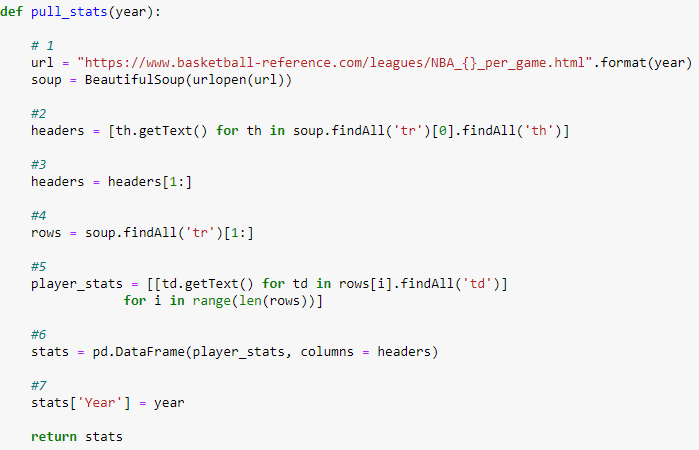
**Scraper for DSC 680 Project 1**

**Part 1: Function to scrape statistics**



1: The input taken by this function is a year that we want the data from. In this section we create the url. Basketball-reference has a consistent url where the only piece that changes in the year which makes it easy to adjust. The url is then opened with Beautiful Soup

2: If you go to the basketball-reference site you will see a table with player stats that we want to scrape from. It is best to do this in Chrome so that we can easily inspect the element to see the html for the page. In this section we identify the header row. Here is our finding from inspecting the element:

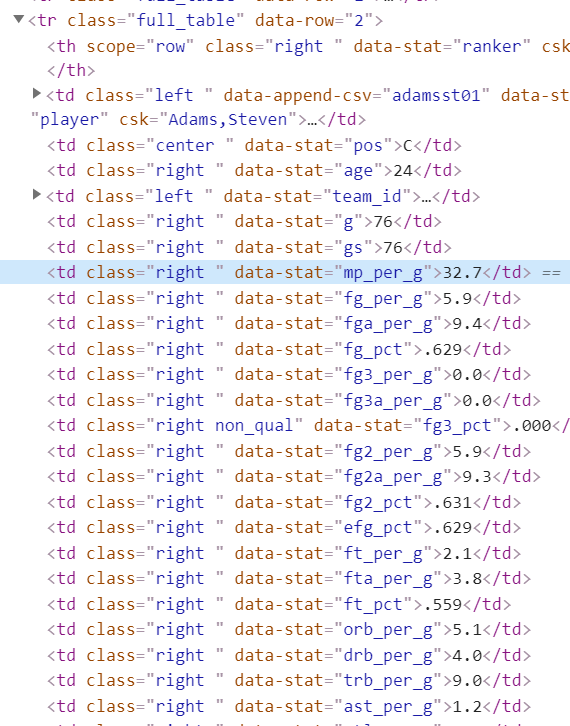


You can see <tr> at the top meaning table row. Within this row we can see several <th> sections which stand for table header. Each <th> is just the name of the column. You can read the code is section two as: Create a list called “headers” containing all of the table headers within the first table row. I recommend going to the website and right clicking the headers, inspecting and looking at this element. It highlights the cell it is talking about which makes it easier to see what is going on.

3: Remove the first row of the headers because it is the index column.

4: Now we find all of the table rows of the data and names them rows. We exclude the first one because it is the headers.

5: We need to get the data from each row. Table data is <td> in html. Below is the html of an example row:



At the top is the table row and then each data point within that row. The “data-stats” show the header the data belongs to so this row is for Steven Adams. This code can be read as: For each row in the data frame, create a list of the data in that row.

6: Create a pandas dataframe using the stats data and the headers that we scraped.

7: Create a column that lists the year the stats are from.

**Part 2: Function to scrape salary**



1: Create an empty list.

2: Go to the espn website. An example url is: <http://www.espn.com/nba/salaries/_/year/2018/page/1>

We can see that there are a couple additional things to tackle on this website. They have a new row of headers after every 10 players and there are 15 pages of data. Thankfully, the url includes the page number so the user can add the number of pages to 15 in this case or else I have it default at 50 and it doesn’t seem to cause an error if you overshoot the pages. We will need to loop through each page.

3: Section three is a very similar process to the pull\_stats function and will create a dataframe with one page worth of salaries.

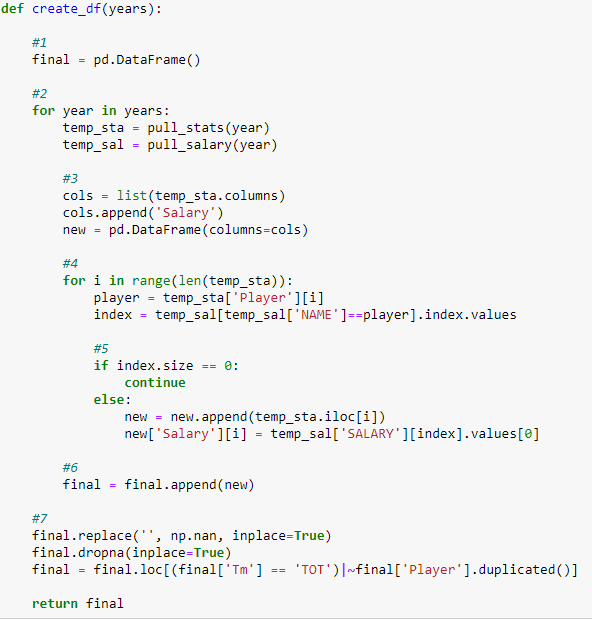
4: The name column also has the players position after a comma so I remove the position because we already have that from the stats data. We then add the data from this page to the list named “full” and start the loop over rat the next page.

5: Once we have all of the data from every page into the “full” list we concatenate that list into a pandas dataframe.

6: Remove any row that has ‘RK’ in the ‘RK’ column because it is a repeat header row.

7: Reset index

**Part 3: Function to create dataframe**



1: Create an empty dataframe

2: This function takes a list of years so this section starts a loop for each year in the list. Create a stat dataframe and salary dataframe for this loops year.

3: Create a list of column names and add a column for salary. Create a temporary dataframe with those column names.

4: A loop that goes through each player in the stats dataframe. It takes the players name and looks for that name in the salary dataframe and saves the index of that players row.

5: The index is saved as an array. If that array is empty (size == 0) then the name was not found in the salary dataframe. The function ignores that player and moves on to the next one. If the array does contain an index, we append the players stats to the temporary “new” dataframe and append the players salary to combine the data.

6: The temporary “new” dataframe is appended to the “final” dataframe. Then the process starts over for the next year in the list.

7: Once all of the data is placed into one dataframe we clean it. I removed any row that had missing data. There are also many players that play for multiple teams each year. Basketball-reference deals with this by giving data for each team but also giving a combined row where the team name is ‘TOT’ for total. Here I remove all duplicate rows except for those that have a team name ‘TOT’ so that I don’t have data from partial years.