:mod:`datetime` --- Basic date and time types

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 4)

Unknown directive type "module".

```
.. module:: datetime
   :synopsis: Basic date and time types.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 7)

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Tim Peters <tim@zope.com>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 8)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Tim Peters <tim@zope.com>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 9)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: A.M. Kuchling <amk@amk.ca>
```

Source code: :source:`Lib/datetime.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 11); *backlink*

Unknown interpreted text role "source".

The :mod:`datetime` module supplies classes for manipulating dates and times.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 17); *backlink*

Unknown interpreted text role "mod".

While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 22)

Unknown directive type "seealso".

```
.. seealso::

   Module :mod:`calendar`
      General calendar related functions.

   Module :mod:`time`
      Time access and conversions.

   Module :mod:`zoneinfo`
      Concrete time zones representing the IANA time zone database.

   Package `dateutil <https://dateutil.readthedocs.io/en/stable/>`_
      Third-party library with expanded time zone and parsing support.
```

Aware and Naive Objects

Date and time objects may be categorized as "aware" or "naive" depending on whether or not they include timezone information.

With sufficient knowledge of applicable algorithmic and political time adjustments, such as time zone and daylight saving time information, an **aware** object can locate itself relative to other aware objects. An aware object represents a specific moment in time that is not open to interpretation. [1]

A **naive** object does not contain enough information to unambiguously locate itself relative to other date/time objects. Whether a naive object represents Coordinated Universal Time (UTC), local time, or time in some other timezone is purely up to the program, just like it is up to the program whether a particular number represents metres, miles, or mass. Naive objects are easy to understand and to work with, at the cost of ignoring some aspects of reality.

For applications requiring aware objects, :class:`.datetime` and :class:`.time` objects have an optional time zone information attribute, :attr:`!tzinfo`, that can be set to an instance of a subclass of the abstract :class:`tzinfo` class. These :class:`tzinfo` objects capture information about the offset from UTC time, the time zone name, and whether daylight saving time is in effect.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 57); *backlink*
>
> Unknown interpreted text role "class".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 57); *backlink*
>
> Unknown interpreted text role "class".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 57); *backlink*
>
> Unknown interpreted text role "attr".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 57); *backlink*
>
> Unknown interpreted text role "class".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 57); *backlink*
>
> Unknown interpreted text role "class".

Only one concrete :class:`tzinfo` class, the :class:`timezone` class, is supplied by the :mod:`datetime` module. The :class:`timezone` class can represent simple timezones with fixed offsets from UTC, such as UTC itself or North American EST and EDT timezones. Supporting timezones at deeper levels of detail is up to the application. The rules for time adjustment across the world are more political than rational, change frequently, and there is no standard suitable for every application aside from UTC.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 63); *backlink*
>
> Unknown interpreted text role "class".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 63); *backlink*
>
> Unknown interpreted text role "class".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 63); *backlink*
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 63); *backlink*
>
> Unknown interpreted text role "class".

Constants

The :mod:`datetime` module exports the following constants:

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 74); *backlink*
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-

main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 76)

Unknown directive type "data".

```
.. data:: MINYEAR

   The smallest year number allowed in a :class:`date` or :class:`.datetime` object.
   :const:`MINYEAR` is ``1``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 82)

Unknown directive type "data".

```
.. data:: MAXYEAR

   The largest year number allowed in a :class:`date` or :class:`.datetime` object.
   :const:`MAXYEAR` is ``9999``.
```

Available Types

An idealized naive date, assuming the current Gregorian calendar always was, and always will be, in effect. Attributes: :attr:`year`, :attr:`month`, and :attr:`day`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 93); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 93); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 93); *backlink*

Unknown interpreted text role "attr".

An idealized time, independent of any particular day, assuming that every day has exactly 24*60*60 seconds. (There is no notion of "leap seconds" here.) Attributes: :attr:`hour`, :attr:`minute`, :attr:`second`, :attr:`microsecond`, and :attr:`.tzinfo`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 101); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 101); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 101); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 101); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 101); *backlink*

Unknown interpreted text role "attr".

A combination of a date and a time. Attributes: :attr:`year`, :attr:`month`, :attr:`day`, :attr:`hour`, :attr:`minute`, :attr:`second`, :attr:`microsecond`, and :attr:`.tzinfo`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 110); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 110); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 110); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 110); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 110); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 110); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 110); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 110); *backlink*

Unknown interpreted text role "attr".

A duration expressing the difference between two :class:`date`, :class:`.time`, or :class:`.datetime` instances to microsecond resolution.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 118); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 118); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 118); *backlink*

Unknown interpreted text role "class".

An abstract base class for time zone information objects. These are used by the :class:`.datetime` and :class:`.time` classes to provide a customizable notion of time adjustment (for example, to account for time zone and/or daylight saving time).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 125); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 125); *backlink*

Unknown interpreted text role "class".

A class that implements the :class:`tzinfo` abstract base class as a fixed offset from the UTC.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 133); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 136)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
```

Objects of these types are immutable.

Subclass relationships:

```
object
    timedelta
    tzinfo
        timezone
    time
    date
        datetime
```

Common Properties

The :class:`date`, :class:`.datetime`, :class:`.time`, and :class:`timezone` types share these common features:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 153); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 153); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 153); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 153); *backlink*

Unknown interpreted text role "class".

- Objects of these types are immutable.
- Objects of these types are hashable, meaning that they can be used as dictionary keys.
- Objects of these types support efficient pickling via the :mod:`pickle` module.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 159); *backlink*

  Unknown interpreted text role "mod".

Determining if an Object is Aware or Naive

Objects of the :class:`date` type are always naive.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 164); *backlink*

Unknown interpreted text role "class".

An object of type :class:`.time` or :class:`.datetime` may be aware or naive.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 166); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 166); *backlink*

Unknown interpreted text role "class".

A :class:`.datetime` object *d* is aware if both of the following hold:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 168); *backlink*

Unknown interpreted text role "class".

1. `d.tzinfo` is not `None`
2. `d.tzinfo.utcoffset(d)` does not return `None`

Otherwise, *d* is naive.

A :class:`.time` object *t* is aware if both of the following hold:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 175); *backlink*

Unknown interpreted text role "class".

1. `t.tzinfo` is not `None`
2. `t.tzinfo.utcoffset(None)` does not return `None`.

Otherwise, *t* is naive.

The distinction between aware and naive doesn't apply to :class:`timedelta` objects.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 182); *backlink*

Unknown interpreted text role "class".

:class:`timedelta` Objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 187); *backlink*

Unknown interpreted text role "class".

A :class:`timedelta` object represents a duration, the difference between two dates or times.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 190); *backlink*

Unknown interpreted text role "class".

All arguments are optional and default to `0`. Arguments may be integers or floats, and may be positive or negative.

Only *days*, *seconds* and *microseconds* are stored internally. Arguments are converted to those units:

- A millisecond is converted to 1000 microseconds.
- A minute is converted to 60 seconds.
- An hour is converted to 3600 seconds.
- A week is converted to 7 days.

and days, seconds and microseconds are then normalized so that the representation is unique, with

- `0 <= microseconds < 1000000`
- `0 <= seconds < 3600*24` (the number of seconds in one day)
- `-999999999 <= days <= 999999999`

The following example illustrates how any arguments besides *days*, *seconds* and *microseconds* are "merged" and normalized into those three resulting attributes:

```
>>> from datetime import timedelta
>>> delta = timedelta(
...     days=50,
...     seconds=27,
...     microseconds=10,
...     milliseconds=29000,
...     minutes=5,
...     hours=8,
...     weeks=2
... )
>>> # Only days, seconds, and microseconds remain
>>> delta
datetime.timedelta(days=64, seconds=29156, microseconds=10)
```

If any argument is a float and there are fractional microseconds, the fractional microseconds left over from all arguments are combined and their sum is rounded to the nearest microsecond using round-half-to-even tiebreaker. If no argument is a float, the conversion and normalization processes are exact (no information is lost).

If the normalized value of days lies outside the indicated range, :exc:`OverflowError` is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 238); *backlink*

Unknown interpreted text role "exc".

Note that normalization of negative values may be surprising at first. For example:

```
>>> from datetime import timedelta
>>> d = timedelta(microseconds=-1)
>>> (d.days, d.seconds, d.microseconds)
(-1, 86399, 999999)
```

Class attributes:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 252)

Unknown directive type "attribute".

```
.. attribute:: timedelta.min

   The most negative :class:`timedelta` object, ``timedelta(-999999999)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 257)

Unknown directive type "attribute".

```
.. attribute:: timedelta.max

   The most positive :class:`timedelta` object, ``timedelta(days=999999999,
   hours=23, minutes=59, seconds=59, microseconds=999999)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 263)

Unknown directive type "attribute".

```
.. attribute:: timedelta.resolution

   The smallest possible difference between non-equal :class:`timedelta` objects,
   ``timedelta(microseconds=1)``.
```

Note that, because of normalization, `timedelta.max` > `-timedelta.min`. `-timedelta.max` is not representable as a :class:`timedelta` object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 268); *backlink*

Unknown interpreted text role "class".

Instance attributes (read-only):

Attribute	Value
`days`	Between -999999999 and 999999999 inclusive
`seconds`	Between 0 and 86399 inclusive
`microseconds`	Between 0 and 999999 inclusive

Supported operations:

Operation	Result
`t1 = t2 + t3`	Sum of *t2* and *t3*. Afterwards *t1*-*t2* == *t3* and *t1*-*t3* == *t2* are true. (1)
`t1 = t2 - t3`	Difference of *t2* and *t3*. Afterwards *t1* == *t2* - *t3* and *t2* == *t1* + *t3* are true. (1)(6)
`t1 = t2 * i or t1 = i * t2`	Delta multiplied by an integer. Afterwards *t1* // i == *t2* is true, provided `i != 0`.
	In general, *t1* * i == *t1* * (i-1) + *t1* is true. (1)
`t1 = t2 * f or t1 = f * t2`	Delta multiplied by a float. The result is rounded to the nearest multiple of timedelta.resolution using round-half-to-even.

Operation	Result
`f = t2 / t3`	Division (3) of overall duration *t2* by interval unit *t3*. Returns a :class:`float` object. **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 309); *backlink*** Unknown interpreted text role "class".
`t1 = t2 / f or t1 = t2 / i`	Delta divided by a float or an int. The result is rounded to the nearest multiple of timedelta.resolution using round-half-to-even.
`t1 = t2 // i` or `t1 = t2 // t3`	The floor is computed and the remainder (if any) is thrown away. In the second case, an integer is returned. (3)
`t1 = t2 % t3`	The remainder is computed as a :class:`timedelta` object. (3) **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 321); *backlink*** Unknown interpreted text role "class".
`q, r = divmod(t1, t2)`	Computes the quotient and the remainder: `q = t1 // t2` (3) and `r = t1 % t2`. q is an integer and r is a :class:`timedelta` object. **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 324); *backlink*** Unknown interpreted text role "class".
`+t1`	Returns a :class:`timedelta` object with the same value. (2) **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 329); *backlink*** Unknown interpreted text role "class".
`-t1`	equivalent to :class:`timedelta`(*-t1.days*, *-t1.seconds*, *-t1.microseconds*), and to *t1** -1. (1)(4) **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 332); *backlink*** Unknown interpreted text role "class".
`abs(t)`	equivalent to +*t* when `t.days >= 0`, and to -*t* when `t.days < 0`. (2)
`str(t)`	Returns a string in the form `[D day[s], ][H]H:MM:SS[.UUUUUU]`, where D is negative for negative `t`. (5)
`repr(t)`	Returns a string representation of the :class:`timedelta` object as a constructor call with canonical attribute values. **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 344); *backlink*** Unknown interpreted text role "class".

Notes:

1. This is exact but may overflow.
2. This is exact and cannot overflow.
3. Division by 0 raises :exc:`ZeroDivisionError`.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 358); *backlink*

   Unknown interpreted text role "exc".

4. -*timedelta.max* is not representable as a :class:`timedelta` object.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 361); *backlink*

   Unknown interpreted text role "class".

5. String representations of :class:`timedelta` objects are normalized similarly to their internal representation. This leads to somewhat unusual results for negative timedeltas. For example:

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 364); *backlink*

   Unknown interpreted text role "class".

   ```
   >>> timedelta(hours=-5)
   datetime.timedelta(days=-1, seconds=68400)
   >>> print(_)
   -1 day, 19:00:00
   ```

6. The expression `t2 - t3` will always be equal to the expression `t2 + (-t3)` except when t3 is equal to `timedelta.max`; in that case the former will produce a result while the latter will overflow.

In addition to the operations listed above, :class:`timedelta` objects support certain additions and subtractions with :class:`date` and :class:`.datetime` objects (see below).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 378); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 378); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 378); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 382)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.2
   Floor division and true division of a :class:`timedelta` object by another
   :class:`timedelta` object are now supported, as are remainder operations and
   the :func:`divmod` function. True division and multiplication of a
   :class:`timedelta` object by a :class:`float` object are now supported.
```

Comparisons of :class:`timedelta` objects are supported, with some caveats.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 389); *backlink*

Unknown interpreted text role "class".

The comparisons `==` or `!=` *always* return a :class:`bool`, no matter the type of the compared object:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 391); *backlink*

Unknown interpreted text role "class".

```
>>> from datetime import timedelta
>>> delta1 = timedelta(seconds=57)
>>> delta2 = timedelta(hours=25, seconds=2)
>>> delta2 != delta1
True
>>> delta2 == 5
False
```

For all other comparisons (such as < and >), when a :class:`timedelta` object is compared to an object of a different type, :exc:`TypeError` is raised:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 402); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 402); *backlink*

Unknown interpreted text role "exc".

```
>>> delta2 > delta1
True
>>> delta2 > 5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '>' not supported between instances of 'datetime.timedelta' and 'int'
```

In Boolean contexts, a :class:`timedelta` object is considered to be true if and only if it isn't equal to `timedelta(0)`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 413); *backlink*

Unknown interpreted text role "class".

Instance methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 418)

Unknown directive type "method".

```
.. method:: timedelta.total_seconds()

   Return the total number of seconds contained in the duration. Equivalent to
   ``td / timedelta(seconds=1)``. For interval units other than seconds, use the
   division form directly (e.g. ``td / timedelta(microseconds=1)``).

   Note that for very large time intervals (greater than 270 years on
   most platforms) this method will lose microsecond accuracy.

   .. versionadded:: 3.2
```

Examples of usage: :class:`timedelta`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 429); *backlink*

Unknown interpreted text role "class".

An additional example of normalization:

```
>>> # Components of another_year add up to exactly 365 days
>>> from datetime import timedelta
>>> year = timedelta(days=365)
>>> another_year = timedelta(weeks=40, days=84, hours=23,
...                          minutes=50, seconds=600)
>>> year == another_year
True
>>> year.total_seconds()
31536000.0
```

Examples of :class:`timedelta` arithmetic:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 444); *backlink*

Unknown interpreted text role "class".

```
>>> from datetime import timedelta
```

```
>>> year = timedelta(days=365)
>>> ten_years = 10 * year
>>> ten_years
datetime.timedelta(days=3650)
>>> ten_years.days // 365
10
>>> nine_years = ten_years - year
>>> nine_years
datetime.timedelta(days=3285)
>>> three_years = nine_years // 3
>>> three_years, three_years.days // 365
(datetime.timedelta(days=1095), 3)
```

:class:`date` Objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 462); *backlink*

Unknown interpreted text role "class".

A :class:`date` object represents a date (year, month and day) in an idealized calendar, the current Gregorian calendar indefinitely extended in both directions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 465); *backlink*

Unknown interpreted text role "class".

January 1 of year 1 is called day number 1, January 2 of year 1 is called day number 2, and so on. [2]

All arguments are required. Arguments must be integers, in the following ranges:

- `MINYEAR <= year <= MAXYEAR`
- `1 <= month <= 12`
- `1 <= day <= number of days in the given month and year`

If an argument outside those ranges is given, :exc:`ValueError` is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 481); *backlink*

Unknown interpreted text role "exc".

Other constructors, all class methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 486)

Unknown directive type "classmethod".

```
.. classmethod:: date.today()

   Return the current local date.

   This is equivalent to ``date.fromtimestamp(time.time())``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 492)

Unknown directive type "classmethod".

```
.. classmethod:: date.fromtimestamp(timestamp)

   Return the local date corresponding to the POSIX timestamp, such as is
   returned by :func:`time.time`.

   This may raise :exc:`OverflowError`, if the timestamp is out
   of the range of values supported by the platform C :c:func:`localtime`
   function, and :exc:`OSError` on :c:func:`localtime` failure.
   It's common for this to be restricted to years from 1970 through 2038. Note
   that on non-POSIX systems that include leap seconds in their notion of a
   timestamp, leap seconds are ignored by :meth:`fromtimestamp`.

   .. versionchanged:: 3.3
      Raise :exc:`OverflowError` instead of :exc:`ValueError` if the timestamp
      is out of the range of values supported by the platform C
      :c:func:`localtime` function. Raise :exc:`OSError` instead of
      :exc:`ValueError` on :c:func:`localtime` failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 511)

Unknown directive type "classmethod".

```
.. classmethod:: date.fromordinal(ordinal)

   Return the date corresponding to the proleptic Gregorian ordinal, where
   January 1 of year 1 has ordinal 1.

   :exc:`ValueError` is raised unless ``1 <= ordinal <=
   date.max.toordinal()``. For any date *d*,
   ``date.fromordinal(d.toordinal()) == d``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 521)

Unknown directive type "classmethod".

```
.. classmethod:: date.fromisoformat(date_string)

   Return a :class:`date` corresponding to a *date_string* given in the format
   ``YYYY-MM-DD``::

      >>> from datetime import date
      >>> date.fromisoformat('2019-12-04')
      datetime.date(2019, 12, 4)

   This is the inverse of :meth:`date.isoformat`. It only supports the format
   ``YYYY-MM-DD``.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 536)

Unknown directive type "classmethod".

```
.. classmethod:: date.fromisocalendar(year, week, day)

   Return a :class:`date` corresponding to the ISO calendar date specified by
   year, week and day. This is the inverse of the function :meth:`date.isocalendar`.

   .. versionadded:: 3.8
```

Class attributes:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 546)

Unknown directive type "attribute".

```
.. attribute:: date.min

   The earliest representable date, ``date(MINYEAR, 1, 1)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 551)

Unknown directive type "attribute".

```
.. attribute:: date.max

   The latest representable date, ``date(MAXYEAR, 12, 31)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 556)

Unknown directive type "attribute".

```
.. attribute:: date.resolution

   The smallest possible difference between non-equal date objects,
   ``timedelta(days=1)``.
```

Instance attributes (read-only):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 564)

Unknown directive type "attribute".

```
.. attribute:: date.year

   Between :const:`MINYEAR` and :const:`MAXYEAR` inclusive.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 569)

Unknown directive type "attribute".

```
.. attribute:: date.month

   Between 1 and 12 inclusive.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 574)

Unknown directive type "attribute".

```
.. attribute:: date.day

   Between 1 and the number of days in the given month of the given year.
```

Supported operations:

Operation	Result
`date2 = date1 + timedelta`	*date2* is `timedelta.days` days removed from *date1*. (1)
`date2 = date1 - timedelta`	Computes *date2* such that `date2 + timedelta == date1`. (2)
`timedelta = date1 - date2`	(3)
`date1 < date2`	*date1* is considered less than *date2* when *date1* precedes *date2* in time. (4)

Notes:

1. *date2* is moved forward in time if `timedelta.days > 0`, or backward if `timedelta.days < 0`. Afterward `date2 - date1 == timedelta.days`. `timedelta.seconds` and `timedelta.microseconds` are ignored. :exc:`OverflowError` is raised if `date2.year` would be smaller than :const:`MINYEAR` or larger than :const:`MAXYEAR`.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 599); *backlink*

   Unknown interpreted text role "exc".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 599); *backlink*

   Unknown interpreted text role "const".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 599); *backlink*

   Unknown interpreted text role "const".

2. `timedelta.seconds` and `timedelta.microseconds` are ignored.
3. This is exact, and cannot overflow. timedelta.seconds and timedelta.microseconds are 0, and date2 + timedelta == date1 after.
4. In other words, `date1 < date2` if and only if `date1.toordinal() < date2.toordinal()`. Date comparison raises :exc:`TypeError` if the other comparand isn't also a :class:`date` object. However, `NotImplemented` is returned instead if the other comparand has a :meth:`timetuple` attribute. This hook gives other kinds of date objects a chance at implementing mixed-type comparison. If not, when a :class:`date` object is compared to an object of a different type, :exc:`TypeError` is raised unless the comparison is `==` or `!=`. The latter cases return :const:`False` or :const:`True`, respectively.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 613); *backlink*

   Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 613); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 613); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 613); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 613); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 613); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 613); *backlink*

Unknown interpreted text role "const".

In Boolean contexts, all :class:`date` objects are considered to be true.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 623); *backlink*

Unknown interpreted text role "class".

Instance methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 627)

Unknown directive type "method".

```
.. method:: date.replace(year=self.year, month=self.month, day=self.day)

   Return a date with the same value, except for those parameters given new
   values by whichever keyword arguments are specified.

   Example::

       >>> from datetime import date
       >>> d = date(2002, 12, 31)
       >>> d.replace(day=26)
       datetime.date(2002, 12, 26)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 640)

Unknown directive type "method".

```
.. method:: date.timetuple()

   Return a :class:`time.struct_time` such as returned by :func:`time.localtime`.

   The hours, minutes and seconds are 0, and the DST flag is -1.

   ``d.timetuple()`` is equivalent to::

     time.struct_time((d.year, d.month, d.day, 0, 0, 0, d.weekday(), yday, -1))

   where ``yday = d.toordinal() - date(d.year, 1, 1).toordinal() + 1``
```

```
   is the day number within the current year starting with ``1`` for January 1st.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 654)

Unknown directive type "method".

```
.. method:: date.toordinal()

   Return the proleptic Gregorian ordinal of the date, where January 1 of year 1
   has ordinal 1. For any :class:`date` object *d*,
   ``date.fromordinal(d.toordinal()) == d``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 661)

Unknown directive type "method".

```
.. method:: date.weekday()

   Return the day of the week as an integer, where Monday is 0 and Sunday is 6.
   For example, ``date(2002, 12, 4).weekday() == 2``, a Wednesday. See also
   :meth:`isoweekday`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 668)

Unknown directive type "method".

```
.. method:: date.isoweekday()

   Return the day of the week as an integer, where Monday is 1 and Sunday is 7.
   For example, ``date(2002, 12, 4).isoweekday() == 3``, a Wednesday. See also
   :meth:`weekday`, :meth:`isocalendar`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 675)

Unknown directive type "method".

```
.. method:: date.isocalendar()

   Return a :term:`named tuple` object with three components: ``year``,
   ``week`` and ``weekday``.

   The ISO calendar is a widely used variant of the Gregorian calendar. [#]_

   The ISO year consists of 52 or 53 full weeks, and where a week starts on a
   Monday and ends on a Sunday. The first week of an ISO year is the first
   (Gregorian) calendar week of a year containing a Thursday. This is called week
   number 1, and the ISO year of that Thursday is the same as its Gregorian year.

   For example, 2004 begins on a Thursday, so the first week of ISO year 2004
   begins on Monday, 29 Dec 2003 and ends on Sunday, 4 Jan 2004::

        >>> from datetime import date
        >>> date(2003, 12, 29).isocalendar()
        datetime.IsoCalendarDate(year=2004, week=1, weekday=1)
        >>> date(2004, 1, 4).isocalendar()
        datetime.IsoCalendarDate(year=2004, week=1, weekday=7)

   .. versionchanged:: 3.9
      Result changed from a tuple to a :term:`named tuple`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 699)

Unknown directive type "method".

```
.. method:: date.isoformat()

   Return a string representing the date in ISO 8601 format, ``YYYY-MM-DD``::

       >>> from datetime import date
       >>> date(2002, 12, 4).isoformat()
       '2002-12-04'

   This is the inverse of :meth:`date.fromisoformat`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 709)

Unknown directive type "method".

```
.. method:: date.__str__()

   For a date *d*, ``str(d)`` is equivalent to ``d.isoformat()``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 714)

Unknown directive type "method".

```
.. method:: date.ctime()

   Return a string representing the date::

       >>> from datetime import date
       >>> date(2002, 12, 4).ctime()
       'Wed Dec  4 00:00:00 2002'

   ``d.ctime()`` is equivalent to::

     time.ctime(time.mktime(d.timetuple()))

   on platforms where the native C
   :c:func:`ctime` function (which :func:`time.ctime` invokes, but which
   :meth:`date.ctime` does not invoke) conforms to the C standard.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 731)

Unknown directive type "method".

```
.. method:: date.strftime(format)

   Return a string representing the date, controlled by an explicit format string.
   Format codes referring to hours, minutes or seconds will see 0 values. For a
   complete list of formatting directives, see
   :ref:`strftime-strptime-behavior`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 739)

Unknown directive type "method".

```
.. method:: date.__format__(format)

   Same as :meth:`.date.strftime`. This makes it possible to specify a format
   string for a :class:`.date` object in :ref:`formatted string
   literals <f-strings>` and when using :meth:`str.format`. For a
   complete list of formatting directives, see
   :ref:`strftime-strptime-behavior`.
```

Examples of Usage: :class:`date`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 747); *backlink*

Unknown interpreted text role "class".

Example of counting days to an event:

```
>>> import time
>>> from datetime import date
>>> today = date.today()
>>> today
datetime.date(2007, 12, 5)
>>> today == date.fromtimestamp(time.time())
True
>>> my_birthday = date(today.year, 6, 24)
>>> if my_birthday < today:
...     my_birthday = my_birthday.replace(year=today.year + 1)
>>> my_birthday
datetime.date(2008, 6, 24)
>>> time_to_birthday = abs(my_birthday - today)
>>> time_to_birthday.days
202
```

More examples of working with :class:`date`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 768); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 770)

Unknown directive type "doctest".

```
.. doctest::

    >>> from datetime import date
    >>> d = date.fromordinal(730920) # 730920th day after 1. 1. 0001
    >>> d
    datetime.date(2002, 3, 11)

    >>> # Methods related to formatting string output
    >>> d.isoformat()
    '2002-03-11'
    >>> d.strftime("%d/%m/%y")
    '11/03/02'
    >>> d.strftime("%A %d. %B %Y")
    'Monday 11. March 2002'
    >>> d.ctime()
    'Mon Mar 11 00:00:00 2002'
    >>> 'The {1} is {0:%d}, the {2} is {0:%B}.'.format(d, "day", "month")
    'The day is 11, the month is March.'

    >>> # Methods for to extracting 'components' under different calendars
    >>> t = d.timetuple()
    >>> for i in t:     # doctest: +SKIP
    ...     print(i)
    2002                # year
    3                   # month
    11                  # day
    0
    0
    0
    0                   # weekday (0 = Monday)
    70                  # 70th day in the year
    -1
    >>> ic = d.isocalendar()
    >>> for i in ic:    # doctest: +SKIP
    ...     print(i)
    2002                # ISO year
    11                  # ISO week number
    1                   # ISO day number ( 1 = Monday )

    >>> # A date object is immutable; all operations produce a new object
    >>> d.replace(year=2005)
    datetime.date(2005, 3, 11)
```

:class:`.datetime` Objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 816); *backlink*

Unknown interpreted text role "class".

A :class:`.datetime` object is a single object containing all the information from a :class:`date` object and a :class:`.time` object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 819); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 819); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 819); *backlink*

Unknown interpreted text role "class".

Like a :class:`date` object, :class:`.datetime` assumes the current Gregorian calendar extended in both directions; like a :class:`.time` object, :class:`.datetime` assumes there are exactly 3600*24 seconds in every day.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 822); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 822); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 822); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 822); *backlink*

Unknown interpreted text role "class".

Constructor:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 828)

Invalid class attribute value for "class" directive: "datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None, *, fold=0)".

```
.. class:: datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None, *, fold=0)

   The *year*, *month* and *day* arguments are required. *tzinfo* may be ``None``, or an
   instance of a :class:`tzinfo` subclass. The remaining arguments must be integers
   in the following ranges:

   * ``MINYEAR <= year <= MAXYEAR``,
   * ``1 <= month <= 12``,
   * ``1 <= day <= number of days in the given month and year``,
   * ``0 <= hour < 24``,
   * ``0 <= minute < 60``,
   * ``0 <= second < 60``,
   * ``0 <= microsecond < 1000000``,
   * ``fold in [0, 1]``.

   If an argument outside those ranges is given, :exc:`ValueError` is raised.

   .. versionadded:: 3.6
      Added the ``fold`` argument.
```

Other constructors, all class methods:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 850)

Unknown directive type "classmethod".

```
.. classmethod:: datetime.today()

   Return the current local datetime, with :attr:`.tzinfo` ``None``.

   Equivalent to::

     datetime.fromtimestamp(time.time())

   See also :meth:`now`, :meth:`fromtimestamp`.

   This method is functionally equivalent to :meth:`now`, but without a
   ``tz`` parameter.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst`, line 863)

Unknown directive type "classmethod".

```
.. classmethod:: datetime.now(tz=None)

   Return the current local date and time.

   If optional argument *tz* is ``None``
   or not specified, this is like :meth:`today`, but, if possible, supplies more
   precision than can be gotten from going through a :func:`time.time` timestamp
   (for example, this may be possible on platforms supplying the C
   :c:func:`gettimeofday` function).
```

```
   If *tz* is not ``None``, it must be an instance of a :class:`tzinfo` subclass,
   and the current date and time are converted to *tz*â€™s time zone.

   This function is preferred over :meth:`today` and :meth:`utcnow`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 879)

Unknown directive type "classmethod".

```
.. classmethod:: datetime.utcnow()

   Return the current UTC date and time, with :attr:`.tzinfo` ``None``.

   This is like :meth:`now`, but returns the current UTC date and time, as a naive
   :class:`.datetime` object. An aware current UTC datetime can be obtained by
   calling ``datetime.now(timezone.utc)``. See also :meth:`now`.

   .. warning::

      Because naive ``datetime`` objects are treated by many ``datetime`` methods
      as local times, it is preferred to use aware datetimes to represent times
      in UTC. As such, the recommended way to create an object representing the
      current time in UTC is by calling ``datetime.now(timezone.utc)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 895)

Unknown directive type "classmethod".

```
.. classmethod:: datetime.fromtimestamp(timestamp, tz=None)

   Return the local date and time corresponding to the POSIX timestamp, such as is
   returned by :func:`time.time`. If optional argument *tz* is ``None`` or not
   specified, the timestamp is converted to the platform's local date and time, and
   the returned :class:`.datetime` object is naive.

   If *tz* is not ``None``, it must be an instance of a :class:`tzinfo` subclass, and the
   timestamp is converted to *tz*â€™s time zone.

   :meth:`fromtimestamp` may raise :exc:`OverflowError`, if the timestamp is out of
   the range of values supported by the platform C :c:func:`localtime` or
   :c:func:`gmtime` functions, and :exc:`OSError` on :c:func:`localtime` or
   :c:func:`gmtime` failure.
   It's common for this to be restricted to years in
   1970 through 2038. Note that on non-POSIX systems that include leap seconds in
   their notion of a timestamp, leap seconds are ignored by :meth:`fromtimestamp`,
   and then it's possible to have two timestamps differing by a second that yield
   identical :class:`.datetime` objects. This method is preferred over
   :meth:`utcfromtimestamp`.

   .. versionchanged:: 3.3
      Raise :exc:`OverflowError` instead of :exc:`ValueError` if the timestamp
      is out of the range of values supported by the platform C
      :c:func:`localtime` or :c:func:`gmtime` functions. Raise :exc:`OSError`
      instead of :exc:`ValueError` on :c:func:`localtime` or :c:func:`gmtime`
      failure.

   .. versionchanged:: 3.6
      :meth:`fromtimestamp` may return instances with :attr:`.fold` set to 1.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 926)

Unknown directive type "classmethod".

```
.. classmethod:: datetime.utcfromtimestamp(timestamp)

   Return the UTC :class:`.datetime` corresponding to the POSIX timestamp, with
   :attr:`.tzinfo` ``None``.  (The resulting object is naive.)

   This may raise :exc:`OverflowError`, if the timestamp is
   out of the range of values supported by the platform C :c:func:`gmtime` function,
   and :exc:`OSError` on :c:func:`gmtime` failure.
   It's common for this to be restricted to years in 1970 through 2038.

   To get an aware :class:`.datetime` object, call :meth:`fromtimestamp`::

     datetime.fromtimestamp(timestamp, timezone.utc)

   On the POSIX compliant platforms, it is equivalent to the following
   expression::

     datetime(1970, 1, 1, tzinfo=timezone.utc) + timedelta(seconds=timestamp)
```

```
except the latter formula always supports the full years range: between
:const:`MINYEAR` and :const:`MAXYEAR` inclusive.

.. warning::

   Because naive ``datetime`` objects are treated by many ``datetime`` methods
   as local times, it is preferred to use aware datetimes to represent times
   in UTC. As such, the recommended way to create an object representing a
   specific timestamp in UTC is by calling
   ``datetime.fromtimestamp(timestamp, tz=timezone.utc)``.

.. versionchanged:: 3.3
   Raise :exc:`OverflowError` instead of :exc:`ValueError` if the timestamp
   is out of the range of values supported by the platform C
   :c:func:`gmtime` function. Raise :exc:`OSError` instead of
   :exc:`ValueError` on :c:func:`gmtime` failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 963)

Unknown directive type "classmethod".

```
.. classmethod:: datetime.fromordinal(ordinal)

   Return the :class:`.datetime` corresponding to the proleptic Gregorian ordinal,
   where January 1 of year 1 has ordinal 1. :exc:`ValueError` is raised unless ``1
   <= ordinal <= datetime.max.toordinal()``. The hour, minute, second and
   microsecond of the result are all 0, and :attr:`.tzinfo` is ``None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 971)

Unknown directive type "classmethod".

```
.. classmethod:: datetime.combine(date, time, tzinfo=self.tzinfo)

   Return a new :class:`.datetime` object whose date components are equal to the
   given :class:`date` object's, and whose time components
   are equal to the given :class:`.time` object's. If the *tzinfo*
   argument is provided, its value is used to set the :attr:`.tzinfo` attribute
   of the result, otherwise the :attr:`~.time.tzinfo` attribute of the *time* argument
   is used.

   For any :class:`.datetime` object *d*,
   ``d == datetime.combine(d.date(), d.time(), d.tzinfo)``. If date is a
   :class:`.datetime` object, its time components and :attr:`.tzinfo` attributes
   are ignored.

   .. versionchanged:: 3.6
      Added the *tzinfo* argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 989)

Unknown directive type "classmethod".

```
.. classmethod:: datetime.fromisoformat(date_string)

   Return a :class:`.datetime` corresponding to a *date_string* in one of the
   formats emitted by :meth:`date.isoformat` and :meth:`datetime.isoformat`.

   Specifically, this function supports strings in the format:

   .. code-block:: none

      YYYY-MM-DD[*HH[:MM[:SS[.fff[fff]]]][+HH:MM[:SS[.ffffff]]]]

   where ``*`` can match any single character.

   .. caution::

     This does *not* support parsing arbitrary ISO 8601 strings - it is only intended
     as the inverse operation of :meth:`datetime.isoformat`. A more full-featured
     ISO 8601 parser, ``dateutil.parser.isoparse`` is available in the third-party package
     `dateutil <https://dateutil.readthedocs.io/en/stable/parser.html#dateutil.parser.isoparse>`__.

   Examples::

       >>> from datetime import datetime
       >>> datetime.fromisoformat('2011-11-04')
       datetime.datetime(2011, 11, 4, 0, 0)
       >>> datetime.fromisoformat('2011-11-04T00:05:23')
       datetime.datetime(2011, 11, 4, 0, 5, 23)
```

```
        >>> datetime.fromisoformat('2011-11-04 00:05:23.283')
        datetime.datetime(2011, 11, 4, 0, 5, 23, 283000)
        >>> datetime.fromisoformat('2011-11-04 00:05:23.283+00:00')
        datetime.datetime(2011, 11, 4, 0, 5, 23, 283000, tzinfo=datetime.timezone.utc)
        >>> datetime.fromisoformat('2011-11-04T00:05:23+04:00')   # doctest: +NORMALIZE_WHITESPACE
        datetime.datetime(2011, 11, 4, 0, 5, 23,
            tzinfo=datetime.timezone(datetime.timedelta(seconds=14400)))

    .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1026)

Unknown directive type "classmethod".

```
.. classmethod:: datetime.fromisocalendar(year, week, day)

   Return a :class:`.datetime` corresponding to the ISO calendar date specified
   by year, week and day. The non-date components of the datetime are populated
   with their normal default values. This is the inverse of the function
   :meth:`datetime.isocalendar`.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1035)

Unknown directive type "classmethod".

```
.. classmethod:: datetime.strptime(date_string, format)

   Return a :class:`.datetime` corresponding to *date_string*, parsed according to
   *format*.

   This is equivalent to::

     datetime(*(time.strptime(date_string, format)[0:6]))

   :exc:`ValueError` is raised if the date_string and format
   can't be parsed by :func:`time.strptime` or if it returns a value which isn't a
   time tuple. For a complete list of formatting directives, see
   :ref:`strftime-strptime-behavior`.
```

Class attributes:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1053)

Unknown directive type "attribute".

```
.. attribute:: datetime.min

   The earliest representable :class:`.datetime`, ``datetime(MINYEAR, 1, 1,
   tzinfo=None)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1059)

Unknown directive type "attribute".

```
.. attribute:: datetime.max

   The latest representable :class:`.datetime`, ``datetime(MAXYEAR, 12, 31, 23, 59,
   59, 999999, tzinfo=None)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1065)

Unknown directive type "attribute".

```
.. attribute:: datetime.resolution

   The smallest possible difference between non-equal :class:`.datetime` objects,
   ``timedelta(microseconds=1)``.
```

Instance attributes (read-only):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1073)

Unknown directive type "attribute".

```
.. attribute:: datetime.year

   Between :const:`MINYEAR` and :const:`MAXYEAR` inclusive.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1078)

Unknown directive type "attribute".

```
.. attribute:: datetime.month

   Between 1 and 12 inclusive.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1083)

Unknown directive type "attribute".

```
.. attribute:: datetime.day

   Between 1 and the number of days in the given month of the given year.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1088)

Unknown directive type "attribute".

```
.. attribute:: datetime.hour

   In ``range(24)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1093)

Unknown directive type "attribute".

```
.. attribute:: datetime.minute

   In ``range(60)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1098)

Unknown directive type "attribute".

```
.. attribute:: datetime.second

   In ``range(60)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1103)

Unknown directive type "attribute".

```
.. attribute:: datetime.microsecond

   In ``range(1000000)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1108)

Unknown directive type "attribute".

```
.. attribute:: datetime.tzinfo

   The object passed as the *tzinfo* argument to the :class:`.datetime` constructor,
   or ``None`` if none was passed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1114)

Unknown directive type "attribute".

```
.. attribute:: datetime.fold

   In ``[0, 1]``. Used to disambiguate wall times during a repeated interval. (A
   repeated interval occurs when clocks are rolled back at the end of daylight saving
   time or when the UTC offset for the current zone is decreased for political reasons.)
   The value 0 (1) represents the earlier (later) of the two moments with the same wall
   time representation.

   .. versionadded:: 3.6
```

Supported operations:

Operation	Result
`datetime2 = datetime1 + timedelta`	(1)
`datetime2 = datetime1 - timedelta`	(2)
`timedelta = datetime1 - datetime2`	(3)
`datetime1 < datetime2`	Compares :class:`.datetime` to :class:`.datetime`. (4) **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1136); *backlink*** Unknown interpreted text role "class". **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1136); *backlink*** Unknown interpreted text role "class".

1. datetime2 is a duration of timedelta removed from datetime1, moving forward in time if `timedelta.days` > 0, or backward if `timedelta.days` < 0. The result has the same :attr:`~.datetime.tzinfo` attribute as the input datetime, and datetime2 - datetime1 == timedelta after. :exc:`OverflowError` is raised if datetime2.year would be smaller than :const:`MINYEAR` or larger than :const:`MAXYEAR`. Note that no time zone adjustments are done even if the input is an aware object.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1140); *backlink*

   Unknown interpreted text role "attr".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1140); *backlink*

   Unknown interpreted text role "exc".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1140); *backlink*

   Unknown interpreted text role "const".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1140); *backlink*

   Unknown interpreted text role "const".

2. Computes the datetime2 such that datetime2 + timedelta == datetime1. As for addition, the result has the same :attr:`~.datetime.tzinfo` attribute as the input datetime, and no time zone adjustments are done even if the input is aware.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1149); *backlink*

   Unknown interpreted text role "attr".

3. Subtraction of a :class:`.datetime` from a :class:`.datetime` is defined only if both operands are naive, or if both are aware. If one is aware and the other is naive, :exc:`TypeError` is raised.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1154); *backlink*

   Unknown interpreted text role "class".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1154); *backlink*

   Unknown interpreted text role "class".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1154); *backlink*

   Unknown interpreted text role "exc".

   If both are naive, or both are aware and have the same :attr:`~.datetime.tzinfo` attribute, the :attr:`~.datetime.tzinfo` attributes are ignored, and the result is a :class:`timedelta` object *t* such that `datetime2 + t == datetime1`. No time zone adjustments are done in this case.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1158); *backlink*

   Unknown interpreted text role "attr".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1158); *backlink*

   Unknown interpreted text role "attr".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1158); *backlink*

   Unknown interpreted text role "class".

   If both are aware and have different :attr:`~.datetime.tzinfo` attributes, `a-b` acts as if *a* and *b* were first converted to naive UTC datetimes first. The result is `(a.replace(tzinfo=None) - a.utcoffset()) - (b.replace(tzinfo=None) - b.utcoffset())` except that the implementation never overflows.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1163); *backlink*

   Unknown interpreted text role "attr".

4. *datetime1* is considered less than *datetime2* when *datetime1* precedes *datetime2* in time.

   If one comparand is naive and the other is aware, :exc:`TypeError` is raised if an order comparison is attempted. For equality comparisons, naive instances are never equal to aware instances.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1172); *backlink*

   Unknown interpreted text role "exc".

   If both comparands are aware, and have the same :attr:`~.datetime.tzinfo` attribute, the common :attr:`~.datetime.tzinfo` attribute is ignored and the base datetimes are compared. If both comparands are aware and have different

:attr:`~.datetime.tzinfo` attributes, the comparands are first adjusted by subtracting their UTC offsets (obtained from `self.utcoffset()`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1176); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1176); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1176); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1182)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   Equality comparisons between aware and naive :class:`.datetime`
   instances don't raise :exc:`TypeError`.
```

Note

In order to stop comparison from falling back to the default scheme of comparing object addresses, datetime comparison normally raises :exc:`TypeError` if the other comparand isn't also a :class:`.datetime` object. However, `NotImplemented` is returned instead if the other comparand has a :meth:`timetuple` attribute. This hook gives other kinds of date objects a chance at implementing mixed-type comparison. If not, when a :class:`.datetime` object is compared to an object of a different type, :exc:`TypeError` is raised unless the comparison is `==` or `!=`. The latter cases return :const:`False` or :const:`True`, respectively.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1188); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1188); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1188); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1188); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1188); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1188); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1188); *backlink*

Unknown interpreted text role "const".

Instance methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1200)

Unknown directive type "method".

```
.. method:: datetime.date()

   Return :class:`date` object with same year, month and day.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1205)

Unknown directive type "method".

```
.. method:: datetime.time()

   Return :class:`.time` object with same hour, minute, second, microsecond and fold.
   :attr:`.tzinfo` is ``None``. See also method :meth:`timetz`.

   .. versionchanged:: 3.6
      The fold value is copied to the returned :class:`.time` object.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1214)

Unknown directive type "method".

```
.. method:: datetime.timetz()

   Return :class:`.time` object with same hour, minute, second, microsecond, fold, and
   tzinfo attributes. See also method :meth:`time`.

   .. versionchanged:: 3.6
      The fold value is copied to the returned :class:`.time` object.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1223)

Unknown directive type "method".

```
.. method:: datetime.replace(year=self.year, month=self.month, day=self.day, \
   hour=self.hour, minute=self.minute, second=self.second, microsecond=self.microsecond, \
   tzinfo=self.tzinfo, *, fold=0)

   Return a datetime with the same attributes, except for those attributes given
   new values by whichever keyword arguments are specified. Note that
   ``tzinfo=None`` can be specified to create a naive datetime from an aware
   datetime with no conversion of date and time data.

   .. versionadded:: 3.6
      Added the ``fold`` argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1236)

Unknown directive type "method".

```
.. method:: datetime.astimezone(tz=None)

   Return a :class:`.datetime` object with new :attr:`.tzinfo` attribute *tz*,
   adjusting the date and time data so the result is the same UTC time as
   *self*, but in *tz*'s local time.

   If provided, *tz* must be an instance of a :class:`tzinfo` subclass, and its
   :meth:`utcoffset` and :meth:`dst` methods must not return ``None``. If *self*
   is naive, it is presumed to represent time in the system timezone.

   If called without arguments (or with ``tz=None``) the system local
```

```
timezone is assumed for the target timezone. The ``.tzinfo`` attribute of the converted
datetime instance will be set to an instance of :class:`timezone`
with the zone name and offset obtained from the OS.

If ``self.tzinfo`` is *tz*, ``self.astimezone(tz)`` is equal to *self*:  no
adjustment of date or time data is performed. Else the result is local
time in the timezone *tz*, representing the same UTC time as *self*:  after
``astz = dt.astimezone(tz)``, ``astz - astz.utcoffset()`` will have
the same date and time data as ``dt - dt.utcoffset()``.

If you merely want to attach a time zone object *tz* to a datetime *dt* without
adjustment of date and time data, use ``dt.replace(tzinfo=tz)``. If you
merely want to remove the time zone object from an aware datetime *dt* without
conversion of date and time data, use ``dt.replace(tzinfo=None)``.

Note that the default :meth:`tzinfo.fromutc` method can be overridden in a
:class:`tzinfo` subclass to affect the result returned by :meth:`astimezone`.
Ignoring error cases, :meth:`astimezone` acts like::

   def astimezone(self, tz):
       if self.tzinfo is tz:
           return self
       # Convert self to UTC, and attach the new time zone object.
       utc = (self - self.utcoffset()).replace(tzinfo=tz)
       # Convert from UTC to tz's local time.
       return tz.fromutc(utc)

.. versionchanged:: 3.3
   *tz* now can be omitted.

.. versionchanged:: 3.6
   The :meth:`astimezone` method can now be called on naive instances that
   are presumed to represent system local time.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1282)

Unknown directive type "method".

```
.. method:: datetime.utcoffset()

   If :attr:`.tzinfo` is ``None``, returns ``None``, else returns
   ``self.tzinfo.utcoffset(self)``, and raises an exception if the latter doesn't
   return ``None`` or a :class:`timedelta` object with magnitude less than one day.

   .. versionchanged:: 3.7
      The UTC offset is not restricted to a whole number of minutes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1292)

Unknown directive type "method".

```
.. method:: datetime.dst()

   If :attr:`.tzinfo` is ``None``, returns ``None``, else returns
   ``self.tzinfo.dst(self)``, and raises an exception if the latter doesn't return
   ``None`` or a :class:`timedelta` object with magnitude less than one day.

   .. versionchanged:: 3.7
      The DST offset is not restricted to a whole number of minutes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1302)

Unknown directive type "method".

```
.. method:: datetime.tzname()

   If :attr:`.tzinfo` is ``None``, returns ``None``, else returns
   ``self.tzinfo.tzname(self)``, raises an exception if the latter doesn't return
   ``None`` or a string object,
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1309)

Unknown directive type "method".

```
.. method:: datetime.timetuple()

   Return a :class:`time.struct_time` such as returned by :func:`time.localtime`.
```

```
``d.timetuple()`` is equivalent to::

  time.struct_time((d.year, d.month, d.day,
                    d.hour, d.minute, d.second,
                    d.weekday(), yday, dst))

where ``yday = d.toordinal() - date(d.year, 1, 1).toordinal() + 1``
is the day number within the current year starting with ``1`` for January
1st. The :attr:`tm_isdst` flag of the result is set according to the
:meth:`dst` method: :attr:`.tzinfo` is ``None`` or :meth:`dst` returns
``None``, :attr:`tm_isdst` is set to ``-1``; else if :meth:`dst` returns a
non-zero value, :attr:`tm_isdst` is set to ``1``; else :attr:`tm_isdst` is
set to ``0``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1328)

Unknown directive type "method".

```
.. method:: datetime.utctimetuple()

   If :class:`.datetime` instance *d* is naive, this is the same as
   ``d.timetuple()`` except that :attr:`tm_isdst` is forced to 0 regardless of what
   ``d.dst()`` returns. DST is never in effect for a UTC time.

   If *d* is aware, *d* is normalized to UTC time, by subtracting
   ``d.utcoffset()``, and a :class:`time.struct_time` for the
   normalized time is returned. :attr:`tm_isdst` is forced to 0. Note
   that an :exc:`OverflowError` may be raised if *d*.year was
   ``MINYEAR`` or ``MAXYEAR`` and UTC adjustment spills over a year
   boundary.

   .. warning::

      Because naive ``datetime`` objects are treated by many ``datetime`` methods
      as local times, it is preferred to use aware datetimes to represent times
      in UTC; as a result, using ``utcfromtimetuple`` may give misleading
      results. If you have a naive ``datetime`` representing UTC, use
      ``datetime.replace(tzinfo=timezone.utc)`` to make it aware, at which point
      you can use :meth:`.datetime.timetuple`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1350)

Unknown directive type "method".

```
.. method:: datetime.toordinal()

   Return the proleptic Gregorian ordinal of the date. The same as
   ``self.date().toordinal()``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1355)

Unknown directive type "method".

```
.. method:: datetime.timestamp()

   Return POSIX timestamp corresponding to the :class:`.datetime`
   instance. The return value is a :class:`float` similar to that
   returned by :func:`time.time`.

   Naive :class:`.datetime` instances are assumed to represent local
   time and this method relies on the platform C :c:func:`mktime`
   function to perform the conversion. Since :class:`.datetime`
   supports wider range of values than :c:func:`mktime` on many
   platforms, this method may raise :exc:`OverflowError` for times far
   in the past or far in the future.

   For aware :class:`.datetime` instances, the return value is computed
   as::

      (dt - datetime(1970, 1, 1, tzinfo=timezone.utc)).total_seconds()

   .. versionadded:: 3.3

   .. versionchanged:: 3.6
      The :meth:`timestamp` method uses the :attr:`.fold` attribute to
      disambiguate the times during a repeated interval.

   .. note::

      There is no method to obtain the POSIX timestamp directly from a
      naive :class:`.datetime` instance representing UTC time. If your
      application uses this convention and your system timezone is not
```

```
   set to UTC, you can obtain the POSIX timestamp by supplying
   ``tzinfo=timezone.utc``::

      timestamp = dt.replace(tzinfo=timezone.utc).timestamp()

   or by calculating the timestamp directly::

      timestamp = (dt - datetime(1970, 1, 1)) / timedelta(seconds=1)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1393)

Unknown directive type "method".

```
.. method:: datetime.weekday()

   Return the day of the week as an integer, where Monday is 0 and Sunday is 6.
   The same as ``self.date().weekday()``. See also :meth:`isoweekday`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1399)

Unknown directive type "method".

```
.. method:: datetime.isoweekday()

   Return the day of the week as an integer, where Monday is 1 and Sunday is 7.
   The same as ``self.date().isoweekday()``. See also :meth:`weekday`,
   :meth:`isocalendar`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1406)

Unknown directive type "method".

```
.. method:: datetime.isocalendar()

   Return a :term:`named tuple` with three components: ``year``, ``week``
   and ``weekday``. The same as ``self.date().isocalendar()``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1412)

Unknown directive type "method".

```
.. method:: datetime.isoformat(sep='T', timespec='auto')

   Return a string representing the date and time in ISO 8601 format:

   - ``YYYY-MM-DDTHH:MM:SS.ffffff``, if :attr:`microsecond` is not 0
   - ``YYYY-MM-DDTHH:MM:SS``, if :attr:`microsecond` is 0

   If :meth:`utcoffset` does not return ``None``, a string is
   appended, giving the UTC offset:

   - ``YYYY-MM-DDTHH:MM:SS.ffffff+HH:MM[:SS[.ffffff]]``, if :attr:`microsecond`
     is not 0
   - ``YYYY-MM-DDTHH:MM:SS+HH:MM[:SS[.ffffff]]``,  if :attr:`microsecond` is 0

   Examples::

       >>> from datetime import datetime, timezone
       >>> datetime(2019, 5, 18, 15, 17, 8, 132263).isoformat()
       '2019-05-18T15:17:08.132263'
       >>> datetime(2019, 5, 18, 15, 17, tzinfo=timezone.utc).isoformat()
       '2019-05-18T15:17:00+00:00'

   The optional argument *sep* (default ``'T'``) is a one-character separator,
   placed between the date and time portions of the result. For example::

      >>> from datetime import tzinfo, timedelta, datetime
      >>> class TZ(tzinfo):
      ...     """A time zone with an arbitrary, constant -06:39 offset."""
      ...     def utcoffset(self, dt):
      ...         return timedelta(hours=-6, minutes=-39)
      ...
      >>> datetime(2002, 12, 25, tzinfo=TZ()).isoformat(' ')
      '2002-12-25 00:00:00-06:39'
      >>> datetime(2009, 11, 27, microsecond=100, tzinfo=TZ()).isoformat()
      '2009-11-27T00:00:00.000100-06:39'

   The optional argument *timespec* specifies the number of additional
```

```
   components of the time to include (the default is ``'auto'``).
   It can be one of the following:

   - ``'auto'``: Same as ``'seconds'`` if :attr:`microsecond` is 0,
     same as ``'microseconds'`` otherwise.
   - ``'hours'``: Include the :attr:`hour` in the two-digit ``HH`` format.
   - ``'minutes'``: Include :attr:`hour` and :attr:`minute` in ``HH:MM`` format.
   - ``'seconds'``: Include :attr:`hour`, :attr:`minute`, and :attr:`second`
     in ``HH:MM:SS`` format.
   - ``'milliseconds'``: Include full time, but truncate fractional second
     part to milliseconds. ``HH:MM:SS.sss`` format.
   - ``'microseconds'``: Include full time in ``HH:MM:SS.ffffff`` format.

   .. note::

      Excluded time components are truncated, not rounded.

   :exc:`ValueError` will be raised on an invalid *timespec* argument::


      >>> from datetime import datetime
      >>> datetime.now().isoformat(timespec='minutes')   # doctest: +SKIP
      '2002-12-25T00:00'
      >>> dt = datetime(2015, 1, 1, 12, 30, 59, 0)
      >>> dt.isoformat(timespec='microseconds')
      '2015-01-01T12:30:59.000000'

   .. versionadded:: 3.6
      Added the *timespec* argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1480)

Unknown directive type "method".

```
.. method:: datetime.__str__()

   For a :class:`.datetime` instance *d*, ``str(d)`` is equivalent to
   ``d.isoformat(' ')``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1486)

Unknown directive type "method".

```
.. method:: datetime.ctime()

   Return a string representing the date and time::

       >>> from datetime import datetime
       >>> datetime(2002, 12, 4, 20, 30, 40).ctime()
       'Wed Dec  4 20:30:40 2002'

   The output string will *not* include time zone information, regardless
   of whether the input is aware or naive.

   ``d.ctime()`` is equivalent to::

     time.ctime(time.mktime(d.timetuple()))

   on platforms where the native C :c:func:`ctime` function
   (which :func:`time.ctime` invokes, but which
   :meth:`datetime.ctime` does not invoke) conforms to the C standard.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1505)

Unknown directive type "method".

```
.. method:: datetime.strftime(format)

   Return a string representing the date and time, controlled by an explicit format
   string. For a complete list of formatting directives, see
   :ref:`strftime-strptime-behavior`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1512)

Unknown directive type "method".

```
.. method:: datetime.__format__(format)
```

```
Same as :meth:`.datetime.strftime`. This makes it possible to specify a format
string for a :class:`.datetime` object in :ref:`formatted string
literals <f-strings>` and when using :meth:`str.format`. For a
complete list of formatting directives, see
:ref:`strftime-strptime-behavior`.
```

Examples of Usage: :class:`.datetime`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1520); *backlink*

Unknown interpreted text role "class".

Examples of working with :class:`~datetime.datetime` objects:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1523); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1525)

Unknown directive type "doctest".

```
.. doctest::

    >>> from datetime import datetime, date, time, timezone

    >>> # Using datetime.combine()
    >>> d = date(2005, 7, 14)
    >>> t = time(12, 30)
    >>> datetime.combine(d, t)
    datetime.datetime(2005, 7, 14, 12, 30)

    >>> # Using datetime.now()
    >>> datetime.now()   # doctest: +SKIP
    datetime.datetime(2007, 12, 6, 16, 29, 43, 79043)   # GMT +1
    >>> datetime.now(timezone.utc)   # doctest: +SKIP
    datetime.datetime(2007, 12, 6, 15, 29, 43, 79060, tzinfo=datetime.timezone.utc)

    >>> # Using datetime.strptime()
    >>> dt = datetime.strptime("21/11/06 16:30", "%d/%m/%y %H:%M")
    >>> dt
    datetime.datetime(2006, 11, 21, 16, 30)

    >>> # Using datetime.timetuple() to get tuple of all attributes
    >>> tt = dt.timetuple()
    >>> for it in tt:   # doctest: +SKIP
    ...     print(it)
    ...
    2006    # year
    11      # month
    21      # day
    16      # hour
    30      # minute
    0       # second
    1       # weekday (0 = Monday)
    325     # number of days since 1st January
    -1      # dst - method tzinfo.dst() returned None

    >>> # Date in ISO format
    >>> ic = dt.isocalendar()
    >>> for it in ic:   # doctest: +SKIP
    ...     print(it)
    ...
    2006    # ISO year
    47      # ISO week
    2       # ISO weekday

    >>> # Formatting a datetime
    >>> dt.strftime("%A, %d. %B %Y %I:%M%p")
    'Tuesday, 21. November 2006 04:30PM'
    >>> 'The {1} is {0:%d}, the {2} is {0:%B}, the {3} is {0:%I:%M%p}.'.format(dt, "day", "month", "time"
    'The day is 21, the month is November, the time is 04:30PM.'
```

The example below defines a :class:`tzinfo` subclass capturing time zone information for Kabul, Afghanistan, which used +4 UTC until 1945 and then +4:30 UTC thereafter:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1576); *backlink*

Unknown interpreted text role "class".

```
from datetime import timedelta, datetime, tzinfo, timezone

class KabulTz(tzinfo):
    # Kabul used +4 until 1945, when they moved to +4:30
    UTC_MOVE_DATE = datetime(1944, 12, 31, 20, tzinfo=timezone.utc)

    def utcoffset(self, dt):
        if dt.year < 1945:
            return timedelta(hours=4)
        elif (1945, 1, 1, 0, 0) <= dt.timetuple()[:5] < (1945, 1, 1, 0, 30):
            # An ambiguous ("imaginary") half-hour range representing
            # a 'fold' in time due to the shift from +4 to +4:30.
            # If dt falls in the imaginary range, use fold to decide how
            # to resolve. See PEP495.
            return timedelta(hours=4, minutes=(30 if dt.fold else 0))
        else:
            return timedelta(hours=4, minutes=30)

    def fromutc(self, dt):
        # Follow same validations as in datetime.tzinfo
        if not isinstance(dt, datetime):
            raise TypeError("fromutc() requires a datetime argument")
        if dt.tzinfo is not self:
            raise ValueError("dt.tzinfo is not self")

        # A custom implementation is required for fromutc as
        # the input to this function is a datetime with utc values
        # but with a tzinfo set to self.
        # See datetime.astimezone or fromtimestamp.
        if dt.replace(tzinfo=timezone.utc) >= self.UTC_MOVE_DATE:
            return dt + timedelta(hours=4, minutes=30)
        else:
            return dt + timedelta(hours=4)

    def dst(self, dt):
        # Kabul does not observe daylight saving time.
        return timedelta(0)

    def tzname(self, dt):
        if dt >= self.UTC_MOVE_DATE:
            return "+04:30"
        return "+04"
```

Usage of `KabulTz` from above:

```
>>> tz1 = KabulTz()

>>> # Datetime before the change
>>> dt1 = datetime(1900, 11, 21, 16, 30, tzinfo=tz1)
>>> print(dt1.utcoffset())
4:00:00

>>> # Datetime after the change
>>> dt2 = datetime(2006, 6, 14, 13, 0, tzinfo=tz1)
>>> print(dt2.utcoffset())
4:30:00

>>> # Convert datetime to another time zone
>>> dt3 = dt2.astimezone(timezone.utc)
>>> dt3
datetime.datetime(2006, 6, 14, 8, 30, tzinfo=datetime.timezone.utc)
>>> dt2
datetime.datetime(2006, 6, 14, 13, 0, tzinfo=KabulTz())
>>> dt2 == dt3
True
```

:class:`.time` Objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1648); *backlink*

Unknown interpreted text role "class".

A :class:`time` object represents a (local) time of day, independent of any particular day, and subject to adjustment via a :class:`tzinfo` object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1651); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1651); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1654)

Invalid class attribute value for "class" directive: "time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None, *, fold=0)".

```
.. class:: time(hour=0, minute=0, second=0, microsecond=0, tzinfo=None, *, fold=0)

   All arguments are optional. *tzinfo* may be ``None``, or an instance of a
   :class:`tzinfo` subclass. The remaining arguments must be integers in the
   following ranges:

   * ``0 <= hour < 24``,
   * ``0 <= minute < 60``,
   * ``0 <= second < 60``,
   * ``0 <= microsecond < 1000000``,
   * ``fold in [0, 1]``.

   If an argument outside those ranges is given, :exc:`ValueError` is raised. All
   default to ``0`` except *tzinfo*, which defaults to :const:`None`.
```

Class attributes:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1672)

Unknown directive type "attribute".

```
.. attribute:: time.min

   The earliest representable :class:`.time`, ``time(0, 0, 0, 0)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1677)

Unknown directive type "attribute".

```
.. attribute:: time.max

   The latest representable :class:`.time`, ``time(23, 59, 59, 999999)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1682)

Unknown directive type "attribute".

```
.. attribute:: time.resolution

   The smallest possible difference between non-equal :class:`.time` objects,
   ``timedelta(microseconds=1)``, although note that arithmetic on
   :class:`.time` objects is not supported.
```

Instance attributes (read-only):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1691)

Unknown directive type "attribute".

```
.. attribute:: time.hour

   In ``range(24)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1696)

Unknown directive type "attribute".

```
.. attribute:: time.minute

   In ``range(60)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1701)

Unknown directive type "attribute".

```
.. attribute:: time.second

   In ``range(60)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1706)

Unknown directive type "attribute".

```
.. attribute:: time.microsecond

   In ``range(1000000)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1711)

Unknown directive type "attribute".

```
.. attribute:: time.tzinfo

   The object passed as the tzinfo argument to the :class:`.time` constructor, or
   ``None`` if none was passed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1717)

Unknown directive type "attribute".

```
.. attribute:: time.fold

   In ``[0, 1]``. Used to disambiguate wall times during a repeated interval. (A
   repeated interval occurs when clocks are rolled back at the end of daylight saving
   time or when the UTC offset for the current zone is decreased for political reasons.)
   The value 0 (1) represents the earlier (later) of the two moments with the same wall
   time representation.

   .. versionadded:: 3.6
```

:class:`.time` objects support comparison of :class:`.time` to :class:`.time`, where *a* is considered less than *b* when *a* precedes *b* in time. If one comparand is naive and the other is aware, :exc:`TypeError` is raised if an order comparison is attempted. For equality comparisons, naive instances are never equal to aware instances.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1727); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1727); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1727); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1727); *backlink*

Unknown interpreted text role "exc".

If both comparands are aware, and have the same :attr:`~time.tzinfo` attribute, the common :attr:`~time.tzinfo` attribute is ignored and the base times are compared. If both comparands are aware and have different :attr:`~time.tzinfo` attributes, the comparands are first adjusted by subtracting their UTC offsets (obtained from `self.utcoffset()`). In order to stop mixed-type comparisons from falling back to the default comparison by object address, when a :class:`.time` object is compared to an object of a different type, :exc:`TypeError` is raised unless the comparison is `==` or `!=`. The latter cases return :const:`False` or :const:`True`, respectively.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1733); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1733); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1733); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1733); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1733); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1733); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1733); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1743)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
  Equality comparisons between aware and naive :class:`~datetime.time` instances
  don't raise :exc:`TypeError`.
```

In Boolean contexts, a :class:`.time` object is always considered to be true.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1747); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1749)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   Before Python 3.5, a :class:`.time` object was considered to be false if it
   represented midnight in UTC. This behavior was considered obscure and
   error-prone and has been removed in Python 3.5. See :issue:`13936` for full
   details.
```

Other constructor:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1758)

Unknown directive type "classmethod".

```
.. classmethod:: time.fromisoformat(time_string)

   Return a :class:`.time` corresponding to a *time_string* in one of the
   formats emitted by :meth:`time.isoformat`. Specifically, this function supports
   strings in the format:

   .. code-block:: none

      HH[:MM[:SS[.fff[fff]]]][+HH:MM[:SS[.ffffff]]]

   .. caution::

     This does *not* support parsing arbitrary ISO 8601 strings. It is only
```

```
      intended as the inverse operation of :meth:`time.isoformat`.

   Examples::

       >>> from datetime import time
       >>> time.fromisoformat('04:23:01')
       datetime.time(4, 23, 1)
       >>> time.fromisoformat('04:23:01.000384')
       datetime.time(4, 23, 1, 384)
       >>> time.fromisoformat('04:23:01+04:00')
       datetime.time(4, 23, 1, tzinfo=datetime.timezone(datetime.timedelta(seconds=14400)))

   .. versionadded:: 3.7
```

Instance methods:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1788)

Unknown directive type "method".

```
.. method:: time.replace(hour=self.hour, minute=self.minute, second=self.second, \
   microsecond=self.microsecond, tzinfo=self.tzinfo, *, fold=0)

   Return a :class:`.time` with the same value, except for those attributes given
   new values by whichever keyword arguments are specified. Note that
   ``tzinfo=None`` can be specified to create a naive :class:`.time` from an
   aware :class:`.time`, without conversion of the time data.

   .. versionadded:: 3.6
      Added the ``fold`` argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1800)

Unknown directive type "method".

```
.. method:: time.isoformat(timespec='auto')

   Return a string representing the time in ISO 8601 format, one of:

   - ``HH:MM:SS.ffffff``, if :attr:`microsecond` is not 0
   - ``HH:MM:SS``, if :attr:`microsecond` is 0
   - ``HH:MM:SS.ffffff+HH:MM[:SS[.ffffff]]``, if :meth:`utcoffset` does not return ``None``
   - ``HH:MM:SS+HH:MM[:SS[.ffffff]]``, if :attr:`microsecond` is 0 and :meth:`utcoffset` does not return

   The optional argument *timespec* specifies the number of additional
   components of the time to include (the default is ``'auto'``).
   It can be one of the following:

   - ``'auto'``: Same as ``'seconds'`` if :attr:`microsecond` is 0,
     same as ``'microseconds'`` otherwise.
   - ``'hours'``: Include the :attr:`hour` in the two-digit ``HH`` format.
   - ``'minutes'``: Include :attr:`hour` and :attr:`minute` in ``HH:MM`` format.
   - ``'seconds'``: Include :attr:`hour`, :attr:`minute`, and :attr:`second`
     in ``HH:MM:SS`` format.
   - ``'milliseconds'``: Include full time, but truncate fractional second
     part to milliseconds. ``HH:MM:SS.sss`` format.
   - ``'microseconds'``: Include full time in ``HH:MM:SS.ffffff`` format.

   .. note::

      Excluded time components are truncated, not rounded.

   :exc:`ValueError` will be raised on an invalid *timespec* argument.

   Example::

      >>> from datetime import time
      >>> time(hour=12, minute=34, second=56, microsecond=123456).isoformat(timespec='minutes')
      '12:34'
      >>> dt = time(hour=12, minute=34, second=56, microsecond=0)
      >>> dt.isoformat(timespec='microseconds')
      '12:34:56.000000'
      >>> dt.isoformat(timespec='auto')
      '12:34:56'

   .. versionadded:: 3.6
      Added the *timespec* argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1844)

Unknown directive type "method".

```
.. method:: time.__str__()

   For a time *t*, ``str(t)`` is equivalent to ``t.isoformat()``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1849)

Unknown directive type "method".

```
.. method:: time.strftime(format)

   Return a string representing the time, controlled by an explicit format
   string. For a complete list of formatting directives, see
   :ref:`strftime-strptime-behavior`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1856)

Unknown directive type "method".

```
.. method:: time.__format__(format)

   Same as :meth:`.time.strftime`. This makes it possible to specify a format string
   for a :class:`.time` object in :ref:`formatted string
   literals <f-strings>` and when using :meth:`str.format`. For a
   complete list of formatting directives, see
   :ref:`strftime-strptime-behavior`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1865)

Unknown directive type "method".

```
.. method:: time.utcoffset()

   If :attr:`.tzinfo` is ``None``, returns ``None``, else returns
   ``self.tzinfo.utcoffset(None)``, and raises an exception if the latter doesn't
   return ``None`` or a :class:`timedelta` object with magnitude less than one day.

   .. versionchanged:: 3.7
      The UTC offset is not restricted to a whole number of minutes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1875)

Unknown directive type "method".

```
.. method:: time.dst()

   If :attr:`.tzinfo` is ``None``, returns ``None``, else returns
   ``self.tzinfo.dst(None)``, and raises an exception if the latter doesn't return
   ``None``, or a :class:`timedelta` object with magnitude less than one day.

   .. versionchanged:: 3.7
      The DST offset is not restricted to a whole number of minutes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1884)

Unknown directive type "method".

```
.. method:: time.tzname()

   If :attr:`.tzinfo` is ``None``, returns ``None``, else returns
   ``self.tzinfo.tzname(None)``, or raises an exception if the latter doesn't
   return ``None`` or a string object.
```

Examples of Usage: :class:`.time`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1890); *backlink*

Unknown interpreted text role "class".

Examples of working with a :class:`.time` object:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1893); *backlink*

Unknown interpreted text role "class".

```
>>> from datetime import time, tzinfo, timedelta
>>> class TZ1(tzinfo):
...     def utcoffset(self, dt):
...         return timedelta(hours=1)
...     def dst(self, dt):
...         return timedelta(0)
...     def tzname(self,dt):
...         return "+01:00"
...     def  __repr__(self):
...         return f"{self.__class__.__name__}()"
...
>>> t = time(12, 10, 30, tzinfo=TZ1())
>>> t
datetime.time(12, 10, 30, tzinfo=TZ1())
>>> t.isoformat()
'12:10:30+01:00'
>>> t.dst()
datetime.timedelta(0)
>>> t.tzname()
'+01:00'
>>> t.strftime("%H:%M:%S %Z")
'12:10:30 +01:00'
>>> 'The {} is {:%H:%M}.'.format("time", t)
'The time is 12:10.'
```

:class:`tzinfo` Objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1923); *backlink*

Unknown interpreted text role "class".

This is an abstract base class, meaning that this class should not be instantiated directly. Define a subclass of :class:`tzinfo` to capture information about a particular time zone.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1928); *backlink*

Unknown interpreted text role "class".

An instance of (a concrete subclass of) :class:`tzinfo` can be passed to the constructors for :class:`.datetime` and :class:`.time` objects. The latter objects view their attributes as being in local time, and the :class:`tzinfo` object supports methods revealing offset of local time from UTC, the name of the time zone, and DST offset, all relative to a date or time object passed to them.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1932); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1932); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1932); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1932); *backlink*

Unknown interpreted text role "class".

You need to derive a concrete subclass, and (at least) supply implementations of the standard :class:`tzinfo` methods needed by the :class:`.datetime` methods you use. The :mod:`datetime` module provides :class:`timezone`, a simple concrete subclass of :class:`tzinfo` which can represent timezones with fixed offset from UTC such as UTC itself or North American EST and EDT.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1938); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1938); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1938); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1938); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1938); *backlink*

Unknown interpreted text role "class".

Special requirement for pickling: A :class:`tzinfo` subclass must have an :meth:`__init__` method that can be called with no arguments, otherwise it can be pickled but possibly not unpickled again. This is a technical requirement that may be relaxed in the future.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1945); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1945); *backlink*

Unknown interpreted text role "meth".

A concrete subclass of :class:`tzinfo` may need to implement the following methods. Exactly which methods are needed depends on the uses made of aware :mod:`datetime` objects. If in doubt, simply implement all of them.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1950); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1950); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1955)

Unknown directive type "method".

```
.. method:: tzinfo.utcoffset(dt)

   Return offset of local time from UTC, as a :class:`timedelta` object that is
   positive east of UTC. If local time is west of UTC, this should be negative.

   This represents the *total* offset from UTC; for example, if a
   :class:`tzinfo` object represents both time zone and DST adjustments,
   :meth:`utcoffset` should return their sum. If the UTC offset isn't known,
   return ``None``. Else the value returned must be a :class:`timedelta` object
   strictly between ``-timedelta(hours=24)`` and ``timedelta(hours=24)``
   (the magnitude of the offset must be less than one day). Most implementations
   of :meth:`utcoffset` will probably look like one of these two::

      return CONSTANT                 # fixed-offset class
      return CONSTANT + self.dst(dt)  # daylight-aware class

   If :meth:`utcoffset` does not return ``None``, :meth:`dst` should not return
   ``None`` either.

   The default implementation of :meth:`utcoffset` raises
   :exc:`NotImplementedError`.

   .. versionchanged:: 3.7
      The UTC offset is not restricted to a whole number of minutes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 1981)

Unknown directive type "method".

```
.. method:: tzinfo.dst(dt)

   Return the daylight saving time (DST) adjustment, as a :class:`timedelta`
   object or
   ``None`` if DST information isn't known.

   Return ``timedelta(0)`` if DST is not in effect.
   If DST is in effect, return the offset as a :class:`timedelta` object
   (see :meth:`utcoffset` for details). Note that DST offset, if applicable, has
   already been added to the UTC offset returned by :meth:`utcoffset`, so there's
   no need to consult :meth:`dst` unless you're interested in obtaining DST info
   separately. For example, :meth:`datetime.timetuple` calls its :attr:`~.datetime.tzinfo`
   attribute's :meth:`dst` method to determine how the :attr:`tm_isdst` flag
   should be set, and :meth:`tzinfo.fromutc` calls :meth:`dst` to account for
   DST changes when crossing time zones.

   An instance *tz* of a :class:`tzinfo` subclass that models both standard and
   daylight times must be consistent in this sense:

   ``tz.utcoffset(dt) - tz.dst(dt)``

   must return the same result for every :class:`.datetime` *dt* with ``dt.tzinfo ==
   tz``  For sane :class:`tzinfo` subclasses, this expression yields the time
   zone's "standard offset", which should not depend on the date or the time, but
   only on geographic location. The implementation of :meth:`datetime.astimezone`
   relies on this, but cannot detect violations; it's the programmer's
   responsibility to ensure it. If a :class:`tzinfo` subclass cannot guarantee
   this, it may be able to override the default implementation of
   :meth:`tzinfo.fromutc` to work correctly with :meth:`astimezone` regardless.

   Most implementations of :meth:`dst` will probably look like one of these two::

      def dst(self, dt):
          # a fixed-offset class:  doesn't account for DST
          return timedelta(0)

   or::

      def dst(self, dt):
          # Code to set dston and dstoff to the time zone's DST
          # transition times based on the input dt.year, and expressed
          # in standard local time.

          if dston <= dt.replace(tzinfo=None) < dstoff:
              return timedelta(hours=1)
          else:
              return timedelta(0)

   The default implementation of :meth:`dst` raises :exc:`NotImplementedError`.

   .. versionchanged:: 3.7
      The DST offset is not restricted to a whole number of minutes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2035)

Unknown directive type "method".

```
.. method:: tzinfo.tzname(dt)

   Return the time zone name corresponding to the :class:`.datetime` object *dt*, as
   a string. Nothing about string names is defined by the :mod:`datetime` module,
   and there's no requirement that it mean anything in particular. For example,
   "GMT", "UTC", "-500", "-5:00", "EDT", "US/Eastern", "America/New York" are all
   valid replies. Return ``None`` if a string name isn't known. Note that this is
   a method rather than a fixed string primarily because some :class:`tzinfo`
   subclasses will wish to return different names depending on the specific value
   of *dt* passed, especially if the :class:`tzinfo` class is accounting for
   daylight time.

   The default implementation of :meth:`tzname` raises :exc:`NotImplementedError`.
```

These methods are called by a :class:`.datetime` or :class:`.time` object, in response to their methods of the same names. A :class:`.datetime` object passes itself as the argument, and a :class:`.time` object passes `None` as the argument. A :class:`tzinfo` subclass's methods should therefore be prepared to accept a *dt* argument of `None`, or of class :class:`.datetime`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2050); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2050); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2050); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2050); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2050); *backlink*

Unknown interpreted text role "class".

When `None` is passed, it's up to the class designer to decide the best response. For example, returning `None` is appropriate if the class wishes to say that time objects don't participate in the :class:`tzinfo` protocols. It may be more useful for `utcoffset(None)` to return the standard UTC offset, as there is no other convention for discovering the standard offset.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2056); *backlink*

Unknown interpreted text role "class".

When a :class:`.datetime` object is passed in response to a :class:`.datetime` method, `dt.tzinfo` is the same object as *self*. :class:`tzinfo` methods can rely on this, unless user code calls :class:`tzinfo` methods directly. The intent is that the :class:`tzinfo` methods interpret *dt* as being in local time, and not need worry about objects in other timezones.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2062); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2062); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2062); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2062); *backlink*

Unknown interpreted text role "class".

There is one more :class:`tzinfo` method that a subclass may wish to override:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2068); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2071)

Unknown directive type "method".

```
.. method:: tzinfo.fromutc(dt)

   This is called from the default :class:`datetime.astimezone()`
   implementation. When called from that, ``dt.tzinfo`` is *self*, and *dt*'s
   date and time data are to be viewed as expressing a UTC time. The purpose
   of :meth:`fromutc` is to adjust the date and time data, returning an
   equivalent datetime in *self*'s local time.

   Most :class:`tzinfo` subclasses should be able to inherit the default
   :meth:`fromutc` implementation without problems. It's strong enough to handle
   fixed-offset time zones, and time zones accounting for both standard and
   daylight time, and the latter even if the DST transition times differ in
   different years. An example of a time zone the default :meth:`fromutc`
   implementation may not handle correctly in all cases is one where the standard
   offset (from UTC) depends on the specific date and time passed, which can happen
   for political reasons. The default implementations of :meth:`astimezone` and
   :meth:`fromutc` may not produce the result you want if the result is one of the
   hours straddling the moment the standard offset changes.

   Skipping code for error cases, the default :meth:`fromutc` implementation acts
   like::

      def fromutc(self, dt):
          # raise ValueError error if dt.tzinfo is not self
          dtoff = dt.utcoffset()
          dtdst = dt.dst()
          # raise ValueError if dtoff is None or dtdst is None
          delta = dtoff - dtdst  # this is self's standard offset
          if delta:
              dt += delta   # convert to standard local time
              dtdst = dt.dst()
              # raise ValueError if dtdst is None
          if dtdst:
              return dt + dtdst
          else:
              return dt
```

In the following :download:`tzinfo_examples.py <../includes/tzinfo_examples.py>` file there are some examples of :class:`tzinfo` classes:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2108); *backlink*

Unknown interpreted text role "download".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2108); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2112)

Unknown directive type "literalinclude".

```
.. literalinclude:: ../includes/tzinfo_examples.py
```

Note that there are unavoidable subtleties twice per year in a :class:`tzinfo` subclass accounting for both standard and daylight time, at the DST transition points. For concreteness, consider US Eastern (UTC -0500), where EDT begins the minute after 1:59 (EST) on the second Sunday in March, and ends the minute after 1:59 (EDT) on the first Sunday in November:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2114); *backlink*

Unknown interpreted text role "class".

```
  UTC   3:MM  4:MM  5:MM  6:MM  7:MM  8:MM
  EST  22:MM 23:MM  0:MM  1:MM  2:MM  3:MM
  EDT  23:MM  0:MM  1:MM  2:MM  3:MM  4:MM

start  22:MM 23:MM  0:MM  1:MM  3:MM  4:MM

  end  23:MM  0:MM  1:MM  1:MM  2:MM  3:MM
```

When DST starts (the "start" line), the local wall clock leaps from 1:59 to 3:00. A wall time of the form 2:MM doesn't really make sense on that day, so `astimezone(Eastern)` won't deliver a result with `hour == 2` on the day DST begins. For example, at the Spring forward transition of 2016, we get:

```
>>> from datetime import datetime, timezone
>>> from tzinfo_examples import HOUR, Eastern
>>> u0 = datetime(2016, 3, 13, 5, tzinfo=timezone.utc)
>>> for i in range(4):
...     u = u0 + i*HOUR
...     t = u.astimezone(Eastern)
```

```
...     print(u.time(), 'UTC =', t.time(), t.tzname())
...
05:00:00 UTC = 00:00:00 EST
06:00:00 UTC = 01:00:00 EST
07:00:00 UTC = 03:00:00 EDT
08:00:00 UTC = 04:00:00 EDT
```

When DST ends (the "end" line), there's a potentially worse problem: there's an hour that can't be spelled unambiguously in local wall time: the last hour of daylight time. In Eastern, that's times of the form 5:MM UTC on the day daylight time ends. The local wall clock leaps from 1:59 (daylight time) back to 1:00 (standard time) again. Local times of the form 1:MM are ambiguous. :meth:`astimezone` mimics the local clock's behavior by mapping two adjacent UTC hours into the same local hour then. In the Eastern example, UTC times of the form 5:MM and 6:MM both map to 1:MM when converted to Eastern, but earlier times have the :attr:`~datetime.fold` attribute set to 0 and the later times have it set to 1. For example, at the Fall back transition of 2016, we get:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2147); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2147); *backlink*

Unknown interpreted text role "attr".

```
>>> u0 = datetime(2016, 11, 6, 4, tzinfo=timezone.utc)
>>> for i in range(4):
...     u = u0 + i*HOUR
...     t = u.astimezone(Eastern)
...     print(u.time(), 'UTC =', t.time(), t.tzname(), t.fold)
...
04:00:00 UTC = 00:00:00 EDT 0
05:00:00 UTC = 01:00:00 EDT 0
06:00:00 UTC = 01:00:00 EST 1
07:00:00 UTC = 02:00:00 EST 0
```

Note that the :class:`.datetime` instances that differ only by the value of the :attr:`~datetime.fold` attribute are considered equal in comparisons.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2169); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2169); *backlink*

Unknown interpreted text role "attr".

Applications that can't bear wall-time ambiguities should explicitly check the value of the :attr:`~datetime.fold` attribute or avoid using hybrid :class:`tzinfo` subclasses; there are no ambiguities when using :class:`timezone`, or any other fixed-offset :class:`tzinfo` subclass (such as a class representing only EST (fixed offset -5 hours), or only EDT (fixed offset -4 hours)).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2172); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2172); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2172); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2172); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2178)

Unknown directive type "seealso".

```
.. seealso::
```

```
:mod:`zoneinfo`
  The :mod:`datetime` module has a basic :class:`timezone` class (for
  handling arbitrary fixed offsets from UTC) and its :attr:`timezone.utc`
  attribute (a UTC timezone instance).

  ``zoneinfo`` brings the *IANA timezone database* (also known as the Olson
  database) to Python, and its usage is recommended.

`IANA timezone database <https://www.iana.org/time-zones>`_
  The Time Zone Database (often called tz, tzdata or zoneinfo) contains code
  and data that represent the history of local time for many representative
  locations around the globe. It is updated periodically to reflect changes
  made by political bodies to time zone boundaries, UTC offsets, and
  daylight-saving rules.
```

:class:`timezone` Objects

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2198); *backlink*

Unknown interpreted text role "class".

The :class:`timezone` class is a subclass of :class:`tzinfo`, each instance of which represents a timezone defined by a fixed offset from UTC.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2201); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2201); *backlink*

Unknown interpreted text role "class".

Objects of this class cannot be used to represent timezone information in the locations where different offsets are used in different days of the year or where historical changes have been made to civil time.

The *offset* argument must be specified as a :class:`timedelta` object representing the difference between the local time and UTC. It must be strictly between `-timedelta(hours=24)` and `timedelta(hours=24)`, otherwise :exc:`ValueError` is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2212); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2212); *backlink*

Unknown interpreted text role "exc".

The *name* argument is optional. If specified it must be a string that will be used as the value returned by the :meth:`datetime.tzname` method.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2217); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2220)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2222)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   The UTC offset is not restricted to a whole number of minutes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2226)

Unknown directive type "method".

```
.. method:: timezone.utcoffset(dt)

   Return the fixed value specified when the :class:`timezone` instance is
   constructed.

   The *dt* argument is ignored. The return value is a :class:`timedelta`
   instance equal to the difference between the local time and UTC.

   .. versionchanged:: 3.7
      The UTC offset is not restricted to a whole number of minutes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2237)

Unknown directive type "method".

```
.. method:: timezone.tzname(dt)

   Return the fixed value specified when the :class:`timezone` instance
   is constructed.

   If *name* is not provided in the constructor, the name returned by
   ``tzname(dt)`` is generated from the value of the ``offset`` as follows. If
   *offset* is ``timedelta(0)``, the name is "UTC", otherwise it is a string in
   the format ``UTCÂ±HH:MM``, where Â± is the sign of ``offset``, HH and MM are
   two digits of ``offset.hours`` and ``offset.minutes`` respectively.

   .. versionchanged:: 3.6
      Name generated from ``offset=timedelta(0)`` is now plain `'UTC'`, not
      ``'UTC+00:00'``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2253)

Unknown directive type "method".

```
.. method:: timezone.dst(dt)

   Always returns ``None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2257)

Unknown directive type "method".

```
.. method:: timezone.fromutc(dt)

   Return ``dt + offset``. The *dt* argument must be an aware
   :class:`.datetime` instance, with ``tzinfo`` set to ``self``.
```

Class attributes:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2264)

Unknown directive type "attribute".

```
.. attribute:: timezone.utc

   The UTC timezone, ``timezone(timedelta(0))``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2269)

Unknown directive type "index".

```
.. index::
   single: % (percent); datetime format
```

:meth:`strftime` and :meth:`strptime` Behavior

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2274); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2274); *backlink*

Unknown interpreted text role "meth".

:class:`date`, :class:`.datetime`, and :class:`.time` objects all support a `strftime(format)` method, to create a string representing the time under the control of an explicit format string.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2277); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2277); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2277); *backlink*

Unknown interpreted text role "class".

Conversely, the :meth:`datetime.strptime` class method creates a :class:`.datetime` object from a string representing a date and time and a corresponding format string.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2281); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2281); *backlink*

Unknown interpreted text role "class".

The table below provides a high-level comparison of :meth:`strftime` versus :meth:`strptime`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2285); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2285); *backlink*

Unknown interpreted text role "meth".

	strftime	**strptime**
Usage	Convert object to a string according to a given format	Parse a string into a :class:`.datetime` object given a corresponding format System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2292); *backlink* Unknown interpreted text role "class".
Type of method	Instance method	Class method

	strftime	strptime
Method of	:class:`date`; :class:`.datetime`; :class:`.time` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2296); *backlink* Unknown interpreted text role "class". **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2296); *backlink* Unknown interpreted text role "class". **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2296); *backlink* Unknown interpreted text role "class".	:class:`.datetime` **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2296); *backlink* Unknown interpreted text role "class".
Signature	`strftime(format)`	`strptime(date_string, format)`

:meth:`strftime` and :meth:`strptime` Format Codes

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2301); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2301); *backlink*

Unknown interpreted text role "meth".

The following is a list of all the format codes that the 1989 C standard requires, and these work on all platforms with a standard C implementation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2307)

Malformed table.

```
+-----------+--------------------------------+-----------------------+-------+
| Directive | Meaning                        | Example               | Notes |
+===========+================================+=======================+=======+
| ``%a``    | Weekday as locale's            || Sun, Mon, ..., Sat    | \(1)  |
|           | abbreviated name.              |  (en_US);             |       |
|           |                                || So, Mo, ..., Sa      |       |
|           |                                |  (de_DE)              |       |
+-----------+--------------------------------+-----------------------+-------+
| ``%A``    | Weekday as locale's full name. || Sunday, Monday, ...,  | \(1)  |
|           |                                |  Saturday (en_US);    |       |
```

		Sonntag, Montag, ..., Samstag (de_DE)	
``%w``	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, ..., 6	
``%d``	Day of the month as a zero-padded decimal number.	01, 02, ..., 31	\(9)
``%b``	Month as locale's abbreviated name.	Jan, Feb, ..., Dec (en_US); Jan, Feb, ..., Dez (de_DE)	\(1)
``%B``	Month as locale's full name.	January, February, ..., December (en_US); Januar, Februar, ..., Dezember (de_DE)	\(1)
``%m``	Month as a zero-padded decimal number.	01, 02, ..., 12	\(9)
``%y``	Year without century as a zero-padded decimal number.	00, 01, ..., 99	\(9)
``%Y``	Year with century as a decimal number.	0001, 0002, ..., 2013, 2014, ..., 9998, 9999	\(2)
``%H``	Hour (24-hour clock) as a zero-padded decimal number.	00, 01, ..., 23	\(9)
``%I``	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, ..., 12	\(9)
``%p``	Locale's equivalent of either AM or PM.	AM, PM (en_US); am, pm (de_DE)	\(1), \(3)
``%M``	Minute as a zero-padded decimal number.	00, 01, ..., 59	\(9)
``%S``	Second as a zero-padded decimal number.	00, 01, ..., 59	\(4), \(9)
``%f``	Microsecond as a decimal number, zero-padded to 6 digits.	000000, 000001, ..., 999999	\(5)
``%z``	UTC offset in the form ``Â±HHMM[SS[.ffffff]]`` (empty string if the object is naive).	(empty), +0000, -0400, +1030, +063415, -030712.345216	\(6)
``%Z``	Time zone name (empty string if the object is naive).	(empty), UTC, GMT	\(6)
``%j``	Day of the year as a zero-padded decimal number.	001, 002, ..., 366	\(9)
``%U``	Week number of the year (Sunday as the first day of the week) as a zero-padded decimal number. All days in a new year preceding the first Sunday are considered to be in week 0.	00, 01, ..., 53	\(7), \(9)
``%W``	Week number of the year (Monday as the first day of the week) as a zero-padded decimal number. All days in a new year preceding the first Monday are considered to be in week 0.	00, 01, ..., 53	\(7), \(9)
``%c``	Locale's appropriate date and time representation.	Tue Aug 16 21:30:00 1988 (en_US); Di 16 Aug 21:30:00 1988 (de_DE)	\(1)
``%x``	Locale's appropriate date representation.	08/16/88 (None); 08/16/1988 (en_US); 16.08.1988 (de_DE)	\(1)
``%X``	Locale's appropriate time representation.	21:30:00 (en_US); 21:30:00 (de_DE)	\(1)
``%%``	A literal ``'%'`` character.	%	

Several additional directives not required by the C89 standard are included for convenience. These parameters all correspond to ISO

8601 date values.

Directive	Meaning	Example	Notes
`%G`	ISO 8601 year with century representing the year that contains the greater part of the ISO week (`%V`).	0001, 0002, ..., 2013, 2014, ..., 9998, 9999	(8)
`%u`	ISO 8601 weekday as a decimal number where 1 is Monday.	1, 2, ..., 7	
`%V`	ISO 8601 week as a decimal number with Monday as the first day of the week. Week 01 is the week containing Jan 4.	01, 02, ..., 53	(8), (9)

These may not be available on all platforms when used with the :meth:`strftime` method. The ISO 8601 year and ISO 8601 week directives are not interchangeable with the year and week number directives above. Calling :meth:`strptime` with incomplete or ambiguous ISO 8601 directives will raise a :exc:`ValueError`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2428); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2428); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2428); *backlink*

Unknown interpreted text role "exc".

The full set of format codes supported varies across platforms, because Python calls the platform C library's :func:`strftime` function, and platform variations are common. To see the full set of format codes supported on your platform, consult the :manpage:`strftime(3)` documentation. There are also differences between platforms in handling of unsupported format specifiers.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2433); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2433); *backlink*

Unknown interpreted text role "manpage".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2439)

Unknown directive type "versionadded".

```
.. versionadded:: 3.6
   ``%G``, ``%u`` and ``%V`` were added.
```

Technical Detail

Broadly speaking, `d.strftime(fmt)` acts like the :mod:`time` module's `time.strftime(fmt, d.timetuple())` although not all objects support a :meth:`timetuple` method.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2445); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2445); *backlink*

Unknown interpreted text role "meth".

For the :meth:`datetime.strptime` class method, the default value is `1900-01-01T00:00:00.000`: any components not specified in the format string will be pulled from the default value. [3]

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2449); *backlink*

Unknown interpreted text role "meth".

Using `datetime.strptime(date_string, format)` is equivalent to:

```
datetime(*(time.strptime(date_string, format)[0:6]))
```

except when the format includes sub-second components or timezone offset information, which are supported in `datetime.strptime` but are discarded by `time.strptime`.

For :class:`.time` objects, the format codes for year, month, and day should not be used, as :class:`time` objects have no such values. If they're used anyway, `1900` is substituted for the year, and `1` for the month and day.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2461); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2461); *backlink*

Unknown interpreted text role "class".

For :class:`date` objects, the format codes for hours, minutes, seconds, and microseconds should not be used, as :class:`date` objects have no such values. If they're used anyway, `0` is substituted for them.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2465); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2465); *backlink*

Unknown interpreted text role "class".

For the same reason, handling of format strings containing Unicode code points that can't be represented in the charset of the current locale is also platform-dependent. On some platforms such code points are preserved intact in the output, while on others `strftime` may raise :exc:`UnicodeError` or return an empty string instead.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2469); *backlink*

Unknown interpreted text role "exc".

Notes:

1. Because the format depends on the current locale, care should be taken when making assumptions about the output value. Field orderings will vary (for example, "month/day/year" versus "day/month/year"), and the output may contain Unicode characters encoded using the locale's default encoding (for example, if the current locale is `ja_JP`, the default encoding could be any one of `eucJP`, `SJIS`, or `utf-8`; use :meth:`locale.getlocale` to determine the current locale's encoding).

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2478); *backlink*

   Unknown interpreted text role "meth".

2. The :meth:`strptime` method can parse years in the full [1, 9999] range, but years < 1000 must be zero-filled to 4-digit width.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2487); *backlink*

   Unknown interpreted text role "meth".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2490)

   Unknown directive type "versionchanged".

   ```
   .. versionchanged:: 3.2
      In previous versions, :meth:`strftime` method was restricted to
      years >= 1900.
   ```

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2494)

   Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   In version 3.2, :meth:`strftime` method was restricted to
   years >= 1000.
```

3. When used with the :meth:`strptime` method, the `%p` directive only affects the output hour field if the `%I` directive is used to parse the hour.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2499); *backlink*

   Unknown interpreted text role "meth".

4. Unlike the :mod:`time` module, the :mod:`datetime` module does not support leap seconds.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2503); *backlink*

   Unknown interpreted text role "mod".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2503); *backlink*

   Unknown interpreted text role "mod".

5. When used with the :meth:`strptime` method, the `%f` directive accepts from one to six digits and zero pads on the right. `%f` is an extension to the set of format characters in the C standard (but implemented separately in datetime objects, and therefore always available).

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2507); *backlink*

   Unknown interpreted text role "meth".

6. For a naive object, the `%z` and `%Z` format codes are replaced by empty strings.

   For an aware object:

   `%z`

   :meth:`utcoffset` is transformed into a string of the form `Â±HHMM[SS[.ffffff]]`, where `HH` is a 2-digit string giving the number of UTC offset hours, `MM` is a 2-digit string giving the number of UTC offset minutes, `SS` is a 2-digit string giving the number of UTC offset seconds and `ffffff` is a 6-digit string giving the number of UTC offset microseconds. The `ffffff` part is omitted when the offset is a whole number of seconds and both the `ffffff` and the `SS` part is omitted when the offset is a whole number of minutes. For example, if :meth:`utcoffset` returns `timedelta(hours=-3, minutes=-30)`, `%z` is replaced with the string `'-0330'`.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2520); *backlink*

   Unknown interpreted text role "meth".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2520); *backlink*

   Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2531)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   The UTC offset is not restricted to a whole number of minutes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2534)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   When the ``%z`` directive is provided to the  :meth:`strptime` method,
   the UTC offsets can have a colon as a separator between hours, minutes
   and seconds.
   For example, ``'+01:00:00'`` will be parsed as an offset of one hour.
   In addition, providing ``'Z'`` is identical to ``'+00:00'``.
```

`%Z`

In :meth:`strftime`, `%Z` is replaced by an empty string if :meth:`tzname` returns `None`; otherwise `%Z` is replaced by the returned value, which must be a string.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2542); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2542); *backlink*

Unknown interpreted text role "meth".

:meth:`strptime` only accepts certain values for `%Z`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2546); *backlink*

Unknown interpreted text role "meth".

1. any value in `time.tzname` for your machine's locale
2. the hard-coded values `UTC` and `GMT`

So someone living in Japan may have `JST`, `UTC`, and `GMT` as valid values, but probably not `EST`. It will raise `ValueError` for invalid values.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2555)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.2
   When the ``%z`` directive is provided to the :meth:`strptime` method, an
   aware :class:`.datetime` object will be produced. The ``tzinfo`` of the
   result will be set to a :class:`timezone` instance.
```

7. When used with the :meth:`strptime` method, `%U` and `%W` are only used in calculations when the day of the week and the calendar year (`%Y`) are specified.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2561); *backlink*

Unknown interpreted text role "meth".

8. Similar to `%U` and `%W`, `%V` is only used in calculations when the day of the week and the ISO year (`%G`) are specified in a :meth:`strptime` format string. Also note that `%G` and `%Y` are not interchangeable.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2566); *backlink*

Unknown interpreted text role "meth".

9. When used with the :meth:`strptime` method, the leading zero is optional for formats `%d`, `%m`, `%H`, `%I`, `%M`, `%S`, `%J`, `%U`, `%W`, and `%V`. Format `%y` does require a leading zero.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]datetime.rst, line 2572); *backlink*

Unknown interpreted text role "meth".

Footnotes

[1] If, that is, we ignore the effects of Relativity

[2] This matches the definition of the "proleptic Gregorian" calendar in Dershowitz and Reingold's book *Calendrical Calculations*, where it's the base calendar for all computations. See the book for algorithms for converting between proleptic Gregorian ordinals and many other calendar systems.

[3] See R. H. van Gent's guide to the mathematics of the ISO 8601 calendar for a good explanation.

[4] Passing `datetime.strptime('Feb 29', '%b %d')` will fail since `1900` is not a leap year.