DATETIME

Python datetime contains below class for dealing with datetime.

1. **datetime.date**

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

1. **datetime.time**

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: hour, minute, second, microsecond, and tzinfo.

1. **datetime.datetime**

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

1. **datetime.timedelta**

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

1. **datetime.tzinfo**

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

1. **datetime.timezone**

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

**Notes:**

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

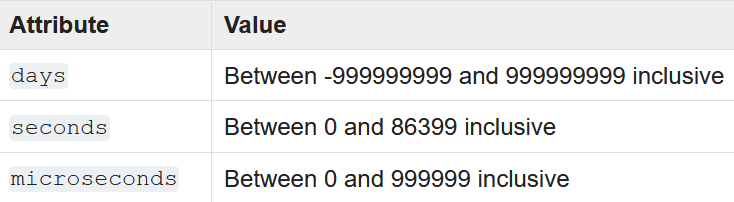
* Objects of these types are immutable.
* Objects of these types are hashable, meaning that they can be used as dictionary keys.
* Objects of these types support efficient pickling via the pickle module.

**timedelta**

datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)

all arguments are optional and default to 0.

On timedelta we have only below attributes accessible.



In case if we create any timedelta onject using days, hours, .. but that will be converted into above three values and each can be accessed using by their attributed as listed above.

**Operation on timedelta object**

Timedelta object supports many operations few of them are-

* Addition
* Subtraction
* Multiplication by number(t1=t1\*mum)
* Division( division of two timedelta object i.e. t1/t2)
* Division(by number or float i.e. -- t1/f)
* abs(t) (create absolute value, something like mod in math)

delta = datetime.timedelta(days=50,seconds=27,microseconds=10,milliseconds=29000,minutes=5, hours=8,weeks=2)

print(delta.days) //64

print(delta.seconds) //29156

print(delta.hours) #this line will give error as hour is not a valid attribute

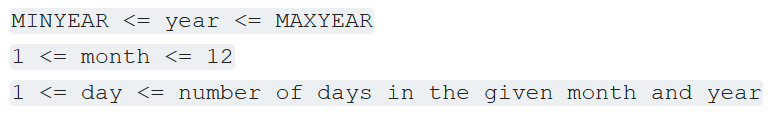
print(delta.microseconds) //10

**date**

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

datetime.date(year, month, day) ---- constructor of creating date

---All arguments are required. Arguments must be integers and they must be in their range



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

**Other constructor and class methods**

1. date.today()

Returns current local date

**Attributes of date class**

* date.year --- year between minyear and maxyear
* date.month ---- month between 1 and 12 inclusive
* date.day ---- day in integer

**Supported operations**

Date object have below supported operations

* addition of two date ---- date1+date2
* subtraction of two data ----- date1-date2
* comparison of two date --- date1 com\_operator date2

**Instance methods of date class**

It has many methods some commonly used are-

1. *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

d = date(2002, 12, 31)

d.replace(day=26) #datetime.date(2002, 12, 26) day num. of d is replaced by give day num.

1. *date.weekday()* ---- returns the weekday, Monday as 0, Tues=1..
2. *date.ctime* ---- Returns string representations of date
3. *date.strftime(format)* --- Retuns the string representation of date – in date, datetime class

**date.strftime(format) ----** this method is available indate, datetime class

This method is used to convert given datetime data into string as per specified format specifiers.

date=datetime.date.today()

str\_time=date.strftime('%Y-%m-%d') # gives data into YYY-MM-DD format

print(str\_time)

str\_time=date.strftime('%Y/%m/%d') # gives data into YYY/MM/DD format

print(str\_time)

str\_time=date.strftime('%d-%m-%Y') # gives data into DD-MM-YYYY format

print(str\_time)

str\_time=date.strftime('%d/%m/%Y') # gives data into DD/MM/YYYY format

print(str\_time)

**strptime(date\_str, format\_of\_dat\_str)**

*This methods is available in datetime class only.*

This method is used to convert the give string into datetime object.

The output data or datetime object depends on the date\_str and format specifier.

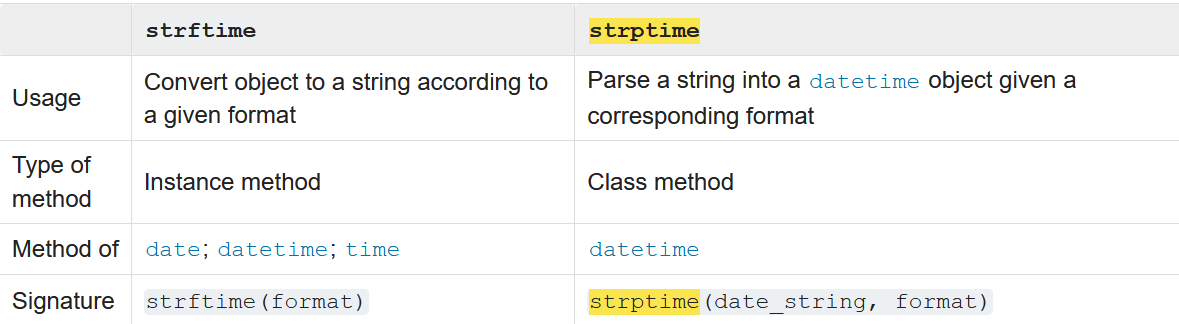
date\_str ----- string that we want to convert the string into date or datetime

format\_of\_date\_str ----- it specifies the in which format date\_str is given.

*This will always return date or datetime in YYYY-MM-DD or YYYY-MM-DD HH-MM-SS format*

*If date\_str doesn’t match with format\_specifier then it retuns ValueError.*

*In case if only datetime is needed then call date() on returned datetime object.*



datetime\_object = datetime.strptime('26032021', '%d%m%Y')

print(datetime\_object)

print(datetime\_object.date())

**Datetime**

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

datetime.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 tzinfo subclass
* The remaining arguments must be integers in the following ranges.

**Class methods of datetime**

It has below class methods.

* datetime.today()

Return the current local datetime, with tzinfo None

* datetime.now(tz=None)

Return the current local date and time\

* datetime.utcnow()

Return the current UTC date and time, with tzinfo None

* datetime.strptime(date\_string, format)

Return a datetime corresponding to date\_string, parsed according to format

**Attributes of datetime**

* datetime.year -------- year value from datetime object
* datetime.month -------- month value from datetime object
* datetime.day -------- day value from datetime object
* datetime.hour -------- hour value from datetime object
* datetime.minute -------- minute value from datetime object
* datetime.seconds -------- seconds value from datetime object
* datetime.microseconds -------- microseconds value from datetime object
* datetime.tzinfo -------- tzinfo value from datetime object

**instance methods of datetime**

datetime.date() ---------gives date value from datetime

datetime.time() ---------gives time value from datetime

datetime.timetz() ---------gives timetx value from datetime

datetime.replace() ---------same as previous

datetime.weekday() ---------gives weekday value from datetime

datetime.strftime() ---------check previous

**TIME**

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

datetime.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 tzinfo subclass.

**class attributes**

time.hour

time.second

time.microsecond