CMSC 201 Section 40

Spring 2020

Homework # 3 – For Loops and Lists

Due Date: Monday, February 24 2020 by 11:59:59 pm

Total Value: 40 points

Creating the hw3 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

Just as you did for Homework 1 and Homework 2, you should create a directory to store your Homework 3 files. We recommend calling it **`hw3`** and creating it inside the **`Homeworks`** directory inside your `201` directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1. (You don't need to make a separate folder for each file. You should store all Homework 3 files in the same **`hw3`** folder.)

Part 1: Calculating the Mean of a set of numbers (worth 8 points)

This code must be submitted as a file labeled hw3_part1.py

Prompt the user to enter a set floating point numbers. First, ask the user how many numbers they will enter. Then, use a loop to prompt for that many numbers, one number at a time, and store those numbers in a list.

Then, calculate the mean of those numbers, and print out the results in an informative and friendly format.

The mean – often called the arithmetic average – of a set of numbers is the sum of the numbers, divided by how many numbers there are. Your program should print out a meaningful message like

“The mean of the numbers you entered is : “ and then print the mean value you calculated.

Part 2: Calculating Media and Mode and Printing a Histogram of a Set of Integers (worth 12 points)

The code for all three sub-parts must be submitted as a single file labeled hw3_part2.py

In this part of the assignment, you will read in a set of integers between 0 and 9, inclusive. You will keep count of how many of each integer you have read in. When all the data items have been entered, you will calculate the median of the numbers that have been entered; calculate the mode of the numbers that have been entered; and then print out a histogram showing how many of each integer have been entered.

Some notes on this:

1. Define CONSTANTS that set the minimum integer allowed and the maximum integer allowed. For this program, the minimum will be set to 0 and the maximum will be set to 9. But it should be possible to rerun your

program with different values for minimum and maximum. HINT: after this program works correctly, try changing the value of the constant holding the minimum integer to 2; and the value of the constant holding the maximum integer to 8. Your program should work perfectly without changing any other line of code!!

2. Ask the user how many integers will be entered, and then prompt for and read that many integers. If the user does not enter an integer and your program crashes, that's okay. (But this is the last homework for which that will be okay.)

3. Have a variable that is a list that counts how many of each integer have been entered. Initialize it to all 0s. That is, if 0 through 9 are allowable integers, your list should be initialized to [0,0,0,0,0,0,0,0,0,0]. Each time you read in another integer, increment the appropriate list element by 1. If the first integer entered is a 5, your list should now look like [0,0,0,0,0,1,0,0,0,0]. If the user then enters a 4, a 5, a 7 and another 4, your list should look like [0,0,0,0,2,2,0,1,0,0].

Part 2a: Calculate the median:

Note: DO NOT USE pre-written python functions that calculate the median for you!! The point of this is to write the code yourself.

The median has an equal number of entries below it and above it in the list. (This might be off depending on the data; that's okay.) To calculate the median for this assignment: divide the total number of integers entered by 2. Note that this is integer division!! Now walk through the list, starting at either end. Sum the list entries you encounter. When you get to the point where the sum is greater than or equal to half the total entries, declare that to be the median. In other words, if your list of integers entered by the user looks like: [1,4,4,2,1,5,0,0,0,9]. You know you have 26 entries. If you start at the beginning , you know that the user entered 0 one time; they entered 1 four times; they entered 2 four times; they entered 3 twice; they entered 4 once; and they entered 5 five times. That’s a total of 17 entries. Since that’s more than half of 26, you can declare that 5 is the median. If you started at the other end of the list, the user entered 9 nine times, then nothing until you get to 5, which was entered 5 times. That’s 14 entries, which is more than half of 26 entries, so you can declare 5 to be the median working this way.

Part 2b: Calculate the mode:

The mode is the integer that was entered most often. Find the list entry that is the highest, and print out a message stating that that integer is the mode.

Part 2c: Print the histogram:

Now print a histogram of the data. Prompt the user to enter a character which will be printed in the histogram to represent data. Let’s say that the user enters “*”.

Print out a series of lines. On each line will be: the integer whose data is on this line; and then a number of “*” characters equal to the number of times that integer was entered.

Suppose we had the data above – the number of times each integer was entered is captured in the list [1,4,4,2,1,5,0,0,0,9]. If the user enters “*” as the character to be printed, your histogram should look like:

```
0 *

1 ****

2 ****

3 **

4 *

5 *****
```

```
6
7
8
9 *********
```

Part 3: Chesapeake Bay (Worth 10 points)

Write a program that prints the numbers from 1 to 30 (inclusive), one per line. However, there are three special cases where instead of printing the number, you print a message instead:

1. If the number you would print is **divisible by 5**, print the message:
   Horseshoe crabs have 5 eyes on the tops of their heads.
2. If the number you would print is **divisible by 4**, print the message:
   Rockfish, also known as stripers, have 4 nostrils.
3. If the number you would print is **divisible by 8**, instead print out:
   Blue crabs have 8 walking legs.

Note that for some instances, both messages #2 and #3 will be printed!

Your code must contain one constant!

Here is a *partial* sample output, showing from 1 to 18.

```
bash-4.1$ python3 hw3_part3.py
1
2
3
Rockfish, also known as stripers, have 4 nostrils.
Horseshoe crabs have 5 eyes on the tops of their heads.
6
7
Rockfish, also known as stripers, have 4 nostrils.
Blue crabs have 8 walking legs.
9
Horseshoe crabs have 5 eyes on the tops of their heads.
11
Rockfish, also known as stripers, have 4 nostrils.
13
14
Horseshoe crabs have 5 eyes on the tops of their heads.
Rockfish, also known as stripers, have 4 nostrils.
Blue crabs have 8 walking legs.
17
18
```

Part 4– Counting Box (Worth 10 points)

Create a program that will output a “counting” box.

The program should prompt the user for these inputs, **in exactly this order:**

1. The width of the box
2. The height of the box

For these inputs, you can assume the following:

- The height and width will be integers greater than zero

Using this width and height, the program will print out a box where there are **`width`** numbers on each line and **`height`** rows. The numbers must count up starting from 0 and should continue counting up (do not restart the numbering).

HINT: You can keep the **`print()`** *function from printing on a new line by using* **`end=" "`** *at the end:* **`print("Hello", end=" ")`**. *If you do want to print a new line, you can call print without an argument:* **`print()`**. *You can put anything you want inside the quotation marks – above, we have used a single space, to separate the numbers in the counting box. You can also use an empty string, a comma, or even whole words!*

(See the next page for sample output.)

Here is some sample output for **hw3_part4.py**, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

```
bash-4.1$ python hw3_part4.py
Please enter a width:  4
Please enter a height: 2
0 1 2 3
4 5 6 7

bash-4.1$ python hw3_part4.py
Please enter a width:  12
Please enter a height: 7
0 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69 70 71
72 73 74 75 76 77 78 79 80 81 82 83

bash-4.1$ python hw3_part4.py
Please enter a width:  11
Please enter a height: 13
0 1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40 41 42 43
44 45 46 47 48 49 50 51 52 53 54
55 56 57 58 59 60 61 62 63 64 65
66 67 68 69 70 71 72 73 74 75 76
77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 93 94 95 96 97 98
99 100 101 102 103 104 105 106 107 108 109
110 111 112 113 114 115 116 117 118 119 120
121 122 123 124 125 126 127 128 129 130 131
132 133 134 135 136 137 138 139 140 141 142
```

(NOTE: The “box” might not actually be a box. The number of digits increases as the value gets larger, and so the box gets wider.)

Submitting

Once your **hw3_part1.py**, **hw3_part2.py**, **hw3_part3.py**, and **hw3_part4.py** files are complete, it is time to turn them in with the **submit** command.

You may also turn in individual files as you complete them. To do so, only **submit** those files that are complete. You may resubmit any or all your files as many times as you like up until the due date/time. Be aware that when you resubmit a file, you are overwriting the last version that you submitted!

You must be logged into your account on GL, and you must be in the same directory as your Homework 3 Python files. To double-check that you are in the directory with the correct files, you can type **ls**.

```
linux1[3]% ls
hw3_part1.py  hw3_part3.py  hw3_part5.py
hw3_part2.py  hw3_part4.py
linux1[4]% 
```

To submit your Homework 3 Python files, we use the **submit** command, where the class is **cs201s**, and the assignment is **HW3**. Type in (all on one line, even if it wraps around the screen) **submit cmsc201s HW3 hw3_part1.py hw3_part2.py hw3_part3.py** , **hw3_part4.py** and press enter.

[redacted]

If you don’t get a confirmation, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 1. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**