pyRFC3339 Documentation

Release 1.1

Kurt Raschke

May 16, 2019

Contents

CHAPTER 1

Description

pyRFC3339 parses and generates **RFC 3339**-compliant timestamps using Python datetime.datetime objects.

```
>>> from pyrfc3339 import generate, parse
>>> from datetime import datetime
>>> import pytz
>>> generate(datetime.utcnow().replace(tzinfo=pytz.utc)) #doctest:+ELLIPSIS
'...T...Z'
>>> parse('2009-01-01T10:01:02Z')
datetime.datetime(2009, 1, 1, 10, 1, 2, tzinfo=<UTC>)
>>> parse('2009-01-01T14:01:02-04:00')
datetime.datetime(2009, 1, 1, 14, 1, 2, tzinfo=<UTC-04:00>)
```

CHAPTER 2

Installation

To install the latest version from PyPI:

```
$ pip install pyRFC3339
```

To install the latest development version:

```
$ pip install https://github.com/kurtraschke/pyRFC3339/tarball/
master#egg=pyRFC3339-dev
```

To build the documentation with Sphinx:

1. `$ pip install -f docs/requirements.txt`
2. `$ python setup.py build_sphinx`

The documentation is also available online at:

`https://pyrfc3339.readthedocs.io/`

CHAPTER 3

Changelog

3.1 1.1 (2018-06-10)

- Drop support for EOL Python releases, add (explicit) support for Python 3.5 and 3.6 (#7#10#11#12)
- Add `utils.FixedOffset.__deepcopy__()` method, to prevent crash on deepcopy (#8)

3.2 1.0 (2015-11-09)

- First formally-tagged release
- Fix *`utils.timedelta_seconds()`* to use `datetime.timedelta.total_seconds()` when the native method is available (#6)
- Documentation and packaging cleanup (#4#5)

3.3 0.2 (2014-02-09)

- Python 3 compatibility (#2)

3.4 0.1 (2011-01-26)

- Initial release

CHAPTER 4

Modules

4.1 pyrfc3339 – Imports

pyRFC3339 parses and generates **RFC 3339**-compliant timestamps using Python `datetime.datetime` objects.

```
>>> from pyrfc3339 import generate, parse
>>> from datetime import datetime
>>> import pytz
>>> generate(datetime.utcnow().replace(tzinfo=pytz.utc)) #doctest:+ELLIPSIS
'...T...Z'
>>> parse('2009-01-01T10:01:02Z')
datetime.datetime(2009, 1, 1, 10, 1, 2, tzinfo=<UTC>)
>>> parse('2009-01-01T14:01:02-04:00')
datetime.datetime(2009, 1, 1, 14, 1, 2, tzinfo=<UTC-04:00>)
```

4.2 pyrfc3339.generator – Generate RFC 3339 timestamps

pyrfc3339.generator.**generate**(*dt*, *utc=True*, *accept_naive=False*, *microseconds=False*)

Generate an **RFC 3339**-formatted timestamp from a `datetime.datetime`.

```
>>> from datetime import datetime
>>> generate(datetime(2009,1,1,12,59,59,0,pytz.utc))
'2009-01-01T12:59:59Z'
```

The timestamp will use UTC unless *utc=False* is specified, in which case it will use the timezone from the `datetime.datetime`'s `tzinfo` parameter.

```
>>> eastern = pytz.timezone('US/Eastern')
>>> dt = eastern.localize(datetime(2009,1,1,12,59,59))
>>> generate(dt)
'2009-01-01T17:59:59Z'
```

(continues on next page)

(continued from previous page)

```
>>> generate(dt, utc=False)
'2009-01-01T12:59:59-05:00'
```

Unless *accept_naive=True* is specified, the *datetime* must not be naive.

```
>>> generate(datetime(2009,1,1,12,59,59,0))
Traceback (most recent call last):
...
ValueError: naive datetime and accept_naive is False
```

```
>>> generate(datetime(2009,1,1,12,59,59,0), accept_naive=True)
'2009-01-01T12:59:59Z'
```

If *accept_naive=True* is specified, the *datetime* is assumed to be UTC. Attempting to generate a local timestamp from a naive datetime will result in an error.

```
>>> generate(datetime(2009,1,1,12,59,59,0), accept_naive=True, utc=False)
Traceback (most recent call last):
...
ValueError: cannot generate a local timestamp from a naive datetime
```

4.3 `pyrfc3339.parser` – Parse RFC 3339 timestamps

`pyrfc3339.parser.`**`parse`**(*timestamp*, *utc=False*, *produce_naive=False*)

Parse an **RFC 3339**-formatted timestamp and return a *datetime.datetime*.

If the timestamp is presented in UTC, then the *tzinfo* parameter of the returned *datetime* will be set to *pytz.utc*.

```
>>> parse('2009-01-01T10:01:02Z')
datetime.datetime(2009, 1, 1, 10, 1, 2, tzinfo=<UTC>)
```

Otherwise, a *tzinfo* instance is created with the appropriate offset, and the *tzinfo* parameter of the returned *datetime* is set to that value.

```
>>> parse('2009-01-01T14:01:02-04:00')
datetime.datetime(2009, 1, 1, 14, 1, 2, tzinfo=<UTC-04:00>)
```

However, if *parse()* is called with *utc=True*, then the returned *datetime* will be normalized to UTC (and its tzinfo parameter set to *pytz.utc*), regardless of the input timezone.

```
>>> parse('2009-01-01T06:01:02-04:00', utc=True)
datetime.datetime(2009, 1, 1, 10, 1, 2, tzinfo=<UTC>)
```

The input is strictly required to conform to **RFC 3339**, and appropriate exceptions are thrown for invalid input.

```
>>> parse('2009-01-01T06:01:02')
Traceback (most recent call last):
...
ValueError: timestamp does not conform to RFC 3339
```

```
>>> parse('2009-01-01T25:01:02Z')
Traceback (most recent call last):
...
ValueError: hour must be in 0..23
```

4.4 pyrfc3339.utils – Utilities for working with timestamps

class pyrfc3339.utils.**FixedOffset**(*hours*, *minutes*)

Represent a timezone with a fixed offset from UTC and no adjustment for DST.

```
>>> FixedOffset(4,0)
<UTC+04:00>
>>> FixedOffset(-4,0)
<UTC-04:00>
>>> FixedOffset(4,30)
<UTC+04:30>
```

```
>>> tz = FixedOffset(-5,0)
>>> tz.dst(None)
datetime.timedelta(0)
```

The class tries to do the right thing with the sign of the time zone offset:

```
>>> FixedOffset(-9,30)
<UTC-09:30>
>>> FixedOffset(-9,-30)
Traceback (most recent call last):
...
ValueError: minutes must not be negative
```

Offsets must thus be normalized so that the minute value is positive:

```
>>> FixedOffset(-8,30)
<UTC-08:30>
```

dst(*dt*)

Return offset for DST. Always returns timedelta(0).

tzname(*dt*)

Return name of timezone.

utcoffset(*dt*)

Return offset from UTC.

pyrfc3339.utils.**timedelta_seconds**(*td*)

Return the offset stored by a `datetime.timedelta` object as an integer number of seconds. Microseconds, if present, are rounded to the nearest second.

Delegates to `timedelta.total_seconds()` if available.

```
>>> timedelta_seconds(timedelta(hours=1))
3600
>>> timedelta_seconds(timedelta(hours=-1))
-3600
>>> timedelta_seconds(timedelta(hours=1, minutes=30))
5400
>>> timedelta_seconds(timedelta(hours=1, minutes=30,
... microseconds=300000))
5400
>>> timedelta_seconds(timedelta(hours=1, minutes=30,
... microseconds=900000))
5401
```

pyrfc3339.utils.**timezone**(*utcoffset*)

Return a string representing the timezone offset. Remaining seconds are rounded to the nearest minute.

```
>>> timezone(3600)
'+01:00'
>>> timezone(5400)
'+01:30'
>>> timezone(-28800)
'-08:00'
```

Python Module Index

p

Index