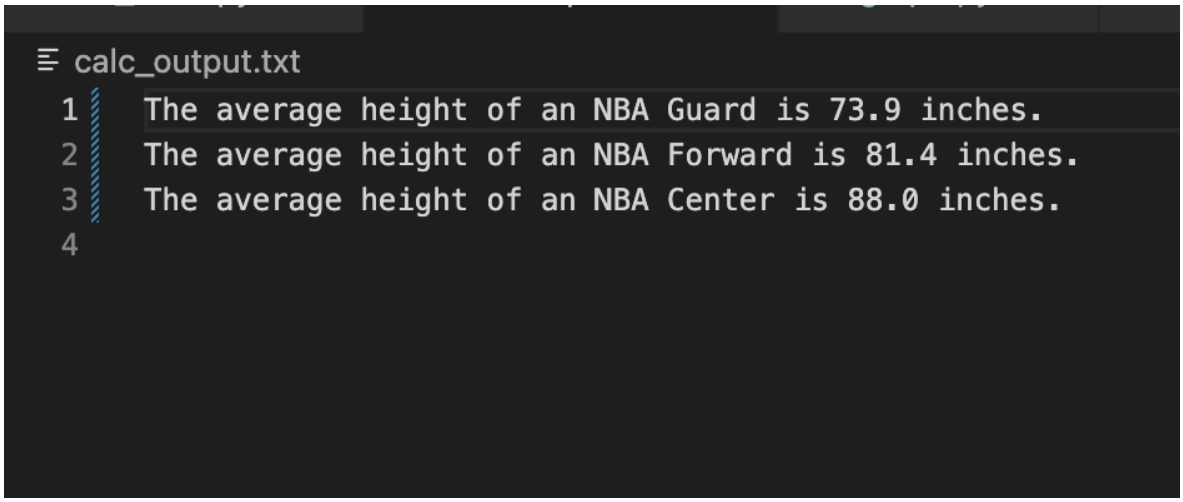
Github Repository: https://github.com/SophAnd/SI-206-Final-Project.git

1. The goals for your project including what APIs/websites you planned to work with and what data you planned to gather (10 points)
    - The goal for our project was to calculate the likelihood of a player have a high average points per game, as set by some bound (for example what is the chances that a person averages over 24 points per game and plays the point guard position. Or what is the probability that a person averages more than 20 points per game they are under 6 feet tall). We had planned on working with a NCAA website and ESPN API but ran into issues because the page had been formatted to only show 50 results unless the user clicked a button which made web scraping very difficult. We had planned on collecting data about the player including their name, the position they played, their height, how many games they have played, as well as how many points they averaged.
2. The goals that were achieved including what APIs/websites you actually worked with and what data you did gather (10 points)
    - The goals we achieved with the APIs/websites we actually worked with included determining if there was a trend between player height and the position they are playing. We narrowed our focus to a player's name, their height, the position they played, and their home state. We collected data about female high school players that are committed to play college basketball and (second website group). We decided to switch to shift our focus because average points per game was an already calculated statistic and could have many different factors that affect it. For example if a player suffers an injury at the beginning of the game the number of games they played increases and their points don't drastically hurt their average points per game. We wanted to gather data to see if a certain build was meant for a certain position and whether or not this was true across different age ranges.
3. The problems that you faced (10 points)
    - One of the problems we faced was having to find new websites to scrape for information. While we tried to solve the issue of loading at least 100 items on the ESPN page, the button didn't modify the URL and we, despite lots of research, were unable to come up with a solution and had to find a different source. We then became confused on whether the websites we had selected were API's or not. We wanted to stay in the field of team sports because we felt that having a topic that we are passionate about will motivate us to put more effort into the project. We searched the github with free api's given to us in the project instructions, but were unable to find a free api that had the data we were looking for. We got close with the Canadian Football League (CFL), but that needed an access code that we were unable to find. I believe that our journey of searching for a website taught us both about the different types of webpages that exist and gave us a better understanding and appreciation of them.

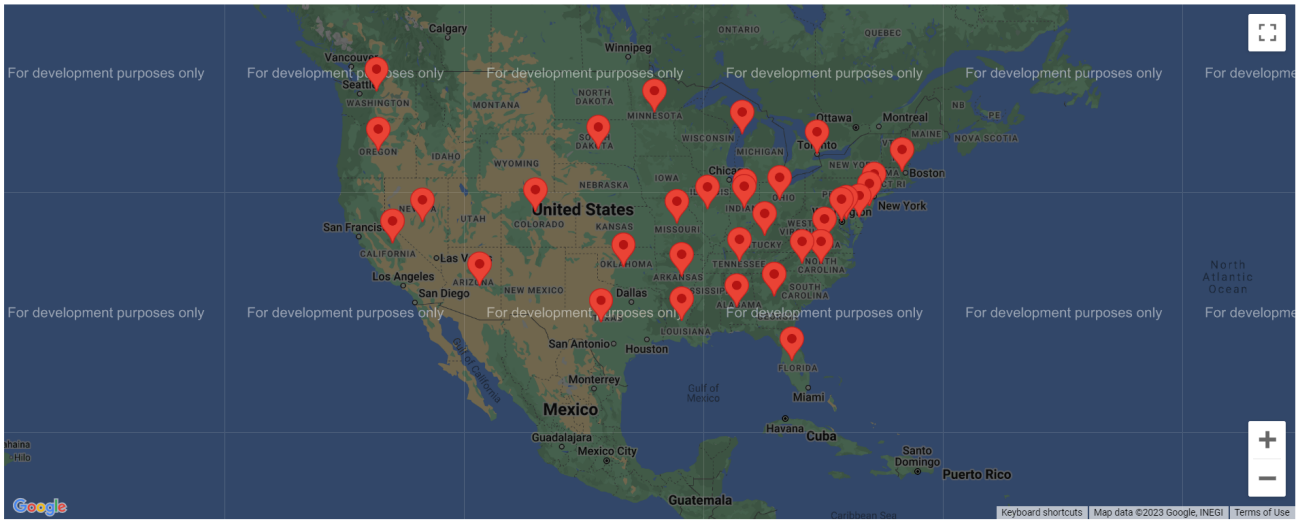
| Issue Description | Result (Issue Solved) |
|---|---|
| Loading data on page error. We didn't know how to scrape the data since the full | We were unable to solve the issue so we decided to move to a different website. |

| | |
|---|---|
| number of results we needed were not accessible until after the user clicked a button. The default HTML wouldn't allow us to scrape normally. | |
| (Saif) I had an issue when trying to scrape my website because it had divided entries to the "table" into an odd and even class. I was confused about how I should get the entries because I wanted to keep it in the same order. I didn't want to have to jump back and forth between a list of odd entries and a separate list of even entries. | I was able to "inspect element" and see that each of these odd and even entries had more specific sub-elements in them that were common to both the odd and even CSS classes. Using this I had what I needed to get the important data as one list and then I could clean it up. |
| (both) We had an issue with creating an integer key because we wanted it to be height but each players height was given as a string and in the form "feet'inches"" or something similar (for example one of the websites used 5'11") which is not an integer. | We decided to use a mix of the .strip() and .split string methods to isolate the feet value from the inches values. Once the values had been isolated we used typecasting using int to convert the string value to an integer. We then used the formula feet*12 + inches to get a final value of height in inches that could serve as the integer key. |

4. The calculations from the data in the database (i.e. a screen shot) (10 points)
    - For the dataset pertaining to NBA players, we were able to calculate the average height of a guard, a forward, and a center. Reference the image below.
    - 

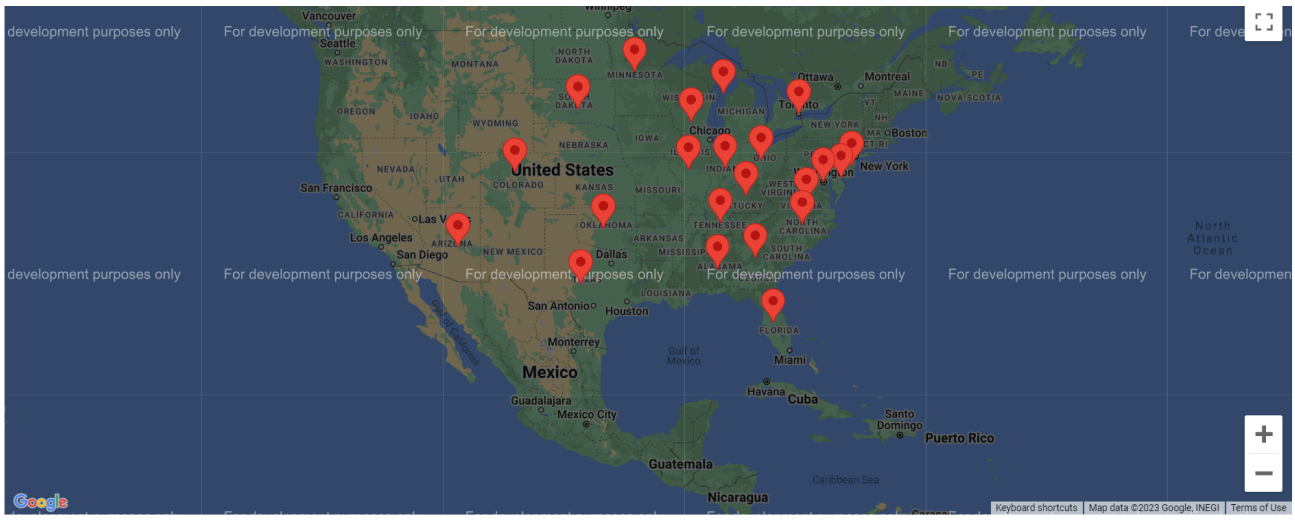
5. The visualization that you created (i.e. screen shot or image file) (10 points)
    - One of the visuals we created was a map that outlined the different locations that the female high school basketball players were pulled from. While this may look similar to the lecture example, a few modifications that were made were that we only focused on North America (given our data set), we didn't map points based on cities, we did them based on states, we modified the website itself as well. We changed the size of the map

in reference to the page (we increased the size of the map canvas) and added a caption that matches the map which represents our data. There are no duplicates which is why it looks like less than 100 rows. We wanted unique locations only because we felt duplicate points would become crowded and unnecessary.




**About this Map**

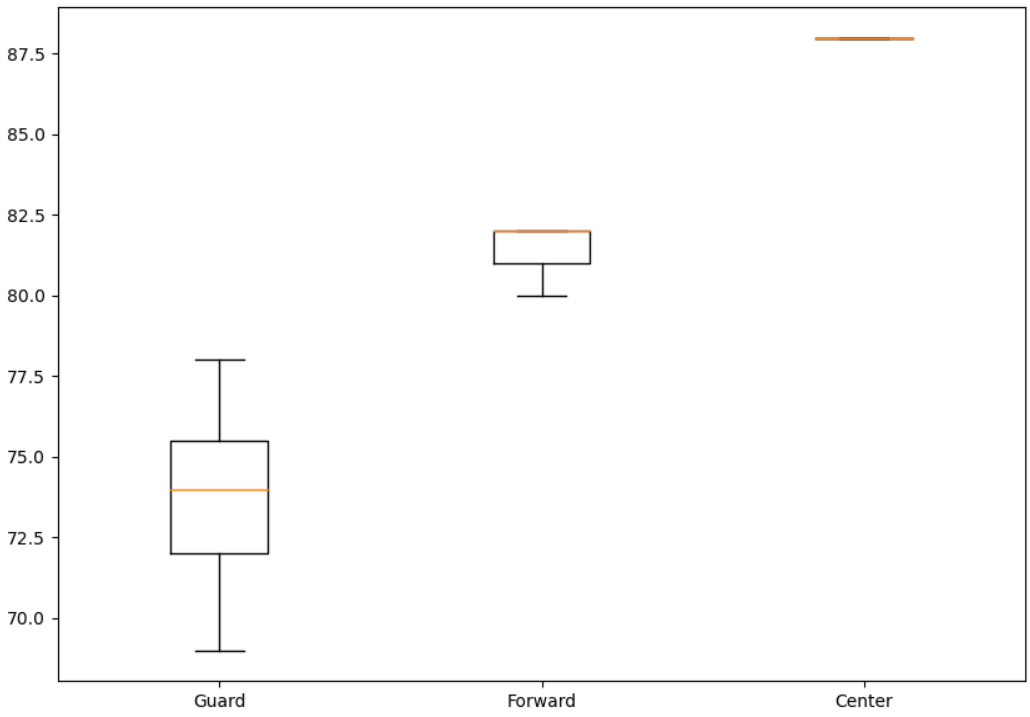
- This is a cool map from: www.py4e.com. It has the locations of all the high school players that we collected data on.
- The second image has the same modification made to it as the first one does (in terms of difference from the lecture recommendations), but this one only contains data about players under six feet tall. As compared with the image above, it seems that the data for this region is centered around the east coast, rather than spanning up the west coast and central America like with the original data set.




**About this Map**

- This is a cool map from: www.py4e.com. This map illustrates states which have players in the top 128 high school female basketball players that are under 6'0. There seems to be an interesting concentration around the east coast. In a sport dominated by height, what positions would shorter plays opt for?

- From the NBA data, we created a visualization using matplotlib to show box plots of heights of players based on their position.




6. Instructions for running your code (10 points)
   - Specify the url you would like to scrape from. Make sure that the bounds of your loops are correct. Once these items have been set you will need to open up DB For SQLite, this will allow you to view the outputs of the program, which is what we care about (the table version). Run the program and then open the SQLite file using DB For SQLite. Having the website that you're scraping from is also helpful so you can verify that the data has been read in correctly and you are getting the values that you want.

7. Documentation for each function that you wrote. This includes describing the input and output for each function (20 points)
   - Saif: I created two void functions for creating and inserting into the database initially. I began by scrapping for the player names, followed by player positions, and finally their heights. Once the data had been cleaned to match the format I wanted I initialized the database in the first void function and then called the insertion function multiple times to make sure that the data I had cleaned was getting added correctly. After the insertion function had finished being called the database was completed. The creation function takes in no parameters and the insertion function takes in 5 parameters: the players name list, the player's height list, the players position list, and the player's "home state" lists or the state their currently playing in, and finally an index which helps the for loop where it ended (also to ensure we never read more than 25 values at a time).

- Sophie: I created six different functions. The first function takes in a url as an input and returns a list of tuples. The second function takes in a url for a website as well and returns the combined result of a two tuple lists (it calls the first function twice and then combines these two results into a master tuple list). The next function takes in a database name and then returns a connection and cursor to the SQLite file. The next definition takes in a tuple list, a cursor to a SQLite file, and a connection to the SQL fike, It returns nothing. The 5th function takes in a cursor and connection to a SQLite file and then creates a table provided that one does not already exist (it will be replaced). It also returns nothing. The last function being the main function, which takes in parameters and doesn't return anything.
- Both: We decided that our integer key (as specified by the project requirements) would be the heights of the players. We ran into an issue with this (detailed below) but were able to make it work out. Once our respective databases had been created we could use the JOIN keyword to see if there were players of a certain height and if there were a new table (with less than 25 insertions) could be created. We could then look at this table and see what positions they played to see if there was a correlation between the size of a player and their respective position.

8. You must also clearly document all resources you used. The documentation should be of the following form (20 points)

| Date Issued | Issue Description | Location of Resource | Result (Issue Solved) |
|---|---|---|---|
| 04/21/2023 | Get a better understanding of what APIs are as well as how they operate | Online resource (work cited included) First reference | Strengthened understanding of APIs helped with high level project details |
| 04/21/2023 | Needed a review of beautiful soup as well as high level practice problems to refresh our memory. | Online resource (work cited included) Second reference | Helped give basic understanding and review of what Beautiful soup is as well as how it works. Examples were pretty basic but were still helpful for revision. |
| 04/21/2023 | Needed specific examples with detailed explanation to get a better understanding of the intricacies of beautiful soup. | Online resource (work cited included) last reference | More practice with web scraping. The website has examples with detailed comments which was helpful when we got stuck on what a line of code was supposed to be doing. |

| | | | |
|---|---|---|---|
| 04/21/2023 | Needed relevant examples of how we should implement ideas. Having practice problems would help us strengthen concepts and we needed guidance on how to approach problems. | Course Resources (RuneStone as well as lecture slides) | Helped give specific and relevant examples of how different components of web scraping work. The RuneStone practice was very helpful in making sure we understand what each line of the code was doing. For beautiful soup as well as with SQLite |

Documentation For Resources Used:

- Besides Visual Studio Code (and the anaconda set up), we used DB Browser For SQLite as well as referencing various online articles. We pulled data from: College Sports Madness and (insert link**). We referenced the RuneStone practice problems and lecture slides heavily as well. We have a list of MLA citations below.

Work Cited:


Amazon Web Services. (n.d.). *What Is An API (Application Programming Interface)?* AWS. Retrieved April 21, 2023, from https://aws.amazon.com/what-is/api/

"Box Plot in Python Using Matplotlib." GeeksforGeeks, GeeksforGeeks, 8 Mar. 2022, https://www.geeksforgeeks.org/box-plot-in-python-using-matplotlib/.

OVER 9000! PythonProgramming.net. (n.d.). *Web scraping and parsing with Beautiful Soup 4 Introduction*. Python programming tutorials. Retrieved April 21, 2023, from https://pythonprogramming.net/introduction-scraping-parsing-beautiful-soup-tutorial/

Richardson, L. (n.d.). *Beautiful Soup documentation*. Beautiful Soup Documentation - Beautiful Soup 4.12.0 documentation. Retrieved April 21, 2023, from https://www.crummy.com/software/BeautifulSoup/bs4/doc/