MatlabTest Documentation

Release 0.1.0

Chance Tarver

Jun 13, 2018

Contents:

1 MatlabTest **1**
- 1.1 Features . . . 1
- 1.2 Credits . . . 1

2 Installation **3**
- 2.1 Stable release . . . 3
- 2.2 From sources . . . 3

3 Usage **5**

4 Contributing **7**
- 4.1 Types of Contributions . . . 7
- 4.2 Get Started! . . . 8
- 4.3 Pull Request Guidelines . . . 9
- 4.4 Tips . . . 9
- 4.5 Deploying . . . 9

5 History **11**
- 5.1 0.1.0 (2018-06-12) . . . 11

6 Test Data **13**
- 6.1 My Handle Class . . . 13

7 Indices and tables **15**

MATLAB Module Index **17**

CHAPTER 1

MatlabTest

This is just a test to see if I could use the python cookiecutter and readthedocs to document matlab code. By importing an extension, I was able to. This is only a basic example.

- Free software: MIT license
- Documentation: https://matlabtest.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 MatlabTest, run this command in your terminal:

```
$ pip install matlabtest
```

This is the preferred method to install MatlabTest, 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 MatlabTest can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/ctarver/matlabtest
```

Or download the tarball:

```
$ curl  -OL https://github.com/ctarver/matlabtest/tarball/master
```

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

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

CHAPTER 3

Usage

To use MatlabTest in a project:

```
import matlabtest
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/ctarver/matlabtest/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/ctarver/matlabtest/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 :)

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/matlabtest.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 matlabtest
$ cd matlabtest/
$ 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 matlabtest 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.

4.3 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/ctarver/matlabtest/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_matlabtest
```

4.5 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 5

History

5.1 0.1.0 (2018-06-12)

- First release on PyPI.

CHAPTER 6

Test Data

This is the test data module.

test_data is a really cool module.

6.1 My Handle Class

This is the handle class definition.

class test_data.**MyHandleClass**(*x*)

Bases: my.super.Class, handle

a handle class

Parameters **x** – a variable

get_x()

how is this displayed?

static my_static_function(*z*)

A static function in *MyHandleClass*.

Parameters **z** – input z

Returns w

x = None

a property

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

MATLAB Module Index

t

test_data, 13

Index

G

get_x() (test_data.MyHandleClass method), 13

M

my_static_function() (test_data.MyHandleClass static method), 13
MyHandleClass (class in test_data), 13

T

test_data (module), 13

X

x (test_data.MyHandleClass attribute), 13