datetime_truncate Documentation

Release 1.0.0

Björn Andersson

December 28, 2013

Contents

1 *period* usage: 3

2 *TimeRange* usage: 5

3 Sugar 7

4 `datetime_periods` 9

4.1 Indices and tables 14

Python Module Index 15

This module aims to help you create time periods from timestamps.

CHAPTER 1

period usage:

Pass in a `datetime.datetime()` object and a period name and it'll return the beginning and end of that period.

```
>>> from datetime_periods import period
>>> period(datetime(2012, 4, 2, second=12), 'minute')
[datetime(2012, 4, 2), datetime(2012, 4, 2, 0, 0, 59)]
>>> period(datetime(2012, 4, 2), 'hour')
[datetime(2012, 4, 2, 0), datetime(2012, 4, 2, 0, 59, 59)]
>>> period(datetime(2012, 4, 2), 'day')
[datetime(2012, 4, 2), datetime(2012, 4, 2, 23, 59, 59)]
>>> period(datetime(2012, 4, 2), 'week')
[datetime(2012, 4, 2), datetime(2012, 4, 8, 23, 59, 59)]
>>> period(datetime(2012, 4, 15), 'month')
[datetime(2012, 4, 1), datetime(2012, 4, 30, 23, 59, 59)]
>>> period(datetime(2012, 4, 2), 'quarter')
[datetime(2012, 4, 1), datetime(2012, 6, 30, 23, 59, 59)]
>>> period(datetime(2012, 9, 1), 'half_year')
[datetime(2012, 7, 1), datetime(2012, 12, 31, 23, 59, 59)]
>>> period(datetime(2012, 7, 1), 'year')
[datetime(2012, 1, 1), datetime(2012, 12, 31, 23, 59, 59)]
```

CHAPTER 2

TimeRange usage:

The `TimeRange` class takes two times, *start* and *stop*, and creates *datetime* objects from them that is smart about when a date should roll over to the following day.

This class can also act like a 2 length list where index 0=start, 1=stop time. This to allow the class to be used for argument expansion and as an iterator.

```
>>> from datetime_periods import TimeRange
>>> tr = TimeRange('17:00', '23:00', '2013-12-25')
>>> tr.start
datetime(2013, 12, 25, 17)
>>> tr.stop
datetime(2013, 12, 25, 23)
>>> tr = TimeRange('17:00', '04:00', '2013-12-25')
>>> tr.start
datetime(2013, 12, 25, 17)
>>> tr.stop
datetime(2013, 12, 26, 4)
>>> tr[0] == tr.start
True
>>> tr[1] == tr.stop
True
```

CHAPTER 3

Sugar

The `sugar` module has sugar functions for all `period` variants available.

Sugar functions for entire period:

- *period_second*
- *period_minute*
- *period_hour*
- *period_day*
- *period_week*
- *period_month*
- *period_quarter*
- *period_half_year*
- *period_year*

Sugar functions for beginning of period:

- *period_beginning_second*
- *period_beginning_minute*
- *period_beginning_hour*
- *period_beginning_day*
- *period_beginning_week*
- *period_beginning_month*
- *period_beginning_quarter*
- *period_beginning_half_year*
- *period_beginning_year*

Sugar functions for end of period:

- *period_end_second*
- *period_end_minute*

- *period_end_hour*
- *period_end_day*
- *period_end_week*
- *period_end_month*
- *period_end_quarter*
- *period_end_half_year*
- *period_end_year*

CHAPTER 4

datetime_periods

class datetime_periods.**TimeRange**(*start*, *stop*, *date=None*, *tzinfo=None*)

Takes two times, `start` and `stop`, and tries to be smart about putting a date to those times.

Stop is always assumed to follow start in chronological order, so if stop is numerically less than start then it must be tomorrow.

Strings are parsed with `dateutil.parser` from the *python-dateutil* package.

This class can also act like a 2 length list where index 0=start, 1=stop time. This to allow the class to be used for argument expansion and as an iterator.

Examples:

```
>>> tr = TimeRange('17:00', '23:00', '2013-12-25')
>>> tr.start
datetime(2013, 12, 25, 17)
>>> tr.stop
datetime(2013, 12, 25, 23)
>>> tr = TimeRange('17:00', '04:00', '2013-12-25')
>>> tr.start
datetime(2013, 12, 25, 17)
>>> tr.stop
datetime(2013, 12, 26, 4)
>>> tr[0] == tr.start
True
>>> tr[1] == tr.stop
True
```

Parameters

- **start** – a `time` object or a time string
- **stop** – a `time` object or a time string
- **date** – a `datetime`, `date`, or date string
- **tzinfo** – None or a tzinfo that will replace the current one in the `timestamp` attributes `start` and `stop`

datetime_periods.**period**(*datetime*, *period_name='day'*)

Takes the given *datetime* and then creates the *period_name* that *datetime* belongs to. If given one in the middle of the day and *period_name* 'day' then it'll be from 00:00:00 till 23:59:59.

Possible values for *period_name*:

- second
- minute
- hour
- day
- week (iso week i.e. monday to sunday)
- month
- quarter
- half_year
- year

Examples:

```
>>> period(datetime(2012, 4, 2), 'hour')
[datetime(2012, 4, 2, 0), datetime(2012, 4, 2, 0, 59, 59)]
>>> period(datetime(2012, 4, 2), 'day')
[datetime(2012, 4, 2), datetime(2012, 4, 2, 23, 59, 59)]
>>> period(datetime(2012, 4, 2), 'week')
[datetime(2012, 4, 2), datetime(2012, 4, 8, 23, 59, 59)]
>>> period(datetime(2012, 4, 2), 'quarter')
[datetime(2012, 4, 1), datetime(2012, 6, 30, 23, 59, 59)]
```

Params datetime A truncated datetime object

Params period_name The period for which to calculate the end for *datetime*

Returns datetime with all fields to second set to the very last before before the next period

Return type `datetime` datetime object

`datetime_periods.`**`period_beginning`**(*datetime*, *truncate_to='day'*)

Truncates a datetime to have the values with higher precision than the one set as *truncate_to* as zero (or one for day and month).

Possible values for *truncate_to*:

- second
- minute
- hour
- day
- week (iso week i.e. to monday)
- month
- quarter
- half_year
- year

Examples:

```
>>> truncate(datetime(2012, 12, 12, 12), 'day')
datetime(2012, 12, 12)
>>> truncate(datetime(2012, 12, 14, 12, 15), 'quarter')
datetime(2012, 10, 1)
>>> truncate(datetime(2012, 3, 1), 'week')
datetime(2012, 2, 27)
```

Params datetime an initialized datetime object

Params truncate_to The highest precision to keep its original data.

Returns datetime with *truncated_to* as the highest level of precision

Return type `datetime` datetime object

This function is an alias for datetime_truncate.truncate.

`datetime_periods.`**`period_end`**(*datetime*, *period_name='day'*)

Returns a datetime where it is the end of *period_name*. Notice that `period_end` assumes that the *datetime* has been run by `truncate` before being passed in. If that is not the case the results might not be what is expected.

Possible values for *period_name*:

- second
- minute
- hour
- day
- week (iso week i.e. monday to sunday)
- month
- quarter
- half_year
- year

Examples:

```
>>> period_end(datetime(2012, 4, 2), 'hour')
datetime(2012, 4, 2, 0, 59, 59)
>>> period_end(datetime(2012, 4, 2), 'day')
datetime(2012, 4, 2, 23, 59, 59)
>>> period_end(datetime(2012, 4, 2), 'week')
datetime(2012, 4, 8, 23, 59, 59)
>>> period_end(datetime(2012, 4, 1), 'quarter')
datetime(2012, 6, 30, 23, 59, 59)
```

Params datetime A truncated datetime object

Params period_name The period for which to calculate the end for *datetime*

Returns datetime with all fields to second set to the very last before before the next period

Return type `datetime` datetime object

`datetime_periods.period.`**`period`**(*datetime*, *period_name='day'*)

Takes the given *datetime* and then creates the *period_name* that *datetime* belongs to. If given one in the middle of the day and *period_name* 'day' then it'll be from 00:00:00 till 23:59:59.

Possible values for *period_name*:

- second
- minute
- hour
- day
- week (iso week i.e. monday to sunday)
- month
- quarter
- half_year
- year

Examples:

```
>>> period(datetime(2012, 4, 2), 'hour')
[datetime(2012, 4, 2, 0), datetime(2012, 4, 2, 0, 59, 59)]
>>> period(datetime(2012, 4, 2), 'day')
[datetime(2012, 4, 2), datetime(2012, 4, 2, 23, 59, 59)]
>>> period(datetime(2012, 4, 2), 'week')
[datetime(2012, 4, 2), datetime(2012, 4, 8, 23, 59, 59)]
>>> period(datetime(2012, 4, 2), 'quarter')
[datetime(2012, 4, 1), datetime(2012, 6, 30, 23, 59, 59)]
```

Params datetime A truncated datetime object

Params period_name The period for which to calculate the end for *datetime*

Returns datetime with all fields to second set to the very last before before the next period

Return type `datetime` datetime object

`datetime_periods.period_end.`**`period_end`**(*datetime*, *period_name='day'*)

Returns a datetime where it is the end of *period_name*. Notice that `period_end` assumes that the *datetime* has been run by `truncate` before being passed in. If that is not the case the results might not be what is expected.

Possible values for *period_name*:

- second
- minute
- hour
- day
- week (iso week i.e. monday to sunday)
- month
- quarter
- half_year
- year

Examples:

```
>>> period_end(datetime(2012, 4, 2), 'hour')
datetime(2012, 4, 2, 0, 59, 59)
>>> period_end(datetime(2012, 4, 2), 'day')
datetime(2012, 4, 2, 23, 59, 59)
>>> period_end(datetime(2012, 4, 2), 'week')
datetime(2012, 4, 8, 23, 59, 59)
>>> period_end(datetime(2012, 4, 1), 'quarter')
datetime(2012, 6, 30, 23, 59, 59)
```

Params datetime A truncated datetime object

Params period_name The period for which to calculate the end for *datetime*

Returns datetime with all fields to second set to the very last before before the next period

Return type `datetime` datetime object

`datetime_periods.sugar.`**`period_beginning_day`**(*datetime*)
Sugar for `datetime_truncate.truncate(datetime, 'day')`

`datetime_periods.sugar.`**`period_beginning_half_year`**(*datetime*)
Sugar for `datetime_truncate.truncate(datetime, 'half')`

`datetime_periods.sugar.`**`period_beginning_hour`**(*datetime*)
Sugar for `datetime_truncate.truncate(datetime, 'hour')`

`datetime_periods.sugar.`**`period_beginning_minute`**(*datetime*)
Sugar for `datetime_truncate.truncate(datetime, 'minute')`

`datetime_periods.sugar.`**`period_beginning_month`**(*datetime*)
Sugar for `datetime_truncate.truncate(datetime, 'month')`

`datetime_periods.sugar.`**`period_beginning_quarter`**(*datetime*)
Sugar for `datetime_truncate.truncate(datetime, 'quarter')`

`datetime_periods.sugar.`**`period_beginning_second`**(*datetime*)
Sugar for `datetime_truncate.truncate(datetime, 'second')`

`datetime_periods.sugar.`**`period_beginning_week`**(*datetime*)
Sugar for `datetime_truncate.truncate(datetime, 'week')`

`datetime_periods.sugar.`**`period_beginning_year`**(*datetime*)
Sugar for `datetime_truncate.truncate(datetime, 'year')`

`datetime_periods.sugar.`**`period_day`**(*datetime*)
Sugar for `period(datetime, 'day')`

`datetime_periods.sugar.`**`period_end_day`**(*datetime*)
Sugar for `period_end(datetime, 'day')`

`datetime_periods.sugar.`**`period_end_half_year`**(*datetime*)
Sugar for `period_end(datetime, 'half')`

`datetime_periods.sugar.`**`period_end_hour`**(*datetime*)
Sugar for `period_end(datetime, 'hour')`

`datetime_periods.sugar.`**`period_end_minute`**(*datetime*)
Sugar for `period_end(datetime, 'minute')`

`datetime_periods.sugar.`**`period_end_month`**(*datetime*)
Sugar for `period_end(datetime, 'month')`

datetime_periods.sugar.**period_end_quarter**(*datetime*)
Sugar for `period_end(datetime, 'quarter')`

datetime_periods.sugar.**period_end_second**(*datetime*)
Sugar for `period_end(datetime, 'second')`

datetime_periods.sugar.**period_end_week**(*datetime*)
Sugar for `period_end(datetime, 'week')`

datetime_periods.sugar.**period_end_year**(*datetime*)
Sugar for `period_end(datetime, 'year')`

datetime_periods.sugar.**period_half_year**(*datetime*)
Sugar for `period(datetime, 'half')`

datetime_periods.sugar.**period_hour**(*datetime*)
Sugar for `period(datetime, 'hour')`

datetime_periods.sugar.**period_minute**(*datetime*)
Sugar for `period(datetime, 'minute')`

datetime_periods.sugar.**period_month**(*datetime*)
Sugar for `period(datetime, 'month')`

datetime_periods.sugar.**period_quarter**(*datetime*)
Sugar for `period(datetime, 'quarter')`

datetime_periods.sugar.**period_second**(*datetime*)
Sugar for `period(datetime, 'second')`

datetime_periods.sugar.**period_week**(*datetime*)
Sugar for `period(datetime, 'week')`

datetime_periods.sugar.**period_year**(*datetime*)
Sugar for `period(datetime, 'year')`

4.1 Indices and tables

- *genindex*
- *search*

Python Module Index

d

datetime_periods.period, 11
datetime_periods.period_end, 12
datetime_periods.sugar, 13