attotimebuilder

Release 0.4.0

Feb 18, 2021

Contents

CHAPTER 1

aniso8601 builder for attodatetimes

CHAPTER 2

Features

- Provides `AttoTimeBuilder` compatible with aniso8601
- Returns `attodatetime` and `attotimedelta` types

CHAPTER 3

Installation

The recommended installation method is to use pip:

```
$ pip install attotimebuilder
```

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 attotimebuilder import AttoTimeBuilder
>>> aniso8601.parse_datetime('1977-06-10T12:00:00', builder=AttoTimeBuilder)
attotime.attodatetime(1977, 6, 10, 12, 0, 0, 0, 0)
```

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

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

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

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

Leap seconds are explicitly not supported:

```
>>> aniso8601.parse_datetime('2018-03-06T23:59:60', builder=AttoTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/attotimebuilder/python2/lib/python2.7/site-packages/
↪aniso8601/time.py", line 131, in parse_datetime
    return builder.build_datetime(datepart, timepart)
  File "attotimebuilder/__init__.py", line 120, in build_datetime
    cls._build_object(time))
  File "/home/nielsenb/Jetfuse/attotimebuilder/python2/lib/python2.7/site-packages/
↪aniso8601/builder.py", line 71, in _build_object
```

(continues on next page)

(continued from previous page)

```
    ss=parsetuple[2], tz=parsetuple[3])
  File "attotimebuilder/__init__.py", line 73, in build_time
    raise LeapSecondError('Leap seconds are not supported.')
aniso8601.exceptions.LeapSecondError: Leap seconds are not supported.
```

4.2 Parsing dates

There is no `attodate` type, so native Python `datetime.date` objects are returned.

To parse a date represented in an ISO 8601 string:

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

Basic format is supported as well:

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

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

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

To parse an ISO 8601 ordinal date:

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

4.3 Parsing times

To parse a time formatted as an ISO 8601 string:

```
>>> import aniso8601
>>> from attotimebuilder import AttoTimeBuilder
>>> aniso8601.parse_time('11:31:14', builder=AttoTimeBuilder)
attotime.attotime(11, 31, 14, 0, 0)
```

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

```
>>> aniso8601.parse_time('113114', builder=AttoTimeBuilder)
attotime.attotime(11, 31, 14, 0, 0)
```

A UTC offset can be specified for times:

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

Reduced accuracy is supported:

```
>>> aniso8601.parse_time('21:42', builder=AttoTimeBuilder)
attotime.attotime(21, 42, 0, 0, 0)
>>> aniso8601.parse_time('22', builder=AttoTimeBuilder)
attotime.attotime(22, 0, 0, 0, 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=AttoTimeBuilder)
attotime.attotime(22, 33, 30, 0, 0.0)
>>> aniso8601.parse_time('23.75', builder=AttoTimeBuilder)
attotime.attotime(23, 45, 0, 0, 0.00)
```

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

```
>>> aniso8601.parse_time('23:59:60', builder=AttoTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/attotimebuilder/python2/lib/python2.7/site-packages/
↪aniso8601/time.py", line 116, in parse_time
    return _RESOLUTION_MAP[get_time_resolution(timestr)](timestr, tz, builder)
  File "/home/nielsenb/Jetfuse/attotimebuilder/python2/lib/python2.7/site-packages/
↪aniso8601/time.py", line 165, in _parse_second_time
    return builder.build_time(hh=hourstr, mm=minutestr, ss=secondstr, tz=tz)
  File "attotimebuilder/__init__.py", line 73, in build_time
    raise LeapSecondError('Leap seconds are not supported.')
aniso8601.exceptions.LeapSecondError: Leap seconds are not supported.
```

4.4 Parsing durations

To parse a duration formatted as an ISO 8601 string:

```
>>> import aniso8601
>>> from attotimebuilder import AttoTimeBuilder
>>> aniso8601.parse_duration('P1Y2M3DT4H54M6S', builder=AttoTimeBuilder)
attotime.attotimedelta(428, 17646)
```

Reduced accuracy is supported:

```
>>> aniso8601.parse_duration('P1Y', builder=AttoTimeBuilder)
attotime.attotimedelta(365)
```

A decimal fraction is allowed on the lowest order element:

```
>>> aniso8601.parse_duration('P1YT3.5M', builder=AttoTimeBuilder)
attotime.attotimedelta(365, 210)
```

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

```
>>> aniso8601.parse_duration('P1YT3,5M', builder=AttoTimeBuilder)
attotime.attotimedelta(365, 210)
```

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

```
>>> aniso8601.parse_duration('P0001-01-02T01:30:5', builder=AttoTimeBuilder)
attotime.attotimedelta(397, 5405)
```

4.5 Parsing intervals

To parse an interval specified by a start and end:

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

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

```
>>> aniso8601.parse_interval('2007-03-01T13:00:00/P1Y2M10DT2H30M',␣
↪builder=AttoTimeBuilder)
(attotime.attodatetime(2007, 3, 1, 13, 0, 0, 0, 0), attotime.attodatetime(2008, 5, 9,␣
↪15, 30, 0, 0, 0))
```

A duration can also be specified by a duration and end time, note that no `attodate` type exists, so dates are returned as native `datetime.date` objects:

```
>>> aniso8601.parse_interval('P1M/1981-04-05', builder=AttoTimeBuilder)
(datetime.date(1981, 4, 5), datetime.date(1981, 3, 6))
```

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=AttoTimeBuilder))
[datetime.date(1981, 3, 6), datetime.date(1981, 4, 5)]
```

The end of an interval is returned as a `attodatetime` 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=AttoTimeBuilder)
(datetime.date(2014, 11, 12), attotime.attodatetime(2014, 11, 12, 4, 54, 6, 500000, 0.
↪0))
>>> aniso8601.parse_interval('2007-03-01/P1.5D', builder=AttoTimeBuilder)
(datetime.date(2007, 3, 1), attotime.objects.attodatetime(2007, 3, 2, 12, 0, 0, 0, 0.
↪0))
```

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

```
>>> aniso8601.parse_repeating_interval('R3/1981-04-05/P1D', builder=AttoTimeBuilder)
<generator object _date_generator at 0x7fba29feed20>
>>> list(aniso8601.parse_repeating_interval('R3/1981-04-05/P1D',␣
↪builder=AttoTimeBuilder))
[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=AttoTimeBuilder))
[attotime.attodatetime(1980, 3, 5, 1, 1, 0, 0, 0), attotime.attodatetime(1980, 3, 4,␣
↪23, 59, 0, 0, 0)]
```

Unbounded intervals are also allowed (Python 2):

```
>>> result = aniso8601.parse_repeating_interval('R/PT1H2M/1980-03-05T01:01:00',␣
↪builder=AttoTimeBuilder)
>>> result.next()
attotime.attodatetime(1980, 3, 5, 1, 1, 0, 0, 0)
>>> result.next()
attotime.attodatetime(1980, 3, 4, 23, 59, 0, 0, 0)
```

or for Python 3:

```
>>> result = aniso8601.parse_repeating_interval('R/PT1H2M/1980-03-05T01:01:00',␣
↪builder=AttoTimeBuilder)
>>> next(result)
attotime.attodatetime(1980, 3, 5, 1, 1, 0, 0, 0)
>>> next(result)
attotime.attodatetime(1980, 3, 4, 23, 59, 0, 0, 0)
```

The above treat years as 365 days and months as 30 days. Fractional months and years are supported accordingly:

```
>>> aniso8601.parse_interval('P1.1Y/2001-02-28', builder=AttoTimeBuilder)
(datetime.date(2001, 2, 28), datetime.date(2000, 1, 23))
>>> aniso8601.parse_interval('2001-02-28/P1Y2.5M', builder=AttoTimeBuilder)
(datetime.date(2001, 2, 28), datetime.date(2002, 5, 14))
```

CHAPTER 5

Development

5.1 Setup

It is recommended to develop using a virtualenv.

Configure the development environment and pull in any required dependencies:

```
$ python setup.py develop
```

5.2 Tests

Tests can be run using the unittest testing framework:

```
$ python -m unittest discover attotimebuilder
```

CHAPTER 6

Contributing

attotimebuilder is an open source project hosted on Bitbucket.

Any and all bugs are welcome on our issue tracker.

CHAPTER 7

References

- aniso8601 and sub-microsecond precision