Project 1: Build a Soccer League Project

Sections of this Guide:

- **How to approach this project** includes detailed guidance to help you think about how to organize your code, project and files.
- **How to succeed at this project** lists the grading requirements for the project, with hints, links to course videos to refresh your memory and helpful resources.

How to Approach This project

After reading through the project instructions we see that there are three main tasks that need to be accomplished. Although there are other requirements for the project, these are the main three:

1. Read the supplied CSV file.

2. Iterate through the players and assign them equally to three teams; the Dragons, Sharks, and Raptors. This step is where the heart of the code lies. A good strategy for solving this is to think about how we would do this in a real life situation.

   How would we take the 18 players, assign them to three different teams and make sure each team gets the same number of experienced players? One way would be to sort them into two groups, experienced and inexperienced. Then take those two groups and split them between the three teams. In terms of useful data structures, both lists and dictionaries can be useful.

   Perhaps iterating over the data in the CSV file and creating lists of experienced and inexperienced players? Each iteration through the CSV file then could allow the list(s) to grow, based on a player's experience. Storing the team names in a dictionary to iterate over might prove to be a useful technique as well when assigning teams.

3. Generate an output text file with a "league roster".

How to succeed at this project

Here are the things you need to do pass this project. Make sure you complete them ***before*** you turn in your project.

- ❑ **Script execution**
  It is a best practice to have your python files run only when they are explicitly ran. If we

have called functions or operations in our script and our script is imported by another file, they will run. To prevent this we can include an `if __name__ == '__main__':` statement and put function calls, print statements, etc. *inside* that `if` block. For example if we do:

```
if __name__ == '__main__':

print("Hello World")
```

The print statement here will only be run if the file is executed directly and **not** if it is imported separately.

For this requirement, make sure the code in `league_builder.py` that needs to be executed to run the program is included in an `if __name__ == '__main__':` block.

- ❏ *Related resource:* Python Documentation for `__main__`

❏ Output a text file

After you process the players and split them into the necessary teams, Dragons, Sharks, and Raptors, you'll need to export the information into a `teams.txt` file.

- ❏ *Related video:* Writing to Files
- ❏ *Related workshop:* Python File I/O Workshops

❏ Team Rosters

You'll want to read in the files from `soccer_players.csv` and assign the players to the appropriate teams. Inside the `teams.txt` file you'll need to have the three required teams with the players evenly divided. For each player it should provide their name, their parent/guardian name, and if they are experienced.

- ❏ *Related video:* Around and Around (from Python Basics)

❏ Comments

Another best practice is to add comments to your code. This not only helps others know what your intent for a piece of code is, but can be a big help when you revisit a project in the future.

- ❏ *Related video:* Shopping List Introduction (at around the 1:25 mark)