* Natty Champs * UVA CS 5010 * Final Project *

Vasudha Manikandan | Latifa Hasan | John Zhang
Summer Chambers | Prabhjot Singh

Introduction

The National Basketball Association (NBA) has consistently been one of the most watched sports around the world. It featured cultural icons of past generations such as Michael Jordan and has inspired generations of fans and young athletes to become avid followers of the sport. The influence and cultural relevance of the NBA continues to this day, but many argue that the magic that existed during the past generations of basketball has been lost. The debate that surround this divisive topic range in variety but can synthesized to these few:

- Who's the best player in NBA history in terms of statistics (not including wins, championships, and MVPS?
- How has basketball changed in terms of various statistics that define the sport?

Although there are many players that we can consider when trying to answer the first question, for the purposes of this project we have decided to limit it to a comparison between all time greats belonging to different generations of basketball: Michael Jordan and LeBron James. Furthermore, in order to answer this question we will be looking at how both of these players differed in terms of comparison to each other and the entire league for a given year.

For the second question, we determined that there were multiple angles to view a change in the nba, however limited it to the following subsets of questions:

- Is basketball as physical?
- Are players generally better?

In order to analyze the first points we examined statistical trends in variables such as 3 point and 2 point field goals as well as overall personal fouls. Hypothetically, if there was an decrease in physicality, an indicator for this could be an increase of 3 point field goals and personal fouls as well as a decrease in 2 point field goals. For the second question, we visualized points per a game and how that has changed over the years. Therefore we define an improvement in players by whether or not there is an increase in our trend of points per a game.

As summarized above, the overall goal of this project is to investigate the characteristics that make up both "old school" and "current" NBA and elaborate on their differences. By doing so we hope to better understand the statistics and trends that revolve around this debate.

The Data
Our data comes from basketball-reference.com, a website that collects data about NBA players and their statistics. The fact that this website collects data on each individual player, allows us to be able to create functions to extract a specific one we would like to know more about. To obtain the data from the website we used web-scraping techniques such as findAll to obtain the column header and getText to get the text into a list. Then we looped through the list of text and attached it to the correct column header. We refined the data a little bit and removed any missing data points.

The data set itself consists of all NBA players from the years 1980 to 2020 and provides a variety of information. The two qualitative variables were the position of the player and the team the player was on during that year. The rest of the variables are quantitative including age, minutes played, points per game, steals per game, etc. All these pieces of data are significant in giving useful information about a selected player. We then can turn this information into knowledge and gain some insight on how the player(s) have done over the years in some of the quantitative categories and how they possibly compare to the rest of the league. All this knowledge and information can help teams choose better players, focus on improving certain statistics of a player, and when march madness rolls around, determine what teams have a higher probability of winning.

Experimental Design
As stated above, this data was web-scraped from the website basketball-reference.com. The web-scraping used the beautifulsoup module to get the information off the website. To organize the information into a more malleable form, the Pandas module was used to put the data into a dataframe. Finally, any missing data was dropped to clean up the dataframe.

After organizing the data, general summary statistics were conducted on the data to figure out some information. Over the years, we found the average number of points per player per game to be 8.06. The average player played for 19.8 minutes. The average age of a player was 26.75. These general statistics help to learn about the league and how it is doing overall.

Now, to get more into the specifics of each individual player, a function called PlayerLookup was created. The purpose of this function is for novice users to learn more about a specific player in a specific year. The user can input a player’s name and year and the output of the function will be a row of their information such as their age, team, position, points, etc.

Now if the user would like to know how this player is in comparison to the rest of the league based on player age and points per game, the user can use our second function called PlayerComp. PlayerComp takes the input from PlayerLookup and places them in an empty dataframe. The second row in the dataframe is the average player in the league for that year. Using this constructed data frame, the output of this function is a bar graph to visualize the difference between the chosen player and the average player in the league.

The third function, yearAvg, takes in three parameters: a year, a statistic category and the top number of players you want to take the average for. This will output a single number that is the average for that statistic for that year's top number of players. This is a step towards a more in depth analysis of the data.

The fourth function created expands more on yearAvg. It is mostly directed towards those who want to delve deeper into the statistics of the NBA. YearlyComparisonTrend takes in four parameters: a start year, an end year, the statistic the user would like to look up and a top number of players. In the function, there is a loop that will go through the range of years set and then for each year the statistic established in the parameter is attached to the according year and placed into a dataframe. After the loop has completed, a line graph is printed to showcase the trend of the statistic over the years.

Each of these functions builds on the other to provide different levels of users knowledge about the NBA league and individual players. We also wanted to present this knowledge in different ways, whether that be through creating a dataframe, a singular output or visualizing on graphs.

Beyond Original Specifications
We wanted to make our project generalizable in such a way that they would be used to examine a variety of different things very easily. Therefore we implemented functions that either asked for user input or functions that were easily adaptable.

Web-Scraped Data
As stated above, this data was web-scraped from the website basketball-reference.com. But within our web-scrape is also the ability to easily change the range of years of the data. This functionality allows for a csv of your preference to be made within the matter of a couple minutes and ready to further investigate using the other functions that were created.

PlayerLookup
Do you need the correct statistical evidence to support why your favorite player is the best in the league? Well this is exactly what this tool is for. Using the

PlayerLookup function enables anyone to search pertinent information within a matter of seconds.

YearlyComparisonTrend
This function allows you to visualize any given statistic and how it has changed over any range of years. This can be done for virtually any numerical variable in our dataset and at the end will give us a line graph. Examples of this can be found in our result section where we evaluate different trends over the years 1980-2020.

Results

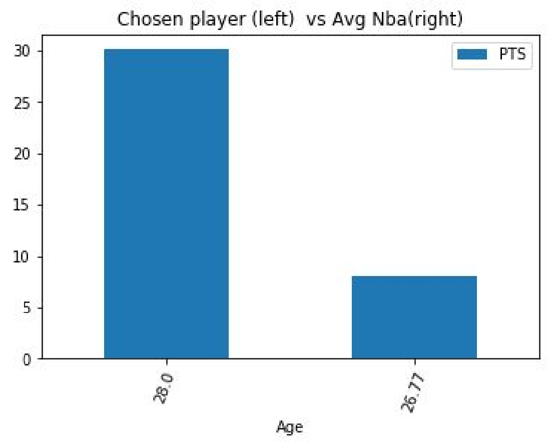
- Who's the best player in NBA history in terms of statistics (not including wins, championships, and MVPS?
    - Jordan (1984 - 2003) vs Lebron (2003-2020)

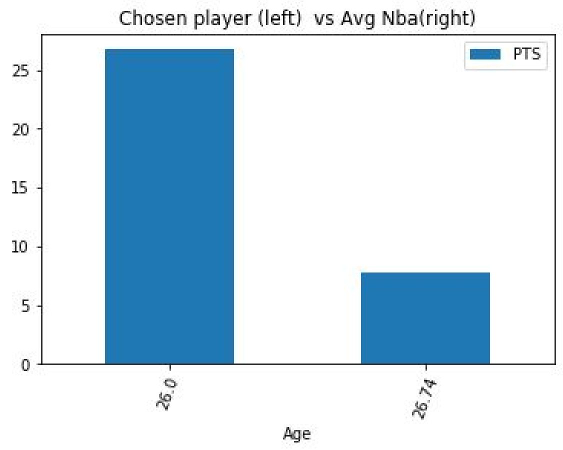
Using the function "return_player", we were able to retrieve the stats for each player over their career as a Pandas dataframe. Then, we used the np.mean() aggregate function to return each player's average stats over their career. By comparing the two dataframes, we were able to evaluate which player had the mean advantage over the other for each statistic. Ignoring unhelpful statistics like age or year and negative statistics like personal fouls or turnovers, we found that Michael Jordan had the advantage in 8 meaningful stats, while LeBron James had the advantage for 9 meaningful stats. Put simply, these players are relatively evenly matched. We could investigate this question further by looking into which stats each player had advantages for:

Advantage Jordan: FG, 2P, FT, FT%, ORB, STL, BLK, PTS
Advantage James: MP, FG%, 3P, 3P%, 2P%, eFG%, DRB, TRB, AST

Since Michael Jordan's and LeBron James' NBA careers didn't quite overlap, it is difficult to interpret each of these statistics in the context of the league each player actually played in (the NBA has changed over time--as we'll discuss a bit later). We can begin to tackle this issue by looking at a year in the middle of Michael Jordan's career (1992) and in LeBron James' career (2011) and compare their average total points to the total points per game of the NBA's average-aged player in that year:

Michael Jordan (1992) LeBron James (2011)

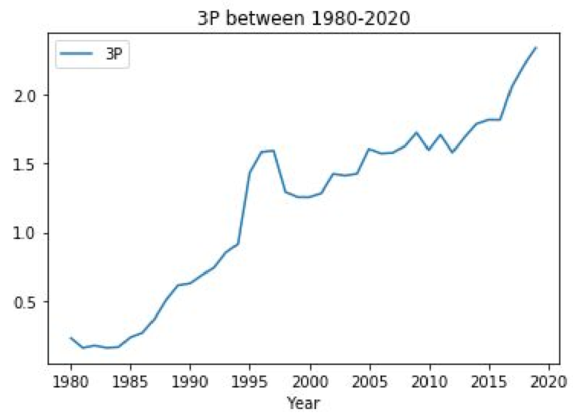


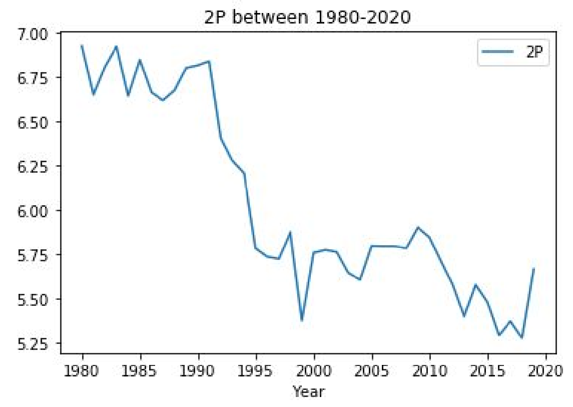


These plots visually depict how both Michael Jordan and LeBron James excelled in comparison to the average-aged NBA player.

- How has basketball changed in terms of various statistics that define the sport?
    - Are 3-pointers or 2-pointers more/less common now?

We can look at the number of 3-point field goals over time as well as the number of 2-point field goals over time.

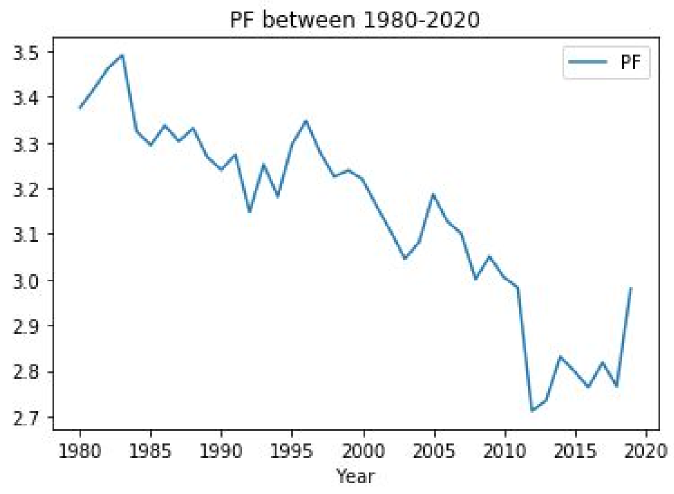




The trend we see from the YearlyComparisonTrend function is that 3 point field goals have increased over the years 1980-2020, while 2 point field goals have decreased.

- Is it as physical?

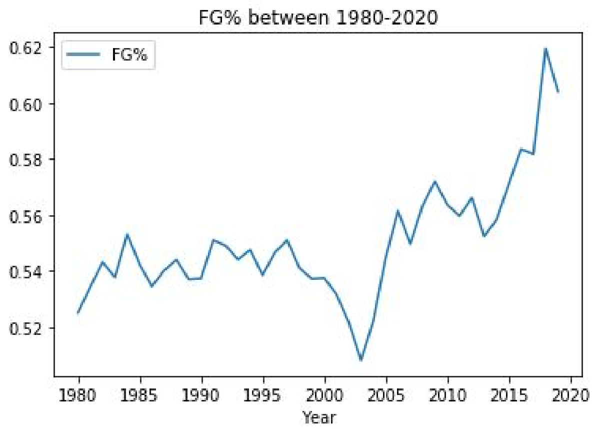
We can also see that personal fouls have decreased over the years.

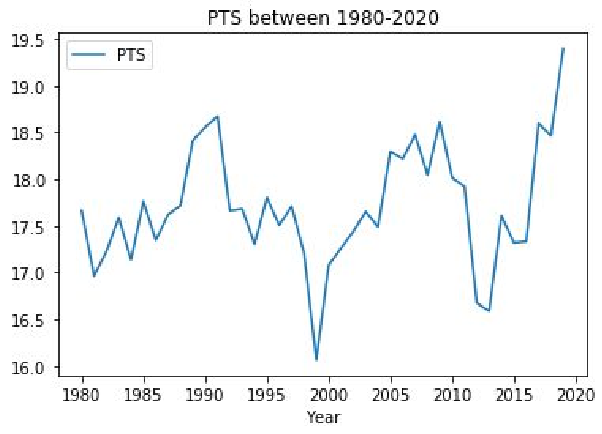


These findings suggest that basketball has become less physical since 1980.

- Are players generally better?

We see that overall, players have improved in field goal percentage as well as in total points scored.





These findings suggest that players are better now (shooting with more accuracy and scoring more) than they were in 1980.

**While these graphs show pretty dramatic differences, one should pay particular attention to the y-axis values to understand the actual magnitude of changes we're observing.

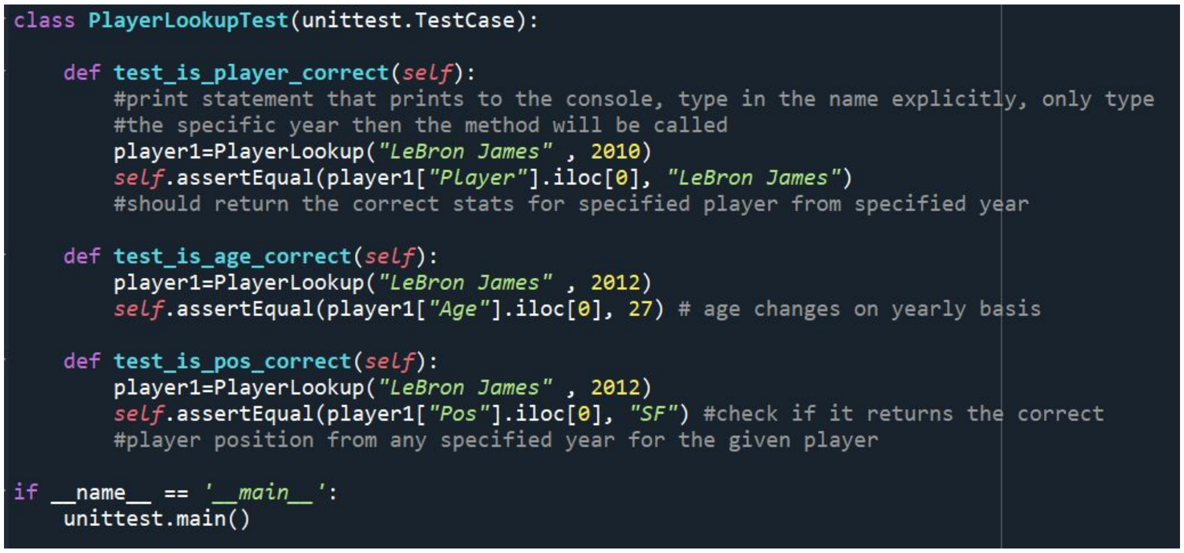
Testing

In order to make sure that components of our project were functioning properly we performed unit tests on our various functions.

PlayerLookup(playername, year)
> The PlayerLookup function allows user input to learn more about specific players. However, since it takes user input, it can be challenging to test the function. Hence, we created an alternate PlayerLookup function that requires two parameters, player name and year, to return the relevant statistics of the player from the year specified in the parameter. Once these are passed, the function should return the player statistics from the given player name and the year. The unit tests developed for this function work to ensure that the correct player statistics are returned.

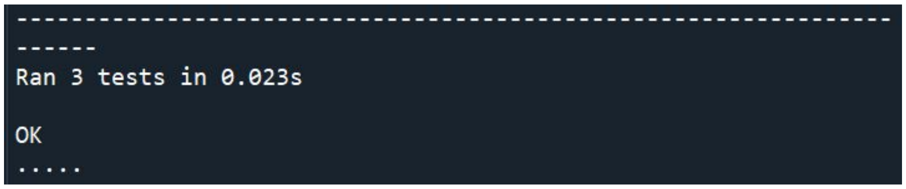
The unit tests for the PlayerLookup() function tested to check the different uses for the method PlayerLookup.

1. We set up the first test to check whether the function returns the correct statistics for the specified player and year. It should not return statistics for any other player/year. We used a self.assertEqual statement to test whether the values entered using the PlayerLookup function would return the correct statistics for the specified player and year.
2. Since age changes on a yearly basis, the function should also be able to reflect the change. The second test checks whether the age changes with a change in the year attribute.
3. The third test checks whether the correct player position is returned when the player name and the year attributes are specified.



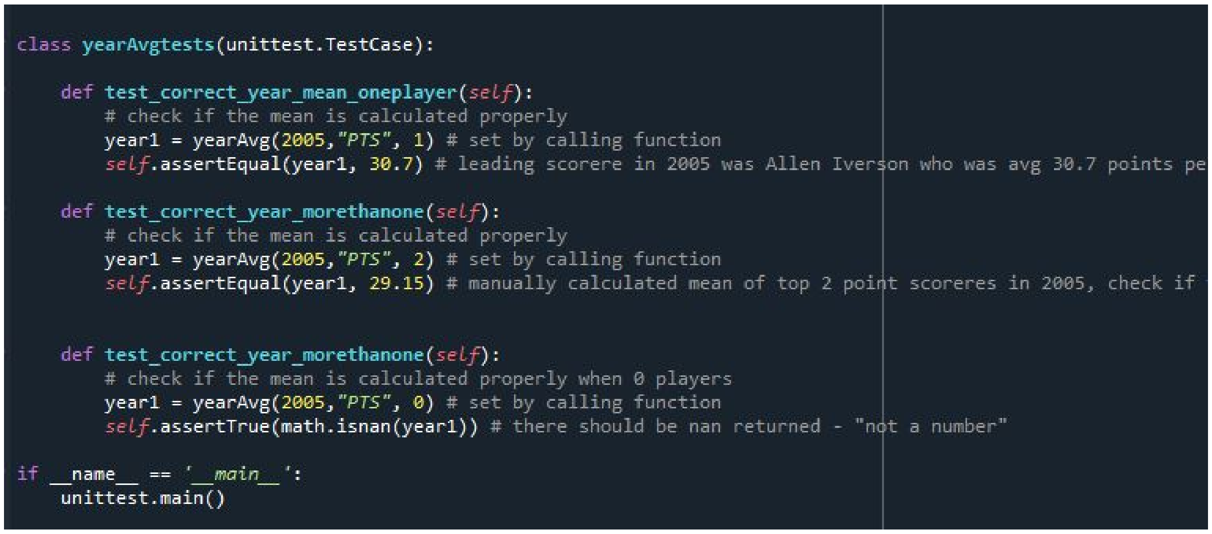
```
class PlayerLookupTest(unittest.TestCase):

    def test_is_player_correct(self):
        #print statement that prints to the console, type in the name explicitly, only type
        #the specific year then the method will be called
        player1=PlayerLookup("LeBron James" , 2010)
        self.assertEqual(player1["Player"].iloc[0], "LeBron James")
        #should return the correct stats for specified player from specified year

    def test_is_age_correct(self):
        player1=PlayerLookup("LeBron James" , 2012)
        self.assertEqual(player1["Age"].iloc[0], 27) # age changes on yearly basis

    def test_is_pos_correct(self):
        player1=PlayerLookup("LeBron James" , 2012)
        self.assertEqual(player1["Pos"].iloc[0], "SF") #check if it returns the correct
        #player position from any specified year for the given player

if __name__ == '__main__':
    unittest.main()
```

The unit tests for each of the test cases above run and return the following output:



```
------------------------------------------------------------
------
Ran 3 tests in 0.023s

OK
.....
```

yearAvg(year, stat_cat, topnum)
The year average function requires 3 parameters which include year of interest, statistical category of interest, and how many of the top players you want included in the analysis. After passing in these, the function should return the average of any given stat from any given year of any given number of top players. Therefore the unit tests developed in this portion focused on making sure the correct mean was returned.

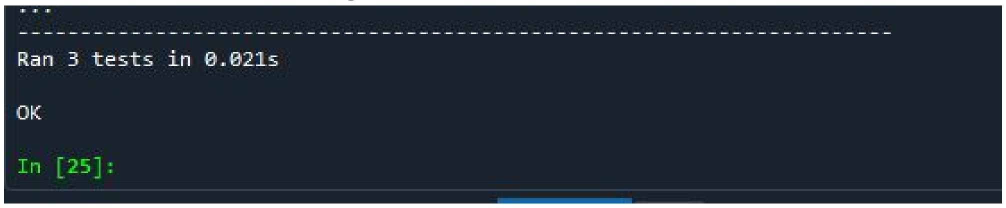


```
class yearAvgtests(unittest.TestCase):

    def test_correct_year_mean_oneplayer(self):
        # check if the mean is calculated properly
        year1 = yearAvg(2005,"PTS", 1) # set by calling function
        self.assertEqual(year1, 30.7) # leading scorere in 2005 was Allen Iverson who was avg 30.7 points pe

    def test_correct_year_morethanone(self):
        # check if the mean is calculated properly
        year1 = yearAvg(2005,"PTS", 2) # set by calling function
        self.assertEqual(year1, 29.15) # manually calculated mean of top 2 point scoreres in 2005, check if


    def test_correct_year_morethanone(self):
        # check if the mean is calculated properly when 0 players
        year1 = yearAvg(2005,"PTS", 0) # set by calling function
        self.assertTrue(math.isnan(year1)) # there should be nan returned - "not a number"

if __name__ == '__main__':
    unittest.main()
```

The main tests revolve around 3 different potential uses for our method yearAvg.

1. When we want to see the stats of only the top player of the league that year, we should be able to pass in 1 for the topum attribute. It should not do anything to the value and only one sample should be obtained. Therefore, a self.assertEqual statement was used to compare the value obtained through the yearAvg function and the value we found by looking up the top scorer of the 2005 season's stats.
2. Our function also should be able to return a mean value if more than 1 players are passed through our topnum attribute. Therefore we now used the top 2 players in the league to calculate our average for points for the 2005 season.
3. The last test is mainly to test for the nan object type that should be returned if 0 is passed through our topnum. Nan stands for "not any number" and may be tested for by using a math method called math.isnan() placing in our value that we would like to check inside the parentheses.

The results after running the tests:



```
...
----------------------------------------------------------------------
Ran 3 tests in 0.021s

OK

In [25]:
```

The idea is that we do not need to test all the numerical statistical columns because if it works for one of them it should work for them all. To add to this, we also did not need to check for the correct data frame as if that was incorrect, then the mean value returned would also most likely be incorrect. Making logical thresholds like this makes unit testing a lot less cumbersome and does not take away from the confidence of the proper functioning of our functions.

Conclusions

We found that while the NBA has changed since 1980, Michael Jordan and LeBron James are relatively evenly matched in terms of number of advantages in overall statistics in the context of their time and in comparison with the average player of their time. We also found that today's basketball is more 3-point-oriented and less 2-point oriented than in the past. Players are less physical, accruing fewer average personal fouls now than in the past. These findings help us understand some of the differences between old-school and modern basketball.

One of the things that we wanted to make sure existed in this project was a way to make any of the tools utilized generalizable to any situation a user might want to test. By doing so anyone seeking to gain quick insight into a player's history or yearly trends could do so with relative ease. One example use case of this could be a NBA team preparing to face off with an all star on another team. They could ideally use the playerlook functionality to gain all the knowledge that they need - and that too with the option of visuals to accompany the results. To add to this, with the emerging popularity of fantasy basketball, this tool could also be used for the general purposes of creating the best team.

Having said that, the functions and tools exemplified in this program can be dramatically improved. One way would be to web scrape or add more information related to specific game logs and winning records for each of the players. That way a more thorough assessment could be made regarding the statistics that make a winning team. In addition to this, all the data obtained for this project revolved around the regular season. However, players may perform differently in the playoffs, which could be drastically different from the types of trends we see in the regular season. Alongside these improvements, more functions could be added that maps multiple player information and yearly trends. This might be a more encapsulating picture of the statistics that surround a year or a player.

References

https://www.basketball-reference.com/leagues/NBA_2020_per_game.html
https://pythonprogramming.net/introduction-scraping-parsing-beautiful-soup-tutorial/
https://pythontic.com/pandas/dataframe-plotting/bar%20chart

Appendix - Stats Legend

Rk -- Rank
Pos -- Position
Age -- Player's age on February 1 of the season
Tm -- Team
G -- Games
GS -- Games Started
MP -- Minutes Played Per Game
FG -- Field Goals Per Game
FGA -- Field Goal Attempts Per Game
FG% -- Field Goal Percentage
3P -- 3-Point Field Goals Per Game
3PA -- 3-Point Field Goal Attempts Per Game
3P% -- 3-Point Field Goal Percentage
2P -- 2-Point Field Goals Per Game
2PA -- 2-Point Field Goal Attempts Per Game
2P% -- 2-Point Field Goal Percentage
eFG% -- Effective Field Goal Percentage
^This statistic adjusts for the fact that a 3-point field goal is worth one more point than a 2-point field goal.
FT -- Free Throws Per Game
FTA -- Free Throw Attempts Per Game
FT% -- Free Throw Percentage
ORB -- Offensive Rebounds Per Game

DRB -- Defensive Rebounds Per Game

TRB -- Total Rebounds Per Game

AST -- Assists Per Game

STL -- Steals Per Game

BLK -- Blocks Per Game

TOV -- Turnovers Per Game

PF -- Personal Fouls Per Game

PTS -- Points Per Game