INF1002 Programming Fundamentals – Lab2

Topics:

1. Advanced data structures in python: list, tuple, dictionary
2. For/while loops
3. List Comprehensions
4. File I/O

Warmup exercises:

The following lab assignment requires the use of all topics discussed so far in the module. You may wish to practice some of the concepts with simple exercises before attempting the lab assignment. You are not required to include these exercises in your submission, though you may wish to do so, to help you in the lab test.

1. Evaluate the following expression

```
List1=['abc', 'bcd', ['123', 567], 789]
List1[1]
List1[2]
List1[2][2]

List1[4]
List1[-3]
List1[2:]
Tuple1=('23',15,8,100)
Tuple1[0]
Tuple1[1:]
Tuple1[:2]
Tuple1[2]=23
Dic={'A01':'xiaoming', 'A02':['mie',
3]}
Dic['A01']
Dic['A02']
Dic['A02'][0]
range(10)
range(2:10:2)
h_letters = [ letter for letter in 'human' ]
print( h_letters)
number_list = [ x for x in range(20) if x % 2 == 0]
print(number_list)
num_list = [y for y in range(100) if y % 2 == 0 if y % 5 == 0]
print(num_list)
obj = ["Even" if i%2==0 else "Odd" for i in range(10)]
print(obj)
```

2. Create one tuple and one list to store five of your favourite fruits (e.g. pear, apple, strawberry, banana, orange) separately

_____

_____

3. Print all the items that you have created from the list and tuple

_____

_____

4. Update one of the items from the list or tuple, e.g. change the ‘apple’ into ‘papaya’

5. Delete one of the items from the list or tuple, e.g. delete pear

6. Sort all the remaining fruits and print the ordered fruits out using for and while loops

7. Use the list comprehension to generate the lists according to below requirements:
   - Find all the numbers from 1-1000 that are divisible by 7
   - Find all the numbers from 1-1000 that have a 3 in them
   - Count the number of spaces in a string

8. Create one dictionary to store 12 months and its corresponding number of days. Use for and while loops to print out all the months and all the number of days.

9. You are going to design one program to check the popular words in a given document. Please download the Lab2_testData.txt from LMS. Note that the given data has a fixed scheme where each line is one long string and each string contains multiple keywords that are separated by “,”. You need to write one program to read this file and calculate the top 5 most frequent keywords and write out these 5 keywords in the end of the file. The following are some hints which may help you design this program.
   - String has a cool function that you can use to split a string separated by a ‘,’. For instance, given one string str1 = “apple, pear, peach”, to get all the keywords in str1, str1.split(‘,’) will return a list of keywords. You can use list1=str1.split(‘,’) to obtain all the keywords and put them into one list.
   - To get the top 5 most frequent keywords, you need to extract all the keywords first and figure out one way to calculate the frequency of each keyword. Then select the top 5 keywords.

   Input: The given text file “Lab2_testData.txt”
   Output: Print out the top 5 keywords both in the screen and write them to a new txt file “top_5.txt”.

   ******** You must try exercise 8 to learn how to do the Files I/O!*******

Lab Assignment:

To help you better practice, you need to perform a set of tasks in one auto-grading system, Gradescope in LMS/xSiTe. Gradescope will provide you immediate feedback of your program, such that you will know the issue of your program.

In this lab, you need to finish three tasks in Gradescope system. Gradescope system will provide you immediate feedback about the correctness of your program. You can view the feedback of your system by clicking the failed case. Gradescope adopts a test cases-based approach to check your program. Your score is depending on the number of test cases that you program can pass. Note that to train you to have a good programming practice, you must write your program strictly according to the requirement of the tasks, including your input and output format. If there is any difference (even one space), your program will fail on the test cases. SO, TRAIN YOURSELF TO BE AN EXACT THINKER!

To help you practice, you are allowed to do multiple attempts for each task. Enjoy your learning!

Note:

1. **It is noted that usage of '*quit()*', '*exit()*' and '*sys.exit()*' functions to quit/exit a conditional statement or function in Gradescope submission results in test case failure.**
2. **Use '*return*' or other means.**

Task 1: Count Popular Characters

Task Description:

In this task, you are going to design one program to check the popular characters in a given string. You need to write one program to calculate the top 5 most frequent characters. The following are some hints that may help you design this program.

- String has a cool function that you can use to return a copy of the string in which all case-based characters have been lowercased.
- To get the top 5 most frequent characters after sorting them, you need to extract all the characters first and figure out one way to calculate the frequency of each character. Then select the top 5 characters.
- The output must in the descending order of character frequency. If there are characters with the same frequency, they must be printed in ascending ASCII order.
- Print out the top 5 characters and their counts in the screen. (Your output should be in one line)

Running Examples:

C:\INF1002\Lab2\CountPopularChars>**python CountPopularChars.py sdsERwweYxcxeewHJesddsdskjjkjrFGe21DS2145o9003gDDS**

d:7,s:7,e:6,j:4,w:3

NOTE: For all the above tasks, please test your program using different input and make sure to document your code properly. Even though the functions will only be introduced next week, you are welcome to use functions to write the program if you know how to use them.

Instructions to submit and auto grade your code to Gradescope:

1. Download the skeleton code file "CountPopularChars.py" from the "Python\Lab\Lab2" xSite folder.
2. Add your code to the function "*def CountPopularChars()*".
3. Do not change file name or function names.
4. Drag and drop your locally tested code to Gradescope. You can submit and auto graded any number of times. Last submission counts.
5. There are 5 test cases and maximum score for this task is 5.
6. Autograder will be timed out after 10 minutes.

Task 2: Even Odd Calculator

Task Description:

In this task, you are assigned to develop one program to help users calculate different information based a series of user provided numbers (integers). Detailed requirement is provided as follows:

Input: A series of numbers

Output: (Your output should be in one line)

- The summation of the even numbers and summation of the odd numbers in the input list
- The difference between the biggest and smallest numbers in the input list
- The count of even numbers and odd numbers in the input list
- The "centered" average of the list of integers. The centered average can be calculated as the mean average of the values, after removing the largest and smallest values in the array. If there are multiple copies of the smallest value, ignore all but keep just one copy, and likewise for the largest value. For instance, [12,2,8,7,100] → 9; [2,2,8,11,100] → 7

To have a better understanding of the loops, please try to implement two programs: one uses "for" and another uses "while" loop.

Running Examples:

C:\INF1002\Lab2\EvenOddCalculator> **python EvenOddCalculator.py 12,2,8,7,100**

The sum of all even numbers is 122, the sum of all odd numbers is 7, the difference between the biggest and smallest number is 98, the total number of even numbers is 4, the total number of odd numbers is 1, the centered average is 9.

C:\INF1002\Lab2\EvenOddCalculator> **python EvenOddCalculator.py 1,2,abcd,8,11,200,301**

Please enter valid integers.

Instructions to submit and auto grade your code to Gradescope:

1. Download the skeleton code file "EvenOddCalculator.py" from the "Python\Lab\Lab2" xSiTe folder.

2. Add your code to the function "*def EvenOddCalculator()*".
3. Do not change file name or function names.
4. Drag and drop your locally tested code to Gradescope. You can submit and auto graded any number of times. Last submission counts.
5. There are 5 test cases and maximum score for this task is 5.
6. Autograder will be timed out after 10 minutes.

Task 3: Leap Year Calculator

Task Description:

In this task, you will develop a program to compute all the leap years within a specified time period. The user will input a start year and an end year. Your task is to determine how many leap years are included in this period and print out those leap years. The rule for determining leap years is as follows:

- A year is called a leap year, if the year is perfectly divisible by 4 - except for years which are both divisible by 100 and not divisible by 400. The second part of the rule effects century years. For example; the century years 1600 and 2000 are leap years, but the century years 1700, 1800, and 1900 are not. This means that three times out of every 400 years there are 8 years between leap years.

More information about the leap years rule can be found online.

Input: Two numbers (one is the start year, and another is the end year)

Output: The number of leap years and all the leap years (Your output should be in one line)

Note: In case of invalid input, print the message "Your input is invalid!".

Running Examples:

C:\INF1002\Lab2\LeapYearCalculator> **python LeapYearCalculator.py 1989 2000**

The number of Leap Years is 3, the Leap Years are 1992, 1996, 2000

Instructions to submit and auto grade your code to Gradescope:

1. Download the skeleton code file "LeapYearCalculator.py" from the "Python\Lab\Lab2" xSiTe folder.
2. Add your code to the function "*def LeapYearCalculator()*".
3. Do not change file name or function names.
4. Drag and drop your locally tested code to Gradescope. You can submit and auto graded any number of times. Last submission counts.
5. There are 5 test cases and maximum score for this task is 5.
6. Autograder will be timed out after 10 minutes.

Final submission and auto grading all three tasks together:

After testing all the above three tasks, submit (drag and drop) all three files *CountPopularChars.py*, *EvenOddCalculator.py* and

LeapYearCalculator.py together and auto graded for a maximum score of 15 for all the 3 tasks.

Note: It is not necessary to zip these 3 files.