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

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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. Help Video: Double click the file to view video  |

**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, page 225).

1. Write a function with the header named addTen.
2. The function will accept an Integer variable named number.
3. 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.
4. The return statement will return the value of number.

Function **a.**integer **a.**addTen (**b.**integer Number)

 Display “Enter a number:”

 Input **c.**number

Set **c.**number = number + 10

Return **d.**number

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

Set number = addTen (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, page 238).

1. Write a function with the header named gender.
2. The function will accept a Boolean variable named answer.
3. The function body will ask the user to enter their gender into the variable type and then determine if they are male or female with an if statement.
4. The return statement will return the value of answer.

Function **a.**boolean **a.**gender(**b.**boolean answer)

 Declare String type

Display “Enter your gender (male or female):”

 Input **c.**type

If (**c.**type == “male”) then

 answer = False

Else

 answer = True

End If

Return **d.**answer

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

Set answer = gender (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, page 240).

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

Declare Integer **a.**myNUmber\_

Declare Real **a.**squareRoot

 Display “Enter a number:”

 Input **b.**myNumber

 Set **c.­­­­­­**squareRoot = sqrt myNUmber

 Display “The square root is”, **d.**squareRoot

**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, page 246).

1. Declare a variable named subtotal, a constant variable named tax set to the rate of .06, and a variable named total.
2. Ask the user to enter the subtotal. Store the input in subtotal.
3. Calculate the total as subtotal + subtotal \* tax.
4. Make a call to the currencyFormat function and pass it total. Since you are not displaying it on this line, simply set the return value to total.
5. Display the total to the screen.

Declare Real **a.**subTotal

Declare Constant Real **a.**tax =.06

Declare Real **a.**total

 Display “Enter the subtotal:”

 Input **b.**subTotal

 Set **c.­­­­­­**total = subTotal + subTotal \*tax

 total = **d.**currencyFormat(total)

 Display “The total is $”, **e.**total

**Lab 7.2 – Functions and Flowcharts**

|  |
| --- |
| Critical Review When creating a flowchart for a program that has functions, draw a separate flowchart for each function.  The starting terminal symbol usually shows the name of the function, along with any parameters that the function has. The ending terminal symbol reads Return, followed by the value or expression being returned. In Raptor, there are built-in procedures and functions that perform a wide variety of tasks on the programmer's behalf, saving development time and reducing the chance for errors. Raptor's buily-in functions can return values, but modules made by the user do not have that ability. Raptor has the following built-in functions.basic math: rem, mod, sqrt, log, abs, ceiling, floortrigonometry: sin, cos, tan, cot, arcsin, arcos, arctan, arccotmiscellaneous: random, Length\_of If you want to learn what each of these functions do, use the Help menu in Raptor and search for the function name. While ceiling and floor round a number to the nearest integer, there is no function in Raptor that will round a number just to two decimal places. The random function in Raptor takes no arguments. To generate a random integer from 1 to n, use floor((random\*n) + 1). For example, you can simulate the roll of a die (random number from 1 to 6) with floor((random \* 6) + 1). Help Video: Double click the file to view video  |

This lab requires you to create the flowchart from page 222 on Using Random Numbers using the RANDOM function. Use an application such as Raptor or Visio.

**Step 1:** Start by reading the pseudocode on page 221 and 222 of your textbook on Using Random Numbers. In addition to simply displaying the random values, your program will also meet the following requirements:

* Allow the two players of the dice game to enter their names in variables named playerOne and playerTwo.
* Based on the random roll of the dice, your program will determine which value is higher or if they tie and declare one player a winner.
* Create structure in your program by creating the following modules:
	+ An inputNames( ) module that will ask the players to enter their names
	+ A rollDice( ) module that will call the RANDOM function and determine the winner. This will be done with a decision statement.
	+ A displayInfo( ) module that will print the winners name to the screen.
* Additionally, your program should allow the same players to play as many times as they want.

**Step 2:** Start Raptor and save your document as *Lab 7-2*. The *.rap* file extension will be added automatically.

**Step 3:** Start by adding a comment box with the necessary variables.

**Step 4:** Add your loop to run multiple times and your module calls in the main module. Your flowchart might look as follows:

**Step 5:** Code the inputNames( ) module so both players can enter their name into the appropriate variable.

**Step 6:** Go to the rollDice( ) module and add an assignment statement. Assign p1number to whatever the RANDOM function return. The assignment box input box should look as follows:

**Step 7:** Add a second assignment statement and do the same for p2number.

**Step 8:** Add a selection statement that will determine which number is larger, or if there is a tie. The best way to do this is to create a nested if else where you first check to see if p1number is equal to p2number. If so, assign winnerName equal to “TIE”. If not, create a second decision to see if p1number is greater than p2number. If so, then winnerName should be set equal to playerOne and if not then winnerName should be set equal to playerTwo.

**Step 9:** Go to the displayInfo( ) module and print the winners name to the screen.

**Step 10:** Paste your finished flowchart in the space below.

 **PASTE FLOWCHART HERE**

 

**Lab 7.3 – Python Code and Random**

The goal of this lab is to convert the Dice Game in Lab 7.2 to Python code.

**Step 1:** Start the IDLE Environment for Python. Prior to entering code, save your file by clicking on File and then Save. Select your location and save this file as *Lab*7-3*.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':

 #initialize variables

 #call to rollDice

 #call to displayInfo

 endProgram = raw\_input('Do you want to end program? (Enter yes or 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 documentation for adding libraries, add the following statement:

import random

**Step 5:** Under the documentation for initialize variables, set endProgram to ‘no’ and playerOne and playerTwo to ‘NO NAME’.

**Step 6:** Under the documentation 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 raw\_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 name, playerTwo name, 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:** Execute your program so that all loops work and paste the final code below

Lab 7-3 The Dice Game

sean srock

import random

#the main function

def main():

 print

 #initialize variables

 endProgram = 'no'

 playerOne = 'NO NAME'

 playerTwo = 'NO NAME'

 #call to inputNames

 playerOne, playerTwo = inputNames(playerOne, playerTwo)

 #while loop to run program again

 while endProgram == 'no':

 #initialize variables

 p1number = 0

 p2number = 0

 winnerName = 'NO NAME'

 #call to rollDice

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

 #call to displayInfo

 displayInfo(winnerName)

 #end of while loop

 endProgram = raw\_input('Do you want to end program? (Enter yes or no): ')

#this function gets the players names

def inputNames(playerOne, playerTwo):

 playerOne = raw\_input('Enter Player 1 Name: ')

 playerTwo = raw\_input('Enter player 2 Name: ')

 return playerOne, playerTwo

#this function will get the random values

def rollDice(p1number, p2number, playerOne, playerTwo, winnerName):

 p1number = random.randint(1, 6)

 p2number = random.randint(1, 6)

 if p1number == p2number:

 winnerName = "TIE"

 elif p1number > p2number:

 winnerName = playerOne

 else:

 winnerName = playerTwo

 return winnerName

#this function displays the winner

def displayInfo(winnerName):

 print 'The winner is', winnerName

# calls main

main()

**Lab 7.4 – Programming Challenge 1 – Math Test**

Write the Flowchart and Python code for the following programming problem based on the pseudocode below.

Help Video for Raptor: Double click the file to view video

Help Video for Python: Double click the file to view video

Write a program that will allow a student to enter their name and then ask them to solve 10 mathematical equations. The program should display two random numbers that are to be added, such as:

 247

+ 129

The program should allow the student to enter the answer. The program should then display whether the answer was right or wrong, and accumulate the correct values. After the 10 questions are asked, calculate the average correct. Then display the student name, the number correct, and the average correct in both decimal and percentage format.

In addition to any system functions you may use, you might consider the following functions:

* A function that allows the student to enter their name.
* A function that gets two random numbers, anywhere from 1 to 500.
* A function that displays the equation and asks the user to enter their answer.
* A function that checks to see if the answer is right and accumulates the number right.
* A function that calculates the results.
* A function that displays the student name, the number right, and the average right.

Your sample output might look as follows (random numbers will be different):

Enter Student Name: Katie

What is the answer to the following equation

424

+

28

What is the sum: 472

Wrong

What is the answer to the following equation

163

+

233

What is the sum: 396

Right

What is the answer to the following equation

285

+

453

What is the sum: 688

Wrong

Etc…(through 10 iterations)

Information for student: Katie

The number right: 5

The average right is 0.50 or 50.0 %

**The Pseudocode**

Module main()

 //Declare local variables

 Declare Integer counter = 0

 Declare String studentName = “NO NAME”

 Declare Real averageRight = 0.0

 Declare Real right = 0.0

 Declare Integer number1 = 0

 Declare Integer number2 = 0

 Declare answer = 0.0

 Set studentName = inputNames()

 //Loop to run program again

 While counter < 10

 //calls functions

Call getNumbers(number1, number2)

Set answer = getAnswer(number1, number2, answer)

Set right = checkAnswer(number1, number2, answer, right)

Set counter = counter + 1

 End While

 Set averageRight = results(right, averageRight)

 Call displayInfo(right, averageRight, studentName)

End Module

Function String inputNames(String studentName)

 Display “Enter Student Name:”

 Input studentName

 Return studentName

End Function

Module getNumber(Integer Ref number1, Integer Ref number2)

 Set number1 = random(1, 500)

 Set number2 = random(1, 500)

End Module

Function Integer getAnswer(Integer number1, Integer number2, Integer answer)

 Display “What is the answer to the following equation”

 Display number1

 Display “+”

 Display number2

 Display “What is the sum:”

 Input answer

 Return answer

End Function

Function Integer checkAnswer(Integer number1, Integer number2, Integer answer, Integer right)

 If answer == number1 + number2 then

 Display “Right”

 Set right = right + 1

 Else

 Display “Wrong”

 End If

 Return right

End Function

Function Real results (Integer right, Real AverageRight)

 Set averageRight = right / 10

 Return averageRight

End Function

Module displayInfo(Integer right, Real averageRight, String studentName)

 Display “Information for student:”, studentName

 Display “The number right:”, right

 Display “The average right is:”, averageRight

End Module

**The Flowchart**

 

**The Python Code**

 **PASTE CODE HERE**

#Lab 7-5 Math Quiz and Averages

sean srock

import random

#the main function

def main():

 print

 #initialize variables

 counter = 0

 studentName = 'NO NAME'

 averageRight = 0

 right = 0

 number1 = 0

 number2 = 0

 #call to inputNames

 studentName = inputNames()

 #while loop to run program again

 while counter < 10:

 #initialize variables

 number1, number2 = getNumbers()

 answer = getAnswer(number1, number2)

 right = checkAnswer(number1, number2, answer, right)

 counter = counter + 1

 #end of while loop

 print

 averageRight = results(right, averageRight)

 displayInfo(right, averageRight, studentName)

#this function gets the players names

def inputNames():

 studentName = raw\_input('Enter Student Name: ')

 return studentName

def getNumbers():

 number1 = random.randint(1, 500)

 number2 = random.randint(1, 500)

 return number1, number2

def getAnswer(number1, number2):

 print 'What is the answer to the following equation'

 print number1

 print '+'

 print number2

 answer= input('What is the sum: ')

 return answer

def checkAnswer(number1, number2, answer, right):

 if answer == number1 + number2:

 print 'Right'

 right = right + 1

 else:

 print 'Wrong'

 return right

def results(right, averageRight):

 averageRight = float(right) / 10

 return averageRight

def displayInfo(right, averageRight, studentName):

 print

 print 'Information for student: ', studentName

 print 'The number right: ', right

 print 'The average right is %.2f'%(averageRight), 'or', averageRight \* 100, '%'

# calls main

main()