CIS 41B - Assignment 1: review of CIS 41A, iterables and callables

Write an application with 2 modules:

- a *scores.py* module for a Scores class that reads data from a file of scores and lets the user selectively print the data.

- a *lab1.py* driver module with a main and functions that work with a Scores object.

Input file

The given input file, *input1.txt*, contains columns of scores for several countries. There are 6 columns in the file, one for each country. Don't use the constant 6 in your code. When I test your program I might use a file with 4 columns or 7 columns.

For each column of data, which is 1 country record:

- The first data is the country name abbreviation

- The rest of the data are scores for the country. As with the number of columns, don't hard code the number of scores.

The columns are separated whitespaces.

Scores class

Write a Scores class in a file called scores.py that does the following tasks:

* Allow the caller to pass in an input filename when the Scores object is created
* Read in data from the file and store it in an appropriate data structure, which should be an instance variable. The code should check for file open success and end the program with an error message if the file open fails.

+*2pts Extra Credit* if your code doesn't use any loop (comprehension doesn't count as a Python loop) to read in and store data. [Hint: take advantage of some of the new concepts covered in the class notes.]

* Have the following 4 public methods:

1. a method that prints all country abbreviations and their corresponding scores, *sorted by the total score* of each country. Each country name and scores are printed on one line, and all data are lined up in columns (see sample output). Your code *should not have to save the total score of each country in a variable*, instead take advantage of some of the concepts covered in the class notes.
2. a method that accepts a score limit and a boolean for going above or below the limit. The method prints the country abbreviation (not the whole record) of countries with scores that are either above or below the limit, depending on the boolean argument. You *should not have to write a loop to search for scores* that match the limit condition, instead take advantage of some of the concepts covered in the class notes.
3. a method that prints the frequency of all the scores. Each unique score is printed with its frequency count on one line, and the scores are lined up in columns (see sample output).
4. a method which is a generator that returns one country's record at a time. A country record is the abbreviation and corresponding scores of one country. The order of the country record being returned is *alphabetical order based on country abbreviation*. You can decide the format of the returned data.

There can be additional private methods if you see the need for them.

+1pt extra credit for using comprehension in the public methods when appropriate. But 1/2pt of extra credit will be deducted for every comprehension that isn't necessary.

In addition to the Scores class, the scores.py file also has a global printName decorator:

* The printName decorator prints the name of the function that's running, for debugging purpose.   
  And since printName will print the method name to screen, it also serves as a reminder to name your method descriptively, which is always a good idea. Do not use "method1", "method2"...
* Use printName with the public methods of 2 and 3 shown above (print sorted by total and print score frequency)

Driver code

In the lab1.py file write the following 6 functions:

1. A function that will:
   * print a menu of 5 choices: 1. Print by total score

2. Print by limit

3. Print one

4. Print score frequency

5. Quit

* + keep prompting the user for a choice of 1-5, until there is a valid choice to return.
  + print an appropriate error message when the choice is not valid.

1. A function that calls the Scores object method 1 to print all records sorted by the total scores.
2. A function that prompts the user for a score limit and a choice of above or below the limit. Then the function calls the Scores object method 2. If the user enters an invalid limit or invalid choice, re-prompt until there is valid input.
3. A function that calls the Scores object method 3 to print the frequency of the scores.
4. A function that will:
   * loop to let the user press the Enter key to see one country record at a time. The loop ends when the user enters any other character, or when there is no more country record.
   * use the Scores object generator (method 4) to get the record for one country and print the country abbreviation and corresponding scores on one line of output (see sample output)
   * if the user presses the Enter key too many times and there is no more data, print a "no more data" message
5. A main function that will:
   * create a Scores object, passing in the file "input1.txt".
   * loop to call the function of item 1 above until the user chooses 5.  
     In the loop, call the appropriate function (items 2, 3, 4, or 5) to process the user choice. The code should *not use an if elif statement* to determine which function to call. Instead use a concept that's in the class notes. (And don't even think of using 4 if statements as a solution, that's only for newbies.)

Lab requirements:

* Have a documentation block at the top of each source file with your name and short description of the file, such as "Scores class to read in data and print selected data" or "Driver code that uses Scores class"
* Have a docstring for each function and public methods. The docstring can be as simple as a one-line description of what the method does: ''' print all records above/below a limit ''' or you can also add descriptions for input arguments and return data.
* The Scores class must have all the methods described above and they should work as described.
* The driver code should have 6 functions and each function should work as described.

See sample output on the next page.

Sample output

1. Print by total score

2. Print by limit

3. Print one

4. Print score frequency

5. Quit

Enter choice: 1

printByTotal # function name, a result of the decorator

ZWE: 20 15 14 8 12 11 4 24 # numbers are in columns, right justified

UKR: 19 15 15 12 12 12 8 20 # lines are sorted by total scores

COL: 23 15 14 8 12 11 11 19

NZL: 19 15 14 13 13 12 11 24

MEX: 20 15 14 11 14 14 12 22

JPN: 22 15 13 13 14 14 13 22

1. Print by total score

2. Print by limit

3. Print one

4. Print score frequency

5. Quit

Enter choice: 2

Enter limit: a # invalid choice

limit must be a number

Enter limit: 10

Above or below 10? (a/b): b

countries with scores below 10

UKR COL ZWE

1. Print by total score

2. Print by limit

3. Print one

4. Print score frequency

5. Quit

Enter choice: 3

Press enter key to print a country record, or enter any character to end:

COL: 23 15 14 8 12 11 11 19

Press enter key to print a country record, or enter any character to end:

JPN: 22 15 13 13 14 14 13 22

Press enter key to print a country record, or enter any character to end: a

# get one country at a time, in alphabetical order

1. Print by total score

2. Print by limit

3. Print one

4. Print score frequency

5. Quit

Enter choice: 2

Enter limit: 22

Above or below 22? (a/b): c # invalid choice

Enter a for above, b for below

Above or below 22? (a/b): a

countries with scores above 22

COL ZWE NZL

1. Print by total score

2. Print by limit

3. Print one

4. Print score frequency

5. Quit

Enter choice: 3

Press enter key to print a country record, or enter any character to end:

MEX: 20 15 14 11 14 14 12 22 # continue from above

Press enter key to print a country record, or enter any character to end:

NZL: 19 15 14 13 13 12 11 24

Press enter key to print a country record, or enter any character to end:

UKR: 19 15 15 12 12 12 8 20

Press enter key to print a country record, or enter any character to end:

ZWE: 20 15 14 8 12 11 4 24

Press enter key to print a country record, or enter any character to end:

End of data # reach the end of all countries

1. Print by total score

2. Print by limit

3. Print one

4. Print score frequency

5. Quit

Enter choice: 3

Press enter key to print a country record, or enter any character to end:

End of data # still at end of all countries

1. Print by total score

2. Print by limit

3. Print one

4. Print score frequency

5. Quit

Enter choice: 4

scoreFrequency # function name, a result of the decorator

4: 1

8: 3

11: 5

12: 7

13: 5

14: 8

15: 7

19: 3

20: 3

22: 3

23: 1

24: 2

1. Print by total score

2. Print by limit

3. Print one

4. Print score frequency

5. Quit

Enter choice: a # invalid input

choice must be number

Enter choice: 12 # invalid input

Enter 1, 2, 3, 4, or 5

Enter choice: 5