beprof Documentation

Release 0.1.3+rev1

Author

Nov 17, 2018

Contents

Contents:

CHAPTER 1

beprof

CHAPTER 2

Overview

docs	
tests	
package	

Overv.io issue board: https://overv.io/DataMedSci/beprof/

Beam Profile Analysing Tools

Library provides methods to work with Beam Profiles which are sets of points (2-D with optional metadata) sorted by one of coordinates. cd beprof is based on nparray class from numpy, and it provides numerous tools for different computations and data analysis.

2.1 Installation

Current version available on testing PyPi server, although once a stable version is ready it will be pushed to official PyPi repo.

For now, installation can be done from this GIT repository, using:

```
pip install setuptools versioneer
pip install git+https://github.com/DataMedSci/beprof.git
```

To unistall, simply use:

```
pip uninstall beprof
```

2.2 Documentation

https://beprof.readthedocs.io/

2.2.1 Features

Once you install beprof, you should be able to import is as a python module Using ipython the code would be i.e.:

```
import beprof
from beprof import curve  #imports curve module
from beprof import profile  #imports profile module
```

Once you import necessary modules, you can use them to work with i.e. profiles:

```
from beprof import profile
dir(profile)
p = profile.Profile([[0, 1], [1, -1], [2, 3], [4, 0]])
print(p)
```

You can also use another modules as numpy or matplotlib to work with beprof:

```
#assuming you already defined p as above
import numpy as np
import matplotlib.pyplot as plt
foo = np.asarray(p)
print(foo.shape())
plt.plot(foo[:,0], foo[:,1])
plt.show()
```

Note that beprof is a library and end-users shouldn't "run it". It is also strongly discouraged to use *git clone* to download code. Git is only for developers, end-users should use pip installation. If you are interested in development - have a look at CONTRIBUTING section.

2.2.2 Credits

This package was created with Cookiecutter and the grzanka/cookiecutter-pip-docker-versioneer project template.

CHAPTER 3

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:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/DataMedSci/beprof/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.

3.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

3.1.4 Write Documentation

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

3.1.5 Submit Feedback

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

3.2 Get Started!

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

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

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

3. If you are using Linux based distributions - installation of appropriate python-dev package may be required.
4. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv beprof
$ pip install versioneer
$ cd beprof/
$ python setup.py develop
```

5. Create a branch for local development:

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

Now you can make your changes locally.

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

```
$ flake8 beprof tests
$ python setup.py test
$ tox
```

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

7. 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
```

8. Submit a pull request through the GitHub website.

3.3 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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/DataMedSci/beprof/pull_requests and make sure that the tests pass for all supported Python versions.

3.4 Tips

To run a subset of tests:

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

Some classes have main methods which can be used for testing. To run them either use PyCharm right click on __main__ or something like this:

```
$ PYTHONPATH=. python beprof/curve.py
```

Note - it only applies when you are trying to run the code after downloading it to local disk e.g. via git.

Another example may be running code this way:

```
$ python -m beprof.curve
```

from source directory.

CHAPTER 4

Credits

4.1 Development Lead

- Leszek Grzanka <grzanka@agh.edu.pl>

4.2 Contributors

- Agnieszka Rudnicka

CHAPTER 5

Indices and tables

- genindex
- modindex
- search