numpytimebuilder

Release 0.4.1

Feb 18, 2021

Contents

CHAPTER 1

aniso8601 builder for NumPy datetimes

CHAPTER 2

Features

- Provides `NumPyTimeBuilder` compatible with aniso8601
- Returns `datetime64` and `timedelta64` NumPy types

CHAPTER 3

Installation

The recommended installation method is to use pip:

```
$ pip install numpytimebuilder
```

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 numpytimebuilder import NumPyTimeBuilder
>>> aniso8601.parse_datetime('1977-06-10T12:00:00', builder=NumPyTimeBuilder)
numpy.datetime64('1977-06-10T12:00:00')
```

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

```
>>> aniso8601.parse_datetime('1977-06-10 12:00:00', delimiter=' ',␣
↪builder=NumPyTimeBuilder)
numpy.datetime64('1977-06-10T12:00:00')
```

Since the NumPy `datetime64` implementaton only supports naive datetimes, timezones are explicitly not supported:

```
>>> aniso8601.parse_datetime('1977-06-10T12:00:00Z', builder=NumPyTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/numpytimebuilder/python2/lib/python2.7/site-packages/
↪aniso8601/time.py", line 131, in parse_datetime
    return builder.build_datetime(datepart, timepart)
  File "numpytimebuilder/__init__.py", line 37, in build_datetime
    raise NotImplementedError('Timezones are not supported by numpy '
NotImplementedError: Timezones are not supported by numpy datetime64 type.
>>> aniso8601.parse_datetime('1979-06-05T08:00:00-08:00', builder=NumPyTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/numpytimebuilder/python2/lib/python2.7/site-packages/
↪aniso8601/time.py", line 131, in parse_datetime
    return builder.build_datetime(datepart, timepart)
  File "numpytimebuilder/__init__.py", line 37, in build_datetime
```

(continues on next page)

(continued from previous page)

```
    raise NotImplementedError('Timezones are not supported by numpy '
NotImplementedError: Timezones are not supported by numpy datetime64 type
```

Leap seconds are not currently supported by the NumPy `datetime64` implementation, so leap seconds are explicitly not supported:

```
>>> aniso8601.parse_datetime('2018-03-06T23:59:60', builder=NumPyTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/numpytimebuilder/python2/lib/python2.7/site-packages/
↪aniso8601/time.py", line 131, in parse_datetime
    return builder.build_datetime(datepart, timepart)
  File "numpytimebuilder/__init__.py", line 120, in build_datetime
    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 numpytimebuilder import NumPyTimeBuilder
>>> aniso8601.parse_date('1984-04-23', builder=NumPyTimeBuilder)
numpy.datetime64('1984-04-23')
```

Basic format is supported as well:

```
>>> aniso8601.parse_date('19840423', builder=NumPyTimeBuilder)
numpy.datetime64('1984-04-23')
```

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

```
>>> aniso8601.parse_date('1986-W38-1', builder=NumPyTimeBuilder)
numpy.datetime64('1986-09-15')
```

To parse an ISO 8601 ordinal date:

```
>>> aniso8601.parse_date('1988-132', builder=NumPyTimeBuilder)
numpy.datetime64('1988-05-11')
```

4.3 Parsing times

NumPy offers no `time64` type, so parsing times is explicitly not supported:

```
>>> import aniso8601
>>> from numpytimebuilder import NumPyTimeBuilder
>>> aniso8601.parse_time('11:31:14', builder=NumPyTimeBuilder)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nielsenb/Jetfuse/numpytimebuilder/python2/lib/python2.7/site-packages/
↪aniso8601/time.py", line 116, in parse_time
```

(continues on next page)

(continued from previous page)

```
    return _RESOLUTION_MAP[get_time_resolution(timestr)](timestr, tz, builder)
  File "/home/nielsenb/Jetfuse/numpytimebuilder/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 "numpytimebuilder/__init__.py", line 32, in build_time
    raise NotImplementedError('No compatible numpy time64 type.')
NotImplementedError: No compatible numpy time64 type.
```

4.4 Parsing durations

The NumPy `timedelta64` type only supports a single component per delta, so durations are returned as a tuple of `timedelta64` objects.

To parse a duration formatted as an ISO 8601 string:

```
>>> import aniso8601
>>> from numpytimebuilder import NumPyTimeBuilder
>>> aniso8601.parse_duration('P1Y2M3DT4H54M6S', builder=NumPyTimeBuilder)
(numpy.timedelta64(428,'D'), numpy.timedelta64(4,'h'), numpy.timedelta64(54,'m'),␣
↪numpy.timedelta64(6,'s'), numpy.timedelta64(0,'ms'), numpy.timedelta64(0,'us'),␣
↪numpy.timedelta64(0,'ns'), numpy.timedelta64(0,'ps'), numpy.timedelta64(0,'fs'),␣
↪numpy.timedelta64(0,'as'))
```

Reduced accuracy is supported:

```
>>> aniso8601.parse_duration('P1Y', builder=NumPyTimeBuilder)
(numpy.timedelta64(365,'D'), numpy.timedelta64(0,'h'), numpy.timedelta64(0,'m'),␣
↪numpy.timedelta64(0,'s'), numpy.timedelta64(0,'ms'), numpy.timedelta64(0,'us'),␣
↪numpy.timedelta64(0,'ns'), numpy.timedelta64(0,'ps'), numpy.timedelta64(0,'fs'),␣
↪numpy.timedelta64(0,'as'))
```

A decimal fraction is allowed on the lowest order element:

```
>>> aniso8601.parse_duration('P1YT3.5M', builder=NumPyTimeBuilder)
(numpy.timedelta64(365,'D'), numpy.timedelta64(0,'h'), numpy.timedelta64(3,'m'),␣
↪numpy.timedelta64(30,'s'), numpy.timedelta64(0,'ms'), numpy.timedelta64(0,'us'),␣
↪numpy.timedelta64(0,'ns'), numpy.timedelta64(0,'ps'), numpy.timedelta64(0,'fs'),␣
↪numpy.timedelta64(0,'as'))
```

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

```
>>> aniso8601.parse_duration('P1YT3,5M', builder=NumPyTimeBuilder)
(numpy.timedelta64(365,'D'), numpy.timedelta64(0,'h'), numpy.timedelta64(3,'m'),␣
↪numpy.timedelta64(30,'s'), numpy.timedelta64(0,'ms'), numpy.timedelta64(0,'us'),␣
↪numpy.timedelta64(0,'ns'), numpy.timedelta64(0,'ps'), numpy.timedelta64(0,'fs'),␣
↪numpy.timedelta64(0,'as'))
```

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

```
>>> aniso8601.parse_duration('P0001-01-02T01:30:5', builder=NumPyTimeBuilder)
(numpy.timedelta64(397,'D'), numpy.timedelta64(1,'h'), numpy.timedelta64(30,'m'),␣
↪numpy.timedelta64(5,'s'), numpy.timedelta64(0,'ms'), numpy.timedelta64(0,'us'),␣
↪numpy.timedelta64(0,'ns'), numpy.timedelta64(0,'ps'), numpy.timedelta64(0,'fs'),␣
↪numpy.timedelta64(0,'as'))
```

The above treat years as 365 days and months as 30 days. Calendar level accuracy is not supported. Fractional years and months are supported accordingly:

```
>>> aniso8601.parse_duration('P2.1Y', builder=NumPyTimeBuilder)
(numpy.timedelta64(766,'D'), numpy.timedelta64(12,'h'), numpy.timedelta64(0,'m'),␣
↪numpy.timedelta64(0,'s'), numpy.timedelta64(0,'ms'), numpy.timedelta64(0,'us'),␣
↪numpy.timedelta64(0,'ns'), numpy.timedelta64(0,'ps'), numpy.timedelta64(0,'fs'),␣
↪numpy.timedelta64(0,'as'))
>>> aniso8601.parse_duration('P1Y0.5M', builder=NumPyTimeBuilder)
(numpy.timedelta64(380,'D'), numpy.timedelta64(0,'h'), numpy.timedelta64(0,'m'),␣
↪numpy.timedelta64(0,'s'), numpy.timedelta64(0,'ms'), numpy.timedelta64(0,'us'),␣
↪numpy.timedelta64(0,'ns'), numpy.timedelta64(0,'ps'), numpy.timedelta64(0,'fs'),␣
↪numpy.timedelta64(0,'as'))
```

4.4.1 Applying durations

The `apply_duration` helper function is provided for applying duration tuples to a `datetime64` object. It takes a `datetime64` (from `parse_datetime`), a duration tuple (from `parse_duration`), and a Python operator to be applied:

```
>>> import aniso8601
>>> import operator
>>> from numpytimebuilder import NumPyTimeBuilder
>>> from numpytimebuilder.util import apply_duration
>>> datetime = aniso8601.parse_datetime('1977-06-10T12:00:00',␣
↪builder=NumPyTimeBuilder)
>>> duration = aniso8601.parse_duration('P3Y2M1DT1H2M3S', builder=NumPyTimeBuilder)
>>> apply_duration(datetime, duration, operator.add)
numpy.datetime64('1980-08-09T13:02:03')
```

Keep in mind the span of representable datetimes decreases as the resolution increases! See the NumPy Datetime Units documentation for more information.

4.5 Parsing intervals

To parse an interval specified by a start and end:

```
>>> import aniso8601
>>> from numpytimebuilder import NumPyTimeBuilder
>>> aniso8601.parse_interval('2007-03-01T13:00:00/2008-05-11T15:30:00')
(numpy.datetime64('2007-03-01T13:00:00'), numpy.datetime64('2008-05-11T15:30:00'))
```

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

```
>>> aniso8601.parse_interval('2007-03-01T13:00:00/P1Y2M10DT2H30M',␣
↪builder=NumPyTimeBuilder)
(numpy.datetime64('2007-03-01T13:00:00'), numpy.datetime64('2008-05-09T15:30:00'))
```

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

```
>>> aniso8601.parse_interval('P1M/1981-04-05', builder=NumPyTimeBuilder)
(numpy.datetime64('1981-04-05'), numpy.datetime64('1981-03-06'))
```

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=NumPyTimeBuilder))
[numpy.datetime64('1981-03-06'), numpy.datetime64('1981-04-05')]
```

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=NumPyTimeBuilder)
(numpy.datetime64('2014-11-12'), numpy.datetime64('2014-11-12T04:54:06.500'))
>>> aniso8601.parse_interval('2007-03-01/P1.5D', builder=NumPyTimeBuilder)
(numpy.datetime64('2007-03-01'), numpy.datetime64('2007-03-02T12:00:00'))
```

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

```
>>> aniso8601.parse_repeating_interval('R3/1981-04-05/P1D', builder=NumPyTimeBuilder)
<generator object _date_generator at 0x7fd76fe9abe0>
>>> list(aniso8601.parse_repeating_interval('R3/1981-04-05/P1D',␣
↪builder=NumPyTimeBuilder))
[numpy.datetime64('1981-04-05'), numpy.datetime64('1981-04-06'), numpy.datetime64(
↪'1981-04-07')]
```

Repeating intervals are allowed to go in the reverse direction:

```
>>> list(aniso8601.parse_repeating_interval('R2/PT1H2M/1980-03-05T01:01:00',␣
↪builder=NumPyTimeBuilder))
[numpy.datetime64('1980-03-05T01:01:00'), numpy.datetime64('1980-03-04T23:59:00')]
```

Unbounded intervals are also allowed (Python 2):

```
>>> result = aniso8601.parse_repeating_interval('R/PT1H2M/1980-03-05T01:01:00',␣
↪builder=NumPyTimeBuilder)
>>> result.next()
numpy.datetime64('1980-03-05T01:01:00')
>>> result.next()
numpy.datetime64('1980-03-04T23:59:00')
```

or for Python 3:

```
>>> result = aniso8601.parse_repeating_interval('R/PT1H2M/1980-03-05T01:01:00',␣
↪builder=NumPyTimeBuilder)
>>> next(result)
numpy.datetime64('1980-03-05T01:01:00')
>>> next(result)
numpy.datetime64('1980-03-04T23:59:00')
```

The above treat years as 365 days and months as 30 days. Calendar level accuracy is not supported. Fractional months and years are supported accordingly:

```
>>> aniso8601.parse_interval('P1.1Y/2001-02-28', builder=NumPyTimeBuilder)
(numpy.datetime64('2001-02-28'), numpy.datetime64('2000-01-24'))
>>> aniso8601.parse_interval('2001-02-28/P1Y2.5M', builder=NumPyTimeBuilder)
(numpy.datetime64('2001-02-28'), numpy.datetime64('2002-05-14'))
```

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 numpytimebuilder
```

CHAPTER 6

Contributing

numpytimebuilder is an open source project hosted on Bitbucket.

Any and all bugs are welcome on our issue tracker.

CHAPTER 7

References

- NumPy datetimes and timedeltas
- aniso8601 and sub-microsecond precision