SI 206 Final Project

Ryan Baxter & Lexi Fogel

Repo link: https://github.com/lexifogel/si206-final

Goals

We unfortunately made our final project plan before thoroughly examining the APIs we planned to use. Therefore, when it came time to access the APIs, we realized we wouldn't be able to pull the data we had originally hoped to. Thus, we deviated from this plan completely when it came time to do the project.

Our original goals were:

- Use API's that look at data for transportation within different U.S. cities. (Los Angeles, Honolulu, Philly)
- Gather information of transportation services offered in each of the three cities
- Create bar charts to show the distribution between the individual states of offered transportation

Ultimately, we ended up shifting over to a sports theme. We utilized an NBA API (balldontlie) and a soccer API (football API from Rapid API) to pull data on individual players across the two sports to compare the relative heights of players, by their primary position played.

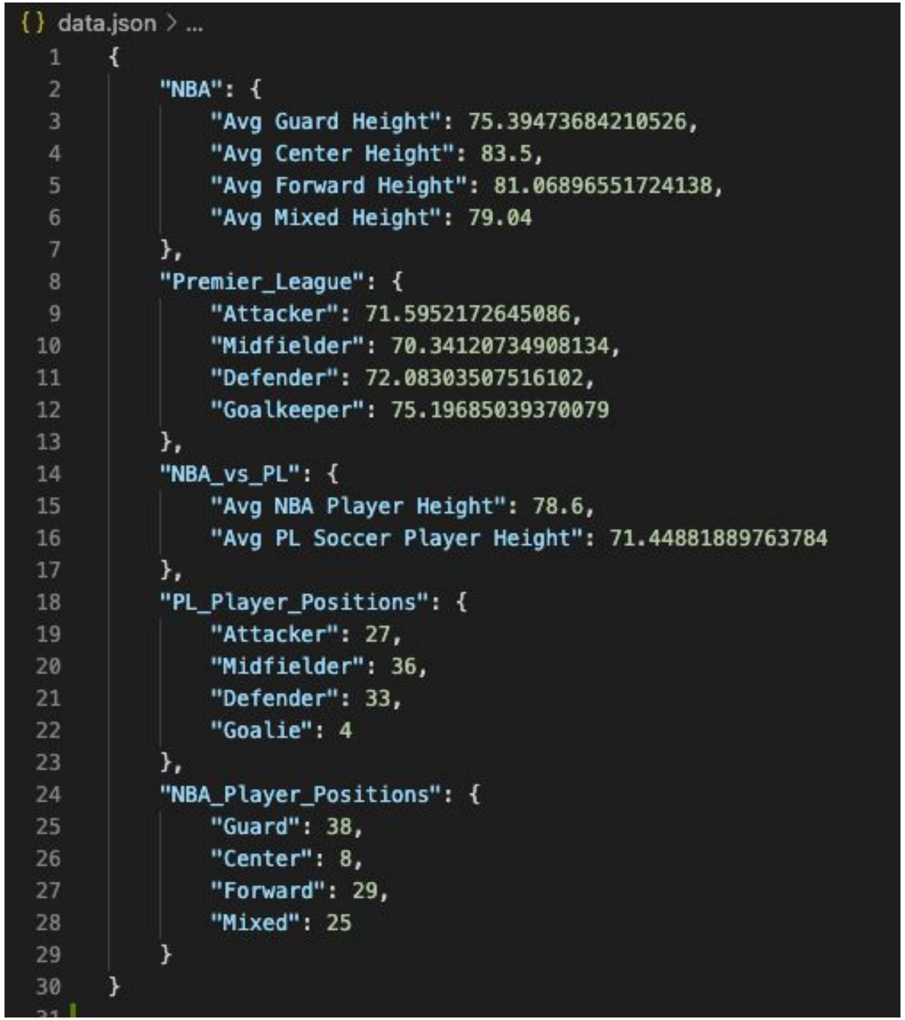
Thus our new goals became:

- Compare the average heights by position for basketball players in the NBA
- Compare the average heights by position for soccer in the Premier League
- Compare the heights of players across the two sports (in the leagues we examined)

Problems

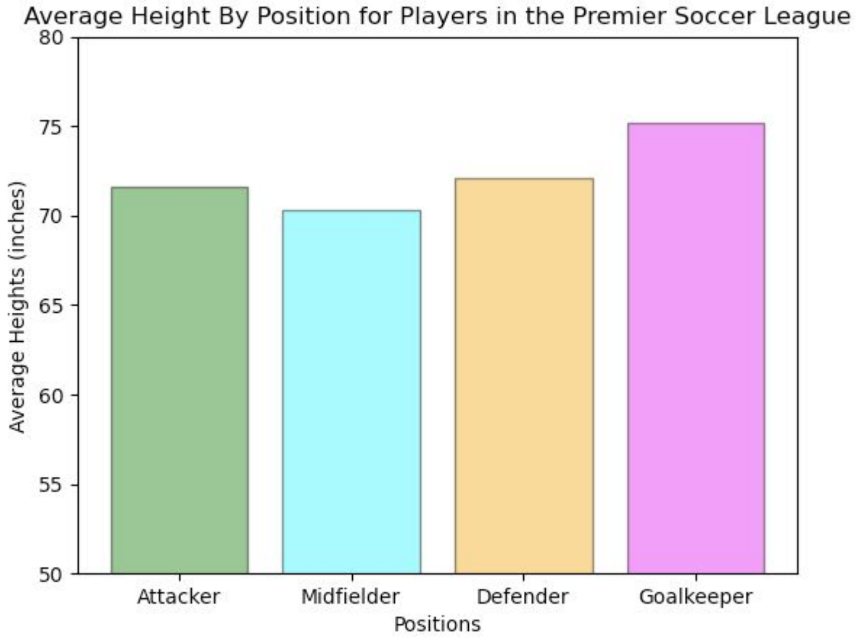
- Problem 1:
    - The first problem we had related to the Soccer API. Originally, we planned to grab the player height data and compare it to the NBA height data. However, we failed to notice that the data for the Soccer heights was in centimeters rather than inches, and thus they weren't directly comparable
        - We solved this problem by adding a conversion equation before sorting the height data which converts the Soccer player height from centimeters to inches so that the units for the two sports were the same
- Problem 2:
    - The second issue we faced was with the NBA API. Our original implementation of NBA_get.py failed to gather the same information (i.e. the same players) in the ID table and the PlayerHeight table because only the createPositionTable() failed to filter out players who did not have a listed height in the API. This resulted in having different players between the two individual tables, and thus being unable to join them and do calculations for each player
        - We solved this problem by adding a check before pushing the data into the database that would "continue" if the listed player did not have a listed height in the API data, ensuring that the two tables included the same people
- Problem 3:
    - The last problem we faced had to do with both the Soccer and NBA APIs. We had issues getting the data and occasionally received an error because we failed to wait long enough for the API connection reset / did not remember the API rate limit that was stated on the API documentation
        - We solved this by printing out a response to the terminal that tells the user to wait 30 seconds before running the code again

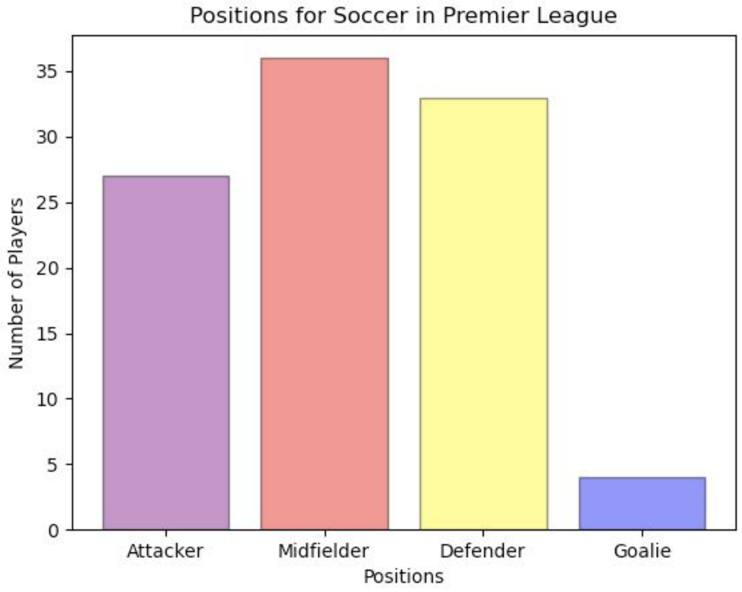
Calculations file (data.json)


data.json > ...

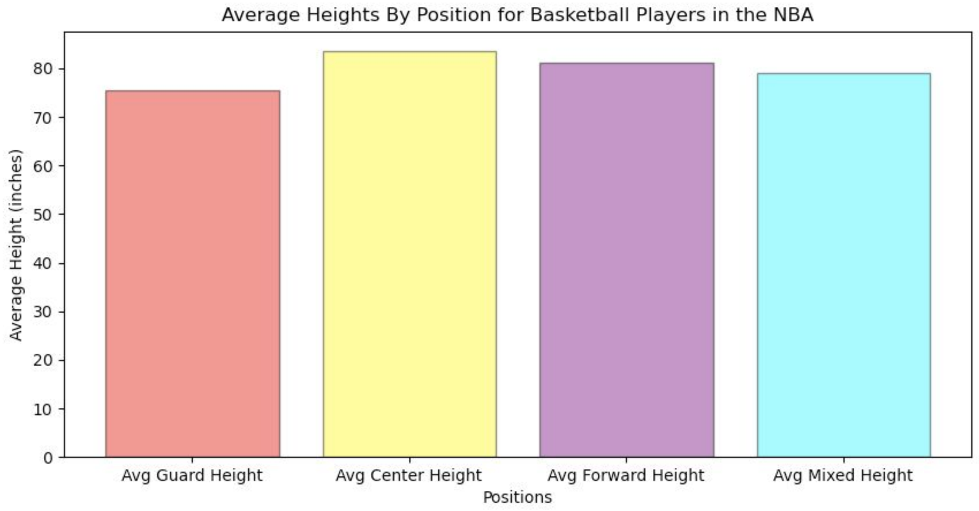
```json
{
    "NBA": {
        "Avg Guard Height": 75.39473684210526,
        "Avg Center Height": 83.5,
        "Avg Forward Height": 81.06896551724138,
        "Avg Mixed Height": 79.04
    },
    "Premier_League": {
        "Attacker": 71.5952172645086,
        "Midfielder": 70.34120734908134,
        "Defender": 72.08303507516102,
        "Goalkeeper": 75.19685039370079
    },
    "NBA_vs_PL": {
        "Avg NBA Player Height": 78.6,
        "Avg PL Soccer Player Height": 71.44881889763784
    },
    "PL_Player_Positions": {
        "Attacker": 27,
        "Midfielder": 36,
        "Defender": 33,
        "Goalie": 4
    },
    "NBA_Player_Positions": {
        "Guard": 38,
        "Center": 8,
        "Forward": 29,
        "Mixed": 25
    }
}
```

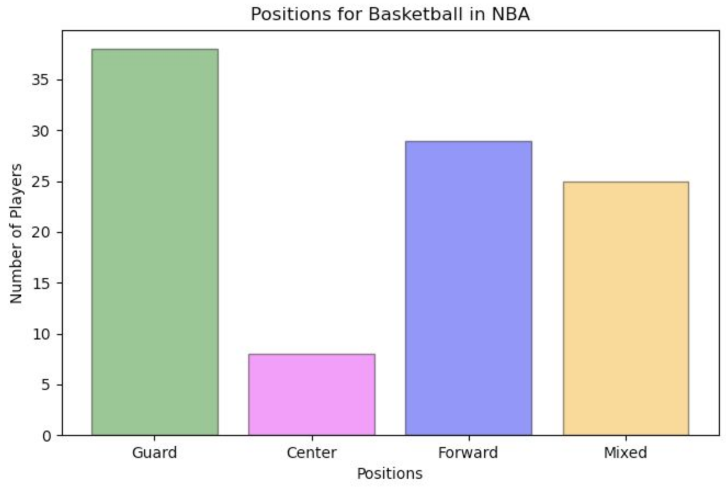
Visualizations -- Premier League



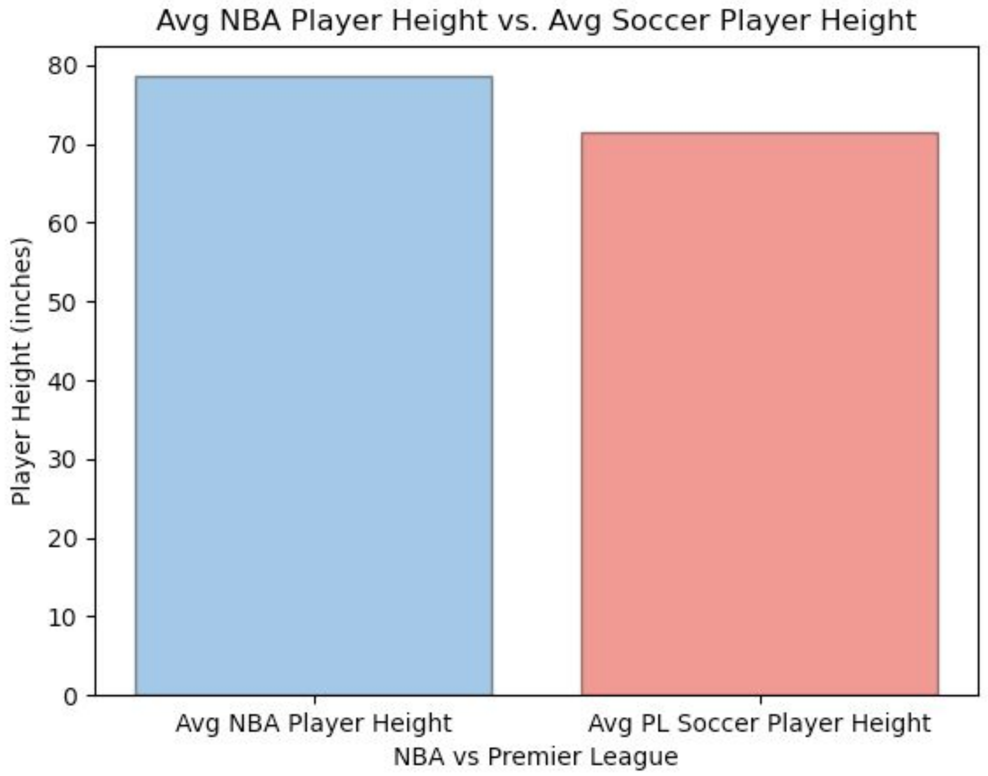


Visualizations -- NBA





Visualizations -- Both!

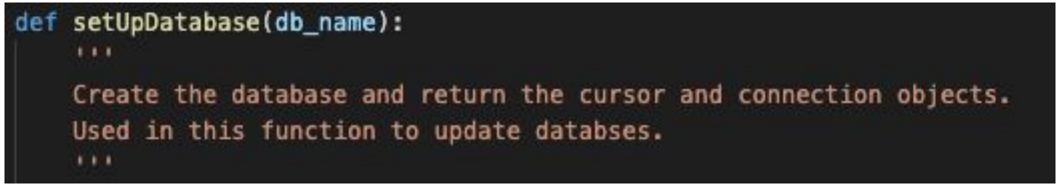


Instructions for running

1) Soccer
   a) Open soccer_get.py
   b) Run this file
   c) Wait 30 seconds
   d) Repeat steps 1b/c three more times, until there are 100 total players in the database
   e) Open soccer_read.py
   f) Run this file to execute the calculations and display the visualizations
2) Basketball
   a) Open NBA_get.py
   b) Run this file
   c) Wait 30 seconds
   d) Repeat steps 2b/c three more times, until there are 100 total players in the database
   e) Open NBA_read.py
   f) Run this file to execute the calculations and display the visualizations
3) Combined
   a) Open NBA_vs_PL.py
   b) Run this file to execute the calculations and display the visualization

Function Documentation

- Each file begins with the setUpDatabase function

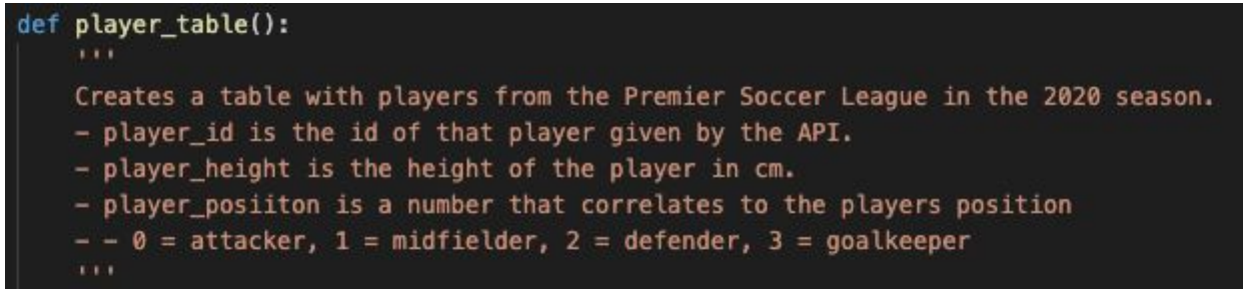


```python
def setUpDatabase(db_name):
    '''
    Create the database and return the cursor and connection objects.
    Used in this function to update databses.
    '''
```

    - Input: The database name
    - Output: The cursor and connection to the database

Function Documentation cont

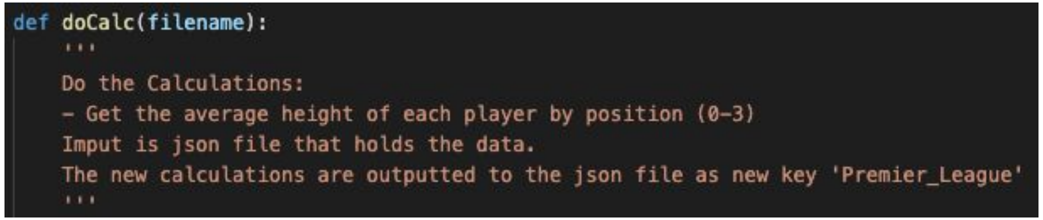
- soccer_get also has the following functions:
    - playertable()



```
def player_table():
    '''
    Creates a table with players from the Premier Soccer League in the 2020 season.
    - player_id is the id of that player given by the API.
    - player_height is the height of the player in cm.
    - player_posiiton is a number that correlates to the players position
    - - 0 = attacker, 1 = midfielder, 2 = defender, 3 = goalkeeper
    '''
```

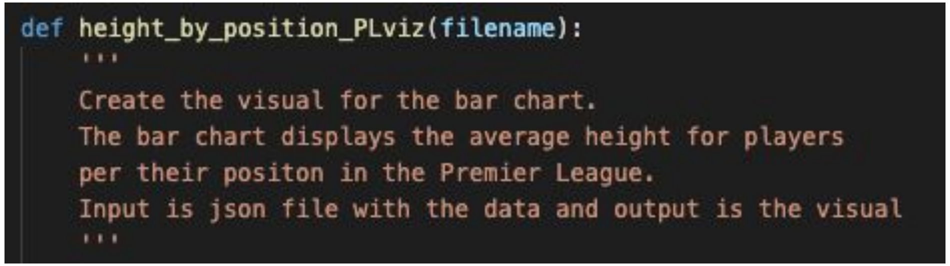
Function Documentation cont

- soccer_read also has the following functions:
  - doCalc(filename)



```
def doCalc(filename):
    '''
    Do the Calculations:
    - Get the average height of each player by position (0-3)
    Imput is json file that holds the data.
    The new calculations are outputted to the json file as new key 'Premier_League'
    '''
```

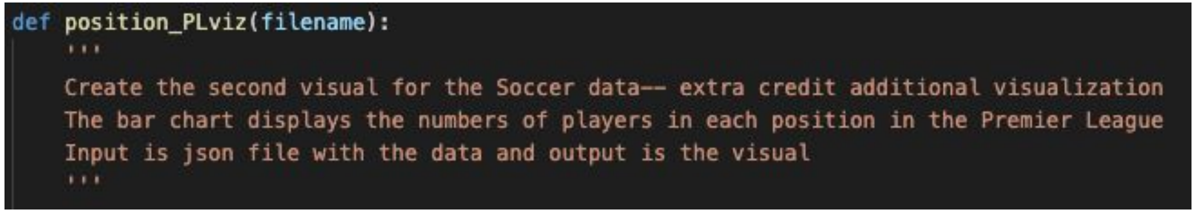
  - height_by_position_PLviz(filename)



```
def height_by_position_PLviz(filename):
    '''
    Create the visual for the bar chart.
    The bar chart displays the average height for players
    per their positon in the Premier League.
    Input is json file with the data and output is the visual
    '''
```

Function Documentation cont

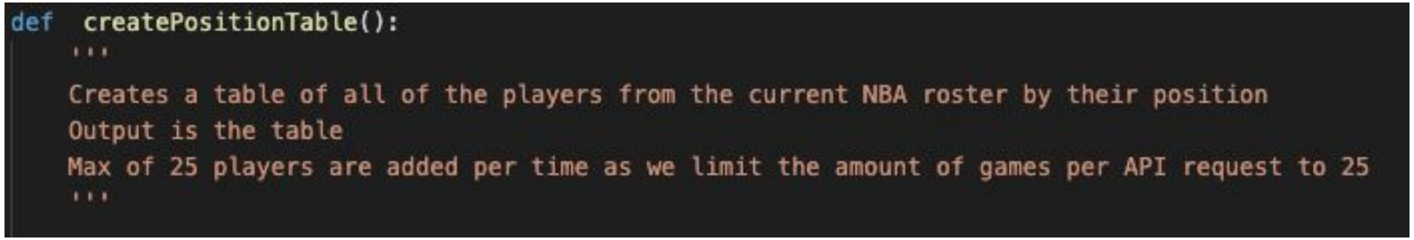
- soccer_read cont:
  - position_PLviz(filename) -- extra credit



```
def position_PLviz(filename):
    '''
    Create the second visual for the Soccer data-- extra credit additional visualization
    The bar chart displays the numbers of players in each position in the Premier League
    Input is json file with the data and output is the visual
    '''
```

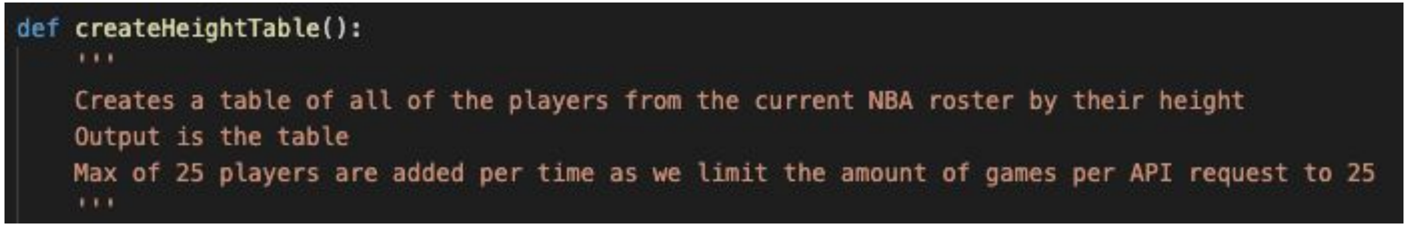
Function Documentation cont

- NBA_get also has the following functions:
  - createPositionTable()



```
def  createPositionTable():
    '''
    Creates a table of all of the players from the current NBA roster by their position
    Output is the table
    Max of 25 players are added per time as we limit the amount of games per API request to 25
    '''
```

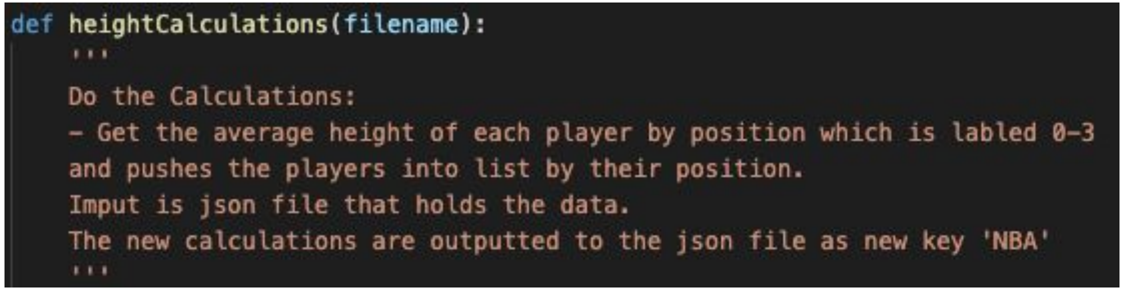
  - createHeightTable()



```
def createHeightTable():
    '''
    Creates a table of all of the players from the current NBA roster by their height
    Output is the table
    Max of 25 players are added per time as we limit the amount of games per API request to 25
    '''
```

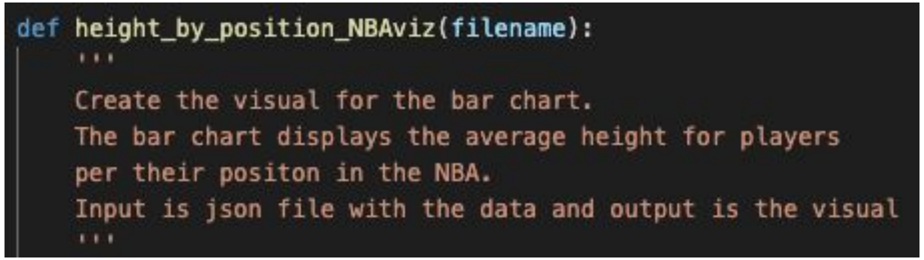
Function Documentation cont

- NBA_read also has the following functions:
  - heightCalculations()



```python
def heightCalculations(filename):
    '''
    Do the Calculations:
    - Get the average height of each player by position which is labled 0-3
    and pushes the players into list by their position.
    Imput is json file that holds the data.
    The new calculations are outputted to the json file as new key 'NBA'
    '''
```

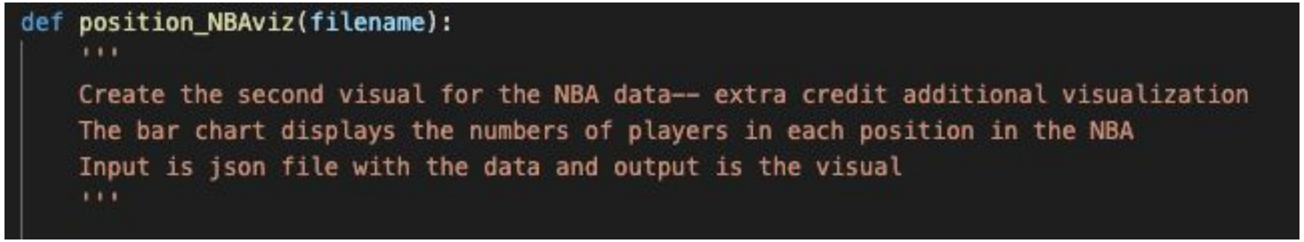
  - height_by_position_NBAviz(filename)



```python
def height_by_position_NBAviz(filename):
    '''
    Create the visual for the bar chart.
    The bar chart displays the average height for players
    per their positon in the NBA.
    Input is json file with the data and output is the visual
    '''
```

Function Documentation cont

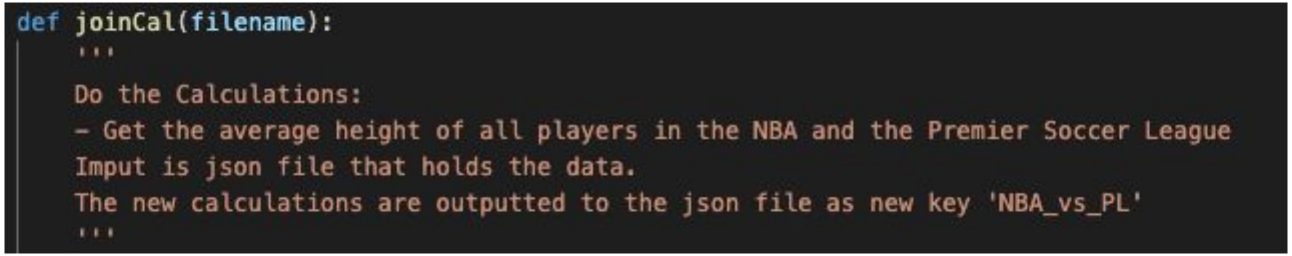
- NBA_read cont:
    - position_NBAviz(filename) -- extra credit



```
def position_NBAviz(filename):
    '''
    Create the second visual for the NBA data-- extra credit additional visualization
    The bar chart displays the numbers of players in each position in the NBA
    Input is json file with the data and output is the visual
    '''
```

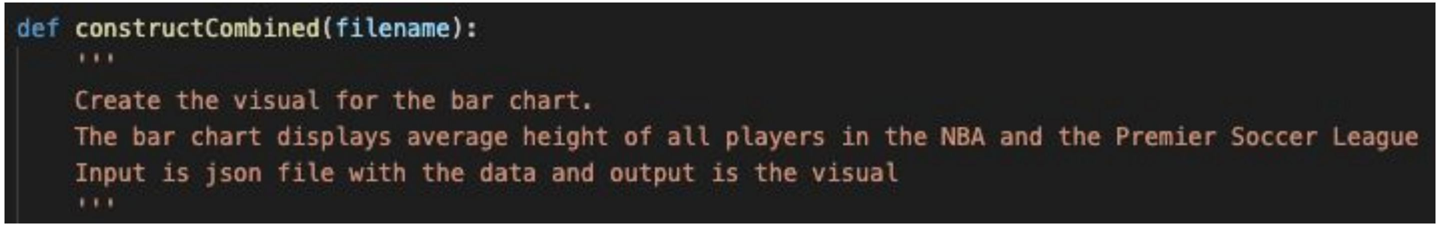
Function Documentation cont

- NBA_vs_PL also has the following functions:
    - joinCal(filename)



```
def joinCal(filename):
    '''
    Do the Calculations:
    - Get the average height of all players in the NBA and the Premier Soccer League
    Imput is json file that holds the data.
    The new calculations are outputted to the json file as new key 'NBA_vs_PL'
    '''
```

- constructCombined(filename)



```
def constructCombined(filename):
    '''
    Create the visual for the bar chart.
    The bar chart displays average height of all players in the NBA and the Premier Soccer League
    Input is json file with the data and output is the visual
    '''
```

Resources

Date	Issue Description	Location of Resource	Result
11/29	Understanding why the APIs were sometime returning faulty requests.	https://www.balldontlie.io/#introduction https://www.api-football.com/documentation-v3#operation/get-teams-statistics	Solved: Had to know the limit we were allowed to request in a given amount of time
12/6	Understanding to change the amount of data in each page in the querystring for the params requirement.	https://www.balldontlie.io/#introduction	Solved: Had to change the querystring = {“per page”: “100”, “page”:str(i)}
12/10	Accessing and using the database created by one file, from another file	https://stackoverflow.com/questions/51065457/how-to-import-a-sql-file-to-python	Solved
12/11	Understanding how to create matplotlib better and clean up our graphs to understand better	https://stackoverflow.com/questions/8575062/how-to-show-matplotlib-plots	Solved