Programing Assignment 3

Submission Date : 28.11.2018
Due Date : 12.12.2018 23:59
Advisors : Dr. Ahmet Selman BOZKIR, Dr. Fuat AKAL

1 Introduction

In this assignment, you will get familiar with file operations, lists and dictionaries while you implement a real world cinema automation application. Furthermore, you will have a hands-on experience on how to design required data structures to store related data for halls and seats in a movie theater while you do some operations such as selling or canceling tickets.

2 Assignment

In this assignment, you will be supposed to design and implement a console based basic movie theater automation which reads an input file (".txt" - will be given by an argument) line by line and executes each line which contains a command and its arguments -if any-. List of available commands is given in Section 3.

In our hypothetic movie theater, arbitrary number of halls can be created with a rectangular layout having the columns named with a sequence of 0, 1, 2, 3, ... while its rows are named with the characters of English alphabet (A, B, C, ..., Z). In Figure 1 this scheme has been illustrated. ***As a result, number of the rows of any hall is limited up to 26.*** In our cinema you should apply two types of fares for each seat (i.e. **student** or **full fare**). According to these tariffs, price of student and full fare tickets will be sold as **5** and **10** Turkish Liras, respectively.

In the following sections, you will find the descriptions of the commands and their arguments. Before delving into details please **note that your program must output the results of the operations to both console and a text file. The name of the output file must be "out.txt" and it must be created at the same folder where your python file exists.**

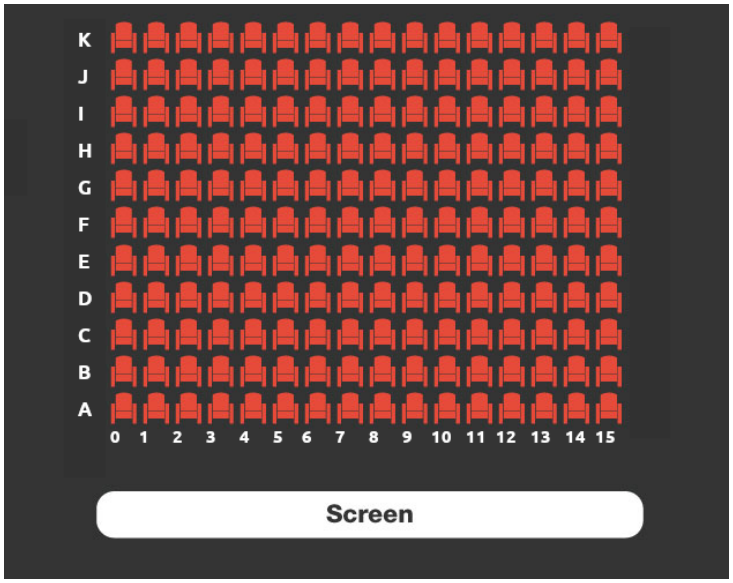

Figure 1: A sample cinema hall with its rows and columns

3 Commands

You are supposed to write a Python program to read an input file and to execute commands inside of it. The name of the input file must be accepted as an argument by your program. The name of your program must be **"main.py"**. The input files will reside in the directory where your "main.py" file resides.

3.1 CREATEHALL

This command expects two arguments in order to create a free hall. Initially all the seat must set to free. The syntax of the command is given below:

```
CREATEHALL hallname number_of_rowsxnumber_of_columns
```

Example command:

```
CREATEHALL bluehall 20x15
```

To this end, when you read such a line you must create a hall which will be called as "bluehall" and having 20 rows and 15 columns resulting of 300 total seats. If the operation is successfully carried out, you must output a success prompt like below:

```
The hall 'bluehall' having 300 seats has been created
```

The hall names will be unique. However, note that, once the hall is defined it will be no more possible to redefine it. So, your program must output an error message if you come across another "CREATEHALL" command involving the same name you have already assigned. The example error message is presented below:

```
Warning: Cannot create the hall for the second time. The cinema has already redhall
```

Since this command expects 2 arguments, supplying more or less than 2 arguments must cause warnings which are exemplified below. So, it is your duty to check command arguments of this and other command types.

```
Error: Not enough parameters for creating a hall!
Error: Too much parameters for creating a hall!
```

3.2 SELLTICKET

The command "SELLTICKET" enables you to sell tickets. This command fundamentally takes four arguments. However, it may take arbitrary number of arguments for multiple seats or seat ranges. The basic syntax is given below:

```
SELLTICKET customer_name full|student hall_name seat*
```

As can be seen, the first argument indicates customer name whereas the second one identifies the fare type. As told before, tickets are categorized into two groups which are either "full" or "student". The third argument stands for the target hall name in which the seat or seats will be purchased. The last argument (may have more than one) specifies the seat (e.g. B13), seats(e.g. B13 B14 A20 C2) or seat ranges (e.g. D2-15). Thus, you must deal with variational number of command arguments and process each of them separately. Some command examples are given below:

```
SELLTICKET ahmet full bluehall B12
SELLTICKET mehmet student bluehall B12 C3 B5 D4
SELLTICKET ayse student imax C2-4 D7 A6
SELLTICKET turgut full imax C3-15
SELLTICKET didem student redhall D7-32
```

Note that, your input file will involve lots of "SELLTICKET" commands each having a unique customer name. At this point your algorithm must handle all command arguments separately and check whether the respective seat or seat ranges are available to sell. As a rule, any command argument indicating individual seat (e.g. B12) should be checked for its availability whereas all the seats in a range must be completely checked before its purchase. In other words, you cannot sell a seat range even one of the seats in it is not available. As a result, output of each "SELLTICKET" command involves one or more than lines each indicating the result of each seat or seat ranges. Meanwhile, you must also check the hall name and number of command arguments against any exceptional case. Moreover, seat ranges or individual seats may exceed the hall sizes. For instance, if the seat label (e.g. C30) does not exists in the related target hall, your program must create a warning for that operation. Some example output lines are given below with respect to example commands above:

```
Success: ahmet has bought B12 at bluehall
Warning: The seat B12 cannot be sold to mehmet since it was already sold!
Success: mehmet has bought C5 at bluehall
Success: mehmet has bought B5 at bluehall
Success: mehmet has bought D4 at bluehall
Success: ayse has bought C2-4 at imax
Success: ayse has bought D7 at imax
Success: ayse has bought A6 at imax
Error: The seats C3-15 cannot be sold to turgut due some of them have already been sold!
Error: The hall 'redhall' has less column than the specified index D7-32!
```

3.3 CANCELTICKET

The command "CANCELTICKET" enables you to cancel some purchases. It has varying number of arguments such as "SELLTICKET". The basic syntax of the command is given below:

```
CANCELTICKET hall_name seat*
```

Similar to "SELLTICKET" command, it takes arbitrary number of command arguments and try to cancel the operation and revert the status of each seat in which each arguments points to. Same exceptional cases hold for "SELLTICKET" also exist for "CANCELTICKET". Furthermore, if the command argument points to any seat which has never been sold, you must output a warning. Some example commands and their respective outputs given in an ordered way below:

```
CANCELTICKET bluehall B12
CANCELTICKET imax C3-4 B5
CANCELTICKET redhall E33
CANCELTICKET bluehall G12-45
```

Output:

```
Success: The seat B12 at 'bluehall' has been canceled and now ready to be sold again
Success: The seats C3-4 at 'imax' have been canceled and now ready to be sold again
Error: The seat B5 at 'imax' has already been free! Nothing to cancel
Error: The hall 'redhall' has less column than the specified index D33!
Error: The hall 'bluehall' has less column than the specified index G12-G45
```

3.4 BALANCE

The command "BALANCE" is used to get a balance report of any hall with its revenues. The command takes only the hall names as arguments and the number of arguments can vary. The syntax of the command is given below:

```
BALANCE hall_name*
```

According to this definition some examples are listed below:

```
BALANCE bluehall
BALANCE imax redhall
```

When your program executes this command it must first parse the command line and list the balance information involving student, full fare and overall revenues. An example output is presented below:

```
Hall report of 'bluehall'
-------------------------
Sum of students = 45, Sum of full fares = 20, Overall = 65
```

As you may observe, the line below the first line has the same string length with the header line of the command output.

3.5 SHOWHALL

The command "SHOWHALL" is used to visualize the current layout of the halls with their seats and their actual status. The command takes only one argument that is the hall name. An instance of the command is given below:

```
SHOWHALL pinkhall
```

In this example output you see the hall layout according to a hall named "pinkhall". Pink hall has has 15 rows and 15 columns. As it can be seen, the rows are labeled with letters of the English alphabet and they are ordered in descending fashion (i.e. O, N, M, L, ..., A). Likewise, the columns must be placed one after another having 1 white space character. The columns must be started from 0 to the count of last column. In this visualization, the empty seats are illustrated with the character of 'X' while the seats sold for for students and full fare tickets can be shown with 'S' and 'F' respectively.

```
O X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
N X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
M X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
L X  X  X  X  S  X  X  X  X  X  X  S  X  X  X
K X  X  X  S  X  X  X  X  X  X  F  X  X  X  X
J X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
I X  X  X  X  F  F  X  X  X  X  X  X  F  X  X
H X  X  X  X  X  X  X  S  S  X  X  X  F  X  X
G X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
F X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
E X  X  X  X  X  X  F  X  F  X  X  X  S  X  X
D X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
C X  X  S  X  S  S  S  S  S  S  X  X  X  X  X
B X  X  X  S  S  X  X  X  X  X  X  X  X  X  X
A X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14
```

You should keep in mind that your output must be formed and listed exactly same as the example given above:

4 Important Notes

- Do not miss the submission deadline! Use the submit system. Submissions as e-mail attachments will not be accepted.
- Test your code with the input and respective output files prior to submission.
- Save all your work until the assignment is graded.
- Your assignment must be original and you must do it with your own individual work. Duplicate or very similar assignments are both going to be considered as cheating. You can ask your questions via Piazza and you are supposed to be aware of everything discussed on Piazza. At Piazza, you cannot share your source code completely or partially.

Assignments will be checked for similarity, and there will be serious consequences if plagiarism is detected.

- You may assume that the input le will be named arbitrarily and keep in mind that it is your task to read the program startup arguments.
- Instructor will run your program on a terminal screen as exampled below:

  ```
  python3 main.py inputblabla.txt
  ```

  So, your program must read the first argument and process the file it points and outputs the result into both console and "out.txt"
- Do not forget to keep your main() function simple, compact and informative. It is also important to write down comments for your code.
- The T.A. of this assignment has right to change any functional and non-functional part of this assignment via reporting on Piazza system. Any kind of these changes are valid and override this document.
- It is your duty to check the Piazza platform against any possible update about this assignment. If any instruction written by the TA violates any condition against this document, the new instruction(s) on Piazza is/are valid!