CS215 Chapter4B Lab – None & Functions w/ Default Params   Name:____________________

1 ALL FUNCTIONS ARE DEFINED AT THE START OF A PROGRAM

1. Convention: All function definitions go at the top of your file, one after the other. In Lab 4A, you probably did not do this. Go back to Lab4A and move both function defintions below so that they are the start of your file.

   - `findLetterGrade(...)`
   - `calculateStadiumSales(...)`

2. Rerun. For some of you, your program will not compile. Why not? Call me over to discuss. If it compiles, move on.

3. I understand I will lose 1 point on every assignment in which all functions of my design are not at the top of my program file.

   Your signature: ____

2 CALLING A METHOD AND UNDERSTANDING NONE

4. Consider the method definition at right.

   For what values of x does this function *explicitly* return a value?

   ____

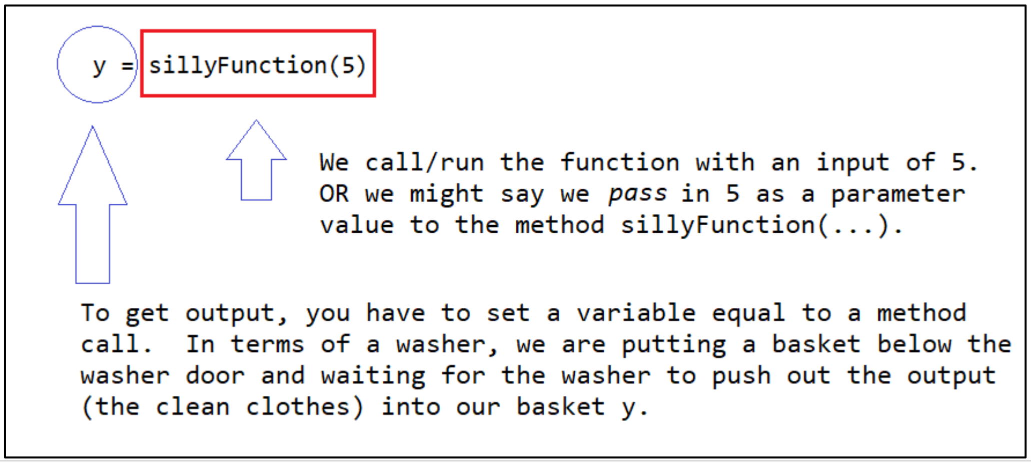
```
def sillyFunction(x):
    '''''This function is NONEsense. :)'''
    if 1<x<9:
        return x**2
    elif x >= 9:
        return x**0.5
```

5. The code above is the function's definition. It does not DO anything until used. As an analogy, consider a washer. It just sits there, doing nothing until it is time to do laundry. At that point, we turn it on via a special knob/button, give it input (dirty wash), and when it's done, we get its output (clean wash).

   A function definition is like an unused washer. It gives you the power to clean clothes, but you yourself have to do a few special things to actually "run the washer".

   In CS lingo, "running a function" is the same thing as "using the function." Also using CS lingo, we "run" or "use" a function by "CALLING THE FUNCTION," that is, by passing in an actual output and getting out actual output from it.

   The red below shows how we CALL a function (turn on the washer) and the blue below shows how we get its output.



```
y = sillyFunction(5)
```

We call/run the function with an input of 5. OR we might say we *pass* in 5 as a parameter value to the method sillyFunction(...).

To get output, you have to set a variable equal to a method call. In terms of a washer, we are putting a basket below the washer door and waiting for the washer to push out the output (the clean clothes) into our basket y.

6. In the example, above, we called sillyFunction with a parameter value of 5. This runs our function with a parameter value of x = 5. The output of sillyFunction will therefore be 25. Why? (Call me over if you don't know.) What are the values of y after the function calls below?

```
y = sillyFunction(3)        ______________________________
y = sillyFunction(9)        ______________________________
y = sillyFunction(16)       ______________________________
y = sillyFunction(-1)       ______________________________
```

7. Check your answers above by running the file LabCh4B.py, replacing the ???'s with the numbers above.

8. Why did you see "None" in the last line of the previous example? Well, Python includes the value "None" which represents the absence of a value. Every Python function returns *something*. Even if you do not include return statement.

9. Add this code at the bottom of LabCh4.py:

   - Ask the user to enter a number.
   - Call sillyFunction with the user's number and save the returned output into the variable y.
   - Analyze y:
     - If y is set to None, print "Silly Function returned None."
     - Otherwise, print "Silly Function returned Something meaningful."

10. Notice we named sillyFunction(…) according to camelCase (first word lowercase, each additional word starts with an uppercase). For now, just like with variable names, you have two options for how to name methods in this course.

   - camelCase: sillyFunction(…) [Use this convention if you name your variables in camelCase.]
   - lowercase with underscores: silly_function(…) [Use this convention if you name your variables in the same convention.]

   Sign the line below.

   I understand I will lose 1 point on every assignment in which any function of my design is not named according to 1 of the conventions above, and that my chosen variable naming conventions should match my chosen function naming convention.

   Your signature: ____

3 DEFAULT PARAMETERS

1. Put these lines of code at the bottom of sillyFunction.py.

```
y = sillyFunction()
print (f"output when no parameter is passed in: {y}")
```

   What happens? ____

2. Without providing a value for the parameter x, sillyFunction cannot run. If you want the user to be able to run the function without an input for x, you just provide an input for x in the method definition. The highlighted code below shows how to provide a default value of -1 to x, so if the user does not provide a value, the method will be run with x set to -1./

```
def sillyFunction(x = -1):
    '''This function is nonesense.'''
    if 1<x<9:
        return x**2
    elif x >= 9:
        return x**0.5
```

3. Rerun the program after adding the highlighted code in the previous example.

```
y = sillyFunction()
print (f"output when no parameter is passed in: {y}")
```

What do the lines above do now? ____

4. Define a new method called rectangleArea (or rectangle_area) AT THE TOP OF YOUR FILE that takes in the 2 obvious parameters below and returns the area. For demonstration purposes, please list length before width in your method signature.

    - a length (with a default value of 5) and a width (with a default value of 10)

5. What is the output of the method calls below? Take a guess and then run your code to determine the answer. Call me over if the answer does not make sense to you.

`y = rectangleArea(2, 3)` ______________________________

`y = rectangleArea(3, 2)` ______________________________

`y = rectangleArea(2)` ______________________________

`y = rectangleArea(width = 2)` ______________________________

`y = rectangleArea()` ______________________________

6. Now, remove the default value for width in the method definition. Rerun the line below.

```
y = rectangleArea(2)
```

What happens? ____

7. The point of the above example: Some arguments in a function may have defaults, and some may not. Any parameters that have a default value must appear to the RIGHT of those that do not have defaults.

Why is this sensible? Because these trailing parameters at the right are pretty much optional, but the initial leftmost non-default parameters must be provided by the user.

8. Update your method signature so that there is a length (NO DEFAULT value) followed by a width (with a default value of 10) . What is the output below?

```
y = rectangleArea(2, 3)          ____________________

y = rectangleArea(2)             ____________________

y = rectangleArea()              ____________________
```

9. Some of you have discovered that 2**5 can be computed via the function call pow(2,5). Both will compute 2^5 = 32. Type the following into the Console area of Spyder. Doing so shows you all the documentation about the pow function. This is how you can get information about functions that you did not write.

```
pow?
```

10. How many parameters can pow take? ____________________

    How many parameters with default values are provided? ____________________

11. After reading the documentation, guess at the output below. Then check your work by coding these values into your file.

```
y = pow (2, 3)          ____________________

y = pow (2)             ____________________

y = pow (2, 3, 5)       ____________________

y = pow (4, 2, 7)       ____________________
```

12. Check out the documentation about the print function by typing the following into the console.

```
print?
```

    How many parameters can print take? ____________________

    How many parameters with default values are provided? ____________________

13. What is printed out below? After this lab, "sep" and "end" should not seem magical anymore.

```
print("a", "b", "c", sep='-', end="***")
print("d", "e", sep='...', end="!!")
```

```
Answer:
```

____

____

SUBMISSION INFO

TO GET CREDIT FOR THIS LAB, UPLOAD THESE 3 DOCUMENTS TO THE SUBMISSION AREA.

- LearningFunctions.py from LabCh4A (with function defs now at top)
- LabCh4B.pdf (with all blanks filled in)
- FunctionsWithDefaultAndNone.py