Search projects

Help Donate Log in Register

# isodate 0.6.0

✓ Latest version

```
pip install isodate
```

Last released: Oct 13, 2017

An ISO 8601 date/time/duration parser and formatter

**Navigation**

Project description

Release history

Download files

**Project links**

Homepage

**Statistics**

GitHub statistics:

## Project description

### ISO 8601 date/time parser

build passing | coverage 94% | pypi v0.6.0 | license BSD

This module implements ISO 8601 date, time and duration parsing. The implementation follows ISO8601:2004 standard, and implements only date/time representations mentioned in the standard. If something is not mentioned there, then it is treated as non existent, and not as an allowed option.

For instance, ISO8601:2004 never mentions 2 digit years. So, it is not intended by this module to support 2 digit years. (while it may still be valid as ISO date, because it is not explicitly forbidden.) Another example is, when no time zone information is given for a time, then it should be interpreted as local time, and not UTC.

As this module maps ISO 8601 dates/times to standard Python data types, like *date*, *time*, *datetime* and *timedelta*, it is not possible to convert all possible ISO 8601 dates/times. For instance, dates before 0001-01-01 are not allowed by the Python *date* and *datetime* classes. Additionally fractional seconds are limited to microseconds. That means if the parser finds for instance nanoseconds it will round it to microseconds.

## Documentation

**Currently there are four parsing methods available.**

- **parse_time:**
  parses an ISO 8601 time string into a *time* object
- **parse_date:**
  parses an ISO 8601 date string into a *date* object
- **parse_datetime:**
  parses an ISO 8601 date-time string into a *datetime* object
- **parse_duration:**
  parses an ISO 8601 duration string into a *timedelta* or *Duration* object.
- **parse_tzinfo:**
  parses the time zone info part of an ISO 8601 string into a *tzinfo* object.

As ISO 8601 allows to define durations in years and months, and *timedelta* does not handle years and months, this module provides a *Duration* class, which can be used almost like a *timedelta* object (with some limitations). However, a *Duration* object can be converted into a *timedelta* object.

There are also ISO formatting methods for all supported data types. Each *xxx_isoformat* method accepts a format parameter. The default format is always the ISO 8601 expanded format. This is the same format used by *datetime.isoformat*:

Stars: 74

Forks: 32

Open issues/PRs: 20

View statistics for this project via Libraries.io, or by using Google BigQuery

### Meta

**License:** BSD License (BSD)

**Author:** Gerhard Weis

### Maintainers

gweis

### Classifiers

**Development Status**
  4 - Beta

**Intended Audience**
  Developers

**License**
  OSI Approved :: BSD License

**Operating System**
  OS Independent

**Programming Language**

Python
Python :: 2.6
Python :: 2.7
Python :: 3.3
Python :: 3.4
Python :: 3.5
Python :: 3.6
Python :: Implementation :: PyPy

**Topic**

Internet
Software Development :: Libraries :: Python Modules

- ***time_isoformat:***

  *Intended to create ISO time strings with default format hh:mm:ssZ.*

- ***date_isoformat:***

  *Intended to create ISO date strings with default format yyyy-mm-dd.*

- ***datetime_isoformat:***

  *Intended to create ISO date-time strings with default format yyyy-mm-ddThh:mm:ssZ.*

- ***duration_isoformat:***

  *Intended to create ISO duration strings with default format PnnYnnMnnDTnnHnnMnnS.*

- ***tz_isoformat:***

  *Intended to create ISO time zone strings with default format hh:mm.*

- ***strftime:***

  *A re-implementation mostly compatible with Python's strftime, but supports only those format strings, which can also be used for dates prior 1900. This method also understands how to format datetime and Duration instances.*

## Installation:

This module can easily be installed with Python standard installation methods.

Either use *python setup.py install* or in case you have *setuptools* or *distribute* available, you can also use *easy_install*.

## Limitations:

> - *The parser accepts several date/time representation which should be invalid according to ISO 8601 standard.*
>   1. *for date and time together, this parser accepts a mixture of basic and extended format. e.g. the date could be in basic format, while the time is accepted in extended format. It also allows short dates and times in date-time strings.*
>   2. *For incomplete dates, the first day is chosen. e.g. 19th century results in a date of 1901-01-01.*
>   3. *negative Duration and timedelta value are not fully supported yet.*

### Further information:

The doc strings and unit tests should provide rather detailed information about the methods and their limitations.

The source release provides a *setup.py* script, which can be used to run the unit tests included.

Source code is available at http://github.com/gweis/isodate.

## CHANGES

### 0.6.0 (2017-10-13)

- support incomplete month date (Fabien Loffredo)
- rely on duck typing when doing duration maths
- support ‘:’ as separator in fractional time zones (usrenmae)

### 0.5.4 (2015-08-06)

- Fix parsing of Periods (Fabien Bochu)
- Make Duration objects hashable (Geoffrey Fairchild)
- Add multiplication to duration (Reinoud Elhorst)

### 0.5.1 (2014-11-07)

- fixed pickling of Duration objects
- raise ISO8601Error when there is no ‘T’ separator in datetime strings (Adrian Coveney)

### 0.5.0 (2014-02-23)

- ISO8601Error are subclasses of ValueError now (Michael Hrivnak)
- improve compatibility across various python variants and versions
- raise exceptions when using fractional years and months in date maths with durations
- renamed method todatetime on Duraction objects to totimedelta

### 0.4.9 (2012-10-30)

- support pickling FixedOffset instances
- make sure parsed fractional seconds are in microseconds
- add leading zeros when formattig microseconds (Jarom Loveridge)

### 0.4.8 (2012-05-04)

- fixed incompatibility of unittests with python 2.5 and 2.6 (runs fine on 2.7 and 3.2)

### 0.4.7 (2012-01-26)

- fixed tzinfo formatting (never pass None into tzinfo.utcoffset())

### 0.4.6 (2012-01-06)

- added Python 3 compatibility via 2to3

### 0.4.5 (2012-01-06)

- made setuptools dependency optional

### 0.4.4 (2011-04-16)

- Fixed formatting of microseconds for datetime objects

### 0.4.3 (2010-10-29)

- Fixed problem with %P formating and fractions (supplied by David Brooks)

### 0.4.2 (2010-10-28)

- Implemented unary - for Duration (supplied by David Brooks)
- Output fractional seconds with '%P' format. (partly supplied by David Brooks)

### 0.4.1 (2010-10-13)

- fixed bug in comparison between timedelta and Duration.
- fixed precision problem with microseconds (reported by Tommi Virtanen)

### 0.4.0 (2009-02-09)

- added method to parse ISO 8601 time zone strings
- added methods to create ISO 8601 conforming strings

### 0.3.0 (2009-1-05)

- Initial release

## TODOs

This to do list contains some thoughts and ideas about missing features, and parts to think about, whether to implement them or not. This list is probably not complete.

## Missing features:

> - *time formating does not allow to create fractional representations.*
> - *parser for ISO intervals.*
> - *currently microseconds are always padded to a length of 6 characters. trailing 0s should be optional*

## Documentation:

> - ***parse_datetime:***
>   - *complete documentation to show what this function allows, but ISO forbids. and vice verse.*
>   - *support other separators between date and time than 'T'*
> - ***parse_date:***
>   - *yeardigits should be always greater than 4*
>   - *dates before 0001-01-01 are not supported*
> - ***parse_duration:***
>   - *alternative formats are not fully supported due to parse_date restrictions*
>   - *standard duration format is fully supported but not very restrictive.*
> - ***Duration:***
>   - *support fractional years and month in calculations*
>   - *implement w3c order relation? (http://www.w3.org/TR/xmlschema-2/#duration-order)*
>   - *refactor to have duration mathematics only at one place.*
>   - *localize __str__ method (does timedelta do this?)*

> - *when is a Duration negative?*
> - *normalize Durations. months [00-12] and years ]-inf,+inf[*