Learning Quiz 28: Recursion

Due Nov 13 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:

- Sections 10.1-10.2: Introduction to Recursion, Examples of Recursion (p. 329-347)

Please also go over Practice Problems 10.1-10.7 in the textbook (solutions 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
LATEST	Attempt 1	2,942 minutes	3 out of 5

Score for this attempt: **3** out of 5
Submitted Nov 21 at 2:24pm
This attempt took 2,942 minutes.

Question 1

1 / 1 pts

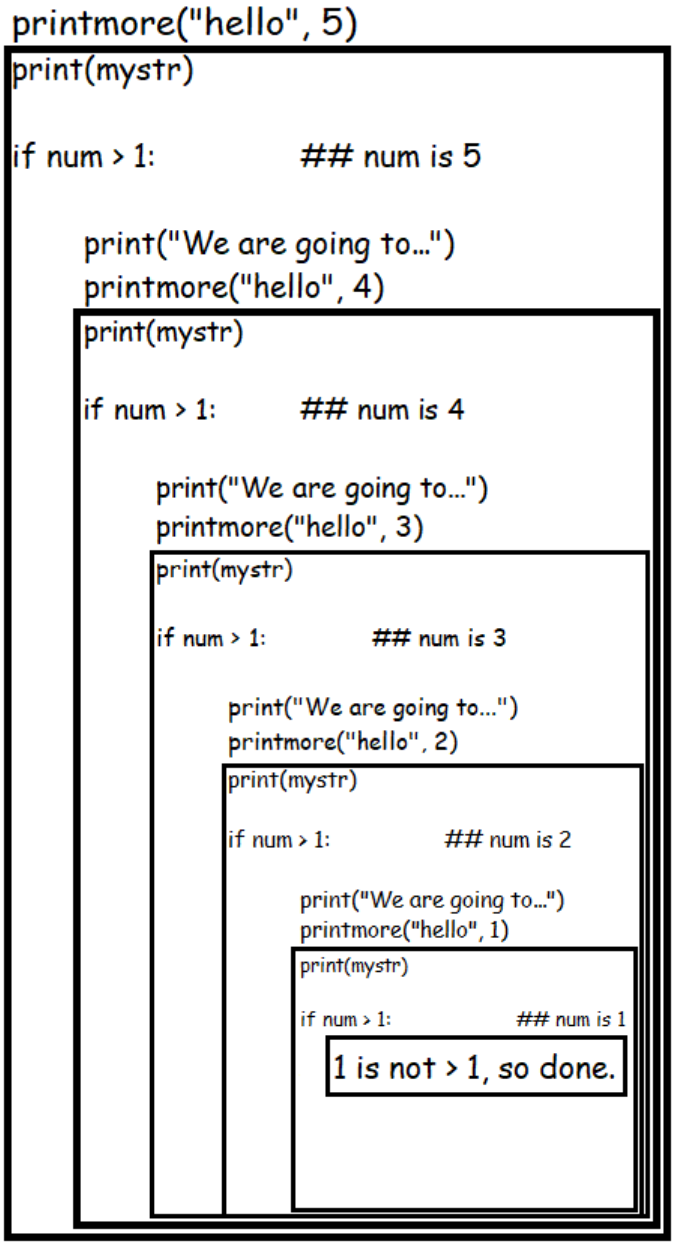
A recursive function is a function that calls itself. In the following example, we look at a function that prints a text x number of times:

```
def printmore(mystr, num):
    ''' prints mystr num number of times '''
    print(mystr)
    if num > 1:
        print("We are going to print", num - 1, "more times.")
        printmore(mystr, num-1)

## test out the function...
printmore("hello",5)
```

Notice that we defined a printmore() function AND the function calls on a printmore() function.

The logic of the printmore() example above can be visualized as:



Suppose we would like to print the following output using a recursive function:

```
## Desired output:
5 4 3 2 1 0
4 3 2 1 0
3 2 1 0
2 1 0
1 0
0

## Line of code to run:
tozero(5)
```

Fill in the blanks in order to get the desired outcome:

```
def tozero(n):
    for i in range(n,-1,-1):
        print(i, end = " ")
    print()
```

if n > 0 :

tozero (n-1)

Answer 1:

Correct! 0

Correct Answer -1

Answer 2:

Correct! tozero

Question 2

1 / 1 pts

Notice that in our functions in Question 1, the recursion eventually stopped because the if statement condition was not satisfied. It is possible that our recursion function may logically never end if a stopping condition is never encountered.

Consider the following modified printmore() function with no stopping condition:

```
def printmore(mystr, num):
    ' prints mystr num number of times '
    print(mystr)
    print("We are going to print", num - 1, "more times.")
    printmore(mystr, num-1)

## test out the function...
printmore("hello",5)
```

You'll notice that (unlike other programming languages), Python will eventually raise an exception: RecursionError.

As a programmer using recursion, it's important to think about your **base case**--what are you working down or up towards, and what happens when you reach the end?

In the following example, we are working towards the number 0. When we reach 0, our base case, we'll print a celebratory statement.

```
def celebrate(n, statement):
    'counts down to 0 by 1 and prints statement at end'
    if n <= 0:       ## base case
        print(statement)
    else:           ## recursive step, n > 0
        print(n)
        celebrate(n-1, statement)

celebrate(10, "Happy New Year!")
```

You can also make the else portion your base case. Compare the following code to the one right before this:

```
def celebrate(n, statement):
    'counts down to 0 by 1 and prints statement at end'
    if n > 0: ## recursive step
        print(n)
        celebrate(n-1, statement)
    else: ## base case
        print(statement)
```

```
celebrate(10, "Happy New Year!")
```

You can even omit the else, as long as there's an end to the recursion:

```
def celebrate(n, statement):
    'counts down to 0 by 1 and prints statement at end'
    if n > 0: ## recursive step
        print(n)
        celebrate(n-1, statement)

celebrate(10, "Happy New Year!")
```

Which of the following will give us a RecursionError?

```
def count100(n):
    if n > 100:
        print("Over 100")
    else:
        print(n)
        count100(n+10)

count100(15)
```

```
def count100(n):
    if n < 100:
        print("Over 100")
    else:
        print(n)
        count100(n-10)

count100(15)
```

Correct!

```
def count100(n):
    if n > 100:
        print("Over 100")
    else:
        print(n)
        count100(n-10)

count100(15)
```

```
def count100(n):
    if n < 100:
        print(n)
        count100(n + 10)
    else:
        print("Over 100")

count100(15)
```

Question 3

0 / 1 pts

The base case is the stopping condition (no more recursions called) of a recursive function. Consider the following code:

```
def myfun(n):
    if n > 0:
        print(n)
        myfun(n-1)
```

What is the base case of the function?

You Answered

- (●) n > 0

  For all values n > 0, myfun(n-1) will run. myfun(n-1) is the recursion.

- ( ) n < 0
- ( ) This function does not have a base case

Correct Answer

- ( ) n <= 0
- ( ) n >= 0

Question 4

0 / 1 pts

You may have noticed by now that functions in Questions 1-3 can also easily be handled by using loops. In fact, it's more efficient to handle tasks by loops. However, it is sometimes easier to read and work with recursion code. Consider the following code that prints out the Fibonacci sequence 1 1 2 3 5 8 13 21 34 55 89 ...:

```
nterm = int(input("How many terms? "))


print("Compare the loop code and output: ")
term0 = 0
term1 = 1
if nterm <=0:
    print(term0)
else:
    print(term0, term1, end=" ")
    for i in range(2,nterm):
        next = term0 + term1
        print(next, end = " ")
        term0 = term1
        term1 = next



print("\nto the recursion code and output:")
def fib(n):
    if n <= 1:
        return(n)
    else:
        return(fib(n-1) + fib(n-2))

for i in range(nterm):
    print(fib(i), end = " ")
```

An interesting recursion problem is the "Tower of Hanoi" which is worth looking into. However, we will not discuss it here.

Consider the following recursive function:

```
def countdown(n):
    'counts down to 0 by 5'
    if n <= 0:       ## base case
        print("Reached 0")
    else:           ## recursive step, n>0
```

```
print(n)
countdown(n-5)
```

Which of the following non-recursive functions achieves the same task?

```
def countdown_norecur(n):
    for i in range(n):
        print(i)
    print("Reached 0")
```

You Answered

```
def countdown_norecur(n):
    for i in range(n,1,-5):
        print(i)
    print("Reached 0")
```

```
def countdown_norecur(n):
    for i in range(n,0,-5):
        print(i-5)
    print("Reached 0")
```

Correct Answer

```
def countdown_norecur(n):
    for i in range(n,0,-5):
        print(i)
    print("Reached 0")
```

```
def countdown_norecur(n):
    for i in range(n,1,-5):
        print(i-5)
    print("Reached 0")
```

```
def countdown_norecur(n):
    for i in range(n):
        print(i+1)
    print("Reached 0")
```

Question 5

1 / 1 pts

Consider the follow non-recursive function:

```
def fun1(n):
    for i in range(n):
        print(i * "*")
```

Which of the following recursive functions will achieve the same task for values n >= 1? (Hint, print each one separately and consider blank lines)

```
def fun2(n):
    if n > 1:
        fun2(n)
    print((n-1) * "*")
```

Correct!

```
def fun2(n):
    if n > 1:
        fun2(n-1)
    print((n-1) * "*")
```

```
def fun2(n):
    if n > 0:
        fun2(n-1)
    print((n-1) * "*")
```

```
def fun2(n):
    if n > 0:
        fun2(n)
    print((n-1) * "*")
```

Quiz Score: **3** out of 5