MOPAC step Documentation

Release 0.1.0

Paul Saxe

Apr 15, 2020

CONTENTS

Contents:

CHAPTER

ONE

MOPAC STEP

A step for a SEAMM flowchart for handling the semiempirical quantum code MOPAC

- Free software: BSD license
- Documentation: https://mopac-step.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the molssi-seamm/cookiecutter-seamm-plugin project template.

Developed by the Molecular Sciences Software Institute (MolSSI), which receives funding from the National Science Foundation under award ACI-1547580

CHAPTER

TWO

INSTALLATION

2.1 MOPAC

In order to run MOPAC, you will need to have it installed on the appropriate computer or computers. Please see the MOPAC website for further details. If you are at Viginia Tech, please contact Paul Saxe (psaxe@vt.edu) for access to the University-wide license. At other institutions it is possible that there is already a license that you can use. Otherwise, for academic users MOPAC is free, but requires a license which you can get from the MOPAC website.

2.2 Stable release

To install MOPAC step, run this command in your terminal:

```
$ pip install mopac_step
```

This is the preferred method to install MOPAC step, 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.

2.3 From sources

The sources for MOPAC step can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/paulsaxe/mopac_step
```

Or download the tarball:

```
$ curl  -OL https://github.com/paulsaxe/mopac_step/tarball/master
```

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

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

CHAPTER

THREE

USAGE

To use MOPAC step in a project:

```
import mopac_step
```

CHAPTER

FOUR

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/paulsaxe/mopac_step/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

MOPAC step could always use more documentation, whether as part of the official MOPAC step 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/paulsaxe/mopac_step/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 *mopac_step* for local development.

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

```
$ git clone git@github.com:your_name_here/mopac_step.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 mopac_step
$ cd mopac_step/
$ 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 mopac_step 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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/paulsaxe/mopac_step/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_mopac_step
```

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- Paul Saxe <psaxe@molssi.org>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

HISTORY

6.1 0.1.0 (2018-01-20)

- First release on PyPI.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search