

CSC 110 – Programming Project – Spring 2022

Movies 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 2006 and 2016 and display information requested by the user.

The Data:

The data file can be found here:

[movies.csv](#)

This csv file contains the following information about movies that were released between 2006 and 2016:

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. Some films list more than one genre, separated by a semi-colon.

Director

The name of the director of the film.

Year

The year that the film was released. This should be represented as an integer.

Run Time

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

Revenue

The amount of money the film made in the box office in the millions of dollars. This should be represented as a float.

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:

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 made by a specified director

This option displays for the user all information about films made by a particular director. The program will prompt the user to enter the name of the director, and ask for reentry if the director is not represented in the data.

2) Find the highest grossing film made in a specific year

This option displays all information about the highest grossing film (highest revenue) made in a specific year. The program will prompt the user to enter the chosen year, and ask for reentry if the year is not in the list.

3) Find all films made in a given year range by a specified genre

This option displays all information about films in a specified genre that were produced within a range of years chosen by the user. The program will prompt the user for the name of the genre, and ask for reentry if the genre is not one of the genres represented in the data. Note that some films list more than one genre. This option should include all films where the genre is specified. 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 years in the range represented in the data, 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 revenue higher than a specified value

This option displays a single number representing the average runtime of films with a revenue higher than the value specified by the user. The program will prompt the user for the revenue threshold and ask for reentry if the entry is not a valid number.

6) Sort all lists by revenue and write the results to a new file

This option sorts the lists by the revenue of the films 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.

Functions

You are required to use functions when writing this program. You will have some freedom to design your functions. One requirement is that the functions that perform tasks requested by the user should NOT print their results. Rather, they should return the requested values to the main function, and have a separate function that prints the results. You should have a main function with no parameters and no return values. Do not include a call to the main function in your code because Gradescope will not be able to test it if you do.

User Input

Any time the user is asked for input, your program should check to make sure the input is valid. You can use exception handling to handle cases where the bad input will cause the program to crash. You can use the `in` function to check that user input is found in the list in question. For example, if you are asking the user to enter the name of a director, you can use the following to check if the director's name is in the list of directors in the data file:

```
if dirName in directorList:
```

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 to 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 made by a specified director

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

Please choose one of the following options:
1 -- Find all films made by a specified director
2 -- Find the highest grossing film made in a specific year
3 -- Find all films made in a given year range in a specified genre
4 -- Search for a film by title
5 -- Find average runtime of films with higher revenue than specified value
6 -- Sort all lists by revenue and write the results to a new file
7 -- Quit
Choice ==> 1

Enter director: Adam
Invalid entry - Try again
Enter director: Adam McKay

The films that meet your criteria are:
```

TITLE	GENRE	DIRECTOR	YEAR	RUNTIME	REVENUE(mil)
Step Brothers	Comedy	Adam McKay	2008	98	\$100.47
Talladega Nights: The Ballad of Ricky Bobby	Action;Comedy;Sport	Adam McKay	2006	108	\$148.21
The Big Short	Biography;Comedy;Drama	Adam McKay	2015	130	\$70.24
The Other Guys	Action;Comedy;Crime	Adam McKay	2010	107	\$119.22

Find the highest grossing film made in a specific year

```
>>> main()
Please enter a file name: movies.csv

Please choose one of the following options:
1 -- Find all films made by a specified director
2 -- Find the highest grossing film made in a specific year
3 -- Find all films made in a given year range in a specified genre
4 -- Search for a film by title
5 -- Find average runtime of films with higher revenue than specified value
6 -- Sort all lists by revenue and write the results to a new file
7 -- Quit
Choice ==> 2

Enter year: a
Invalid entry - Try again
Enter year: 1909
Year out of range, must be between 2006 and 2016
Enter year: 2010

The film that meets your criteria is:
```

TITLE	GENRE	DIRECTOR	YEAR	RUNTIME	REVENUE(mil)
Toy Story 3	Animation;Adventure;Comedy	Lee Unkrich	2010	103	\$414.98

Find all films made in a given year range in a specified genre

```
>>> main()
Please enter a file name: movies.csv

Please choose one of the following options:
1 -- Find all films made by a specified director
2 -- Find the highest grossing film made in a specific year
3 -- Find all films made in a given year range in a specified genre
4 -- Search for a film by title
5 -- Find average runtime of films with higher revenue than specified value
6 -- Sort all lists by revenue and write the results to a new file
7 -- Quit
Choice ==> 3

Enter year range to search (oldest year first)
Year1: a
Invalid entry - Try again
Year1: 2019
Year out of range, must be between 2006 and 2016
Year1: 2011
Year2: a
Invalid entry - Try again
Year2: 2009
Second year should be after first year - try again
Year1: 2011
Year2: 2013
Enter genre: Fun
Invalid entry - Try again
Enter genre: Horror

The films that meet your criteria are:
```

TITLE	GENRE	DIRECTOR	YEAR	RUNTIME	REVENUE(mil)
Carrie	Drama;Horror	Kimberly Peirce	2013	100	\$35.27
Dark Shadows	Comedy;Fantasy;Horror	Tim Burton	2012	113	\$79.71
Evil Dead	Fantasy;Horror	Fede Alvarez	2013	91	\$54.24
Final Destination 5	Horror;Thriller	Steven Quale	2011	92	\$42.58
Horns	Drama;Fantasy;Horror	Alexandre Aja	2013	120	\$0.16
Mama	Horror;Thriller	Andrés Muschietti	2013	100	\$71.59
Oculus	Horror;Mystery	Mike Flanagan	2013	104	\$27.69
Resident Evil: Retribution	Action;Horror;Sci-Fi	Paul W.S. Anderson	2012	96	\$42.35
Scream 4	Horror;Mystery	Wes Craven	2011	111	\$38.18
Sinister	Horror;Mystery	Scott Derrickson	2012	110	\$48.06
Texas Chainsaw 3D	Horror;Thriller	John Luessenhop	2013	92	\$34.33
The Cabin in the Woods	Horror	Drew Goddard	2012	95	\$42.04
The Conjuring	Horror;Mystery;Thriller	James Wan	2013	112	\$137.39
The Green Inferno	Adventure;Horror	Eli Roth	2013	100	\$7.19
The Mortal Instruments: City of Bones	Action;Fantasy;Horror	Harald Zwart	2013	130	\$31.17
The Purge	Horror;Sci-Fi;Thriller	James DeMonaco	2013	85	\$64.42
Under the Skin	Drama;Horror;Sci-Fi	Jonathan Glazer	2013	108	\$2.61
Underworld Awakening	Action;Fantasy;Horror	Mans Marling	2012	88	\$62.32
Warm Bodies	Comedy;Horror;Romance	Jonathan Levine	2013	98	\$66.36
World War Z	Action;Adventure;Horror	Marc Forster	2013	116	\$202.35

Search for a film by title (note that the file is sorted by title)

```
>>> main()
Please enter a file name: movies.csv

Please choose one of the following options:
1 -- Find all films made by a specified director
2 -- Find the highest grossing film made in a specific year
3 -- Find all films made in a given year range in a specified genre
4 -- Search for a film by title
5 -- Find average runtime of films with higher revenue than specified value
6 -- Sort all lists by revenue and write the results to a new file
7 -- Quit
Choice ==> 4

Enter title: Westworld

No such film exists.

Please choose one of the following options:
1 -- Find all films made by a specified director
2 -- Find the highest grossing film made in a specific year
3 -- Find all films made in a given year range in a specified genre
4 -- Search for a film by title
5 -- Find average runtime of films with higher revenue than specified value
6 -- Sort all lists by revenue and write the results to a new file
7 -- Quit
Choice ==> 4

Enter title: Hidden Figures

The film that meets your criteria is:
```

TITLE	GENRE	DIRECTOR	YEAR	RUNTIME	REVENUE(mil)
Hidden Figures	Biography;Drama;History	Theodore Melfi	2016	127	\$169.27

Find the average runtime of films with revenue higher than a specified value

```
>>> main()
Please enter a file name: movies.csv

Please choose one of the following options:
1 -- Find all films made by a specified director
2 -- Find the highest grossing film made in a specific year
3 -- Find all films made in a given year range in a specified genre
4 -- Search for a film by title
5 -- Find average runtime of films with higher revenue than specified value
6 -- Sort all lists by revenue and write the results to a new file
7 -- Quit
Choice ==> 5

Enter revenue limit (millions): $100
The average runtime for films with revenue higher than $ 100.00 million is 121.95 minutes.
```

```
>>> main()
Please enter a file name: movies.csv

Please choose one of the following options:
1 -- Find all films made by a specified director
2 -- Find the highest grossing film made in a specific year
3 -- Find all films made in a given year range in a specified genre
4 -- Search for a film by title
5 -- Find average runtime of films with higher revenue than specified value
6 -- Sort all lists by revenue and write the results to a new file
7 -- Quit
Choice ==> 5

Enter revenue limit (millions): $980
No films have revenue higher than $ 980.00 million.
```

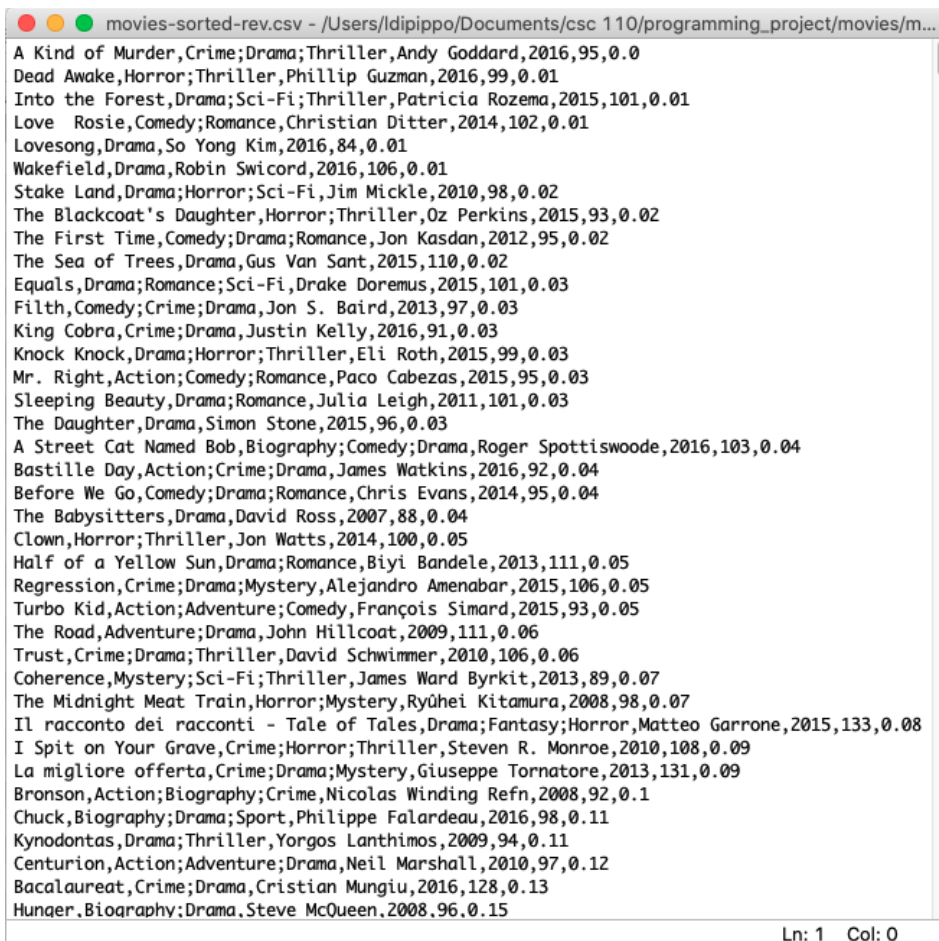
Sort all lists by revenue and write the results to a new file

```
>>> main()
Please enter a file name: movies.csv

Please choose one of the following options:
1 -- Find all films made by a specified director
2 -- Find the highest grossing film made in a specific year
3 -- Find all films made in a given year range in a specified genre
4 -- Search for a film by title
5 -- Find average runtime of films with higher revenue than specified value
6 -- Sort all lists by revenue and write the results to a new file
7 -- Quit
Choice ==> 6

Sorted data has been written to the file: movies-sorted-rev.csv.
```

The output file that is generated will be called: `movies-sorted-rev.csv` and it will look like this:



```
movies-sorted-rev.csv - /Users/ldipippo/Documents/csc 110/programming_project/movies/m...
A Kind of Murder,Crime;Drama;Thriller,Andy Goddard,2016,95,0.0
Dead Awake,Horror;Thriller,Phillip Guzman,2016,99,0.01
Into the Forest,Drama;Sci-Fi;Thriller,Patricia Rozema,2015,101,0.01
Love Rosie,Comedy;Romance,Christian Ditter,2014,102,0.01
Lovesong,Drama,So Yong Kim,2016,84,0.01
Wakefield,Drama,Robin Swicord,2016,106,0.01
Stake Land,Drama;Horror;Sci-Fi,Jim Mickle,2010,98,0.02
The Blackcoat's Daughter,Horror;Thriller,Oz Perkins,2015,93,0.02
The First Time,Comedy;Drama;Romance,Jon Kasdan,2012,95,0.02
The Sea of Trees,Drama,Gus Van Sant,2015,110,0.02
Equals,Drama;Romance;Sci-Fi,Drake Doremus,2015,101,0.03
Filth,Comedy;Crime;Drama,Jon S. Baird,2013,97,0.03
King Cobra,Crime;Drama,Justin Kelly,2016,91,0.03
Knock Knock,Drama;Horror;Thriller,Eli Roth,2015,99,0.03
Mr. Right,Action;Comedy;Romance,Paco Cabezas,2015,95,0.03
Sleeping Beauty,Drama;Romance,Julia Leigh,2011,101,0.03
The Daughter,Drama,Simon Stone,2015,96,0.03
A Street Cat Named Bob,Biography;Comedy;Drama,Roger Spottiswoode,2016,103,0.04
Bastille Day,Action;Crime;Drama,James Watkins,2016,92,0.04
Before We Go,Comedy;Drama;Romance,Chris Evans,2014,95,0.04
The Babysitters,Drama,David Ross,2007,88,0.04
Clown,Horror;Thriller,Jon Watts,2014,100,0.05
Half of a Yellow Sun,Drama;Romance,Biyi Bandele,2013,111,0.05
Regression,Crime;Drama;Mystery,Alejandro Amenabar,2015,106,0.05
Turbo Kid,Action;Adventure;Comedy,François Simard,2015,93,0.05
The Road,Adventure;Drama,John Hillcoat,2009,111,0.06
Trust,Crime;Drama;Thriller,David Schwimmer,2010,106,0.06
Coherence,Mystery;Sci-Fi;Thriller,James Ward Byrkit,2013,89,0.07
The Midnight Meat Train,Horror;Mystery,Ryûhei Kitamura,2008,98,0.07
Il racconto dei racconti - Tale of Tales,Drama;Fantasy;Horror,Matteo Garrone,2015,133,0.08
I Spit on Your Grave,Crime;Horror;Thriller,Steven R. Monroe,2010,108,0.09
La migliore offerta,Crime;Drama;Mystery,Giuseppe Tornatore,2013,131,0.09
Bronson,Action;Biography;Crime,Nicolas Winding Refn,2008,92,0.1
Chuck,Biography;Drama;Sport,Philippe Falardeau,2016,98,0.11
Kynodontas,Drama;Thriller,Yorgos Lanthimos,2009,94,0.11
Centurion,Action;Adventure;Drama,Neil Marshall,2010,97,0.12
Bacalaureat,Crime;Drama,Cristian Mungiu,2016,128,0.13
Hunger,Biography;Drama,Steve McQueen,2008,96,0.15
```

Ln: 1 Col: 0

Quit when the user chooses, and handle errors in user input

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

Please choose one of the following options:
1 -- Find all films made by a specified director
2 -- Find the highest grossing film made in a specific year
3 -- Find all films made in a given year range in a specified genre
4 -- Search for a film by title
5 -- Find average runtime of films with higher revenue than specified value
6 -- Sort all lists by revenue and write the results to a new file
7 -- Quit
Choice ==> a
Invalid entry - Try again
Choice ==> x
Invalid entry - Try again
Choice ==> 9
Choice must be between 1 and 7
Choice ==> 0
Choice must be between 1 and 7
Choice ==> 7

Good-bye
>>> |
```

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 should create a separate function that will print the results. So the functions that you write to implement each of the menu options will NOT print results. They will return the appropriate information so that you can print the results by calling the print results function. Be sure to design your print function so that it will be able to print any of the types of results that might be returned.
- 4) 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 director in directorList:
```

- 5) Here are a few hints about how to sort your data by revenue:
 - a) The parameters of the function should be the list of revenues. The function should return a list of indexes sorted by rearranging them the same way you will rearrange the revenues list. That is, suppose the revenues list looks like this:

```
[103.3,104.7,205.4,201.6,200.8,202.3]
```


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:

```
[0, 1, 4, 3, 5, 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:

```
Annie, 103.3
Beethoven, 104.7
Once, 200.8
Moneyball, 201.6
Showtime, 202.3
Carrie, 205.4
```

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

- 6) 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()
```

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

```
print(titles[i].ljust(45), genres[i].ljust(35),
      directors[i].ljust(24), str(years[i]).ljust(8),
      str(runtimes[i]).ljust(8), (" $" + str(revs[i])).rjust(12))
```

`rjust(45)` justifies the text to the right and allows 45 characters for it
`ljust(45)` 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.

What to Submit:

Please submit your code in a file called `projMovies.py` to Gradescope in the assignment called *Programming Project - Movies*.

NOTE: DO NOT use Gradescope to test and debug your code. You should be doing all of your testing and debugging in IDLE. Once you are confident that the code is correct, you can submit to Gradescope.