**Module 03: Lab Project**

Algenis A. Romero

Cybersecurity Workforce Certification Training (CWCT)

Ivy Tech Community College

CWCT 140 - Introduction to Python for Cyber Security

Instructor David Lieberman

August 2023

**Module 3 Lab Project   
CWCT140  
Python Essentials**

In part I of this week’s lab, we will apply our newly minted control statements to the task of plotting bar graphs on the console. See the output below:

Text

Description automatically generated with medium confidence

Part I: Write a Python program to produce this output

This exercise will introduce you to subroutines -- a Python element we haven’t covered yet. Creating and calling a subroutine is extremely easy to do

The code snippet below lays out the structure of the program. The **PlotBars** subroutine accepts a data list and does the work of printing the bar graphs. The main program builds a data list and calls the **PlotBars** subroutine. In the program below, use the pseudo-code to build out the function. Run the Main program to test it.

# Function to plot bar graphs

def plotBars(data\_list, plotsymbol):

    # use a for loop to print 40 blank lines

    # this "clears" out the console window

    #

    # for each value in the list:

    #   print "value" number of plotsymbols using string replication

    print()

# Main Program

# Build a list of numbers between 1 and 50

theList = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

plotBars(theList, "\*")

print("The data is:", theList)

print()

# Function to plot bar graphs  
def plotBars(data\_list, plotsymbil):  
 # Use a for loop to print 40 blank lines  
 # this "clears" out the console windows  
 for b in range(40):  
 print()  
 # for each value in the list print "value" using string replication  
 for val in range(len(data\_list)):  
 print(data\_list[val] \* plotsymbil)  
 # print(list\_lengh)  
  
# Data list  
thelist = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]  
  
# Call the function and print the bars.  
plotBars(thelist, "\*")  
print("The data is: ", thelist)  
print()

**Paste Snapshots of Results Here**

A screen shot of a computer

Description automatically generated

Part II: Order Data Using a Bubble Sort

A bubble sort algorithm is designed to place a list of numbers in order – from lowest to highest. We can use the **plotBars** function above to show us the progress of a bubble sort.

Add the bubble sort subroutine shown below to the program. Call the bubble sort subroutine from the main program. You should see a series of bar graph plots. Each plot shows the progress of the bubble sort operation. When the bubble sort is finished, the last bar graph will reflect a properly ordered list (low to high)

def bubbleSort(sorted\_list):

    for outer\_index in range(len(sorted\_list)):

        # call plotBars subroutine

        for inner\_index in range(0, len(sorted\_list) - outer\_index - 1):

            if sorted\_list[inner\_index] > sorted\_list[inner\_index + 1]:

                temp = sorted\_list[inner\_index]

                sorted\_list[inner\_index] = sorted\_list[inner\_index+1]

                sorted\_list[inner\_index+1] = temp

    # call plotBars subroutine

def bubbleSort(sorted\_list):  
 for outer\_index in range(len(sorted\_list)):  
 # Sort subroutine  
 for inner\_index in range(0, len(sorted\_list) - outer\_index - 1):  
 if sorted\_list[inner\_index] > sorted\_list[inner\_index + 1]:  
 temp = sorted\_list[inner\_index]  
 sorted\_list[inner\_index] = sorted\_list[inner\_index + 1]  
 sorted\_list[inner\_index + 1] = temp  
 plotBars(sorted\_list, "\*")  
  
  
# Data list  
thelist = [50, 40, 30, 20, 10, 5, 4, 3, 2, 1]  
  
# Call the function and print the bars.  
bubbleSort(thelist)  
print("The data was: ", thelist)  
print()

**Paste Snapshot of Result Here**

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated