Python Datetime Module

The Datetime module allows us to work with date and time objects. It provides three additional data types: date, time and datetime.

Invoking the library

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

date()

The date method return a date object with the year, month and day attributes:

datetime.date(year: int, month: int, day: int)

In [2]:
```
from datetime import date
```

In [3]:
```
obj = date(2022, 12, 1)

print("year",obj.year)

print("Month",obj.month)

print("day",obj.day)
```

```
year 2022
Month 12
day 1
```

time()

The time method return a time object with the hour, minute, second, microsecond and tzinfo attributes:

datetime.time(hour: int, minute: int, second: int)

In [4]:
```
from datetime import time
```

In [5]:
```
obj = time(10, 20, 33)

print("Hour:",obj.hour)

print("Minute:",obj.minute)

print("Seconds:",obj.second)
```

```
Hour: 10
Minute: 20
Seconds: 33
```

datetime()

The datetime returns an object with both, the date and time objects attributes:

datetime.datetime(year: int, month: int, day: int, hour: int, minute: int, second: int)

datetime.datetime(year, month, day, hour, minute, second)

In [6]:
```
from datetime import datetime
```

In [7]:
```
obj = datetime(2024, 12, 1, 15, 35, 59)

print("Year:",obj.year)

print("Month:",obj.month)

print("Day:",obj.day)

print("Hour:",obj.hour)

print("mintue",obj.minute)

print("Second:",obj.second)
```

```
Year: 2024
Month: 12
Day: 1
Hour: 15
mintue 35
Second: 59
```

now() and today()

now and today methods return a datetime object with system's exact day and time:

In [8]:
```
from datetime import datetime
```

In [9]:
```
now = datetime.now()
now
```

Out[9]:
```
datetime.datetime(2024, 4, 12, 20, 7, 40, 937959)
```

In [10]:
```
print("Date:",now.date())

print("Time:",now.time())

print("Year:",now.year)

print("Month:",now.month)

print("Day:",now.day)

print("Hour:",now.hour)

print("Minute:",now.minute)

print("Second:",now.second)

print("Microsecond:",now.microsecond)
```

```
Date: 2024-04-12
Time: 20:07:40.937959
Year: 2024
Month: 4
Day: 12
Hour: 20
Minute: 7
Second: 40
Microsecond: 937959
```

Additionally, now can take a timezone object as an optional parameter:

In [11]:
```
from datetime import datetime, timezone
```

In [12]:
```
print("Indian Time utc:",datetime.now(timezone.utc))
```

```
Indian Time utc: 2024-04-12 14:37:40.983502+00:00
```

strftime() and strptime()

You can easily transform between strings and datetime objects with the strftime and strptime methods.

strftime()

strftime allow us to create human formatted strings out of a Python datetime object:

In [13]:
```
from datetime import datetime
```

In [14]:
```
now = datetime.now()

print(now)


print(now.strftime("%d-%b-%Y"))


print(now.strftime("%d-%m-%Y"))


print(now.strftime("%d-%b-%Y"))


print(now.strftime("%d-%m-%Y"))


print(now.strftime("%m/%d/%Y"))


print(now.strftime("%b/%d/%Y - %H:%M:%S"))
```

```
2024-04-12 20:07:41.008757
12-Apr-2024
12-04-2024
12-Apr-2024
12-04-2024
04/12/2024
Apr/12/2024 - 20:07:41
```

strptime()

The strptime method creates a datetime object from a string.

A string representing a datetime object.

The python format code equivalent to that string.

In [15]:
```
from datetime import datetime
```

In [16]:
```
datetime_str = '12-Jul-2023'

print("date_month_year",datetime.strptime(datetime_str, '%d-%b-%Y'))


datetime_str = 'Jul/12/2023 - 14:38:37'

print("date_month_year_hour_minute",datetime.strptime(datetime_str, "%b/%d/%Y - %H:%M:%S"))
```

```
date_month_year 2023-07-12 00:00:00
date_month_year_hour_minute 2023-07-12 14:38:37
```

timedelta()

The timedelta object represents the difference between two dates or times.

In [17]:
```
from datetime import datetime
```

In [18]:
```
date_1 = datetime.strptime('12-Jul-2023', '%d-%b-%Y')
date_2 = datetime.strptime('01-Jan-2024', '%d-%b-%Y')

print("Firest date:",date_1)

print("Second date:",date_2)

difference = date_2 - date_1

difference

print("difference between Two dates",difference.days)
```

```
Firest date: 2023-07-12 00:00:00
Second date: 2024-01-01 00:00:00
difference between Two dates 173
```

timedelta can add days, seconds and microseconds to a datetime object:

In [19]:
```
from datetime import datetime, timedelta
```

In [20]:
```
now = datetime.now()
```

```
print("Present Time&hour",now)

print("adding The days,seconds",now + timedelta(days=10, seconds=15))
```

```
Present Time&hour 2024-04-12 20:07:41.144903
adding The days,seconds 2024-04-22 20:07:56.144903
```

And can subtract days, seconds and microseconds to a datetime object:

In [21]:
```
from datetime import datetime, timedelta
```

In [22]:
```
now = datetime.now()

print("present_time",now)

print("Subtract the days",now - timedelta(days=10, seconds=15))
```

```
present_time 2024-04-12 20:07:41.180702
Subtract the days 2024-04-02 20:07:26.180702
```

Python strftime cheatsheet

Code	Example	Description
%a	Sun	Weekday as locale's abbreviated name.
%A	Sunday	Weekday as locale's full name.
%w	0	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.
%d	08	Day of the month as a zero-padded decimal number.
%-d	8	Day of the month as a decimal number. (Platform specific)
%b	Sep	Month as locale's abbreviated name.
%B	September	Month as locale's full name.
%m	09	Month as a zero-padded decimal number.
%-m	9	Month as a decimal number. (Platform specific)
%y	13	Year without century as a zero-padded decimal number.
%Y	2013	Year with century as a decimal number.
%H	07	Hour (24-hour clock) as a zero-padded decimal number.
%-H	7	Hour (24-hour clock) as a decimal number. (Platform specific)
%I	07	Hour (12-hour clock) as a zero-padded decimal number.
%-I	7	Hour (12-hour clock) as a decimal number. (Platform specific)
%p	AM	Locale's equivalent of either AM or PM.
%M	06	Minute as a zero-padded decimal number.
%-M	6	Minute as a decimal number. (Platform specific)
%S	05	Second as a zero-padded decimal number.
%-S	5	Second as a decimal number. (Platform specific)
%f	000000	Microsecond as a decimal number, zero-padded to 6 digits.

Code	Example	Description
%z	+0000	UTC offset in the form ±HHMM[SS[.ffffff]] (empty string if the object is naive).
%Z	UTC	Time zone name (empty string if the object is naive).
%j	251	Day of the year as a zero-padded decimal number.
%-j	251	Day of the year as a decimal number. (Platform specific)
%U	36	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.
%-U	36	Week number of the year (Sunday as the first day of the week) as a decimal number. All days in a new year preceding the first Sunday are considered to be in week 0. (Platform specific)
%W	35	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.
%-W	35	Week number of the year (Monday as the first day of the week) as a decimal number. All days in a new year preceding the first Monday are considered to be in week 0. (Platform specific)
%c	Sun Sep 8 07:06:05 2013	Locale's appropriate date and time representation.
%x	09/08/13	Locale's appropriate date representation.
%X	07:06:05	Locale's appropriate time representation.
%%	%	A literal '%' character.

Calender

In [25]:
```
import calendar
```

In [26]:
```
year=2024
month=9
```

In [27]:
```
print(calendar.month(year,month))
```

```
   September 2024
Mo Tu We Th Fr Sa Su
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
```

In [28]:
```
print(calendar.monthrange(year,month))
```

```
(6, 30)
```

In [29]:
```
calendar.firstweekday()
```

Out[29]:
```
0
```

In [31]:
```
calendar.isleap(2024)
```

Out[31]:
```
True
```

In [32]:
```
calendar.leapdays(2019,2025)
```

Out[32]:
```
2
```

In [34]:
```
print ("The calendar of year 2024 is :")
print (calendar.calendar(2024))
```

```
The calendar of year 2024 is :2024
                                  2024

      January               February               March
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7                1  2  3  4                  1  2  3
 8  9 10 11 12 13 14    5  6  7  8  9 10 11    4  5  6  7  8  9 10
15 16 17 18 19 20 21   12 13 14 15 16 17 18   11 12 13 14 15 16 17
22 23 24 25 26 27 28   19 20 21 22 23 24 25   18 19 20 21 22 23 24
29 30 31               26 27 28 29            25 26 27 28 29 30 31

       April                  May                    June
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7             1  2  3  4  5                  1  2
 8  9 10 11 12 13 14    6  7  8  9 10 11 12    3  4  5  6  7  8  9
15 16 17 18 19 20 21   13 14 15 16 17 18 19   10 11 12 13 14 15 16
22 23 24 25 26 27 28   20 21 22 23 24 25 26   17 18 19 20 21 22 23
29 30                  27 28 29 30 31         24 25 26 27 28 29 30

        July                  August                September
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7                1  2  3  4                     1
 8  9 10 11 12 13 14    5  6  7  8  9 10 11    2  3  4  5  6  7  8
15 16 17 18 19 20 21   12 13 14 15 16 17 18    9 10 11 12 13 14 15
22 23 24 25 26 27 28   19 20 21 22 23 24 25   16 17 18 19 20 21 22
29 30 31               26 27 28 29 30 31      23 24 25 26 27 28 29
                                              30

      October               November               December
Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su   Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6                   1  2  3                     1
 7  8  9 10 11 12 13    4  5  6  7  8  9 10    2  3  4  5  6  7  8
14 15 16 17 18 19 20   11 12 13 14 15 16 17    9 10 11 12 13 14 15
21 22 23 24 25 26 27   18 19 20 21 22 23 24   16 17 18 19 20 21 22
28 29 30 31            25 26 27 28 29 30      23 24 25 26 27 28 29
                                              30 31
```

M.RAMA GOPALA KRISHNA