IF100
Take Home Exam 4
Due June 17th 2022 Friday 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 "`housePrices.txt`", which stores the prices of houses in Iran. Your task is to develop a Python program that displays the most expensive 5 neighborhoods for the house specification that the user has provided.

In your program, the user will provide two inputs for the number of rooms and the availability of a parking spot. By using provided specifications (room number and parking spot), you are supposed to find the average house price that matches requests for each address, and then print the most expensive 5 neighborhoods (addresses) based on these average values. The user may choose one or more specifications (room number and parking spot) as "any" which means all choices for that spec will be taken into consideration. If there are no houses that match specification, then the program should inform the user regarding unavailability.

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 `housePrices.txt`.

Each line of the input file will contain 5 pieces of information in the following format:

`Address_Room_Parking_Elevator:Price(USD)`

Ex: **`Shahran_1_True_True:61666.67`**

- As we said, there are 5 pieces of information on each line for a specific house: neighborhood name, number of rooms, has a parking spot or not, has an

elevator or not and price of the house, respectively. **Each piece of information is separated with an underline ("_") except price of the house. It is separated with a column (":")**.

- The example given above represents a house that is located on Shahran that has a parking spot, elevator and has exactly 1 room with a price of $6166.67.
- 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 duplications inside of 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 "`housePrices.txt`". While trying your code on CodeRunner, do not provide any path to open the file. You just need to execute the following command: **`open("housePrices.txt")`**
- Your program should display the following prompt to get the asked number of room input from the user.

```
Enter the room number:
```

    - If the user enters an invalid input, then your program should keep asking till a valid one is entered. The valid inputs are nonnegative integers or "any" string.
        - If "any" is entered, all existing room numbers should be taken into consideration.

        - Keep in mind that "any" is **case insensitive** which means "aNy" or "any" or "ANY" or "ANy" need to be considered as the same valid input.

- Your program should display the following prompt to get the if parking spot is necessary or not from the user.

```
Enter parking option:
```

    - If the user enters an invalid parking spot option, then your program should keep asking till a valid one is entered.

        - For the parking option, the user may enter either "True" or "False" or "any". If "any" is entered, all existing options ("True" and "False") should be taken into consideration.

        - Keep in mind that, "True", "False" and "any" are **case insensitive** which means "any"-"ANY" or "True"-"TrUE" or "False"-"fALSE" need to be considered as the same valid input.

- When inputs are taken, your program will find the average house price for each address for given specifications. After you calculate the average house price for each address, you should only display the most expensive 5 addresses based on their average prices. You should display the results **exactly** in the same format:

```
MostExpensiveAdressName1 AveragePriceOfAdress1

MostExpensiveAdressName2 AveragePriceOfAdress2

MostExpensiveAdressName3 AveragePriceOfAdress3

MostExpensiveAdressName4 AveragePriceOfAdress4

MostExpensiveAdressName5 AveragePriceOfAdress5
```

- There should be a horizontal tab character (`"\t"`) between the address name and the average price value. Additionally, the price values should be printed with 2 decimal places.

- `MostExpensiveAdressName1` is the most expensive address name while `MostExpensiveAdressName5` is the most expensive fifth address name.

- If there are less than 5 addresses (but at least 1) which satisfies the given criterias, then you should print them all in the correct order.

- You may assume that the average house prices for each address will be different from each other.

- If there are no houses that matches the asked specifications, display the following output:

```
No results for these options
```

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

```
Enter the room number: 2
Enter parking option: True
Zaferanieh  539444.44
Velenjak  415833.33
Dezashib  375000.00
Niavaran  353622.81
Araj  350000.00
```

Sample Run 2

```
Enter the room number: 3
Enter parking option: False
Lavasan 900000.00
Velenjak  523333.33
Araj  446666.67
Gheitarieh  416666.67
Farmanieh 230000.00
```

Sample Run 3

```
Enter the room number: any
```

```
Enter parking option: any
Gandhi  2333333.33
Lavasan 1600000.00
Mahmoudieh  1115555.56
Vanak 1090000.00
Elahieh 892878.43
```

Sample Run 4

```
Enter the room number: an
Enter the room number: anny
Enter the room number: AnY
Enter parking option: any
Gandhi  2333333.33
Lavasan 1600000.00
Mahmoudieh  1115555.56
Vanak 1090000.00
Elahieh 892878.43
```

Sample Run 5

```
Enter the room number: 5
Enter parking option: True
Abazar  3033333.33
Zafar 2666666.67
Mahmoudieh  2500000.00
Gandhi  2333333.33
Vanak 1916666.67
```

Sample Run 6

```
Enter the room number: 6
Enter parking option: any
No result for these options
```

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