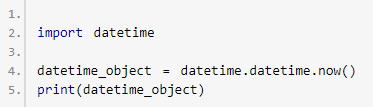
**CHAPTER 9**

**PYTHON DATE AND TIME**

Python has a module named **datetime** to work with dates and times. Let's create a few simple programs related to date and time before we dig deeper.

**Example 1: Get Current Date and Time**

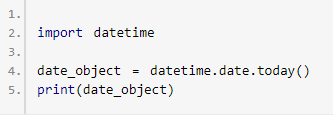


When you run the program, the output will be something like:

2018-12-19 09:26:03.478039

Here, we have imported **datetime** module using import datetime statement. One of the classes defined in the datetime module is datetime class. We then used now()method to create a datetime object containing the current local date and time.

**Example 2: Get Current Date**



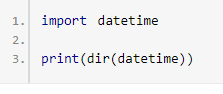
When you run the program, the output will be something like:

2018-12-19

In this program, we have used today() method defined in the date class to get a dateobject containing the current local date.

**What's inside datetime?**

We can use [dir()](https://www.programiz.com/python-programming/methods/built-in/dir) function to get a list containing all attributes of a module.



When you run the program, the output will be:

['MAXYEAR', 'MINYEAR', '\_\_builtins\_\_', '\_\_cached\_\_', '\_\_doc\_\_', '\_\_file\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_spec\_\_', '\_divide\_and\_round', 'date', 'datetime', 'datetime\_CAPI', 'time', 'timedelta', 'timezone', 'tzinfo']

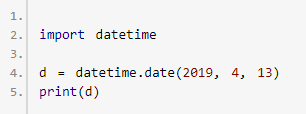
Commonly used classes in the datetime module are:

* date Class
* time Class
* datetime Class
* timedelta Class

**datetime.date Class**

You can instantiate date objects from the date class. A date object represents a date (year, month and day).

**Example 3: Date object to represent a date**

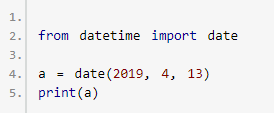


When you run the program, the output will be:

2019-04-13

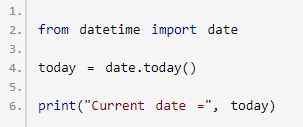
If you are wondering, date() in the above example is a constructor of the date class. The constructor takes three arguments: year, month and day. The variable a is a date object.

We can only import date class from the datetime module. Here's how:



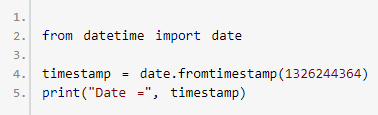
**Example 4: Get current date**

You can create a date object containing the current date by using a classmethod named today(). Here's how:



**Example 5: Get date from a timestamp**

We can also create date objects from a timestamp. A Unix timestamp is the number of seconds between a particular date and January 1, 1970 at UTC. You can convert a timestamp to date using fromtimestamp() method.

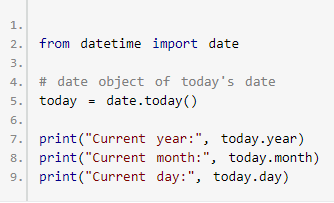


When you run the program, the output will be:

Date = 2012-01-11

**Example 6: Print today's year, month and day**

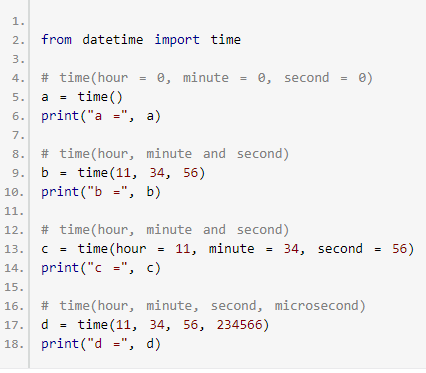
We can get year, month, day, day of the week etc. from the date object easily. Here's how:



**datetime.time**

A time object instantiated from the time class represents the local time.

**Example 7: Time object to represent time**



When you run the program, the output will be:

a = 00:00:00

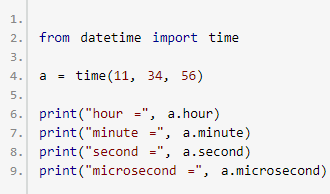
b = 11:34:56

c = 11:34:56

d = 11:34:56.234566

**Example 8: Print hour, minute, second and microsecond**

Once you create a time object, you can easily print its attributes such as hour, minute etc.



When you run the example, the output will be:

hour = 11

minute = 34

second = 56

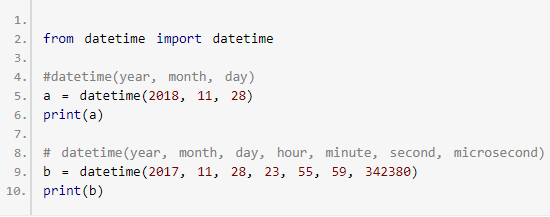
microsecond = 0

Notice that we haven't passed microsecond argument. Hence, its default value 0 is printed.

**datetime.datetime**

The datetime module has a class named dateclass that can contain information from both **date** and **time** objects.

**Example 9: Python datetime object**



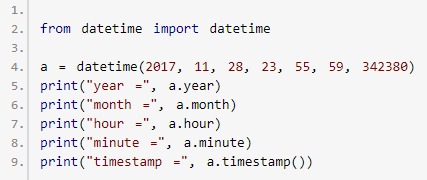
When you run the program, the output will be:

2018-11-28 00:00:00

2017-11-28 23:55:59.342380

The first three arguments year, month and day in the datetime() constructor are mandatory.

**Example 10: Print year, month, hour, minute and timestamp**



When you run the program, the output will be:

year = 2017

month = 11

day = 28

hour = 23

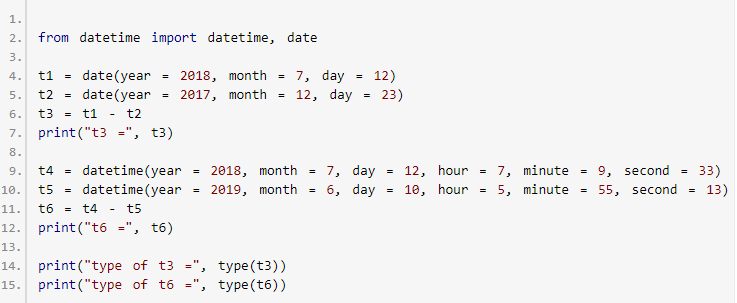
minute = 55

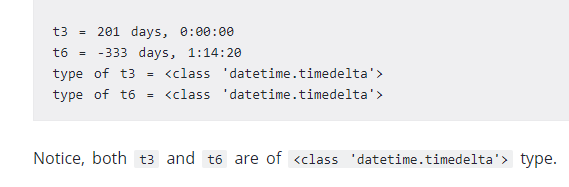
timestamp = 1511913359.34238

**datetime.timedelta**

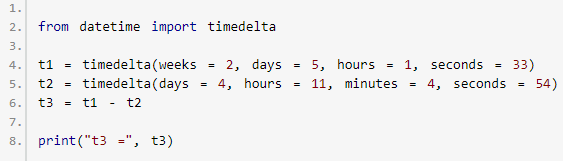
A timedelta object represents the difference between two dates or times.

**Example 11: Difference between two dates and times**

****When you run the program, the output will be:



**Example 12: Difference between two timedelta objects**

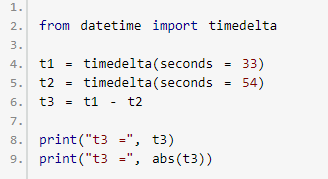


When you run the program, the output will be:

t3 = 14 days, 13:55:39

Here, we have created two timedelta objects t1 and t2, and their difference is printed on the screen.

**Example 13: Printing negative timedelta object**



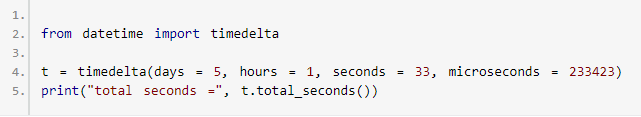
When you run the program, the output will be:

t3 = -1 day, 23:59:39

t3 = 0:00:21

**Example 14: Time duration in seconds**

You can get the total number of seconds in a timedelta object using total\_seconds() method.



When you run the program, the output will be: total seconds = 435633.233423

You can also find sum of two dates and times using + operator. Also, you can multiply and divide a timedelta object by integers and floats.

**Python format datetime**

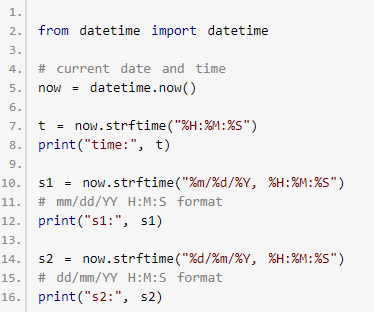
The way date and time is represented may be different in different places, organizations etc. It's more common to use mm/dd/yyyy in the US, whereas dd/mm/yyyy is more common in the UK.

Python has strftime() and strptime() methods to handle this.

**Python strftime() - datetime object to string**

The strftime() method is defined under classes date, datetime and time. The method creates a formatted string from a given date, datetime or time object.

**Example 15: Format date using strftime()**



When you run the program, the output will be something like:

time: 04:34:52

s1: 12/26/2018, 04:34:52

s2: 26/12/2018, 04:34:52

Here, %Y, %m, %d, %H etc. are format codes. The strftime() method takes one or more format codes and returns a formatted string based on it.

In the above program, t, s1 and s2 are strings.

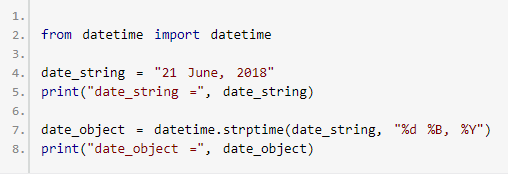
* %Y - year [0001,..., 2018, 2019,..., 9999]
* %m - month [01, 02, ..., 11, 12]
* %d - day [01, 02, ..., 30, 31]
* %H - hour [00, 01, ..., 22, 23
* %M - month [00, 01, ..., 58, 59]
* %S - second [00, 01, ..., 58, 59]

To learn more about strftime() and format codes, visit: [Python strftime()](https://www.programiz.com/python-programming/datetime/strftime).

**Python strptime() - string to datetime**

The strptime() method creates a datetime object from a given string (representing date and time).

**Example 16: strptime()**



When you run the program, the output will be:

date\_string = 21 June, 2018

date\_object = 2018-06-21 00:00:00

The strptime() method takes two arguments:

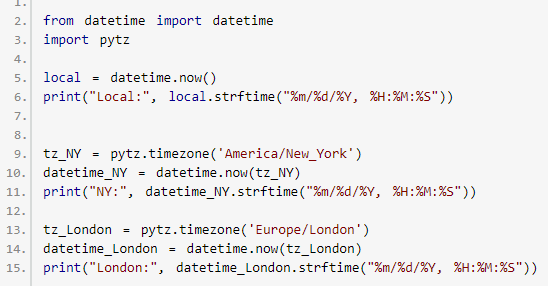
1. a string representing date and time
2. format code equivalent to the first argument

By the way, %d, %B and %Y format codes are used for day, month(full name) and yearrespectively.

Visit [Python strptime()](https://www.programiz.com/python-programming/datetime/strptime) to learn more.

**Handling timezone in Python**

Suppose, you are working on a project and need to display date and time based on their timezone. Rather than trying to handle timezone yourself, we suggest you to use a third-party [pytZ module](http://pytz.sourceforge.net/).



When you run the program, the output will be something like

Local time: 2018-12-20 13:10:44.260462

America/New\_York time: 2018-12-20 13:10:44.260462

Europe/London time: 2018-12-20 13:10:44.260462

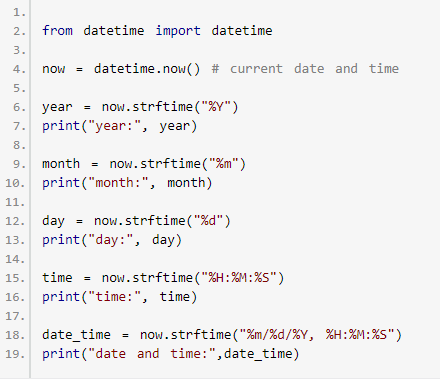
Here, datetime\_NY and datetime\_London are datetime objects containing the current date and time of their respective timezone.

# **Python strftime()**

The strftime() method returns a string representing date and time using [date](https://www.programiz.com/python-programming/datetime#date), [time](https://www.programiz.com/python-programming/datetime#time) or [datetime](https://www.programiz.com/python-programming/datetime" \l "datetime) object.

**Example 1: datetime to string using strftime()**

The program below converts a datetime object containing current date and time to different string formats.



When you run the program, the output will something like be:

year: 2018

month: 12

day: 24

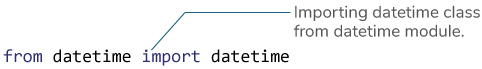
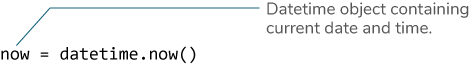
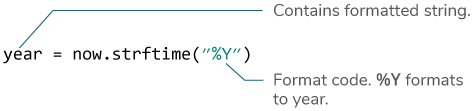
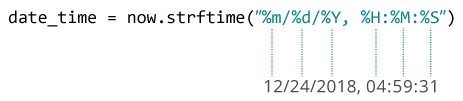
time: 04:59:31

date and time: 12/24/2018, 04:59:31

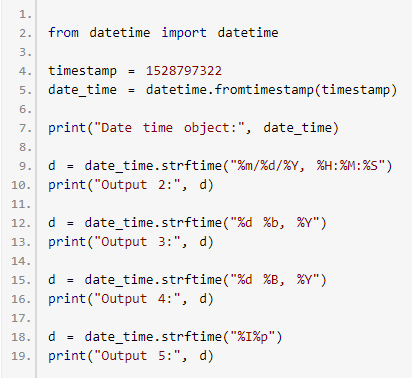
Here, year, day, time and date\_time are strings, whereas now is a datetime object.

**How strftime() works?**

In the above program, %Y, %m, %d etc. are format codes. The strftime() method takes one or more format codes as an argument and returns a formatted string based on it.

1. We imported datetime class from the datetime module. It's because the object of datetime class can access strftime() method.  
     
   
2. The datetime object containing current date and time is stored in now variable.  
     
   
3. The strftime() method can be used to create formatted strings.  
     
   
4. The string you pass to the strftime() method may contain more than one format codes.  
     
   

**Example 2: Creating string from a timestamp**



When you run the program, the output will be:

Date time object: 2018-06-12 09:55:22

Output 2: 06/12/2018, 09:55:22

Output 3: 12 Jun, 2018

Output 4: 12 June, 2018

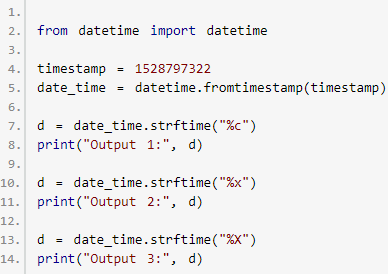
Output 5: 09AM

**Format Code List**

The table below shows all the codes that you can pass to the strftime() method.

| **Directive** | **Meaning** | **Example** |
| --- | --- | --- |
| %a | Abbreviated weekday name. | Sun, Mon, ... |
| %A | Full weekday name. | Sunday, Monday, ... |
| %w | Weekday as a decimal number. | 0, 1, ..., 6 |
| %d | Day of the month as a zero-padded decimal. | 01, 02, ..., 31 |
| %-d | Day of the month as a decimal number. | 1, 2, ..., 30 |
| %b | Abbreviated month name. | Jan, Feb, ..., Dec |
| %B | Full month name. | January, February, ... |
| %m | Month as a zero-padded decimal number. | 01, 02, ..., 12 |
| %-m | Month as a decimal number. | 1, 2, ..., 12 |
| %y | Year without century as a zero-padded decimal number. | 00, 01, ..., 99 |
| %-y | Year without century as a decimal number. | 0, 1, ..., 99 |
| %Y | Year with century as a decimal number. | 2013, 2019 etc. |
| %H | Hour (24-hour clock) as a zero-padded decimal number. | 00, 01, ..., 23 |
| %-H | Hour (24-hour clock) as a decimal number. | 0, 1, ..., 23 |
| %I | Hour (12-hour clock) as a zero-padded decimal number. | 01, 02, ..., 12 |
| %-I | Hour (12-hour clock) as a decimal number. | 1, 2, ... 12 |
| %p | Locale’s AM or PM. | AM, PM |
| %M | Minute as a zero-padded decimal number. | 00, 01, ..., 59 |
| %-M | Minute as a decimal number. | 0, 1, ..., 59 |
| %S | Second as a zero-padded decimal number. | 00, 01, ..., 59 |
| %-S | Second as a decimal number. | 0, 1, ..., 59 |
| %f | Microsecond as a decimal number, zero-padded on the left. | 000000 - 999999 |
| %z | UTC offset in the form +HHMM or -HHMM. |  |
| %Z | Time zone name. |  |
| %j | Day of the year as a zero-padded decimal number. | 001, 002, ..., 366 |
| %-j | Day of the year as a decimal number. | 1, 2, ..., 366 |
| %U | Week number of the year (Sunday as the first day of the week). 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). 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. | Mon Sep 30 07:06:05 2013 |
| %x | Locale’s appropriate date representation. | 09/30/13 |
| %X | Locale’s appropriate time representation. | 07:06:05 |
| %% | A literal '%' character. | % |

**Example 3: Locale's appropriate date and time**



When you run the program, the output will be:

Output 1: Tue Jun 12 09:55:22 2018

Output 2: 06/12/18

Output 3: 09:55:22

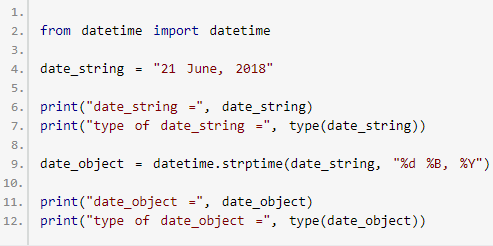
Format codes %c, %x and %X are used for locale's appropriate date and time representation. We also recommend you to check [Python strptime()](https://www.programiz.com/python-programming/datetime/strptime). The strptime() method creates a datetime object from a string.

# **Python strptime()**

The strptime() method creates a [datetime](https://www.programiz.com/python-programming/datetime" \l "datetime) object from a given string.

**Note:** You cannot create datetime object from every string. The string needs to be in a certain format.

**Example 1: string to datetime object**



When you run the program, the output will be:

date\_string = 21 June, 2018

type of date\_string = <class 'str'>

date\_object = 2018-06-21 00:00:00

type of date\_object = <class 'datetime.datetime'>

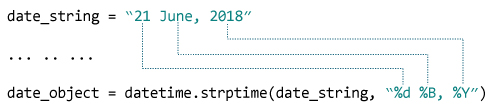
**How strptime() works?**

The strptime() class method takes two arguments:

* string (that be converted to datetime)
* format code

Based on the string and format code used, the method returns its equivalent datetimeobject.

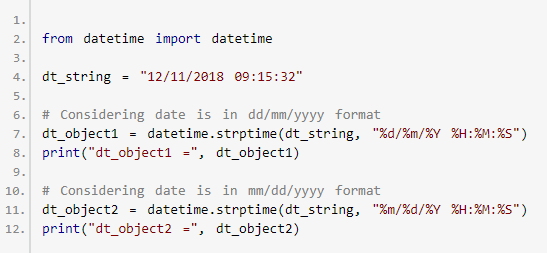
In the above example:



Here,

* %d - Represents day of the month. **Example:** 01, 02, ..., 31
* %B - Month's name in full. **Example:** January, February etc.
* %Y - Year in four digits. **Example:** 2018, 2019 etc.

**Example 2: string to datetime object**



When you run the program, the output will be:

dt\_object1 = 2018-11-12 09:15:32

dt\_object2 = 2018-12-11 09:15:32

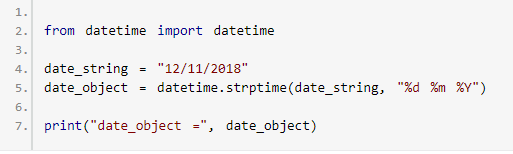
**Format Code List**

The table below shows all the format codes that you can use.

| **Directive** | **Meaning** | **Example** |
| --- | --- | --- |
| %a | Abbreviated weekday name. | Sun, Mon, ... |
| %A | Full weekday name. | Sunday, Monday, ... |
| %w | Weekday as a decimal number. | 0, 1, ..., 6 |
| %d | Day of the month as a zero-padded decimal. | 01, 02, ..., 31 |
| %-d | Day of the month as a decimal number. | 1, 2, ..., 30 |
| %b | Abbreviated month name. | Jan, Feb, ..., Dec |
| %B | Full month name. | January, February, ... |
| %m | Month as a zero-padded decimal number. | 01, 02, ..., 12 |
| %-m | Month as a decimal number. | 1, 2, ..., 12 |
| %y | Year without century as a zero-padded decimal number. | 00, 01, ..., 99 |
| %-y | Year without century as a decimal number. | 0, 1, ..., 99 |
| %Y | Year with century as a decimal number. | 2013, 2019 etc. |
| %H | Hour (24-hour clock) as a zero-padded decimal number. | 00, 01, ..., 23 |
| %-H | Hour (24-hour clock) as a decimal number. | 0, 1, ..., 23 |
| %I | Hour (12-hour clock) as a zero-padded decimal number. | 01, 02, ..., 12 |
| %-I | Hour (12-hour clock) as a decimal number. | 1, 2, ... 12 |
| %p | Locale’s AM or PM. | AM, PM |
| %M | Minute as a zero-padded decimal number. | 00, 01, ..., 59 |
| %-M | Minute as a decimal number. | 0, 1, ..., 59 |
| %S | Second as a zero-padded decimal number. | 00, 01, ..., 59 |
| %-S | Second as a decimal number. | 0, 1, ..., 59 |
| %f | Microsecond as a decimal number, zero-padded on the left. | 000000 - 999999 |
| %z | UTC offset in the form +HHMM or -HHMM. |  |
| %Z | Time zone name. |  |
| %j | Day of the year as a zero-padded decimal number. | 001, 002, ..., 366 |
| %-j | Day of the year as a decimal number. | 1, 2, ..., 366 |
| %U | Week number of the year (Sunday as the first day of the week). 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). 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. | Mon Sep 30 07:06:05 2013 |
| %x | Locale’s appropriate date representation. | 09/30/13 |
| %X | Locale’s appropriate time representation. | 07:06:05 |
| %% | A literal '%' character. | % |

**ValueError in strptime()**

If the string (first argument) and the format code (second argument) passed to the strptime() doesn't match, you will get ValueError. For example:



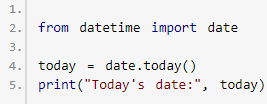
If you run this program, you will get an error.

ValueError: time data '12/11/2018' does not match format '%d %m %Y'

# **How to get current date and time in Python?**

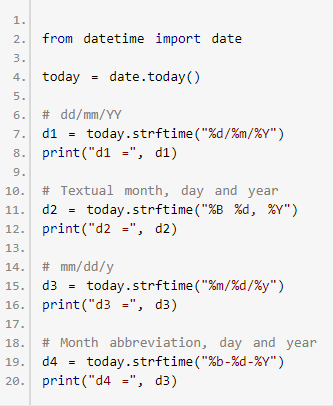
There are a number of ways you can take to get current date. We will use date class of the [datetime](https://www.programiz.com/python-programming/datetime) module to accomplish this task.

**Example 1: Python get today's date**



Here, we imported date class from the datetime module. Then, we used date.today()method to get the current local date. By the way, today variable will be a date object. You can use [strftime()](https://www.programiz.com/python-programming/datetime/strftime) method to create string representing date in different formats from this object.

**Example 2: Current date in different formats**



When you run the program, the output will be something like:

d1 = 25/12/2018

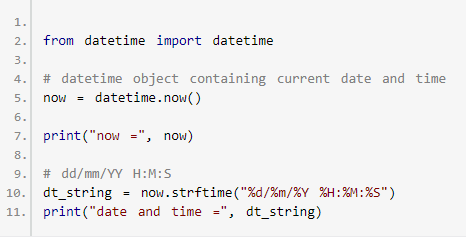
d2 = December 25, 2018

d3 = 12/25/18

d4 = 12/25/18

If you need to get current date and time, you can use datetime class of the datetimemodule.

**Example 3: Get current date and time**

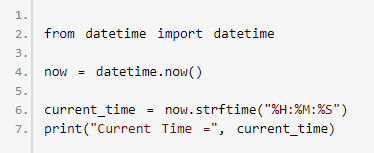


Here, we have used datetime.now() to get current date and time. Then, we used strftime()method to create a string representing date and time in another format.

# **Python Get Current time**

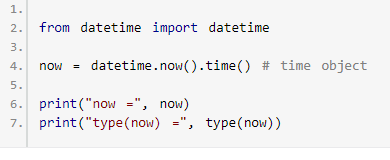
There are a number of ways you can take to get current time in Python.

**Example 1: Current time using datetime object**



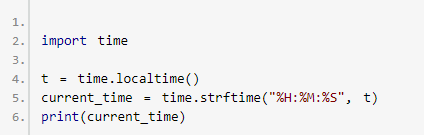
In the above example, we have imported datetime class from the [datetime](https://www.programiz.com/python-programming/datetime) module. Then, we used now() method to get a datetime object containing current date and time. Using [datetime.strftime()](https://www.programiz.com/python-programming/datetime/strftime) method, we then created a string representing current time.

If you need to create a time object containing current time, you can do something like this.



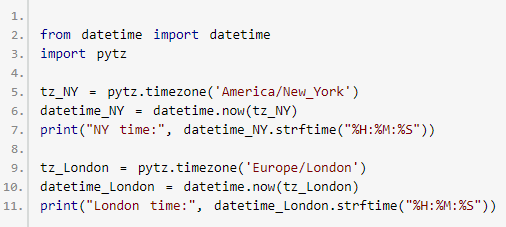
**Example 2: Current time using time module**

You can also get the current time using time module.



**Example 3: Current time of a timezone**

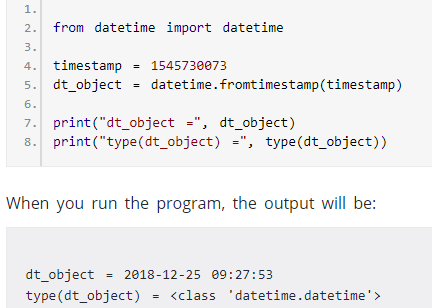
If you need to find current time of a certain timezone, you can use [pytZ module](http://pytz.sourceforge.net/).



# **Python timestamp to datetime and vice-versa**

It's pretty common to store date and time as a timestamp in a database. A Unix timestamp is the number of seconds between a particular date and January 1, 1970 at UTC.

**Example 1: Python timestamp to datetime**

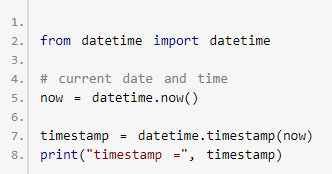
****

Here, we have imported datetime class from the [datetime](https://www.programiz.com/python-programming/datetime) module. Then, we used datetime.fromtimestamp() classmethod which returns the local date and time (datetime object). This object is stored in dt\_object variable.

**Note:** You can easily create a string representing date and time from a datetime object using [strftime()](https://www.programiz.com/python-programming/datetime/strftime) method.

**Example 2: Python datetime to timestamp**

You can get timestamp from a datetime object using datetime.timestamp() method.

****

**Python time Module**

Python has a module named time to handle time-related tasks. To use functions defined in the module, we need to import the module first. Here's how:

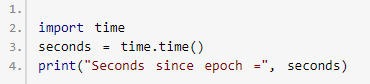
1. import time

Here are commonly used time-related functions.

**Python time.time()**

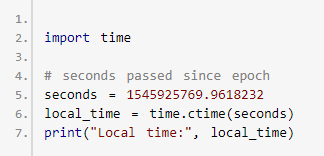
The time() function returns the number of seconds passed since epoch.

For Unix system, January 1, 1970, 00:00:00 at **UTC** is epoch (the point where time begins).



**Python time.ctime()**

The time.ctime() function takes seconds passed since epoch as an argument and returns a string representing local time.

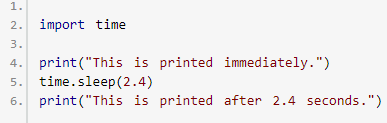


If you run the program, the output will be something like:

Local time: Thu Dec 27 15:49:29 2018

**Python time.sleep()**

The sleep() function suspends (delays) execution of the current thread for the given number of seconds.



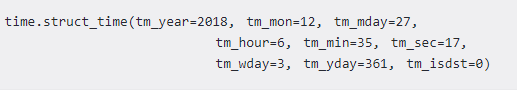
To learn more, visit: [Python sleep()](https://www.programiz.com/python-programming/time/sleep).

Before we talk about other time-related functions, let's explore time.struct\_time class in brief.

**time.struct\_time Class**

Several functions in the time module such as gmtime(), asctime() etc. either take time.struct\_time object as an argument or return it.

Here's an example of time.struct\_time object.

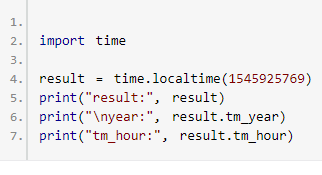


| Index | Attribute | Values |
| --- | --- | --- |
| 0 | tm\_year | 0000, ...., 2018, ..., 9999 |
| 1 | tm\_mon | 1, 2, ..., 12 |
| 2 | tm\_mday | 1, 2, ..., 31 |
| 3 | tm\_hour | 0, 1, ..., 23 |
| 4 | tm\_min | 0, 1, ..., 59 |
| 5 | tm\_sec | 0, 1, ..., 61 |
| 6 | tm\_wday | 0, 1, ..., 6; Monday is 0 |
| 7 | tm\_yday | 1, 2, ..., 366 |
| 8 | tm\_isdst | 0, 1 or -1 |

The values (elements) of the time.struct\_time object are accessible using both indices and attributes.

**Python time.localtime()**

The localtime() function takes the number of seconds passed since epoch as an argument and returns struct\_time in **local time**.



When you run the program, the output will be something like:

result: time.struct\_time(tm\_year=2018, tm\_mon=12, tm\_mday=27, tm\_hour=15, tm\_min=49, tm\_sec=29, tm\_wday=3, tm\_yday=361, tm\_isdst=0)

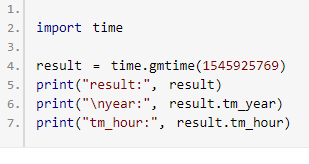
year: 2018

tm\_hour: 15

If no argument or None is passed to localtime(), the value returned by time() is used.

**Python time.gmtime()**

The gmtime() function takes the number of seconds passed since epoch as an argument and returns struct\_time in **UTC**.



When you run the program, the output will be:

result = time.struct\_time(tm\_year=2018, tm\_mon=12, tm\_mday=28, tm\_hour=8, tm\_min=44, tm\_sec=4, tm\_wday=4, tm\_yday=362, tm\_isdst=0)

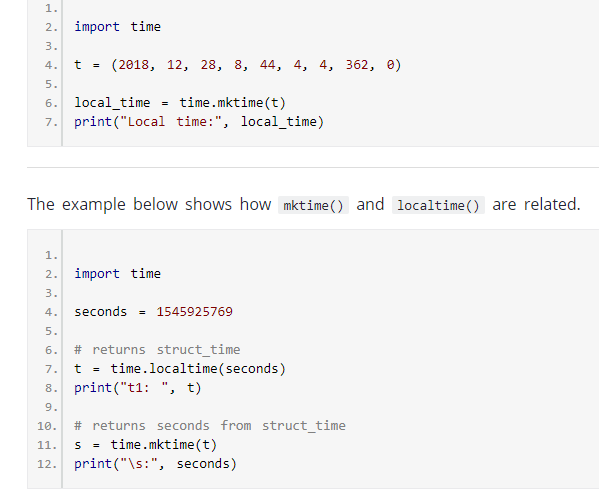
year = 2018

tm\_hour = 8

If no argument or None is passed to gmtime(), the value returned by time() is used.

**Python time.mktime()**

The mktime() function takes struct\_time (or a tuple containing 9 elements corresponding to struct\_time) as an argument and returns the seconds passed since epoch in local time. Basically, it's the inverse function of localtime().



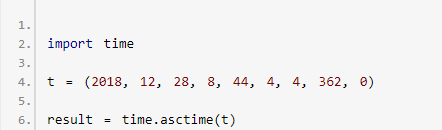
When you run the program, the output will be something like:

t1: time.struct\_time(tm\_year=2018, tm\_mon=12, tm\_mday=27, tm\_hour=15, tm\_min=49, tm\_sec=29, tm\_wday=3, tm\_yday=361, tm\_isdst=0)

s: 1545925769.0

**Python time.asctime()**

The asctime() function takes struct\_time (or a tuple containing 9 elements corresponding to struct\_time) as an argument and returns a string representing it. Here's an example:

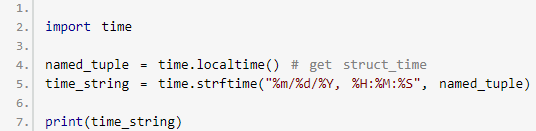


When you run the program, the output will be:

Result: Fri Dec 28 08:44:04 2018

**Python time.strftime()**

The strftime() function takes struct\_time (or tuple corresponding to it) as an argument and returns a string representing it based on the format code used. For example,



When you run the program, the output will be something like:

12/28/2018, 09:47:41

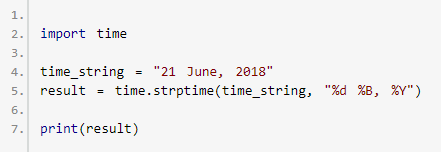
Here, %Y, %m, %d, %H etc. are format codes.

* %Y - year [0001,..., 2018, 2019,..., 9999]
* %m - month [01, 02, ..., 11, 12]
* %d - day [01, 02, ..., 30, 31]
* %H - hour [00, 01, ..., 22, 23
* %M - month [00, 01, ..., 58, 59]
* %S - second [00, 01, ..., 58, 61]

To learn more, visit: [time.strftime()](https://docs.python.org/3/library/time.html" \l "time.strftime).

**Python time.strptime()**

The strptime() function parses a string representing time and returns struct\_time.



When you run the program, the output will be:

time.struct\_time(tm\_year=2018, tm\_mon=6, tm\_mday=21, tm\_hour=0, tm\_min=0, tm\_sec=0, tm\_wday =3, tm\_yday=172, tm\_isdst=-1)

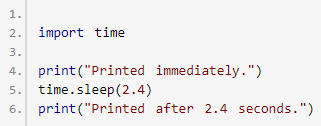
# **Python sleep()**

**The sleep() function suspends (waits) execution of the current thread for a given number of seconds.**

Python has a module named [time](https://www.programiz.com/python-programming/time) which provides several useful functions to handle time-related tasks. One of the popular functions among them is sleep().

The sleep() function suspends execution of the current thread for a given number of seconds.

**Example 1: Python sleep()**



Here's how this program works:

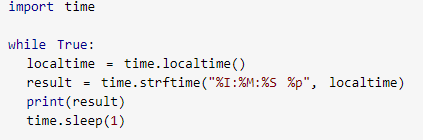
* "Printed immediately" is printed
* Suspends (Delays) execution for 2.4 seconds.
* "Printed after 2.4 seconds" is printed.

As you can see from the above example, sleep() takes a floating-point number as an argument.

**Before Python 3.5**, the actual suspension time may be less than the argument specified to the time() function.

**Since Python 3.5**, the suspension time will be at least the seconds specified.

**Example 2: Python create a digital clock**



In the above program, we computed and printed the current local time inside the infinite [while loop](https://www.programiz.com/python-programming/while-loop). Then, the program waits for 1 second. Again, the current local time is computed and printed. This process goes on.

When you run the program, the output will be something like:

02:10:50 PM

02:10:51 PM

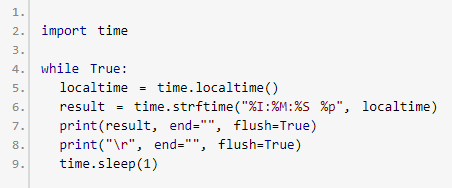
02:10:52 PM

02:10:53 PM

02:10:54 PM

... .. ...

Here is a slightly modified better version of the above program.



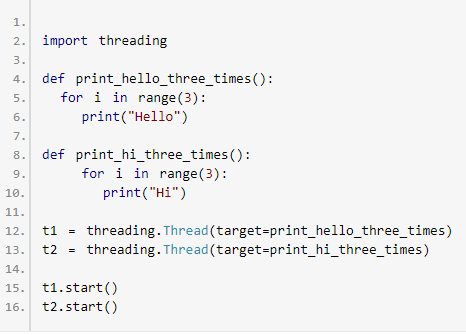
To learn more, visit [digital clock in Python shell](https://stackoverflow.com/questions/37515587/run-a-basic-digital-clock-in-the-python-shell)

**Multithreading in Python**

Before talking about sleep() in multithreaded programs, let's talk about processes and threads. A computer program is a collection of instructions. A process is the execution of those instructions. A thread is a subset of the process. A process can have one or more threads.

**Example 3: Python multithreading**

All the programs above in this article are single-threaded programs. Here's an example of a multithreaded Python program.



When you run the program, the output will be something like:

Hello

Hello

Hi

Hello

Hi

Hi

The above program has two threads t1 and t2. These threads are run using t1.start()and t2.start() statements.

Note that, t1 and t2 run concurrently and you might get different output.

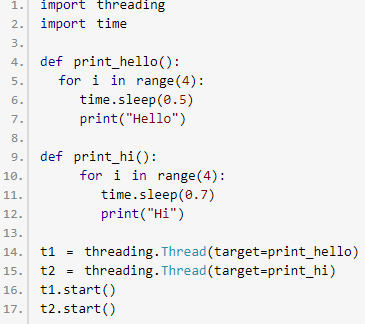
Visit this page to learn more about [Multithreading in Python](https://stackoverflow.com/questions/2846653/how-to-use-threading-in-python).

**time.sleep() in multithreaded programs**

The sleep() function suspends execution of the current thread for a given number of seconds.

In case of single-threaded programs, sleep() suspends execution of the thread and process. However, the function suspends a thread rather than the whole process in multithreaded programs.

**Example 4: sleep() in a multithreaded program**



The above program has two threads. We have used time.sleep(0.5) and time.sleep(0.75) to suspend execution of these two threads for 0.5 seconds and 0.7 seconds respectively