relativetimebuilder Documentation

Release 2.0.0

Brandon Nielsen

Jun 11, 2019

Contents

CHAPTER 1

aniso8601 builder for dateutil relativedeltas

CHAPTER 2

Features

- Provides `RelativeTimeBuilder` compatible with aniso8601
- Returns dateutil `relativedelta` objects for durations

CHAPTER 3

Installation

The recommended installation method is to use pip:

```
$ pip install relativetimebuilder
```

Alternatively, you can download the source (git repository hosted at Bitbucket) and install directly:

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

CHAPTER 4

Use

4.1 Parsing datetimes

To parse a typical ISO 8601 datetime string:

```
>>> import aniso8601
>>> from relativetimebuilder import RelativeTimeBuilder
>>> aniso8601.parse_datetime('1977-06-10T12:00:00', builder=RelativeTimeBuilder)
datetime.datetime(1977, 6, 10, 12, 0)
```

Alternative delimiters can be specified, for example, a space:

```
>>> aniso8601.parse_datetime('1977-06-10 12:00:00', delimiter=' ',␣
↪builder=RelativeTimeBuilder)
datetime.datetime(1977, 6, 10, 12, 0)
```

Both UTC (Z) and UTC offsets for timezones are supported:

```
>>> aniso8601.parse_datetime('1977-06-10T12:00:00Z', builder=RelativeTimeBuilder)
datetime.datetime(1977, 6, 10, 12, 0, tzinfo=+0:00:00 UTC)
>>> aniso8601.parse_datetime('1979-06-05T08:00:00-08:00', builder=RelativeTimeBuilder)
datetime.datetime(1979, 6, 5, 8, 0, tzinfo=-8:00:00 UTC)
```

Leap seconds are explicitly not supported:

```
>>> aniso8601.parse_datetime('2018-03-06T23:59:60', builder=RelativeTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/time.py", line 132, in␣
↪parse_datetime
    return builder.build_datetime(datepart, timepart)
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/builders/python.py",␣
↪line 181, in build_datetime
    cls._build_object(time))
```

(continues on next page)

(continued from previous page)

```
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/builders/__init__.py",␣
↪line 64, in _build_object
    ss=parsetuple[2], tz=parsetuple[3])
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/builders/python.py",␣
↪line 141, in build_time
    raise LeapSecondError('Leap seconds are not supported.')
aniso8601.exceptions.LeapSecondError: Leap seconds are not supported.
```

4.2 Parsing dates

To parse a date represented in an ISO 8601 string:

```
>>> import aniso8601
>>> from relativetimebuilder import RelativeTimeBuilder
>>> aniso8601.parse_date('1984-04-23', builder=RelativeTimeBuilder)
datetime.date(1984, 4, 23)
```

Basic format is supported as well:

```
>>> aniso8601.parse_date('19840423', builder=RelativeTimeBuilder)
datetime.date(1984, 4, 23)
```

To parse a date using the ISO 8601 week date format:

```
>>> aniso8601.parse_date('1986-W38-1', builder=RelativeTimeBuilder)
datetime.date(1986, 9, 15)
```

To parse an ISO 8601 ordinal date:

```
>>> aniso8601.parse_date('1988-132', builder=RelativeTimeBuilder)
datetime.date(1988, 5, 11)
```

4.3 Parsing times

To parse a time formatted as an ISO 8601 string:

```
>>> import aniso8601
>>> from relativetimebuilder import RelativeTimeBuilder
>>> aniso8601.parse_time('11:31:14', builder=RelativeTimeBuilder)
datetime.time(11, 31, 14)
```

As with all of the above, basic format is supported:

```
>>> aniso8601.parse_time('113114', builder=RelativeTimeBuilder)
datetime.time(11, 31, 14)
```

A UTC offset can be specified for times:

```
>>> aniso8601.parse_time('17:18:19-02:30', builder=RelativeTimeBuilder)
datetime.time(17, 18, 19, tzinfo=-2:30:00 UTC)
>>> aniso8601.parse_time('171819Z', builder=RelativeTimeBuilder)
datetime.time(17, 18, 19, tzinfo=+0:00:00 UTC)
```

Reduced accuracy is supported:

```
>>> aniso8601.parse_time('21:42', builder=RelativeTimeBuilder)
datetime.time(21, 42)
>>> aniso8601.parse_time('22', builder=RelativeTimeBuilder)
datetime.time(22, 0)
```

A decimal fraction is always allowed on the lowest order element of an ISO 8601 formatted time:

```
>>> aniso8601.parse_time('22:33.5', builder=RelativeTimeBuilder)
datetime.time(22, 33, 30)
>>> aniso8601.parse_time('23.75', builder=RelativeTimeBuilder)
datetime.time(23, 45)
```

Leap seconds are explicitly not supported and attempting to parse one raises a `LeapSecondError`:

```
>>> aniso8601.parse_time('23:59:60', builder=RelativeTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/time.py", line 117, in␣
↪parse_time
    return _RESOLUTION_MAP[get_time_resolution(timestr)](timestr, tz, builder)
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/time.py", line 166, in _
↪parse_second_time
    return builder.build_time(hh=hourstr, mm=minutestr, ss=secondstr, tz=tz)
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/builders/python.py",␣
↪line 141, in build_time
    raise LeapSecondError('Leap seconds are not supported.')
aniso8601.exceptions.LeapSecondError: Leap seconds are not supported.
```

4.4 Parsing durations

Parsing durations returns `relativedelta` objects from dateutil for calendar level accuracy.

To parse a duration formatted as an ISO 8601 string:

```
>>> import aniso8601
>>> from relativetimebuilder import RelativeTimeBuilder
>>> aniso8601.parse_duration('P1Y2M3DT4H54M6S', builder=RelativeTimeBuilder)
relativedelta(years=+1, months=+2, days=+3, hours=+4, minutes=+54, seconds=+6)
```

Reduced accuracy is supported:

```
>>> aniso8601.parse_duration('P1Y', builder=RelativeTimeBuilder)
relativedelta(years=+1)
```

A decimal fraction is allowed on the lowest order element:

```
>>> aniso8601.parse_duration('P1YT3.5M', builder=RelativeTimeBuilder)
relativedelta(years=+1, minutes=+3.5)
```

The decimal fraction can be specified with a comma instead of a full-stop:

```
>>> aniso8601.parse_duration('P1YT3,5M', builder=RelativeTimeBuilder)
relativedelta(years=+1, minutes=+3.5)
```

Decimal fractions are not supported for years or months as calendar level accuracy would not be guaranteed:

```
>>> aniso8601.parse_duration('P1Y2.5M', builder=RelativeTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/duration.py", line 30,␣
↪in parse_duration
    return _parse_duration_prescribed(isodurationstr, builder)
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/duration.py", line 75,␣
↪in _parse_duration_prescribed
    return _parse_duration_prescribed_notime(durationstr, builder)
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/duration.py", line 119,␣
↪in _parse_duration_prescribed_notime
    PnW=weekstr, PnD=daystr)
  File "relativetimebuilder/__init__.py", line 24, in build_duration
    raise RelativeValueError('Fractional months and years are not '
relativetimebuilder.RelativeValueError: Fractional months and years are not defined␣
↪for relative durations.
>>> aniso8601.parse_duration('P1.5Y', builder=RelativeTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/duration.py", line 30,␣
↪in parse_duration
    return _parse_duration_prescribed(isodurationstr, builder)
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/duration.py", line 75,␣
↪in _parse_duration_prescribed
    return _parse_duration_prescribed_notime(durationstr, builder)
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/duration.py", line 119,␣
↪in _parse_duration_prescribed_notime
    PnW=weekstr, PnD=daystr)
  File "relativetimebuilder/__init__.py", line 24, in build_duration
    raise RelativeValueError('Fractional months and years are not '
relativetimebuilder.RelativeValueError: Fractional months and years are not defined␣
↪for relative durations.
```

Parsing a duration from a combined date and time is supported as well:

```
>>> aniso8601.parse_duration('P0001-01-02T01:30:5', builder=RelativeTimeBuilder)
relativedelta(years=+1, months=+1, days=+2, hours=+1, minutes=+30, seconds=+5)
```

4.5 Parsing intervals

Intervals are built using `relativedelta` objects from dateutil for calendar level accuracy.

To parse an interval specified by a start and end:

```
>>> import aniso8601
>>> from relativetimebuilder import RelativeTimeBuilder
>>> aniso8601.parse_interval('2007-03-01T13:00:00/2008-05-11T15:30:00',␣
↪builder=RelativeTimeBuilder)
(datetime.datetime(2007, 3, 1, 13, 0), datetime.datetime(2008, 5, 11, 15, 30))
```

Intervals specified by a start time and a duration are supported:

```
>>> aniso8601.parse_interval('2007-03-01T13:00:00/P1Y2M10DT2H30M',␣
↪builder=RelativeTimeBuilder)
(datetime.datetime(2007, 3, 1, 13, 0), datetime.datetime(2008, 5, 11, 15, 30))
```

A duration can also be specified by a duration and end time:

```
>>> aniso8601.parse_interval('P1M/1981-04-05', builder=RelativeTimeBuilder)
(datetime.date(1981, 4, 5), datetime.date(1981, 3, 5))
```

Notice that the result of the above parse is not in order from earliest to latest. If sorted intervals are required, simply use the `sorted` keyword as shown below:

```
>>> sorted(aniso8601.parse_interval('P1M/1981-04-05', builder=RelativeTimeBuilder))
[datetime.date(1981, 3, 5), datetime.date(1981, 4, 5)]
```

The end of an interval is returned as a `datetime` when required to maintain the resolution specified by a duration, even if the duration start is given as a date:

```
>>> aniso8601.parse_interval('2014-11-12/PT4H54M6.5S', builder=RelativeTimeBuilder)
(datetime.date(2014, 11, 12), datetime.datetime(2014, 11, 12, 4, 54, 6, 500000))
>>> aniso8601.parse_interval('2007-03-01/P1.5D', builder=RelativeTimeBuilder)
(datetime.date(2007, 3, 1), datetime.datetime(2007, 3, 2, 12, 0))
```

Repeating intervals are supported as well, and return a generator:

```
>>> aniso8601.parse_repeating_interval('R3/1981-04-05/P1D',␣
↪builder=RelativeTimeBuilder)
<generator object _date_generator at 0x7f0862919fa0>
>>> list(aniso8601.parse_repeating_interval('R3/1981-04-05/P1D',␣
↪builder=RelativeTimeBuilder))
[datetime.date(1981, 4, 5), datetime.date(1981, 4, 6), datetime.date(1981, 4, 7)]
```

Repeating intervals are allowed to go in the reverse direction:

```
>>> list(aniso8601.parse_repeating_interval('R2/PT1H2M/1980-03-05T01:01:00',␣
↪builder=RelativeTimeBuilder))
[datetime.datetime(1980, 3, 5, 1, 1), datetime.datetime(1980, 3, 4, 23, 59)]
```

Unbounded intervals are also allowed (Python 2):

```
>>> result = aniso8601.parse_repeating_interval('R/PT1H2M/1980-03-05T01:01:00',␣
↪builder=RelativeTimeBuilder)
>>> result.next()
datetime.datetime(1980, 3, 5, 1, 1)
>>> result.next()
datetime.datetime(1980, 3, 4, 23, 59)
```

or for Python 3:

```
>>> result = aniso8601.parse_repeating_interval('R/PT1H2M/1980-03-05T01:01:00',␣
↪builder=RelativeTimeBuilder)
>>> next(result)
datetime.datetime(1980, 3, 5, 1, 1)
>>> next(result)
datetime.datetime(1980, 3, 4, 23, 59)
```

Intervals are calculated with calendar level accuracy:

```
>>> aniso8601.parse_interval('2003-01-27/P1M', builder=RelativeTimeBuilder)
(datetime.date(2003, 1, 27), datetime.date(2003, 2, 27))
>>> aniso8601.parse_interval('2003-01-31/P1M', builder=RelativeTimeBuilder)
(datetime.date(2003, 1, 31), datetime.date(2003, 2, 28))
>>> aniso8601.parse_interval('P1Y/2001-02-28', builder=RelativeTimeBuilder)
(datetime.date(2001, 2, 28), datetime.date(2000, 2, 28))
```

Fractional years and months do not make sense for relative intervals:

```
>>> aniso8601.parse_interval('P1.1Y/2001-02-28', builder=RelativeTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/interval.py", line 40,␣
↪in parse_interval
    intervaldelimiter, datetimedelimiter)
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/interval.py", line 98,␣
↪in _parse_interval
    return builder.build_interval(end=enddate, duration=duration)
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/builders/python.py",␣
↪line 311, in build_interval
    durationobject = cls._build_object(duration)
  File "/home/nielsenb/Jetfuse/aniso8601/aniso8601/aniso8601/builders/__init__.py",␣
↪line 71, in _build_object
    TnS=parsetuple[6])
  File "relativetimebuilder/__init__.py", line 24, in build_duration
    raise RelativeValueError('Fractional months and years are not '
relativetimebuilder.RelativeValueError: Fractional months and years are not defined␣
↪for relative durations.
```

CHAPTER 5

Development

5.1 Setup

It is recommended to develop using a virtualenv.

5.2 Tests

Tests can be run using *setuptools <https://setuptools.readthedocs.io/en/latest/setuptools.html>*:

```
$ python setup.py test
```

CHAPTER 6

Contributing

RelativeTimeBuilder is an open source project hosted on Bitbucket.

Any and all bugs are welcome on our issue tracker.