Python library – datetime (*)

Datetime has the following components (classes):

- **date -** handles the year, month, and day.
- **time -** handles the hour, minute, second, microsecond, and tzinfo (time zone information).
- **datetime** - handles a combination of date and time
- **timedelta –** handles the duration of time.
- **tzinfo –** handles time zones.
- **timezone** – implements tzinfo as a fixed UTC offset

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

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. Naive objects are easy to understand and to work with, at the cost of ignoring some aspects of reality

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

(*) https://docs.python.org/3/library/datetime.html

(*) https://realpython.com/python-datetime/

Python library – datetime creation

```
In [1]: import datetime as dt

In [2]: dt.date(year=2021, month=12, day=20)
        dt.date(2021,12,20)
Out[2]: datetime.date(2021, 12, 20)

In [3]: dt.time(hour=15, minute=25, second=43)
        dt.time(15, 25, 43)
Out[3]: datetime.time(15, 25, 43)

In [4]: dt.datetime(year=2021, month=12, day=20, hour=15, minute=25, second=43)
        dt.datetime(2021, 12, 20, 15, 25, 43)
Out[4]: datetime.datetime(2021, 12, 20, 15, 25, 43)
```

Other ways to create Datetime instances

```
In [5]: today = dt.date.today()
        today
Out[5]: datetime.date(2021, 12, 26)

In [6]: today_time = dt.datetime.now()
        today_time
Out[6]: datetime.datetime(2021, 12, 26, 13, 36, 37, 711967)

In [7]: dt.datetime.combine(today, today_time.time())
Out[7]: datetime.datetime(2021, 12, 26, 13, 36, 37, 711967)
```

Creating from a string date

```
In [8]: # YYYY-MM-DD HH:MM:SS (in ISO 8601 format)
        dt.date.fromisoformat("2021-12-22")
Out[8]: datetime.date(2021, 12, 22)

In [9]: date_str = "12/22/2021 21:13:03"
        date_fmt_str = "%m/%d/%Y %H:%M:%S"
        dt.datetime.strptime(date_str, date_fmt_str)
Out[9]: datetime.datetime(2021, 12, 22, 21, 13, 3)
```

Python library – datetime – format codes

Directive	Description	Example
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	08

Directive	Description	Example
%f	Microsecond 000000-999999	548513
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%C	Century	20
%x	Local version of date	12/31/18
%X	Local version of time	17:41:00
%%	A % character	%
%G	ISO 8601 year	2018
%u	ISO 8601 weekday (1-7)	1
%V	ISO 8601 weeknumber (01-53)	01

(*) http://w3schools.com

Python library – datetime - timedelta

Date arithmetic

In [10]:
```
today = dt.datetime.now()
today
```

Out[10]:
```
datetime.datetime(2021, 12, 23, 12, 29, 8, 397528)
```

In [11]:
```
# adding two days
after_tomorrow = dt.timedelta(days=+2)
today + after_tomorrow
```

Out[11]:
```
datetime.datetime(2021, 12, 25, 12, 29, 8, 397528)
```

In [12]:
```
# subtracting two days
before_yesterday = dt.timedelta(days=-2)
today + before_yesterday
```

Out[12]:
```
datetime.datetime(2021, 12, 21, 12, 29, 8, 397528)
```

In [13]:
```
# adding two days and subtracting three hours
delta = dt.timedelta(days=+2, hours=-3)
today + delta
```

Out[13]:
```
datetime.datetime(2021, 12, 25, 9, 29, 8, 397528)
```

Python library – datetime - date

date objects

```
In [14]: # datetime.date(year, month, day)
         dt.date(2021,12,15)
Out[14]: datetime.date(2021, 12, 15)

In [15]: # Current local date
         dt.date.today()
Out[15]: datetime.date(2021, 12, 26)

In [16]: # Local date from timestamp
         d1 = dt.date.fromtimestamp(time.time())
         d1
Out[16]: datetime.date(2021, 12, 26)

In [17]: # date attributes
         d1.year, d1.month, d1.day
Out[17]: (2021, 12, 26)

In [18]: # Returns the day of the week (Monday is 0)
         d1.weekday()
Out[18]: 6

In [19]: # Returns the day of the week (Monday is 1)
         d1.isoweekday()
Out[19]: 7

In [20]: # Returns the year, week and weekday
         # The first week is the first week containing a Thursday
         d1.isocalendar()
Out[20]: (2021, 51, 7)

In [21]: # Returns a date string in ISO format YYYY-MM-DD
         d1.isoformat()
Out[21]: '2021-12-26'

In [22]: #Return a date string controlled by the format
         d1.strftime("%d/%m/%Y")
Out[22]: '26/12/2021'
```

Python library – datetime - datetime

datetime objects

```
In [23]: # datetime.datetime(year,month,day,hour=0,minute=0
         # second=0, microsecond=0,tzinfo=None,*,fold=0)
         dt.datetime(2021,12,26,16,6,45)
Out[23]: datetime.datetime(2021, 12, 26, 16, 6, 45)

In [24]: # Returns the current local date and time
         dt1 = dt.datetime.today()
         dt1
Out[24]: datetime.datetime(2021, 12, 26, 16, 59, 41, 554500)

In [25]: # returns the current UTC date and time
         dt.datetime.utcnow()
Out[25]: datetime.datetime(2021, 12, 26, 16, 59, 41, 570490)

In [26]: # Local date from timestamp
         dt.datetime.fromtimestamp(time.time())
Out[26]: datetime.datetime(2021, 12, 26, 16, 59, 41, 584483)
```

datetime object attributes

```
In [27]: dt1.year, dt1.month, dt1.day
Out[27]: (2021, 12, 26)

In [28]: dt1.hour, dt1.minute, dt1.second, dt1.microsecond
Out[28]: (16, 59, 41, 554500)

In [29]: dt1.tzinfo, dt1.fold
Out[29]: (None, 0)
```

Python library – datetime - datetime

datetime object methods

```
In [30]: # Returns date object with the same date
         dt1.date()
Out[30]: datetime.date(2021, 12, 26)

In [31]: # Returns time object with the same time
         dt1.time()
Out[31]: datetime.time(16, 59, 41, 554500)

In [32]: # Returns time object with tzinfo attributes
         dt1.timetz()
Out[32]: datetime.time(16, 59, 41, 554500)

In [33]: # Returns a datetime object with new tzinfo atribute tz
         dt1 = dt1.astimezone(tz=None)

In [34]: # Returns dst - self.tzinfo.dst(self)
         dt1.dst()

In [35]: # Returns self.tzinfo.tzname(self)
         dt1.tzname()
Out[35]: 'GMT Standard Time'

In [36]: # returns a struct_time
         dt1.timetuple()
Out[36]: time.struct_time(tm_year=2021, tm_mon=12, tm_mday=26, tm_hour=16,
         tm_min=59, tm_sec=41, tm_wday=6, tm_yday=360, tm_isdst=-1)
```

```
In [37]: # Returns the day of the week (Monday is 0)
         dt1.weekday()
Out[37]: 6

In [38]: # Returns the day of the week (Monday is 1)
         dt1.isoweekday()
Out[38]: 7

In [39]: # Returns the year, week and weekday
         # The first week is the first week containing a Thursday
         dt1.isocalendar()
Out[39]: (2021, 51, 7)

In [40]: # Returns a date string in ISO format
         # YYYY-MM-DDTHH:MM:SS:ffffff
         dt1.isoformat()
Out[40]: '2021-12-26T16:59:41.554500+00:00'

In [41]: # Returns a date string
         dt1.ctime()
Out[41]: 'Sun Dec 26 16:59:41 2021'

In [42]: #Return a date string controlled by the format
         dt1.strftime("%A, %d. %B %Y %I:%M%p")
Out[42]: 'Sunday, 26. December 2021 04:59PM'
```

Python library – datetime - time

time objects

In [43]:
```
# datetime.time(hour=0,minute=0,second=0,
# microsecond=0,tzinfo=None,*,fold=0)
t1 = dt.time(16, 42, 55, 253765)
t1
```

Out[43]:
```
datetime.time(16, 42, 55, 253765)
```

time object attributes

In [44]:
```
t1.hour, t1.minute, t1.second, t1.microsecond
```

Out[44]:
```
(16, 42, 55, 253765)
```

In [45]:
```
t1.tzinfo, t1.fold
```

Out[45]:
```
(None, 0)
```

time object methods

In [46]:
```
# Returns a time string  in ISO 8601 format
t1.isoformat()
```

Out[46]:
```
'16:42:55.253765'
```

In [47]:
```
#Return a time string controlled by the format
t1.strftime("%H:%M:%S.%f")
```

Out[47]:
```
'16:42:55.253765'
```

In [48]:
```
# Returns dst - self.tzinfo.dst(None)
t1.dst()
```

In [49]:
```
# Returns self.tzinfo.tzname(None)
t1.tzname()
```