Shadow Project Documentation

Release 0.1.1

Fabien Mathieu

Jan 13, 2020

Contents:

CHAPTER 1

Shadow Project

A dummy project to evaluate the Cookie Cutter workflow

- Free software: GNU General Public License v3
- Documentation: https://shadow-project.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Shadow Project, run this command in your terminal:

```
$ pip install shadow_project
```

This is the preferred method to install Shadow Project, as it will always install the most recent stable release.

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

2.2 From sources

The sources for Shadow Project can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/balouf/shadow_project
```

Or download the tarball:

```
$ curl  -OL https://github.com/balouf/shadow_project/tarball/master
```

Once you have a copy of the source, you can install it with:

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

CHAPTER 3

Usage

To use Shadow Project in a project:

```
import shadow_project
```

CHAPTER 4

Reference

4.1 A Nice Section

class shadow_project.**MyClass1**(*a: float*, *b: float*)

A whatever-you-are-doing.

Parameters

- **a** – the *a* of the system. Must be nonnegative.
- **b** – the *b* of the system.

Variables **my_string** (*str*) – a nice string.

Raises **ValueError** – if a is negative.

Note: document the __init__() method in the docstring of the class itself, because the docstring of the __init__() method does not appear in the documentation.

- Refer to a class this way: *MyClass2*.
- Refer to a method this way: *addition()*.
- Refer to a method in another class: *MyClass2.addition()*.
- Refer to an parameter or variable this way: a.

```
>>> my_object = MyClass1(a=5, b=3)
```

A_NICE_CONSTANT = 42

This is a nice constant.

A_VERY_NICE_CONSTANT = 51

addition() → float

Add a and b.

Returns a + b.

```
>>> my_object = MyClass1(a=5, b=3)
>>> my_object.addition()
8
```

divide_a_by_c_and_add_d(*c: float*, *d: float*) → float
Divide a by something and add something else.

Parameters

- **c** – a non-zero number. If you want to say many things about this parameter, you must indent the following lines, like this.
- **d** – a beautiful number.

Returns a / c + d.

Raises **ZeroDivisionError** – if c = 0.

This function gives an example of Sphinx documentation with typical features.

```
>>> my_object = MyClass1(a=5, b=3)
>>> my_object.divide_a_by_c_and_add_d(c=2, d=10)
12.5
```

4.2 Another Nice Section

class shadow_project.**MyClass2**(*a: float*, *b: float*)
A whatever-you-are-doing.

Parameters

- **a** – the *a* of the system.
- **b** – the *b* of the system.

```
>>> my_object = MyClass2(a = 5, b = 3)
```

addition() → float
Add a and b.

Returns a + b.

```
>>> my_object = MyClass2(a=5, b=3)
>>> my_object.addition()
8
```

class shadow_project.**MyClass3**(*a: float*, *b: float*)
A whatever-you-are-doing.

Parameters

- **a** – the *a* of the system.
- **b** – the *b* of the system.

```
>>> my_object = MyClass3(a = 5, b = 3)
```

addition() → float
Add a and b

Returns `a + b`.

```
>>> my_object = MyClass3(a=5, b=3)
>>> my_object.addition()
8
```

CHAPTER 5

Contributing

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

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/balouf/shadow_project/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

Shadow Project could always use more documentation, whether as part of the official Shadow Project 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 at https://github.com/balouf/shadow_project/issues.

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, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *shadow_project* for local development.

1. Fork the *shadow_project* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/shadow_project.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv shadow_project
$ cd shadow_project/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 shadow_project tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/balouf/shadow_project/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_shadow_project
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Fabien Mathieu <fabien.mathieu@nokia-bell-labs.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.1.1 (2018-03-05)

- Second release on PyPI.

7.2 0.1.0 (2018-03-05)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index

A

D

M