CMSC 201 Section 60

Homework 3 - Lists and Loops

Due Date: Monday, October 3rd, 2022 by 11:59:59 PM
Value: 40 points

This assignment falls under the standard cmsc201 academic integrity policy. This means you should not discuss/show/copy/distribute your solutions, your code or your main ideas for the solutions to any other student. Also, you should not post these problems on any forum, internet solutions website, etc.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
"""
File:     FILENAME.py
Author:   YOUR NAME
Date:     THE DATE
Section:  YOUR DISCUSSION SECTION NUMBER
E-mail:   YOUR_EMAIL@umbc.edu
Description:
  DESCRIPTION OF WHAT THE PROGRAM DOES
"""
```

Creating Your HW3 Directory

```
linux3[1]% cd cmsc201/Homeworks
linux3[2]% mkdir hw3
linux3[3]% cd hw3
linux3[4]% 
```

Submission Details

Submit the files under the following titles:
(These are case sensitive as usual.)
submit cmsc201 HW3 {files go here}

Problem 1 - Frequency	freq.py
Problem 2 - Positive or Negative or ?	posOrNeg.py
Problem 3 - Finding the highest value	highest.py
Problem 4 - Finding the largest different	diff.py

For example you would do:

```
linux1[4]% submit cmsc201 HW3 freq.py posOrNeg.py highest.py
diff.py
Submitting freq.py...OK
Submitting posOrNeg.py...OK
Submitting highest.py...OK
Submitting diff.py...OK
linux1[5]% 
```

From here, you can use **`emacs`** to start creating and writing your different Homework 3 Python programs.

You don't need to and should not make a separate folder for each file. You should store all of the Homework 3 files in the same **`hw3`** folder.

Coding Standards

Coding standards for CMSC 201 can be found here.

For now, you should pay special attention to the sections about:

- Naming Conventions
- Use of Whitespace
- Comments (specifically, File Header Comments)
- Variable names.

Make sure to include **`if __name__ == '__main__':`** at the beginning of each file and indent your code inside it.

Input Validation

For this assignment, you do **not** need to worry about **SOME** “input validation.”

If the user enters a different type of data than what you asked for, your program may crash. This is acceptable.

Unlike in HW1 you didn't have to worry about any input validation since you didn't have if statements, but now you do, so you can worry a little about it. You can try to prevent invalid input, but only when that invalid input is of the correct type.

For example, if your program asks the user to enter a whole number, it is acceptable if your program crashes if they enter something else like “dog” or “twenty” or “88.2” instead.

But it's a good idea to try to catch a zero division error for instance, or entering negative numbers when only positive numbers are allowed.

Here is what that error might look like:

```
Please enter a number: twenty
Traceback (most recent call last):
  File "test_file.py", line 10, in <module>
    num = int(input("Please enter a number: "))
ValueError: invalid literal for int() with base 10: 'twenty'
```

Allowed Built-ins/Methods/etc

- Declaring and assigning variables, ints, floats, bools, strings.
- Casting int(x), str(x), float(x), (technically bool(x))
- Using +, -, *, /, //, %, **; +=, -=, *=, /=, //=, %=, **= where appropriate
- Print, with string formatting, with end= or sep=:
    - '{}'.format(var), '%d' % some_int, f-strings
    - Really the point is that we don't care how you format strings in Python
    - Ord, chr, but you won't need them this time.
- Input, again with string formatting in the prompt, casting the returned value.
- Using the functions provided to you in the starter code.
- Comparisons ==, <=, >=, >, <, !=, in
- Logical and, or, not
- if/elif/else, nested if statements
- Using import with libraries and specific functions **as allowed** by the project/homework.
- String Operations:
    - upper(), lower()
    - concatenation +, +=
- For loops, both *for i* and *for each* type.
- Lists, list(), indexing, i.e. my_list[i] or my_list[3]
    - 2d-lists if you want them/need them my_2d[i][j]
    - Append, remove
    - .join()
- split(), lower(), upper() for strings
- len() for strings and lists

Forbidden Built-ins/Methods/etc

This is not a complete listing, but it includes:

While loops

- sentinel values, boolean flags to terminate while loops

list slicing

If you have read this section, then you know the secret word is: scandalous.

String operations, strip(), join(), isupper(), islower()

- **string slicing**

Dictionaries

- creation using dict(), or {}, copying using dict(other_dict)
- .get(value, not_found_value) method
- accessing, inserting elements, removing elements.

break, continue

methods outside those permitted within allowed types

- for instance str.endswith
- list.index, list.count, etc.

Keywords you definitely don't need: await, as, assert, async, class, except, finally, global, lambda, nonlocal, raise, try, yield

The *is* keyword is forbidden, not because it's necessarily bad, but because it doesn't behave as you might expect (it's not the same as ==).

built in functions: any, all, breakpoint, callable, classmethod, compile, exec, delattr, divmod, enumerate, filter, map, max, min, isinstance, issubclass, iter, locals, oct, next, memoryview, property, repr, reversed, round, set, setattr, sorted, staticmethod, sum, super, type, vars, zip

exit() or quit()

If something is not on the allowed list, not on this list, then it is probably forbidden.

The forbidden list can always be overridden by a particular problem, so if a problem allows something on this list, then it is allowed for that problem.

Problem 1 - Frequency

It will be important to search through data (in a list) in order to make decisions. In this program, called freq.py, it will search through a list of integers after values are entered into the list after a value is given by the user to find within that list.

For example, if the user entered data into the list such as:

[1, 3, 2, 4, 3, 4, 2, 1, 2, 1, 4, 3, 1]

Then the user enters a number to determine the frequency of that instance within that list such as:

```
Please enter a number to check the frequency: 1
We found 4 value(s) of 1 within the data list given
```

Your program will collect six values from the user and enter them into a list. Then the program will ask for a number to look up, it will respond with the frequency found. 0 is a legit value.

Sample Output

```
linux[0]$ python3 freq.py
Please enter value #1 into the data list: 1
Please enter value #2 into the data list: 3
Please enter value #3 into the data list: 2
Please enter value #4 into the data list: 4
Please enter value #5 into the data list: 3
Please enter value #6 into the data list: 4
Please enter a number to check the frequency: 2
We found 1 value(s) of 2 within the data list given

linux[1]$ python3 freq.py
Please enter value #1 into the data list: 1
Please enter value #2 into the data list: 3
Please enter value #3 into the data list: 2
Please enter value #4 into the data list: 4
Please enter value #5 into the data list: 3
Please enter value #6 into the data list: 4
Please enter a number to check the frequency: 5
We found 0 value(s) of 5 within the data list given
```

Problem 2 - Positive or Negative or ?

In this program, called posOrNeg.py, will search through a list of integer values entered into the list. The program will determine whether the value is positive or negative. First, the user will be asked how many values to enter and then enter values. The program will do the rest in displaying if the values were positive or negative in the order given within the list. 0 is neither a positive nor negative number.

Your program will collect a number of values from the user, and enter them into a list. Then the program will respond with if the data is positive, negative or neither! Your program should work with ***ANY*** list length.

Sample Output

```
linux[0]$ python3 posOrNeg.py
How many items do you wish to enter into the list? 6
Please enter value #1 into the data list: 1
Please enter value #2 into the data list: -3
Please enter value #3 into the data list: 2
Please enter value #4 into the data list: -4
Please enter value #5 into the data list: -3
Please enter value #6 into the data list: 4
Value #1 [1] is positive
Value #2 [-3] is negative
Value #3 [2] is positive
Value #4 [-4] is negative
Value #5 [-3] is negative
Value #6 [4] is positive

linux[1]$ python3 posOrNeg.py
How many items do you wish to enter into the list? 8
Please enter value #1 into the data list: 1
Please enter value #2 into the data list: -3
Please enter value #3 into the data list: 2
Please enter value #4 into the data list: -4
Please enter value #5 into the data list: -3
Please enter value #6 into the data list: 4
Please enter value #7 into the data list: 0
Please enter value #8 into the data list: -14

Value #1 [1] is positive
Value #2 [-3] is negative
Value #3 [2] is positive
Value #4 [-4] is negative
Value #5 [-3] is negative
Value #6 [4] is positive
Value #7 [0] is unsigned
Value #8 [-14] is negative
```

Problem 3 - Finding the highest value

In this program, called highest.py, it will traverse through a list of integer values entered into the list, and the program will determine the highest value within that list. First, the user will be asked how many values to enter and then enter values. The program will do the rest in displaying the highest value within the list.

Your program will collect a number of values from the user, and enter them into a list. Then the program will respond with the highest value in the list ***without using the sort() method***. Your program should work with ***ANY*** list length.

Sample Output

```
linux[0]$ python3 highest.py
How many items do you wish to enter into the list? 6
Please enter value #1 into the data list: 1
Please enter value #2 into the data list: -3
Please enter value #3 into the data list: 2
Please enter value #4 into the data list: -4
Please enter value #5 into the data list: -3
Please enter value #6 into the data list: 4
The highest value is 4

linux[1]$ python3 highest.py
How many items do you wish to enter into the list? 8
Please enter value #1 into the data list: -1
Please enter value #2 into the data list: -3
Please enter value #3 into the data list: -2
Please enter value #4 into the data list: -4
Please enter value #5 into the data list: -3
Please enter value #6 into the data list: -4
Please enter value #7 into the data list: 0
Please enter value #8 into the data list: -14
The highest values is 0
```

Problem 4 - Finding the largest difference

In this program called diff.py, will search through a list of FLOAT values entered into the list. The program will determine the largest difference between two consecutive values throughout the list. First, the user will be asked how many values to enter and then enter values. The program will do the rest in displaying the largest difference between the two values consecutively within the list.

An an example: if the input is:

index	[0]	[1]	[2]	[3]	[4]	[5]
value	1.34	-3.23	2.34	-4.87	-3.98	4.37
diff.	X	4.57	5.57	7.21	0.89	8.35

Your program should work with ***ANY*** list length.

Sample Output

```
linux[0]$ python3 diff.py
How many items do you wish to enter into the list? 6
Please enter value #1 into the data list: 1.34
Please enter value #2 into the data list: -3.23
Please enter value #3 into the data list: 2.34
Please enter value #4 into the data list: -4.87
Please enter value #5 into the data list: -3.98
Please enter value #6 into the data list: 4.37
The highest difference between values is 8.35

linux[1]$ python3 diff.py
How many items do you wish to enter into the list? 8
Please enter value #1 into the data list: -1.34
Please enter value #2 into the data list: -3.23
Please enter value #3 into the data list: -2.43
Please enter value #4 into the data list: -4.87
Please enter value #5 into the data list: -3.98
Please enter value #6 into the data list: -4.37
Please enter value #7 into the data list: 0
Please enter value #8 into the data list: -14.99
The highest difference between values is -14.99
```