

CSC 110 – Programming Project - Spring 2023

Airline Flight Scheduling

Objective:

The objective of this project is to write a program in Python that will read in a data file with all flights from Providence to Orlando and display information requested by the user.

The Data:

The data file can be found here:

[flights.csv](#)

This .csv file contains the following information for flights from Providence to Orlando.

Airline

The name of the airline, which is represented as a string.

Flight Number

Each airline has a unique flight number assigned to a particular flight. Each airline may have more than one flight scheduled on the same day from Providence to Orlando.

Departure Time

The time of day that the flight is scheduled to depart from Providence. Time is represented in 24 hour clock time, so 14:00 represents 2:00PM.

Arrival Time

The time of day that the flight is scheduled to arrive in Orlando.

Price

The cost of the flight.

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 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 Specific Flight

This option asks the user to enter an airline and a flight number, and then displays for the user the information about that flight. If the user enters an airline or flight number that does not exist, the program should let the user know.

2 - Flights Shorter than a Specified Duration

This option will ask the user to specify a maximum flight duration and then display flight information for each of the flights whose duration is less than or equal to that maximum. Duration is computed by taking the difference between the arrival time and the departure time.

3 - Cheapest Flight by a Given Airline

This option finds the flight with the lowest price flown by a specified airline and displays the information for that flight. The user should be prompted to enter an airline and then the program should make sure the user chooses a correct option. You may not use the built-in `min` function or `sort` function.

4 - Flights Departing After a Specified Time

This option will provide the user with a list of flights that depart after a specified time. Your program should ensure that the time entered by the user is in a valid time format. That is, the user input should be in the HH:MM format where HH is between 00 and 24 and MM is between 00 and 60.

5 - Average Price of All Flights

This option finds the average price of all available flights. You may not use the built-in `sum` or `average` functions.

6 - Sort All Flights by Departure Time and Write to a New File

This option creates an output file called `time-sorted-flights.csv`, and writes to the file the flights sorted by departure time. 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 departure time. 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 an airline, you can use the following to check if the airline name is in the list of airlines in the data file:

```
if airlineName in airlineList:
```

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 if the user chooses an invalid option 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 also include a description of any algorithms that are used in the code.
- 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 show you what the results should look like for each of the options on the menu.

Check for Valid Input

```
>>> main()
Please enter a file name: flight.csv
Invalid file name try again ...
Please enter a file name: flights.csv

Please choose one of the following options:
1 -- Find flight information by airline and flight number
2 -- Find flights shorter than a specified duration
3 -- Find the cheapest flight by a given airline
4 -- Find flight departing after a specified time
5 -- Find the average price of all flights
6 -- Write a file with flights sorted by departure time
7 -- Quit
Choice ==> a
Entry must be a number
Choice ==> 9
Entry must be between 1 and 7
Choice ==> z
Entry must be a number
Choice ==> 11
Entry must be between 1 and 7
Choice ==> 7

Thank you for flying with us
>>> |
```

1 – Find Specific Flight

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

Please choose one of the following options:
1 -- Find flight information by airline and flight number
2 -- Find flights shorter than a specified duration
3 -- Find the cheapest flight by a given airline
4 -- Find flight departing after a specified time
5 -- Find the average price of all flights
6 -- Write a file with flights sorted by departure time
7 -- Quit
Choice ==> 1

Enter airline name: united
Invalid input -- try again
Enter airline name: United
Enter flight number: aaaa
Invalid input -- try again
Enter flight number: 4103

The flight that meets your criteria is:

AIRLINE  FLT#    DEPART  ARRIVE PRICE
United   4103     11:00   16:16 $ 321
```

2 - Flights Shorter than a Specified Duration

```
Please choose one of the following options:
1 -- Find flight information by airline and flight number
2 -- Find flights shorter than a specified duration
3 -- Find the cheapest flight by a given airline
4 -- Find flight departing after a specified time
5 -- Find the average price of all flights
6 -- Write a file with flights sorted by departure time
7 -- Quit
Choice ==> 2
```

```
Enter maximum duration (in minutes): a
Entry must be a number
Enter maximum duration (in minutes): 200
```

The flights that meet your criteria are:

AIRLINE	FLT#	DEPART	ARRIVE	PRICE
JetBlue	1075	17:00	20:06	\$ 280
JetBlue	475	7:30	10:42	\$ 340

```
Please choose one of the following options:
1 -- Find flight information by airline and flight number
2 -- Find flights shorter than a specified duration
3 -- Find the cheapest flight by a given airline
4 -- Find flight departing after a specified time
5 -- Find the average price of all flights
6 -- Write a file with flights sorted by departure time
7 -- Quit
Choice ==> 2
```

```
Enter maximum duration (in minutes): 60
```

No flights meet your criteria

3 - Cheapest Flight by a Given Airline

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

```
Please choose one of the following options:
1 -- Find flight information by airline and flight number
2 -- Find flights shorter than a specified duration
3 -- Find the cheapest flight by a given airline
4 -- Find flight departing after a specified time
5 -- Find the average price of all flights
6 -- Write a file with flights sorted by departure time
7 -- Quit
Choice ==> 3
```

```
Enter airline name: usair
Invalid input -- try again
Enter airline name: USAir
```

The flight that meets your criteria is:

AIRLINE	FLT#	DEPART	ARRIVE	PRICE
USAir	1269	6:15	10:57	\$ 210

4 - Flights Departing After a Specified Time

Please choose one of the following options:
1 -- Find flight information by airline and flight number
2 -- Find flights shorter than a specified duration
3 -- Find the cheapest flight by a given airline
4 -- Find flight departing after a specified time
5 -- Find the average price of all flights
6 -- Write a file with flights sorted by departure time
7 -- Quit
Choice ==> 4

Enter earliest departure time: a
Invalid time - Try again aa:aa
Invalid time - Try again 2:30
Invalid time - Try again 14:30

The flights that meet your criteria are:

AIRLINE	FLT#	DEPART	ARRIVE	PRICE
Delta	5138	16:20	22:10	\$ 212
JetBlue	1075	17:00	20:06	\$ 280
USAir	1865	16:56	21:34	\$ 300
USAir	3289	18:55	23:41	\$ 300
USAir	3863	15:35	20:54	\$ 302
USAir	3826	17:45	23:19	\$ 302
United	2869	16:55	21:33	\$ 406

Please choose one of the following options:
1 -- Find flight information by airline and flight number
2 -- Find flights shorter than a specified duration
3 -- Find the cheapest flight by a given airline
4 -- Find flight departing after a specified time
5 -- Find the average price of all flights
6 -- Write a file with flights sorted by departure time
7 -- Quit
Choice ==> 4

Enter earliest departure time: 22:00

No flights meet your criteria

5 - Average Price of All Flights

Please choose one of the following options:
1 -- Find flight information by airline and flight number
2 -- Find flights shorter than a specified duration
3 -- Find the cheapest flight by a given airline
4 -- Find flight departing after a specified time
5 -- Find the average price of all flights
6 -- Write a file with flights sorted by departure time
7 -- Quit
Choice ==> 5

The average price is \$ 308.41

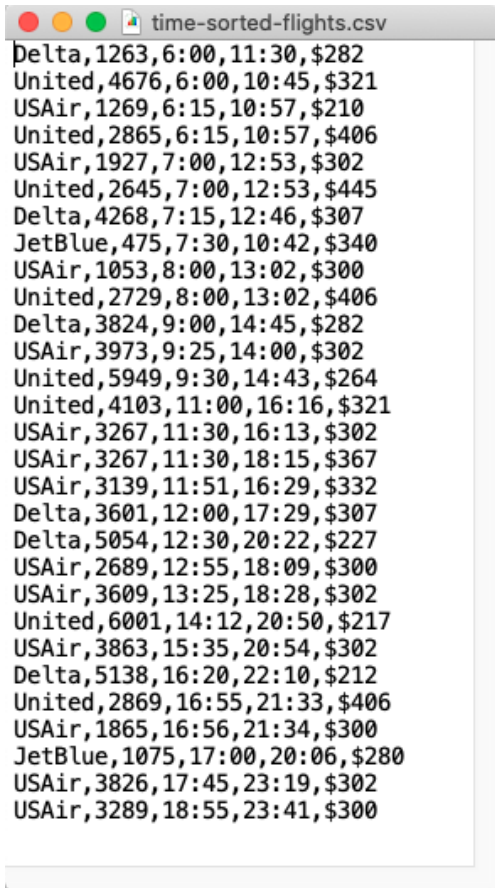
6 - Sort All Flights by Departure Time and Write to a New File

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

Please choose one of the following options:
1 -- Find flight information by airline and flight number
2 -- Find flights shorter than a specified duration
3 -- Find the cheapest flight by a given airline
4 -- Find flight departing after a specified time
5 -- Find the average price of all flights
6 -- Write a file with flights sorted by departure time
7 -- Quit
Choice ==> 6

Sorted data has been written to file: time-sorted-flights.csv
```

The output file should look like this:



The screenshot shows a text editor window titled 'time-sorted-flights.csv'. The window contains a list of flight records, each on a new line. Each record consists of the airline name, flight number, departure time, arrival time, and price, separated by commas. The records are sorted by departure time in ascending order. The data is as follows:

Airline	Flight Number	Departure Time	Arrival Time	Price
Delta	1263	6:00	11:30	\$282
United	4676	6:00	10:45	\$321
USAir	1269	6:15	10:57	\$210
United	2865	6:15	10:57	\$406
USAir	1927	7:00	12:53	\$302
United	2645	7:00	12:53	\$445
Delta	4268	7:15	12:46	\$307
JetBlue	475	7:30	10:42	\$340
USAir	1053	8:00	13:02	\$300
United	2729	8:00	13:02	\$406
Delta	3824	9:00	14:45	\$282
USAir	3973	9:25	14:00	\$302
United	5949	9:30	14:43	\$264
United	4103	11:00	16:16	\$321
USAir	3267	11:30	16:13	\$302
USAir	3267	11:30	18:15	\$367
USAir	3139	11:51	16:29	\$332
Delta	3601	12:00	17:29	\$307
Delta	5054	12:30	20:22	\$227
USAir	2689	12:55	18:09	\$300
USAir	3609	13:25	18:28	\$302
United	6001	14:12	20:50	\$217
USAir	3863	15:35	20:54	\$302
Delta	5138	16:20	22:10	\$212
United	2869	16:55	21:33	\$406
USAir	1865	16:56	21:34	\$300
JetBlue	1075	17:00	20:06	\$280
USAir	3826	17:45	23:19	\$302
USAir	3289	18:55	23:41	\$300

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 from either the main function or some other function that calls the computation functions. 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) Departure and arrival times are stored as strings. You will need to write a function to convert a time string like '10:34' to an integer so that you can do comparisons and sort based on these times. Since all of the times are assumed to be in the same day, you can compute a numeric time that represents the number of minutes since the beginning of the day. So '10:34' would be 10 hours and 34 minutes since the beginning of the day, which is 634 minutes since the beginning of the day. Your function should take the time string as a parameter and return the numerical time value.
- 5) Here are a few hints about how to sort your data by departure time:
 - a) The parameters of the function should be the list of departure times. The function should return a list of indexes sorted by rearranging them the same way you will rearrange the departures list. That is, suppose the departures list looks like this:

```
['10:15', '10:11', '10:10', '10:03', '10:16', '10:06']
```

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 departure times will be rearranged, so we would end up with:

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

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.

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, you can use the following syntax:

```
print(airlines[i].ljust(8), flnums[i].ljust(6),  
      departures[i].rjust(7), arrivals[i].rjust(7),  
      "$", str(prices[i]).rjust(3))
```

`rjust(7)` justifies the text to the right and allows 7 characters for it

`ljust(8)` 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 `projAirline.py` to Gradescope in the assignment called *Programming Project - Airline Scheduling*.

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.

Grading the Assignment:

See the [Grading Rubric for the Airline Programming Project](#) to see how the assignment will be graded.