Python Lunar Documentation

Release 0.6.0

Michael Reuter

Apr 08, 2020

Contents

Information for completing the Astronomical League's Lunar and Lunar II observing programs.

CHAPTER 1

Features

- Offer moon information based on location and date/time.

1.1 Contents:

1.1.1 Installation

At the command line either via easy_install or pip:

```
$ easy_install pylunar
$ pip install pylunar
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv pylunar
$ pip install pylunar
```

1.1.2 Usage

To use Python Lunar in a project:

```
>>> import pylunar
```

Location: Boston, MA, USA

```
>>> mi = pylunar.MoonInfo((42, 21, 30), (-71, 3, 35))
```

Local Time: July 18, 2016 at 21:45

API requires UTC, so add 4 hours since on Daylight Savings (changes day)

```
>>> mi.update((2016, 7, 19, 1, 45, 0))
>>> mi.age()
14.613899383497483
>>> mi.fractional_phase()
0.9900636126401263
>>> mi.phase_name()
'WAXING_GIBBOUS'
```

This package also contains the `LunarFeatureContainer` class which holds features on the moon for the Astronomical League's Lunar Club and Lunar II observing programs. To create a container for the Lunar Club program, do.

```
>>> lc = pylunar.LunarFeatureContainer("Lunar")
>>> lc.load()
>>> len(lc)
90
```

There are 90 features available to this observing program. The container allows one to filter those features based on the position of the lunar terminator with respect to a given feature. The `LunarFeatureContainer.load()` method can be passed a `MoonInfo` instance to perform that filtering.

```
>>> lc.load(mi)
>>> len(lc)
12
```

A container for Lunar II can be created by passing the *LunarII* string to the constructor of `LunarFeatureContainer`.

Feature instances (`LunarFeature`) can be obtained from the container in usual manner.

```
>>> for feature in lc:
...     print(feature)
Name = Grimaldi
Lat/Long = (-5.38, -68.36)
Type = Crater
Delta Lat/Long = (5.72, 5.74)
Name = Mare Crisium
Lat/Long = (16.18, 59.10)
Type = Mare
Delta Lat/Long = (14.85, 19.02)
...
```

1.1.3 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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/mareuter/pylunar/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" is open to whoever wants to implement it.

Implement Features

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

Write Documentation

Python Lunar could always use more documentation, whether as part of the official Python Lunar 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/mareuter/pylunar/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 *pylunar* for local development.

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

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

3. Create a branch for local development:

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

Now you can make your changes locally.

4. When you're done making changes, check that your changes pass style and unit tests, including testing other Python versions with tox:

```
$ tox
```

To get tox, just pip install it.

5. 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
```

6. 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 2.7, and 3.5, and for PyPy. Check https://travis-ci.org/mareuter/pylunar under pull requests for active pull requests or run the `tox` command and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test test/test_pylunar.py
```

1.1.4 Credits

Development Lead

- Michael Reuter <mareuternh@gmail.com>

Contributors

None yet. Why not be the first?

1.1.5 History

0.6.0 (2020-04-07)

- Drop Python 2 and 3.4 and 3.5 support
- Add Python 3.7 and 3.8 support
- Change to 3-clause BSD license
- Switch to pytest from tox
- Add Github workflows for build and package upload
- Remove Travis

0.5.1 (2018-04-20)

- Add changelog updates

0.5.0 (2018-04-20)

- LunarFeature additions
  - Latitude and Longitude ranges
  - Feature angle
- MoonInfo additions
  - Libration phase angle
  - Libration visibility check
  - Updated is_visible to use libration visibility check

0.4.1 (2017-05-30)

- Corrected moon state after rise/set function call
- Made landing sites always visible once visible

0.4.0 (2017-05-28)

- Added landing sites to feature database
- Expanding LunarFeature content
- MoonInfo object additions
  - right ascension and declination
  - solar elongation
  - earth distance
  - rise, transit and set times
  - angular size
  - magnitude
  - sub-solar latitude

0.3.1 (2017-05-15)

- Ensure feature DB included in package

0.3.0 (2017-05-15)

- MoonInfo object additions
  - time of day
  - is feature visible

- LunarFeatureContainer object changes
  - Made constructor club related
  - Load call can check if feature is visible using MoonInfo instance

0.2.1 (2017-04-20)

- Changed mechanism to determine phase name

0.2.0 (2017-04-16)

- MoonInfo object additions
  - phase name
  - time from new moon
  - time to new moon
  - time to full moon

0.1.0 (2016-07-18)

- MoonInfo object that provides basic lunar information
  - age
  - altitude
  - azimuth
  - colongitude
  - fractional phase
  - libration latitude
  - libration longitude
  - next four lunar phases

1.2 Feedback

If you have any suggestions or questions about **Python Lunar** feel free to email me at mareuternh@gmail.com.

If you encounter any errors or problems with **Python Lunar**, please let me know! Open an Issue at the GitHub http://github.com/mareuter/pylunar main repository.