Module 5 Lab: Repetition Structures

This lab accompanies Chapter 3 - Control Statements and Program Development in *Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and The Cloud*.

Module 5 Lab Part 1 –Repetition Structures Pseudocode: Condition Controlled Loops

Critical Review

A repetition structure causes a statement or set of statements to execute repeatedly.

Repetition structures are used to perform the same task over and over.

Repetition structures are commonly called loops

A condition-controlled loop uses a true/false condition to control the number of times that it repeats.

The general structure of a While loop with a condition-controlled statement is:

```
//Declare loop control variable
While condition
     Statement
     Statement
     Etc.
     //Ask Question that changes the loop control variable
End While
```

The general structure of a Do While loop with a condition-controlled statement is:

```
//Declare loop control variable
Do
     Statement
     Statement
     Etc.
     //Ask Question that changes the loop control variable
While Condition
```

This lab requires you to implement a condition controlled loop.

Note that while many software languages have a do-while-loop, Python does not.

Step 1: Examine the following `code` from **Decisions and Boolean Logic Lab from Module 4 Part 2**. Loops are commonly used to call blocks of code multiple times. The common design pattern is to use a while or for loop around the block of code.

```
//Declare local variables
Declare Real monthlySales
Declare Real storeAmount
Declare Real empAmount
Declare Real salesIncrease

//Code to be repeated
# include code to get the monthly Sales
# include code to get the Increase in Sales
# include code to Calculate the Store Bonus
# include code to Calculate the Employee Bonus
# include code to print out all the results
```

Step 2: In the space provided below, create a loop control variable named `keepGoing` of the data type `String`. Initialize this variable to "y".

```
keepGoing = “ y ”                    //insert code to the left
```

Step 3: In the space provided, write a `While` statement.

```
// Declare local variables
Declare Real monthlySales
Declare Real storeAmount
Declare Real empAmount
Declare Real salesIncrease
keepGoing = “ y ”                    //insert code to the left

// Function calls
While keepGoing == “ y ” :
   # include code to get the monthly Sales
   # include code to get the Increase in Sales
   # include code to Calculate the Store Bonus
   # include code to Calculate the Employee Bonus
   # include code to print out all the results
   Display "Do you want to run the program again? (Enter y for yes)."
   Input keepGoing
End While
```

Step 4: In the space provided below, create a loop control variable named `keepGoing` of the data type string. Initialize this variable to "y".

```
keepGoing =  “ y ”                  //insert code to the left
```

Step 5: In the space provided, write a `Do-While` statement.

```
// Declare local variables
Declare Real monthlySales
Declare Real storeAmount
Declare Real empAmount
Declare Real salesIncrease
keepGoing =  “ y ”                  //insert code to the left

// Do-While Loop
Do
   # include code to get the monthly Sales
   # include code to get the Increase in Sales
   # include code to Calculate the Store Bonus
   # include code to Calculate the Employee Bonus
   # include code to print out all the results
   Display "Do you want to run the program again? (Enter y for yes)."
   Input keepGoing
While keepGoing ==  “ y ”
```

Module 5 Lab Part 2 –Repetition Structures Pseudocode: Count Controlled Loops

Critical Review

A count-controlled loop repeats a specific number of times.

The loop keeps a count of the number of times that it iterates, and when the count reaches a specified amount the loop stops.

A variable, known as a counter variable, is used to store the number of iterations that it has performed.

The three actions that take place are initialization, test, and increment.

- Initialization: Before the loop begins, the counter variable is initialized to a starting value.
- Test: The loop tests the counter variable by comparing it to a maximum value.
- Increment: To increment a variable means to increase its value. This is done by adding one to the loop control variable.

Any loop can be used with a count-controlled loop.

A running total is a sum of numbers that accumulates with each iteration of a loop. The variable used to keep the running total is called an accumulator.

This lab requires you to write a complete program using a condition controlled loop, a counter controlled loop, and an accumulator. The program is as follows:

```
Write a program that will allow a grocery store to keep track of
the total number of bottles collected for seven days.

The program will calculate the total number of bottles returned
for the week and the amount paid out (the total returned times
.10 cents).

The output of the program should include the total number of
bottles returned and the total paid out.

The program will ask the user if they have more data to enter and
will end the program if they do not.
```

Step 1: In the pseudocode below, declare the following variables under the documentation for Step 1.

- A variable called `totalBottles` that is initialized to 0
  - o This variable will store the accumulated bottle values
- A variable called `counter` and that is initialized to 1
  - o This variable will control the loop
- A variable called `todayBottles` that is initialized to 0
  - o This variable will store the number of bottles returned on a day
- A variable called `totalPayout` that is initialized to 0

    - This variable will store the calculated value of totalBottles times .10
- A variable called `keepGoing` that is initialized to "y"
    - This variable will be used to run the program again

Step 2: In the pseudocode below, make calls to the following sections of code under the documentation for Step 2.

- Code to set the variable `totalBottles` that contains the total number of bottles for the week.
- Code to set the variable `totalPayout` using the variable `totalBottles.` It returns the total payout for the week.
- Code that prints out the total number of bottles collected and the total payout using the variables `totalBottles` and `totalPayout.`

Step 3: In the pseudocode below, write a condition controlled while loop around your function calls using the `keepGoing` variable under the documentation for Step 3.

Complete Steps 1-3 below:

```
// Step 1: Declare variables below
totalBottles = 0
counter = 1
todayBottles = 0
totalPayout = 0
keepGoing =  “ y ”

// Step 3: Loop to run program again
While keepGoing ==  “ y ”
   // Step 2: Code to set value of variables
   # code to set value of variable totalBottles
   # code to set value of variable totalPayout
   # code to print the number of total bottles and total payout

   Display "Do you want to enter another week’s worth of data?"
   Display "(Enter y or n)"
   Input keepGoing
End While
```

Step 4: In the pseudocode below, write the missing lines, including:

a. The 3 missing variable declarations and initializing them to 0
b. The missing condition (Hint: should run seven iterations)
c. The missing input variable
d. The missing accumulator
e. The increment statement for the counter

```
// getBottles code
    NBR_OF_DAYS = 7
    // Declare and initialize totalBottles, todayBottles, counter to 0
       a. totalBottles = 0
          todayBottles = 0
          counter = 0
    While b. counter < NBR_OF_DAYS:
       Display "Enter number of bottles returned for day #",
                counter, ":"
       Input c. todayBottles
       d. totalBottles = totalBottles + todayBottles
       e. counter += 1

     End While
```

Step 5: In the pseudocode below, write the missing lines, including:

a. The missing calculation

```
// calcPayout code
     PAYOUT_PER_BOTTLE = .10
     totalPayout = 0 // resets to 0 for multiple runs
     a. totalPayout = totalPayout * PAYOUT_PER_BOTTLE
```

Step 6: In the pseudocode below, write the missing lines, including:

a. The missing display statement
b. The missing display statement

```
// printInfo code
     a. Display  " Total bottles returned in a week are  " , totalBottles
     b. Display  " Total payout for the returned bottles is $ " , totalPayout
```

Module 5 Lab Part 3 – Python Code

Convert the Bottle Return program as developed above to Python code.

Be sure to convert the variable names and the module/function names from **camelCase** to **snake_case** in your 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 *CIS129_YourName_Lab5.py*. Be sure to include the .py extension.

Step 2: Document the parts of the program:

- The first few lines of your program to include your name, the date, and a brief description of what the program does.
- The purpose of each section of code should be documented
- The purpose of each while-loop should be documented

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

```
# Lab 5 The Bottle Return Program

# Step 1: Declare variables below
total_bottles = 0
counter = 1
today_bottles = 0
total_payout = 0
keep_going = " y "

# Step 2: Loop to run program again
While keep_going == " y "
      # Step 3: Code to set value of variables
      # code to set value of variable totalBottles
      # code to set value of variable totalPayout
      # code to print the number of total bottles and total payout

      Display "Do you want to enter another week's worth of data?"
      Display "(Enter y or n)"
      Input keep_going
End While
```

Step 4: Click Run and Run Module to see how your program processes. Test with the exact values below to verify the expected output:

```
>>>
Enter number of bottles for day #1: 346
Enter number of bottles for day #2: 238
Enter number of bottles for day #3: 638
Enter number of bottles for day #4: 890
Enter number of bottles for day #5: 1035
Enter number of bottles for day #6: 899
Enter number of bottles for day #7: 536

The total number of bottles collected is 4582
The total paid out is $ 458.2

Do you want to enter another week's worth of data?
(Enter y or n): y

Enter number of bottles for day #1: 425
Enter number of bottles for day #2: 342
Enter number of bottles for day #3: 235
Enter number of bottles for day #4: 539
Enter number of bottles for day #5: 485
Enter number of bottles for day #6: 321
Enter number of bottles for day #7: 128

The total number of bottles collected is 2475
The total paid out is $ 247.5

Do you want to enter another week's worth of data?
(Enter y or n): n
>>>
```

Step 5: Does the output of your program look EXACTLY like the above sample output?

Step 6: **Submit this completed word document and .py source file to D2L.**