**Advanced data and coding  
Final assessment  
Gabriel Prieto Ch.**

**Rationale of the project**

The project intends to interactively display how much presidents and world leaders around the world earn in their annual wage, and see if their wages correlate somehow with the average national wage of their own country. The user will have the opportunity to explore different alternatives, pointed below:

1. A comparison between two different World Leaders.
2. A comparison between a World Leader and the national average data of his/her nation.
3. A comparison between a World Leader and the national average data of another country.
4. A comparison between a World Leader and the input of the user, thought to be his/her own salary.
5. A page of both a scatterplot and a column graph that contains the final overview and conclusion of the project.

In order to make it work, I first needed the data to work with. Specifically, in order to do a comparison, I needed the World Leaders’ salaries, the average national wages of all the countries, and Gini coefficient (index that measure distribution of wealth in a country).

This information, along with extra data, was found in the following webpages.

Countries of the World: <https://www.countries-ofthe-world.com/all-countries.html>

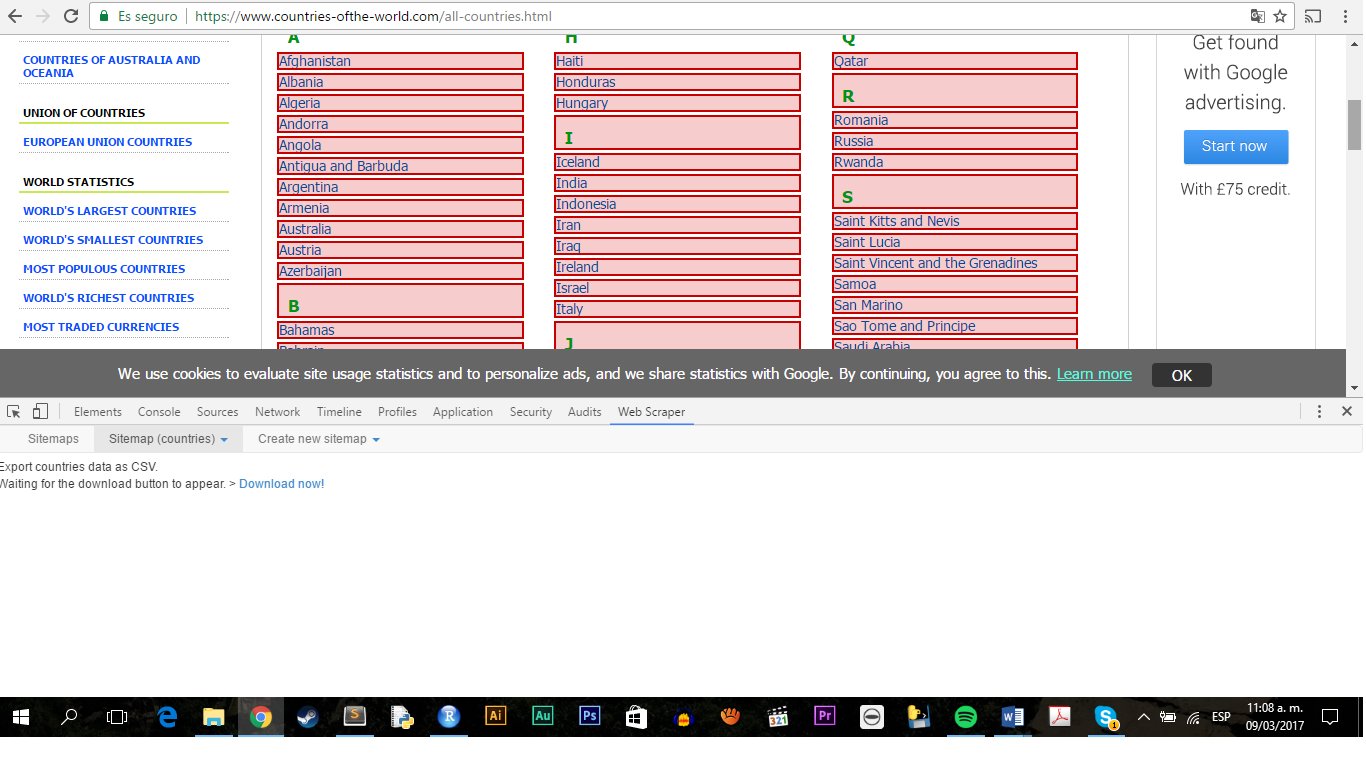
World Leaders’ salaries: <http://www.paywizard.org/main/salary/vip-check/world-leaders-salaries>

Average national monthly wages in the world: <https://www.statista.com/statistics/226956/average-world-wages-in-purchasing-power-parity-dollars/>

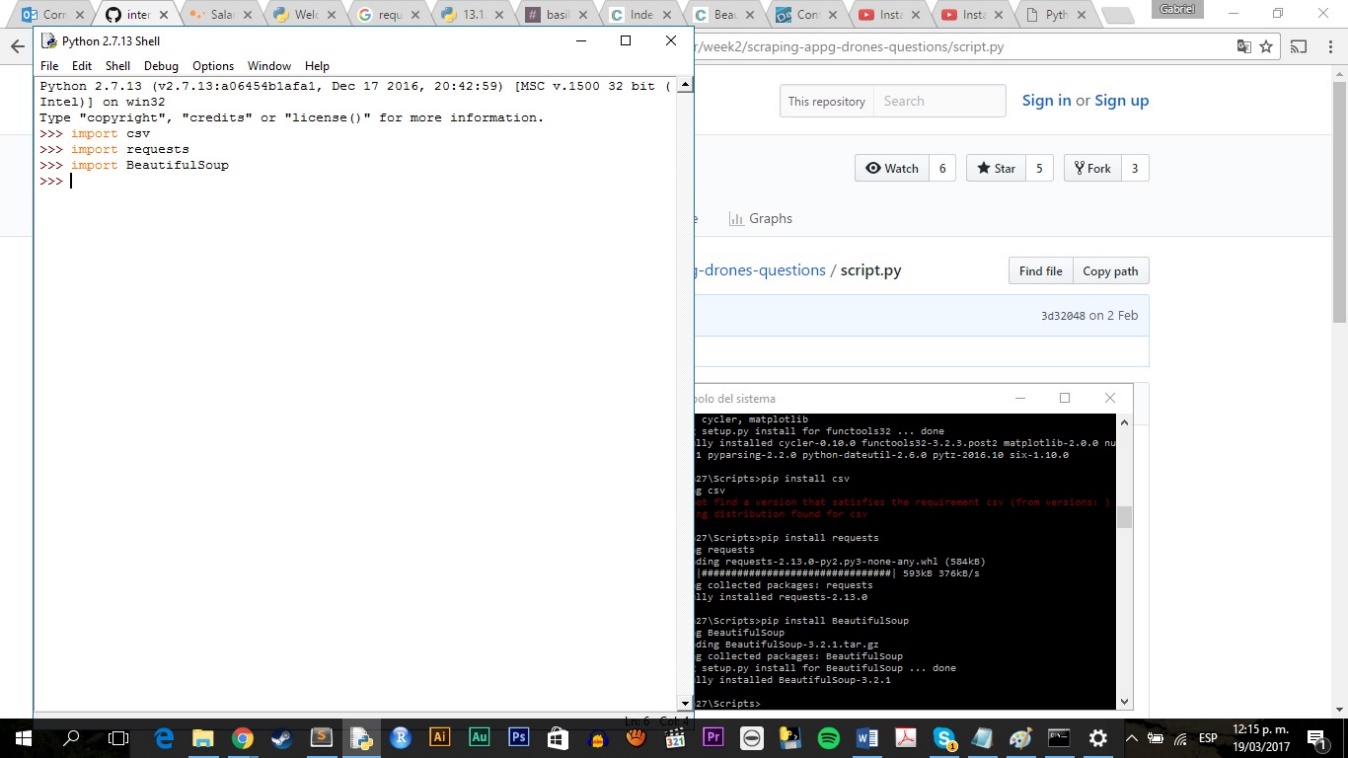
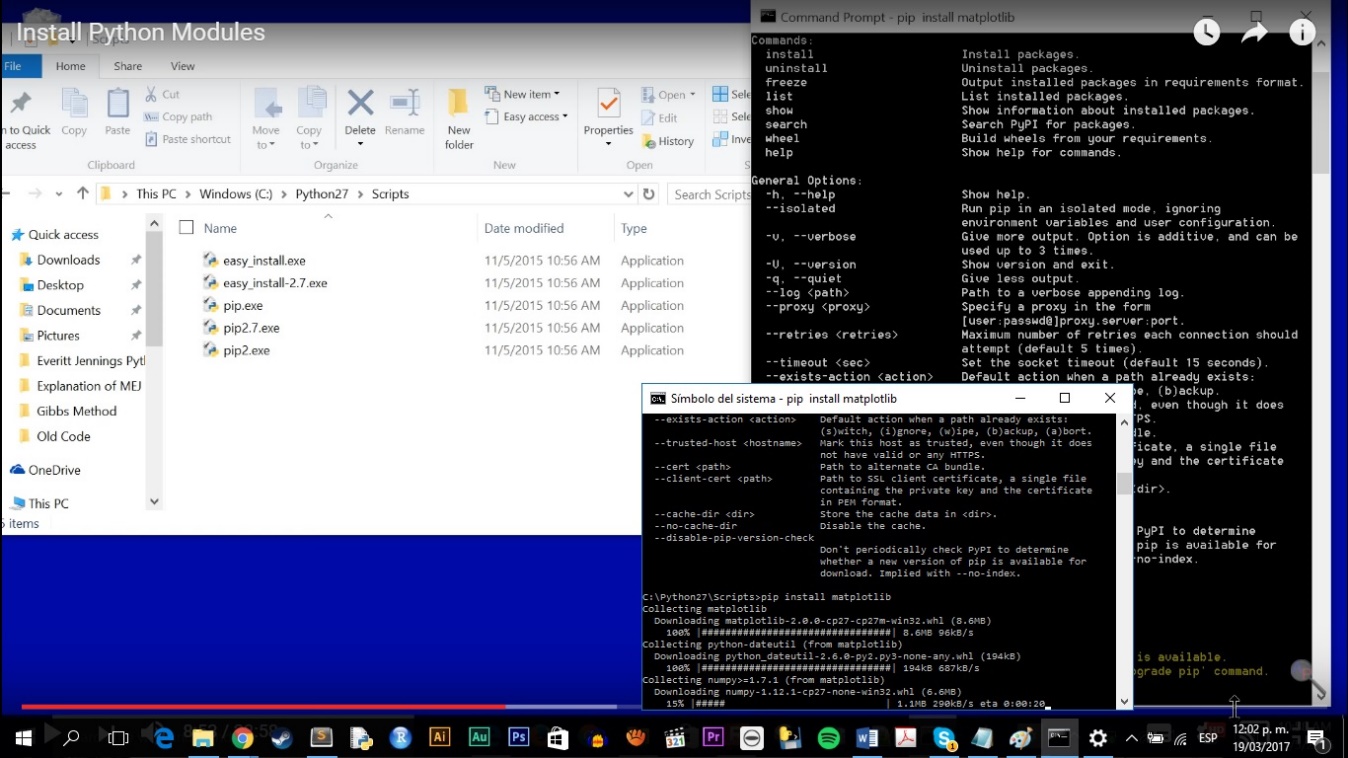
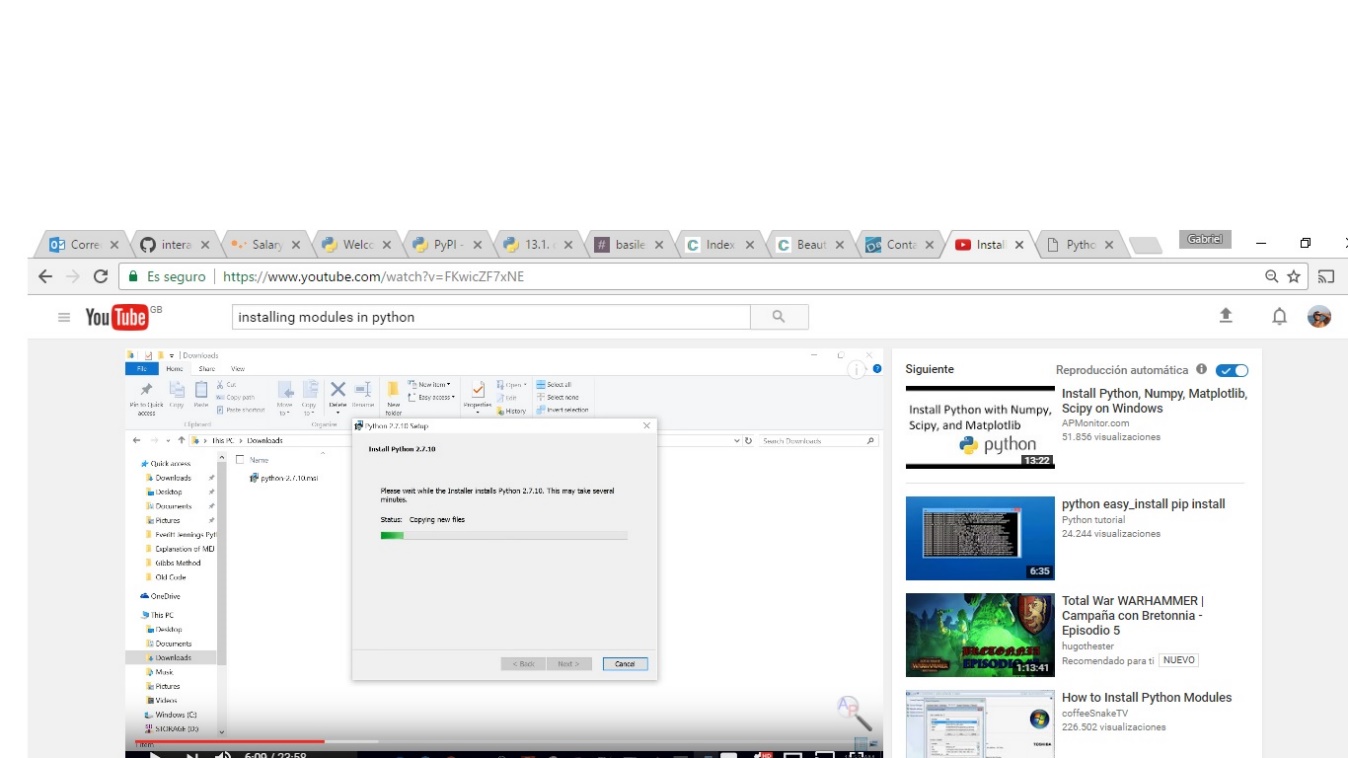
Gini coefficient of countries: <https://en.wikipedia.org/wiki/List_of_countries_by_average_wage>

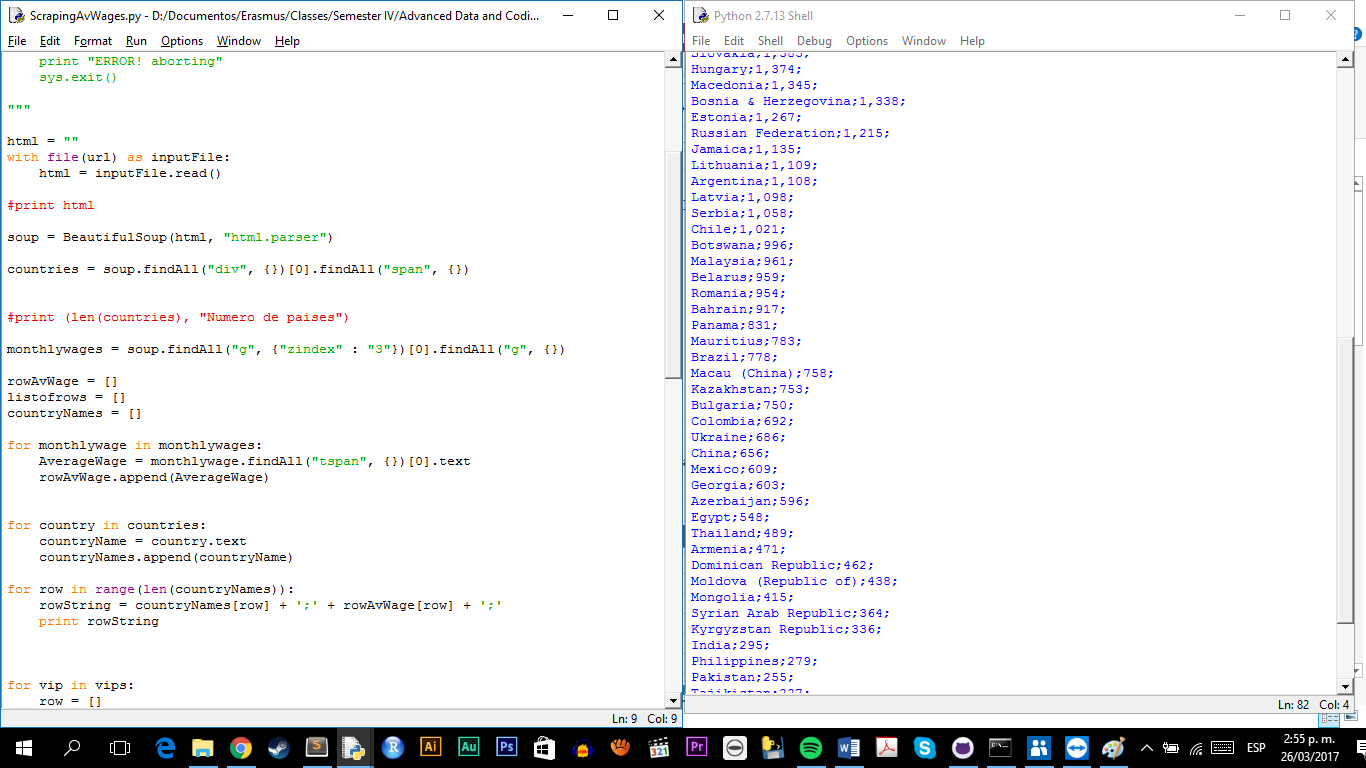
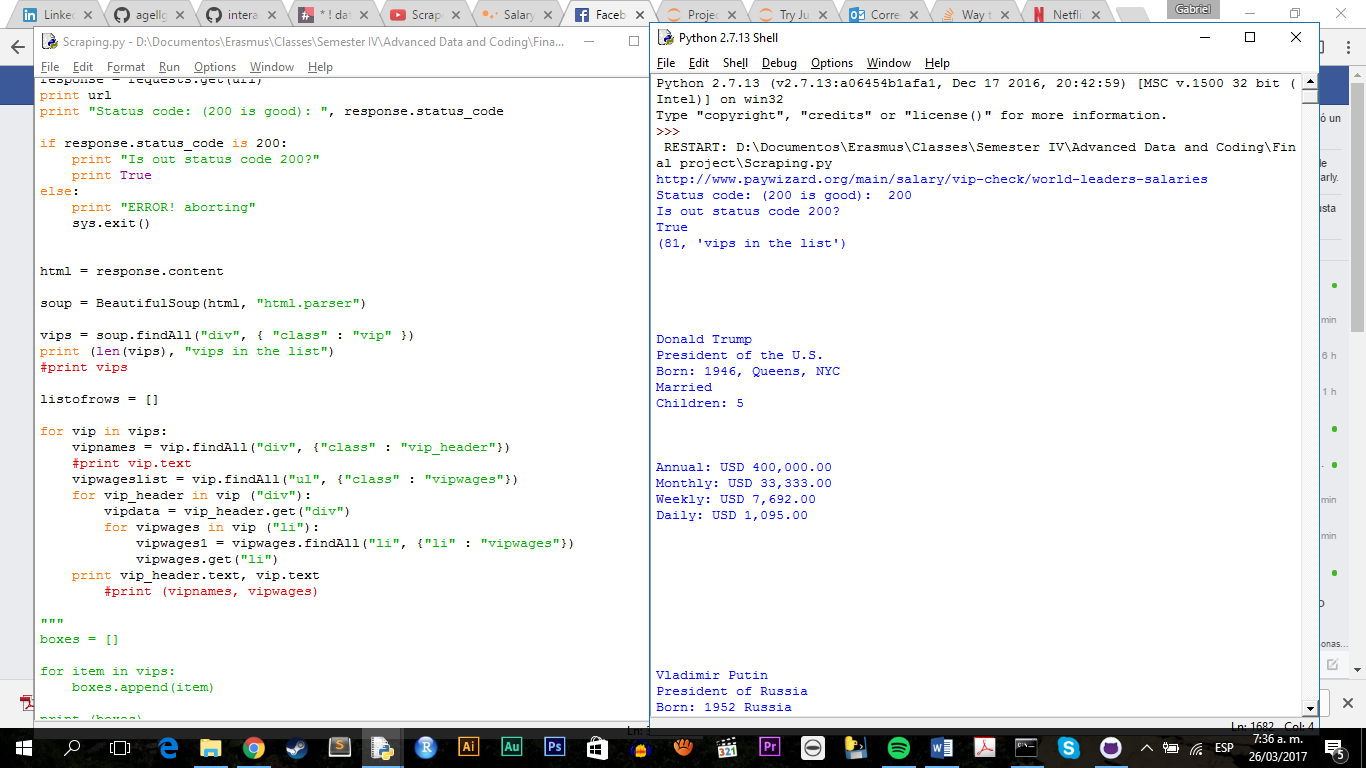
**SCRAPING**

In order to acquire the information from those webpages, I opted for using scraping techniques. Initially, I tried using a the scraping tool Webscraper, but that tool only worked for the Wikipedia pages.



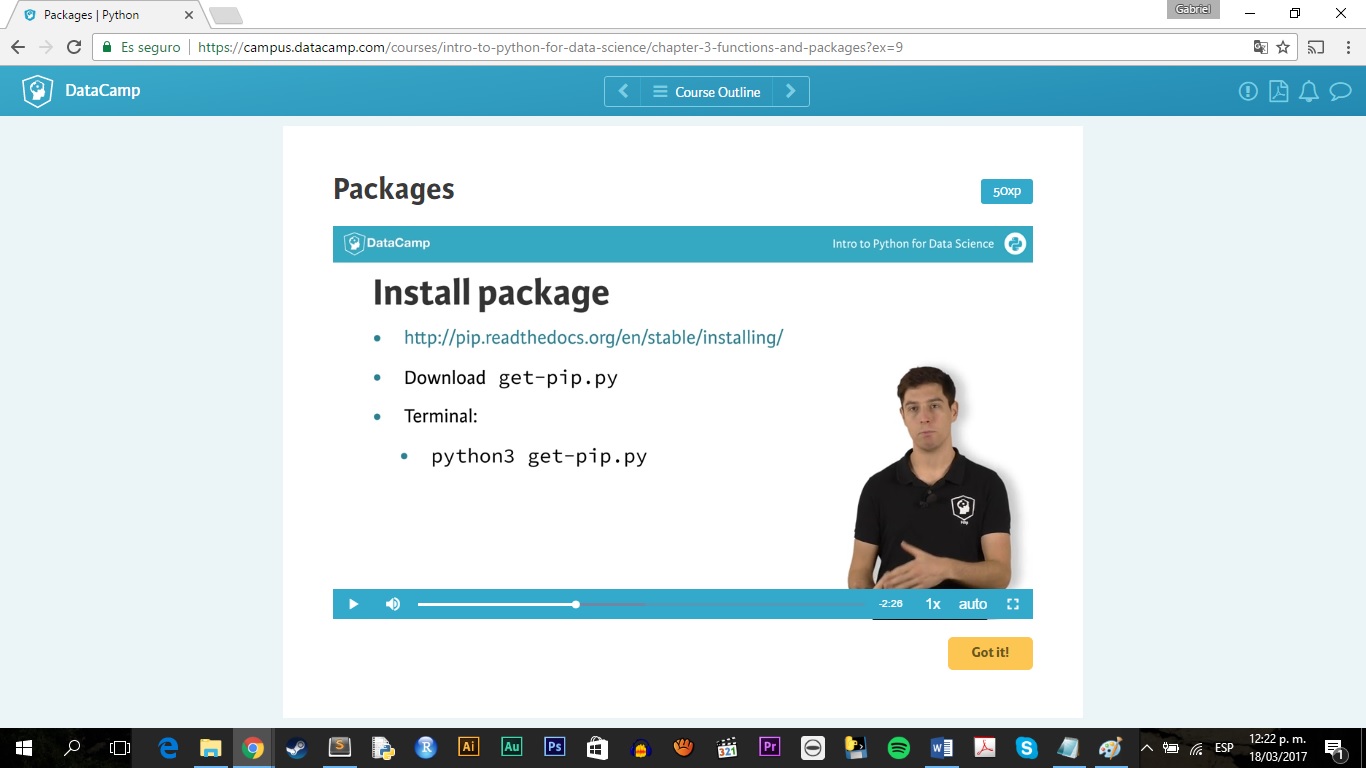
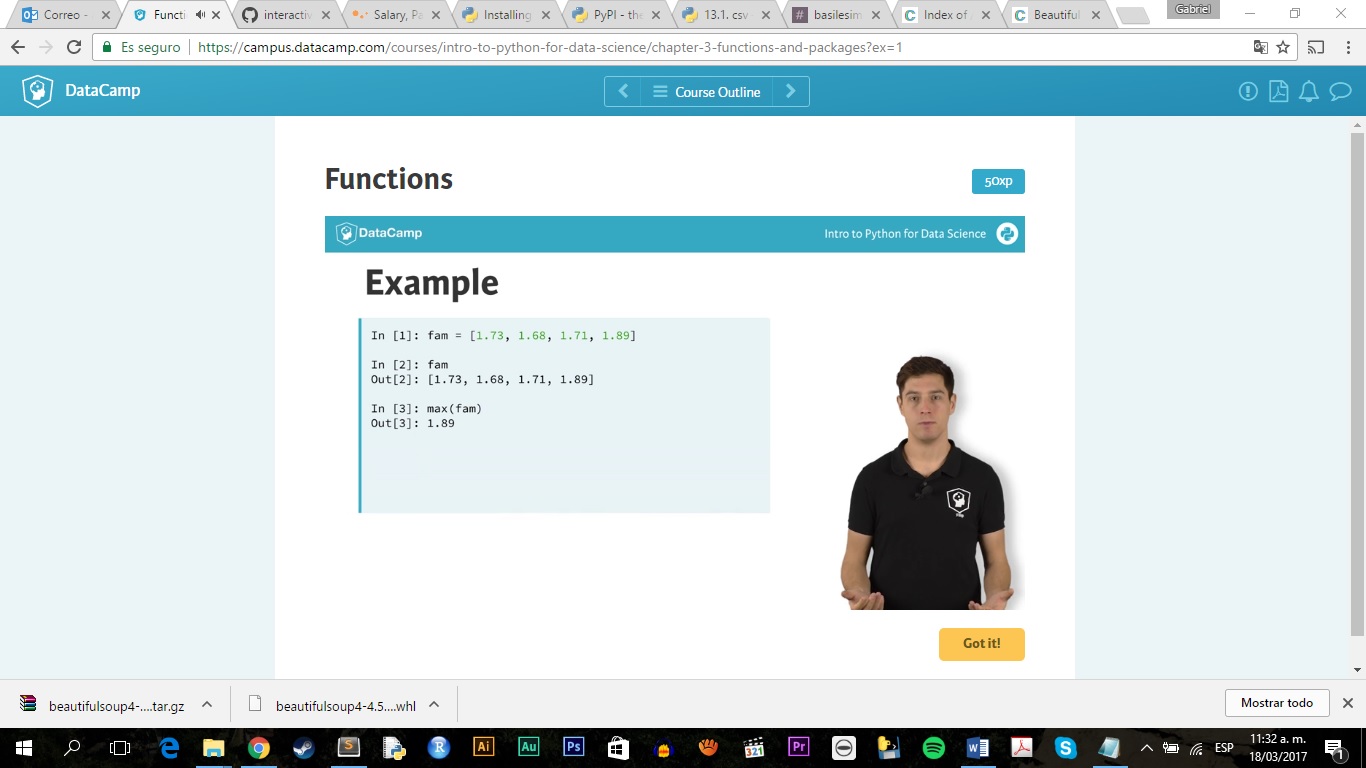
After several tries with other scraping tools, I choose Python as my method for scraping. I followed several tutorials of self-learning, in order to fully understand how Python worked. I realized I needed to install some extra modules for Python, which I did, following more examples on the web on how to do it.





Methods and modules in Python – how to implement them.

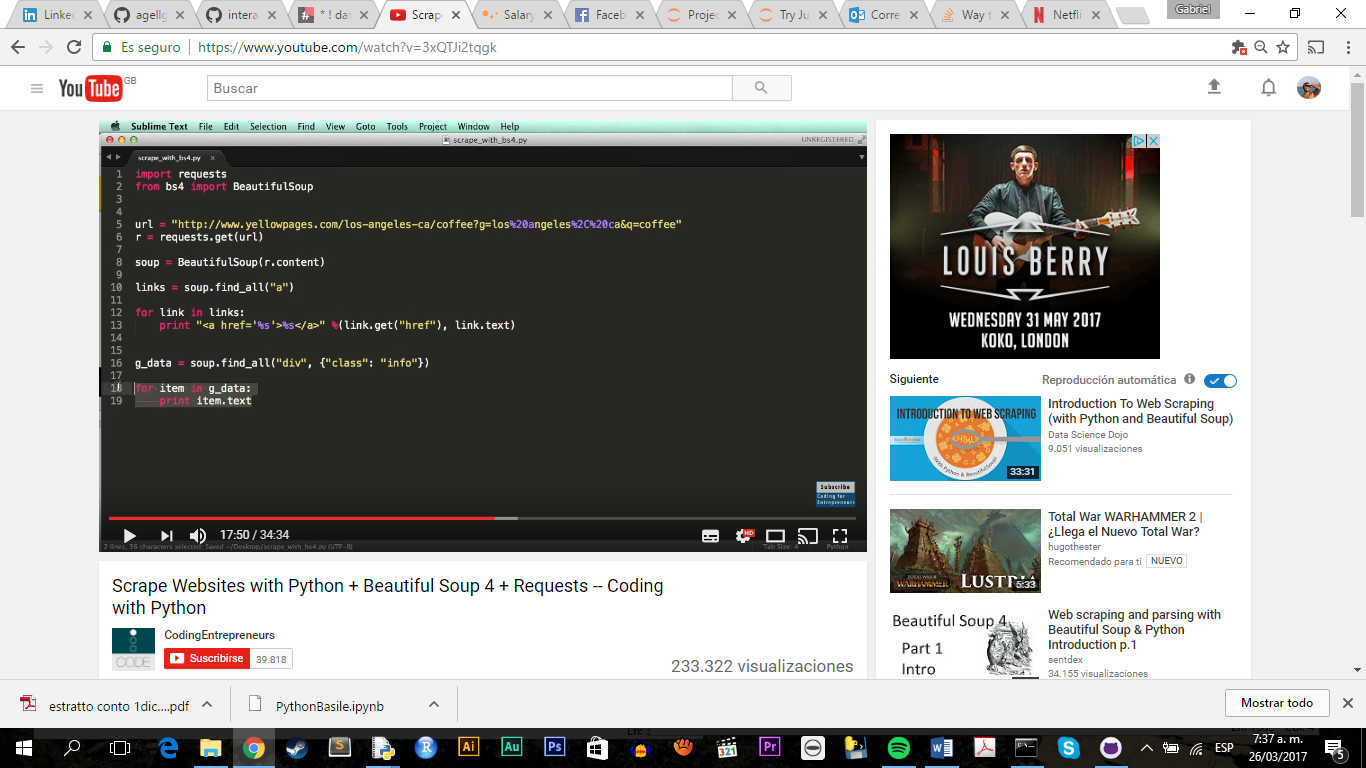
<https://campus.datacamp.com/courses/intro-to-python-for-data-science/chapter-3-functions-and-packages?ex=9>



Scraping in Python videos: self-learning

<https://www.youtube.com/watch?v=FKwicZF7xNE>

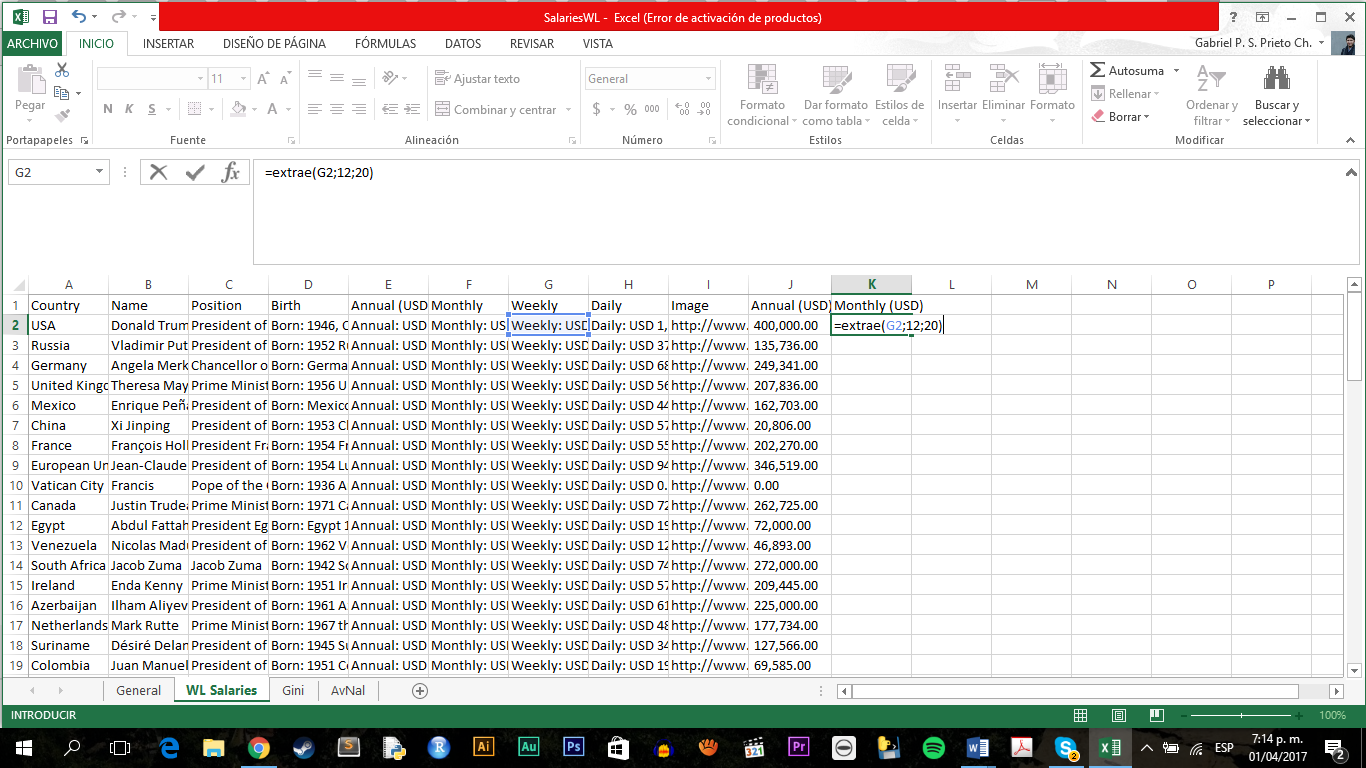
<https://www.youtube.com/watch?v=3xQTJi2tqgk>



**CLEANING**

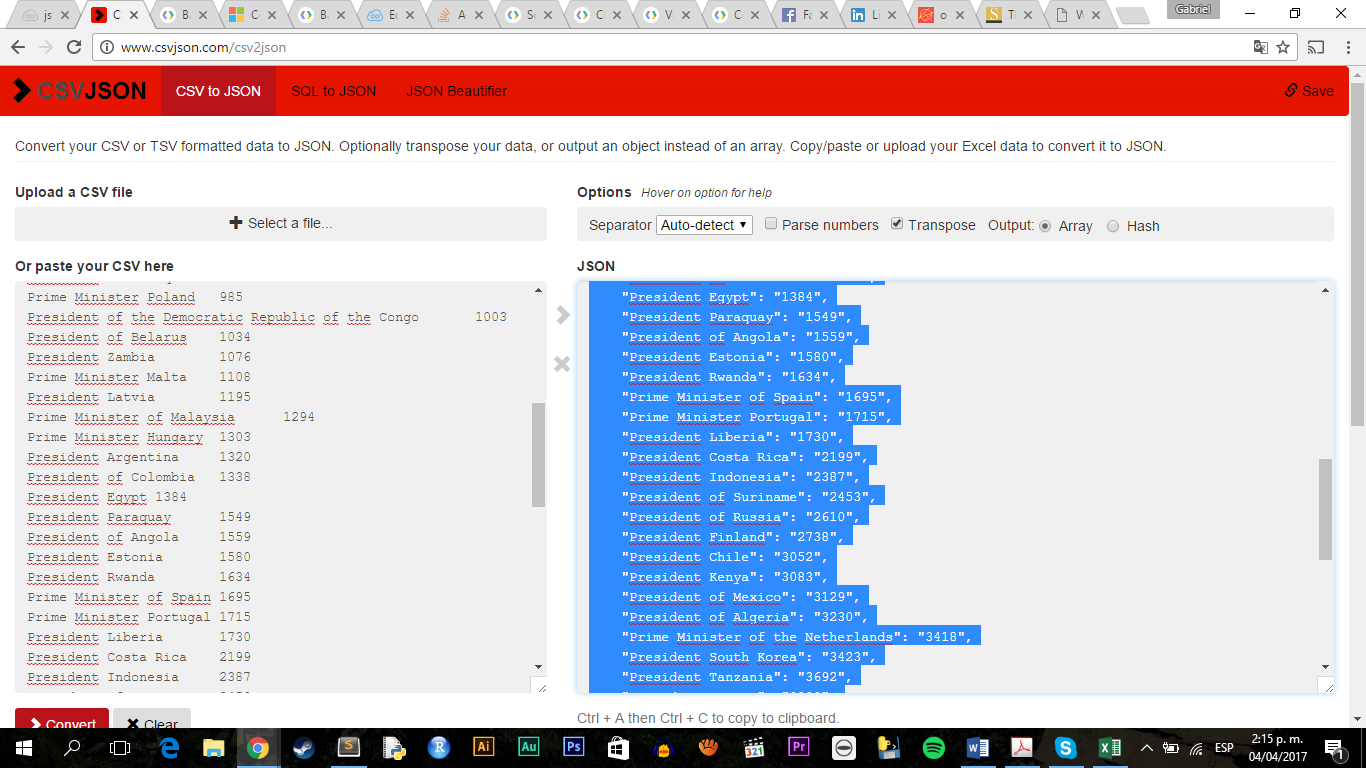
Once I managed to gather the data required for the project, I pasted it in an Excel sheet. There I started cleaning it manually; followed by this, I created a new column with a tag named ‘Country’. This was the tag used to merge all the four tables together.

For merging them together, I used the function Extract() from Excel to clean up the data related to salaries of world leaders. For the dots, and commas within the numbers, I used the FindAndReplace() function, and replaced them with nothing.



Merged all the datasets using the created tag (Country) with the Excel function VLookUp(). This function drops an error (#N/A) in the destiny cell every time the function wasn’t able to find retrievable data for a specific country. To clean it, I opted for copying the final values of the dataset in a new sheet, and using the function FindAndReplace(), to replace them with nothing.

All this process resulted in a clean, merged data set. This data was then transferred to a JSON file.



**CREATING THE CHARTS**

For the charts, I opted to use the Google API. The links below show the instructions I followed to create my charts.

<https://developers.google.com/chart/interactive/docs/roles>

<https://developers.google.com/chart/interactive/docs/drawing_charts>

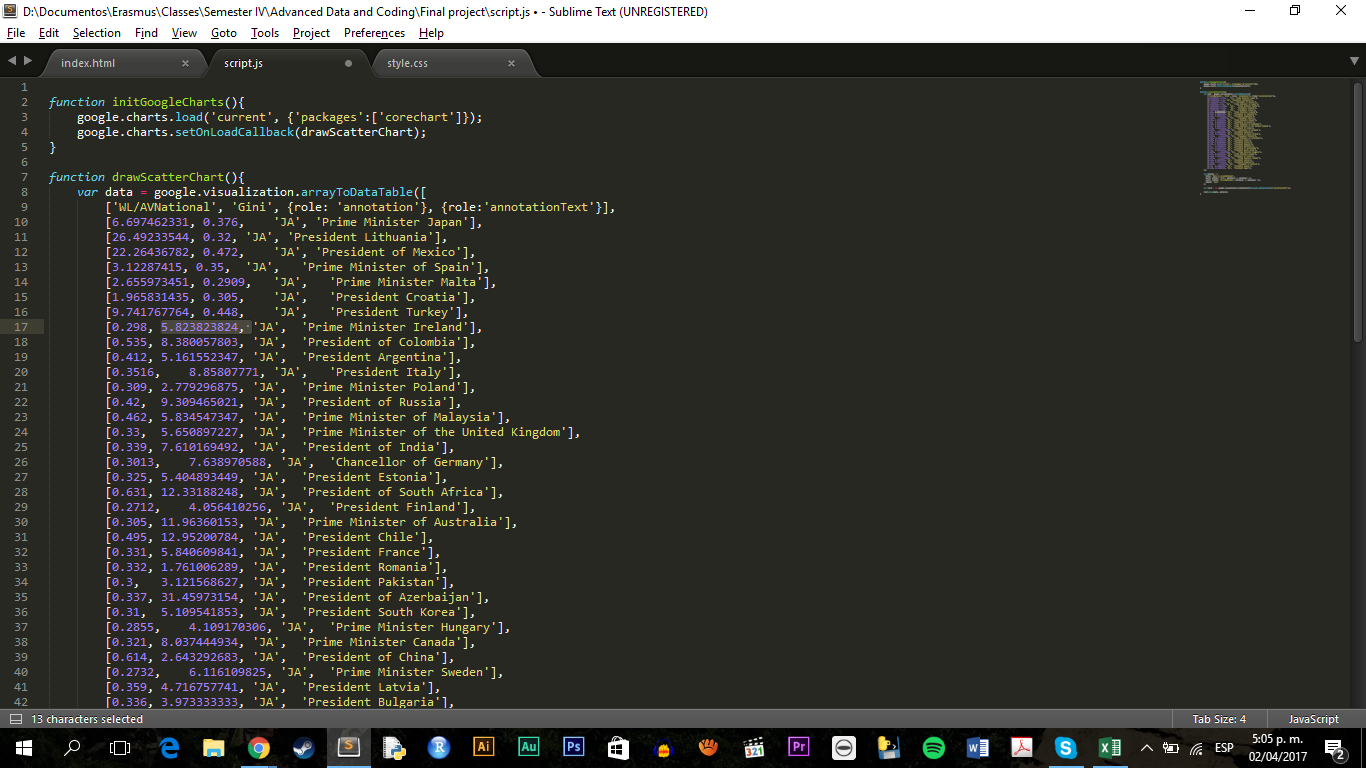
<https://developers.google.com/chart/interactive/docs/gallery/scatterchart>

<http://stackoverflow.com/questions/12848203/adding-labels-to-google-scatter-charts>

<https://developers.google.com/chart/interactive/docs/gallery/barchart>

**CREATING THE WEBPAGE**

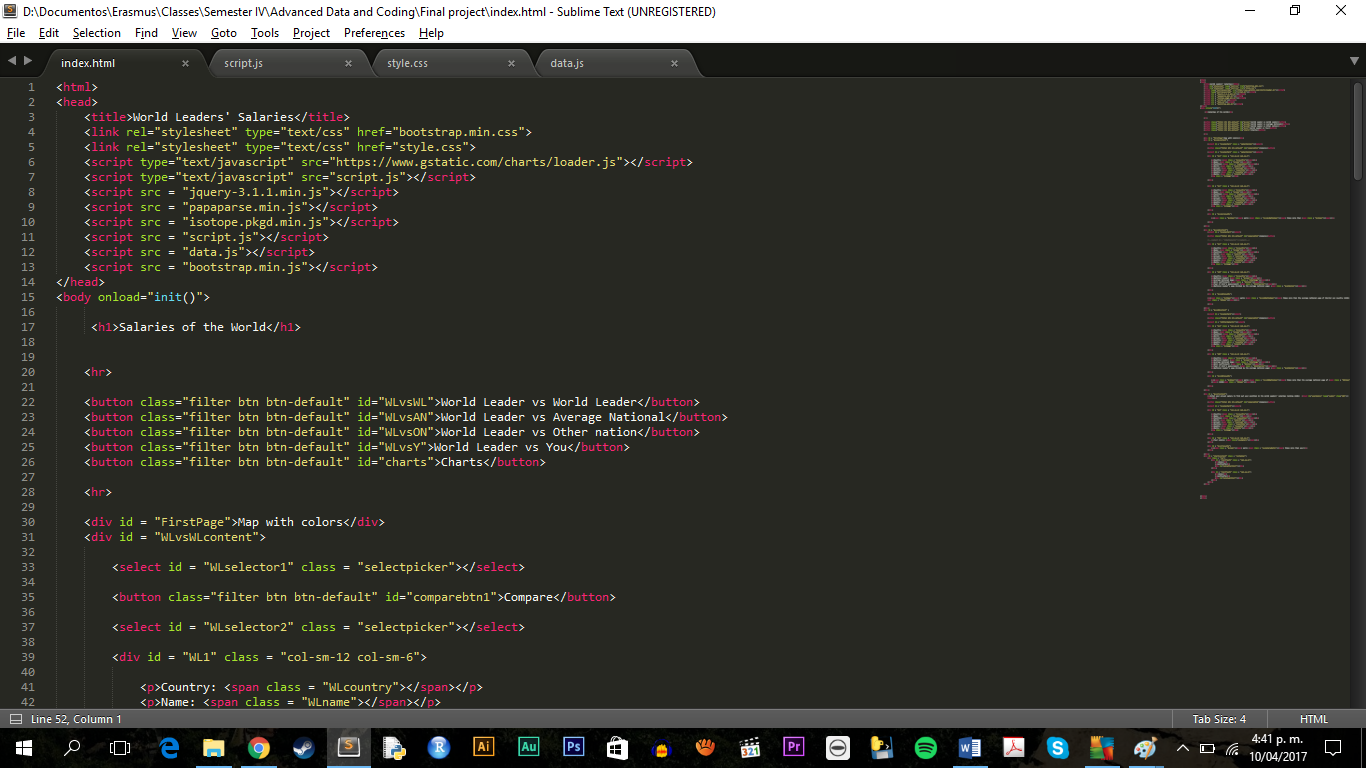
In order to visualize the project, I decided to use three coding languages: html, CSS, and Javascript.



As the main body, I used an HTML file named index; to this, I connected my other three files: Style.css, data.js (which contains the data to be analyzed in a JSON file) and script.js. I also connected the following libraries:

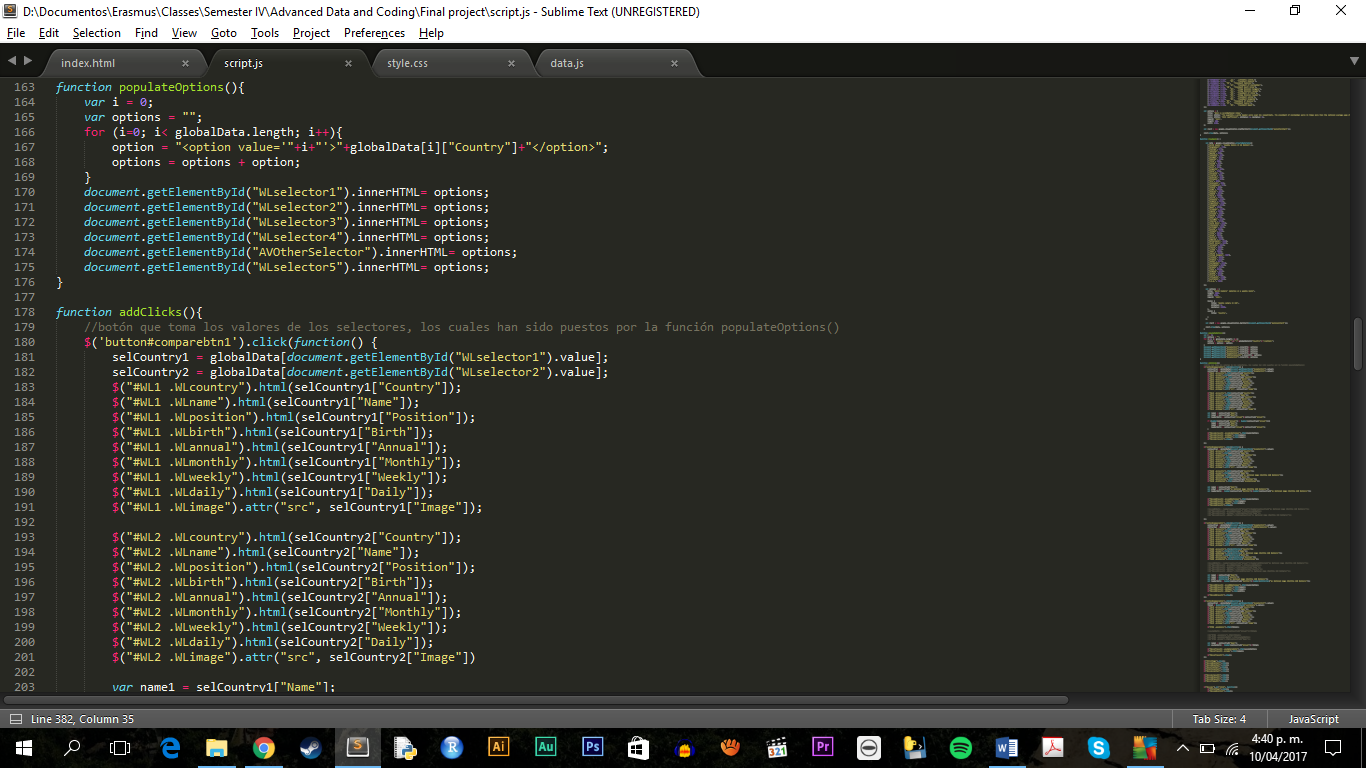
* **<link rel="stylesheet" type="text/css" href="bootstrap.min.css">**
* **<link rel="stylesheet" type="text/css" href="style.css">**
* <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
* **<script type="text/javascript" src="script.js"></script>**
* **<script src = "jquery-3.1.1.min.js"></script>**
* <script src = "papaparse.min.js"></script>
* <script src = "isotope.pkgd.min.js"></script>
* **<script src = "script.js"></script>**
* **<script src = "data.js"></script>**
* **<script src = "bootstrap.min.js"></script>**

Libraries in bold are the ones used for specifically for this project. Other libraries where also loaded in the project, as I recycled code I had been developing during class: in order to avoid problems when executing my project because of a lack of libraries, I opted to load them.



It is more likely that the project would work without the libraries that are not in bold, but I decided it was a safer choice to leave them.

In the script.js, I created 5 functions that modify html, depending on the users’ interaction with my webpage.

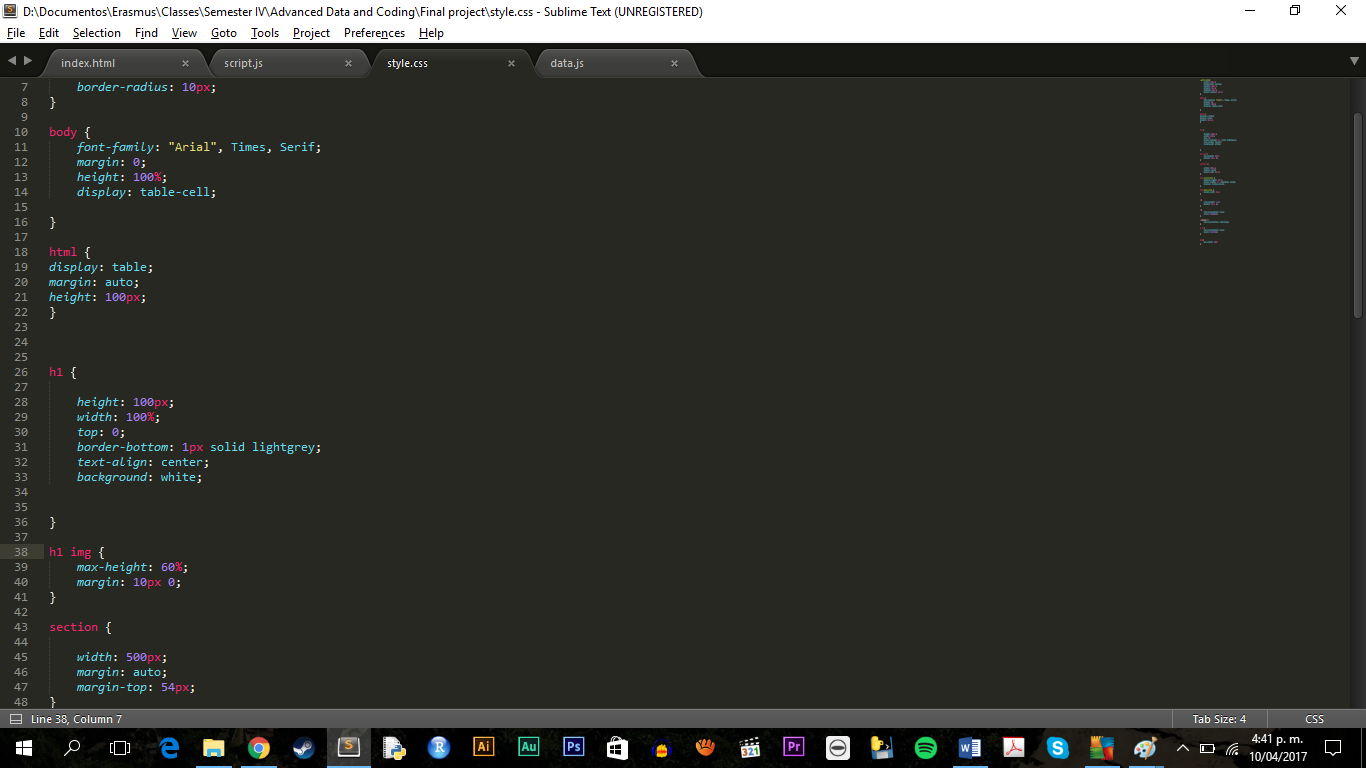


The function init() starts the other four functions. DrawScatterChart() draws the scatterplot chart, DrawBasic() draws a bar chart, PopulateOptions() populates the <select> tags on each page of my project and addClicks() is in charge of running the functionality of the buttons within the project.

Lines of code that appear commented were previous attempts of making the code work. However, after making them work, I managed to improve them. Nevertheless, I thought it would be important to leave some of the 1.0 code behind, in order to show the enhancement process this project went through.

**FINAL VISUALIZATIONS**

For the final visualizations, I opted to use mainly CSS, and Bootstrap.



The final result of the project can be seen here:

<https://github.com/agellgab/FinalDataProject>

Due to time constraints, the biggest weakness of the project is its poor visuals. Therefore, there is a great window of opportunity for redevelopment and enhancement in this matter.