Python Humble Utils Documentation

Release 3.1.0

Nikita P. Shupeyko

Jun 28, 2021

Contents

Contents:

CHAPTER 1

Python Humble Utils

Python utils for everyday use.

- Documentation.
- Please, open issues before sending emails to the maintainers: You will get a much faster response!

1.1 Feature Areas

- File operations.
- File/directory paths extraction.
- File/directory paths randomization.
- String case conversions.
- Python class convenience shortcuts.
- py.test fixtures and helpers.

1.2 Installation

```
$ pip install python-humble-utils
```

or install from sources:

```
$ python setup.py install
```

Refer to Installation for detailed instructions.

1.3 Usage

```
import os
from pathlib import Path

from python_humble_utils.filesystem import yield_file_paths
from python_humble_utils.strings import camel_or_pascal_case_to_snake_case


# ...

file_paths = yield_file_paths(
    dir_path=Path("dir") / "with" / "scripts",
    allowed_file_extensions=(".sh", ".bash"),
    recursively=True
)
assert set(file_paths) == set(("s1.sh", "s2.bash", "s3.bash"))

s = camel_or_pascal_case_to_snake_case("camelCasedString")
assert s == "camel_cased_string"

s = camel_or_pascal_case_to_snake_case("PascalCasedString")
assert s == "pascal_cased_string"

# ...
```

1.4 Contributing

Your contributions are very much welcome! Refer to Contributing for more details.

1.5 Code of Conduct

All those using `python-humble-utils`, including its codebase and project management ecosystem are expected to follow the Python Community Code of Conduct.

1.6 Acknowledgements

This package was initially scaffolded via Cookiecutter with audreyr/cookiecutter-pypackage template.

CHAPTER 2

python_humble_utils package

2.1 Subpackages

2.1.1 python_humble_utils.vendor package

Submodules

python_humble_utils.vendor.pytest module

python_humble_utils.vendor.pytest.**generate_tmp_file_path**(*tmpdir_factory*, *file_name_with_extension: str*, *tmp_dir_path: Optional[pathlib.Path] = None*) → pathlib.Path

Generate file path relative to a temporary directory.

Parameters

- **tmpdir_factory** – py.test's *tmpdir_factory* fixture.
- **file_name_with_extension** – file name with extension e.g. *file_name.ext*.
- **tmp_dir_path** – path to directory (relative to the temporary one created by *tmpdir_factory*) where the generated file path should reside. # noqa

Returns file path.

Module contents

2.2 Submodules

2.3 python_humble_utils.classes module

python_humble_utils.classes.**get_all_subclasses**(*cls: Type[CT_co], including_self: bool = False*) → Collection[Type[CT_co]]

Get all subclasses.

Parameters

- **cls** – class to lookup subclasses of.
- **including_self** – whether or not the the :param cls: itself is to be accounted for.

Returns

param cls subclasses.

2.4 python_humble_utils.filesystem module

python_humble_utils.filesystem.**create_or_update_file**(*file_path: str, file_content: str = '', file_content_encoding: str = 'utf-8'*) → None

Create or update file.

Parameters

- **file_path** – path to the file.
- **file_content** – file content.
- **file_content_encoding** – file content encoding e.g. *latin-1*.

python_humble_utils.filesystem.**generate_random_dir_path**(*root_dir_path: Optional[pathlib.Path] = None, subdir_count: int = 0, random_string_generator: Optional[Callable[[], str]] = None*) → pathlib.Path

Generate a random directory path.

Parameters

- **root_dir_path** – root dir path; by default, the current dir path is used
- **subdir_count** – a number of subdirectories to generate in the directory root.
- **random_string_generator** – random number generator; by default, the UUID4 hex is used

Returns directory root path.

python_humble_utils.filesystem.**read_file**(*file_path: str, as_single_line: bool = False*) → str

Read file content.

Parameters

- **file_path** – path to the file.
- **as_single_line** – whether or not the file is to be read as a single line.

Returns file content.

python_humble_utils.filesystem.**yield_file_paths**(*dir_path: pathlib.Path, allowed_file_extensions: Collection[str], recursively: bool = False*) → Iterable[pathlib.Path]

Yield file paths.

Parameters

- **dir_path** – path to the containing directory.
- **allowed_file_extensions** – file extensions to match against e.g. *['.abc', '.def']*.
- **recursively** – whether or not the directory is to be recursively traversed.

Returns file paths.

2.5 python_humble_utils.objects module

python_humble_utils.objects.**flatten**(*obj: Any, flatten_dicts_by_values: bool = True, coerce: Optional[Callable[[T], M]] = None*) → Iterable[M]

Flatten an arbitrarily complex object.

Parameters

- **obj** – an obj to flatten.
- **flatten_dicts_by_values** – if True, mapping will be flattened by values, otherwise by keys.
- **coerce** – a callable used to coerce items of the resulting iterable to

Returns a recursively-constructed iterable of the object's constituents.

python_humble_utils.objects.**get_all_instances**(*cls: Type[T]*) → Sequence[T]

Get all class instances.

2.6 python_humble_utils.strings module

python_humble_utils.strings.**camel_or_pascal_case_to_snake_case**(*s: str*) → str

Convert *camelCased* or *PascalCased* string to *snake_case*.

Based on https://stackoverflow.com/a/1176023/1557013.

Parameters **s** – string in *camelCase* or *PascalCase*.

Returns string in *snake_case*.

python_humble_utils.strings.**camel_or_pascal_case_to_space_delimited**(*s: str*) → str

Convert *camelCased* or *PascalCased* string to space-delimited.

Based on https://stackoverflow.com/a/9283563/1557013.

Parameters **s** – string in *camelCase* or *PascalCase*.

Returns space-delimited string.

2.7 Module contents

CHAPTER 3

Installation

Note: *The commands below are presumed to be run relative to the project root unless explicitly stated otherwise.* `./` *also refers to the project root.*

3.1 From PyPI

To install the latest release, run

```
$ pip install python-humble-utils
```

Or, install a specific version via

```
$ pip install python-humble-utils==<version>
```

If you don't have pip installed, this Python installation guide can guide you through the process.

3.2 From Sources

1. Obtain sources
   - cloning the repository:

     ```
     $ git clone git://github.com/webyneter/python-humble-utils
     ```

   - or, downloading a tarball:

     ```
     $ curl -OL https://github.com/webyneter/python-humble-utils/tarball/master
     ```

2. Install via:

```
$ python setup.py install
```

CHAPTER 4

Usage

```
import os
from pathlib import Path

from python_humble_utils.filesystem import yield_file_paths
from python_humble_utils.strings import camel_or_pascal_case_to_snake_case


# ...

file_paths = yield_file_paths(
    dir_path=Path("dir") / "with" / "scripts",
    allowed_file_extensions=(".sh", ".bash"),
    recursively=True
)
assert set(file_paths) == set(("s1.sh", "s2.bash", "s3.bash"))

s = camel_or_pascal_case_to_snake_case("camelCasedString")
assert s == "camel_cased_string"

s = camel_or_pascal_case_to_snake_case("PascalCasedString")
assert s == "pascal_cased_string"

# ...
```

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

Note: *The commands below are presumed to be run relative to the project root unless explicitly stated otherwise.* `./` *also refers to the project root.*

5.1 Ways You Can Help Us

5.1.1 Report Bugs

Create an issue corresponding to the bug you have found complying with the project's issue template.

5.1.2 Fix Bugs

Look through the GitHub issues: Anything tagged with `bug` and without an `Assignee` is open to whoever wants to implement it.

Make sure to submit clean, concise, well-tested pull requests only, so it is easier for contributors to review it; strive to deliver as short and atomic pull requests as possible as this will dramatically increase the likelihood of those being merged.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with `enhancement` and/or `help wanted` and/or without an `Assignee` is open to whoever wants to implement it.

5.1.4 Write Documentation

We could always use more documentation, whether as part of the official docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue.

If you are proposing a feature,

- explain in detail how it would work;
- keep the scope as narrow as possible, to make it easier to implement;
- remember that this is a volunteer-driven project, so contributions are welcome!

5.2 Workflow

See *Developing Locally* for detailed instructions on setting up local environment.

Oncer you are all set up,

1. create a branch:

```
$ git checkout -b <issue id>-<issue title>
```

2. make the contribution;
3. follow *Running tox with Multiple Python Distributions* to run tests comprehensively;
4. commit changes to the branch:

```
$ git add .
$ git commit -m "<detailed description of your changes>"
```

5. push the branch to GitHub:

```
$ git push origin <issue id>-<issue title>
```

6. submit a pull request via GitHub or any other git GUI tool you prefer.

5.3 Guidelines

Upon submission, make sure the PR meets these guidelines:

1. the PR does not decrease code coverage (unless there is a very specific reason to);
2. the docs (both programmatic and manual) are updated, if needed.

CHAPTER 6

Developing Locally

Note: *The commands below are presumed to be run relative to the project root unless explicitly stated otherwise.* `./` *also refers to the project root.*

6.1 Environment Setup

1. Fork us on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:<your username>/python-humble-utils.git
```

3. Create a virtualenv ; assuming you have virtualenvwrapper installed, this is how you do it:

```
$ mkvirtualenv python_humble_utils
$ cd <cloned project root>
$ setvirtualenvproject
```

4. Initialize environment:

```
$ python setup.py develop
```

6.2 Scenarios

6.2.1 Updating Requirements

Project requirements must be declared and pinned in `./requirements*.txt`.

To install/upgrade/uninstall dependencies into/in/from the environment:

```
$ make install
```

6.2.2 Running `tox` with Multiple Python Distributions

Running `tox` locally requires a number of Python distributions to be available, which is a challenge, to say the least. pyenv helps overcome this major obstacle.

1. Follow pyenv installation instructions to install `pyenv` system-wide.
2. Install all versions of Python the project is tested against by `tox` (see `./tox.ini`).
3. Run `tox`:

   ```
   $ make test-all
   ```

CHAPTER 7

Project Makefile

Note: *The commands below are presumed to be run relative to the project root unless explicitly stated otherwise.* `./` *also refers to the project root.*

To facilitate smooth development workflow, we provide a `Makefile` defining a number of convenience commands.

- `clean` running
    1. `clean-build` (build artifact removal);
    2. `clean-pyc` (compilation artifact removal);
    3. `clean-test` (test and coverage artifact removal).

  Specifically:

```
$ make clean
$ make clean-build
$ make clean-pyc
$ make clean-test
```

- `lint` checking codebase compliance with PEP8 via flake8:

```
$ make lint
```

- `test` running py.test:

```
$ make test
```

- `test-all` running tox:

```
$ make test-all
```

- `coverage` running coverage:

```
$ make coverage
```

- `docs` generating project docs via Sphinx:

```
$ make docs
```

- `servedocs` serving docs live via watchdog:

```
$ make servedocs
```

- `setup-release` packaging and releasing the project to PyPI:

```
$ make setup-release
```

- `setup-dist` builds source and wheel packages via setuptools:

```
$ make setup-dist
```

- `setup-install` installing the package to the current environment:

```
$ make setup-install
```

- `install` keeping local environment dependencies in sync with those defined in `./requirements*.txt`:

```
$ make install
```

CHAPTER 8

Credits

8.1 Development Lead

- Nikita P. Shupeyko <webyneter@gmail.com>

8.2 Contributors

None yet. Why not be the first?

CHAPTER 9

History

9.1 v3.0.1

v3.0.1.

- Sync usage guides.
- Update ‘HISTORY.rst.

9.2 v3.0.0

v3.0.0.

Breaking changes:

- **The following functions have been removed as part of the Remove redundant utilities/utilities making no sense effort:**
  - *extract_file_name_with_extension*
  - *extract_file_name_and_extension*
  - *extract_file_dir_path*
  - *parse_tuple_from_string*
  - *generate_hex_uuid_4*
  - *generate_random_file_name_with_extension*
  - *get_class_name*
  - *get_class_qualname*
- Package structure has been altered.
- Drop support for Python 3.5.

Other changes:

- Update the docs.
- Switch from Python 3.6+ typing.Collection back to typing.Sequence for backward compatibility with Python 3.5.
- Support Python 3.8.
- Switch to native Python paths.
- Support Python 3.7.
- Upgrade all dependencies to the latest versions to date.

9.3 v2.0.0

v2.0.0.

- Move tests to the root.
- Pick another naming scheme for tests.
- Convert NamedTuple's to classes and document them.
- Clean up docstrings.
- Stop indexing tests in docs.

9.4 v1.0.4

v1.0.4.

- Update HISTORY.

9.5 v1.0.3

v1.0.3.

- Fix deployment to the new PyPI.
- Mention python-humble-utils in "Open Source Projects using Hypothesis".
- Move bumpversion configuration from setup.cfg to .bumpversion.cfg.
- Enclose notes in Note blocks.
- Remove entrypoint section from setup.py.
- Add insert utility.

9.6 v1.0.2

v1.0.2.

- Add Code Climate badge to README.

9.7 v1.0.1

v1.0.1.

- Fix README not rendered properly on PyPI.

9.8 v1.0.0

v1.0.0.

- Bump package Development Status.
- Test package deployment locally.
- Fix relative paths notice.
- Add Gitter badge.
- Fill in HISTORY.

9.9 v0.5.0

v0.5.0.

- Document python_humble_utils package.
- Introduce local requirements.
- Stop using pip-tools.
- Point out that all paths in docs are relative to the project root.
- Prevent pip-tools from injecting indirect requirements.
- Target stable docs version only.
- Fix README not rendered on PyPI.
- Ensure codecov evaluates coverage against payload files only.

9.10 v0.4.0

v0.4.0.

- Support Python 3.6.

9.11 v0.3.0

v0.3.0.

- Setup ReadTheDocs.

9.12 v0.2.0

v0.2.0.

- First release on PyPI.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index