Final Project

Learning Objective

Demonstrate knowledge of the Python programming concepts learned throughout the semester. **You are only allowed to use Python concepts taught this semester.**

Assignment Description

Write a program that allows the user to play multiple games with already established players. Read a CSV (comma-separated values) file with information about the players, play games, and update the players' scores.

You are provided with a CSV with data about players.

- Each row represents one line in a table, representing player information.
- The first row in the CSV file represents the header that contains the information of this data.

The menu options displayed to the user are determined by information in a text file named menu_options.txt. The data is separated by a colon. The first item is an uppercase letter, and the second item is a description of the option.

Text File
```
A: Display player information by ID
B: Display the lowest scoring player
C: Display the highest scoring player
D: Display top players
E: Find players
P: Play Game
Q: Quit
```

Create four Python files: **gameplay.py**, **helper.py**, **display.py**, and **main_*last*_*first*.py**.

In the helper.py file, create the following functions to read the CSV file, and perform tasks: **create_options_dict(), get_user_option(), find_players(), read_player_data(), play_game().**

In the display.py file, create the following functions to read the text file and interact with the user: **display_user_menu(), display_player_by_ID(), display_player(), display_smallest_value(), display_largest_value(), display_top_scores().**

In the gameplay.py file, you will write your own code and game inspired by a previous lab or assignment. More guidance on this will appear in this file below.

In the main_last_first.py file, create the **main()** function which calls the functions in the other files.

Requirements

Use Python concepts from this semester's course materials including slide decks, videos, assignments, and labs. They are all available on Brightspace. If you need help, go to the office hours of the instructor or any learning assistant (LA) or post on Piazza.

Do NOT import and use the CSV or other modules to process the CSV file.
Do NOT collaborate, share code, or exchange solutions with others.
Do NOT use methods, functions, ideas, that we have not covered in the class, class codes, or in class slides.
Do NOT use an AI program such as ChatGPT to help complete this project. This is the final project of our class, so show your own work to finish it.

Sample Output (user input shown in **green and underlined**)

```
Welcome to our gaming hub!

A ->  Display player information by ID
B ->  Display the lowest scoring player
C ->  Display the highest scoring player
D ->  Display top players
E ->  Find players
P ->  Play Game
Q ->  Quit
Option: Z
Option: 1
Option:   a
Enter Player ID: 1500
Not Accepted.

A ->  Display player information by ID
B ->  Display the lowest scoring player
C ->  Display the highest scoring player
D ->  Display top players
E ->  Find players
P ->  Play Game
Q ->  Quit
Option:   A
Enter Player ID: 10
William [#10]
      The game 1 score is 710
      The game 2 score is 730
      The game 3 score is 700
      Total score is 2140

A ->  Display player information by ID
B ->  Display the lowest scoring player
C ->  Display the highest scoring player
D ->  Display top players
E ->  Find players
P ->  Play Game
Q ->  Quit
Option: b
Select from this list: ['game1_score', 'game2_score', 'game3_score']
Enter a key: game1_score
Oliver [#8]
      The game 1 score is 470
      The game 2 score is 490
      The game 3 score is 480
```

```
     Total score is 1440

A ->  Display player information by ID
B ->  Display the lowest scoring player
C ->  Display the highest scoring player
D ->  Display top players
E ->  Find players
P ->  Play Game
Q ->  Quit
Option: c
Select from this list: ['game1_score', 'game2_score', 'game3_score']
Enter a key: hi
Enter a key: key
Enter a key: game2_score
Link [#92]
     The game 1 score is 880
     The game 2 score is 1200
     The game 3 score is 860
     Total score is 2940

A ->  Display player information by ID
B ->  Display the lowest scoring player
C ->  Display the highest scoring player
D ->  Display top players
E ->  Find players
P ->  Play Game
Q ->  Quit
Option: d
How many top scores (max is 100) do you want to display?
Enter a number: two
Enter a number: 1001
Enter a number: 3
1. 2940
2. 2920
3. 2920

A ->  Display player information by ID
B ->  Display the lowest scoring player
C ->  Display the highest scoring player
D ->  Display top players
E ->  Find players
P ->  Play Game
Q ->  Quit
Option: e
Find players based on the following attributes: ['name', 'hobby']
Enter a key:
```

```
Enter a key: total
Enter a key: hobby
Enter a search phrase:    GARDENING
2 player(s) contain(s) Gardening in key hobby
Michael [#2]
      The game 1 score is 510
      The game 2 score is 540
      The game 3 score is 560
      Total score is 1610
TheGamester [#73]
      The game 1 score is 980
      The game 2 score is 990
      The game 3 score is 960
      Total score is 2930

A ->  Display player information by ID
B ->  Display the lowest scoring player
C ->  Display the highest scoring player
D ->  Display top players
E ->  Find players
P ->  Play Game
Q ->  Quit
Option: e
Find players based on the following attributes: ['name', 'hobby']
Enter a key: name
Enter a search phrase: abcabc
No player contains Abcabc in key name

A ->  Display player information by ID
B ->  Display the lowest scoring player
C ->  Display the highest scoring player
D ->  Display top players
E ->  Find players
P ->  Play Game
Q ->  Quit
Option: q
```

SAMPLE OUTPUT 2 - GAME PLAY PART

```
Welcome to our gaming hub!

A ->  Display player information by ID
B ->  Display the lowest scoring player
C ->  Display the highest scoring player
D ->  Display top players
E ->  Find players
P ->  Play Game
```

```
Q ->  Quit
Option: P
Choose index of player 1: 700
Choose index of player 1: 7
Choose index of player 2: 100

Playing Game 1.
… The rest will print out the game of your own with modifications to an
existing lab or assignment. Assume player 1 wins.
Emma wins 10 points!

A ->  Display player information by ID
B ->  Display the lowest scoring player
C ->  Display the highest scoring player
D ->  Display top players
E ->  Find players
P ->  Play Game
Q ->  Quit
Option: Q
```

Steps

1. When in doubt, check the sample output. If you can interpret a step in multiple ways, you are free to choose your own approach. Your code and sample output can differ in slight prints and/or spacings; go with whichever is more comfortable for you. However, try to match the sample output with only small variations.
2. In PyCharm (Community Edition), open your existing ITP116 project.
3. Create a new directory called **project_*last_first*** where *last* is your last/family name and *first* is your preferred first name. Use all lowercase letters. You will have points deducted if this is not correct.
4. In the directory, you will create **multiple Python files**. At the top of **each file**, put comments in the following format and replace the name, email, section, and filename with your actual information:

```
# ITP 116, Spring 2025
# Final Project
# Name: First Last
# Email: username@usc.edu
# Filename: filename.py (update the name based on the file)
# Description: (you fill in)
```

5. Put the **csv** file in your project_*last_first* directory.
   - Download the file from Brightspace under the item for the Final Project.
   - Drag it onto your project_*last_first* directory in PyCharm.

- If needed, use the Refactor -> Rename tool in PyCharm to change the filename.
- **Do not open it in Excel or Google Sheets. It can change the encoding. If you did, delete the previous version and download it again from Brightspace.** If you encounter an error named UnicodeDecodeError, you can add an argument for the encoding parameter (encoding="UTF-8") when calling your functions.

6. Put the **menu_options.txt** file in your project_*last_first* directory.
    - Download the file from Brightspace by right-clicking on it and selecting the "Save Link As..." or "Save As..." option.
    - If needed, change the filename to menu_options.txt.
    - Navigate to your project_*last_first* folder.
    - Click the Save button.

7. Create a Python file entitled **helper.py**. At the top of this file, add the comment block that must include your name. Import necessary files and modules. In this file, define the following functions:

   Define the **read_player_data()** function.

- Parameter: csv_file is the name of the CSV file to read and set a default value of "players.csv"
- Return: a Dataframe object, holding players information.
- Open the csv_file using Pandas.
- Return this DataFrame.
- This function will be called once in the main() function in the main_*last_first*.py file.

   Define the **create_options_dict()** function.

- Parameter: textFileStr is the name of the text file to read and has a default value of "menu_options.txt"
- Return: a dictionary where the keys are options (strings) and the values are descriptions (strings)
- Open the textFileStr in read mode.
- Note that the data in this text file is separated by a colon.
- Read the file line by line. Put this information in a dictionary variable. Close file afterwards.
- They keys are the uppercase letters, and the values are the descriptions of the options.
- You will call this function in the main function in the main_last_first.py file.

   Define the **get_user_option()** function.

- Parameter: optionsDict is a dictionary holding the options that the user can choose
- Return: a string that is a valid option entered by the user
- Get input from the user using the following prompt:

```
Option:
```

- Use the appropriate loop to ask the user for input until they enter an acceptable input. The acceptable inputs are the keys of the optionsDict dictionary.
- Also allow the user to enter extra spaces before and after their input or enter lower- or upper-case letters.
- You will call this function in the main function in the main_last_first.py file.
- Here is an example with user input in green and underlined text ("Q" is one of the keys):

```
Option: AAA
Option:  bee
Option: 1
Option: x
Option:   q
```

Define the **play_game()** function.

- Parameter: data is a Dataframe object, holding players information.
- Return value: None
- Ask for player 1 and player 2 ids as input (check whether the value is between 1 and 120 (inclusive)). Index player "id" column, convert it to a list after using .values, and then create a loop until the user enters acceptable player ids within that list.
- Player 1 will be you, the player. Player 2 is assumed to be the opponent. Therefore, if we lose, player 2 is assumed to win.
- Call your functions from gameplay.py to play the game. At the end, your functions should return either "win" or "lose" back to the play_game() function.
- Print a victory message for the winner. For example, Emma is the player with ID 7, therefore, when the player with ID 7 wins a game, we print the message below:

```
Emma wins 10 points!
```

- If the returned value is "lose", this means player 1 lost and player 2 won. Update the "previous score" of player 1 to 0 and the "previous score" of player 2 to 10 in the data object. Add 10 to game1_score for the winning person.
- Note, you can use .loc to locate the correct id user, and the attribute you want to change such as "previous_score" to update the scores.
- This function will be called in the main_last_first.py file.

8. Create a Python file titled **gameplay.py**. At the top of this file, include a comment block that contains your name. You will improve one of our previous labs or assignments and integrate it with the rest of this project using this python file. You will "gamify" the lab/assignment of your choice (starting from lab 4 or assignment 4), so that it fits into this project. You may directly copy

and paste your previous lab or assignment solution and add new functions. Be sure to import any necessary files and modules. This file is worth 40 points for your project.

1. You are expected to enhance one of your previous labs or assignments and integrate it into the rest of your project using this Python file. The goal is to "gamify" the chosen lab or assignment so that it fits within this project. If it is already a game (zombies, or card games, etc.) you can improve them with more ideas.
2. The game you create should have either "win" or "lose" outcomes. User input is required. The game must vary with each playthrough. You can choose whether to create a one-player (1P) or two-player (2P) game.
3. The option "P" in the main menu should call play_game() function in the helper.py. After asking for player IDs in that function, you should call the functions in your gameplay.py. The structure of gameplay.py is up to you.
4. You will be graded based on the number of unique methods and ideas you incorporate into this Python file.
5. Your file should include at least **3 new functions**, each demonstrating significant effort and applying coding techniques available in our slides. Try to use different data types, functions, methods, and so on. Be sure to include **clear comments within each function** to help graders understand your approach.
6. At the end of the game, return "win" or "lose" to the play_game() function so that you can print the winner and update the player scores.

   Send a private Piazza message (or join office hours) if you would like to go over your ideas with me.

   There are some things that you are NOT allowed to do:

   • Do NOT use a project that is an assessment (such as homework, assignment, lab, or final project) from another course, or previous semesters. This is an academic violation.

   • Do NOT share ideas/code with other classmates. There is no collaboration with other people except course staff.

9. Create a Python file entitled **display.py**. At the top of this file, add the comment block that must include your name. Import necessary files and modules. In this file, define the following functions:

   Define the **display_user_menu()** function.

- Parameter: optionsDict is a dictionary holding the options that the user can choose
- Return value: None
- Display the options to the user using the optionsDict parameter.
- The keys contain letters (strings) for the options, while the values contain short descriptions (strings) for each letter.
- Print the key (letter) followed by " **->** " and then the value (description).
- You will call this function in the main function in the main_last_first.py file.

Create the **display_player()** function.

- Parameter: data is a pandas Series object holding a single player information.
- Return value: None
- Display the information for the player in the following format (the words surrounded by double quotes are going to be replaced by the player information):

```
“name” [#“id”]
   The game 1 score is “game1_score”
   The game 2 score is “game2_score”
   The game 3 score is “game3_Score”
   Total score is “game1_score” + “game2_score” + “game3_score”
```

- To achieve this, use Pandas to access rows/columns/cells. Warning: Make sure your data type is Series! If **data** argument is a dataframe this part will be more challenging than necessary.
- Here is an example for the player Link, id 92:

```
Link [#92]
   The game 1 score is 880
   The game 2 score is 1200
   The game 3 score is 860
   Total score is 2940
```

Define the **display_smallest_value()** function.

- Parameter: data is a Dataframe object, holding players information.
- Return value: None
- Create a list variable to hold strings game1_score, game2_score, game3_score. Print the following using this variable:

```
Select from this list:
['game1_score', 'game2_score', 'game3_score']
```

- Ask user input until they enter an input in the list.
- Allow the user to enter upper or lowercase letters and enter extra space.
- Here is an example (user input shown in green and underlined):

```
Enter a key:  year
Enter a key: 2023
Enter a key:  game1_SCORE
```

- Return the row with the smallest value in the selected key. You can use loops, or pandas .min() function and conditioning to achieve this. There are multiple ways to solve this.
- Call the display_player() function to print the player information with the lowest game score. **Note, display_player function accepts a Series object. Make sure you end up with a Series object of a single player information instead of a DataFrame object. For example, if you**

have a DataFrame object with a single row, you can turn this to a Series using the following syntax:

player = DataFrame.iloc[0]

This hint applies to display_largest_value() function as well.

- Call this function in the main_last_first.py file.
- Here is the expected output when the key entered is "game1_score" because this user has the lowest game1 score in the dataset:

```
Oliver [#8]
   The game 1 score is 470
   The game 2 score is 490
   The game 3 score is 480
   Total score is 1440
```

Define the **display_largest_value()** function.

- Parameter: data is a Dataframe object, holding players information.
- Return value: None
- Similar to the display_smallest_value() function. You should focus on the largest value instead of the smallest.
- Call this function in the main_last_first.py file.

Create the **display_player_by_ID()** function.

- Parameter: data is a Dataframe object, holding players information.
- Return value: None
- Display all the information of the user for a selected id.
- Ask user input for the player id. If the input is not an integer, or not between 1 and 120 (inclusive), print “Not accepted.” There is no loop, and we leave the function in case of not accepted input.
- If the user id exists, condition the DataFrame to return the row with the correct player information.
- Call the display_player() function to display the information of the player with the correct id.
- **Note, display_player function accepts a Series object. Make sure you end up with a Series object of a single player information instead of a DataFrame object. For example, if you have a DataFrame object with a single row, you can turn this to a Series using the following syntax:**

  player = DataFrame.iloc[0]
- This function will be called in the main_last_first.py file.

Define the **display_top_scores()** function.

- Parameter: data is a Dataframe object, holding players information.
- Return value: None
- Display the following message to the user:

```
How many top scores (max is 100) do you want to display?
```

Get an integer from the user (repeat until they enter an integer between 1 to 100 inclusive). Here is an example (user input shown in green):

```
Enter a number: ten
Enter a number: 1001
Enter a number:    2
```

You can add all the game scores first using pandas indexing and convert the column to a list after using .values. Then, sort the list, and print the top scores using a for loop:

```
1. 2940
2. 2920
```

This function will be called in the main_last_first.py file.

Define the **find_players()** function.

- Parameter: data is a Dataframe object, holding players information.
- Return value: None
- Create a list to hold `'name'` and `'hobby'` strings. Print below using this list:

```
Find players based on the following attributes:
['name', 'hobby']
```

- Have the user enter text for a search phrase. Allow the user to enter upper or lower case text and allow the user to enter extra spaces before and after their input. The key should be `'name'` or `'hobby'`
- Here is an example showing the user input in `green and underlined` text:

```
Enter a key: id
Enter a key:
Enter a key:  Hobby
Enter a search phrase:   GARDENING
```

- Use .contains method to condition your Pandas object considering the key selected. Name this filtered object filtered_data.
- If there are no players that meet the criteria, then print a message (be careful with the upper and lower cases). Try using the len() function. Here is an example when the search text is "Cookie":

```
No player contains Cookie in key hobby
```

- Otherwise, using a loop, display the information of the players using the display_player() function.
- **Note, display_player function accepts a Series object. Make sure you end up with a Series object of a single player information instead of a DataFrame object. For example, if you have a DataFrame object with a single row, you can turn this to a Series using the following syntax:**

player = DataFrame.iloc[0]

- Note that names and hobbies have the first letter uppercased in the csv file, think about which method would be useful here.
- This function will be called in the main_last_first.py file.

- Here is an example showing the two players displayed when the search phrase is "GARDENING":

```
Found 2 player(s) that contain(s) Gardening in key hobby
Michael [#2]
   The game 1 score is 510
   The game 2 score is 540
   The game 3 score is 560
    Total score is 1610

TheGamester [#73]
   The game 1 score is 980
   The game 2 score is 990
   The game 3 score is 960
    Total score is 2930
```

Create a Python file entitled **main_*last*_*first*.py** where *last* is your last/family name and *first* is your first name. Use lowercase letters. Make sure to add the proper comment block at the top of the file. Import the **helper** and **display** files.

Define the **main()** function.

- Parameter: None
- Return value: None
- Display the following message to the user:

```
Welcome to our gaming hub!
```

- Call the helper.**read_player_data()** function to read the CSV file and create a Pandas object. If your CSV file is named "players.csv", then you do not need to have an argument. Make sure to capture the returned Pandas object in a variable.
- Call the helper.**create_options_dict**() function to create the dictionary for the options and capture the return value in a variable.
- Use a loop to display the options, get the user's option, and respond to the user's option until the user enters "Q" or "q". To display the options, use the display.display_user_menu() function.
- To get input from the user, use the helper.get_user_option() function.
- Use branching to call the appropriate function depending on the user's input: A, B, C, ..., P.
- Don't forget to call the **main()** function.

10. Be sure to comment your code. This means that there should be comments at the top of the files you created as well as throughout your code. Put a comment block at the top of each Python file. **Put a comment block before each function stating the parameters, return values, and what that function does.** Points will be deducted for not having comments.
11. Follow coding conventions. You should use camelCase or snake_case for variable names. You are welcome to create any variables that you need. Variables should NOT start with an uppercase letter.
12. Test the program. Look at the Sample Output below. Projects that do not run are subject to a 20% penalty.
13. Prepare your submission:
    - Find the **project_*last_first*** folder on your computer and compress it. This cannot be done within PyCharm. This folder MUST have the following files: **gameplay.py, helper.py, main_*last_first*.py,** and **display.py**.
    - On Windows, use **File Explorer** to select the folder. Right-click and select the Send to -> Compressed (zipped) folder option. This will create a zip file.
    - On Mac OS, use **Finder** to select the folder. Right-click and select the Compress "*FolderName*" option. This will create a zip file.
14. Upload the zip file to Brightspace. Make sure you click Submit.

Grading

- This assignment is worth 100 points. With extra credit, you may earn up to 110 points.
- Make sure that the program runs. Points will be taken off if the graders have to edit the source code to test your program.
- Make sure to submit your assignment correctly as described above. Points will be taken off for improper submission.
- Do not use `outside-the-class` material. More information is available in the guidelines document on Brightspace.
- Overall, do not use ideas that we did not cover in the class, feel free to check the slides and videos to make sure you will not lose points because of this.
- This content is protected and may not be shared, uploaded, or distributed. Posting this project on sites such as Chegg and Course Hero is an academic integrity violation. Do NOT use an AI tool such as ChatGPT.

Category	Item	Points
helper.py	read_player_data()	3
	play_game()	8
	create_options_dict()	4
	get_user_option()	5
	find_players()	5
gameplay.py	various functions	40
display.py	display_top_scores()	4
	display_user_menu()	5
	display_player_by_ID()	5
	display_player()	5
	display_smallest_value()	5
	display_largest_value()	5
main_*last*_*first*.py	main()	6
	Total (including 10 extra credit)	**110**

Extra Credit - 10 points

If your project has all the functions detailed above, it does not use any outside-the-class material, and follows the instructions above as closely as possible, you will receive +10 to your overall project grade.