datetime module

datetime module is a predefined library. In python every program is called one module. Datetime is predefined module or program with predefined datetime functionality.
datetime module is a default module which comes with python software.
The datetime module supplies classes for manipulating dates and times.

```
>>> import datetime
>>>
```

datetime module provides the following classes or data types.

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

date datatype or class
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.

Date object is created with following attributes/variables/properties.

1. Year
2. Month
3. Day

Example:

```
import datetime

date1=datetime.date(2023,9,10)
print(type(date1))
print(date1)
print(f'Day {date1.day}')
print(f'Month {date1.month}')
print(f'Year {date1.year}')
```

Output:

```
<class 'datetime.date'>
2023-09-10
Day 10
Month 9
Year 2023
```

date.today()
Return the current local date (System date).

```
# write a code which return current date or system date

import datetime

current_date=datetime.date.today()
print(current_date)
```

time class

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*, *tzinfo=None*)

All arguments are optional. *tzinfo* may be None, or an instance of a tzinfo subclass. The remaining arguments must be integers in the following ranges:

1. 0 <= hour < 24,
2. 0 <= minute < 60,

3. 0 <= second < 60,
4. 0 <= microsecond < 1000000,

Time object is created with following attributes or properties

1. Hour
2. Minute
3. Second
4. Microsecond
5. Tzinfo

Example:

```
# Write a program to create time object

import datetime

t1=datetime.time()
print(t1)
t2=datetime.time(10,12,49)
print(t2)
print(f'Hours {t1.hour}')
print(f'Minutes {t1.minute}')
print(f'Seconds {t1.second}')
print(f'Hours {t2.hour}')
print(f'Minute {t2.minute}')
print(f'Seconds {t2.minute}')
```

Output:

```
00:00:00
10:12:49
Hours 0
Minutes 0
Seconds 0
Hours 10
Minute 12
Seconds 12
```

Timezone

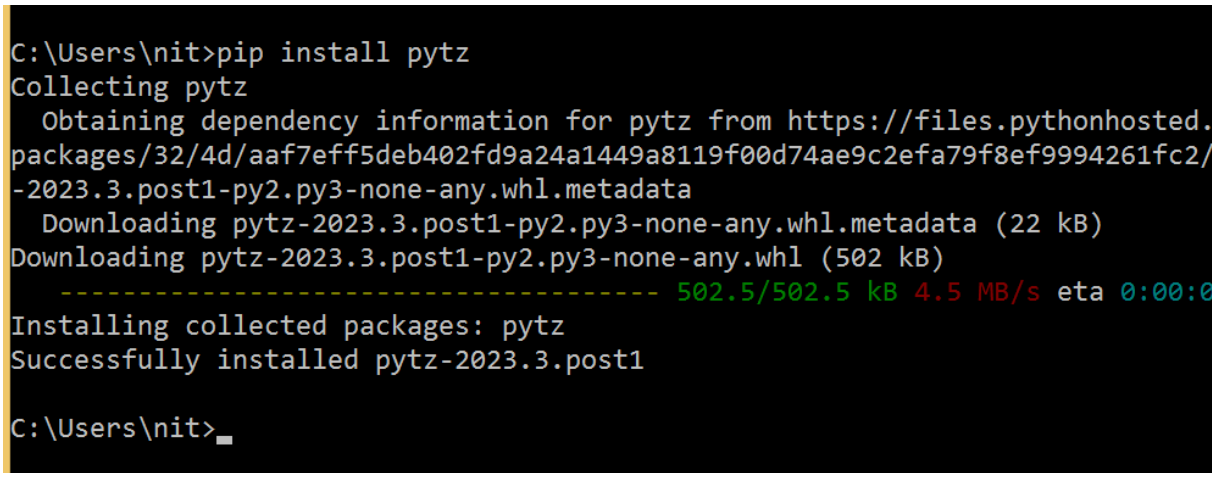
What is timezone in Python?

A time zone represents the standardized time depending on which part of the world is being considered.

In simple terms, timezone refers to the local time of a region. UTC (Coordinated Universal Time) is the astronomical time based on earth's rotation, is the standard against which the world's region-based time is coordinated.

How work with timezone?

Install pytz module



Pytz is third party module, which implements functionality of timezone

pytz.timezone('region'): Create the timezone object of a particular region

How to find timezone information?

>>> pytz.all_timezones
['Africa/Abidjan', 'Africa/Accra', 'Africa/Addis_Ababa', 'Africa/Algiers', 'Africa/Asmara', 'Africa/Asmera', 'Africa/Bamako', 'Africa/Bangui', 'Africa/Banjul', 'Africa/Bissau', 'Africa/Blantyre', 'Africa/Brazzaville', 'Africa/Bujumbura', 'Africa/Cairo', 'Africa/Casablanca', 'Africa/Ceuta', 'Africa/Conakry', 'Africa/Dakar', 'Africa/Dar_es_Salaam', 'Africa/Djibouti', 'Africa/Douala', 'Africa/El_Aaiun', 'Africa/Freetown', 'Africa/Gaborone', 'Africa/Harare', 'Africa/Johannesburg', 'Africa/Juba', 'Africa/Kampala', 'Africa/Khartoum', 'Africa'...]

Example:

```
# Creating time using timezone

import datetime
import pytz

tz=pytz.timezone("US/Central")
print(tz)
t1=datetime.time(10,5,30,0,tz)
print(t1)
```

datetime

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, tzinfo=None)

datetime.today()

Return the current local datetime, with tzinfo None.

datetime.now(*tz=None*)

Return the current local date and time.

Example:

```
import datetime
import pytz

dt1=datetime.datetime.today()
print(dt1)
dt2=datetime.datetime.now()
print(dt2)
tz=pytz.timezone("US/Central")
dt3=datetime.datetime.now(tz)
print(dt3)
```

Output:

```
2023-09-10 22:52:20.633309
2023-09-10 22:52:20.782410
```

2023-09-10 12:22:20.982543-05:00

Example:

```
import datetime

dt1=datetime.datetime(2023,9,10)
print(dt1)
dt2=datetime.datetime(2023,9,10,10,50,40)
print(dt2)
d1=dt2.date()
t1=dt2.time()
print(d1)
print(t1)
print(d1.year,d1.month,d1.day)
print(t1.hour,t1.minute,t1.second)
```

Output:

```
2023-09-10 00:00:00
2023-09-10 10:50:40
2023-09-10
10:50:40
2023 9 10
10 50 40
```

timedelta

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

d1=datetime.date.today()
print(d1)
days=3
d1=d1+datetime.timedelta(days=days)
print(d1)
weeks=4
d1=d1+datetime.timedelta(weeks=weeks)
print(d1)
weeks=52
d1=d1+datetime.timedelta(weeks=weeks)
print(d1)
d1=d1-datetime.timedelta(days=days)
print(d1)
d1=d1-datetime.timedelta(weeks=4)
print(d1)
d1=d1-datetime.timedelta(weeks=52)
print(d1)
```

Output:

```
2023-09-10
2023-09-13
2023-10-11
2024-10-09
2024-10-06
2024-09-08
2023-09-10
```

strftime()

it is predefined function of datetime class. It 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 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)
%%	A literal '%' character.	%

Example:

```
import datetime

dt=datetime.datetime.today()
print(dt)
print(dt.strftime("%a"))
print(dt.strftime("%A"))
print(dt.strftime("%A %d"))
print(dt.strftime("%w"))
print(dt.strftime("%A %d %b"))
print(dt.strftime("%A %d %B"))
print(dt.strftime("%A %d %B %Y"))
print(dt.strftime("%d/%m/%Y"))
print(dt.strftime("%H:%M:%S"))
print(dt.strftime("%A %d %B %Y %H:%M:%S"))
print(dt.strftime("%I:%M:%S %p"))
print(dt.strftime("%j"))
print(dt.strftime("%U"))
print(dt.strftime("%c"))
print(dt.strftime("%x"))
print(dt.strftime("%X"))
```

Output:

```
2023-09-10 23:35:49.229210
Sun
Sunday
Sunday 10
0
Sunday 10 Sep
```

Sunday 10 September
Sunday 10 September 2023
10/09/2023
23:35:49
Sunday 10 September 2023 23:35:49
11:35:49 PM
253
37
Sun Sep 10 23:35:49 2023
09/10/23
23:35:49