




PYTHON



DATE



Python provides the **datetime** module work with real dates and times. In real-world applications, we need to work with the date and time. Python enables us to schedule our Python script to run at a particular timing.

In Python, the date is not a data type, but we can work with the date objects by importing the module named with **datetime, time, and calendar**.

In this section of the tutorial, we will discuss how to work with the date and time objects in Python.

THE **DATETIME** CLASSES ARE CLASSIFIED IN THE SIX MAIN CLASSES.

date - It is a naive ideal date. It consists of the year, month, and day as attributes.

time - It is a perfect time, assuming every day has precisely $24*60*60$ seconds. It has hour, minute, second, microsecond, and **tzinfo** as attributes.

datetime - It is a grouping of date and time, along with the attributes year, month, day, hour, minute, second, microsecond, and tzinfo.

timedelta - It represents the difference between two dates, time or datetime instances to microsecond resolution.

tzinfo - It provides time zone information objects.

timezone - It is included in the new version of Python. It is the class that implements the **tzinfo** abstract base class.

TICK

In Python, the time instants are counted since 12 AM, 1st January 1970. The function **time()** of the module `time` returns the total number of ticks spent since 12 AM, 1st January 1970. A tick can be seen as the smallest unit to measure the time.



```
import time;
```

```
#prints the number of ticks spent since 12 AM, 1st January 1970
```

```
print(time.time())
```

HOW TO GET THE CURRENT TIME?

The `localtime()` functions of the `time` module are used to get the current time tuple. Consider the following example.

```
import time;

#returns a time tuple

print(time.localtime(time.time()))
```

THE TIME IS TREATED AS THE TUPLE OF 9 NUMBERS.

Index	Attribute	Values
0	Year	4 digit (for example 2018)
1	Month	1 to 12
2	Day	1 to 31
3	Hour	0 to 23
4	Minute	0 to 59
5	Second	0 to 60
6	Day of weak	0 to 6
7	Day of year	1 to 366
8	Daylight savings	-1, 0, 1 , or -1

GETTING FORMATTED TIME

The time can be formatted by using the **asctime()** function of the time module. It returns the formatted time for the time tuple being passed.

```
import time
```

```
    #returns the formatted time
```

```
print(time.asctime(time.localtime(time.time())))
```

PYTHON SLEEP TIME

The **sleep()** method of time module is used to stop the execution of the script for a given amount of time. The output will be delayed for the number of seconds provided as the float.

```
import time
```

```
for i in range(0,5):
```

```
    print(i)
```

```
    #Each element will be printed after 1 second
```

```
    time.sleep(1)
```

THE DATETIME MODULE

The **datetime** module enables us to create the custom date objects, perform various operations on dates like the comparison, etc.

To work with dates as date objects, we have to import **the datetime** module into the python source code.

Consider the following example to get the **datetime** object representation for the current time.

```
import datetime
```

```
#returns the current datetime object
```

```
print(datetime.datetime.now())
```



DATETIME MODULE

CREATING DATE OBJECTS


We can create the date objects bypassing the desired date in the datetime constructor for which the date objects are to be created.

Example

```
import datetime
```

```
#returns the datetime object for the specified date
```

```
print(datetime.datetime(2020,04,04))
```



We can also specify the time along with the date to create the datetime object. Consider the following example.

```
import datetime  
  
#returns the datetime object for the specified time  
  
print(datetime.datetime(2020,4,4,1,26,40))
```

In the above code, we have passed in **datetime()** function year, month, day, hour, minute, and millisecond attributes in a sequential manner.

COMPARISON OF TWO DATES

We can compare two dates by using the comparison operators like `>`, `>=`, `<`, and `<=`.

```
from datetime import datetime as dt
```

```
#Compares the time. If the time is in between 8AM and 4PM, then it prints working h  
ours otherwise it prints fun hours
```

```
if dt(dt.now().year,dt.now().month,dt.now().day,8)<dt.now()<dt(dt.now().year,dt.now().  
month,dt.now().day,16):
```

```
    print("Working hours....")
```

```
else:
```

```
    print("fun hours")
```



THE CALENDAR MODULE



Python provides a calendar object that contains various methods to work with the calendars.

Consider the following example to print the calendar for the last month of 2018.

```
import calendar;
cal = calendar.month(2020,3)
#printing the calendar of December 2018
print(cal)
```

PRINTING THE CALENDAR OF WHOLE YEAR

The `prcal()` method of `calendar` module is used to print the calendar of the entire year. The year of which the calendar is to be printed must be passed into this method.

```
import calendar
```

```
#printing the calendar of the year 2019
```

```
s = calendar.prcal(2020)
```



CONTINUE IN NEXT UNIT