CS384 2024 Lab 4 - July Dec 2024

****

Problem 4.1: Student Grades Dictionary
Write a Python program to create a dictionary where the keys are student names and the values are lists containing their grades. Implement functions to:
Student Names and Marks need to be taken as input. Assume case insensitive for names. So Anmol and anmol are same. Store name as lowercase for key value.

1. Add a new student with grades.
2. Update the grades of an existing student.
3. Calculate the average grade of each student.
4. Print all students with their average grades.
5. Sort students by their grades in descending order without using inbuilt function.

Example:

```
students = {
  'Anmol': [85, 90, 88],
  'Naresh': [78, 81, 85],
  'Neha': [92, 87, 90]
}
```

Output:

```
Anmol - Average: 87.67
Naresh - Average: 81.33
Neha - Average: 89.67
```

****

Problem 4.2: Frequency Analysis and Anagram Grouping
Write a Python program that takes a list of strings as input and uses a dictionary to categorize the strings into groups of anagrams. Then, for each group, calculate the total frequency of all characters. Finally, identify and return the group with the highest total character frequency. An anagram is a word or phrase formed by rearranging the letters of another word or phrase, typically using all the original letters exactly once. In other words, anagrams are words or phrases that contain the same letters, but in a different order.

Here are some examples:

1. Original word: Listen
Anagram: Silent

2. Original word: Acts
Anagram: Cats

Take input words from the users.

Example ["listen", "silent", "enlist", "inlets"]. These are the group of anagram for the word enlist (of any of the word thereof). Now In total of 4 words in this list ["listen", "silent", "enlist", "inlets"], we need the combined frequency of each character. So each character appears 4 times.

```
'enlist': {'l': 4, 'i': 4, 's': 4, 't': 4, 'e': 4, 'n': 4}
'goolge': {'g':4, 'o':4, 'l':2, 'e':2}
.. nd so on.
```

The expected output is:

# words should be stored in the order of the anagrams like this irrespective of user input. So user can enter cat, listen, tac, google ... but then it should be stored like this.

```
#Output
words = ["listen", "silent", "enlist", "inlets", "google", "goolge", "cat", "tac", "act"]

#Anagram Dictionary
{
   'enlist': ['listen', 'silent', 'enlist', 'inlets'],
   'goolge': ['google', 'goolge'],
   'act': ['cat', 'tac', 'act']
}
```