**IF100**

**Take Home Exam 5**

**Due January 17th 2022 Monday 23:59 (Sharp Deadline)**

**Introduction**

The aim of this homework is to practice on file operations and dictionaries. In addition, please try to utilize functions since they would reduce the complexity of your program.

**Description**

You are given a file named "songs.txt", which stores the top 584 most listened songs between 2010-2019. Your task is to develop a Python program that recommends songs with respect to the given genres.

In your program, the user will provide genre as an input. If the entered genre is not available in the "songs.txt" file, then your program will print an appropriate error message. Otherwise, it will suggest a song that is from the preferred genre. The suggested song must have the highest popularity score among the other songs. If there are more than one song in the preferred genre that has the highest popularity score, your program should then check the beats per minute score of those songs and pick the song that has the highest beats per minute score. On the other hand, you may assume that there will be no pairs of songs with the same popularity scores *and* the same beats per minute scores (at the same time) from the same genre.

The program will keep suggesting songs recursively until "quit" is entered as an input in a case sensitive manner. Please see the "Sample Runs" section in order to understand the flow of the program, the inputs and the outputs in a better way.

**Input File**

You will be given only one input file sample. The name of this input file will be songs.txt.

Each line of the input file will contain song information as in the following format;

song\_name | genre | popularity\_score | beats\_per\_minute\_score

**Ex:** \* Señorita | canadian pop | 79 | 117

* There are 4 pieces of information on each line for a specific song: name, genre, popularity and beats per minute, respectively. **Each piece of information is separated with a pipe that has spaces on both sides (" | ")**.
* Notice that each line starts with a star followed by a space ("\* ").
* The example given above represents that the song Señorita from the canadian pop genre has a popularity score of 79 with 117 beats per minute.
* You can assume that the file information will be given in the correct format. Thus, you don't have to perform any format check on the file content.
* You can assume that there are no pairs of songs with the same popularity scores *and* the same beats per minute scores from the same genre.
* Notice that the song genre is case sensitive, which means "electropop" and "ElecTRoPop" cannot be considered as same genres.
* Song names are unique. That is, each song occurs **only once** in the file.
* You can assume that there will be no empty lines in the file. However, you cannot make any assumptions on the number of lines of this file. Keep this in mind while you're preprocessing the content of the file.

**Input and Output**

The inputs of the program and their order are explained below. It is extremely important to follow this order since we automatically process your programs. ***Thus your work will be graded as 0 unless the order is entirely correct***. Please see the "Sample Runs" section for some examples.

The prompts of the input statements to be used has to be exactly the same as the prompts of the "Sample Runs".

* Do not change the file name. It should be "songs.txt".
* Your program should display the following prompt to get the genre input from the user.

Please enter the genre that you want to listen:

* If the user enters an invalid genre, then your program should keep asking till a valid one is entered.
  + When the user enters an invalid genre, the following error must be displayed, where *genre* is the input.

There is no such genre *genre*.

* + To close the program, "quit" must be taken as an input, in a case sensitive manner. Your program should print the following message after the quit operation is executed.

The program is closed.

* When suggesting a song, either the one from the same genre with the highest popularity score, or the one from the same genre with the highest beats per minute score if there are multiple songs with the same highest popularity score, your program should print the following message, where *song* is the name of the song, *popularity* is the popularity score and *bpm* is the beats per minute score.

You can listen *song* that has *popularity* popularity and *bpm* beats per minute.

**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. You have to display the required information in the same order and with the same words and characters as below.

**Sample Run 1**

Please enter the genre that you want to listen: **a**

There is no such genre a.

Please enter the genre that you want to listen: **pOp**

There is no such genre pOp.

Please enter the genre that you want to listen: **POP**

There is no such genre POP.

Please enter the genre that you want to listen: **pop**

You can listen Memories that has 99 popularity and 91 beats per minute.

Please enter the genre that you want to listen: **qUiT**

There is no such genre qUiT.

Please enter the genre that you want to listen: **QUİT**

There is no such genre QUİT.

Please enter the genre that you want to listen: **quit**

The program is closed.

**Sample Run 2**

Please enter the genre that you want to listen: **dance pop**

You can listen Lose You To Love Me that has 97 popularity and 102 beats per minute.

Please enter the genre that you want to listen: **Dance pop**

There is no such genre Dance pop.

Please enter the genre that you want to listen: **quit**

The program is closed.

**Sample Run 3**

Please enter the genre that you want to listen: **moroccan pop**

You can listen Boom Boom that has 53 popularity and 110 beats per minute.

Please enter the genre that you want to listen: **quit**

The program is closed.

**Sample Run 4**

Please enter the genre that you want to listen: **canadian latin**

You can listen Behind Your Back that has 18 popularity and 98 beats per minute.

Please enter the genre that you want to listen: **Quit**

There is no such genre Quit.

Please enter the genre that you want to listen: **edm**

You can listen Higher Love that has 88 popularity and 104 beats per minute.

Please enter the genre that you want to listen: **electro house**

You can listen Tired that has 70 popularity and 124 beats per minute.

Please enter the genre that you want to listen: **quit**

The program is closed.

**Sample Run 5**

Please enter the genre that you want to listen: **hip hop**

You can listen Me, Myself & I that has 78 popularity and 112 beats per minute.

Please enter the genre that you want to listen: **quit**

The program is closed.

**Sample Run 6**

Please enter the genre that you want to listen: **escape room**

You can listen Truth Hurts that has 90 popularity and 158 beats per minute.

Please enter the genre that you want to listen: **quit**

The program is closed.

**What and where to submit?**

You should prepare (or at least test) your program using Python 3.x.x. We will use Python 3.x.x while testing your take-home exam. Let us repeat,

* You must use Google Colab to develop your code from scratch (from beginning till the end), and then submit it **through SUCourse+ only**! Once you are done with developing your code on Google Colab, then you will copy your code to the CodeRunner to see if your program can produce the correct outputs. At the end, you will submit your code through CodeRunner (and SUCourse+). You should keep your Google Colab file until the end of the semester, we might want to look at this. If you fail to provide this Google Colab file anytime in the semester, you may not earn any credits from this Take Home Exam.
* In the CodeRunner, there are some visible and invisible (hidden) test cases. You will see your final grade (including hidden test cases) before submitting your code. Thus, it will be possible to know your THE grade before submitting your solution.
* You don't have to submit the txt file. Just paste your code to CodeRunner.
* **There is no re-submission**. You don't have to complete your task in one time, you can continue from where you left last time but you should not press submit before finalizing it. Therefore, you should make sure that it’s your final solution version before you submit it.
* **General Take-Home Exam Rules**
* Successful submission is one of the requirements of the take-home exam. If, for some reason, you cannot successfully submit your take-home exam and we cannot grade it, your grade will be 0.
* There is NO late submission. You need to submit your take-home exam before the deadline. Please be careful that SUCourse+ time and your computer time may have 1-2 minutes differences. You need to take this time difference into consideration.
* Do NOT submit your take-home exam via email or in hardcopy! SUCourse+ is the only way that you can submit your take-home exam.
* If your code does not work because of a syntax error, then we cannot grade it; and thus, your grade will be 0.
* Please submit your **own** work only. It is really easy to find "similar" programs!
* Plagiarism will not be tolerated. Please check our plagiarism policy given in the syllabus of the course.

Good luck!

Özgün Yargı

& IF100 Instructors