Learning Quiz 22: Encapsulation in Functions

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:

- Section 7.1: Encapsulation in Functions (p. 204-210)

Please also go over Practice Problem 7.1 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
LATEST	Attempt 1	11 minutes	4.25 out of 5

Score for this attempt: **4.25** out of 5
Submitted Nov 9 at 6:18pm
This attempt took 11 minutes.

Question 1

0.25 / 1 pts

Consider the following function:

```
def squareSum(x,y):
    sqsum = x**2 + y**2
    print("If x =", x, "and y =", y, "then the squared sum is", sqsum)
    return sqsum

squareSum(5,6)
```

Notice that although we didn't explicitly create a x or y variable, Python still successfully runs the function called squareSum().

However, if we try to access the x or y value outside of the function, we get an error saying that x and y have not been created.

```
def squareSum(x,y):
    sqsum = x**2 + y**2
    print("If x =", x, "and y =", y, "then the squared sum is", sqsum)
    return sqsum

squareSum(5,6)
print(x)
```

This is because the x and y variables exist only within the function. The are called **local** variables.

Consider the following program:

```
x = 7
a = 3
b = x*a
def avg5(a,b,c,d):
    y = 5
    return (y + a + b + c + d)/4
d = avg5(1,2,3,4)
```

Which of the following variables exist outside of the function after running the lines above?

Correct Answer ☐ a

Correct Answer ☐ b

Correct Answer ☐ d

☐ c

☐ y

Correct! ☑ x

Question 2

1 / 1 pts

Variables can be defined both inside and outside of functions. Variables defined outside of the function (in the interpreter shell) are called **global** variables. Global variables can still be accessed inside the function:

```
x = 3
def outx():
    print("Variable x is",x)
outx()
```

This is true regardless of whether the variable is created before or after the function definition:

```
def outx():
    print(x)
x = 3
outx()
```

However, the variable needs to be created before calling the function. The following will produce an error because x is defined after calling the function:

```
def outx():
    print(x)
outx()
x = 3
```

Consider the following program:

```
x = 7
##### Space 1 #####
def outx():
    print(x)
##### Space 2 #####
x = 5
##### Space 3 #####
x = 3
##### Space 4 #####
```

Where should we call on our function outx() in order to print the value 5?

- Space 2
- Space 1
- Space 4
- [✓ Space 3]

Correct!

Question 3

1 / 1 pts

If we define a variable inside a function, that definition will only last within the function. In the below example notice that x equals 3 outside of the function:

```
x = 3
def outx():
    x = 10
    print("Print 1: Variable x is", x)
outx()
print("Print 2: Variable x outside of the function is", x)
```

Notice that x refers to the value 10 only within the function. In this case, we have two different variable x's-- one in the interpreter shell and one in the function.

You can also create local (function-copy) variables through arguments. In the following code, although we set y = 7, the argument 8 in outxy(8) will make function-copy y equal to 8.

```
x = 3
y = 7
def outxy(y):
    x = 10
    print("Print 1: Variable x is", x)
    print("Print 1: Variable y is", y)
outxy(8)
print("Print 2: Variable x outside of the function is", x)
print("Print 2: Variable y outside of the function is", y)
```

Consider the following program:

```
x = 10
y = 20
z = 30
def calcVar(x):
    y = 100
    return x * y * z
newVar = calcVar(4)
```

What is stored in variable newVar?

30000

None

Correct!

[✓ 12000]

6000

2400

Question 4

1 / 1 pts

In the first code example from Question 2, we were able to run the following program:

```
x = 3
def outx():
    print("Variable x is",x)
outx()
```

In the first code example from Question 3, we were able to run the following program:

```
x = 3
def outx():
    x = 10
    print("Print 1: Variable x is", x)
outx()
print("Print 2: Variable x outside of the function is", x)
```

However, if we define a separate local copy of x within the function (as in question 3), we cannot access the outer value x even if it's before we create the local x.

```
x = 3
def outx():
    print("Line 1: Variable x is", x)        #This will give us an error
    x = 10
    print("Line 2: Variable x is now", x)
outx()
print("Line 3: Variable x outside of the function is", x)
```

Consider the following program:

```
##### Space 1 #####
x = "orange"
def outx():
    ##### Space 2 #####
    print(x)
    ##### Space 3 #####
##### Space 4 #####
outx()
##### Space 5 #####
```

Considering each space separately, what will print if the code x = "apple" is inserted in each indicated space.

Correct!

Space 1 orange

Correct!	**Space 2**	apple
Correct!	**Space 3**	ERROR
Correct!	**Space 4**	apple
Correct!	**Space 5**	orange

Question 5

1 / 1 pts

Each function creates a new "namespace". As a result, it is possible to have multiple "x" variables across different functions. The following program will run as usual with functions thisFun() referring to the interpreter shell "x" and thatFun() and otherFun() referring to the function-defined "x".

```
x = 10
def thisFun():
    print(x)

def thatFun():
    x = 15
    print(x)

def otherFun(x):
    print(x)
```

You can also have functions call on each other. The same namespace rules apply where "x" may refer to a different value across functions.

```
def h(x):
    print('In h, x is', x)
    return x

def g(x):
    print('Start g')
    h(x-1)
    print('In g, x is', x)
    return x

def f(x):
    print('Start f')
    print('In f, x is', x)
    print(g(x-1) * h(x+4))

f(3)
```

Consider the following program:

```
def h(x):
    print("Output 1")

def g(x):
    h(x-5)
    print("Output 2")

def f(x):
    g(x*2)
    print("Output 3")
    return x%3
```

```
x = f(7)
print("Output 4")
```

What is the value of x at that each of the following print-output numbers?

Correct!	**Output 1**	9
Correct!	**Output 2**	14
Correct!	**Output 3**	7
Correct!	**Output 4**	1

Other Incorrect Match Options:
- 3
- 4
- 2

Quiz Score: **4.25** out of 5