CSC 110 – Programming Project – Spring 2020

Movie Data

Objective:
The objective of this project is to write a program in Python that will read in a data file with movies that were released between 2000 and 2009 and display information requested by the user.

The Data:
The data file can be found here:

http://www.cs.uri.edu/~cingiser/csc110/assignments/movies.txt

This text file contains the following information about movies that were released between 2000 and 2009:

Title
The name of the movie, which is represented as a string. Some film titles are represented with more than one word.

Genre
The genre is the type of movie, such as Drama, Comedy, Documentary, etc.

Run Time
The run time is the overall length of the film in minutes. This should be represented as an integer.

Rating
This is the MPAA rating of the film, such as R, PG, PG-13. Some films have a rating of Unrated.

Studio
This is the studio that produced the film.

Year
This is the year that the film was released. This value should be stored as an integer.

The Program:
For this assignment, you will present the user with a list of options and you will display the results of the action chosen by the user. The program should continue to run until the user chooses to quit. The menu that you present to the user should look something like this:

```
>>> main()
Please enter a file name: movies
Invalid file name try again ...
Please enter a file name: movies.txt

Please choose one of the following options:
1 -- Find all films produced by a certain studio
2 -- Find the longest film of a specific genre
3 -- Find all films made in a given year range with a specific rating
4 -- Search for a film by title
5 -- Find average runtime of films with a certain rating
6 -- Sort all lists by year and write the results to a new file
7 -- Quit
Choice ==>
```

As you can see from the image, there are 7 different options offered to the user. Here is a brief description of each.

1) Find all films produced by a certain studio
   This option displays for the user all information about films made by a chosen studio. The program will prompt the user to enter the name of the studio, and ask for reentry if the studio is not in the list.

2) Find the longest film of a specific genre
   This option displays all information about the longest film of a specific genre. The program will prompt the user to enter the chosen genre, and ask for reentry if the genre is not in the list.

3) Find all films made in a given year range with a chosen rating
   This option displays all information about films with a chosen rating that were produced within a range of years chosen by the user. The program will prompt the user for the rating, and ask for reentry if the rating is not in the list. It will then prompt the user for the two years that represent the range. It should check to make sure that the years are valid numbers, and that the first year entered by the user is earlier than the second year.

4) Search for a film by title (note that the file is sorted by title)
   This option displays all information about the film with a specific title. The program will prompt the user for the title of a film. If that title does not exist in the list, the program will display a message indicating such. Because the file is sorted by title, the program should use the appropriate search algorithm.

5) Find the average runtime of films with a certain rating
   This option displays a single number representing the average runtime of films with a rating specified by the user. The program will prompt the user for the rating and ask for reentry if the rating is not in the list.

6) Sort all lists by year produced and write the results to a new file
   This option sorts the lists by the year the films were produced and writes the resulting sorted lists to a file. Note that you should not change the sorting order of the lists in your program. Rather, you should create a list of indexes that you sort based on the order of the year. You can use any sorting algorithm that you like. You may not use the python built-in sort function.

Program Requirements:
Your program should meet the following requirements:

1) Your program should *work correctly*. Your program should do at least the following correctly:
   a. Display the menu of options to the user.
   b. Ask the user the make a choice.
   c. Execute the choice made by the user and display the results.
   d. Continue until the user chooses to quit.
   e. Display an error message any time the user enters a value that is not valid, and allow the user to try again.

2) Your program should use good *modular design*. It should use functions for each of the main tasks of the program. Note that some of the tasks in this program are very similar, and if you design your code carefully, you can use the same function to perform several of the options the user may choose.

3) Your program should be *well-documented*. This should include your name and an overall description of the program at the top of the file. It should have a comment for each function describing the expected input parameters, what it returns, and if applicable, a description of any algorithms that are used in the function.

4) Your program should use *well-named* functions and variables. The code should be simple to read and understand what is going on given the names of the functions, and variables along with the comments in the code.

Examples:
The following screen shots give you an idea of what you should display for each of the options offered by the program. Note: not all results are shown in examples below.

Find all films produced by a certain studio

```
Please choose one of the following options:
1 -- Find all films produced by a certain studio
2 -- Find the longest film of a specific genre
3 -- Find all films made in a given year range with a specific rating
4 -- Search for a film by title
5 -- Find average runtime of films with a certain rating
6 -- Sort all lists by year and write the results to a new file
7 -- Quit
Choice ==> 1

Enter studio:fox
Invalid choice - try again:
Enter studio:Fox Searchlight

The films that meet your criteria are:

TITLE                                GENRE          TIME  RATING   STUDIO                 YEAR
28 Days Later                        SciFi/Fantasy  113   R        Fox Searchlight        2003
Once                                 Musical        85    R        Fox Searchlight        2007
Waking Life                          Drama          99    R        Fox Searchlight        2001
```

Find the longest film of a specific genre

```
Please choose one of the following options:
1 -- Find all films produced by a certain studio
2 -- Find the longest film of a specific genre
3 -- Find all films made in a given year range with a specific rating
4 -- Search for a film by title
5 -- Find average runtime of films with a certain rating
6 -- Sort all lists by year and write the results to a new file
7 -- Quit
Choice ==> 2

Enter genre:drama
Invalid choice - try again:
Enter genre:Drama
Longest film with genre  Drama  :

The films that meet your criteria are:

TITLE                                GENRE          TIME  RATING   STUDIO                 YEAR
Ray                                  Drama          152   PG-13    Universal Pictures     2004
```

Find all films made in a given year range with a specific rating

```
Please choose one of the following options:
1 -- Find all films produced by a certain studio
2 -- Find the longest film of a specific genre
3 -- Find all films made in a given year range with a specific rating
4 -- Search for a film by title
5 -- Find average runtime of films with a certain rating
6 -- Sort all lists by year and write the results to a new file
7 -- Quit
Choice ==> 3

Enter year range to search (oldest year first)
Year1: 2004
Year2: 2002
Second year should be after first year - try again
Year1: 2001
Year2: 2003
Enter rating:PG-13

The films that meet your criteria are:

TITLE                              GENRE        TIME  RATING  STUDIO                   YEAR
Dumb and Dumberer                  Comedy       85    PG-13   New Line Cinema          2003
Hollywood Homicide                 Action       116   PG-13   Columbia Pictures        2003
Hulk                               Action       138   PG-13   Universal Pictures       2003
K-19: The Widowmaker               Drama        138   PG-13   Paramount Pictures       2002
Malibu's Most Wanted               Comedy       86    PG-13   Warner Bros.             2003
National Security                  Drama        88    PG-13   Sony Pictures            2003
Pootie Tang                        Comedy       81    PG-13   Paramount Pictures       2001
Showtime                           Comedy       95    PG-13   Warner Bros. Pictures    2002
Simone                             Drama        117   PG-13   New Line Cinema          2002
Something's Gotta Give             Comedy       128   PG-13   Sony Pictures            2003
The Dish                           Drama        101   PG-13   Warner Bros. Pictures    2001
The Mothman Prophecies             Mystery      119   PG-13   Screen Gems              2002
The Recruit                        Mystery      115   PG-13   Touchstone Pictures      2003
The Sum of All Fears               Drama        124   PG-13   Paramount Pictures       2002
Undertaking Betty                  Drama        94    PG-13   Miramax Films            2003
What Women Want                    Comedy       127   PG-13   Paramount Pictures       2001
```

Search for a film by title

```
Please choose one of the following options:
1 -- Find all films produced by a certain studio
2 -- Find the longest film of a specific genre
3 -- Find all films made in a given year range with a specific rating
4 -- Search for a film by title
5 -- Find average runtime of films with a certain rating
6 -- Sort all lists by year and write the results to a new file
7 -- Quit
Choice ==> 4

Enter title:hulk
Invalid choice - try again:
Enter title:Hulk

The films that meet your criteria are:

TITLE                              GENRE        TIME  RATING  STUDIO                   YEAR
Hulk                               Action       138   PG-13   Universal Pictures       2003
```

Find average runtime of films with a certain rating

```
Please choose one of the following options:
1 -- Find all films produced by a certain studio
2 -- Find the longest film of a specific genre
3 -- Find all films made in a given year range with a specific rating
4 -- Search for a film by title
5 -- Find average runtime of films with a certain rating
6 -- Sort all lists by year and write the results to a new file
7 -- Quit
Choice ==> 5

Enter rating:pg
Invalid choice - try again:
Enter rating:PG
The average runtime for films with a  PG rating is 99.88
```

Sort all lists by year and write the results to a new file

```
Please choose one of the following options:
1 -- Find all films produced by a certain studio
2 -- Find the longest film of a specific genre
3 -- Find all films made in a given year range with a specific rating
4 -- Search for a film by title
5 -- Find average runtime of films with a certain rating
6 -- Sort all lists by year and write the results to a new file
7 -- Quit
Choice ==> 6

Enter name of output file: out-year.txt
```

Notes and Hints:

1) As mentioned in option 6 above, you may not use the python built-in sort function to sort your data.
2) You may not use the python built-in index function to find an item in a list. You should write the search code yourself.
3) You may use the python "in" function to determine if an element is in a particular list. For example, if you want to find out if a given `genre` is in the `genreList`, you can use:

```
if genre in genreList:
```

4) Here are a few hints about how to sort your data by year:

a) The parameters of the function should be the list of years. The function should return a list of indexes sorted by rearranging them the same way you will rearrange the years list. That is, suppose the years list looks like this:

```
[2003,2004,2005,2001,2000,2002]
```

and the titles list looks like this:

```
['Annie','Beethoven','Carrie','Moneyball','Once','Showtime']
```

You should create a list of indexes as follows:

```
[0,1,2,3,4,5]
```

Then your sorting algorithm will rearrange the list of indexes the same way the list of years will be rearranged, so we would end up with:

```
[4,3,5,0,1,2]
```

Your function will return this list of indexes.

b) You will have another function that takes as a parameter the list of indexes and all of the other lists and writes to an output file with all of the lists in the order specified by the index list. So, in the example above, your output file would look something like this:

```
Once,2000
Moneyball,2001
Showtime,2002
Annie,2003
Beethoven,2004
Carrie,2005
```

Of course, all of the other information about each film would be in the file as well.

5) To make a copy of a list, use the following syntax:

```
newList = oldList.copy()
```

If you have lists inside the list, then you will need to make a deep copy. This does a recursive copy and copies all lists that are inside of the original list:

```
newList = oldList.deepcopy()
```

6) To format your output for the movie data program, you can use the following syntax:

```
print(titles[i].ljust(40), genres[i].ljust(14),
str(runtimes[i]).ljust(5),ratings[i].ljust(8),studios[i].ljust(25
```

`rjust(40)` justifies the text to the right and allows 40 characters for it

`ljust(40)` left justifies the text (that is the letter L, not the number 1 at the beginning of the function name)

You can only justify a string, so if you want to print an integer or float, you have to convert it to a string first.

Extra Credit: (5 pts)

If you complete all of the required elements of the program, you can attempt this extra credit.

Write a function to find the studio that has produced the most films. Add an option to the menu to allow the user to choose this function. Print the result on the screen.

What to Submit:

Please submit a single `.py` file with the file name as follows:

`lastname_firstinitial_movies.py`

depending on which project you have chosen and replacing "`lastname`" with your own last name and "`firstinitial`" with your own first initial.