IF100
Files & Dictionary - Practice #2

Introduction

The aim of this example is to practice file operations and dictionaries in Python. The use of file operations is due to the nature of the problem; that is, you cannot finish this practicing assignment without using file operations. You may implement the solution without using a dictionary but that would be less efficient.

Description

Formula One, abbreviated as F1, is a worldwide known car racing regulated by Fédération Internationale de l'Automobile (FIA). For more information and detailed regulations, please refer to http://www.formula1.com.

In this exercise, you will write a program to generate a sorted Formula One pilot point table after reading race results from a file. This point table will consist of information about the performance status of Formula One pilots. Your program will read rankings of several races from an input file, and by processing these race results, for each pilot, your program will compute the total points earned. Then, your program will generate a sorted point table and print it on the shell. The details of the input file format and the rules of processing is explained below.

Input File

Your program will read race results from an input file. There will be several race results in the same file, where each race result begins with a constant line:

```
***RACE***
```

After this line, there comes the ranking list of the pilots for a particular race, line by line. There is only one pilot name on each line of this ranking list. At the end of the pilot ranking list of a particular race, there is an empty line. The first pilot name in this list is the winner of the race, the second name in this list is the runner-up (2^{nd} place), etc. An example race result is as the following. In this example race, Massa wins, Heidfeld is in the 2^{nd} place, Alonso is 3^{rd}, etc.

The number of pilots in the rank list may vary in each race, but you may assume that there will be at least 8 pilots in the ranking list of a particular race. Besides, a particular pilot will always be listed at most once in a particular race. However, you cannot make any other assumptions on this race data.

Please note that there will be several race results in the same input file. For a sample input file, you may check out the *race_results.txt* provided with this assignment, which is partially taken from Formula One 2008. You can observe from this sample that there are different numbers of pilots in each race iist. Therefore, your program should work for any number of pilots. Moreover, your program should also work for any pilot names. In the sample results file, there are seven (7) races, but we can have any number of races, it is NOT a fixed number. The only guarantee is that there will be at least 8 pilots in each race.

THEREFORE, PLEASE DO NOT MAKE ANY ASSUMPTIONS ON THE NUMBER OF PILOTS, THE NAMES OF THE PILOTS AND THE NUMBER OF RACES; JUST READ THE RACE RESULTS FILE AND PROCESS IT.

There will be no white space in the pilot names. In other words, we will use a single word for a pilot name. We will not use non-English letters in the pilot names. In this assignment, you may assume that the pilot names will be given in this way (i.e. with no blank within them and non-English letters will not be used).

The structure of the input file, the associated rules, assumptions and the things that you cannot assume, which are all explained above, are fixed such that you cannot change them or make any other assumptions. No format check for the contents of the input file will be required. You may assume that all the data will be in the correct format.

The name of the input file will be a *keyboard* input. In other words, your program will prompt for this input.

Processing

For the ones who have no idea about Formula One rules, results and ranking system, let us give some basic information:

- In a race, all the pilots that finish the race are ranked (according to their times, but we do not use times in this assignment)

- According to the ranking list of a race, first 8 pilots get some points, but the rest gets nothing. Points of the first eight are 10, 8, 6, 5, 4, 3, 2, 1, respectively.

- The point table is a sorted list of pilots. The order is determined by the points that each pilot gained through all races in that season. If there is a tie in points, some other criteria are used to determine the order. These details are explained below.

As mentioned above, there will be several race results in the input file. A particular race starts with ***RACE*** tag. After this tag, we will have the pilot names (one pilot name per line) that determine the ranking of this race. You need to process the first 8 pilots of each race. Each rank has different points, but only the first 8 pilots get points. Thus, you have to be aware of the rank of the pilots. You can use a dictionary to store the data for several pilots.

After your program finishes reading and processing all of the race results from the input file and storing the necessary data in the dictionary, it should print the points table that includes the ranking of the pilots with respect to the *points* earned, in descending manner. Thus, pilots with higher *points* must be at higher positions in the table. If multiple pilots have the same *point*, then your program should display those pilots in alphabetical order (ascending) with respect to their names.

Hint: You can use built-in ***len()*** and ***max()*** functions together with the ***pop()*** method on dictionary objects and the ***sort()*** method on list objects.

Output

After the pilots are sorted, your program should display the point table. The format of this table is as described below.

The first line is the header of the table. This line is fixed and given below.

RANK - NAME - POINTS

After this header line, the point table will be displayed in rank order (from top to bottom). You should not display the pilots with zero points. Rank values should start from 1. You should display separate pilot data in separate lines.

For each pilot, 3 pieces of information should be displayed: (1) rank of the pilot in the table, (2) name of the pilot, and (3) points that the pilot gained. These 3 pieces of information must be displayed in this order and there must be spaces and dashes in between them.

Together with this assignment document, we provide a sample input file. Please see "Sample Runs" section in order to understand the flow of the program, the inputs and the outputs in a better way.

We highly recommend the use of dictionaries.

We are very aware that this homework can also be done by:

1) Using lists instead, which makes the implementation much complicated.

2) Using files only by parsing several times, which is very inefficient.

Also, we would like to remind that this practice example is given for your preparation for the final exam and dictionaries will be an important part of the exam. Doing this practice example by avoiding using dictionaries will not help you much in this direction.

We also highly recommend the use of functions.

Sample Runs

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are inputs taken from the user.

Sample Run

```
Please enter the filename for the race results: race_results.txt
RANK - NAME - POINTS
1 - Massa - 50
2 - Alonso - 34
3 - Hamilton - 34
4 - Heidfeld - 34
5 - Kubica - 25
6 - Raikkonen - 25
7 - Glock - 15
8 - Vettel - 15
9 - Rosberg - 10
10 - Coulthard - 8
11 - Trulli - 7
12 - Kovalainen - 6
13 - Webber - 5
14 - Bourdais - 2
15 - Barrichello - 1
16 - Nakajima - 1
17 - Piquet - 1
```

Programming and Coding Advice

It would be easier for you to implement the algorithm of this problem if you **first** try to **draw the flowchart** or **write the pseudocode** so that you can go over your solution to see if there are any errors.

Additionally, this homework is not short to implement and it will be very difficult and problematic to code the solution without decomposing the process into smaller pieces. Indeed, it is always a good idea to **use decomposition and pattern recognition** in programming, regardless of the length of the problem. Just define subproblems in the whole problem, and try to solve these subproblems before.