pint Documentation

Release 0.5.1+186.g6ae4f8a.dirty

PINT Developers

Mar 12, 2017

Contents

Contents:

CHAPTER 1

PINT

PINT is not TEMPO3

PINT is a project to develop a new pulsar timing solution based on python and modern libraries. It is in the very early stages of development, but it can already produce residuals from most "normal" timing models that agree with Tempo and Tempo2 to within ~10 nanoseconds.

The primary reasons we are developing PINT are: - To have a robust system to check high-precision timing results that is completely independent of TEMPO and TEMPO2 - To make a system that is easy to extend and modify due to a good design and the use of a modern programming language, techniques, and libraries

Most users will want to install PINT as follows:

```
python setup.py install --user
```

To build PINT so that it can be run from the local directory (without installing), such as for developers or to run the tests, use:

```
python setup.py build_ext --inplace
```

For more information on installing and developing PINT, see the Wiki and the documentation in the doc subdirectory in the distribution. Also, follow along with the examples in the examples subdirectory, particularly the IPython notebook `examples/PINT_walkthrough.ipynb`.

CHAPTER 2

Installation

Stable release

To install pint, run this command in your terminal:

```
$ pip install pint
```

This is the preferred method to install pint, 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.

From sources

The sources for pint can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/nanograv/pint
```

Or download the tarball:

```
$ curl  -OL https://github.com/nanograv/pint/tarball/master
```

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

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

CHAPTER 3

Usage

To use pint in a project:

```
import pint
```

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/nanograv/pint/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

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

Get Started!

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

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

```
$ git clone git@github.com:your_name_here/pint.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 pint
$ cd pint/
$ pip install -r requirements_dev.txt
$ pip install -r requirements.txt
$ 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 the tests. Also check that any new docs are formatted correctly:

```
$ make test
$ make docs
```

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.

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/nanograv/pint/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To track and checkout another user's branch:

```
$ git remote add other-user-username https://github.com/other-user-username/pint.git
$ git fetch other-user-username
$ git checkout --track -b branch-name other-user-username/branch-name
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search