Jacob Birnbach
Create Task

3a.

The question that prompted my program was, “Who is better at basketball, Michael Jordan or Lebron James”, an often disputed topic among basketball analysts. Since Jordan and Lebron played during different time periods, it is very difficult to determine who is better. My program attempts to answer this question using statistics. In this video, my program calculates and compares Jordan and Lebron’s career average points per game with the mean points per game of all players during the era in which the specified player played in. Jordan is compared with players from 1985 to 2003 and Lebron is compared with players from 2004 to 2021. My program then uses these statistics and parameters to calculate the z-score for each player. The player with the higher z-score is the better player for the specified stat type. My program also produces graphical outputs that show where each player’s career average lies on the normal distribution of all players that played during their career.

3b.

First Code Segment

```
def get_year_averages(self,year):

       #calculates stat average for specific year

       url =
'https://www.basketball-reference.com/leagues/NBA_{}_per_game.html'.format(year)

       stats = get_data(url)[self.stat_type]

       #converts data to list

       arr = []

       for num in stats:

           if num != None and num != '':

               arr.append(float(num))

       #returns list containing mean and standard deviation of season averages

       mean_sd = []

       mean_sd.append(st.mean(arr))
```

```
        mean_sd.append(st.pstdev(arr))

        return mean_sd
```

Second Code Segment

```
    def decade_averages(self):

        current_year = self.year_f

        y_means = []

        y_stdevs = []

        while current_year <= self.year_l:

            year_stats = self.get_year_averages(current_year)

            y_means.append(year_stats[0])

            y_stdevs.append(year_stats[1])

            current_year +=1

        self.years_mean = st.mean(y_means)

        self.years_std = st.mean(y_stdevs)

        return 0
```

3b.

After the screen scraping algorithm creates a Pandas data frame of every player’s per game stat from a specific year, my program indexes the column stats using the stat type keyword (PTS, AST, STL, etc). I then store the stats in my “arr[]” list and calculate a mean and standard deviation of the list. I store these descriptive stats in another list called “mean_sd[]” where the mean corresponds to the first element and the standard deviation is the second element. My “get_year_aveages()” method returns this list. This method is then called many times in my “decade_averages()“ method which uses the player object’s first year and last year attributes and iterates over every year in the range to calculate a mean and standard deviation for every year in the player’s career using the “get_year_averages()” method. It then appends the mean and standard deviation for every year to the “y_means” and “y_sd lists”. It then calculates the average mean and average standard deviation of these lists and stores them in the player object's "years_mean" and "years_std" attributes. Without these lists, I would have no way to store and calculate the NBA league averages for every year in the player's career. Without these parameters, I would have no way to calculate each player's z-score and there would be no way to determine which player is better.

3c.

Compare Z Scores Function

```
#calculates z-score
```

```
def calc_z_score(x, mu, sigma):

   return (x-mu)/sigma

#compares both player z scores and decides who is better (larger z score)

def compare_z_scores(p1, p2):

   if p1.decade_averages() or p2.decade_averages() != 0:

       print('Calculation Error Exiting')

       exit(1)

   p1_z = calc_z_score(p1.mean,p1.years_mean,p1.years_std)

   p2_z = calc_z_score(p2.mean,p2.years_mean,p2.years_std)

   message = '{} {} with a z-score of {} is better than {} {} with a z-score of {}
when comparing {}'

   if p1_z > p2_z:

       message = message.format(p1.first_name, p1.last_name, round(p1_z,3),
p2.first_name, p2.last_name, round(p2_z,3), p1.stat_type)

   elif p2_z > p1_z:

       message = message.format(p2.first_name, p2.last_name, round(p2_z,3),
p1.first_name, p1.last_name, round(p1_z,2), p1.stat_type)

   else:
       message = 'Both players have the same z-score so they are equally as good'

   return message
```

Instantiation of Player objects and call to compare z scores function

```
P1 = Player(p1_first_name,p1_last_name,p1_FY,p1_LY, TYPE)

P2 = Player(p2_first_name,p2_last_name,p2_FY,p2_LY, TYPE)

print(compare_z_scores(P1, P2))

plot_z_score(P1, 'p1')

plot_z_score(P2, 'p2')
```

The “compare_z_scores()” function calculates and compares both player objects’ z scores to determine who is the better player. The “decade_averages()” method is called for each player object and iterates over every year in the player’s year range to calculate an average mean and standard deviation for the entered stat type during the player’s career. It then stores these values in the player object’s “years_mean” and “years_std” attributes. The “calc_z_score()” function uses these attributes to calculate a z-score for each player, which is stored in variables called “p1_z” and “p2_z”. These z-scores are compared and the player with the higher z score is formatted into the “message” string as the better player. This output message is then returned and printed on the console. Without this procedure, I would have no way of calculating and comparing each player’s z-score, as well as producing an output message to the user.

3d.

The outcomes of this procedure depend on how good the entered player was compared to the other players they played with. If player 1’s z-score is greater than player 2’s z-score, their name and z-score will be formatted into the first 3 place holders in the output message.
When comparing points per game, if Lebron James is player 1 and Michael Jordan is player 2, player 2 will have a higher z-score than player 1 because Jordan’s z-score is greater than Lebron’s z-score. Since Michael Jordan was player 2, the condition p2_z > p1_z will be met and the output message will format Michael Jordan’s name and z-score as the better player.
When comparing career 3p%, if Stephen Curry is player 1 and Shaqille O’Neal is player 2, the condition p1_z > p2_z will be met. Since Stephen Curry is a significantly better 3 point shooter than Shaq and Curry is player 1, the output message will format Stephen Curry’s name and z-score as the better player.