months Documentation

Release 1.0.0

Kyle Stark

Oct 18, 2019

Contents

Contents:

CHAPTER 1

months

Python library for representing specific months

- Free software: MIT license
- Documentation: https://months.readthedocs.org.

1.1 Features

- Represent specific months along with their years
- Convert to and from native datetime and date objects
- Convenient math operations for adding / subtracting month intervals
- Convenient operations for displaying months

CHAPTER 2

Installation

At the command line:

```
$ easy_install months
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv months
$ pip install months
```

CHAPTER 3

Usage

To use months in a project:

```
import months

month = months.Month(2015, 4)
print(month.full_display)            # April 2015
print(month.month_abbr)              # Apr
print(month + 9)                     # 2016-01
print(month.start_date)              # datetime.date(2015, 4, 1)
print(month.n_days)                  # 30
print(month.dates[-1])               # datetime.date(2015, 4, 30)
print(month.nth(-1))                 # datetime.date(2015, 4, 30)
print(month.to(2015, 5))             # [Month(2015, 4), Month(2015, 5)]
print(month.distance(month + 3))     # 3
print(month.gregorian_month_number)  # 24172
print(int(month))                    # 201504
print(float(month))                  # 201504.0
```

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/kstark/months/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” is open to whoever wants to implement it.

4.1.3 Implement Features

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

4.1.4 Write Documentation

months could always use more documentation, whether as part of the official months 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/kstark/months/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 *months* for local development.

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

```
$ git clone git@github.com:your_name_here/months.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 months
$ cd months/
$ 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 months tests
$ python setup.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, and 3.4, and for PyPy. Check https://travis-ci.org/kstark/months/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_months
```

CHAPTER 5

Credits

5.1 Development Lead

- Kyle Stark <kstarked@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

CHAPTER 7

2.0.0 (2019-10-18)

- Thanks to nolanbconaway for their contributions!
- **New methods to handle relations between months:**
  - `month.to(other)` for generating intervals of months.
  - `month.distance(other)` for computing distance between months.
- **New methods for month date info**
  - `month.n_days` to return the number of days in the month.
  - `month.dates` to return a list of all days in the month.
  - `month.nth(day)` to return a specific day in the month.
- **`__int__` and `__float__` methods added.**
  - Also a `month.gregorian_month_number` method to compute number of months since year 0.

CHAPTER 8

1.1.0 (2019-10-18)

- Support up to Python 3.8, drop explicit support for 2.6/3.2/3.3
- Raise TypeError on invalid addition/subtraction instead of ValueError

CHAPTER 9

1.0.0 (2015-04-13)

- Documentation added
- 2.6 support added
- Tests for bad math added

CHAPTER 10

0.1.0 (2015-04-13)

- First release on PyPI.

CHAPTER 11

API Documentation

class months.**Month**(*year*, *month*)

Represent a specific month of a year.

Provides various utilities for generating, manipulating, and displaying months.

abbr_display

Return the abbreviated calendar name of the month and the year.

```
>>> Month(2015, 4).full_display
'Apr 2015'
```

dates

Return a tuple of all days in the month.

```
>>> Month(2018, 1).dates[:2]
(datetime.date(2018, 1, 1), datetime.date(2018, 1, 2))
```

distance(*self*, **args*, ***kwargs*)

Return the number of months distance between months.

This will always be a positive number. Accepts two-element lists/tuples or Month objects.

```
>>> Month(2018, 1).distance(Month(2018, 12))
11
>>> Month(2018, 5).distance(2018, 1)
4
```

Parameters

other [Month, date, datetime, tuple] A Month-like object.

Returns

n_months [int] Integer number of months distance.

end_date
Return a datetime.date object for the last day of the month.

classmethod from_date(*cls*, *date*)
Return a Month instance from given a date or datetime object.

Parameters

date [date or datetime] A date/datetime object as implemented via the standard lib module.

Returns

month [Month] The month object for that date.

classmethod from_today(*cls*)
Return a Month instance from today's date (local time).

classmethod from_utc_today(*cls*)
Return a Month instance from today's date (UTC time).

full_display
Return the calendar name of the month along with the year.

```
>>> Month(2015, 4).full_display
'April 2015'
```

gregorian_month_number
Return the number of months since the start of Gregorian year 1.

Year 0 and month 0 are invalid. So the first month of year 1 is 1, and the first month of year -1 is -1.

```
>>> Month(1, 1).gregorian_month_number
1
>>> Month(2, 2).gregorian_month_number
14
>>> Month(-1, 2).gregorian_month_number
-2
```

month_abbr
Return the abbreviated calendar name of the month.

```
>>> Month(2015, 4).month_abbr
'Apr'
```

month_name
Return the calendar name of the month.

```
>>> Month(2015, 4).month_name
'April'
```

n_days
Return the number of days in the month.

```
>>> Month(2018, 1).n_days
31
```

nth(*self*, *day*)
Get date object for nth day of month.

Accepts nonzero integer values between +- `month.n_days`.

```
>>> Month(2018, 1).nth(1) == Month(2018, 1).start_date
True
>>> Month(2018, 1).nth(8)
datetime.date(2018, 1, 8)
```

```
>>> Month(2018, 1).nth(-2)
datetime.date(2018, 1, 30)
```

Parameters

day [int] Day of the month.

Returns

date [datetime.date] Date object for the day of the month.

range
Return a tuple of the first and last days of the month.

start_date
Return a datetime.date object for the first day of the month.

to(*self*, **args*, ***kwargs*)
Generate a list of all months between two months, inclusively.

Accepts two-element lists/tuples, date-like objects, or Month objects. If months are provided out of order (like `june_18.to.march_18`) then the list will also be in reverse order.

```
>>> Month(2018, 1).to(Month(2018, 2))
[Month(2018, 1), Month(2018, 2)]
>>> Month(2018, 3).to(2018, 1)
[Month(2018, 3), Month(2018, 2), Month(2018, 1)]
```

Parameters

other [Month, date, datetime, tuple] A Month-like object.

Returns

months [list] List of months spanning the two objects, inclusively.

CHAPTER 12

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index