Module time in Python

Python has a time module used to handle time-related tasks. TipsMake.com will work with you to find out the details and functions related to the time specified in this module. Let's follow it!

The next tutorial in the series of lessons on handling Date / Time in Python, TipsMake.com will work with you to find out details about **module time** and functions related to the time specified in this module. Let's follow it!

Python has a **time module used to handle time-related tasks** . To use the functions defined in the module, first import this module, do the following:

```
import time
```

Here are the most commonly used time-related functions.

The most used functions

1. time.time ()
2. time.ctime ()
3. time.sleep ()
4. Class time.struct_time
5. time.localtime ()
6. time.gmtime ()
7. time.mktime ()
8. time.asctime ()
9. time.strftime ()
10. time.strptime ()

time.time ()

The time () function returns the number of seconds from the *epoch*, or the timestamp value

For Unix systems, 00:00:00 on January 1, 1970 UTC hours are called *epochs* (the start of time).

```
import time seconds = time.time() print("So giay tinh tu epoch:", seconds)
```

The result will look like:

```
So giay tinh tu epoch: 1562922590.7720907
```

time.ctime ()

This method converts a time represented by the number of seconds from the epoch to a string representation.

```
import time # s? giây tính t?
epoch # viet boi TipsMake.com seconds = 1562983783.9618232 local_time = time.cti
```

Run the program, the result returns the date and time corresponding to the number of seconds transmitted:

```
Local time: Sat Jul 13 09:09:43 2019
```

If not passing *seconds*, the program returns the current time value.

time.sleep ()

The sleep () function stops executing the current thread in the number of seconds passed.

```
import time print ("Start :", time.ctime()) time.sleep(3) print ("End :", time.ct
```

This method does not return any values, but only the executable delay. Try running the program to see the delay.

```
Start : Sat Jul 13 09:33:52 2019 End : Sat Jul 13 09:33:57 2019
```

Before going on to other time-related functions, learn through **class time.struct_time** .

Class time.struct_time

Some functions in module time, such as *gmtime (), asctime ()* . have *time.struct_time* is the returned object.

Example of *time.struct_time* result .

```
time.struct_time(tm_year=2018, tm_mon=12, tm_mday=27, tm_hour=6, tm_min=35, tm_se
```

Index
Properties
Describe
0 `tm_year` Current year: 0000, ., 2018, ., 9999 1 `tm_mon` Current month: 1, 2, ., 12 2 `tm_mday` Current day: 1, 2, ., 31 3 `tm_hour` Current time: 0, 1, ., 23 4 `tm_min` Current time: 0, 1, ., 59 5 `tm_sec` Current seconds: 0, 1, ., 61 6 `tm_wday` Day of the week : 0, 1, ., 6; Monday is calculated at 0 7 `tm_yday` Day of the year: 1, 2, ., 366 8 `tm_isdst` Determine DST: 0, 1 or -1

time.localtime ()

The localtime () function in the *time* module takes the number of seconds passed into the argument and returns *struct_time* in local time.

```
import time result = time.localtime(1562983783) print("Ket qua:", result) print("
```

Run the program, the result is:

```
Ket qua: time.struct_time(tm_year=2019, tm_mon=7, tm_mday=13, tm_hour=9, tm_min=9
```

If no number of seconds is provided or the value is transferred, the current time returned from the *time ()* function will be used.

time.gmtime ()

The function gmtime () in the *time* module takes the number of seconds passed as an argument and returns the *struct_time* by UTC time.

```
import time result = time.gmtime(1562983783) print("Ket qua:", result) print("nNa
```

Run the program, the result is:

```
Ket qua: time.struct_time(tm_year=2019, tm_mon=7, tm_mday=13, tm_hour=2, m_min=9,
```

If no number of seconds is provided or the value is transferred, the current time returned from the *time ()* function will be used.

time.mktime ()

The mktime () function in the *time* module takes *struct_time* (or a tuple containing 9 elements corresponding to *struct_time)* as the argument and returns the number of seconds from the local time epoch. This is the inverse function of *localtime ()*.

```
import time t = (2019, 7, 13, 9, 9, 43, 5, 194, 0) local_time = time.mktime(t) pr
```

Run the program, the result is:

```
 Gio dia phuong: 1562983783.0
```

The example below shows how *mktime ()* and *localtime ()* are related.

```
 import time seconds = 1562983783 # tr? v?
 struct_time # viet boi TipsMake.com t = time.localtime(seconds) print("t1: ", t
? v? giây t? struct_time s = time.mktime(t) print("ns:", seconds)
```

Result:

```
t1: time.struct_time(tm_year=2019, tm_mon=7, tm_mday=13, tm_hour=9, tm_min=9, tm
```

time.asctime ()

The asctime () function in the *time* module takes *struct_time* (or a tuple containing 9 elements corresponding to *struct_time)* as the argument and returns a string representing that time.

```
import time t = (2019, 7, 13, 9, 9, 43, 5, 194, 0) result = time.asctime(t) print
```

Program results returned:

```
 Ket qua: Sat Jul 13 09:09:43 2019
```

time.strftime ()

The strftime () function in the *time* module takes *struct_time* (or a tuple corresponding to *struct_time)* as the argument and returns a string representing the time based on the input format code.

```
 import time named_tuple = time.localtime() # l?
y struct_time time_string = time.strftime("%m/%d/%Y, %H:%M:%S", named_tuple) prin
```

Run the program, the result is:

```
 07/15/2019, 08:46:58
```

In this example,% Y,% m,% d,% H are **format codes** .

1. % Y: year [0001, ., 2018, 2019, ., 9999]
2. % m: month [01, 02, ., 11, 12]
3. % d: date [01, 02, ., 30, 31]
4. % H: hour [00, 01, ., 22, 23
5. % M: month [00, 01, ., 58, 59]
6. % S: seconds [00, 01, ., 58, 61]

Read more: strftime function () in the datetime Python module

time.strptime ()

The strptime () function in the *time* module analyzes a string representing a time, time, and returns *struct_time*.

```
import time time_string = "17 July, 2019" result = time.strptime(time_string, "%
```

The returned result has the form:

```
time.struct_time(tm_year=2019, tm_mon=7, tm_mday=17, tm_hour=0, tm_min=0, tm_sec=
```

Read more: strptime () function in the datetime Python module

Previous lesson: Convert the timestamp value in Python

Next lesson: Function sleep () in Python

You finished reading the article "**Module time in Python**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.