

---

**latest :**

**AndiEcker**

**Mar 19, 2024**



# CONTENTS

<b>1</b>	<b>features and use-cases</b>	<b>3</b>
<b>2</b>	<b>screenshots of apps developed with ae portions</b>	<b>5</b>
<b>3</b>	<b>code maintenance guidelines</b>	<b>7</b>
3.1	portions code requirements . . . . .	7
3.2	design pattern and software principles . . . . .	7
3.3	contributing . . . . .	7
3.4	create new namespace . . . . .	11
3.5	register a new namespace portion . . . . .	11
<b>4</b>	<b>registered namespace package portions</b>	<b>13</b>
4.1	ae.base . . . . .	14
4.2	ae.deep . . . . .	30
4.3	ae.django_utils . . . . .	40
4.4	ae.droid . . . . .	41
4.5	ae.notify . . . . .	41
4.6	ae.valid . . . . .	45
4.7	ae.files . . . . .	46
4.8	ae.paths . . . . .	53
4.9	ae.core . . . . .	72
4.10	ae.lockname . . . . .	86
4.11	ae.dynamicod . . . . .	88
4.12	ae.i18n . . . . .	91
4.13	ae.parse_date . . . . .	96
4.14	ae.literal . . . . .	97
4.15	ae.progress . . . . .	102
4.16	ae.updater . . . . .	106
4.17	ae.console . . . . .	109
4.18	ae.sys_core . . . . .	122
4.19	ae.sys_data . . . . .	128
4.20	ae.sys_core_sh . . . . .	173
4.21	ae.sys_data_sh . . . . .	184
4.22	ae.db_core . . . . .	196
4.23	ae.db_oracle . . . . .	206
4.24	ae.db_pg . . . . .	208
4.25	ae.transfer_service . . . . .	211
4.26	ae.sideload_server . . . . .	220
4.27	ae.gui_app . . . . .	225
4.28	ae.gui_help . . . . .	252

4.29	ae.kivy_gsl	272
4.30	ae.kivy_dyn_chi	280
4.31	ae.kivy_relief_canvas	282
4.32	ae.kivy	285
4.33	ae.kivy.il8n	286
4.34	ae.kivy.widgets	287
4.35	ae.kivy.apps	306
4.36	ae.kivy.behaviors	313
4.37	ae.kivy.tours	321
4.38	ae.kivy_auto_width	328
4.39	ae.kivy_file_chooser	334
4.40	ae.kivy_iterable_displayer	337
4.41	ae.kivy_qr_displayer	338
4.42	ae.kivy_sideload	339
4.43	ae.kivy_user_prefs	345
4.44	ae.lisz_app_data	345
4.45	ae.enaml_app	363
4.46	ae.enaml_app.functions	367
<b>5</b>	<b>indices and tables</b>	<b>369</b>
	<b>Python Module Index</b>	<b>371</b>
	<b>Index</b>	<b>373</b>

welcome to the documentation of the portions (app/service modules and sub-packages) of this freely extendable ae namespace ([PEP 420](#)).



## FEATURES AND USE-CASES

the portions of this namespace are simplifying your Python application or service in the areas/domains:

- \* data processing/validation
- \* file handling
- \* i18n (localization)
- \* configuration settings
- \* console
- \* logging
- \* database access
- \* networking
- \* multi-platform/-OS
- \* context help
- \* app tours
- \* user preferences (font size, color, theming, ...)
- \* QR codes
- \* sideloading





## SCREENSHOTS OF APPS DEVELOPED WITH AE PORTIONS

Fig. 2.1: Gls!Test app demo

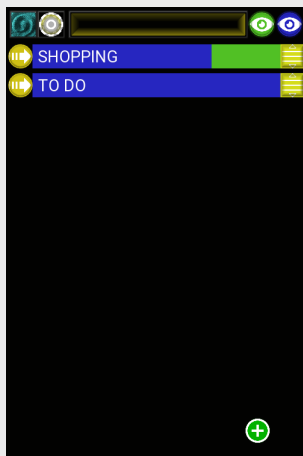


Fig. 2.2: kivy lisz app root list



Fig. 2.3: fruits sub-list



Fig. 2.4: using light theme

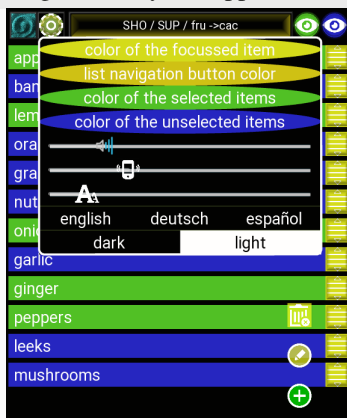


Fig. 2.5: lisz user preferences

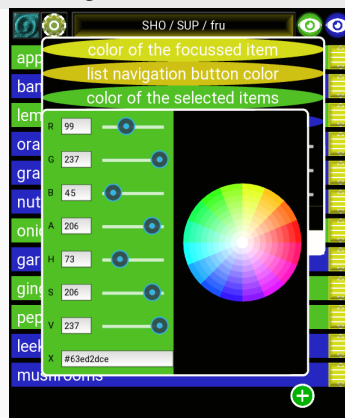
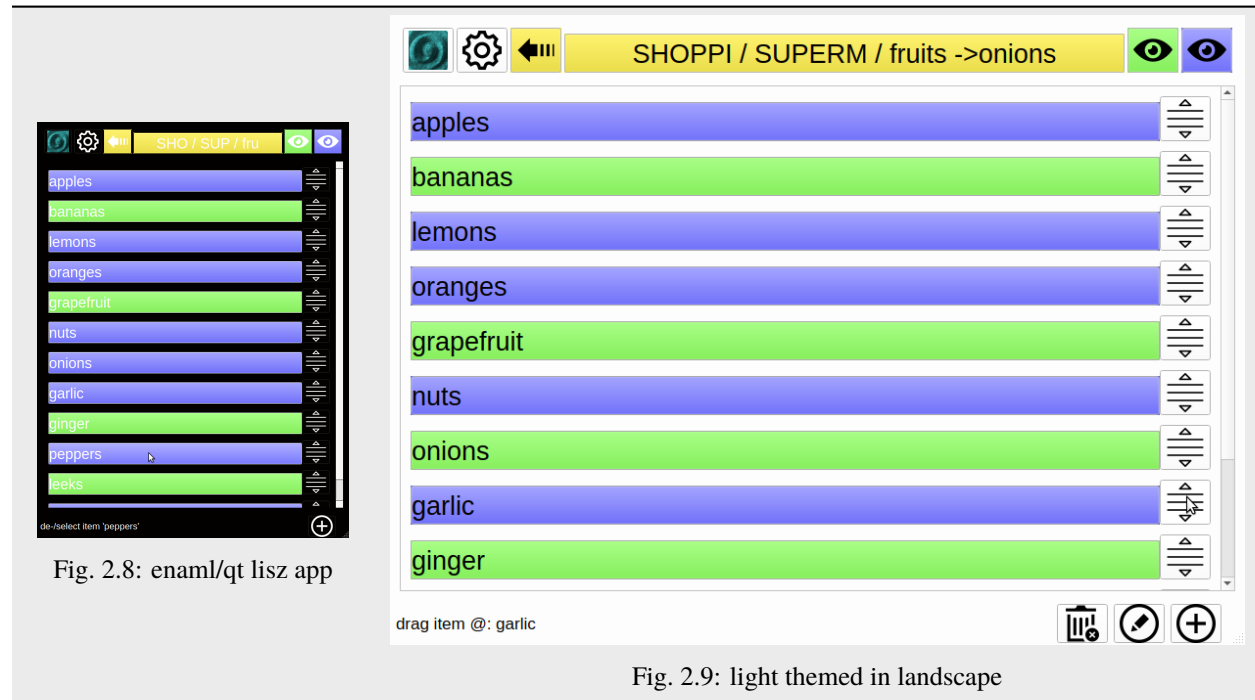


Fig. 2.6: kivy lisz color editor



Fig. 2.7: bigger font size



## CODE MAINTENANCE GUIDELINES

### 3.1 portions code requirements

- pure python
- fully typed ([PEP 526](#))
- fully *documented*
- 100 % test coverage
- multi thread safe
- code checks (using pylint and flake8)

### 3.2 design pattern and software principles

- DRY - don't repeat yourself
- KIS - keep it simple

### 3.3 contributing

we want to make it as easy and fun as possible for you to contribute to this project.

#### 3.3.1 reporting bugs

before you create a new issue, please check to see if you are using the latest version of this project; the bug may already be resolved.

also search for similar problems in the issue tracker; it may already be an identified problem.

include as much information as possible into the issue description, at least:

1. version numbers (of Python and any involved packages).
2. small self-contained code example that reproduces the bug.
3. steps to reproduce the error.
4. any traceback/error/log messages shown.

### 3.3.2 requesting new features

1. on the git repository host server create new issue, providing a clear and detailed explanation of the feature you want and why it's important to add.
2. if you are able to implement the feature yourself (refer to the [contribution steps](#) section below).

### 3.3.3 contribution steps

thanks for your contribution – we'll get your merge request reviewed. you could also review other merge requests, just like other developers will review yours and comment on them. based on the comments, you should address them. once the reviewers approve, the maintainers will merge.

before you start make sure you have a [GitLab account](#).

contribution can be done either with the `git-repo-manager` tool or directly by using the `git` command and the Gitlab server.

#### using the git repository manager *grm*

1. fork and clone the repository of this project to your computer

in your console change the working directory to your project's parent folder. then run the following command with the `new_feature_or_fix` part replaced by an appropriate branch name, describing shortly your contribution:

```
grm -b new_feature_or_fix fork ae-group/ae_ae
```

---

**Note:** the `grm fork` action will also add the forked repository as the remote `upstream` to your local repository.

---

after the repository fork you change your current working directory to the new working tree root folder, created by the `grm fork` action, and execute the `grm renew` action. this will prepare a new package version of the project and upgrade the project files created from templates to its latest version.

2. code and check

now use your favorite IDE/Editor to implement the new feature or code the bug fix. don't forget to amend the project with new unit and integrity tests, and ensure they pass, by executin from time to time the `grm check` action.

3. publish your changes

before you initiate a push/merge request against the Gitlab server, execute the `grm prepare` action, which will create, with the help of the `git diff` command, a `.commit_msg.txt` file in the working tree root of your project, containing a short summary in the first line followed with a blank line and a list of the project files that got added, changed or deleted.

---

**Hint:** the `.commit_msg.txt` file can be amended by any text editor before you run the `grm commit` action. for changes initiated by an issue please include the issue number (in the format `fixes #<issue-number>`) into this file. you may use [Markdown](#) syntax in this file for simple styling.

---

to finally commit and upload your changes run the following three `grm` actions in the root folder of your project:

```

grm commit
grm push
grm request

```

the `grm commit` command is first executing a `grm check` action to do a final check of the project resources and to run the unit and integrity tests. if all these checks pass then a new git commit will be created, including your changes to the project. `grm push` will then push the commit to your `origin` remote repository (your fork) and `grm request` will finally create a new merge/pull request against the upstream remote repository (the forked one).

**Hint:** to complete the workflow a maintainer of the project has to execute the `grm release` action. this will merge your changes into the main branch *develop* of the upstream repository and then release a new version of the project onto PyPI.

more detailed information of the features of the `grm` tool are available within [the grm user manual](#).

### using *git* and *Gitlab*

alternatively to the `grm` tool you could directly use the [git command suite](#) and the [Gitlab website](#) to achieve the same (with a lot more of typing and fiddling ;-):

1. fork the [upstream repository](#) into your user account.
2. clone your forked repo as `origin` remote to your computer, and add an `upstream` remote for the destination repo by running the following commands in the console of your local machine:

```

git clone https://gitlab.com/<YourGitLabUserName>/ae_ae.git
git remote add upstream https://gitlab.com/ae-group/ae_ae.git

```

3. checkout out a new local feature branch and update it to the latest version of the `develop` branch:

```

git checkout -b <new_feature_or_fix_branch_name> develop
git pull --rebase upstream develop

```

please keep your code clean by staying current with the `develop` branch, where code will be merged. if you find another bug, please fix it in a separated branch instead.

4. push the branch to your fork. treat it as a backup:

```

git push origin <new_feature_or_fix_branch_name>

```

5. code

implement the new feature or the bug fix; include tests, and ensure they pass.

6. check

run the basic code style and typing checks locally (pylint, mypy and flake8) before you commit.

7. commit

for every commit please write a short summary in the first line followed with a blank line and then more detailed descriptions of the change. for bug fixes please include any issue number (in the format `#nnn`) in your summary:

```

git commit -m "issue #123: put change summary here (can be a issue title)"

```

latest :

---

---

**Note:** **never leave the commit message blank!** provide a detailed, clear, and complete description of your changes!

---

8. publish your changes (prepare a Merge Request)

before submitting a [merge request](#), update your branch to the latest code:

```
git pull --rebase upstream develop
```

if you have made many commits, we ask you to squash them into atomic units of work. most issues should have one commit only, especially bug fixes, which makes them easier to back port:

```
git checkout develop
git pull --rebase upstream develop
git checkout <new_feature_or_fix_branch_name>
git rebase -i develop
```

push changes to your fork:

```
git push -f
```

9. issue/make a GitLab Merge Request:

- navigate to your fork where you just pushed to
- click *Merge Request*
- in the branch field write your feature branch name (this is filled with your default branch name)
- click *Update Commit Range*
- ensure the changes you implemented are included in the *Commits* tab
- ensure that the *Files Changed* tab incorporate all of your changes
- fill in some details about your potential patch including a meaningful title
- click *New merge request*.

### 3.3.4 release to PyPI

the release of a new/changed project will automatically be initiated by the GitLab CI, using the two protected vars PYPI\_USERNAME and PYPI\_PASSWORD (marked as masked) from the users group of this namespace, in order to provide the user name and password of the maintainers PyPI account (on Gitlab.com at Settings/CI\_CD/Variables).

### 3.3.5 useful links and resources

- [General GitLab documentation](#)
- [GitLab workflow documentation](#)
- grm (git repository manager) module [project repository](#) and [user manual](#)

## 3.4 create new namespace

a **PEP 420** namespace splits the codebase of a library or framework into multiple project repositories, called portions (of the namespace).

---

**Hint:** the *aedev* namespace is providing the *grm* tool to create and maintain any namespace and its portions.

---

the id of a new namespace consists of letters only and has to be available on PYPI. the group-name name gets by default generated from the namespace name plus the suffix '-group', so best choose an id that results in a group name that is available on your repository server.

## 3.5 register a new namespace portion

follow the steps underneath to add and register a new module as portion onto the *ae* namespace:

1. open a console window and change the current directory to the parent directory of your projects root folders.
2. choose a not-existing/unique name for the new portion (referred as *<portion-name>* in the next steps).
3. run `grm --namespace=ae --project=<portion_name> new-module` to register the portion name within the namespace, to create a new project folder *ae\_<portion-name>* (providing initial project files created from templates) and to get a pre-configured git repository (with the remote already set and the initial files unstaged, to be extended, staged and finally committed).
4. run `cd ae_<portion-name>` to change the current to the working tree root of the new portion project.
5. run `pyenv local venv_name` (or any other similar tool) to create/prepare a local virtual environment.
6. fans of TDD are then coding unit tests in the prepared test module *test\_ae\_<portion-name>.py*, situated within the *tests* sub-folder of your new code project folder.
7. extend the file *<portion\_name>.py* situated in the *ae* sub-folder to implement the new portion.
8. run `grm check-integrity` to run the linting and unit tests (if they fail go one or two steps back).
9. run `grm prepare`, then amend the commit message within the file *.commit\_msg.txt*, then run `grm commit` and `grm push` to commit and upload your new portion to your personal remote/server repository fork, and finally run `grm request` to request the merge/pull into the forked/upstream repository in the users group *ae-group* (at <https://gitlab.com/ae-group>).

the registration of a new portion to the *ae* namespace has to be done by a namespace maintainer.

registered portions will automatically be included into the *ae namespace documentation*, available at [ReadTheDocs](#).





## REGISTERED NAMESPACE PACKAGE PORTIONS

the following list contains all registered portions of the ae namespace, plus additional modules of each portion.

**Hint:** a note on the ordering: portions with no dependencies are at the begin of the following list. the portions that are depending on other portions of the ae namespace are listed more to the end.

---

<i>ae.base</i>	basic constants, helper functions and context manager
<i>ae.deep</i>	easy handling of deeply nested data structures
<i>ae.django_utils</i>	helpers for django projects
<i>ae.droid</i>	android constants and helper functions
<i>ae.notify</i>	send notifications via email, telegram or whatsapp
<i>ae.valid</i>	data validation helper functions
<i>ae.files</i>	generic file object helpers
<i>ae.paths</i>	generic file path helpers
<i>ae.core</i>	application core constants, helper functions and base classes
<i>ae.lockname</i>	named threading locks
<i>ae.dynamicod</i>	dynamic execution of code blocks and expressions
<i>ae.i18n</i>	internationalization / localization helpers
<i>ae.parse_date</i>	parse date strings more flexible and less strict
<i>ae.literal</i>	literal type detection and evaluation
<i>ae.progress</i>	display progress of long running processes
<i>ae.updater</i>	application environment updater
<i>ae.console</i>	console application environment
<i>ae.sys_core</i>	dynamic system configuration, initialization and connection
<i>ae.sys_data</i>	external system data structures
<i>ae.sys_core_sh</i>	SiHOT PMS system core xml interface
<i>ae.sys_data_sh</i>	SiHOT system data interface
<i>ae.db_core</i>	database connection and data manipulation base classes
<i>ae.db_ora</i>	database system core layer to access Oracle databases
<i>ae.db_pg</i>	postgres database layer
<i>ae.transfer_service</i>	transfer client and server services
<i>ae.sideload_server</i>	sideloading server
<i>ae.gui_app</i>	base class for python applications with a graphical user interface
<i>ae.gui_help</i>	main app base class with context help for flow and app state changes
<i>ae.kivy_gls1</i>	add glsl shaders to your kivy widget

continues on next page

Table 4.1 – continued from previous page

<code>ae.kivy_dyn_chi</code>	dynamic children mix-in for kivy container widgets
<code>ae.kivy_relief_canvas</code>	inner/outer elliptic/square reliefs for any kivy widget
<code>ae.kivy</code>	core application classes and widgets for GUIApp-conform Kivy apps
<code>ae.kivy.i18n</code>	ae.kivy.i18n module
<code>ae.kivy.widgets</code>	ae.kivy.widgets module
<code>ae.kivy.apps</code>	ae.kivy.apps module
<code>ae.kivy.behaviors</code>	ae.kivy.behaviors module
<code>ae.kivy.tours</code>	ae.kivy.tours module
<code>ae.kivy_auto_width</code>	automatic width mix-in classes for kivy widgets
<code>ae.kivy_file_chooser</code>	extended kivy file chooser widget
<code>ae.kivy_iterable_displayer</code>	iterable displayer widget
<code>ae.kivy_qr_displayer</code>	qr code displayer widget
<code>ae.kivy_sideloading</code>	kivy mixin and widgets to integrate a sideloading server in your app
<code>ae.kivy_user_prefs</code>	user preferences widgets for your kivy app
<code>ae.lisz_app_data</code>	lisz demo app data handling
<code>ae.enaml_app</code>	enaml application widgets, helper functions and classes
<code>ae.enaml_app.functions</code>	enaml application helper functions.

## 4.1 ae.base

### 4.1.1 basic constants, helper functions and context manager

this module is pure python, has no external dependencies, and is providing base constants, common helper functions and context managers.

#### base constants

ISO format strings for *date* and *datetime* values are provided by the constants `DATE_ISO` and `DATE_TIME_ISO`.

the `UNSET` constant is useful in cases where `None` is a valid data value and another special value is needed to specify that e.g. an argument or attribute has no (valid) value or did not get specified/passed.

default values to compile file and folder names for a package or an app project are provided by the constants: `DOCS_FOLDER`, `TESTS_FOLDER`, `TEMPLATES_FOLDER`, `BUILD_CONFIG_FILE`, `PACKAGE_INCLUDE_FILES_PREFIX`, `PY_EXT`, `PY_INIT`, `PY_MAIN`, `CFG_EXT` and `INI_EXT`.

the constants `PACKAGE_NAME`, `PACKAGE_DOMAIN` and `PERMISSIONS` are mainly used for apps running on mobile devices. to avoid redundancies, these values get loaded from the *build config file* - if it exists in the current working directory.

## base helper functions

to write more compact and readable code for the most common file I/O operations, the helper functions `read_file()` and `write_file()` are wrapping Python's built-in `open()` function and its context manager.

the function `duplicates()` returns the duplicates of an iterable type.

`norm_line_sep()` is converting any combination of line separators of a string to a single new-line character.

`norm_name()` converts any string into a name that can be used e.g. as file name or as method/attribute name.

to normalize a file path, in order to remove `.`, `..` placeholders, to resolve symbolic links or to make it relative or absolute, call the function `norm_path()`.

`camel_to_snake()` and `snake_to_camel()` providing name conversions of class and method names.

to encode unicode strings to other codecs the functions `force_encoding()` and `to_ascii()` can be used.

the `round_traditional()` function get provided by this module for traditional rounding of float values. the function signature is fully compatible to Python's `round()` function.

the function `instantiate_config_parser()` ensures that the `ConfigParser` instance is correctly configured, e.g. to support case-sensitive config variable names and to use `ExtendedInterpolation` for the interpolation argument.

`app_name_guess()` guesses the name of a running Python application from the application environment, with the help of `build_config_variable_values()`, which determines config-variable-values from the build spec file of an app project.

## operating system constants and helpers

the string `os_platform` provides the OS where your app is running, extending Python's `sys.platform()` for mobile platforms like Android and iOS.

`os_host_name()`, `os_local_ip()` and `os_user_name()` are determining machine and user information from the OS.

use `env_str()` to determine the value of an OS environment variable with automatic variable name conversion. other helper functions provided by this namespace portion to determine the values of the most important system environment variables for your application are `sys_env_dict()` and `sys_env_text()`.

## android-specific constants and helper functions

some helper functions of this module are provided to be used for the Android OS.

the helper function `start_app_service()` is starting a service in its own, separate thread. with the function `request_app_permissions()` you can ensure that all your Android permissions will be requested. the module `ae.kivy.apps` does this automatically on app startup. on other platforms than Android it will have no effect to call these functions.

---

**Note:** importing this module on an Android OS, it is monkey patching the `shutil` module to prevent crashes.

---

links to other android code and service examples and documentation:

- [`https://python-for-android.readthedocs.io/en/latest/`](https://python-for-android.readthedocs.io/en/latest/) \_\_
- [`https://github.com/kivy/python-for-android/tree/develop/pythonforandroid/recipes/android/src/android`](https://github.com/kivy/python-for-android/tree/develop/pythonforandroid/recipes/android/src/android) \_\_
- [`https://github.com/tshirtman/kivy\\_service\\_osc/blob/master/src/main.py`](https://github.com/tshirtman/kivy_service_osc/blob/master/src/main.py) \_\_

latest :

---

- [`https://blog.kivy.org/2014/01/building-a-background-application-on-android-with-kivy/`](https://blog.kivy.org/2014/01/building-a-background-application-on-android-with-kivy/)\_\_
- [`https://github.com/Android-for-Python/Android-for-Python-Users`](https://github.com/Android-for-Python/Android-for-Python-Users)\_\_
- [`https://github.com/Android-for-Python/INDEX-of-Examples`](https://github.com/Android-for-Python/INDEX-of-Examples)\_\_

big thanks to [Robert Flatt](#) for his investigations, findings and documentations to code and build Kivy apps for the Android OS, and to [Gabriel Pettier](#) for his service osc example.

### generic context manager

the context manager `in_wd()` allows to switch the current working directory temporarily. the following example demonstrates a typical usage, together with a temporary path, created with the help of Pythons `TemporaryDirectory` class:

```
with tempfile.TemporaryDirectory() as tmp_dir, in_wd(tmp_dir):  
    # within the context the tmp_dir is set as the current working directory  
    assert os.getcwd() == tmp_dir  
# current working directory set back to the original path and the temporary directory_  
↪ got removed
```

### call stack inspection

`module_attr()` dynamically determines a reference to an attribute (variable, function, class, ...) in a module.

`module_name()`, `stack_frames()`, `stack_var()` and `stack_vars()` are inspecting the call stack frames to determine e.g. variable values of the callers of a function/method.

---

**Hint:** the `AppBase` class uses these helper functions to determine the version and title of an application, if these values are not specified in the instance initializer.

---

another useful helper function provided by this portion to inspect and debug your code is `full_stack_trace()`.

## Module Attributes

<code>DOCS_FOLDER</code>	project documentation root folder name
<code>TESTS_FOLDER</code>	name of project folder to store unit/integration tests
<code>TEMPLATES_FOLDER</code>	template folder name, used in template and namespace root projects to maintain and provide common file templates
<code>BUILD_CONFIG_FILE</code>	gui app build config file
<code>PACKAGE_INCLUDE_FILES_PREFIX</code>	file/folder names prefix included into setup package_data/ae_updater
<code>PY_CACHE_FOLDER</code>	python cache folder name
<code>PY_EXT</code>	file extension for modules and hooks
<code>PY_INIT</code>	init-module file name of a python package
<code>PY_MAIN</code>	main-module file name of a python executable
<code>CFG_EXT</code>	CFG config file extension
<code>INI_EXT</code>	INI config file extension
<code>DATE_ISO</code>	ISO string format for date values (e.g.
<code>DATE_TIME_ISO</code>	ISO string format for datetime values
<code>DEF_ENCODE_ERRORS</code>	default encode error handling for UnicodeEncodeErrors
<code>DEF_ENCODING</code>	encoding for <i>force_encoding()</i> that will always work independent from destination (console, file sys, ...).
<code>NAME_PARTS_SEP</code>	name parts separator character, e.g.
<code>SKIPPED_MODULES</code>	skipped modules used as default by <i>module_name()</i> , <i>stack_var()</i> and <i>stack_vars()</i>
<code>UNSET</code>	pseudo value used for attributes/arguments if <i>None</i> is needed as a valid value
<code>os_platform</code>	operating system / platform string (see <i>_os_platform()</i> ).

## Functions

<code>app_name_guess()</code>	guess/try to determine the name of the currently running app (w/o assessing not yet initialized app instance).
<code>build_config_variable_values(*names_defaults)</code>	determine build config variable values from the <code>buildozer.spec</code> file in the current directory.
<code>camel_to_snake(name)</code>	convert name from CamelCase to snake_case.
<code>deep_dict_update(data, update)</code>	update the optionally nested data dict in-place with the items and sub-items from the update dict.
<code>dummy_function(*_args, **_kwargs)</code>	null function accepting any arguments and returning <code>None</code> .
<code>duplicates(values)</code>	determine all duplicates in the iterable specified in the values argument.
<code>env_str(name[, convert_name])</code>	determine the string value of an OS environment variable, optionally preventing invalid variable name.
<code>force_encoding(text[, encoding, errors])</code>	force/ensure the encoding of text (str or bytes) without any UnicodeDecodeError/UnicodeEncodeError.
<code>full_stack_trace(ex)</code>	get full stack trace from an exception.
<code>import_module(import_name[, path])</code>	search, import and execute a Python module dynamically without adding it to <code>sys.modules</code> .

continues on next page

Table 4.2 – continued from previous page

<code>in_wd(new_cwd)</code>	context manager to temporary switch the current working directory / cwd.
<code>instantiate_config_parser()</code>	instantiate and prepare config file parser.
<code>main_file_paths_parts(portion_name)</code>	determine tuple of supported main/version file name path part tuples.
<code>module_attr(import_name[, attr_name])</code>	determine dynamically a reference to a module or to any attribute (variable/func/class) declared in the module.
<code>module_file_path([local_object])</code>	determine the absolute path of the module from which this function got called.
<code>module_name(*skip_modules[, depth])</code>	find the first module in the call stack that is <i>not</i> in <code>skip_modules</code> .
<code>norm_line_sep(text)</code>	convert any combination of line separators in the <code>text</code> arg to new-line characters.
<code>norm_name(name[, allow_num_prefix])</code>	normalize name to start with a letter/alphabetic/underscore and to contain only alphanumeric/underscore chars.
<code>norm_path(path[, make_absolute, ...])</code>	normalize path, replacing <code>../</code> parts or the tilde character (for home folder) and transform to relative/abs.
<code>now_str([sep])</code>	return the current timestamp as string (to use as suffix for file and variable/attribute names).
<code>os_host_name()</code>	determine the operating system host/machine name.
<code>os_local_ip()</code>	determine ip address of this system/machine in the local network (LAN or WLAN).
<code>os_user_name()</code>	determine the operating system username.
<code>project_main_file(import_name[, project_path])</code>	determine the main module file path of a project package, containing the project <code>__version__</code> module variable.
<code>read_file(file_path[, extra_mode, encoding, ...])</code>	returning content of the text/binary file specified by <code>file_path</code> argument as string.
<code>request_app_permissions(*_args, **_kwargs)</code>	null function accepting any arguments and returning None.
<code>round_traditional(num_value[, num_digits])</code>	round numeric value traditional.
<code>snake_to_camel(name[, back_convertible])</code>	convert name from snake_case to CamelCase.
<code>stack_frames([depth])</code>	generator returning the call stack frame from the level given in <code>depth</code> .
<code>stack_var(name, *skip_modules[, scope, depth])</code>	determine variable value in calling stack/frames.
<code>stack_vars(*skip_modules[, find_name, ...])</code>	determine all global and local variables in a calling stack/frames.
<code>start_app_service(*_args, **_kwargs)</code>	null function accepting any arguments and returning None.
<code>sys_env_dict()</code>	returns dict with python system run-time environment values.
<code>sys_env_text([ind_ch, ind_len, key_ch, ...])</code>	compile formatted text block with system environment info.
<code>to_ascii(unicode_str)</code>	converts unicode string into ascii representation.
<code>write_file(file_path, content[, extra_mode, ...])</code>	(over)write the file specified by <code>file_path</code> with text or binary/bytes content.

## Classes

*UnsetType()*

(singleton) UNSET (type) object class.

**DOCS\_FOLDER = 'docs'**

project documentation root folder name

**TESTS\_FOLDER = 'tests'**

name of project folder to store unit/integration tests

**TEMPLATES\_FOLDER = 'templates'**

template folder name, used in template and namespace root projects to maintain and provide common file templates

**BUILD\_CONFIG\_FILE = 'buildozer.spec'**

gui app build config file

**PACKAGE\_INCLUDE\_FILES\_PREFIX = 'ae\_'**

file/folder names prefix included into setup package\_data/ae\_updater

**PY\_CACHE\_FOLDER = '\_\_pycache\_\_'**

python cache folder name

**PY\_EXT = '.py'**

file extension for modules and hooks

**PY\_INIT = '\_\_init\_\_.py'**

init-module file name of a python package

**PY\_MAIN = '\_\_main\_\_.py'**

main-module file name of a python executable

**CFG\_EXT = '.cfg'**

CFG config file extension

**INI\_EXT = '.ini'**

INI config file extension

**DATE\_ISO = '%Y-%m-%d'**

ISO string format for date values (e.g. in config files/variables)

**DATE\_TIME\_ISO = '%Y-%m-%d %H:%M:%S.%f'**

ISO string format for datetime values

**DEF\_ENCODE\_ERRORS = 'backslashreplace'**

default encode error handling for UnicodeEncodeErrors

**DEF\_ENCODING = 'ascii'**

encoding for *force\_encoding()* that will always work independent from destination (console, file sys, ...).

**NAME\_PARTS\_SEP = '\_'**

name parts separator character, e.g. for *norm\_name()*

---

```
SKIPPED_MODULES = ('ae.base', 'ae.paths', 'ae.dynamicod', 'ae.core', 'ae.console',
'ae.gui_app', 'ae.gui_help', 'ae.kivy', 'ae.kivy.apps', 'ae.kivy.behaviors',
'ae.kivy.i18n', 'ae.kivy.tours', 'ae.kivy.widgets', 'ae.enaml_app', 'ae.beeware_app',
'ae.pyglet_app', 'ae.pygobject_app', 'ae.dabo_app', 'ae.qpython_app', 'ae.appjar_app',
'importlib._bootstrap', 'importlib._bootstrap_external')
```

skipped modules used as default by `module_name()`, `stack_var()` and `stack_vars()`

### class UnsetType

Bases: `object`

(singleton) UNSET (type) object class.

`__bool__()`

ensure to be evaluated as False, like None.

`__len__()`

ensure to be evaluated as empty.

**UNSET = <ae.base.UnsetType object>**

pseudo value used for attributes/arguments if *None* is needed as a valid value

### app\_name\_guess()

guess/try to determine the name of the currently running app (w/o assessing not yet initialized app instance).

#### Return type

`str`

#### Returns

application name/id or “unguessable” if not guessable.

**build\_config\_variable\_values(\*names\_defaults, section='app')**

determine build config variable values from the `buildozer.spec` file in the current directory.

#### Parameters

- **names\_defaults** `// (Tuple[str, Any])` – tuple of tuples of build config variable names and default values.
- **section** `// (str)` – name of the spec file section, using ‘app’ as default.

#### Return type

`Tuple[Any, ...]`

#### Returns

tuple of build config variable values (using the passed default value if not specified in the `BUILD_CONFIG_FILE` spec file or if the spec file does not exist in cwd).

**camel\_to\_snake(name)**

convert name from CamelCase to snake\_case.

#### Parameters

**name** `// (str)` – name string in CamelCaseFormat.

#### Return type

`str`

#### Returns

name in snake\_case\_format.



**deep\_dict\_update**(*data*, *update*)

update the optionally nested data dict in-place with the items and sub-items from the update dict.

**Parameters**

- **data** (dict) – dict to be updated/extended. non-existing keys of dict-sub-items will be added.
- **update** (dict) – dict with the [sub-]items to update in the data dict.

---

**Hint:** the module/portion `ae.deep` is providing more deep update helper functions.

---

**dummy\_function**(\*\_args, \*\*\_kwargs)

null function accepting any arguments and returning None.

**Parameters**

- **\_args** – ignored positional arguments.
- **\_kwargs** – ignored keyword arguments.

**Returns**

always None.

**duplicates**(*values*)

determine all duplicates in the iterable specified in the *values* argument.

inspired by Ritesh Kumars answer to <https://stackoverflow.com/questions/9835762>.

**Parameters**

**values** (Iterable) – iterable (list, tuple, str, ...) to search for duplicate items.

**Return type**

list

**Returns**

list of the duplicate items found (can contain the same duplicate multiple times).

**env\_str**(*name*, *convert\_name=False*)

determine the string value of an OS environment variable, optionally preventing invalid variable name.

**Parameters**

- **name** (str) – name of an OS environment variable.
- **convert\_name** (bool) – pass True to prevent invalid variable names by converting Camel-Case names into SNAKE\_CASE, lower-case into upper-case and all non-alpha-numeric characters into underscore characters.

**Return type**

Optional[str]

**Returns**

string value of OS environment variable if found, else None.

**force\_encoding**(*text*, *encoding='ascii'*, *errors='backslashreplace'*)

force/ensure the encoding of text (str or bytes) without any UnicodeDecodeError/UnicodeEncodeError.

**Parameters**

- **text** (Union[str, bytes]) – text as str/bytes.
- **encoding** (str) – encoding (def= `DEF_ENCODING`).

- **errors** (str) – encode error handling (def= `DEF_ENCODE_ERRORS`).

**Return type**`str`**Returns**

text as str (with all characters checked/converted/replaced to be encode-able).

**full\_stack\_trace(ex)**

get full stack trace from an exception.

**Parameters**

**ex** (Exception) – exception instance.

**Return type**`str`**Returns**

str with stack trace info.

**import\_module(import\_name, path=<ae.base.UnsetType object>)**

search, import and execute a Python module dynamically without adding it to sys.modules.

**Parameters**

- **import\_name** (str) – dot-name of the module to import.
- **path** (Union[str, UnsetType, None]) – optional file path of the module to import. if this arg is not specified or has the default value (`UNSET`), then the path will be determined from the import name. specify `None` to prevent the module search.

**Return type**`Optional[ModuleType]`**Returns**

a reference to the loaded module or `None` if module could not be imported.

**instantiate\_config\_parser()**

instantiate and prepare config file parser.

**Return type**`ConfigParser`**in\_wd(new\_cwd)**

context manager to temporary switch the current working directory / cwd.

**Parameters**

**new\_cwd** (str) – path to the directory to switch to (within the context/with block). an empty string gets interpreted as the current working directory.

**Return type**`Generator[None, None, None]`**main\_file\_paths\_parts(portion\_name)**

determine tuple of supported main/version file name path part tuples.

**Parameters**

**portion\_name** (str) – portion or package name.

**Return type**`Tuple[Tuple[str, ...], ...]`

**Returns**

tuple of tuples of main/version file name path parts.

**module\_attr**(*import\_name*, *attr\_name*="")

determine dynamically a reference to a module or to any attribute (variable/func/class) declared in the module.

**Parameters**

- **import\_name** (str) – import-/dot-name of the distribution/module/package to load/import.
- **attr\_name** (str) – name of the attribute declared within the module. do not specify or pass an empty string to get/return a reference to the imported module instance.

**Return type**

Optional[Any]

**Returns**

module instance or module attribute value or None if module not found or UNSET if module attribute doesn't exist.

---

**Note:** a previously not imported module will *not* be added to *sys.modules* by this function.

---

**module\_file\_path**(*local\_object*=None)

determine the absolute path of the module from which this function got called.

**Parameters**

**local\_object** (Optional[Callable]) – optional local module, class, method, function, traceback, frame, or code object of the calling module (passing *lambda: 0* also works). omit to use instead the `__file__` module variable (which will not work if the module is frozen by py2exe or PyInstaller).

**Return type**

str

**Returns**

module path (inclusive module file name) or empty string if path not found/determinable.

**module\_name**(\**skip\_modules*, *depth*=0)

find the first module in the call stack that is *not* in *skip\_modules*.

**Parameters**

- **skip\_modules** (str) – module names to skip (def=this ae.core module).
- **depth** (int) – the calling level from which on to search. the default value 0 refers the frame and the module of the caller of this function. pass 1 or an even higher value if you want to get the module name of a function/method in a deeper level in the call stack.

**Return type**

Optional[str]

**Returns**

the module name of the call stack level specified by *depth*.

**norm\_line\_sep**(*text*)

convert any combination of line separators in the *text* arg to new-line characters.

**Parameters**

**text** (str) – string containing any combination of line separators ('\\r\\n' or '\\r').

**Return type**`str`**Returns**

normalized/converted string with only new-line (`'\n'`) line separator characters.

**norm\_name**(*name*, *allow\_num\_prefix=False*)

normalize name to start with a letter/alphabetic/underscore and to contain only alphanumeric/underscore chars.

**Parameters**

- **name** *(str)* – any string to be converted into a valid variable/method/file/... name.
- **allow\_num\_prefix** *(bool)* – pass True to allow leading digits in the returned normalized name.

**Return type**`str`**Returns**

cleaned/normalized/converted name string (e.g. for a variable-/method-/file-name).

**norm\_path**(*path*, *make\_absolute=True*, *remove\_base\_path=""*, *remove\_dots=True*, *resolve\_sym\_links=True*)

normalize path, replacing `../` parts or the tilde character (for home folder) and transform to relative/abs.

**Parameters**

- **path** *(str)* – path string to normalize/transform.
- **make\_absolute** *(bool)* – pass False to not convert path to an absolute path.
- **remove\_base\_path** *(str)* – pass a valid base path to return a relative path, even if the argument values of *make\_absolute* or *resolve\_sym\_links* are *True*.
- **remove\_dots** *(bool)* – pass False to not replace/remove the `.` and `..` placeholders.
- **resolve\_sym\_links** *(bool)* – pass False to not resolve symbolic links, passing True implies a *True* value also for the *make\_absolute* argument.

**Return type**`str`**Returns**

normalized path string: absolute if *remove\_base\_path* is empty and either *make\_absolute* or *resolve\_sym\_links* is *True*; relative if *remove\_base\_path* is a base path of *path* or if *path* got specified as relative path and neither *make\_absolute* nor *resolve\_sym\_links* is *True*.

---

**Hint:** the *normalize()* function additionally replaces *PATH\_PLACEHOLDERS*.

---

**now\_str**(*sep=""*)

return the current timestamp as string (to use as suffix for file and variable/attribute names).

**Parameters**

**sep** *(str)* – optional prefix and separator character (separating date from time and in time part the seconds from the microseconds).

**Return type**`str`**Returns**

timestamp as string (length=20 + 3 \* len(sep)).

**os\_host\_name()**

determine the operating system host/machine name.

**Return type**

`str`

**Returns**

machine name string.

**os\_local\_ip()**

determine ip address of this system/machine in the local network (LAN or WLAN).

inspired by answers of SO users @dml and @fatal\_error to the question: <https://stackoverflow.com/questions/166506>.

**Return type**

`str`

**Returns**

ip address of this machine in the local network (WLAN or LAN/ethernet) or empty string if this machine is not connected to any network.

**\_os\_platform()**

determine the operating system where this code is running (used to initialize the `os_platform` variable).

**Return type**

`str`

**Returns**

operating system (extension) as string:

- `'android'` for all Android systems.
- `'cygwin'` for MS Windows with an installed Cygwin extension.
- `'darwin'` for all Apple Mac OS X systems.
- `'freebsd'` for all other BSD-based unix systems.
- `'ios'` for all Apple iOS systems.
- `'linux'` for all other unix systems (like Arch, Debian/Ubuntu, Suse, ...).
- `'win32'` for MS Windows systems (w/o the Cygwin extension).

**os\_platform = 'linux'**

operating system / platform string (see `_os_platform()`).

this string value gets determined for most of the operating systems with the help of Python's `sys.platform()` function and additionally detects the operating systems iOS and Android (not supported by Python).

**os\_user\_name()**

determine the operating system username.

**Return type**

`str`

**Returns**

username string.

**project\_main\_file(import\_name, project\_path="")**

determine the main module file path of a project package, containing the project `__version__` module variable.

**Parameters**

- **import\_name** (str) – import name of the module/package (including namespace prefixes for namespace packages).
- **project\_path** (str) – optional path where the project of the package/module is situated. not needed if the current working directory is the root folder of either the import\_name project or of a sister project (under the same project parent folder).

**Return type**

str

**Returns**

absolute file path/name of main module or empty string if no main/version file found.

**read\_file**(file\_path, extra\_mode="", encoding=None, error\_handling='ignore')

returning content of the text/binary file specified by file\_path argument as string.

**Parameters**

- **file\_path** (str) – file path/name to load into a string or a bytes array.
- **extra\_mode** (str) – extra open mode flag characters appended to “r” onto open() mode argument. pass “b” to read the content of a binary file returned as bytes array. in binary mode the argument passed in **error\_handling** will be ignored.
- **encoding** (Optional[str]) – encoding used to load and convert/interpret the file content.
- **error\_handling** (str) – for files opened in text mode pass ‘strict’ or None to return None (instead of an empty string) for the cases where either a decoding *ValueError* exception or any *OSError*, *FileNotFoundError* or *PermissionError* exception got raised. the default value ‘ignore’ will ignore any decoding errors (missing some characters) and will return an empty string on any file/os exception. this parameter will be ignored if the **extra\_mode** argument contains the ‘b’ character (to read the file content as binary/bytes-array).

**Return type**

Union[str, bytes]

**Returns**

file content string or bytes array.

**Raises**

- **FileNotFoundError** – if file does not exist.
- **OSError** – if **file\_path** is misspelled or contains invalid characters.
- **PermissionError** – if current OS user account lacks permissions to read the file content.
- **ValueError** – on decoding errors.

**round\_traditional**(num\_value, num\_digits=0)

round numeric value traditional.

needed because python round() is working differently, e.g. round(0.075, 2) == 0.07 instead of 0.08 inspired by <https://stackoverflow.com/questions/31818050/python-2-7-round-number-to-nearest-integer>.

**Parameters**

- **num\_value** (float) – float value to be round.
- **num\_digits** (int) – number of digits to be round (def=0 - rounds to an integer value).

**Return type**

float

**Returns**

rounded value.

**snake\_to\_camel**(*name*, *back\_convertible=False*)

convert name from snake\_case to CamelCase.

**Parameters**

- **name** (str) – name string composed of parts separated by an underscore character (*NAME\_PARTS\_SEP*).
- **back\_convertible** (bool) – pass *True* to get the first character of the returned name in lower-case if the snake name has no leading underscore character (and to allow the conversion between snake and camel case without information loss).

**Return type**

str

**Returns**

name in camel case.

**stack\_frames**(*depth=1*)

generator returning the call stack frame from the level given in *depth*.

**Parameters**

**depth** (int) – the stack level to start; the first returned frame by this generator. the default value (1) refers the next deeper stack frame, respectively the one of the caller of this function. pass 2 or a higher value if you want to start with an even deeper frame/level.

**Return type**

Generator

**Returns**

generated frames of the call stack.

**stack\_var**(*name*, *\*skip\_modules*, *scope=""*, *depth=1*)

determine variable value in calling stack/frames.

**Parameters**

- **name** (str) – variable name to search in the calling stack frames.
- **skip\_modules** (str) – module names to skip (def=see *SKIPPED\_MODULES* module constant).
- **scope** (str) – pass ‘locals’ to only check for local variables (ignoring globals) or ‘globals’ to only check for global variables (ignoring locals). the default value (an empty string) will not restrict the scope, returning either a local or global value.
- **depth** (int) – the calling level from which on to search. the default value (1) refers the next deeper stack frame, which is the caller of the function. pass 2 or an even higher value if you want to start the variable search from a deeper level in the call stack.

**Return type**

Optional[Any]

**Returns**

the variable value of a deeper level within the call stack or UNSET if the variable was not found.

**stack\_vars**(*\*skip\_modules*, *find\_name=""*, *min\_depth=1*, *max\_depth=0*, *scope=""*)

determine all global and local variables in a calling stack/frames.

**Parameters**

- **skip\_modules**¶ (*str*) – module names to skip (def=see [SKIPPED\\_MODULES](#) module constant).
- **find\_name**¶ (*str*) – if passed then the returned stack frame must contain a variable with the passed name.
- **scope**¶ (*str*) – scope to search the variable name passed via [find\\_name](#). pass ‘locals’ to only search for local variables (ignoring globals) or ‘globals’ to only check for global variables (ignoring locals). passing an empty string will find the variable within either locals and globals.
- **min\_depth**¶ (*int*) – the call stack level from which on to search. the default value (1) refers the next deeper stack frame, respectively to the caller of this function. pass 2 or a higher value if you want to get the variables from a deeper level in the call stack.
- **max\_depth**¶ (*int*) – the maximum depth in the call stack from which to return the variables. if the specified argument is not zero and no [skip\\_modules](#) are specified then the first deeper stack frame that is not within the default [SKIPPED\\_MODULES](#) will be returned. if this argument and [find\\_name](#) get not passed then the variables of the top stack frame will be returned.

**Return type**

`Tuple[Dict[str, Any], Dict[str, Any], int]`

**Returns**

tuple of the global and local variable dicts and the depth in the call stack.

**sys\_env\_dict()**

returns dict with python system run-time environment values.

**Return type**

`Dict[str, Any]`

**Returns**

python system run-time environment values like `python_ver`, `argv`, `cwd`, `executable`, `frozen` and `bundle_dir` (if bundled with pyinstaller).

---

**Hint:** see also <https://pyinstaller.readthedocs.io/en/stable/runtime-information.html>

---

**sys\_env\_text(ind\_ch=' ', ind\_len=12, key\_ch='=', key\_len=15, extra\_sys\_env\_dict=None)**

compile formatted text block with system environment info.

**Parameters**

- **ind\_ch**¶ (*str*) – indent character (default=” “).
- **ind\_len**¶ (*int*) – indent depths (default=12 characters).
- **key\_ch**¶ (*str*) – key-value separator character (default=”=”).
- **key\_len**¶ (*int*) – key-name minimum length (default=15 characters).
- **extra\_sys\_env\_dict**¶ (`Optional[Dict[str, str]]`) – dict with additional system info items.

**Return type**

`str`

**Returns**

text block with system environment info.



**to\_ascii**(*unicode\_str*)

converts unicode string into ascii representation.

useful for fuzzy string compare; inspired by MiniQuark's answer in: <https://stackoverflow.com/questions/517923/what-is-the-best-way-to-remove-accents-in-a-python-unicode-string>

**Parameters**

**unicode\_str** (str) – string to convert.

**Return type**

str

**Returns**

converted string (replaced accents, diacritics, ... into normal ascii characters).

**write\_file**(*file\_path*, *content*, *extra\_mode=""*, *encoding=None*)

(over)write the file specified by *file\_path* with text or binary/bytes content.

**Parameters**

- **file\_path** (str) – file path/name to write the passed content into (overwriting any previous content!).
- **content** (Union[str, bytes]) – new file content either passed as string or list of line strings (will be concatenated with the line separator of the current OS: os.linesep).
- **extra\_mode** (str) – open mode flag characters. passed unchanged to the *mode* argument of `open()` if this argument starts with 'a', else this argument value will be appended to 'w'.
- **encoding** (Optional[str]) – encoding used to write/convert/interpret the file content to write.

**Raises**

- **FileExistsError** – if file exists already and is write-protected.
- **FileNotFoundError** – if parts of the file path do not exist.
- **OSError** – if *file\_path* is misspelled or contains invalid characters.
- **PermissionError** – if current OS user account lacks permissions to read the file content.
- **ValueError** – on decoding errors.

**request\_app\_permissions**(\**\_args*, \*\**\_kwargs*)

null function accepting any arguments and returning None.

**Parameters**

- **\_args** – ignored positional arguments.
- **\_kwargs** – ignored keyword arguments.

**Returns**

always None.

**start\_app\_service**(\**\_args*, \*\**\_kwargs*)

null function accepting any arguments and returning None.

**Parameters**

- **\_args** – ignored positional arguments.
- **\_kwargs** – ignored keyword arguments.

latest :

---

### Returns

always None.

## 4.2 ae.deep

### 4.2.1 easy handling of deeply nested data structures

this `ae` namespace portion is pure python, depends only on the Python runtime and the `ae.base` portion, and provides functions for to read, update and delete values of deep data structures. more helper function to prepare and convert data structures between different systems are available in the `ae.sys_data` module.

the root and node objects of deep data structures consisting of sequences (like list, tuple, ...), mappings (dict, ...) and data (class) objects. the leaf data objects are mostly simple types like int, float or string.

### deep data structure example

the following deep data structure is composed of the data class `Person`, a member list and two dictionaries:

```
>>> from dataclasses import dataclass
>>> @dataclass
... class Person:
...     first_name: str
...     hobbies: List[str]
```

```
>>> member_hobbies = [
...     "dancing",
...     "music",          # ...
... ]
```

```
>>> member_peter = Person(
...     first_name="Peter",
...     hobbies=member_hobbies,
...     # ...
... )
```

```
>>> member_list = [
...     member_peter,      # ...
... ]
```

```
>>> club_data = {
...     'city': "Madrid",
...     'members': member_list, # ...
... }
```

```
>>> clubs_mapping = {
...     'fun-club': club_data, # ...
... }
```

putting the above data structures together, results in a deep data structure, in where `clubs_mapping` represents the root object.

the nodes of this deep data structure get referenced by `club_data`, `member_list`, `member_peter` and `member_hobbies`.

the fields `city`, `first_name` and the items 0 and 1 of `member_hobbies` (referencing the values "dancing" and "music") are finally representing the leafs of this data structure:

```
>>> clubs_mapping == {
...     'fun-club': {
...         'city': "Madrid",
...         'members': [
...             Person(
...                 first_name="Peter",
...                 hobbies=[
...                     "dancing",
...                     "music",
...                 ]
...             ),
...         ]
...     }
... }
True
```

### referencing a deep data object

there are two types of paths to reference the data items within a deep data structure: `object key lists` and `key path strings`.

to get any node object or leaf value within a deep data structure, referenced by a key path string, call the functions `key_path_object()`, which expects a data structure in its first argument `obj` and a key path string in its `key_path` second argument.

in the following example, the function `key_path_object()` determines the first name object from the `member_list` data node:

```
>>> key_path_object(member_list, '0.first_name')
'Peter'
```

to determine the same object via an object key list, use the function `key_list_object()`:

```
>>> key_list_object([(member_list, 0),
...                  (member_peter, 'first_name')])
'Peter'
```

use the function `key_path_string()` to convert an object key list into a key path string. the following example determines the same `first_name` data leaf object with an object key list:

```
>>> key_path = key_path_string([(member_list, 0),
...                             (member_peter, 'first_name')])
...
>>> print(repr(key_path))
'0.first_name'
>>> key_path_object(member_list, key_path)
'Peter'
```

e.g. the more deep/complex key path string `'fun-club.members.0.first_name.4'`, references the 5th character of the leaf object "Peter", this time from the root node of the example data structure (`clubs_mapping`):

latest :

```
>>> key_path_object(clubs_mapping, 'fun-club.members.0.first_name.4')
'r'
```

the same char object, referenced above with a key path string, can also be referenced with an object key list, with the help of the function `key_path_string()`:

```
>>> key_path_string([
...     (clubs_mapping, 'fun-club'),      # clubs_mapping['fun-club'] == club_data
...     (club_data, 'members'),          # club_data['members'] == member_list
...     (member_list, 0),                # member_list[0] == member_peter
...     (member_peter, 'first_name'),    # member_peter.first_name == "Peter"
...     ("Peter", 4),                    # "Peter"[4] == "r"
... ])
'fun-club.members.0.first_name.4'
```

## helpers to examine deep data structures

the `deep_search()` function allows to scan and inspect all the elements of any deep data structure. `deep_search()` can also be very useful for discovering internals of the Python language/libraries or to debug and test deep and complex data structures.

`object_items()` is another useful helper function which is returning a list of key-value pairs of any type of data node object.

## helpers to change data in deep data structures

use the function `replace_object()` to change/replace a single node or leaf object within a deep data structure. alternatively you could use `key_path_object()`, by passing the new value as additional argument to it.

for multiple/bulk changes use the function `deep_replace()`, which is traversing/scanning the entire data structure.

the function `deep_update()` merges two deep data structures.

to wipe any node/leaf from a deep data structure use the function `pop_object()`, which returns the old/removed node, item or attribute value. another option to remove objects from a data structure is to use `deep_update()` with data:~ae.base.UNSET values in its `updating_obj` argument.

more details you find in the respective docstring of these functions.

## Module Attributes

<code>DataLeafTypesType</code>	list/tuple of types of deep data leaves
<code>KeyType</code>	index/attribute types of deep data structures
<code>KeyFilterCallable</code>	callable to filter item key/index and attribute names
<code>ObjKeysType</code>	object key list of tuples of: object, key/index/attribute
<code>ObjCheckCallable</code>	<code>deep_replace()/deep_search()</code> parameter type
<code>MutableDataTypes</code>	deep data structure node types
<code>DEFAULT_LEAF_TYPES</code>	default leaf types of deep data structures
<code>KEY_PATH_SEPARATORS</code>	separator characters in key path string

## Functions

<code>deep_replace(obj, replace_with[, ...])</code>	replace values (bottom up) within the passed (deeply nested) data structure.
<code>deep_search(obj, found[, leaf_types, ...])</code>	search key and/or value within the passed (deeply nested) data object structure.
<code>deep_update(obj, updating_obj[, leaf_types, ...])</code>	merge the <code>updating_obj</code> data structure into the similar structured node object <code>obj</code> .
<code>key_filter_default(index_or_attr)</code>	default key filter callable, returning True to filter out key/index/attribute id/name.
<code>key_list_object(obj_keys)</code>	determine object in a deep nested data structure via an object key list.
<code>key_path_object(obj, key_path[, new_value])</code>	determine object in a deep nested data structure via a key path string, and optionally assign a new value to it.
<code>key_path_string(obj_keys)</code>	convert obj keys path into deep object key path string.
<code>key_value(key_str)</code>	convert key string (mapping key, sequence index, obj attribute) into its value (str, int, float, tuple, ..).
<code>next_key_of_path(key_path)</code>	parse key_path to determine the next item key/index, the path separator and the rest of the key path.
<code>object_item_value(obj, key[, default_value])</code>	determine value of a data object item/attribute.
<code>object_items(obj[, leaf_types, key_filter])</code>	determine the items of a data object, with mapping keys, sequence indexes or attribute names and its values.
<code>pop_object(obj_keys)</code>	delete sub-attribute/item of a data object.
<code>replace_object(obj_keys, new_value)</code>	set sub-attribute/item with a mutable parent object to a new value within a deeply nested object/data structure.
<code>value_filter_default(value, obj_keys)</code>	default item value filter callable, returning True to filter out values to be scanned deeper.

### DataLeafTypesType

list/tuple of types of deep data leaves

alias of `Tuple[Type, ...]`

### KeyType

index/attribute types of deep data structures

alias of `Union[str, int, tuple]`

### KeyFilterCallable

callable to filter item key/index and attribute names

alias of `Callable[[Union[str, int, tuple]], bool]`

### ObjKeysType

object key list of tuples of: object, key/index/attribute

alias of `List[Tuple[Any, Any]]`

### ObjCheckCallable

`deep_replace()/deep_search()` parameter type

alias of `Callable[[List[Tuple[Any, Any]], Any, Any], Any]`

**MutableDataTypes** = (`<class 'collections.abc.MutableMapping'>`, `<class 'collections.abc.MutableSequence'>`)

deep data structure node types

```
DEFAULT_LEAF_TYPES = (<class 'bytes'>, <class 'int'>, <class 'float'>, <class 'set'>,
<class 'str'>, <class 'type'>)
```

default leaf types of deep data structures

```
KEY_PATH_SEPARATORS = ('.', '[', ']')
```

separator characters in key path string

```
key_filter_default(index_or_attr)
```

default key filter callable, returning True to filter out key/index/attribute id/name.

this function is the default value for the `key_filter` parameter of the deep scanning and traversing functions `deep_replace()`, `deep_search()`, `deep_update()` and `object_items()`.

if you are using your own filter callable you could call this function from it to prevent endless recursion, especially if your deep data structures contains circular- or self-referencing objects, like e.g.: `rtype: bool`

- self- or doubly-linked data structures (e.g. `kivy.app.App.proxy_ref` (self-linked) or `ae.kivy.apps.KivyMainApp.framework_app` and `ae.kivy.apps.FrameworkApp.main_app`).
- back-linked data structures, like e.g. the `parent` property in Kivy widget trees.
- magic (two leading underscore characters) or internal (one leading underscore) attributes (e.g. via `WindowSDL._WindowBase__instance`).

```
value_filter_default(value, obj_keys)
```

default item value filter callable, returning True to filter out values to be scanned deeper.

this function is the default value for the `value_filter` parameter of the deep scanning and traversing functions `deep_replace()`, `deep_search()` and `deep_update()`.

by using your own filter callable make sure to prevent endless processing or recursion, especially if your deep data structures contains circular- or self-referencing objects. e.g. some data value types have to be excluded from to be deeper processed to prevent `RecursionError` (endless recursion, e.g. on `str` values because `str[i]` is `str`, on `int` because `int.denominator` is `int`).

### Return type

`bool`

```
deep_replace(obj, replace_with, leaf_types=(<class 'bytes'>, <class 'int'>, <class 'float'>, <class 'set'>, <class
'str'>, <class 'type'>), key_filter=<function key_filter_default>, value_filter=<function
value_filter_default>, obj_keys=None)
```

replace values (bottom up) within the passed (deeply nested) data structure.

### Parameters

- **obj** *Any* – mutable sequence or mapping data structure to be deep searched and replaced. can contain any combination of deep nested data objects. mutable node objects (e.g. `dict/list`) as well as the immutable types not included in `leaf_types` will be recursively deep searched (top down) by passing their items one by one to the callback function specified by `replace_with`.
- **replace\_with** *Callable[[List[Tuple[Any, Any]], Any, Any], Any]* – called for each item with the 3 arguments object key list, key in parent data-structure, and the object/value. any return value other than `UNSET` will be used to overwrite the node/leaf object in the data-structure.
- **leaf\_types** *Tuple[Type, ...]* – tuple of leaf types to skip from to be searched deeper. the default value of this parameter is specified in the modul constant `DEFAULT_LEAF_TYPES`.

- **key\_filter** (Callable[[Union[str, int, tuple]], bool]) – called for each sub-item/-attribute of the data structure specified by *obj*. return True for item-key/item-index/attribute-name to be filtered out. by default all attribute/key names starting with an underscore character will be filtered out (see default callable `key_filter_default()`).
- **value\_filter** (Callable[[Any, List[Tuple[Any, Any]]], bool]) – called for each sub-item/-attribute of the data structure specified by *obj*. return True for items/attributes values to be filtered out. by default empty values, excluded values (see EXCLUDED\_VALUE\_TYPES) and already scanned objects will be filtered out (see default callable `value_filter_default()`).
- **obj\_keys** (Optional[List[Tuple[Any, Any]]]) – used (internally only) to pass the parent data-struct path in recursive calls.

**Return type**`int`**Returns**

the number of levels of the first mutable data objects above the changed data object, or 0 of the changed data object is mutable.

**Raises**

ValueError if no mutable parent object is in data structure (*obj*).

**Raises**

AttributeError e.g. if *obj* is of type `int`, and *int* is missing in `leaf_types`.

---

**Note:** make sure to prevent overwrites on internal objects of the Python runtime, on some of them the Python interpreter could even crash (e.g. with: exit code 134 (interrupted by signal 6: SIGABRT)).

---

**deep\_search**(*obj*, *found*, *leaf\_types*=(<class 'bytes'>, <class 'int'>, <class 'float'>, <class 'set'>, <class 'str'>, <class 'type'>), *key\_filter*=<function `key_filter_default`>, *value\_filter*=<function `value_filter_default`>, *obj\_keys*=None)

search key and/or value within the passed (deeply nested) data object structure.

**Parameters**

- **obj** (Any) – root object to start the top-down deep search from, which can contain any combination of deep nested elements/objects. for each sub-element the callable passed into *found* will be executed. if the callable returns True then the data path, the key and the value will be stored in a tuple and added to the search result list (finally returned to the caller of this function).
- for iterable objects of type dict/tuple/list, the sub-items will be searched, as well as the attributes determined via the Python `dir()` function. to reduce the number of items/attributes to be searched use the parameters *leaf\_types* and/or *key\_filter*.
- **found** (Callable[[List[Tuple[Any, Any]], Any, Any], Any]) – called for each item with 3 arguments (data-struct-path, key in data-structure, value), and if the return value is True then the data/object path, the last key and value will be added as a new item to the returned list.
- **leaf\_types** (Tuple[Type, ...]) – tuple of leaf types to skip from to be searched deeper. the default value of this parameter is specified in the modul constant `DEFAULT_LEAF_TYPES`.
- **key\_filter** (Callable[[Union[str, int, tuple]], bool]) – called for each sub-item/-attribute of the data structure specified by *obj*. return True for item-key/item-index/attribute-name to be filtered out. by default all attribute/key names starting with an underscore character will be filtered out (see default callable `key_filter_default()`).

- **value\_filter** *¶* (Callable[[Any, List[Tuple[Any, Any]]], bool) – called for each sub-item/-attribute of the data structure specified by *obj*. return True for items/attributes values to be filtered out. by default empty values, excluded values (see EXCLUDED\_VALUE\_TYPES) and already scanned objects will be filtered out (see default callable *value\_filter\_default()*).
- **obj\_keys** *¶* (Optional[List[Tuple[Any, Any]]]) – used (internally only) to pass the parent data-struct path in recursive calls.

**Return type**

List[Tuple[List[Tuple[Any, Any]], Any, Any]]

**Returns**

list of tuples (data-struct-path, key, value); one tuple for each found item within the passed *obj* argument. an empty list will be returned if no item got found.

**deep\_update**(*obj*, *updating\_obj*, *leaf\_types*=(<class 'bytes'>, <class 'int'>, <class 'float'>, <class 'set'>, <class 'str'>, <class 'type'>), *key\_filter*=<function *key\_filter\_default*>, *value\_filter*=<function *value\_filter\_default*>, *obj\_keys*=None)

merge the *updating\_obj* data structure into the similar structured node object *obj*.

**Parameters**

- **obj** *¶* (Any) – deep data object to update.
- **updating\_obj** *¶* (Any) – data structure similar structured like the *obj* argument with update values. a UNSET value will delete the item from the *obj* argument.
- **leaf\_types** *¶* (Tuple[Type, ...]) – tuple of leaf types to skip from to be searched deeper. the default value of this parameter is specified in the modul constant *DEFAULT\_LEAF\_TYPES*.
- **key\_filter** *¶* (Callable[[Union[str, int, tuple]], bool]) – called for each sub-item/-attribute of the data structure specified by *obj*. return True for item-key/item-index/attribute-name to be filtered out. by default all attribute/key names starting with an underscore character will be filtered out (see default callable *key\_filter\_default()*).
- **value\_filter** *¶* (Callable[[Any, List[Tuple[Any, Any]]], bool) – called for each sub-item/-attribute of the data structure specified by *obj*. return True for items/attributes values to be filtered out. by default empty values, excluded values (see EXCLUDED\_VALUE\_TYPES) and already scanned objects will be filtered out (see default callable *value\_filter\_default()*).
- **obj\_keys** *¶* (Optional[List[Tuple[Any, Any]]]) – used (internally only) to pass the parent data-struct path in recursive calls.

**key\_list\_object**(*obj\_keys*)

determine object in a deep nested data structure via an object key list.

**Parameters**

**obj\_keys** *¶* (List[Tuple[Any, Any]]) – object key list.

**Return type**

Any

**Returns**

recalculated object referenced by the first object and the keys of *obj\_keys* or *~ae.base.UNSET* if not found.

**Raises**

TypeError if key does not match the object type in any item of *obj\_keys*. ValueError if *obj\_keys* is not of type *ObjKeysType*.



---

**Hint:** to include changes on immutable data structures, the returned object value gets recalculated, starting from the first object (`obj_keys`[0][0]`), going deeper via the key only (while ignoring all other child objects in the object key list specified by :paramref:`obj\_keys`).

---

**key\_path\_object** (*obj*, *key\_path*, *new\_value*=<ae.base.UnsetType object>)

determine object in a deep nested data structure via a key path string, and optionally assign a new value to it.

#### Parameters

- **obj** *(Any)* – initial data object to search in (and its sub-objects).
- **key\_path** *(str)* – composed key string containing dict keys, tuple/list/str indexes and object attribute names, separated by a dot character, like shown in the following examples:

```
>>> class AClass:
...     str_attr_name_a = "a_attr_val"
...     dict_attr = {'a_str_key': 3, 999: "value_with_int_key", '999
↳ ': "..str_key"}
```

```
>>> class BClass:
...     str_attr_name_b = "b_b_b_b_b"
...     b_obj = AClass()
```

```
>>> b = BClass()
```

```
>>> assert key_path_object(b, 'str_attr_name_b') == "b_b_b_b_b"
>>> assert key_path_object(b, 'b_obj.str_attr_name_a') == "a_attr_val"
↳ "
>>> assert key_path_object(b, 'b_obj.str_attr_name_a.5') == "r" #_
↳ 6th chr of a_attr_val
>>> assert key_path_object(b, 'b_obj.dict_attr.a_str_key') == 3
```

the item key or index value of lists and dictionaries can alternatively be specified in Python syntax, enclosed in [ and ]:

```
>>> assert key_path_object(b, 'b_obj.dict_attr["a_str_key"]') == 3
>>> assert key_path_object(b, 'b_obj.dict_attr[\'a_str_key\']') == 3
>>> assert key_path_object(b, 'b_obj.dict_attr[999]') == "value_with_
↳ int_key"
>>> assert key_path_object(b, 'b_obj.dict_attr["999"]') == "..str_key"
↳ "
```

only dict key strings that are not can be misinterpreted as number can be specified without the high commas (enclosing the key string), like e.g.:

```
>>> assert key_path_object(b, 'b_obj.dict_attr[a_str_key]') == 3
```

- **new\_value** *(Optional[Any])* – optional new value - replacing the found object. the old object value will be returned.

---

**Note:** immutable objects, like tuples, that are embedding in obj will be automatically updated/replaced up in the data tree structure until a mutable object (list, dict or object) get

found.

---

**Return type**

*Any*

**Returns**

specified object/value (the old value if *new\_value* got passed) or *UNSET* if not found/exists (key path string is invalid).

**key\_path\_string**(*obj\_keys*)

convert obj keys path into deep object key path string.

**Parameters**

**obj\_keys** *(List[Tuple[Any, Any]])* – object key list to convert.

**Return type**

*str*

**Returns**

key path string of the object keys path specified by the *obj\_keys* argument.

**key\_value**(*key\_str*)

convert key string (mapping key, sequence index, obj attribute) into its value (str, int, float, tuple, ..).

**Return type**

*Any*

**next\_key\_of\_path**(*key\_path*)

parse *key\_path* to determine the next item key/index, the path separator and the rest of the key path.

**Parameters**

**key\_path** *(str)* – data object key/index path string to parse.

**Return type**

*Tuple[Any, str, str]*

**Returns**

tuple of key/index, the separator character and the (unparsed) rest of the key path (possibly an empty string).

**Raises**

*IndexError* if the argument of *key\_path* is an empty string.

**object\_items**(*obj*, *leaf\_types*=(<class 'bytes'>, <class 'int'>, <class 'float'>, <class 'set'>, <class 'str'>, <class 'type'>), *key\_filter*=<function *key\_filter\_default*>)

determine the items of a data object, with mapping keys, sequence indexes or attribute names and its values.

**Parameters**

- **obj** *(Any)* – data structure/node object (list, dict, set, tuple, data object, ...).
- **leaf\_types** *(Tuple[Type, ...])* – tuple of leaf types to skip from to be searched deeper. the default value of this parameter is specified in the modul constant *DEFAULT\_LEAF\_TYPES*.
- **key\_filter** *(Callable[[Union[str, int, tuple]], bool])* – called for each sub-item/-attribute of the data structure specified by *obj*. return True for item-key/item-index/attribute-name to be filtered out. by default all attribute/key names starting with an underscore character will be filtered out (see default callable *key\_filter\_default*()).

**Return type**

*List[Tuple[Any, Any]]*

**Returns**

items view of the data object specified in the `obj` argument.

**object\_item\_value**(*obj*, *key*, *default\_value*=<*ae.base.UnsetType object*>)

determine value of a data object item/attribute.

**Parameters**

- **obj** *(Any)* – data structure object to get item/attribute value from.
- **key** *(Any)* – mapping key, attribute name or sequence index of the item/attribute.
- **default\_value** *(Any)* – default value to return if the item/attribute does not exist in `obj`.

**Return type**

*Any*

**Returns**

data object item/attribute value or the `default_value` if not found.

**pop\_object**(*obj\_keys*)

delete sub-attribute/item of a data object.

**Parameters**

**obj\_keys** *(List[Tuple[Any, Any]])* – list of (object, key) tuples identifying an element within a deeply nested data structure or object hierarchy. the root of the data/object structure is the object at list index 0 and the element to be deleted is identified by the object and key in the last list item of this argument.

the referenced data structure can contain even immutable node objects (like tuples) which will be accordingly changed/replaced if affected/needed. for that at least one object/element above the immutable object in the deep data structure has to be mutable, else a `ValueError` will be raised.

**Return type**

*List[Any]*

**Returns**

list of deleted values or UNSET if not found.

**Raises**

`ValueError` if no immutable parent object got found or if the `obj_keys`' is empty.  
`IndexError` if the one of the specified indexes in :`paramref:`obj_keys`` does not exist.

**replace\_object**(*obj\_keys*, *new\_value*)

set sub-attribute/item with a mutable parent object to a new value within a deeply nested object/data structure.

**Parameters**

- **obj\_keys** *(List[Tuple[Any, Any]])* – list of (object, key) tuples identifying an element within a deeply nested data structure or object hierarchy. the root of the data/object structure is the object at list index 0 and the element to be changed is identified by the object and key in the last list item of this argument.

the referenced data structure can contain immutable data node objects (like tuples) which will be accordingly changed/replaced if affected/needed.

at least one object/element, situated above of replaced data object within the deep data structure, has to be mutable, else a `ValueError` will be raised.

- **new\_value** *(Any)* – value to be assigned to the element referenced by the last list item of the argument in `obj_keys`.

**Return type**`int`**Returns**

the number of levels of the first mutable data objects above the changed data object, or 0 if the changed data object is mutable.

**Raises**

ValueError if no immutable parent object got found or if the object key list in the `obj_keys` argument is empty.

## 4.3 ae.django\_utils

### 4.3.1 helpers for django projects

this module is providing helper functions for your django projects.

#### Functions

<code>generic_language(lang_code)</code>	remove the country specific part - if exists - from the passed language code.
<code>requested_language()</code>	determine the currently requested short translation language code, fallback to english/'en' if unset.
<code>set_url_part(url[, fragment, query])</code>	add or replace fragment and/or query string of url.

#### **generic\_language(*lang\_code*)**

remove the country specific part - if exists - from the passed language code.

**Parameters**

**lang\_code** *(str)* – lower-case language code with optional hyphen and country specific part.

**Return type**`str`**Returns**

stripped/short language code (e.g. return “en” if `lang_code == “en-gb”`)

#### **requested\_language()**

determine the currently requested short translation language code, fallback to english/'en' if unset.

**Return type**`str`**Returns**

short (without country specific part) and lower-case language code

#### **set\_url\_part(*url*, *fragment*="", *query*="", *\*\*set\_part\_values*)**

add or replace fragment and/or query string of url.

**Parameters**

- **url** *(str)* – url to complete by adding and/or updating parts of it.
- **fragment** *(str)* – id of the fragment to add/replace.
- **query** *(str)* – query string to add/replace.

- **set\_part\_values** – other parts like ‘scheme’, ‘netloc’, ‘path’, ‘params’ (see :func:urllib.parse.urlparse).

**Return type**`str`**Returns**

url with added/replaced parts.

## 4.4 ae.droid

### 4.4.1 android constants and helper functions

to include this ae namespace portion into your project add the following import statement into your main module (main.py) of your application project:

```
import ae.droid
```

this import will ensure that all your permissions will be requested on app startup. on other platforms than Android it will have no effect.

big thanks to [Robert Flatt](#) for his investigations, findings and documentations to code and build Kivy apps for the Android OS, and to [Gabriel Pettier](#) for his service osc example.

**links to android service examples and documentation:**

- [https://github.com/tshirtman/kivy\\_service\\_osc/blob/master/src/main.py](https://github.com/tshirtman/kivy_service_osc/blob/master/src/main.py)
- <https://python-for-android.readthedocs.io/en/latest/services/#arbitrary-scripts-services>
- <https://blog.kivy.org/2014/01/building-a-background-application-on-android-with-kivy/>

## 4.5 ae.notify

### 4.5.1 send notifications via email, telegram or whatsapp

this pure python module depends mainly on the Standard Python Libraries `email` and `smtplib` and the external `requests` module.

an instance of the `Notifications` has to be created for each notification sender in your app, providing the sender's credentials for each used notification channel (service).

the notification channels and the receiver(s) can be specified individually for each notification message to send.

## Module Attributes

<code>DEF_ENC_PORT</code>	standard SMTP port
<code>DEF_ENC_SERVICE_NAME</code>	standard SMTP service name
<code>SSL_ENC_PORT</code>	port to use SMTP via SSL
<code>SSL_ENC_SERVICE_NAME</code>	service name in <code>Notifications.smtp_server_uri</code> of SMTP via SSL
<code>TLS_ENC_PORT</code>	port to use SMTP via TLS
<code>TLS_ENC_SERVICE_NAME</code>	service name in <code>Notifications.smtp_server_uri</code> of SMTP via TLS
<code>TELEGRAM_MESSAGE_MAX_LEN</code>	maximum length of Telegram notification message body
<code>WHATSAPP_MESSAGE_MAX_LEN</code>	maximum length of Whatsapp notification message body

## Classes

<code>Notifications([smtp_server_uri, mail_from, ...])</code>	a single instance of this class can be used to handle all notifications of an app/service.
---	--

**DEF\_ENC\_PORT = 25**

standard SMTP port

**DEF\_ENC\_SERVICE\_NAME = 'smtp'**

standard SMTP service name

**SSL\_ENC\_PORT = 465**

port to use SMTP via SSL

**SSL\_ENC\_SERVICE\_NAME = 'smtps'**

service name in `Notifications.smtp_server_uri` of SMTP via SSL

**TLS\_ENC\_PORT = 587**

port to use SMTP via TLS

**TLS\_ENC\_SERVICE\_NAME = 'smtpTLS'**

service name in `Notifications.smtp_server_uri` of SMTP via TLS

**TELEGRAM\_MESSAGE\_MAX\_LEN = 4096**

maximum length of Telegram notification message body

**WHATSAPP\_MESSAGE\_MAX\_LEN = 65536**

maximum length of Whatsapp notification message body

**\_body\_mime\_type\_conversion(msg\_body, mime\_type)**

convert content of notification message body.

### Parameters

- **msg\_body** (str) – message body string.
- **mime\_type** (str) – mime type to convert to, if it has the “to” prefix in front of the resulting mime type.

**Return type**`Tuple[str, str]`**Returns**

tuple of converted message body and resulting mime type (removing the “to” prefix).

```
class Notifications(smtp_server_uri="", mail_from="", local_mail_host="", telegram_token="",
                   whatsapp_token="", whatsapp_sender="")
```

Bases: `object`

a single instance of this class can be used to handle all notifications of an app/service.

```
__init__(smtp_server_uri="", mail_from="", local_mail_host="", telegram_token="", whatsapp_token="",
          whatsapp_sender="")
```

initialize one or more different services for a sender of multiple notifications to individual receivers.

**Parameters**

- **smtp\_server\_uri** (str) – host and optional port and user credentials of email SMTP server to use, in the format `[service://{}][{}user{}:password{}@{}mail_server_host{}:mail_server_port]`. default SMTP ports: 25/DEF\_ENC\_PORT, port 587/TSL\_ENC\_PORT for E-SMTP/TLS or 465/SSL\_ENC\_PORT for smtps/SSL.
- **mail\_from** (str) – email sender address.
- **local\_mail\_host** (str) – FQDN of the local email host in the SMTP HELO/EHLO command.
- **telegram\_token** (str) – token for the Telegram cloud API obtained from the @BotFather bot.
- **whatsapp\_token** (str) – token for the WhatsApp cloud API obtained from the developer portal.
- **whatsapp\_sender** (str) – sender phone number id for the WhatsApp cloud API obtained from the developer portal.

```
send_notification(msg_body, receiver, subject="", mime_type='to_html')
```

send a notification message with optional subject to receiver via the specified service.

**Parameters**

- **msg\_body** (str) – message body. line breaks are converted (br-tag <-> newline character) in accordance with mime\_type.
- **receiver** (str) – receiver address in the format `service:address=name`, where `service` is `mailto`, `telegram` or `whatsapp`, `address` is an email address, a chat id or phone number and `name` is the name of the receiving person.
- **subject** (str) – optional subject text. added to the top of the msg\_body for messenger services, separated by an empty line. if not specified or specified as empty string or as a single space character, then it will not be added to the top of the message body.
- **mime\_type** (str) – mime type ('html' or 'plain'), and optional conversion to it (if starts with 'to').

**Return type**`str`**Returns**

error message on error or empty string if notification got send successfully.

**send\_email**(*msg\_body*, *address*, *subject*, *name*, *mime\_type*='to\_html')

send email to the passed address.

**Parameters**

- **msg\_body** (str) – message body text. for new lines use newline char in plain and <br> in html mime\_type.
- **address** (str) – email address of the receiver.
- **subject** (str) – email subject text.
- **name** (str) – name of the receiver.
- **mime\_type** (str) – mime type ('html' or 'plain'), and optional conversion to it (if starts with 'to').

**Return type**

str

**Returns**

error message on error or empty string if notification email got send successfully.

**send\_telegram**(*msg\_body*, *chat\_id*, *name*, *mime\_type*='to\_html')

send message to the passed telegram chat id.

**Parameters**

- **msg\_body** (str) – message body text. in 'html' mime\_type message texts are only a few tags support by Telegram (see <https://core.telegram.org/bots/api#html-style>). on top of that you can also include the following tags, which will be either converted or removed:  
\* br: will be converted into a new line character. \* div: will be removed.
- **chat\_id** (str) – chat id of the telegram receiver or group.
- **name** (str) – name of the receiver.
- **mime\_type** (str) – mime type ('html' or 'plain'), and optional conversion to it (if starts with 'to').

**Return type**

str

**Returns**

error message on error or empty string if notification got send successfully.

---

**Hint:** see <https://www.heise.de/select/ct/2023/8/2231816070982959290> for useful bots and tips. use <https://telemetr.io/>, <https://lyzem.com/> or <https://tgstat.com/> to search/find public channels/groups.

---

**send\_whatsapp**(*msg\_body*, *receiver\_id*, *name*, *mime\_type*='to\_html')

send message to the passed WhatsApp user/group.

**Parameters**

- **msg\_body** (str) – message body text. for new lines use newline char in plain and <br> in html mime\_type.
- **receiver\_id** (str) – phone number with country code (and a leading '+') of the WhatsApp receiver or the id of the WA group (last part of the URL to invite somebody into the group).
- **name** (str) – name of the receiver.



- **mime\_type** (str) – mime type ('html' or 'plain'), and optional conversion to it (if starts with 'to'). recognized/converted html tags are b, br, i and pre.

**Return type**

str

**Returns**

error message on error or empty string if notification got send successfully.

using WA Business API (see: <https://developers.facebook.com/docs/whatsapp/on-premises/reference/messages> and [https://github.com/Neurotech-HQ/heyoo/blob/58ad576c3dfaf05bad5f342bc8614cf0ba02e6ae/heyoo/\\_\\_init\\_\\_.py#L43](https://github.com/Neurotech-HQ/heyoo/blob/58ad576c3dfaf05bad5f342bc8614cf0ba02e6ae/heyoo/__init__.py#L43)) has the restriction that the receiver has first to send a message to the sender to get a window of 24 hours. and using pyWhatKit's webbrowser-based approach will not work on PythonAnywhere because web.whatsapp.com is not in their whitelist (<https://www.pythonanywhere.com/whitelist/>)

## 4.6 ae.valid

### 4.6.1 data validation helper functions

this module is pure Python and has no dependencies.

the two slightly bigger helper functions provided by this namespace portion are `correct_email()` and `correct_phone()`, which are useful to check if a string contains a valid email address or phone number.

they also allow you to automatically correct an email address or a phone number to a valid format. more sophisticated helpers for the validation of email addresses, phone numbers and post addresses are available in the `ae.validation` namespace portion.

#### Functions

<code>correct_email(email[, changed, removed])</code>	check and correct email address from a user input (removing all comments).
<code>correct_phone(phone[, changed, removed, ...])</code>	check and correct phone number from a user input (removing all invalid characters including spaces).

**correct\_email**(email, changed=False, removed=None)

check and correct email address from a user input (removing all comments).

special conversions that are not returned as changed/corrected are: the domain part of an email will be corrected to lowercase characters, additionally emails with all letters in uppercase will be converted into lowercase.

regular expressions are not working for all edge cases (see the answer to this SO question: <https://stackoverflow.com/questions/201323/using-a-regular-expression-to-validate-an-email-address>) because RFC822 is very complex (even the reg expression recommended by RFC 5322 is not complete; there is also a more readable form given in the informational RFC 3696). additionally a regular expression does not allow corrections. therefore this function is using a procedural approach (using recommendations from RFC 822 and [https://en.wikipedia.org/wiki/Email\\_address](https://en.wikipedia.org/wiki/Email_address)).

**Parameters**

- **email** (str) – email address to check and correct.
- **changed** (bool) – optional flag if email address got changed (before calling this function) - will be returned unchanged if email did not get corrected.

- **removed** (Optional[List[str]]) – optional list declared by caller to pass back all the removed characters including the index in the format “<index>:<removed\_character(s)>”.

**Return type**

Tuple[str, bool]

**Returns**

tuple of (possibly corrected email address, flag if email got changed/corrected).

**correct\_phone**(*phone*, *changed=False*, *removed=None*, *keep\_1st\_hyphen=False*)

check and correct phone number from a user input (removing all invalid characters including spaces).

**Parameters**

- **phone** (str) – phone number to check and correct.
- **changed** (bool) – optional flag if phone got changed (before calling this function) - will be returned unchanged if phone did not get corrected.
- **removed** (Optional[List[str]]) – optional list declared by caller to pass back all the removed characters including the index in the format “<index>:<removed\_character(s)>”.
- **keep\_1st\_hyphen** (bool) – pass True to keep at least the first occurring hyphen character.

**Return type**

Tuple[str, bool]

**Returns**

tuple of (possibly corrected phone number, flag if phone got changed/corrected).

## 4.7 ae.files

### 4.7.1 generic file object helpers

this namespace portion is pure Python providing helpers for file object and content managing. it only depends on the *ae.base* namespace portion.

---

**Hint:** more helper functions to manage directory/folder structures are provided by the *ae.paths* portion.

---

the helper function *copy\_bytes()* provides recoverable copies of binary files and file streams, with progress callbacks for each copied bytes chunk/buffer.

*file\_lines()* and *read\_file\_text()* are helpers to read/load text file contents.

the function *write\_file\_text()* stores a string to a text file.

the helper function *file\_transfer\_progress()* puts the amount of transferred bytes in a short and user readable format, to be displayed as progress string in a file transfer progress.

*RegisteredFile* and *CachedFile* encapsulate and optionally cache the contents of a file within a file object. instances of these classes are compatible with the file objects provided by Python's *pathlib* module. but also pure path strings can be used as file objects (see also the *FileObject* type).

all these types of file objects are supported by the files register class *FilesRegister* from the *ae.paths* portion.

## registered file

a registered file object represents a single file on your file system and can be instantiated from one of the classes *RegisteredFile* or *CachedFile* provided by this module/portion:

```
from ae.files import RegisteredFile

rf = RegisteredFile('path/to/the/file_name.extension')

assert str(rf) == 'path/to/the/file_name.extension'
assert rf.path == 'path/to/the/file_name.extension'
assert rf.stem == 'file_name'
assert rf.ext == '.extension'
assert rf.properties == {}
```

file properties will be automatically attached to each file object instance with the instance attribute *properties*. in the last example it results in an empty dictionary because the *path* of this file object does not contain folder names with an underscore character.

## file properties

file property names and values are automatically determined from the names of their sub-folders, specified in the *path* attribute. every sub-folder name containing an underscore character in the format <property-name>\_<value> will be interpreted as a file property:

```
rf = RegisteredFile('property1_69/property2_3.69/property3_whatever/file_name.ext')
assert rf.properties['property1'] == 69
assert rf.properties['property2'] == 3.69
assert rf.properties['property3'] == 'whatever'
```

the property types *int*, *float* and *string* are recognized and converted into a property value. boolean values can be specified as 1 and 0 integers.

## cached file

a cached file created from the *CachedFile* behaves like a *registered file* and additionally provides the possibility to cache parts or the whole file content as well as the file pointer of the opened file:

```
cf = CachedFile('integer_69/float_3.69/string_whatever/file_name.ext')

assert str(cf) == 'integer_69/float_3.69/string_whatever/file_name.ext'
assert cf.path == 'integer_69/float_3.69/string_whatever/file_name.ext'
assert cf.stem == 'file_name'
assert cf.ext == '.ext'
assert cf.properties['integer'] == 69
assert cf.properties['float'] == 3.69
assert cf.properties['string'] == 'whatever'
```

pn instantiation of the *CachedFile* file object the default file object loader function *\_default\_object\_loader()* will be used, which opens a file stream via Python's *open()* built-in. alternatively you can specify a specific file object loader with the *object\_loader* parameter or by assigning a callable directly to the *object\_loader* attribute:

latest :

```
cf = CachedFile('integer_69/float_3.69/string_whatever/file_name.ext',
                object_loader=lambda cached_file_obj: my_open_method(cached_file_obj.
↪path))
```

the cached file object is accessible via the *loaded\_object* attribute of the cached file object instance:

```
assert isinstance(cf.loaded_object, TextIOWrapper)
cf.loaded_object.seek(...)
cf.loaded_object.read(...)

cf.loaded_object.close()
```

## Module Attributes

<i>FileObject</i>	file object type, e.g.
<i>PropertyType</i>	types of file property values
<i>PropertiesType</i>	dict of file properties
<i>FilenameOrStream</i>	file name or file stream pointer

## Functions

<i>copy_bytes</i> (src_file, dst_file, *, ...)	recoverable copy of a file or stream (file-like object), optionally with progress callbacks.
<i>file_lines</i> (file_path[, encoding])	returning lines of the text file specified by file_path argument as tuple.
<i>file_transfer_progress</i> (transferred_bytes[, ...])	return string to display the transfer progress of transferred bytes in short and user readable format.
<i>read_file_text</i> (file_path[, encoding, ...])	returning content of the text file specified by file_path argument as string.
<i>write_file_text</i> (text_or_lines, file_path[, ...])	write the passed text string or list of line strings into the text file specified by file_path argument.

## Classes

<i>CachedFile</i> (file_path[, object_loader, ...])	represents a cacheables registered file object - see also <i>cached file</i> examples.
<i>RegisteredFile</i> (file_path, **kwargs)	represents a single file - see also <i>registered file</i> examples.

## FileObject

file object type, e.g. a file path str or any class or callable where the returned instance/value is either a string or an object with a *stem* attribute (holding the file name w/o extension), like e.g. *CachedFile*, *RegisteredFile*, *pathlib.Path* or *pathlib.PurePath*.

alias of `Union[str, RegisteredFile, CachedFile, Path, PurePath, Any]`

**PropertyType**

types of file property values

alias of `Union[int, float, str]`

**PropertiesType**

dict of file properties

alias of `Dict[str, Union[int, float, str]]`

**FilenameOrStream**

file name or file stream pointer

alias of `Union[str, BinaryIO]`

**copy\_bytes**(*src\_file*, *dst\_file*, \*, *transferred\_bytes*=0, *total\_bytes*=0, *buf\_size*=16384, *overwrite*=False, *move\_file*=False, *recoverable*=False, *errors*=None, *progress\_func*=<function dummy\_function>, *\*\*progress\_kwargs*)

recoverable copy of a file or stream (file-like object), optionally with progress callbacks.

**Parameters**

- **src\_file** (Union[str, BinaryIO]) – source file name or opened stream (file-like) object. if passing a non-seekable stream together with a non-zero value in *transferred\_bytes* then the source stream has to be set to the correct position before you call this function. if passing any source stream then also the total file/stream size has to be passed into the *total\_bytes* parameter. source file streams does also not support a True value in the *move\_file* argument.
- **dst\_file** (Union[str, BinaryIO]) – destination file name or opened stream (file-like) object. recoverable copies and copies with a True value in the *overwrite* argument are not allowed; always use a destination file name if you need a recoverable/overwriting copy.
- **transferred\_bytes** (int) – file offset at which the copy process starts. if not passed for recoverable copies, then *copy\_bytes* will determine this value from the file length of the destination file.
- **total\_bytes** (int) – source file size in bytes (needed only if *src\_file* is a stream).
- **buf\_size** (int) – size of copy buffer/chunk in bytes (that get copied before each progress callback).
- **overwrite** (bool) – pass True to allow to overwrite of destination file. if the destination file exists already then this function will return an error (when this argument get not passed or is False).
- **move\_file** (bool) – pass True to delete source file on complete copying (only works if source is a stream).
- **recoverable** (bool) – pass True to allow recoverable file copy (only working if source is a stream).
- **errors** (Optional[List[str]]) – pass empty list to get a list of detailed error messages.
- **progress\_func** (Callable) – optional callback to dispatch or break/cancel the copy progress for large files. if the callback returns a non-empty value it will be interpreted as cancel reason, the copy process will be stopped and an error will be returned.
- **progress\_kwargs** – optional additional kwargs passed to the progress function. the kwargs *total\_bytes* and *transferred\_bytes* will be updated before the callback.

**Return type**`str`**Returns**

destination file name/stream as string or empty string on error.

---

**Hint:** this function is extending the compatible Python functions `shutil.copyfileobj()`, `shutil.copyfile()`, `shutil.copy()`, `shutil.copy2()` and `http.server.SimpleHTTPRequestHandler.copyfile()` with recoverability and a progress callback. it can also be used as argument for the `copy_function` parameter of e.g. `shutil.copytree()` and `shutil.move()`.

---

**file\_lines**(*file\_path*, *encoding=None*)

returning lines of the text file specified by *file\_path* argument as tuple.

**Parameters**

- **file\_path** (str) – file path/name to parse/load.
- **encoding** (Optional[str]) – encoding used to load and convert/interpret the file content.

**Return type**`Tuple[str, ...]`**Returns**

tuple of the lines found in the specified file or empty tuple if the file could not be found or opened.

**file\_transfer\_progress**(*transferred\_bytes*, *total\_bytes=0*, *decimal\_places=3*)

return string to display the transfer progress of transferred bytes in short and user readable format.

**Parameters**

- **transferred\_bytes** (int) – number of transferred bytes.
- **total\_bytes** (int) – number of total bytes.
- **decimal\_places** (int) – number of decimal places (should be between 0 and 3).

**Return type**`str`**Returns**

formatted string to display progress of currently running transfer.

**read\_file\_text**(*file\_path*, *encoding=None*, *error\_handling='ignore'*)

returning content of the text file specified by *file\_path* argument as string.

**Parameters**

- **file\_path** (str) – file path/name to load into a string.
- **encoding** (Optional[str]) – encoding used to load and convert/interpret the file content.
- **error\_handling** (str) – pass 'strict' or *None* to return *None* (instead of an empty string) for the cases where either a decoding *ValueError* exception or any *OSError*, *FileNotFoundError* or *PermissionError* exception got raised. the default value 'ignore' will ignore any decoding errors (missing some characters) and will return an empty string on any file/os exception.

**Return type**`Optional[str]`

**Returns**

file content string. if the file could not be decoded, found or opened, then return empty string or None (None only if 'strict' got passed to the [error\\_handling](#) parameter).

**write\_file\_text**(*text\_or\_lines*, *file\_path*, *encoding=None*)

write the passed text string or list of line strings into the text file specified by *file\_path* argument.

**Parameters**

- **text\_or\_lines** (Union[str, List[str], Tuple[str]]) – new file content either passed as string or list of line strings (will be concatenated with the line separator of the current OS: os.linesep).
- **file\_path** (str) – file path/name to write the passed content into (overwriting any previous content!).
- **encoding** (Optional[str]) – encoding used to write/convert/interpret the file content to write.

**Return type**

bool

**Returns**

True if the content got written to the file, False on any file/OS error.

**class RegisteredFile**(*file\_path*, *\*\*kwargs*)

Bases: [object](#)

represents a single file - see also [registered file](#) examples.

**\_\_init\_\_**(*file\_path*, *\*\*kwargs*)

initialize registered file\_obj instance.

**Parameters**

- **file\_path** (str) – file path string.
- **kwargs** – not supported, only there to have compatibility to [CachedFile](#) to detect invalid kwargs.

**path**: str

file path

**stem**: str

file basename without extension

**ext**: str

file name extension

**properties**: Dict[str, Union[int, float, str]]

file properties

**\_\_eq\_\_**(*other*)

allow equality checks.

**Parameters**

**other** (Union[str, RegisteredFile, CachedFile, Path, PurePath, Any]) – other file object to compare this instance with.

**Return type**

bool

**Returns**

True if both objects are of this type and contain a file with the same path, else False.

**\_\_repr\_\_()**

for config var storage and eval recovery.

**Returns**

evaluatable/recoverable representation of this object.

**\_\_str\_\_()**

return file path.

**Returns**

file path string of this file object.

**add\_property(property\_name, str\_value)**

add a property to this file object instance.

**Parameters**

- **property\_name** *(str)* – stem of the property to add.
- **str\_value** *(str)* – literal of the property value (int/float/str type will be detected).

**\_\_hash\_\_** = None

**\_default\_object\_loader(file\_obj)**

file object loader that is opening the file and keeping the handle of the opened file.

**Parameters**

**file\_obj** *(Union[str, RegisteredFile, CachedFile, Path, PurePath, Any])* – file object (path string or obj with *path* attribute holding the complete file path).

**Returns**

file handle to the opened file.

**class CachedFile(file\_path, object\_loader=<function \_default\_object\_loader>, late\_loading=True)**

Bases: *RegisteredFile*

represents a cacheables registered file object - see also *cached file* examples.

**\_\_init\_\_(file\_path, object\_loader=<function \_default\_object\_loader>, late\_loading=True)**

create cached file object instance.

**Parameters**

- **file\_path** *(str)* – path string of the file.
- **object\_loader** *(Callable[[CachedFile], Any])* – callable converting the file\_obj into a cached object (available via *loaded\_object*).
- **late\_loading** *(bool)* – pass False to convert/load file\_obj cache early, directly at instantiation.

**path:** *str*

file path

**stem:** *str*

file basename without extension

**ext:** *str*

file name extension



**properties:** `Dict[str, Union[int, float, str]]`

file properties

**property loaded\_object:** `Any`

loaded object class instance property.

**Returns**

loaded and cached file object.

## 4.8 ae.paths

### 4.8.1 generic file path helpers

this pure python namespace portion is providing useful *path helper functions* as well as *generic system paths* for most platforms, like e.g.:

- android OS
- iOS
- linux
- macOS
- Windows

the only external hard dependency of this module are the ae namespace portions *ae.base* and *ae.files*. optional dependencies are:

- on android OS the PyPi package *jnius*, needed by the functions *user\_data\_path()* and *user\_docs\_path()*.
- the *plyer* PyPi package, needed by the function *add\_common\_storage\_paths()*.

### path helper functions

the generator functions *coll\_files()*, *coll\_folders()* and *coll\_items()* are building the fundament for most of the file and folder collection functionality, provided by this module. for example the function *path\_files()* is using these generators to determine the files within a folder structure that are matching the specified wildcards and *path part placeholders*. similarly the function *path\_folders()* for folders, and *path\_items()* to collect both, file and folder names.

use the functions *copy\_files()* and *move\_files()* to duplicate and move multiple files or entire file path trees. these two functions are based on *copy\_file()* and *move\_file()*. the functions *copy\_tree()* and *move\_tree()* providing an alternative way to copy or move entire directory trees.

the helper function *normalize()* converts path strings containing path placeholders into regular path strings, resolving symbolic links, or is converting paths string from absolute paths to relative paths and vice versa.

to determine if the path of a file or folder is matching a glob-like path pattern/mask with wildcards, the functions *path\_match()* and *paths\_match()* can be used. useful specially for cases where you don't have direct access to the file system.

file paths for series of files, e.g. for logging, can be determined via the *series\_file\_name()* function.

the function *skip\_py\_cache\_files()* can be used in path file collections to skip the files situated in the Python cache folder (*PY\_CACHE\_FOLDER* respectively *\_\_pycache\_\_*).

## generic system paths

generic system paths are determined by the following helper functions:

- `app_data_path()`: application data path.
- `app_docs_path()`: application documents path.
- `user_data_path()`: user data path.
- `user_docs_path()`: user documents path.

these system paths together with additional generic paths like e.g. the current working directory, storage paths provided by the *plyer* package, or user and application paths, are provided as *path placeholders*, which get stored within the `PATH_PLACEHOLDERS` dict by calling the function `add_common_storage_paths()`.

`path_name()` and `placeholder_path()` are converting regular path strings or parts of it into path placeholders.

## file/folder collection and classification

more complex collections of files and folder paths, and the grouping of them, can be done with the classes *Collector*, described in the underneath section *collecting files*, and *FilesRegister*, described in the section *files register*.

use the *Collector class* for temporary quick file path searches on your local file systems as well as on remote servers/hosts. one implementation example is e.g. the method `deployed_code_files()` of the *aedev.pythonanywhere* module.

the class *FilesRegister* helps you to create and cache file path registers permanently, to quickly find at any time the best fitting match for a requested purpose. for example the *gui\_app* module is using it to dynamically select image/font/audio/... resource files depending on the current user preferences, hardware and/or software environment.

## collecting files

to collect file names in the current working directory create an instance of the *Collector* class and call its `collect()` method with a file or folder path, which can contain wildcards:

```
.. code-block:: python
```

```
from ae.paths import Collector coll = Collector() coll.collect('*.*.png') image_files_list = coll.files
```

after that a list containing the found file names can then be retrieved from the *files* attribute.

`collect()` can be called multiple times to accumulate and extend the *files* list:

```
.. code-block:: python
```

```
coll = Collector() coll.collect('.png') coll.collect('.jpg') image_files_list = coll.files
```

multiple calls of *Collector.collect()* can be joined into one code line, because it is returning its instance. the following statement is equivalent to the last example:

```
.. code-block:: python
```

```
image_files_list = Collector().collect('.png').collect('.jpg').files
```

by specifying the `**` wildcard entire folder trees can be scanned. the following example is collection all the files, including the hidden ones, in the folder tree under the current working directory:

```
.. code-block:: python
```

```
test_files = Collector().collect('/.*').collect('/.*').files
```

**Hint:** the second call of the `Collector.collect()` method in this example has to be done only if you are using Python's `glob.glob()` as the searcher callback, which excludes hidden files (with a leading dot) from to match the `*` wildcard.

`Collector` is by default only returning files. to also collect folder paths in a deep folder tree, you have to pass the collecting generator function to the optional `item_collector` parameter of the `Collector` class. the accumulated files and folders can then be retrieved from their respective instance attributes `files`, `paths` and `selected`:

```
.. code-block:: python
```

```
coll = Collector(item_collector=coll_items) coll.collect(...) ... files = coll.files folders = coll.paths
file_and_folder_items = coll.selected
```

the found files are provided by the `Collector.files` instance attribute. found folders will be separately collected within the `Collector` instance attribute `paths`. the `selected` attribute contains all found files and folders in a single list.

### collect from multiple locations

in a single call of `collect()`, providing the method parameters `append` or `select`, you can scan multiple combinations of path prefixes and suffixes, which can both contain wildcards and folder names, whereas the suffixes containing also parts of the file names to search for.

**Hint:** the wildcards `*`, `**` and `?` are allowed in the prefixes as well as in suffixes.

the resulting file paths are relative or absolute, depending on if the specified prefix(es) containing absolute or relative paths.

in the following example determines the relative paths of all folders directly underneath the current working directory with a name that contains the string `'xxx'` or is starting with `'yyy'` or is ending with `'zzz'`:

```
.. code-block:: python
```

```
coll = Collector(item_collector=coll_folders) coll.collect('', append=('xxx', 'yyy*', '*zzz')) folders =
coll.paths
```

**Hint:** replace empty string in the first argument of `collect()` with `'{cwd}'` to get absolute paths.

the following example is collecting the absolute paths of files with the name `xxx.cfg` from the first found location/folder, starting to search in the current working directory, then in the folder above the application data folder, and finally in a folder with the name of the main application underneath the user data folder:

```
.. code-block:: python
```

```
coll = Collector() coll.collect("{cwd}", "{app}/..", "{usr}/{main_app_name}", append="xxx.cfg")
found_files = coll.files
```

**latest :**

---

to set or change the generic path placeholder parts values, e.g. of the main application name (*{main\_app\_name}*) and the application data path (*{app}*), you simply specify their corresponding values as kwargs in the construction of the *Collector* instance:

```
.. code-block:: python
```

```
coll = Collector(main_app_name=..., app=...)
```

additionally you can specify any other path placeholders that will be automatically used and replaced by the *Collector* instance:

```
.. code-block:: python
```

```
coll = Collector(any_other_placeholder=...)
```

by default only the found file(s)/folder(s) of the first combination will be collected. to collect all files instead, pass an empty tuple to the method argument *only\_first\_of* of :meth:`~Collector.collect`:

```
.. code-block:: python
```

```
coll.collect(..., append=..., [select=..., ] only_first_of=())
```

add one of the strings *'prefix'*, *'append'* or *'select'* to the *only\_first\_of* tuple argument to collect only the files/folders of the first combination of the specified prefixes, append-suffixes and select-suffixes.

by using the *select* argument the found files and folders will additionally be collected in the *Collector* instance attribute *selected*.

combinations collected via the *select* argument that are not existing will be logged. the results are provided by the instance attributes *failed*, *prefix\_failed* and *suffix\_failed*.

## files register

a files register is an instance of the *FilesRegister*, providing property based file collection and selection, which is e.g. used by the *ae.gui\_app* ae namespace portion to find and select resource files like icon/image or sound files.

files can be collected from various places by a single instance of the class *FilesRegister*:

```
.. code-block:: python
```

```
from ae.paths import FilesRegister
```

```
file_reg = FilesRegister('first/path/to/collect') file_reg.add_paths('second/path/to/collect/files/from')
```

```
registered_file = file_reg.find_file('file_name')
```

in this example the *FilesRegister* instance collects all files that are existing in any sub-folders underneath the two provided paths. then the *find\_file()* method will return a file object of type *RegisteredFile* of the last collected file with the stem (base name w/o extension) *'file\_name'*.

multiple files with the same stem can be collected and registered e.g. with different formats, to be selected by the app by their different properties, which are specified in the folder names underneath the collected paths. assuming your application is providing an icon image in two sizes, provided within the following folder structure, situated in the current working directory:

```
resources/  
  size_72/  
    app_icon.jpg
```

(continues on next page)

(continued from previous page)

```
size_150/
  app_icon.png
```

when you then create an instance of `FilesRegister` both image files from the `resources` folder will get registered, interpreting the sub-folder names (`size_*`) as properties or attributes for the registered files, where `size` will result as the property name and the string after the underscore as the property value.

to retrieve the paths of the application image file with the size 72, call the `find_file()` method:

```
.. code-block:: python
```

```
file_reg = FilesRegister('resources') app_icon_image_path = file_reg.find_file('app_icon', dict(size=72))
```

as a shortcut you can alternatively call the object directly (leaving `find_file` away):

```
.. code-block:: python
```

```
app_icon_image_path = file_reg('app_icon', dict(size=150))
```

the resulting file path in `app_icon_image_path` will be “`resources/size_72/app_icon.jpg`” in the penultimate example and “`resources/size_150/app_icon.png`” in the last example.

an instance of `FilesRegister` (`file_reg`) behaves like a dict object, where the item key is the file stem (`'app_icon'`) and the item value is a list of instances of `RegisteredFile`. both files in the resources folder are provided as one dict item:

```
.. code-block:: python
```

```
file_reg = FilesRegister('resources') assert 'app_icon' in file_reg assert len(file_reg) == 1 assert
len(file_reg['app_icon']) == 2 assert isinstance(file_reg['app_icon'][0], RegisteredFile)
```

for more complex selections you can use callables passed into the `property_matcher` and `file_sorter` arguments of `find_file()`.

## Module Attributes

<i>APPEND_TO_END_OF_FILE_LIST</i>	special flag default value for the <i>first_index</i> argument of the <i>add_*</i> methods of <i>FilesRegister</i> to append new file objects to the end of the name's register file object list.
<i>INSERT_AT_BEGIN_OF_FILE_LIST</i>	special flag default value for the <i>first_index</i> argument of the <i>add_*</i> methods of <i>FilesRegister</i> to insert new file objects always at the begin of the name's register file object list.
<i>COLLECTED_FOLDER</i>	item type value for a folder directory node
<i>CollArgType</i>	argument type passed to most callable arguments of <i>coll_items()</i> except of <i>searcher</i>
<i>CollCreatorReturnType</i>	type of the return value of the <i>creator</i> callable, which will be returned to the caller
<i>CollYieldType</i>	type of collected item, which is either <i>COLLECTED_FOLDER</i> for a folder or the file extension for a file
<i>CollYieldItems</i>	type of collected item iterator yielding tuples of (CollYieldType, item[_path])
<i>SearcherRetType</i>	type of the return value of the callable <i>searcher</i> argument of <i>coll_items()</i>
<i>SearcherType</i>	type of the callable <i>searcher</i> argument of <i>coll_items()</i>
<i>copy_file</i> (src, dst, *[, follow_symlinks])	Copy data and metadata.
<i>copy_tree</i> (src, dst[, symlinks, ignore, ...])	Recursively copy a directory tree and return the destination directory.
<i>move_file</i> (src, dst[, copy_function])	Recursively move a file or directory to another location.
<i>move_tree</i> (src, dst[, copy_function])	Recursively move a file or directory to another location.
<i>PATH_PLACEHOLDERS</i>	placeholders of user-, os- and app-specific system paths and file name parts

## Functions

<code>add_common_storage_paths()</code>	add common storage paths to <i>PATH_PLACEHOLDERS</i> depending on the operating system (OS).
<code>app_data_path()</code>	determine the os-specific absolute path of the {app} directory where user app data can be stored.
<code>app_docs_path()</code>	determine the os-specific absolute path of the {ado} directory where user documents app are stored.
<code>coll_files(file_mask[, file_class])</code>	determine existing file(s) underneath the folder specified by <i>file_mask</i> .
<code>coll_folders(folder_mask[, folder_class])</code>	determine existing folder(s) underneath the folder specified by <i>folder_mask</i> .
<code>coll_item_type(item_path)</code>	classify path item to be either file, folder or non-existent/excluded/skipped.
<code>coll_items(item_mask[, searcher, selector, ...])</code>	determine path-/file-like item(s) specified with optional wildcards by <i>item_mask</i> .
<code>copy_files(src_folder, dst_folder[, ...])</code>	copy files from <i>src_folder</i> into optionally created <i>dst_folder</i> , optionally overwriting destination files.
<code>move_files(src_folder, dst_folder[, overwrite])</code>	move files from <i>src_folder</i> into optionally created <i>dst_folder</i> , optionally overwriting destination files.
<code>normalize(path[, make_absolute, ...])</code>	normalize/transform path replacing <i>PATH_PLACEHOLDERS</i> and the tilde character (for home folder).
<code>path_files(file_mask[, file_class])</code>	determine existing file(s) underneath the folder specified by <i>file_mask</i> .
<code>path_folders(folder_mask[, folder_class])</code>	determine existing folder(s) underneath the folder specified by <i>folder_mask</i> .
<code>path_items(item_mask[, selector, creator])</code>	determine existing file/folder item(s) underneath the folder specified by <i>item_mask</i> .
<code>path_join(*parts)</code>	join path parts preventing trailing path separator if last part is empty string.
<code>path_match(path, mask)</code>	return True if the specified path matches the specified path mask/pattern.
<code>path_name(path)</code>	determine placeholder key name of the specified path.
<code>paths_match(paths, masks)</code>	filter the paths matching at least one of the specified glob-like wildcard masks.
<code>placeholder_key(path)</code>	determine <i>PATH_PLACEHOLDERS</i> key of specified path.
<code>placeholder_path(path)</code>	replace begin of path string with the longest prefix found in <i>PATH_PLACEHOLDERS</i> .
<code>series_file_name(file_path[, digits, ...])</code>	determine non-existent series file name with a unique series index.
<code>skip_py_cache_files(file_path)</code>	file exclude callback for the files under Python's cache folders.
<code>user_data_path()</code>	determine the os-specific absolute path of the {usr} directory where user data can be stored.
<code>user_docs_path()</code>	determine the os-specific absolute path of the {doc} directory where the user is storing the personal documents.

latest :

---

## Classes

<code>Collector([item_collector])</code>	file/folder collector class
<code>FilesRegister(*add_path_args[, ...])</code>	file register catalog - see also <i>files register</i> examples.

**APPEND\_TO\_END\_OF\_FILE\_LIST = 9223372036854775807**

special flag default value for the *first\_index* argument of the *add\_\** methods of *FilesRegister* to append new file objects to the end of the name's register file object list.

**INSERT\_AT\_BEGIN\_OF\_FILE\_LIST = -9223372036854775807**

special flag default value for the *first\_index* argument of the *add\_\** methods of *FilesRegister* to insert new file objects always at the begin of the name's register file object list.

**COLLECTED\_FOLDER = None**

item type value for a folder|directory|node

**CollArgType = typing.Any**

argument type passed to most callable arguments of *coll\_items()* except of *searcher*

**CollCreatorReturnType = typing.Any**

type of the return value of the *creator* callable, which will be returned to the caller

**CollYieldType**

type of collected item, which is either *COLLECTED\_FOLDER* for a folder or the file extension for a file

alias of *Optional[str]*

**CollYieldItems**

type of collected item iterator yielding tuples of (CollYieldType, item[\_path])

alias of *Iterable[tuple[Optional[str], Any]]*

**SearcherRetType**

type of the return value of the callable *searcher* argument of *coll\_items()*

alias of *Iterable[Any]*

**SearcherType**

type of the callable *searcher* argument of *coll\_items()*

alias of *Callable[[str], Iterable[Any]]*

**coll\_item\_type(item\_path)**

classify path item to be either file, folder or non-existent/excluded/skipped.

**Parameters**

*item\_path* *(str)* – file/folder path string.

**Return type**

*Optional[str]*

**Returns**

COLLECTED\_FOLDER for folders, the file extension for files or None if not found.

**coll\_items(item\_mask, searcher=functools.partial(<function glob>, recursive=True), selector=<class 'str'>, type\_detector=<function coll\_item\_type>, creator=<class 'str'>, \*\*creator\_kwargs)**

determine path-/file-like item(s) specified with optional wildcards by *item\_mask*.

**Parameters**



- **item\_mask** (str) – file path mask with optional `glob()` wildcards, the '~' shortcut for the home folder path and any *path placeholders*, which is specifying the files/folders to collect. use the '\*\*' glob as path to include also items from sub-folders deeper than one level.
- **searcher** (Callable[[str], Iterable[Any]]) – callable to convert/resolve path with optional wildcards in `item_mask` into multiple item file/folder path strings.
- **type\_detector** (Callable[[Any], Optional[str]]) – callable to typify/classify a found item to be stored in the first tuple items of the returned list. if not passed then `path_item_type()` will be used.
- **selector** (Callable[[Any], Union[bool, Any]]) – called with each found file/folder name to check if it has to be added to the returned list. the default argument (str) results in returning every file/folder found by `glob()`.
- **creator** (Callable[[Any], Any]) – each found file/folder will be passed as argument to this class/callable and the instance/return-value will be appended as an item to the returned item list. if not passed then the `str` class will be used, which means that the items of the returned list will be strings of the file/folder path and name. passing a class, like e.g. `ae.files.CachedFile`, `ae.files.CachedFile` or `pathlib.Path`, will create instances of this class. alternatively you can pass a callable which will be called on each found file/folder. in this case the return value of the callable will be inserted in the related item of the returned list. silly mypy does not support `Union[Type[Any], Callable[[str, KwArg()], Any]]`.
- **creator\_kwargs** – additional/optional kwargs passed onto the used `item_class` apart from the item name.

**Return type**

`Iterable[tuple[Optional[str], Any]]`

**Returns**

iterator/generator yielding found and selected file system items as instances of the item creator class (passed as `creator` argument, defaulting to the `str` class).

**coll\_files**(*file\_mask*, *file\_class*=<class 'str'>, *\*\*file\_kwargs*)

determine existing file(s) underneath the folder specified by *file\_mask*.

**Parameters**

- **file\_mask** (str) – glob file mask (with optional glob wildcards and *PATH\_PLACEHOLDERS*) specifying the files to collect (by default including the sub-folders).
- **file\_class** (Union[Type[Any], Callable]) – factory used for the returned list items (see *coll\_items.creator*). silly mypy does not support `Union[Type[Any], Callable[[str, KwArg()], Any]]`.
- **file\_kwargs** – additional/optional kwargs apart from file name passed onto the used `item_class`.

**Return type**

`Iterable[tuple[Optional[str], Any]]`

**Returns**

list of files of the class specified by *file\_mask*.

**coll\_folders**(*folder\_mask*, *folder\_class*=<class 'str'>, *\*\*folder\_kwargs*)

determine existing folder(s) underneath the folder specified by *folder\_mask*.

**Parameters**

- **folder\_mask** (str) – glob folder mask (with optional glob wildcards and *PATH\_PLACEHOLDERS*) specifying the folders to collect (by default including the sub-folders).
- **folder\_class** (Union[Type[Any], Callable]) – class or factory used for the returned list items (see *creator*). silly mypy does not support Union[Type[Any], Callable[[str, KwArg()], Any]].
- **folder\_kwargs** – additional/optional kwargs apart from file name passed onto the used item\_class.

**Return type**

Iterable[tuple[Optional[str], Any]]

**Returns**

list of folders of the class specified by *folder\_mask*.

**add\_common\_storage\_paths()**

add common storage paths to *PATH\_PLACEHOLDERS* depending on the operating system (OS).

the following storage paths are provided by the *plyer* PyPi package (not all of them are available in each OS):

- *application*: user application directory.
- *documents*: user documents directory.
- *downloads*: user downloads directory.
- *external\_storage*: external storage root directory.
- *home*: user home directory.
- *music*: user music directory.
- *pictures*: user pictures directory.
- *root*: root directory of the operating system partition.
- *sdcard*: SD card root directory (only available in Android if sdcard is inserted).
- *videos*: user videos directory.

additionally storage paths that are only available on certain OS (inspired by the method *get\_drives*, implemented in [https://github.com/kivy-garden/filebrowser/blob/master/kivy\\_garden/filebrowser/\\_\\_init\\_\\_.py](https://github.com/kivy-garden/filebrowser/blob/master/kivy_garden/filebrowser/__init__.py)):

- *Linux*: external storage devices/media mounted underneath the system partition root in /mnt or /media.
- *Apple Mac OSX or iOS*: external storage devices/media mounted underneath the system partition root in /Volume.
- *MS Windows*: additional drives mapped as the drive partition name.

**app\_data\_path()**

determine the os-specific absolute path of the {app} directory where user app data can be stored.

---

**Hint:** use *app\_docs\_path()* instead to get a more public path to the user.

---

**Return type**

str

**Returns**

path string of the user app data folder.

**app\_docs\_path()**

determine the os-specific absolute path of the {ado} directory where user documents app are stored.

---

**Hint:** use `app_data_path()` instead to get a more hidden path to the user.

---

**Return type**

`str`

**Returns**

path string of the user documents app folder.

**copy\_file(src, dst, \*, follow\_symlinks=True)**

alias for `shutil.copy2()` (compatible to `shutil.copy()`, `shutil.copyfile()` and `ae.files.copy_bytes()`).

**copy\_tree(src, dst, symlinks=False, ignore=None, copy\_function=<function copy2>, ignore\_dangling\_symlinks=False, dirs\_exist\_ok=False)**

alias for `shutil.copytree()`.

**move\_file(src, dst, copy\_function=<function copy2>)**

alias for `shutil.move()` (see also `move_tree()`).

**move\_tree(src, dst, copy\_function=<function copy2>)**

another alias for `shutil.move()` (see also `move_file()`).

**copy\_files(src\_folder, dst\_folder, overwrite=False, copier=<function copy2>)**

copy files from `src_folder` into optionally created `dst_folder`, optionally overwriting destination files.

**Parameters**

- **src\_folder** *(str)* – path to source folder/directory where the files get copied from. placeholders in `PATH_PLACEHOLDERS` will be recognized and substituted.
- **dst\_folder** *(str)* – path to destination folder/directory where the files get copied to. all placeholders in `PATH_PLACEHOLDERS` are recognized and will be substituted.
- **overwrite** *(bool)* – pass True to overwrite existing files in the destination folder/directory. on False the files will only get copied if they not exist in the destination.
- **copier** *(Callable)* – copy/move function with `src_file` and `dst_file` parameters, returning file path/name.

**Return type**

`List[str]`

**Returns**

list of copied files, with their destination path.

**move\_files(src\_folder, dst\_folder, overwrite=False)**

move files from `src_folder` into optionally created `dst_folder`, optionally overwriting destination files.

**Parameters**

- **src\_folder** *(str)* – path to source folder/directory where the files get moved from. placeholders in `PATH_PLACEHOLDERS` will be recognized and substituted. please note that the source folders itself will neither be moved nor removed (but will be empty after the operation finished).

- **dst\_folder** (str) – path to destination folder/directory where the files get moved to. all placeholders in *PATH\_PLACEHOLDERS* are recognized and will be substituted.
- **overwrite** (bool) – pass True to overwrite existing files in the destination folder/directory. on False the files will only get moved if they not exist in the destination.

**Return type**`List[str]`**Returns**

list of moved files, with their destination path.

**normalize**(path, make\_absolute=True, remove\_base\_path="", remove\_dots=True, resolve\_sym\_links=True)  
normalize/transform path replacing *PATH\_PLACEHOLDERS* and the tilde character (for home folder).

**Parameters**

- **path** (str) – path string to normalize/transform.
- **make\_absolute** (bool) – pass False to not convert path to an absolute path.
- **remove\_base\_path** (str) – pass a valid base path to return a relative path, even if the argument values of *make\_absolute* or *resolve\_sym\_links* are True.
- **remove\_dots** (bool) – pass False to not replace/remove the . and .. placeholders.
- **resolve\_sym\_links** (bool) – pass False to not resolve symbolic links, passing True implies a True value also for the *make\_absolute* argument.

**Return type**`str`**Returns**

normalized path string: absolute if *remove\_base\_path* is empty and either *make\_absolute* or *resolve\_sym\_links* is True; relative if *remove\_base\_path* is a base path of *path* or if *path* got passed as relative path and neither *make\_absolute* nor *resolve\_sym\_links* is True.

**path\_files**(file\_mask, file\_class=<class 'str'>, \*\*file\_kwargs)

determine existing file(s) underneath the folder specified by *file\_mask*.

**Parameters**

- **file\_mask** (str) – glob file mask (with optional glob wildcards and *PATH\_PLACEHOLDERS*) specifying the files to collect (by default including the sub-folders).
- **file\_class** (Union[Type[Any], Callable]) – factory used for the returned list items (see *path\_items.creator*). silly mypy does not support Union[Type[Any], Callable[[str, KwArg()], Any]].
- **file\_kwargs** – additional/optional kwargs apart from file name passed onto the used item\_class.

**Return type**`List[Any]`**Returns**list of files of the class specified by *file\_mask*.

**path\_folders**(folder\_mask, folder\_class=<class 'str'>, \*\*folder\_kwargs)

determine existing folder(s) underneath the folder specified by *folder\_mask*.

**Parameters**

- **folder\_mask** (str) – glob folder mask (with optional glob wildcards and [PATH\\_PLACEHOLDERS](#)) specifying the folders to collect (by default including the sub-folders).
- **folder\_class** (Union[Type[Any], Callable]) – class or factory used for the returned list items (see [creator](#)). silly mypy does not support Union[Type[Any], Callable[[str, KwArg()], Any]].
- **folder\_kwargs** – additional/optional kwargs apart from file name passed onto the used item\_class.

**Return type**

List[Any]

**Returns**

list of folders of the class specified by folder\_mask.

**path\_items**(item\_mask, selector=<class 'str'>, creator=<class 'str'>, \*\*creator\_kwargs)

determine existing file/folder item(s) underneath the folder specified by item\_mask.

**Parameters**

- **item\_mask** (str) – file path mask (with optional glob wildcards and [PATH\\_PLACEHOLDERS](#)) specifying the files/folders to collect.
- **selector** (Callable[[str], Any]) – called with each found file/folder name to check if it has to be added to the returned list. the default argument (str) results in returning every file/folder found by glob().
- **creator** (Union[Type[Any], Callable]) – each found file/folder will be passed as argument to this class/callable and the instance/return-value will be appended as an item to the returned item list. if not passed then the str class will be used, which means that the items of the returned list will be strings of the file/folder path and name. passing a class, like e.g. [ae.files.CachedFile](#), [ae.files.CachedFile](#) or [pathlib.Path](#), will create instances of this class. alternatively you can pass a callable which will be called on each found file/folder. in this case the return value of the callable will be inserted in the related item of the returned list. silly mypy does not support Union[Type[Any], Callable[[str, KwArg()], Any]].
- **creator\_kwargs** – additional/optional kwargs passed onto the used item\_class apart from the item name.

**Return type**

List[Any]

**Returns**

list of found and selected items of the item class (item\_mask).

**path\_join**(\*parts)

join path parts preventing trailing path separator if last part is empty string.

**Parameters**

**parts** (str) – path parts to join.

**Return type**

str

**Returns**

joined path string.

---

**Hint:** although [os.path.join\(\)](#) is implemented in C, this function is faster:: .. code-block:: python

```
import os
import timeit
from ae.paths import path_join
paths_secs = timeit.timeit('path_join("test", "sub_test", "sub_sub_test")', globals=globals())
os_secs = timeit.timeit('os.path.join("test", "sub_test", "sub_sub_test")', globals=globals())
assert paths_secs < os_secs
```

even if you import `os.path.join()` without the namespace prefixes, like this:: .. code-block:: python

```
from os.path import join as os_join
os_secs = timeit.timeit('os_join("test", "sub_test", "sub_sub_test")',
globals=globals())
assert paths_secs < os_secs
```

```
_path_match_replacement = re.compile('(\\/\\\\\\\\\\\\\\\\*\\\\\\\\\\\\\\\\*|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*/\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\?|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\[\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\]|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\[\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\?\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\]|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\[!\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\[\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\])')

```

pre-compiled regular expression for `path_match()`, inspired by the great SO answer of Pugsley (see <https://stackoverflow.com/questions/27726545/63212852#63212852>)

**path\_match**(*path*, *mask*)

return True if the specified path matches the specified path mask/pattern.

## Parameters

- **path** (str) – path string to match.
- **mask** (str) – path mask/pattern including glob-like wildcards.

### Return type

bool

## Returns

True if the path specified by `path` matches the mask/pattern specified by the `mask` argument.

**path\_name**(*path*)

determine placeholder key name of the specified path.

## Parameters

**path** (str) – path string to determine name of (can contain placeholders).

### Return type

str

## Returns

name (respectively dict key in *PATH\_PLACEHOLDERS*) of the found path or empty string if not found.

**paths\_match**(*paths*, *masks*)

filter the paths matching at least one of the specified glob-like wildcard masks.

## Parameters

- **paths** (Sequence[str]) – sequence of path strings to be checked if they match at least one pattern/mask, specified by the `masks` argument.
- **masks** (Sequence[str]) – sequence of path masks/pattern with glob-like wildcards.

### Return type

```
Iterable[str]
```

## Returns

list of the paths specified by `paths` that match at least one mask, specified by the `masks` argument.

**placeholder\_key**(*path*)

determine *PATH\_PLACEHOLDERS* key of specified path.

**Parameters**

**path** (str) – path string starting with a [PATH\\_PLACEHOLDERS](#) path prefix.

**Return type**

str

**Returns**

placeholder key (if found as path prefix), else empty string.

**placeholder\_path(path)**

replace begin of path string with the longest prefix found in [PATH\\_PLACEHOLDERS](#).

**Parameters**

**path** (str) – path string (optionally including sub-folders and file name).

**Return type**

str

**Returns**

path string with replaced placeholder prefix (if found).

**series\_file\_name(file\_path, digits=2, marker=' ', create=False)**

determine non-existent series file name with a unique series index.

**Parameters**

- **file\_path** (str) – file path and name (optional with extension).
- **digits** (int) – number of digits used for the series index.
- **marker** (str) – marker that will be put at the end of the file name and before the series index.
- **create** (bool) – pass True to create the file (to reserve the series index).

**Return type**

str

**Returns**

file path extended with unique/new series index.

**skip\_py\_cache\_files(file\_path)**

file exclude callback for the files under Python's cache folders.

**Parameters**

**file\_path** (str) – path to file to check for exclusion, relative to the project root folder.

**Return type**

bool

**Returns**

True if the file specified in `file_path` has to be excluded, else False.

**user\_data\_path()**

determine the os-specific absolute path of the {usr} directory where user data can be stored.

---

**Hint:** this path is not accessible on Android devices, use [user\\_docs\\_path\(\)](#) instead to get a more public path to the user.

---

**Return type**

str

latest :

---

**Returns**

path string of the user data folder.

**user\_docs\_path()**

determine the os-specific absolute path of the {doc} directory where the user is storing the personal documents.

---

**Hint:** use `user_data_path()` instead to get a more hidden user data.

---

**Return type**

`str`

**Returns**

path string of the user documents folder.

```
PATH_PLACEHOLDERS = {'ado': '/home/docs/Documents/docs', 'app':  
'/home/docs/.config/docs', 'app_name': 'docs', 'cwd':  
'/home/docs/checkouts/readthedocs.org/user_builds/ae/checkouts/latest/docs', 'doc':  
'/home/docs/Documents', 'usr': '/home/docs/.config'}
```

placeholders of user-, os- and app-specific system paths and file name parts

```
class Collector(item_collector=<function coll_files>, **placeholders)
```

Bases: `object`

file/folder collector class

```
__init__(item_collector=<function coll_files>, **placeholders)
```

create new file/folder/item collector instance with individual (extended or overriding) placeholders.

**Parameters**

- **item\_collector** `Callable[[str], Iterable[tuple[Optional[str], Any]]]` – callable to determine the item type to collect. the default is the `coll_files()` function. pass e.g. `coll_folders()` to collect only folders or `coll_items()` to collect both (files and folders). overload the arguments of these functions (with `partial`) to adapt/change their default arguments.
- **placeholders** – all other kwargs are placeholders with their names:replacements as keys:values. the placeholders provided by `PATH_PLACEHOLDERS` are available too (but will be overwritten by these arguments).

**paths:** `List[Any]`

list of found/collected folders

**files:** `List[Any]`

list of found/collected files

**selected:** `List[Any]`

list of found/collected file/folder item instances

**failed**

number of not found select-combinations

**prefix\_failed:** `Dict[str, int]`

not found select-combinations count for each prefix



**suffix\_failed:** `Dict[str, int]`

not found select-combinations count for each suffix

### placeholders

path part placeholders of this Collector instance

**check\_add**(*item\_mask*, *select=False*)

check if item mask match file/folder(s) and if yes append accordingly to collecting instance lists.

#### Parameters

- **item\_mask** `(str)` – file/folder mask, optionally including wildcards in the glob.glob format.
- **select** `(bool)` – pass True to additionally add found files/folders into *selected*.

#### Return type

`bool`

#### Returns

True if at least one file/folder got found/added, else False.

**\_collect\_appends**(*prefix*, *appends*, *only\_first\_of*)

**\_collect\_selects**(*prefix*, *selects*, *only\_first\_of*)

**collect**(\**prefixes*, *append=()*, *select=()*, *only\_first\_of=('append', 'prefix', 'select')*)

collect additional files/folders by combining the given prefixes with all the given append/select suffixes.

---

**Note:** all arguments of this method can either be passed either as tuples, or for a single value as string.

---

#### Parameters

- **prefixes** `(str)` – tuple of file/folder paths to be used as prefix.
- **append** `(Union[str, Tuple[str, ...]])` – tuple of file/folder names to be used as append suffix.
- **select** `(Union[str, Tuple[str, ...]])` – tuple of file/folder names to be used as select suffix. in contrary to **append** is logging not found **prefixes** combinations in the instance attributes *failed*, *prefix\_failed* and *suffix\_failed*.
- **only\_first\_of** `(Union[str, Tuple[str, ...]])` – tuple with the strings *‘prefix’*, *‘append’* or *‘select’* or one of these strings. if it contains the string *‘prefix’* then only the files/folders of the first combination will be collected. if it contains *‘append’* then only the files/folders of the first combination done with the suffixes passed into the *append* argument will be collected. if it contains *‘select’* then only the files/folders of the first combination done with the suffixes passed into the *select* argument will be collected. pass empty tuple to collect all combinations.

#### Return type

*Collector*

each of the passed *prefixes* will be combined with the suffixes specified in *append* and in *select*. the resulting file/folder paths that are exist, will then be added to the appropriate instance attribute, either *files* for a file or *paths* for a folder.

additionally the existing file/folder paths from the combinations of *prefixes* and *select* will be added in the *selected* list attribute.

---

**Hint:** more details and some examples are available in the doc string of this [module](#).

---

**property error\_message:** `str`

returns an error message if an error occurred.

**Returns**

error message string if collection failure/error occurred, else an empty string.

**class FilesRegister**(\*add\_path\_args, property\_matcher=None, file\_sorter=None, \*\*add\_path\_kwargs)

Bases: `dict`

file register catalog - see also [files register](#) examples.

**\_\_init\_\_**(\*add\_path\_args, property\_matcher=None, file\_sorter=None, \*\*add\_path\_kwargs)

create files register instance.

this method gets redirected with [add\\_path\\_args](#) and [add\\_path\\_kwargs](#) arguments to [add\\_paths\(\)](#).

**Parameters**

- **add\_path\_args** – if passed then [add\\_paths\(\)](#) will be called with this args tuple.
- **property\_matcher** (Optional[Callable[[Union[str, [RegisteredFile](#), [CachedFile](#), Path, PurePath, Any]], bool]]) – property matcher callable, used as default value by [find\\_file\(\)](#) if not passed there.
- **file\_sorter** (Optional[Callable[[Union[str, [RegisteredFile](#), [CachedFile](#), Path, PurePath, Any]], Any]]) – file sorter callable, used as default value by [find\\_file\(\)](#) if not passed there.
- **add\_path\_kwargs** – passed onto call of [add\\_paths\(\)](#) if the [add\\_path\\_args](#) got provided by the caller.

**\_\_call\_\_**(\*find\_args, \*\*find\_kwargs)

add\_path\_args and kwargs will be completely redirected to [find\\_file\(\)](#).

**Return type**

Union[str, [RegisteredFile](#), [CachedFile](#), Path, PurePath, Any, None]

**add\_file**(file\_obj, first\_index=9223372036854775807)

add a single file to the list of this dict mapped by the file-name/stem as dict key.

**Parameters**

- **file\_obj** (Union[str, [RegisteredFile](#), [CachedFile](#), Path, PurePath, Any]) – either file path string or any object with a *stem* attribute.
- **first\_index** (int) – pass list index -n-1..n-1 to insert [file\\_obj](#) in the name's list. values greater than n (==len(file\_list)) will append the file\_obj to the end of the file object list and values less than n-1 will insert the file\_obj to the start of the file.

**add\_files**(files, first\_index=9223372036854775807)

add files from another [FilesRegister](#) instance.

**Parameters**

- **files** (Iterable[Union[str, [RegisteredFile](#), [CachedFile](#), Path, PurePath, Any]]) – iterable with file objects to be added.

- **first\_index** (int) – pass list index -n-1..n-1 to insert the first file\_obj in each name's register list. values greater than n (==len(file\_list)) will append the file\_obj to the end of the file object list. the order of the added items will be unchanged if this value is greater or equal to zero. negative values will add the items from *files* in reversed order and **after** the item specified by this index value (so passing -1 will append the items to the end in reversed order, while passing -(n+1) will insert them at the beginning in reversed order).

**Return type**`List[str]`**Returns**

list of paths of the added files.

**add\_paths**(\*file\_path\_masks, first\_index=9223372036854775807, file\_class=<class 'ae.files.RegisteredFile'>, \*\*init\_kwargs)

add files found in the folder(s) specified by the *file\_path\_masks* args.

**Parameters**

- **file\_path\_masks** (str) – file path masks (with optional wildcards and *PATH\_PLACEHOLDERS*) specifying the files to collect (by default including the sub-folders).
- **first\_index** (int) – pass list index -n-1..n-1 to insert the first file\_obj in each name's register list. values greater than n (==len(file\_list)) will append the file\_obj to the end of the file object list. the order of the added items will be unchanged if this value is greater or equal to zero. negative values will add the found items in reversed order and **after** the item specified by this index value (so passing -1 will append the items to the end in reversed order, while passing -(n+1) will insert them at the beginning in reversed order).
- **file\_class** (Type[Union[str, *RegisteredFile*, *CachedFile*, *Path*, *PurePath*, *Any*]]) – the used file object class (see *FileObject*). each found file object will be passed to the class constructor (callable) and added to the list which is an item of this dict.
- **init\_kwargs** – additional/optional kwargs passed onto the used *file\_class*. pass e.g. the object\_loader to use, if *file\_class* is *CachedFile* (instead of the default: *RegisteredFile*).

**Return type**`List[str]`**Returns**

list of paths of the added files.

**add\_register**(files\_register, first\_index=9223372036854775807)

add files from another *FilesRegister* instance.

**Parameters**

- **files\_register** (*FilesRegister*) – files register instance containing the file\_obj to be added.
- **first\_index** (int) – pass list index -n-1..n-1 to insert the first file\_obj in each name's register list. values greater than n (==len(file\_list)) will append the file\_obj to the end of the file object list. the order of the added items will be unchanged if this value is greater or equal to zero. negative values will add the found items in reversed order and **after** the item specified by this index value (so passing -1 will append the items to the end in reversed order, while passing -(n+1) will insert them at the beginning in reversed order).

**Return type**`List[str]`

**Returns**

list of paths of the added files.

**find\_file**(*name*, *properties=None*, *property\_matcher=None*, *file\_sorter=None*)

find file\_obj in this register via properties, property matcher callables and/or file sorter.

**Parameters**

- **name** (str) – file name (stem without extension) to find.
- **properties** (Optional[Dict[str, Union[int, float, str]]]) – properties to select the correct file.
- **property\_matcher** (Optional[Callable[[Union[str, RegisteredFile, CachedFile, Path, PurePath, Any]], bool]]) – callable to match the correct file.
- **file\_sorter** (Optional[Callable[[Union[str, RegisteredFile, CachedFile, Path, PurePath, Any]], Any]]) – callable to sort resulting match results.

**Return type**

Union[str, RegisteredFile, CachedFile, Path, PurePath, Any, None]

**Returns**

registered/cached file object of the first found/correct file.

**reclassify**(*file\_class=<class 'ae.files.CachedFile'>*, *\*\*init\_kwargs*)

re-instantiate all name's file registers items to instances of the class *file\_class*.

**Parameters**

- **file\_class** (Type[Union[str, RegisteredFile, CachedFile, Path, PurePath, Any]]) – the new file object class (see *FileObject*). each found file object will be passed to the class constructor (callable) and the return value will then replace the file object in the file list.
- **init\_kwargs** – additional/optional kwargs passed onto the used file\_class. pass e.g. the object\_loader to use, if *file\_class* is CachedFile (the default file object class).

## 4.9 ae.core

### 4.9.1 application core constants, helper functions and base classes

this module declares practical constants, tiny helper functions and app base classes, which are reducing the code of your application (and of other ae namespace modules/portions).

**core constants**

there are three debug level constants: *DEBUG\_LEVEL\_DISABLED*, *DEBUG\_LEVEL\_ENABLED* and *DEBUG\_LEVEL\_VERBOSE*. short names for all debug level constants are provided by the dict *DEBUG\_LEVELS*. the debug level of your application can be either set in your code or optionally data-driven externally (using the *config files* or *config options* of the module *ae.console*).

to use the *python logging module* in conjunction with this module the constant *LOGGING\_LEVELS* is providing a mapping between the debug levels and the python logging levels.

the encoding of strings into byte-strings (to output them to the console/stdout or to file contents) can be tricky sometimes. to not lose any logging output because of invalid characters this module will automatically handle any

`UnicodeEncodeError` exception for you. invalid characters will then automatically be converted to the default encoding (specified by `DEF_ENCODING`) with the default error handling method specified by `DEF_ENCODE_ERRORS` (both defined in the `ae.base` namespace portion/module).

## core helper functions

the `print_out()` function, which is fully compatible to python's `print()`, is using the encode helpers `force_encoding()` and `to_ascii()` to auto-correct invalid characters.

the function `hide_dup_line_prefix()` is very practical if you want to remove or hide redundant line prefixes in your log files, to make them better readable.

## application base classes

the classes `AppBase` and `SubApp` are applying logging and debugging features to your application. create in your application one instance of `AppBase` to represent the main application task. if your application needs a separate logging/debugging configuration for sub-threads or sub-tasks then create an `SubApp` instance for each of these sub-apps.

sub-apps are very flexible and not tied to any fix use-case. they can be created e.g. for each sub-task or application thread. you could also create a `SubApp` instance for each of your external systems, like a database server or to connect your application onto different test environments or to your live/production system (e.g. for system comparison and maintenance).

both application classes are automatically catching and handling any exceptions and run-time errors: only if any critical exception/error cannot be handled then the `shutdown()` method will make sure that all sub-apps and threads get terminated and joined. additionally all print-out buffers will be flushed to include all the info of the critical error (the last debug and error messages) into the standard error/output and into any activated log files.

## basic usage of an application base class

at the top of your python application main file/module create an instance of the class `AppBase`:

```
""" docstring at the top of the main module of your application """
from ae.core import AppBase

__version__ = '1.2.3'

ca = AppBase()
```

in the above example the `AppBase` instance will automatically use the docstring title of the module as application title and the string in the module variable `__version__` as application version. to overwrite these defaults pass your application title and version string via the arguments `app_title` and `app_version` to the instantiation of `AppBase`:

```
ca = AppBase(app_title="title of this app instance", app_version='3.2.1')
```

other automatically initialized instance attributes of `AppBase` are documented underneath in the `class docstring`. they include e.g. the *date and time when the instance got created*, the *name/id of this application instance* or the *application path*.

## application class hierarchy

for most use cases you will not instantiate from `AppBase` directly - instead you will instantiate one of the extended application classes that are inherited from this base class.

the class `ConsoleApp` e.g. inherits from `AppBase` and is adding configuration options and variables to it. so in your console application it is recommended to directly use instances of `ConsoleApp` instead of `AppBase`.

for applications with an GUI use instead one of the classes `KivyMainApp`, `EnamlMainApp` or `TogaMainApp`.

## application logging

print-outs are an essential tool for the debugging and logging of your application at run-time. in python the print-outs are done with the `print()` function or with the python `logging` module. these print-outs get per default send to the standard output and error streams of your OS and so displayed on your system console/shell. the `print_out()` function and the `print_out()` method of this `core` module are adding two more sophisticated ways for print-outs to the console/log-files.

using `AppBase` is making the logging much easier and also ensures that print-outs of any imported library or package will be included within your log files. this is done by redirecting the standard output and error streams to your log files with the help of the `_PrintingReplicator` class.

head-less server applications like web servers are mostly not allowed to use the standard output streams. for some these applications you could redirect the standard output and error streams to a log file by using the OS redirection character (`>`):

```
python your_application.py >log_std_out.log 2>log_std_err.log
```

but because most web servers doesn't allow you to use this redirection, you can alternatively specify the `suppress_stdout` parameter as `True` in the instantiation of an `AppBase` instance. additionally you can call the `init_logging()` method to activate a log file. after that all print-outs of your application and libraries will only appear in your log file.

also in complex applications, where huge print-outs to the console can get lost easily, you want to use a log file instead. but even a single log file can get messy to read, especially for multithreading server applications. for that `SubApp` is allowing you to create for each thread a separate sub-app instance with its own log file.

using this module ensures that any crashes or freezes happening in your application will be fully logged. apart from the gracefully handling of `UnicodeEncodeError` exceptions, the `Python faulthandler` will be enabled automatically to catch system errors and to dump a traceback of them to the console and any activated log file.

## activate ae log file

ae log files are text files using by default the encoding of your OS console/shell. to activate the redirection of your applications print-outs into an ae log file for a `AppBase` instance you simply specify the file name of the log file in the `init_logging()` method call:

```
app = AppBase()
app.init_logging(log_file_name='my_log_file.log')
```

## activate ae logging features

for multi-threaded applications include the thread-id of the printing thread automatically into your log files by passing a True value to the `multi_threading` argument. to additionally also suppress any print-outs to the standard output/error streams pass True to the `suppress_stdout` argument:

```
app = AppBase(multi_threading=True, suppress_stdout=True)
app.init_logging(log_file_name='my_log_file.log')
```

the ae log files provided by this module are automatically rotating if the size of a log file succeeds the value in MBytes defined in the `LOG_FILE_MAX_SIZE`. to adapt this value to your needs you can specify the maximum log file size in MBytes with the argument `log_file_size_max` in your call of `init_logging()`:

```
app.init_logging(log_file_name='my_log_file.log', log_file_size_max=9.)
```

by using the `ConsoleApp` class instead of `AppBase` you can alternatively store the logging configuration of your application within a *configuration variable* or a *configuration option*. the order of precedence to find the appropriate logging configuration of each app instance is documented [here](#).

## using python logging module

if you prefer to use instead the python logging module for the print-outs of your application, then pass a `python logging configuration dictionary` with the individual configuration of your logging handlers, files and loggers to the `py_logging_params` argument of the `init_logging()` method:

```
app.init_logging(py_logging_params=my_py_logging_config_dict)
```

passing the python logging configuration dictionary to one of the `AppBase` instances created by your application will automatically disable the ae log file of this instance.

## application debugging

to use the debug features of `core` you simply have to import the needed *debug level constant* to pass it at instantiation of your `AppBase` or `SubApp` class to the `debug_level` argument:

```
app = AppBase(..., debug_level=:data:`DEBUG_LEVEL_ENABLED`)      # same for
↪:class:`SubApp`
```

by passing `DEBUG_LEVEL_ENABLED` the print-outs (and log file contents) will be more detailed, and even more verbose if you use instead the debug level `DEBUG_LEVEL_VERBOSE`.

the debug level can be changed at any time in your application code by directly assigning the new debug level to the `debug_level` property. if you prefer to change the debug levels dynamically, then use the `ConsoleApp` instead of `AppBase`, because `ConsoleApp` provides this property as a *configuration file variable* and *command line option*. this way you can specify *the actual debug level* without the need to change (and re-build) your application code.

## Module Attributes

<code>DEBUG_LEVEL_DISABLED</code>	lowest debug level - only display logging levels ERROR/CRITICAL.
<code>DEBUG_LEVEL_ENABLED</code>	minimum debugging info - display logging levels WARNING or higher.
<code>DEBUG_LEVEL_VERBOSE</code>	verbose debug info - display logging levels INFO/DEBUG or higher.
<code>DEBUG_LEVELS</code>	numeric ids and names of all supported debug levels.
<code>LOGGING_LEVELS</code>	association between ae debug levels and python logging levels.
<code>HIDDEN_CREDENTIALS</code>	credential keys that are hidden in print/repr output (not if verbose)
<code>MAX_NUM_LOG_FILES</code>	maximum number of <i>ae log files</i>
<code>LOG_FILE_MAX_SIZE</code>	max.
<code>LOG_FILE_IDX_WIDTH</code>	width of rotating log file index within log file name; adding +3 to ensure index range up to factor 10 <sup>3</sup> .
<code>ori_std_out</code>	original sys.stdout on app startup
<code>ori_std_err</code>	original sys.stderr on app startup
<code>log_file_lock</code>	log file rotation multi-threading lock
<code>po(*objects[, sep, end, file, flush, ...])</code>	alias of function <code>print_out()</code>
<code>APP_KEY_SEP</code>	separator character used in <i>app_key</i> of <i>AppBase</i> instance
<code>app_inst_lock</code>	app instantiation multi-threading lock

## Functions

<code>activate_multi_threading()</code>	activate multi-threading for all app instances (normally done at main app startup).
<code>hide_dup_line_prefix(last_line, current_line)</code>	replace duplicate characters at the start of two strings with spaces.
<code>logger_late_init()</code>	check if logging modules got initialized already and if not then do it now.
<code>main_app_instance()</code>	determine the main instance of the <i>AppBase</i> in the current running application.
<code>po(*objects[, sep, end, file, flush, ...])</code>	alias of function <code>print_out()</code>
<code>print_out(*objects[, sep, end, file, flush, ...])</code>	universal/unbreakable print function - replacement for the built-in python function <code>print()</code> .
<code>registered_app_names()</code>	determine the app names of all registered/running applications.



## Classes

<code>AppBase([app_title, app_name, app_version, ...])</code>	provides easy logging and debugging for your application.
<code>SubApp([app_title, app_name, app_version, ...])</code>	separate/additional sub-app/thread/task with own/individual logging/debug configuration.

**DEBUG\_LEVEL\_DISABLED:** `int = 0`

lowest debug level - only display logging levels ERROR/CRITICAL.

**DEBUG\_LEVEL\_ENABLED:** `int = 1`

minimum debugging info - display logging levels WARNING or higher.

**DEBUG\_LEVEL\_VERBOSE:** `int = 2`

verbose debug info - display logging levels INFO/DEBUG or higher.

**DEBUG\_LEVELS:** `Dict[int, str] = {0: 'disabled', 1: 'enabled', 2: 'verbose'}`

numeric ids and names of all supported debug levels.

**LOGGING\_LEVELS:** `Dict[int, int] = {0: 30, 1: 20, 2: 10}`

association between ae debug levels and python logging levels.

**HIDDEN\_CREDENTIALS = ('password', 'token')**

credential keys that are hidden in print/repr output (not if verbose)

**hide\_dup\_line\_prefix**(*last\_line*, *current\_line*)

replace duplicate characters at the start of two strings with spaces.

### Parameters

- **last\_line** `(str)` – last line string (e.g. the last line of text/log file).
- **current\_line** `(str)` – current line string.

### Return type

`str`

### Returns

current line string but duplicate characters at the beginning are replaced by space chars.

**MAX\_NUM\_LOG\_FILES:** `int = 69`

maximum number of *ae log files*

**LOG\_FILE\_MAX\_SIZE:** `int = 15`

max. size in MB of rotating *ae log files*

**LOG\_FILE\_IDX\_WIDTH:** `int = 5`

width of rotating log file index within log file name; adding +3 to ensure index range up to factor 10<sup>3</sup>.

**ori\_std\_out:** `TextIO = <_io.TextIOWrapper name='<stdout>' mode='w' encoding='utf-8'>`

original sys.stdout on app startup

**ori\_std\_err:** `TextIO = <_io.TextIOWrapper name='<stderr>' mode='w' encoding='utf-8'>`

original sys.stderr on app startup

**log\_file\_lock:** `RLock = <unlocked _thread.RLock object owner=0 count=0>`

log file rotation multi-threading lock

**`_LOGGER = None`**

python logger for this module gets lazy/late initialized and only if requested by caller

**`logger_late_init()`**

check if logging modules got initialized already and if not then do it now.

**`_MULTI_THREADING_ACTIVATED: bool = False`**

flag if threading is used in your application

**`activate_multi_threading()`**

activate multi-threading for all app instances (normally done at main app startup).

**`_deactivate_multi_threading()`**

disable multi threading (needed to reset app environment in unit testing).

**`print_out(*objects, sep=' ', end='\n', file=None, flush=False, encode_errors_def='backslashreplace', logger=None, app=None, **kwargs)`**

universal/unbreakable print function - replacement for the **built-in python function `print()`**.

#### Parameters

- **`objects`** – tuple of objects to be printed. if the first object is a string that starts with a `\r` character then the print-out will be only sent to the standard output (and will not be added to any active log files - see also **`end`** argument).
- **`sep`** (**`str`**) – separator character between each printed object/string (def=" ").
- **`end`** (**`str`**) – finalizing character added to the end of this print-out (def="\n"). pass `\r` to suppress the print-out into *ae log file* or to any activated python logger - useful for console/shell processing animation (see `tcp.TcpServer.run()`).
- **`file`** (**`Optional[TextIO]`**) – output stream object to be printed to (def=None which will use standard output streams). if given then the redirection to all active log files and python logging loggers will be disabled (even if the **`logger`** argument is specified).
- **`flush`** (**`bool`**) – flush stream after printing (def=False).
- **`encode_errors_def`** (**`str`**) – default error handling to encode (def=:data:DEF\_ENCODE\_ERRORS).
- **`logger`** (**`Optional[Logger]`**) – used logger to output *objects* (def=None). ignored if the **`file`** argument gets specified/passed.
- **`app`** (**`Optional[AppBase]`**) – the app instance from where this print-out got initiated.
- **`kwargs`** – catch unsupported kwargs for debugging (all items will be printed to all the activated logging/output streams).

this function is silently handling and autocorrecting string encode errors for output/log streams which are not supporting unicode. any instance of **`AppBase`** is providing this function as a method with the ***same name***). it is recommended to call/use this instance method instead of this function.

in multithreading applications this function prevents dismembered/fluttered print-outs from different threads.

---

**Note:** this function has an alias named **`po()`**.

---

**`po(*objects, sep=' ', end='\n', file=None, flush=False, encode_errors_def='backslashreplace', logger=None, app=None, **kwargs)`**

alias of function **`print_out()`**

**APP\_KEY\_SEP:** `str = '@'`

separator character used in `app_key` of `AppBase` instance

**\_APP\_INSTANCES:** `WeakValueDictionary[str, AppBase] = <WeakValueDictionary>`

dict that is weakly holding references to all `AppBase` instances created at run time.

gets automatically initialized in `AppBase.__init__()` to allow log file split/rotation and `debug_level` access at application thread or module level.

the first created `AppBase` instance is called the main app instance. `_MAIN_APP_INST_KEY` stores the dict key of the main instance.

**\_MAIN\_APP\_INST\_KEY:** `str = ''`

key in `_APP_INSTANCES` of main `AppBase` instance

**app\_inst\_lock:** `RLock = <unlocked _thread.RLock object owner=0 count=0>`

app instantiation multi-threading lock

**main\_app\_instance()**

determine the main instance of the `AppBase` in the current running application.

**Return type**

`Optional[AppBase]`

**Returns**

main/first-instantiated `AppBase` instance or None (if app is not fully initialized yet).

**registered\_app\_names()**

determine the app names of all registered/running applications.

**Return type**

`List[str]`

**\_register\_app\_instance(app)**

register new `AppBase` instance in `_APP_INSTANCES`.

**Parameters**

`app` *(AppBase)* – `AppBase` instance to register

**\_unregister\_app\_instance(app\_key)**

unregister/remove `AppBase` instance from within `_APP_INSTANCES`.

**Parameters**

`app_key` *(str)* – app key of the instance to remove.

**Return type**

`Optional[AppBase]`

**Returns**

removed `AppBase` instance.

**\_shut\_down\_sub\_app\_instances(timeout=None)**

shut down all `SubApp` instances.

**Parameters**

`timeout` *(Optional[float])* – timeout float value in seconds used for the `SubApp` shutdowns and for the acquisition of the threading locks of *the ae log file* and the *app instances*.

**class \_PrintingReplicator(sys\_out\_obj=<\_io.TextIOWrapper name='<stdout>' mode='w' encoding='utf-8'>)**

Bases: `object`

replacement of standard/error stream replicating print-outs to all active logging streams (log files/buffers).

```
__init__(sys_out_obj=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='utf-8'>)
```

initialise a new T-stream-object

**Parameters**

**sys\_out\_obj** *(TextIO)* – standard output/error stream to be replicated (def=sys.stdout)

```
write(any_str)
```

write string to ae logging and standard output streams.

automatically suppressing UnicodeEncodeErrors if console/shell or log file has different encoding by forcing re-encoding with DEF\_ENCODE\_ERRORS.

**Parameters**

**any\_str** *(Union[str, bytes])* – string or bytes to output.

**Return type**

*None*

```
__getattr__(attr)
```

get attribute value from standard output stream.

**Parameters**

**attr** *(str)* – name of the attribute to retrieve/return.

**Return type**

*Any*

**Returns**

value of the attribute.

```
_APP_THREADS: WeakValueDictionary[int, Thread] = <WeakValueDictionary>
```

weak dict to keep the references of all application threads. added to prevent the joining of unit testing threads in the test teardown (resetting app environment).

```
_register_app_thread()
```

add new app thread to \_APP\_THREADS if not already added.

```
_join_app_threads(timeout=None)
```

join/finish all app threads and finally deactivate multi-threading.

**Parameters**

**timeout** *(Optional[float])* – timeout float value in seconds for thread joining (def=None - block/no-timeout).

---

**Note:** this function has to be called by the main app instance only.

---

```
class AppBase(app_title="", app_name="", app_version="", sys_env_id="", debug_level=0, multi_threading=False, suppress_stdout=False)
```

Bases: *object*

provides easy logging and debugging for your application.

most applications only need a single instance of this class; apps with threads could create separate instances for each thread.

instance Attributes (ordered alphabetically - ignoring underscore characters):

- *app\_key* id/key of this application instance.
- *app\_name* basename (without the file name extension) of the executable.

- `app_path` file path of app executable.
- `app_title` application title/description.
- `app_version` application version (set via the `AppBase.app_version` argument).
- `debug_level` debug level of this instance.
- `_last_log_line_prefix` last ae log file line prefix that got print-out to the log of this app instance.
- `_log_buf_stream` ae log file buffer stream.
- `_log_file_index` index of the current rotation ae log file backup.
- `_log_file_name` path and file name of the ae log file.
- `_log_file_size_max` maximum size in MBytes of an ae log file.
- `_log_file_stream` ae log file TextIO output stream.
- `_log_with_timestamp` log timestamp line prefix if True or a non-empty strftime compatible format string.
- `py_log_params` python logging config dictionary.
- `_nul_std_out` null stream used to prevent print-outs to `standard output`.
- `_shut_down` flag set to True if this application instance got already shutdown.
- `startup_beg` datetime of begin of the instantiation/startup of this app instance.
- `startup_end` datetime of end of the instantiation/startup of this application instance.
- `suppress_stdout` flag set to True if this application does not print to stdout/console.
- `sys_env_id` system environment id of this application instance.

```

_last_log_line_prefix: str = ''
    prefix of the last printed log line

_log_buf_stream: Optional[StringIO] = None
    log file buffer stream instance

_log_file_stream: Optional[TextIO] = None
    log file stream instance

_log_file_index: int = 0
    log file index (for rotating logs)

_log_file_size_max: float = 15
    maximum log file size in MBytes (rotating log files)

_log_file_name: str = ''
    log file name

_log_with_timestamp: Union[bool, str] = False
    True of strftime format string to enable timestamp

_nul_std_out: Optional[TextIO] = None
    logging null stream

py_log_params: Dict[str, Any] = {}
    dict of config parameters for py logging

_shut_down: bool = False
    True if this app instance got shut down already

```

```
__init__(app_title="", app_name="", app_version="", sys_env_id="", debug_level=0, multi_threading=False,
        suppress_stdout=False)
```

initialize a new [AppBase](#) instance.

#### Parameters

- **app\_title** (str) – application title/description setting the attribute [app\\_title](#). if not specified then the docstring of your app's main module will be used (see [example](#)).
- **app\_name** (str) – application instance name to set the attribute [app\\_name](#). if not specified then base name of the main module file name will be used.
- **app\_version** (str) – application version string to set the attribute [app\\_version](#). if not specified then value of a global variable with the name `__version__` will be used (if declared in the actual call stack).
- **sys\_env\_id** (str) – system environment id to set the instance attribute [sys\\_env\\_id](#). the default value of this argument is an empty string.
- **debug\_level** (int) – default debug level to set the instance attribute [debug\\_level](#). the default value of this argument is [DEBUG\\_LEVEL\\_DISABLED](#).
- **multi\_threading** (bool) – pass True if instance is used in multi-threading app.
- **suppress\_stdout** (bool) – pass True (for wsgi apps) to prevent any python print outputs to stdout.

**startup\_beg:** [datetime](#)

begin of app startup datetime

**app\_path:** [str](#)

path to folder of your main app code file

**app\_title:** [str](#) = ''

title/description of this app instance

**app\_name:** [str](#) = ''

name of this app instance

**app\_version:** [str](#) = ''

version of this app instance

**\_debug\_level:** [int](#) = 2

debug level of this app instance

**sys\_env\_id:** [str](#) = ''

system environment id of this app instance

**suppress\_stdout:** [bool](#) = True

flag to suppress prints to stdout

**startup\_end:** [Optional\[datetime\]](#) = None

end datetime of the application startup

**\_\_del\_\_()**

deallocate this app instance by calling [AppBase.shutdown\(\)](#).

**property active\_log\_stream:** [StringIO](#) | [TextIO](#) | None

check if ae logging is active and if yes then return the currently used log stream (read-only property).

**Returns**

log file or buf stream if logging is activated, else None.

**property app\_key:** `str`

determine the key of this application class instance (read-only property).

**Returns**

application key string.

**property debug\_level:** `int`

debug level property:

**Getter**

return the current debug level of this app instance.

**Setter**

change the debug level of this app instance.

**property debug:** `bool`

True if app is in debug mode.

**property verbose:** `bool`

True if app is in verbose debug mode.

**call\_method**(*callback*, \*args, \*\*kwargs)

call passed callable/method with the passed args, catching and logging exceptions preventing app exit.

**Parameters**

- **callback** `(Union[Callable, str])` – either a callable or the name of the main app method of this instance to call.
- **args** – args passed to the main app method to be called.
- **kwargs** – kwargs passed to the main app method to be called.

**Return type**

`Any`

**Returns**

return value of the called method or None if method throws exception/does not exist.

**init\_logging**(*py\_logging\_params=None*, *log\_file\_name=""*, *log\_file\_size\_max=15*,  
*log\_with\_timestamp=False*, *disable\_buffering=False*)

initialize logging system.

**Parameters**

- **py\_logging\_params** `(Optional[Dict[str, Any]])` – config dict for python logging configuration. if this dict is not empty then python logging is configured with the given options in this dict and all the other kwargs are ignored.
- **log\_file\_name** `(str)` – default log file name for ae logging (def="" - ae logging disabled).
- **log\_file\_size\_max** `(float)` – max. size in MB of ae log file (def=LOG\_FILE\_MAX\_SIZE).
- **log\_with\_timestamp** `(Union[bool, str])` – add timestamp prefix to each log line if True or a non-empty strftime compatible format string.
- **disable\_buffering** `(bool)` – pass True to disable ae log buffering at app startup.

log files and config values will be initialized as late as possible in `log_file_check()`, e.g. indirectly triggered by a request to a config variable via `_parse_args()` (like `logFile`).

### `log_line_prefix()`

compile prefix of log print-out line for this `AppBase` instance.

the line prefix consists of (depending on the individual values of either a module variable or of an attribute this app instance):

- `_MULTI_THREADING_ACTIVATED`: if True then the thread id gets printed surrounded with angle brackets (< and >), right aligned and space padded to minimal 6 characters.
- `sys_env_id`: if not empty then printed surrounded with curly brackets ({ and }), left aligned and space padded to minimal 4 characters.
- `_log_with_timestamp`: if (a) True or (b) a non-empty string then the system time (determined with `now()`) gets printed in the format specified either by the (a) the `DATE_TIME_ISO` constant or (b) by the string in this attribute.

this method is using the instance attribute `_last_log_line_prefix` to keep a copy of the last printed log line prefix to prevent the printout of duplicate characters in consecutive log lines.

#### Return type

`str`

#### Returns

log file line prefix string including one space as separator character at the end.

### `log_file_check(curr_stream=None)`

check and possibly correct log file status and the passed currently used stream.

#### Parameters

`curr_stream` (Optional[TextIO]) – currently used stream.

#### Return type

Optional[TextIO]

#### Returns

stream passed into `curr_stream` or new/redirected stream of `curr_stream` or None if `curr_stream` is None.

for already opened log files check if the log file is big enough and if yes then do a file rotation. if log file is not opened but log file name got already set, then check if log startup buffer is active and if yes then create log file, pass log buffer content to log file and close the log buffer.

### `print_out(*objects, file=None, **kwargs)`

app-instance-specific print-outs.

#### Parameters

- `objects` – objects to be printed out.
- `file` (Optional[TextIO]) – output stream object to be printed to (def=None). passing None on a main app instance will print the objects to the standard output and any active log files, but on a `SubApp` instance with an active log file the print-out will get redirected exclusively/only to log file of this `SubApp` instance.
- `kwargs` – all the other supported kwargs of this method are documented [at the print\\_out\(\) function of this module](#).

this method has an alias named `po()`



**po**(\*objects, file=None, \*\*kwargs)

alias of method [print\\_out\(\)](#)

**debug\_out**(\*objects, minimum\_debug\_level=1, \*\*kwargs)

special debug version of [builtin print\(\)](#) function.

this method will print out the passed objects only if the [current debug level](#) of this app instance is higher than the value passed into the [minimum\\_debug\\_level](#) argument. in this case the print-out will be delegated onto the [print\\_out\(\)](#).

#### Parameters

- **objects** – objects to be printed out.
- **minimum\_debug\_level** ([int](#)) – minimum debug level to print the passed objects.
- **kwargs** – all the supported kwargs of this method are documented at the [print\\_out\(\)](#) function of the [core](#) module (including the [file](#) argument).

this method has an alias named [dpo\(\)](#).

**dpo**(\*objects, minimum\_debug\_level=1, \*\*kwargs)

alias of method [debug\\_out\(\)](#)

**verbose\_out**(\*objects, \*\*kwargs)

special verbose debug version of [builtin print\(\)](#) function.

#### Parameters

- **objects** – objects to be printed out.
- **kwargs** – the [file](#) argument is documented at the [print\\_out\(\)](#) method of the [AppBase](#) class. all other supported kwargs of this method are documented at the [print\\_out\(\)](#) function of the [core](#) module.

this method has an alias named [vpo\(\)](#).

**vpo**(\*objects, \*\*kwargs)

alias of method [verbose\\_out\(\)](#)

**shutdown**(exit\_code=0, timeout=None)

shutdown this app instance and if it is the main app instance then also any created sub-app-instances.

#### Parameters

- **exit\_code** ([Optional\[int\]](#)) – set application OS exit code - ignored if this is NOT the main app instance (def=0). pass None to prevent call of `sys.exit(exit_code)`.
- **timeout** ([Optional\[float\]](#)) – timeout float value in seconds used for the thread termination/joining, for the [SubApp](#) shutdowns and for the acquisition of the threading locks of [the ae log file](#) and the [app instances](#).

**\_std\_out\_err\_redirection**(redirect)

enable/disable the redirection of the standard output/error TextIO streams if needed.

#### Parameters

- **redirect** ([bool](#)) – pass True to enable or False to disable the redirection.

**\_append\_eof\_and\_flush\_file**(stream\_file, stream\_name)

add special end-of-file marker and flush the internal buffers to the file stream.

#### Parameters

- **stream\_file** ([TextIO](#)) – file stream.

- **stream\_name** (str) – name of the file stream (only used for debugging/error messages).

#### **\_flush\_and\_close\_log\_buf()**

flush and close ae log buffer and pass content to log stream if opened.

#### **\_open\_log\_file()**

open the ae log file with path and file name specified by *\_log\_file\_name*.

tries to create a log sub-folder - if specified in *\_log\_file\_name* and the folder does not exist (folder creation is limited to one folder level).

---

**Note:** a already existing file with the same file name will be overwritten (file contents get lost!).

---

#### **\_close\_log\_file()**

close the ae log file.

#### **\_rename\_log\_file()**

rename rotating log file while keeping first/startup log and log file count below MAX\_NUM\_LOG\_FILE.

**class SubApp**(app\_title="", app\_name="", app\_version="", sys\_env\_id="", debug\_level=0, multi\_threading=False, suppress\_stdout=False)

Bases: *AppBase*

separate/additional sub-app/thread/task with own/individual logging/debug configuration.

create an instance of this class for every extra thread and task where your application needs separate logging and/or debug configuration - additional to the main app instance.

all members of this class are documented at the *AppBase* class.

**startup\_beg:** *datetime*

begin of app startup datetime

**app\_path:** *str*

path to folder of your main app code file

## 4.10 ae.lockname

### 4.10.1 named threading locks

named locks are used in multi-threaded applications and based on the Python threading lock classes *threading.Lock* and *threading.RLock*. the advantage of the named locks in contrary to Python threading locks is that a lock don't need to create and store a reference of a Python threading lock object - the *NamedLocks* does this automatically for your application and does keep track of all the named locks of your application in its class variables.

so a named lock get exclusively identified only by an unique string. and to create other blocking locks you only need a reference to the *NamedLocks* class.

named locks are very useful e.g. if you want to lock a certain record of database table. for this you simply create a new instance of the *NamedLocks* class and as unique string you can use the table name followed by the primary key of the record to lock:

```

named_lock = NamedLocks()
if named_lock.acquire(table_name + primary_key)

    ...    # locked database transaction code goes here

named_lock.release(table_name + primary_key)

```

if now any other process of your application want to lock the same record (same table name and primary key) then it will be blocked until the process that acquired this named lock first is releasing the table record lock.

alternatively and especially if your application want to create multiple named locks you can use the class `NamedLocks` as a context manager, passing all the named lock strings to the constructor:

```

with NamedLocks(table_name1 + primary_key1, table_name2 + primary_key2, ...):
    ...    # locked database transaction

```

## Classes

<code>NamedLocks(*lock_names[, reentrant_locks, ...])</code>	manage all named locks of your application.
--	---

**class** `NamedLocks(*lock_names, reentrant_locks=True, sys_lock=False)`

Bases: `object`

manage all named locks of your application.

migrated from <https://stackoverflow.com/users/355230/martineau> answer in stackoverflow on the question <https://stackoverflow.com/questions/37624289/value-based-thread-lock>.

**Note:** currently the `sys_lock` feature is not implemented. use either `ae.lockfile` or the github extension `portalocker` (see <https://github.com/WoLpH/portalocker>) or the encapsulating extension `ilock` (<https://github.com/symonsoft/ilock>). more on system wide named locking: <https://stackoverflow.com/questions/6931342/system-wide-mutex-in-python-on-linux>.

**locks\_change\_lock:** `ClassVar[allocate_lock] = <unlocked _thread.lock object>`

threading lock class variable used to change status of all NamedLock instances

**active\_locks:** `ClassVar[Dict[str, Union[allocate_lock, RLock]]] = {}`

class variable keeping a dictionary of all active RLock/Lock instances

**active\_lock\_counters:** `ClassVar[Dict[str, int]] = {}`

lock counters class variable for reentrant locks

**\_\_init\_\_**(\*lock\_names, reentrant\_locks=True, sys\_lock=False)

prepare new named lock(s).

### Parameters

- **lock\_names** `(str)` – unique lock strings to be prepared to be locked by `__enter__()`.
- **reentrant\_locks** `(bool)` – pass False to use non-reentrant locks (True=reentrant locks).
- **sys\_lock** `(bool)` – pass True to prepare system lock (works for several independent applications). CURRENTLY NOT IMPLEMENTED.

**\_lock\_names**

tuple of lock names

**\_lock\_class:** `Type[Union[allocate_lock, RLock]]`

used threading lock class

**\_sys\_lock**

True if lock will be system-wide (not only application-wide)

**\_print\_func**

print function used to show debug and error messages

**\_\_enter\_\_()**

locking context enter method.

**\_\_exit\_\_(exc\_type, exc\_val, exc\_tb)**

locking context exit method.

**dpo(\*args, \*\*kwargs)**

print function which is suppressing printout if debug level is too low.

**acquire(lock\_name, \*args, \*\*kwargs)**acquire the named lock specified by the *lock\_name* argument.**Parameters**

- **lock\_name** `(str)` – name of the lock to acquire.
- **args** – passed to the acquire method of the underlying `RLock` respectively `Lock` class instance.
- **kwargs** – passed to the acquire method of the underlying `RLock` or `Lock` class instance.

**Return type**`bool`**Returns**

True if named lock got acquired successfully, else False.

**release(lock\_name)**release the named lock specified by the *lock\_name* argument.**Parameters****lock\_name** `(str)` – name of the lock to release.

## 4.11 ae.dynamicod

### 4.11.1 dynamic execution of code blocks and expressions

this ae namespace portion provides useful helper functions to evaluate Python expressions and execute Python code dynamically at application run-time.

dynamically executed code block or expression string offers convenience for powerful system and application configuration and for data-driven architectures.

for the dynamic execution of functions and code blocks the helper functions `try_call()`, `try_exec()` and `exec_with_return()` are provided. the helper function `try_eval()` evaluates dynamic expressions.

**Note: security considerations**

make sure that any dynamically executed code is from a secure source to prevent code injections of malware. treat configuration files from untrusted sources with extreme caution and only execute them after a complete check and/or within a sandbox.

**Hint:** these functions are e.g. used by the *Literal* class to dynamically determine literal values.

**Module Attributes**

<i>base_globals</i>	default if no global variables get passed in dynamic code/expression evaluations
---------------------	--

**Functions**

<i>exec_with_return</i> (code_block[, ...])	execute python code block and return the resulting value of its last code line.
<i>try_call</i> (callee, *args[, ignored_exceptions])	execute callable while ignoring specified exceptions and return callable return value.
<i>try_eval</i> (expr[, ignored_exceptions, ...])	evaluate expression string ignoring specified exceptions and return evaluated value.
<i>try_exec</i> (code_block[, ignored_exceptions, ...])	execute python code block string ignoring specified exceptions and return value of last code line in block.

**exec\_with\_return**(code\_block, ignored\_exceptions=(), glo\_vars=None, loc\_vars=None)

execute python code block and return the resulting value of its last code line.

**Parameters**

- **code\_block** *(str)* – python code block to execute.
- **ignored\_exceptions** *(Tuple[Type[Exception], ...])* – tuple of ignored exceptions.
- **glo\_vars** *(Optional[Dict[str, Any]])* – optional globals() available in the code execution.
- **loc\_vars** *(Optional[Dict[str, Any]])* – optional locals() available in the code execution.

**Return type**

*Optional[Any]*

**Returns**

value of the expression at the last code line or UNSET if either code block is empty, only contains comment lines, or one of the ignorable exceptions raised or if last code line is no expression.

inspired by this SO answer <https://stackoverflow.com/questions/33409207/how-to-return-value-from-exec-in-function/52361938#52361938>.

**try\_call**(*callee*, \**args*, *ignored\_exceptions*=(), \*\**kwargs*)

execute callable while ignoring specified exceptions and return callable return value.

**Parameters**

- **callee** (Callable) – pointer to callable (either function pointer, lambda expression, a class, ...).
- **args** – function arguments tuple.
- **ignored\_exceptions** (Tuple[Type[Exception], ...]) – tuple of ignored exceptions.
- **kwargs** – function keyword arguments dict.

**Return type**

Optional[Any]

**Returns**

function return value or UNSET if an ignored exception got thrown.

**try\_eval**(*expr*, *ignored\_exceptions*=(), *glo\_vars*=None, *loc\_vars*=None)

evaluate expression string ignoring specified exceptions and return evaluated value.

**Parameters**

- **expr** (str) – expression to evaluate.
- **ignored\_exceptions** (Tuple[Type[Exception], ...]) – tuple of ignored exceptions.
- **glo\_vars** (Optional[Dict[str, Any]]) – optional globals() available in the expression evaluation.
- **loc\_vars** (Optional[Dict[str, Any]]) – optional locals() available in the expression evaluation.

**Return type**

Optional[Any]

**Returns**

function return value or UNSET if an ignored exception got thrown.

**try\_exec**(*code\_block*, *ignored\_exceptions*=(), *glo\_vars*=None, *loc\_vars*=None)

execute python code block string ignoring specified exceptions and return value of last code line in block.

**Parameters**

- **code\_block** (str) – python code block to be executed.
- **ignored\_exceptions** (Tuple[Type[Exception], ...]) – tuple of ignored exceptions.
- **glo\_vars** (Optional[Dict[str, Any]]) – optional globals() available in the code execution.
- **loc\_vars** (Optional[Dict[str, Any]]) – optional locals() available in the code execution.

**Return type**

Optional[Any]

**Returns**

function return value or UNSET if an ignored exception got thrown.

**base\_globals**

default if no global variables get passed in dynamic code/expression evaluations

## 4.12 ae.i18n

### 4.12.1 internationalization / localization helpers

on importing this portion it will automatically determine the default locale (language) and encoding of your operating system and user configuration.

the functions `default_language()` and `default_encoding()` - provided by this portion - are determining or changing the default language and translation texts encoding.

additional languages will be automatically loaded by the function `load_language_texts()`.

#### translation texts locale paths

multiple paths can be provided by your app as well as any python package and namespace portion to store translation texts. by default they are situated in a sub-folder with the name `loc` underneath of your app/package root folder. to load them call the function `register_package_translations()` from the module using the locale texts.

---

**Hint:** see e.g. the ae namespace portion `ae.gui_help` loading package/module specific translation messages.

---



---

**Hint:** the `on_app_build()` app event is used by `gui_help` to load the app-specific translation texts on app startup.

---

to specify additional locale folders you can use the function `register_translations_path()`.

in a locale folder have to exist at least one sub-folder with a name of the language code for each supported language (e.g. 'loc/en' for english).

in each of these sub-folders at least one message translation file with a file name ending in the string specified by the constant `MSG_FILE_SUFFIX` has to exist.

#### translatable message texts and f-strings

simple message text strings can be enclosed in the code of your application with the `get_text()` function provided by this portion/module:

```
from ae.i18n import get_text

message = get_text("any translatable message displayed to the app user.")
print(message)           # prints the translated message text
```

for more complex messages with placeholders you can use the `get_f_string()` function:

```
from ae.i18n import get_f_string

my_var = 69
print(get_f_string("The value of my_var is {my_var}."))
```

translatable message can also be provided in various pluralization forms. to get a pluralized message you have to pass the `count` keyword argument of `get_text()`:

latest :

---

```
print(get_text("child", count=1))      # translated into "child" (in english) or e.g.
↳ "Kind" in german
print(get_text("child", count=3))      # -> "children" (in english) or e.g. "Kinder" (in
↳ german)
```

for pluralized message translated by the `get_f_string()` function, the count value have to be passed in the `count` item of the `loc_vars`:

```
print(get_f_string("you have {count} children", loc_vars=dict(count=1)))
# -> "you have 1 child" or e.g. "Sie haben 1 Kind"
print(get_f_string("you have {count} children", loc_vars={'count': 3}))
# -> "you have 3 children" or "Sie haben 3 Kinder"
```

you can load several languages into your app run-time. to get the translation for a language that is not the current default language you have to pass the `language` keyword argument with the desired language code onto the call of `get_text()` (or `get_f_string()`):

```
print(get_text("message", language='es'))  # returns the spanish translation text of
↳ "message"
print(get_text("message", language='de'))  # returns the german translation text of
↳ "message"
```

the helper function `translation()` can be used to determine if a translation exists for a message text.

---

**Hint:** the ae portion `ae.kivy.i18n` is implementing translation messages for kv files and provides additional helper functions and methods.

---

## Module Attributes

<code>MsgType</code>	type of message literals in translation text files
<code>LanguageMessages</code>	type of the data structure storing the loaded messages
<code>DEF_ENCODING</code>	encoding of the messages in your app code
<code>DEF_LANGUAGE</code>	language code of the messages in your app code
<code>LOADED_TRANSLATIONS</code>	message text translations of all loaded languages
<code>MSG_FILE_SUFFIX</code>	name suffix of translation text files
<code>TRANSLATIONS_PATHS</code>	file paths to search for translations
<code>default_locale</code>	language and encoding code of the current language/locale



## Functions

<code>default_encoding([new_enc])</code>	get and optionally set the default message text encoding.
<code>default_language([new_lang])</code>	get and optionally set the default language code.
<code>get_f_string(f_str[, key_suffix, language, ...])</code>	translate passed f-string into a message string of the passed / default language.
<code>get_text(text[, count, key_suffix, language])</code>	translate passed text string into the current language.
<code>load_language_file(file_name, encoding, language)</code>	load file content encoded with the given encoding into the specified language.
<code>load_language_texts([language, encoding, ...])</code>	load translation texts for the given language and optional domain.
<code>plural_key(count)</code>	convert number in <code>count</code> into a dict key to access the correct plural form.
<code>register_package_translations()</code>	call from module scope of the package to register/add translations resources path.
<code>register_translations_path([translation_path])</code>	add/register the passed root path as new resource of translation texts.
<code>translation(text[, language])</code>	determine translation for passed text string and language.

## MsgType

type of message literals in translation text files

alias of `Union[str, Dict[str, str]]`

## LanguageMessages

type of the data structure storing the loaded messages

alias of `Dict[str, Union[str, Dict[str, str]]]`

`DEF_ENCODING = 'UTF-8'`

encoding of the messages in your app code

`DEF_LANGUAGE = 'en'`

language code of the messages in your app code

`LOADED_TRANSLATIONS: Dict[str, Dict[str, str] | Dict[str, str]] = {}`

message text translations of all loaded languages

`MSG_FILE_SUFFIX = 'Msg.txt'`

name suffix of translation text files

`TRANSLATIONS_PATHS: List[str] = ['/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_help/loc',  
'/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/kivy_qr_display/loc',  
'/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/kivy_sideload/loc']`

file paths to search for translations

`default_locale: List[str] = ['en', 'UTF-8']`

language and encoding code of the current language/locale

`default_encoding(new_enc='')`

get and optionally set the default message text encoding.

**Parameters**

**new\_enc** (str) – new default encoding to be set. kept unchanged if not passed.

**Return type**

str

**Returns**

old default encoding (current if **new\_enc** get not passed).

**default\_language**(new\_lang="")

get and optionally set the default language code.

**Parameters**

**new\_lang** (str) – new default language code to be set. kept unchanged if not passed.

**Return type**

str

**Returns**

old default language (or current one if **new\_lang** get not passed).

**get\_text**(text, count=None, key\_suffix="", language="")

translate passed text string into the current language.

**Parameters**

- **text** (str) – text message to be translated.
- **count** (Optional[int]) – pass int value if the translated text has variants for their pluralization. the count value will be converted into an amount/pluralize key by the function `plural_key()`.
- **key\_suffix** (str) – suffix to the key used if the translation is a dict.
- **language** (str) – language code to load (def=current language code in 1st item of `default_locale`).

**Return type**

str

**Returns**

translated text message or the value passed into **text** if no translation text got found for the current language.

**get\_f\_string**(f\_str, key\_suffix="", language="", glo\_vars=None, loc\_vars=None)

translate passed f-string into a message string of the passed / default language.

**Parameters**

- **f\_str** (str) – f-string to be translated and evaluated.
- **key\_suffix** (str) – suffix to the key used if the translation is a dict.
- **language** (str) – language code to load (def=current language code in 1st item of `default_locale`).
- **glo\_vars** (Optional[Dict[str, Any]]) – global variables used in the conversion of the f-string expression to a string. the globals() of the caller of the callee will be available too and get overwritten by the items of this argument.
- **loc\_vars** (Optional[Dict[str, Any]]) – local variables used in the conversion of the f-string expression to a string. the locals() of the caller of the callee will be available too and get overwritten by the items of this argument. pass a numeric value in the **count** item of this dict for pluralized translated texts (see also **count** parameter of the function `get_text()`).

**Return type**`str`**Returns**

translated text message including evaluated and formatted variables/expressions of the f-string passed-in `f_str`. if no translation text got found for the current language then the original text message will be returned. any syntax errors and exceptions occurring in the conversion of the f-string are silently ignored.

**load\_language\_file**(*file\_name*, *encoding*, *language*)

load file content encoded with the given encoding into the specified language.

**Parameters**

- **file\_name** `(str)` – file name, inclusive path and extension to load.
- **encoding** `(str)` – encoding id string.
- **language** `(str)` – language id string.

**load\_language\_texts**(*language*="", *encoding*="", *domain*="", *reset*=False)

load translation texts for the given language and optional domain.

**Parameters**

- **language** `(str)` – language code of the translation texts to load. use the default language if not passed.
- **encoding** `(str)` – encoding to use to load message file.
- **domain** `(str)` – optional domain id, e.g. the id of an app, attached process or a user. if passed then it will be used as prefix for the message file name to be loaded additionally and after the default translation texts get loaded (overwriting the default translations).
- **reset** `(bool)` – pass True to clear all previously added language/locale messages.

**Return type**`str`**Returns**

language code of the loaded/default language.

**plural\_key**(*count*)

convert number in `count` into a dict key to access the correct plural form.

**Parameters**

**count** `(Optional[int])` – number of items used in the current context or None (resulting in empty string).

**Return type**`str`**Returns**

dict key (prefix) within the MsgType part of the translation data structure.

**register\_package\_translations**()

call from module scope of the package to register/add translations resources path.

no parameters needed because we use here `stack_var()` helper function to determine the the module file path via the `__file__` module variable of the caller module in the call stack. in this call we have to overwrite the default value (`SKIPPED_MODULES`) of the `skip_modules` parameter to not skip ae portions that are providing package resources and are listed in the `SKIPPED_MODULES`, like e.g. `ae.gui_app` and `ae.gui_help` (passing empty string "" to overwrite default skip list).

latest :

---

**register\_translations\_path**(*translation\_path=""*)

add/register the passed root path as new resource of translation texts.

**Parameters**

**translation\_path** (str) – root path of a translations folder structure to register, using cwd if not specified.

**Return type**

bool

**Returns**

True if the translation folder structure exists and got properly added/registered, else False.

**translation**(*text, language=""*)

determine translation for passed text string and language.

**Parameters**

- **text** (str) – text message to be translated.
- **language** (str) – language code to load (def=current language code in 1st item of *default\_locale*).

**Return type**

Union[str, Dict[str, str], None]

**Returns**

None if not found else translation message or dict with plural forms.

## 4.13 ae.parse\_date

### 4.13.1 parse date strings more flexible and less strict

this module is pure python and has no namespace external dependencies.

the *parse\_date()* helper function is converting a wide range of date and datetime string literal formats into the built-in types *datetime.datetime* and *datetime.date*.

this function extends (and fully replaces) Python's standard method *strptime()* and supports multiple date formats which are much more flexible interpreted.

#### Functions

---

*parse\_date*(literal, \*additional\_formats[, ...])

parse a date literal string, returning the represented date/datetime or None if date literal is invalid.

---

**parse\_date**(*literal, \*additional\_formats, replace=None, ret\_date=False, dt\_seps=('T', ' '), ti\_sep=':', ms\_sep='.', tz\_sep='+'*)

parse a date literal string, returning the represented date/datetime or None if date literal is invalid.

this function checks/corrects the passed date/time literals to support a wider range of ISO and additional date/time formats as Python's *strptime()*.

---

**Hint:** Python's `strptime()` to parse date and time strings into `datetime.date` or `datetime.datetime` objects is very strict and does not respect the formatting alternatives of ISO8601 (see <https://bugs.python.org/issue15873> and <https://github.com/boxed/iso8601>).

---

additionally a `datetime.date` object can be created/returned automatically if no time info is specified in the date string/literal (see `ret_date` parameter).

#### Parameters

- **literal** (str) – date literal string in the format of `DATE_ISO`, `DATE_TIME_ISO` or in one of the additional formats passed into the `additional_formats` arguments.
- **additional\_formats** (str) – additional date literal format string masks (supported mask characters are documented at the `format` argument of the python method `strptime()`).
- **replace** (Optional[Dict[str, Any]]) – dict of replace keyword arguments for `datetime.datetime.replace()` call. pass e.g. `dict(microsecond=0, tzinfo=None)` to set the microseconds of the resulting date to zero and to remove the timezone info.
- **ret\_date** (Optional[bool]) – request return value type: `True=datetime.date`, `False=datetime.datetime` (the default) or `None=determine type from literal` (short date if `dt_seps` are not in literal).
- **dt\_seps** (Tuple[str, ...]) – tuple of supported separator characters between the date and time literal parts.
- **ti\_sep** (str) – separator character of the time parts (hours/minutes/seconds) in literal.
- **ms\_sep** (str) – microseconds separator character.
- **tz\_sep** (str) – time-zone separator character.

#### Return type

`Union[date, datetime, None]`

#### Returns

represented date/datetime or `None` if date literal is invalid.

## 4.14 ae.literal

### 4.14.1 literal type detection and evaluation

a number, calendar date or other none-text-value gets represented by a literal string, if entered as user input or has to be stored, e.g. as *configuration variable*.

the *Literal* class implemented by this portion converts such a *evaluable literal string* into the corresponding value (and type).

latest :

---

## Functions

<code>evaluable_literal(literal)</code>	check evaluable format of literal string, possibly return appropriate evaluation function and stripped literal.
---	---

## Classes

<code>Literal([literal_or_value, value_type, name])</code>	convert literal string into the corresponding value (and type).
--	---

### `evaluable_literal(literal)`

check evaluable format of literal string, possibly return appropriate evaluation function and stripped literal.

#### Parameters

**literal** *(str)* – string to be checked if it is in the *evaluable literal format* and if it has to be stripped.

#### Return type

`Tuple[Optional[Callable], Optional[str]]`

#### Returns

tuple of evaluation/execution function and the (optionally stripped) literal string (removed triple high-commas on expression/code-blocks) - if *literal* is in one of the supported *evaluable literal formats* - else the tuple (None, <empty string>).

**class** `Literal(literal_or_value=None, value_type=None, name='LiT')`

Bases: `object`

convert literal string into the corresponding value (and type).

pass the literal string on instantiation as the first (the *literal\_or\_value*) argument:

```
>>> number = Literal("3")
>>> number
Literal('3')
>>> number.value
3
>>> type(number.value)
<class 'int'>
```

*Literal* will interpret the value type from the specified literal string. the corresponding *int* value provides the *value* attribute.

to make sure that a number-like literal will be interpreted as a string enclose it in high-commas. the following example will therefore result as a string type:

```
>>> number = Literal("'3'")
>>> number
Literal("'3'")
>>> number.value
'3'
>>> type(number.value)
<class 'str'>
```

another way to ensure the correct value type, is to specify it with the optional *second argument*:

```
>>> number = Literal("3", str)
>>> number
Literal('3')
>>> number.value
'3'
```

alternatively assign the *evaluable literal string* after the instantiation, directly to the *value* attribute of the *Literal* instance:

```
>>> number = Literal(value_type=str)
>>> number.value = "3"
>>> number.value
'3'
```

any type can be specified as the literal value type:

```
>>> my_list = Literal(value_type=list)
>>> my_dict = Literal(value_type=dict)
>>> my_datetime = Literal(value_type=datetime.datetime)
>>> class MyClass:
...     pass
>>> my_instance = Literal(value_type=MyClass)
```

the value type get automatically determined also for *evaluable python expression literal* . for example the following literal gets converted into a *datetime* object:

```
>>> datetime_value = Literal('(datetime.datetime.now())')
```

also if assigned directly to the *value* attribute:

```
>>> date_value = Literal()
>>> date_value.value = '(datetime.date.today())'
```

---

**Note:** the literal string of the last two examples has to start and end with round brackets, to mark it as a *evaluable literal*.

---

to convert calendar date literal strings into one of the supported ISO formats (*DATE\_TIME\_ISO* or *DATE\_ISO*), the expected value type has to be specified:

```
>>> date_value = Literal('2033-12-31', value_type=datetime.date)
>>> assert date_value.value == datetime.date(2033, 12, 31)
```

a *ValueError* exception will be raised if the conversion fails, or if the result cannot be converted into the requested value type:

```
>>> date_literal = Literal(value_type=datetime.date)
>>> date_literal.value = "invalid-date-literal"
>>> date_value = date_literal.value
Traceback (most recent call last):
...
ValueError
```

all supported literal formats are documented at the [value](#) property/attribute.

**\_\_init\_\_**(*literal\_or\_value=None, value\_type=None, name='LiT'*)

create new Literal instance.

#### Parameters

- **literal\_or\_value** (Optional[Any]) – initial literal (evaluable string expression) or value of this instance.
- **value\_type** (Optional[Type]) – type of the value of this instance (def=determined latest by/in the [value](#) property getter).
- **name** (str) – name of the literal (only used for debugging/error-message).

**\_\_repr\_\_**()

Return repr(self).

**property value:** [Any](#)

property representing the value of this Literal instance.

#### Setter

assign literal or a new value; can be either a value literal string or directly the represented/resulting value. if the assigned value is not a string and the value type of this instance got still unspecified then this instance will be restricted to the type of the assigned value. assigning a None value will be ignored - neither the literal nor the value will change with that!

#### Getter

return the literal value; on the first call the literal will be evaluated (lazy/late) and the value type will be set if still unspecified. further getter calls will directly return the already converted literal value.

if the literal of this [Literal](#) instance coincide with one of the following evaluable formats then the value and the type of the value gets automatically recognized. an evaluable formatted literal strings has to start and end with one of the character pairs shown in the following table:

starts with	ends with	evaluation value type
(	)	tuple literal or expression
[	]	list literal
{	}	dict literal
'	'	string literal
“	”	string literal
“”	“”	code block with return
“””	“””	code block with return

#### other supported literals and values

literals with type restriction to a boolean type are evaluated as python expression. this way literal strings like 'True', 'False', '0' and '1' will be correctly recognized and converted into a boolean value.

literal strings that representing a date value (with type restriction to either [datetime.datetime](#) or [datetime.date](#)) will be converted with the [parse\\_date\(\)](#) function and should be formatted in one of the standard date formats (defined via the [ae.base](#) constants [DATE\\_TIME\\_ISO](#) and [DATE\\_ISO](#)).

literals and values that are not in one of the above formats will finally be passed to the constructor of the restricted type class to try to convert them into their representing value.



**append\_value**(*item\_value*)

add new item to the list value of this Literal instance (lazy *self.value* getter call function pointer).

**Parameters**

**item\_value** *(Any)* – value of the item to be appended to the value of this Literal instance.

**Return type**

*Any*

**Returns**

the value (==list) of this Literal instance.

this method gets e.g. used by the *ConsoleApp* method *add\_option()* to have a function pointer to this literal value with lazy/late execution of the value getter (value.append cannot be used in this case because the list could have be changed before it get finally read/used).

---

**Note:** this method calls the append method of the value object and will therefore only work if the value is of type *list* (or a compatible type).

---

**convert\_value**(*lit\_or\_val*)

set/change the literal/value of this *Literal* instance and return the represented value.

**Parameters**

**lit\_or\_val** *(Any)* – the new value to be set.

**Return type**

*Any*

**Returns**

the final/converted value of this Literal instance.

this method gets e.g. used by the *ConsoleApp* method *add\_option()* to have a function pointer to let the *ArgumentParser* convert a configuration option literal into the represented value.

**type\_mismatching\_with**(*value*)

check if this literal instance would reject the passed value because of type mismatch.

**Parameters**

**value** *(Any)* – new literal value.

**Return type**

*bool*

**Returns**

True if the passed value would have a type mismatch or if literal type is still not set, else False.

**\_determine\_value**(*lit\_or\_val*)

check passed value if it is still a literal determine the represented value.

**Parameters**

**lit\_or\_val** *(Any)* – new literal value or the representing literal string.

**Return type**

*Any*

**Returns**

determined/converted value or *self.\_lit\_or\_val* if value could not be recognized/converted.

`_chk_val_reset_else_set_type(value)`

reset and return passed value if is None, else determine value type and set type (if not already set).

**Parameters**

**value** *(Any)* – just converted new literal value to be checked and if ok used to set an unset type.

**Return type**

*Any*

**Returns**

passed value or the stored literal/value if passed value is None.

## 4.15 ae.progress

### 4.15.1 display progress of long running processes

this module is simplifying the display of progress messages on the command console/shell of your OS for long running processes.

#### basic usage of progress portion

create an instance of the *Progress* for each process/thread your application spawns:

```
from ae.core import SubApp
from ae.progress import Progress

app = SubApp(...)
...
progress = Progress(app, ...)
```

now you can call the *next()* method within your long running process on each processed item/percentage:

```
while process_is_not_finished:
    ...
    progress.next()
```

optionally you can request to print an end-message by calling the *finished()* method of your *Progress* instance as soon as the process is finished:

```
...
progress.finished()
```

the above code snippets are printing a start message to your console at the instantiation of *Progress*. then every call of the method *next()* will print the next message and finally the method *finished()* will print an end message.

to use the integrated error counter and automatic printout of an error message to the console, pass the message text of any occurring error to the argument *error\_msg* of *next()* or *finished()*.

## progress instantiation

the *first argument* expects the instance of the application class (either *AppBase*, *SubApp* or *ConsoleApp*) that spawns the process.

the next three arguments are configuring a run or item counter. and the other arguments may be used to adopt the format of the displayed messages to your needs.

## process run counter

depending on the type of process you want to show progress differently, e.g. on each processed item or after passing a certain percentage value and either as incrementing or decrementing number. for that *Progress* provides a counter which can be flexible configured with the arguments *start\_counter* and *total\_count*.

specifying only *start\_counter* results in a countdown. e.g. to display the number of items waiting to be processed::

```
progress = Progress(app, number_of_items_to_be_processed)
```

by only specifying *total\_count* you get an incremental process run counter:

```
progress = Progress(app, total_count=number_of_items_to_be_processed)
```

to display a percentage on the console in 5 percent steps specify *total\_count* as 100 percent and *delta* as +5:

```
progress = Progress(app, total_count=100, delta=5)
```

## individual message templates

progress displays 5 types of messages on the console. to overwrite one of the generic default message templates, simply pass your template on instantiation of *Progress* into the argument that is displayed underneath directly after the message type:

- **start - *start\_msg***  
start message printed on process start and class instantiation
- **next - *next\_msg***  
next run message (printed on each call of *next()*)
- **end - *end\_msg***  
finished message (printed on call of *finished()*)
- **error - *err\_msg***  
error message on any occurring error (printed on each call of either *next()* or *finished()*)
- **nothing-to-do - *nothing\_to\_do\_msg***  
nothing-to-do message printed on process start (class instantiation) if total count is zero

the following table shows which progress state placeholders you can use in which message type:

placeholder	available in message type
run_counter	start, next, end, error
total_count	start, next, end, error
processed_id	next, end, error
err_counter	next, end, error
err_msg	next, end, error

latest :

---

**Hint:** only the *nothing-to-do* message type does not provide any placeholders.

---

## Classes

<code>Progress(app_base[, start_counter, ...])</code>	display progress on the console/log output for a long running tasks.
---	--

```
class Progress(app_base, start_counter=0, total_count=0, delta=-1, start_msg="", next_msg="",
               end_msg='Finished processing of {total_count} having {err_counter} failures: ¡{err_msg}!',
               err_msg='{err_counter} errors on processing {total_count} items',
               current={run_counter}: ¡{err_msg}!', nothing_to_do_msg="")
```

Bases: `object`

display progress on the console/log output for a long running tasks.

```
__init__(app_base, start_counter=0, total_count=0, delta=-1, start_msg="", next_msg="",
         end_msg='Finished processing of {total_count} having {err_counter} failures: ¡{err_msg}!',
         err_msg='{err_counter} errors on processing {total_count} items',
         current={run_counter}: ¡{err_msg}!', nothing_to_do_msg="")
```

prepare print-outs for a new progress (long running process with incrementing or decrementing item counter).

### Parameters

- **app\_base** `(AppBase)` – instance of the application class that is spawning the long-running process.
- **start\_counter** `(int)` – process item counter start value. counter decrements on each call of `next()` (if `total_count` not specified).
- **total\_count** `(int)` – number of items that will be processed with an incrementing counter. by passing a positive integer the process item counter will be incremented on each call of `next()`.
- **delta** `(int)` – difference to decrement/increment on each call of `next()`.
- **start\_msg** `(str)` – start message template with placeholders.
- **next\_msg** `(Optional[str])` – next message - if an empty string get passed then a default message will be provided with placeholders - pass `None` if you want to suppress the print-out of a next message.
- **end\_msg** `(str)` – end message template with placeholders, pass `None` if you want to suppress the print-out of an end message (in this case only a new line will be printed).
- **err\_msg** `(str)` – error message template with placeholders.
- **nothing\_to\_do\_msg** `(str)` – message template printed-out if the values of the two arguments `start_counter` and `total_count` are not specified or are both less or equal to zero.

**\_app:** `AppBase`

used application class instance

**\_next\_msg**

next message template

**\_end\_msg**

end message template

**\_err\_msg**

error message template

**\_err\_counter**

error counter

**\_run\_counter**

item, percentage or run counter

**\_total\_count**

total count of item/percentage/run counter

**\_delta**

delta value to increment/decrement run counter

**next**(*processed\_id*="", *error\_msg*="", *next\_msg*="", *delta*=0)

log the processing of the next item of this long-running task.

**Parameters**

- **processed\_id** (str) – id(s) of the next item (to be displayed on console/logging output).
- **error\_msg** (str) – pass the error message to display if the next item produced any errors. if an error message get passed then the **\_err\_counter** will be incremented.
- **next\_msg** (str) – message to output (use instance message if not passed/empty).
- **delta** (int) – delta for decrement/increment process run counter (use instance default if not passed).

**finished**(*processed\_id*="", *error\_msg*="")

display end of processing for the current item.

**Parameters**

- **processed\_id** (str) – id(s) of the next item (to be displayed on console/logging output).
- **error\_msg** (str) – optional error message to display if current items produced any error. if an error message get passed then the **\_err\_counter** will be incremented.

**get\_end\_message**(*processed\_id*="", *error\_msg*="")

determine message text for finishing the currently processed item.

**Parameters**

- **processed\_id** (str) – id(s) of the next item (to be displayed on console/logging output).
- **error\_msg** (str) – optional error message to display if current items produced any error.

**Return type**

str

**Returns**

message text to display.

## 4.16 ae.updater

### 4.16.1 application environment updater

this module is providing check functions running on app startup for easy deployment of Python application updates.

#### updater check functions

update any files on the destination machine with the help of the check functions `check_moves()` and `check_overwrites()`.

the check function `check_local_updates()` checks if your deployment package contains a Python update script that will be executed (and only one time after an app update) on the next startup of your application.

for a temporary work-around or bug-fix you can deploy your application update with a bootstrap Python script which will be executed on every startup of your application. the detection and execution of such bootstrap script is done by the function `check_local_bootstrap()`-

the function `check_all()` combines all the above checks.

the following skeleton of a main app module demonstrates a typical usage of `check_all()`:

```
from python_and_3rd_party_libs import ...
from ae.updater import check_all
from project_local_libs import ...

check_all()

app = WhateverApp(app_name=...)
...
app.run_app()
```

replace the `check_all()` call with the needed check function(s) if your app only needs certain checks.

---

**Note:** make sure that the check function(s) get called before you initialize any app instances if you want to update any *config variables*, like *application status* or user preferences.

---

..hint: more info you find in the doc-strings of each of the check functions.

## Functions

<code>check_all([copy_src_folder, ...])</code>	check all outstanding scripts to be executed and files to be moved/overwritten.
<code>check_copies([src_folder, dst_folder])</code>	check on new or missing files to be copied from <code>src_folder</code> to the <code>dst_folder</code> .
<code>check_local_bootstraps()</code>	check if <code>ae_bootstrap</code> script exists in the current working directory to be executed on app startup.
<code>check_local_updates()</code>	check if <code>ae_updater</code> script exists in the current working directory to be executed and deleted.
<code>check_moves([src_folder, dst_folder])</code>	check on missing files to be moved from <code>src_folder</code> to the <code>dst_folder</code> .
<code>check_overwrites([src_folder, dst_folder])</code>	check on files to be moved from the source directory and overwritten within the destination directory.

**check\_copies**(`src_folder='ae_updater_copies', dst_folder=''`)

check on new or missing files to be copied from `src_folder` to the `dst_folder`.

### Parameters

- **src\_folder** *(str)* – path to source folder/directory where the files get copied from. If not specified then `COPIES_SRC_FOLDER_NAME` will be used.
- **dst\_folder** *(str)* – path to destination folder/directory where the files get copied to. If not specified or if you pass an empty string then the user data/preferences path (`{usr}`) will be used.

### Return type

`List[str]`

### Returns

list of moved files, with their destination path.

**check\_moves**(`src_folder='ae_updater_moves', dst_folder=''`)

check on missing files to be moved from `src_folder` to the `dst_folder`.

### Parameters

- **src\_folder** *(str)* – path to source folder/directory where the files get moved from. If not specified then `MOVES_SRC_FOLDER_NAME` will be used. Please note that the source folder itself will neither be moved nor removed (but will be empty after the operation finished).
- **dst\_folder** *(str)* – path to destination folder/directory where the files get moved to. If not specified or if you pass an empty string then the user data/preferences path (`{usr}`) will be used.

### Return type

`List[str]`

### Returns

list of moved files, with their destination path.

**check\_overwrites**(`src_folder='ae_updater_overwrites', dst_folder=''`)

check on files to be moved from the source directory and overwritten within the destination directory.

### Parameters

- **src\_folder** (str) – path to source folder/directory where the files get moved from. If not specified then MOVES\_SRC\_FOLDER\_NAME will be used. Please note that the source folder itself will neither be moved nor removed (but will be empty after the operation finished).
- **dst\_folder** (str) – path to destination folder/directory where the files get moved to. If not specified or if you pass an empty string then the user data/preferences path ({usr}) will be used.

**Return type**

List[str]

**Returns**

list of moved and possibly overwritten files, with their destination path.

**check\_local\_updates()**

check if ae\_updater script exists in the current working directory to be executed and deleted.

**Return type**

bool

**Returns**

return value (True) of executed run\_updater method (if module&function exists), else False.

**check\_local\_bootstraps()**

check if ae\_bootstrap script exists in the current working directory to be executed on app startup.

**Return type**

bool

**Returns**

return value (True) of executed run\_updater function (if module&function exists) else False.

**check\_all(copy\_src\_folder="", move\_src\_folder="", over\_src\_folder="", dst\_folder="")**

check all outstanding scripts to be executed and files to be moved/overwritten.

**Parameters**

- **copy\_src\_folder** (str) – path to source folder/directory where the files get copied from. if not specified or if you pass an empty string then COPIES\_SRC\_FOLDER\_NAME will be used.
- **move\_src\_folder** (str) – path to source folder/directory where the files get moved from. if not specified or if you pass an empty string then MOVES\_SRC\_FOLDER\_NAME will be used.
- **over\_src\_folder** (str) – path to source folder/directory where the files get moved from and overwritten to. if not specified then OVERWRITES\_SRC\_FOLDER\_NAME will be used.
- **dst\_folder** (str) – path to destination folder/directory where the files get moved to. if not specified or if you pass an empty string then the user data/preferences path ({usr}) will be used.

**Return type**

List[str]

**Returns**

list of processed (copied, moved or overwritten) files, with their destination path.



## 4.17 ae.console

### 4.17.1 console application environment

an instance of the *ConsoleApp* class is representing a python application with dynamically configurable logging, debugging features (inherited from *AppBase*), command line arguments and config files and options.

the helper function *sh\_exec()* provided by this portion simplifies the execution of shell/console commands.

#### define command line arguments and options

the methods *add\_argument()* and *add\_option()* are defining command line arguments and *config options*, finally parsed/loaded by calling *run\_app()*:

```
ca = ConsoleApp(app_title="command line arguments demo", app_version="3.6.9")
ca.add_argument('argument_name_or_id', help="Help text for this command line argument")
ca.add_option('option_name_or_id', "help text for this command line option", "default_
↪value")
...
ca.run_app()
```

the values of the command line arguments and options are determined via the methods *get\_argument()* and *get\_option()*. additional configuration values, stored in *INI/CFG files*, are accessible via the *get\_variable()* method.

#### auto-collected app name, title and version

if one of the kwargs *app\_title* or *app\_version* is not specified in the init call of the *ConsoleApp* class instance, then they will automatically get determined from your app main module: the app title from the docstring title, and the application version string from the *\_\_version\_\_* variable:

```
""" module docstring title """
from ae.console import ConsoleApp

__version__ = '1.2.3'

ca = ConsoleApp()

assert ca.app_title == "module docstring title"
assert ca.app_version == '1.2.3'
```

*ConsoleApp* also determines on instantiation the name/id of your application, if not explicitly specified in *app\_name*. other application environment vars/options (like e.g. the application startup folder path and the current working directory path) will be automatically initialized and provided via the app instance.

## configuration files, sections, variables and options

a config file consists of config sections, each section provides config variables and config options to parametrize your application at run-time.

### config files

configuration files can be shared between apps or used exclusively by one app. the following file names are recognized and loaded automatically on app initialization:

config file	used for .... config variables and options
<any_path_name_and_ext>	application/domain specific
<app_name>.ini	application specific (read-/write-able)
<app_name>.cfg	application specific (read-only)
.app_env.cfg	application/suite specific (read-only)
.sys_env.cfg	general system (read-only)
.sys_env<SYS_ENV_ID>.cfg	the system with SYS_ID (read-only)

the above table is ordered by the preference to search/get the value of a config variable/option. so the values stored in the domain/app specific config file will always precede/overwrite any application and system specific values.

app/domain-specific config files have to be specified explicitly, either on initialization of the [ConsoleApp](#) instance via the kwarg `additional_cfg_file`, or by calling the method `add_cfg_files()`. they can have any file extension and can be placed into any accessible folder.

all the other config files have to have the specified name with a `.ini` or `.cfg` file extension, and get recognized in the current working directory, in the user data directory (see `ae.paths.user_data_path()`) and in the application installation directory.

### config sections

this module is supporting the `config file format` of Python's built-in `ConfigParser` class, extended by more complex config value types. the following examples shows a config file with two config sections containing one config option (named `log_file`) and two config variables (`configVar1` and `configVar2`):

```
[aeOptions]
log_file = './logs/your_log_file.log'
configVar1 = ['list-element1', ('list-element2-1', 'list-element2-2', ), {}]

[YourSectionName]
configVar2 = {'key1': 'value 1', 'key2': 2222, 'key3': datetime.datetime.now()}
```

the config section `aeOptions` (defined by `MAIN_SECTION_NAME`) is the default or main section, storing the values of any pre-defined `config option` and of some `config variables`.

## config variables

config variables can store complex data types. in the example config file above the config variable *configVar1* holds a list with 3 elements: the first element is a string, the second element a tuple, and the third element an empty dict.

all the values, of which its *repr* string can be evaluated with the built-in `eval()` function, can be stored in a config variable, by calling the `set_variable()` method. to read/fetch their value, call the method `get_variable()` with the name and section names of the config variable. you can specify the type of an config variable via the value passed into *value* argument or by the see *special encapsulated strings*, respectively the config value literal.

the following config variables are pre-defined in the *main config section* and recognized by *this module*, some of them also by the module/portion *ae.core*:

- *debug\_level*: debug logging verbosity level *config option*
- *log\_file*: ae logging file name (this is also a *config option* - set-able as command line arg)
- *logging\_params*: *general ae logging configuration parameters (py and ae logging)*
- *py\_logging\_params*: *python logging configuration*
- *onboarding\_tour\_started*: count the onboarding tour starts since the installation of the app. will be reset to zero after a user registration (by calling `register_user()`)
- *registered\_users*: users registered with their OS user name as user id (see `register_user()`)
- *user\_id*: id of the app user (default is determined from the *system user name* <*ae.base.os\_user\_name*>)
- *user\_specific\_cfg\_vars*: list of config variables storing an individual value for each registered user (see section *user specific config variables*)

**Note:** the value of a config variable can be overwritten by defining an OS environment variable with a name that is equal to the *snake+upper-case converted names* of the config-section and -variable. e.g. declare an OS environment variable with the name *AE\_OPTIONS\_LOG\_FILE* to overwrite the value of the *pre-defined config option/variable* *log\_file*.

## config options

config options are config variables, defined persistently in the config section *aeOptions*. specifying them on the command line, preceding the option name with two leading hyphen characters, and using an equal character between the name and the option value, overwrites the value stored in the config file:

```
$ your_application --log_file='your_new_log_file.log'
```

the default value of a not specified config option gets searched first in the config files (the exact search order is documented in the doc-string of the method `add_cfg_files()`), or if not found then the default value will be used, that is specified in the definition of the config option (the call of `add_option()`).

the method `get_option()` determines the value of a config option:

```
my_log_file_name = ca.get_option('log_file')
```

use the `set_option()` if you want to change the value of a configuration option at run-time. to read the default value of a config option or variable directly from the available configuration files use the method `get_variable()`. the default value of a config option or variable can also be set or changed directly from within your application by calling the `set_variable()` method.

latest :

---

## pre-defined configuration options

for a more verbose logging to the console output specify, either on the command line or in a config files, the config option *debug\_level* (or as short option *-D*) with a value of 2 (for verbose). the supported config option values are documented [here](#).

the value of the second pre-defined config option *log\_file* specifies the log file path/file\_name. also this option can be abbreviated on the command line with the short *-L* option id.

---

**Note:** after an explicit definition of the optional config option *user\_id* via *add\_option()* it will be automatically used to initialize the *user\_id* attribute.

---

## user specific config variables

config variables specified in the set *user\_specific\_cfg\_vars* get automatically recognized as user-specific. override the method `:meth`~ConsoleApp._init_default_user_cfg_vars`` in your main app instance to define or revoke which config variables the app is storing individually for each user.

---

**Hint:** to permit individual sets of user-specific config variables for a user (or group) add the config variable *user\_specific\_cfg\_vars* in the user-specific config file section(s). don't forget in this special case to also add there also this config variable, e.g. as ('*aeOptions*', '*user\_specific\_cfg\_vars*').

---

## Module Attributes

<i>MAIN_SECTION_NAME</i>	default name of main config section
--------------------------	-------------------------------------

## Functions

<i>config_value_string</i> (value)	convert passed value to a string to store them in a config/ini file.
<i>sh_exec</i> (command_line[, extra_args, ...])	execute command in the current working directory of the OS console/shell.

## Classes

<i>ConsoleApp</i> ([app_title, app_name, ...])	provides command line arguments and options, config options, logging and debugging for your application.
--	--

**MAIN\_SECTION\_NAME:** `str = 'aeOptions'`  
default name of main config section

**config\_value\_string**(*value*)

convert passed value to a string to store them in a config/ini file.

**Parameters**

**value** *(Any)* – value to convert to ini variable string/literal.

**Return type**

*str*

**Returns**

ini variable literal string.

---

**Note:** *Literal* converts the returned string format back into the representing value.

---

**sh\_exec**(*command\_line*, *extra\_args=()*, *console\_input=""*, *lines\_output=None*, *cae=None*, *shell=False*)

execute command in the current working directory of the OS console/shell.

**Parameters**

- **command\_line** *(str)* – command line string to execute on the console/shell. could contain command line args separated by whitespace characters (alternatively use *extra\_args*).
- **extra\_args** *(Sequence)* – optional sequence of extra command line arguments.
- **console\_input** *(str)* – optional string to be sent to the stdin stream of the console/shell.
- **lines\_output** *(Optional[List[str]])* – optional list to return the lines printed to stdout/stderr on execution.
- **cae** *(Optional[Any])* – optional *ConsoleApp* instance, only used for logging. to suppress any logging output pass *UNSET*.
- **shell** *(bool)* – pass True to execute command in the default OS shell (see *subprocess.run()*).

**Return type**

*int*

**Returns**

return code of the executed command or 126 if execution raised any other exception.

**class ConsoleApp**(*app\_title=""*, *app\_name=""*, *app\_version=""*, *sys\_env\_id=""*, *debug\_level=0*, *multi\_threading=False*, *suppress\_stdout=False*, *cfg\_opt\_eval\_vars=None*, *additional\_cfg\_files=()*, *cfg\_opt\_val\_stripper=None*, *formatter\_class=None*, *epilog=""*, *\*\*logging\_params*)

Bases: *AppBase*

provides command line arguments and options, config options, logging and debugging for your application.

most applications only need a single instance of this class. each instance is encapsulating a *ConfigParser* and a *ArgumentParser* instance. so only apps with threads and different sets of config options for each thread could create a separate instance of this class.

instance attributes (ordered alphabetically - ignoring underscore characters):

- *\_arg\_parser* *ArgumentParser* instance.
- *cfg\_opt\_choices* valid choices for pre-/user-defined options.
- *cfg\_opt\_eval\_vars* additional dynamic variable values that are getting set via the *cfg\_opt\_eval\_vars* argument of the method *ConsoleApp.\_\_init\_\_()* and get then used in the evaluation of *evaluable config option values*.

- `_cfg_files` iterable of config file names that are getting loaded and parsed (specify additional configuration/INI files via the `additional_cfg_files` argument).
- `cfg_options` pre-/user-defined options (dict of `Literal` instances defined via `add_option()`).
- `_cfg_parser` ConfigParser instance.
- `_main_cfg_fnam` main config file name.
- `_main_cfg_mod_time` last modification datetime of main config file.
- `_cfg_opt_val_stripper` callable to strip option values.
- `_parsed_arguments` ArgumentParser.parse\_args() return.

```
__init__(app_title="", app_name="", app_version="", sys_env_id="", debug_level=0, multi_threading=False,
        suppress_stdout=False, cfg_opt_eval_vars=None, additional_cfg_files=(),
        cfg_opt_val_stripper=None, formatter_class=None, epilog="", **logging_params)
```

initialize a new `ConsoleApp` instance.

#### Parameters

- **`app_title`** (str) – application title/description to set the instance attribute `app_title`.  
if not specified then the docstring of your app’s main module will be used (see *example*).
- **`app_name`** (str) – application instance name to set the instance attribute `app_name`.  
if not specified then base name of the main module file name will be used.
- **`app_version`** (str) – application version string to set the instance attribute `app_version`.  
if not specified then value of a global variable with the name `__version__` will be used (*if declared in the actual call stack*).
- **`sys_env_id`** (str) – system environment id to set the instance attribute `sys_env_id`.  
this value is also used as file name suffix to load all the system config variables in `sys_env<suffix>.cfg`. pass e.g. ‘LIVE’ to init this ConsoleApp instance with config values from `sys_envLIVE.cfg`.  
the default value of this argument is an empty string.

---

**Note:** if the argument value results as empty string then the value of the optionally defined OS environment variable `AE_OPTIONS_SYS_ENV_ID` will be used as default.

---

- **`debug_level`** (int) – default debug level to set the instance attribute `debug_level`.  
the default value of this argument is `DEBUG_LEVEL_DISABLED`.
- **`multi_threading`** (bool) – pass True if instance is used in multi-threading app.
- **`suppress_stdout`** (bool) – pass True (for wsgi apps) to prevent any python print outputs to stdout.
- **`cfg_opt_eval_vars`** (Optional[dict]) – dict of additional application specific data values that are used in eval expressions (e.g. `AcuSihotMonitor.ini`).
- **`additional_cfg_files`** (Iterable) – iterable of additional CFG/INI file names (opt. incl. abs/rel. path).
- **`cfg_opt_val_stripper`** (Optional[Callable]) – callable to strip/reformat/normalize the option choices values.

- **formatter\_class** (Optional[Any]) – alternative formatter class passed onto ArgumentParser instantiation.
- **epilog** (str) – optional epilog text for command line arguments/options help text (passed onto ArgumentParser instantiation).
- **logging\_params** – all other kwargs are interpreted as logging configuration values - the supported kwargs are all the method kwargs of *init\_logging()*.

**\_cfg\_parser**

ConfigParser instance

**cfg\_options: Dict[str, Literal]**

all config options

**cfg\_opt\_choices: Dict[str, Iterable]**

all valid config option choices

**cfg\_opt\_eval\_vars: dict**

app-specific vars for init of cfg options

**\_cfg\_files: list**

list of all found INI/CFG files

**\_main\_cfg\_fnam: str**

def main config file name

**\_main\_cfg\_mod\_time: float**

main config file modification datetime

**\_cfg\_opt\_val\_stripper: Optional[Callable]**

callable to strip or normalize config option choice values

**\_parsed\_arguments: Optional[Namespace]**

storing returned namespace of ArgumentParser.parse\_args() call, used to retrieve command line args

**\_arg\_parser: ArgumentParser**

ArgumentParser instance

**\_init\_default\_user\_cfg\_vars()**

init user default config variables.

override this method to add module-/app-specific config vars that can be set individually per user.

**\_init\_logging(logging\_params)**

determine and init logging config.

**Parameters**

**logging\_params** (Dict[str, Any]) – logging config dict passed as args by user that will be amended with cfg values.

**Return type**

Optional[str]

**Returns**

None if py logging is active, log file name if ae logging is set in cfg or args or empty string if no logging got configured in cfg/args.

the logging configuration can be specified in several alternative places. the precedence on various existing configurations is (the highest precedence first):

- *log\_file configuration option* specifies the name of the used ae log file (will be read after initialisation of this app instance)
- *logging\_params configuration variable* dict with a *py\_logging\_params* key to activate python logging
- *logging\_params configuration variable* dict with the ae log file name in the key *log\_file\_name*
- *py\_logging\_params configuration variable* to use the python logging module
- *log\_file configuration variable* specifying ae log file
- *logging\_params* dict passing the python logging configuration in the key *py\_logging\_params* to this method
- *logging\_params* dict passing the ae log file in the logging key *log\_file\_name* to this method

**\_\_del\_\_()**

deallocate this app instance by calling *ae.core.AppBase.shutdown()*.

**property debug\_level:** *int*

debug level property:

**Getter**

return the current debug level of this app instance.

**Setter**

change the debug level of this app instance.

**add\_cfg\_files(\*additional\_cfg\_files)**

extend list of available and additional config files (in *\_cfg\_files*).

**Parameters**

**additional\_cfg\_files** (*str*) – domain/app-specific config file names to be defined/registered additionally.

**Return type**

*str*

**Returns**

empty string on success else line-separated list of error message text.

underneath the search order of the config files variable value - the first found one will be returned:

1. the domain/app-specific *config files* added in your app code by this method. these files will be searched for the config option value in reversed order - so the last added *config file* will be the first one where the config value will be searched.
2. *config files* added via *additional\_cfg\_files* argument of *ConsoleApp.\_\_init\_\_()* (searched in the reversed order).
3. <app\_name>.INI file in the <app\_dir>
4. <app\_name>.CFG file in the <app\_dir>
5. <app\_name>.INI file in the <usr\_dir>
6. <app\_name>.CFG file in the <usr\_dir>
7. <app\_name>.INI file in the <cwd>
8. <app\_name>.CFG file in the <cwd>
9. .sys\_env.cfg in the <app\_dir>
10. .sys\_env<sys\_env\_id>.cfg in the <app\_dir>



11. `.app_env.cfg` in the `<app_dir>`
12. `.sys_env.cfg` in the `<usr_dir>`
13. `.sys_env<sys_env_id>.cfg` in the `<usr_dir>`
14. `.app_env.cfg` in the `<usr_dir>`
15. `.sys_env.cfg` in the `<cwd>`
16. `.sys_env<sys_env_id>.cfg` in the `<cwd>`
17. `.app_env.cfg` in the `<cwd>`
18. `.sys_env.cfg` in the parent folder of the `<cwd>`
19. `.sys_env<sys_env_id>.cfg` in the parent folder of the `<cwd>`
20. `.app_env.cfg` in the parent folder of the `<cwd>`
21. `.sys_env.cfg` in the parent folder of the parent folder of the `<cwd>`
22. `.sys_env<sys_env_id>.cfg` in the parent folder of the parent folder of the `<cwd>`
23. `.app_env.cfg` in the parent folder of the parent folder of the `<cwd>`
24. `value` argument passed into the `add_opt()` method call (defining the option)
25. `default_value` argument passed into this method (only if `add_option` didn't get called)

**legend of the placeholders in the above search order lists** (see also [`ae.paths.PATH\_PLACEHOLDERS`](#)):

- `<cwd>` is the current working directory of your application (determined with `os.getcwd()`)
- `<app_name>` is the base app name without extension of your main python code file.
- `<app_dir>` is the application data directory (APPDATA/`<app_name>` in Windows, `~/.config/<app_name>` in Linux).
- `<usr_dir>` is the user data directory (APPDATA in Windows, `~/.config` in Linux).
- `<sys_env_id>` is the specified argument of `ConsoleApp.__init__()`.

**`cfg_section_variable_names`**(*section*, *cfg\_parser=None*)

determine current config variable names/keys of the passed config file section.

#### Parameters

- **`section`** (str) – config file section name.
- **`cfg_parser`** (Optional[ConfigParser]) – ConfigParser instance to use (def=self.\_cfg\_parser).

#### Return type

Tuple[str, ...]

#### Returns

tuple of all config variable names.

**`_get_cfg_parser_val`**(*name*, *section*, *default\_value=None*, *cfg\_parser=None*)

determine thread-safe the value of a config variable from the config file.

#### Parameters

- **`name`** (str) – name/option\_id of the config variable.
- **`section`** (str) – name of the config section.

- **default\_value** (Optional[Any]) – default value to return if config value is not specified in any config file.
- **cfg\_parser** (Optional[ConfigParser]) – ConfigParser instance to use (def=self.\_cfg\_parser).

**Return type**

Any

**load\_cfg\_files**(*config\_modified=True*)

(re)load and parse all config files.

**Parameters**

**config\_modified** (bool) – pass False to prevent the refresh/overwrite the initial config file modified date.

**is\_main\_cfg\_file\_modified**()

determine if main config file got modified.

**Return type**

bool

**Returns**

True if the content of the main config file got modified/changed.

**get\_variable**(*name, section=None, default\_value=None, cfg\_parser=None, value\_type=None*)

get value of *config option*, OS environ or *config variable*.

**Parameters**

- **name** (str) – name of a *config option* or of an existing/declared *config variable*.
- **section** (Optional[str]) – name of the *config section*. defaulting to the app options section (**MAIN\_SECTION\_NAME**) if not specified or if None or empty string passed.
- **default\_value** (Optional[Any]) – default value to return if config value is not specified in any config file.
- **cfg\_parser** (Optional[ConfigParser]) – optional ConfigParser instance to use (def=\_cfg\_parser).
- **value\_type** (Optional[Type]) – optional type of the config value. only used for *config variables* and ignored for *config options*.

**Return type**

Any

**Returns**

variable value which will be searched in the *config options*, the OS environment and in the *config variables* in the following order and manner:

- **config option** with a name equal to the *name* argument (only if the passed *section* value is either empty, None or equal to **MAIN\_SECTION\_NAME**).
- **OS environment variable** with a matching snake+upper-cased name, compiled from the *section* and *name* arguments, separated by an underscore character.
- **config variable** with a name and section equal to the values passed into the *name* and *section* arguments.

if no variable could be found then a None value will be returned.

this method has an alias named **get\_var**() .

**get\_var**(*name*, *section=None*, *default\_value=None*, *cfg\_parser=None*, *value\_type=None*)

alias of method [get\\_variable\(\)](#)

#### Return type

[Any](#)

**set\_variable**(*name*, *value*, *cfg\_fname=None*, *section=None*, *old\_name=""*)

set/change the value of a [config variable](#) and if exists the related config option.

if the passed string in [name](#) is the id of a defined [config option](#) and [section](#) is either empty or equal to the value of [MAIN\\_SECTION\\_NAME](#) then the value of this config option will be changed too.

if the section does not exist it will be created (in contrary to Pythons ConfigParser).

#### Parameters

- **name** [\(str\)](#) – name/option\_id of the config value to set.
- **value** [\(Any\)](#) – value to assign to the config value, specified by the [name](#) argument.
- **cfg\_fname** [\(Optional\[str\]\)](#) – file name (def= [\\_main\\_cfg\\_fname](#)) to save the new option value to.
- **section** [\(Optional\[str\]\)](#) – name of the [config section](#). defaulting to the app options section ([MAIN\\_SECTION\\_NAME](#)) if not specified or if None or empty string passed.
- **old\_name** [\(str\)](#) – old name/option\_id that has to be removed (used to rename config option name/key).

#### Return type

[str](#)

#### Returns

empty string on success else error message text.

this method has an alias named [set\\_var\(\)](#).

**set\_var**(*name*, *value*, *cfg\_fname=None*, *section=None*, *old\_name=""*)

alias of method [set\\_variable\(\)](#)

#### Return type

[str](#)

**add\_argument**(*\*args*, *\*\*kwargs*)

define new command line argument.

original/underlying args/kwargs of [argparse.ArgumentParser](#) are used - please see the description/definition of [add\\_argument\(\)](#).

this method has an alias named [add\\_arg\(\)](#).

**add\_arg**(*\*args*, *\*\*kwargs*)

alias of method [add\\_argument\(\)](#)

**get\_argument**(*name*)

determine the command line parameter value.

#### Parameters

- **name** [\(str\)](#) – argument id of the parameter.

#### Return type

[Any](#)

**Returns**

value of the parameter.

this method has an alias named `get_arg()`.

**get\_arg**(*name*)

alias of method `get_argument()`

**Return type**

*Any*

**add\_option**(*name, desc, value, short\_opt="", choices=None, multiple=False*)

defining and adding a new config option for this app.

**Parameters**

- **name** (str) – string specifying the option id and short description of this new option. the name value will also be available as long command line argument option (case-sens.).
- **desc** (str) – description and command line help string of this new option.
- **value** (Any) – default value and type of the option. returned by `get_option()` if this option is not specified as command line argument nor exists as config variable in any config file. pass *UNSET* to define a boolean flag option, specified without a value on the command line. the resulting value will be *True* if the option will be specified on the command line, else *False*. specifying a value on the command line results in a *SystemExit* on parsing.
- **short\_opt** (Union[str, UnsetType]) – short option character. if not passed or passed as '' then the first character of the name will be used. passing *UNSET* or *None* prevents the declaration of a short option. please note that the short options 'h', 'D' and 'L' are already used internally by the classes `ArgumentParser` and `ConsoleApp`.
- **choices** (Optional[Iterable]) – list of valid option values (optional, default=allow all values).
- **multiple** (bool) – True if option can be added multiple times to command line (optional, default=False).

the value of a config option can be of any type and gets represented by an instance of the *Literal* class. supported value types and literals are documented [here](#).

this method has an alias named `add_opt()`.

**add\_opt**(*name, desc, value, short\_opt="", choices=None, multiple=False*)

alias of method `add_option()`

**\_change\_option**(*name, value*)

change config option and any references to it.

**get\_option**(*name, default\_value=None*)

determine the value of a config option specified by its name (option id).

**Parameters**

- **name** (str) – name/id of the config option.
- **default\_value** (Optional[Any]) – default value of the option (if not defined with `add_option()`).

**Return type**

*Any*

**Returns**

first found value of the option identified by *name*. the returned value has the same type as the value specified in the *add\_option()* call. if not given on the command line, then it gets search next in default config section (*MAIN\_SECTION\_NAME*) of the collected config files (the exact search order is documented in the doc-string of the method *add\_cfg\_files()*). if not found in the config file then the default value specified of the option definition (the *add\_option()* call) will be used. the other default value, specified in the *default\_value* kwarg of this method, will be returned only if the option name/id never got defined.

this method has an alias named *get\_opt()*.

**get\_opt**(*name*, *default\_value=None*)

alias of method *get\_option()*

**Return type**

*Any*

**set\_option**(*name*, *value*, *cfg\_fnam=None*, *save\_to\_config=True*)

set or change the value of a config option.

**Parameters**

- **name** (str) – id of the config option to set.
- **value** (Any) – value to assign to the option, identified by *name*.
- **cfg\_fnam** (Optional[str]) – config file name to save new option value. if not specified then the default file name of *set\_variable()* will be used.
- **save\_to\_config** (bool) – pass False to prevent to save the new option value also to a config file. the value of the config option will be changed in any case.

**Return type**

str

**Returns**

"/empty string on success else error message text.

this method has an alias named *set\_opt()*.

**set\_opt**(*name*, *value*, *cfg\_fnam=None*, *save\_to\_config=True*)

alias of method *set\_option()*

**Return type**

str

**parse\_arguments()**

parse all command line args.

this method get normally only called once and after all the options have been added with *add\_option()*. *add\_option()* will then set the determined config file value as the default value and then the following call of this method will overwrite it with command line argument value, if given.

**property user\_id**

id of the user of this app.

**load\_user\_cfg()**

load users configuration.

**register\_user**(\*\**user\_data*)

register the current user and create/copy a new set of user specific config vars.

**Parameters**

**user\_data** – user data dict.

---

**Note:** this method will overwrite an existing user with the same user id, with the passed user data and the config variable values of the current/default user.

---

**user\_section**(*section, name*)

return the user section name if the passed (section, name) setting id is user-specific.

**Parameters**

- **section** (*str*) – section name.
- **name** (*str*) – variable name.

**Return type**

*str*

**Returns**

passed section name or user-specific section name.

**app\_env\_dict**()

collect run-time app environment data and settings - for app logging and debugging.

**Return type**

*Dict[str, Any]*

**Returns**

dict with app environment data/settings.

**run\_app**()

prepare app run. call after definition of command line arguments/options and before run of app code.

**show\_help**()

print help message, listing defined command line args and options, to console output/stream.

includes command line args defined with [add\\_argument\(\)](#), options defined with [add\\_option\(\)](#) and the args/kwargs defined with the respective [ArgumentParser](#) methods (see description/definition of [print\\_help\(\)](#) of [ArgumentParser](#)).

## 4.18 ae.sys\_core

### 4.18.1 dynamic system configuration, initialization and connection

this module allows the dynamic (data-driven) configuration and connection of your application to all their needed external systems (like servers, databases, cloud stores, ...).

## system classes

the *SystemBase* class represents a single system. the creation of instances of this class is automatically done *dynamically* by the (mostly unique) instance of the class *UsedSystems*.

each instance of *SystemBase* is using a system-specific connector class to establish an internal connection to the represented system. for the implementation of a connector class, simply inherit from the abstract base class *SystemConnectionBase* - also provided by this module.

by creating an instance of the class *UsedSystems* you get a dictionary-like object with all the properties for each configured system. the items of this dict are instances of the class *SystemBase*.

## dynamic system properties

the properties of your systems are specified within config *files* as *variables*.

the only mandatory config variable is a dict with the name *availableSystems*, with the system-ids as dict keys. the items of this dict are also dicts with string keys and called *system configs*. only the key *name* is mandatory in these system configs. the following system config dict keys are supported:

- **name** : system name (long name of system-id, only used for repr() and debugging).
- **credential\_keys** : List[str] of credential keys needed to connect to this system.
- **feature\_keys** : List[str] of feature keys supported by this system.
- **available\_rec\_types** : Dict[str, str] of supported rec type ids and names.
- **connector\_module** : name of the connector package.module (def=`ae.sys_data_<system-id-in-lower-case>`).
- **connector\_class** : name of the connector class (def=`<system-id>SysConnector`),

the value of each credential and feature key gets automatically dynamically read from a config option or variable. the name of these config options/variables consists of the key string, prefixed with the system id in lower case. so if e.g. the value of a credential key 'User' for the system id 'Abc' will be loaded from a config variable with the name *abcUser*. all the credential and feature config variables can be either stored in the main or the systems section of the config files - see also *config\_value()*.

an example of a system configuration you find in the config file *test.cfg*, situated in the tests folder of this package.

## multiple system configurations

to provide multiple system configurations for the same application (suite) you can use individual config files for each application instance with a separate config file (see `.sys_env<SYS_ENV_ID>.cfg` [here](#)).

## Module Attributes

---

*SYS\_SECTION\_NAME*

section name for system config variables

---

## Functions

<code>config_value(var_name, console_app)</code>	determine system configuration setting value from application config options/variables.
--	---

## Classes

<code>SystemBase(sys_id, console_app, credentials)</code>	instance represents an external system, including their system properties and a connection object.
<code>SystemConnectorBase(system)</code>	abstract system connector base class - sub-class establishes the connection to an external system.
<code>UsedSystems(console_app, *selected_systems, ...)</code>	an instance of this class is keeping a dictionary of all the found and used systems of this application.

**SYS\_SECTION\_NAME = 'aeSystems'**

section name for system config variables

**config\_value(var\_name, console\_app)**

determine system configuration setting value from application config options/variables.

### Parameters

- **var\_name** (str) – config variable name. variable value gets searched in the application instance environment, first in the *application config options*, then in the *application config variables* (first within the section MAIN\_SECTION\_NAME/aeOptions and then within SYS\_SECTION\_NAME/aeSystems).
- **console\_app** (ConsoleApp) – ConsoleApp instance of the application providing these config options/variables.

### Return type

Any

### Returns

config variable value if found else None.

**class SystemBase(sys\_id, console\_app, credentials, features=(), rec\_types=None)**

Bases: object

instance represents an external system, including their system properties and a connection object.

**instances:** int = 0

**\_\_init\_\_(sys\_id, console\_app, credentials, features=(), rec\_types=None)**

create new *SystemBase* instance.

### Parameters

- **sys\_id** (str) – unique str to identify a system (also used as prefix/suffix).
- **console\_app** (ConsoleApp) – ConsoleApp instance of the application using these systems.
- **credentials** (Dict[str, str]) – dict to access system, containing e.g. user name, password, token, dsn...



- **features** (Sequence[str]) – optional list with special features for this system (see SDF\_\* constants).
- **rec\_types** (Optional[Dict[str, str]]) – optional dict of available record types for this system.

**sys\_id**

system id

**console\_app**

application instance and config environment

**credentials**

credentials to connect to this system

**features**

system specific features and configs

**available\_rec\_types:** Dict[str, str]

record types available in this system

**connection:** Any

object with opt. connect() and disconnect()/close() methods

**conn\_error:** str

error message of last system access or dis-/connect

**\_\_repr\_\_()**

representation string of this *SystemBase* instance.

**Return type**

str

**Returns**

representation string of this instance.

**connect**(sys\_config, force\_reconnect=False)

connect this instance to his system using a system connector class.

**Parameters**

- **sys\_config** (Dict[str, str]) – dict with the *dynamic config properties of this system*.
- **force\_reconnect** (bool) – optional; pass True to force re-connection (even if self.connection is already initialized).

**Return type**

str

**Returns**

error message or empty string in case of no errors while re-connecting.

**disconnect()**

disconnect this external system.

**Return type**

str

**Returns**

error message string or empty string if no errors occurred on disconnection.

---

```
class SystemConnectorBase(system)
```

Bases: `ABC`

abstract system connector base class - sub-class establishes the connection to an external system.

**instances:** `int = 0`

**\_\_init\_\_**(system)

create new `SystemBase` instance.

**Parameters**

**system** `(SystemBase)` – instance of `SystemBase` class that is representing the system using this connection.

**system**

SystemBase instance

**console\_app**

application instance and config environment

**last\_err\_msg:** `str`

last system connection error message(s)

**\_\_repr\_\_**()

representation string of this `SystemBase` instance.

**Return type**

`str`

**Returns**

representation string of this instance.

**abstract connect()**

abstract method - raising `NotImplementedError`, so has to be implemented by the sub-class.

**Return type**

`str`

```
class UsedSystems(console_app, *selected_systems, **entered_credentials)
```

Bases: `OrderedDict`

an instance of this class is keeping a dictionary of all the found and used systems of this application.

the keys of this dictionary-like class are the system ids of your application. the items are instances of the `SystemBase` class.

each instance is providing additionally the following instance attributes:

- `console_app` - instance of the `ConsoleApp` class that is using these systems.
- `available_systems` - system properties loaded from config files.

all the system-specific properties are stored within the `SystemBase` instances.

**\_\_init\_\_**(console\_app, \*selected\_systems, \*\*entered\_credentials)

create new instance of `UsedSystems`.

**Parameters**

- **console\_app** `(ConsoleApp)` – `ConsoleApp` instance of the application using these systems.

- **selected\_systems** (str) – optional iterable of system id strings of the available systems that have to be initialized for the app specified by the `console_app` argument. if no system id get passed then all available systems will be initialized from the config variable `availableSystems`.
- **entered\_credentials** (str) – optional dict of credentials entered by a user at run-time (overwriting the values set in the related config variable `availableSystems`). the keys of this dictionary starting with the system id in lower case, followed by one of the key strings specified in the `credential_keys` config list (see *dynamic system properties*).

**\_load\_merge\_cred\_feat**(*entered\_credentials*)

load and initialize all used system configs and merge entered credentials into it.

**Parameters**

**entered\_credentials** (Dict[str, str]) – passed over unchanged from *UsedSystems* - see *entered\_credentials*.

**\_add\_system**(*sys\_id*, *credentials*, *features*=())

add new *SystemBase* instance to this *UsedSystems* instance.

**Parameters**

- **sys\_id** (str) – system id.
- **credentials** (Dict[str, str]) – credentials of the new system to add.
- **features** (Sequence[str]) – optional list with special features for this system (see SDF\_\* constants).

**connect**(*force\_reconnect*=False)

connect all the selected systems of this instance.

**Parameters**

**force\_reconnect** (bool) – optional; pass True to force re-connection (even if self.connection is initialized).

**Return type**

str

**Returns**

error message or empty string in case of no errors while re-connecting.

**disconnect**()

disconnect all available and already connected systems.

**Return type**

str

**Returns**

error message string or empty string if no errors occurred on disconnection.

## 4.19 ae.sys\_data

### 4.19.1 external system data structures

this module allows your application to easily manage data structures to interface data flows from and onto external systems.

an external system can be nearly anything: like e.g.: a database or web server or an application or software suite with an interface for data exchanges; even data files can be used as an external system.

the same data can be represented differently on different systems. this module allows you to specify individual data structures for each system and provides hooks to easily integrate system-specific data conversion functions.

#### data structures

use the classes of this module to define any kind of simple data structures - like lists, mappings and trees:

- a list can get represented by an instance of one of the classes *Records* or *Values*. each *Records* instance is a sequence of 0..n *Record* instances. a *Values* instance is a sequence of 1..n *Value* instances.
- a mapping get represented by an instance of the class *Record*, whereas each mapping item gets represented by an instance of the private class *\_Field*.
- a node of a tree structure can be represented by an instance of the classes *Values*, *Records* or *Record*.

by combining these simple data structures you can build any complex data structures. the root of such a complex data structure can be defined by an instance of either *Records* or *Record*.

the leaves of any simple or complex data structure are represented by instances of the *Value* class.

the following diagram is showing a complex data structure with all the possible combinations (of a single system):

#### data reference index paths

an *index path* is a tuple of indexes/keys that is referencing one part or a value of a data structure.

the items of an index path are either of type int (specifying a list index in a *Values* or *Records* instance) or of type str (specifying a field name of a *Record* instance).

e.g. the index path to reference in the above graph example the field **DA** from the record **Record (root)** would be ('D', 0, 'DA'). the first item is referencing the field **D** in the root record. the second item **0** is referencing the first item of the *Records* instance/value of field **D**, which is a *Record* instance. and finally the last item **DA** specifies the field name in this *Record* instance.

the hierarchically highest placed *Record* instance in a data structure is called the *root record*, and the *Value* instances at the lowest level in a data structure are the leaves of a data structure.

each *\_Field* is providing apart from its field value also a reference to its root record as well as a *root index*. the root index is an index path to reference the *\_Field* instance from the root record.

if the format/representation of the data value or of any references differs at one of the used systems then a *\_Field* instance allows you to store a separate system-specific value. to access a system-specific value or reference in your data structure you have to specify additionally to the index path also a *system identifier* and optionally a *direction identifier*.

## system and direction identifiers

a *system id* is a user-defined string that is uniquely identifying a system and should consist of at least two characters (see `_ASP_SYS_MIN_LEN`).

a *direction id* is specifying the data flow direction. the two pre-defined direction ids are:

- `FAD_FROM` for pulling data from a system or
- `FAD_ONTO` for pushing data onto a system.

the methods `pull()` and `push()` of the `Record` class are used to pull/push and convert system-specific data from/onto a system.

## Module Attributes

<code>ACTION_INSERT</code>	insert action
<code>ACTION_UPDATE</code>	update action
<code>ACTION_UPSERT</code>	insert or update (if already exists) action
<code>ACTION_DELETE</code>	delete action
<code>ACTION_SEARCH</code>	search action
<code>ACTION_PARSE</code>	parse action
<code>ACTION_BUILD</code>	build action
<code>ACTION_PULL</code>	pull-from-system action
<code>ACTION_PUSH</code>	push-to-system action
<code>ACTION_COMPARE</code>	compare action
<code>FAT_IDX</code>	main/system field name within parent Record or list index within Records/Values
<code>FAT_VAL</code>	main/system field value - storing one of the <code>VALUE_TYPES</code> instance
<code>FAT_CLEAR_VAL</code>	field default/clear value (init by <code>_Field.set_clear_val()</code> , used by <code>clear_leaves()</code> )
<code>FAT_REC</code>	root Record instance
<code>FAT_RCX</code>	field index path ( <code>idx_path</code> ) from the root Record instance
<code>FAT_CAL</code>	calculator callable
<code>FAT_CHK</code>	validator callable
<code>FAT_CNV</code>	system value converter callable
<code>FAT_FLT</code>	field filter callable
<code>FAT_SQE</code>	SQL expression to fetch field value from db
<code>ALL_FATS</code>	tuple of all pre-defined field aspect types/prefixes
<code>FAD_FROM</code>	FROM field aspect direction
<code>FAD_ONTO</code>	ONTO field aspect direction
<code>IDX_PATH_SEP</code>	separator character used for <code>idx_path</code> values (especially if field has a Record value).
<code>ALL_FIELDS</code>	special key of <code>fields_patches</code> argument of <code>Record.copy()</code> to allow aspect value patching for all fields.
<code>CALLABLE_SUFFIX</code>	suffix for aspect keys - used by <code>_Field.set_aspects()</code>
<code>IDX_TYPES</code>	tuple of classes/types used for system data index path items
<code>PATH_TYPES</code>	path items, plus field name path, plus index path
<code>AspectKeyType</code>	type of <code>_Field</code> aspect key
<code>IdxItemType</code>	types of <code>idx_path</code> items
<code>IdxPathType</code>	<code>idx_path</code> type

continues on next page

Table 4.3 – continued from previous page

<i>IdxTypes</i>	types of either <i>idx_path</i> or <i>idx_path</i> items
<i>NodeType</i>	node types ( <i>NODE_TYPES</i> )
<i>ListType</i>	list/sequence types ( <i>LIST_TYPES</i> )
<i>ValueType</i>	value types ( <i>VALUE_TYPES</i> )
<i>NodeChildType</i>	node child types
<i>FieldCallable</i>	callable with <i>_Field</i> argument
<i>FieldValCallable</i>	callable with <i>_Field</i> and additional argument
<i>VALUE_TYPES</i>	tuple of classes/types used for system data values
<i>LIST_TYPES</i>	tuple of classes/types used for system data lists
<i>NODE_TYPES</i>	tuple of classes/types used for system data nodes
<i>NODE_CHILD_TYPES</i>	tuple of classes/types used for system data node children
<i>PARENT_TYPES</i>	value types that can have children (Value excluded)

## Functions

<i>aspect_key</i> (type_or_key[, system, direction])	compiles an aspect key from the given args
<i>aspect_key_direction</i> (key)	determines the direction id string from an aspect key.
<i>aspect_key_system</i> (key)	determines the system id string from an aspect key.
<i>compose_current_index</i> (node, idx_path, ...)	determine tuple with the current indexes.
<i>deeper</i> (deepness, instance)	check and calculate resulting/remaining deepness for <i>Record/_Field/Records.copy()</i> when going one level deeper.
<i>field_name_idx_path</i> (field_name[, ...])	converts a field name path string into an index path tuple.
<i>field_names_idx_paths</i> (field_names)	return list of the full <i>idx_path</i> names for all the fields specified in the <i>field_names</i> argument.
<i>get_current_index</i> (node)	get current index of passed <i>node</i> .
<i>idx_path_field_name</i> (idx_path[, add_sep])	convert index path tuple/list into field name string.
<i>init_current_index</i> (node, idx_path, use_curr_idx)	determine current index of <i>node</i> and if not set then initialize to the first index path item.
<i>set_current_index</i> (node[, idx, add])	set current index of <i>node</i> .
<i>string_to_records</i> (str_val, field_names[, ...])	convert formatted string into a <i>Records</i> instance containing several <i>Record</i> instances.
<i>template_idx_path</i> (idx_path[, is_sub_rec])	check/determine if <i>idx_path</i> is referring to template item.
<i>use_current_index</i> (node, idx_path, use_curr_idx)	determine index path of <i>node</i> by using current index of <i>node</i> if exists and is enabled by <i>use_curr_idx</i> arg.
<i>use_rec_default_root_rec_idx</i> (rec, root_rec)	helper function to determine resulting root record and root index.
<i>use_rec_default_sys_dir</i> (rec, system, direction)	helper function to determine resulting system/direction.

## Classes

<i>Record</i> ([template, fields, system, ...])	instances of this mapping class are used to represent record-like data structures.
<i>Records</i> ([seq])	Records class.
<i>Value</i> ([iterable])	represents a value.
<i>Values</i> ([seq])	ordered/mutable sequence/list, which contains 0..n instances of the class <i>Value</i> .

**ACTION\_INSERT** = 'INSERT'

insert action

**ACTION\_UPDATE** = 'UPDATE'

update action

**ACTION\_UPSERT** = 'UPSERT'

insert or update (if already exists) action

**ACTION\_DELETE** = 'DELETE'

delete action

**ACTION\_SEARCH** = 'SEARCH'

search action

**ACTION\_PARSE** = 'PARSE'

parse action

**ACTION\_BUILD** = 'BUILD'

build action

**ACTION\_PULL** = 'PULL'

pull-from-system action

**ACTION\_PUSH** = 'PUSH'

push-to-system action

**ACTION\_COMPARE** = 'COMPARE'

compare action

**FAT\_IDX** = 'idx'

main/system field name within parent Record or list index within Records/Values

**FAT\_VAL** = 'vle'

main/system field value - storing one of the VALUE\_TYPES instance

**FAT\_CLEAR\_VAL** = 'vwc'

field default/clear value (init by `_Field.set_clear_val()`, used by `clear_leaves()`)

**FAT\_REC** = 'rrd'

root Record instance

**FAT\_RCX** = 'rrx'

field index path (idx\_path) from the root Record instance

**FAT\_CAL** = 'clc'

calculator callable

**FAT\_CHK = 'chk'**

validator callable

**FAT\_CNV = 'cnv'**

system value converter callable

**FAT\_FLT = 'flt'**

field filter callable

**FAT\_SQE = 'sqc'**

SQL expression to fetch field value from db

**ALL\_FATS = ('idx', 'vle', 'vwc', 'rrd', 'rrx', 'clc', 'chk', 'cnv', 'flt', 'sqc')**

tuple of all pre-defined field aspect types/prefixes

**FAD\_FROM = 'From'**

FROM field aspect direction

**FAD\_ONTO = 'Onto'**

ONTO field aspect direction

**IDX\_PATH\_SEP = '/'**

separator character used for idx\_path values (especially if field has a Record value).

don't use dot char because this is used e.g. to separate system field names in xml element name paths.

**ALL\_FIELDS = '\*\*'**

special key of fields\_patches argument of Record.copy() to allow aspect value patching for all fields.

**CALLABLE\_SUFFIX = '()'**

suffix for aspect keys - used by \_Field.set\_aspects()

**\_ASP\_TYPE\_LEN = 3**

aspect key type string length

**\_ASP\_DIR\_LEN = 4**

aspect key direction string length

**\_ASP\_SYS\_MIN\_LEN = 2**

aspect key system id string length

**IDX\_TYPES: Tuple[Type, ...] = (<class 'int'>, <class 'str'>)**

tuple of classes/types used for system data index path items

**PATH\_TYPES = (<class 'int'>, <class 'str'>, <class 'tuple'>)**

path items, plus field name path, plus index path

**AspectKeyType**

type of \_Field aspect key

**IdxItemType**

types of idx\_path items

alias of Union[int, str]

**IdxPathType**

idx\_path type

alias of Tuple[Union[int, str], ...]



**IdxTypes**

types of either `idx_path` or `idx_path` items

alias of `Union[int, str, Tuple[Union[int, str], ...]]`

**NodeType**

node types (`NODE_TYPES`)

alias of `Union[Record, Records]`

**ListType**

list/sequence types (`LIST_TYPES`)

alias of `Union[Values, Records]`

**ValueType**

value types (`VALUE_TYPES`)

alias of `Union[Value, Values, Record, Records]`

**NodeChildType**

node child types

alias of `Union[_Field, Record]`

**FieldCallable**

callable with `_Field` argument

alias of `Callable[[_Field], Any]`

**FieldValCallable**

callable with `_Field` and additional argument

alias of `Callable[[_Field, Any], Any]`

**aspect\_key**(*type\_or\_key*, *system=""*, *direction=""*)

compiles an aspect key from the given args

**Parameters**

- **type\_or\_key** *(str)* – either `FAT_*` type or full key (including already the system and direction)-
- **system** *(str)* – system id string (if `type_or_key` is a pure `FAT_*` constant).
- **direction** *(str)* – direction string `FAD_*` constant (if `type_or_key` is a pure `FAT_*` constant).

**Return type**

`str`

**Returns**

compiled aspect key as string.

**aspect\_key\_system**(*key*)

determines the system id string from an aspect key.

**Parameters**

**key** *(str)* – aspect key string.

**Return type**

`str`

**Returns**

system id (SDI\_\* constant).

**aspect\_key\_direction(*key*)**

determines the direction id string from an aspect key.

**Parameters**

**key** *//* (*str*) – aspect key string.

**Return type**

*str*

**Returns**

direction id (FAD\_\* constant).

**deeper(*deepness*, *instance*)**

check and calculate resulting/remaining deepness for Record/\_Field/Records.copy() when going one level deeper.

**Parameters**

- **deepness** *//* (*int*) – <0 will be returned unchanged until last level is reached: -1==full deep copy, -2==deep copy until deepest Value, -3==deep copy until deepest \_Field.
- **instance** *//* (*Any*) – instance to be processed/copied (if this method is returning != 0/zero).

**Return type**

*int*

**Returns**

if deepness == 0 then return 0, if deepness < 0 then return 0 if the deepest level is reached, else (deepness > 0) return deepness - 1.

**field\_name\_idx\_path(*field\_name*, *return\_root\_fields=False*)**

converts a field name path string into an index path tuple.

**Parameters**

- **field\_name** *//* (*Union[int, str, Tuple[Union[int, str], ...]]*) – field name str or field name index/path string or field index tuple or int (for Records index).
- **return\_root\_fields** *//* (*bool*) – pass True to also return len()==1-tuple for fields with no deeper path (def=False).

**Return type**

*Tuple[Union[int, str], ...]*

**Returns**

index path tuple (idx\_path) or empty tuple if the field has no deeper path and return\_root\_fields==False.

**field\_names\_idx\_paths(*field\_names*)**

return list of the full idx paths names for all the fields specified in the field\_names argument.

**Parameters**

**field\_names** *//* (*Sequence[Tuple[Union[int, str], ...]]*) – sequence/list/tuple of field (main or system) names.

**Return type**

*List[Tuple[Union[int, str], ...]]*

**Returns**

list of their idx paths names.

**idx\_path\_field\_name**(*idx\_path*, *add\_sep=False*)

convert index path tuple/list into field name string.

**Parameters**

- **idx\_path** (Tuple[Union[int, str], ...]) – index path to convert.
- **add\_sep** (bool) – pass True to always separate index with IDX\_PATH\_SEP. the default value False will only put a separator char if the field value is a Record (to separate the root field name from the subfield name).

**Return type**

str

**Returns**

field name string.

**compose\_current\_index**(*node*, *idx\_path*, *use\_curr\_idx*)

determine tuple with the current indexes.

**Parameters**

- **node** (Union[Values, Records, Record]) – root node/list (Record or Records/Values instance) to process.
- **idx\_path** (Tuple[Union[int, str], ...]) – index path relative to root node passed in *node* arg.
- **use\_curr\_idx** (List) – list of index counters within *idx\_path* where the current index has to be used.

**Return type**

Tuple[Union[int, str], ...]

**Returns**

tuple of current indexes.

**get\_current\_index**(*node*)

get current index of passed *node*.

**Parameters**

**node** (Union[Values, Records, Record]) – instance of Record or Records (real node) or Values (simple list).

**Return type**

Union[int, str, None]

**Returns**

current index value.

**init\_current\_index**(*node*, *idx\_path*, *use\_curr\_idx*)

determine current index of *node* and if not set then initialize to the first index path item.

**Parameters**

- **node** (Union[Values, Records, Record]) – root node/list (Record or Records/Values instance) to process.
- **idx\_path** (Tuple[Union[int, str], ...]) – index path relative to root node passed in *node* arg.
- **use\_curr\_idx** (Optional[List]) – list of index counters within *idx\_path* where the current index has to be used.

**Return type**`Tuple[Union[int, str], ...]`**Returns**

tuple of current indexes.

**set\_current\_index**(*node*, *idx=None*, *add=None*)set current index of *node*.**Parameters**

- **node** `(Union[Values, Records, Record])` – root node/list (Record or Records/Values instance) to process.
- **idx** `(Union[int, str, None])` – index value to set (str for field name; int for list index); if given *add* will be ignored.
- **add** `(Optional[int])` – value to add to list index; will be ignored if the *idx* argument is specified.

**Return type**`Union[int, str]`**Returns**

the finally set/new index value.

**use\_current\_index**(*node*, *idx\_path*, *use\_curr\_idx*, *check\_idx\_type=False*, *delta=1*)determine index path of *node* by using current index of *node* if exists and is enabled by *use\_curr\_idx* arg.**Parameters**

- **node** `(Union[Values, Records, Record])` – root node/list (Record or Records/Values instance) to process.
- **idx\_path** `(Tuple[Union[int, str], ...])` – index path relative to root node passed in *node* arg.
- **use\_curr\_idx** `(Optional[List])` – list of index counters within *idx\_path* where the current index has to be used.
- **check\_idx\_type** `(bool)` – pass True to additionally check if the index type is correct (def=False).
- **delta** `(int)` – value to decrease the list index counters within *use\_curr\_idx* (def=1).

**Return type**`Tuple[Union[int, str], ...]`**Returns**

tuple of current indexes.

**string\_to\_records**(*str\_val*, *field\_names*, *rec\_sep=''*, *fld\_sep=''*, *root\_rec=None*, *root\_idx=()*)convert formatted string into a *Records* instance containing several *Record* instances.**Parameters**

- **str\_val** `(str)` – formatted string to convert.
- **field\_names** `(Sequence)` – list/tuple of field names of each record
- **rec\_sep** `(str)` – character(s) used in *str\_val* to separate the records.
- **fld\_sep** `(str)` – character(s) used in *str\_val* to separate the field values of each record.
- **root\_rec** `(Optional[Record])` – root to which the returned records will be added.

- **root\_idx** (Tuple[Union[int, str], ...]) – index path from root where the returned records will be added.

**Return type***Records***Returns**converted *Records* instance.**template\_idx\_path**(idx\_path, is\_sub\_rec=False)check/determine if *idx\_path* is referring to template item.**Parameters**

- **idx\_path** (Tuple[Union[int, str], ...]) – index path to check.
- **is\_sub\_rec** (bool) – pass True to only check sub-record-fields (will then return always False for root-fields).

**Return type**

bool

**Returns**True if *idx\_path* is referencing a template item (with index zero/0), else False.**use\_rec\_default\_root\_rec\_idx**(rec, root\_rec, idx=(), root\_idx=(), met="")

helper function to determine resulting root record and root index.

**Parameters**

- **rec** (*Record*) – current *Record* instance.
- **root\_rec** (Optional[*Record*]) – default root record (def=`rec`).
- **idx** (Optional[Tuple[Union[int, str], ...]]) – current index of *rec*.
- **root\_idx** (Optional[Tuple[Union[int, str], ...]]) – default root index.
- **met** (str) – calling method/function name (used only for assert error message, def="").

**Return type**Tuple[*Record*, Tuple[Union[int, str], ...]]**Returns**

resulting root record and root index (as tuple).

**use\_rec\_default\_sys\_dir**(rec, system, direction)

helper function to determine resulting system/direction.

**Parameters**

- **rec** (Optional[*Record*]) – current *Record* instance.
- **system** (Optional[str]) – default system id.
- **direction** (Optional[str]) – default direction (see FAD\_\* constants).

**Return type**

Tuple[str, str]

**Returns**

resulting system and direction (as tuple).

**class Value**(*iterable=()*, /)

Bases: `list`

represents a value.

this class inherits directly from the Python list class. each instance can hold either a (single/atomic) value (which can be anything: numeric, char/string or any object) or a list of these single/atomic values.

**\_\_getitem\_\_**(*key*)

determine atomic value.

**Parameters**

**key** `(Union[slice, int])` – list index if value is a list.

**Return type**

`Union[Any, MutableSequence[Any]]`

**Returns**

list item value.

**\_\_setitem\_\_**(*key, value*)

set/initialize list item identified by *key* to the value passed in *value*.

**Parameters**

- **key** `(Union[slice, int])` – list index if value is a list.
- **value** `(Any)` – the new value of the list item.

**Return type**

`None`

**\_\_repr\_\_**()

representation which can be used to serialize and re-create *Value* instance.

**Return type**

`str`

**Returns**

Value representation string.

**\_\_str\_\_**()

string representation of this *Value* instance.

**Return type**

`str`

**Returns**

Value string.

**property initialized**

flag if this *Value* instance got already initialized.

**Returns**

True if already set to a value, else False.

**node\_child**(*idx\_path, moan=False, \*\*\_\_*)

check if *idx\_path* is correct (has to be empty) and if yes then return self.

this method is to simplify the data structure hierarchy implementation.

**Parameters**

- **idx\_path** `(Tuple[Union[int, str], ...])` – this argument has to be an empty tuple/list.

- **moan** (bool) – pass True to raise AssertionError if *idx\_path* is not empty.

**Return type**`Optional[Value]`**Returns**self or None (if *idx\_path* is not empty and *moan* == False).**value**(\**idx\_path*, \*\*\_\_)check if *idx\_path* is correct (has to be empty) and if yes then return self.

this method is to simplify the data structure hierarchy implementation.

**Parameters****idx\_path** (Union[int, str]) – this argument has to be an empty tuple.**Return type**`Optional[Value]`**Returns**self or None (if *idx\_path* is not empty and *moan* == False).**val**(\**idx\_path*, \*\*\_\_)check if *idx\_path* is correct (either empty or contain one int) and if yes then return list item.

this method is to simplify the data structure hierarchy implementation.

**Parameters****idx\_path** – this argument is either empty or contains a list index.**Returns**

atomic/single value or list item value or empty string.

**set\_val**(*val*, \**idx\_path*, \*\*\_\_)

set a/the value of this instance.

**Parameters**

- **val** (Any) – simple/atomic value to be set.
- **idx\_path** (Union[int, str]) – this argument is either empty or contains a list index.

**Returns**

self.

**copy**(\*\_, \*\*\_\_)

copy the value of this Value instance into a new one.

**Returns**

new Value instance containing the same immutable value.

**clear\_leaves**(\*\*\_\_)

clear/reset the value of this instance.

use self[-1] = “ to clear only the newest/top val.

**Returns**

self.

**class Values**(*seq*=())Bases: `list`ordered/mutable sequence/list, which contains 0..n instances of the class `Value`.

**\_\_init\_\_**(*seq=()*)

create new *Values* instance.

**Parameters**

**seq** (Iterable) – Iterable used to initialize the new instance (pass list, tuple or other iterable).

**\_\_repr\_\_**()

Return repr(self).

**Return type**

str

**\_\_str\_\_**()

Return str(self).

**Return type**

str

**node\_child**(*idx\_path*, *use\_curr\_idx=None*, *moan=False*, *selected\_sys\_dir=None*)

determine and return node instance specified by *idx\_path* if exists in this instance or underneath.

**Parameters**

- **idx\_path** (Union[int, str, Tuple[Union[int, str], ...]]) – index path to the node, relative to this instance.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.
- **moan** (bool) – flag to check data integrity; pass True to raise AssertionError if not.
- **selected\_sys\_dir** (Optional[dict]) – optional dict to return the currently selected system/direction.

**Return type**

Union[Value, Values, Record, Records, None]

**Returns**

found node instance or None if not found.

**value**(*\*idx\_path*, *system="*, *direction="*, *\*\*kwargs*)

determine the ValueType instance referenced by *idx\_path* of this *Values/Records* instance.

**Parameters**

- **idx\_path** (Union[int, str]) – index path items.
- **system** (str) – system id.
- **direction** (str) – direction id.
- **kwargs** – extra args (will be passed to underlying data structure).

**Return type**

Union[Value, Values, Record, Records, None]

**Returns**

found Value or Values instance, or None if not found.

**set\_value**(*value*, *\*idx\_path*, *system="*, *direction="*, *protect=False*, *root\_rec=None*, *root\_idx=()*, *use\_curr\_idx=None*)

set the ValueType instance referenced by *idx\_path* of this *Values/Records* instance.



**Parameters**

- **value** (Union[Value, Values, Record, Records]) – ValueType instance to set/change.
- **idx\_path** (Union[int, str]) – index path items.
- **system** (str) – system id.
- **direction** (str) – direction id.
- **protect** (bool) – pass True to prevent replacement of already existing ValueType.
- **root\_rec** (Optional[Record]) – root record.
- **root\_idx** (Tuple[Union[int, str], ...]) – root index.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.

**Return type**

Union[Values, Records]

**Returns**

self (this instance of Values or Records).

**val**(\*idx\_path, system="", direction="", flex\_sys\_dir=True, use\_curr\_idx=None, \*\*kwargs)  
determine the user/system value referenced by idx\_path of this Values/Records instance.

**Parameters**

- **idx\_path** (Union[int, str]) – index path items.
- **system** (str) – system id.
- **direction** (str) – direction id.
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.
- **kwargs** – extra args (will be passed to underlying data structure).

**Return type**

Any

**Returns**

found user/system value, or None if not found or empty string if value was not set yet.

**set\_val**(val, \*idx\_path, protect=False, extend=True, use\_curr\_idx=None)  
set the user/system value referenced by idx\_path of this Values instance.

**Parameters**

- **val** (Any) – user/system value to set/change.
- **idx\_path** (Union[int, str]) – index path items.
- **protect** (bool) – pass True to prevent replacement of already existing ValueType.
- **extend** (bool) – pass True to allow extension of data structure.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.

**Return type**

Values

**Returns**

self (this instance of *Values*).

**copy**(*deepness=0*, *root\_rec=None*, *root\_idx=()*, *\*\*kwargs*)

copy the values/records of this *Values/Records* instance.

**Parameters**

- **deepness** (int) – deep copy levels: <0==see deeper(), 0==only copy current instance, >0==deep copy to deepness value - *\_Field* occupies two deepness: 1st=*\_Field*, 2nd=Value.
- **root\_rec** (Optional[*Record*]) – destination root record.
- **root\_idx** (Tuple[Union[int, str], ...]) – destination index path (tuple of field names and/or list/Records indexes).
- **kwargs** – additional arguments (will be passed on - most of them used by *Record.copy()*).

**Return type**

Union[*Values*, *Records*]

**Returns**

new instance of self (which is an instance of *Values* or *Records*).

**clear\_leaves**(*system=""*, *direction=""*, *flex\_sys\_dir=True*, *reset\_lists=True*)

clear/reset the user/system values of all the leaves of this *Values/Records* instance.

**Parameters**

- **system** (str) – system id.
- **direction** (str) – direction id (FAD\_\* constants).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **reset\_lists** (bool) – pass False to prevent the reset of all the underlying lists.

**Return type**

Union[*Values*, *Records*]

**Returns**

**class Record**(*template=None*, *fields=None*, *system=""*, *direction=""*, *action=""*, *root\_rec=None*, *root\_idx=()*, *field\_items=False*)

Bases: *OrderedDict*

instances of this mapping class are used to represent record-like data structures.

isinstance(..., dict) not working if using MutableMapping instead of OrderedDict as super class. and dict cannot be used as super class because instance as kwarg will then not work: see the Groxx's answer in stackoverflow question at <https://stackoverflow.com/questions/3387691/how-to-perfectly-override-a-dict/47361653#47361653>.

**\_\_init\_\_**(*template=None*, *fields=None*, *system=""*, *direction=""*, *action=""*, *root\_rec=None*, *root\_idx=()*, *field\_items=False*)

create new Record instance, which is an ordered collection of *\_Field* items.

**Parameters**

- **template** (Optional[*Records*]) – pass Records instance to use first item/[0] as template (after deep copy / values cleared).

- **fields** (Optional[Iterable]) – OrderedDict/dict of `_Field` instances (field order is not preserved when using dict) or Record instance (fields will be referenced, not copied!) or list of (field\_name, fld\_or\_val) tuples.
- **system** (str) – main/current system of this record,
- **direction** (str) – interface direction of this record.
- **action** (str) – current action (see ACTION\_INSERT, ACTION\_SEARCH, ACTION\_DELETE, ...)
- **root\_rec** (Optional[Record]) – root record of this record (def=self will be a root record).
- **root\_idx** (Optional[Tuple[Union[int, str], ...]]) – root index of this record (def=()).
- **field\_items** (bool) – pass True to get Record items - using `__getitem__()` - as of type `_Field` (not as `val()`).

**\_\_repr\_\_()**

Return repr(self).

**Return type**

str

**\_\_str\_\_()**

Return str(self).

**Return type**

str

**\_\_contains\_\_(idx\_path)**

True if the dictionary has the specified key, else False.

**Return type**

bool

**\_\_getitem\_\_(key)**

`x.__getitem__(y) <==> x[y]`

**Return type**

Any

**\_\_setitem\_\_(key, value)**

Set self[key] to value.

**node\_child(idx\_path, use\_curr\_idx=None, moan=False, selected\_sys\_dir=None)**

get the node child specified by `idx_path` relative to this `Record` instance.

**Parameters**

- **idx\_path** (Union[int, str, Tuple[Union[int, str], ...]]) – index path or field name index string.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.
- **moan** (bool) – flag to check data integrity; pass True to raise AssertionError if not.
- **selected\_sys\_dir** (Optional[dict]) – optional dict to return the currently selected system/direction.

**Return type**`Union[Value, Values, Record, Records, _Field, None]`**Returns**

found node instance or None if not found.

**set\_node\_child**(*fld\_or\_val*, \**idx\_path*, *system=None*, *direction=None*, *protect=False*, *root\_rec=None*, *root\_idx=()*, *use\_curr\_idx=None*, *to\_value\_type=False*)

set/replace the child of the node specified by *idx\_path* with the value of *fld\_or\_val*.**Parameters**

- **fld\_or\_val** (Any) – field or value - to be set.
- **idx\_path** (Union[int, str]) – index path of the node child to set/replace.
- **system** (Optional[str]) – system id (pass None to use default system id of this *Record* instance).
- **direction** (Optional[str]) – direction id (pass None to use default direction id of this *Record* instance).
- **protect** (bool) – pass True to prevent the replacement of a node child.
- **root\_rec** (Optional[Record]) – root record of this data structure.
- **root\_idx** (Optional[Tuple[Union[int, str], ...]]) – root index path of this node.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.
- **to\_value\_type** (bool) – pass True ensure conversion of *fld\_or\_val* to a Value instance.

**Return type**`Record`**Returns**

self (this Record instance).

**value**(\**idx\_path*, *system=None*, *direction=None*, \*\**kwargs*)

search the Value specified by *idx\_path* and return it if found.**Parameters**

- **idx\_path** (Union[int, str]) – index path of Value.
- **system** (Optional[str]) – system id (pass None to use default system id of this *Record* instance).
- **direction** (Optional[str]) – direction id (pass None to use default direction id of this *Record* instance).
- **kwargs** – additional keyword args (to be passed onto underlying data structure).

**Return type**`Union[Value, Values, Record, Records, None]`**Returns**

found Value instance or None if not found.

**set\_value**(*value*, \**idx\_path*, *system=None*, *direction=None*, *protect=False*, *root\_rec=None*, *root\_idx=()*, *use\_curr\_idx=None*)

set/replace the Value instance of the node specified by *idx\_path*.**Parameters**

- **value** (Union[Value, Values, Record, Records]) – Value instance to be set/replaced.
- **idx\_path** (Union[int, str]) – index path of the Value to be set.
- **system** (Optional[str]) – system id (pass None to use default system id of this Record instance).
- **direction** (Optional[str]) – direction id (pass None to use default direction id of this Record instance).
- **protect** (bool) – pass True to protect existing node value from to be changed/replaced.
- **root\_rec** (Optional[Record]) – root Record instance of this data structure.
- **root\_idx** (Tuple[Union[int, str], ...]) – root index to this node/Record instance.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.

**Return type***Record***Returns**

self (this Record instance).

**val**(\*idx\_path, system=None, direction=None, flex\_sys\_dir=True, use\_curr\_idx=None, \*\*kwargs)

determine the user/system value referenced by *idx\_path* of this *Record* instance.

**Parameters**

- **idx\_path** (Union[int, str]) – index path items.
- **system** (Optional[str]) – system id (pass None to use default system id of this Record instance).
- **direction** (Optional[str]) – direction id (pass None to use default direction id of this Record instance).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.
- **kwargs** – extra args (will be passed to underlying data structure).

**Return type***Any***Returns**

found user/system value, or None if not found or empty string if value was not set yet.

**set\_val**(val, \*idx\_path, system=None, direction=None, flex\_sys\_dir=True, protect=False, extend=True, converter=None, root\_rec=None, root\_idx=(), use\_curr\_idx=None, to\_value\_type=False)

set the user/system value referenced by *idx\_path* of this *Record* instance.

**Parameters**

- **val** (Any) – user/system value to be set/replaced.
- **idx\_path** (Union[int, str]) – index path of the Value to be set.
- **system** (Optional[str]) – system id (pass None to use default system id of this Record instance).
- **direction** (Optional[str]) – direction id (pass None to use default direction id of this Record instance).

- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **protect** (bool) – pass True to protect existing node value from to be changed/replaced.
- **extend** (bool) – pass False to prevent extension of data structure.
- **converter** (Optional[Callable[[*\_Field*, Any], Any]]) – converter callable to convert user values between systems.
- **root\_rec** (Optional[Record]) – root Record instance of this data structure.
- **root\_idx** (Optional[Tuple[Union[int, str], ...]]) – root index to this node/Record instance.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.
- **to\_value\_type** (bool) – pass True ensure conversion of *val* to a Value instance.

**Return type***Record***Returns**

self (this Record instance).

**\_add\_field**(*field*, *idx*="")add *\_Field* instance to this Record instance.**Parameters**

- **field** (*\_Field*) – *\_Field* instance to add.
- **idx** (str) – name/key/idx string to map and identify this field (mostly identical to *field.name()*).

**Return type***Record***Returns**

self.

**add\_fields**(*fields*, *root\_rec*=None, *root\_idx*=())

adding fields to this Record instance.

**Parameters**

- **fields** (Iterable) – either a dict, a Record or a list with (key/field\_name, val/*\_Field*) tuples. key strings containing digits/numbers are interpreted as name/idx paths (then also the specified sub-Records/-Fields will be created). values can be either *\_Field* instances or field values.
- **root\_rec** (Optional[Record]) – root record of this record (def=self will be a root record).
- **root\_idx** (Tuple[Union[int, str], ...]) – root index of this record (def=()).

**Return type***Record***Returns**

self

**add\_system\_fields**(*system\_fields*, *sys fld\_indexes=None*, *system=None*, *direction=None*, *extend=True*)

add/set fields to this *Record* instance from the field definition passed into *system\_fields*.

make sure before you call this method that this *Record* instance has the system and direction attributes specified/set.

#### Parameters

- **system\_fields** (Iterable[Iterable[Any]]) – tuple/list of tuples/lists with system and main field names and optional field aspects. the index of the field names and aspects within the inner tuples/lists get specified by *sys fld\_indexes*.
- **sys fld\_indexes** (Optional[Dict]) – mapping/map-item-indexes of the inner tuples of *system\_fields*. keys are field aspect types (FAT\_\* constants), optionally extended with a direction (FAD\_\* constant) and a system (SDI\_\* constant). if the value aspect key (FAT\_VAL) contains a callable then it will be set as the calculator (FAT\_CAL) aspect; if contains a field value then also the *clear\_val* of this field will be set to the specified value.
- **system** (Optional[str]) – system of the fields to be added - if not specified *system* will be used.
- **direction** (Optional[str]) – direction (FAD constants) of the fields to be added - if not passed used *self.direction*.
- **extend** (bool) – True=add not existing fields, False=apply new system aspects only on existing fields.

#### Return type

*Record*

#### Returns

*self*

**collect\_system\_fields**(*sys fld\_name\_path*, *path\_sep*)

compile list of system *\_Field* instances of this *Record* instance.

#### Parameters

- **sys fld\_name\_path** (Sequence) – sequence/tuple/list of system field names/keys.
- **path\_sep** (str) – system field name/key separator character(s).

#### Return type

List[\_Field]

#### Returns

list of :class:`\_Field` instances which are having system field name/keys set.

**compare\_leaves**(*rec*, *field\_names=()*, *exclude\_fields=()*)

compare the leaf 'compare' values of this *Record* instance with the one passed into *rec*.

'Compare values' are simplified user/system values generated by *Record.compare\_val()*.

#### Parameters

- **rec** (*Record*) – other *Record* instance to compare to.
- **field\_names** (Iterable) – field names to include in compare; pass empty tuple (==default) to include all fields.
- **exclude\_fields** (Iterable) – field names to exclude from compare.

**Return type**`List[str]`**Returns**list of str with the differences between self and *rec*.**compare\_val**(\**idx\_path*)determine normalized/simplified user/system value of the node specified by *idx\_path*.**Parameters****idx\_path** (Union[int, str]) – index path to the node.**Return type**`Any`**Returns**

normalized/simplified compare value.

**copy**(*deepness*=0, *root\_rec*=None, *root\_idx*=(), *onto\_rec*=None, *filter\_fields*=None, *fields\_patches*=None)

copy the fields of this record.

**Parameters**

- **deepness** (int) – deep copy level: <0==see deeper(), 0==only copy this record instance, >0==deep copy to deepness value - `_Field` occupies two deepness: 1st=`_Field`, 2nd=`Value`.
- **root\_rec** (Optional[Record]) – destination root record - using *onto\_rec*/new record if not specified.
- **root\_idx** (Tuple[Union[int, str], ...]) – destination root index (tuple/list with index path items: field names, list indexes).
- **onto\_rec** (Optional[Record]) – destination record; pass None to create new Record instance.
- **filter\_fields** (Optional[Callable[[Field], Any]]) – method called for each copied field (return True to filter/hide/not-include into copy).
- **fields\_patches** (Optional[Dict[str, Dict[str, Union[str, Value, Values, Record, Records, Callable[[Field], Any]]]]) – dict[field\_name\_or\_ALL\_FIELDS:dict[aspect\_key:val\_or\_callable]] to set/overwrite aspect values in each copied `_Field` instance. the keys of the outer dict are either field names or the `ALL_FIELDS` value; aspect keys ending with the `CALLABLE_SUFFIX` have a callable in the dict item that will be called for each field with the field instance as argument; the return value of the callable will then be used as the (new) aspect value. set the aspect value that stores the field value (aspect key == `FAT_VAL`) by passing a data structure instance (of type `ValueType`).

**Return type**`Record`**Returns**

new/extended record instance.

**clear\_leaves**(*system*="", *direction*="", *flex\_sys\_dir*=True, *reset\_lists*=True)clear the leaf values including this `Record` instance and all deeper data structures.**Parameters**

- **system** (str) – system id (pass None to use default system id of this `Record` instance).
- **direction** (str) – direction id (pass None to use default direction id of this `Record` instance).



- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **reset\_lists** (bool) – pass False to prevent reset of sub-lists.

**Return type***Record***Returns**

**leaves**(*system=None, direction=None, flex\_sys\_dir=True*)  
generate leaves/\_Field-instances of this *Record* instance.

**Parameters**

- **system** (Optional[str]) – system id (pass None to use default system id of this *Record* instance).
- **direction** (Optional[str]) – direction id (pass None to use default direction id of this *Record* instance).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.

**Return type***Generator[\_Field, None, None]***Returns**

leaf/\_Field-instance generator.

**leaf\_indexes**(*\*idx\_path, system=None, direction=None, flex\_sys\_dir=True*)  
generate leaf/\_Field-index paths for all fields of this *Record* instance.

**Parameters**

- **idx\_path** (Union[int, str]) – index path to be added as base index path (index path to this *Record* instance).
- **system** (Optional[str]) – system id (pass None to use default system id of this *Record* instance).
- **direction** (Optional[str]) – direction id (pass None to use default direction id of this *Record* instance).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.

**Return type***Generator[Tuple[Union[int, str], ...], None, None]***Returns**

leaf/\_Field-instance generator.

**leaf\_names**(*system="", direction="", col\_names=(), field\_names=(), exclude\_fields=(), name\_type=None*)  
compile a tuple of name types (specified by *name\_type*) for this *Record* instance.

**Parameters**

- **system** (str) – system id (pass None to use default system id of this *Record* instance).
- **direction** (str) – direction id (pass None to use default direction id of this *Record* instance).
- **col\_names** (Iterable) – system field/column names to include; pass empty tuple (==default) to include all.
- **field\_names** (Iterable) – field names to include; pass empty tuple (==default) to include all fields.

- **exclude\_fields** (Iterable) – field names to exclude.
- **name\_type** (Optional[str]) – type of name to be included/returned - see available name types underneath.

**Return type**

Tuple[Union[int, str, Tuple[Union[int, str], ...]], ...]

**Returns**

tuple of field names/indexes of the included/found leaves.

possible values for the *name\_type* argument are:

- 's': user/system field/column name.
- 'f': field name.
- 'r': root name (index path converted into string by *idx\_path\_field\_name()*).
- 'S': index path with user/system field/column name of leaf.
- 'F': index path tuple.
- **None: names depends on each leaf:**
  - root name if leaf is not a root field.
  - user/system name if *system* is not empty/None.
  - field name.

**merge\_leaves**(*rec*, *system=""*, *direction=""*, *flex\_sys\_dir=True*, *extend=True*)

merge the leaves of the other record in *rec* into this *Record* instance.

**Parameters**

- **rec** (*Record*) – other *Record* to merge into this one.
- **system** (str) – system id (pass None to use default system id of this *Record* instance).
- **direction** (str) – direction id (pass None to use default direction id of this *Record* instance).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **extend** (bool) – pass False to prevent extension of this data structure.

**Return type**

*Record*

**Returns**

self (this *Record* instance).

**match\_key**(*match\_fields*)

make tuple of user/system values for all the fields in *match\_fields*.

**Parameters**

**match\_fields** (Iterable) – Iterable with field names/index paths.

**Return type**

tuple

**Returns**

tuple of user/system values.

**merge\_values**(*rec*, *system*="", *direction*="", *flex\_sys\_dir*=True, *extend*=True)

merge user/system values of the other record in *rec* into this *Record* instance.

#### Parameters

- **rec** (Record) – other *Record* to merge into this one.
- **system** (str) – system id (pass None to use default system id of this *Record* instance).
- **direction** (str) – direction id (pass None to use default direction id of this *Record* instance).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **extend** (bool) – pass False to prevent extension of this data structure.

#### Return type

*Record*

#### Returns

self (this *Record* instance).

**missing\_fields**(*required\_fields*=())

check field-names/index-paths specified in *required\_fields*.

#### Parameters

**required\_fields** (Iterable) – list/tuple/iterable of field names or index paths of required fields.

#### Return type

List[Tuple[Union[int, str], ...]]

#### Returns

list of index paths for the ones that are missing or having an empty/None value.

**pop**(*idx*, *default*=None)

check if field name exists and if yes then remove the *\_Field* instance from this *Record* instance.

#### Parameters

- **idx** (str) – field name.
- **default** (Optional[\_Field]) – return value if a field with the name specified by *idx* does not exist in this *Record*.

#### Return type

Optional[\_Field]

#### Returns

removed *\_Field* instance if found, else None.

this method got added because the *OrderedDict*.pop() method does not call *\_\_delitem\_\_*() (see also *\_\_contains\_\_*())

**pull**(*from\_system*)

pull all user/system values and convert them into field values.

#### Parameters

**from\_system** (str) – system id of the system to pull from.

#### Return type

*Record*

#### Returns

self.

**push**(*onto\_system*)

push field values of this *Record* instance to the related user/system values (converted).

**Parameters**

**onto\_system** (str) – system id of the system to push to.

**Return type**

*Record*

**Returns**

self.

**set\_current\_system\_index**(*sys\_fld\_name\_prefix*, *path\_sep*, *idx\_val=None*, *idx\_add=1*)

check and if possible set the current system index of this *Record* instance.

**Parameters**

- **sys\_fld\_name\_prefix** (str) – user/system field name prefix.
- **path\_sep** (str) – user/system name path separator.
- **idx\_val** (Optional[int]) – new current system index value to set (if passed then *idx\_add* will be ignored).
- **idx\_add** (Optional[int]) – increment current system index value with the passed int value.

**Return type**

Optional[*Record*]

**Returns**

self (this *Record* instance) if current index got changed, else None.

**set\_env**(*system=None*, *direction=None*, *action=None*, *root\_rec=None*, *root\_idx=()*)

set the environment (system/direction/action) of this *Record* instance.

**Parameters**

- **system** (Optional[str]) – system id (pass None to leave unchanged).
- **direction** (Optional[str]) – direction id (pass None to leave unchanged).
- **action** (Optional[str]) – action id (pass None to leave unchanged).
- **root\_rec** (Optional[*Record*]) – root *Record* instance of this data structure.
- **root\_idx** (Optional[Tuple[Union[int, str], ...]]) – root index to this node/*Record* instance.

**Return type**

*Record*

**Returns**

self.

**sql\_columns**(*from\_system*, *col\_names=()*)

return list of sql column names for given system.

**Parameters**

- **from\_system** (str) – system from which the data will be selected/fetched.
- **col\_names** (Iterable) – optionally restrict to select columns to names given in this list.

**Return type**

List[str]

**Returns**

list of sql column names.

**sql\_select**(*from\_system*, *col\_names*=())

return list of sql column names/expressions for given system.

**Parameters**

- **from\_system** (str) – system from which the data will be selected/fetched.
- **col\_names** (Iterable) – optionally restrict to select columns to names given in this list.

**Return type**

List[str]

**Returns**

list of sql column names/expressions.

**to\_dict**(*filter\_fields*=None, *key\_type*=<class 'str'>, *push\_onto*=True, *use\_system\_key*=True, *put\_system\_val*=True, *put\_empty\_val*=False, *system*=None, *direction*=None)

copy Record leaf values into a dict.

**Parameters**

- **filter\_fields** (Optional[Callable[[\_Field], Any]]) – callable returning True for each field that need to be excluded in returned dict, pass None to include all fields (if *put\_empty\_val* == True).
- **key\_type** (Union[Type[str], Type[tuple], None]) – type of dict keys: None=field name, tuple=index path tuple, str=index path string (def).
- **push\_onto** (bool) – pass False to prevent self.push(system).
- **use\_system\_key** (bool) – pass False to put leaf field name/index; def=True to use system field name/keys, specified by the system/direction args.
- **put\_system\_val** (bool) – pass False to include/use main field val; def=True to include system val specified by the system/direction args.
- **put\_empty\_val** (bool) – pass True to also include fields with an empty value (None/”).
- **system** (Optional[str]) – system id to determine included leaf and field val (if *put\_system\_val* == True).
- **direction** (Optional[str]) – direction id to determine included leaf and field val (if *put\_system\_val* == True).

**Return type**

Dict[Union[int, str, Tuple[Union[int, str], ...], Any]

**Returns**

dict of filtered leaf user/system values with field names/idx\_path-tuples as their key.

**update**(*mapping*=(), *\*\*kwargs*)

update this *Record* instance - overwriting/extending OrderedDict/super().update() to return self.

**Parameters**

- **mapping** – mapping to use to update this *Record* instance.
- **kwargs** – optional keyword arguments.

**Returns**

self (this *Record* instance).

**class** `Records(seq=())`

Bases: `Values`

Records class.

each instance of this `Records` class is a list of 0..n `Record` instances.

the not overwritten methods of the inherited `Values` class are also available - like e.g. `Values.node_child()` or `Values.val()`.

**\_\_init\_\_**(`seq=()`)

create new `Records` instance.

#### Parameters

**seq** (`Iterable`) – Iterable used to initialize the new instance (pass list, tuple or other iterable).

**\_\_getitem\_\_**(`key`)

`x.__getitem__(y) <==> x[y]`

#### Return type

`Union[Value, Values, Record, Records, List[Union[Value, Values, Record, Records]]]`

**\_\_setitem\_\_**(`key, value`)

Set `self[key]` to value.

**set\_node\_child**(`rec_or_fld_or_val, *idx_path, system="", direction="", protect=False, root_rec=None, root_idx=(), use_curr_idx=None`)

set/replace the child of the node specified by `idx_path` with the value of `rec_or_fld_or_val`.

#### Parameters

- **rec\_or\_fld\_or\_val** – record, field or value - to be set.
- **idx\_path** (`Union[int, str]`) – index path of the node child to set/replace.
- **system** (`str`) – system id (pass None to use default system id of this `Records` instance).
- **direction** (`str`) – direction id (pass None to use default direction id of this `Records` instance).
- **protect** (`bool`) – pass True to prevent the replacement of a node child.
- **root\_rec** (`Optional[Record]`) – root record of this data structure.
- **root\_idx** (`Optional[Tuple[Union[int, str], ...]]`) – root index path of this node.
- **use\_curr\_idx** (`Optional[list]`) – list of counters to specify if and which current indexes have to be used.

#### Return type

`Records`

#### Returns

self (this `Records` instance).

**set\_val**(`val, *idx_path, system="", flex_sys_dir=True, protect=False, extend=True, converter=None, root_rec=None, root_idx=(), use_curr_idx=None`)

set the user/system value referenced by `idx_path` of this `Records` instance.

#### Parameters

- **val** (`Any`) – user/system value to be set/replaced.

- **idx\_path** (Union[int, str]) – index path of the Value to be set.
- **system** (str) – system id (pass None to use default system id of this *Records* instance).
- **direction** (str) – direction id (pass None to use default direction id of this *Records* instance).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **protect** (bool) – pass True to protect existing node value from to be changed/replaced.
- **extend** (bool) – pass False to prevent extension of data structure.
- **converter** (Optional[Callable[[\_Field, Any], Any]]) – converter callable to convert user values between systems.
- **root\_rec** (Optional[Record]) – root Record instance of this data structure.
- **root\_idx** (Optional[Tuple[Union[int, str], ...]]) – root index to this node/Records instance.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.

**Return type***Records***Returns**self (this *Records* instance).**append\_record**(root\_rec, root\_idx=(), from\_rec=None, clear\_leaves=True)add/append Record to this *Records* instance.**Parameters**

- **root\_rec** (Record) – root Record instance of this data structure.
- **root\_idx** (Tuple[Union[int, str], ...]) – root index to this node/Records instance.
- **from\_rec** (Optional[Record]) – *Record* instance to append, if not passed then use a copy of the first/template Record of this *Records* instance, else create new *Record* instance.
- **clear\_leaves** (bool) – pass False to not clear the leaf values.

**Return type***Record***Returns**added/appended *Record* instance.**compare\_records**(records, match\_fields, field\_names=(), exclude\_fields=(), record\_comparator=None)compare the records of this instance with the ones passed in the *records* argument.**Parameters**

- **records** (*Records*) – other instance of *Records* to be compared against self.
- **match\_fields** (Iterable) – iterable with field names/index paths to determine each *Record* id/pkey.
- **field\_names** (Iterable) – iterable with field names/index paths that get compared.
- **exclude\_fields** (Iterable) – iterable with field names/index-paths that get excluded from to be compared.

- **record\_comparator** (Optional[Callable[[Record, Record], List[str]]]) – optional callable for additional compare (called for each Record).

**Return type**`List[str]`**Returns**

list of differences.

**index\_match\_fields**(*match\_fields*)create/initialize match index for this *Records* instance (stored in *match\_index* attribute).**Parameters**

- **match\_fields** (Iterable) – iterable with field names/index paths to determine each *Record* id/pkey.

**Return type***Records***Returns**self (this *Records* instance).**leaves**(*system*="", *direction*="", *flex\_sys\_dir*=True)generate leaves/\_Field-instances of this *Records* instance.**Parameters**

- **system** (str) – system id (pass None to use default system id of the underlying *Record* instances).
- **direction** (str) – direction id (pass None to use default direction of the underlying *Record* instances).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.

**Return type**`Generator[_Field, None, None]`**Returns**

leaf/\_Field-instance generator.

**leaf\_indexes**(*\*idx\_path*, *system*="", *direction*="", *flex\_sys\_dir*=True)generate leaf/\_Field-index paths for all fields of this *Records* instance.**Parameters**

- **idx\_path** (Union[int, str]) – index path to be added as base index path (index path to this *Records* instance).
- **system** (str) – system id (pass None to use default system id of the underlying *Record* instances).
- **direction** (str) – direction id (pass None to use default direction of the underlying *Record* instances).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.

**Return type**`Generator[Tuple[Union[int, str], ...], None, None]`**Returns**

leaf/\_Field-instance generator.



**merge\_records**(records, match\_fields=())

merge the records passed in *records* into this *Records* instance.

**Parameters**

- **records** (Records) – records to be merged in.
- **match\_fields** (Iterable) – match fields used to identify already existing records (merge values in these cases).

**Return type**

Records

**Returns**

self.

**set\_env**(system="", direction="", root\_rec=None, root\_idx=())

set the environment (system/direction/action) of each record underneath this *Records* instance.

**Parameters**

- **system** (Optional[str]) – system id (pass None to leave unchanged).
- **direction** (Optional[str]) – direction id (pass None to leave unchanged).
- **root\_rec** (Optional[Record]) – root Record instance of this data structure.
- **root\_idx** (Optional[Tuple[Union[int, str], ...]]) – root index to this node/Record instance.

**Return type**

Records

**Returns**

self.

**class \_Field**(root\_rec=None, root\_idx=(), allow\_values=False, \*\*aspects)

Bases: object

Internal/Private class used by *Record* to create record field instances.

an instance of *\_Field* is representing one record field. the field properties are internally stored within a private dict (*\_Field.\_aspects*) and are called the ‘aspects’ of a field.

field aspects get used by a field instance e.g. to: \* store field value(s) \* define callable(s) to convert, filter or validate field values \* associate the root record and root index \* store any other user-defined field properties (like sql column expressions, comments, ...)

the FAT\_ALL constant contains all pre-defined aspects (see the other FAT\_\* constants defined at the top of this module). these values are called ‘aspect keys’ and are used as dict keys in the private dict.

each aspect can additionally have a separate property for each system/direction - in this case the aspect key gets extended with direction/system ids. the two available direction ids are pre-defined by the constants *FAD\_FROM* and *FAD\_ONTO*. the system ids are not defined in this module, they have to be defined by the application. aspect keys can be compiled with the function *aspect\_key()*.

**\_\_init\_\_**(root\_rec=None, root\_idx=(), allow\_values=False, \*\*aspects)

**\_\_repr\_\_**()

Return repr(self).

**\_\_str\_\_**()

Return str(self).

`__getitem__(key)`

`node_child(idx_path, use_curr_idx=None, moan=False, selected_sys_dir=None)`

get the node child specified by `idx_path` relative to this `_Field` instance.

#### Parameters

- `idx_path` (Union[int, str, Tuple[Union[int, str], ...]]) – index path or field name index string.
- `use_curr_idx` (Optional[list]) – list of counters to specify if and which current indexes have to be used.
- `moan` (bool) – flag to check data integrity; pass True to raise AssertionError if not.
- `selected_sys_dir` (Optional[dict]) – optional dict to return the currently selected system/direction.

#### Return type

Union[Value, Values, Record, Records, \_Field, None]

#### Returns

found node instance or None if not found.

`value(*idx_path, system="", direction="", flex_sys_dir=False)`

search the Value specified by `idx_path` and return it if found.

#### Parameters

- `idx_path` (Union[int, str]) – index path of Value.
- `system` (str) – system id (" stands for the main/system-independent value).
- `direction` (str) – direction id (" stands for the main/system-independent value).
- `flex_sys_dir` (bool) – pass True to allow fallback to main (non-user/-system) value.

#### Return type

Union[Value, Values, Record, Records, None]

#### Returns

found Value instance of None if not found.

`set_value(value, *idx_path, system="", direction="", protect=False, root_rec=None, root_idx=(), use_curr_idx=None)`

set/replace the Value instance of the node specified by `idx_path`.

#### Parameters

- `value` (Union[Value, Values, Record, Records]) – Value instance to be set/replaced.
- `idx_path` (Union[int, str]) – index path of the Value to be set.
- `system` (str) – system id (" stands for the main/system-independent value).
- `direction` (str) – direction id (" stands for the main/system-independent value).
- `protect` (bool) – pass True to protect existing node value from to be changed/replaced.
- `root_rec` (Optional[Record]) – root Record instance of this data structure.
- `root_idx` (Tuple[Union[int, str], ...]) – root index to this node/Record instance.
- `use_curr_idx` (Optional[list]) – list of counters to specify if and which current indexes have to be used.

**Return type**`_Field`**Returns**self (this `_Field` instance).**val**(*\*idx\_path*, *system=""*, *direction=""*, *flex\_sys\_dir=True*, *use\_curr\_idx=None*, *\*\*kwargs*)determine the user/system value referenced by *idx\_path*.**Parameters**

- **idx\_path** (Union[int, str]) – index path items.
- **system** (str) – system id ("" stands for the main/system-independent value).
- **direction** (str) – direction id ("" stands for the main/system-independent value).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.
- **kwargs** – extra args (will be passed to underlying data structure).

**Return type**`Any`**Returns**

found user/system value, or None if not found or empty string if value was not set yet.

**set\_val**(*val*, *\*idx\_path*, *system=""*, *direction=""*, *flex\_sys\_dir=True*, *protect=False*, *extend=True*, *converter=None*, *root\_rec=None*, *root\_idx=()*, *use\_curr\_idx=None*, *to\_value\_type=False*)set the user/system value referenced by *idx\_path*.**Parameters**

- **val** (Any) – user/system value to be set/replaced.
- **idx\_path** (Union[int, str]) – index path of the Value to be set.
- **system** (str) – system id ("" stands for the main/system-independent value).
- **direction** (str) – direction id ("" stands for the main/system-independent value).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **protect** (bool) – pass True to protect existing node value from to be changed/replaced.
- **extend** (bool) – pass False to prevent extension of data structure.
- **converter** (Optional[Callable[[\_Field, Any], Any]]) – converter callable to convert user values between systems.
- **root\_rec** (Optional[Record]) – root Record instance of this data structure.
- **root\_idx** (Tuple[Union[int, str], ...]) – root index to this node/Record instance.
- **use\_curr\_idx** (Optional[list]) – list of counters to specify if and which current indexes have to be used.
- **to\_value\_type** (bool) – pass True ensure conversion of *val* to a Value instance.

**Return type**`_Field`**Returns**self (this `_Field` instance).

**leaf\_value**(*system*="", *direction*="", *flex\_sys\_dir*=False)

determine the leaf value of this field (and optionally system/direction).

**Parameters**

- **system** (str) – system id (" stands for the main/system-independent value).
- **direction** (str) – direction id (" stands for the main/system-independent value).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.

**Return type**

Union[Value, Values, Record, Records, None]

**Returns**

the main or user/system value of this leaf/field or None if deeper value exists and *flex\_sys\_dir* is False.

used also to check if a deeper located sys field exists in the current data structure.

**leaves**(*system*="", *direction*="", *flex\_sys\_dir*=True)

generate all sub-leaves/\_Field-instances underneath this *\_Field* instance.

**Parameters**

- **system** (str) – system id (" stands for the main/system-independent value).
- **direction** (str) – direction id (" stands for the main/system-independent value).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.

**Return type**

Generator[\_Field, None, None]

**Returns**

leaf/\_Field-instance generator.

**leaf\_indexes**(\**idx\_path*, *system*="", *direction*="", *flex\_sys\_dir*=True)

generate leaf/\_Field-index paths for all subfields underneath (if exist) or this *\_Field* instance.

**Parameters**

- **idx\_path** (Union[int, str]) – index path to this *\_Field* instance.
- **system** (str) – system id (" stands for the main/system-independent value).
- **direction** (str) – direction id (" stands for the main/system-independent value).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.

**Return type**

Generator[Tuple[Union[int, str], ...], None, None]

**Returns**

leaf/\_Field-instance generator.

**find\_aspect\_key**(\**aspect\_types*, *system*="", *direction*="")

search for the passed *aspect\_types* in this *\_Field* instance.

**Parameters**

- **aspect\_types** (str) – aspect types (dict key prefixes) to search for.
- **system** (Optional[str]) – system id (" stands for the main/system-independent value).
- **direction** (Optional[str]) – direction id (" stands for the main/system-independent value).

**Return type**`Optional[str]`**Returns**

the full aspect key of the first found aspect that is matching the passed aspect type and optionally also the passed system and direction ids.

the search will be done in the following order: \* all passed aspect types including the passed system/direction ids. \* all passed aspect types including the passed system and both directions (if direction id get not passed). \* all passed aspect types including the passed system and without direction id. \* all passed aspect types without system and direction ids.

**set\_env**(*system=""*, *direction=""*, *root\_rec=None*, *root\_idx=()*)

set the environment (system/direction/action) of each record underneath this `_Field` instance.

**Parameters**

- **system** (Optional[str]) – system id (don't pass or pass None to leave unchanged).
- **direction** (Optional[str]) – direction id (don't pass or pass None to leave unchanged).
- **root\_rec** (Optional[Record]) – root Record instance of this field, system and direction.
- **root\_idx** (Optional[Tuple[Union[int, str], ...]]) – root index to this node/\_Field instance.

**Return type**`_Field`**Returns**

self (this `_Field` instance).

**set\_system\_root\_rec\_idx**(*system=None*, *direction=None*, *root\_rec=None*, *root\_idx=()*)

set the root record and index of the data structure where this `_Field` instance is included.

**Parameters**

- **system** (Optional[str]) – system id (don't pass or pass None to leave unchanged).
- **direction** (Optional[str]) – direction id (don't pass or pass None to leave unchanged).
- **root\_rec** (Optional[Record]) – new root Record instance of this data structure (pass None to leave unchanged).
- **root\_idx** (Optional[Tuple[Union[int, str], ...]]) – root index to this node/Record instance (pass None to determine).

**Return type**`_Field`**Returns**

self (this `_Field` instance).

**aspect\_exists**(*\*aspect\_types*, *system=""*, *direction=""*, *flex\_sys\_dir=False*)

check if aspect exists and return full aspect key (including system/direction) if yes.

**Parameters**

- **aspect\_types** (str) – aspect types (dict key prefixes) to search for.
- **system** (str) – system id (def=" stands for the main/system-independent value).
- **direction** (str) – direction id (def=" stands for the main/system-independent value).

- **flex\_sys\_dir** (Optional[bool]) – pass False to prevent fallback to system-independent value.

**Return type**

Optional[str]

**Returns**

the full aspect key of the first found aspect that is matching the passed aspect type and optionally also the passed system and direction ids.

**Returns**

**aspect\_value**(\**aspect\_types*, *system*="", *direction*="", *flex\_sys\_dir*=False)

**Parameters**

- **aspect\_types** (str) – aspect types (dict key prefixes) to search for.
- **system** (str) – system id (def="" stands for the main/system-independent value).
- **direction** (str) – direction id (def="" stands for the main/system-independent value).
- **flex\_sys\_dir** (Optional[bool]) – pass False to prevent fallback to system-independent value.

**Return type**

Any

**Returns**

the value of the first found aspect that is matching the passed aspect type and optionally also the passed system and direction ids or None if not found.

**del\_aspect**(*type\_or\_key*, *system*="", *direction*="")

remove aspect from this field.

**Parameters**

- **type\_or\_key** (str) – either FAT\_\* type or full key (including already the system and direction)-
- **system** (str) – system id string (if type\_or\_key is a pure FAT\_\* constant).
- **direction** (str) – direction string FAD\_\* constant (if type\_or\_key is a pure FAT\_\* constant).

**Return type**

Any

**Returns**

the aspect value of the removed aspect.

**set\_aspect**(*aspect\_value*, *type\_or\_key*, *system*="", *direction*="", *protect*=False, *allow\_values*=False)

set/change the value of an aspect identified by *type\_or\_key*, *system* and *direction*.

**Parameters**

- **aspect\_value** (Any) – the value to set on the aspect.
- **type\_or\_key** (str) – either FAT\_\* type or full key (including already the system and direction)-
- **system** (str) – system id string (if type\_or\_key is a pure FAT\_\* constant).
- **direction** (str) – direction string FAD\_\* constant (if type\_or\_key is a pure FAT\_\* constant).

- **protect** (bool) – pass True to prevent overwrite of already existing/set aspect value.
- **allow\_values** (bool) – pass True to allow change of field value aspect (*FAT\_VAL* aspect key).

**Return type***\_Field***Returns**self (this *\_Field* instance).**set\_aspects**(*allow\_values=False, \*\*aspects*)set multiple aspects provided in *aspects*.**Parameters**

- **allow\_values** (bool) – pass True to allow change of field value aspect (*FAT\_VAL* aspect key).
- **aspects** (Any) – dict of aspects where the dict key is the full/complete aspect key.

**Return type***\_Field***Returns**self (this *\_Field* instance).**add\_aspects**(*allow\_values=False, \*\*aspects*)add multiple aspects provided in *aspects*.**Parameters**

- **allow\_values** (bool) – pass True to allow change of field value aspect (*FAT\_VAL* aspect key).
- **aspects** (Any) – dict of aspects where the dict key is the full/complete aspect key.

**Return type***\_Field***Returns**self (this *\_Field* instance).**name**(*system="", direction="", flex\_sys\_dir=True*)

determine one of the names of this field.

**Parameters**

- **system** (str) – system id (def="" stands for the main/system-independent value).
- **direction** (str) – direction id (def="" stands for the main/system-independent value).
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.

**Return type**

str

**Returns**

main or system-specific name of this field.

**del\_name**(*system, direction=""*)

remove system-specific name from this field.

**Parameters**

- **system** (str) – system id (has to be non-empty system id).
- **direction** (str) – direction id (def=” stands for both directions).

**Return type***\_Field***Returns**self (this *\_Field* instance).**has\_name**(name, selected\_sys\_dir=None)check if this field has a name identical to the one passed in *name*.**Parameters**

- **name** (str) – name to search for.
- **selected\_sys\_dir** (Optional[dict]) – pass dict to get back the system and direction ids of the found name.

**Return type**

Optional[str]

**Returns**

full aspect key of the found name (including system/direction) or None if not found.

**set\_name**(name, system="", direction="", protect=False)

set/change one of the names of this field.

**Parameters**

- **name** (str) – the new name of this field.
- **system** (str) – system id (def=” stands for the main/system-independent value).
- **direction** (str) – direction id (def=” stands for the main/system-independent value).
- **protect** (bool) – pass True to prevent overwrite of already set/existing name.

**Return type***\_Field***Returns**self (this *\_Field* instance).**root\_rec**(system="", direction="")

determine and return root record of this field with given system and direction ids.

**Parameters**

- **system** (str) – system id (def=” stands for the main/system-independent value).
- **direction** (str) – direction id (def=” stands for the main/system-independent value).

**Return type**

Optional[Record]

**Returns**

root record instance or None if not set.

**set\_root\_rec**(rec, system="", direction="")

set/change the root record of this field, system and direction.

**Parameters**

- **rec** (Record) – root record instance.



- **system** (str) – system id (def=" stands for the main/system-independent value).
- **direction** (str) – direction id (def=" stands for the main/system-independent value).

**Return type***\_Field***Returns**self (this *\_Field* instance).**root\_idx**(system="", direction="")return the root index of this field for the specified *system* and *direction* ids.**Parameters**

- **system** (str) – system id (def=" stands for the main/system-independent value).
- **direction** (str) – direction id (def=" stands for the main/system-independent value).

**Return type**

Tuple[Union[int, str], ...]

**Returns**

root index of this field.

**set\_root\_idx**(idx\_path, system="", direction="")

set/change the root record of this field, system and direction.

**Parameters**

- **idx\_path** (Tuple[Union[int, str], ...]) – root index for this field/system/direction.
- **system** (str) – system id (def=" stands for the main/system-independent value).
- **direction** (str) – direction id (def=" stands for the main/system-independent value).

**Return type***\_Field***Returns**self (this *\_Field* instance).**calculator**(system="", direction="")return the calculation callable for this field, *system* and *direction*.**Parameters**

- **system** (str) – system id (def=" stands for the main/system-independent value).
- **direction** (str) – direction id (def=" stands for the main/system-independent value).

**Return type**

Optional[Callable[[\_Field], Any]]

**Returns**

callable used to calculate the value of this field or None if not set/exists/found.

**set\_calculator**(calculator, system="", direction="", protect=False)

set/change the field value calculator of this field, system and direction.

**Parameters**

- **calculator** (Callable[[\_Field], Any]) – new callable used to calculate the value of this field.
- **system** (str) – system id (def=" stands for the main/system-independent value).

- **direction** (str) – direction id (def=" stands for the main/system-independent value).
- **protect** (bool) – pass True to prevent overwrite of already set/existing calculator callable.

**Return type***\_Field***Returns**self (this *\_Field* instance).**clear\_val**(system="", direction="")return the initial field value (the clear value) of this field, *system* and *direction*.**Parameters**

- **system** (str) – system id (def=" stands for the main/system-independent value).
- **direction** (str) – direction id (def=" stands for the main/system-independent value).

**Return type***Any***Returns**

the found clear/init value or None if not set/found.

**set\_clear\_val**(clr\_val, system="", direction="")set/change the clear/init value of this field, *system* and *direction*.**Parameters**

- **clr\_val** – new clear/init value of this field.
- **system** (str) – system id (def=" stands for the main/system-independent value).
- **direction** (str) – direction id (def=" stands for the main/system-independent value).

**Return type***\_Field***Returns**self (this *\_Field* instance).**\_ensure\_system\_value**(system, direction="", root\_rec=None, root\_idx=())check if a field value for the specified *system/direction* exists and if not then create it.**Parameters**

- **system** (str) – system id (def=" stands for the main/system-independent value).
- **direction** (str) – direction id (def=" stands for the main/system-independent value).
- **root\_rec** (Optional[Record]) – root Record instance of this field, *system* and *direction*.
- **root\_idx** (Tuple[Union[int, str], ...]) – root index to this node/*\_Field* instance.

**converter**(system, direction="")return the converter callable for this field, *system* and *direction*.**Parameters**

- **system** (str) – system id (def=" stands for the main/system-independent value).
- **direction** (str) – direction id (def=" stands for the main/system-independent value).

**Return type**`Optional[Callable[[_Field, Any], Any]]`**Returns**

callable used to convert the field value between systems or None if not set.

separate system-specific representations of the field value can be (automatically) converted by specifying a converter callable aspect.

**set\_converter**(*converter*, *system*, *direction*="", *protect*=True, *root\_rec*=None, *root\_idx*=())

set/change the field value converter of this field, system and direction.

**Parameters**

- **converter** `Callable[[_Field, Any], Any]` – new callable used to convert the value of this field between systems.
- **system** `(str)` – system id (def="" stands for the main/system-independent value).
- **direction** `(str)` – direction id (def="" stands for the main/system-independent value).
- **protect** `(bool)` – pass False to allow the overwriting of an already set converter callable.
- **root\_rec** `(Optional[Record])` – root Record instance of this field, system and direction.
- **root\_idx** `(Tuple[Union[int, str], ...])` – root index to this node/*\_Field* instance.

**Return type**`_Field`**Returns**

self (this *\_Field* instance).

**convert**(*val*, *system*, *direction*)

convert field value from/onto system.

**Parameters**

- **val** `(Any)` – field value to convert.
- **system** `(str)` – system to convert from/onto.
- **direction** `(str)` – conversion direction (from or onto - see *FAD\_FROM* and *FAD\_ONTO*).

**Return type**`Any`**Returns**

converted field value.

**filterer**(*system*="", *direction*="")

return the filter callable for this field, *system* and *direction*.

**Parameters**

- **system** `(str)` – system id (def="" stands for the main/system-independent value).
- **direction** `(str)` – direction id (def="" stands for the main/system-independent value).

**Return type**`Optional[Callable[[_Field], Any]]`**Returns**

callable used to filter this field/parent-record or None if not set.

**set\_filterer**(*filterer*, *system*="", *direction*="", *protect*=False)

set/change the filterer callable of this field, *system* and *direction*.

**Parameters**

- **filterer** *¶* (Callable[[\_Field], Any]) – new callable used to filter this field or the parent record.
- **system** *¶* (str) – system id (def="" stands for the main/system-independent value).
- **direction** *¶* (str) – direction id (def="" stands for the main/system-independent value).
- **protect** *¶* (bool) – pass True to prevent overwrite of already set/existing filter callable.

**Return type**

*Field*

**Returns**

self (this *Field* instance).

**sql\_expression**(*system*="", *direction*="")

return the sql column expression for this field, *system* and *direction*.

**Parameters**

- **system** *¶* (str) – system id (def="" stands for the main/system-independent value).
- **direction** *¶* (str) – direction id (def="" stands for the main/system-independent value).

**Return type**

Optional[str]

**Returns**

sql column expression string if set or None if not set.

**set\_sql\_expression**(*sql\_expression*, *system*="", *direction*="", *protect*=False)

set/change sql column expression of this field, *system* and *direction*.

**Parameters**

- **sql\_expression** *¶* (str) – new sql column expression used to fetch associated db column of this field.
- **system** *¶* (str) – system id (def="" stands for the main/system-independent value).
- **direction** *¶* (str) – direction id (def="" stands for the main/system-independent value).
- **protect** *¶* (bool) – pass True to prevent overwrite of already set/existing filter callable.

**Return type**

*Field*

**Returns**

self (this *Field* instance).

**validator**(*system*="", *direction*="")

return the validation callable for this field, *system* and *direction*.

**Parameters**

- **system** *¶* (str) – system id (def="" stands for the main/system-independent value).
- **direction** *¶* (str) – direction id (def="" stands for the main/system-independent value).

**Return type**

Optional[Callable[[\_Field, Any], Any]]

**Returns**

validation callable if set or None if not set.

**set\_validator**(*validator*, *system*="", *direction*="", *protect*=False, *root\_rec*=None, *root\_idx*=())

set/change the field value validator of this field, system and direction.

**Parameters**

- **validator** (Callable[[\_Field, Any], Any]) – new callable used to validate the value of this field, systems and direction.
- **system** (str) – system id (def="" stands for the main/system-independent value).
- **direction** (str) – direction id (def="" stands for the main/system-independent value).
- **protect** (bool) – pass False to allow the overwriting of an already set converter callable.
- **root\_rec** (Optional[Record]) – root Record instance of this field, system and direction.
- **root\_idx** (Tuple[Union[int, str], ...]) – root index to this node/\_Field instance.

**Return type**

\_Field

**Returns**

self (this \_Field instance).

**validate**(*val*, *system*="", *direction*="")

validate field value for specified *system* and *direction*.

**Parameters**

- **val** (Any) – field value to validate (if ok to be set as new field value).
- **system** (str) – system id of new field value.
- **direction** (str) – direction id of new field value.

**Return type**

bool

**Returns**

True if *val* is ok to be set as new field value else False.

**append\_record**(*system*="", *direction*="", *flex\_sys\_dir*=True, *root\_rec*=None, *root\_idx*=())

append new record to the *Records* value of this field/system/direction.

**Parameters**

- **system** (str) – system id of the field value to extend.
- **direction** (str) – direction id of the field value to extend.
- **flex\_sys\_dir** (bool) – pass False to prevent fallback to system-independent value.
- **root\_rec** (Optional[Record]) – root Record instance of this field, system and direction.
- **root\_idx** (Tuple[Union[int, str], ...]) – root index to this node/\_Field instance.

**Return type**

Record

**Returns**

added/appended Record instance.

**clear\_leaves**(*system=""*, *direction=""*, *flex\_sys\_dir=True*, *reset\_lists=True*)

clear/reset field values and if *reset\_lists == True* also Records/Values lists to one item.

#### Parameters

- **system** (str) – system of the field value to clear, pass None to clear all field values.
- **direction** (str) – direction of the field value to clear.
- **flex\_sys\_dir** (bool) – if True then also clear field value if system is given and field has no system value.
- **reset\_lists** (bool) – if True/def then also clear Records/lists to one item.

#### Return type

[\*\\_Field\*](#)

#### Returns

self (this [\*\\_Field\*](#) instance).

**copy**(*deepness=0*, *root\_rec=None*, *root\_idx=()*, *\*\*kwargs*)

copy the aspects (names, indexes, values, ...) of this field.

#### Parameters

- **deepness** – deep copy level: <0==see [deeper\(\)](#), 0==only copy current instance, >0==deep copy to deepness value - [\*\\_Field\*](#) occupies two deepness: 1st=[\*\\_Field\*](#), 2nd=Value.
- **root\_rec** (Optional[[\*Record\*](#)]) – destination root record.
- **root\_idx** (Tuple[Union[int, str], ...]) – destination index path (tuple of field names and/or list/Records/Values indexes).
- **kwargs** – additional arguments (will be passed on - most of them used by [\*Record\*](#).  
[\*copy\(\)\*](#)).

#### Return type

[\*\\_Field\*](#)

#### Returns

new/copied [\*\\_Field\*](#) instance.

**parent**(*system=""*, *direction=""*, *value\_types=None*)

determine one of the parent [\*ValueType\*](#) instances in this data structure above of this field.

#### Parameters

- **system** (str) – system id (def="" stands for the main/system-independent value).
- **direction** (str) – direction id (def="" stands for the main/system-independent value).
- **value\_types** (Optional[Tuple[Type[Union[[\*Value\*](#), [\*Values\*](#), [\*Record\*](#), [\*Records\*](#)]], ...]]) – pass tuple of [\*ValueType\*](#) to restrict search to one of the passed types.

#### Return type

Union[[\*Value\*](#), [\*Values\*](#), [\*Record\*](#), [\*Records\*](#), None]

#### Returns

found parent instance or None if not set or if type is not of passed *value\_types*.

**pull**(*from\_system*, *root\_rec*, *root\_idx*)

pull the system-specific value (specified by *from\_system*) into the main value of this field.

#### Parameters

- **from\_system** (str) – system id of the system to pull from.
- **root\_rec** (Record) – root Record instance of this field, system and direction.
- **root\_idx** (Tuple[Union[int, str], ...]) – root index to this node/\_Field instance.

**Return type**

\_Field

**Returns**

self (this \_Field instance).

**push**(onto\_system, root\_rec, root\_idx)

push the main value of this field onto the system-specific value (specified by onto\_system).

**Parameters**

- **onto\_system** (str) – system id of the system to pull from.
- **root\_rec** (Record) – root Record instance of this field, system and direction.
- **root\_idx** (Tuple[Union[int, str], ...]) – root index to this node/\_Field instance.

**Return type**

\_Field

**Returns**

self (this \_Field instance).

**string\_to\_records**(str\_val, field\_names, rec\_sep=',', fld\_sep=' ', system="", direction="")

convert formatted string into a Records instance containing several Record instances.

**Parameters**

- **str\_val** (str) – formatted string to convert.
- **field\_names** (Sequence) – list/tuple of field names of each record
- **rec\_sep** (str) – character(s) used in str\_val to separate the records.
- **fld\_sep** (str) – character(s) used in str\_val to separate the field values of each record.
- **system** (str) – system id (def="" stands for the main/system-independent value).
- **direction** (str) – direction id (def="" stands for the main/system-independent value).

**Return type**

Records

**Returns**

converted Records instance.

**record\_field\_val**(\*idx\_path, system="", direction="")

get/determine the value of any field specified via idx\_path within this data structure.

**Parameters**

- **idx\_path** (Union[int, str]) – index path of the field.
- **system** (str) – system id (def="" stands for the main/system-independent value).
- **direction** (str) – direction id (def="" stands for the main/system-independent value).

**Return type**

Any

**Returns**

the field value if found else None.

this method has an alias named `rfv()`.

**rfv**(\*idx\_path, system="", direction="")

alias of method `record_field_val()`

**Return type**

`Any`

**system\_record\_val**(\*idx\_path, system="", direction="", use\_curr\_idx=None)

get/determine the current value of a/this field within this data structure.

**Parameters**

- **idx\_path** `(Union[int, str])` – index path of the field.
- **system** `(str)` – system id (def=" stands for the main/system-independent value).
- **direction** `(str)` – direction id (def=" stands for the main/system-independent value).
- **use\_curr\_idx** `(Optional[list])` – list of counters to specify if and which current indexes have to be used.

**Return type**

`Any`

**Returns**

the currently selected field value if found else None.

this method has an alias named `srv()`.

**srv**(\*idx\_path, system="", direction="", use\_curr\_idx=None)

alias of method `system_record_val()`

**Return type**

`Any`

**in\_actions**(\*actions, system="", direction="")

determine if current data structure is in one of the passed *actions*.

**Parameters**

- **actions** `(str)` – tuple of actions (see ACTION\_ constants).
- **system** `(str)` – system id (def=" stands for the main/system-independent value).
- **direction** `(str)` – direction id (def=" stands for the main/system-independent value).

**Return type**

`bool`

**Returns**

True if the data structure has set one of the passed *actions* else False.

this method has an alias named `ina()`.

**ina**(\*actions, system="", direction="")

alias of method `in_actions()`

**Return type**

`bool`



**current\_records\_idx**(*system*="", *direction*="")

determine current index of *Records* instance, situated above of this field in this data structure.

#### Parameters

- **system** (str) – system id (def="" stands for the main/system-independent value).
- **direction** (str) – direction id (def="" stands for the main/system-independent value).

#### Return type

`Union[int, str, None]`

#### Returns

full index path or None if no current index exists above this field.

this method has an alias named *crx*().

**crx**(*system*="", *direction*="")

alias of method *current\_records\_idx*()

#### Return type

`Union[int, str, None]`

**VALUE\_TYPES** = (<class 'ae.sys\_data.Value'>, <class 'ae.sys\_data.Values'>, <class 'ae.sys\_data.Record'>, <class 'ae.sys\_data.Records'>)

tuple of classes/types used for system data values

**LIST\_TYPES** = (<class 'ae.sys\_data.Values'>, <class 'ae.sys\_data.Records'>)

tuple of classes/types used for system data lists

**NODE\_TYPES** = (<class 'ae.sys\_data.Record'>, <class 'ae.sys\_data.Records'>)

tuple of classes/types used for system data nodes

**NODE\_CHILD\_TYPES** = (<class 'ae.sys\_data.\_Field'>, <class 'ae.sys\_data.Record'>)

tuple of classes/types used for system data node children

**PARENT\_TYPES** = (<class 'ae.sys\_data.Values'>, <class 'ae.sys\_data.Record'>, <class 'ae.sys\_data.Records'>)

value types that can have children (Value excluded)

## 4.20 ae.sys\_core\_sh

### 4.20.1 SiHOT PMS system core xml interface

This portion is very old and needs refactoring and much more unit tests.

The classes provided by this portion are allowing the implementation of client and server components to communicate with the Sihot PMS system.

#### TODO:

- use other xml library because xml.etree the xml modules in the Python standard library are not secure against maliciously constructed data - the problem here is that the xml generated by the Sihot system is not 100% conform to the xml standards.
- refactor SihotXmlParser and inherited classes: migrating the attributes oc, tn, id, rc, hn, ... to a dict.
- inject cae app instance into \_SihotTcpClient, RequestXmlHandler and TcpServer (replacing ae.core.po()).

## Module Attributes

<i>SDI_SH</i>	Sihot System Interface Id
<i>SH_DEF_SEARCH_FIELD</i>	default search field for external systems (used by <code>sys_data_sh.cl_field_data()</code> )
<i>SDF_SH_SERVER_ADDRESS</i>	Sihot Server address or ip
<i>SDF_SH_KERNEL_PORT</i>	Sihot Kernel Interface port
<i>SDF_SH_WEB_PORT</i>	Sihot Web interfaces port
<i>SDF_SH_CLIENT_PORT</i>	Sihot Server client port
<i>SXML_DEF_ENCODING</i>	encoding used by the Sihot xml interface
<i>ERR_MESSAGE_PREFIX_CONTINUE</i>	error message prefix for ignorable errors
<i>TCP_CONNECTION_BROKEN_MSG</i>	error message fragment added if connection is broken

## Functions

<i>elem_to_attr</i> (elem)	convert element string to attribute string by converting into lower-case and replacing hyphens with underscores.
----------------------------	--

## Classes

<i>AvailCatInfo</i> (cae[, use_kernel])	build xml request and send it to get available room categories from the Sihot server.
<i>AvailCatInfoResponse</i> (cae)	processing response of CATINFO operation code of the WEB interface
<i>CatRoomResponse</i> (cae)	parser for Sihot room category responses.
<i>CatRooms</i> (cae[, use_kernel])	built room category request and send it to Sihot server.
<i>ConfigDict</i> (cae[, use_kernel])	build and send request for the Sihot configuration settings.
<i>ConfigDictResponse</i> (cae)	parser to process Sihot configuration setting responses.
<i>PostMessage</i> (cae[, use_kernel])	build and send request to add a message into the Sihot system message/notification queue.
<i>Request</i> (cae)	xml parser for generic requests from SIHOT.
<i>RequestXmlHandler</i> (request, client_address, ...)	server component base class to receive xml data from the Sihot system.
<i>ResChange</i> (cae)	xml parser to process Sihot server reservation change notifications.
<i>ResKernelGet</i> (cae)	build and send generic request to the Sihot kernel interface.
<i>ResKernelResponse</i> (cae)	response to the RESERVATION-GET oc/request of the KERNEL interface.
<i>ResResponse</i> (cae)	xml parser for kernel or web interface responses.
<i>RoomChange</i> (cae)	xml parser to process Sihot server room change notifications.
<i>ShSysConnector</i> (system)	connector class for the Sihot system.
<i>SihotXmlBuilder</i> (cae[, use_kernel])	generic class to build and send Sihot xml requests.
<i>SihotXmlParser</i> (cae)	XMLParser interface used by client to parse the responses from the Sihot server.
<i>TcpServer</i> (ip, port, cls_xml_handler[, ...])	tcp server class to process xml sent by the Sihot client.

**SDI\_SH = 'Sh'**

Sihot System Interface Id

**SH\_DEF\_SEARCH\_FIELD = 'ShId'**

default search field for external systems (used by `sys_data_sh.cl_field_data()`)

**SDF\_SH\_SERVER\_ADDRESS = 'shServerIP'**

Sihot Server address or ip

**SDF\_SH\_KERNEL\_PORT = 'shServerKernelPort'**

Sihot Kernel Interface port

**SDF\_SH\_WEB\_PORT = 'shServerPort'**

Sihot Web interfaces port

**SDF\_SH\_CLIENT\_PORT = 'shClientPort'**

Sihot Server client port

**SXML\_DEF\_ENCODING = 'cp1252'**

encoding used by the Sihot xml interface

**ERR\_MESSAGE\_PREFIX\_CONTINUE = 'CONTINUE:'**

error message prefix for ignorable errors

**TCP\_CONNECTION\_BROKEN\_MSG = 'socket connection broken!'**

error message fragment added if connection is broken

**\_TCP\_MAXBUFLen = 8192**

tcp buffer length

**\_TCP\_END\_OF\_MSG\_CHAR = b'\x04'**

end-of-message character of the Sihot xml interface

**\_DEBUG\_RUNNING\_CHARS = '|/-\\'**

progress animation characters for console output

**elem\_to\_attr(*elem*)**

convert element string to attribute string by converting into lower-case and replacing hyphens with underscores.

#### Parameters

**elem** *(str)* – element string.

#### Return type

*str*

#### Returns

attribute string of the passed element string.

**class \_SihotTcpClient(*server\_ip, server\_port, timeout=3.6, encoding='utf-8', debug\_level=0*)**

Bases: *object*

local sihot tcp client used by *SihotXmlBuilder.send\_to\_server()*.

**error\_message = ''**

**received\_xml = ''**

**\_\_init\_\_(*server\_ip, server\_port, timeout=3.6, encoding='utf-8', debug\_level=0*)**

**send\_to\_server**(*xml*)

send passed xml string to the Sihot server.

**Parameters**

**xml** *(str)* – xml string to send.

**Return type**

*str*

**Returns**

error message or empty string if no errors occurred.

**\_receive\_response**(*sock*)

receive response from Sihot server.

**Parameters**

**sock** – used socket for the connection to the Sihot server.

**Return type**

*str*

**Returns**

received string.

**class RequestXmlHandler**(*request, client\_address, server*)

Bases: *BaseRequestHandler*

server component base class to receive xml data from the Sihot system.

**error\_message** = ''

**notify**()

print error message to console output.

**handle**()

receive xml string sent by the Sihot system to this server component.

**abstract handle\_xml**(*xml\_from\_client*)

abstract method to be implemented by the inheriting class.

**Parameters**

**xml\_from\_client** *(bytes)* – xml request sent from Sihot xml client as a bytes string.

**Return type**

*bytes*

**Returns**

xml response to the Sihot xml client as bytes string.

**class \_ThreadedServer**(*server\_address, RequestHandlerClass, bind\_and\_activate=True*)

Bases: *ThreadingMixIn, TCPServer*

local tcp server threading mixin class used by *TcpServer* class.

**class TcpServer**(*ip, port, cls\_xml\_handler, debug\_level=0*)

Bases: *object*

tcp server class to process xml sent by the Sihot client.

**\_\_init\_\_**(*ip, port, cls\_xml\_handler, debug\_level=0*)

**run**(*display\_animation=False*)

run/start the server.

**Parameters**

**display\_animation** (bool) – pass True to display progress animation at console output.

**class** **SihotXmlParser**(*cae*)

Bases: **object**

XMLParser interface used by client to parse the responses from the Sihot server.

**\_\_init\_\_**(*cae*)

**parse\_xml**(*xml*)

parse the xml response string sent by the Sihot server.

**Parameters**

**xml** (str) – xml string to parse.

**get\_xml**()

get the xml string to be parsed.

**Return type**

str

**Returns**

xml string to be parsed.

**start**(*tag, \_attrib*)

parse next opening xml tag; called for each opening tag.

**Parameters**

- **tag** (str) – tag string.
- **\_attrib** (Dict[str, str]) – attribute string (not used in Sihot xml elements).

**Return type**

Optional[str]

**Returns**

None if the tag got parsed and recognized/processed else the unrecognized tag string.

**data**(*data*)

process parsed data string; called on each chunk (separated by XMLParser on spaces, special chars, ...).

**Parameters**

**data** (str) – data string chunk.

**Return type**

Optional[str]

**Returns**

None if data chunk string got parsed and recognized else the unprocessed data string.

**end**(*tag*)

parser detected end tag of element; called for each closing tag.

**Parameters**

**tag** (str) – closing element tag string.

**Return type**

Optional[str]

**Returns**

closing element tag string.

**close()**

end of xml string reached; called when all data has been parsed.

**Return type**

*SihotXmlParser*

**Returns**

this *SihotXmlParser* instance.

**server\_error()**

get the server error code string.

**Return type**

*str*

**Returns**

'0' if no error occurred, else the Sihot error return code as string.

**server\_err\_msg()**

get the server error message.

**Return type**

*str*

**Returns**

empty string if no error occurred, else the Sihot error message string.

**class Request(cae)**

Bases: *SihotXmlParser*

xml parser for generic requests from SIHOT.

**get\_operation\_code()**

return the Sihot operation code of the xml request string sent to the Sihot server.

**Return type**

*str*

**Returns**

Sihot operation code string.

**class RoomChange(cae)**

Bases: *SihotXmlParser*

xml parser to process Sihot server room change notifications.

**\_\_init\_\_(cae)**

**class ResChange(cae)**

Bases: *SihotXmlParser*

xml parser to process Sihot server reservation change notifications.

**\_\_init\_\_(cae)**

**start(tag, attrib)**

parser detected start tag of next/new xml element; called for each opening tag.

**Return type**

*None*

**data**(*data*)  
process parsed element data chunk.

**Return type**  
`None`

**class ResResponse**(*cae*)

Bases: `SihotXmlParser`

xml parser for kernel or web interface responses.

**\_\_init\_\_**(*cae*)

**class AvailCatInfoResponse**(*cae*)

Bases: `SihotXmlParser`

processing response of CATINFO operation code of the WEB interface

**\_\_init\_\_**(*cae*)

**data**(*data*)  
process parsed element data chunk.

**Return type**  
`Optional[str]`

**class CatRoomResponse**(*cae*)

Bases: `SihotXmlParser`

parser for Sihot room category responses.

**\_\_init\_\_**(*cae*)

**end**(*tag*)  
parser detected end tag of element.

**class ConfigDictResponse**(*cae*)

Bases: `SihotXmlParser`

parser to process Sihot configuration setting responses.

**\_\_init\_\_**(*cae*)

**end**(*tag*)  
parser detected end tag of Sihot config element.

**Return type**  
`Optional[str]`

**class ResKernelResponse**(*cae*)

Bases: `SihotXmlParser`

response to the RESERVATION-GET oc/request of the KERNEL interface.

**\_\_init\_\_**(*cae*)

**class SihotXmlBuilder**(*cae, use\_kernel=False*)

Bases: `object`

generic class to build and send Sihot xml requests.

**tn:** `str = '1'`

**\_\_init\_\_**(*cae, use\_kernel=False*)

create an instance of this class.

#### Parameters

- **cae** ( *ConsoleApp* ) – instance of the running *ConsoleApp* app.
- **use\_kernel** ( *bool* ) – pass True to use the Sihot kernel interface (False==use Sihot sxml interface).

**beg\_xml**(*operation\_code, add\_inner\_xml="", transaction\_number=""*)

create a new xml request string including xml header, operation code and transaction number.

#### Parameters

- **operation\_code** ( *str* ) – Sihot operation code of the new xml element.
- **add\_inner\_xml** ( *str* ) – inner xml block/elements.
- **transaction\_number** ( *str* ) – Sihot transaction number string.

**end\_xml**()

terminate a Sihot xml request string.

**add\_tag**(*tag, val=""*)

add a new xml element tag with the passed value.

**send\_to\_server**(*response\_parser=None*)

send the built xml request to the Sihot server.

#### Parameters

**response\_parser** ( *Optional[SihotXmlParser]* ) – used parser to parse the response from the Sihot server (def=SihotXmlParser).

#### Return type

*str*

#### Returns

error string or empty string if no errors occurred.

**static new\_tag**(*tag, val="", opening=True, closing=True*)

create new xml element with the passed tag and value.

#### Parameters

- **tag** ( *str* ) – tag of the new xml element.
- **val** ( *Any* ) – value of the new xml element.
- **opening** ( *bool* ) – pass False to NOT add the element opening tag.
- **closing** ( *bool* ) – pass False to NOT add the element closing tag.

#### Returns

new xml element or element part/fragment.

**static convert\_value\_to\_xml\_string**(*value*)

convert any element value type to the corresponding xml string, replacing & < > characters with escapes.

#### Parameters

**value** ( *Any* ) – element value to be converted.



**Return type**`str`**Returns**

element value as string.

**property xml:** `str`

property to determine the currently built xml string.

**Getter**

return built xml string.

**Setter**

change xml string.

**class AvailCatInfo**(*cae*, *use\_kernel=False*)Bases: `SihotXmlBuilder`

build xml request and send it to get available room categories from the Sihot server.

**avail\_rooms**(*hotel\_id*="", *room\_cat*="", *from\_date*=`datetime.date(2024, 3, 19)`, *to\_date*=`datetime.date(2024, 3, 19)`)

determine available rooms for the specified hotel, room category and date range.

**Parameters**

- **hotel\_id** `(str)` – Sihot hotel id or empty string to get available rooms of all hotels.
- **room\_cat** `(str)` – Sihot room category or empty string to get available rooms of all categories.
- **from\_date** `(date)` – start date of the date range: defaulting to today.
- **to\_date** `(date)` – end date of the date range; defaulting to today.

**Return type**`Union[dict, str]`**Returns**Sihot response as dict created by `AvailCatInfoResponse` with the results or an error message string if an error occurred.**response:** `Optional[SihotXmlParser]`**class CatRooms**(*cae*, *use\_kernel=False*)Bases: `SihotXmlBuilder`

built room category request and send it to Sihot server.

**get\_cat\_rooms**(*hotel\_id*='1', *from\_date*=`datetime.date(2024, 3, 19)`, *to\_date*=`datetime.date(2024, 3, 19)`, *scope*="")

determine configured room categories of the Sihot system for the specified hotel and date range.

**Parameters**

- **hotel\_id** `(str)` – Sihot hotel id to get the room categories.
- **from\_date** `(date)` – start date of the date range: defaulting to today.
- **to\_date** `(date)` – end date of the date range; defaulting to today.
- **scope** `(str)` – scope string to request additional information - see Sihot documentation.

**Return type**`Union[dict, str]`**Returns**

Sihot response as dict created by `CatRoomResponse` with the results or an error message string if an error occurred.

**response:** `Optional[SihotXmlParser]`

**class ConfigDict**(*cae, use\_kernel=False*)

Bases: `SihotXmlBuilder`

build and send request for the Sihot configuration settings.

**get\_key\_values**(*config\_type, hotel\_id='1', language='EN'*)

determine a configuration setting of the Sihot system for the specified hotel and language.

**Parameters**

- **config\_type** (str) – Sihot config setting type - see Sihot documentation.
- **hotel\_id** (str) – Sihot hotel id to get the configuration setting.
- **language** (str) – language id for the configuration description texts - see Sihot documentation.

**Return type**`Union[dict, str]`**Returns**

Sihot response as dict created by `ConfigDictResponse` with the results or an error message string if an error occurred.

**response:** `Optional[SihotXmlParser]`

**class PostMessage**(*cae, use\_kernel=False*)

Bases: `SihotXmlBuilder`

build and send request to add a message into the Sihot system message/notification queue.

**post\_message**(*msg, level=3, system='sys\_core\_sh\_module'*)

build and send request to add a message into the Sihot system message/notification queue.

**Parameters**

- **msg** (str) – message text string to add to the Sihot system messages.
- **level** (int) – importance level.
- **system** (str) – message sender system id/string.

**Return type**`str`**Returns**

error message or empty string if no error occurred.

**response:** `Optional[SihotXmlParser]`

**class ResKernelGet**(*cae*)

Bases: `SihotXmlBuilder`

build and send generic request to the Sihot kernel interface.

**\_\_init\_\_**(*cae*)

create an instance of this class.

**Parameters**

- **cae** ( *ConsoleApp* ) – instance of the running *ConsoleApp* app.
- **use\_kernel** – pass True to use the Sihot kernel interface (False==use Sihot sxml interface).

**fetch\_res\_no**(*obj\_id*, *scope*='GET')

determine reservation and guest data for the passed reservation object id.

**Parameters**

- **obj\_id** ( *str* ) – Sihot reservation object id.
- **scope** – search scope string (see 7.3.1.2 in Sihot KERNEL interface doc V 9.0).

**Return type**

*Tuple*[*Any*, ...]

**Returns**

either the reservation ids as tuple of (hotel\_id, res\_id, sub\_id, gds\_no) or the tuple (None, “error”) if the reservation was not found.

**response:** *Optional*[*SihotXmlParser*]

**class ShSysConnector**(*system*)

Bases: *SystemConnectorBase*

connector class for the Sihot system.

**connect**()

not needed - lazy connection

**Return type**

*str*

**static clients\_match\_field\_init**(*match\_fields*)

check and return the match field for a client search.

**Parameters**

**match\_fields** ( *Sequence* ) – tuple/list of length 1 with the match field name.

**Return type**

*str*

**Returns**

match field name or an error message if an error occurred.

**last\_err\_msg:** *str*

last system connection error message(s)

## 4.21 ae.sys\_data\_sh

### 4.21.1 Sihot system data interface

This portion is very old and needs refactoring and much more unit tests.

The sys record data classes provided by this portion are based on client and server components (to communicate with the Sihot PMS system) provided by the [ae.sys\\_core\\_sh](#) portion of the ae namespace package.

#### TODO:

- split into several small modules.
- enhance type annotation.
- add much more unit tests.

#### Module Attributes

<a href="#"><i>SH_CLIENT_MAP</i></a>	default map for ClientFromSihot.elem_fld_map instance.
<a href="#"><i>SH_RES_MAP</i></a>	reservation map

#### Functions

<a href="#"><i>add_sh_options</i></a> (cae[, client_port, ...])	add config/command line options.
<a href="#"><i>client_data</i></a> (cae, obj_id)	fetch the client data record from Sihot for the client specified by its Sihot object id.
<a href="#"><i>complete_res_data</i></a> (rec)	complete reservation data row (rec) with default values.
<a href="#"><i>convert_date_from_sh</i></a> (xml_string)	convert date string sent by the Sihot system into a datetime.date value.
<a href="#"><i>convert_date_onto_sh</i></a> (date)	convert datetime.date value into a date string in the format of the Sihot system.
<a href="#"><i>date_range_chunks</i></a> (date_from, date_till, ...)	split date range into manageable chunks respecting the passed maximum days length and yield them back.
<a href="#"><i>elem_path_join</i></a> (elem_names)	convert list of element names to element path.
<a href="#"><i>gds_no_to_ids</i></a> (cae, hotel_id, gds_no)	determine all reservation ids for a reservation with the passed GDS number.
<a href="#"><i>gds_no_to_obj_id</i></a> (cae, hotel_id, gds_no)	determine the Sihot object id of a reservation identified by the GDS number.
<a href="#"><i>hotel_and_res_id</i></a> (shd)	determine hotel and reservation ids.
<a href="#"><i>obj_id_to_res_no</i></a> (cae, obj_id)	using RESERVATION-GET oc from KERNEL interface (see 7.3 in SIHOT KERNEL interface doc).
<a href="#"><i>pax_count</i></a> (shd)	determine the number of PAX of the passed reservation record.
<a href="#"><i>print_sh_options</i></a> (cae)	print the current config options values to the console.
<a href="#"><i>res_no_to_ids</i></a> (cae, hotel_id, res_id, sub_id)	determine all reservation ids for a reservation with the passed reservation number.
<a href="#"><i>res_no_to_obj_id</i></a> (cae, hotel_id, res_id, sub_id)	determine the Sihot object id of a reservation identified by their reservation number.
<a href="#"><i>res_search</i></a> (cae, date_from[, date_till, ...])	search reservations with the criteria specified by the parameters.

## Classes

<i>BulkFetcherBase</i> (cae[, add_kernel_port])	helper base class for bulk data fetches.
<i>ClientFetch</i> (cae)	build xml and send to the Sihot system to fetch a client record.
<i>ClientFromSihot</i> (cae[, elem_map])	extended xml parser converting client data directly into a sys data Record structure.
<i>ClientSearch</i> (cae)	search client.
<i>ClientToSihot</i> (cae)	extended xml builder class to send/push client data to Sihot.
<i>FldMapXmlBuilder</i> (cae[, use_kernel, elem_map])	extended xml builder base class.
<i>FldMapXmlParser</i> (cae, elem_map)	extended xml parser converting data directly into a sys data Record structure.
<i>GuestBulkFetcher</i> (cae[, add_kernel_port])	WIP/NotUsed/NoTests: the problem is with GUEST-SEARCH is that there is no way to bulk fetch all guests because the search criteria is not providing range search to split in slices.
<i>ResBulkFetcher</i> (cae[, allow_future_arrivals])	reservation bulk fetch.
<i>ResFetch</i> (cae[, use_kernel])	fetch reservation.
<i>ResFromSihot</i> (cae)	extended xml parser converting reservation data directly into a sys data Record structure.
<i>ResSearch</i> (cae[, use_kernel])	search reservation.
<i>ResSender</i> (cae)	helper class to send reservation records to Sihot.
<i>ResToSihot</i> (cae)	extended xml builder to send reservation data to Sihot.

### convert\_date\_from\_sh(*xml\_string*)

convert date string sent by the Sihot system into a datetime.date value.

#### Return type

date

### convert\_date\_onto\_sh(*date*)

convert datetime.date value into a date string in the format of the Sihot system.

#### Return type

str

```
SH_CLIENT_MAP = (('OBJID', 'ShId', None, <function <lambda>>), ('MATCHCODE', 'AcuId'),
('T-SALUTATION', 'Salutation'), ('T-TITLE', 'Title'), ('T-GUEST', 'GuestType', '1'),
('NAME-1', 'Surname'), ('NAME-2', 'Forename'), ('STREET', 'Street'), ('PO-BOX', 'POBox'),
('ZIP', 'Postal'), ('CITY', 'City'), ('T-COUNTRY-CODE', 'Country'), ('T-STATE', 'State',
None, <function <lambda>>), ('T-LANGUAGE', 'Language'), ('T-NATION', 'Nationality', None,
<function <lambda>>), ('COMMENT', 'Comment'), ('COMMUNICATION/', None, None, <function
<lambda>>), ('PHONE-1', 'Phone'), ('PHONE-2', 'WorkPhone'), ('FAX-1', 'Fax'), ('EMAIL-1',
'Email'), ('EMAIL-2', 'EmailB'), ('MOBIL-1', 'MobilePhone'), ('MOBIL-2', 'MobilePhoneB'),
('/COMMUNICATION', None, None, <function <lambda>>), ('ADD-DATA/', None, None, <function
<lambda>>), ('T-PERSON-GROUP', None, '1A'), ('D-BIRTHDAY', 'DOB', None, None, <function
<lambda>>, <function <lambda>>), ('INTERNET-PASSWORD', 'Password'), ('MATCH-ADM',
'RciId'), ('MATCH-SM', 'SfId'), ('/ADD-DATA', None, None, <function <lambda>>))
```

default map for ClientFromSihot.elem\_fld\_map instance. as read-only constant by AcuClientToSihot using the SIHOT KERNEL interface because SiHOT WEB V9 has missing fields: initials (CD\_INIT1/2) and profession (CD\_INDUSTRY1/2).

latest :

```
SH_RES_MAP = (('SIHOT-Document.ID', 'ResHotelId'), ('ARESLIST/',), ('RESERVATION/',),
('RESERVATION.RES-HOTEL', 'ResHotelId'), ('RESERVATION.RES-NR', 'ResId', None, <function
<lambda>>), ('RESERVATION.SUB-NR', 'ResSubId', None, <function <lambda>>),
('RESERVATION.OBJID', 'ResObjId', None, <function <lambda>>), ('GDSNO', 'ResGdsNo'),
('RESERVATION.GUEST-ID', 'ShId'), ('RESERVATION.GUEST-OBJID', 'ShId'),
('RESERVATION.MATCHCODE', 'AcuId'), ('RESERVATION.NAME', 'Surname'), ('VOUCHERNUMBER',
'ResVoucherNo', None, <function <lambda>>), ('EXT-KEY', 'ResGroupNo', None, <function
<lambda>>), ('FLAGS', None, 'IGNORE-OVERBOOKING'), ('RT', 'ResStatus'), ('CAT',
'ResRoomCat'), ('PCAT', 'ResPriceCat', None, <function <lambda>>), ('ALLOTMENT-EXT-NO',
'ResAllotmentNo', '', <function <lambda>>), ('PAYMENT-INST', 'ResAccount', None,
<function <lambda>>), ('SALES-DATE', 'ResBooked', None, <function <lambda>>, <function
<lambda>>, <function <lambda>>), ('RATE-SEGMENT', 'ResRateSegment', None, <function
<lambda>>, ''), ('RATE/', None, None, <function <lambda>>), ('RATE.R', 'ResBoard', None,
<function <lambda>>), ('RATE.ISDEFAULT', None, 'Y', <function <lambda>>), ('/RATE', None,
None, <function <lambda>>), ('RATE/', None, None, <function <lambda>>), ('RATE.R',
'ResRateBoard', None, <function <lambda>>), ('RATE.ISDEFAULT', None, 'Y', <function
<lambda>>), ('RATE.DAYS/', None, None, <function <lambda>>), ('RATE.DAYS.D', ('ResRates',
0, 'RateDay'), None, <function <lambda>>, <function <lambda>>, <function <lambda>>),
('RATE.DAYS.PRICE', ('ResRates', 0, 'RateAmount'), None, <function <lambda>>),
('RATE./DAYS', None, None, <function <lambda>>), ('/RATE', None, None, <function
<lambda>>), ('RATE/', None, None, <function <lambda>>), ('RATE.R', None, 'GSC', <function
<lambda>>), ('RATE.ISDEFAULT', None, 'N', <function <lambda>>), ('/RATE', None, None,
<function <lambda>>), ('RESCHANNELLIST/', None, None, <function <lambda>>),
('RESCHANNEL/', None, None, <function <lambda>>), ('RESCHANNEL.IDX', None, 1, <function
<lambda>>), ('RESCHANNEL.MATCHCODE', None, 'RCI', <function <lambda>>),
('RESCHANNEL.ISPRICEOWNER', None, 1, <function <lambda>>), ('RESCHANNEL.IDX', None, 1,
<function <lambda>>), ('RESCHANNEL.MATCHCODE', None, 'MAR01', <function <lambda>>),
('RESCHANNEL.ISPRICEOWNER', None, 1, <function <lambda>>), ('RESCHANNEL.IDX', None, 2,
<function <lambda>>), ('RESCHANNEL.MATCHCODE', None, 'TSP', <function <lambda>>),
('RESCHANNEL.ISPRICEOWNER', None, 1, <function <lambda>>), ('/RESCHANNEL', None, None,
<function <lambda>>), ('/RESCHANNELLIST', None, None, <function <lambda>>), ('ARR',
'ResArrival', None, None, <function <lambda>>, <function <lambda>>), ('DEP',
'ResDeparture', None, None, <function <lambda>>, <function <lambda>>), ('NOROOMS', None,
1), ('NOPAX', 'ResAdults', None, None, <function <lambda>>, <function <lambda>>),
('NOCHILDS', 'ResChildren', None, <function <lambda>>, <function <lambda>>, <function
<lambda>>), ('TEC-COMMENT', 'ResLongNote', None, <function <lambda>>), ('COMMENT',
'ResNote', None, <function <lambda>>), ('MARKETCODE-NO', 'ResMktSegment', None, <function
<lambda>>), ('MARKETCODE', 'ResMktSegment'), ('SOURCE', 'ResSource', None, <function
<lambda>>), ('NN', 'ResMktGroupNN', None, <function <lambda>>), ('CHANNEL',
'ResMktGroup', None, <function <lambda>>), ('EXT-REFERENCE', 'ResFlightArrComment', None,
<function <lambda>>), ('ARR-TIME', 'ResFlightETA'), ('PICKUP-TIME-ARRIVAL',
'ResFlightETA', None, <function <lambda>>), ('PICKUP-TYPE-ARRIVAL', None, 1, <function
<lambda>>), ('PERSON/', None, None, <function <lambda>>), ('PERSON.GUEST-ID',
('ResPersons', 0, 'PersShId'), None, <function <lambda>>), ('PERSON.MATCHCODE',
('ResPersons', 0, 'PersAcuId'), None, <function <lambda>>), ('PERSON.NAME',
('ResPersons', 0, 'PersSurname'), None, <function <lambda>>), ('PERSON.NAME2',
('ResPersons', 0, 'PersForename'), None, <function <lambda>>), ('PERSON.AUTO-GENERATED',
('ResPersons', 0, 'AutoGen'), '1', <function <lambda>>), ('PERSON.ROOM-SEQ',
('ResPersons', 0, 'RoomSeq'), None, <function <lambda>>), ('PERSON.ROOM-PERS-SEQ',
('ResPersons', 0, 'RoomPersSeq'), None, <function <lambda>>), ('PERSON.PERS-TYPE',
('ResPersons', 0, 'TypeOfPerson'), None, <function <lambda>>), ('PERSON.RN',
('ResPersons', 0, 'RoomNo'), None, <function <lambda>>), ('PERSON.DOB', ('ResPersons', 0,
'PersDOB'), None, <function <lambda>>, <function <lambda>>, <function <lambda>>),
('PERSON.COUNTRY-CODE', ('ResPersons', 0, 'PersCountry'), None, <function <lambda>>),
('PERSON.EMAIL', ('ResPersons', 0, 'PersEmail'), None, <function <lambda>>),
('PERSON.LANG', ('ResPersons', 0, 'PersLanguage'), None, <function <lambda>>),
('PERSON.PHONE', ('ResPersons', 0, 'PersPhone'), None, <function <lambda>>),
('PERSON.PERS-RATE.R', ('ResPersons', 0, 'Board'), None, <function <lambda>>),
('PERSON.PICKUP-COMMENT-ARRIVAL', ('ResPersons', 0, 'FlightArrComment'), None, <function
<lambda>>), ('PERSON.PICKUP-TIME-ARRIVAL', ('ResPersons', 0, 'FlightETA'), None,
```

reservation map

**add\_sh\_options**(*cae*, *client\_port*=0, *add\_kernel\_port*=False, *add\_maps\_and\_kernel\_usage*=False)

add config/command line options.

**Parameters**

- **cae** (ConsoleApp) – app instance.
- **client\_port** (int) – client port.
- **add\_kernel\_port** (bool) – pass True to add also port option for the kernel interface.
- **add\_maps\_and\_kernel\_usage** (bool) – pass True to add also options for kernel switch and maps.

**print\_sh\_options**(*cae*)

print the current config options values to the console.

**client\_data**(*cae*, *obj\_id*)

fetch the client data record from Sihot for the client specified by its Sihot object id.

**Parameters**

- **cae** (ConsoleApp) – app instance.
- **obj\_id** (str) – Sihot client object id.

**Return type**

Record

**Returns**

client record data structure.

**complete\_res\_data**(*rec*)

complete reservation data row (rec) with default values.

**Parameters**

- **rec** (Record) – reservation data Record instance.

**Return type**

Record

**Returns**

completed reservation data Record instance.

**un-changed fields:**

ResRoomNo, ResNote, ResLongNote, ResFlightArrComment (flight no...), ResAllotmentNo, ResVoucherNo.

**mandatory fields:**

ShId or AcuId or Surname (to specify the orderer of the reservation), ResHotelId, ResArrival, ResDeparture, ResRoomCat, ResMktSegment, ResGdsNo, ResAdults, ResChildren.

**optional fields:**

ResPersons0PersSurname and ResPersons0PersForename (surname and forename) ResPersons1PersSurname and ResPersons1PersForename ( ... )

**optional auto-populated fields:**

see the default values - specified in default\_values dict underneath.

**elem\_path\_join**(*elem\_names*)

convert list of element names to element path.

**Parameters**

**elem\_names** *//* (*List*[*str*]) – list of element names.

**Return type**

*str*

**Returns**

element path.

**hotel\_and\_res\_id**(*shd*)

determine hotel and reservation ids.

**Return type**

*Tuple*[*Optional*[*str*], *Optional*[*str*]]

**pax\_count**(*shd*)

determine the number of PAX of the passed reservation record.

**Return type**

*int*

**date\_range\_chunks**(*date\_from*, *date\_till*, *fetch\_max\_days*)

split date range into manageable chunks respecting the passed maximum days length and yield them back.

**Return type**

*Iterator*[*Tuple*[*date*, *date*]]

**gds\_no\_to\_ids**(*cae*, *hotel\_id*, *gds\_no*)

determine all reservation ids for a reservation with the passed GDS number.

**Return type**

*Dict*[*str*, *Any*]

**gds\_no\_to\_obj\_id**(*cae*, *hotel\_id*, *gds\_no*)

determine the Sihot object id of a reservation identified by the GDS number.

**Return type**

*str*

**res\_no\_to\_ids**(*cae*, *hotel\_id*, *res\_id*, *sub\_id*)

determine all reservation ids for a reservation with the passed reservation number.

**Return type**

*Union*[*str*, *Dict*[*str*, *Any*]]

**res\_no\_to\_obj\_id**(*cae*, *hotel\_id*, *res\_id*, *sub\_id*)

determine the Sihot object id of a reservation identified by their reservation number.

**Return type**

*str*

**res\_search**(*cae*, *date\_from*, *date\_till*=*None*, *mkt\_sources*=*None*, *mkt\_groups*=*None*, *max\_los*=28, *search\_flags*="", *search\_scope*="", *chunk\_pause*=1)

search reservations with the criteria specified by the parameters.

**Parameters**

- **cae** *//* – instance of the application environment specifying searched Sihot server.



- **date\_from** (date) – date of first day of included arrivals.
- **date\_till** (Optional[date]) – date of last day of included arrivals.
- **mkt\_sources** (Optional[List[str]]) – list of market source codes.
- **mkt\_groups** (Optional[List[str]]) – list of market group codes.
- **max\_los** (int) – integer with maximum length of stay.
- **search\_flags** (str) – string with search flag words (separated with semicolon).
- **search\_scope** (str) – string with search scope words (separated with semicolon).
- **chunk\_pause** (int) – integer with seconds to pause between fetch of date range chunks.

**Return type**

Union[str, Records]

**Returns**

string with error message if error or Records/list of Sihot reservations (Record instances).

**obj\_id\_to\_res\_no**(cae, obj\_id)

using RESERVATION-GET oc from KERNEL interface (see 7.3 in SIHOT KERNEL interface doc).

**Parameters**

- **cae** (ConsoleApp) – Console App Environment instance.
- **obj\_id** (str) – Sihot Reservation Object Id.

**Return type**

tuple

**Returns**

reservation ids as tuple of (hotel\_id, res\_id, sub\_id, gds\_no) or (None, “error”) if not found

**\_strip\_err\_msg**(error\_msg)**Return type**

str

**class FldMapXmlParser**(cae, elem\_map)Bases: *SihotXmlParser*

extended xml parser converting data directly into a sys data Record structure.

**\_\_init\_\_**(cae, elem\_map)**clear\_rec**()

clear the record data.

**property rec**: *Record*

return the record data.

**start**(tag, attrib)

process start of new xml element.

**Return type**

Optional[str]

**data**(data)

process data of xml element.

**Return type**  
Optional[str]

**end(tag)**

process end of xml element.

**Return type**  
Optional[str]

```
class ClientFromSihot(cae, elem_map=((('OBJID', 'ShId', None, <function <lambda>>), ('MATCHCODE',
'AcuId'), ('T-SALUTATION', 'Salutation'), ('T-TITLE', 'Title'), ('T-GUEST', 'GuestType',
'I'), ('NAME-1', 'Surname'), ('NAME-2', 'Forename'), ('STREET', 'Street'), ('PO-BOX',
'POBox'), ('ZIP', 'Postal'), ('CITY', 'City'), ('T-COUNTRY-CODE', 'Country'),
('T-STATE', 'State', None, <function <lambda>>), ('T-LANGUAGE', 'Language'),
('T-NATION', 'Nationality', None, <function <lambda>>), ('COMMENT', 'Comment'),
('COMMUNICATION', None, None, <function <lambda>>), ('PHONE-1', 'Phone'),
('PHONE-2', 'WorkPhone'), ('FAX-1', 'Fax'), ('EMAIL-1', 'Email'), ('EMAIL-2',
'EmailB'), ('MOBIL-1', 'MobilePhone'), ('MOBIL-2', 'MobilePhoneB'),
('/COMMUNICATION', None, None, <function <lambda>>), ('ADD-DATA', None,
None, <function <lambda>>), ('T-PERSON-GROUP', None, 'IA'), ('D-BIRTHDAY',
'DOB', None, None, <function <lambda>>, <function <lambda>>),
('INTERNET-PASSWORD', 'Password'), ('MATCH-ADM', 'RciId'), ('MATCH-SM',
'SfId'), ('ADD-DATA', None, None, <function <lambda>>)))
```

Bases: [FldMapXmlParser](#)

extended xml parser converting client data directly into a sys data Record structure.

```
__init__(cae, elem_map=((('OBJID', 'ShId', None, <function <lambda>>), ('MATCHCODE', 'AcuId'),
('T-SALUTATION', 'Salutation'), ('T-TITLE', 'Title'), ('T-GUEST', 'GuestType', 'I'), ('NAME-1',
'Surname'), ('NAME-2', 'Forename'), ('STREET', 'Street'), ('PO-BOX', 'POBox'), ('ZIP', 'Postal'),
('CITY', 'City'), ('T-COUNTRY-CODE', 'Country'), ('T-STATE', 'State', None, <function
<lambda>>), ('T-LANGUAGE', 'Language'), ('T-NATION', 'Nationality', None, <function
<lambda>>), ('COMMENT', 'Comment'), ('COMMUNICATION', None, None, <function
<lambda>>), ('PHONE-1', 'Phone'), ('PHONE-2', 'WorkPhone'), ('FAX-1', 'Fax'), ('EMAIL-1',
'Email'), ('EMAIL-2', 'EmailB'), ('MOBIL-1', 'MobilePhone'), ('MOBIL-2', 'MobilePhoneB'),
('/COMMUNICATION', None, None, <function <lambda>>), ('ADD-DATA', None, None,
<function <lambda>>), ('T-PERSON-GROUP', None, 'IA'), ('D-BIRTHDAY', 'DOB', None, None,
<function <lambda>>, <function <lambda>>), ('INTERNET-PASSWORD', 'Password'),
('MATCH-ADM', 'RciId'), ('MATCH-SM', 'SfId'), ('ADD-DATA', None, None, <function
<lambda>>)))
```

**end(tag)**

process end of xml element.

**Return type**  
Optional[str]

```
class ResFromSihot(cae)
```

Bases: [FldMapXmlParser](#)

extended xml parser converting reservation data directly into a sys data Record structure.

```
__init__(cae)
```

**end(tag)**

process end of xml element.

**Return type**  
`Optional[str]`

**class ClientFetch(cae)**

Bases: `SihotXmlBuilder`

build xml and send to the Sihot system to fetch a client record.

**\_\_init\_\_**(cae)

create an instance of this class.

**Parameters**

- **cae** `(ConsoleApp)` – instance of the running `ConsoleApp` app.
- **use\_kernel** – pass True to use the Sihot kernel interface (False==use Sihot sxml interface).

**fetch\_client**(obj\_id, field\_names=())

return Record with guest data OR str with error message in case of error.

**Return type**  
`Union[str, Record]`

**response:** `Optional[SihotXmlParser]`

**class ClientSearch(cae)**

Bases: `SihotXmlBuilder`

search client.

**\_\_init\_\_**(cae)

create an instance of this class.

**Parameters**

- **cae** – instance of the running `ConsoleApp` app.
- **use\_kernel** – pass True to use the Sihot kernel interface (False==use Sihot sxml interface).

**search\_clients**(matchcode="", exact\_matchcode=True, name="", forename="", surname="", guest\_no="", email="", guest\_type="", flags='FIND-ALSO-DELETED-GUESTS', order\_by="", limit=0, field\_names=('ShId'), \*\*kwargs)

invoke the client search.

**Return type**  
`Union[str, list, Records]`

**client\_id\_by\_matchcode**(matchcode)

determine client id for a client identified by its match code.

**Return type**  
`Optional[str]`

**response:** `Optional[SihotXmlParser]`

**class ResFetch(cae, use\_kernel=False)**

Bases: `SihotXmlBuilder`

fetch reservation.

**fetch\_res**(*ho\_id*, *gds\_no*="", *res\_id*="", *sub\_id*="", *scope*='USEISODATE')

invoke request and fetch of reservation data.

**Return type**

`Union[str, Record]`

**fetch\_by\_gds\_no**(*ho\_id*, *gds\_no*, *scope*='USEISODATE')

fetch reservation identified by their GDS number.

**Return type**

`Union[str, Record]`

**fetch\_by\_res\_id**(*ho\_id*, *res\_id*, *sub\_id*, *scope*='USEISODATE')

fetch reservation identified by their reservation number and sub-number.

**Return type**

`Union[str, Record]`

**response:** `Optional[SihotXmlParser]`

**class ResSearch**(*cae*, *use\_kernel*=False)

Bases: `SihotXmlBuilder`

search reservation.

**search\_res**(*hotel\_id*="", *from\_date*=datetime.date(2024, 3, 19), *to\_date*=datetime.date(2024, 3, 19),  
*matchcode*="", *name*="", *gds\_no*="", *flags*="", *scope*="", *guest\_id*="")

invoke search of reservation.

**Return type**

`Union[str, Records]`

**response:** `Optional[SihotXmlParser]`

**class FldMapXmlBuilder**(*cae*, *use\_kernel*=False, *elem\_map*=None)

Bases: `SihotXmlBuilder`

extended xml builder base class.

**\_\_init\_\_**(*cae*, *use\_kernel*=False, *elem\_map*=None)

create an instance of this class.

**Parameters**

- **cae** `(ConsoleApp)` – instance of the running `ConsoleApp` app.
- **use\_kernel** `(bool)` – pass True to use the Sihot kernel interface (False==use Sihot sxml interface).

**get\_warnings**()

get warning messages as concatenated string.

**Return type**

`str`

**wipe\_warnings**()

wipe all collected warning messages.

**prepare\_rec**(*rec*)

prepare record to push to Sihot.

**prepare\_map\_xml**(*rec*, *include\_empty\_values=True*)

prepare and return xml string.

**Return type**

`str`

**response:** `Optional[SihotXmlParser]`

**class ClientToSihot**(*cae*)

Bases: `FldMapXmlBuilder`

extended xml builder class to send/push client data to Sihot.

**\_\_init\_\_**(*cae*)

create an instance of this class.

**Parameters**

- *cae* – instance of the running `ConsoleApp` app.
- **use\_kernel** – pass True to use the Sihot kernel interface (False==use Sihot sxml interface).

**static \_complete\_client\_data**(*rec*)

**prepare\_rec**(*rec*)

prepare client record for the push to Sihot.

**\_prepare\_guest\_xml**(*rec*, *fld\_name\_suffix=""*)

prepare extra guest data.

**\_prepare\_guest\_link\_xml**(*mc1*, *mc2*)

prepare guest links.

**\_send\_link\_to\_sihot**(*pk1*, *pk2*)

**Return type**

`str`

**\_send\_person\_to\_sihot**(*rec*, *first\_person=""*)

send client data of one person to Sihot, passing *AcuId* of first person to send 2nd person.

**Return type**

`str`

**send\_client\_to\_sihot**(*rec*)

send client data to Sihot.

**Return type**

`str`

**response:** `Optional[SihotXmlParser]`

**\_warning\_msgs:** `List[str]`

**class ResToSihot**(*cae*)

Bases: `FldMapXmlBuilder`

extended xml builder to send reservation data to Sihot.

**\_\_init\_\_**(*cae*)

create an instance of this class.

**Parameters**

- **cae** (*ConsoleApp*) – instance of the running *ConsoleApp* app.
- **use\_kernel** – pass True to use the Sihot kernel interface (False==use Sihot sxml interface).

**\_add\_sihot\_configs**(*rec*)

**prepare\_rec**(*rec*)

prepare reservation data to be sent/pushed to Sihot system.

**\_prepare\_res\_xml**(*rec*)

**\_sending\_res\_to\_sihot**(*rec*)

**Return type**

str

**\_handle\_error**(*rec, err\_msg*)

**Return type**

str

**\_ensure\_clients\_exist\_and\_updated**(*rec, ensure\_client\_mode*)

**Return type**

str

**send\_res\_to\_sihot**(*rec, ensure\_client\_mode=0*)

send reservation to Sihot system.

**Return type**

str

**static res\_id\_label**()

reservation ids log and debug message label.

**Return type**

str

**static res\_id\_values**(*rec*)

reservation ids log and debug message data.

**Return type**

str

**res\_id\_desc**(*rec, err\_msg, separator='\n\n', indent=8*)

extended reservation ids log and debug message data.

**Return type**

str

**wipe\_gds\_errors**()

wipe collected gds errors.

**response:** Optional[*SihotXmlParser*]

---

```

    _warning_msgs: List[str]

class BulkFetcherBase(cae, add_kernel_port=True)
    Bases: object
    helper base class for bulk data fetches.
    __init__(cae, add_kernel_port=True)

    add_options()
        add command line option.

    load_options()
        load command line option.

    print_options()
        print command line option to console output.

class GuestBulkFetcher(cae, add_kernel_port=True)
    Bases: BulkFetcherBase

    WIP/NotUsed/NoTests: the problem is with GUEST-SEARCH is that there is no way to bulk fetch all guests
    because the search criteria is not providing range search to split in slices. Fetching all 600k clients is resulting
    in a timeout error after 30 minutes (see Sihot interface SDF_SH_TIMEOUT/'shTimeout' option value)

    fetch_all()
        fetch and return all found records.

        Return type
        Records

class ResBulkFetcher(cae, allow_future_arrivals=True)
    Bases: BulkFetcherBase
    reservation bulk fetch.
    __init__(cae, allow_future_arrivals=True)

    add_options()
        add command line options.

    load_options()
        load command line options.

    print_options()
        print command line options.

    date_range_str()
        determine date range as string value.

        Return type
        str

    fetch_all()
        fetch bulk reservation data.

        Return type
        Records

```

latest :

---

```
class ResSender(cae)
    Bases: ResToSihot
    helper class to send reservation records to Sihot.

    send_rec(rec)
        send reservation record.

        Return type
        Tuple[str, str]

    response: Optional[SihotXmlParser]

    _warning_msgs: List[str]

    _gds_errors: Dict[str, Tuple[Record, str]]

    get_res_no()
        determine reservation number of sent reservation.

        Return type
        tuple
```

## 4.22 ae.db\_core

### 4.22.1 database connection and data manipulation base classes

This module is providing generic base classes and helper methods to implement and integrate any database driver that is compatible to the **Python DB API**. The base classes are providing already all attributes and methods to handle database connections, data selections and manipulations.

Additionally this package is defining generalized interfaces to switch dynamically between databases without the need to adapt the code of your application - simply by selecting another database connector class within your *configuration files*.

#### basic usage

The abstract base class *DbBase* allows you to implement a db-specific class with few lines of code. Simply inherit from *DbBase* and implement a *connect()* method that is setting the *conn* instance attribute to a **Python DB API** compatible connection object.

For a fictive specific database driver package *SpecificDbDriver* that is providing a *connect* method, a minimal example would look like:

```
from ae.db_core import DbBase
...
class SpecificDb(DbBase):
    def connect(self) -> str:
        self.last_err_msg = ''
        try:
            self.conn = SpecificDbDriver.connect(**self.connect_params())
        except Exception as ex:
            self.last_err_msg = f"SpecificDb-connect() error: {ex} for {self}"
        else:
```

(continues on next page)



(continued from previous page)

```

self.create_cursor()
return self.last_err_msg

```

The helper method `connect_params()` provided by this module is helping you to merge any specified database system credentials and features into a dict to be passed as connection parameters to the database-specific connection class/function/factory of the database driver.

Another helper `create_cursor()` provided by `DbBase` can be used in the `connect()` method to create a Python DB API-compatible cursor object and assign it to the `cursor` attribute.

In this implementation your class `SpecificDb` does automatically also inherit useful methods from `DbBase` to execute `INSERT`, `DELETE`, `UPDATE` and `UPSERT` commands, to run `SELECT` queries and to call `stored procedures`.

---

**Hint:** Examples of an implementation of a specific database class are e.g. the portion packages `ae.db_pg` for Postgres and `ae.db_ora` for Oracle.

---

## connection parameters

`DbBase` supports the following connection parameter keys:

- **user** : user name or database account name.
- **password** : user account password.
- **dbname** : name of the database to connect.
- **host** : host name or IP address of the database server.
- **port**: port number of the database server.

Some database drivers are using default values for most of these connection parameters, if they are not specified. The mandatory connection parameters can differ for different database drivers. Most of them need at least a user name and password.

---

**Hint:** Most database drivers like e.g. `psycopg2` or `pg8000` for Postgres databases are supporting all of these connection parameter keys. For others like e.g. `cx_Oracle` which are not supporting some of them (like e.g. `host` and `dbname`) you may also need to overwrite the `connect_params()` method to convert/adopt unsupported connection parameter keys.

---

Alternatively you can provide the connection parameters needed by the database driver connector with a single database url string in the [SQLAlchemy Database URL format](#):

```
dialect+driver://user:password@host:port/dbname
```

So if your connection parameters dict - respective your database system credentials and features - providing a `url` key then `connect_params()` will parse and split the url into separate connection parameters.

---

**Note:** When you specify a connection parameter as separate key and also within the `url` value then the separate key value will have preference. The scheme part (dialect+driver) including the following colon character of the URL string are not evaluated and can be omitted.

---

latest :

---

## bind variables format

Bind variables in your sql query strings have to be formatted in the **named parameter style**. If your database driver is only supporting the *pyformat* parameter style, then overload the `__init__` method and set the *param\_style* to 'pyformat' and all sql query strings will be automatically converted by *DbBase* before they get sent to the database:

```
class SpecificDb(DbBase):
    def __init__(self, ...)
        super().__init__(...)
        self.param_style = 'pyformat'
    ...
```

## Module Attributes

<i>NAMED_BIND_VAR_PREFIX</i>	character to mark a bind variable in a sql query
<i>CHK_BIND_VAR_PREFIX</i>	bind variable name prefix, to allow for same column/-name a new/separate value in e.g.

## Functions

<i>connect_args_from_params</i> (conn_params)	split dict with database connection parameters into credentials and features.
---	---

## Classes

<i>DbBase</i> (system)	abstract database connector base class for the <i>ae</i> namespace database system layers.
------------------------	--

**NAMED\_BIND\_VAR\_PREFIX:** `str = ':'`

character to mark a bind variable in a sql query

**CHK\_BIND\_VAR\_PREFIX:** `str = 'CV_'`

bind variable name prefix, to allow for same column/-name a new/separate value in e.g. the SET clause and an old value in the WHERE clause. Gets added to bind variables in filters/chk\_values and extra where clauses.

**connect\_args\_from\_params**(conn\_params)

split dict with database connection parameters into credentials and features.

### Parameters

**conn\_params** *Dict*[`str`, `Any`] – connection params dict.

### Return type

*Tuple*[*Dict*[`str`, `str`], *List*[`str`]]

### Returns

tuple of credentials dict and features list.

**\_normalize\_col\_values**(*col\_values*)

convert empty strings into real None values.

**Parameters**

**col\_values** *(Dict[str, Any])* – dict of column values (dict key is the column name).

**Return type**

*Dict[str, Any]*

**Returns**

col\_values dict where empty values got replaced with None (also in original dict).

**\_prepare\_in\_clause**(*sql, bind\_vars=None, additional\_col\_values=None*)

replace list bind variables used in an IN clause with separate bind variables for each list item.

**Parameters**

- **sql** *(str)* – query to be executed.
- **bind\_vars** *(Optional[Dict[str, Any]])* – dict of all available bind variables for the execution.
- **additional\_col\_values** *(Optional[Dict[str, Any]])* – additional bind variables.

**Return type**

*Tuple[str, Dict[str, Any]]*

**Returns**

tuple of adapted query string and joined bind variables.

**\_rebind**(*chk\_values=None, where\_group\_order="", bind\_vars=None, extra\_bind=None*)

merge where\_group\_order string with chk\_values filter dict and merge/rename bind variables.

**Parameters**

- **chk\_values** *(Optional[Dict[str, Any]])* – dict with column\_name: value items, used to filter/restrict the resulting rows.

This method compiles this dict into a sql WHERE clause expression. If you also passed additional sql clauses into the *where\_group\_order* then it will be merged with the compiled expression. The names of the sql parameters and related bind variables are build from the column names/keys of this dict, and will be prefixed with *CHK\_BIND\_VAR\_PREFIX*.

Passing None or a empty dict to this and to the *extra\_bind* arguments will disable any filtering. If only this argument is None/empty then the first item of the *extra\_bind* dict will be used as filter.

- **where\_group\_order** *(str)* – sql part with optional WHERE/GROUP/ORDER clauses (the part after the WHERE), including bind variables in the *named parameter style*.
- **bind\_vars** *(Optional[Dict[str, Any]])* – dict with bind variables (variable name has to be prefixed in *where\_group\_order* with *CHK\_BIND\_VAR\_PREFIX*).
- **extra\_bind** *(Optional[Dict[str, Any]])* – additional dict with bind variables (variable name has NOT to be prefixed/adapted in *where\_group\_order* argument).

**Return type**

*Tuple[Optional[Dict[str, Any]], str, Dict[str, Any]]*

**Returns**

tuple of corrected/rebound values of chk\_values, where\_group\_order and bind\_vars.

**class** `DbBase(system)`

Bases: `SystemConnectorBase`, `ABC`

abstract database connector base class for the *ae* namespace database system layers.

This class inherits from `SystemConnectorBase` the following attributes:

**system**: instance of the related `ae-sys_core.SystemBase` class.

**console\_app**: instance of the application using this database system.

**last\_err\_msg**: the error message string of the last error or an empty string if no error occurred in the last database action/operation.

**\_\_init\_\_**(system)

create instance of generic database object (base class for real database like e.g. postgres or oracle).

#### Parameters

**system** `(SystemBase)` – `SystemBase` instance. Providing useful attributes, like e.g.:

**credentials**: dict with database driver specific account credentials.

**features**: list of features. Features will also be passed as connection parameters to the database driver. The main differences to credentials are that a feature without any value will be interpreted as boolean True and that features can have any value that can be specified as a literal (like int or even datetime).

**console\_app**: instance of the application using this database system.

**conn**

database driver connection

**curs**

database driver cursor

**param\_style**: `str`

database driver bind variable/parameter style

**abstract connect()**

sub-class has to implement this connect method

#### Return type

`str`

**\_adapt\_sql**(sql, bind\_vars)

replace the parameter style of bind variables from *pyformat* into *named*.

#### Parameters

- **sql** `(str)` – query to scan for named bind variables.

- **bind\_vars** `(Dict[str, Any])` – dict of all available bind variables.

#### Return type

`str`

#### Returns

adapted query string.

---

**Note:** For database drivers - like `psycopg2` - that are support only the *pyformat* parameter style syntax (in the format `%(bind_var)s`) the sql query string will be adapted, by converting all bind variables from the parameter style *named* into *pyformat*.

The returned query will be unchanged for all other database drivers (that are directly supporting the *named* parameter style).

**call\_proc**(*proc\_name*, *proc\_args*, *ret\_dict=None*)

execute stored procedure on database server.

**Parameters**

- **proc\_name** (str) – name of the stored procedure.
- **proc\_args** (Sequence) – tuple of parameters/arguments passed to the stored procedure.
- **ret\_dict** (Optional[Dict[str, Any]]) – optional dict - if passed then the dict item with the key *return* will be set/updated to the value returned from the stored procedure/database.

**Return type**

str

**Returns**

empty string if no error occurred else the error message.

**close**(*commit=True*)

close the connection to the database driver and server.

**Parameters**

**commit** (bool) – pass False to prevent commit (and also execute rollback) before closure of connection.

**Return type**

str

**Returns**

empty string if no error occurred else the error message.

**connect\_params**()

merges self.system.credentials with self.system.features into a database driver connection parameters dict.

**Return type**

Dict[str, Any]

**Returns**

new dict with the items from self.system.credentials and then extended with the entries of self.system.features.

Credential values are always of type str, only features can be of any type. Features without a specified value are set True. All keys of the returned dict will be in lower-case.

**create\_cursor**()

allow sub-class to create Python DB API-conform database driver cursor

**cursor\_description**()

return description text if opened cursor or None if cursor is closed or not yet opened.

**Return type**

Optional[str]

**fetch\_all**()

fetch all the rows found from the last executed SELECT query.

**Return type**

Sequence[Tuple]

**Returns**

empty list on error or if query result is empty, else a list of database rows.

**fetch\_value**(*col\_idx=0*)

fetch the value of a column of the first/next row of the found rows of the last SELECT query.

**Parameters**

**col\_idx** (int) – index of the column with the value to fetch and return.

**Return type**

Any

**Returns**

value of the column at index *col\_idx*.

**execute\_sql**(*sql, commit=False, bind\_vars=None*)

execute sql query with optional bind variables.

**Parameters**

- **sql** (str) – sql query to execute.
- **commit** (bool) – pass True to execute a COMMIT command directly after the query execution.
- **bind\_vars** (Optional[Dict[str, Any]]) – optional dict with bind variables.

**Return type**

str

**Returns**

empty string if no error occurred else the error message.

**delete**(*table\_name, chk\_values=None, where\_group\_order="", bind\_vars=None, commit=False*)

execute a DELETE command against a table.

**Parameters**

- **table\_name** (str) – name of the database table.
- **chk\_values** (Optional[Dict[str, Any]]) – dict of column names/values to identify the record(s) to delete.
- **where\_group\_order** (str) – extra sql added after the WHERE clause (merged with *chk\_values* by *\_rebind()*). This string can include additional WHERE expressions with extra bind variables.
- **bind\_vars** (Optional[Dict[str, Any]]) – dict of extra bind variables (key=variable name, value=value).
- **commit** (bool) – bool value to specify if commit should be done. Pass True to commit.

**Return type**

str

**Returns**

last error message or empty string if no errors occurred.

**insert**(*table\_name, col\_values, returning\_column="", commit=False*)

execute an INSERT command to add one record to a database table.

**Parameters**

- **table\_name** (str) – name of the database table.

- **col\_values** (Dict[str, Any]) – dict of inserted column values with the column name as key.
- **returning\_column** (str) – name of column which value will be returned by next `fetch_all()`/`fetch_value()` call.
- **commit** (bool) – bool value to specify if commit should be done. Pass True to commit.

**Return type**

str

**Returns**

last error message or empty string if no errors occurred.

**select**(*from\_join*="", *cols*=(), *chk\_values*=None, *where\_group\_order*="", *bind\_vars*=None, *hints*="")

execute a SELECT query against a database table.

**Parameters**

- **from\_join** (str) – name(s) of the involved database table(s), optional with JOIN clause(s). Passing an empty string results in a SELECT statement without the FROM keyword.
- **cols** (Sequence[str]) – sequence of the column names that will be selected and included in the resulting data-rows.
- **chk\_values** (Optional[Dict[str, Any]]) – dict of column names/values to identify selected record(s).
- **where\_group\_order** (str) – extra sql added after the WHERE clause (merged with `chk_values` by `_rebind()`). This string can include additional WHERE expressions with extra bind variables, ORDER BY and GROUP BY expressions. These special/extra bind variables have to be specified in the `bind_vars` argument and have to be prefixed with the string 'CV\_' in the WHERE clause (see also the `CHK_BIND_VAR_PREFIX` data constant in this module).
- **bind\_vars** (Optional[Dict[str, Any]]) – dict of extra bind variables (key=variable name, value=value).
- **hints** (str) – optional SELECT optimization hint string.

**Return type**

str

**Returns**

last error message or empty string if no errors occurred. Use the methods `fetch_all()` or `fetch_value()` to retrieve the resulting data-rows.

**update**(*table\_name*, *col\_values*, *chk\_values*=None, *where\_group\_order*="", *bind\_vars*=None, *commit*=False, *locked\_cols*=())

execute an UPDATE command against a database table.

**Parameters**

- **table\_name** (str) – name of the database table.
- **col\_values** (Dict[str, Any]) – dict of inserted/updated column values with the column name as key.
- **chk\_values** (Optional[Dict[str, Any]]) – dict of column names/values to identify affected record(s). If not passed then the first name/value of `col_values` is used as primary key check/filter value.

- **where\_group\_order** (str) – extra sql added after the WHERE clause (merged with `chk_values` by `_rebind()`). This string can include additional WHERE expressions with extra bind variables, ORDER BY and GROUP BY expressions. These special/extra bind variables have to be specified in the `bind_vars` argument and have to be prefixed with the string 'CV\_' (see also the `CHK_BIND_VAR_PREFIX` data constant in this module).
- **bind\_vars** (Optional[Dict[str, Any]]) – dict of extra bind variables (key=variable name, value=value).
- **commit** (bool) – bool value to specify if commit should be done. Pass True to commit.
- **locked\_cols** (Sequence[str]) – list of column names not be overwritten on update of column value is not empty.

**Return type**

str

**Returns**

last error message or empty string if no errors occurred.

**upsert**(*table\_name*, *col\_values*, *chk\_values*, *where\_group\_order*="", *bind\_vars*=None, *returning\_column*="", *commit*=False, *locked\_cols*=(), *multiple\_row\_update*=True)

execute an INSERT or UPDATE command against a record of a database table (UPDATE if record already exists).

**Parameters**

- **table\_name** (str) – name of the database table.
- **col\_values** (Dict[str, Any]) – dict of inserted/updated column values with the column name as key.
- **chk\_values** (Dict[str, Any]) – dict of column names/values to identify affected record(s), also used to check if record already exists (and data has to be updated instead of inserted). If not passed then the first name/value of `col_values` is used as primary key check/filter value.
- **where\_group\_order** (str) – extra sql added after the WHERE clause (merged with `chk_values` by `_rebind()`). This string can include additional WHERE expressions with extra bind variables, ORDER BY and GROUP BY expressions. These special/extra bind variables have to be specified in the `bind_vars` argument and have to be prefixed with the string 'CV\_' in the query string (see also the `CHK_BIND_VAR_PREFIX` data constant in this module).
- **bind\_vars** (Optional[Dict[str, Any]]) – dict of extra bind variables (key=variable name, value=value).
- **returning\_column** (str) – name of column which value will be returned by next `fetch_all`/`fetch_value`() call.
- **commit** (bool) – bool value to specify if commit should be done. Pass True to commit record changes.
- **locked\_cols** (Sequence[str]) – list of column names not be overwritten on update of column value is not empty.
- **multiple\_row\_update** (bool) – allow update of multiple records with the same `chk_values`.

**Return type**

str



**Returns**

last error message or empty string if no errors occurred.

**commit**(*reset\_last\_err\_msg=False*)

commit the current transaction if the database driver supports/implements a commit method.

**Parameters**

**reset\_last\_err\_msg** (bool) – pass True to reset the last error message of this instance before the commit.

**Return type**

str

**Returns**

last error message or empty string if no error happened.

**rollback**(*reset\_last\_err\_msg=False*)

roll the current transaction back if the DB driver supports transactions and does have a rollback method.

**Parameters**

**reset\_last\_err\_msg** (bool) – pass True to reset the last error message of this instance before the rollback.

**Return type**

str

**Returns**

last error message or empty string if no error happened.

**get\_row\_count**()

determine rowcount of last executed query.

**Return type**

int

**Returns**

the number of affected rows of the last query.

**selected\_column\_names**()

determine the column names for the last executed SELECT query.

**Return type**

List[str]

**Returns**

list of column names - order by column index.

**static thread\_lock\_init**(*table\_name, chk\_values*)

created named locking instance for passed table and filter/check expression.

**Return type**

NamedLocks

**Returns**

NamedLocks instance for this table and filter.

## 4.23 ae.db\_ora

### 4.23.1 database system core layer to access Oracle databases

the class *OraDb* of this namespace portion is a thin layer that is extending the *DbBase* of the module *ae.db\_core* to connect to an Oracle database.

**Hint:** this namespace portion is using the *cx\_Oracle* package as the database driver. the *cx\_Oracle* package has to have at least version 5 or higher.

#### basic usage of the oracle database layer

to create an instance of the class *OraDb* you first have to create a *SystemBase* instance. this can be done either programmatically by providing an application instance (of the class *ConsoleApp* or an inherited sub-class of it) plus any database parameters, like required credentials and any database configuration features/options:

```
app = ConsoleApp()
system = SystemBase('system-id', app, dict(User='user name', Password='password', ...), .
→ ..)
```

alternatively provide all system-specific info within the *ae config files* and let *UsedSystems* load it:

```
system = used_systems['system-id']
```

finally pass the database parameters in *system* to create an instance of *OraDb*:

```
ora_db = OraDb(system)
```

then call the *connect()* method of this instance to connect to the Oracle database server:

```
error_message = ora_db.connect()
if error_message:
    print(error_message)
```

if the connection could not be established then *connect()* is returning an error message string. if the return value is an empty string then you can use all the methods provided by *DbBase*, like e.g. *update()*:

```
error_message = ora_db.update('my_table', {'my_col': 'new value'})
if error_message:
    print(error_message)
    error_message = ora_db.rollback()
```

an explicit call of *rollback()* is only needed if you use transactions. in this case you should also use *commit()* at the end of each transaction to store any data updates:

```
error_message = ora_db.commit()
```

alternatively you can use the *commit* argument that is provided by the *DbBase* DML methods: by passing a *True* value to this argument, the method will automatically execute a *commit()* call for you if no error occurred in the DML method:

```
error_message = ora_db.update('table`, {'column': 369}, commit=True)
```

finally after all database actions are done you can close the connection to the databases server with the `close()` method:

```
error_message = ora_db.close()
```

## Classes

*OraDb*(system)

Oracle database class, based on *DbBase*

**class** *OraDb*(system)

Bases: *DbBase*

Oracle database class, based on *DbBase*

**\_\_init\_\_**(system)

create instance of oracle database object.

### Parameters

**system** (SystemBase) – instance of a *SystemBase* class.

*SystemBase* (defined in the module *ae.sys\_core*) is providing the credentials and features, which get retrieved from *config files*, then converted by *connect\_params()* into *connection parameters* to connect to the Postgres database.

if you experiencing the following unicode encoding error:

```
'charmap' codec can't decode byte 0x90 in position 2: character maps to
↳<undefined>
```

don't try to create a type handler like recommended in some places - still got same error after adding the following method to replace the *self.conn.outputtypehandler* of the database driver:

```
def output_type_handler(cursor, name, default_type, size, precision, scale):
    if default_type in (cx_Oracle.STRING, cx_Oracle.FIXED_CHAR):
        return cursor.var(cx_Oracle.NCHAR, size, cursor.arraysize)
```

luckily, finally found workaround by setting the following OS environment variable to the character set of the used Oracle server (here UTF8):

```
os.environ["NLS_LANG"] = ".AL32UTF8"
```

**connect**()

connect this instance to the database driver.

### Return type

str

**prepare\_ref\_param**(value)

prepare special Oracle reference parameter.

### Parameters

**value** (Union[datetime, int, float, str]) – the input value passed into the reference parameter of the called stored procedure.

**Return type***Any***Returns**

a handle to the reference variable.

the following code snippet shows how to use this method together with `get_value()` to retrieve the returned value of a reference parameter:

```
ora_db = OraDb(...)
*ref_var* = ora_db.prepare_ref_param("input_value")
err_msg = ora_db.call_proc('STORED_PROCEDURE', (*ref_var*, ...))
if not err_msg:
    output_value = ora_db.get_value(*ref_var*)
```

**static `get_value(var)`**

get output value from a reference variable passed into a stored procedure.

**Parameters****var** – handle to a reference variable.**Return type***Any***Returns**

output value of the reference variable.

**static `set_value(var, value)`**

set the input value of a reference variable to pass into a stored procedure.

**Parameters**

- **var** (*Any*) – handle to the reference variable to set.
- **value** (*Union[datetime, int, float, str]*) – value to set as input value of the reference variable.

**param\_style: `str`**

database driver bind variable/parameter style

**last\_err\_msg: `str`**

last system connection error message(s)

## 4.24 ae.db\_pg

### 4.24.1 postgres database layer

this module provides the class `PostgresDb` to connect and interact with a Postgres database server.

the communication with the database driver is done with great help of the `psycopg2` pypi package.

## basic usage of ae.db\_pg

to create an instance of the class *PostgresDb* you first have to create a *SystemBase* instance. this can be done either programmatically by providing an application instance (of the class *ConsoleApp* or an inherited subclass of it) plus any database parameters, like required credentials and any database configuration features/options:

```
app = ConsoleApp()
system = SystemBase('system-id', app, dict(User='user name', Password='password',
↪ Dbname=...), ...)
```

alternatively provide all system-specific info within the *ae config files* and let *UsedSystems* load it:

```
system = used_systems['system-id']
```

finally pass the database parameters in *system* to create an instance of *PostgresDb*:

```
pg_db = PostgresDb(system)
```

then call the *connect()* method of this instance to connect to the Postgres database server:

```
error_message = pg_db.connect()
if error_message:
    print(error_message)
```

if the connection could not be established then *connect()* is returning an error message string. if the return value is an empty string then you can use all the methods provided by *DbBase*, like e.g. *update()*:

```
error_message = pg_db.update('my_table', {'my_col': 'new value'})
if error_message:
    print(error_message)
    error_message = pg_db.rollback()
```

an explicit call of *rollback()* is only needed if you use transactions (autocommit is False). in this case you should also use *commit()* at the end of each transaction to store any data updates:

```
error_message = pg_db.commit()
```

alternatively you can use the *commit* argument that is provided by the *DbBase* DML methods: by passing a *True* value to this argument, the method will automatically execute a *commit()* call for you if no error occurred in the DML method:

```
error_message = pg_db.update('table', {'column': 369}, commit=True)
```

finally after all database actions are done you can close the connection to the databases server with the *close()* method:

```
error_message = pg_db.close()
```

## Classes

<i>PostgresDb</i> (system)	an instance of this class represents a Postgres database.
----------------------------	---

**class** `PostgresDb`(system)

Bases: *DbBase*

an instance of this class represents a Postgres database.

**\_\_init\_\_**(system)

create instance of Postgres database object

### Parameters

**system** (*SystemBase*) – instance of a *SystemBase* class.

*SystemBase* (defined in the module *ae.sys\_core*) is providing the credentials and features, which get retrieved from *config files*, then converted by *connect\_params()* into *connection parameters* to connect to the Postgres database.

for connections via SSL to the Postgres server you have to add either the connection parameters **sslmode**, **sslcert** and **sslkey** or **sslrootcert** and **sslerl** (depending on the configuration of your server).

**features** : optional list of features.

**param\_style**: *str*

database driver bind variable/parameter style

**connect()**

connect this instance to the Postgres database server, using the credentials provided at instantiation.

### Return type

*str*

### Returns

error message in case of error or empty string if not.

**execute\_sql**(sql, commit=False, bind\_vars=None, auto\_commit=False)

execute sql query or sql command.

### Parameters

- **sql** (*str*) – sql query or command to execute.
- **commit** (*bool*) – pass True to commit (after INSERT or UPDATE queries).
- **bind\_vars** (*Optional[Dict[str, Any]]*) – dict of bind variables (key=variable name, value=value).
- **auto\_commit** (*bool*) – pass True activate auto-commit-mode for this postgres session.

### Return type

*str*

### Returns

last error message or empty string if no errors occurred.

**Hint:** overwriting generic `execute_sql` for Postgres because if `auto_commit` is False then a db error is invalidating the connection until it gets rolled back (optionally to a save-point). unfortunately `psycopg2`

does not provide/implement save-points. could be done alternatively with execute(“SAVEPOINT NonAutoCommErrRollback”) but RELEASE/ROLLBACK makes it complicated (see also <https://stackoverflow.com/questions/2370328/continuing-a-transaction-after-primary-key-violation-error>):

```
save_point = None if auto_commit else self.conn.setSavepoint(
    ↪ 'NonAutoCommErrRollback')
super().execute_sql(sql, commit=commit, auto_commit=auto_commit, bind_vars=bind_
    ↪ vars)
if save_point:
    if self.last_err_msg:
        self.conn.rollback(save_point)
    else:
        self.conn.releaseSavepoint(save_point)
return self.last_err_msg
```

therefore KISS - a simple rollback will do it also.

**last\_err\_msg:** str

last system connection error message(s)

## 4.25 ae.transfer\_service

### 4.25.1 transfer client and server services

this ae portion is providing client and server services to transfer files and text messages between two devices in the same local network.

the number of parallel running file and message transfers is only limited by the available resources of the involved devices.

if a file transfers gets interrupted it can be recovered later and without the need to resend the already transferred file content.

standard file paths - like e.g. the documents or downloads folders - are getting automatically adopted to the specific path structures of each involved device and operating system.

#### transfer service life cycle

the transfer service can be invoked in different ways: standalone as a separate process or attached and embedded into a controlling application.

#### run transfer service in standalone mode

execute this module to run the transfer services as a separate standalone process via:

```
python transfer_service.py [--bind=...] [--port=...] [--buf_len=...]
```

the following command line options are overwriting the default server address and socket buffer length (see also [service\\_factory\(\)](#)):

- ‘bind’ to restrict the incoming connections to an ip address/range (overwriting the default [SERVER\\_BIND](#)).

**latest :**

---

- 'port' to specify the socket port (overwriting the default port `SERVER_PORT`).
- 'buf\_len' to specify the socket buffer length (overwriting the default buffer length `SOCKET_BUF_LEN`).

after that the transfer service will be able to receive files send from another process or device.

---

**Note:** on Android a standalone transfer service has to be started as android service.

---

### run transfer service attached to any app

alternatively you can run the transfer service server in a separate thread within respectively attached to your application:

```
from ae.transfer_service import service_factory

transfer_service_app = service_factory()
transfer_service_app.set_option('port', 12345, save_to_config=False)
transfer_service_app.set_option('buf_len', 34567, save_to_config=False)
transfer_service_app.start_server(threaded=True)
```

### pause or stop transfer service

to manually pause the transfer service, store the app instance of the transfer service app (`transfer_service_app` in the example above) and call its `stop_server()` method:

```
transfer_service_app.stop_server()
```

to fully stop the transfer service and terminate the transfer service app call instead its `shutdown()` method:

```
transfer_service_app.shutdown()
```

---

**Hint:** the `shutdown()` method of the base app instance (`AppBase`) automatically ensures a clean shutdown of the transfer service server on app quit/exit.

---

### send file to another transfer service server

to send files from one transfer server to another running transfer server, a separate client process has to be started on the device storing the file to be send.

to initiate the file transfer the client process has to make a tcp connection to the transfer server running on the same device, specifying the path of the file to send and the remote ip of the receiving transfer server and finally call the remote procedure `send_file` like shown in the following example:

```
import socket
from ae.transfer_service import connect_and_request

request_kwargs = dict(method_name='send_file', file_path='path_to_file/file_name.ext',
↳ remote_ip='192.168.3.123')
with socket.socket() as sock:
```

(continues on next page)



(continued from previous page)

```

response_kwargs = connect_and_request(sock, request_kwargs)

if 'error' in response_kwargs:
    # handle error (e.g. display to user or add to a log file)

```

if the file transfer failed then the transfer kwargs dict returned by `connect_and_request()` will contain an `error` key containing the error message text.

## implemented remote procedures

the following remote procedures are provided by the transfer service server:

- `cancel_request`: cancel running file transfer.
- `pending_requests`: get log info the progress/status of all currently running file transfers.
- `recv_file`: receive file from other transfer service server.
- `recv_message`: receive text message from other transfer service server.
- `send_file`: send file to other transfer service server.
- `send_message`: send text message to other transfer service server.

**Hint:** the demo app `ComPartY` is using all provided remote procedures.

## Module Attributes

<code>CONNECTION_TIMEOUT</code>	default timeout (in seconds) to connect and request a server process
<code>CONNECT_ERR_PREFIX</code>	error message string prefix if error happened directly in <code>connect_and_request()</code> helper (no protocol error).
<code>ENCODING_KWARGS</code>	default encoding and encoding error handling strategy
<code>SERVER_BIND</code>	setting BIND to "" or None to allow connections for all available interfaces
<code>SERVER_PORT</code>	server listening port
<code>SHUTDOWN_TIMEOUT</code>	timeout (in seconds) to shutdown/stop the console app
<code>SOCKET_BUF_LEN</code>	buf length for socket receives and sends
<code>TRANSFER_KWARGS_DATE_TIME_NAME_PARTS</code>	kwarg name suffix for values automatically converted
<code>TRANSFER_KWARGS_LINE_END_CHAR</code>	end of line/command character as string
<code>TRANSFER_KWARGS_LINE_END_BYTE</code>	
<code>TransferKwargs</code>	command/action format of requests and responses
<code>requests_lock</code>	locking requests TransferKwargs in <code>TransferServiceApp.reqs_and_logs</code>
<code>server_app</code>	transfer service server app

## Functions

<code>clean_log_str(log_str)</code>	remove high-commas and backslashes from the passed string to add it to the logs (preventing duplicates).
<code>connect_and_request(sock, request_kwargs[, ...])</code>	connect to remote, send first command/action and return response as transfer kwargs dict.
<code>recv_bytes(sock[, buf_len])</code>	receive all bytes from the passed client socket instance until connection lost or line end reached.
<code>service_factory([task_id_func])</code>	create server app instance including the command line options <i>bind</i> and <i>port</i> .
<code>transfer_kwargs_error(transfer_kwargs, err_msg)</code>	add/append error to transfer kwargs dict without overwriting any previous error.
<code>transfer_kwargs_from_literal(transfer_kwargs_lit)</code>	convert dict literal (created with <code>transfer_kwargs_literal()</code> ) to transfer kwargs dict.
<code>transfer_kwargs_literal(transfer_kwargs)</code>	convert dict to str literal to be sent via sockets (re-instantiable via <code>transfer_kwargs_from_literal()</code> ).
<code>transfer_kwargs_update(*variables, **kwargs)</code>	update multiple transfer dicts with the same keys/values (locking with <code>requests_lock</code> ).

## Classes

<code>ThreadedTCPRequestHandler(request, ...)</code>	server request handler.
<code>TransferServiceApp([app_title, app_name, ...])</code>	server service app class

**CONNECTION\_TIMEOUT = 2.7**

default timeout (in seconds) to connect and request a server process

**CONNECT\_ERR\_PREFIX = 'transfer\_service.connect\_and\_request() exception '**

error message string prefix if error happened directly in `connect_and_request()` helper (no protocol error).

**ENCODING\_KWARGS = {'encoding': 'UTF-8', 'errors': 'ignore'}**

default encoding and encoding error handling strategy

**SERVER\_BIND = ''**

setting BIND to '' or None to allow connections for all available interfaces

**SERVER\_PORT = 36969**

server listening port

**SHUTDOWN\_TIMEOUT = 3.9**

timeout (in seconds) to shutdown/stop the console app

**SOCKET\_BUF\_LEN = 16384**

buf length for socket receives and sends

**TRANSFER\_KWARGS\_DATE\_TIME\_NAME\_PARTS = ('\_date', '\_time')**

kwarg name suffix for values automatically converted

**TRANSFER\_KWARGS\_LINE\_END\_CHAR = '\n'**

end of line/command character as string

**TRANSFER\_KWARGS\_LINE\_END\_BYTE = b'\n'**

**TransferKwargs**

command/action format of requests and responses

alias of `Dict[str, Any]`

**requests\_lock** = <unlocked `_thread.lock` object>

locking requests TransferKwargs in `TransferServiceApp.reqs_and_logs`

**clean\_log\_str**(*log\_str*)

remove high-commas and backslashes from the passed string to add it to the logs (preventing duplicates).

**Parameters**

**log\_str** `(Union[str, bytes])` – log string or bytes array to clean up.

**Return type**

`str`

**Returns**

cleaned log string.

**connect\_and\_request**(*sock, request\_kwargs, buf\_len=16384, timeout=2.7*)

connect to remote, send first command/action and return response as transfer kwargs dict.

**Parameters**

- **sock** `(socket)` – new socket instance.
- **request\_kwargs** `(Dict[str, Any])` – first/initial request kwargs dict. if the key ‘server\_address’ is not provided (with the server address as (host, ip) tuple), then (‘local-host’, SERVER\_PORT) is used. if the key ‘local\_ip’ is not specified then the local ip address will be used.
- **buf\_len** `(int)` – socket buffer length. if not passed then `SOCKET_BUF_LEN` will be used. pass zero/0 to use the buf length defined via the ‘buf\_len’ command line option.
- **timeout** `(Optional[float])` – timeout in seconds or None to use socket/system default timeout. if not passed then the default timeout specified by `CONNECTION_TIMEOUT` will be used.

**Return type**

`Dict[str, Any]`

**Returns**

response transfer kwargs dict.

**recv\_bytes**(*sock, buf\_len=16384*)

receive all bytes from the passed client socket instance until connection lost or line end reached.

**Parameters**

- **sock** `(socket)` – socket instance.
- **buf\_len** `(int)` – socket buffer length. if not passed then `SOCKET_BUF_LEN` will be used. pass zero/0 to use the buf length defined via the ‘buf\_len’ command line option.

**Return type**

`bytes`

**Returns**

received bytes.

**service\_factory**(*task\_id\_func=None*)

create server app instance including the command line options *bind* and *port*.

**Parameters**

**task\_id\_func** *//* (Optional[Callable[[str, str, str], str]]) – callable to return an unique id for a transfer request task.

**Return type**

*TransferServiceApp*

**Returns**

transfer service app instance.

**transfer\_kwargs\_error**(*transfer\_kwargs, err\_msg*)

add/append error to transfer kwargs dict without overwriting any previous error. :type  
 \_sphinx\_paramlinks\_ae.transfer\_service.transfer\_kwargs\_error.transfer\_kwargs: Dict[str, Any] :param  
 \_sphinx\_paramlinks\_ae.transfer\_service.transfer\_kwargs\_error.transfer\_kwargs: request/response transfer  
 kwargs dict. :type \_sphinx\_paramlinks\_ae.transfer\_service.transfer\_kwargs\_error.err\_msg: str :param  
 \_sphinx\_paramlinks\_ae.transfer\_service.transfer\_kwargs\_error.err\_msg: error message to add.

**transfer\_kwargs\_from\_literal**(*transfer\_kwargs\_lit*)

convert dict literal (created with *transfer\_kwargs\_literal()*) to transfer kwargs dict.

**Parameters**

**transfer\_kwargs\_lit** *//* (str) – request/response dict literal string.

**Return type**

Dict[str, Any]

**Returns**

transfer kwargs dict.

**transfer\_kwargs\_literal**(*transfer\_kwargs*)

convert dict to str literal to be sent via sockets (re-instantiable via *transfer\_kwargs\_from\_literal()*).

---

**Note:** to ensure security (and prevent injections) only the following basic types can be used: *int*, *float*, *boolean*, *str*, *bytes*, *list*, *tuple* and *dict*. date/time values are only allowed as dict value and of the type *datetime.datetime*; additionally the key of this dict value has to contain one of the fragments defined in *TRANSFER\_KWARGS\_DATE\_TIME\_NAME\_PARTS*.

---

**Parameters**

**transfer\_kwargs** *//* (Dict[str, Any]) – request/response transfer kwargs dict.

**Return type**

str

**Returns**

literal string of transfer kwargs dict terminated with TRANSFER\_KWARGS\_LINE\_END\_CHAR.

**transfer\_kwargs\_update**(\**variables*, \*\**kwargs*)

update multiple transfer dicts with the same keys/values (locking with *requests\_lock*).

**Parameters**

- **variables** *//* – transfer kwargs variables/dicts.
- **kwargs** *//* – kwargs to update.

**class ThreadedTCPRequestHandler**(*request, client\_address, server*)

Bases: [StreamRequestHandler](#)

server request handler.

self.rfile is a file-like object created by the handler; to use e.g. `readline()` instead of `raw recv()` likewise, self.wfile is a file-like object used to write back to the client.

**handle()**

handle a single request

**class TransferServiceApp**(*app\_title="", app\_name="", app\_version="", sys\_env\_id="", debug\_level=0, multi\_threading=False, suppress\_stdout=False, cfg\_opt\_eval\_vars=None, additional\_cfg\_files=(), cfg\_opt\_val\_stripper=None, formatter\_class=None, epilog="", \*\*logging\_params*)

Bases: [ConsoleApp](#)

server service app class

**reqs\_and\_logs:** [List](#)[[Dict](#)[[str](#), [Any](#)]]

list of transfer\_kwargs of currently processed requests

**server\_instance:** [Optional](#)[[ThreadingTCPServer](#)]

server class instance

**server\_thread:** [Optional](#)[[Thread](#)]

server thread (main or separate thread)

**cancel\_request**(*request\_kwargs, handler*)

cancel running request.

#### Parameters

- **request\_kwargs** [\(Dict\[str, Any\]\)](#) – request transfer kwargs dict, specifying via *rt\_id\_to\_cancel* the request to cancel.
- **handler** [\(StreamRequestHandler\)](#) – request handler instance.

#### Return type

[Dict](#)[[str](#), [Any](#)]

#### Returns

response kwargs, having *error* key if request could not be found/cancelled.

**static id\_of\_task**(*action, object\_type, object\_key*)

compile the id of a transfer request task.

#### Parameters

- **action** [\(str\)](#) – action or log level string.
- **object\_type** [\(str\)](#) – task object type ('log' for log entries, 'file' for file transfers, else 'message').
- **object\_key** [\(str\)](#) – task key (file path for file transfers, message string for messages and timestamp for log entries).

#### Return type

[str](#)

#### Returns

unique key identifying the task/log-entry.

**log**(*log\_level*, *message*)

print log message and add it to reqs\_and\_logs (to be read by controller app).

---

**Note:** please note that you have to use `print()` or one of the console print methods of the `AppBase` (like e.g. `verbose_out()`, respective `self.vpo`) instead of this method for the logging of low level transport methods/functions (like e.g. `pending_requests()`, `response_to_request()`, `handle()` or `recv_bytes()`). this will prevent the duplication of a log message, because each call of this method creates a new entry in `reqs_and_logs` which will be sent to the controlling app via the low level transport methods (which would recursively grow the sent messages until the system freezes), especially if the transfer kwargs are included into the log message.

---

#### Parameters

- **log\_level** (str) – ‘print’ always prints, ‘debug’ prints if `self.debug`, ‘verbose’ prints if `self.verbose`.
- **message** (str) – message to print.

**pending\_requests**(*request\_kwargs*, *handler*)

determine currently running/pending server requests and debug log messages (in debug mode only).

#### Parameters

- **request\_kwargs** (Dict[str, Any]) – request transfer kwargs dict.
- **handler** (StreamRequestHandler) – request handler class instance.

#### Return type

Dict[str, Any]

#### Returns

response transfer kwargs dict keys.

**recv\_file**(*request\_kwargs*, *handler*)

receive binary file content from client.

#### Parameters

- **request\_kwargs** (Dict[str, Any]) – request transfer kwargs dict with the following keys:
  - ‘file\_path’: file path (can contain path placeholders).
  - ‘total\_bytes’: total file length in bytes.
  - ‘series\_file’: optionally, pass any value to ensures new file name.
- **handler** (StreamRequestHandler) – request handler class instance.

#### Return type

Dict[str, Any]

#### Returns

response transfer kwargs dict keys (request kwargs extended with additional keys):

- ‘error’: error message string if an error occurred.
- ‘transferred\_bytes’: start offset on recovered transfer (previously received bytes).

**recv\_message**(*request\_kwargs*, *handler*)

receive message from client/peer.

**Return type**

`Dict[str, Any]`

**response\_to\_request**(*request\_lit*, *handler*)

process request to this server and return response string.

---

**Note:** this method is running in a separate thread (created by the server to process this request).

---

**Parameters**

- **request\_lit** `(str)` – request string, which is a dict literal with ‘*method\_name*’ key.
- **handler** `(StreamRequestHandler)` – stream request handler instance.

**Return type**

`str`

**Returns**

response string with transfer kwargs as literal. if an error occurred then the returned response kwargs contains an ‘*error*’ key which stores the related error message.

**send\_file**(*request\_kwargs*, *handler*)

send binary file content to remote server.

**Return type**

`Dict[str, Any]`

**send\_message**(*request\_kwargs*, *handler*)

send message to remote server.

**Return type**

`Dict[str, Any]`

**shutdown**(*exit\_code=0*, *timeout=None*)

overwritten to stop a running transfer service server/threads on shutdown of this app instance.

**Parameters**

- **exit\_code** `(Optional[int])` – set application OS exit code - see [shutdown\(\)](#).
- **timeout** `(Optional[float])` – timeout float value in seconds - see [shutdown\(\)](#).

**start\_server**(*threaded=False*)

start server and run until main app [stop\\_server\(\)](#).

**Parameters**

**threaded** `(bool)` – optionally pass True to use separate thread for the server process.

**Return type**

`str`

**Returns**

empty string/"" if server instance/thread got started else error message string.

**stop\_server**()

stop/pause transfer service server - callable also if not running to reset/prepare this app instance.

latest :

---

**server\_app:** *TransferServiceApp* | **None** = **None**  
transfer service server app

## 4.26 ae.sideload\_server

### 4.26.1 sideloading server

this ae namespace portion provides a simple http server to download a file to another device in the same local network, using any web browser.

by adding this module to your android app you can distribute e.g. the APK file of your app directly to another android device to install it there - without requiring an app store or an internet connection.

The implementation is inspired by the accepted answer to <https://stackoverflow.com/questions/18543640/how-would-i-create-a-python-web-server-that-downloads-a-file-on-any-get-request>.

Useful debugging tools can be found at <https://www.maketecheasier.com/transferring-files-using-python-http-server>.

#### sideloading server lifecycle

The sideloading server provided by this ae namespace portion can be started by first creating an instance of the side-loading app and then call its *start\_server()* method:

```
from ae.sideload_server import server_factory

sideloading_server_app = server_factory()
sideloading_server_app.start_server()
```

By calling *start\_server()* without arguments a default file mask (*DEFAULT\_FILE\_MASK*) will be used, which specifies the APK file of the main app situated in the *Downloads* folder of the device.

The web address that has to be entered in the web browser of the receiving device can be determined with the *server\_url()* method:

```
client_url = sideloading_server_app.server_url()
```

While the server is running the *client\_progress()* method is providing the progress state of any currently running sideloading task. More detailed info of the running sideloading process can be gathered by calling the *fetch\_log\_entries()* method.

to temporarily stop the sideloading server simple call the *stop\_server()* method. On app exit additionally call the method *shutdown()*.

---

**Hint:** The ae namespace portion *kivy\_sideload* is providing a package for an easy integration of this sideloading server into your kivy app.

An example usage of this sideloading server and the *ae.kivy\_sideload* package is integrated into the demo apps *GlsITester*, *Lisz* and *ComPartY*.

---



## Module Attributes

<code>DEFAULT_FILE_MASK</code>	default glob file mask to sideloading file
<code>FILE_COUNT_MISMATCH</code>	err msg part of <code>SideloadServerApp.start_server()</code>
<code>SERVER_BIND</code>	setting BIND to " or None to allow connections for all local devices
<code>SERVER_PORT</code>	http server listening port (a port number under 1024 requires root privileges)
<code>SOCKET_BUF_LEN</code>	buf length for socket sends in <code>response_to_request()</code>
<code>SHUTDOWN_TIMEOUT</code>	timeout(in seconds) to shut down/stop the console app and http sideloading server
<code>SideloadKwargs</code>	command/log format of sideloading requests

## Functions

<code>server_factory([task_id_func])</code>	create server app instance.
<code>update_handler_progress([handler, ...])</code>	default progress update callback.

## Classes

<code>SideloadServerApp([app_title, app_name, ...])</code>	server service app class
<code>SimpleHTTPRequestHandler(request, ...)</code>	server request handler.

**DEFAULT\_FILE\_MASK = '{downloads}/{main\_app\_name}\*.apk'**

default glob file mask to sideloading file

**FILE\_COUNT\_MISMATCH = ' files found matching '**

err msg part of `SideloadServerApp.start_server()`

**SERVER\_BIND = ''**

setting BIND to " or None to allow connections for all local devices

**SERVER\_PORT = 36996**

http server listening port (a port number under 1024 requires root privileges)

**SOCKET\_BUF\_LEN = 16384**

buf length for socket sends in `response_to_request()`

**SHUTDOWN\_TIMEOUT = 3.9**

timeout(in seconds) to shut down/stop the console app and http sideloading server

**SideloadKwargs**

command/log format of sideloading requests

alias of `Dict[str, Any]`

**server\_factory**(*task\_id\_func=None*)

create server app instance.

**Parameters**

**task\_id\_func** (Optional[Callable[[str, str, str], str]]) – callable to return an unique id for a transfer request task.

**Return type**

*SideloadServerApp*

**Returns**

sideloading server app instance.

**update\_handler\_progress**(*handler=None, transferred\_bytes=-3, total\_bytes=-3, \*\*kwargs*)

default progress update callback.

**Parameters**

- **handler** (Optional[SimpleHTTPRequestHandler]) – server request handler - containing the attributes to be updated.
- **transferred\_bytes** (int) – already transferred bytes or error code.
- **total\_bytes** (int) – total sideloading bytes.
- **kwargs** – additional/optional kwargs (e.g. client\_ip).

**class SimpleHTTPRequestHandler**(*request, client\_address, server*)

Bases: *BaseHTTPRequestHandler*

server request handler.

**progress\_total:** int = -1

total file size in bytes to be transferred

**progress\_transferred:** int = -1

transferred bytes in this sideloading progress thread

**do\_GET()**

handler callback for a http GET command request.

**log\_request**(*code='-', size='-'*)

overwrite to prevent console output if not in debug mode.

**class SideloadServerApp**(*app\_title="", app\_name="", app\_version="", sys\_env\_id="", debug\_level=0, multi\_threading=False, suppress\_stdout=False, cfg\_opt\_eval\_vars=None, additional\_cfg\_files=(), cfg\_opt\_val\_stripper=None, formatter\_class=None, epilog="", \*\*logging\_params*)

Bases: *ConsoleApp*

server service app class

**client\_handlers:** Dict[str, SimpleHTTPRequestHandler]

handler for each active client connection

**file\_path:** str

path to the found sideloading file

**load\_requests:** List[Dict[str, Any]]

sideloading requests and error/debug log

**progress\_callback**(\*\**kwargs*)

progress event callback

**server\_instance**: **Optional**[**ThreadingHTTPServer**]

server class instance

**server\_thread**: **Optional**[**Thread**]

server thread (main or separate thread)

**client\_progress**(*client\_ip*)

determine sideloading progress (transferred bytes) for the client with the passed ip address.

This method can be used alternatively to the callback *progress\_callback*.

#### Parameters

**client\_ip** (str) – client ip address.

#### Return type

**Tuple**[int, int]

#### Returns

tuple of two int values: status and file size of the sideloading file. A positive status value signifies the number of already transferred bytes. A negative value in this first int signals an error. Possibles error codes are: \* -1 if sideloading task did not start yet \* -2 if not exists or sideloading task has already finished \* -3 if the default Please note that the file size given in the second int can be 0 if an error has occurred or if the sideloading task is not yet fully created/started.

**static id\_of\_task**(*action, object\_type, object\_key*)

compile the id of the sideloading request task.

#### Parameters

- **action** (str) – action or log level string.
- **object\_type** (str) – task object type (currently only ‘log’ is implemented/supported).
- **object\_key** (str) – task key (for log entries the timestamp of the log entry is used).

#### Return type

str

#### Returns

unique key identifying the task/log-entry.

**log**(*log\_level, message*)

print log message and optionally add it to the requests (to be read by controller app).

#### Parameters

- **log\_level** (str) – ‘print’ always prints, ‘debug’ prints if *self.debug* is True and ‘verbose’ prints a message to the log if *self.verbose* is True.
- **message** (str) – message to print.

**fetch\_log\_entries**()

collect last log entries, return them and remove them from this app instance.

#### Return type

**Dict**[str, **Dict**[str, Any]]

#### Returns

**Dict**[request\_tasks\_id, sideloading\_request\_kwargs] with fetched requests log entries.

### **response\_to\_request(handler)**

process request to this server and return response string.

---

**Note:** this method may run in a separate thread (created by the server to process this request).

---

#### **Parameters**

**handler** (SimpleHTTPRequestHandler) – stream request handler instance.

### **server\_url()**

determine an url string to put into the address field of any browser to start sideloading.

#### **Return type**

str

#### **Returns**

server url string.

### **shutdown(exit\_code=0, timeout=None)**

overwritten to stop any running sideloading server instance/threads on shutdown of this app instance.

#### **Parameters**

- **exit\_code** (Optional[int]) – set application OS exit code - see [shutdown\(\)](#).
- **timeout** (Optional[float]) – timeout float value in seconds - see [shutdown\(\)](#).

### **start\_server(file\_mask="", threaded=False, progress=<function update\_handler\_progress>)**

start http file sideloading server to run until [stop\\_server\(\)](#) get called.

#### **Parameters**

- **file\_mask** (str) – optional glob.glob file mask to specify the sideloading file. If not passed then the [DEFAULT\\_FILE\\_MASK](#) will be used which specifies an APK file of the main app situated in the *Downloads* folder of the device. The sideloading server will only be started if the file mask matches exactly one file. The file path of the matched file can be accessed via the [file\\_path](#) attribute.
- **threaded** (bool) – optionally pass True to use separate thread to run the server instance.
- **progress** (Callable) – optional callback executed for each transferred/side-loaded chunk. If not specified then the default callback method [update\\_handler\\_progress\(\)](#) will be called to update the progress attributes of the request handler, which can be polled via the [client\\_progress\(\)](#) method.

#### **Return type**

str

#### **Returns**

empty string/"" if server instance/thread got started else the error message string.

---

**Note:** if [threaded](#) is *True* then the [progress](#) callback get executed in its own thread.

---

### **stop\_server()**

stop/pause sideloading server - callable also if not running to reset/prepare this app instance.

## 4.27 ae.gui\_app

### 4.27.1 base class for python applications with a graphical user interface

the abstract base class *MainAppBase* provided by this ae namespace portion allows the integration of any Python GUI framework into the ae namespace.

on overview about the available GUI-framework-specific ae namespace portion implementations can be found in the documentation of the ae namespace portion *ae.lisz\_app\_data*.

#### extended console application environment

the abstract base class *MainAppBase* inherits directly from the ae namespace class *ae console application environment class*. the so inherited helper methods are useful to log, configure and control the run-time of your GUI app via command line arguments.

---

**Hint:** please see the documentation of *config options* and *config files* in the *ae.console* namespace portion/module for more detailed information.

---

*MainAppBase* adds on top of the *ConsoleApp* the concepts of *application events*, *application status* and *application flow*, explained further down.

#### application events

the events described in this section are fired on application startup and shutdown. additional events get fired e.g. in relation to the app states (documented further down in the section *app state events*) or on start or stop of an *app tour*.

the following application events are fired exactly one time at startup in the following order:

- *on\_app\_init*: fired **after** *ConsoleApp* app instance got initialized (detected config files) and **before** the image and sound resources and app states get loaded and the GUI framework app class instance gets initialized.
- *on\_app\_run*: fired **from within** the method *run\_app()*, **after** the parsing of the command line arguments and options, and **before** all portion resources got imported.
- *on\_app\_build*: fired **after** all portion resources got loaded/imported, and **before** the framework event loop of the used GUI framework gets started.
- *on\_app\_started*: fired **after** all app initializations, and the start of and the initial processing of the framework event loop.

---

**Note:** the application events *on\_app\_build* and *on\_app\_started* have to be fired by the used GUI framework.

---



---

**Hint:** depending on the used gui framework there can be more app start events. e.g. the *ae.kivy.apps* module fires the events *on\_app\_built()* and *on\_app\_start()* (all of them fired after *on\_app\_run()* and *on\_app\_build()*). see also *kivy application events*.

---

when an application gets stopped then the following events get fired in the following order:

- *on\_app\_exit*: fired **after\* framework win got closed and just \*\*before** the event loop of the GUI framework will be stopped and the app shutdown.

latest :

---

- *on\_app\_quit*: fired **after** the event loop of the GUI framework got stopped and before the `AppBase.shutdown()` method will be called.

---

**Note:** the *on\_app\_exit* events will only be fired if the app is explicitly calling the *stop\_app()* method.

---

---

**Hint:** depending on the used gui framework there can be more events. e.g. the *apps* module fires the events *on\_app\_stop()* and clock tick later *on\_app\_stopped()* (both of them before *on\_app\_quit()* get fired). see also *kivy application events*.

---

## application status

any application- and user-specific configurations like e.g. the last window position/size, the app theme/font/language or the last selected flow within your app, could be included in the application status.

this namespace portion introduces the section *aeAppState* in the app *config files*, where any status values can be stored persistently to be recovered on the next startup of your application.

---

**Hint:** the section name *aeAppState* is declared by the *APP\_STATE\_SECTION\_NAME* constant. if you need to access this config section directly then please use this constant instead of the hardcoded section name.

---

## app state variables

this module is providing/pre-defining the following application state variables:

- *app\_state\_version*
- *flow\_id*
- *flow\_id\_ink*
- *flow\_path*
- *flow\_path\_ink*
- *font\_size*
- *lang\_code*
- *light\_theme*
- *selected\_item\_ink*
- *sound\_volume*
- *unselected\_item\_ink*
- *vibration\_volume*
- *win\_rectangle*

which app state variables are finally used by your app project is (fully data-driven) depending on the app state *config variables* detected in all the *config files* that are found/available at run-time of your app. the names of all the available application state variables can be determined with the main app helper method *app\_state\_keys()*.

if no config-file is provided then this package ensures at least the proper initialization of the following app state variables:

- *font\_size*
- *lang\_code*
- *light\_theme*
- *win\_rectangle*

if your application is e.g. supporting a user-defined font size, using the provided/pre-defined app state variable *font\_size*, then it has to call the method `change_app_state()` with the argument of *app\_state\_name* set to *font\_size* every time when the user has changed the font size of your app.

---

**Hint:** the two built-in app state variables are *flow\_id* and *flow\_path* will be explained detailed in the next section.

---

the `load_app_states()` method is called on instantiation from the implemented main app class to load the values of all app state variables from the *config files*, and is then calling `:meth:~MainAppBase.setup_app_states`` for pass them into their corresponding instance attributes.

use the main app instance attribute to read/get the actual value of a single app state variable. the actual values of all app state variables as a dict is determining the method `retrieve_app_states()`, and can be saved into the *config files* for the next app run via the method `save_app_states()` - this could be done e.g. after the app state has changed or at least on quitting the application.

always call the method `change_app_state()` to change an app state value to ensure:

- (1) the propagation to any duplicated (observable/bound) framework property and
- (2) the event notification of the related (optionally declared) main app instance method.

## app state constants

this module is also providing some pre-defined constants that can be optionally used in your application in relation to the app states data store and for the app state config variables *app\_state\_version*, *font\_size* and *light\_theme*:

- *APP\_STATE\_SECTION\_NAME*
- *APP\_STATE\_VERSION\_VAR\_NAME*
- *MIN\_FONT\_SIZE*
- *MAX\_FONT\_SIZE*
- *THEME\_LIGHT\_BACKGROUND\_COLOR*
- *THEME\_LIGHT\_FONT\_COLOR*
- *THEME\_DARK\_BACKGROUND\_COLOR*
- *THEME\_DARK\_FONT\_COLOR*

## app state events

there are three types of notification events get fired in relation to the app state variables, using the method names:

- *on\_<app\_state\_name>*: fired if the user of the app is changing the value of an app state variable.
- *on\_<app\_state\_name>\_save*: fired if an app state gets saved to the config file.
- *on\_app\_state\_version\_upgrade*: fired if the user upgrades a previously installed app to a higher version.

the method name of the app state change notification event consists of the prefix *on\_* followed by the variable name of the app state. so e.g. on a change of the *font\_size* app state the notification event *on\_font\_size* will be fired/called (if exists as a method of the main app instance). these events don't provide any event arguments.

the second event gets fired for each app state value just after the app states getting retrieved from the app class instance, and before they get stored into the main config file. the method name of this event includes also the name of the app state with the suffix *\_save*, so e.g. for the app state *flow\_id* the event method name will result in *on\_app\_state\_flow\_id\_save()*. this event is providing one event argument with the value of the app state. if the event method returns a value that is not *None* then this value will be stored/saved.

the third event gets fired on app startup when the app got upgraded to a higher version of the app state variable *APP\_STATE\_VERSION\_VAR\_NAME* (*app\_state\_version*). it will be called providing the version number for each version to upgrade, starting with the version of the previously installed main config file, until the upgrade version of the main config file get reached. so if e.g. the previously installed app state version was 3 and the new version number is 6 then this event will be fired 3 times with the argument 3, 4 and 5. it can be used e.g. to change or add app state variables or to adapt the app environment.

## application flow

to control the current state and UX flow (or context) of your application, and to persist it until the next app start, *MainBaseApp* provides two *app state variables*: *flow\_id* to store the currently working flow and *flow\_path* to store the history of nested flows.

an application flow is represented by an id string that defines three things: (1) the action to enter into the flow, (2) the data or object that gets currently worked on and (3) an optional key string that is identifying/indexing a widget or data item of your application context/flow.

---

**Note:** never concatenate a flow id string manually, use the *id\_of\_flow()* function instead.

---

the flow id is initially an empty string. as soon as the user is starting a new work flow or the current selection your application should call the method *change\_flow()* passing the flow id string into the *new\_flow\_id* argument to change the app flow.

for more complex applications you can specify a path of nested flows. this flow path gets represented by the app state variable *flow\_path*, which is a list of flow id strings.

to enter into a deeper/nested flow you simply call *change\_flow()* with one of the actions defined in *ACTIONS\_EXTENDING\_FLOW\_PATH*.

to go back to a previous flow in the flow path call *change\_flow()* passing one of the actions defined in *ACTIONS\_REDUCING\_FLOW\_PATH*.



## application flow change events

the flow actions specified by `ACTIONS_CHANGING_FLOW_WITHOUT_CONFIRMATION` don't need a flow change confirmation event handler:

- `'enter'` or `'leave'` extend/reduce the flow path.
- `'focus'` pass/change the input focus.
- `'suggest'` for autocompletion or other suggestions.

all other flow actions need to be confirmed to be changed by `change_flow()`, either by a custom flow change confirmation method/event-handler or by declaring a popup class. the name of the event handler and of the popup class gets determined from the flow id.

---

**Hint:** the name of the flow change confirmation method that gets fired when the app want to change the flow (via the method `change_flow()`) gets determined by the function `flow_change_confirmation_event_name()`, whereas the name of the popup class get determined by the function `flow_popup_class_name()`.

---

if the flow-specific change confirmation event handler does not exist or returns in a boolean `False` or `None` then `on_flow_change()` will be called. if this call also returns `False` then the action of the new flow id will be searched within `ACTIONS_CHANGING_FLOW_WITHOUT_CONFIRMATION` and if not found then the flow change will be rejected and `change_flow()` returns `False`.

if in contrary either the flow change confirmation event handler exists and does return `True` or `on_flow_change()` returns `True` or the flow action of the new flow id is in `ACTIONS_CHANGING_FLOW_WITHOUT_CONFIRMATION` then the flow id and path will be changed accordingly.

after the flow id/path change confirmation the method `change_flow()` checks if the optional `event_kwargs` key `changed_event_name` got specified and if yes then it calls this method.

finally, if a confirmed flow change results in a `'focus'` flow action then the event `on_flow_widget_focused` will be fired. this event can be used by the GUI framework to set the focus to the widget associated with the new focus flow id.

## flow actions `'open'` and `'close'`

to display an instance of a properly named popup class, simply initiate the change the app flow to an appropriate flow id (with an `'open'` flow action). in this case no change confirmation event handler is needed, because `on_flow_change()` is then automatically opening the popup.

when the popup is visible the flow path will be extended with the respective flow id.

calling the `close` method of the popup will hide it. on closing the popup the flow id will be reset and the opening flow id will be removed from the flow path.

all popup classes are providing the events `on_pre_open`, `on_open`, `on_pre_dismiss` and `on_dismiss`. the `on_dismiss` event handler can be used for data validation: returning a non-`False` value from it will cancel the close.

---

**Hint:** see the documentation of each popup class for more details on the features of popup classes (for Kivy apps e.g. `FlowDropDown`, `FlowPopup` or `FlowSelector`).

---

## key press events

to provide key press events to the applications that will use the new GUI framework you have to catch the key press events of the framework, convert/normalize them and then call the `key_press_from_framework()` with the normalized modifiers and key args.

the `modifiers` arg is a string that can contain several of the following sub-strings, always in the alphabetic order (like listed below):

- Alt
- Ctrl
- Meta
- Shift

the `key` arg is a string that is specifying the last pressed key. if the key is not representing a single character but a command key, then `key` will be one of the following strings:

- escape
- tab
- backspace
- enter
- del
- enter
- up
- down
- right
- left
- home
- end
- pgup
- pgdown

on call of `key_press_from_framework()` this method will try to dispatch the key press event to your application. first it will check the app instance if it has declared a method with the name `on_key_press_of_<modifiers>_<key>` and if so it will call this method.

if this method does return False (or any other value resulting in False) then method `key_press_from_framework()` will check for a method with the same name in lower-case and if exists it will call this method.

if also the second method does return False, then it will try to call the event method `on_key_press` of the app instance (if exists) with the modifiers and the key as arguments.

if the `on_key_press` method does also return False then `key_press_from_framework()` will finally pass the key press event to the original key press handler of the GUI framework for further processing.

## integrate new gui framework

to integrate a new Python GUI framework you have to declare a new class that inherits from `MainAppBase` and implements at least the abstract method `init_app()`.

additionally and to load the resources of the app (after the portions resources got loaded) the event `on_app_build` has to be fired, executing the `MainAppBase.on_app_build()` method. this could be done directly from within `init_app()` or by redirecting one of the events of the app instance of the GUI framework.

a minimal implementation of the `init_app()` method would look like the following:

```
def init_app(self):
    self.call_method('on_app_build')
    return None, None
```

most GUI frameworks are providing classes that need to be instantiated on application startup, like e.g. the instance of the GUI framework app class, the root widget or layout of the main GUI framework window(s). to keep a reference to these instances within your main app class you can use the attributes `framework_app`, `framework_root` and `framework_win` of the class `MainAppBase`.

the initialization of the attributes `framework_app`, `framework_root` and `framework_win` is optional and can be done e.g. within `init_app()` or in the `on_app_build` application event fired later by the framework app instance.

---

**Note:** if `framework_win` is set to a window instance, then the window instance has to provide a `close` method, which will be called automatically by the `stop_app()`.

---

a typical implementation of a framework-specific main app class looks like:

```
from new_gui_framework import NewFrameworkApp, MainWindowClassOfNewFramework

class NewFrameworkMainApp(MainAppBase):
    def init_app(self):
        self.framework_app = NewFrameworkAppClass()
        self.framework_win = MainWindowClassOfNewFramework()

        # return callables to start/stop the event loop of the GUI framework
        return self.framework_app.start, self.framework_app.stop
```

in this example the `on_app_build` application event gets fired either from within the `start` method of the framework app instance or by an event provided by the GUI framework.

`init_app()` will be executed only once at the main app class instantiation. only the main app instance has to initialize the GUI framework to prepare the app startup and has to return at least a callable to start the event loop of the GUI framework.

---

**Hint:** although not recommended because of possible namespace conflicts, one could e.g. alternatively integrate the framework application class as a mixin to the main app class.

---

to initiate the app startup the `run_app()` method has to be called from the main module of your app project. `run_app()` will then start the GUI event loop by calling the first method that got returned by `init_app()`.

**latest :**

---

## optional configuration and extension

most of the base implementation helper methods can be overwritten by either the inheriting framework portion or directly by user main app class.

## base resources for your gui app

this portion is also providing base resources for commonly used images and sounds.

the image file resources provided by this portion are taken from:

- [iconmonstr](#).

the sound files provides by this portion are taken from:

- [Erokia](#) at [freesound.org](#).
- [plasterbrain](#) at [freesound.org](#).

---

**Hint:** the i18n translation texts of this module are provided by the ae namespace portion [ae.gui\\_help](#).

---

TODO: implement OS-independent detection of dark/light screen mode and automatic notification on day/night mode switch. - see <https://github.com/albertosottile/darkdetect> for MacOS, MSWindows and Ubuntu - see <https://github.com/kvdroid/Kvdroid/blob/master/kvdroid/tools/darkmode.py> for Android

## Module Attributes

<i>APP_STATE_SECTION_NAME</i>	config section name to store app state
<i>APP_STATE_VERSION_VAR_NAME</i>	config variable name to store the current application state version
<i>COLOR_BLACK</i>	!= 0/1 to differentiate from framework pure black/white colors
<i>THEME_DARK_BACKGROUND_COLOR</i>	dark theme background color in rgba(0.0 .
<i>THEME_DARK_FONT_COLOR</i>	dark theme font color in rgba(0.0 .
<i>THEME_LIGHT_BACKGROUND_COLOR</i>	light theme background color in rgba(0.0 .
<i>THEME_LIGHT_FONT_COLOR</i>	light theme font color in rgba(0.0 .
<i>MIN_FONT_SIZE</i>	minimum (s.a.
<i>MAX_FONT_SIZE</i>	
<i>ACTIONS_EXTENDING_FLOW_PATH</i>	flow actions that are extending the flow path.
<i>ACTIONS_REDUCING_FLOW_PATH</i>	flow actions that are shrinking/reducing the flow paths.
<i>ACTIONS_CHANGING_FLOW_WITHOUT_CONFIRMATION</i>	flow actions that are processed without the need to be confirmed.
<i>FLOW_KEY_SEP</i>	separator character between flow action/object and flow key
<i>FLOW_ACTION_RE</i>	regular expression detecting invalid characters in flow action string
<i>FLOW_OBJECT_RE</i>	regular expression detecting invalid characters in flow object string
<i>HIDDEN_GLOBALS</i>	tuple of global/module variable names that are hidden in <i>global_variables()</i>
<i>PORTIONS_IMAGES</i>	register of image files found in portions/packages at import time
<i>PORTIONS_SOUNDS</i>	register of sound files found in portions/packages at import time
<i>USER_NAME_MAX_LEN</i>	maximal length of a username/id (restricting <i>user_id</i> )
<i>AppStateType</i>	app state config variable type
<i>EventKwargsType</i>	change flow event kwargs type
<i>ColorRGB</i>	color red, green and blue parts
<i>ColorRGBA</i>	ink is rgb color and alpha
<i>ColorOrInk</i>	color or ink type

## Functions

<i>ellipse_polar_radius</i> (ell_a, ell_b, radian)	calculate the radius from polar for the given ellipse and radian.
<i>ensure_tap_kwarg_refs</i> (init_kwarg, tap_widget)	ensure that the passed widget.__init__ kwarg dict contains a reference to itself within kwarg['tap_kwarg'].
<i>flow_action</i> (flow_id)	determine the action string of a flow_id.
<i>flow_action_split</i> (flow_id)	split flow id string into action part and the rest.
<i>flow_change_confirmation_event_name</i> (flow_id)	determine the name of the event method for the change confirmation of the passed flow_id.
<i>flow_class_name</i> (flow_id, name_suffix)	determine class name for the given flow id and class name suffix.
<i>flow_key</i> (flow_id)	return the key of a flow id.
<i>flow_key_split</i> (flow_id)	split flow id into action with object and flow key.
<i>flow_object</i> (flow_id)	determine the object string of the passed flow_id.
<i>flow_path_id</i> (flow_path[, path_index])	determine the flow id of the newest/last entry in the flow_path.
<i>flow_path_strip</i> (flow_path)	return copy of passed flow_path with all non-enter actions stripped from the end.
<i>flow_popup_class_name</i> (flow_id)	determine name of the Popup class for the given flow id.
<i>id_of_flow</i> (action[, obj, key])	create flow id string.
<i>register_package_images</i> ()	call from module scope of the package to register/add image/img resources path.
<i>register_package_sounds</i> ()	call from module scope of the package to register/add sound file resources.
<i>replace_flow_action</i> (flow_id, new_action)	replace action in given flow id.
<i>update_tap_kwarg</i> (widget[, popup_kwarg])	update or simulate widget's tap_kwarg property and return the updated dictionary (for kv rule of tap_kwarg).

## Classes

<i>MainAppBase</i> (**console_app_kwarg)	abstract base class to implement a GUIApp-conform app class
--	---

**APP\_STATE\_SECTION\_NAME** = 'aeAppState'

config section name to store app state

**APP\_STATE\_VERSION\_VAR\_NAME** = 'app\_state\_version'

config variable name to store the current application state version

**COLOR\_BLACK** = [0.009, 0.006, 0.003, 1.0]

!= 0/1 to differentiate from framework pure black/white colors

**THEME\_DARK\_BACKGROUND\_COLOR** = [0.009, 0.006, 0.003, 1.0]

dark theme background color in rgba(0.0 ... 1.0)

**THEME\_DARK\_FONT\_COLOR** = [0.999, 0.996, 0.993, 1.0]

dark theme font color in rgba(0.0 ... 1.0)

**THEME\_LIGHT\_BACKGROUND\_COLOR** = [0.999, 0.996, 0.993, 1.0]

light theme background color in rgba(0.0 ... 1.0)

---

```
THEME_LIGHT_FONT_COLOR = [0.009, 0.006, 0.003, 1.0]
```

```
    light theme font color in rgba(0.0 ... 1.0)
```

```
MIN_FONT_SIZE = 15.0
```

```
    minimum (s.a. min_font_size) and
```

```
MAX_FONT_SIZE = 99.0
```

```
ACTIONS_EXTENDING_FLOW_PATH = ['add', 'confirm', 'edit', 'enter', 'open', 'show',
    'suggest']
```

```
    flow actions that are extending the flow path.
```

```
ACTIONS_REDUCING_FLOW_PATH = ['close', 'leave']
```

```
    flow actions that are shrinking/reducing the flow paths.
```

```
ACTIONS_CHANGING_FLOW_WITHOUT_CONFIRMATION = ['', 'enter', 'focus', 'leave', 'suggest']
```

```
    flow actions that are processed without the need to be confirmed.
```

```
FLOW_KEY_SEP = ':'
```

```
    separator character between flow action/object and flow key
```

```
FLOW_ACTION_RE = re.compile('[a-z0-9]+')
```

```
    regular expression detecting invalid characters in flow action string
```

```
FLOW_OBJECT_RE = re.compile('[A-Za-z0-9_]+')
```

```
    regular expression detecting invalid characters in flow object string
```

```
HIDDEN_GLOBALS = ('ABC', 'abstractmethod', '_add_base_globals', 'Any', '__builtins__',
    '__cached__', 'Callable', '_d_', 'Dict', '__doc__', '__file__', 'List', '__loader__',
    'module_globals', '__name__', 'Optional', '__package__', '__path__', '__spec__', 'Tuple',
    'Type', '__version__')
```

```
    tuple of global/module variable names that are hidden in global_variables()
```

latest :

---

```
PORTIONS_IMAGES = {'add_item': [RegisteredFile('/home/docs/checkouts/readthedocs.org/
user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/img/add_item.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/add_item.png')], 'close_popup':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/close_popup.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/close_popup.png')], 'copy_node':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/copy_node.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/copy_node.png')], 'cut_node':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/cut_node.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/cut_node.png')], 'delete_item':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/delete_item.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/delete_item.png')], 'drag_item':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/drag_item.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/drag_item.png')], 'edit_font_size':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/enaml_app/img/edit_font_size.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/enaml_app/img/light_1/edit_font_size.png')], 'edit_item':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/edit_item.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/edit_item.png')], 'enter_item':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/enter_item.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/enter_item.png')], 'export_node':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/export_node.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/export_node.png')], 'filter_off':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/filter_off.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/filter_off.png')], 'filter_on':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/filter_on.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/filter_on.png')], 'font_size':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/kivy_user_prefs/img/light_1/font_size.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/kivy_user_prefs/img/font_size.png')], 'help_circled':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/help_circled.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/help_circled.png')], 'help_icon':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/help_icon.png'),
RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/
python3.9/site-packages/ae/gui_app/img/light_1/help_icon.png')], 'icon_view':
```



register of image files found in portions/packages at import time

```
PORTIONS_SOUNDS = {'added': [RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/added.wav')],
'debug_draw': [RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/debug_draw.wav')], 'debug_save':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/debug_save.wav')], 'deleted':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/deleted.wav')], 'edited':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/edited.wav')], 'enter_item':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/enter_item.wav')], 'error':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/error.wav')], 'filter_off':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/filter_off.wav')], 'filter_on':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/filter_on.wav')], 'leave_item':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/leave_item.wav')], 'touched':
[RegisteredFile('/home/docs/checkouts/readthedocs.org/user_builds/ae/envs/latest/lib/python3.9/site-packages/ae/gui_app/snd/touched.wav')]]}
```

register of sound files found in portions/packages at import time

**USER\_NAME\_MAX\_LEN** = 12

maximal length of a username/id (restricting `user_id`)

### AppStateType

app state config variable type

alias of `Dict[str, Any]`

### EventKwargsType

change flow event kwargs type

alias of `Dict[str, Any]`

### ColorRGB

color red, green and blue parts

alias of `Union[Tuple[float, float, float], List[float]]`

### ColorRGBA

ink is rgb color and alpha

alias of `Union[Tuple[float, float, float, float], List[float]]`

### ColorOrInk

color or ink type

alias of `Union[Tuple[float, float, float], List[float], Tuple[float, float, float, float]]`

### ellipse\_polar\_radius(*ell\_a*, *ell\_b*, *radian*)

calculate the radius from polar for the given ellipse and radian.

### Parameters

- `ell_a` (float) – ellipse x-radius.
- `ell_b` (float) – ellipse y-radius.
- `radian` (float) – radian of angle.

**Return type**

float

**Returns**ellipse radius at the angle specified by `radian`.**ensure\_tap\_kwargs\_refs**(*init\_kwargs*, *tap\_widget*)ensure that the passed `widget.__init__` kwargs dict contains a reference to itself within `kwargs['tap_kwargs']`.**Parameters**

- `init_kwargs` (Dict[str, Any]) – kwargs of the widgets `__init__` method.
- `tap_widget` (Any) – reference to the tap widget.

this alternative version is only 10 % faster but much less clean than the current implementation:

```
if 'tap_kwargs' not in init_kwargs:
    init_kwargs['tap_kwargs'] = {}
tap_kwargs = init_kwargs['tap_kwargs']

if 'tap_widget' not in tap_kwargs:
    tap_kwargs['tap_widget'] = tap_widget

if 'popup_kwargs' not in tap_kwargs:
    tap_kwargs['popup_kwargs'] = {}
popup_kwargs = tap_kwargs['popup_kwargs']
if 'opener' not in popup_kwargs:
    popup_kwargs['opener'] = tap_kwargs['tap_widget']
```

**flow\_action**(*flow\_id*)determine the action string of a `flow_id`.**Parameters**`flow_id` (str) – flow id.**Return type**

str

**Returns**

flow action string.

**flow\_action\_split**(*flow\_id*)

split flow id string into action part and the rest.

**Parameters**`flow_id` (str) – flow id.**Return type**

Tuple[str, str]

**Returns**

tuple of (flow action string, flow obj and key string)

**flow\_change\_confirmation\_event\_name**(*flow\_id*)

determine the name of the event method for the change confirmation of the passed *flow\_id*.

**Parameters**

**flow\_id** (str) – flow id.

**Return type**

str

**Returns**

tuple with 2 items containing the flow action and the object name (and id).

**flow\_class\_name**(*flow\_id*, *name\_suffix*)

determine class name for the given flow id and class name suffix.

**Parameters**

- **flow\_id** (str) – flow id.
- **name\_suffix** (str) – class name suffix.

**Return type**

str

**Returns**

name of the class. please note that the flow action *open* will not be added to the returned class name.

**flow\_key**(*flow\_id*)

return the key of a flow id.

**Parameters**

**flow\_id** (str) – flow id string.

**Return type**

str

**Returns**

flow key string.

**flow\_key\_split**(*flow\_id*)

split flow id into action with object and flow key.

**Parameters**

**flow\_id** (str) – flow id to split.

**Return type**

Tuple[str, str]

**Returns**

tuple of (flow action and object string, flow key string).

**flow\_object**(*flow\_id*)

determine the object string of the passed *flow\_id*.

**Parameters**

**flow\_id** (str) – flow id.

**Return type**

str

**Returns**

flow object string.

**flow\_path\_id**(*flow\_path*, *path\_index=-1*)

determine the flow id of the newest/last entry in the flow\_path.

**Parameters**

- **flow\_path** (List[str]) – flow path to get the flow id from.
- **path\_index** (int) – index in the flow\_path.

**Return type**

str

**Returns**

flow id string or empty string if flow path is empty or index does not exist.

**flow\_path\_strip**(*flow\_path*)

return copy of passed flow\_path with all non-enter actions stripped from the end.

**Parameters****flow\_path** (List[str]) – flow path list to strip.**Return type**

List[str]

**Returns**

stripped flow path copy.

**flow\_popup\_class\_name**(*flow\_id*)

determine name of the Popup class for the given flow id.

**Parameters****flow\_id** (str) – flow id.**Return type**

str

**Returns**name of the Popup class. please note that the action *open* will not be added to the returned class name.**id\_of\_flow**(*action*, *obj=""*, *key=""*)

create flow id string.

**Parameters**

- **action** (str) – flow action string.
- **obj** (str) – flow object (defined by app project).
- **key** (str) – flow index/item\_id/field\_id/... (defined by app project).

**Return type**

str

**Returns**

complete flow\_id string.

**register\_package\_images**()

call from module scope of the package to register/add image/img resources path.

no parameters needed because we use here `stack_var()` helper function to determine the module file path via the `__file__` module variable of the caller module in the call stack. in this call we have to overwrite the default value (`SKIPPED_MODULES`) of the `skip_modules` parameter to not skip ae portions that are providing package

resources and are listed in the `SKIPPED_MODULES`, like e.g. `ae.gui_app` and `ae.gui_help` (passing empty string "" to overwrite default skip list).

### **register\_package\_sounds()**

call from module scope of the package to register/add sound file resources.

no parameters needed because we use here `stack_var()` helper function to determine the module file path via the `__file__` module variable of the caller module in the call stack. in this call we have to overwrite the default value (`SKIPPED_MODULES`) of the `skip_modules` parameter to not skip ae portions that are providing package resources and are listed in the `SKIPPED_MODULES`, like e.g. `ae.gui_app` `ae.gui_help` (passing empty string "" to overwrite default skip list).

### **replace\_flow\_action(flow\_id, new\_action)**

replace action in given flow id.

#### **Parameters**

- **flow\_id** (str) – flow id.
- **new\_action** (str) – action to be set/replaced within passed flow id.

#### **Returns**

flow id with new action and object/key from passed flow id.

### **update\_tap\_kwargs(widget, popup\_kwargs=None, \*\*tap\_kwargs)**

update or simulate widget's tap\_kwargs property and return the updated dictionary (for kv rule of tap\_kwargs).

#### **Parameters**

- **widget** (Any) – widget with tap\_kwargs property to be updated.
- **popup\_kwargs** (Optional[Dict[str, Any]]) – dict with items to update popup\_kwargs key of tap\_kwargs
- **tap\_kwargs** – additional tap\_kwargs items to update.

#### **Return type**

Dict[str, Any]

#### **Returns**

### **class MainAppBase(\*\*console\_app\_kwargs)**

Bases: `ConsoleApp, ABC`

abstract base class to implement a GUIApp-conform app class

**app\_state\_version:** int = 0

version number of the app state variables in <config>.ini

**flow\_id:** str = ''

id of the current app flow (entered by the app user)

**flow\_id\_ink:** Union[Tuple[float, float, float], List[float], Tuple[float, float, float, float]] = [0.99, 0.99, 0.69, 0.69]

rgba color for flow id / drag&drop node placeholder

**flow\_path\_ink:** Union[Tuple[float, float, float], List[float], Tuple[float, float, float, float]] = [0.99, 0.99, 0.39, 0.48]

rgba color for flow\_path/drag&drop item placeholder

**font\_size:** float = 21.0

font size used toolbar and flow screens

---

```

lang_code:  str = ''
    optional language code (e.g. 'es_ES' for Spanish)

light_theme:  bool = False
    True=light theme/background, False=dark theme

selected_item_ink:  Union[Tuple[float, float, float], List[float], Tuple[float,
float, float, float]] = [0.69, 1.0, 0.39, 0.18]
    rgba color for list items (selected)

unselected_item_ink:  Union[Tuple[float, float, float], List[float], Tuple[float,
float, float, float]] = [0.39, 0.39, 0.39, 0.18]
    rgba color for list items (unselected)

sound_volume:  float = 0.12
    sound volume of current app (0.0=mute, 1.0=max)

vibration_volume:  float = 0.3
    vibration volume of current app (0.0=mute, 1.0=max)

win_rectangle:  tuple = (0, 0, 1920, 1080)
    window coordinates (x, y, width, height)

framework_app:  Any = None
    app class instance of the used GUI framework

framework_win:  Any = None
    window instance of the used GUI framework

framework_root:  Any = None
    app root layout widget

image_files:  Optional[FilesRegister] = None
    image/icon files

sound_files:  Optional[FilesRegister] = None
    sound/audio files

__init__(**console_app_kwargs)
    create instance of app class.

    Parameters
    console_app_kwargs – kwargs to be passed to the __init__ method of ConsoleApp.

_exit_code:  int
    init by stop_app() and passed onto OS by run_app()

_last_focus_flow_id:  str
    id of the last valid focused window/widget/item/context

flow_path:  List[str]
    list of flow ids, reflecting recent user actions

_start_event_loop:  Optional[Callable]
    callable to start event loop of GUI framework

_stop_event_loop:  Optional[Callable]
    callable to start event loop of GUI framework

```

**\_init\_default\_user\_cfg\_vars()**

init user default config variables.

override this method to add module-/app-specific config vars that can be set individually per user.

**abstract init\_app**(*framework\_app\_class=None*)

initialize framework app instance and root window/layout, return GUI event loop start/stop methods.

**Parameters**

**framework\_app\_class** *(Optional[Any])* – class to create app instance (optionally extended by app project).

**Return type**

*Tuple[Optional[Callable], Optional[Callable]]*

**Returns**

tuple of two callable, the 1st to start and the 2nd to stop/exit the GUI event loop.

**app\_state\_keys()**

determine current config variable names/keys of the app state section *APP\_STATE\_SECTION\_NAME*.

**Return type**

*Tuple[str, ...]*

**Returns**

tuple of all app state item keys (config variable names).

**static backup\_config\_resources()**

backup config files and image/sound/translations resources to {ado}<now\_str>.

config files are collected from {ado}, {usr} or {cwd} (the first found file name only - see/sync-with *ae.console.ConsoleApp.add\_cfg\_files()*).

resources are copied from {ado} or {cwd} (only the first found resources root path).

**Return type**

*str*

**change\_app\_state**(*app\_state\_name, state\_value, send\_event=True, old\_name=""*)

change app state to *state\_value* in self.<app\_state\_name> and app\_states dict.

**Parameters**

- **app\_state\_name** *(str)* – name of the app state to change.
- **state\_value** *(Any)* – new value of the app state to change.
- **send\_event** *(bool)* – pass False to prevent send/call of the main\_app.on\_<app\_state\_name> event.
- **old\_name** *(str)* – pass to add state to the main config file: old state name to rename/migrate or *UNSET* to only add a new app state variable with the name specified in *app\_state\_name*.

**change\_observable**(*name, value, is\_app\_state=False*)

change observable attribute/member/property in framework\_app instance (and shadow copy in main app).

**Parameters**

- **name** *(str)* – name of the observable attribute/member or key of an observable dict property.
- **value** *(Any)* – new value of the observable.

- **is\_app\_state** (bool) – pass True for an app state observable.

**change\_flow**(*new\_flow\_id*, *\*\*event\_kwargs*)

try to change/switch the current flow id to the value passed in *new\_flow\_id*.

#### Parameters

- **new\_flow\_id** (str) – new flow id (maybe overwritten by flow change confirmation event handlers by assigning a flow id to event\_kwargs['flow\_id']).
- **event\_kwargs** – optional args to pass additional data or info onto and from the flow change confirmation event handler.

the following keys are currently supported/implemented by this module/portion (additional keys can be added by the modules/apps using this method):

- *changed\_event\_name*: optional main app event method name to be called if the flow got confirmed and changed.
- *count*: optional number used to render a pluralized help text for this flow change (this number gets also passed to the help text formatter by/in *change\_flow()*).
- *edit\_widget*: optional widget instance for edit/input.
- *flow\_id*: process *flow\_path* as specified by the *new\_flow\_id* argument, but then overwrite this flow id with this event arg value to set *flow\_id*.
- *popup\_kwargs*: optional dict passed to the Popup *\_\_init\_\_* method, like e.g. dict(opener=opener\_widget\_of\_popup, data=...).
- *popups\_to\_close*: optional sequence of widgets to be closed by this method after flow change got confirmed.
- 'reset\_last\_focus\_flow\_id': pass *True* to reset the last focus flow id, pass *False* or *None* to ignore the last focus id (and not use to set flow id) or pass a flow id string value to change the last focus flow id to the passed value.
- *tap\_widget*: optional tapped button widget instance (initiating this flow change).

some of these keys get specified directly on the call of this method, e.g. via *tap\_kwargs* or *tap\_kwargs*, where others get added by the flow change confirmation handlers/callbacks.

#### Return type

bool

#### Returns

True if flow got confirmed by a declared custom flow change confirmation event handler (either event method or Popup class) of the app and changed accordingly, else False.

some flow actions are handled internally independent of the return value of a custom event handler, like e.g. 'enter' or 'leave' will always extend or reduce the flow path and the action 'focus' will give the indexed widget the input focus (these exceptions are configurable via *ACTIONS\_CHANGING\_FLOW\_WITHOUT\_CONFIRMATION*).

**static class\_by\_name**(*class\_name*)

search class name in framework modules as well as in app main.py to return class object.

#### Parameters

**class\_name** (str) – name of the class.

#### Return type

Optional[Type]



**Returns**

class object with the specified class name or *UNSET* if not found.

**static dpi\_factor()**

dpi scaling factor - override if the used GUI framework supports dpi scaling.

**Return type**

*float*

**close\_popups(classes=())**

close all opened popups (starting with the foremost popup).

**Parameters**

**classes** *(tuple)* – optional class filter - if not passed then only the first foremost widgets underneath the app win with an *open* method will be closed. pass tuple to restrict found popup widgets to certain classes. like e.g. by passing (Popup, DropDown, FlowPopup) to get all popups of an app (in Kivy use Factory.WidgetClass if widget is declared only in kv lang).

**find\_image(image\_name, height=32.0, light\_theme=True)**

find best fitting image in img app folder (see also *img\_file()* for easier usage).

**Parameters**

- **image\_name** *(str)* – name of the image (file name without extension).
- **height** *(float)* – preferred height of the image/icon.
- **light\_theme** *(bool)* – preferred theme (dark/light) of the image.

**Return type**

*Optional[RegisteredFile]*

**Returns**

image file object (RegisteredFile/CachedFile) if found else None.

**find\_sound(sound\_name)**

find sound by name.

**Parameters**

**sound\_name** *(str)* – name of the sound to search for.

**Return type**

*Optional[RegisteredFile]*

**Returns**

cached sound file object (RegisteredFile/CachedFile) if sound name was found else None.

**find\_widget(match)**

search the widget tree returning the first matching widget in reversed z-order (top-/foremost first).

**Parameters**

**match** *(Callable[[Any], bool])* – callable called with the widget as argument, returning True if widget matches.

**Return type**

*Optional[Any]*

**Returns**

first found widget in reversed z-order (top-most widget first).

**flow\_path\_action**(*flow\_path=None, path\_index=-1*)

determine the action of the last (newest) entry in the *flow\_path*.

**Parameters**

- **flow\_path** (Optional[List[str]]) – optional flow path to get the flow action from (default=self.flow\_path).
- **path\_index** (int) – optional index in the *flow\_path* (default=-1).

**Return type**

str

**Returns**

flow action string or empty string if flow path is empty or index does not exist.

**global\_variables**(\*\**patches*)

determine generic/most-needed global variables to evaluate expressions/macros.

**Parameters**

**patches** – dict of variable names and values to add/replace on top of generic globals.

**Return type**

Dict[str, Any]

**Returns**

dict of global variables patched with *patches*.

**img\_file**(*image\_name, font\_size=None, light\_theme=None*)

shortcutting *find\_image()* method w/o bound property to get image file path.

**Parameters**

- **image\_name** (str) – image name (file name stem).
- **font\_size** (Optional[float]) – optional font size in pixels.
- **light\_theme** (Optional[bool]) – optional theme (True=light, False=dark).

**Return type**

str

**Returns**

file path of image file or empty string if image file not found.

**key\_press\_from\_framework**(*modifiers, key*)

dispatch key press event, coming normalized from the UI framework.

**Parameters**

- **modifiers** (str) – modifier keys.
- **key** (str) – key character.

**Return type**

bool

**Returns**

True if key got consumed/used else False.

**load\_app\_states**()

load application state from the config files to prepare app.run\_app

**load\_images()**

load images from app folder img.

**load\_sounds()**

load audio sounds from app folder snd.

**load\_translations(*lang\_code*)**

load translation texts for the passed language code.

**Parameters**

**lang\_code** (str) – the new language code to be set (passed as flow key). empty on first app run/start.

**mix\_background\_ink()**

remix background ink if one of the basic back colours change.

**on\_app\_build()**

default/fallback flow change confirmation event handler.

**on\_app\_exit()**

default/fallback flow change confirmation event handler.

**on\_app\_init()**

default/fallback flow change confirmation event handler.

**on\_app\_quit()**

default/fallback flow change confirmation event handler.

**on\_app\_run()**

default/fallback flow change confirmation event handler.

**on\_app\_started()**

app initialization event - the last one on app startup.

**on\_debug\_level\_change(*level\_name*, *\_event\_kwargs*)**

debug level app state change flow change confirmation event handler.

**Parameters**

- **level\_name** (str) – the new debug level name to be set (passed as flow key).
- **\_event\_kwargs** (Dict[str, Any]) – unused event kwargs.

**Return type**

bool

**Returns**

True to confirm the debug level change.

**on\_flow\_change(*flow\_id*, *event\_kwargs*)**

checking if exists a Popup class for the new flow and if yes then open it.

**Parameters**

- **flow\_id** (str) – new flow id.
- **event\_kwargs** (Dict[str, Any]) – optional event kwargs; the optional item with the key *popup\_kwargs* will be passed onto the `__init__` method of the found Popup class.

**Return type**

bool

**Returns**

True if Popup class was found and displayed.

this method is mainly used as the last fallback clicked flow change confirmation event handler of a Flow-Button.

**on\_flow\_id\_ink()**

redirect flow id back ink app state color change event handler to actualize mixed\_back\_ink.

**on\_flow\_path\_ink()**

redirect flow path back ink app state color change event handler to actualize mixed\_back\_ink.

**static on\_flow\_popup\_close(*\_flow\_key*, *\_event\_kwargs*)**

default popup close handler of FlowPopup widget, ensuring update of *flow\_path*.

**Parameters**

- ***\_flow\_key*** *(str)* – unused flow key.
- ***\_event\_kwargs*** *(Dict[str, Any])* – unused popup args.

**Return type**

*bool*

**Returns**

always returning True.

**on\_lang\_code\_change(*lang\_code*, *\_event\_kwargs*)**

language app state change flow change confirmation event handler.

**Parameters**

- ***lang\_code*** *(str)* – the new language code to be set (passed as flow key). empty on first app run/start.
- ***\_event\_kwargs*** *(Dict[str, Any])* – unused event kwargs.

**Return type**

*bool*

**Returns**

True to confirm the language change.

**on\_light\_theme\_change(*\_flow\_key*, *event\_kwargs*)**

app theme app state change flow change confirmation event handler.

**Parameters**

- ***\_flow\_key*** *(str)* – flow key.
- ***event\_kwargs*** *(Dict[str, Any])* – event kwargs with key 'light\_theme' containing True|False for light|dark theme.

**Return type**

*bool*

**Returns**

True to confirm change of flow id.

**on\_selected\_item\_ink()**

redirect selected item back ink app state color change event handler to actualize mixed\_back\_ink.

**on\_unselected\_item\_ink()**

redirect unselected item back ink app state color change event handler to actualize mixed\_back\_ink.

**on\_user\_add(*user\_name*, *event\_kwargs*)**

called on close of UserNameEditorPopup to check user input and create/register the current os user.

**Parameters**

- **user\_name** (str) – new user id.
- **event\_kwargs** (Dict[str, Any]) – optionally pass True in the *unique\_user\_name* key to prevent duplicate usernames.

**Return type**

bool

**Returns**

True if user got registered else False.

**play\_beep()**

make a short beep sound, should be overwritten by GUI framework.

**play\_sound(*sound\_name*)**

play audio/sound file, should be overwritten by GUI framework.

**Parameters**

**sound\_name** (str) – name of the sound to play.

**play\_vibrate(*pattern*=(0.0, 0.09, 0.21, 0.3, 0.09, 0.09, 0.21, 0.09))**

play vibrate pattern, should be overwritten by GUI framework.

**Parameters**

**pattern** (Tuple) – optional tuple of pause and vibrate time sequence - use error pattern if not passed.

**open\_popup(*popup\_class*, *\*\*popup\_kwargs*)**

open Popup/DropDown, calling the *open/show* method of the instance created from the passed popup class.

**Parameters**

- **popup\_class** (Type) – class of the Popup/DropDown widget/window.
- **popup\_kwargs** – args to instantiate and show/open the popup.

**Return type**

Any

**Returns**

created and displayed/opened popup class instance.

---

**Hint:** overwrite this method if framework is using different method to open popup window or if a widget in the Popup/DropDown need to get the input focus.

---

**popups\_opened(*classes*=( ))**

determine all popup-like container widgets that are currently opened.

**Parameters**

**classes** (Tuple) – optional class filter - if not passed then only the widgets underneath win/root with an *open* method will be added. pass tuple to restrict found popup widgets to certain classes. like e.g. by passing (Popup, DropDown, FlowPopup) to get all popups of an ae/Kivy app (in Kivy use Factory.WidgetClass if widget is declared only in kv lang).

**Return type**`List`**Returns**

list of the foremost opened/visible popup class instances (children of the app window), matching the classes or having an *open* method, ordered by their z-coordinate (most front widget first).

**retrieve\_app\_states()**

determine the state of a running app from the main app instance and return it as dict.

**Return type**`Dict[str, Any]`**Returns**

dict with all app states available in the config files.

**run\_app()**

startup main and framework applications.

**save\_app\_states()**

save app state in config file.

**Return type**`str`**Returns**

empty string if app status could be saved into config files else error message.

**setup\_app\_states(app\_states)**

put app state variables into main app instance to prepare framework app.run\_app.

**Parameters**

**app\_states** *¶* (`Dict[str, Any]`) – dict of app states.

**show\_message(message, title="", is\_error=True)**

display (error) message popup to the user.

**Parameters**

- **message** *¶* (`str`) – message string to display.
- **title** *¶* (`str`) – title of message box.
- **is\_error** *¶* (`bool`) – pass False to not emit error tone/vibration.

**stop\_app(exit\_code=0)**

quit this application.

**Parameters**

**exit\_code** *¶* (`int`) – optional exit code.

**upgraded\_config\_app\_state\_version()**

determine app state version of an app upgrade.

**Return type**`int`**Returns**

value of app state variable APP\_STATE\_VERSION\_VAR\_NAME if the app got upgraded (and has a config file from a previous app installation), else 0.

**widget\_by\_attribute**(*att\_name*, *att\_value*)

determine the first (top-most) widget having the passed attribute name and value.

**Parameters**

- **att\_name** (str) – name of the attribute of the searched widget.
- **att\_value** (str) – attribute value of the searched widget.

**Return type**

Optional[Any]

**Returns**

widget that has the specified attribute with the specified value or None if not found.

**widget\_by\_flow\_id**(*flow\_id*)

determine the first (top-most) widget having the passed flow\_id.

**Parameters**

**flow\_id** (str) – flow id value of the searched widget's *tap\_flow\_id*/*focus\_flow\_id* attribute.

**Return type**

Optional[Any]

**Returns**

widget that has a *tap\_flow\_id*/*focus\_flow\_id* attribute with the value of the passed flow id or None if not found.

**widget\_by\_app\_state\_name**(*app\_state\_name*)

determine the first (top-most) widget having the passed app state name (*app\_state\_name*).

**Parameters**

**app\_state\_name** (str) – app state name of the widget's *app\_state\_name* attribute.

**Return type**

Optional[Any]

**Returns**

widget that has a *app\_state\_name* attribute with the passed app state name or None if not found.

**widget\_children**(*wid*, *only\_visible=False*)

determine the children of widget or its container (if exists) in z-order (top-/foremost first).

**Parameters**

- **wid** (Any) – widget to determine the children from.
- **only\_visible** (bool) – pass True to only return visible widgets.

**Return type**

List

**Returns**

list of children widgets of the passed widget.

**static widget\_pos**(*wid*)

return the absolute window x and y position of the passed widget.

**Parameters**

**wid** – widget to determine the position of.

**Return type**

Tuple[float, float]

**Returns**

tuple of x and y screen/window coordinate.

**widgets\_enclosing\_rectangle(widgets)**

calculate the minimum bounding rectangle all the passed widgets.

**Parameters**

**widgets** *¶* (`Union[list, tuple]`) – list/tuple of widgets to determine the minimum bounding rectangle for.

**Return type**

`Tuple[float, float, float, float]`

**Returns**

tuple of floats with the x, y, width, height values of the bounding rectangle.

**static widget\_size(wid)**

return the size (width and height) in pixels of the passed widget.

**Parameters**

**wid** *¶* – widget to determine the size of.

**Return type**

`Tuple[float, float]`

**Returns**

tuple of width and height in pixels.

**static widget\_visible(wid)**

determine if the passed widget is visible (has width and height and (visibility or opacity) set).

**Parameters**

**wid** *¶* (`Any`) – widget to determine visibility of.

**Return type**

`bool`

**Returns**

True if widget is visible (or visibility cannot be determined), False if hidden.

**win\_pos\_size\_change(\*win\_pos\_size)**

screen resize handler called on window resize or when app will exit/stop via closed event.

**Parameters**

**win\_pos\_size** *¶* – window geometry/coordinates: x, y, width, height.

## 4.28 ae.gui\_help

### 4.28.1 main app base class with context help for flow and app state changes

the class `HelpAppBase` provided by this namespace portion is extending your application with a context-sensitive help functionality.

the data-driven approach allows ad-hoc-changes of your app's help texts without the need of code changes or recompilation. this gets achieved within `HelpAppBase` by overriding the main app class methods `change_flow()` and `change_app_state()`.

so to add help support to the widgets of your app you only need to add/provide the help texts with a help id that is matching the value of the `help_id` attribute of the widget you need help for.



additionally, you can provide a separate i18n translation message file for each of the supported languages to make your help texts multilingual.

### help layout implementation example

*HelpAppBase* inherits from *MainAppBase* while still being independent of the used GUI framework/library.

**Note:** the user interface for this help system has to be provided externally on top of this module. it can either be implemented directly in your app project or in a separate framework-specific module.

use *HelpAppBase* as base class of the GUI framework specific main application class and implement the abstract methods *init\_app()* and *ensure\_top\_most\_z\_index()*:

```
from ae.gui_help import HelpAppBase

class MyMainApp(HelpAppBase):
    def init_app(self, framework_app_class=None):
        self.framework_app = framework_app_class()
        ...
        return self.framework_app.run, self.framework_app.stop

    def ensure_top_most_z_index(self, widget):
        framework_method_to_push_widget_to_top_most(widget)
        ...
```

to activate the help mode the widget to display the help texts have to be assigned to the main app attribute *help\_layout* and to the framework app property *help\_layout* via the *change\_observable()* method:

```
main_app.change_observable('help_layout', HelpScreenContainerOrWindow())
```

the *help\_layout* property is also used as a flag of the help mode activity. by assigning *None* to these attributes the help mode will be deactivated:

```
main_app.change_observable('help_layout', None)
```

use the attribute *help\_activator* to provide and store a widget that allows the user to toggle the help mode activation. the *help\_display()* is using it as fallback widget if no help target (or widget to be explained) got found.

**Hint:** the de-/activation method *help\_activation\_toggle()* together with the classes *HelpBehavior*, *HelpToggler* and *Tooltip* are demonstrating a typical implementation of help activator and help text tooltip widgets.

## additional helper functions

the helper functions `anchor_layout_x()`, `anchor_layout_y()`, `anchor_points()` and `anchor_spec()` are calculating framework-independent the position and direction of the targeting tooltip anchor arrow and its layout box.

## flow change context message id

the message id to identify the help texts for each flow button is composed by the `id_of_flow_help()`, using the prefix marker string defined by the module variable `FLOW_HELP_ID_PREFIX` followed by the flow id of the flow widget.

for example the message id for a flow button with the flow action `'open'`, the object `'item'` and the (optional) flow key `'123456'` is resulting in the following help text message id:

```
'help_flow#open_item:123456'
```

if there is no need for a detailed message id that is taking the flow key into account, then simply create a help text message id without the flow key.

the method `help_display()` does first search for a message id including the flow key in the available help text files and if not found it will automatically fall back to use a message id without the flow key:

```
'help_flow#open_item'
```

**Hint:** more information regarding the flow id you find in the doc string of the module `ae.gui_app` in the section *application flow*.

## application state change context message id

the message ids for app state change help texts are using the prefix marker string defined by the module variable `APP_STATE_HELP_ID_PREFIX`, followed by the name of the app state and are composed via the method `id_of_state_help()`.

## pluralize-able help texts

each message id can optionally have several help texts for their pluralization. for that simply add a `count` item to the `help_vars` property of the help target widget and then define a help text for the all the possible count cases in your message text file like shown in the following example:

```
{
    'message_id': {
        'zero':      "help text if {count} == 0",    # {count} will be_
↪replaced with `0`
        'one':      "help text if count == 1",
        'many':     "help text if count > 1",
        'negative': "help text if count < 0",
        '':         "fallback help text if count is None",
    },
    ...
}
```

the provided *count* value can also be included/displayed in the help text, like shown in the ‘zero’ count case of the example.

### pre- and post-change help texts

to display a different help message before and after the change of the flow id or the app state define a message dict with the keys ‘’ (an empty string) and ‘*after*’ like shown in the following example:

```
{
    'message_id': {
        '': "help text displayed before the flow/app-state change.",
        'after': "help text displayed after the flow/app-state change",
    },
    ...
}
```

if you want to move/change the help target to another widget after a change, then use instead of ‘*after*’ the ‘*next\_help\_id*’ message dict key:

```
{
    'message_id': {
        '': "help text before the change",
        'next_help_id': "help_flow#next_flow_id",
    },
    ...
}
```

in this case the help target will automatically change to the widget specified by the flow id in the ‘*next\_help\_id*’ key, if the user was tapping the second time on the first/initial help target widget.

### i18n help texts

the displayed help messages related to the message id will automatically get translated into the default language of the current system/environment.

the declaration and association of message ids and their related help messages is done with the help of the namespace portion *ae.i18n*.

---

**Hint:** more details on these and other features of this help system, e.g. the usage of f-strings in the help texts, is documented in the doc string of the *ae.i18n* module.

a more complex example app demonstrating the features of this context help system can be found in the repository of the [kivy lizs demo](#) app.

---

## 4.28.2 app tours

the following classes provided by this portion building a solid fundament to implement tours for your app:

- *TourBase* - abstract base class of all app tours.
- *TourDropdownFromButton* - abstract base class for tours on dropdowns.
- *OnboardingTour* - minimal app onboarding tour, extendable with app specific tour pages.
- *UserPreferencesTour* - minimal user preferences dropdown tour.

### app tour start and stop events

the following main app event methods get called (if they exist) in relation to the start/stop of an app tour:

- *on\_tour\_init*: fired when the app tour instance got initialized and the app states backup got saved.
- *on\_tour\_start*: fired after tour start() method get called; delay id configurable via *tour\_start\_delay* page data.
- *on\_tour\_exit*: fired after an app tour got finished and the app states got restored to the values of the tour start. fired delayed letting UI events get processed; delay seconds configurable via the *tour\_exit\_delay* page data value.

### UI-specific implementation

to complete the implementation of the app tours, the UI-specific framework has to provide a tour layout class, which is highlighting the widget explained and to display a tooltip and the tour page texts.

---

**Hint:** the *TourOverlay* class is a quite complete implementation of a tour layout class.

---

optionally for the ‘user\_registration’ page of the *OnboardingTour* an open username editor flow has to be implemented.

### Module Attributes

<i>CLOSE_POPUP_FLOW_ID</i>	flow id to close opened dropdown/popup
<i>APP_STATE_HELP_ID_PREFIX</i>	message id prefix for app state change help texts
<i>FLOW_HELP_ID_PREFIX</i>	message id prefix for flow change help texts
<i>TOUR_PAGE_HELP_ID_PREFIX</i>	message id prefix of tour page text/dict
<i>TOUR_START_DELAY_DEF</i>	default value of tour start delay in seconds
<i>TOUR_EXIT_DELAY_DEF</i>	default value of tour exit delay in seconds
<i>IGNORED_HELP_FLOWS</i>	tuple of flow ids never search/show help text for
<i>IGNORED_HELP_STATES</i>	tuple of app state names never searched help for
<i>REGISTERED_TOURS</i>	map(name: class) of all registered tour classes
<i>AnchorSpecType</i>	(see return value of <i>anchor_spec()</i> )
<i>ExplainedMatcherType</i>	single explained widget matcher type

## Functions

<i>anchor_layout_x</i> (anchor_spe, layout_width, ...)	calculate x position of the layout box of an anchor.
<i>anchor_layout_y</i> (anchor_spe, layout_height, ...)	calculate y position of the layout box of an anchor.
<i>anchor_points</i> (font_size, anchor_spe)	recalculate points of the anchor triangle drawing.
<i>anchor_spec</i> (wid_x, wid_y, wid_width, ...)	calculate anchor center pos (x, y) and anchor direction to the targeted widget.
<i>help_id_tour_class</i> (help_id)	determine the tour class if passed help id has attached tour pages.
<i>help_sub_id</i> (help_id)	determine sub id (flow id, tour id or app state name) of the current/specified/passed help id.
<i>id_of_flow_help</i> (flow_id)	compose help id for specified flow id.
<i>id_of_state_help</i> (app_state_name)	compose help id for app state name/key.
<i>id_of_tour_help</i> (page_id)	compose help id for specified tour page id.
<i>register_tour_class</i> (tour_class)	register app tour class.
<i>tour_help_translation</i> (page_id)	determine help translation for the passed page id (flow id or app state name).
<i>tour_id_class</i> (tour_id)	determine the tour class of the passed tour id.
<i>translation_short_help_id</i> (help_id)	check if a help text exists for the passed help id.
<i>widget_page_id</i> (wid)	determine tour page id of passed widget.

## Classes

<i>HelpAppBase</i> (**console_app_kwargs)	main app help base class.
<i>OnboardingTour</i> (main_app)	onboarding tour for first app start.
<i>TourBase</i> (main_app)	abstract tour base class, automatically registering sub-classes as app tours.
<i>TourDropdownFromButton</i> (main_app)	generic tour base class to auto-explain a dropdown menu, starting with the button opening the dropdown.
<i>UserPreferencesTour</i> (main_app)	user preferences menu tour.

**CLOSE\_POPUP\_FLOW\_ID** = 'close\_flow\_popup'

flow id to close opened dropdown/popup

**APP\_STATE\_HELP\_ID\_PREFIX** = 'help\_app\_state#'

message id prefix for app state change help texts

**FLOW\_HELP\_ID\_PREFIX** = 'help\_flow#'

message id prefix for flow change help texts

**TOUR\_PAGE\_HELP\_ID\_PREFIX** = 'tour\_page#'

message id prefix of tour page text/dict

**TOUR\_START\_DELAY\_DEF** = 0.15

default value of tour start delay in seconds

**TOUR\_EXIT\_DELAY\_DEF** = 0.45

default value of tour exit delay in seconds

**IGNORED\_HELP\_FLOWS** = ('close\_flow\_popup',)

tuple of flow ids never search/show help text for

```
IGNORED_HELP_STATES = ('flow_id', 'flow_path', 'win_rectangle')
```

tuple of app state names never searched help for

```
REGISTERED TOURS = {'AnimatedOnboardingTour': <class
'ae.kivy.tours.AnimatedOnboardingTour'>, 'OnboardingTour': <class
'ae.gui_help.OnboardingTour'>, 'SideloadMenuTour': <class
'ae.kivy_sideload.SideloadMenuTour'>, 'TourDropdownFromButton': <class
'ae.gui_help.TourDropdownFromButton'>, 'UserPreferencesTour': <class
'ae.gui_help.UserPreferencesTour'>}
```

map(name: class) of all registered tour classes

### AnchorSpecType

(see return value of [anchor\\_spec\(\)](#))

alias of `Tuple[float, float, str]`

### ExplainedMatcherType

single explained widget matcher type

alias of `Union[Callable[[Any], bool], str]`

```
anchor_layout_x(anchor_spe, layout_width, win_width)
```

calculate x position of the layout box of an anchor.

#### Parameters

- **anchor\_spe** `(Tuple[float, float, str])` – [AnchorSpecType](#) instance ([anchor\\_spec\(\)](#) return) with anchor position/direction.
- **layout\_width** `(float)` – anchor layout width.
- **win\_width** `(float)` – app window width.

#### Return type

`float`

#### Returns

absolute x coordinate within the app window of anchor layout.

```
anchor_layout_y(anchor_spe, layout_height, win_height)
```

calculate y position of the layout box of an anchor.

#### Parameters

- **anchor\_spe** `(Tuple[float, float, str])` – [AnchorSpecType](#) tuple with anchor position and direction.
- **layout\_height** `(float)` – anchor layout height.
- **win\_height** `(float)` – app window height.

#### Return type

`float`

#### Returns

absolute y coordinate in the app window of anchor layout.

```
anchor_points(font_size, anchor_spe)
```

recalculate points of the anchor triangle drawing.

#### Parameters

- **font\_size** (float) – font\_size to calculate size (radius == hypotenuse / 2) of the anchor triangle.
- **anchor\_spec** (Tuple[float, float, str]) – anchor specification tuple: x/y coordinates and direction - see [anchor\\_spec\(\)](#) return.

**Return type**

Tuple[float, ...]

**Returns**

tuple of the three x and y coordinates of the anchor triangle edges.

**anchor\_spec**(wid\_x, wid\_y, wid\_width, wid\_height, win\_width, win\_height)

calculate anchor center pos (x, y) and anchor direction to the targeted widget.

**Parameters**

- **wid\_x** (float) – absolute x coordinate in app window of targeted widget.
- **wid\_y** (float) – absolute y coordinate in app window of targeted widget.
- **wid\_width** (float) – width of targeted widget.
- **wid\_height** (float) – height of targeted widget.
- **win\_width** (float) – app window width.
- **win\_height** (float) – app window height.

**Return type**

Tuple[float, float, str]

**Returns**

tooltip anchor specification tuple ([AnchorSpecType](#)) with the three items:

- anchor\_x (anchor center absolute x coordinate in window),
- anchor\_y (anchor center absolute y coordinate in window) and
- anchor\_dir (anchor direction: 'r'=right, 'i'=increase-y, 'l'=left, 'd'=decrease-y)

---

**Note:** the direction in the y-axis got named increase for higher y values and *decrease* for lower y values to support different coordinate systems of the GUI frameworks.

e.g. Kivy has the y-axis zero value at the bottom of the app window, whereas in enaml/Qt it is at the top.

---

**help\_id\_tour\_class**(help\_id)

determine the tour class if passed help id has attached tour pages.

**Parameters**

**help\_id** (str) – help id to determine the tour class from.

**Return type**

Optional[Any]

**Returns**

tour class of an existing tour for the passed help id or None if no associated tour exists.

**help\_sub\_id**(help\_id)

determine sub id (flow id, tour id or app state name) of the current/specified/passed help id.

opposite of [id\\_of\\_flow\\_help\(\)](#) / [id\\_of\\_state\\_help\(\)](#) / [id\\_of\\_tour\\_help\(\)](#).

**Parameters**

**help\_id** (str) – help id to extract the sub id from.

**Return type**

str

**Returns**

flow id, tour id, app state name or empty string if help id does not contain a sub id.

**id\_of\_flow\_help**(flow\_id)

compose help id for specified flow id.

**Parameters**

**flow\_id** (str) – flow id to make help id for.

**Return type**

str

**Returns**

help id for the specified flow id.

**id\_of\_state\_help**(app\_state\_name)

compose help id for app state name/key.

**Parameters**

**app\_state\_name** (str) – name of the app state variable.

**Return type**

str

**Returns**

help id for the specified app state.

**id\_of\_tour\_help**(page\_id)

compose help id for specified tour page id.

**Parameters**

**page\_id** (str) – tour page id to make help id for.

**Return type**

str

**Returns**

help id for the specified tour page.

**register\_tour\_class**(tour\_class)

register app tour class.

**Parameters**

**tour\_class** (Type[TourBase]) – tour class to register.

**tour\_help\_translation**(page\_id)

determine help translation for the passed page id (flow id or app state name).

**Parameters**

**page\_id** (str) – tour page id.

**Return type**

Union[str, Dict[str, str], None]

**Returns**

help translation text/dict (if exists) or None if translation not found.



**tour\_id\_class**(*tour\_id*)

determine the tour class of the passed tour id.

**Parameters**

**tour\_id** (str) – tour/flow id to determine tour class for.

**Return type**

Optional[Any]

**Returns**

tour class of an existing tour for the passed tour id or None if no tour exists.

**translation\_short\_help\_id**(*help\_id*)

check if a help text exists for the passed help id.

**Parameters**

**help\_id** (str) – help id to check if a translation/help texts exists.

**Return type**

Tuple[Union[str, Dict[str, str], None], str]

**Returns**

tuple of translation text/dict (if exists) and maybe shortened help id(removed detail) or tuple of (None, help\_id) if translation not found.

**widget\_page\_id**(*wid*)

determine tour page id of passed widget.

**Parameters**

**wid** (Optional[Any]) – widget to determine tour page id from (can be None).

**Return type**

str

**Returns**

tour page id or empty string if widget has no page id or is None.

**class TourBase**(*main\_app*)

Bases: object

abstract tour base class, automatically registering subclasses as app tours.

subclass this generic, UI-framework-independent base class to bundle pages of a tour and make sure that the attr:~*TourBase.page\_ids* and *page\_data* attributes are correctly set. a UI-framework-dependent tour overlay/layout instance, created and assigned to *main\_app.tour\_layout*, will automatically create an instance of your tour-specific subclass on tour start.

**classmethod \_\_init\_subclass\_\_**(\*\*kwargs)

register tour class; called on declaration of tour subclass.

**\_\_init\_\_**(*main\_app*)

**auto\_switch\_pages:** Union[bool, int]

enable/disable automatic switch of tour pages.

set to *True*, *1* or *-1* to automatically switch tour pages; *True* and *1* will switch to the next page until the last page is reached, while *-1* will switch back to the previous pages until the first page is reached; *-1* and *1* automatically toggles at the first/last page the to other value (endless ping-pong until back/next button gets pressed by the user).

the seconds to display each page before switching to the next one can be specified via the item value of the the dict *page\_data* dict with the key '*next\_page\_delay*'.

**page\_data:** `Dict[str, Any]`

additional/optional help variables (in *help\_vars* key), tour and page text/layout/timing settings.

the class attribute values are default values for all tour pages and get individually overwritten for each tour page by the i18n translations attributes on tour page change via *load\_page\_data()*.

supported/implemented dict keys:

- *app\_flow\_delay*: time in seconds to wait until app flow change is completed (def=1.2, >0.9 for auto-width).
- *back\_text*: caption of tour previous page button (def=get\_text('back')).
- *fade\_out\_app*: set to 0.0 to prevent the fade out of the app screen (def=1.0).
- *help\_vars*: additional help variables, e.g. *help\_translation* providing context help translation dict/text.
- *next\_text*: caption of tour next page button (def=get\_text('next')).
- *next\_page\_delay*: time in seconds to read the current page before next request\_auto\_page\_switch() (def=9.6).
- *page\_update\_delay*: time in seconds to wait until tour layout/overlay is completed (def=0.9).
- *tip\_text* or "" (empty string): tour page tooltip text fstring message text template. alternatively put as first character a '=' character followed by a tour page flow id to initialize the tip\_text to the help translation text of the related flow widget, and the *self* help variable to the related flow widget instance.
- *tour\_start\_delay*: seconds between tour.start() and on\_tour\_start main app event (def=TOUR\_START\_DELAY\_DEF).
- *tour\_exit\_delay*: seconds between tour.stop() and the on\_tour\_exit main app event (def=TOUR\_EXIT\_DELAY\_DEF).

**pages\_explained\_matchers:** `Dict[str, Union[Callable[[Any], bool], str], Tuple[Union[Callable[[Any], bool], str], ...]]`

matchers (specified as callable or id-string) to determine the explained widget(s) of each tour page.

each key of this dict is a tour page id (for which the explained widget(s) will be determined).

the value of each dict item is a matcher or a tuple of matchers. each matcher specifies a widget to be explained/targeted/highlighted. for matcher tuples the minimum rectangle enclosing all widgets get highlighted.

the types of matchers, to identify any visible widget, are:

- *find\_widget()* matcher callable (scanning framework\_win.children)
- evaluation expression resulting in *find\_widget()* matcher callable
- widget id string, declared via kv lang, identifying widget in framework\_root.ids
- page id string, compiled from widgets app state/flow/focus via *widget\_page\_id()* to identify widget

**page\_ids:** `List[str]`

list of tour page ids, either initialized via this class attribute or dynamically.

**page\_idx:** `int`

index of the current tour page (in *page\_ids*)

**last\_page\_idx:** `Optional[int]`

last tour page index (*None* on tour start)

**main\_app**

shortcut to main app instance

**layout**

tour overlay layout instance

**top\_popup**

top most popup widget (in app simulation)

**backup\_app\_states()**

backup current states of this app, including flow.

**cancel\_auto\_page\_switch\_request** (*reset=True*)

cancel auto switch callback if requested, called e.g. from tour layout/overlay next/back buttons.

**cancel\_delayed\_setup\_layout\_call()**

cancel delayed setup layout call request.

**property last\_page\_id:** `str` | `None`

determine last displayed tour page id.

**load\_page\_data()**

load page before switching to it (and maybe reload after preparing app flow and before setup of layout).

**next\_page()**

switch to next tour page.

**prev\_page()**

switch to previous tour page.

**request\_auto\_page\_switch()**

initiate automatic switch to next tour page.

**restore\_app\_states()**

restore app states of this app - saved via [\*backup\\_app\\_states\(\)\*](#).

**setup\_app\_flow()**

setup app flow and load page data to prepare a tour page.

**setup\_explained\_widget()**

determine and set the explained widget for the actual tour page.

**Return type**

`list`

**Returns**

list of explained widget instances.

**setup\_layout()**

setup/prepare tour overlay/layout after switch of tour page.

**setup\_texts()**

setup texts in tour layout from `page_data`.

**start()**

prepare app tour start.

**stop()**

stop/cancel tour.

**teardown\_app\_flow()**

restore app flow and app states before tour finishing or before preparing/switching to prev/next page.

**update\_page\_ids()**

update/change page ids on app flow setup (before tour page loading and the tour overlay/layout setup).

override this method to dynamically change the page\_ids in a running tour. after adding/removing a page the attribute values of *last\_page\_idx* and *page\_idx* have to be corrected accordingly.

**class TourDropdownFromButton(main\_app)**

Bases: *TourBase*

generic tour base class to auto-explain a dropdown menu, starting with the button opening the dropdown.

**determine\_page\_ids** = *'\_v\_'*

**setup\_app\_flow()**

manage the opening state of the dropdown (open dropdown, only close it if opening button get explained).

**setup\_layout()**

prepare layout for all tour pages - first page explains opening dropdown button.

**\_saved\_app\_states**: *Dict[str, Any]*

**auto\_switch\_pages**: *Union[bool, int]*

enable/disable automatic switch of tour pages.

set to *True*, *1* or *-1* to automatically switch tour pages; *True* and *1* will switch to the next page until the last page is reached, while *-1* will switch back to the previous pages until the first page is reached; *-1* and *1* automatically toggles at the first/last page the to other value (endless ping-pong until back/next button gets pressed by the user).

the seconds to display each page before switching to the next one can be specified via the item value of the the dict *page\_data* dict with the key *'next\_page\_delay'*.

**page\_data**: *Dict[str, Any]*

additional/optional help variables (in *help\_vars* key), tour and page text/layout/timing settings.

the class attribute values are default values for all tour pages and get individually overwritten for each tour page by the *il8n* translations attributes on tour page change via *load\_page\_data()*.

supported/implemented dict keys:

- *app\_flow\_delay*: time in seconds to wait until app flow change is completed (def=1.2, >0.9 for auto-width).
- *back\_text*: caption of tour previous page button (def=get\_text('back')).
- *fade\_out\_app*: set to 0.0 to prevent the fade out of the app screen (def=1.0).
- *help\_vars*: additional help variables, e.g. *help\_translation* providing context help translation dict/text.
- *next\_text*: caption of tour next page button (def=get\_text('next')).
- *next\_page\_delay*: time in seconds to read the current page before next request\_auto\_page\_switch() (def=9.6).
- *page\_update\_delay*: time in seconds to wait until tour layout/overlay is completed (def=0.9).
- *tip\_text* or "" (empty string): tour page tooltip text fstring message text template. alternatively put as first character a '=' character followed by a tour page flow id to initialize the tip\_text to the help translation text of the related flow widget, and the *self* help variable to the related flow widget instance.

- *tour\_start\_delay*: seconds between `tour.start()` and `on_tour_start` main app event (def=TOUR\_START\_DELAY\_DEF).
- *tour\_exit\_delay*: seconds between `tour.stop()` and the `on_tour_exit` main app event (def=TOUR\_EXIT\_DELAY\_DEF).

**pages\_explained\_matchers:** `Dict[str, Union[Callable[[Any], bool], str], Tuple[Union[Callable[[Any], bool], str], ...]]`

matchers (specified as callable or id-string) to determine the explained widget(s) of each tour page.

each key of this dict is a tour page id (for which the explained widget(s) will be determined).

the value of each dict item is a matcher or a tuple of matchers. each matcher specifies a widget to be explained/targeted/highlighted. for matcher tuples the minimum rectangle enclosing all widgets get highlighted.

the types of matchers, to identify any visible widget, are:

- *find\_widget()* matcher callable (scanning `framework_win.children`)
- evaluation expression resulting in *find\_widget()* matcher callable
- widget id string, declared via kv lang, identifying widget in `framework_root.ids`
- page id string, compiled from widgets app state/flow/focus via *widget\_page\_id()* to identify widget

**page\_ids:** `List[str]`

list of tour page ids, either initialized via this class attribute or dynamically.

**page\_idx:** `int`

index of the current tour page (in *page\_ids*)

**last\_page\_idx:** `Optional[int]`

last tour page index (*None* on tour start)

**class OnboardingTour**(*main\_app*)

Bases: *TourBase*

onboarding tour for first app start.

**\_\_init\_\_**(*main\_app*)

overridden to handle onboarding tour starts since app installation.

**page\_idx:** `int`

index of the current tour page (in *page\_ids*)

**setup\_app\_flow()**

overridden to open user preferences dropdown in *responsible\_layout* tour page.

**teardown\_app\_flow()**

overridden to close the opened user preferences dropdown on leaving *layout\_font\_size* tour page.

**update\_page\_ids()**

overridden to remove 2nd-/well-done-page (only showing once on next-page-jump from 1st-/welcome-page).

**\_saved\_app\_states:** `Dict[str, Any]`

**auto\_switch\_pages:** `Union[bool, int]`

enable/disable automatic switch of tour pages.

set to *True*, *1* or *-1* to automatically switch tour pages; *True* and *1* will switch to the next page until the last page is reached, while *-1* will switch back to the previous pages until the first page is reached; *-1* and *1* automatically toggles at the first/last page the to other value (endless ping-pong until back/next button gets pressed by the user).

the seconds to display each page before switching to the next one can be specified via the item value of the the dict `page_data` dict with the key `'next_page_delay'`.

**page\_data:** `Dict[str, Any]`

additional/optional help variables (in `help_vars` key), tour and page text/layout/timing settings.

the class attribute values are default values for all tour pages and get individually overwritten for each tour page by the `i18n` translations attributes on tour page change via `load_page_data()`.

supported/implemented dict keys:

- `app_flow_delay`: time in seconds to wait until app flow change is completed (def=1.2, >0.9 for auto-width).
- `back_text`: caption of tour previous page button (def=get\_text('back')).
- `fade_out_app`: set to 0.0 to prevent the fade out of the app screen (def=1.0).
- `help_vars`: additional help variables, e.g. `help_translation` providing context help translation dict/text.
- `next_text`: caption of tour next page button (def=get\_text('next')).
- `next_page_delay`: time in seconds to read the current page before next request\_auto\_page\_switch() (def=9.6).
- `page_update_delay`: time in seconds to wait until tour layout/overlay is completed (def=0.9).
- `tip_text` or "" (empty string): tour page tooltip text fstring message text template. alternatively put as first character a '=' character followed by a tour page flow id to initialize the `tip_text` to the help translation text of the related flow widget, and the `self` help variable to the related flow widget instance.
- `tour_start_delay`: seconds between `tour.start()` and `on_tour_start` main app event (def=TOUR\_START\_DELAY\_DEF).
- `tour_exit_delay`: seconds between `tour.stop()` and the `on_tour_exit` main app event (def=TOUR\_EXIT\_DELAY\_DEF).

**pages\_explained\_matchers:** `Dict[str, Union[Callable[[Any], bool], str, Tuple[Union[Callable[[Any], bool], str], ...]]]`

matchers (specified as callable or id-string) to determine the explained widget(s) of each tour page.

each key of this dict is a tour page id (for which the explained widget(s) will be determined).

the value of each dict item is a matcher or a tuple of matchers. each matcher specifies a widget to be explained/targeted/highlighted. for matcher tuples the minimum rectangle enclosing all widgets get highlighted.

the types of matchers, to identify any visible widget, are:

- `find_widget()` matcher callable (scanning `framework_win.children`)
- evaluation expression resulting in `find_widget()` matcher callable
- widget id string, declared via kv lang, identifying widget in `framework_root.ids`
- page id string, compiled from widgets app state/flow/focus via `widget_page_id()` to identify widget

**page\_ids:** `List[str]`

list of tour page ids, either initialized via this class attribute or dynamically.

**last\_page\_idx:** `Optional[int]`

last tour page index (*None* on tour start)

**class** `UserPreferencesTour(main_app)`

Bases: `TourDropdownFromButton`

user preferences menu tour.

**\_\_init\_\_**(`main_app`)

**auto\_switch\_pages:** `Union[bool, int]`

enable/disable automatic switch of tour pages.

set to *True*, *1* or *-1* to automatically switch tour pages; *True* and *1* will switch to the next page until the last page is reached, while *-1* will switch back to the previous pages until the first page is reached; *-1* and *1* automatically toggles at the first/last page the to other value (endless ping-pong until back/next button gets pressed by the user).

the seconds to display each page before switching to the next one can be specified via the item value of the the dict `page_data` dict with the key `'next_page_delay'`.

**\_saved\_app\_states:** `Dict[str, Any]`

**page\_data:** `Dict[str, Any]`

additional/optional help variables (in `help_vars` key), tour and page text/layout/timing settings.

the class attribute values are default values for all tour pages and get individually overwritten for each tour page by the i18n translations attributes on tour page change via `load_page_data()`.

supported/implemented dict keys:

- `app_flow_delay`: time in seconds to wait until app flow change is completed (def=1.2, >0.9 for auto-width).
- `back_text`: caption of tour previous page button (def=get\_text('back')).
- `fade_out_app`: set to 0.0 to prevent the fade out of the app screen (def=1.0).
- `help_vars`: additional help variables, e.g. `help_translation` providing context help translation dict/text.
- `next_text`: caption of tour next page button (def=get\_text('next')).
- `next_page_delay`: time in seconds to read the current page before next request\_auto\_page\_switch() (def=9.6).
- `page_update_delay`: time in seconds to wait until tour layout/overlay is completed (def=0.9).
- `tip_text` or "" (empty string): tour page tooltip text fstring message text template. alternatively put as first character a '=' character followed by a tour page flow id to initialize the `tip_text` to the help translation text of the related flow widget, and the `self` help variable to the related flow widget instance.
- `tour_start_delay`: seconds between `tour.start()` and `on_tour_start` main app event (def=TOUR\_START\_DELAY\_DEF).
- `tour_exit_delay`: seconds between `tour.stop()` and the `on_tour_exit` main app event (def=TOUR\_EXIT\_DELAY\_DEF).

```
pages_explained_matchers: Dict[str, Union[Callable[[Any], bool], str],
    Tuple[Union[Callable[[Any], bool], str], ...]]
```

matchers (specified as callable or id-string) to determine the explained widget(s) of each tour page.

each key of this dict is a tour page id (for which the explained widget(s) will be determined).

the value of each dict item is a matcher or a tuple of matchers. each matcher specifies a widget to be explained/targeted/highlighted. for matcher tuples the minimum rectangle enclosing all widgets get highlighted.

the types of matchers, to identify any visible widget, are:

- `find_widget()` matcher callable (scanning `framework_win.children`)
- evaluation expression resulting in `find_widget()` matcher callable
- widget id string, declared via kv lang, identifying widget in `framework_root.ids`
- page id string, compiled from widgets app state/flow/focus via `widget_page_id()` to identify widget

```
page_ids: List[str]
```

list of tour page ids, either initialized via this class attribute or dynamically.

```
page_idx: int
```

index of the current tour page (in `page_ids`)

```
last_page_idx: Optional[int]
```

last tour page index (*None* on tour start)

```
class HelpAppBase(**console_app_kwargs)
```

Bases: `MainAppBase`, `ABC`

main app help base class.

```
displayed_help_id: str = ''
```

message id of currently explained/focused target widget in help mode

```
help_activator: Any = None
```

help mode de-/activator button widget

```
help_layout: Optional[Any] = None
```

help text container widget in active help mode else *None*

```
tour_layout: Optional[Any] = None
```

tour layout/overlay widget in active tour mode else *None*

```
tour_overlay_class: Optional[Type] = None
```

UI-framework-specific tour overlay class, set by main app subclass

```
_next_help_id: str = ''
```

last app-state/flow change to show help text on help mode activation

```
_closing_popup_open_flow_id: str = ''
```

flow id of just closed popup

```
abstract call_method_delayed(delay, callback, *args, **kwargs)
```

delayed call of passed callable/method with args/kwargs catching and logging exceptions preventing app exit.

#### Parameters



- **delay** (float) – delay in seconds before calling the callable/method specified by [callback](#).
- **callback** (Union[Callable, str]) – either callable or name of the main app method to call.
- **args** – args passed to the callable/main-app-method to be called.
- **kwargs** – kwargs passed to the callable/main-app-method to be called.

**Return type**[Any](#)**Returns**

delayed call event object instance, providing a *cancel* method to allow the cancellation of the delayed call within the delay time.

**abstract ensure\_top\_most\_z\_index(widget)**

ensure visibility of the passed widget to be the top most in the z index/order.

**Parameters**

**widget** ([Any](#)) – the popup/dropdown/container widget to be moved to the top.

**abstract help\_activation\_toggle()**

button tapped event handler to switch help mode between active and inactive (also inactivating tour).

**change\_app\_state(app\_state\_name, state\_value, send\_event=True, old\_name="")**

change app state via [change\\_app\\_state\(\)](#), show help text in active help mode.

all parameters are documented in the overwritten method [change\\_app\\_state\(\)](#).

**change\_flow(new\_flow\_id, \*\*event\_kwargs)**

change/switch flow id - overriding [change\\_flow\(\)](#).

more detailed documentation of the parameters you find in the overwritten method [change\\_app\\_state\(\)](#).

this method returns True if flow changed and got confirmed by a declared custom event handler (either event method or Popup class) of the app, if the help mode is *not* active or the calling widget is selected in active help mode, else False.

**Return type**[bool](#)**help\_app\_state\_display(help\_vars, changed=False)**

actualize the help layout if active, before and after the change of the app state.

**Parameters**

- **help\_vars** (Dict[str, Any]) – locals (args/kwargs) of overwritten [change\\_flow\(\)](#) method.

**items passed to the help text formatter:**

– *count*: optional number used to render a pluralized help text for this app state change.

- **changed** ([bool](#)) – False before change of the app state, pass True if app state got just/already changed.

**Return type**[bool](#)**Returns**

True if help mode and layout is active and found target widget is locked, else False.

**help\_display**(*help\_id*, *help\_vars*, *key\_suffix*="", *must\_have*=False)

display help text to the user in activated help mode.

**Parameters**

- **help\_id** (str) – help id to show help text for.
- **help\_vars** (Dict[str, Any]) – variables used in the conversion of the f-string expression to a string. optional items passed to the help text formatter: \* *count*: optional number used to render a pluralized help text. \* *self*: target widget to show help text for.
- **key\_suffix** (str) – suffix to the key used if the translation is a dict.
- **must\_have** (bool) – pass True to display error help text and console output if no help text exists.

**Return type**

bool

**Returns**

True if help text got found and displayed.

**help\_flow\_display**(*help\_vars*, *changed*=False)

actualize the help layout if active, exclusively called by [change\\_flow\(\)](#).

**Parameters**

- **help\_vars** (Dict[str, Any]) – locals (args/kwargs) of overwritten [change\\_flow\(\)](#) method.
- **changed** (bool) – False before change to new flow, pass True if flow got changed already.

**Return type**

bool

**Returns**

True if help layout is active and found target widget is locked, else False.

**help\_is\_inactive**(*help\_id*)

check if help mode is inactive and reserve/note-down current help id for next help mode activation.

**Parameters**

**help\_id** (str) – help id to be reserved for next help activation with empty help id.

**Return type**

bool

**Returns**

True if help mode is inactive, else False.

**help\_target\_and\_id**(*help\_vars*)

find help widget/target and help id on help mode activation.

**Parameters**

**help\_vars** (Dict[str, Any]) – optional help vars.

**Return type**

Tuple[Any, str]

**Returns**

tuple of help target widget and help id.

**help\_widget**(*help\_id*, *help\_vars*)

ensure/find help target widget via attribute name/value and extend *help\_vars*.

**Parameters**

- **help\_id** (str) – widget.help\_id attribute value to detect widget and call stack locals.
- **help\_vars** (Dict[str, Any]) – help env variables, to be extended with event activation stack frame locals and a ‘self’ key with the help target widget.

**Return type**

Any

**Returns**

found help target widget or self.help\_activator if not found.

**key\_press\_from\_framework**(*modifiers*, *key*)

overwritten ae.gui\_app.MainAppBase method to suppress key press events in help or app tour mode.

**Parameters**

- **modifiers** (str) – modifier keys.
- **key** (str) – key character.

**Return type**

bool

**Returns**

True if key got consumed/used else False.

**on\_app\_started**()

app initialization event - the last one on app startup.

**on\_flow\_popup\_close**(*\_flow\_key*, *\_event\_kwargs*)

overwritten popup close handler of FlowPopup widget to reset help widget/text.

**Parameters**

- **\_flow\_key** (str) – (unused)
- **\_event\_kwargs** (Dict[str, Any]) – (unused)

**Return type**

bool

**Returns**

always True.

**save\_app\_states**()

override MainAppBase method to not overwrite app states if app tour is active.

**Return type**

str

**start\_app\_tour**(*tour\_class=None*)

start new app tour, automatically cancelling a currently running app tour.

**Parameters**

**tour\_class** (Optional[Type[TourBase]]) – optional tour (pages) class, default: tour of current help id or *OnboardingTour*.

**Return type**

bool

**Returns**

True if UI-framework support tours/has `tour_overlay_class` set and tour got started.

**widget\_by\_page\_id**(*page\_id*)

determine the first (top-most) widget having the passed tour page id.

**Parameters**

**page\_id** (str) – widgets tour page id from *tap\_flow\_id/focus\_flow\_id/app\_state\_name* attribute.

**Return type**

Optional[Any]

**Returns**

widget that has a *tap\_flow\_id/focus\_flow\_id/app\_state\_name* attribute with the value of the passed page id or None if not found.

**widget\_tourable\_children\_page\_ids**(*parent\_widget*)

determine all visible and tourable children widgets of the passed parent and its child container widgets.

**Parameters**

**parent\_widget** (Any) – parent widget to determine all children that are tourable.

**Return type**

List

**Returns**

list of page ids of tourable children of the passed parent widget.

## 4.29 ae.kivy\_glsl

### 4.29.1 add glsl shaders to your kivy widget

this ae namespace portion provides the mixin class *ShadersMixin* that can be combined with any Kivy widget to display GLSL-/shader-based graphics, gradients and animations.

additionally some *built-in shaders* are integrated into this portion. more shader examples can be found in the *glsl* sub-folder of the *GlslTester* demo application.

#### usage of ShadersMixin class

to add the *ShadersMixin* mixin class to a Kivy widget in your python code file you have to specify it in the declaration of your widget class. the following example is extending Kivy's *BoxLayout* layout with a shader:

```
from kivy.uix.boxlayout import BoxLayout
from ae.kivy_glsl import ShadersMixin

class MyBoxLayoutWithShader(ShadersMixin, BoxLayout):
```

alternatively you can declare a shader for your widget as a new kv rule within a kv file:

```
<MyBoxLayoutWithShader@ShadersMixin+BoxLayout>
```

to register a shader, call the *ShadersMixin.add\_shader()* method:

```
shader_id = widget_instance.add_shader()
```

by default `add_shader()` is using the built-in `plasma hearts` shader, provided by this portion. the next example is instead using the built-in `plunge waves` shader:

```
from ae.kivy_gls1 import BUILT_IN_SHADERS

widget_instance.add_shader(shader_code=BUILT_IN_SHADERS['plunge_waves'])
```

alternatively you can use your own shader code by specifying it on call of the method `add_shader()` either as code block string to the *paramref*: `~ShadersMixin.add_shader.shader_code` argument or as file name to the *paramref*: `~ShadersMixin.add_shader.shader_file` argument.

animation shaders like the built-in `plunge waves` and `plasma hearts` shaders need to be refreshed by a timer. the refreshing frequency can be specified via the `update_freq` parameter. to disable the automatic creation of a timer event pass a zero value to this argument.

---

**Hint:** the demo apps `ComPartY` and `Gls1Tester` are disabling the automatic timer event for each shader and using instead a Kivy clock timer to update the frames of all active shaders.

---

store the return value of `add_shader()` to stop, pause or to delete the shader later. the following examples demonstrates the deletion of a shader by calling the `del_shader()` method:

```
widget_instance.del_shader(shader_id)
```

---

**Note:** you can activate multiple shaders for the same widget. the visibility and intensity of each shader depends then on the implementation of the shader codes and the values of the input arguments (especially *alpha* and *tex\_col\_mix*) for each shader (see parameter `glsl_dyn_args`).

---

## shader compilation errors and renderer crashes

on some devices (mostly on Android) the shader script does not compile. the success property of Kivy's shader class is then set to `False` and an error message like the following gets printed on to the console output:

```
[ERROR ] [Shader      ] <fragment> failed to compile (gl:0)
[INFO  ] [Shader      ] fragment shader: <b"0:27(6): error: ....
```

some common failure reasons are:

- missing declaration of used uniform input variables.
- non-input/output variables declared on module level (they should be moved into main or any other function).

if a compile error occurred then the `run_state` value of the shader id dict will be set to `'error'` - check the value of the `error_message` key of the shader id dict for more details on the error.

in other cases the shader code compiles fine but then the renderer is crashing in the `vbo.so` library and w/o printing any Python traceback to the console - see also [this Kivy issues](#)).

if the crash only happens on some Adreno GPUs then see issues [p4a #2723](#) and [kivy #8080](#) and a possible fix in [kivy #8098](#). sometimes these type of crashes can be prevented if the texture of the widget (or of the last shader) gets fetched (w/ the function `texture2D(texture0, tex_coord0)`) - even if it is not used for the final `gl_FragColor` output variable. in

latest :

---

some cases additional to fetch the texture, the return value of the *texture2D* call has to be accessed at least once at the first render cycle.

## built-in shaders

the **circled alpha** shader is a simple gradient pixel shader without any time-based animations.

the **plunge waves** shader is animated and inspired by the kivy pulse shader example (Danguafer/Silexars, 2010) <https://github.com/kivy/kivy/blob/master/examples/shader/shadertree.py>.

the animated **plasma hearts** shader is inspired by the kivy plasma shader example <https://github.com/kivy/kivy/blob/master/examples/shader/plasma.py>.

---

**Hint:** the **GlsITester** and **ComPartY** applications are demonstrating the usage of this portion.

---

the literals of the built-in shaders got converted into constants, following the recommendations given in the accepted answer of [this SO question](#).

## Module Attributes

<i>DEFAULT_FPS</i>	default frames-per-second
<i>ShaderIdType</i>	shader internal data and id
<i>BUILT_IN_SHADERS</i>	dict of built-in shader code blocks - key specifies shader in <i>shader_code</i> .
<i>RENDER_SHAPES</i>	supported render shapes, specified in <i>render_shape</i> .
<i>SHADER_PARAMETER_MATCHER</i>	match uniform var declarations
<i>HIDDEN_SHADER_PARAMETERS</i>	uniform vars that are not editable by the user

## Functions

<i>shader_parameter_alias</i> (shader_code, arg_name)	check for alias for the passed shader argument name in the specified shader code.
<i>shader_parameters</i> (shader_code)	shader arg names of the shader code in the specified shader code.

## Classes

<i>ShadersMixin</i> ()	shader mixin base class
------------------------	-------------------------

**DEFAULT\_FPS = 30.0**

default frames-per-second

**ShaderIdType**

shader internal data and id

alias of `Dict[str, Any]`

```
4.20 archive y_sqr(x / HUNDRED) * THOUSAND;\n float m2 = y / resolution.y;\n float m3 = x
```

dict of built-in shader code blocks - key specifies shader in [shader\\_code](#).

```
RENDER_SHAPES = (<class 'kivy.graphics.vertex_instructions.Ellipse'>, <class
'kivy.graphics.vertex_instructions.Rectangle'>, <class
'kivy.graphics.vertex_instructions.RoundedRectangle'>)
```

supported render shapes, specified in [render\\_shape](#).

```
SHADER_PARAMETER_MATCHER = re.compile('^uniform (float|vec2|vec4) ([a-z_]+);',
re.MULTILINE)
```

match uniform var declarations

```
HIDDEN_SHADER_PARAMETERS = ('resolution', 'time', 'win_pos')
```

uniform vars that are not editable by the user

```
shader_parameter_alias(shader_code, arg_name)
```

check for alias for the passed shader argument name in the specified shader code.

#### Parameters

- **shader\_code** [\(str\)](#) – shader code to determine the parameter alias names (from comment of uniform declaration).
- **arg\_name** [\(str\)](#) – shader arg name.

#### Return type

[str](#)

#### Returns

alias (if found) or shader arg name (if not).

```
shader_parameters(shader_code)
```

shader arg names of the shader code in the specified shader code.

#### Parameters

**shader\_code** [\(str\)](#) – shader code from which to determine the declared uniform parameter names.

#### Return type

[Tuple\[str, ...\]](#)

#### Returns

tuple of shader arg names of the shader code passed into the argument [shader\\_code](#), plus the [start\\_time](#) argument which controls, independent of to be included in the shader code, how the [time](#) argument get prepared in each render frame (see also [start\\_time](#)).

```
class ShadersMixin
```

Bases: [object](#)

shader mixin base class

**canvas:** [Any](#)

**center:** [Tuple\[float, float\]](#)

**fbind:** [Callable](#)

**parent:** [Any](#)

**pos:** [list](#)



**size:** `list`

**to\_window:** `Callable`

**unbind\_uid:** `Callable`

**added\_shaders**

list of shader-ids/kwarg-dicts for each shader

**running\_shaders:** `List[Dict[str, Any]] = []`

list/pool of active/running shaders/render-contexts

**\_pos\_fbind\_uid:** `int = 0`

**\_size\_fbind\_uid:** `int = 0`

**add\_shader**(*add\_to*="", *shader\_code*='uniform float alpha;\nuniform float contrast;\nuniform float tex\_col\_mix;\nuniform float time;\nuniform vec2 center\_pos;\nuniform vec2 win\_pos;\nuniform vec2 resolution;\nuniform vec4 tint\_ink;\n\nconst float THOUSAND = 963.9;\nconst float HUNDRED = 69.3;\nconst float TEN = 9.9;\nconst float TWO = 1.83;\nconst float ONE = 0.99;\n\nvoid main(void)\n{\n vec2 pix\_pos = (frag\_modelview\_mat \* gl\_FragCoord).xy - win\_pos;\n vec2 rel\_center = center\_pos - win\_pos;\n float x = abs(pix\_pos.x - rel\_center.x);\n float y = abs(pix\_pos.y - rel\_center.y - resolution.y);\n float m1 = x + y + cos(sin(time) \* TWO) \* HUNDRED + sin(x / HUNDRED) \* THOUSAND;\n float m2 = y / resolution.y;\n float m3 = x / resolution.x + time \* TWO;\n float c1 = abs(sin(m2 + time) / TWO + cos(m3 / TWO - m2 - m3 + time));\n float c2 = abs(sin(c1 + sin(m1 / THOUSAND + time) + sin(y / HUNDRED + time) + sin((x + y) / HUNDRED) \* TWO));\n float c3 = abs(sin(c2 + cos(m2 + m3 + c2) + cos(m3) + sin(x / THOUSAND)));;\n vec4 tex = texture2D(texture0, tex\_coord0);\n float dis = TWO \* distance(pix\_pos, rel\_center) / max(resolution.x, resolution.y);\n vec4 col = vec4(c1, c2, c3, contrast \* (ONE - dis)) \* tint\_ink \* TWO;\n col = mix(tex, col, tex\_col\_mix);\n gl\_FragColor = vec4(col.rgb, col.a \* sqrt(alpha));\n}\n', *shader\_file*="", *start\_time*=0.0, *update\_freq*=30.0, *run\_state*='running', *render\_shape*=<class 'kivy.graphics.vertex\_instructions.Rectangle'>, \*\**glsl\_dyn\_args*)

add/register a new shader for the mixing-in widget.

### Parameters

- **add\_to** (str) – “ to add to current canvas, ‘before’ and ‘after’ to the before/after canvas of the widget instance mixed-into. if the canvas does not exist then the shaders render context will be set as a current canvas.
- **shader\_code** (str) – fragment shader code block or key of BUILT\_IN\_SHADERS dict if first character is ‘=’. this argument will be ignored if *shader\_file* is not empty.
- **shader\_file** (str) – filename with the glsl shader code (with “-VERTEX” or “-FRAGMENT” sections) to load.
- **start\_time** (Optional[float]) – base/start time. passing the default value zero is syncing the *time* glsl parameter of this shader with `kivy.clock.Clock.get_boottime()`. pass `None` to initialize this argument to the current `Clock` boot time, to start the *time* glsl argument at zero.
- **update\_freq** (float) – shader render update frequency. pass 0.0 to disable creation of an update timer.
- **run\_state** (str) – optional shader run state (default='running'), pass 'paused'/'error' to not run it.
- **render\_shape** (Union[Any, str]) – pass one of the supported shapes (*RENDER\_SHAPES*) either as shape class or str.

- **glsl\_dyn\_args** – extra/user dynamic shader parameters, depending on the used shader code. the keys of this dict are the names of the corresponding glsl input variables in your shader code. the built-in shaders (provided by this module) providing the following glsl input variables:
  - *'alpha'*: opacity (float, 0.0 - 1.0).
  - *'center\_pos'*: center position in Window coordinates (tuple(float, float)).
  - *'contrast'*: color contrast (float, 0.0 - 1.0).
  - *'mouse'*: mouse pointer position in Window coordinates (tuple(float, float)).
  - *'resolution'*: width and height in Window coordinates (tuple(float, float)).
  - *'tex\_col\_mix'*: **factor (float, 0.0 - 1.0) to mix the kivy input texture** and the calculated color. a value of 1.0 will only show the shader color, whereas 0.0 will result in the color of the input texture (uniform texture0).
  - *'tint\_ink'*: tint color with color parts in the range 0.0 till 1.0.
  - *'time'*: **animation time (offset to `start_time`) in seconds. if** specified as constant (non-dynamic) value then you have to call the `next_tick()` method to increment the timer for this shader.

pass a callable to provide a dynamic/current value, which will be called on each rendering frame without arguments and the return value will be passed into the glsl shader.

---

**Note:** don't pass *int* values because some renderer will interpret them as *0.0*.

---

#### Return type

`Dict[str, Any]`

#### Returns

index (id) of the created/added render context.

#### `_compile_shader(shader_id)`

try to compile glsl shader file/code - raise `ValueError` if compilation failed.

#### Parameters

**shader\_id** (`Dict[str, Any]`) – shader id (internal data dict with either *shader\_file* (preference) or *shader\_code* key representing the shader code to use).

#### Return type

`RenderContext`

#### Returns

kivy/glsl render context with the compiled shader attached to it.

#### `del_shader(shader_id)`

remove *shader\_id* added via `add_shader`.

#### Parameters

**shader\_id** (`Dict[str, Any]`) – id of the shader to remove (returned by `add_shader()`). ignoring if the passed shader got already removed.

#### `next_tick(increment=0.03333333333333333)`

increment glsl *time* input argument if `running_shaders` get updated manually/explicitly by the app.

**Parameters**

**increment**¶ (float) – delta in seconds for the next refresh of all running\_shaders with a *time* constant.

**on\_added\_shaders**(\*\_args)

added\_shaders list property changed event handler.

**on\_parent**(\*\_args)

parent changed event handler.

**play\_shader**(shader\_id)

create new render context canvas and add it to the widget canvas to display shader output.

**Parameters**

**shader\_id**¶ (Dict[str, Any]) – shader id and internal data dict with either *shader\_file* (preference) or *shader\_code* key representing the shader code to use. a shader dict with the *shader\_code* key is a fragment shader (w/o a vertex shader), that will be automatically prefixed with the Kivy fragment shader header file template, if the \$HEADER\$ placeholder is not included in the shader code (even if it is commented out, like in the following glsl code line: `#ifdef GL_ES //$HEADER$`).

**Return type**

str

**Returns**

“” if shader is running/playing or error message string.

**\_pos\_changed**(\*\_args)

pos changed event handler.

**\_refresh\_glsl**(shader\_id, \_dt)

timer/clock event handler to animate and sync one canvas shader.

**refresh\_running\_shaders**()

manually update all running\_shaders.

**refresh\_shader**(shader\_id)

update the shader arguments for the current animation frame.

**Parameters**

**shader\_id**¶ (Dict[str, Any]) – dict with render context, rectangle and glsl input arguments.

**Return type**

str

**Returns**

empty string if arguments got passed to the shader without errors, else error message.

**\_size\_changed**(\*\_args)

size changed event handler.

**stop\_shader**(shader\_id, set\_run\_state=True)

stop shader by removing it from started shaders.

**Parameters**

- **shader\_id**¶ (Dict[str, Any]) – id of the shader to stop. ignoring if the passed shader got already stopped.

- **set\_run\_state** (bool) – pass False to prevent that the *run\_state* of the *shader\_id* gets changed to *paused* (for internal use to refresh all running shaders).

#### **update\_shaders()**

stop/unbind all shaders, then restart/bind the shaders having their *run\_state* as *running*.

## 4.30 ae.kivy\_dyn\_chi

### 4.30.1 dynamic children mix-in for kivy container widgets

this ae portion is providing the mixin class *DynamicChildrenBehavior* to add children widgets dynamically and data-driven to your kivy popup widget (like DropDowns, Popups, Menus, Selectors).

#### Module Attributes

<i>AttrMapType</i>	child attribute (value of the 'kwargs' and 'attributes' keys)
<i>DataItemValueType</i>	item dict value (str for 'cls' key, dict for 'kwargs'/'attributes')
<i>DataItemType</i>	<i>DynamicChildrenBehavior.child_data_maps</i> item type
<i>ChildrenDataType</i>	<i>DynamicChildrenBehavior.child_data_maps</i> type

#### Classes

<i>DynamicChildrenBehavior</i> (**kwargs)	mixin class for the dynamic creation/refresh of child widgets from a data map.
---	--

#### **AttrMapType**

child attribute (value of the 'kwargs' and 'attributes' keys)

alias of `Dict[str, Any]`

#### **DataItemValueType**

item dict value (str for 'cls' key, dict for 'kwargs'/'attributes')

alias of `Union[str, Dict[str, Any]]`

#### **DataItemType**

*DynamicChildrenBehavior.child\_data\_maps* item type

alias of `Dict[str, Union[str, Dict[str, Any]]]`

#### **ChildrenDataType**

*DynamicChildrenBehavior.child\_data\_maps* type

alias of `List[Dict[str, Union[str, Dict[str, Any]]]]`

**\_child\_data\_dict**(*child\_data, cls, key, defaults*)

determine child data dict values and put children default values for the keys not specified in child data.

**Return type**

`Dict[str, Any]`

**class DynamicChildrenBehavior**(\*\**kwargs*)

Bases: `object`

mixin class for the dynamic creation/refresh of child widgets from a data map.

at least one of the classes that is mixing in this class has to inherit from `Widget` (or `EventDispatcher`) to get the `child_data_maps` attribute correctly initialized and firing on change.

**child\_data\_defaults:** `List[Dict[str, Union[str, Dict[str, Any]]]]`

child data default values for all the children with the same *cls* key that are specified via the `ae.kivy_dyn_chi.DynamicChildrenBehavior.child_data_maps` property. if the *cls* key is missing or its item value is empty then the defaults in the other item values will be used for all children.

`child_data_defaults` is a `ListProperty` and defaults to an empty list.

**child\_data\_maps:** `List[Dict[str, Union[str, Dict[str, Any]]]]`

list of child data dicts to instantiate the children of the inheriting layout/widget.

each child data dict is defining its own widget with the following keys:

- *cls*: either the class name or the class/type object of the widget to be created dynamically.
- *kwargs*: dict of keyword arguments that will be passed to the constructor method of the widget. all values in this dict with the magic string `'replace_with_data_map_popup'` will be replaced with the instance of the container before it gets passed to the `__init__` method of the child (see `deep_replace()`).
- *attributes*: dict of attributes where the key is specifying the attribute name/path and the value the finally assigned attribute value. the attribute name (the key of this dict) can be a deep/combined attribute/index path which allows to update deeper objects within the child object (via `key_path_object()`). all values in this dict with the magic string `'replace_with_data_map_popup'` will be replaced with the instance of the container before the child attributes get updated. all values in this dict with the magic string `'replace_with_data_map_child'` will be replaced with the instance of the child before they get applied to it (via `deep_replace()`).

`child_data_maps` is a `ListProperty` and defaults to an empty list.

**bind:** `Callable`

**container:** `Widget`

**\_container:** `Widget`

**\_\_init\_\_**(\*\**kwargs*)

add dynamic creation and refresh of children to this layout (Popup/Dropdown/...) widget.

**refresh\_child\_data\_widgets**(\**args*, \*\**init\_kwargs*)

recreate dynamic children of the passed widget.

**Parameters**

- **\_args** – not needed extra args (only passed if this function get called as event handler).
- **init\_kwargs** – container kwargs (passed from `__init__()` method).

## 4.31 ae.kivy\_relief\_canvas

### 4.31.1 inner/outer elliptic/square reliefs for any kivy widget

the *ReliefCanvas* mixin class of this ae namespace portion can be added to any square or elliptic Kivy widget to draw an inner and/or outer relief, in order to convert your widget to have an outstanding or sunken 3D-appearance.

#### Module Attributes

<i>ReliefColors</i>	tuple of (top, bottom) relief colors or empty tuple
<i>ReliefBrightness</i>	top and bottom brightness/darken factor

#### Functions

<i>relief_colors</i> ([color_or_ink, darken_factors])	calculate the (top and bottom) colors used for the relief lines/drawings.
---	---

#### Classes

<i>ReliefCanvas</i> (**kwargs)	relief canvas mixin class.
--------------------------------	----------------------------

#### ReliefColors

tuple of (top, bottom) relief colors or empty tuple

alias of `Union[Tuple[Union[Tuple[float, float, float], List[float]], Union[Tuple[float, float, float], List[float]]], Tuple]`

#### ReliefBrightness

top and bottom brightness/darken factor

alias of `Tuple[float, float]`

**relief\_colors**(color\_or\_ink=(0, 0, 0), darken\_factors=(0.6, 0.3))

calculate the (top and bottom) colors used for the relief lines/drawings.

#### Parameters

- **color\_or\_ink** *(Union[Tuple[float, float, float], List[float], Tuple[float, float, float, float]])* – color used to calculate the relief colors from, which will first be lightened until one of the color parts (R, G or B) reach the value 1.0; then darken factors will be applied to the color parts. If not passed then grey colors will be returned.

---

**Note:** if the alpha value of paramref:~*relief\_colors.color\_or\_ink* is zero then no relief colors will be calculated and an empty tuple will be returned (disabling relief).

---

- **darken\_factors** *(Tuple[float, float])* – two factors to darken (1) the top and (2) the bottom relief color parts.

**Return type**

`Union[Tuple[Union[Tuple[float, float, float], List[float]], Union[Tuple[float, float, float], List[float]]], Tuple]`

**Returns**

tuple with darkened colors calculated from `color_or_ink` and `darken_factors` or an empty tuple if the alpha value of `color_or_ink` has a zero value.

**class ReliefCanvas**(\*\*kwargs)

Bases: `object`

relief canvas mixin class.

to activate the drawing of a relief you have to specify two colors, one for the top part and another one for the bottom part of the relief, which are both stored in a single kivy property. the function `relief_colors()` can be used to calculate lightened and darkened values of the surface color of the widget:

```
MySquareRaisedWidgetWithColoredSurface:
    surface_color: 0.9, 0.6, 0.3, 1.0
    relief_square_outer_colors: relief_colors(color_or_ink=self.surface_color)
```

this will result in a raised widget with a square outer relief where the top/left relief color get a lightened value and the bottom/right relief a darkened value of the color specified by `surface_color`.

using the default values will result in raised widgets with the inner part sunken, simulating the light source in the top left window border/corner. to make a sunken widget for the same light source position you simply have to flip the items of the `darken_factors` argument of the `relief_colors()` function.

the following example shows this for a round/elliptic button widget:

```
MyRoundSunkenButton:
    relief_ellipse_outer_colors: relief_colors(darken_factors=(0.3, 0.6))
```

the other color attributes of this mixin class control the relief colors for the inner part of a square shaped widget (`relief_square_inner_colors`) and for the inner part of an elliptic shape widget (`relief_ellipse_inner_colors`).

the depth of the outer raise/sunk effect can be controlled with the `relief_square_outer_lines` property/attribute. `relief_square_inner_lines` controls the raise/sunk depth of the inner surface of a square widget. `relief_ellipse_inner_lines` and `relief_ellipse_outer_lines` are doing the same for widgets with a round/elliptic shape.

the properties `relief_square_inner_offset` and `relief_ellipse_inner_offset` are specifying the width of the widget border (the part between the outer and the inner relief) in pixels.

---

**Note:** at least one of the classes that is mixing in this class has to inherit from `Widget` (or `EventDispatcher`) to get the widgets `pos`, `size`, `canvas` properties and the `bind` method.

---

**relief\_pos\_size**

list/tuple of optional relief position and size (x, y, width, height) in pixels.

if not specified or empty list/tuple, than the pos/size values of the mixing-in widget will be used instead.

`relief_pos_size` is a `ListProperty` and defaults to an empty list.

**relief\_ellipse\_inner\_colors:** `Union[Tuple[Union[Tuple[float, float, float], List[float]], Union[Tuple[float, float, float], List[float]]], Tuple]`

list/tuple of ellipse inner (top, bottom) rgb colors.

*relief\_ellipse\_inner\_colors* is a *ObjectProperty* and defaults to an empty tuple.

**relief\_ellipse\_inner\_lines**

number of ellipse inner lines/pixels to be drawn.

*relief\_ellipse\_inner\_lines* is a *NumericProperty* and defaults to '3sp'.

**relief\_ellipse\_inner\_offset**

number of pixels left unchanged at the border of the inner elliptic surface before the inner relief starts.

*relief\_ellipse\_inner\_offset* is a *NumericProperty* and defaults to '1sp'.

**relief\_ellipse\_outer\_colors:** *Union*[*Tuple*[*Union*[*Tuple*[float, float, float], *List*[float]], *Union*[*Tuple*[float, float, float], *List*[float]]], *Tuple*]

list/tuple of ellipse outer (top, bottom) rgb colors.

*relief\_ellipse\_outer\_colors* is a *ObjectProperty* and defaults to an empty tuple.

**relief\_ellipse\_outer\_lines**

number of ellipse outer lines/pixels to be drawn.

*relief\_ellipse\_outer\_lines* is a *NumericProperty* and defaults to '3sp'.

**relief\_square\_inner\_colors**

list/tuple of square inner (top, bottom) rgb colors.

*relief\_square\_inner\_colors* is a *ObjectProperty* and defaults to an empty tuple.

**relief\_square\_inner\_lines**

number of square inner lines/pixels to be drawn.

*relief\_square\_inner\_lines* is a *NumericProperty* and defaults to '3sp'.

**relief\_square\_inner\_offset**

number of pixels left unchanged at the border of the square inner surface before the inner relief starts.

*relief\_square\_inner\_offset* is a *NumericProperty* and defaults to '1sp'.

**relief\_square\_outer\_colors:** *Union*[*Tuple*[*Union*[*Tuple*[float, float, float], *List*[float]], *Union*[*Tuple*[float, float, float], *List*[float]]], *Tuple*]

list/tuple of square outer (top, bottom) rgb colors.

*relief\_square\_outer\_colors* is a *ObjectProperty* and defaults to an empty tuple.

**relief\_square\_outer\_lines:** *NumericProperty*

number of square outer lines/pixels to be drawn.

*relief\_square\_outer\_lines* is a *NumericProperty* and defaults to '3sp'.

**bind:** *Any*

**canvas:** *Any*

**pos:** *list*

**size:** *list*

**\_\_init\_\_**(\*\**kwargs*)

**\_relief\_refresh**(\**args*)

pos/size or color changed event handler.



**`_relief_ellipse_inner_refresh`**(*add\_instruction, top\_color, bottom\_color, wid\_x, wid\_y, wid\_width, wid\_height*)

ellipse pos/size or color changed event handler.

**`_relief_ellipse_outer_refresh`**(*add\_instruction, top\_color, bottom\_color, wid\_x, wid\_y, wid\_width, wid\_height*)

ellipse pos/size or color changed event handler.

**`_relief_square_inner_refresh`**(*add\_instruction, top\_color, bottom\_color, wid\_x, wid\_y, wid\_width, wid\_height*)

square pos/size or color changed event handler.

**`_relief_square_outer_refresh`**(*add\_instruction, top\_color, bottom\_color, wid\_x, wid\_y, wid\_width, wid\_height*)

square pos/size or color changed event handler.

## 4.32 ae.kivy

### 4.32.1 core application classes and widgets for GUIApp-conform Kivy apps

this ae portion is implementing the Kivy-framework-specific parts for apps with multilingual context-sensitive help, user onboarding, tours, walkthroughs and tutorials.

by extending and joining the app classes *MainAppBase*, *HelpAppBase* and *App*, it is providing additional *config variables*, some useful constants, behaviors and widgets for your multi-platform apps.

this portion is composed of the following modules:

- *i18n*: internationalization (i18n) function *get\_txt()* for python and kv code
- *behaviors*: widget behavior classes
- *widgets*: generic widget classes and some useful constants
- *tours*: app tour widget classes
- *apps*: providing the two application classes (*FrameworkApp* and *KivyMainApp*)

#### unit tests

unit tests are currently still incomplete and need at least V 2.0 of OpenGL and the *Kivy framework* installed.

---

**Note:** unit tests are currently not passing at the gitlab CI because is failing to set up a properly running OpenGL graphics/window system on the python image that all ae portions are using.

---

## 4.33 ae.kivy.i18n

### 4.33.1 ae.kivy.i18n module

this module is adding translatable f-strings to the python and kv code of your app, via the helper function `get_txt()` and the `_GetTextBinder` class.

#### Module Attributes

<code>get_txt(text[, count, language, loc_vars])</code>	instantiate global i18n translation callable and language switcher helper
---	---

#### class \_GetTextBinder

Bases: Observable

redirect `ae.i18n.get_f_string()` to an instance of this class.

kivy currently only support a single one automatic binding in kv files for all function names ending with `_` (see `watched_keys` extension in `kivy/lang/parser.py` line 201; e.g. `f_` would get recognized by the `lang_tr` re pattern, but kivy will only add the `_` symbol to `watched_keys` and therefore `f_` not gets bound.) to allow both - f-strings and simple `get_text` messages - this module binds `ae.i18n.get_f_string()` to the `get_txt` symbol (instead of `ae.i18n.get_text()`).

`get_txt` can be used as translation callable, but also to switch the current default language. additionally `get_txt` is implemented as an observer that automatically updates any translations messages of all active/visible kv rules on switch of the language at app run-time.

inspired by (see also discussion at <https://github.com/kivy/kivy/issues/1664>):

- <https://github.com/tito/kivy-gettext-example>
- [https://github.com/Kovak/kivy\\_i18n\\_test](https://github.com/Kovak/kivy_i18n_test)
- <https://git.bluedynamics.net/phil/woodmaster-trainer/-/blob/master/src/ui/kivy/i18n.py>

**observers:** `List[Tuple[Callable, tuple, dict]] = []`

list of bound observer tuples (func, args, kwargs)

**\_bound\_uid** = -1

**fbind**(name, func, \*args, \*\*kwargs)

override fbind (fast bind) from Observable to collect and separate `_` bindings.

#### Parameters

- **name** `(str)` – attribute name to be bound.
- **func** `(Callable)` – observer notification function (to be called if attribute changes).
- **args** – args to be passed to the observer.
- **kwargs** – kwargs to be passed to the observer.

#### Return type

`int`

#### Returns

unique id of this binding.

**funbind**(*name*, *func*, \**args*, \*\**kwargs*)

override fast unbind.

#### Parameters

- **name** (str) – bound attribute name.
- **func** (Callable) – observer notification function (called if attribute changed).
- **args** – args to be passed to the observer.
- **kwargs** – kwargs to be passed to the observer.

**switch\_lang**(*lang\_code*)

change language and update kv rules properties.

#### Parameters

**lang\_code** (str) – language code to switch this app to.

**\_\_call\_\_**(*text*, *count*=None, *language*="", *loc\_vars*=None, \*\**kwargs*)

translate text into the current-default or the passed language.

#### Parameters

- **text** (str) – text to translate.
- **count** (Optional[int]) – optional count for pluralization.
- **language** (str) – language code to translate the passed text to (def=current default language).
- **loc\_vars** (Optional[Dict[str, Any]]) – local variables used in the conversion of the f-string expression to a string. the *count* item of this dict will be overwritten by the value of the *count* parameter (if this argument got specified).
- **kwargs** – extra kwargs (e.g. *glo\_vars* or *key\_suffix* - see *get\_f\_string()*).

#### Return type

str

#### Returns

translated text.

**get\_txt**(*text*, *count*=None, *language*="", *loc\_vars*=None, \*\**kwargs*) = <ae.kivy.i18n.\_GetTextBinder object>

instantiate global i18n translation callable and language switcher helper

## 4.34 ae.kivy.widgets

### 4.34.1 ae.kivy.widgets module

this module provides constants and widgets for your multi-platform apps.

the generic constants for animations and vibration patterns (mostly used on mobile platforms).

most of the widgets provided by this module are based on the widgets of the [Kivy framework](#), extended to work with *app state variables*, e.g. to support app styles and theming (dark or light) and user definable font sizes. some of them also change the *application flow*.

by importing this module the following generic widgets will be registered in the kivy widget class factory maps, to be available in the kv language for your app:

- *AppStateSlider*: extended version of *Slider*, changing the value of *app state variables*.
- *FlowButton*: button to change the application flow.
- *FlowDropDown*: attachable menu-like popup, based on *DropDown*.
- *FlowInput*: dynamic kivy widget based on *TextInput* with application flow support.
- *FlowPopup*: dynamic auto-content-sizing popup to query user input or to show messages.
- *FlowSelector*: attachable popup used for dynamic elliptic auto-spreading menus and toolbars.
- *FlowToggler*: toggle button based on *ImageLabel* and *ToggleButtonBehavior* to change the application flow or any flag or application state.
- *HelpToggler* is a toggle button widget that switches the app's help and tour mode on and off.
- *ImageLabel*: dynamic kivy widget extending the Kivy *Label* widget with an image.
- *MessageShowPopup*: simple message box widget based on *FlowPopup*.
- *OptionalButton*: dynamic kivy widget based on *FlowButton* which can be dynamically hidden.
- *ShortenedButton*: dynamic kivy widget based on *FlowButton* shortening the button text.
- *Tooltip* displays text blocks that are automatically positioned next to any widget to providing e.g. i18n context help texts or app tour/onboarding info.
- *UserNameEditorPopup*: popup window used e.g. to enter new user, finally registered in the app config files.

### tooltip popup to display context-sensitive help and app tour texts

the tooltip popup widget class *Tooltip* allows you to target any widget by pointing with an arrow to it. the position and size of this widget gets automatically calculated from the targeted widget position and size and the tooltip text size. and if the screen/window size is not big enough then the tooltip texts get scrollable.

---

**Hint:** use cases of the class *Tooltip* are e.g. the help texts prepared and displayed by the method *help\_display()* as well as the “explaining widget” tooltips in an app tour.

---

### help activation and de-activation

use the widget class *HelpToggler* provided by this module to toggle the active state of the help mode.

---

**Hint:** the *HelpToggler* class is using the low-level touch events to prevent the dispatch of the Kivy events *on\_press*, *on\_release* and *on\_dismiss*, to allow to show help texts for opened dropdowns and popups, without closing/dismissing them.

---

to attach help texts to your widget instances add the behavior class *HelpBehavior*.

## Module Attributes

<code>ANI_SINE_DEEPER_REPEAT3</code>	sine 3 x deeper repeating animation, used e.g.
<code>CRITICAL_VIBRATE_PATTERN</code>	very long/~2.4s vibrate pattern for critical error notification (sending SOS to the mobile world;)
<code>ERROR_VIBRATE_PATTERN</code>	long/~2s vibrate pattern for error notification.
<code>LOVE_VIBRATE_PATTERN</code>	short/~1.2s vibrate pattern for fun/love notification.
<code>MAIN_KV_FILE_NAME</code>	default file name of the main kv file of your app

## Classes

<code>AbsolutePosSizeBinder(*widgets[, ...])</code>	propagate changes of <i>pos/size</i> properties of one or more widgets plus their parents to attributes/callbacks.
<code>AppStateSlider(**kwargs)</code>	slider widget with help text to change app state value.
<code>ExtTextInputCutCopyPaste(**kwargs)</code>	overwrite/extend <code>kivy.uix.textinput.TextInputCutCopyPaste</code> w/ translatable and autocomplete options.
<code>FlowButton(**kwargs)</code>	button to change the application flow.
<code>FlowDropDown(**kwargs)</code>	flow based widget class to implement dynamic menu-like user selections and toolbars.
<code>FlowInput(**kwargs)</code>	text input/edit widget with optional autocompletion.
<code>FlowPopup(**kwargs)</code>	popup for dynamic and auto-sizing dialogs and other top-most or modal windows.
<code>FlowSelector(**kwargs)</code>	attachable popup used for dynamic elliptic auto-spreading menus and toolbars.
<code>FlowToggler(**kwargs)</code>	toggle button changing flow id.
<code>HelpToggler(**kwargs)</code>	widget to activate and deactivate the help mode.
<code>ImageLabel(**kwargs)</code>	base label used for all labels and buttons - declared in <code>widgets.kv</code> and also in this module to inherit from.
<code>MessageShowPopup(**kwargs)</code>	flow popup to display info or error messages.
<code>Tooltip(**kwargs)</code>	semi-transparent and optional click-through container to display help and tour page texts.

**ANI\_SINE\_DEEPER\_REPEAT3** = <kivy.animation.Sequence object>

sine 3 x deeper repeating animation, used e.g. to animate help layout (see *Tooltip* widget)

**CRITICAL\_VIBRATE\_PATTERN** = (0.0, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.24, 0.12, 0.24, 0.12, 0.24, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12)

very long/~2.4s vibrate pattern for critical error notification (sending SOS to the mobile world;)

**ERROR\_VIBRATE\_PATTERN** = (0.0, 0.09, 0.09, 0.18, 0.18, 0.27, 0.18, 0.36, 0.27, 0.45)

long/~2s vibrate pattern for error notification.

**LOVE\_VIBRATE\_PATTERN** = (0.0, 0.12, 0.12, 0.21, 0.03, 0.12, 0.12, 0.12)

short/~1.2s vibrate pattern for fun/love notification.

**MAIN\_KV\_FILE\_NAME** = 'main.kv'

default file name of the main kv file of your app

---

```
class AbsolutePosSizeBinder(*widgets, bind_window_size=False)
```

Bases: `object`

propagate changes of *pos/size* properties of one or more widgets plus their parents to attributes/callbacks.

create an instance of this class passing the widget(s) to observe on change of their pos/size. then call the methods `pos_to_attribute()`, `pos_to_callback()`, `size_to_attribute()` and `size_to_callback()` to specify the propagation of the changed *pos* and/or *size*. to remove the change propagation call the method `unbind()`.

---

**Note:** the *pos* attribute/callback propagations are providing absolute window coordinates.

---

```
__init__(*widgets, bind_window_size=False)
```

instantiate binder specifying the monitored widget(s).

#### Parameters

- **widgets** `(Widget)` – widget(s) to observe changes of their *pos* and *size* properties. if specified more than one widget then the pos/size coordinates of the rectangle that is enclosing all specified widgets are propagated.
- **bind\_window\_size** `(bool)` – pass True to propagate pos and size changes if window size changes.

```
_bind()
```

```
_propagate(wid, value, attributes, callbacks)
```

```
_wid_pos_changed(wid, new_pos)
```

propagate *pos* property change to target attributes and subscribed observers.

#### Parameters

- **wid** `(Widget)` – bound widget or a ScrollView that is embedding the bound widget, which pos changed.
- **new\_pos** `(List[float])` – new position of the bound widget/ScrollView (unused).

```
_wid_size_changed(wid, new_size)
```

propagate *size* property change to target attributes and subscribed observers.

#### Parameters

- **wid** `(Widget)` – bound widget or a ScrollView that is embedding the bound widget, which pos changed.
- **new\_size** `(List[float])` – new position of the bound widget/ScrollView (unused).

```
_rel_pos_changed(_rel, _new_pos)
```

propagate *pos* property change of relative/scrollable layout/container.

#### Parameters

- **\_rel** `(Widget)` – relative layout or a scroll view, embedding bound widget(s), which pos changed.
- **\_new\_pos** `(list)` – new position of the RelativeLayout/ScrollView (unused).

```
_rel_size_changed(_rel, _new_size)
```

propagate size change of relative/scrollable layout/container.

#### Parameters

- **`_rel`** (Widget) – relative layout or a scroll view, embedding bound widget(s), which size changed.
- **`_new_size`** (list) – new size of the RelativeLayout/ScrollView (unused).

**`pos_to_attribute`**(*target, attribute, converter=None*)

request the propagation of the changed (absolute) widget(s) position to an object attribute.

#### Parameters

- **`target`** (Any) – the object which attribute will be changed on change of *pos*.
- **`attribute`** (str) – the name of the attribute to assign the new/changed absolute position.
- **`converter`** (Optional[Callable[[Widget, List[float]], Any]]) – optional pos value converter, returning the final value assigned to the attribute.

**`pos_to_callback`**(*callback*)

bind callable to *pos* change event.

#### Parameters

**`callback`** (Callable[[Widget, List[float]], Any]) – callable to be called when pos changed with the changed widget and pos as arguments.

**`size_to_attribute`**(*target, attribute, converter=None*)

request the propagation of the changed widget(s) size to an object attribute.

#### Parameters

- **`target`** (Any) – the object which attribute will be changed on change of *size*.
- **`attribute`** (str) – the name of the attribute to assign the new/changed size.
- **`converter`** (Optional[Callable[[Widget, List[float]], Any]]) – optional pos value converter, returning the final value assigned to the attribute.

**`size_to_callback`**(*callback*)

bind callable to *size* change event.

#### Parameters

**`callback`** (Callable[[Widget, List[float]], Any]) – callable to be called when size changed with the changed widget and size as arguments.

**`unbind`**()

unbind the widget(s) of this binder instance.

---

**Note:** this instance can be destroyed after the call of this method. for new bindings create a new instance.

---

**`class AppStateSlider`**(\*\**kwargs*)

Bases: [HelpBehavior](#), [ShadersMixin](#), [Slider](#)

slider widget with help text to change app state value.

**`app_state_name`**

name of the app state to be changed by this slider value

**`__str__`**()

added for easier debugging.

**on\_value**(\*args)

value changed event handler.

**Parameters**

**args** – tuple of instance and new value.

**class HelpToggler**(\*\*kwargs)

Bases: [ReliefCanvas](#), [Image](#)

widget to activate and deactivate the help mode.

To prevent dismiss of opened popups and dropdowns at help mode activation, this singleton instance has to:

- be registered in its `__init__` to the [help\\_activator](#) attribute and
- have a [on\\_touch\\_down\(\)](#) method that is eating the activation touch event (returning True) and
- a [on\\_touch\\_down\(\)](#) method not passing an activation touch in all DropDown/Popup widgets.

**ani\_value**

float value (range: 0.0 - 1.0) to animate this button in help/tour mode

**\_\_init\_\_**(\*\*kwargs)

initialize an instance of this class and also [help\\_activator](#).

**ani\_start**()

start animation of this button.

**ani\_stop**()

stop animation of this button.

**on\_touch\_down**(touch)

touch down event handler to toggle help mode while preventing dismiss of open dropdowns/popups.

**Parameters**

**touch** ([MotionEvent](#)) – touch event.

**Return type**

[bool](#)

**Returns**

True if touch happened on this button (and will get no further processed => eaten).

**class ImageLabel**(\*\*kwargs)

Bases: [ReliefCanvas](#), [ShadersMixin](#), [Label](#)

base label used for all labels and buttons - declared in widgets.kv and also in this module to inherit from.

---

**Note:** hide-able label needs extra handling, because even setting width/height to zero the text can still be visible, especially in dark mode and even with having the text-color-alpha==0. to fully hide the texture in all cases, set either the text to an empty string or the opacity to zero.

---

**\_\_repr\_\_**()

added for easier debugging of [FlowButton](#) and [FlowToggler](#) widgets.

**class FlowButton**(\*\*kwargs)

Bases: [HelpBehavior](#), [SlideSelectBehavior](#), [TouchableBehavior](#), [ButtonBehavior](#), [ImageLabel](#)

button to change the application flow.



**long\_tap\_flow\_id**

flow id that will be set when this button gets long tap event

**tap\_flow\_id**

the new flow id that will be set when this button get tapped

**tap\_kwargs**

kwargs dict passed to event handler (change\_flow) when button get tapped

**\_\_init\_\_**(\*\*kwargs)

**on\_long\_tap**(touch)

long tap/click default handler.

**Parameters**

**touch** *¶* (*MotionEvent*) – motion/touch event data with the touched widget in *touch.grab\_current*.

**on\_release**()

overridable touch release event handler.

**class FlowDropDown**(\*\*kwargs)

Bases: *ContainerChildrenAutoWidthBehavior*, *DynamicChildrenBehavior*, *SlideSelectBehavior*, *ReliefCanvas*, *DropDown*

flow based widget class to implement dynamic menu-like user selections and toolbars.

**close\_kwargs**

kwargs passed to all close action flow change event handlers

**content**

layout container

**menu\_items**

container/content children, like buttons, text inputs or sliders

**parent\_popup\_to\_close**

parent popup widget instance to be closed if this dropdown closes

**\_\_init\_\_**(\*\*kwargs)

**\_\_repr\_\_**()

added for easier debugging.

**\_real\_dismiss**(\*args)

overridden to ensure that return value of on\_dismiss-dispatch get recognized.

**dismiss**(\*args)

override DropDown method to prevent dismiss of any dropdown/popup while clicking on activator widget.

**Parameters**

**args** *¶* – args to be passed to DropDown.dismiss().

**close**(\*args)

override DropDown method to prevent dismiss of any dropdown/popup while clicking on activator widget.

**Parameters**

**args** *¶* – args to be passed to DropDown.dismiss().

**on\_container**(*instance, value*)

sync *content* widget and *menu\_items* list with container widget.

**Parameters**

- **instance** (Widget) – self.
- **value** (Widget) – new/changed *container* widget.

**on\_dismiss**()

default dismiss/close default event handler.

**Return type**

Optional[bool]

**Returns**

True to prevent/cancel the dismiss/close.

**on\_touch\_down**(*touch*)

prevent the processing of a touch on the help activator widget by this dropdown.

**Parameters**

**touch** (MotionEvent) – motion/touch event data.

**Return type**

bool

**Returns**

True if event got processed/used.

**\_reposition**(*\*args*)

fixing Dropdown bug - see issue #7382 and PR #7383. TODO: remove if PR gets merged and distributed.

**class ExtTextInputCutCopyPaste**(*\*\*kwargs*)

Bases: *TextInputCutCopyPaste*

overwrite/extend *kivy.uix.textinput.TextInputCutCopyPaste* w/ translatable and autocomplete options.

**\_\_init\_\_**(*\*\*kwargs*)

create *Bubble* instance to display the cut/copy/paste options.

the monkey patch of *TextInputCutCopyPaste* which was done in *FlowInput.\_show\_cut\_copy\_paste()* has to be temporarily reset before the *super()* call below, to prevent endless recursion because else the other *super(cls, instance)* call (in python2 style within *TextInputCutCopyPaste.\_\_init\_\_()*) results in the same instance (instead of the overwritten instance).

**on\_parent**(*instance, value*)

overwritten to translate *BubbleButton* texts and to add extra menus to add/delete ac texts.

**Parameters**

- **instance** (Widget) – self.
- **value** (Widget) – kivy main window.

**class FlowInput**(*\*\*kwargs*)

Bases: *HelpBehavior*, *ShadersMixin*, *TextInput*

text input/edit widget with optional autocompletion.

until version 0.1.43 of this portion the background and text color of *FlowInput* did automatically get switched by a change of the `light_theme` app state. now all colors left unchanged (before only the ones with `<unchanged>`):

```
* background_color: Window.clearcolor          # default: 1, 1, 1, 1
* cursor_color: app.font_color                 # default: 1, 0, 0, 1
* disabled_foreground_color: <unchanged>       # default: 0, 0, 0, .5
* foreground_color: app.font_color             # default: 0, 0, 0, 1
* hint_text_color: <unchanged>                 # default: 0.5, 0.5, 0.5, 1
↪ 1.0
* selection_color: <unchanged>                 # default: 0.1843, 0.6549,
↪ 0.8313, .5
```

to implement a dark background for the dark theme we would need also to change the images in the properties:

`background_active`, `background_disabled_normal` and `self.background_normal`.

the images/colors of the bubble that is showing e.g. on long press of the `TextInput` widget (cut/copy/paste/...) kept unchanged - only the `font_size` get adapted and the bubble button texts get translated. for that the class *ExtTextInputCutCopyPaste* provided by this portion inherits from the original bubble class `TextInputCutCopyPaste`.

the original bubble class is getting monkey patched shortly/temporarily in the moment of the instantiation to translate the bubble menu options, change the font sizes and add additional menu options to memorize/forget auto-completion texts.

#### **focus\_flow\_id**

flow id that will be set when this widget get focus

#### **unfocus\_flow\_id**

flow id that will be set when this widget lost focus

#### **auto\_complete\_texts: List[str]**

list of autocompletion texts

#### **auto\_complete\_selector\_index\_ink: Union[Tuple[float, float, float], List[float], Tuple[float, float, float, float]]**

color and alpha used to highlight the currently selected text of all matching autocompletion texts

#### **\_ac\_dropdown: Any = None**

singleton `FlowDropDown` instance for all `TextInput` instances

#### **\_matching\_ac\_texts: List[str] = []**

one list instance for all `TextInput` instances is enough

#### **\_matching\_ac\_index: int = 0**

index of selected text in the dropdown matching texts list

#### **\_\_init\_\_(\*\*kwargs)**

#### **\_\_repr\_\_()**

added for easier debugging.

#### **\_change\_selector\_index(delta)**

change/update/set the index of the matching texts in the opened autocompletion dropdown.

**Parameters**

**delta** (int) – index delta value between old and new index (e.g. pass +1 to increment index). set index to zero if the old/last index was on the last item in the matching list.

**\_delete\_ac\_text**(*ac\_text=""*)

**delete\_text\_from\_ac**(\**\_args*)

check if current text is in autocompletion list and if yes then remove it.

called by FlowInput kbd event handler and from menu button added by ExtTextInputCutCopyPaste.on\_parent().

**Parameters**

**\_args** – unused event args.

**extend\_ac\_with\_text**(\**\_args*)

add non-empty text to autocompletion texts.

**Parameters**

**\_args** – unused event args.

**keyboard\_on\_key\_down**(*window, keycode, text, modifiers*)

overwritten TextInput/FocusBehavior kbd event handler.

**Parameters**

- **window** (Any) – keyboard window.
- **keycode** (Tuple[int, str]) – pressed key as tuple of (numeric key code, key name string).
- **text** (str) – pressed key value string.
- **modifiers** (List[str]) – list of modifier keys (pressed or locked).

**Return type**

bool

**Returns**

True if key event get processed/used by this method.

**keyboard\_on\_textinput**(*window, text*)

overridden to suppress any user input if tour is running/active.

**on\_focus**(*\_self, focus*)

change flow on text input change of focus.

**Parameters**

- **\_self** (Widget) – unused dup ref to self.
- **focus** (bool) – True if this text input got focus, False on unfocus.

**on\_text**(*\_self, text*)

TextInput.text change event handler.

**Parameters**

- **\_self** (Widget) – unneeded duplicate reference to TextInput/self.
- **text** (str) – new/current text property value.

**\_select\_ac\_text**(*selector*)

put selected autocompletion text into text input and close \_ac\_dropdown

---

```
_show_cut_copy_paste(*args, **kwargs)
```

Show a bubble with cut copy and paste buttons

```
class FlowPopup(**kwargs)
```

Bases: `ModalBehavior`, `DynamicChildrenBehavior`, `SlideSelectBehavior`, `ReliefCanvas`, `BoxLayout`

popup for dynamic and auto-sizing dialogs and other top-most or modal windows.

the scrollable `container` (a `ScrollView` instance) can only have one child, referenced by the `content` attribute, which can be any widget (e.g. a label). use a layout for `content` to display multiple widgets. set `optimal_content_width` and/or `optimal_content_height` to make the popup size as small as possible, using e.g. `minimum_width` respectively `minimum_height` if `content` is a layout that is providing and updating this property, or `text_size_guess()` if it is a label or button widget.

---

**Hint:** `texture_size` could provide a more accurate size than `text_size_guess()`, but should be used with care to prevent recursive property change loops.

---

this class is very similar to `Popup` and can be used as replacement, incompatible are the following attributes of `Popup` and `ModalView`:

- `background`: FlowPopup has no `BorderImage`.
- `border`: FlowPopup is using a `RoundedRectangle`.
- `title_align`: is 'center' and could be changed via the `title_bar` id.
- `title_color` is the `app.font_color`.
- `title_font` is the default font.
- `title_size` is the default button height (`button_height`).

#### Events

**`on_pre_open:`**

fired before the FlowPopup is opened and got added to the main window.

**`on_open:`**

fired when the FlowPopup is opened.

**`on_pre_dismiss:`**

fired before the FlowPopup is closed.

**`on_dismiss:`**

fired when the FlowPopup is closed. if the callback returns True, the popup will stay opened.

**`close_kwargs`**

kwargs passed to all close action flow change event handlers.

`close_kwargs` is a `DictProperty`. the default depends the action of the penultimate flow id in the `ae.gui_app.flow_path`: is empty or 'enter' dict then it defaults to an empty flow, else to an empty dict.

**`container: Widget`**

popup scrollable layout underneath the title bar and the parent of the `content` container.

`container` is an `ObjectProperty` and is read-only.

**content**

popup main content container, displayed as a child of the scrollable layout *container*.

*content* is an *ObjectProperty* and has to be specified either in the kv language as children or via the *content* kwarg.

**menu\_items**

sequence of the content widgets and close button.

*menu\_items* is an *ObjectProperty* and includes by default the content widgets as well as the close button of this popup.

**optimal\_content\_width**

width of the content to be fully displayed/visible.

*optimal\_content\_width* is a *NumericProperty*. if *0* or *None* or not explicitly set then it defaults to the main window width and - in landscape orientation - minus the *side\_spacing* and the width needed by the *query\_data\_maps* widgets.

**optimal\_content\_height**

height of the content to be fully displayed/visible.

*optimal\_content\_height* is a *NumericProperty*. if *0* or *None* or not explicitly set then it defaults to the main window height minus the height of *title* and - in portrait orientation - minus the *side\_spacing* and the height needed by the *query\_data\_maps* widgets.

**parent\_popup\_to\_close**

parent popup widget instance to be closed if this popup closes.

*parent\_popup\_to\_close* is a *ObjectProperty* and defaults to *None*.

**query\_data\_maps: List[Dict[str, Any]]**

list of child data dicts to instantiate the query widgets (most likely *FlowButton*) of this popup.

*query\_data\_maps* is a *ListProperty* and defaults to an empty list.

**separator\_height**

height of the separator.

*separator\_height* is a *NumericProperty* and defaults to 3sp.

**side\_spacing**

padding in pixels from Window.width in landscape-orientation, and from Window.height in portrait-orientation.

*side\_spacing* is a *NumericProperty* and defaults to 192sp.

**title**

title string of the popup.

*title* is a *StringProperty* and defaults to an empty string.

**\_anim\_alpha**

internal opacity/alpha for fade-in/-out animations

**\_anim\_duration**

internal time in seconds for fade-in/-out animations

**\_max\_height**

popup max height (calculated from Window/side\_spacing)

**\_max\_width**

popup max width (calculated from Window/side\_spacing)

**\_\_events\_\_** = ('on\_pre\_open', 'on\_open', 'on\_pre\_dismiss', 'on\_dismiss')

**\_\_init\_\_**(\*\*kwargs)

**background\_color**

background ink tuple in the format (red, green, blue, alpha).

the *background\_color* is a *ColorProperty* and defaults to clearcolor.

**overlay\_color**

ink (color + alpha) tuple in the format (red, green, blue, alpha) used for dimming of the main window.

*overlay\_color* is a *ColorProperty* and defaults to the current color value clearcolor with an alpha of 0.6 (set in *\_\_init\_\_()*).

**separator\_color**

color used by the separator between title and the content-/container-layout.

*separator\_color* is a *ColorProperty* and defaults to the current value of the *font\_color* property.

**\_\_repr\_\_()**

added for easier debugging.

**add\_widget**(widget, index=0, canvas=None)

add container and content widgets.

first call set container from kv rule, 2nd the content, 3rd raise error.

**Parameters**

- **widget**¶ (*Widget*) – widget instance to be added.
- **index**¶ (*int*) – index kwarg of *kivy.uix.widget.Widget()*.
- **canvas**¶ (*Optional[str]*) – canvas kwarg of *kivy.uix.widget.Widget()*.

**close**(\*args, \*\*kwargs)

close/dismiss container/layout (ae.gui\_app popup handling compatibility for all GUI frameworks).

---

**Note:** prevents close/dismiss of any dropdown/popup while clicking on help activator widget.

---

**Parameters**

- **\_args**¶ – arguments (to have compatible signature for DropDown/Popup/ModalView widgets).
- **kwargs**¶ – keyword arguments (compatible signature for DropDown/Popup/ModalView widgets).

**dismiss**(\*args, \*\*kwargs)

alias method of *close()*

**on\_\_anim\_alpha**(instance, value)

\_anim\_alpha changed event handler.

**on\_content**(*\_instance, value*)

optional single widget (to be added to the container layout) set directly or via FlowPopup kwargs.

**on\_dismiss**()

default dismiss/close event handler.

**Return type**

`Optional[bool]`

**Returns**

return True to prevent/cancel the dismiss/close.

**on\_open**()

open default event handler.

**on\_pre\_dismiss**()

pre close/dismiss event handler.

**on\_pre\_open**()

pre open default event handler.

**open**(\**\_args*, \*\**kwargs*)

start optional open animation after calling open method if exists in inheriting container/layout widget.

**Parameters**

- ***\_args*** – unused argument (to have compatible signature for Popup/ModalView and DropDown widgets passing the parent widget).
- ***kwargs*** – extra arguments that are removed before to be passed to the inheriting open method:
  - `'animation': False` will disable the fade-in-animation (default=True).

**class FlowSelector**(\*\**kwargs*)

Bases: `ModalBehavior`, `DynamicChildrenBehavior`, `FlowButton`

attachable popup used for dynamic elliptic auto-spreading menus and toolbars.

this app flow based menu-like popup consists of a central button and animated elliptic-auto-spreading menu items.

any widget class can be used for the menu items of this class, although ShortenedButton instances are best-prepared to auto-shorten the *text* property.

**Events**

***on\_pre\_open*:**

fired before the FlowSelector is opened and got added to the main window.

***on\_open*:**

fired when the FlowSelector is opened.

***on\_pre\_dismiss*:**

fired before the FlowSelector is closed.

***on\_dismiss*:**

fired when the FlowSelector is closed. if the callback returns True, the menu will stay opened.

inspired by <https://github.com/kivy-garden/garden.modernmenu>



**attached\_widget**

widget from which this instance got opened.

The `open()` method will automatically set this property whilst `close()` will set it back to `None`.

**close\_kwargs**

kwargs passed to all close action flow change event handlers.

`close_kwargs` is a `DictProperty`. the default depends the action of the penultimate flow id in the `ae.gui_app.flow_path`: is empty or 'enter' dict then it defaults to an empty flow, else to an empty dict.

**is\_open**

`True` if the `open()` method of this instance got called. `close()` sets this value to `False`.

`is_open` is a `BooleanProperty` and defaults to `False`.

**parent\_popup\_to\_close**

parent menu/popup widget instance to be closed if this menu closes.

`parent_popup_to_close` is a `ListProperty` and defaults to an empty list.

**radian\_offset**

start/end angle offset (in radians) for the elliptically positioned items of an elliptic menu.

`radian_offset` is a `NumericProperty` and defaults to 9 degrees (as radian, respectively  $\tau * 9.0 / 360.0$ ).

**separator\_height**

line width of the border of the menu-back-button and of the connectors between the menu and its items.

`separator_height` is a `NumericProperty` and defaults to 1sp.

**scale\_x**

spread/widen factor of the menu item ellipse in x direction.

`scale_x` is a `NumericProperty` and defaults to 1.0.

**scale\_y**

spread/widen factor of the menu item ellipse in y direction.

`scale_y` is a `NumericProperty` and defaults to 1.0.

**\_anim\_alpha**

internal opacity/alpha for fade-in/-out animations

**\_anim\_duration**

internal time in seconds for fade-in/-out animations

**\_creation\_direction = 1.0**

creation-ellipse-direction of the menu buttons/items

**\_start\_radian = 0.0**

angle of the first menu item

**\_item\_radian = 0.0**

angle of each menu item

**\_\_events\_\_ = ('on\_pre\_open', 'on\_open', 'on\_pre\_dismiss', 'on\_dismiss')****\_\_init\_\_(\*\*kwargs)**

**overlay\_color**

ink (color + alpha) tuple in the format (red, green, blue, alpha) used for dimming of the main window.

*overlay\_color* is a *ColorProperty* and defaults to the current color value *clearcolor* with an alpha of 0.6 (set in *\_\_init\_\_()*).

**separator\_color**

color used to draw the border of the menu-back-button.

*separator\_color* is a *ColorProperty* and defaults to the current value of the *font\_color* property.

**container: Widget**

parent widget of the menu items (for compatibility with other popups/dropdowns).

*container* is an *ObjectProperty* and is read-only.

**menu\_items**

sequence of the menu items widgets (for compatibility with *SlideSelectBehavior*).

*menu\_items* is an *ListProperty* and defaults to the items specified via the *child\_data\_maps* property and the kv language.

**\_attached\_pos(widget, \_pos)****\_attached\_size(widget, \_size)****\_finalize\_close(\*\_args)**

final real dismiss after animations are finished.

**\_item\_moved(\_anim, item, progress)**

draw line from menu-center to animated item-center.

**\_layout\_items(\*\_args)**

calculate the radians/angles and positions of the menu-items.

given an offset angle of 9 degrees the (start-angle direction menu-item-angle) for the 9 block regions from top-left ... center/middle ... bottom-right would be:

Table 4.4: possibly blocked window regions

	left	center	right
top	0 -72	180+180	180 +72
middle	81-162	90+360	99+162
bottom	72 -72	180-180	108 +72

**add\_widget(widget, index=0, canvas=None)**

manage children added via kv, python and *ae.kivy\_dyn\_chi.DynamicChildrenBehavior*.

**close(\*\_args, \*\*kwargs)**

close/dismiss menu (ae.gui\_app popup handling compatibility for all GUI frameworks).

---

**Note:** prevents close/dismiss of any dropdown/popup while clicking on help activator widget.

---

**Parameters**

- *\_args* – arguments (to have compatible signature for DropDown/Popup/ModalView widgets).

- **kwargs** – keyword arguments (compatible signature for Drop-Down/Popup/ModalView widgets): *force*: pass *True* to force closing, ignoring return value of *dispatch('on\_dismiss')* *animation*: pass *False* to close this menu without fade-out animation

**dismiss(\*\_args, \*\*kwargs)**

alias method of *close()*

**on\_dismiss()**

dismiss/close default event handler.

**Return type**

*Optional*[*bool*]

**Returns**

return *True* to prevent/cancel the dismiss/close.

**on\_open()**

open default event handler.

**on\_pre\_dismiss()**

pre close/dismiss event handler.

**on\_pre\_open()**

pre open default event handler.

**on\_release()**

touch release default event handler.

**open(attach\_to, \*\*kwargs)**

display flow selector menu items, with animation and as a popup in modal mode.

**Parameters**

- **attach\_to** (*Widget*) – the widget to which this menu gets attached to.
- **kwargs** (*Dict*[*str*, *Any*]) – extra arguments that are removed before to be passed to the inheriting open method:
  - *'animation'*: *False* will disable the fade-in-animation (default=*True*).

**remove\_widget(widget)**

sync self.menu\_items with self.children.

**touch\_pos\_is\_inside(pos)**

is touch inside of this widget or a group of sub-widgets. overwritten to also include the menu items.

**Parameters**

**pos** (*List*[*float*]) – touch position (x, y) in window coordinates.

**Return type**

*bool*

**Returns**

*True* if this menu and its items would process a touch event at *pos*.

**class FlowToggler(\*\*kwargs)**

Bases: *HelpBehavior*, *SlideSelectBehavior*, *TouchableBehavior*, *ToggleButtonBehavior*, *ImageLabel*

toggle button changing flow id.

**long\_tap\_flow\_id**

flow id that will be set when this button gets long tap event

**tap\_flow\_id**

the new flow id that will be set when this toggle button get released

**tap\_kwargs**

kwargs dict passed to event handler (change\_flow) when button get tapped

**\_\_init\_\_**(\*\*kwargs)

**on\_long\_tap**(touch)

long tap/click default handler.

**Parameters**

**touch** *¶* ([MotionEvent](#)) – motion/touch event data with the touched widget in *touch.grab\_current*.

**on\_release**()

overridable touch release event handler.

**class MessageShowPopup**(\*\*kwargs)

Bases: [FlowPopup](#)

flow popup to display info or error messages.

**\_container**: [Widget](#)

**attach\_to**: [Optional](#)[[Widget](#)]

**\_layout\_finished**: [bool](#)

**\_opened\_item**: [Optional](#)[[Widget](#)]

**\_touch\_moved\_outside**: [bool](#)

**message**

popup window message text to display

**title**

popup window title text to display

**class Tooltip**(\*\*kwargs)

Bases: [ScrollView](#)

semi-transparent and optional click-through container to display help and tour page texts.

**tip\_text**

tooltip text string to display.

*tip\_text* is a [StringProperty](#) and defaults to an empty string.

**anchor\_spe**

anchor pos and direction, see [AnchorSpecType](#) (read-only)

**has\_tour**

True if a tour exists for the current app flow/help context (read-only)

**tap\_thru**

True if user can tap widgets behind/covered by this tooltip win (read-only)

**tour\_start\_pos**

screen position of the optionally displayed tour start button (read-only)

**tour\_start\_size**

size of the optionally displayed tour start button (read-only)

**\_\_init\_\_**(\*\**kwargs*)**targeted\_widget**

target widget to display tooltip text for (mostly a button, but could any, e.g. a layout widget).

*targeted\_widget* is a `ObjectProperty` and defaults to the main app `help_activator`.

**\_actual\_pos**(\**args*)**Return type**

`Tuple[float, float]`

**collide\_tap\_thru\_toggler**(*touch\_x*, *touch\_y*)

check if touch is on the tap through toggler pseudo button.

**Parameters**

- **touch\_x** *(float)* – window x position of touch.
- **touch\_y** *(float)* – window y position of touch.

**Return type**

`bool`

**Returns**

True if user touched the tap through toggler.

**collide\_tour\_start\_button**(*touch\_x*, *touch\_y*)

check if touch is on the tap through toggler pseudo button.

**Parameters**

- **touch\_x** *(float)* – window x position of touch.
- **touch\_y** *(float)* – window y position of touch.

**Return type**

`bool`

**Returns**

True if user touched the tap through toggler.

**on\_size**(\**args*)

(re-)position `help_activator` tooltip correctly after help text loading and layout resizing.

**on\_targeted\_widget**(\**args*)

targeted widget changed event handler.

**Parameters**

**\_args** – change event args (unused).

**on\_touch\_down**(*touch*)

check for additional events added by this class.

**Parameters**

**touch** *(MotionEvent)* – motion/touch event data.

**Return type**`bool`**Returns**

True if event got processed/used.

## 4.35 ae.kivy.apps

### 4.35.1 ae.kivy.apps module

this module is providing two application classes, one of them extending the `Kivy App class`. the other app class is used as main app class, extending `HelpAppBase` with additional attributes and helper methods.

#### application classes

the class `KivyMainApp` is implementing a main app class, reducing the amount of code needed to create a Python application based on the `Kivy framework`.

`KivyMainApp` is based on the following classes:

- the abstract base class `HelpAppBase` which adds context-sensitive help.
- the abstract base class `MainAppBase` which adds *application status*, *app state variables*, *app state constants*, *application flow* and *application events*.
- `ConsoleApp` is adding *config files*, *config variables* and *config options*.
- `AppBase` is adding *application logging* and *application debugging*.

this namespace portion is also encapsulating the `Kivy App class` via the `FrameworkApp` class. this Kivy app class instance can be directly accessed from the main app class instance via the `framework_app` attribute.

#### kivy app config variables

all the *config variables* and app constants inherited from the base app classes are available.

---

**Hint:** please see the documentation of the namespace portions/modules `ae.console`, `ae.gui_app` and `ae.gui_help` for more detailed information on all the inherited *config variables*, *config options*, *config files* and *app state constants*.

---

the additional *config variables* `win_min_width` and `win_min_height`, added by this portion, you can optionally restrict the minimum size of the kivy main window of your app. their default values are set on app startup in the method `on_app_run()`.

more constants provided by this portion are declared in the `widgets` module.

## kivy application events

the main app class is firing *application events*, additional to the ones provided by *MainAppBase*, by redirecting events of Kivy's *App* class. these framework app events get fired after the event *on\_app\_run()*, in the following order (the Kivy event/callback-method name is given in brackets):

- *on\_app\_build* (kivy.app.App.build, after the main kv file get loaded).
- *on\_app\_built* (kivy.app.App.build, after the root widget get build).
- *on\_app\_start* (kivy.app.App.on\_start)
- *on\_app\_started* (one clock tick after *on\_app\_start*/kivy.app.App.on\_start)
- *on\_app\_pause* (kivy.app.App.on\_pause)
- *on\_app\_resume* (kivy.app.App.on\_resume)
- *on\_app\_stop* (kivy.app.App.on\_stop)
- *on\_app\_stopped* (one clock tick after *on\_app\_stop*)

## Classes

<i>FrameworkApp</i> (main_app, **kwargs)	Kivy framework app class proxy redirecting events and callbacks to the main app class instance.
<i>KivyMainApp</i> (**console_app_kwargs)	Kivy application

**class FrameworkApp**(main\_app, \*\*kwargs)

Bases: *App*

Kivy framework app class proxy redirecting events and callbacks to the main app class instance.

### app\_states

duplicate of MainAppBase app state for events/binds

### button\_height

default button height, dynamically calculated from font size

### displayed\_help\_id

help id of the currently explained/help-target widget

### font\_color

rgba color of the font used for labels/buttons/...

### help\_layout

layout widget if help mode is active else None

### landscape

True if app win width is bigger than the app win height

### max\_font\_size

maximum font size in pixels bound to window size

### min\_font\_size

minimum - “ -

**mixed\_back\_ink**

background color mixed from available back inks

**tour\_layout**

overlay layout widget if tour is active else None

**\_\_init\_\_**(*main\_app*, *\*\*kwargs*)

init kivy app

**main\_app**

set reference to KivyMainApp instance

**build()**

kivy build app callback.

**Return type**

`Widget`

**Returns**

root widget (Main instance) of this app.

**key\_press\_from\_kivy**(*keyboard*, *key\_code*, *\_scan\_code*, *key\_text*, *modifiers*)

convert and redistribute key down/press events coming from `Window.on_key_down`.

**Parameters**

- **keyboard** `(Any)` – used keyboard.
- **key\_code** `(int)` – key code of pressed key.
- **\_scan\_code** `(int)` – key scan code of pressed key.
- **key\_text** `(Optional[str])` – key text of pressed key.
- **modifiers** `(List[str])` – list of modifier keys (including e.g. ‘capslock’, ‘num-lock’, ...)

**Return type**

`bool`

**Returns**

True if key event got processed used by the app, else False.

**key\_release\_from\_kivy**(*keyboard*, *key\_code*, *\_scan\_code*)

key release/up event.

**Return type**

`bool`

**Returns**

return value of call to `on_key_release` (True if ke got processed/used).

**on\_pause()**

app pause event automatically saving the app states.

emits the `on_app_pause` event.

**Return type**

`bool`

**Returns**

True.



**on\_resume()**

app resume event automatically loading the app states.

emits the *on\_app\_resume* event.

**Return type**

`bool`

**Returns**

True.

**on\_start()**

kivy app start event.

called after *run\_app()* method, after Kivy created the main layout (by calling its *build()* method) and has attached it to the main window.

emits the events: *on\_app\_start* and *on\_app\_started*.

**on\_stop()**

quit app event automatically saving the app states.

emits the *on\_app\_stopped* event whereas the method *stop\_app()* emits the *on\_app\_stop* event.

**win\_pos\_size\_change(\*\_)**

resize handler updates: *win\_rectangle*, *landscape*.

**class KivyMainApp**(*\*\*console\_app\_kwargs*)

Bases: *HelpAppBase*

Kivy application

**documents\_root\_path:** `str = '.'`

root file path for app documents, e.g. for import/export

**get\_txt\_**(*text*, *count=None*, *language=""*, *loc\_vars=None*, *\*\*kwargs*): `Any =`  
`<ae.kivy.i18n._GetTextBinder object>`

make i18n translations available via main app instance

**kbd\_input\_mode:** `str = 'scale'`

optional app state to set Window[Base].softinput\_mode

**tour\_overlay\_class**

Kivy main app tour overlay class

alias of *TourOverlay*

**\_debug\_enable\_clicks:** `int = 0`

**init\_app**(*framework\_app\_class=<class 'ae.kivy.apps.FrameworkApp'>*)

initialize framework app instance and prepare app startup.

**Parameters**

**framework\_app\_class**¶ (*Type[FrameworkApp]*) – class to create app instance (optionally extended by app project).

**Return type**

*Tuple[Optional[Callable], Optional[Callable]]*

**Returns**

callable to start and stop/exit the GUI event loop.

**app\_env\_dict()**

collect run-time app environment data and settings.

**Return type**

`Dict[str, Any]`

**Returns**

dict with app environment data/settings.

**call\_method\_delayed(delay, callback, \*args, \*\*kwargs)**

delayed call of passed callable/method with args/kwargs catching and logging exceptions preventing app exit.

**Parameters**

- **delay** `(float)` – delay in seconds before calling the callable/method specified by `callback`.
- **callback** `(Union[Callable, str])` – either callable or name of the main app method to call.
- **args** – args passed to the callable/main-app-method to be called.
- **kwargs** – kwargs passed to the callable/main-app-method to be called.

**Return type**

`Any`

**Returns**

delayed call event (in Kivy of `Type[ClockEvent]`) providing a *cancel* method to allow the cancellation of the delayed call within the delay time.

**change\_light\_theme(light\_theme)**

change font and window clear/background colors to match ‘light’/‘black’ themes.

**Parameters**

**light\_theme** `(bool)` – pass True for light theme, False for black theme.

**static class\_by\_name(class\_name)**

resolve kv widgets

**Return type**

`Optional[Type]`

**static dpi\_factor()**

dpi scaling factor - overridden to use Kivy’s dpi scaling.

**Return type**

`float`

**ensure\_top\_most\_z\_index(widget)**

ensure visibility of the passed widget to be the foremost in the z index/order.

**Parameters**

**widget** `(Widget)` – widget to check and possibly correct to be the foremost one.

if other dropdown/popup opened after the passed widget/layout, then only correct z index/order to show this widget/layout as popup (in front, as foremost widget). if the passed widget has a method named *activate\_modal* (like e.g. `ae.kivy.behaviors.ModalBehavior.activate_modal()`) then it will be called.

**global\_variables(\*\*patches)**

overridden to add Kivy-specific globals.

**Return type**

`Dict[str, Any]`

**help\_activation\_toggle()**

button tapped event handler to switch help mode between active and inactive (also inactivating tour).

**load\_sounds()**

override to preload audio sounds from app folder snd into sound file cache.

**on\_app\_build()**

kivy App build event handler called at the beginning of `kivy.app.App.build()`.

**on\_app\_built()**

kivy App build event handler called at the end of `kivy.app.App.build()`.

**on\_app\_pause()**

kivy `on_pause()` event handler.

**on\_app\_resume()**

kivy `on_resume()` event handler.

**on\_app\_run()**

run app event handler - used to set the user preference app states and initial window pos and size.

**on\_app\_start()**

app start event handler - triggered by `FrameworkApp.on_start()`.

**on\_app\_started()**

kivy `on_start()` event handler (called after `on_app_build/on_app_built`).

**on\_app\_stopped()**

kivy `on_stop()` event handler (called after `on_app_stop`).

**on\_flow\_widget\_focused()**

set focus to the widget referenced by the current flow id.

**on\_kbd\_input\_mode\_change(mode, \_event\_kwargs)**

language app state change event handler.

**Parameters**

- **mode** `(str)` – the new softinput\_mode string (passed as flow key).
- **\_event\_kwargs** `(Dict[str, Any])` – unused event kwargs.

**Return type**

`bool`

**Returns**

True to confirm the language change.

**on\_lang\_code()**

language code app-state-change-event-handler to refresh kv rules.

**on\_light\_theme()**

theme app-state-change-event-handler.

**on\_user\_preferences\_open**(*\_flow\_id*, *\_event\_kwargs*)

enable debug mode after clicking 3 times within 6 seconds.

**Parameters**

- **\_flow\_id** (str) – (unused).
- **\_event\_kwargs** (Dict[str, Any]) – (unused).

**Return type**

bool

**Returns**

False for *on\_flow\_change()* get called, opening user preferences popup.

**play\_beep**()

make a short beep sound.

**play\_sound**(*sound\_name*)

play audio/sound file.

**play\_vibrate**(*pattern*=(0.0, 0.09, 0.09, 0.18, 0.18, 0.27, 0.18, 0.36, 0.27, 0.45))

play vibrate pattern.

**open\_popup**(*popup\_class*, *\*\*popup\_kwargs*)

open Popup or DropDown using the *open* method. overwriting the main app class method.

**Parameters**

- **popup\_class** (Type[Union[FlowPopup, Popup, DropDown]]) – class of the Popup or DropDown widget.
- **popup\_kwargs** – args to be set as attributes of the popup class instance plus an optional *opener* kwarg that will pass the popup opener widget to the popup.open() method; if *opener* gets not specified then the framework window will be used.

**Return type**

Widget

**Returns**

created and displayed/opened popup class instance.

**text\_size\_guess**(*text*, *font\_size*=0.0, *padding*=(0.0, 0.0))

quickly roughly pre-calculate texture size of a multi-line string without rendering.

**Parameters**

- **text** (str) – text string which can contain line feed characters.
- **font\_size** (float) – the font size to pseudo-render the passed text; using the value of *font\_size* as default if not passed.
- **padding** (Tuple[float, float]) – optional padding in pixels for x and y coordinate (totals for left+right/top+bottom).

**Return type**

Tuple[float, float]

**Returns**

roughly the size (width, height) to display the string passed into *text*. more exactly size would need to use internal render methods of Kivy, like e.g. *\_get\_text\_width()* and *get\_extents()*.

**static** `widget_pos(wid)`

return widget's window x/y position (overridden for absolute coordinates relative/scrollable layouts).

**Parameters**

`wid` – widget to determine the position of.

**Return type**

`Tuple[float, float]`

**Returns**

tuple of x and y screen coordinate.

## 4.36 ae.kivy.behaviors

### 4.36.1 ae.kivy.behaviors module

this module provides the following behavior classes:

- *HelpBehavior* extends and prepares any Kivy widget to show an individual help text for it.
- *ModalBehavior* is a generic mix-in class that provides modal behavior to any container widget.
- *SlideSelectBehavior*: quickly navigate in elliptically-shaped sub-/menus, alternatively starting with a long touch, then slide to the menu item to select and release.
- *TouchableBehavior*: extends toggle-/touch-behavior of *ButtonBehavior*.

#### help behaviour mixin

to show a i18n translatable help text for a Kivy widget create a subclass of the widget and add the mixin-/behavior-class *HelpBehavior*. the following example is attaching a help text to the Kivy *Button* widget:

```
from kivy.uix.button import Button
from ae.kivy.widgets import HelpBehavior

class ButtonWithHelpText(HelpBehavior, Button):
    ...
```

alternatively you can archive this via the definition of a new kv-lang rule, like shown underneath:

```
<ButtonWithHelpText@HelpBehavior+Button>
```

**Note:** to automatically lock and mark the widget you want to add help texts for, this mixin class has to be specified as the first inheriting class in the class or rule declaration.

latest :

---

## modal behavior mixin

to convert a container widget into a modal dialog, add the *ModalBehavior* mix-in class, provided by this ae namespace portion.

the following code snippet demonstrates a typical implementation:

```
class MyContainer(ModalBehavior, BoxLayout):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)

    def open(self):
        self.activate_esc_key_close()
        self.activate_modal()

    def close(self):
        self.deactivate_esc_key_close()
        self.deactivate_modal()
```

calling the method *activate\_esc\_key\_close()* in the *open* method of a container class allows the user to close the popup by pressing the Escape key (or Back on Android). this optional feature can be reverted by calling the *deactivate\_esc\_key\_close()* method in your *close* method.

to additionally activate the modal mode call the method *activate\_modal()*. the modal mode can be deactivated by calling the *deactivate\_modal()* method.

all touch, mouse and keyboard user interactions will be consumed or filtered after activating the modal mode. therefore it is recommended to also visually change the GUI while in the modal mode, which has to be implemented by the mixing-in container widget.

---

**Hint:** usage examples of the *ModalBehavior* mix-in are e.g. the classes *TourOverlay* and *FlowPopup*.

---

## Module Attributes

<i>TOUCH_VIBRATE_PATTERN</i>	very short/~0.3s vibrate pattern for button and toggler touch.
------------------------------	--

## Classes

<i>HelpBehavior()</i>	behaviour mixin class for widgets providing help texts.
<i>ModalBehavior()</i>	mix-in to allow close on Escape/Back key and to optionally provide a modal mode to a container widget.
<i>SlideSelectBehavior(**kwargs)</i>	quickly navigate in sub-/menus, starting with a long touch, then slide to the menu item to select and release.
<i>TouchableBehavior(**kwargs)</i>	touch-/toggle-button mix-in class with shaders, animations and additional events for double/triple/long touches.

**TOUCH\_VIBRATE\_PATTERN = (0.0, 0.09, 0.09, 0.06, 0.03, 0.03)**

very short/~0.3s vibrate pattern for button and toggler touch.

## **class HelpBehavior**

Bases: `object`

behaviour mixin class for widgets providing help texts.

### **help\_id**

unique help id of the widget.

The correct identification of each help-aware widget presuppose that the attribute `help_id` has a unique value for each widget instance. This is done automatically for the widgets provided by the module `ae.kivy.widgets` by converting the app flow or app state of these widgets into a help id (see e.g. the implementation of the class `FlowButton`).

`help_id` is a `StringProperty` and defaults to an empty string.

### **help\_lock**

this property is True if the help mode is active and this widget is not the help target.

`help_lock` is a `BooleanProperty` and defaults to the value `False`.

### **help\_vars**

dict of extra data to displayed/render the help text of this widget.

The `help_vars` is a dict which can be used to provide extra context data to dynamically generate, translate and display individual help texts.

`help_vars` is a `DictProperty` and defaults to an empty dict.

### **\_shader\_args**

shader internal data / id

### **collide\_point: Callable**

### **on\_touch\_down(touch)**

prevent any processing if touch is done on the help activator widget or in active help mode.

#### **Parameters**

**touch** `¶` (`MotionEvent`) – motion/touch event data.

#### **Return type**

`bool`

#### **Returns**

True if event got processed/used.

## **class ModalBehavior**

Bases: `object`

mix-in to allow close on Escape/Back key and to optionally provide a modal mode to a container widget.

to make the container widget's modal state more obvious, add in your container widget an overlay color with an alpha between 0.3 and 0.9, together with the following canvas instructions:

**canvas:**

#### **Color:**

rgba: root.my\_overlay\_color[:3] + [root.my\_overlay\_color[-1] if self.is\_modal else 0]

#### **Rectangle:**

size: Window.size if self.is\_modal else (0, 0)

two rectangles will be needed to not overlay/fade-out the help activator button:

**canvas:**

**Color:**

rgba: self.my\_overlay\_color[:3] + [self.my\_overlay\_color[-1] if self.is\_modal else 0]

**Rectangle:**

**size:**

Window.width if self.is\_modal else 0, Window.height -  
app.main\_app.help\_activator.height if self.is\_modal else 0

**Rectangle:**

pos: app.main\_app.help\_activator.right, app.main\_app.help\_activator.y width:  
Window.width - app.main\_app.help\_activator.width if self.is\_modal else 0 height:  
app.main\_app.help\_activator.height

**center:** `List`

center position of `Widget`

**close:** `Callable`

method to dismiss the container widget (provided by self/container-widget)

**collide\_point:** `Callable`

method to detect collisions with other widgets of `Widget`

**disabled:** `bool`

disabled property of `Widget`

**fbind:** `Callable`

fast binding method of `Widget`

**funbind:** `Callable`

fast unbinding method of `Widget`

**unbind\_uid:** `Callable`

even faster unbinding method of `Widget`

**auto\_dismiss**

determines if the container is automatically dismissed when the user hits the Esc/Back key or clicks outside it.

`auto_dismiss` is a `BooleanProperty` and defaults to True.

**is\_modal**

flag if modal mode is active. use `activate_modal()` and `deactivate_modal()` to change this value.

`is_modal` is a `BooleanProperty` and defaults to False.

**\_center\_aligned:** `bool = False`

True if self will be repositioned to Window center

**\_fast\_bound\_center\_uid:** `int = 0`

if of center (pos and size) fbind/unbind\_uid

**\_touch\_started\_inside:** `Optional[bool] = None`

flag if touch started inside this widget or -group



**\_align\_center**(\*\_args)

reposition container to Window center.

**Parameters**

**\_args** – unused (passed only on bound window resize events)

**\_on\_key\_down**(\_window, key, \_scancode, \_codepoint, \_modifiers)

close/dismiss this popup if back/Esc key get pressed - allowing stacking with DropDown/FlowDropDown.

**activate\_esc\_key\_close**()

activate key press handler, calling self.close() if Escape/Back key get pressed.

**activate\_modal**(align\_center=True)

activate or renew modal mode for the mixing-in container widget.

**Parameters**

**align\_center** (bool) – pass False to prevent the automatic alignment of **center** to center on reposition or resize of self or on resize of Window.

**deactivate\_esc\_key\_close**()

deactivate keyboard event handler, activated via **activate\_esc\_key\_close**().

**deactivate\_modal**()

de-activate modal mode for the mixing-in container.

**on\_touch\_down**(touch)

touch down event handler, prevents the processing of a touch on the help activator widget by this popup.

**Parameters**

**touch** (MotionEvent) – motion/touch event data.

**Return type**

bool

**Returns**

True if event got processed/used.

**on\_touch\_move**(touch)

touch move event handler.

**Return type**

bool

**on\_touch\_up**(touch)

touch up event handler.

**Return type**

bool

**touch\_pos\_is\_inside**(pos)

check if the touch pos is inside of this widget or a group of sub-widgets.

**Parameters**

**pos** (List[float]) – touch position (x, y) in window coordinates.

**Return type**

bool

**Returns**

True if this widget or group would process a touch event at pos.

---

```
class SlideSelectBehavior(**kwargs)
```

Bases: `object`

quickly navigate in sub-/menus, starting with a long touch, then slide to the menu item to select and release.

the slide-select feature of this class allows a quicker select of any menu item, by opening any popup via the `on_long_tap()` event, then move the pointer/finger onto the menu item to select to finally release the touch. to enable this feature specify the touch event in the `touch_event` key of the `popup_kwargs` dict in the `change_flow()` call, e.g. by adding the following lines in your kv code onto the `FlowButton/FlowToggler` that is opening the popup:

```
on_long_tap:
    app.main_app.change_flow(id_of_flow('open', 'my_menu'),
    **update_tap_kwargs(self, popup_kwargs=dict(touch_event=args[1])))
```

---

**Note:** has to be inherited (to be in the MRO) before `ButtonBehavior`, respectively `ToggleButtonBehavior`, for the touch event get grabbed properly.

---

**attach\_to:** `Optional[Widget]`

**close:** `Callable`

**collide\_point:** `Callable`

**dispatch:** `Callable`

**to\_widget:** `Callable`

**\_\_init\_\_**(\*\*kwargs)

set normal pressed state shader on widget initialization.

**static** `_cancel_slide_select_closer(touch)`

**static** `_cancel_slide_select_opener(touch)`

`_grab_and_open(touch, item, first_close, *_args)`

**static** `_ungrab_and_close(touch, popup, *_args)`

**on\_touch\_move**(touch)

disable long touch on mouse/finger moves.

**Parameters**

**touch** `MotionEvent` – motion/touch event data.

**Return type**

`bool`

**Returns**

True if event got processed/used.

**on\_touch\_up**(touch)

disable long touch on mouse/finger up.

**Parameters**

**touch** `MotionEvent` – motion/touch event data.

**Return type**

`bool`

**Returns**

True if event got processed/used.

**class TouchableBehavior**(\*\*kwargs)

Bases: `object`

touch-/toggle-button mix-in class with shaders, animations and additional events for double/triple/long touches.

**Events*****on\_double\_tap:***

fired with the touch-down MotionEvent instance arg when a button get tapped twice within short time.

***on\_triple\_tap:***

fired with the touch-down MotionEvent instance arg when a button get tapped three times within short time.

***on\_long\_tap:***

fired with the touch-down MotionEvent instance arg when a button get tapped more than 2.4 seconds.

***on\_alt\_tap:***

fired with the touch-down MotionEvent instance arg when a button get either double, triple or long tapped.

---

**Note:** has to be inherited (to be in the MRO) before `ButtonBehavior`, respectively `ToggleButtonBehavior`, for the touch event get grabbed properly.

---

**add\_shader:** `Callable`

**center\_x:** `float`

**center\_y:** `float`

**collide\_point:** `Callable`

**del\_shader:** `Callable`

**disabled:** `bool`

**dispatch:** `Callable`

**main\_app:** `Any`

**state:** `str`

**\_touch\_anim**

widget-got-touched-animation

**\_touch\_x**

x pos moving in touch animation from initial touch to center pos

**\_touch\_y**

y pos moving in touch animation from initial touch to center pos

**\_\_events\_\_** = ('on\_alt\_tap', 'on\_double\_tap', 'on\_long\_tap', 'on\_triple\_tap')

**\_\_init\_\_**(\*\*kwargs)

set normal pressed state shader on widget initialization.

**down\_shader**

shader running if button is in pressed state 'down'.

*down\_shader* is a `DictProperty` and defaults to the `firestorm` shader. set to `None` to not render the default shader on button press/down.

**normal\_shader**

shader running if button is in pressed state 'normal'.

*normal\_shader* is a `DictProperty` and defaults to the `plunge wave` shader. set to `None` to not render a shader on button release/up.

**static \_cancel\_long\_touch\_clock**(touch)

**Return type**

`bool`

**on\_alt\_tap**(touch)

default handler for alternative tap (double, triple or long tap/click).

**Parameters**

**touch** `(MotionEvent)` – motion/touch event data with the touched widget in *touch.grab\_current*.

**on\_double\_tap**(touch)

double tap/click default handler.

**Parameters**

**touch** `(MotionEvent)` – motion/touch event data with the touched widget in *touch.grab\_current*.

**on\_down\_shader**(\*args)

button down state shader changed event handler.

**on\_long\_tap**(touch)

long tap/click default handler.

**Parameters**

**touch** `(MotionEvent)` – motion/touch event data with the touched widget in *touch.grab\_current*.

**on\_normal\_shader**(\*args)

button normal state shader changed event handler.

**on\_state**(\_widget, \_value)

button pressed state changed event handler, switching between 'normal' and 'down' state shader.

**Parameters**

- **\_widget** `(Any)` – button widget (is self).
- **\_value** `(str)` – new state value (either 'normal' or 'down').

**on\_touch\_down**(touch)

add sound, vibration and animation, check if tour is running and additional double/triple/alt touch events.

**Parameters**

**touch** `(MotionEvent)` – motion/touch event data.

**Return type**`bool`**Returns**

True if event got processed/used.

**on\_touch\_move(touch)**

disable long touch on mouse/finger moves.

**Parameters****touch** *(MotionEvent)* – motion/touch event data.**Return type**`bool`**Returns**

True if event got processed/used.

**on\_touch\_up(touch)**

disable long touch on mouse/finger up.

**Parameters****touch** *(MotionEvent)* – motion/touch event data.**Return type**`bool`**Returns**

True if event got processed/used.

**on\_triple\_tap(touch)**

triple tap/click default handler.

**Parameters****touch** *(MotionEvent)* – motion/touch event data with the touched widget in *touch.grab\_current*.**\_update\_shader()**

update shader on changed shader or button state.

## 4.37 ae.kivy.tours

### 4.37.1 ae.kivy.tours module

this module provides the following classes to augment the user interface of your apps with animated product tours, tutorials, walkthroughs and user onboarding/welcome features:

- *AnimatedTourMixin*
- *AnimatedOnboardingTour*
- *TourOverlay*

the class *TourOverlay* is implementing an overlay layout widget to display the animations, shaders, tour page texts, tooltip text and the navigation buttons of an active/running app tour.

the *AnimatedTourMixin* can be mixed-into a tour class that inherits from *TourBase* to extend it with animation and glsl shader features.

latest :

the class *AnimatedOnboardingTour* is based on *OnboardingTour* and *AnimatedTourMixin* to extend the generic app onboarding tour class with animations. it provides a generic app onboarding tour that covers the core features, that can be easily extended with app-specific tour pages.

to integrate a more app-specific onboarding tour into your app, simply declare a class with a name composed by the name of your app (app\_name) in camel-case, followed by the suffix '*OnboardingTour*'.

## Module Attributes

<i>PageAnimationType</i>	tuple of a widget id string and an <i>Animation</i> instance/evaluation-expression.
<i>PageAnimationsType</i>	tuple of <i>PageAnimationType</i> items
<i>WidgetValues</i>	a key of this dict specifies the name, the dict value the value of a widget property/attribute.
<i>DEF_FADE_OUT_APP</i>	default of tour layout fade out app screen factor

## Functions

<i>ani_start_check</i> (ani, wid)	start animation if needed else skip animation start.
<i>animated_widget_values</i> (wid, ani)	determine from a widget the attribute/property values animated/changed by an animation.
<i>restore_widget_values</i> (wid, values)	restore property values of a widget.

## Classes

<i>AnimatedOnboardingTour</i> (main_app)	onboarding tour, extended with animations and glsl shaders.
<i>AnimatedTourMixin</i> (main_app)	tour class mixin to add individual shaders to the tour layout and their children widgets.
<i>TourOverlay</i> (main_app[, tour_class])	tour layout/view overlay singleton class to display an active/running modal app tour with optional glsl shaders.

## PageAnimationType

tuple of a widget id string and an *Animation* instance/evaluation-expression.

if the first character of the widget id is a @ then the repeat attribute of the *Animation* instance will be set to True. the rest of the widget id string specifies the widget to be animated which is either:

- one of the widgets of the *TourOverlay* layout class, identified by the on of the following strings: 'next\_but', 'page\_lbl', 'tap\_pointer', 'prev\_but', 'title\_lbl', 'tooltip', 'tour\_page\_texts'.
- the explained widget if an empty string is given.
- the *TourOverlay* layout class instance for any other string (e.g. 'layout' or 'overlay').

alternative to an animation instance, a evaluation string can be specified. these evaluations allow to use the following globals: *Animation* (also abbreviated as A), Clock, *layout*, sp, Window and a reference to the instance of this app tour via *tour*.

alias of `Tuple[str, Union[Animation, str]]`

**PageAnimationsType**

tuple of *PageAnimationType* items

alias of `Tuple[str, Union[Animation, str], ...]`

**WidgetValues**

a key of this dict specifies the name, the dict value the value of a widget property/attribute.

alias of `Dict[str, Union[list, tuple, dict, float]]`

**DEF\_FADE\_OUT\_APP = 0.39**

default of tour layout fade out app screen factor

**ani\_start\_check(ani, wid)**

start animation if needed else skip animation start.

**Parameters**

- **ani** *(Animation)* – *Animation* instance.
- **wid** *(Widget)* – widget to start/skip the animation for.

**animated\_widget\_values(wid, ani)**

determine from a widget the attribute/property values animated/changed by an animation.

**Parameters**

- **wid** *(Widget)* – widget of which the animation property values will get retrieved.
- **ani** *(Union[Animation, CompoundAnimation])* – *Animation*/kivy.animation.CompoundAnimation instance.

**Return type**

`Dict[str, Union[list, tuple, dict, float]]`

**Returns**

dict with widget property names and values.

**restore\_widget\_values(wid, values)**

restore property values of a widget.

**Parameters**

- **wid** *(Widget)* – widget of which the animation property values will get restored.
- **values** *(Dict[str, Union[list, tuple, dict, float]])* – attribute/property values to restore on the widget.

**class AnimatedTourMixin(main\_app)**

Bases: *object*

tour class mixin to add individual shaders to the tour layout and their children widgets.

**layout:** *Widget*

**main\_app:** *Any*

**page\_ids:** *List[str]*

**page\_idx:** *int*

**setup\_texts:** *Callable*

`__init__(main_app)`

**pages\_animations:** `Dict[Optional[str], Tuple[Tuple[str, Union[Animation, str]], ...]]`

dict of compound animation instances of the pages of this tour.

the key of this dict is the page id or None (for animations available in all pages of this tour). each value of this dict is of the type *PageAnimationsType*.

**pages\_shaders:** `Dict[Optional[str], Tuple[Tuple[str, Dict[str, Any]], ...]]`

dict of widget shaders for the pages of this tour.

the key of this dict is the page id or None (for shaders available in all pages of this tour). each value of this dict is a tuple of tuples of widget id and `add_shader()`-kwargs.

the widget id string specifies the widget to which a shader will be added, which is either:

- one of the widgets of the *TourOverlay* layout class, identified by the on of the following strings: `'next_but'`, `'page_lbl'`, `'tap_pointer'`, `'prev_but'`, `'title_lbl'`, `'tooltip'`, `'tour_page_texts'`.
- the explained widget if an empty string is given.
- the *TourOverlay* layout class instance for any other string (e.g. `'layout'` or `'overlay'`).

before the `add_shader()`-kwargs dict will be passed to the *add\_shader()* method, all their non-string values, specifying as strings, will be evaluated/converted automatically. the evaluation provides the following globals: *layout*, *sp*, *Clock*, *Window* and the *tour* instance.

**switch\_next\_animations:** `Dict[Optional[str], Tuple[Tuple[str, Union[Animation, str]], ...]]`

dict of compound animation instances for the next page switch transition of the pages of this tour.

the key of this dict is the page id or None (for animations available in all pages of this tour). each value of this dict is of the type *PageAnimationsType*.

**\_add\_animations(animations)**

add animations to the tour page currently displayed in the tour layout/overlay.

#### Parameters

**animations** *(Tuple[Tuple[str, Union[Animation, str]], ...])* – tuple of tuples of widget id and animation instance/evaluation-string.

#### Returns

length of the longest animation added (in seconds).

**next\_page()**

overridden to add demo animations before/on switch to the next tour page.

**setup\_explained\_widget()**

overridden to bind pos/size of explained widget(s) to the tour layout/overlay placeholder.

#### Return type

*list*

#### Returns

list of explained widget instances.

**setup\_page\_shaders\_and\_animations()**

setup shaders and animations of the current page.

specified in *pages\_shaders* and *pages\_animations*.



**setup\_layout()**

overridden to set up animations and shaders of the current tour page.

**simulate\_text\_input**(*text\_input, text\_to\_delay, text\_to\_insert="", deltas=(1.8, 0.6, 0.3)*)

simulate the typing of texts by a user entered into an explained TextInput widget of a tour page.

**Parameters**

- **text\_input** (TextInput) – text input widget, either of type TextInput or FlowInput.
- **text\_to\_delay** (str) – text string to be inserted delayed by the seconds specified in deltas[0].
- **text\_to\_insert** (str) – text string to be inserted directly into the passed text input widget.
- **deltas** (Tuple[float, ...]) – delay deltas in seconds between each character to simulate text inputted by a user. first delta default is a bit higher to finish navigation button y-pos-animation.

**tap\_animation**(*wid\_id="", pos\_delay=2.34, press\_delay=0.69, release\_delay=0.39*)

create a compound animation instance simulating a user touch/tap on the specified widget.

**Parameters**

- **wid\_id** (str) – specifies the widget to be tap simulated: either a widget id string (first item of the PageAnimationType tuple), or (if prefixed with a column character) tap/focus/ state id of a widget, or an empty string (specifies the currently explained widget).
- **pos\_delay** (float) – time in seconds to position/move the pointer from the next button to the widget.
- **press\_delay** (float) – time in seconds of the button press simulation animation.
- **release\_delay** (float) – time in seconds of the button release simulation animation.

**Return type**

Tuple[str, Union[Animation, str]]

**Returns**

compound animation instance simulating a tap.

---

**Note:** use as animation evaluation expression, to get the widget values on setup-time of the page (not tour).

---

**teardown\_shaders\_and\_animations()**

teardown all added shaders and animations of current tour page (including switch next page animations).

**teardown\_app\_flow()**

overridden to teardown the animations of the current/last-shown tour page.

**class AnimatedOnboardingTour**(*main\_app*)

Bases: *AnimatedTourMixin, OnboardingTour*

onboarding tour, extended with animations and glsl shaders.

**\_\_init\_\_**(*main\_app*)

overridden to handle onboarding tour starts since app installation.

**next\_page**()

overriding to remove next button size animation only visible in the first tour after app re/start.

**setup\_layout**()

overridden to update layout texts if app window/screen orientation (`app.landscape`) changes.

**teardown\_shaders\_and\_animations**()

overridden to unbind `setup_texts()` on leaving the responsible `_layout` tour page.

**layout**: `Widget`

tour overlay layout instance

**main\_app**: `Any`

shortcut to main app instance

**page\_ids**: `List[str]`

list of tour page ids, either initialized via this class attribute or dynamically.

**page\_idx**: `int`

index of the current tour page (in `page_ids`)

**\_added\_animations**: `List[Tuple[Widget, Animation, WidgetValues]]`

**\_added\_shaders**: `List[Tuple[Widget, ShaderIdType]]`

**pages\_animations**: `Dict[Optional[str], PageAnimationsType]`

dict of compound animation instances of the pages of this tour.

the key of this dict is the page id or None (for animations available in all pages of this tour). each value of this dict is of the type `PageAnimationsType`.

**pages\_shaders**: `Dict[Optional[str], Tuple[Tuple[str, ShaderIdType], ...]]`

dict of widget shaders for the pages of this tour.

the key of this dict is the page id or None (for shaders available in all pages of this tour). each value of this dict is a tuple of tuples of widget id and `add_shader()`-kwargs.

the widget id string specifies the widget to which a shader will be added, which is either:

- one of the widgets of the `TourOverlay` layout class, identified by the one of the following strings: `'next_but'`, `'page_lbl'`, `'tap_pointer'`, `'prev_but'`, `'title_lbl'`, `'tooltip'`, `'tour_page_texts'`.
- the explained widget if an empty string is given.
- the `TourOverlay` layout class instance for any other string (e.g. `'layout'` or `'overlay'`).

before the `add_shader()`-kwargs dict will be passed to the `add_shader()` method, all their non-string values, specifying as strings, will be evaluated/converted automatically. the evaluation provides the following globals: `layout`, `sp`, `Clock`, `Window` and the `tour` instance.

**switch\_next\_animations**: `Dict[Optional[str], PageAnimationsType]`

dict of compound animation instances for the next page switch transition of the pages of this tour.

the key of this dict is the page id or None (for animations available in all pages of this tour). each value of this dict is of the type `PageAnimationsType`.

**class TourOverlay**(*main\_app*, *tour\_class=None*, *\*\*kwargs*)

Bases: [ModalBehavior](#), [ShadersMixin](#), [FloatLayout](#)

tour layout/view overlay singleton class to display an active/running modal app tour with optional glsl shaders.

**ani\_value**

animated float value between 0.0 and 1.0, used e.g. by [AnimatedTourMixin.pages\\_animations](#).

[ani\\_value](#) is a [NumericProperty](#) and is read-only.

**explained\_pos**

window position (absolute x, y window coordinates) of the targeted/explained/highlighted widget.

[explained\\_pos](#) is a [ListProperty](#) and is read-only.

**explained\_size**

widget size (width, height) of the targeted/explained/highlighted widget.

[explained\\_size](#) is a [ListProperty](#) and is read-only.

**fade\_out\_app**

fade out app screen factor: 0.0 prevents fade out of the areas around [TourPageTexts](#) and the explained widget.

1.0 results in maximum app screen fade out. configurable for individual tour page via [page\\_data\['fade\\_out\\_app'\]](#).

[fade\\_out\\_app](#) is a [NumericProperty](#) and defaults to 0.39.

**label\_height**

height in pixels of the page text labels and text lines.

[label\\_height](#) is a [NumericProperty](#) and is read-only.

**navigation\_disabled**

if this flag is True then the back/next buttons in the tour layout/overlay are disabled.

[navigation\\_disabled](#) is a [BooleanProperty](#) and is read-only.

**close:** [Callable](#)

method to dismiss the container widget (provided by self/container-widget)

**tour\_instance**

holding the [TourBase](#) instance of the current tour, initialized by [start\\_tour\(\)](#).

[tour\\_instance](#) is a [ObjectProperty](#) and is read-only.

**\_\_init\_\_**(*main\_app*, *tour\_class=None*, *\*\*kwargs*)

prepare app and tour overlay (singleton instance of this class) to start tour.

**Parameters**

- **main\_app**¶ ([HelpAppBase](#)) – main app instance.
- **tour\_class**¶ ([Optional](#)[[Type](#)[[TourBase](#)]]) – optional tour (pages) class, default: tour class of current help id or [OnboardingTour](#).

**explained\_widget**

explained widget instance on actual tour (page).

[explained\\_widget](#) is a [ObjectProperty](#) and is read-only.

**next\_page()**

switch to next tour page.

**on\_navigation\_disabled(\*\_args)**

navigation button disabled change event, used to hide page texts (blend-in-anim in `page_updated()`).

**page\_updated()**

callback from `setup_layout()` for UI-specific patches, after tour layout/overlay setup.

**prev\_page()**

switch to previous tour page.

**start\_tour(tour\_cls=None)**

reset app state and prepare tour to start.

**Parameters**

**tour\_cls** *(Optional[Type[TourBase]])* – optional tour (pages) class, default: tour of currently shown help id or OnboardingTour.

**Return type**

`bool`

**Returns**

True if tour exists and got started.

**stop\_tour()**

stop tour and restore the initially backed-up app state.

## 4.38 ae.kivy\_auto\_width

### 4.38.1 automatic width mix-in classes for kivy widgets

this ae portion is providing classes to mix them into any kivy widget with a *texture* property (like e.g. `Label` or `Button`), to automatically size and resize widgets and/or to display a long text as scrolling ticker within a tall widget.

#### automatic font size iteration with animation

mix-in the `AutoFontSizeBehavior` into any kivy widget with a *texture* property, to automatically grow or shrink the font size to fully fill the width/height of the widget with the size of their texture.

more details see in the documentation of the `AutoFontSizeBehavior` class.

#### automatic container width with opening animation

the class `ContainerChildrenAutoWidthBehavior` determines the optimal width of a container widget, so that the text of any children widget is fully visible/displayed.

the optimal container width is determined by increasing width of the container in iterations, which are implemented through a kivy `Animation`. as soon as the texts of all children are fully displayed (or the maximum width is reached) the animation stops.

more details see in the documentation of the `ContainerChildrenAutoWidthBehavior` class.

## automatic ticker animation

mix-in the *SimpleAutoTickerBehavior* class to automatic slide the texture of a widget if it is too big to be completely/fully displayed, like in a news-ping-pong-ticker.

for more details check the documentation of the *SimpleAutoTickerBehavior* class.

## Classes

<i>AutoFontSizeBehavior</i> (**kwargs)	mix-in to interpolate the optimal font size so that the texture is filling the full width of the widget.
<i>ContainerChildrenAutoWidthBehavior</i> ()	detect minimum width for the complete display of the textures of all children at opening with animation.
<i>SimpleAutoTickerBehavior</i> (**kwargs)	mix-in class to slide texture in a ping-pong-like-ticker animation, if too long to be displayed completely.

**class** *AutoFontSizeBehavior*(\*\*kwargs)

Bases: *object*

mix-in to interpolate the optimal font size so that the texture is filling the full width of the widget.

the desired spacing (left plus right) between the texture border and the widget borders can be set via the *auto\_font\_text\_spacing* property. additional padding can be added via the *padding\_x* property of the mixing-in widget.

to disable the animation set the length of the *auto\_font\_anim\_duration* to zero or very short value.

the minimum and maximum of the texture font size can be restricted by setting the attributes *auto\_font\_min\_size* and *auto\_font\_max\_size*.

**bind:** *Callable*

**font\_size:** *float*

**texture\_size:** *tuple*

**texture\_update:** *Callable*

**width:** *float*

**height:** *float*

**auto\_font\_anim\_duration:** *float*

duration in seconds of the font size grow/shrink animation.

*auto\_font\_anim\_duration* is a *NumericProperty* and defaults to 0.9 seconds.

**auto\_font\_text\_spacing:** *float*

horizontal padding in pixels between widget and texture width (including the additional horizontal *padding\_x*).

*auto\_font\_text\_spacing* is a *NumericProperty* and defaults to 18sp.

**\_font\_size\_anim:** *Optional[Animation]* = None

**\_font\_anim\_mode:** *int* = 0

`_last_font_size: float = 0.0`

`__init__(**kwargs)`

`auto_font_max_size: float`

maximum font size.

`auto_font_max_size` is a `NumericProperty` and defaults to `min(MAX_FONT_SIZE, max_font_size)`.

`auto_font_min_size: float`

minimum font size.

`auto_font_min_size` is a `NumericProperty` and defaults to `max(MIN_FONT_SIZE, min_font_size)`.

`_font_size_adjustable()`

check if font size need/has to be adjustable.

`_start_font_anim(*_args)`

delayed anim check

`_stop_font_anim()`

`_font_size_progress(_anim, _self, _progress)`

animation on \_progress event handler.

**class ContainerChildrenAutoWidthBehavior**

Bases: `object`

detect minimum width for the complete display of the textures of all children at opening with animation.

this mix-in class can be added to any type of container or layout widget to provide a consistent API with `open()` and `close()` methods, a `on_complete_opened()` event and a `container` attribute.

---

**Note:** a `container` attribute will be automatically created for container classes without it.

---

the animation starts when the `open()` method get called. this call will be forwarded via `super()` to the container if it has an `open` method.

at animation start the width of this container will be set to value of the `auto_width_start` attribute. then the container width increases via the running animation until, either:

- the container width is greater than the value of the `auto_width_minimum` attribute and the textures of all children are fully visible or
- the container width reaches the app window width minus the window padding specified in the `auto_width_window_padding` attribute.

the window width gets bound to the container width to ensure proper displaying if the window width changes.

#### Events

**`on_complete_opened:`**

fired when the container width animation is finished or stopped because all children are fully visible.

**`container: Widget`**

widget to add the dynamic children to (provided by the widget to be mixed into)

**`dismiss: Callable`**

optional method provided by the widget to be mixed into

**dispatch:** `Callable`

event dispatch method, provided by the widget to be mixed into

**opacity:** `float`

opacity of the widget to be mixed into

**parent:** `Widget`

parent of this widget/container.

**width:** `float`

width of the widget to be mixed into (mostly a parent of `self.container`)

**auto\_width\_anim\_duration:** `float`

duration in seconds of the auto-width-animation.

`auto_width_anim_duration` is a `NumericProperty` and defaults to 0.9 seconds.

**auto\_width\_window\_padding:** `float`

horizontal padding in pixels between the window and the container.

`auto_width_window_padding` is a `NumericProperty` and defaults to 96sp.

**auto\_width\_minimum:** `float`

minimum container width in pixels (before the width animation will be stopped).

`auto_width_minimum` is a `NumericProperty` and defaults to 369sp.

**auto\_width\_child\_padding:** `float`

horizontal padding in pixels between child widget and child texture.

`auto_width_child_padding` is a `NumericProperty` and defaults to 87sp.

**auto\_width\_start:** `float`

container width in pixels at the start of the width animation.

`auto_width_start` is a `NumericProperty` and defaults to 3sp.

**\_width\_anim:** `Animation` = `None`

**\_complete\_width:** `float` = `0.0`

**\_\_events\_\_** = ('on\_complete\_opened',)

**close**(\*args, \*\*kwargs)

close/dismiss container/layout (ae.gui\_app popup handling compatibility for all GUI frameworks).

#### Parameters

- `_args` – unused argument (to have compatible signature for Drop-Down/Popup/ModalView widgets).
- `_kwargs` – unused argument (to have compatible signature for Drop-Down/Popup/ModalView widgets).

**on\_complete\_opened()**

dispatch event default handler, called on opening when the final width got determined.

**open**(\*args, \*\*kwargs)

open container, optionally starting auto-width-animation.

#### Parameters

- `_args` – unused argument (to have compatible signature for Popup/ModalView and DropDown widgets passing the parent widget).
- `kwargs` – optional extra arguments:
  - `'animation'`: *False* will disable the *width* and *open()*-animations (default=True).

**reset\_width\_detection()**

call to reset the last detected minimum container width (e.g. if the children text got changed).

**\_detect\_complete\_width()**

check clients textures until widest child texture got detected.

**Return type**

`float`

**Returns**

0.0 until complete width got detected, then the last detected minimum container width.

**\_on\_complete\_opened(\*\_args)**

open animation completion callback/event.

**\_on\_win\_width(\*\_args)**

Window.width event handler.

**\_open\_width\_progress(\_anim, \_self, \_progress)**

animation on\_progress event handler.

**\_win\_width\_bind()**

bind Window width property to container width.

**class SimpleAutoTickerBehavior(\*\*kwargs)**

Bases: `object`

mix-in class to slide texture in a ping-pong-like-ticker animation, if too long to be displayed completely.

if the *text* or *size* of the widget where this class get mixed in changes then this instance is first determining the number of characters that can be displayed completely in this widget. this is done with a kivy animation. the duration of this animation can be set via the property `auto_ticker_length_anim_duration`.

to adjust the padding space between the widget border and their texture width, the property `auto_ticker_text_spacing` can be set accordingly.

after determining the maximum number of characters that can be displayed and storing this value into the private attribute `_ticker_text_length` a second animation - the offset animation - gets started to slide/scroll the text. the speed of the offset animation can be set via the property `auto_ticker_offset_anim_speed`.

---

**Note:** while the ticker animations are running the *text* property of the widget is only containing the visible part of the full initial text string. use the private attribute `_ori_text` to determine the full text string.

---

**bind:** `Callable`

**get\_root\_window:** `Callable`

**text:** `str`

**texture\_size:** `tuple`

**texture\_update:** `Callable`



**unbind:** `Callable`

**width:** `float`

**auto\_ticker\_length\_anim\_duration:** `float`  
duration in seconds of the iteration animation to determine the maximum text length.  
*auto\_ticker\_length\_anim\_duration* is a `NumericProperty` and defaults to 0.9 seconds.

**auto\_ticker\_offset\_anim\_speed:** `float`  
speed of the ticker text offset animation in characters per second.  
*auto\_ticker\_length\_anim\_duration* is a `NumericProperty` and defaults to 9.6.

**auto\_ticker\_text\_spacing:** `float`  
horizontal padding between widget and texture width in pixels.  
*auto\_ticker\_text\_spacing* is a `NumericProperty` and defaults to 18sp.

**\_bound\_properties:** `Dict[str, Callable]`  
properties of the mixing in widget to bind

**\_length\_anim:** `Optional[Animation] = None`  
shorten length animation

**\_min\_text\_len:** `int = 6`  
minimal length of shortened text

**\_offset\_anim:** `Optional[Animation] = None`  
ticker text offset animation

**\_ori\_text:** `str = ''`  
original/full text string

**\_ticker\_text\_offset:** `int = 0`  
current animated offset in the ticker text

**\_ticker\_text\_length:** `int = 6`  
number of characters that are completely visible in widget

**\_ticker\_text Updating:** `bool = False`  
flag to block restart of ticker on internal update of *text* property

**\_\_init\_\_**(*\*\*kwargs*)

**\_bind\_properties**()

**\_start\_length\_anim**(*\*\_args*)

**\_start\_offset\_anim**(*offset\_on\_complete=0*)

**\_stop\_length\_anim**(*reset=True*)

**\_stop\_offset\_anim**(*reset=True*)

**\_text\_changed**(*\*\_args*)  
called on change of label text. `assert _args[1] == self.text`

**\_ticker\_length\_progress**(*\_anim, \_self, progress*)

latest :

---

```
_ticker_max_offset()

    Return type
    int

_ticker_offset_progress(_anim, _self, progress)

_ticker_text_update(text)

_unbind_properties()
```

## 4.39 ae.kivy\_file\_chooser

### 4.39.1 extended kivy file chooser widget

This ae namespace portion provides the *FileChooserPopup* widget (*chooser\_popup*) which is embedding Kivy's *FileChooser* class in a dropdown window (*FlowDropDown*), and extending it with a path selector and a button to switch between list and icon view.

#### file chooser dropdown usage

The *FileChooserPopup* widget can be used like any Kivy *DropDown* widget - see the python and kv lang examples in the doc strings of the *dropdown* module. Additionally all the features of the *FlowDropDown* like e.g. the *child\_data\_maps* are available.

Alternatively (and without the need to explicitly instantiate the file chooser dropdown widget) you simply have to change the application flow to *id\_of\_flow*('open', 'file\_chooser') to open this file chooser (see also *application flow*):

```
main_app.change_flow(id_of_flow('open', 'file_chooser'),
                    **update_tap_kwargs(open_button))
```

The variable *open\_button* in this example represents a button widget instance that opens the file chooser dropdown (and to which the file chooser gets attached to).

Use the *submit\_to* property to distinguish multiple usages of the file chooser in a single app:

```
main_app.change_flow(id_of_flow('open', 'file_chooser'),
                    **update_tap_kwargs(open_button,
                                       popup_kwargs=dict(submit_to=submit_to_str_or_
↪ callable)))
```

The variable *submit\_to\_str\_or\_callable* of the above example can be either a string or a callable. If you pass a callable, *FileChooserPopup* will call it if the user has selected a file (by touching or double clicking on a file entry). This callback receives two arguments: the file path of just selected file and the *FileChooser* dropdown widget instance and can be declared like:

```
def submit_to_callable(file_path: str, chooser_popup: Widget):
```

Passing a string to *submit\_to* (or if it get not specified at all) the hard-coded *on\_file\_chooser\_submit* event handler callback method of your main app instance will be executed with the same two arguments:

```
def on_file_chooser_submit(self, file_path: str, chooser_popup: Widget):
    if chooser_popup.submit_to == 'usage1':
        usage1_object_or_process.file_path = file_path
        chooser_popup.dismiss()
    elif chooser_popup.submit_to == 'usage2':
        ...
    elif chooser_popup.submit_to == '':      # w/o specifying `submit_to`
        ...
```

Use the key of the `tap_flow_id` property of the `FlowButton` to provide a separate `help_text` for each individual button.

The `filters` property of Kivy's `kivy.uix.filechooser.FileChooser` can be used to filter the files displayed in this file chooser widget.

The path selector dropdown (`FileChooserPathSelectPopup`) situated at the top of this file chooser dropdown is providing all common OS and app specific paths that are registered in the `PATH_PLACEHOLDERS` dict. The keys of this dict will be displayed as shortcut path names instead of the full path strings. Additionally translation texts can be provided for the shortcut path names to display them in the language selected by the app user.

To extend the path selector dropdown with additional paths you can either register them within `PATH_PLACEHOLDERS`, or you add them to the optional app state variable `file_chooser_paths` by calling the method `register_file_path()`.

By adding the list `file_chooser_paths` to the *app state variables* of your app, the paths provided by the path selector widget will automatically maintain and keep the OS and user paths persistent between app runs.

to record and remember the last selected path add also the app state `file_chooser_initial_path` to the *:ref: `app state variables`* of your app.

Override the method `_init_default_user_cfg_vars()` within the main app instance of your app to make these two persistent *app state variables* user-specific:

```
def _init_default_user_cfg_vars(self):
    super()._init_default_user_cfg_vars()
    self.user_specific_cfg_vars |= {
        (APP_STATE_SECTION_NAME, 'file_chooser_initial_path'),
        (APP_STATE_SECTION_NAME, 'file_chooser_paths'),
    }
```

**Hint:** you don't need to override `_init_default_user_cfg_vars()` if your app is embedding the `ae` portion `ae.kivy_sideload`.

## Classes

<code>FileChooserPathSelectPopup(**kwargs)</code>	file chooser path selector dropdown.
<code>FileChooserPopup(**kwargs)</code>	file chooser drop down container.

**class** `FileChooserPopup(**kwargs)`

Bases: `FlowDropDown`

file chooser drop down container.

**initial\_path**

initial file path displayed on opening

**submit\_to**

callable or string to identify which action/part requested the selected file

**filters**

see `kivy.uix.filechooser.FileChooser.filters`.

**static on\_file\_chooser\_entry\_added**(*view\_entries*)

on\_entry\_added/on\_subentry\_to\_entry event handler to patch theme-related properties of Kivy File-Chooser.

**Parameters**

**view\_entries** *(List[Widget])* – list of view entries for a node (icon or label) of the file chooser.

---

**Note:** This method get called for each node in the moment when a file entry widget (`FileListEntry` or `FileIconEntry`) gets added to an instance of Kivy's `FileChooser` widget class.

Therefore the patches done here are not affected if the user preferences (e.g. the font size or light/dark theme) get changed while a file chooser instance is displayed. In this case the user has to simply close and reopen/re-instantiate the file chooser to display the nodes with the just changed user preferences.

Theme adaption is still missing for the file chooser progress: all font sizes and colors of the currently used `FileChooserProgressBase` are hard-coded, so a theme-aware progress class has to be implemented (and assigned to the `progress_cls` property).

---

**static register\_file\_path**(*file\_path*, *main\_app*)

set folder of the passed file path as new initial path and add it to path history.

**Parameters**

- **file\_path** *(str)* – file path (mostly the last just selected file) of which the folder will be registered.
- **main\_app** *(Any)* – main app instance.

**\_container:** `Widget`

**\_layout\_finished:** `bool`

**\_opened\_item:** `Optional[Widget]`

**\_touch\_moved\_outside:** `bool`

**class FileChooserPathSelectPopup**(*\*\*kwargs*)

Bases: `FlowDropDown`

file chooser path selector dropdown.

**paths**

list of file paths in the path selection dropdown

**\_container:** `Widget`

**\_layout\_finished:** `bool`

```

_opened_item: Optional[Widget]

_touch_moved_outside: bool

```

## 4.40 ae.kivy\_iterable\_displayer

### 4.40.1 iterable displayer widget

The popup widget provided by this ae namespace portion displays items and sub-items of any type of iterables, like dicts, lists, sets and tuples.

#### iterable displayer usage

To open a popup displaying the keys/indexes and values of an iterable simple instantiate *IterableDisplayerPopup*. You can specify a popup window title string via the *title* kwarg and pass the iterable to the *data* kwarg (or property):

```
dict_displayer = IterableDisplayerPopup(title="popup window title", data=iterable_data)
```

A widget will be automatically instantiated for each sub-item of *iterable\_data* to display the item key and value. The used widget class is depending on the type of the sub-item. For non-iterable sub-items the *IterableDisplayerLabel* widget will be used. If instead a sub-item is containing another iterable then *IterableDisplayerPopup* will use the *IterableDisplayerButton* class, which when tapped displays another instance of *IterableDisplayerPopup* with the sub-sub-items.

**Note:** The string in the *title* property may be shortened automatically by *FlowPopup*, depending on the width of the popup layout and the *font\_size* app state.

#### Classes

<i>IterableDisplayerPopup</i> (**kwargs)	FlowPopup displaying iterable data - useful for quick prototyping and debugging.
--	--

```
class IterableDisplayerPopup(**kwargs)
```

Bases: *FlowPopup*

FlowPopup displaying iterable data - useful for quick prototyping and debugging.

**data**

the iterable (dict, list, set, tuple) from which the items will be shown

**static compile\_data\_maps**(data)

re-create data maps if the *data* attribute changes.

**Parameters**

**data** *(Union[dict, list, set, tuple])* – dict/list/set/tuple data to display (==self.data binding).

**Returns**

list of dicts to be assigned to self.child\_data\_maps.

latest :

---

```
_container: Widget
attach_to: Optional[Widget]
_layout_finished: bool
_opened_item: Optional[Widget]
_touch_moved_outside: bool
```

## 4.41 ae.kivy\_qr\_displayer

### 4.41.1 qr code displayer widget

the popup widget *QrDisplayerPopup* provided by this ae namespace portion is displaying QR codes.

the *QrDisplayerPopup* is inherited from *ae.kivy.widgets.FlowPopup* and is embedding the Kivy Garden *kivy\_garden.qrcode* module.

#### qr displayer popup usage

to display a QR code instantiate *QrDisplayerPopup* specifying in the *title* property of the popup the string to encode to a QR image and in the *qr\_content* property a short string describing the content of the string to encode. after that call the *open* method:

```
qr_displayer = QrDisplayerPopup(title="string to encode", qr_content="what to encode")
qr_displayer.open()
```

alternatively you can simply change the application flow to *id\_of\_flow('open', 'qr\_displayer')* (see also *application flow*):

```
main_app.change_flow(id_of_flow('open', 'qr_displayer'),
                    popup_kwargs=dict(title="string to encode",
                                       qr_content="what to encode"))
```

the label texts used by this popup widget are automatically translated into the german and spanish language via the translation texts provided in the resources of this ae namespace portion.

---

**Note:** if your app is providing i18n translations then the *qr\_content* string has to be translated (e.g. by using *get\_txt()* or *get\_text()*) before it gets passed to the popup kwargs.

---

to support additional languages simply add the translations texts to your app's translation texts resources or submit a PR to add them to this ae namespace portion. alternatively you could put different wording by specifying also the english translation text.

---

**Hint:** apart from *root.qr\_content* you can also use *root.title* in the translation texts to repeat/mention the string to encode in the text content.

---

## Classes

<code>QrDisplayerPopup(**kwargs)</code>	qr code displayer.
---	--------------------

```
class QrDisplayerPopup(**kwargs)
    Bases: FlowPopup
    qr code displayer.
    qr_content
        string to name the content that get displayed as QR code
    _container: Widget
    attach_to: Optional[Widget]
    _layout_finished: bool
    _opened_item: Optional[Widget]
    _touch_moved_outside: bool
```

## 4.42 ae.kivy\_sideloadimg

### 4.42.1 kivy mixin and widgets to integrate a sideloading server in your app

this namespace portion provides widgets and a mixin class for you main app instance to easily integrate and control the *ae sideloading server* <ae.sideloadimg\_server> into your *main app*.

#### kivy sideloading integration into your main app class

add the *SideloadimgMainAppMixin* mixin provided by this ae namespace portion to your main app class:

```
class MyMainAppClass(SideloadimgMainAppMixin, KivyMainApp):
```

the sub app of the sideloading server will then automatically be instantiated when your app starts and will initialize the *sideloading\_app* attribute with this sub app instance.

---

**Hint:** if you prefer to instantiate the sideloading server sub app manually then specify *SideloadimgMainAppMixin* after *KivyMainApp* in the declaration of your main app class.

---

adding *sideloading\_active* to the *:ref: `app state variables`* of your app's *config files* will ensure that the running status of the sideloading server gets automatically stored persistent on pause or stop of the app for the next app start.

the running status of the sideloading server will be restored in the app start event handler method (*on\_app\_run()*).

to manually start it offering the APK of the embedding app call the *on\_sideloadimg\_server\_start()* method passing an empty string and dict:

```
self.on_sideloadimg_server_start("", {})
```

latest :

---

---

**Hint:** when you pass the dict with a number in a 'port' key then this number will be used as server listening port.

---

if no 'port' gets specified then *SideloadMainAppMixin* will calculate an individual port number from the first character of the *app\_name* of the app mixing in this class. this is to prevent the server socket error *[Errno 98] Address already in use* if two different applications with sideloading are running on the same device and want to offer sideloading.

to manually pause the sideloading server call the *on\_sideloading\_server\_stop()* method passing an empty string and dict:

```
self.on_sideloading_server_stop("", {})
```

### usage of the sideloading button

this ae namespace portion is additionally providing the *SideloadButton* flow button widget to integrate it in your Kivy app. This button can be used to:

- start or stop the sideloading server,
- select a file for sideloading via the *FileChooserPopup*.
- display file info like full file path and file length.
- display the URL of your sideloading server as QR code to allow connections from other devices.

to optionally integrate this *SideloadButton* into your app add it to the root layout in your app's main kv file with the *id sideloading\_button*:

```
MyRootLayout:
    ...
    SideloadButton:
        id: sideloading_button
```

if the sideloading server is not active and the user is clicking the *SideloadButton* then this portion will first check if the *Downloads* folder of the device is containing an APK file for the running app and if yes then the sideloading server will be started providing the found APK file.

if the sideloading server is instead already running/active and the user is tapping on the *SideloadButton* then a dropdown menu will be shown with options to (1) display info of the sideloading file, (2) select a new file, (3) display the sideloading server URL as QR code or (4) stop the sideloading server.

### dependencies/requirements in *buildozer.spec*

to build an Android APK with the kivy sideloading server integrated, make sure that the following external packages are specified in the *requirements* setting of the *[app]* section of your *buildozer.spec* file.

- ae.kivy\_file\_chooser
- ae.kivy\_iterable\_displayer
- ae.kivy\_qr\_displayer
- ae.kivy\_sideloading
- ae.sideloading\_server
- kivy\_garden.qrcode



- `qrcode`

additionally, the following packages and `ae` namespace portions required by the above packages have to be included:

```
qrcode, kivy_garden.qrcode,
ae.base, ae.files, ae.paths, ae.deep, ae.dynamicod, ae.i18n,
ae.updater, ae.core, ae.literal, ae.console, ae.parse_date, ae.gui_app,
ae.gui_help, ae.kivy_auto_width, ae.kivy_dyn_chi,
ae.kivy_relief_canvas, ae.kivy, ae.kivy_user_prefs, ae.kivy_gls1,
ae.kivy_file_chooser, ae.sideload_server, ae.kivy_sideload,
ae.kivy_iterable_displayer, ae.kivy_qr_displayer
```

## sideloading server life cycle

to activate the sideloading server to offer a different file, specify the path (or glob file mask) of the file to be offered/available via sideloading in the `sideloading_file_mask` attribute and then call the method `on_sideload_server_start()`. this method will check if the specified file exists and if yes then it will start the sideloading server. if you specify a file mask instead of a concrete file path then this method will check if exists exactly one file matching the file mask.

after the start of the sideloading server the `sideloading_file_ext` attribute will contain the file extension of the file available via sideloading.

the sideloading server will automatically be shut down on quit/close of the embedding app. you can alternatively stop the sideloading server manually at any time by calling the `on_sideload_server_stop()` method.

## Classes

<code>SideloadMainAppMixin()</code>	mixin class with default methods for the main app class.
<code>SideloadMenuPopup(**kwargs)</code>	dropdown menu to control sideloading server.
<code>SideloadMenuTour(main_app)</code>	user preferences menu tour.

**class** `SideloadMenuPopup(**kwargs)`

Bases: `FlowDropDown`

dropdown menu to control sideloading server.

`__init__`(\*\*kwargs)

`_container`: `Widget`

`_layout_finished`: `bool`

`_opened_item`: `Optional[Widget]`

`_touch_moved_outside`: `bool`

**class** `SideloadMenuTour(main_app)`

Bases: `TourDropdownFromButton`

user preferences menu tour.

`__init__`(main\_app)

**page\_ids:** List[str]

list of tour page ids, either initialized via this class attribute or dynamically.

**\_saved\_app\_states:** Dict[str, Any]

**auto\_switch\_pages:** Union[bool, int]

enable/disable automatic switch of tour pages.

set to *True*, *1* or *-1* to automatically switch tour pages; *True* and *1* will switch to the next page until the last page is reached, while *-1* will switch back to the previous pages until the first page is reached; *-1* and *1* automatically toggles at the first/last page the to other value (endless ping-pong until back/next button gets pressed by the user).

the seconds to display each page before switching to the next one can be specified via the item value of the the dict *page\_data* dict with the key '*next\_page\_delay*'.

**page\_data:** Dict[str, Any]

additional/optional help variables (in *help\_vars* key), tour and page text/layout/timing settings.

the class attribute values are default values for all tour pages and get individually overwritten for each tour page by the i18n translations attributes on tour page change via *load\_page\_data()*.

supported/implemented dict keys:

- *app\_flow\_delay*: time in seconds to wait until app flow change is completed (def=1.2, >0.9 for auto-width).
- *back\_text*: caption of tour previous page button (def=get\_text('back')).
- *fade\_out\_app*: set to 0.0 to prevent the fade out of the app screen (def=1.0).
- *help\_vars*: additional help variables, e.g. *help\_translation* providing context help translation dict/text.
- *next\_text*: caption of tour next page button (def=get\_text('next')).
- *next\_page\_delay*: time in seconds to read the current page before next request\_auto\_page\_switch() (def=9.6).
- *page\_update\_delay*: time in seconds to wait until tour layout/overlay is completed (def=0.9).
- *tip\_text* or "" (empty string): tour page tooltip text fstring message text template. alternatively put as first character a '=' character followed by a tour page flow id to initialize the tip\_text to the help translation text of the related flow widget, and the *self* help variable to the related flow widget instance.
- *tour\_start\_delay*: seconds between tour.start() and on\_tour\_start main app event (def=TOUR\_START\_DELAY\_DEF).
- *tour\_exit\_delay*: seconds between tour.stop() and the on\_tour\_exit main app event (def=TOUR\_EXIT\_DELAY\_DEF).

**pages\_explained\_matchers:** Dict[str, Union[ExplainedMatcherType, Tuple[ExplainedMatcherType, ...]]]

matchers (specified as callable or id-string) to determine the explained widget(s) of each tour page.

each key of this dict is a tour page id (for which the explained widget(s) will be determined).

the value of each dict item is a matcher or a tuple of matchers. each matcher specifies a widget to be explained/targeted/highlighted. for matcher tuples the minimum rectangle enclosing all widgets get highlighted.

the types of matchers, to identify any visible widget, are:

- `find_widget()` matcher callable (scanning `framework_win.children`)
- evaluation expression resulting in `find_widget()` matcher callable
- widget id string, declared via kv lang, identifying widget in `framework_root.ids`
- page id string, compiled from widgets app state/flow/focus via `widget_page_id()` to identify widget

**page\_idx:** `int`  
index of the current tour page (in `page_ids`)

**last\_page\_idx:** `Optional[int]`  
last tour page index (*None* on tour start)

**class SideloadMainAppMixin**

Bases: `object`

mixin class with default methods for the main app class.

**app\_name:** `str`

**change\_app\_state:** `Callable`

**change\_flow:** `Callable`

**dpo:** `Callable`

**framework\_root:** `Widget`

**get\_opt:** `Callable`

**show\_message:** `Callable`

**user\_specific\_cfg\_vars:** `set`

**vpo:** `Callable`

**file\_chooser\_initial\_path:** `str = ''`  
used by `file_chooser` to select side-loaded file

**file\_chooser\_paths:** `List[str] = []`  
recently used paths as app state for file chooser

**sideloading\_active:** `tuple = ()`  
app state flag if sideloading server is running

**sideloading\_app:** `SideloadServerApp`  
http sideloading server console app

**sideloading\_file\_ext:** `str = '.'`  
extension of selected sideloading file

**sideloading\_file\_mask:** `str = ''`  
file mask of sideloading file

**\_init\_default\_user\_cfg\_vars()**

**on\_app\_run()**  
run app event.

**on\_app\_started()**

initialize and start shaders after kivy app, window and widget root got initialized.

**on\_debug\_level\_change**(*level\_name*, *\_event\_kwargs*)

debug level app state change flow change confirmation event handler.

**Parameters**

- **level\_name** (str) – the new debug level name to be set (passed as flow key).
- **\_event\_kwargs** (Dict[str, Any]) – unused event kwargs.

**Return type**

bool

**Returns**

True to confirm the debug level change.

**on\_file\_chooser\_submit**(*file\_path*, *chooser\_popup*)

event callback from FileChooserPopup.on\_submit() on selection of file.

**Parameters**

- **file\_path** (str) – path string of selected file.
- **chooser\_popup** (Widget) – file chooser popup/container widget.

**on\_sideloading\_server\_start**(*\_flow\_key*, *event\_kwargs*)

start the sideloading server.

**Parameters**

- **\_flow\_key** (str) – unused/empty flow key.
- **event\_kwargs** (Dict[str, Any]) – event kwargs: \* 'port': TCP/IP server listening port. \* 'tap\_widget': button instance that initiated the start of the server.

**Return type**

bool

**Returns**

always True to confirm change of flow id.

**on\_sideloading\_server\_stop**(*\_flow\_key*, *\_event\_kwargs*)

stop a running sideloading http server.

**Return type**

bool

**Parameters**

- **\_flow\_key** (str) – unused/empty flow key.
- **\_event\_kwargs** (Dict[str, Any]) – unused event kwargs.

:re change of flow id.

## 4.43 ae.kivy\_user\_prefs

### 4.43.1 user preferences widgets for your kivy app

This namespace portion is providing a set of widgets to allow the users of your app to change their personal app settings/preferences, like the theme, the font size, the language and the used colors.

To use it in your app import this module, which can be done either in one of the modules of your app via:

```
import ae.kivy_user_prefs
```

Alternatively and when you use the [Kivy framework](#) for your app, you can import it within your main KV file, like this:

```
#: import _any_dummy_name ae.kivy_user_prefs
```

**Note:** The i18n translation texts of the namespace portion get registered on importing. When you import this portion from the main KV file and your app is overwriting a translation text of this portion, then you have to make sure that the translation texts of your main app get registered after the import of this portion. For that reason [MainAppBase](#) is using the *on\_app\_build* event to load the application resources, which gets fired after Kivy has imported the main KV file.

The user preferences are implemented as a [FlowDropDown](#) via the widget *UserPreferencesPopup*.

To integrate it in your app you simply add the *UserPreferencesButton* widget to the main KV file of your app.

### user preferences debug mode

The user preferences are activating a debug mode when you click/touch the *UserPreferencesButton* button more than three times within 6 seconds.

This debug mode activation is implemented in the *on\_user\_preferences\_open()* event handler method declared in the *ae.kivy.apps* module. It can be disabled for your app by simply overriding this method with an empty method in your main app class.

## 4.44 ae.lisz\_app\_data

### 4.44.1 lisz demo app data handling

this module provides common constants, functions and methods for the showcase/demo application *Lisz*, which is demonstrating the usage of the GUI framework packages provided by the [ae namespace](#).

### usage demonstration of ae namespace portions

the usage of the following ae namespace portions get demonstrated by this application:

- [ae.base](#): basic constants and helper functions
- [ae.files](#): file collection, grouping and caching
- [ae.deep](#): deep data structure search and replace
- [ae.i18n](#): internationalization / localization helpers

- `ae.paths`: generic file path helpers
- `ae.dynamicod`: evaluation and execution helper functions
- `ae.updater`: application environment updater
- `ae.core`: application core constants, helper functions and base classes
- `ae.literal`: literal type detection and evaluation
- `ae.console`: console application environment
- `ae.parse_date`: parse date strings more flexible and less strict
- `ae.gui_app`: base class for python applications with a graphical user interface
- `ae.gui_help`: main app base class with context help for flow and app state changes

---

**Hint:** the Kivy variant of this demo app uses additionally the following `ae` namespace portions: `ae.kivy_auto_width`, `ae.kivy_dyn_chi`, `ae.kivy_relief_canvas`, `ae.kivy` and `ae.kivy_user_prefs`.

---

### features of the lisz demo app

- internationalization of texts, user messages, help texts, button/label texts (`ae.i18n`)
- easy mapping of files in complex folder structures (`ae.files`, `ae.paths`)
- providing help screens (`ae.gui_help`, `ae.kivy.widgets`)
- colors changeable by user (`ae.kivy_user_prefs`, `ae.enaml_app`)
- font and button sizes are changeable by user (`ae.gui_app`)
- dark and light theme switchable by user
- sound output support with sound volume configurable by user
- recursive item data tree manipulation: add, edit and delete item
- each item can be selected/check marked
- filtering of selected/checked and unselected/unchecked items
- an item represents either a sub-node (sub-list) or a leaf of the data tree
- item order changeable via drag & drop
- item can be moved to the parent or a sub-node
- easy navigation within the item tree (up/down navigation in tree and quick jump)

### lisz application data model

the lisz demo app is managing a recursive item tree - a list of lists - that can be used e.g. as to-do or shopping list.

to keep this demo app simple, the data managed by the lisz application is a minimalistic tree structure that gets stored as an *application status*, without the need of any database. the root node and with that the whole recursive data structure gets stored in the app state variable *root\_node*.

the root of the tree structure is a list of the type *LiszNode* containing list items of type *LiszItem*. a *LiszItem* element represents a dict of the type *Dict[str, Any]*.

each *LiszItem* element of the tree structure is either a leaf or a node. and each node is a sub-list with a recursive structure identical to the root node and of the type *LiszNode*.

the following graph is showing an example data tree:

the above example tree structure is containing the root node items *A* (which is a leaf) and *C* (which is a sub-node).

the node *C* consists of the items *CA* and *CB* where *CA* is a leaf and *CB* is a node.

the first item of the node *CB* is another sub-node with the leaf item *CBA*.

#### 4.44.2 GUI framework demo implementations

the plan is to integrate the following GUI frameworks on top of the abstract base class (implemented in the *ae.gui\_app* portion of the *ae* namespace):

- *Kivy* based on the *Kivy* framework: *kivy lisz demo app*
- *Enaml* based on the *enaml* framework: *enaml lisz demo app*
- *Beeware Toga* based on the *beeware* framework: *beeware toga lisz demo app*
- *Dabo* based on the *dabo* framework: *dabo lisz demo app*
- *pyglet*
- *pygobject*
- *AppJar*

the main app base mixin class *LiszDataMixin* provided by this module is used to manage the common data structures, functions and methods for the various demo applications variants based on *ae.gui\_app* and the related GUI framework implementation portions (like e.g. *ae.kivy.apps* and *ae.enaml\_app*) of the *ae* namespace.

### 4.44.3 gui framework implementation variants

#### kivy

the `kivy lisz` app is based on the `Kivy` framework, a `pypi` package documented [here](#).

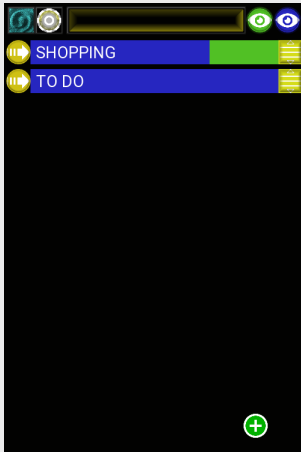


Fig. 4.1: kivy lisz app root list



Fig. 4.2: fruits sub-list

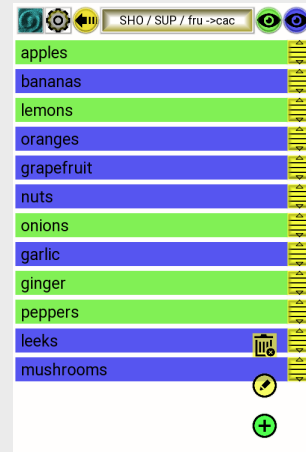


Fig. 4.3: using light theme



Fig. 4.4: lisz user preferences

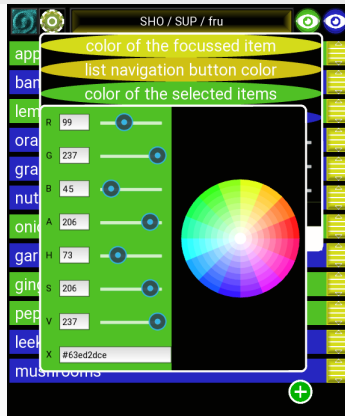


Fig. 4.5: kivy lisz color editor



Fig. 4.6: bigger font size

#### kivy wish list

- `kv` language loop pseudo widget (like `enaml` is providing) to easily generate sets of similar widgets.



## enaml

the `enaml lisz demo app` is based on the `enaml` framework, a pypi package documented here at [ReadTheDocs](#).

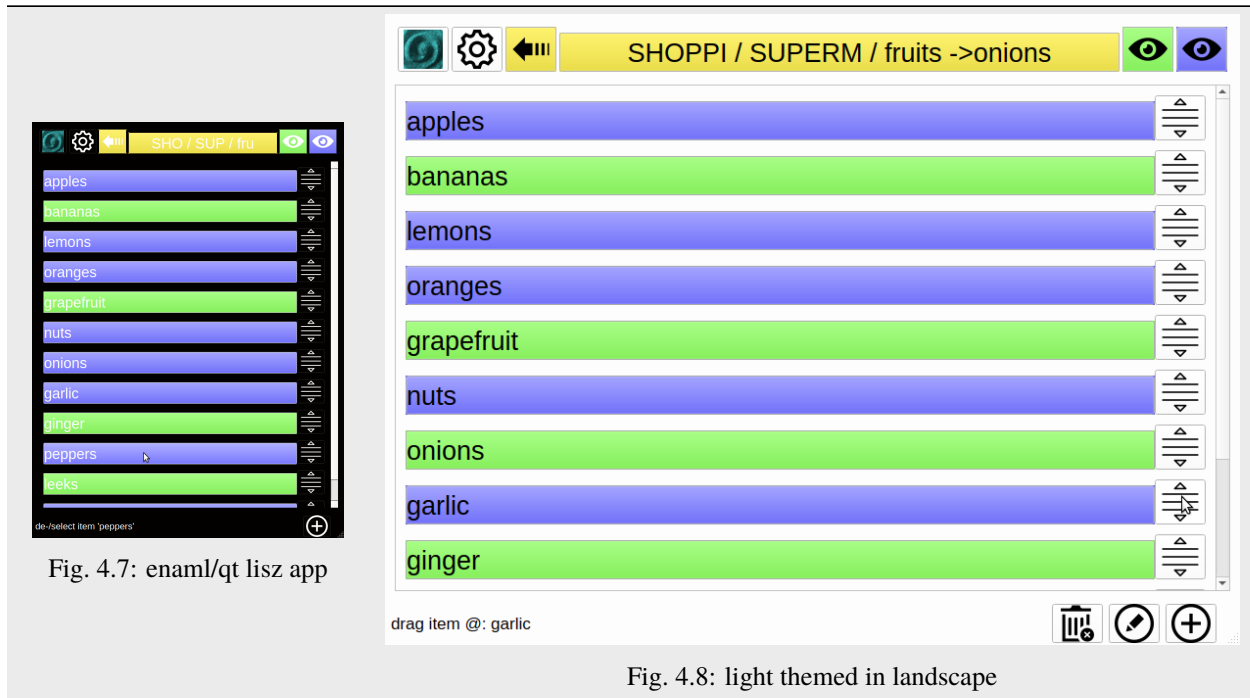


Fig. 4.7: enaml/qt lisz app

Fig. 4.8: light themed in landscape

## automatic update of widget attributes

dependencies have to be executed/read\_from, so e.g. the icon attribute will not be updated if `app.app_state_light_theme` gets changed:

```
icon << main_app.cached_icon('font_size') or app.app_state_light_theme
```

in contrary the icon will be updated by the following two statements:

```
icon << main_app.cached_icon('font_size') if app.app_state_light_theme else main_app.  
cached_icon('font_size')  
icon << app.app_state_light_theme == None or main_app.cached_icon('font_size')
```

KeyEvent implementation based on this SO answer posted by the enamlx author `frmdstryr/Jairus Martin`: <https://stackoverflow.com/questions/20380940/how-to-get-key-events-when-using-enaml>. alternative and more complete implementation can be found in the `enamlx` package (<https://github.com/frmdstryr/enamlx>).

latest :

---

## enaml wish list

- type and syntax checking, code highlighting and debugging of enaml files within PyCharm.
- fix freezing of linux/Ubuntu system in debugging of opening/opened PopuViews in PyCharm.

## Module Attributes

<i>FLOW_PATH_ROOT_ID</i>	pseudo item id needed for flow path jumper and for drop onto leave item button
<i>FLOW_PATH_TEXT_SEP</i>	flow path separator for <i>flow_path_text()</i>
<i>FOCUS_FLOW_PREFIX</i>	prefix shown in front of flow key of focused item
<i>INVALID_ITEM_ID_PREFIX_CHARS</i>	invalid initial chars in item id (to detect id   literal in flow key)
<i>NODE_FILE_PREFIX</i>	file name prefix for node imports/exports
<i>NODE_FILE_EXT</i>	file extension for node imports/exports
<i>IMPORT_NODE_MAX_FILE_LEN</i>	maximum file length of importable <file.NODE_FILE_EXT> file
<i>IMPORT_NODE_MAX_ITEMS</i>	maximum number of items to import or paste from clipboard
<i>LiszItem</i>	node item data (nid) type
<i>LiszNode</i>	node/list type
<i>NodeFileInfo</i>	tuple of (node-parent-name, node-list, file_path, error-message)
<i>NodeFilesInfo</i>	list of node file info tuples

## Functions

<i>check_item_id</i> (item_id)	check if the passed item id string is valid.
<i>correct_item_id</i> (item_id)	strip and replace extra/invalid characters from the passed item id string.
<i>flow_path_items_from_text</i> (text)	parse and interpret text block for (optional) flow path text and node data (in pprint.pformat/repr format).
<i>item_sel_filter</i> (item)	callable to filter selected LiszItems.
<i>item_unsel_filter</i> (item)	callable to filter unselected LiszItems.

## Classes

<i>LiszDataMixin</i> ()	lisz data model - independent from used GUI framework.
-------------------------	--

**FLOW\_PATH\_ROOT\_ID** = '^^^^'

pseudo item id needed for flow path jumper and for drop onto leave item button

**FLOW\_PATH\_TEXT\_SEP** = ' / '

flow path separator for *flow\_path\_text()*

**FOCUS\_FLOW\_PREFIX** = '->'

prefix shown in front of flow key of focused item

**INVALID\_ITEM\_ID\_PREFIX\_CHARS** = '[{'

invalid initial chars in item id (to detect id | literal in flow key)

**NODE\_FILE\_PREFIX** = 'node\_'

file name prefix for node imports/exports

**NODE\_FILE\_EXT** = '.txt'

file extension for node imports/exports

**IMPORT\_NODE\_MAX\_FILE\_LEN** = 8192

maximum file length of importable <file.NODE\_FILE\_EXT> file

**IMPORT\_NODE\_MAX\_ITEMS** = 12

maximum number of items to import or paste from clipboard

**LiszItem**

node item data (nid) type

alias of `Dict[str, Any]`

**LiszNode**

node/list type

alias of `List[Dict[str, Any]]`

**NodeFileInfo**

tuple of (node-parent-name, node-list, file\_path, error-message)

alias of `Tuple[str, List[Dict[str, Any]], str, str]`

**NodeFilesInfo**

list of node file info tuples

alias of `List[Tuple[str, List[Dict[str, Any]], str, str]]`

**check\_item\_id**(*item\_id*)

check if the passed item id string is valid.

**Parameters**

**item\_id** `(str)` – item id to check.

**Return type**

`str`

**Returns**

“” if all chars in the specified *item\_id* argument are valid, else one of the translated message strings.

**correct\_item\_id**(*item\_id*)

strip and replace extra/invalid characters from the passed item id string.

**Parameters**

**item\_id** `(str)` – item id string to correct.

**Return type**

`str`

**Returns**

corrected item id (can result in an empty string).

**flow\_path\_items\_from\_text**(*text*)

parse and interpret text block for (optional) flow path text and node data (in pprint.pformat/repr format).

**Parameters**

**text** *//* (*str*) – text block to be parsed. the text block can optionally be prefixed with an extra line (separated by a new line ‘n’ character) containing the destination flow path in text format (using FLOW\_PATH\_TEXT\_SEP to separate the flow path items).

the (rest of the) text block represents the node/item data in one of the following formats:

- single text line (interpreted as single leaf item).
- multiple text lines (interpreted as multiple leaf items).
- dict repr string, starting with ‘{’ character.
- list repr string, starting with ‘[’ character.

**Return type**

*Tuple*[*str*, *str*, *List*[*Dict*[*str*, *Any*]]]

**Returns**

tuple of error message (empty string if no error occurred), flow path (empty string if root or not given) and node list.

**item\_sel\_filter**(*item*)

callable to filter selected LiszItems.

**Parameters**

**item** *//* (*Dict*[*str*, *Any*]) – item data structure to check.

**Return type**

*bool*

**Returns**

True if item is a selected leaf or if item is a node with only selected sub-leaves, else False.

**item\_unsel\_filter**(*item*)

callable to filter unselected LiszItems.

**Parameters**

**item** *//* (*Dict*[*str*, *Any*]) – item data structure to check.

**Return type**

*bool*

**Returns**

True if item is an unselected leaf or if item is a node with only unselected leaves, else False.

**class LiszDataMixin**

Bases: *object*

lisz data model - independent from used GUI framework.

**root\_node:** *List*[*Dict*[*str*, *Any*]] = []

root of lisz data structure

**current\_node\_items:** *List*[*Dict*[*str*, *Any*]]

node item data of the current node / sub list (stored as app state via root\_node)

**filtered\_indexes:** *List*[*int*]

indexes of the filtered/displayed items in the current node

**filter\_selected:** `bool = False`  
 True to hide/filter selected node items  
**filter\_unselected:** `bool = False`  
 True to hide/filter unselected node items  
**debug\_level:** `int`  
 debug\_level  
**flow\_id:** `str`  
 current attr:flow id <ae.gui\_app.MainAppBase.flow\_id>  
**flow\_path:** `List[str]`  
*flow path* ref. current node  
**\_refreshing\_data:** `bool = False`  
 DEBUG True while running *refresh\_all()* method  
**call\_method:** `Callable`  
**call\_method\_delayed:** `Callable`  
**change\_app\_state:** `Callable`  
**change\_flow:** `Callable`  
**flow\_path\_action:** `Callable`  
**play\_sound:** `Callable`  
**refresh\_node\_widgets:** `Callable`  
**add\_item**(nid, node\_to\_add\_to=None, new\_item\_index=0)  
 add item (leaf or node) to currently displayed node.

#### Parameters

- **nid** *(Dict[str, Any])* – LszItem to add (has to have a non-empty item id).
- **node\_to\_add\_to** *(Optional[List[Dict[str, Any]])* – node where the passed item will be added to (def=current node).
- **new\_item\_index** *(int)* – index where the new item will be inserted (default=0, ignored if item already exists).

#### Return type

`str`

#### Returns

error message if any error happened, else empty string.

**add\_items**(items, node\_to\_add\_to=None)  
 add item to currently displayed node.

#### Parameters

- **items** *(List[Dict[str, Any]])* – LszNode list to add (each item has to have a non-empty item id).
- **node\_to\_add\_to** *(Optional[List[Dict[str, Any]])* – node where the passed item will be added to (def=current node).

**Return type**`str`**Returns**

error message if any error happened (multiple error messages are separated by \n), else empty string.

**change\_sub\_node\_sel**(*node, set\_sel\_to*)

change the selection of all the sub-leaves of the passed node to the specified value.

**Parameters**

- **node** (List[Dict[str, Any]]) – node of which to change the selection of the sub-item leaves.
- **set\_sel\_to** (bool) – True will only toggle the unselected sub-item leaves, False only the selected ones.

**current\_item\_or\_node\_literal**()

return the currently focused/displayed item or node as repr string.

**Return type**`str`**Returns**

pformat repr string of the currently focused item id/node or of the displayed node.

**delete\_items**(\**item\_ids, parent\_node=None, node\_only=False*)

delete either complete items or sub node of the items (identified by the passed item ids).

**Parameters**

- **item\_ids** (str) – tuple of item ids to identify the items/sub-nodes to be deleted.
- **parent\_node** (Optional[List[Dict[str, Any]]]) – node from where the item has to be removed from (default=current node).
- **node\_only** (bool) – True if only delete the sub-node of the identified item, False to delete the item.

**Return type**`List[Dict[str, Any]]`**edit\_validate**(*old\_item\_index, new\_id=None, want\_node=None, parent\_node=None, new\_item\_index=0*)

validate the user changes after adding a new item or editing an existing item.

**Parameters**

- **old\_item\_index** (int) – index in the current node of the edited item or -1 if a new item (to be added).
- **new\_id** (Optional[str]) – new/edited id string.
- **want\_node** (Optional[bool]) – True if the new/edited item will have a sub-node, False if not.
- **parent\_node** (Optional[List[Dict[str, Any]]]) – node where the edited/added item as to be updated/inserted (default=current list).
- **new\_item\_index** (int) – index where the new item have to be inserted (default=0, ignored in edit item mode).

**Return type**`str`

**Returns**

empty string on successful edit validation or on cancellation of new item (with empty id string), else error string or *'request\_delete\_confirmation\_for\_item'* if the user has to confirm the deletion after the user wiped the item id string or *'request\_delete\_confirmation\_for\_node'* if the user has to confirm the removal of the sub-node.

**export\_node**(*flow\_path*, *file\_path*='.', *node*=None)

export node specified by the passed *flow\_path* argument.

**Parameters**

- **flow\_path** (List[str]) – flow path of the node to export.
- **file\_path** (str) – folder to store the node data into (def=current working directory).
- **node** (Optional[List[Dict[str, Any]]]) – explicit/filtered node items (if not passed then all items will be exported).

**Return type**

str

**Returns**

empty string if node got exported without errors, else the error message/raised exception.

**find\_item\_index**(*item\_id*, *searched\_node*=None)

determine list index of the passed item id in the searched node or in the current node.

**Parameters**

- **item\_id** (str) – item id to find.
- **searched\_node** (Optional[List[Dict[str, Any]]]) – searched node. if not passed then the current node will be searched instead.

**Return type**

int

**Returns**

item list index in the searched node or -1 if item id was not found.

**flow\_key\_text**(*flow\_id*, *landscape*)

determine the shortest possible text fragment of the passed flow key that is unique in the current node.

used to display unique part of the key of the focused item/node.

**Parameters**

- **flow\_id** (str) – flow id to get key to check from (pass the observed value to update GUI automatically, either *self.app\_state\_flow\_id* or *self.app\_states['flow\_id']*).
- **landscape** (bool) – True if window has landscape shape (resulting in larger abbreviation). pass the observed attribute, mostly situated in the *framework\_win* (e.g. *self.framework\_win.landscape*).

**Return type**

str

**Returns**

display text containing flow key.

**flow\_path\_from\_text**(*text*, *skip\_check=False*)

restore the full complete flow path from the shortened flow keys generated by *flow\_path\_text()*.

**Parameters**

- **text** (str) – flow path text - like returned by *flow\_path\_text()*.
- **skip\_check** (bool) – pass True to skip the check if the flow path exists in the current self.root\_node.

**Return type**

List[str]

**Returns**

flow path list.

**flow\_path\_node**(*flow\_path=None*, *create=False*)

determine the node specified by the passed flow path, optionally create missing nodes of the flow path.

**Parameters**

- **flow\_path** (Optional[List[str]]) – flow path list.
- **create** (bool) – pass True to create missing nodes. only on False this method will return empty list on invalid/broken flow\_path.

**Return type**

List[Dict[str, Any]]

**Returns**

node list at flow\_path (if found -or- created is True and no-creation-errors) or empty list (if flow\_path not found and created is False or on creation error).

**flow\_path\_quick\_jump\_nodes**()

determine the current flow paths of all nodes excluding the current, to quick-jump from current node.

**Return type**

List[str]

**Returns**

list of flow path texts of all nodes apart from the current one.

**flow\_path\_text**(*flow\_path*, *min\_len=3*, *display\_root=False*, *separator=' / '*)

generate shortened display text from the passed flow path.

**Parameters**

- **flow\_path** (List[str]) – flow path list.
- **min\_len** (int) – minimum length of node ids (flow id keys). pass zero value to not shorten ids.
- **display\_root** (bool) – pass True to return FLOW\_PATH\_ROOT\_ID on empty/root path.
- **separator** (str) – path item separator (default=FLOW\_PATH\_TEXT\_SEP).

**Return type**

str

**Returns**

shortened display text string of the passed flow path (which can be converted back to a flow path list with the method *flow\_path\_from\_text()*).



**focus\_neighbour\_item**(*delta*)

move flow id to previous/next displayed/filtered node item.

**Parameters**

**delta** (int) – moving step (if greater 0 then forward, else backward).

**global\_variables**(\*\**patches*)

overridden to add app-specific globals.

**Return type**

Dict[str, Any]

**importable\_node\_files**(*folder\_path*='.')

load and check all nodes found in the app folder documents of this app.

**Parameters**

**folder\_path** (str) – path to the folder where the node files are situated (def=current working directory).

**Return type**

List[Tuple[str, List[Dict[str, Any]], str, str]]

**Returns**

list of node file tuples of (node\_name, node, file\_path, error-message).

**static import\_file\_info**(*file\_path*, *node\_name*='')

load node file and determine node content.

**Parameters**

- **file\_path** (str) – path to the node file to import.
- **node\_name** (str) – optional name/id of the parent node name. if not passed then it will be determined from the file name (removing `NODE_FILE_PREFIX` and file extension).

**Return type**

Tuple[str, List[Dict[str, Any]], str, str]

**Returns**

node file info tuple as: (node name, node-data or empty list, file path, error message).

**import\_items**(*node*, *parent*=None, *item\_index*=0)

import passed node items into the passed parent/destination node at the given index.

**Parameters**

- **node** (List[Dict[str, Any]]) – node with items to import/add.
- **parent** (Optional[List[Dict[str, Any]]]) – destination node to add the node items to (def=current node list).
- **item\_index** (int) – list index in the destination node where the items have to be inserted (default=0).

**Return type**

str

**Returns**

empty string if all items of node got imported correctly, else error message string.

**import\_node**(*node\_id*, *node*, *parent=None*, *item\_index=0*)

import passed node as new node into the passed parent node at the given index.

**Parameters**

- **node\_id** (str) – id of the new node to import/add.
- **node** (List[Dict[str, Any]]) – node with items to import/add.
- **parent** (Optional[List[Dict[str, Any]]]) – destination node to add the new node to (def=current node list).
- **item\_index** (int) – list index in the parent node where the items have to be inserted (default=0).

**Return type**

str

**Returns**

empty string if node got added/imported correctly, else error message string.

**item\_by\_id**(*item\_id*, *searched\_node=None*)

search item in either the passed or the current node.

**Parameters**

- **item\_id** (str) – item id to find.
- **searched\_node** (Optional[List[Dict[str, Any]]]) – searched node. if not passed then the current node will be searched instead.

**Return type**

Dict[str, Any]

**Returns**

found item or if not found a new dict with the single key=value 'id'=item\_id.

**move\_item**(*dragged\_node*, *dragged\_id*, *dropped\_path=None*, *dropped\_id=""*)

move item id from passed dragged\_node to the node and index specified by dropped\_path and dropped\_id.

**Parameters**

- **dragged\_node** (List[Dict[str, Any]]) – node where the item got dragged/moved from.
- **dragged\_id** (str) – id of the dragged/moved item.
- **dropped\_path** (Optional[List[str]]) – optional destination/drop node path, if not passed use dragged\_node.
- **dropped\_id** (str) – optional destination item where the dragged item will be moved before it. if empty string passed or not passed then the item will be placed at the end of the destination node.

**Return type**

bool

**node\_info**(*node*, *what=()*, *recursive=True*)

determine statistics info for the node specified by *flow\_path*.

**Parameters**

- **node** (List[Dict[str, Any]]) – node to get info for.

- **what** (Tuple[str, ...]) – pass tuple of statistic info fields to include only these into the returned dict (passing an empty tuple or nothing will include all the following fields):
  - 'count': number of items (nodes and leaves) in this node (including sub-nodes).
  - 'leaf\_count': number of sub-leaves.
  - 'node\_count': number of sub-nodes.
  - 'selected\_leaf\_count': number of selected sub-leaves.
  - 'unselected\_leaf\_count': number of unselected sub-leaves.
  - 'names': list of all sub-item/-node names/ids.
  - 'leaf\_names': list of all sub-leaf names.
  - 'selected\_leaf\_names': list of all selected sub-leaf names.
  - 'unselected\_leaf\_names': list of all unselected sub-leaf names.
- **recursive** (bool) – pass False if only the passed node has to be investigated.

**Return type**

Dict[str, Union[int, str, List[str]]]

**Returns**dict with the node info specified by the *what* argument.**on\_app\_state\_root\_node\_save**(root\_node)

shrink root\_node app state variable before it get saved to the config file.

**Return type**

List[Dict[str, Any]]

**on\_filter\_toggle**(toggle\_attr, \_event\_kwargs)

toggle filter on click of either the selected or the unselected filter button.

note that the inverted filter may be toggled to prevent both filters active.

**Parameters**

- **toggle\_attr** (str) – specifying the filter button to toggle, either 'filter\_selected' or 'filter\_unselected'.
- **\_event\_kwargs** (Dict[str, Any]) – unused.

**Return type**

bool

**Returns**

True to process flow id change.

**on\_item\_enter**(\_key, event\_kwargs)

entering sub node from current node.

**Parameters**

- **\_key** (str) – flow key (item id).
- **event\_kwargs** (dict) – event kwargs.

**Return type**

bool

**Returns**

True to process/change flow id.

**on\_item\_leave**(*\_key*, *event\_kwargs*)

leaving sub node, setting current node to parent node.

**Parameters**

- **\_key** (str) – flow key (item id).
- **event\_kwargs** (dict) – event kwargs.

**Return type**

bool

**Returns**

True to process/change flow id.

**on\_item\_sel\_toggle**(*item\_id*, *event\_kwargs*)

toggle selection of leaf item.

**Parameters**

- **item\_id** (str) – item id of the leaf to toggle selection for.
- **event\_kwargs** (dict) – event kwargs.

**Return type**

bool

**Returns**

True to process/change flow id.

**on\_item\_sel\_change**(*item\_id*, *event\_kwargs*)

toggle, set or reset in the current node the selection of a leaf item or of the sub-leaves of a node item.

**Parameters**

- **item\_id** (str) – item id of the leaf/node to toggle selection for.
- **event\_kwargs** (dict) – event kwargs, containing a *set\_sel\_to* key with a boolean value, where True will select and False deselect the item (or the sub-items if the item is a non-empty node).

**Return type**

bool

**Returns**

True to process/change flow id.

this flow change event can be used alternatively to [on\\_item\\_sel\\_toggle\(\)](#) for more sophisticated lisz app implementations, like e.g. the [kivy lisz demo app](#) .

**on\_item\_sel\_confirm**(*item\_id*, *event\_kwargs*)

confirming sub-list item de-/selection from ItemSelConfirmPopup

**Return type**

bool

**on\_key\_press**(*modifiers*, *key\_code*)

check key press event to be handled and processed as command/action.

**Parameters**

- **modifiers** (str) – modifier keys string.

- **key\_code** (str) – key code string.

**Return type**

bool

**Returns**

True if key event got processed/used, else False.

**on\_node\_extract**(flow\_path\_text, event\_kwargs)

extract the leaves of the node specified by flow\_path\_text.

**Parameters**

- **flow\_path\_text** (str) – flow path text or list literal (identifying the start node to extract from).
- **event\_kwargs** (dict) – extra arguments specifying extract options (only *extract\_type* is mandatory):

*extract\_type* specifies extract destination and an optional filter on un-/selected items. the first part defines the extract action (copy/cut/delete/export/share) and an optional second part (separated by an underscore) the filter. e.g. the following string values can be passed for a 'copy' extract action:

- 'copy' is copying all items of the specified node to the clipboard.
- 'copy\_sel' is copying only the selected items of the node to the clipboard.
- 'copy\_unsel' is copying only the unselected items to the clipboard.

*recursive* specifies if *False* that the extraction affects only the leaves of the current node specified by flow\_path\_text and if *True* the extraction affects also the leaves of the sub-nodes (default=True).

*export\_path* specifies the destination folder of the export action (default='.'/CWD).

**Return type**

bool

**Returns**

True to process/change flow.

**on\_node\_jump**(flow\_path\_text, event\_kwargs)

FlowButton clicked event handler restoring flow path from the flow key.

**Parameters**

- **flow\_path\_text** (str) – flow path text (identifying where to jump to).
- **event\_kwargs** (Dict[str, Any]) – event arguments (used to reset flow id).

**Return type**

bool

**Returns**

always True to process/change flow.

**refresh\_all()**

changed flow event handler refreshing currently displayed items after changing current node/flow path.

**refresh\_current\_node\_items\_from\_flow\_path()**

refresh current node including the depending on display node.

**shrink\_node\_size**(*node*, *item\_filter*=None, *recursive*=True)

shrink node size by removing unneeded items and *sel* keys, e.g. to export or save space in config file.

#### Parameters

- **node** (List[Dict[str, Any]]) – start or root node to shrink (in-place!).
- **item\_filter** (Optional[Callable[[Dict[str, Any]], bool]]) – pass callable to remove items from the passed node and its sub-nodes. the callable is getting each item as argument and has to return True to remove it from its node.
- **recursive** (bool) – pass False if only the passed start node has to be shrunk.

**sub\_item\_ids**(*node*=None, *item\_ids*=(), *leaves\_only*=True, *hide\_sel\_val*=None, *recursive*=True, *sub\_ids*=None)

return item names/ids of the specified items including their sub-node items (if exists and recursive==True).

used to determine the affected item ids if user want to delete or de-/select the sub-items of the item(s) specified by the passed arguments.

#### Parameters

- **node** (Optional[List[Dict[str, Any]]]) – searched node, if not passed use the current node as default.
- **item\_ids** (Tuple[str, ...]) – optional item id filter, if passed only items with an id in this tuple will be returned. this filter will not be used for sub-node filtering (if recursive==True).
- **leaves\_only** (bool) – pass False to also include/return node item ids.
- **hide\_sel\_val** (Optional[bool]) – pass False/True to exclude un-/selected leaf items from the returned list of ids. if None or not passed then all found items will be included.
- **recursive** (bool) – pass False if only the passed start node has to be investigated/included.
- **sub\_ids** (Optional[List[str]]) – already found sub item ids (used for the recursive calls of this method).

#### Return type

List[str]

#### Returns

list of found item ids.

**toggle\_item\_sel**(*node\_idx*)

toggle the item selection of the item identified by the list index in the current node.

#### Parameters

- **node\_idx** (int) – list index of the item in the current node to change the selection for.

## 4.45 ae.enaml\_app

### 4.45.1 enaml application widgets, helper functions and classes

the enaml module *widgets* is providing widgets to write themed applications which can switch their font colors and backgrounds at run-time between dark and light.

another set of widgets provided by this namespace portion allows the automatic change of the application flow with only few lines of code.

to convert colors between the enaml and other formats the functions declared in the *functions* of this package can be used.

#### main application class for GUIApp-conform Enaml app

the classes *FrameworkApp* and *EnamlMainApp* of this ae portion are bundling and adding useful attributes and methods for your application and are extendable by creating a subclass.

the class *EnamlMainApp* is implementing a main app class that is reducing the amount of code needed to create a Python application based on the *enaml framework*.

*EnamlMainApp* is based on the following classes:

- the abstract base class *MainAppBase* which is providing *application status* (including *app state variables* and *app state constants*), *application flow* and *application events*.
- the class *ConsoleApp* is adding *config files*, *config variables* and *config options*.
- the class *AppBase* is adding *application logging* and *application debugging*.

the main app class *EnamlMainApp* is also encapsulating the enaml app class for the Qt widget set (<enaml.QtApplication>) within the *FrameworkApp* class.

an instance of the Enaml app class can be directly accessed from the main app class instance via the *framework\_app* attribute.

#### enaml application events

this portion is firing *application events* additional to the ones provided by *MainAppBase*. these framework app events get fired after *on\_app\_run()* in the following order:

- *on\_app\_build* (fired on start of the application event loop).
- *on\_app\_stopped* (fired after the main application window got closed)

#### Functions

---

*convert\_key\_event\_to\_code*(event)

converts the Qt key-press/-release event into a modifiers and key code string.

---

## Classes

<code>EnamlMainApp(**console_app_kwargs)</code>	enaml application main base class
<code>FrameworkApp(*args, **kwargs)</code>	enaml framework application class with atom member/attribute support.

### `convert_key_event_to_code(event)`

converts the Qt key-press/-release event into a modifiers and key code string.

#### Return type

`Tuple[str, str]`

### `class FrameworkApp(*args, **kwargs)`

Bases: `QtApplication`

enaml framework application class with atom member/attribute support.

#### `app_state_flow_id`

A value of type *str*.

By default, bytes will NOT be promoted to strings. Pass `strict=False` to the constructor to enable loose string checking.

#### `app_state_flow_path`

A List member which supports container notifications.

#### `app_state_font_size`

A value of type *float*.

By default, ints and longs will be promoted to floats. Pass `strict=True` to the constructor to enable strict float checking.

#### `app_state_light_theme`

A value of type *bool*.

#### `app_state_sound_volume`

A value of type *float*.

By default, ints and longs will be promoted to floats. Pass `strict=True` to the constructor to enable strict float checking.

#### `app_state_win_rectangle`

A member which allows tuple values.

If item validation is used, then assignment will create a copy of the original tuple before validating the items, since validation may change the item values.

#### `app_state_flow_id_ink`

A member which allows list values.

Assigning to a list creates a copy. The original list will remain unmodified. This is similar to the semantics of the assignment operator on the C++ STL container classes.

#### `app_state_flow_path_ink`

A member which allows list values.

Assigning to a list creates a copy. The original list will remain unmodified. This is similar to the semantics of the assignment operator on the C++ STL container classes.



**app\_state\_selected\_item\_ink**

A member which allows list values.

Assigning to a list creates a copy. The original list will remain unmodified. This is similar to the semantics of the assignment operator on the C++ STL container classes.

**app\_state\_unselected\_item\_ink**

A member which allows list values.

Assigning to a list creates a copy. The original list will remain unmodified. This is similar to the semantics of the assignment operator on the C++ STL container classes.

**landscape**

saved via win\_rectangle app state

**max\_font\_size**

maximum font size in pixels bound to window size

**min\_font\_size**

minimum - “ -

**mixed\_back\_ink**

mixed color - used for user pref dropdown opening button

```
__atom_specific_members__ = frozenset({'app_state_flow_id', 'app_state_flow_id_ink',
'app_state_flow_path', 'app_state_flow_path_ink', 'app_state_font_size',
'app_state_light_theme', 'app_state_selected_item_ink', 'app_state_sound_volume',
'app_state_unselected_item_ink', 'app_state_win_rectangle', 'landscape',
'max_font_size', 'min_font_size', 'mixed_back_ink'})
```

```
__slotnames__ = []
```

```
class EnamlMainApp(**console_app_kwargs)
```

Bases: *MainAppBase*

enaml application main base class

```
_original_key_press_handler: Optional[Callable] = None
```

```
_original_key_release_handler: Optional[Callable] = None
```

```
_original_win_resize_handler: Optional[Callable] = None
```

```
init_app(framework_app_class=<class 'ae.enaml_app.FrameworkApp'>)
```

initialize framework app instance and root window/layout, return GUI event loop start/stop methods.

**Return type**

*Tuple[Optional[Callable], Optional[Callable]]*

```
cached_icon(icon_name, size, light)
```

get cached image/icon object.

**Return type**

*Optional[Icon]*

```
call_method_delayed(_delay, callback, *args, **kwargs)
```

delay not implemented - for now redirect to direct call.

**Return type**

*Any*

**focus\_widget**(*widget*)

set input/keyboard focus to the passed widget.

**Parameters**

**widget**¶ (*Widget*) – widget/window that will receive the focus.

**focused\_widget**()

enaml/qt focus debug helper method determining tool tip of the current qt widget with focus

**Return type**

*str*

**Returns**

tool tip string of current focus or app window status.

**key\_press\_from\_enaml**(*event*)

convert/normalize enaml/Qt key press/down event and pass it to MainAppBase key press dispatcher.

**key\_release\_from\_enaml**(*event*)

convert/normalize enaml/Qt key release/up event and pass it to MainAppBase key release dispatcher.

**load\_images**()

overwrite un-cached image file register to use cached image files instead.

**on\_app\_run**()

run/start app event handler.

**on\_flow\_widget\_focused**()

set focus to the widget referenced by the current flow id.

**on\_font\_size\_change**(*\_flow\_key, event\_kwargs*)

font size app state flow change confirmation event handler.

**Parameters**

- **\_flow\_key**¶ (*str*) – flow key.
- **event\_kwargs**¶ (*Dict[str, Any]*) – event kwargs with key '*font\_size*' containing the font\_size in pixels.

**Return type**

*bool*

**Returns**

True to confirm change of flow id.

**open\_popup**(*popup\_class, \*\*popup\_kwargs*)

open Popup and set focus to the first widget.

**Parameters**

- **popup\_class**¶ (*Type*) – class of the Popup widget/window.
- **popup\_kwargs**¶ – args to instantiate and show/open the popup.

**Return type**

*Widget*

**Returns**

instance of the popup widget.

**play\_sound**(*sound\_name*)

play audio/sound file.

**user\_preference\_color\_selected**(*color\_name, dialog*)

ColorDialog callback.

**win\_activated**(*main\_window*)

main window activated event handler, called only once on app startup via widgets.enaml/ThemeMainWindow.

**Parameters**

**main\_window** *Any* –

**Returns**

**win\_closed**(*changed*)

callback fired on close of Main window to save/restore framework\_win.geometry on app exit/start.

**Parameters**

**changed** *dict* – qt/enaml changed event dict.

---

**Note:** neither self.framework\_app.stop() nor self.framework\_app.\_qapp.exit(exit\_code) trigger window closed event.

---

**win\_resize\_from\_enaml**(*event*)

convert/normalize enaml/Qt key press/down event and pass it to MainAppBase key press dispatcher.

## 4.46 ae.enaml\_app.functions

enaml application helper functions.

### Functions

<b>ae_rgba</b> (color)	convert enaml Color instance into ae color format.
<b>enaml_rgba</b> (red, green, blue, alpha)	convert ae color rgba floats into enaml compatible rgba integers.
<b>style_rgba</b> (red, green, blue, alpha)	convert ae color rgba floats into Qt style compatible rgba color string.

**ae\_rgba**(*color*)

convert enaml Color instance into ae color format.

**Parameters**

**color** *Color* – enaml Color instance.

**Return type**

*Tuple*[float, float, float, float]

**Returns**

rgba tuple with 4 float values (0.0 ... 1.0).

**enaml\_rgba**(*red, green, blue, alpha*)

convert ae color rgba floats into enaml compatible rgba integers.

**Parameters**

- **red** (float) – red color value (0.0 ... 1.0).
- **green** (float) – green color value (0.0 ... 1.0).
- **blue** (float) – blue color value (0.0 ... 1.0).
- **alpha** (float) – alpha/opacity value (0.0 ... 1.0).

**Return type**

Color

**Returns**

rgba enaml Color instance (0 ... 255).

**style\_rgba**(*red, green, blue, alpha*)

convert ae color rgba floats into Qt style compatible rgba color string.

**Parameters**

- **red** (float) – red color value (0.0 ... 1.0).
- **green** (float) – green color value (0.0 ... 1.0).
- **blue** (float) – blue color value (0.0 ... 1.0).
- **alpha** (float) – alpha/opacity value (0.0 ... 1.0).

**Return type**

str

**Returns**

rgba enaml Color instance (0 ... 255).

## INDICES AND TABLES

- [portion repositories at gitlab.com](#)
- [genindex](#)
- [modindex](#)
- [ae namespace projects and documentation](#)
- [aede namespace projects and documentation](#)



## PYTHON MODULE INDEX

### a

- ae.base, 14
- ae.console, 109
- ae.core, 72
- ae.db\_core, 196
- ae.db\_ora, 206
- ae.db\_pg, 208
- ae.deep, 30
- ae.django\_utils, 40
- ae.droid, 41
- ae.dynamicod, 88
- ae.enaml\_app, 363
- ae.enaml\_app.functions, 367
- ae.files, 46
- ae.gui\_app, 225
- ae.gui\_help, 252
- ae.i18n, 91
- ae.kivy, 285
- ae.kivy.apps, 306
- ae.kivy.behaviors, 313
- ae.kivy.i18n, 286
- ae.kivy.tours, 321
- ae.kivy.widgets, 287
- ae.kivy\_auto\_width, 328
- ae.kivy\_dyn\_chi, 280
- ae.kivy\_file\_chooser, 334
- ae.kivy\_gls, 272
- ae.kivy\_iterable\_displayer, 337
- ae.kivy\_qr\_displayer, 338
- ae.kivy\_relief\_canvas, 282
- ae.kivy\_sideload, 339
- ae.kivy\_user\_prefs, 345
- ae.lisz\_app\_data, 345
- ae.literal, 97
- ae.lockname, 86
- ae.notify, 41
- ae.parse\_date, 96
- ae.paths, 53
- ae.progress, 102
- ae.sideload\_server, 220
- ae.sys\_core, 122
- ae.sys\_core\_sh, 173
- ae.sys\_data, 128
- ae.sys\_data\_sh, 184
- ae.transfer\_service, 211
- ae.updater, 106
- ae.valid, 45

**latest :**

---



## Symbols

- `_APP_INSTANCES` (in module `ae.core`), 79
- `_APP_THREADS` (in module `ae.core`), 80
- `_ASP_DIR_LEN` (in module `ae.sys_data`), 132
- `_ASP_SYS_MIN_LEN` (in module `ae.sys_data`), 132
- `_ASP_TYPE_LEN` (in module `ae.sys_data`), 132
- `_DEBUG_RUNNING_CHARS` (in module `ae.sys_core_sh`), 175
- `_Field` (class in `ae.sys_data`), 157
- `_GetTextBinder` (class in `ae.kivy.i18n`), 286
- `_LOGGER` (in module `ae.core`), 77
- `_MAIN_APP_INST_KEY` (in module `ae.core`), 79
- `_MULTI_THREADING_ACTIVATED` (in module `ae.core`), 78
- `_PrintingReplicator` (class in `ae.core`), 79
- `_SihotTcpClient` (class in `ae.sys_core_sh`), 175
- `_TCP_END_OF_MSG_CHAR` (in module `ae.sys_core_sh`), 175
- `_TCP_MAXBUFLLEN` (in module `ae.sys_core_sh`), 175
- `_ThreadedServer` (class in `ae.sys_core_sh`), 176
- `__atom_specific_members__` (*FrameworkApp* attribute), 365
- `__bool__`() (*UnsetType* method), 20
- `__call__`() (*FilesRegister* method), 70
- `__call__`() (*\_GetTextBinder* method), 287
- `__contains__`() (*Record* method), 143
- `__del__`() (*AppBase* method), 82
- `__del__`() (*ConsoleApp* method), 116
- `__enter__`() (*NamedLocks* method), 88
- `__eq__`() (*RegisteredFile* method), 51
- `__events__` (*ContainerChildrenAutoWidthBehavior* attribute), 331
- `__events__` (*FlowPopup* attribute), 299
- `__events__` (*FlowSelector* attribute), 301
- `__events__` (*TouchableBehavior* attribute), 319
- `__exit__`() (*NamedLocks* method), 88
- `__getattr__`() (*\_PrintingReplicator* method), 80
- `__getitem__`() (*Record* method), 143
- `__getitem__`() (*Records* method), 154
- `__getitem__`() (*Value* method), 138
- `__getitem__`() (*\_Field* method), 157
- `__hash__` (*RegisteredFile* attribute), 52
- `__init__`() (*AbsolutePosSizeBinder* method), 290
- `__init__`() (*AnimatedOnboardingTour* method), 325
- `__init__`() (*AnimatedTourMixin* method), 323
- `__init__`() (*AppBase* method), 81
- `__init__`() (*AutoFontSizeBehavior* method), 330
- `__init__`() (*AvailCatInfoResponse* method), 179
- `__init__`() (*BulkFetcherBase* method), 195
- `__init__`() (*CachedFile* method), 52
- `__init__`() (*CatRoomResponse* method), 179
- `__init__`() (*ClientFetch* method), 191
- `__init__`() (*ClientFromSihot* method), 190
- `__init__`() (*ClientSearch* method), 191
- `__init__`() (*ClientToSihot* method), 193
- `__init__`() (*Collector* method), 68
- `__init__`() (*ConfigDictResponse* method), 179
- `__init__`() (*ConsoleApp* method), 114
- `__init__`() (*DbBase* method), 200
- `__init__`() (*DynamicChildrenBehavior* method), 281
- `__init__`() (*ExtTextInputCutCopyPaste* method), 294
- `__init__`() (*FilesRegister* method), 70
- `__init__`() (*FldMapXmlBuilder* method), 192
- `__init__`() (*FldMapXmlParser* method), 189
- `__init__`() (*FlowButton* method), 293
- `__init__`() (*FlowDropDown* method), 293
- `__init__`() (*FlowInput* method), 295
- `__init__`() (*FlowPopup* method), 299
- `__init__`() (*FlowSelector* method), 301
- `__init__`() (*FlowToggler* method), 304
- `__init__`() (*FrameworkApp* method), 308
- `__init__`() (*HelpToggler* method), 292
- `__init__`() (*Literal* method), 100
- `__init__`() (*MainAppBase* method), 242
- `__init__`() (*NamedLocks* method), 87
- `__init__`() (*Notifications* method), 43
- `__init__`() (*OnboardingTour* method), 265
- `__init__`() (*OraDb* method), 207
- `__init__`() (*PostgresDb* method), 210
- `__init__`() (*Progress* method), 104
- `__init__`() (*Record* method), 142
- `__init__`() (*Records* method), 154
- `__init__`() (*RegisteredFile* method), 51
- `__init__`() (*ReliefCanvas* method), 284
- `__init__`() (*ResBulkFetcher* method), 195

---

`__init__()` (*ResChange* method), 178  
`__init__()` (*ResFromSihot* method), 190  
`__init__()` (*ResKernelGet* method), 182  
`__init__()` (*ResKernelResponse* method), 179  
`__init__()` (*ResResponse* method), 179  
`__init__()` (*ResToSihot* method), 193  
`__init__()` (*RoomChange* method), 178  
`__init__()` (*SideloadMenuPopup* method), 341  
`__init__()` (*SideloadMenuTour* method), 341  
`__init__()` (*SihotXmlBuilder* method), 180  
`__init__()` (*SihotXmlParser* method), 177  
`__init__()` (*SimpleAutoTickerBehavior* method), 333  
`__init__()` (*SlideSelectBehavior* method), 318  
`__init__()` (*SystemBase* method), 124  
`__init__()` (*SystemConnectorBase* method), 126  
`__init__()` (*TcpServer* method), 176  
`__init__()` (*Tooltip* method), 305  
`__init__()` (*TouchableBehavior* method), 319  
`__init__()` (*TourBase* method), 261  
`__init__()` (*TourOverlay* method), 327  
`__init__()` (*UsedSystems* method), 126  
`__init__()` (*UserPreferencesTour* method), 267  
`__init__()` (*Values* method), 139  
`__init__()` (*\_Field* method), 157  
`__init__()` (*\_PrintingReplicator* method), 79  
`__init__()` (*\_SihotTcpClient* method), 175  
`__init_subclass__()` (*TourBase* class method), 261  
`__len__()` (*UnsetType* method), 20  
`__repr__()` (*FlowDropDown* method), 293  
`__repr__()` (*FlowInput* method), 295  
`__repr__()` (*FlowPopup* method), 299  
`__repr__()` (*ImageLabel* method), 292  
`__repr__()` (*Literal* method), 100  
`__repr__()` (*Record* method), 143  
`__repr__()` (*RegisteredFile* method), 52  
`__repr__()` (*SystemBase* method), 125  
`__repr__()` (*SystemConnectorBase* method), 126  
`__repr__()` (*Value* method), 138  
`__repr__()` (*Values* method), 140  
`__repr__()` (*\_Field* method), 157  
`__setitem__()` (*Record* method), 143  
`__setitem__()` (*Records* method), 154  
`__setitem__()` (*Value* method), 138  
`__slotnames__` (*FrameworkApp* attribute), 365  
`__str__()` (*AppStateSlider* method), 291  
`__str__()` (*Record* method), 143  
`__str__()` (*RegisteredFile* method), 52  
`__str__()` (*Value* method), 138  
`__str__()` (*Values* method), 140  
`__str__()` (*\_Field* method), 157  
`_ac_dropdown` (*FlowInput* attribute), 295  
`_actual_pos()` (*Tooltip* method), 305  
`_adapt_sql()` (*DbBase* method), 200  
`_add_animations()` (*AnimatedTourMixin* method), 324  
`_add_field()` (*Record* method), 146  
`_add_sihot_configs()` (*ResToSihot* method), 194  
`_add_system()` (*UsedSystems* method), 127  
`_added_animations` (*AnimatedOnboardingTour* attribute), 326  
`_added_shaders` (*AnimatedOnboardingTour* attribute), 326  
`_align_center()` (*ModalBehavior* method), 316  
`_anim_alpha` (*FlowPopup* attribute), 298  
`_anim_alpha` (*FlowSelector* attribute), 301  
`_anim_duration` (*FlowPopup* attribute), 298  
`_anim_duration` (*FlowSelector* attribute), 301  
`_app` (*Progress* attribute), 104  
`_append_eof_and_flush_file()` (*AppBase* method), 85  
`_arg_parser` (*ConsoleApp* attribute), 115  
`_attached_pos()` (*FlowSelector* method), 302  
`_attached_size()` (*FlowSelector* method), 302  
`_bind()` (*AbsolutePosSizeBinder* method), 290  
`_bind_properties()` (*SimpleAutoTickerBehavior* method), 333  
`_body_mime_type_conversion()` (in module *ae.notify*), 42  
`_bound_properties` (*SimpleAutoTickerBehavior* attribute), 333  
`_bound_uid` (*\_GetTextBinder* attribute), 286  
`_cancel_long_touch_clock()` (*TouchableBehavior* static method), 320  
`_cancel_slide_select_closer()` (*SlideSelectBehavior* static method), 318  
`_cancel_slide_select_opener()` (*SlideSelectBehavior* static method), 318  
`_center_aligned` (*ModalBehavior* attribute), 316  
`_cfg_files` (*ConsoleApp* attribute), 115  
`_cfg_opt_val_stripper` (*ConsoleApp* attribute), 115  
`_cfg_parser` (*ConsoleApp* attribute), 115  
`_change_option()` (*ConsoleApp* method), 120  
`_change_selector_index()` (*FlowInput* method), 295  
`_child_data_dict()` (in module *ae.kivy\_dyn\_chi*), 280  
`_chk_val_reset_else_set_type()` (*Literal* method), 101  
`_close_log_file()` (*AppBase* method), 86  
`_closing_popup_open_flow_id` (*HelpAppBase* attribute), 268  
`_collect_appends()` (*Collector* method), 69  
`_collect_selects()` (*Collector* method), 69  
`_compile_shader()` (*ShadersMixin* method), 278  
`_complete_client_data()` (*ClientToSihot* static method), 193  
`_complete_width` (*ContainerChildrenAutoWidthBehavior* attribute), 331  
`_container` (*DynamicChildrenBehavior* attribute), 281  
`_container` (*FileChooserPathSelectPopup* attribute), 336

- `_container` (*FileChooserPopup* attribute), 336
- `_container` (*IterableDisplayerPopup* attribute), 337
- `_container` (*MessageShowPopup* attribute), 304
- `_container` (*QrDisplayerPopup* attribute), 339
- `_container` (*SideloadMenuPopup* attribute), 341
- `_creation_direction` (*FlowSelector* attribute), 301
- `_deactivate_multi_threading()` (in module *ae.core*), 78
- `_debug_enable_clicks` (*KivyMainApp* attribute), 309
- `_debug_level` (*AppBase* attribute), 82
- `_default_object_loader()` (in module *ae.files*), 52
- `_delete_ac_text()` (*FlowInput* method), 296
- `_delta` (*Progress* attribute), 105
- `_detect_complete_width()` (*ContainerChildrenAutoWidthBehavior* method), 332
- `_determine_value()` (*Literal* method), 101
- `_end_msg` (*Progress* attribute), 105
- `_ensure_clients_exist_and_updated()` (*ResToSihot* method), 194
- `_ensure_system_value()` (*\_Field* method), 166
- `_err_counter` (*Progress* attribute), 105
- `_err_msg` (*Progress* attribute), 105
- `_exit_code` (*MainAppBase* attribute), 242
- `_fast_bound_center_uid` (*ModalBehavior* attribute), 316
- `_finalize_close()` (*FlowSelector* method), 302
- `_flush_and_close_log_buf()` (*AppBase* method), 86
- `_font_anim_mode` (*AutoFontSizeBehavior* attribute), 329
- `_font_size_adjustable()` (*AutoFontSizeBehavior* method), 330
- `_font_size_anim` (*AutoFontSizeBehavior* attribute), 329
- `_font_size_progress()` (*AutoFontSizeBehavior* method), 330
- `_gds_errors` (*ResSender* attribute), 196
- `_get_cfg_parser_val()` (*ConsoleApp* method), 117
- `_grab_and_open()` (*SlideSelectBehavior* method), 318
- `_handle_error()` (*ResToSihot* method), 194
- `_init_default_user_cfg_vars()` (*ConsoleApp* method), 115
- `_init_default_user_cfg_vars()` (*MainAppBase* method), 242
- `_init_default_user_cfg_vars()` (*SideloadMainAppMixin* method), 343
- `_init_logging()` (*ConsoleApp* method), 115
- `_item_moved()` (*FlowSelector* method), 302
- `_item_radian` (*FlowSelector* attribute), 301
- `_join_app_threads()` (in module *ae.core*), 80
- `_last_focus_flow_id` (*MainAppBase* attribute), 242
- `_last_font_size` (*AutoFontSizeBehavior* attribute), 329
- `_last_log_line_prefix` (*AppBase* attribute), 81
- `_layout_finished` (*FileChooserPathSelectPopup* attribute), 336
- `_layout_finished` (*FileChooserPopup* attribute), 336
- `_layout_finished` (*IterableDisplayerPopup* attribute), 338
- `_layout_finished` (*MessageShowPopup* attribute), 304
- `_layout_finished` (*QrDisplayerPopup* attribute), 339
- `_layout_finished` (*SideloadMenuPopup* attribute), 341
- `_layout_items()` (*FlowSelector* method), 302
- `_length_anim` (*SimpleAutoTickerBehavior* attribute), 333
- `_load_merge_cred_feat()` (*UsedSystems* method), 127
- `_lock_class` (*NamedLocks* attribute), 88
- `_lock_names` (*NamedLocks* attribute), 87
- `_log_buf_stream` (*AppBase* attribute), 81
- `_log_file_index` (*AppBase* attribute), 81
- `_log_file_name` (*AppBase* attribute), 81
- `_log_file_size_max` (*AppBase* attribute), 81
- `_log_file_stream` (*AppBase* attribute), 81
- `_log_with_timestamp` (*AppBase* attribute), 81
- `_main_cfg_fnam` (*ConsoleApp* attribute), 115
- `_main_cfg_mod_time` (*ConsoleApp* attribute), 115
- `_matching_ac_index` (*FlowInput* attribute), 295
- `_matching_ac_texts` (*FlowInput* attribute), 295
- `_max_height` (*FlowPopup* attribute), 298
- `_max_width` (*FlowPopup* attribute), 298
- `_min_text_len` (*SimpleAutoTickerBehavior* attribute), 333
- `_next_help_id` (*HelpAppBase* attribute), 268
- `_next_msg` (*Progress* attribute), 104
- `_normalize_col_values()` (in module *ae.db\_core*), 198
- `_nul_std_out` (*AppBase* attribute), 81
- `_offset_anim` (*SimpleAutoTickerBehavior* attribute), 333
- `_on_complete_opened()` (*ContainerChildrenAutoWidthBehavior* method), 332
- `_on_key_down()` (*ModalBehavior* method), 317
- `_on_win_width()` (*ContainerChildrenAutoWidthBehavior* method), 332
- `_open_log_file()` (*AppBase* method), 86
- `_open_width_progress()` (*ContainerChildrenAutoWidthBehavior* method), 332
- `_opened_item` (*FileChooserPathSelectPopup* attribute), 336
- `_opened_item` (*FileChooserPopup* attribute), 336
- `_opened_item` (*IterableDisplayerPopup* attribute), 338
- `_opened_item` (*MessageShowPopup* attribute), 304
- `_opened_item` (*QrDisplayerPopup* attribute), 339
- `_opened_item` (*SideloadMenuPopup* attribute), 341
- `_ori_text` (*SimpleAutoTickerBehavior* attribute), 333

- `_original_key_press_handler` (*EnamlMainApp attribute*), 365
- `_original_key_release_handler` (*EnamlMainApp attribute*), 365
- `_original_win_resize_handler` (*EnamlMainApp attribute*), 365
- `_os_platform()` (in module *ae.base*), 25
- `_parsed_arguments` (*ConsoleApp attribute*), 115
- `_path_match_replacement` (in module *ae.paths*), 66
- `_pos_changed()` (*ShadersMixin method*), 279
- `_pos_fbind_uid` (*ShadersMixin attribute*), 277
- `_prepare_guest_link_xml()` (*ClientToSihot method*), 193
- `_prepare_guest_xml()` (*ClientToSihot method*), 193
- `_prepare_in_clause()` (in module *ae.db\_core*), 199
- `_prepare_res_xml()` (*ResToSihot method*), 194
- `_print_func` (*NamedLocks attribute*), 88
- `_propagate()` (*AbsolutePosSizeBinder method*), 290
- `_real_dismiss()` (*FlowDropDown method*), 293
- `_rebind()` (in module *ae.db\_core*), 199
- `_receive_response()` (*\_SihotTcpClient method*), 176
- `_refresh_gls1()` (*ShadersMixin method*), 279
- `_refreshing_data` (*LiszDataMixin attribute*), 353
- `_register_app_instance()` (in module *ae.core*), 79
- `_register_app_thread()` (in module *ae.core*), 80
- `_rel_pos_changed()` (*AbsolutePosSizeBinder method*), 290
- `_rel_size_changed()` (*AbsolutePosSizeBinder method*), 290
- `_relief_ellipse_inner_refresh()` (*ReliefCanvas method*), 284
- `_relief_ellipse_outer_refresh()` (*ReliefCanvas method*), 285
- `_relief_refresh()` (*ReliefCanvas method*), 284
- `_relief_square_inner_refresh()` (*ReliefCanvas method*), 285
- `_relief_square_outer_refresh()` (*ReliefCanvas method*), 285
- `_rename_log_file()` (*AppBase method*), 86
- `_reposition()` (*FlowDropDown method*), 294
- `_run_counter` (*Progress attribute*), 105
- `_saved_app_states` (*OnboardingTour attribute*), 265
- `_saved_app_states` (*SideloadingMenuTour attribute*), 342
- `_saved_app_states` (*TourDropdownFromButton attribute*), 264
- `_saved_app_states` (*UserPreferencesTour attribute*), 267
- `_select_ac_text()` (*FlowInput method*), 296
- `_send_link_to_sihot()` (*ClientToSihot method*), 193
- `_send_person_to_sihot()` (*ClientToSihot method*), 193
- `_sending_res_to_sihot()` (*ResToSihot method*), 194
- `_shader_args` (*HelpBehavior attribute*), 315
- `_show_cut_copy_paste()` (*FlowInput method*), 296
- `_shut_down` (*AppBase attribute*), 81
- `_shut_down_sub_app_instances()` (in module *ae.core*), 79
- `_size_changed()` (*ShadersMixin method*), 279
- `_size_fbind_uid` (*ShadersMixin attribute*), 277
- `_start_event_loop` (*MainAppBase attribute*), 242
- `_start_font_anim()` (*AutoFontSizeBehavior method*), 330
- `_start_length_anim()` (*SimpleAutoTickerBehavior method*), 333
- `_start_offset_anim()` (*SimpleAutoTickerBehavior method*), 333
- `_start_radian` (*FlowSelector attribute*), 301
- `_std_out_err_redirection()` (*AppBase method*), 85
- `_stop_event_loop` (*MainAppBase attribute*), 242
- `_stop_font_anim()` (*AutoFontSizeBehavior method*), 330
- `_stop_length_anim()` (*SimpleAutoTickerBehavior method*), 333
- `_stop_offset_anim()` (*SimpleAutoTickerBehavior method*), 333
- `_strip_err_msg()` (in module *ae.sys\_data\_sh*), 189
- `_sys_lock` (*NamedLocks attribute*), 88
- `_text_changed()` (*SimpleAutoTickerBehavior method*), 333
- `_ticker_length_progress()` (*SimpleAutoTickerBehavior method*), 333
- `_ticker_max_offset()` (*SimpleAutoTickerBehavior method*), 333
- `_ticker_offset_progress()` (*SimpleAutoTickerBehavior method*), 334
- `_ticker_text_length` (*SimpleAutoTickerBehavior attribute*), 333
- `_ticker_text_offset` (*SimpleAutoTickerBehavior attribute*), 333
- `_ticker_text_update()` (*SimpleAutoTickerBehavior method*), 334
- `_ticker_text_updating` (*SimpleAutoTickerBehavior attribute*), 333
- `_total_count` (*Progress attribute*), 105
- `_touch_anim` (*TouchableBehavior attribute*), 319
- `_touch_moved_outside` (*FileChooserPathSelectPopup attribute*), 337
- `_touch_moved_outside` (*FileChooserPopup attribute*), 336
- `_touch_moved_outside` (*IterableDisplayerPopup attribute*), 338
- `_touch_moved_outside` (*MessageShowPopup attribute*), 304
- `_touch_moved_outside` (*QrDisplayerPopup attribute*), 339
- `_touch_moved_outside` (*SideloadingMenuPopup attribute*), 341



\_touch\_started\_inside (*ModalBehavior* attribute), 316  
 \_touch\_x (*TouchableBehavior* attribute), 319  
 \_touch\_y (*TouchableBehavior* attribute), 319  
 \_unbind\_properties() (*SimpleAutoTickerBehavior* method), 334  
 \_ungrab\_and\_close() (*SlideSelectBehavior* static method), 318  
 \_unregister\_app\_instance() (in module *ae.core*), 79  
 \_update\_shader() (*TouchableBehavior* method), 321  
 \_warning\_msgs (*ClientToSihot* attribute), 193  
 \_warning\_msgs (*ResSender* attribute), 196  
 \_warning\_msgs (*ResToSihot* attribute), 194  
 \_wid\_pos\_changed() (*AbsolutePosSizeBinder* method), 290  
 \_wid\_size\_changed() (*AbsolutePosSizeBinder* method), 290  
 \_width\_anim (*ContainerChildrenAutoWidthBehavior* attribute), 331  
 \_win\_width\_bind() (*ContainerChildrenAutoWidthBehavior* method), 332

## A

AbsolutePosSizeBinder (class in *ae.kivy.widgets*), 289  
 acquire() (*NamedLocks* method), 88  
 ACTION\_BUILD (in module *ae.sys\_data*), 131  
 ACTION\_COMPARE (in module *ae.sys\_data*), 131  
 ACTION\_DELETE (in module *ae.sys\_data*), 131  
 ACTION\_INSERT (in module *ae.sys\_data*), 131  
 ACTION\_PARSE (in module *ae.sys\_data*), 131  
 ACTION\_PULL (in module *ae.sys\_data*), 131  
 ACTION\_PUSH (in module *ae.sys\_data*), 131  
 ACTION\_SEARCH (in module *ae.sys\_data*), 131  
 ACTION\_UPDATE (in module *ae.sys\_data*), 131  
 ACTION\_UPSERT (in module *ae.sys\_data*), 131  
 ACTIONS\_CHANGING\_FLOW\_WITHOUT\_CONFIRMATION (in module *ae.gui\_app*), 235  
 ACTIONS\_EXTENDING\_FLOW\_PATH (in module *ae.gui\_app*), 235  
 ACTIONS\_REDUCING\_FLOW\_PATH (in module *ae.gui\_app*), 235  
 activate\_esc\_key\_close() (*ModalBehavior* method), 317  
 activate\_modal() (*ModalBehavior* method), 317  
 activate\_multi\_threading() (in module *ae.core*), 78  
 active\_lock\_counters (*NamedLocks* attribute), 87  
 active\_locks (*NamedLocks* attribute), 87  
 active\_log\_stream (*AppBase* property), 82  
 add\_arg() (*ConsoleApp* method), 119  
 add\_argument() (*ConsoleApp* method), 119  
 add\_aspects() (*\_Field* method), 163  
 add\_cfg\_files() (*ConsoleApp* method), 116  
 add\_common\_storage\_paths() (in module *ae.paths*), 62  
 add\_fields() (*Record* method), 146  
 add\_file() (*FilesRegister* method), 70  
 add\_files() (*FilesRegister* method), 70  
 add\_item() (*LiszDataMixin* method), 353  
 add\_items() (*LiszDataMixin* method), 353  
 add\_opt() (*ConsoleApp* method), 120  
 add\_option() (*ConsoleApp* method), 120  
 add\_options() (*BulkFetcherBase* method), 195  
 add\_options() (*ResBulkFetcher* method), 195  
 add\_paths() (*FilesRegister* method), 71  
 add\_property() (*RegisteredFile* method), 52  
 add\_register() (*FilesRegister* method), 71  
 add\_sh\_options() (in module *ae.sys\_data\_sh*), 187  
 add\_shader (*TouchableBehavior* attribute), 319  
 add\_shader() (*ShadersMixin* method), 277  
 add\_system\_fields() (*Record* method), 146  
 add\_tag() (*SihotXmlBuilder* method), 180  
 add\_widget() (*FlowPopup* method), 299  
 add\_widget() (*FlowSelector* method), 302  
 added\_shaders (*ShadersMixin* attribute), 277  
 ae.base  
     module, 14  
 ae.console  
     module, 109  
 ae.core  
     module, 72  
 ae.db\_core  
     module, 196  
 ae.db\_ora  
     module, 206  
 ae.db\_pg  
     module, 208  
 ae.deep  
     module, 30  
 ae.django\_utils  
     module, 40  
 ae.droid  
     module, 41  
 ae.dynamicod  
     module, 88  
 ae.enaml\_app  
     module, 363  
 ae.enaml\_app.functions  
     module, 367  
 ae.files  
     module, 46  
 ae.gui\_app  
     module, 225  
 ae.gui\_help  
     module, 252  
 ae.i18n  
     module, 91  
 ae.kivy  
     module, 285

---

ae.kivy.apps  
    module, 306

ae.kivy.behaviors  
    module, 313

ae.kivy.i18n  
    module, 286

ae.kivy.tours  
    module, 321

ae.kivy.widgets  
    module, 287

ae.kivy\_auto\_width  
    module, 328

ae.kivy\_dyn\_chi  
    module, 280

ae.kivy\_file\_chooser  
    module, 334

ae.kivy\_gls  
    module, 272

ae.kivy\_iterable\_displayer  
    module, 337

ae.kivy\_qr\_displayer  
    module, 338

ae.kivy\_relief\_canvas  
    module, 282

ae.kivy\_sideload  
    module, 339

ae.kivy\_user\_prefs  
    module, 345

ae.lisz\_app\_data  
    module, 345

ae.literal  
    module, 97

ae.lockname  
    module, 86

ae.notify  
    module, 41

ae.parse\_date  
    module, 96

ae.paths  
    module, 53

ae.progress  
    module, 102

ae.sideload\_server  
    module, 220

ae.sys\_core  
    module, 122

ae.sys\_core\_sh  
    module, 173

ae.sys\_data  
    module, 128

ae.sys\_data\_sh  
    module, 184

ae.transfer\_service  
    module, 211

ae.updater  
    module, 106

ae.valid  
    module, 45

ae\_rgba() (in module *ae.enaml\_app.functions*), 367

ALL\_FATS (in module *ae.sys\_data*), 132

ALL\_FIELDS (in module *ae.sys\_data*), 132

anchor\_layout\_x() (in module *ae.gui\_help*), 258

anchor\_layout\_y() (in module *ae.gui\_help*), 258

anchor\_points() (in module *ae.gui\_help*), 258

anchor\_spe (Tooltip attribute), 304

anchor\_spec() (in module *ae.gui\_help*), 259

AnchorSpecType (in module *ae.gui\_help*), 258

ANI\_SINE\_DEEPER\_REPEAT3 (in module *ae.kivy.widgets*), 289

ani\_start() (HelpToggler method), 292

ani\_start\_check() (in module *ae.kivy.tours*), 323

ani\_stop() (HelpToggler method), 292

ani\_value (HelpToggler attribute), 292

ani\_value (TourOverlay attribute), 327

animated\_widget\_values() (in module *ae.kivy.tours*), 323

AnimatedOnboardingTour (class in *ae.kivy.tours*), 325

AnimatedTourMixin (class in *ae.kivy.tours*), 323

app\_data\_path() (in module *ae.paths*), 62

app\_docs\_path() (in module *ae.paths*), 62

app\_env\_dict() (ConsoleApp method), 122

app\_env\_dict() (KivyMainApp method), 309

app\_inst\_lock (in module *ae.core*), 79

app\_key (AppBase property), 83

APP\_KEY\_SEP (in module *ae.core*), 78

app\_name (AppBase attribute), 82

app\_name (SideloadMainAppMixin attribute), 343

app\_name\_guess() (in module *ae.base*), 20

app\_path (AppBase attribute), 82

app\_path (SubApp attribute), 86

app\_state\_flow\_id (FrameworkApp attribute), 364

app\_state\_flow\_id\_ink (FrameworkApp attribute), 364

app\_state\_flow\_path (FrameworkApp attribute), 364

app\_state\_flow\_path\_ink (FrameworkApp attribute), 364

app\_state\_font\_size (FrameworkApp attribute), 364

APP\_STATE\_HELP\_ID\_PREFIX (in module *ae.gui\_help*), 257

app\_state\_keys() (MainAppBase method), 243

app\_state\_light\_theme (FrameworkApp attribute), 364

app\_state\_name (AppStateSlider attribute), 291

APP\_STATE\_SECTION\_NAME (in module *ae.gui\_app*), 234

app\_state\_selected\_item\_ink (FrameworkApp attribute), 364

app\_state\_sound\_volume (FrameworkApp attribute), 364

- app\_state\_unselected\_item\_ink (*FrameworkApp* attribute), 365  
 app\_state\_version (*MainAppBase* attribute), 241  
 APP\_STATE\_VERSION\_VAR\_NAME (in module *ae.gui\_app*), 234  
 app\_state\_win\_rectangle (*FrameworkApp* attribute), 364  
 app\_states (*FrameworkApp* attribute), 307  
 app\_title (*AppBase* attribute), 82  
 app\_version (*AppBase* attribute), 82  
 AppBase (class in *ae.core*), 80  
 append\_record() (*\_Field* method), 169  
 append\_record() (*Records* method), 155  
 APPEND\_TO\_END\_OF\_FILE\_LIST (in module *ae.paths*), 60  
 append\_value() (*Literal* method), 100  
 AppStateSlider (class in *ae.kivy.widgets*), 291  
 AppStateType (in module *ae.gui\_app*), 237  
 aspect\_exists() (*\_Field* method), 161  
 aspect\_key() (in module *ae.sys\_data*), 133  
 aspect\_key\_direction() (in module *ae.sys\_data*), 134  
 aspect\_key\_system() (in module *ae.sys\_data*), 133  
 aspect\_value() (*\_Field* method), 162  
 AspectKeyType (in module *ae.sys\_data*), 132  
 attach\_to (*IterableDisplayerPopup* attribute), 338  
 attach\_to (*MessageShowPopup* attribute), 304  
 attach\_to (*QrDisplayerPopup* attribute), 339  
 attach\_to (*SlideSelectBehavior* attribute), 318  
 attached\_widget (*FlowSelector* attribute), 300  
 AttrMapType (in module *ae.kivy\_dyn\_chi*), 280  
 auto\_complete\_selector\_index\_ink (*FlowInput* attribute), 295  
 auto\_complete\_texts (*FlowInput* attribute), 295  
 auto\_dismiss (*ModalBehavior* attribute), 316  
 auto\_font\_anim\_duration (*AutoFontSizeBehavior* attribute), 329  
 auto\_font\_max\_size (*AutoFontSizeBehavior* attribute), 330  
 auto\_font\_min\_size (*AutoFontSizeBehavior* attribute), 330  
 auto\_font\_text\_spacing (*AutoFontSizeBehavior* attribute), 329  
 auto\_switch\_pages (*OnboardingTour* attribute), 265  
 auto\_switch\_pages (*SideloadingMenuTour* attribute), 342  
 auto\_switch\_pages (*TourBase* attribute), 261  
 auto\_switch\_pages (*TourDropdownFromButton* attribute), 264  
 auto\_switch\_pages (*UserPreferencesTour* attribute), 267  
 auto\_ticker\_length\_anim\_duration (*SimpleAutoTickerBehavior* attribute), 333  
 auto\_ticker\_offset\_anim\_speed (*SimpleAutoTickerBehavior* attribute), 333  
 auto\_ticker\_text\_spacing (*SimpleAutoTickerBehavior* attribute), 333  
 auto\_width\_anim\_duration (*ContainerChildrenAutoWidthBehavior* attribute), 331  
 auto\_width\_child\_padding (*ContainerChildrenAutoWidthBehavior* attribute), 331  
 auto\_width\_minimum (*ContainerChildrenAutoWidthBehavior* attribute), 331  
 auto\_width\_start (*ContainerChildrenAutoWidthBehavior* attribute), 331  
 auto\_width\_window\_padding (*ContainerChildrenAutoWidthBehavior* attribute), 331  
 AutoFontSizeBehavior (class in *ae.kivy\_auto\_width*), 329  
 avail\_rooms() (*AvailCatInfo* method), 181  
 available\_rec\_types (*SystemBase* attribute), 125  
 AvailCatInfo (class in *ae.sys\_core\_sh*), 181  
 AvailCatInfoResponse (class in *ae.sys\_core\_sh*), 179
- ## B
- background\_color (*FlowPopup* attribute), 299  
 backup\_app\_states() (*TourBase* method), 263  
 backup\_config\_resources() (*MainAppBase* static method), 243  
 base\_globals (in module *ae.dynamicod*), 90  
 beg\_xml() (*SihotXmlBuilder* method), 180  
 bind (*AutoFontSizeBehavior* attribute), 329  
 bind (*DynamicChildrenBehavior* attribute), 281  
 bind (*ReliefCanvas* attribute), 284  
 bind (*SimpleAutoTickerBehavior* attribute), 332  
 build() (*FrameworkApp* method), 308  
 BUILD\_CONFIG\_FILE (in module *ae.base*), 19  
 build\_config\_variable\_values() (in module *ae.base*), 20  
 BUILT\_IN\_SHADERS (in module *ae.kivy\_gsl*), 274  
 BulkFetcherBase (class in *ae.sys\_data\_sh*), 195  
 button\_height (*FrameworkApp* attribute), 307
- ## C
- cached\_icon() (*EnamlMainApp* method), 365  
 CachedFile (class in *ae.files*), 52  
 calculator() (*\_Field* method), 165  
 call\_method (*LiszDataMixin* attribute), 353  
 call\_method() (*AppBase* method), 83  
 call\_method\_delayed (*LiszDataMixin* attribute), 353  
 call\_method\_delayed() (*EnamlMainApp* method), 365  
 call\_method\_delayed() (*HelpAppBase* method), 268  
 call\_method\_delayed() (*KivyMainApp* method), 310  
 call\_proc() (*DbBase* method), 201  
 CALLABLE\_SUFFIX (in module *ae.sys\_data*), 132  
 camel\_to\_snake() (in module *ae.base*), 20

- `cancel_auto_page_switch_request()` (*TourBase method*), 263
- `cancel_delayed_setup_layout_call()` (*TourBase method*), 263
- `cancel_request()` (*TransferServiceApp method*), 217
- `canvas` (*ReliefCanvas attribute*), 284
- `canvas` (*ShadersMixin attribute*), 276
- `CatRoomResponse` (*class in ae.sys\_core\_sh*), 179
- `CatRooms` (*class in ae.sys\_core\_sh*), 181
- `center` (*ModalBehavior attribute*), 316
- `center` (*ShadersMixin attribute*), 276
- `center_x` (*TouchableBehavior attribute*), 319
- `center_y` (*TouchableBehavior attribute*), 319
- `CFG_EXT` (*in module ae.base*), 19
- `cfg_opt_choices` (*ConsoleApp attribute*), 115
- `cfg_opt_eval_vars` (*ConsoleApp attribute*), 115
- `cfg_options` (*ConsoleApp attribute*), 115
- `cfg_section_variable_names()` (*ConsoleApp method*), 117
- `change_app_state` (*LiszDataMixin attribute*), 353
- `change_app_state` (*SideloadMainAppMixin attribute*), 343
- `change_app_state()` (*HelpAppBase method*), 269
- `change_app_state()` (*MainAppBase method*), 243
- `change_flow` (*LiszDataMixin attribute*), 353
- `change_flow` (*SideloadMainAppMixin attribute*), 343
- `change_flow()` (*HelpAppBase method*), 269
- `change_flow()` (*MainAppBase method*), 244
- `change_light_theme()` (*KivyMainApp method*), 310
- `change_observable()` (*MainAppBase method*), 243
- `change_sub_node_sel()` (*LiszDataMixin method*), 354
- `check_add()` (*Collector method*), 69
- `check_all()` (*in module ae.updater*), 108
- `check_copies()` (*in module ae.updater*), 107
- `check_item_id()` (*in module ae.lisz\_app\_data*), 351
- `check_local_bootstraps()` (*in module ae.updater*), 108
- `check_local_updates()` (*in module ae.updater*), 108
- `check_moves()` (*in module ae.updater*), 107
- `check_overwrites()` (*in module ae.updater*), 107
- `child_data_defaults` (*DynamicChildrenBehavior attribute*), 281
- `child_data_maps` (*DynamicChildrenBehavior attribute*), 281
- `ChildrenDataType` (*in module ae.kivy\_dyn\_chi*), 280
- `CHK_BIND_VAR_PREFIX` (*in module ae.db\_core*), 198
- `class_by_name()` (*KivyMainApp static method*), 310
- `class_by_name()` (*MainAppBase static method*), 244
- `clean_log_str()` (*in module ae.transfer\_service*), 215
- `clear_leaves()` (*\_Field method*), 169
- `clear_leaves()` (*Record method*), 148
- `clear_leaves()` (*Value method*), 139
- `clear_leaves()` (*Values method*), 142
- `clear_rec()` (*FldMapXmlParser method*), 189
- `clear_val()` (*\_Field method*), 166
- `client_data()` (*in module ae.sys\_data\_sh*), 187
- `client_handlers` (*SideloadServerApp attribute*), 222
- `client_id_by_matchcode()` (*ClientSearch method*), 191
- `client_progress()` (*SideloadServerApp method*), 223
- `ClientFetch` (*class in ae.sys\_data\_sh*), 191
- `ClientFromSihot` (*class in ae.sys\_data\_sh*), 190
- `clients_match_field_init()` (*ShSysConnector static method*), 183
- `ClientSearch` (*class in ae.sys\_data\_sh*), 191
- `ClientToSihot` (*class in ae.sys\_data\_sh*), 193
- `close` (*ModalBehavior attribute*), 316
- `close` (*SlideSelectBehavior attribute*), 318
- `close` (*TourOverlay attribute*), 327
- `close()` (*ContainerChildrenAutoWidthBehavior method*), 331
- `close()` (*DbBase method*), 201
- `close()` (*FlowDropDown method*), 293
- `close()` (*FlowPopup method*), 299
- `close()` (*FlowSelector method*), 302
- `close()` (*SihotXmlParser method*), 178
- `close_kwargs` (*FlowDropDown attribute*), 293
- `close_kwargs` (*FlowPopup attribute*), 297
- `close_kwargs` (*FlowSelector attribute*), 301
- `CLOSE_POPUP_FLOW_ID` (*in module ae.gui\_help*), 257
- `close_popups()` (*MainAppBase method*), 245
- `coll_files()` (*in module ae.paths*), 61
- `coll_folders()` (*in module ae.paths*), 61
- `coll_item_type()` (*in module ae.paths*), 60
- `coll_items()` (*in module ae.paths*), 60
- `CollArgType` (*in module ae.paths*), 60
- `CollCreatorReturnType` (*in module ae.paths*), 60
- `collect()` (*Collector method*), 69
- `collect_system_fields()` (*Record method*), 147
- `COLLECTED_FOLDER` (*in module ae.paths*), 60
- `Collector` (*class in ae.paths*), 68
- `collide_point` (*HelpBehavior attribute*), 315
- `collide_point` (*ModalBehavior attribute*), 316
- `collide_point` (*SlideSelectBehavior attribute*), 318
- `collide_point` (*TouchableBehavior attribute*), 319
- `collide_tap_thru_toggler()` (*Tooltip method*), 305
- `collide_tour_start_button()` (*Tooltip method*), 305
- `CollYieldItems` (*in module ae.paths*), 60
- `CollYieldType` (*in module ae.paths*), 60
- `COLOR_BLACK` (*in module ae.gui\_app*), 234
- `ColorOrInk` (*in module ae.gui\_app*), 237
- `ColorRGB` (*in module ae.gui\_app*), 237
- `ColorRGBA` (*in module ae.gui\_app*), 237
- `commit()` (*DbBase method*), 205
- `compare_leaves()` (*Record method*), 147
- `compare_records()` (*Records method*), 155



- compare\_val() (*Record method*), 148  
 compile\_data\_maps() (*IterableDisplayerPopup static method*), 337  
 complete\_res\_data() (*in module ae.sys\_data\_sh*), 187  
 compose\_current\_index() (*in module ae.sys\_data*), 135  
 config\_value() (*in module ae.sys\_core*), 124  
 config\_value\_string() (*in module ae.console*), 112  
 ConfigDict (*class in ae.sys\_core\_sh*), 182  
 ConfigDictResponse (*class in ae.sys\_core\_sh*), 179  
 conn (*DbBase attribute*), 200  
 conn\_error (*SystemBase attribute*), 125  
 connect() (*DbBase method*), 200  
 connect() (*OraDb method*), 207  
 connect() (*PostgresDb method*), 210  
 connect() (*ShSysConnector method*), 183  
 connect() (*SystemBase method*), 125  
 connect() (*SystemConnectorBase method*), 126  
 connect() (*UsedSystems method*), 127  
 connect\_and\_request() (*in module ae.transfer\_service*), 215  
 connect\_args\_from\_params() (*in module ae.db\_core*), 198  
 CONNECT\_ERR\_PREFIX (*in module ae.transfer\_service*), 214  
 connect\_params() (*DbBase method*), 201  
 connection (*SystemBase attribute*), 125  
 CONNECTION\_TIMEOUT (*in module ae.transfer\_service*), 214  
 console\_app (*SystemBase attribute*), 125  
 console\_app (*SystemConnectorBase attribute*), 126  
 ConsoleApp (*class in ae.console*), 113  
 container (*ContainerChildrenAutoWidthBehavior attribute*), 330  
 container (*DynamicChildrenBehavior attribute*), 281  
 container (*FlowPopup attribute*), 297  
 container (*FlowSelector attribute*), 302  
 ContainerChildrenAutoWidthBehavior (*class in ae.kivy\_auto\_width*), 330  
 content (*FlowDropDown attribute*), 293  
 content (*FlowPopup attribute*), 297  
 convert() (*\_Field method*), 167  
 convert\_date\_from\_sh() (*in module ae.sys\_data\_sh*), 185  
 convert\_date\_onto\_sh() (*in module ae.sys\_data\_sh*), 185  
 convert\_key\_event\_to\_code() (*in module ae.enaml\_app*), 364  
 convert\_value() (*Literal method*), 101  
 convert\_value\_to\_xml\_string() (*SihotXmlBuilder static method*), 180  
 converter() (*\_Field method*), 166  
 copy() (*\_Field method*), 170  
 copy() (*Record method*), 148  
 copy() (*Value method*), 139  
 copy() (*Values method*), 142  
 copy\_bytes() (*in module ae.files*), 49  
 copy\_file() (*in module ae.paths*), 63  
 copy\_files() (*in module ae.paths*), 63  
 copy\_tree() (*in module ae.paths*), 63  
 correct\_email() (*in module ae.valid*), 45  
 correct\_item\_id() (*in module ae.lisz\_app\_data*), 351  
 correct\_phone() (*in module ae.valid*), 46  
 create\_cursor() (*DbBase method*), 201  
 credentials (*SystemBase attribute*), 125  
 CRITICAL\_VIBRATE\_PATTERN (*in module ae.kivy.widgets*), 289  
 crx() (*\_Field method*), 173  
 current\_item\_or\_node\_literal() (*LiszDataMixin method*), 354  
 current\_node\_items (*LiszDataMixin attribute*), 352  
 current\_records\_idx() (*\_Field method*), 172  
 curs (*DbBase attribute*), 200  
 cursor\_description() (*DbBase method*), 201
- ## D
- data (*IterableDisplayerPopup attribute*), 337  
 data() (*AvailCatInfoResponse method*), 179  
 data() (*FldMapXmlParser method*), 189  
 data() (*ResChange method*), 178  
 data() (*SihotXmlParser method*), 177  
 DataItemType (*in module ae.kivy\_dyn\_chi*), 280  
 DataItemValueType (*in module ae.kivy\_dyn\_chi*), 280  
 DataLeafTypesType (*in module ae.deep*), 33  
 DATE\_ISO (*in module ae.base*), 19  
 date\_range\_chunks() (*in module ae.sys\_data\_sh*), 188  
 date\_range\_str() (*ResBulkFetcher method*), 195  
 DATE\_TIME\_ISO (*in module ae.base*), 19  
 DbBase (*class in ae.db\_core*), 199  
 deactivate\_esc\_key\_close() (*ModalBehavior method*), 317  
 deactivate\_modal() (*ModalBehavior method*), 317  
 debug (*AppBase property*), 83  
 debug\_level (*AppBase property*), 83  
 debug\_level (*ConsoleApp property*), 116  
 debug\_level (*LiszDataMixin attribute*), 353  
 DEBUG\_LEVEL\_DISABLED (*in module ae.core*), 77  
 DEBUG\_LEVEL\_ENABLED (*in module ae.core*), 77  
 DEBUG\_LEVEL\_VERBOSE (*in module ae.core*), 77  
 DEBUG\_LEVELS (*in module ae.core*), 77  
 debug\_out() (*AppBase method*), 85  
 deep\_dict\_update() (*in module ae.base*), 20  
 deep\_replace() (*in module ae.deep*), 34  
 deep\_search() (*in module ae.deep*), 35  
 deep\_update() (*in module ae.deep*), 36  
 deeper() (*in module ae.sys\_data*), 134  
 DEF\_ENC\_PORT (*in module ae.notify*), 42  
 DEF\_ENC\_SERVICE\_NAME (*in module ae.notify*), 42

DEF\_ENCODE\_ERRORS (in module *ae.base*), 19  
 DEF\_ENCODING (in module *ae.base*), 19  
 DEF\_ENCODING (in module *ae.i18n*), 93  
 DEF\_FADE\_OUT\_APP (in module *ae.kivy.tours*), 323  
 DEF\_LANGUAGE (in module *ae.i18n*), 93  
 default\_encoding() (in module *ae.i18n*), 93  
 DEFAULT\_FILE\_MASK (in module *ae.sideloadng\_server*), 221  
 DEFAULT\_FPS (in module *ae.kivy\_gsl*), 274  
 default\_language() (in module *ae.i18n*), 94  
 DEFAULT\_LEAF\_TYPES (in module *ae.deep*), 33  
 default\_locale (in module *ae.i18n*), 93  
 del\_aspect() (*\_Field* method), 162  
 del\_name() (*\_Field* method), 163  
 del\_shader (*TouchableBehavior* attribute), 319  
 del\_shader() (*ShadersMixin* method), 278  
 delete() (*DbBase* method), 202  
 delete\_items() (*LiszDataMixin* method), 354  
 delete\_text\_from\_ac() (*FlowInput* method), 296  
 determine\_page\_ids (*TourDropDownFromButton* attribute), 264  
 disabled (*ModalBehavior* attribute), 316  
 disabled (*TouchableBehavior* attribute), 319  
 disconnect() (*SystemBase* method), 125  
 disconnect() (*UsedSystems* method), 127  
 dismiss (*ContainerChildrenAutoWidthBehavior* attribute), 330  
 dismiss() (*FlowDropDown* method), 293  
 dismiss() (*FlowPopup* method), 299  
 dismiss() (*FlowSelector* method), 303  
 dispatch (*ContainerChildrenAutoWidthBehavior* attribute), 330  
 dispatch (*SlideSelectBehavior* attribute), 318  
 dispatch (*TouchableBehavior* attribute), 319  
 displayed\_help\_id (*FrameworkApp* attribute), 307  
 displayed\_help\_id (*HelpAppBase* attribute), 268  
 do\_GET() (*SimpleHTTPRequestHandler* method), 222  
 DOCS\_FOLDER (in module *ae.base*), 19  
 documents\_root\_path (*KivyMainApp* attribute), 309  
 down\_shader (*TouchableBehavior* attribute), 320  
 dpi\_factor() (*KivyMainApp* static method), 310  
 dpi\_factor() (*MainAppBase* static method), 245  
 dpo (*SideloadngMainAppMixin* attribute), 343  
 dpo() (*AppBase* method), 85  
 dpo() (*NamedLocks* method), 88  
 dummy\_function() (in module *ae.base*), 21  
 duplicates() (in module *ae.base*), 21  
 DynamicChildrenBehavior (class in *ae.kivy\_dyn\_chi*), 281

## E

edit\_validate() (*LiszDataMixin* method), 354  
 elem\_path\_join() (in module *ae.sys\_data\_sh*), 187  
 elem\_to\_attr() (in module *ae.sys\_core\_sh*), 175

ellipse\_polar\_radius() (in module *ae.gui\_app*), 237  
 enaml\_rgba() (in module *ae.enaml\_app.functions*), 367  
 EnamlMainApp (class in *ae.enaml\_app*), 365  
 ENCODING\_KWARGS (in module *ae.transfer\_service*), 214  
 end() (*CatRoomResponse* method), 179  
 end() (*ClientFromSihot* method), 190  
 end() (*ConfigDictResponse* method), 179  
 end() (*FldMapXmlParser* method), 190  
 end() (*ResFromSihot* method), 190  
 end() (*SihotXmlParser* method), 177  
 end\_xml() (*SihotXmlBuilder* method), 180  
 ensure\_tap\_kwargs\_refs() (in module *ae.gui\_app*), 238  
 ensure\_top\_most\_z\_index() (*HelpAppBase* method), 269  
 ensure\_top\_most\_z\_index() (*KivyMainApp* method), 310  
 env\_str() (in module *ae.base*), 21  
 ERR\_MESSAGE\_PREFIX\_CONTINUE (in module *ae.sys\_core\_sh*), 175  
 error\_message (*\_SihotTcpClient* attribute), 175  
 error\_message (*Collector* property), 70  
 error\_message (*RequestXmlHandler* attribute), 176  
 ERROR\_VIBRATE\_PATTERN (in module *ae.kivy.widgets*), 289  
 evaluable\_literal() (in module *ae.literal*), 98  
 EventKwargsType (in module *ae.gui\_app*), 237  
 exec\_with\_return() (in module *ae.dynamicod*), 89  
 execute\_sql() (*DbBase* method), 202  
 execute\_sql() (*PostgresDb* method), 210  
 explained\_pos (*TourOverlay* attribute), 327  
 explained\_size (*TourOverlay* attribute), 327  
 explained\_widget (*TourOverlay* attribute), 327  
 ExplainedMatcherType (in module *ae.gui\_help*), 258  
 export\_node() (*LiszDataMixin* method), 355  
 ext (*CachedFile* attribute), 52  
 ext (*RegisteredFile* attribute), 51  
 extend\_ac\_with\_text() (*FlowInput* method), 296  
 ExtTextInputCutCopyPaste (class in *ae.kivy.widgets*), 294

## F

FAD\_FROM (in module *ae.sys\_data*), 132  
 FAD\_ONTO (in module *ae.sys\_data*), 132  
 fade\_out\_app (*TourOverlay* attribute), 327  
 failed (*Collector* attribute), 68  
 FAT\_CAL (in module *ae.sys\_data*), 131  
 FAT\_CHK (in module *ae.sys\_data*), 131  
 FAT\_CLEAR\_VAL (in module *ae.sys\_data*), 131  
 FAT\_CNV (in module *ae.sys\_data*), 132  
 FAT\_FLT (in module *ae.sys\_data*), 132  
 FAT\_IDX (in module *ae.sys\_data*), 131  
 FAT\_RCX (in module *ae.sys\_data*), 131  
 FAT\_REC (in module *ae.sys\_data*), 131

- FAT\_SQE (in module *ae.sys\_data*), 132
- FAT\_VAL (in module *ae.sys\_data*), 131
- fbind (*ModalBehavior* attribute), 316
- fbind (*ShadersMixin* attribute), 276
- fbind() (*\_GetTextBinder* method), 286
- features (*SystemBase* attribute), 125
- fetch\_all() (*DbBase* method), 201
- fetch\_all() (*GuestBulkFetcher* method), 195
- fetch\_all() (*ResBulkFetcher* method), 195
- fetch\_by\_gds\_no() (*ResFetch* method), 192
- fetch\_by\_res\_id() (*ResFetch* method), 192
- fetch\_client() (*ClientFetch* method), 191
- fetch\_log\_entries() (*SideloadingServerApp* method), 223
- fetch\_res() (*ResFetch* method), 191
- fetch\_res\_no() (*ResKernelGet* method), 183
- fetch\_value() (*DbBase* method), 202
- field\_name\_idx\_path() (in module *ae.sys\_data*), 134
- field\_names\_idx\_paths() (in module *ae.sys\_data*), 134
- FieldCallable (in module *ae.sys\_data*), 133
- FieldValCallable (in module *ae.sys\_data*), 133
- file\_chooser\_initial\_path (*SideloadingMainAppMixin* attribute), 343
- file\_chooser\_paths (*SideloadingMainAppMixin* attribute), 343
- FILE\_COUNT\_MISMATCH (in module *ae.sideload\_server*), 221
- file\_lines() (in module *ae.files*), 50
- file\_path (*SideloadingServerApp* attribute), 222
- file\_transfer\_progress() (in module *ae.files*), 50
- FileChooserPathSelectPopup (class in *ae.kivy\_file\_chooser*), 336
- FileChooserPopup (class in *ae.kivy\_file\_chooser*), 335
- FilenameOrStream (in module *ae.files*), 49
- FileObject (in module *ae.files*), 48
- files (*Collector* attribute), 68
- FilesRegister (class in *ae.paths*), 70
- filter\_selected (*LiszDataMixin* attribute), 352
- filter\_unselected (*LiszDataMixin* attribute), 353
- filtered\_indexes (*LiszDataMixin* attribute), 352
- filterer() (*\_Field* method), 167
- filters (*FileChooserPopup* attribute), 336
- find\_aspect\_key() (*\_Field* method), 160
- find\_file() (*FilesRegister* method), 72
- find\_image() (*MainAppBase* method), 245
- find\_item\_index() (*LiszDataMixin* method), 355
- find\_sound() (*MainAppBase* method), 245
- find\_widget() (*MainAppBase* method), 245
- finished() (*Progress* method), 105
- FldMapXmlBuilder (class in *ae.sys\_data\_sh*), 192
- FldMapXmlParser (class in *ae.sys\_data\_sh*), 189
- flow\_action() (in module *ae.gui\_app*), 238
- FLOW\_ACTION\_RE (in module *ae.gui\_app*), 235
- flow\_action\_split() (in module *ae.gui\_app*), 238
- flow\_change\_confirmation\_event\_name() (in module *ae.gui\_app*), 238
- flow\_class\_name() (in module *ae.gui\_app*), 239
- FLOW\_HELP\_ID\_PREFIX (in module *ae.gui\_help*), 257
- flow\_id (*LiszDataMixin* attribute), 353
- flow\_id (*MainAppBase* attribute), 241
- flow\_id\_ink (*MainAppBase* attribute), 241
- flow\_key() (in module *ae.gui\_app*), 239
- FLOW\_KEY\_SEP (in module *ae.gui\_app*), 235
- flow\_key\_split() (in module *ae.gui\_app*), 239
- flow\_key\_text() (*LiszDataMixin* method), 355
- flow\_object() (in module *ae.gui\_app*), 239
- FLOW\_OBJECT\_RE (in module *ae.gui\_app*), 235
- flow\_path (*LiszDataMixin* attribute), 353
- flow\_path (*MainAppBase* attribute), 242
- flow\_path\_action (*LiszDataMixin* attribute), 353
- flow\_path\_action() (*MainAppBase* method), 245
- flow\_path\_from\_text() (*LiszDataMixin* method), 355
- flow\_path\_id() (in module *ae.gui\_app*), 239
- flow\_path\_ink (*MainAppBase* attribute), 241
- flow\_path\_items\_from\_text() (in module *ae.lisz\_app\_data*), 352
- flow\_path\_node() (*LiszDataMixin* method), 356
- flow\_path\_quick\_jump\_nodes() (*LiszDataMixin* method), 356
- FLOW\_PATH\_ROOT\_ID (in module *ae.lisz\_app\_data*), 350
- flow\_path\_strip() (in module *ae.gui\_app*), 240
- flow\_path\_text() (*LiszDataMixin* method), 356
- FLOW\_PATH\_TEXT\_SEP (in module *ae.lisz\_app\_data*), 350
- flow\_popup\_class\_name() (in module *ae.gui\_app*), 240
- FlowButton (class in *ae.kivy.widgets*), 292
- FlowDropDown (class in *ae.kivy.widgets*), 293
- FlowInput (class in *ae.kivy.widgets*), 294
- FlowPopup (class in *ae.kivy.widgets*), 297
- FlowSelector (class in *ae.kivy.widgets*), 300
- FlowToggler (class in *ae.kivy.widgets*), 303
- focus\_flow\_id (*FlowInput* attribute), 295
- FOCUS\_FLOW\_PREFIX (in module *ae.lisz\_app\_data*), 350
- focus\_neighbour\_item() (*LiszDataMixin* method), 356
- focus\_widget() (*EnamlMainApp* method), 365
- focused\_widget() (*EnamlMainApp* method), 366
- font\_color (*FrameworkApp* attribute), 307
- font\_size (*AutoFontSizeBehavior* attribute), 329
- font\_size (*MainAppBase* attribute), 241
- force\_encoding() (in module *ae.base*), 21
- framework\_app (*MainAppBase* attribute), 242
- framework\_root (*MainAppBase* attribute), 242
- framework\_root (*SideloadingMainAppMixin* attribute), 343
- framework\_win (*MainAppBase* attribute), 242



FrameworkApp (class in *ae.enaml\_app*), 364  
 FrameworkApp (class in *ae.kivy.apps*), 307  
 full\_stack\_trace() (in module *ae.base*), 22  
 funbind (ModalBehavior attribute), 316  
 funbind() (*\_GetTextBinder* method), 286

## G

gds\_no\_to\_ids() (in module *ae.sys\_data\_sh*), 188  
 gds\_no\_to\_obj\_id() (in module *ae.sys\_data\_sh*), 188  
 generic\_language() (in module *ae.django\_utils*), 40  
 get\_arg() (ConsoleApp method), 120  
 get\_argument() (ConsoleApp method), 119  
 get\_cat\_rooms() (CatRooms method), 181  
 get\_current\_index() (in module *ae.sys\_data*), 135  
 get\_end\_message() (Progress method), 105  
 get\_f\_string() (in module *ae.i18n*), 94  
 get\_key\_values() (ConfigDict method), 182  
 get\_operation\_code() (Request method), 178  
 get\_opt (SideloadMainAppMixin attribute), 343  
 get\_opt() (ConsoleApp method), 121  
 get\_option() (ConsoleApp method), 120  
 get\_res\_no() (ResSender method), 196  
 get\_root\_window (SimpleAutoTickerBehavior attribute), 332  
 get\_row\_count() (DbBase method), 205  
 get\_text() (in module *ae.i18n*), 94  
 get\_txt (in module *ae.kivy.i18n*), 287  
 get\_txt\_ (KivyMainApp attribute), 309  
 get\_value() (OraDb static method), 208  
 get\_var() (ConsoleApp method), 118  
 get\_variable() (ConsoleApp method), 118  
 get\_warnings() (FldMapXmlBuilder method), 192  
 get\_xml() (SihotXmlParser method), 177  
 global\_variables() (KivyMainApp method), 310  
 global\_variables() (LiszDataMixin method), 357  
 global\_variables() (MainAppBase method), 246  
 GuestBulkFetcher (class in *ae.sys\_data\_sh*), 195

## H

handle() (RequestXmlHandler method), 176  
 handle() (ThreadedTCPRequestHandler method), 217  
 handle\_xml() (RequestXmlHandler method), 176  
 has\_name() (*\_Field* method), 164  
 has\_tour (Tooltip attribute), 304  
 height (AutoFontSizeBehavior attribute), 329  
 help\_activation\_toggle() (HelpAppBase method), 269  
 help\_activation\_toggle() (KivyMainApp method), 311  
 help\_activator (HelpAppBase attribute), 268  
 help\_app\_state\_display() (HelpAppBase method), 269  
 help\_display() (HelpAppBase method), 269  
 help\_flow\_display() (HelpAppBase method), 270

help\_id (HelpBehavior attribute), 315  
 help\_id\_tour\_class() (in module *ae.gui\_help*), 259  
 help\_is\_inactive() (HelpAppBase method), 270  
 help\_layout (FrameworkApp attribute), 307  
 help\_layout (HelpAppBase attribute), 268  
 help\_lock (HelpBehavior attribute), 315  
 help\_sub\_id() (in module *ae.gui\_help*), 259  
 help\_target\_and\_id() (HelpAppBase method), 270  
 help\_vars (HelpBehavior attribute), 315  
 help\_widget() (HelpAppBase method), 270  
 HelpAppBase (class in *ae.gui\_help*), 268  
 HelpBehavior (class in *ae.kivy.behaviors*), 315  
 HelpToggler (class in *ae.kivy.widgets*), 292  
 HIDDEN\_CREDENTIALS (in module *ae.core*), 77  
 HIDDEN\_GLOBALS (in module *ae.gui\_app*), 235  
 HIDDEN\_SHADER\_PARAMETERS (in module *ae.kivy\_gsl*), 276  
 hide\_dup\_line\_prefix() (in module *ae.core*), 77  
 hotel\_and\_res\_id() (in module *ae.sys\_data\_sh*), 188

## I

id\_of\_flow() (in module *ae.gui\_app*), 240  
 id\_of\_flow\_help() (in module *ae.gui\_help*), 260  
 id\_of\_state\_help() (in module *ae.gui\_help*), 260  
 id\_of\_task() (SideloadServerApp static method), 223  
 id\_of\_task() (TransferServiceApp static method), 217  
 id\_of\_tour\_help() (in module *ae.gui\_help*), 260  
 idx\_path\_field\_name() (in module *ae.sys\_data*), 134  
 IDX\_PATH\_SEP (in module *ae.sys\_data*), 132  
 IDX\_TYPES (in module *ae.sys\_data*), 132  
 IdxItemType (in module *ae.sys\_data*), 132  
 IdxPathType (in module *ae.sys\_data*), 132  
 IdxTypes (in module *ae.sys\_data*), 132  
 IGNORED\_HELP\_FLOWS (in module *ae.gui\_help*), 257  
 IGNORED\_HELP\_STATES (in module *ae.gui\_help*), 257  
 image\_files (MainAppBase attribute), 242  
 ImageLabel (class in *ae.kivy.widgets*), 292  
 img\_file() (MainAppBase method), 246  
 import\_file\_info() (LiszDataMixin static method), 357  
 import\_items() (LiszDataMixin method), 357  
 import\_module() (in module *ae.base*), 22  
 import\_node() (LiszDataMixin method), 357  
 IMPORT\_NODE\_MAX\_FILE\_LEN (in module *ae.lisz\_app\_data*), 351  
 IMPORT\_NODE\_MAX\_ITEMS (in module *ae.lisz\_app\_data*), 351  
 importable\_node\_files() (LiszDataMixin method), 357  
 in\_actions() (*\_Field* method), 172  
 in\_wd() (in module *ae.base*), 22  
 ina() (*\_Field* method), 172  
 index\_match\_fields() (Records method), 156

INI\_EXT (in module *ae.base*), 19  
 init\_app() (*EnamlMainApp* method), 365  
 init\_app() (*KivyMainApp* method), 309  
 init\_app() (*MainAppBase* method), 243  
 init\_current\_index() (in module *ae.sys\_data*), 135  
 init\_logging() (*AppBase* method), 83  
 initial\_path (*FileChooserPopup* attribute), 335  
 initialized (*Value* property), 138  
 insert() (*DbBase* method), 202  
 INSERT\_AT\_BEGIN\_OF\_FILE\_LIST (in module *ae.paths*), 60  
 instances (*SystemBase* attribute), 124  
 instances (*SystemConnectorBase* attribute), 126  
 instantiate\_config\_parser() (in module *ae.base*), 22  
 INVALID\_ITEM\_ID\_PREFIX\_CHARS (in module *ae.lisz\_app\_data*), 351  
 is\_main\_cfg\_file\_modified() (*ConsoleApp* method), 118  
 is\_modal (*ModalBehavior* attribute), 316  
 is\_open (*FlowSelector* attribute), 301  
 item\_by\_id() (*LiszDataMixin* method), 358  
 item\_sel\_filter() (in module *ae.lisz\_app\_data*), 352  
 item\_unsel\_filter() (in module *ae.lisz\_app\_data*), 352  
 IterableDisplayPopup (class in *ae.kivy\_iterable\_displayer*), 337

## K

kbd\_input\_mode (*KivyMainApp* attribute), 309  
 key\_filter\_default() (in module *ae.deep*), 34  
 key\_list\_object() (in module *ae.deep*), 36  
 key\_path\_object() (in module *ae.deep*), 37  
 KEY\_PATH\_SEPARATORS (in module *ae.deep*), 34  
 key\_path\_string() (in module *ae.deep*), 38  
 key\_press\_from\_enaml() (*EnamlMainApp* method), 366  
 key\_press\_from\_framework() (*HelpAppBase* method), 271  
 key\_press\_from\_framework() (*MainAppBase* method), 246  
 key\_press\_from\_kivy() (*FrameworkApp* method), 308  
 key\_release\_from\_enaml() (*EnamlMainApp* method), 366  
 key\_release\_from\_kivy() (*FrameworkApp* method), 308  
 key\_value() (in module *ae.deep*), 38  
 keyboard\_on\_key\_down() (*FlowInput* method), 296  
 keyboard\_on\_textinput() (*FlowInput* method), 296  
 KeyFilterCallable (in module *ae.deep*), 33  
 KeyType (in module *ae.deep*), 33  
 KivyMainApp (class in *ae.kivy\_apps*), 309

## L

label\_height (*TourOverlay* attribute), 327  
 landscape (*FrameworkApp* attribute), 307, 365  
 lang\_code (*MainAppBase* attribute), 241  
 LanguageMessages (in module *ae.i18n*), 93  
 last\_err\_msg (*OraDb* attribute), 208  
 last\_err\_msg (*PostgresDb* attribute), 211  
 last\_err\_msg (*ShSysConnector* attribute), 183  
 last\_err\_msg (*SystemConnectorBase* attribute), 126  
 last\_page\_id (*TourBase* property), 263  
 last\_page\_idx (*OnboardingTour* attribute), 267  
 last\_page\_idx (*SideloadMenuTour* attribute), 343  
 last\_page\_idx (*TourBase* attribute), 262  
 last\_page\_idx (*TourDropdownFromButton* attribute), 265  
 last\_page\_idx (*UserPreferencesTour* attribute), 268  
 layout (*AnimatedOnboardingTour* attribute), 326  
 layout (*AnimatedTourMixin* attribute), 323  
 layout (*TourBase* attribute), 263  
 leaf\_indexes() (*\_Field* method), 160  
 leaf\_indexes() (*Record* method), 149  
 leaf\_indexes() (*Records* method), 156  
 leaf\_names() (*Record* method), 149  
 leaf\_value() (*\_Field* method), 159  
 leaves() (*\_Field* method), 160  
 leaves() (*Record* method), 149  
 leaves() (*Records* method), 156  
 light\_theme (*MainAppBase* attribute), 242  
 LIST\_TYPES (in module *ae.sys\_data*), 173  
 ListType (in module *ae.sys\_data*), 133  
 LiszDataMixin (class in *ae.lisz\_app\_data*), 352  
 LiszItem (in module *ae.lisz\_app\_data*), 351  
 LiszNode (in module *ae.lisz\_app\_data*), 351  
 Literal (class in *ae.literal*), 98  
 load\_app\_states() (*MainAppBase* method), 246  
 load\_cfg\_files() (*ConsoleApp* method), 118  
 load\_images() (*EnamlMainApp* method), 366  
 load\_images() (*MainAppBase* method), 246  
 load\_language\_file() (in module *ae.i18n*), 95  
 load\_language\_texts() (in module *ae.i18n*), 95  
 load\_options() (*BulkFetcherBase* method), 195  
 load\_options() (*ResBulkFetcher* method), 195  
 load\_page\_data() (*TourBase* method), 263  
 load\_requests (*SideloadServerApp* attribute), 222  
 load\_sounds() (*KivyMainApp* method), 311  
 load\_sounds() (*MainAppBase* method), 247  
 load\_translations() (*MainAppBase* method), 247  
 load\_user\_cfg() (*ConsoleApp* method), 121  
 loaded\_object (*CachedFile* property), 53  
 LOADED\_TRANSLATIONS (in module *ae.i18n*), 93  
 locks\_change\_lock (*NamedLocks* attribute), 87  
 log() (*SideloadServerApp* method), 223  
 log() (*TransferServiceApp* method), 217  
 log\_file\_check() (*AppBase* method), 84

LOG\_FILE\_IDX\_WIDTH (in module *ae.core*), 77  
 log\_file\_lock (in module *ae.core*), 77  
 LOG\_FILE\_MAX\_SIZE (in module *ae.core*), 77  
 log\_line\_prefix() (*AppBase* method), 84  
 log\_request() (*SimpleHTTPRequestHandler* method), 222  
 logger\_late\_init() (in module *ae.core*), 78  
 LOGGING\_LEVELS (in module *ae.core*), 77  
 long\_tap\_flow\_id (*FlowButton* attribute), 292  
 long\_tap\_flow\_id (*FlowToggler* attribute), 303  
 LOVE\_VIBRATE\_PATTERN (in module *ae.kivy.widgets*), 289

## M

main\_app (*AnimatedOnboardingTour* attribute), 326  
 main\_app (*AnimatedTourMixin* attribute), 323  
 main\_app (*FrameworkApp* attribute), 308  
 main\_app (*TouchableBehavior* attribute), 319  
 main\_app (*TourBase* attribute), 262  
 main\_app\_instance() (in module *ae.core*), 79  
 main\_file\_paths\_parts() (in module *ae.base*), 22  
 MAIN\_KV\_FILE\_NAME (in module *ae.kivy.widgets*), 289  
 MAIN\_SECTION\_NAME (in module *ae.console*), 112  
 MainAppBase (class in *ae.gui\_app*), 241  
 match\_key() (*Record* method), 150  
 max\_font\_size (*FrameworkApp* attribute), 307, 365  
 MAX\_FONT\_SIZE (in module *ae.gui\_app*), 235  
 MAX\_NUM\_LOG\_FILES (in module *ae.core*), 77  
 menu\_items (*FlowDropDown* attribute), 293  
 menu\_items (*FlowPopup* attribute), 298  
 menu\_items (*FlowSelector* attribute), 302  
 merge\_leaves() (*Record* method), 150  
 merge\_records() (*Records* method), 156  
 merge\_values() (*Record* method), 150  
 message (*MessageShowPopup* attribute), 304  
 MessageShowPopup (class in *ae.kivy.widgets*), 304  
 min\_font\_size (*FrameworkApp* attribute), 307, 365  
 MIN\_FONT\_SIZE (in module *ae.gui\_app*), 235  
 missing\_fields() (*Record* method), 151  
 mix\_background\_ink() (*MainAppBase* method), 247  
 mixed\_back\_ink (*FrameworkApp* attribute), 307, 365  
 ModalBehavior (class in *ae.kivy.behaviors*), 315  
 module  
   ae.base, 14  
   ae.console, 109  
   ae.core, 72  
   ae.db\_core, 196  
   ae.db\_ora, 206  
   ae.db\_pg, 208  
   ae.deep, 30  
   ae.django\_utils, 40  
   ae.droid, 41  
   ae.dynamicod, 88  
   ae.enaml\_app, 363

ae.enaml\_app.functions, 367  
 ae.files, 46  
 ae.gui\_app, 225  
 ae.gui\_help, 252  
 ae.i18n, 91  
 ae.kivy, 285  
 ae.kivy.apps, 306  
 ae.kivy.behaviors, 313  
 ae.kivy.i18n, 286  
 ae.kivy.tours, 321  
 ae.kivy.widgets, 287  
 ae.kivy\_auto\_width, 328  
 ae.kivy\_dyn\_chi, 280  
 ae.kivy\_file\_chooser, 334  
 ae.kivy\_gls, 272  
 ae.kivy\_iterable\_displayer, 337  
 ae.kivy\_qr\_displayer, 338  
 ae.kivy\_relief\_canvas, 282  
 ae.kivy\_sideload, 339  
 ae.kivy\_user\_prefs, 345  
 ae.lisz\_app\_data, 345  
 ae.literal, 97  
 ae.lockname, 86  
 ae.notify, 41  
 ae.parse\_date, 96  
 ae.paths, 53  
 ae.progress, 102  
 ae.sideload\_server, 220  
 ae.sys\_core, 122  
 ae.sys\_core\_sh, 173  
 ae.sys\_data, 128  
 ae.sys\_data\_sh, 184  
 ae.transfer\_service, 211  
 ae.updater, 106  
 ae.valid, 45  
 module\_attr() (in module *ae.base*), 23  
 module\_file\_path() (in module *ae.base*), 23  
 module\_name() (in module *ae.base*), 23  
 move\_file() (in module *ae.paths*), 63  
 move\_files() (in module *ae.paths*), 63  
 move\_item() (*LiszDataMixin* method), 358  
 move\_tree() (in module *ae.paths*), 63  
 MSG\_FILE\_SUFFIX (in module *ae.i18n*), 93  
 MessageType (in module *ae.i18n*), 93  
 MutableDataTypes (in module *ae.deep*), 33

## N

name() (*\_Field* method), 163  
 NAME\_PARTS\_SEP (in module *ae.base*), 19  
 NAMED\_BIND\_VAR\_PREFIX (in module *ae.db\_core*), 198  
 NamedLocks (class in *ae.lockname*), 87  
 navigation\_disabled (*TourOverlay* attribute), 327  
 new\_tag() (*SihotXmlBuilder* static method), 180  
 next() (*Progress* method), 105



next\_key\_of\_path() (in module *ae.deep*), 38  
 next\_page() (*AnimatedOnboardingTour* method), 326  
 next\_page() (*AnimatedTourMixin* method), 324  
 next\_page() (*TourBase* method), 263  
 next\_page() (*TourOverlay* method), 327  
 next\_tick() (*ShadersMixin* method), 278  
 node\_child() (*\_Field* method), 158  
 node\_child() (*Record* method), 143  
 node\_child() (*Value* method), 138  
 node\_child() (*Values* method), 140  
 NODE\_CHILD\_TYPES (in module *ae.sys\_data*), 173  
 NODE\_FILE\_EXT (in module *ae.lisz\_app\_data*), 351  
 NODE\_FILE\_PREFIX (in module *ae.lisz\_app\_data*), 351  
 node\_info() (*LiszDataMixin* method), 358  
 NODE\_TYPES (in module *ae.sys\_data*), 173  
 NodeChildType (in module *ae.sys\_data*), 133  
 NodeFileInfo (in module *ae.lisz\_app\_data*), 351  
 NodeFilesInfo (in module *ae.lisz\_app\_data*), 351  
 NodeType (in module *ae.sys\_data*), 133  
 norm\_line\_sep() (in module *ae.base*), 23  
 norm\_name() (in module *ae.base*), 24  
 norm\_path() (in module *ae.base*), 24  
 normal\_shader (*TouchableBehavior* attribute), 320  
 normalize() (in module *ae.paths*), 64  
 Notifications (class in *ae.notify*), 43  
 notify() (*RequestXmlHandler* method), 176  
 now\_str() (in module *ae.base*), 24

## O

obj\_id\_to\_res\_no() (in module *ae.sys\_data\_sh*), 189  
 ObjCheckCallable (in module *ae.deep*), 33  
 object\_item\_value() (in module *ae.deep*), 39  
 object\_items() (in module *ae.deep*), 38  
 ObjKeyType (in module *ae.deep*), 33  
 observers (*\_GetTextBinder* attribute), 286  
 on\_\_anim\_alpha() (*FlowPopup* method), 299  
 on\_added\_shaders() (*ShadersMixin* method), 279  
 on\_alt\_tap() (*TouchableBehavior* method), 320  
 on\_app\_build() (*KivyMainApp* method), 311  
 on\_app\_build() (*MainAppBase* method), 247  
 on\_app\_built() (*KivyMainApp* method), 311  
 on\_app\_exit() (*MainAppBase* method), 247  
 on\_app\_init() (*MainAppBase* method), 247  
 on\_app\_pause() (*KivyMainApp* method), 311  
 on\_app\_quit() (*MainAppBase* method), 247  
 on\_app\_resume() (*KivyMainApp* method), 311  
 on\_app\_run() (*EnamlMainApp* method), 366  
 on\_app\_run() (*KivyMainApp* method), 311  
 on\_app\_run() (*MainAppBase* method), 247  
 on\_app\_run() (*SideloadMainAppMixin* method), 343  
 on\_app\_start() (*KivyMainApp* method), 311  
 on\_app\_started() (*HelpAppBase* method), 271  
 on\_app\_started() (*KivyMainApp* method), 311  
 on\_app\_started() (*MainAppBase* method), 247  
 on\_app\_started() (*SideloadMainAppMixin* method), 343  
 on\_app\_state\_root\_node\_save() (*LiszDataMixin* method), 359  
 on\_app\_stopped() (*KivyMainApp* method), 311  
 on\_complete\_opened() (*ContainerChildrenAutoWidthBehavior* method), 331  
 on\_container() (*FlowDropDown* method), 293  
 on\_content() (*FlowPopup* method), 299  
 on\_debug\_level\_change() (*MainAppBase* method), 247  
 on\_debug\_level\_change() (*SideloadMainAppMixin* method), 344  
 on\_dismiss() (*FlowDropDown* method), 294  
 on\_dismiss() (*FlowPopup* method), 300  
 on\_dismiss() (*FlowSelector* method), 303  
 on\_double\_tap() (*TouchableBehavior* method), 320  
 on\_down\_shader() (*TouchableBehavior* method), 320  
 on\_file\_chooser\_entry\_added() (*FileChooserPopup* static method), 336  
 on\_file\_chooser\_submit() (*SideloadMainAppMixin* method), 344  
 on\_filter\_toggle() (*LiszDataMixin* method), 359  
 on\_flow\_change() (*MainAppBase* method), 247  
 on\_flow\_id\_ink() (*MainAppBase* method), 248  
 on\_flow\_path\_ink() (*MainAppBase* method), 248  
 on\_flow\_popup\_close() (*HelpAppBase* method), 271  
 on\_flow\_popup\_close() (*MainAppBase* static method), 248  
 on\_flow\_widget\_focused() (*EnamlMainApp* method), 366  
 on\_flow\_widget\_focused() (*KivyMainApp* method), 311  
 on\_focus() (*FlowInput* method), 296  
 on\_font\_size\_change() (*EnamlMainApp* method), 366  
 on\_item\_enter() (*LiszDataMixin* method), 359  
 on\_item\_leave() (*LiszDataMixin* method), 360  
 on\_item\_sel\_change() (*LiszDataMixin* method), 360  
 on\_item\_sel\_confirming() (*LiszDataMixin* method), 360  
 on\_item\_sel\_toggle() (*LiszDataMixin* method), 360  
 on\_kbd\_input\_mode\_change() (*KivyMainApp* method), 311  
 on\_key\_press() (*LiszDataMixin* method), 360  
 on\_lang\_code() (*KivyMainApp* method), 311  
 on\_lang\_code\_change() (*MainAppBase* method), 248  
 on\_light\_theme() (*KivyMainApp* method), 311  
 on\_light\_theme\_change() (*MainAppBase* method), 248  
 on\_long\_tap() (*FlowButton* method), 293  
 on\_long\_tap() (*FlowToggler* method), 304  
 on\_long\_tap() (*TouchableBehavior* method), 320

- `on_navigation_disabled()` (*TourOverlay* method), 328
  - `on_node_extract()` (*LiszDataMixin* method), 361
  - `on_node_jump()` (*LiszDataMixin* method), 361
  - `on_normal_shader()` (*TouchableBehavior* method), 320
  - `on_open()` (*FlowPopup* method), 300
  - `on_open()` (*FlowSelector* method), 303
  - `on_parent()` (*ExtTextInputCutCopyPaste* method), 294
  - `on_parent()` (*ShadersMixin* method), 279
  - `on_pause()` (*FrameworkApp* method), 308
  - `on_pre_dismiss()` (*FlowPopup* method), 300
  - `on_pre_dismiss()` (*FlowSelector* method), 303
  - `on_pre_open()` (*FlowPopup* method), 300
  - `on_pre_open()` (*FlowSelector* method), 303
  - `on_release()` (*FlowButton* method), 293
  - `on_release()` (*FlowSelector* method), 303
  - `on_release()` (*FlowToggler* method), 304
  - `on_resume()` (*FrameworkApp* method), 308
  - `on_selected_item_ink()` (*MainAppBase* method), 248
  - `on_sideloading_server_start()` (*Sideloading-MainAppMixin* method), 344
  - `on_sideloading_server_stop()` (*Sideloading-MainAppMixin* method), 344
  - `on_size()` (*Tooltip* method), 305
  - `on_start()` (*FrameworkApp* method), 309
  - `on_state()` (*TouchableBehavior* method), 320
  - `on_stop()` (*FrameworkApp* method), 309
  - `on_targeted_widget()` (*Tooltip* method), 305
  - `on_text()` (*FlowInput* method), 296
  - `on_touch_down()` (*FlowDropDown* method), 294
  - `on_touch_down()` (*HelpBehavior* method), 315
  - `on_touch_down()` (*HelpToggler* method), 292
  - `on_touch_down()` (*ModalBehavior* method), 317
  - `on_touch_down()` (*Tooltip* method), 305
  - `on_touch_down()` (*TouchableBehavior* method), 320
  - `on_touch_move()` (*ModalBehavior* method), 317
  - `on_touch_move()` (*SlideSelectBehavior* method), 318
  - `on_touch_move()` (*TouchableBehavior* method), 321
  - `on_touch_up()` (*ModalBehavior* method), 317
  - `on_touch_up()` (*SlideSelectBehavior* method), 318
  - `on_touch_up()` (*TouchableBehavior* method), 321
  - `on_triple_tap()` (*TouchableBehavior* method), 321
  - `on_unselected_item_ink()` (*MainAppBase* method), 248
  - `on_user_add()` (*MainAppBase* method), 249
  - `on_user_preferences_open()` (*KivyMainApp* method), 311
  - `on_value()` (*AppStateSlider* method), 291
  - OnboardingTour* (class in *ae.gui\_help*), 265
  - opacity* (*ContainerChildrenAutoWidthBehavior* attribute), 331
  - `open()` (*ContainerChildrenAutoWidthBehavior* method), 331
  - `open()` (*FlowPopup* method), 300
  - `open()` (*FlowSelector* method), 303
  - `open_popup()` (*EnamlMainApp* method), 366
  - `open_popup()` (*KivyMainApp* method), 312
  - `open_popup()` (*MainAppBase* method), 249
  - optimal\_content\_height* (*FlowPopup* attribute), 298
  - optimal\_content\_width* (*FlowPopup* attribute), 298
  - OraDb* (class in *ae.db\_ora*), 207
  - ori\_std\_err* (in module *ae.core*), 77
  - ori\_std\_out* (in module *ae.core*), 77
  - os\_host\_name()* (in module *ae.base*), 24
  - os\_local\_ip()* (in module *ae.base*), 25
  - os\_platform* (in module *ae.base*), 25
  - os\_user\_name()* (in module *ae.base*), 25
  - overlay\_color* (*FlowPopup* attribute), 299
  - overlay\_color* (*FlowSelector* attribute), 301
- ## P
- PACKAGE\_INCLUDE\_FILES\_PREFIX* (in module *ae.base*), 19
  - page\_data* (*OnboardingTour* attribute), 266
  - page\_data* (*SideloadingMenuTour* attribute), 342
  - page\_data* (*TourBase* attribute), 261
  - page\_data* (*TourDropDownFromButton* attribute), 264
  - page\_data* (*UserPreferencesTour* attribute), 267
  - page\_ids* (*AnimatedOnboardingTour* attribute), 326
  - page\_ids* (*AnimatedTourMixin* attribute), 323
  - page\_ids* (*OnboardingTour* attribute), 266
  - page\_ids* (*SideloadingMenuTour* attribute), 341
  - page\_ids* (*TourBase* attribute), 262
  - page\_ids* (*TourDropDownFromButton* attribute), 265
  - page\_ids* (*UserPreferencesTour* attribute), 268
  - page\_idx* (*AnimatedOnboardingTour* attribute), 326
  - page\_idx* (*AnimatedTourMixin* attribute), 323
  - page\_idx* (*OnboardingTour* attribute), 265
  - page\_idx* (*SideloadingMenuTour* attribute), 343
  - page\_idx* (*TourBase* attribute), 262
  - page\_idx* (*TourDropDownFromButton* attribute), 265
  - page\_idx* (*UserPreferencesTour* attribute), 268
  - page\_updated()* (*TourOverlay* method), 328
  - PageAnimationsType* (in module *ae.kivy.tours*), 322
  - PageAnimationType* (in module *ae.kivy.tours*), 322
  - pages\_animations* (*AnimatedOnboardingTour* attribute), 326
  - pages\_animations* (*AnimatedTourMixin* attribute), 324
  - pages\_explained\_matchers* (*OnboardingTour* attribute), 266
  - pages\_explained\_matchers* (*SideloadingMenuTour* attribute), 342
  - pages\_explained\_matchers* (*TourBase* attribute), 262
  - pages\_explained\_matchers* (*TourDropDownFromButton* attribute), 265



- pages\_explained\_matchers (*UserPreferencesTour* attribute), 267
- pages\_shaders (*AnimatedOnboardingTour* attribute), 326
- pages\_shaders (*AnimatedTourMixin* attribute), 324
- param\_style (*DbBase* attribute), 200
- param\_style (*OraDb* attribute), 208
- param\_style (*PostgresDb* attribute), 210
- parent (*ContainerChildrenAutoWidthBehavior* attribute), 331
- parent (*ShadersMixin* attribute), 276
- parent() (*\_Field* method), 170
- parent\_popup\_to\_close (*FlowDropDown* attribute), 293
- parent\_popup\_to\_close (*FlowPopup* attribute), 298
- parent\_popup\_to\_close (*FlowSelector* attribute), 301
- PARENT\_TYPES (in module *ae.sys\_data*), 173
- parse\_arguments() (*ConsoleApp* method), 121
- parse\_date() (in module *ae.parse\_date*), 96
- parse\_xml() (*SihotXmlParser* method), 177
- path (*CachedFile* attribute), 52
- path (*RegisteredFile* attribute), 51
- path\_files() (in module *ae.paths*), 64
- path\_folders() (in module *ae.paths*), 64
- path\_items() (in module *ae.paths*), 65
- path\_join() (in module *ae.paths*), 65
- path\_match() (in module *ae.paths*), 66
- path\_name() (in module *ae.paths*), 66
- PATH\_PLACEHOLDERS (in module *ae.paths*), 68
- PATH\_TYPES (in module *ae.sys\_data*), 132
- paths (*Collector* attribute), 68
- paths (*FileChooserPathSelectPopup* attribute), 336
- paths\_match() (in module *ae.paths*), 66
- pax\_count() (in module *ae.sys\_data\_sh*), 188
- pending\_requests() (*TransferServiceApp* method), 218
- placeholder\_key() (in module *ae.paths*), 66
- placeholder\_path() (in module *ae.paths*), 67
- placeholders (*Collector* attribute), 69
- play\_beep() (*KivyMainApp* method), 312
- play\_beep() (*MainAppBase* method), 249
- play\_shader() (*ShadersMixin* method), 279
- play\_sound (*LiszDataMixin* attribute), 353
- play\_sound() (*EnamlMainApp* method), 366
- play\_sound() (*KivyMainApp* method), 312
- play\_sound() (*MainAppBase* method), 249
- play\_vibrate() (*KivyMainApp* method), 312
- play\_vibrate() (*MainAppBase* method), 249
- plural\_key() (in module *ae.i18n*), 95
- po() (*AppBase* method), 84
- po() (in module *ae.core*), 78
- pop() (*Record* method), 151
- pop\_object() (in module *ae.deep*), 39
- popups\_opened() (*MainAppBase* method), 249
- PORTIONS\_IMAGES (in module *ae.gui\_app*), 235
- PORTIONS\_SOUNDS (in module *ae.gui\_app*), 237
- pos (*ReliefCanvas* attribute), 284
- pos (*ShadersMixin* attribute), 276
- pos\_to\_attribute() (*AbsolutePosSizeBinder* method), 291
- pos\_to\_callback() (*AbsolutePosSizeBinder* method), 291
- post\_message() (*PostMessage* method), 182
- PostgresDb (class in *ae.db\_pg*), 210
- PostMessage (class in *ae.sys\_core\_sh*), 182
- prefix\_failed (*Collector* attribute), 68
- prepare\_map\_xml() (*FldMapXmlBuilder* method), 192
- prepare\_rec() (*ClientToSihot* method), 193
- prepare\_rec() (*FldMapXmlBuilder* method), 192
- prepare\_rec() (*ResToSihot* method), 194
- prepare\_ref\_param() (*OraDb* method), 207
- prev\_page() (*TourBase* method), 263
- prev\_page() (*TourOverlay* method), 328
- print\_options() (*BulkFetcherBase* method), 195
- print\_options() (*ResBulkFetcher* method), 195
- print\_out() (*AppBase* method), 84
- print\_out() (in module *ae.core*), 78
- print\_sh\_options() (in module *ae.sys\_data\_sh*), 187
- Progress (class in *ae.progress*), 104
- progress\_callback() (*SideloadingServerApp* method), 222
- progress\_total (*SimpleHTTPRequestHandler* attribute), 222
- progress\_transferred (*SimpleHTTPRequestHandler* attribute), 222
- project\_main\_file() (in module *ae.base*), 25
- properties (*CachedFile* attribute), 52
- properties (*RegisteredFile* attribute), 51
- PropertiesType (in module *ae.files*), 49
- PropertyType (in module *ae.files*), 48
- pull() (*\_Field* method), 170
- pull() (*Record* method), 151
- push() (*\_Field* method), 171
- push() (*Record* method), 151
- PY\_CACHE\_FOLDER (in module *ae.base*), 19
- PY\_EXT (in module *ae.base*), 19
- PY\_INIT (in module *ae.base*), 19
- py\_log\_params (*AppBase* attribute), 81
- PY\_MAIN (in module *ae.base*), 19
- Python Enhancement Proposals
- PEP 249, 196
  - PEP 249#paramstyle, 198
  - PEP 420, 1, 11
  - PEP 526, 7
- ## Q
- qr\_content (*QrDisplayerPopup* attribute), 339
- QrDisplayerPopup (class in *ae.kivy\_qr\_displayer*), 339

query\_data\_maps (*FlowPopup* attribute), 298

## R

radian\_offset (*FlowSelector* attribute), 301

read\_file() (in module *ae.base*), 26

read\_file\_text() (in module *ae.files*), 50

rec (*FldMapXmlParser* property), 189

received\_xml (*\_SihotTcpClient* attribute), 175

reclassify() (*FilesRegister* method), 72

Record (class in *ae.sys\_data*), 142

record\_field\_val() (*\_Field* method), 171

Records (class in *ae.sys\_data*), 153

recv\_bytes() (in module *ae.transfer\_service*), 215

recv\_file() (*TransferServiceApp* method), 218

recv\_message() (*TransferServiceApp* method), 218

refresh\_all() (*LiszDataMixin* method), 361

refresh\_child\_data\_widgets() (*DynamicChildrenBehavior* method), 281

refresh\_current\_node\_items\_from\_flow\_path() (*LiszDataMixin* method), 361

refresh\_node\_widgets (*LiszDataMixin* attribute), 353

refresh\_running\_shaders() (*ShadersMixin* method), 279

refresh\_shader() (*ShadersMixin* method), 279

register\_file\_path() (*FileChooserPopup* static method), 336

register\_package\_images() (in module *ae.gui\_app*), 240

register\_package\_sounds() (in module *ae.gui\_app*), 241

register\_package\_translations() (in module *ae.i18n*), 95

register\_tour\_class() (in module *ae.gui\_help*), 260

register\_translations\_path() (in module *ae.i18n*), 95

register\_user() (*ConsoleApp* method), 121

registered\_app\_names() (in module *ae.core*), 79

REGISTERED TOURS (in module *ae.gui\_help*), 258

RegisteredFile (class in *ae.files*), 51

release() (*NamedLocks* method), 88

relief\_colors() (in module *ae.kivy\_relief\_canvas*), 282

relief\_ellipse\_inner\_colors (*ReliefCanvas* attribute), 283

relief\_ellipse\_inner\_lines (*ReliefCanvas* attribute), 284

relief\_ellipse\_inner\_offset (*ReliefCanvas* attribute), 284

relief\_ellipse\_outer\_colors (*ReliefCanvas* attribute), 284

relief\_ellipse\_outer\_lines (*ReliefCanvas* attribute), 284

relief\_pos\_size (*ReliefCanvas* attribute), 283

relief\_square\_inner\_colors (*ReliefCanvas* attribute), 284

relief\_square\_inner\_lines (*ReliefCanvas* attribute), 284

relief\_square\_inner\_offset (*ReliefCanvas* attribute), 284

relief\_square\_outer\_colors (*ReliefCanvas* attribute), 284

relief\_square\_outer\_lines (*ReliefCanvas* attribute), 284

ReliefBrightness (in module *ae.kivy\_relief\_canvas*), 282

ReliefCanvas (class in *ae.kivy\_relief\_canvas*), 283

ReliefColors (in module *ae.kivy\_relief\_canvas*), 282

remove\_widget() (*FlowSelector* method), 303

RENDER\_SHAPES (in module *ae.kivy\_gsl*), 276

replace\_flow\_action() (in module *ae.gui\_app*), 241

replace\_object() (in module *ae.deep*), 39

reqs\_and\_logs (*TransferServiceApp* attribute), 217

Request (class in *ae.sys\_core\_sh*), 178

request\_app\_permissions() (in module *ae.base*), 29

request\_auto\_page\_switch() (*TourBase* method), 263

requested\_language() (in module *ae.django\_utils*), 40

requests\_lock (in module *ae.transfer\_service*), 215

RequestXmlHandler (class in *ae.sys\_core\_sh*), 176

res\_id\_desc() (*ResToSihot* method), 194

res\_id\_label() (*ResToSihot* static method), 194

res\_id\_values() (*ResToSihot* static method), 194

res\_no\_to\_ids() (in module *ae.sys\_data\_sh*), 188

res\_no\_to\_obj\_id() (in module *ae.sys\_data\_sh*), 188

res\_search() (in module *ae.sys\_data\_sh*), 188

ResBulkFetcher (class in *ae.sys\_data\_sh*), 195

ResChange (class in *ae.sys\_core\_sh*), 178

reset\_width\_detection() (*ContainerChildrenAutoWidthBehavior* method), 332

ResFetch (class in *ae.sys\_data\_sh*), 191

ResFromSihot (class in *ae.sys\_data\_sh*), 190

ResKernelGet (class in *ae.sys\_core\_sh*), 182

ResKernelResponse (class in *ae.sys\_core\_sh*), 179

response (*AvailCatInfo* attribute), 181

response (*CatRooms* attribute), 182

response (*ClientFetch* attribute), 191

response (*ClientSearch* attribute), 191

response (*ClientToSihot* attribute), 193

response (*ConfigDict* attribute), 182

response (*FldMapXmlBuilder* attribute), 193

response (*PostMessage* attribute), 182

response (*ResFetch* attribute), 192

response (*ResKernelGet* attribute), 183

response (*ResSearch* attribute), 192

response (*ResSender* attribute), 196

response (*ResToSihot* attribute), 194

- response\_to\_request() (*SideloadingServerApp* method), 223  
 response\_to\_request() (*TransferServiceApp* method), 219  
 ResResponse (class in *ae.sys\_core\_sh*), 179  
 ResSearch (class in *ae.sys\_data\_sh*), 192  
 ResSender (class in *ae.sys\_data\_sh*), 195  
 restore\_app\_states() (*TourBase* method), 263  
 restore\_widget\_values() (in module *ae.kivy.tours*), 323  
 ResToSihot (class in *ae.sys\_data\_sh*), 193  
 retrieve\_app\_states() (*MainAppBase* method), 250  
 rfv() (*\_Field* method), 172  
 rollback() (*DbBase* method), 205  
 RoomChange (class in *ae.sys\_core\_sh*), 178  
 root\_idx() (*\_Field* method), 165  
 root\_node (*LiszDataMixin* attribute), 352  
 root\_rec() (*\_Field* method), 164  
 round\_traditional() (in module *ae.base*), 26  
 run() (*TcpServer* method), 176  
 run\_app() (*ConsoleApp* method), 122  
 run\_app() (*MainAppBase* method), 250  
 running\_shaders (*ShadersMixin* attribute), 277
- ## S
- save\_app\_states() (*HelpAppBase* method), 271  
 save\_app\_states() (*MainAppBase* method), 250  
 scale\_x (*FlowSelector* attribute), 301  
 scale\_y (*FlowSelector* attribute), 301  
 SDF\_SH\_CLIENT\_PORT (in module *ae.sys\_core\_sh*), 175  
 SDF\_SH\_KERNEL\_PORT (in module *ae.sys\_core\_sh*), 175  
 SDF\_SH\_SERVER\_ADDRESS (in module *ae.sys\_core\_sh*), 175  
 SDF\_SH\_WEB\_PORT (in module *ae.sys\_core\_sh*), 175  
 SDI\_SH (in module *ae.sys\_core\_sh*), 174  
 search\_clients() (*ClientSearch* method), 191  
 search\_res() (*ResSearch* method), 192  
 SearcherRetType (in module *ae.paths*), 60  
 SearcherType (in module *ae.paths*), 60  
 select() (*DbBase* method), 203  
 selected (*Collector* attribute), 68  
 selected\_column\_names() (*DbBase* method), 205  
 selected\_item\_ink (*MainAppBase* attribute), 242  
 send\_client\_to\_sihot() (*ClientToSihot* method), 193  
 send\_email() (*Notifications* method), 43  
 send\_file() (*TransferServiceApp* method), 219  
 send\_message() (*TransferServiceApp* method), 219  
 send\_notification() (*Notifications* method), 43  
 send\_rec() (*ResSender* method), 196  
 send\_res\_to\_sihot() (*ResToSihot* method), 194  
 send\_telegram() (*Notifications* method), 44  
 send\_to\_server() (*\_SihotTcpClient* method), 175  
 send\_to\_server() (*SihotXmlBuilder* method), 180  
 send\_whatsapp() (*Notifications* method), 44  
 separator\_color (*FlowPopup* attribute), 299  
 separator\_color (*FlowSelector* attribute), 302  
 separator\_height (*FlowPopup* attribute), 298  
 separator\_height (*FlowSelector* attribute), 301  
 series\_file\_name() (in module *ae.paths*), 67  
 server\_app (in module *ae.transfer\_service*), 219  
 SERVER\_BIND (in module *ae.sideloading\_server*), 221  
 SERVER\_BIND (in module *ae.transfer\_service*), 214  
 server\_err\_msg() (*SihotXmlParser* method), 178  
 server\_error() (*SihotXmlParser* method), 178  
 server\_factory() (in module *ae.sideloading\_server*), 221  
 server\_instance (*SideloadingServerApp* attribute), 223  
 server\_instance (*TransferServiceApp* attribute), 217  
 SERVER\_PORT (in module *ae.sideloading\_server*), 221  
 SERVER\_PORT (in module *ae.transfer\_service*), 214  
 server\_thread (*SideloadingServerApp* attribute), 223  
 server\_thread (*TransferServiceApp* attribute), 217  
 server\_url() (*SideloadingServerApp* method), 224  
 service\_factory() (in module *ae.transfer\_service*), 215  
 set\_aspect() (*\_Field* method), 162  
 set\_aspects() (*\_Field* method), 163  
 set\_calculator() (*\_Field* method), 165  
 set\_clear\_val() (*\_Field* method), 166  
 set\_converter() (*\_Field* method), 167  
 set\_current\_index() (in module *ae.sys\_data*), 136  
 set\_current\_system\_index() (*Record* method), 152  
 set\_env() (*\_Field* method), 161  
 set\_env() (*Record* method), 152  
 set\_env() (*Records* method), 157  
 set\_filterer() (*\_Field* method), 167  
 set\_name() (*\_Field* method), 164  
 set\_node\_child() (*Record* method), 144  
 set\_node\_child() (*Records* method), 154  
 set\_opt() (*ConsoleApp* method), 121  
 set\_option() (*ConsoleApp* method), 121  
 set\_root\_idx() (*\_Field* method), 165  
 set\_root\_rec() (*\_Field* method), 164  
 set\_sql\_expression() (*\_Field* method), 168  
 set\_system\_root\_rec\_idx() (*\_Field* method), 161  
 set\_url\_part() (in module *ae.django\_utils*), 40  
 set\_val() (*\_Field* method), 159  
 set\_val() (*Record* method), 145  
 set\_val() (*Records* method), 154  
 set\_val() (*Value* method), 139  
 set\_val() (*Values* method), 141  
 set\_validator() (*\_Field* method), 169  
 set\_value() (*\_Field* method), 158  
 set\_value() (*OraDb* static method), 208  
 set\_value() (*Record* method), 144  
 set\_value() (*Values* method), 140  
 set\_var() (*ConsoleApp* method), 119

- set\_variable() (*ConsoleApp* method), 119
- setup\_app\_flow() (*OnboardingTour* method), 265
- setup\_app\_flow() (*TourBase* method), 263
- setup\_app\_flow() (*TourDropdownFromButton* method), 264
- setup\_app\_states() (*MainAppBase* method), 250
- setup\_explained\_widget() (*AnimatedTourMixin* method), 324
- setup\_explained\_widget() (*TourBase* method), 263
- setup\_layout() (*AnimatedOnboardingTour* method), 326
- setup\_layout() (*AnimatedTourMixin* method), 324
- setup\_layout() (*TourBase* method), 263
- setup\_layout() (*TourDropdownFromButton* method), 264
- setup\_page\_shaders\_and\_animations() (*AnimatedTourMixin* method), 324
- setup\_texts (*AnimatedTourMixin* attribute), 323
- setup\_texts() (*TourBase* method), 263
- SH\_CLIENT\_MAP (in module *ae.sys\_data\_sh*), 185
- SH\_DEF\_SEARCH\_FIELD (in module *ae.sys\_core\_sh*), 175
- sh\_exec() (in module *ae.console*), 113
- SH\_RES\_MAP (in module *ae.sys\_data\_sh*), 185
- shader\_parameter\_alias() (in module *ae.kivy\_gsl*), 276
- SHADER\_PARAMETER\_MATCHER (in module *ae.kivy\_gsl*), 276
- shader\_parameters() (in module *ae.kivy\_gsl*), 276
- ShaderIdType (in module *ae.kivy\_gsl*), 274
- ShadersMixin (class in *ae.kivy\_gsl*), 276
- show\_help() (*ConsoleApp* method), 122
- show\_message (*SideloadMainAppMixin* attribute), 343
- show\_message() (*MainAppBase* method), 250
- shrink\_node\_size() (*LiszDataMixin* method), 361
- ShSysConnector (class in *ae.sys\_core\_sh*), 183
- shutdown() (*AppBase* method), 85
- shutdown() (*SideloadServerApp* method), 224
- shutdown() (*TransferServiceApp* method), 219
- SHUTDOWN\_TIMEOUT (in module *ae.sideload\_server*), 221
- SHUTDOWN\_TIMEOUT (in module *ae.transfer\_service*), 214
- side\_spacing (*FlowPopup* attribute), 298
- sideloading\_active (*SideloadMainAppMixin* attribute), 343
- sideloading\_app (*SideloadMainAppMixin* attribute), 343
- sideloading\_file\_ext (*SideloadMainAppMixin* attribute), 343
- sideloading\_file\_mask (*SideloadMainAppMixin* attribute), 343
- SideloadKwargs (in module *ae.sideload\_server*), 221
- SideloadMainAppMixin (class in *ae.kivy\_sideload*), 343
- SideloadMenuPopup (class in *ae.kivy\_sideload*), 341
- SideloadMenuTour (class in *ae.kivy\_sideload*), 341
- SideloadServerApp (class in *ae.sideload\_server*), 222
- SihotXmlBuilder (class in *ae.sys\_core\_sh*), 179
- SihotXmlParser (class in *ae.sys\_core\_sh*), 177
- SimpleAutoTickerBehavior (class in *ae.kivy\_auto\_width*), 332
- SimpleHttpRequestHandler (class in *ae.sideload\_server*), 222
- simulate\_text\_input() (*AnimatedTourMixin* method), 325
- size (*ReliefCanvas* attribute), 284
- size (*ShadersMixin* attribute), 276
- size\_to\_attribute() (*AbsolutePosSizeBinder* method), 291
- size\_to\_callback() (*AbsolutePosSizeBinder* method), 291
- skip\_py\_cache\_files() (in module *ae.paths*), 67
- SKIPPED\_MODULES (in module *ae.base*), 19
- SlideSelectBehavior (class in *ae.kivy.behaviors*), 317
- snake\_to\_camel() (in module *ae.base*), 27
- SOCKET\_BUF\_LEN (in module *ae.sideload\_server*), 221
- SOCKET\_BUF\_LEN (in module *ae.transfer\_service*), 214
- sound\_files (*MainAppBase* attribute), 242
- sound\_volume (*MainAppBase* attribute), 242
- sql\_columns() (*Record* method), 152
- sql\_expression() (*\_Field* method), 168
- sql\_select() (*Record* method), 153
- srv() (*\_Field* method), 172
- SSL\_ENC\_PORT (in module *ae.notify*), 42
- SSL\_ENC\_SERVICE\_NAME (in module *ae.notify*), 42
- stack\_frames() (in module *ae.base*), 27
- stack\_var() (in module *ae.base*), 27
- stack\_vars() (in module *ae.base*), 27
- start() (*FldMapXmlParser* method), 189
- start() (*ResChange* method), 178
- start() (*SihotXmlParser* method), 177
- start() (*TourBase* method), 263
- start\_app\_service() (in module *ae.base*), 29
- start\_app\_tour() (*HelpAppBase* method), 271
- start\_server() (*SideloadServerApp* method), 224
- start\_server() (*TransferServiceApp* method), 219
- start\_tour() (*TourOverlay* method), 328
- startup\_beg (*AppBase* attribute), 82
- startup\_beg (*SubApp* attribute), 86
- startup\_end (*AppBase* attribute), 82
- state (*TouchableBehavior* attribute), 319
- stem (*CachedFile* attribute), 52



stem (*RegisteredFile* attribute), 51  
 stop() (*TourBase* method), 263  
 stop\_app() (*MainAppBase* method), 250  
 stop\_server() (*SideloadServerApp* method), 224  
 stop\_server() (*TransferServiceApp* method), 219  
 stop\_shader() (*ShadersMixin* method), 279  
 stop\_tour() (*TourOverlay* method), 328  
 string\_to\_records() (*\_Field* method), 171  
 string\_to\_records() (in module *ae.sys\_data*), 136  
 style\_rgba() (in module *ae.enaml\_app.functions*), 368  
 sub\_item\_ids() (*LiszDataMixin* method), 362  
 SubApp (class in *ae.core*), 86  
 submit\_to (*FileChooserPopup* attribute), 336  
 suffix\_failed (*Collector* attribute), 68  
 suppress\_stdout (*AppBase* attribute), 82  
 switch\_lang() (*\_GetTextBinder* method), 287  
 switch\_next\_animations (*AnimatedOnboardingTour* attribute), 326  
 switch\_next\_animations (*AnimatedTourMixin* attribute), 324  
 SXML\_DEF\_ENCODING (in module *ae.sys\_core\_sh*), 175  
 sys\_env\_dict() (in module *ae.base*), 28  
 sys\_env\_id (*AppBase* attribute), 82  
 sys\_env\_text() (in module *ae.base*), 28  
 sys\_id (*SystemBase* attribute), 125  
 SYS\_SECTION\_NAME (in module *ae.sys\_core*), 124  
 system (*SystemConnectorBase* attribute), 126  
 system\_record\_val() (*\_Field* method), 172  
 SystemBase (class in *ae.sys\_core*), 124  
 SystemConnectorBase (class in *ae.sys\_core*), 125

## T

tap\_animation() (*AnimatedTourMixin* method), 325  
 tap\_flow\_id (*FlowButton* attribute), 293  
 tap\_flow\_id (*FlowToggler* attribute), 304  
 tap\_kwargs (*FlowButton* attribute), 293  
 tap\_kwargs (*FlowToggler* attribute), 304  
 tap\_thru (*Tooltip* attribute), 304  
 targeted\_widget (*Tooltip* attribute), 305  
 TCP\_CONNECTION\_BROKEN\_MSG (in module *ae.sys\_core\_sh*), 175  
 TcpServer (class in *ae.sys\_core\_sh*), 176  
 teardown\_app\_flow() (*AnimatedTourMixin* method), 325  
 teardown\_app\_flow() (*OnboardingTour* method), 265  
 teardown\_app\_flow() (*TourBase* method), 263  
 teardown\_shaders\_and\_animations() (*AnimatedOnboardingTour* method), 326  
 teardown\_shaders\_and\_animations() (*AnimatedTourMixin* method), 325  
 TELEGRAM\_MESSAGE\_MAX\_LEN (in module *ae.notify*), 42  
 template\_idx\_path() (in module *ae.sys\_data*), 137  
 TEMPLATES\_FOLDER (in module *ae.base*), 19  
 TESTS\_FOLDER (in module *ae.base*), 19

text (*SimpleAutoTickerBehavior* attribute), 332  
 text\_size\_guess() (*KivyMainApp* method), 312  
 texture\_size (*AutoFontSizeBehavior* attribute), 329  
 texture\_size (*SimpleAutoTickerBehavior* attribute), 332  
 texture\_update (*AutoFontSizeBehavior* attribute), 329  
 texture\_update (*SimpleAutoTickerBehavior* attribute), 332  
 THEME\_DARK\_BACKGROUND\_COLOR (in module *ae.gui\_app*), 234  
 THEME\_DARK\_FONT\_COLOR (in module *ae.gui\_app*), 234  
 THEME\_LIGHT\_BACKGROUND\_COLOR (in module *ae.gui\_app*), 234  
 THEME\_LIGHT\_FONT\_COLOR (in module *ae.gui\_app*), 234  
 thread\_lock\_init() (*DbBase* static method), 205  
 ThreadedTCPRequestHandler (class in *ae.transfer\_service*), 216  
 tip\_text (*Tooltip* attribute), 304  
 title (*FlowPopup* attribute), 298  
 title (*MessageShowPopup* attribute), 304  
 TLS\_ENC\_PORT (in module *ae.notify*), 42  
 TLS\_ENC\_SERVICE\_NAME (in module *ae.notify*), 42  
 tn (*SihotXmlBuilder* attribute), 179  
 to\_ascii() (in module *ae.base*), 28  
 to\_dict() (*Record* method), 153  
 to\_widget (*SlideSelectBehavior* attribute), 318  
 to\_window (*ShadersMixin* attribute), 277  
 toggle\_item\_sel() (*LiszDataMixin* method), 362  
 Tooltip (class in *ae.kivy.widgets*), 304  
 top\_popup (*TourBase* attribute), 263  
 touch\_pos\_is\_inside() (*FlowSelector* method), 303  
 touch\_pos\_is\_inside() (*ModalBehavior* method), 317  
 TOUCH\_VIBRATE\_PATTERN (in module *ae.kivy.behaviors*), 314  
 TouchableBehavior (class in *ae.kivy.behaviors*), 319  
 TOUR\_EXIT\_DELAY\_DEF (in module *ae.gui\_help*), 257  
 tour\_help\_translation() (in module *ae.gui\_help*), 260  
 tour\_id\_class() (in module *ae.gui\_help*), 260  
 tour\_instance (*TourOverlay* attribute), 327  
 tour\_layout (*FrameworkApp* attribute), 308  
 tour\_layout (*HelpAppBase* attribute), 268  
 tour\_overlay\_class (*HelpAppBase* attribute), 268  
 tour\_overlay\_class (*KivyMainApp* attribute), 309  
 TOUR\_PAGE\_HELP\_ID\_PREFIX (in module *ae.gui\_help*), 257  
 TOUR\_START\_DELAY\_DEF (in module *ae.gui\_help*), 257  
 tour\_start\_pos (*Tooltip* attribute), 304  
 tour\_start\_size (*Tooltip* attribute), 305  
 TourBase (class in *ae.gui\_help*), 261  
 TourDropdownFromButton (class in *ae.gui\_help*), 264  
 TourOverlay (class in *ae.kivy.tours*), 326

- TRANSFER\_KWARGS\_DATE\_TIME\_NAME\_PARTS (in module *ae.transfer\_service*), 214
- transfer\_kwargs\_error() (in module *ae.transfer\_service*), 216
- transfer\_kwargs\_from\_literal() (in module *ae.transfer\_service*), 216
- TRANSFER\_KWARGS\_LINE\_END\_BYTE (in module *ae.transfer\_service*), 214
- TRANSFER\_KWARGS\_LINE\_END\_CHAR (in module *ae.transfer\_service*), 214
- transfer\_kwargs\_literal() (in module *ae.transfer\_service*), 216
- transfer\_kwargs\_update() (in module *ae.transfer\_service*), 216
- TransferKwargs (in module *ae.transfer\_service*), 214
- TransferServiceApp (class in *ae.transfer\_service*), 217
- translation() (in module *ae.i18n*), 96
- translation\_short\_help\_id() (in module *ae.gui\_help*), 261
- TRANSLATIONS\_PATHS (in module *ae.i18n*), 93
- try\_call() (in module *ae.dynamicod*), 89
- try\_eval() (in module *ae.dynamicod*), 90
- try\_exec() (in module *ae.dynamicod*), 90
- type\_mismatching\_with() (Literal method), 101
- ## U
- unbind (*SimpleAutoTickerBehavior* attribute), 332
- unbind() (*AbsolutePosSizeBinder* method), 291
- unbind\_uid (*ModalBehavior* attribute), 316
- unbind\_uid (*ShadersMixin* attribute), 277
- unfocus\_flow\_id (*FlowInput* attribute), 295
- unselected\_item\_ink (*MainAppBase* attribute), 242
- UNSET (in module *ae.base*), 20
- UnsetType (class in *ae.base*), 20
- update() (*DbBase* method), 203
- update() (*Record* method), 153
- update\_handler\_progress() (in module *ae.sideload\_server*), 222
- update\_page\_ids() (*OnboardingTour* method), 265
- update\_page\_ids() (*TourBase* method), 264
- update\_shaders() (*ShadersMixin* method), 280
- update\_tap\_kwargs() (in module *ae.gui\_app*), 241
- upgraded\_config\_app\_state\_version() (*MainAppBase* method), 250
- upsert() (*DbBase* method), 204
- use\_current\_index() (in module *ae.sys\_data*), 136
- use\_rec\_default\_root\_rec\_idx() (in module *ae.sys\_data*), 137
- use\_rec\_default\_sys\_dir() (in module *ae.sys\_data*), 137
- UsedSystems (class in *ae.sys\_core*), 126
- user\_data\_path() (in module *ae.paths*), 67
- user\_docs\_path() (in module *ae.paths*), 68
- user\_id (*ConsoleApp* property), 121
- USER\_NAME\_MAX\_LEN (in module *ae.gui\_app*), 237
- user\_preference\_color\_selected() (*EnamlMainApp* method), 367
- user\_section() (*ConsoleApp* method), 122
- user\_specific\_cfg\_vars (*SideloadMainAppMixin* attribute), 343
- UserPreferencesTour (class in *ae.gui\_help*), 267
- ## V
- val() (*\_Field* method), 159
- val() (*Record* method), 145
- val() (*Value* method), 139
- val() (*Values* method), 141
- validate() (*\_Field* method), 169
- validator() (*\_Field* method), 168
- Value (class in *ae.sys\_data*), 137
- value (*Literal* property), 100
- value() (*\_Field* method), 158
- value() (*Record* method), 144
- value() (*Value* method), 139
- value() (*Values* method), 140
- value\_filter\_default() (in module *ae.deep*), 34
- VALUE\_TYPES (in module *ae.sys\_data*), 173
- Values (class in *ae.sys\_data*), 139
- ValueType (in module *ae.sys\_data*), 133
- verbose (*AppBase* property), 83
- verbose\_out() (*AppBase* method), 85
- vibration\_volume (*MainAppBase* attribute), 242
- vpo (*SideloadMainAppMixin* attribute), 343
- vpo() (*AppBase* method), 85
- ## W
- WHATSAPP\_MESSAGE\_MAX\_LEN (in module *ae.notify*), 42
- widget\_by\_app\_state\_name() (*MainAppBase* method), 251
- widget\_by\_attribute() (*MainAppBase* method), 250
- widget\_by\_flow\_id() (*MainAppBase* method), 251
- widget\_by\_page\_id() (*HelpAppBase* method), 272
- widget\_children() (*MainAppBase* method), 251
- widget\_page\_id() (in module *ae.gui\_help*), 261
- widget\_pos() (*KivyMainApp* static method), 312
- widget\_pos() (*MainAppBase* static method), 251
- widget\_size() (*MainAppBase* static method), 252
- widget\_tourable\_children\_page\_ids() (*HelpAppBase* method), 272
- widget\_visible() (*MainAppBase* static method), 252
- widgets\_enclosing\_rectangle() (*MainAppBase* method), 252
- WidgetValues (in module *ae.kivy.tours*), 323
- width (*AutoFontSizeBehavior* attribute), 329
- width (*ContainerChildrenAutoWidthBehavior* attribute), 331
- width (*SimpleAutoTickerBehavior* attribute), 333

---

`win_activated()` (*EnamlMainApp* method), 367  
`win_closed()` (*EnamlMainApp* method), 367  
`win_pos_size_change()` (*FrameworkApp* method),  
309  
`win_pos_size_change()` (*MainAppBase* method), 252  
`win_rectangle` (*MainAppBase* attribute), 242  
`win_resize_from_enaml()` (*EnamlMainApp* method),  
367  
`wipe_gds_errors()` (*ResToSihot* method), 194  
`wipe_warnings()` (*FldMapXmlBuilder* method), 192  
`write()` (*\_PrintingReplicator* method), 80  
`write_file()` (in module *ae.base*), 29  
`write_file_text()` (in module *ae.files*), 51

## X

`xml` (*SihotXmlBuilder* property), 181