datetime and calendar module

Python provides built-ins data types like,

1. Int
2. Float
3. Complex
4. Bool
5. NoneType

These are called numeric data types. These data types are used to represent numbers.

In application development in order to represent date and time, python provides to predefined data types or classes date and time. These classes or data types are available in a predefined module called datetime.

This module comes with python software.
datetime module provides the following classes or data types

1. date
2. time
3. datetime
4. timedelta

date

date object or data type represent date.

Syntax: date(year,month,day)

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, ValueError is raised

Note: The value of MINYEAR is 1 and MAXYEAR 9999

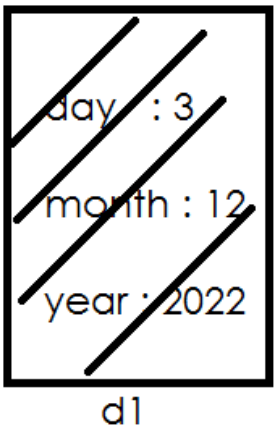
Date object is created with three properties or attributes and these attributes are read only.

1. day
2. month
3. year

```
d1=datetime.date(2022,12,3)
```

```
print(d1.day)
print(d1.month)
print(d1.year)

3
12
2022
```



d1

```
d1.month=11 # Error
```

Example:

```
>>> import datetime
>>> d1=datetime.date(2022,12,3)
>>> print(d1)
2022-12-03
>>> print(d1.year,d1.month,d1.day)
2022 12 3
>>> print(f'{d1.day}/{d1.month}/{d1.year}')
3/12/2022
>>> d1.day=2
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    d1.day=2
AttributeError: attribute 'day' of 'datetime.date' objects is not writable
```

date.fromisoformat(*date_string*)

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

```
# write a program to read details of n students
# each student having rollno,name,course,doj
import datetime
student_dict={}
n=int(input("Enter how many students ?"))
for i in range(n):
    rollno=int(input("Rollno "))
    name=input("Name ")
    course=input("Course ")
    doj=datetime.date.fromisoformat(input("Date of Joining YYYY-MM-DD"))
    student_dict[rollno]=[name,course,doj]

print(student_dict)
```

Output:

```
Enter how many students ?2
Rollno 1
Name naresh
Course java
Date of Joining YYYY-MM-DD2022-05-04
Rollno 2
Name sursh
Course python
Date of Joining YYYY-MM-DD2022-06-03
{1: ['naresh', 'java', datetime.date(2022, 5, 4)], 2: ['sursh', 'python',
datetime.date(2022, 6, 3)]}
```

Example:

```
>>> import datetime
>>> studList=[['ramesh',datetime.date(2022,11,4)],
          ['suresh',datetime.date(2022,11,4)],
          ['naresh',datetime.date(2022,10,5)],
...           ['rajesh',datetime.date(2022,10,5)]]
>>> for stud in studList:
...     name,d=stud
...     if d.month==11:
...         print(name)
```

...
...
ramesh
suresh

Example:

```
d1=datetime.date(2022,12,3)
d1=d1+1
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    d1=d1+1
TypeError: unsupported operand type(s) for +: 'datetime.date' and 'int'
```

date.today()

Return the current local date or return system date

```
>>> d2=datetime.date.today()
>>> print(d2)
>>> print(d2.year,d2.month,d2.day)
2022 12 3
```

time date type or object

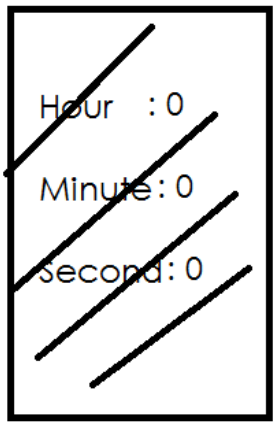
A time object represents a (local) time of day, independent of any particular day.

datetime.time(hour=0, minute=0, second=0, microsecond=0)

All the arguments of time function are default or optional.
Time is an immutable object, after creating time we cannot do any changes.

t1=datetime.time()

print(t1)
0:0:0
print(t1.hour) 0
print(t1.minute) 0
print(t1.second) 0



time object
t1

t1.hour=4 # Error

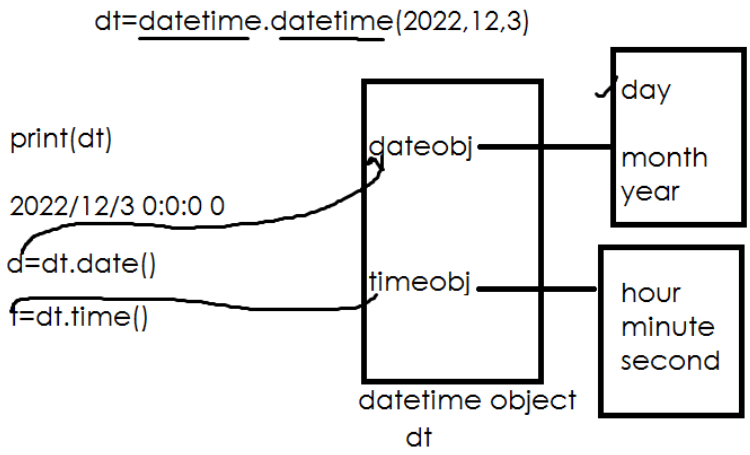
```
>>> t1=datetime.time(7,8,9)
>>> print(t1)
07:08:09
>>> print(t1.hour,t1.minute,t1.second)
7 8 9
```

time.fromisoformat(*time_string*)
Return a time corresponding to a *time_string* in one of the formats emitted by time.isoformat() (HH:MM:SS). This function is used to convert string representation of time to time object

```
>>> t2=datetime.time.fromisoformat("07:20:40")
>>> print(t2)
07:20:40
```

datetime object or date type
This object represents date and time.

datetime(year,month,day,hour=0,minute=0,second=,microseconds=0)



datetime.today()
Return the current local datetime

datetime.now(*tz=None*)
Return the current local date and time.

Example:

```
>>> import datetime
>>> dt1=datetime.datetime.today()
>>> dt2=datetime.datetime.now()
>>> print(dt1)
2022-12-03 19:27:37.078386
>>> print(dt2)
2022-12-03 19:27:48.066475
>>> d1=dt1.date()
>>> t1=dt1.time()
>>> print(d1)
2022-12-03
>>> print(type(d1))
<class 'datetime.date'>
>>> print(t1)
19:27:37.078386
>>> print(type(t1))
<class 'datetime.time'>
>>> print(type(dt1))
<class 'datetime.datetime'>
```

timedelta

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

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

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.

Example:

```
>>> d1=datetime.date(2022,12,3)
>>> days=datetime.timedelta(days=10)
>>> print(d1)
2022-12-03
>>> d1=d1+days
>>> print(d1)
2022-12-13
>>> d1=d1-days
>>> print(d1)
2022-12-03
>>> m=datetime.timedelta(weeks=4)
>>> d1=d1+m
>>> print(d1)
2022-12-31
```

date.strftime(*format*)
Return a string representing the date, controlled by an explicit format string.

Directive	Meaning	Example

%a	Weekday as locale’s abbreviated name.	Sun, Mon, …, Sat (en_US); So, Mo, …, Sa (de_DE)
%A	Weekday as locale’s full name.	Sunday, Monday, …, 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
%b	Month as locale’s abbreviated name.	Jan, Feb, …, Dec (en_US); Jan, Feb, …, Dez (de_DE)
%B	Month as locale’s full name.	January, February, …, December (en_US); Januar, Februar, …, Dezember (de_DE)
%m	Month as a zero-padded decimal number.	01, 02, …, 12
%y	Year without century as a zero-padded decimal number.	00, 01, …, 99
%Y	Year with century as a decimal number.	0001, 0002, …, 2013, 2014, …, 9998, 9999
%H	Hour (24-hour clock) as a zero-padded decimal number.	00, 01, …, 23
%I	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, …, 12
%p	Locale’s equivalent of either AM or PM.	AM, PM (en_US); am, pm (de_DE)
%M	Minute as a zero-padded decimal number.	00, 01, …, 59
%S	Second as a zero-padded decimal number.	00, 01, …, 59

%f	Microsecond as a decimal number, zero-padded to 6 digits.	000000, 000001, …, 999999
%z	UTC offset in the form ±HHMM[SS[.ffffff]] (empty string if the object is naive).	(empty), +0000, -0400, +1030, +063415, -030712.345216
%Z	Time zone name (empty string if the object is naive).	(empty), UTC, GMT
%j	Day of the year as a zero-padded decimal number.	001, 002, …, 366
%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
%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
%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)
%x	Locale’s appropriate date representation.	08/16/88 (None); 08/16/1988 (en_US); 16.08.1988 (de_DE)
%X	Locale’s appropriate time representation.	21:30:00 (en_US); 21:30:00 (de_DE)

Example:

```
>>> import datetime
>>> d1=datetime.date.today()
>>> print(d1)
2022-12-03
>>> print(d1.strftime("%a"))
Sat
```

```
>>> print(d1.strftime("%A"))
Saturday
>>> print(d1.strftime("%w"))
6
>>> print(d1.strftime("%d"))
03
>>> print(d1.strftime("%d %A"))
03 Saturday
>>> print(d1.strftime("%d %A %b"))
03 Saturday Dec
>>> print(d1.strftime("%d %A %B"))
03 Saturday December
>>> print(d1.strftime("%d/%m"))
03/12
>>> print(d1.strftime("%d/%m/%y"))
03/12/22
>>> print(d1.strftime("%d/%m/%Y"))
03/12/2022
>>> print(d1.strftime("%d %A %B %Y"))
03 Saturday December 2022
>>> dt=datetime.datetime.today()
>>> print(dt)
2022-12-03 20:05:40.680835
>>> print(dt.strftime("%H:%M:%S"))
20:05:40
>>> print(dt.strftime("%I:%M:%S"))
08:05:40
>>> print(dt.strftime("%j"))
337
>>> print(dt.strftime("%U"))
48
>>> print(dt.strftime("%W"))
48
>>> print(dt.strftime("%c"))
Sat Dec  3 20:05:40 2022
>>> print(dt.strftime("%x"))
12/03/22
>>> print(dt.strftime("%X"))
20:05:40
```

calendar module

“calendar” module comes with python software.

This module allows you to output calendars like the Unix cal program, and provides additional useful functions related to the calendar.

calendar.month(*theyear*, *themonth*, *w=0*, *l=0*)
Returns a month’s calendar in a multi-line string using the formatmonth() of the TextCalendar class.

calendar.calendar(*year*, *w=2*, *l=1*, *c=6*, *m=3*)
Returns a 3-column calendar for an entire year as a multi-line string using the formatyear() of the TextCalendar class

```
>>> import calendar
>>> mcal=calendar.month(2022,12)
print(type(mcal))
<class 'str'>
>>> print(mcal)
   December 2022
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 31

>>> print(calendar.month(2022,12))
   December 2022
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 31

>>> ycal=calendar.calendar(2022,m=2)
>>> print(ycal)
```

2022

January

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
31						

February

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						

March

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	31			

April

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	

May

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	31					

June

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			

July

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	31

August

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	31				

September

Mo	Tu	We	Th	Fr	Sa	Su
			1	2	3	4
5	6	7	8	9	10	11

October

Mo	Tu	We	Th	Fr	Sa	Su
					1	2
3	4	5	6	7	8	9

```
12 13 14 15 16 17 18      10 11 12 13 14 15 16
19 20 21 22 23 24 25      17 18 19 20 21 22 23
26 27 28 29 30            24 25 26 27 28 29 30
                    31

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

Example:

```
import calendar

f=open("calout.txt","w")
ycal=calendar.calendar(2022,m=2)
print(ycal,file=f,flush=True)
```

Output:

The output is saved inside calout.txt file, which is created in current working directory

calendar.HTMLCalendar(*firstweekday=0*)

This class can be used to generate HTML calendars.

formatyear(*theyear*, *width=3*)

Return a year's calendar as an HTML table. *width* (defaulting to 3) specifies the number of months per row

formatmonth(theyear, themonth, withyear=True)

Return a month's calendar as an HTML table. If withyear is true the year will be included in the header, otherwise just the month name will be used.

```
>>> import calendar
>>> hcal=calendar.HTMLCalendar()
>>> s=hcal.formatmonth(2022,12)
```

```
>>> print(s)
<table border="0" cellpadding="0" cellspacing="0" class="month">
<tr><th colspan="7" class="month">December 2022</th></tr>
<tr><th class="mon">Mon</th><th class="tue">Tue</th><th
class="wed">Wed</th><th class="thu">Thu</th><th class="fri">Fri</th><th
class="sat">Sat</th><th class="sun">Sun</th></tr>
<tr><td class="noday"> </td><td class="noday"> </td><td
class="noday"> </td><td class="thu">1</td><td class="fri">2</td><td
class="sat">3</td><td class="sun">4</td></tr>
<tr><td class="mon">5</td><td class="tue">6</td><td
class="wed">7</td><td class="thu">8</td><td class="fri">9</td><td
class="sat">10</td><td class="sun">11</td></tr>
<tr><td class="mon">12</td><td class="tue">13</td><td
class="wed">14</td><td class="thu">15</td><td class="fri">16</td><td
class="sat">17</td><td class="sun">18</td></tr>
<tr><td class="mon">19</td><td class="tue">20</td><td
class="wed">21</td><td class="thu">22</td><td class="fri">23</td><td
class="sat">24</td><td class="sun">25</td></tr>
<tr><td class="mon">26</td><td class="tue">27</td><td
class="wed">28</td><td class="thu">29</td><td class="fri">30</td><td
class="sat">31</td><td class="noday"> </td></tr>
</table>
```

Example:

```
import calendar

f=open("monthcal.html","w")
hcal=calendar.HTMLCalendar()
mcal=hcal.formatmonth(2022,12)
print(mcal,file=f,flush=True)
```

Output:

Output is saved inside monthcal.html file, which is created in current working directory

Example:

```
import calendar

f=open("yearcal.html","w")
hcal=calendar.HTMLCalendar()
```

```
ycal=hcal.formatyear(2022)
print(ycal,file=f,flush=True)
```

Output

Output is saved insdie yearcal.html file