Squelette Documentation

Release 0.0.0

Louis Paternault

Feb 19, 2024

CONTENTS

This package is a template package for my new python projects.

CHAPTER

ONE

FEATURES

1.1 Binary

- There are several ways to call the binary.
  - The package contains a `__main__` module, so you can use `python -m squelette [ARGS]`.
  - There is an entry point in the setup file. That way, the binary is correctly installed, whatever the platform, and once installed, you can use `squelette [ARGS]`.
  - In the `bin` directory, not shipped when distributing the program, there is a `squelette` binary. In development environment, you can use `./bin/squelette [ARGS]`.
  - The package is translated using gettext.
- My usual `logging` configuration is set.

1.2 Tests

Several parts are tested; everything is configured in `tox.ini`.

- First of all, tox is used to aggregate all of the following tests. To perform all the tests (with different python versions), in the root project directory, run:

```
$ tox
```

- To run `unittest`, in the root directory, run:

```
$ python -m unittest
```

- Unittest call also run `doctest` present accross modules. Thus, you can freely add doctests anywhere in your program, and they will be performed along with unittests or tox. This part is performed in `squelette/test.py`.
- It can be a bit cumbersome to add tests for a program or library acting on files. File `test/test_add/__init__.py` scans module directories for couples of `.in` and `.out`, and creates one `unittest.TestCase` per couple. Using one test case per couple, when a test fails, it is easier to track which couple of files caused the problem than if tests where performed in a loop in a single test case.

  Note that for simple cases (like this one), this could also have been achieved by using `unittest.TestCase.subTest()`.
- Tox also checks that *Code formatting* runs without any error.
- Tox also checks that *Documentation* is compiled without any warning or error.

1.3 Release and Distribution

- Hatch is used with `pyproject.toml` file, with my usual configuration. Among others:
  - The `README.md` file content is used as the long description (don't repeat yourself).
  - Entry points are used, so that binaries are automatically installed the right way, disregarding specific platform.
- My usual release process is described in file `release.rst`.
- Stdeb is used to build quick and dirty debian packages. It is configured in `stdeb.cfg`.
- To access packages data, no matter what release are, they are placed in package sub directories, and accessed using importlib.resources. An example is available in file `squelette/__main__.py`.
- A build hook is used to automatically compile gettext message catalogs while building the python package: `hatch_build_babel.py`.

1.4 Documentation

- My usual `README.rst` is provided, with links, basic documentation, and badges.
- The documentation is built using sphinx.
- Sphinx compilation is tested by *tox*.
- Readthedocs is configured using `a YAML file`.

1.5 Code formatting

- The code is Pylint compliant. Configuration is done in the `pylintrc` file.
- The code is formatted using black. If pre-commit is properly installed and configured (i.e. `pip install pre-commit` to install, and `pre-commit install` to set up pre-commit hooks), black is automatically run each time you `git commit`.
- *Tox* tests that pylint and black raises no error.

CHAPTER

TWO

DOWNLOAD AND INSTALL

See the main project page for instructions, and changelog.

CHAPTER

THREE

USAGE

Here are the command line options for *squelette*.

Python package template

```
usage: squelette [-h] [--version] [NUMBER ...]
```

3.1 Positional Arguments

NUMBER List of numbers to add

3.2 Named Arguments

--version Show version

As a template, this program sums numbers given in argument.