How to Get time of a Python program's execution

In this article, we will learn to calculate the time taken by a program to execute in Python. We will use some built-in functions with some custom codes as well. Let's first have a quick look over how the program's execution affects the time in Python.

Programmers must have often suffered from **"Time Limit Exceeded"** error while building program scripts. In order to resolve this issue, we must optimize our programs to perform better. For that, we might need to know how much time the program is taking for its execution. Let us discuss different functions supported by Python to calculate the running time of a program in python**.**

The time of a Python program's execution measure could be inconsistent depending on the following factors:

1. The same program can be evaluated using different algorithms
2. Running time varies between algorithms
3. Running time varies between implementations
4. Running time varies between computers
5. Running time is not predictable based on small inputs

Calculate Execution Time using time() Function

We calculate the execution time of the program using `time.time()` function. It imports the `time` module which can be used to get the current time. The below example stores the starting time before the for loop executes, then it stores the ending time after the print line executes. The difference between the ending time and starting time will be the running time of the program. `time.time()` function is best used on ***nix.**

```
import time

#starting time
start = time.time()

for i in range(3):
    print("Hello")

# end time
end = time.time()

# total time taken
print("Execution time of the program is- ", end-start)
```

OUTPUT:

```
Hello
Hello
Hello
Execution time of the program is- 1.430511474609375e-05
```

Calculate execution time using timeit() function

We calculate the execution time of the program using `timeit()` function. It imports the `timeit` module. The result is the execution time in seconds. This assumes that your program takes at least a tenth of a second to run.

The below example creates a variable and wraps the entire code including imports inside triple quotes. The test code acts as a string. Now, we call the `time.timeit()` function. The `timeit()` function accepts the test code as an argument, executes it, and records the execution time. The value of the number argument is set to 100 cycles.


```
import timeit

test_code = """
a = range(100000)
```

```
b = []
for i in a:
    b.append(i+2)
"""

total_time = timeit.timeit(test_code, number=200)
print("Execution time of the program is-", total_time)
```

OUTPUT:

```
Execution time of the program is- 4.26646219700342
```

Calculate execution time using time.clock() Function

Another function of the time module to measure the time of a program's execution is `time.clock()` function. `time.clock()` measures CPU time on Unix systems, not wall time. This function is mainly used for benchmarking purposes or timing algorithms. `time.clock()` may return slightly better accuracy than `time.time()`. It returns the processor time, which allows us to calculate only the time used by this process. It is best used on Windows.

```
import time

t0= time.clock()
print("Hello")

t1 = time.clock() - t0

print("Time elapsed: ", t1 - t0) # CPU seconds elapsed (floating point)
```

OUTPUT:

```
Hello

Time elapsed: -0.02442
```

Note:

`time.clock()` is "Deprecated since version 3.3". The behavior of this function depends on the platform. Instead, we can use **perf_counter()** or **process_time()** depending on the requirements or have a well-defined behavior.

`time.perf_counter()` - It returns the value (in fractional seconds) of a performance counter, i.e. a clock with the highest available resolution to measure a short duration. It does include time elapsed during sleep and is system-wide.

`time.process_time()` - It returns the value (in fractional seconds) of the sum of the system and user CPU time of the current process. It does not include time elapsed during sleep. **For example,**

```
start = time.process_time()
... do something
elapsed = (time.process_time() - start)
```

Calculate execution time using datetime.now() Function

We calculate the elapsed time using `datetime.datetime.now()` from the `datetime` module available in Python. It does not make the script a multi-line string like in `timeit()`. This solution is slower than the `timeit()` since calculating the difference in time is included in the execution time. The output is represented as days, hours, minutes, etc

The below example saves the current time before any execution in a variable. Then call `datetime.datetime.now()` after the program execution to find the difference between the end and start time of execution.


```
import datetime

start = datetime.datetime.now()

list1 = [4, 2, 3, 1, 5]
list1.sort()

end = datetime.datetime.now()
print(end-start)
```

OUTPUT:

```
0:00:00.000007
```

Calculate execution time using %%time

We use `%%time` command to calculate the time elapsed by the program. This command is basically for the users who are working on **Jupyter Notebook**. This will only capture the wall time of a particular cell.

```
%%time
[ x**2 for x in range(10000)]
```

```
Wall time: 19 ms
Out[11]: [2,
 3,
 4,
 5,
 6,
 7,
 8,
 9,
 10,
 11,
 12,
 13,
 14,
 15,
 16,
 17,
```

Why is timeit() the best way to measure the execution time of Python code?

1. You can also use `time.clock()` on Windows and `time.time()` on Mac or Linux. However, `timeit()` will automatically use either `time.clock()` or `time.time()` in the background depending on the operating system.

2. `timeit()` disables the garbage collector which could otherwise skew the results.

3. `timeit()` repeats the test many times to minimize the influence of other tasks running on your operating system.

Conclusion

In this article, we learned to calculate the time of execution of any program by using functions such as `time()`, `clock()`, `timeit()`, `%%time` etc. We also discussed the optimization of the python script. We learned about various functions and their uniqueness.

← Get Year from Date

Convert Seconds to Minutes →

Game Development

Data Structure & Alog.

Operating System

Computer Network

Computer Architecture

More...