Dates and Times

Austin Bingham
COFOUNDER - SIXTY NORTH

@austin_bingham

Robert Smallshire
COFOUNDER - SIXTY NORTH

@robsmallshire

Date and Time Types

Date and Time Types

`date`

Proleptic Gregorian calendar

Extended backward and forward infinitely

Use with care with historical dates

The last county to adopt the Gregorian calendar was Saudi Arabia in 2016

Date and Time Types

`date`

`time`

`datetime`

Timezones: Naïve vs. Aware

Naïve

Lacking timezone and daylight saving time information

Implicit

The precise meaning of naïve times is not explicit and is only "correct" by convention within a program

Aware

They carry information about timezone and daylight saving time

Date and Time Types

`date`

`time`

`datetime`

`tzinfo`

`timedelta`

These types are all immutable and can not be modified once they're created.

datetime.date

```
3
>>> d.day
3
>>> d.weekday()
1
>>> d.isoweekday()
2
>>> d.isoformat()
'2020-03-03'
>>> d.strftime('%A %d %B %Y')
'Tuesday 03 March 2020'
>>> "The date is {:%A %d %B %Y}".format(d)
'The date is Tuesday 03 March 2020'
>>> d.strftime('%A %-d %B %Y')
'Tuesday 3 March 2020'
>>> "{date:%A} {date.day} {date:%B} {date.year}".format(date=d)
'Tuesday 3 March 2020'
>>> datetime.date.min
datetime.date(1, 1, 1)
>>> datetime.date.max
datetime.date(9999, 12, 31)
>>> datetime.date.resolution
datetime.timedelta(days=1)
>>>
```

Platform Dependencies

These options are not portable due to platform dependencies

Many platforms are different in subtle ways

datetime.time

```
>>> datetime.time(hour=23, minute=59, second=59, microsecond=999999)
datetime.time(23, 59, 59, 999999)
>>> t = datetime.time(10, 32, 47, 675623)
>>> t.hour
10
>>> t.minute
32
>>> t.second
47
>>> t.microsecond
675623
>>> t.isoformat()
'10:32:47.675623'
>>> t.strftime('%Hh%Mm%Ss')
'10h32m47s'
>>> "{t.hour}h{t.minute}m{t.second}s".format(t=t)
'10h32m47s'
>>> datetime.time.min
datetime.time(0, 0)
>>> datetime.time.max
datetime.time(23, 59, 59, 999999)
>>> datetime.time.resolution
datetime.timedelta(microseconds=1)
>>>
```

Composite Date and Time

`datetime.datetime`

Combines a `date` and a `time` into a single object

datetime.datetime

```
>>> from datetime import datetime
>>> datetime
<class 'datetime.datetime'>
>>> datetime.time
<method 'time' of 'datetime.datetime' objects>
>>>
```

datetime.datetime

```
datetime.datetime(2020, 3, 3, 12, 28, 48, 302882)
>>> datetime.datetime.fromordinal(5)
datetime.datetime(1, 1, 5, 0, 0)
>>> datetime.datetime.fromtimestamp(3635352)
datetime.datetime(1970, 2, 12, 2, 49, 12)
>>> datetime.datetime.utcfromtimestamp(3635352)
datetime.datetime(1970, 2, 12, 1, 49, 12)
>>> d = datetime.date.today()
>>> t = datetime.time(8, 15)
>>> datetime.datetime.combine(d, t)
datetime.datetime(2020, 3, 3, 8, 15)
>>> dt = datetime.datetime.strptime("Monday 6 January 2014, 12:13:31", "%A %d %B
 %Y, %H:%M:%S")
>>> dt
datetime.datetime(2014, 1, 6, 12, 13, 31)
>>> dt.date()
datetime.date(2014, 1, 6)
>>> dt.time()
datetime.time(12, 13, 31)
>>> dt.day
6
>>> dt.isoformat()
'2014-01-06T12:13:31'
>>>
```

Durations

`datetime.timedelta`

Represents durations of time

Constructor accepts days, seconds, microseconds, milliseconds, minutes, hours, and weeks

We **strongly** recommend using keyword arguments

datetime.timedelta

```
>>> datetime.timedelta(milliseconds=1, microseconds=1000)
datetime.timedelta(microseconds=2000)
>>> td = datetime.timedelta(weeks=1, minutes=2, milliseconds=5500)
>>> td
datetime.timedelta(days=7, seconds=125, microseconds=500000)
>>> td.days
7
>>> td.seconds
125
>>> td.microseconds
500000
>>> str(td)
'7 days, 0:02:05.500000'
>>> repr(td)
'datetime.timedelta(days=7, seconds=125, microseconds=500000)'
>>>
```

Date and Time Arithmetic

Date and Time Arithmetic

```
>>> a = datetime.datetime(year=2014, month=5, day=8, hour=14, minute=22)
>>> b = datetime.datetime(year=2014, month=3, day=14, hour=12, minute=9)
>>> d = a - b
>>> d
datetime.timedelta(days=55, seconds=7980)
>>> d.total_seconds()
4759980.0
>>> datetime.date.today() + datetime.timedelta(weeks=1) * 3
datetime.date(2020, 3, 24)
>>> f = datetime.time(14, 30, 0)
>>> g = datetime.time(15, 45, 0)
>>> f - g
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for -: 'datetime.time' and 'datetime.time
'
>>>
```

Time Zones

Timezones

Use `tzinfo` for "aware" objects

Fundamentally political entities

No exhaustive timezone data in Python

Third-party modules like `pytz` or `dateutil`

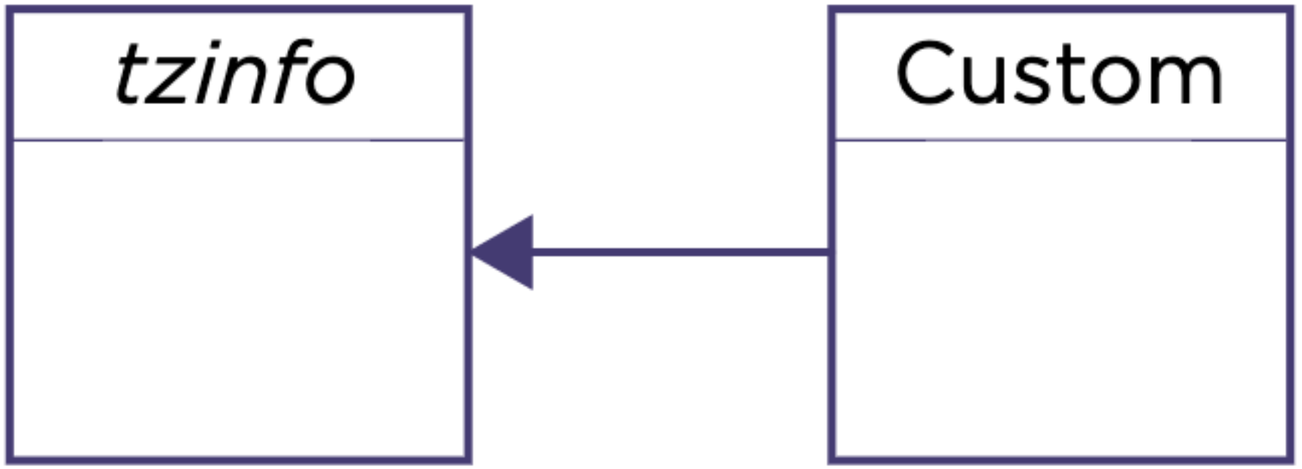
Python 3 contains rudimentary support

`tzinfo` and `timezone`

Timezones

```
>>> cet = datetime.timezone(datetime.timedelta(hours=1), "CET")
>>> cet
datetime.timezone(datetime.timedelta(seconds=3600), 'CET')
>>> departure = datetime.datetime(year=2014, month=1, day=7, hour=11, minute=30,
 tzinfo=cet)
>>> arrival = datetime.datetime(year=2014, month=1, day=7, hour=13, minute=5, tz
info=datetime.timezone.utc)
>>> arrival - departure
datetime.timedelta(seconds=9300)
>>> str(arrival - departure)
'2:35:00'
>>>
```

More Complete Timezone Support



docs.python.org/3/library/datetime.html#tzinfo-objects

pytz dateutil

Summary

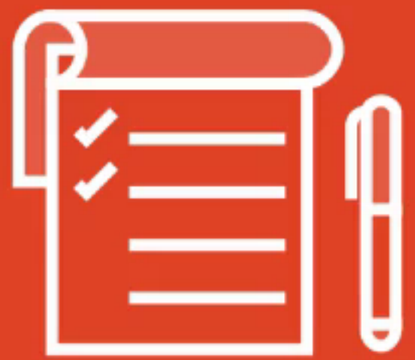
`datetime` provides support for working with dates and times

`datetime.date` represents calendar dates

`datetime.time` represents times of day

`datetime.datetime` combines date and time information

Summary



`datetime.timedelta` represents durations

`timedelta` allows certain kinds of arithmetic with times

`datetime.tzinfo` is an abstract class for working with timezones

`datetime.timezone` is a simple implementation of `tzinfo`

Python does not include exhaustive timezone information