Python Timeit() with Examples

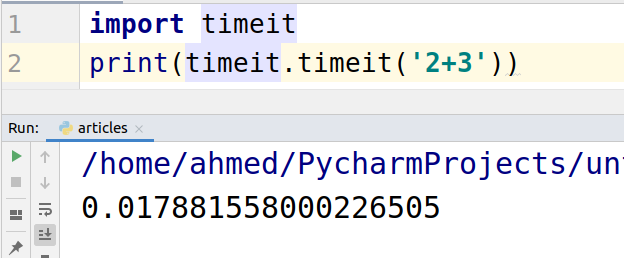
There are many algorithms to do a specific task. How do you measure which algorithm is better? One way is to compare the execution time of the algorithms. For measuring the execution time of the algorithm there is a built-in function in Python. Timeit runs the given code statements 1 million times and delivers the minimum time take from the set. Let’s take a look at the syntax of the function.

|  |
| --- |
| timeit.timeit(stmt, setup,timer, number) |

The first input “stmt” takes the code as input for which you want to measure execution time. Let’s run a simple program.

|  |
| --- |
| import timeit print(timeit.timeit('2+3')) |

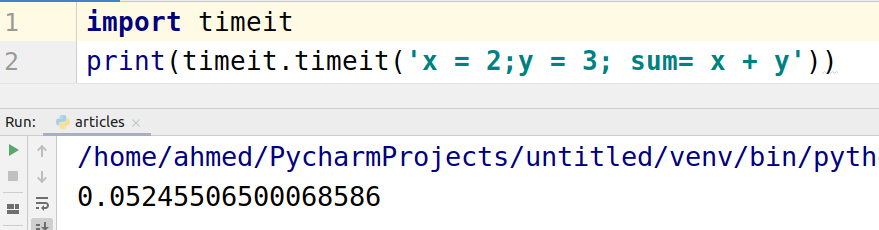
In the above code, a simple addition command is given as input. The time taken to perform simple addition is shown below.



## Multiple lines as input

In the below example, timeit function is taking multiple lines of code as input. While giving multiple lines as the input you just need to separate the statements by a semicolon.

|  |
| --- |
| import timeit print(timeit.timeit('x = 2;y = 3; sum= x + y')) |

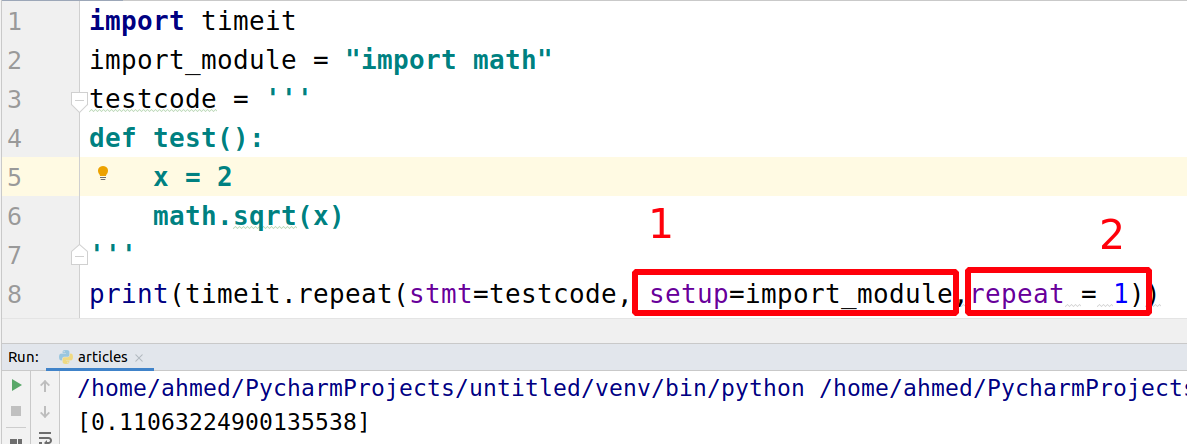


## Using triple quotes

Let’s take a look at the code in which we can pass a whole program or function as input to the timeit function.

|  |
| --- |
| import timeit import\_module = "import math" testcode = ''' def test():  x = 2  math.sqrt(x) ''' print(timeit.repeat(stmt=testcode, setup=import\_module,repeat = 1)) |

In the below screenshot 1 represents the second input of the timeit function. The “Setup” takes the necessary dependencies of the code that are required.



## Comparison of two codes

Now let’s take a look at how to compare two different sets of code. We are going to compare two functions, addition and multiplication.

|  |
| --- |
| import timeit   def addition(x, y):  result = x + y  return result   def multiplication(x, y):  result = x \* y  return result   def addition\_time():  # compute addition time  SETUP\_CODE = ''' from \_\_main\_\_ import addition '''  TEST\_CODE = ''' addition(2,3) '''   times = timeit.repeat(setup=SETUP\_CODE,  stmt=TEST\_CODE,  repeat=3,  number=10000)   # minimum time  print('addition time: {}'.format(min(times)))   # compute multiplication time def multiplication\_time():  SETUP\_CODE = ''' from \_\_main\_\_ import multiplication'''  TEST\_CODE = ''' multiplication(2,3)  '''   # timeit.repeat statement  times = timeit.repeat(setup=SETUP\_CODE,  stmt=TEST\_CODE,  repeat=3,  number=10000)   # minimum time  print('multiplication time: {}'.format(min(times)))   if \_\_name\_\_ == "\_\_main\_\_":  multiplication\_time()  addition\_time() |

The following is the output.

