Lab 06 – Functions

Value: *10 points*

This week's lab will put into practice the concepts you learned about functions: creating functions, passing parameters, and scope.

(Having concepts explained in a new and different way can often lead to a better understanding, so make sure to pay attention as your TA explains.)

Part 1A: Review – Scope

Everything in python3 has a ***scope*** - the places in the program in which it is accessible. For example, you can create a constant outside of functions or the main program.

```
MAX_VAL = 8

def other_fxn ():

# code in other_fxn () has access to MAX_VAL

     print ( "The maximum allowed value is" , MAX_VAL)

if __name__ == "__main__":

# code in main block has access to MAX_VAL

     guess = int ( input ( "Guess the max value: " ))

     while guess != MAX_VAL:

          print ( "Whoops, you guessed wrong." )

          guess = int ( input ( "Guess again: " ))
```

That constant is now a ***global*** constant, which means it can be accessed by any line of code in the file. So the main program can access it, as well as any other functions that you might write. Remember, for this course you are only allowed to have constants be global - regular variables (that aren't constants) should only be declared inside functions.

Local variables are only accessible to code within their same scope. If a variable is declared in the main program, another function called `print_info()` will not be able to access it. In the same way, a variable in `print_info()` will not be accessible to the code in the main block

```
def print_info ():

# this variable can't be accessed by main()

     var_for_print_info = 5

if __name__ == "__main__":

# this variable can't be accessed by print_info

     var_for_main = 17
```

Part 1B: Review – Functions

A function is a way of compartmentalizing our code: a well-written function does *one* thing, and does it very well. A function allows us to write a piece of code once, and to then use, or "call," the function whenever we want to use that code.

A function has a few key parts:

1. Function name
   - This is how we call the function. It tells python3 that we want it to use that function and execute its…
2. Function body
   - This is the code that makes up the function. This is what the function does when called.
3. Parameters (optional)
   - A function uses parameters to take in information from the code that called it. This is one of the ways that data is passed from one piece of code to another. A function can have no parameters, one parameter, or it could have a hundred!

Let's take a look at some example functions and how they work:

```
def print_name (name):
    print ( "Hello, my name is" , name)
```

This function is called **print_name()** , and it takes in one formal parameter (**name**). In order to use the code in this function, we must call the function and pass it an ***argument*** . The argument could be a variable, or it could be a literal string (one with quotation marks around it).

Here's the **print_name()** function again, but this time we also have a main program that calls the function multiple times.

```
def print_name (name):
        print ( "Hello, my name is" , name)
if __name__ == "__main__":
        user = input( "What's your name? " )
        prez = "Hrabowski"
        print_name(user)
        print_name(prez)
        print_name( "John Jacob Jingleheimer Schmidt" )
```

Note that we have called the function multiple ways: both with variables and with a string literal. Note also that the variable names we passed as actual parameters (**prez** and **user**) do not need to match the name of the formal parameter.

Here is the output for the code above:

Hello, my name is YOUR_NAME_HERE
Hello, my name is Hrabowski

Hello, my name is John Jacob Jingleheimer Schmidt

Part 2: Exercise

In this lab, you'll be downloading a file and completing it by writing three function definitions, and then writing three function calls in the main program.

Tasks

Starting:

□ Copy the **lab6.py** file from the location specified by your TA

□ It should be renamed to be **names.py**

□ Complete the file header comment at the top

Functions:

□ Write the code for **sum_a_list()**

□ Write the code for **get_string_lengths()**

□ Write the code for **get_names()**

□ Call the function **sum_a_list()**

□ Call the function **get_string_lengths()**

□ Call the function **get_names()**

General:

□ Run and test your code as needed

□ Show your work to your TA

Part 3A: Downloading the File

First, create the lab06 folder using the mkdir command -- the folder needs to be inside your Labs folder as well.

Next, copy the lab6.py file into your lab06 folder. The file is available on the class github page. You can download it from github and then scp (secure copy) it into your GL account.

The first thing you should do in your file is complete the file header comment, filling in your name, section number, email, and the date.

Part 3B: Creating Functions

At this point, if you try to run the file, you will get an error. That is because the file is only partially completed for you.

You will need to update the file to complete the three function definitions and the main program. Read the function header comments to see the details about the three functions.

You will also need to write calls to each of these functions. The places where these calls need to happen in the main program are indicated for you. You shouldn't need to write any other code.

Part 4: Completing Your Lab

You must use the **submit** command to complete your lab. You will have until midnight on Thursday, March 26, to complete and submit your work. Submit it as lab6.

Tasks

Starting:

□ Copy the **lab6.py** file
□ It should be renamed to be **names.py**
□ Complete the file header comment at the top

Functions:

□ Write the code for **sum_a_list()**
□ Write the code for **get_string_lengths()**
□ Call the function **sum_a_list()**
□ Call the function **get_string_lengths()**

General:

□ Run and test your code as needed
□ Submit your work to your TA