Lab: Functions

This lab accompanies Chapter 6 of *Starting Out with Programming Logic & Design*.

Lab 7.1 – Functions and Pseudocode

Critical Review

You have been coding with modules in pseudocode and functions when using Python.

You modules in pseudocode can be made into functions by returning a value.

A function is a special type of module that returns a value back to the part of the program that called it.

Most programming languages provide a library of prewritten functions that perform commonly needed tasks.

Library functions are built into the programming language and you can call them as needed. They are commonly performed tasks.

Writing Your Own Function that Returns an Integer

Step 1: A function contains three parts: a header, a body, and a return statement. The first is a function header which specifies the data type of the value that is to be returned, the name of the function, and any parameter variables used by the function to accept arguments. The body is comprised of one or more statements that are executed when the function is called. In the following space, complete the following: (Reference: Writing Your Own Functions).

- Write a function with the header named `addTen`.
- The function will accept an `Integer` variable named `number`.
- The function body will ask the user to enter a number and the add 10 to the number. The answer will be stored in the `variable` number.
- The `return` statement will return the value of `number`.

```
Function a._____  b._____ (c._____)
      Display "Enter a number:"
      Input d._____
      Set e._____ = number + 10
Return f._____
```

Step 2: In the following space, write a function call to your function from Step 1.

```
Set number = _____(_____)
```

Writing Your Own Function that Returns a Boolean Value

Step 1: A Boolean function will either return a true or a false value. You can use these functions to test a condition. They are useful for simplifying complex conditions that are tested in decision and repetition structures. In the following space, complete the following: (Reference: Returning Boolean Values).

- Write a function named `gender`. The function accepts no arguments.
- The function will declare a local Boolean variable named `answer`, and a local `String` variable named `type`.
- The function body will ask the user to enter their gender, and store the user's input in the `type` variable.
- The function will determine whether the user entered "male" or "female" with an `if` statement.
- The `return` statement will return the value of `answer`.

```
Function a._____  b._____ ()
      Declare Boolean answer
      Declare String type
      Display "Enter your gender (male or female):"
      Input c._____
      If (d._____ == "male") then
            answer = False
      Else
            answer = True
      End If

Return e._____
```

Step 2: In the following space, write a function call to your function from Step 1.

```
Set answer = _____ ()
```

Using Mathematical Library Function: `sqrt`

Step 1: The `sqrt` function accepts an argument and returns the square root of the argument. In the following space, complete the following: (Reference: The sqrt Function).

a. Declare a variable named `myNumber` and a variable named `squareRoot` of the data type `Real`.
b. Ask the user to enter a number of which they want to find the square root. Store the input in `myNumber`.
c. Call the `sqrt` function to determine the square root of `myNumber`.
d. Display the square root to the screen.

```
Declare Integer a._____
Declare Real b._____
Display "Enter a number:"
Input c._____
```

```
Set d._____ = _____
Display "The square root is ", e._____
```

Using Formatting Functions

Step 1: Most languages provide one or more functions that format numbers in some way. A common use of formatting functions is to format numbers as currency amounts. While a specific programming language will have its own name for formatting currency, use the function `currencyFormat` for pseudocode. In the following space, complete the following: (Reference: Formatting Functions).

- Declare a variable named `subtotal`, a constant variable named `tax` set to the rate of .06, and a variable named `total`.
- Ask the user to enter the subtotal. Store the input in `subtotal`.
- Calculate the total as `subtotal + subtotal * tax`.
- Make a call to the `currencyFormat` function and pass it `total`. Display the value that is returned from the function on the screen.

```
Declare Real a._____
Declare Constant Real b._____
Declare Real c._____
Display "Enter the subtotal:"
Input d._____
Set e._____ = _____
Display "The total is $", f._____(_____)
```

Lab 7.2 – Python Code and Random

Critical Review

A value-returning function is a function that returns a value back to the part of the program that called it. In Python, you have been using value-returning functions and those that do not.

Recall the function calls from Lab 6-4. The first call returns number back to the `number` variable. The second call just displays a value and there is no need to return a value.

```
number = getNumber(number)  # value returning function
printAverage(averageScores) # function returns no value
```

Standard Library Functions

Python comes with a *standard library* of functions that have already been written for you. These functions, known as *library functions,* make a programmer's job easier because they perform many of the tasks that programmers commonly need to perform. In fact, you have already used several of Python's library functions. Some of the functions that you have used are `input`, `input`, and `range`. Python has many other library functions.

The `random` Function

In order to use the `random` function in Python, you must import the `random` library. This loads the library into memory so that you can use the functions that exist within it. To do this, simply add the following line to the top of your code:

```
import random
```

One of the functions in the `random` library is the `random.random.int( )` module. This function accepts two arguments with the first being the starting number and the second being the ending number. The following is how you would get a random number between 1 and 6.

```
p1number = random.randint(1, 6)
```

Writing your own Value-Returning Functions

We have already written our own value returning functions that return one variable to the place where the function was called.

However, you can also return more than one value in Python. The function call might look as follows:

```
playerOne, playerTwo = inputNames(playerOne, playerTwo)
```

The `return` statement looks as follows:

The goal of this lab is to convert the Dice Game to Python code.

Step 1: Start Visual Studio Code.. Prior to entering code, save your file by clicking on File and then Save. Select your location and save this file as *Lab7.py*. Be sure to include the .py extension.

Step 2: Document the first few lines of your program to include your name, the date, and a brief description of what the program does.

Step 3: Start your program with the following code for main:

```
# Lab 7-3 The Dice Game
# add libraries needed

# the main function
def main():
    print()

    # initialize variables


    # call to inputNames


    # while loop to run program again
    while endProgram == 'no':

        # populate variables


        # call to rollDice


        # call to displayInfo


        endProgram = input('Do you want to end program? (yes/no): ')


#this function gets the players names


#this function will get the random values


#this function displays the winner


# calls main
main()
```

Step 4: Under the comment for adding libraries, add the following statement:

```
import random
```

Step 5: Under the comment for initialize variables, set `endProgram` to 'no' and `playerOne` and `playerTwo` to 'NO NAME'.

Step 6: Under the comment for making a call to inputNames, set the function call to both playerOne and playerTwo and pass both variables to the function as arguments. This must be done because both values need to be returned from the function. This is done as follows:

```
playerOne, playerTwo = inputNames(playerOne, playerTwo)
```

Step 7: Inside your `while` loop, set `winnersName` to 'NO NAME' and `p1number` and `p2number` to 0.

Step 8: Make a call to `rollDice` and pass the necessary variables needed in this function. This function should be set to the `winnerName` as that variable will be returned from the function. This is done as follows:

```
winnerName = rollDice(p1number, p2number, playerOne, playerTwo,
                      winnerName)
```

Step 9: Make a call to `displayInfo` and pass it `winnerName`.

Step 10: The next step is to write the function that will allow both players to enter their names. Write a function heading that matches your function call in Step 6, making sure to accept two arguments. The body of this function will use the `input` function to take in both players names, and one `return` statement that returns both `playerOne` and `playerTwo` variable The `return` statement should look as follows:

```
return playerOne, playerTwo
```

Step 11: The next function to code is the `rollDice` function. Write the function header to match the function call in Step 8. This function body will call the `random` function to determine `p1number` and `p2number`. The code should look as follows:

```
p1number = random.randint(1, 6)
p2number = random.randint(1, 6)
```

Step 12: Next, inside this function write a nested `if else` statement that will set `winnerName` to either `playerOne`, `playerTwo`, or "TIE".

Step 13: The final step in this function is to return `winnerName`.

Step 14: The final function to code is the `displayInfo` function. Write the function header to match the call made in Step 9. The body of the function should simply print the `winnerName` variable to the screen.

Step 15: **Submit this completed word document and .py source file to D2L.**