Learning Quiz 29: Run Time Analysis

Due Nov 27 at 1pm **Points** 5 **Questions** 5 **Available** until Dec 4 at 11:59pm **Time Limit** None
Allowed Attempts Unlimited

Instructions

Prior to completing this quiz, be sure to read:

- Section 10.3: Run Time Analysis (p. 347-353)

Please also go over Practice Problem 10.8 in the textbook (solution at the end of the chapter) before attempting this quiz.

This quiz was created for learning purposes. You may attempt this quiz as many times as you would like. The highest score *prior to the deadline* will count towards the final course grade. No late submissions will be accepted.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	less than 1 minute	5 out of 5
LATEST	Attempt 2	less than 1 minute	5 out of 5
	Attempt 1	8 minutes	3 out of 5

Score for this attempt: **5** out of 5
Submitted Nov 24 at 10:22am
This attempt took less than 1 minute.

Question 1

1 / 1 pts

As a programmer, a major question that you will eventually encounter is, "Is my program efficient?" Efficiency usually refers to the program's run time. A smaller run time means that the program completes in a shorter amount of time. In Learning Quiz 28, we compare recursive functions to loops and find that they are able to achieve the same results. In this quiz, we will work towards comparing the run time of recursion versus loops.

Which of the following functions is recursive?

```
def count1(n):
    for i in range(1,n+1):
        print(i)

def count2(n):
    if n>2:
        countrecur(n-1)
    print(n)
```

○ count1()

Correct!

[✓ count2()]

Question 2

1 / 1 pts

In the code below, we will need to import a module to help us keep track of the run time of our program. Consider the code below. Fill in the blank with one word for the module that we will need to import.

```
import ___________

def count1(n):
    for i in range(1,n+1):
        print(i)

def count2(n):
    if n>2:
        count2(n-1)
    print(n)

start1 = time.time()
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
stop1 = time.time()

start2 = time.time()
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
stop2 = time.time()
```

Correct!

time

Correct Answers

time

Question 3

1 / 1 pts

Given the code from Question 2, how would you calculate the run time of running count1(990) ten times?

start2 - start1

Correct!

[✓ stop1 - start1]

start1 - stop1

stop2 - stop1

Question 4

1 / 1 pts

The run time of a program is affected by a lot of factors, including our computer specs and any programs we might have running in the background. Therefore, one computer might compute all functions many times faster than another computer, regardless of whether the algorithm is efficient or not. Ideally, we would like to compare the run time of programs with everything else held equal (same computer, no programs running in the background, computer isn't overheating, etc.). Unfortunately, this is almost never possible. We will better be able to compare the efficiency of our programs by running the algorithm multiple times and comparing the overall result.

After importing the time module and creating the count1() and count2() functions from above, run the following lines of code a few times on your computer at least 5 times:

```
start1 = time.time()
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
count1(990)
stop1 = time.time()

start2 = time.time()
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
count2(990)
stop2 = time.time()

print(stop1 - start1)
print(stop2 - start2)
```

You will notice that count1() is generally faster than count2(), though this may vary slightly depending on your computer.

Correct!

[✓ I have completed this exercise]

Question 5

1 / 1 pts

In the questions above, we limited our count1() and count2() functions to n = 990. Thus, our functions only printed out values up to 990. If we try to test the run time count2(1000), we encounter a RecursionError because Python, by default, will not allow more than 997 recursive calls in depth. To change that, add the following lines to your code:

```
import sys
sys.setrecursionlimit(1500)    ## or whatever your desired stacking value is
```

Note, this is dangerous because you may encounter a memory issue.

True/False: count1(5000), as written in Question 1, will also encounter a RecursionError

True

Correct! [✓ False]

Quiz Score: **5** out of 5