Traniee Assessment Exam 2

Introduction

The aim of this take-home exam is to practice on file operations and dictionaries in Python. The use of file operations is due to the nature of the problem; that's why you cannot finish this take-home exam without using file operations. You may implement the solution without using a dictionary, but that would be less efficient and much harder for you to code. Besides, beware that if we use a larger file in auto grading, your program may fail if you use lists instead of a dictionary.

Description

In this take-home exam, you will implement a Python program that advises you to buy a product of your choice from the best seller by looking at their ratings, and the program should keep asking for an input until the special input '`exit`' is entered.

Together with this description document, you are also given two input files, "products.txt", "ratings.txt". Your program can use these files to create the dictionaries of ratings and products that each company sells (please do not read these files from Google Drive, instead upload them to the Colab and read them. Otherwise your code will not work during grading and trying on GradeChecker). Then, your program will take a string input (refers to the item name that you want to buy) from the user, and find the company which sells the product that the user asked for. If there are multiple companies which sell the wanted product, then you should find the company with the best rating score. Finally, your program will print the name of the company that it finds.

To perform this operation, your program should read the given files, and keep the information in a format that can be used to find the companies selling the product asked, and select the company that has the highest rating.

If there is only one company selling the product or there is no company selling the product, you should display a message accordingly. If there are multiple companies selling the product, the program should select the company with the highest rating by using the information in ratings.txt and display the name of the company selected in a message.

Input Files

You will be given two input file samples, products.txt and ratings.txt. For both files you can assume there will be no empty lines in the files. However, you can not make any assumptions on the number of the lines of the files.

In ***products.txt*** there will be the company names and the products they are selling in a certain format. In every line there will be one company's information in the format given below.

companyname-item_#,item_#,item_#

Each line will start with the company name, then the products names will be given seperated with commas. There will be a dash character ("-") after the company name, you do not have to perform any input check. You can assume that company names will be unique. Meaning that; in the file there will be no companies with the same name. Each item name given on the same line will be separated by comma (","). Also, you can assume that the item names that the companies selling will be different from each other. You cannot make assumptions about the number of the items that the companies are selling other than there will be at least one item that each company will be selling. You may assume that company names and item names do not contain dash characters.

In the ***ratings.txt*** file, there will be company names and their ratings in a certain format. In every line there will be one company's information in the format given below.

companyname,rating/10

Each line will start with the company name, after a comma (",") the rating score of the company will be given over 10. You do not have to perform any input check, you may assume that it will be a numeric value in the [0, 10] range (both borders are included). You can also assume that company names given in the file will be unique. Also, you can assume that each company will have a different rating score.

For a sample input file, you may check out the products.txt and ratings.txt provided with this assignment.

Please note that *we may use other files while grading* your take-home exams (in the same format and structure surely). The content of the files will most probably change for grading purposes, but the name of the files will remain the same.

User Inputs

Your program will process the input files and ask for a string input being the name of the product. The program should keep asking this string input until `'exit'` is given as input. You do not need to perform any input check for the user input.

Processing the File Contents and Creating the Dictionaries

First and a very important step of this take home exam, is to read the files and store the information in dictionaries by processing the content of the files. You will need to use the string operations to parse the information given in the files. One way to solve this assignment with dictionaries would be creating two dictionaries (just a suggestion, number of dictionaries will depend on your algorithm). In one dictionary you could keep a list of companies as values who sell the product which is given as keys. This way you could use this dictionary to find the companies which sell the product given as user input. In other words, in this dictionary key will be a string which is an item name, values will be a list of strings (list of companies which sell this product). In the second dictionary you could keep the names of the companies as keys and their rating scores as values. You could use this dictionary to find the company with the highest rating score. Don't forget to process ratings as floating numbers (instead of strings). You may see samples of these dictionaries below.

```
items_companies = {
      "item_1" : ["comp_1", "comp_2"],
      "item_2" : ["comp_4", "comp_1", "comp_77"],
      .
      .
}
companies_ratings = {
      "comp_1" : 7.23,
      "comp_2" : 9.45,
      .
      .
}
```

As we stated before, this is just a suggestion. Maybe you may use a different number of dictionaries with different structures. It all depends on your algorithm.

The use of dictionaries is mandatory in this assignment.

We are very aware that this take-home exam can also be done by:

- Using multiple lists instead of a dictionary, which makes the implementation much more complicated.

We would like to remind that dictionaries will be a very important concept for the remaining part of the Data Analytics (DA) program. Doing this take-home exam by avoiding dictionary use will not help you much in this direction.

Additionally, GradeChecker includes a timeout method which terminates the execution if it takes longer than a predefined threshold. Using lists (multiple lists instead of a single dictionary) may result in your code being not executed entirely by GradeChecker, which would yield a grade of 0. Meanwhile, you can of course use the list data structure in your program for purposes other than storing the dataset(s).

Although it's not mandatory, we highly recommend the use of functions.

Outputs

If none of the companies sell the product entered by the user, the program should print

`"None of the companies sell this product."`

If there is only one company selling the product entered as input then the program should print

`"Only companyX sells this product."`

If several (more than one) companies sell the product entered as input, then the program should find the company with the best rating among them, and print

`"We suggest you to buy` *`productX`* `from` *`companyX`* `because it has the highest ranking as X.XX".`

Of course, the words ended with X should be replaced with the correct values.

Important Remarks on Working with Files

When it comes to working with files, there are some differences in the way we use Google Colab and GradeChecker.

While working in Google Colab, your code does not run on your computer, hence there is no direct access to the files you have on your computer. However, you may upload your text file(s) to Google Colab by running this two line snippet on a separate code cell (you may also use Colab's user interface to upload your files without the following lines):

```
from google.colab import files
uploaded = files.upload()
```

Every time you run this code, you will be prompted to upload a file from your computer. Please note that this snippet may not work for certain browsers, so Chrome/Chromium is recommended. Please also be reminded that you need to repeat the upload process every time you reopen your Colab file or you reconnect to the environment.

If you used the two lines to upload the files, and you are going to try your code on GradeChecker; then you should remove this code cell (or make them as comment lines) before you download the .py version of your code, as this snippet only works on Google Colab and it will cause error on GradeChecker.

Another note on GradeChecker is that GradeChecker does not have the txt file(s) related to the take-home exams already. So, while uploading your code to GradeChecker, you should upload the required text files together with your .py file (*3 files in total for this take-home exam: your main .py file, `product.txt` and `ratings.txt`*) and select your .py file as the main file to be executed. You should also upload your txt files to the SUCourse plus as well.

In short, if you use files.upload() method to upload the txt file to Google Colab, do not forget to erase the related lines before you upload your .py to GradeChecker and also to SUCourse.

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 product you want to buy:item_25
We suggest you to buy item_25 from comp_503 because it has the highest ranking as 10.0
Please enter the product you want to buy:exit
Goodbye!
```

Sample Run 2

```
Please enter the product you want to buy:item_87
None of the companies sell this product.
Please enter the product you want to buy:exit
Goodbye!
```

Sample Run 3

```
Please enter the product you want to buy:item_23
Only comp_339 sells this product.
Please enter the product you want to buy:exit
Goodbye!
```

Sample Run 4

```
Please enter the product you want to buy:item_8
We suggest you to buy item_8 from comp_682 because it has the highest ranking as 9.88
Please enter the product you want to buy:item_1
We suggest you to buy item_1 from comp_503 because it has the highest ranking as 10.0
Please enter the product you want to buy:item_0
```

None of the companies sell this product.
Please enter the product you want to buy:***item_50***
We suggest you to buy item_50 from comp_697 because it has the highest ranking as 9.9
Please enter the product you want to buy:***exit***
Goodbye!

Good Luck!