datetime module

it is a predefined module which comes with python software
this module is used to work with date and time
The datetime module supplies classes for manipulating dates and times.
While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.
The following classes provided by this module

1. date class
2. time class
3. datetime class
4. timedelta class
5. tzinfo class

date and time classes are immutable data types.

date class or date data type
A date object represents a date (year, month and day)
class datetime.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.

***classmethod* date.today()**
Return the current local date.

```
>>> import datetime
>>> datetime.date.today()
datetime.date(2022, 1, 7)
>>> d=datetime.date.today()
>>> print(d)
2022-01-07
>>> d.year
2022
>>> d.month
1
```

```
>>> d.day
7
>>>
```

time Objects

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

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

All arguments are optional

0 <= hour < 24,
0 <= minute < 60,
0 <= second < 60,
0 <= microsecond < 1000000

```
>>> import datetime
>>> d1=datetime.date(1974,1,10)
>>> d1
datetime.date(1974, 1, 10)
>>> print(d1)
1974-01-10
>>> print(d1.day,d1.month,d1.year)
10 1 1974
>>> t1=datetime.time()
>>> print(t1)
00:00:00
>>> t2=datetime.time(12,30,40)
>>> print(t2)
12:30:40
>>> print(t2.hour,t2.minute,t2.second)
12 30 40
>>>
```

timedelta Objects

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

class 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:

```
import datetime
def main():
    d1=datetime.date(2010,6,12)
    print(d1)
    days=datetime.timedelta(4)
    d1=d1+days
    print(d1)
    month=datetime.timedelta(weeks=4)
    d1=d1+month
    print(d1)
    year=datetime.timedelta(days=365)
    d1=d1+year
    print(d1)
    d1=d1-days
    print(d1)
    d1=d1-month
    print(d1)
    d1=d1-year
    print(d1)

main()
```

Output:

```
2010-06-12
2010-06-16
```

2010-07-14
2011-07-14
2011-07-10
2011-06-12
2010-06-12

datetime Objects

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

class datetime.datetime(*year*, *month*, *day*, *hour=0*, *minute=0*, *second=0*, *microsecond=0*)

```
>>> d1=datetime.datetime(2022,1,8)
>>> print(d1)
2022-01-08 00:00:00
>>> d2=datetime.datetime(2022,1,8,6,45,23)
>>> print(d2)
2022-01-08 06:45:23
>>> d3=datetime.datetime.today()
>>> print(d3)
2022-01-08 18:44:49.989881
>>> print(d3.year,d3.month,d3.day)
2022 1 8
>>> print(d3.hour,d3.minute,d3.second)
18 44 49
>>>
```

```
>>> dt1=datetime.datetime.today()
>>> print(dt1)
2022-01-08 18:48:59.396697
>>> td=datetime.timedelta(hours=1)
>>> dt1=dt1+td
>>> print(dt1)
2022-01-08 19:48:59.396697
>>>
```

Formatting date and time
strftime() is a predefined method of datetime object.

This method is used to format date and time

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 on the left.	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 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)

```
>>> td=datetime.timedelta(hours=1)
>>> dt1=dt1+td
>>> print(dt1)
```

```
2022-01-08 19:48:59.396697
>>> dt=datetime.datetime.today()
>>> print(dt)
2022-01-08 18:55:16.566477
>>> print(dt.strftime("%a"))
Sat
>>> print(dt.strftime("%A"))
Saturday
>>> print(dt.strftime("%A %d"))
Saturday 08
>>> print(dt.strftime("%A %d %B"))
Saturday 08 January
>>> print(dt.strftime("%A %d %b"))
Saturday 08 Jan
>>> print(dt.strftime("%d-%m"))
08-01
>>> print(dt.strftime("%d-%m-%Y"))
08-01-2022
>>> print(dt.strftime("%A %d %B %Y"))
Saturday 08 January 2022
>>> print(dt.strftime("%H"))
18
>>> print(dt.strftime("%I"))
06
>>> print(dt.strftime("%I:%M:%S %p"))
06:55:16 PM
>>> print(dt.strftime("%j"))
008
>>> print(dt.strftime("%U"))
01
>>> print(dt.strftime("%c"))
Sat Jan  8 18:55:16 2022
>>> print(dt.strftime("%x"))
01/08/22
>>> print(dt.strftime("%X"))
18:55:16
>>>
```

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.
This module provides 3 classes

1. Calendar
2. TextCalendar
3. HTMLCalendar

>>> import calendar
>>> print(calendar.month(2022,1))
January 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.calendar(2022))
2022

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 1 2 3 4 5 6 1 2 3 4 5 6
3 4 5 6 7 8 9 7 8 9 10 11 12 13 7 8 9 10 11 12 13
10 11 12 13 14 15 16 14 15 16 17 18 19 20 14 15 16 17 18 19 20
17 18 19 20 21 22 23 21 22 23 24 25 26 27 21 22 23 24 25 26 27
24 25 26 27 28 29 30 28 28 29 30 31
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 1 1 2 3 4 5
4 5 6 7 8 9 10 2 3 4 5 6 7 8 6 7 8 9 10 11 12
11 12 13 14 15 16 17 9 10 11 12 13 14 15 13 14 15 16 17 18 19
18 19 20 21 22 23 24 16 17 18 19 20 21 22 20 21 22 23 24 25 26
25 26 27 28 29 30 23 24 25 26 27 28 29 27 28 29 30
30 31

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

October

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					

November

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			

December

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