Names and IDs of all participants.

P1 – Leyla Ellazova – 444831

P2 – Islam Islamov – 444601

		Short description
Short description		The webpage consists of information related to different football(soccer) leagues and players, as well as their nationalities. We will be focusing on players only and all the info related to them. Link to website - https://en.soccerwiki.org/
Short description of scrapers	Beautiful Soup	Initially we import all the stuff we will need later, these are: 1.BS from bs4 2.requests 3.pandas as pd Then we will carry on with HTTP headers, which let the client and the server pass additional information with an HTTP request or response. Next, we will create global containers for our scraped content. Then an empty csv file where we can put our data. Next are the URLs, which are followed by our get_stats function that will return us the most useful data for later analysis. Then we will parse the HTML for operations using html.parser. Then we will get our information by inspecting webpage. Then we will loop through and get the data we need (here we have all the rows in our table). It will basically find us the stuff we need. Then we will append our player stats (update player stats). From here we will carry on with writing(saving) player stats to a file (created earlier). Then we will include headers to our csv and create a data frame for it. And after that we just run our scrapper, which will generate us a file of 100 top players ranked by SOCCERWIKI, regardless of position(on field) or anything else.
	Scrapy	First, we start with importing scrapy and pandas as pd. Second, we set url and next urls that we will get later. Next, create pages dynamically by looping through range of 15(because the code fetches 15 set of results from the server). After that, we define a custom class Soccer, which inherits from the spider class(scrapy) Also, we add start urls, the output which will be written into the csv file. We define the spider's name as `soccer_spider.` Then initialize the empty list (player stats for appending data later on). We parse the response. And then use the css and xpath html selectors to find the information we need by looping through rows under the body. Then we make another dictionary for stats(empty). Then we will use join method in order to convert our list to the string. Then we update our stats file that we created earlier. We get top 100 players by RATING (the highest to lowest). Then we also included the headers to our final csv file. Next, we created data frame(pandas) for stats of the players. Next, we wrote to our output file using pandas.

	Selenium	Again, first we import all the useful tools. Pandas and Selenium. Then we add options in order to ignore some of the errors: ssl errors, certificate errors. Also, we will allow specific level of errors to occur now when we run the codes. Then we start the driver with our own options. The reason for this is that selenium automatically opens the browser and we bypass the default unsecure configuration, and also ignore the minor warnings. Then we get the stats from website, by starting from the URL that we input here. Then we get the next pages dynamically, through each 15 pages (15,30,45,60,75 etc.) Next, we use the stats function in order to loop through the pages and get the url data for each of them. Next, we get the NAME, CLUB, POSITION from the stats body, however, the remaining stuff will be scraped in next part(other_stats), such as HEIGHT, FOOT, AGE and RATING. After all these steps we finally write all the scraped data to our csv file. We also include column names or Headers. The print statements are for testing and debugging (information messages) Messages can be omitted if needed, however they are added to indicate when each part of the code is finished.
Short technical description of output		The idea was to scrap players and their names, clubs, positions, ratings, age, foot preferences and height(cm). The information will be saved to csv file to make it easier to do further analysis. When it comes to output, it is basically top 100 players according to SoccerWiki.org and their stats (based on rating). As follows: **NAME CLUB POSITION HEIGHT FOOT AGE RATING**
Elementary data analysis		So, we decided not to do so basic analysis since we needed to do the correlation test and the clustering too. First, we did clustering, which gave us the optimal number of clusters of 3, which was checked with elbow method, since it is the most basic one. Next, after the clustering we did the correlation test between the ages of players and their ratings, we wanted to know if there was a relation between them. Turns out that there was almost no relationship between them, since both correlation test gave the same results of ~0.189 meaning that the correlation is very weak. (Pearson) However, we assume that if there was a chance to scrape more players the numbers would be different, as 100 rows in our opinion is not that much.
Job Distribution		

		Initially we were trying to do everything separately, however we soon changed our minds since we had some minor errors, which were affecting the later analysis. Leyla did mainly selenium part and Islam did beautiful soup part. When it comes to scrapy, it was done together, since this was the part, we struggled the most at. The codes are very user friendly and are easy to understand unless you drastically change anything inside the files the results should not change from those that we obtained.