Homework #4 - BeautifulSoup + Pandas

Re-submit Assignment

Due	Nov 1, 2019 by 11:59pm	**Points**	10	**Submitting**	a file upload	**File Types**	html

Complete these exercises and submit a single Jupyter Notebook file (in .html format, not .ipynb) that contains your responses by **Midnight (11:59PM) on Friday 11/1**. Late assignments will be penalized up to 2 points per day (20%), unless prior arrangements have been made to submit the assignment after the deadline.

The notebook should be well organized. Each section should be **clearly labeled with the exercise (and part) that it addresses** (e.g., Exercise #1a, #1b, #2) in a Markdown cell block. Use (clear and concise) comments as needed to help describe each step of your process. All notebook cells that contain essential steps should be executed and the output should be visible, so as to demonstrate your successful completion of the exercise. If you cannot complete an exercise in its entirety, you should make an effort to demonstrate your intermediate progress in order to maximize partial credit, and move forward as best as possible. You may submit any written answers to the exercises in the notebook as text cells.

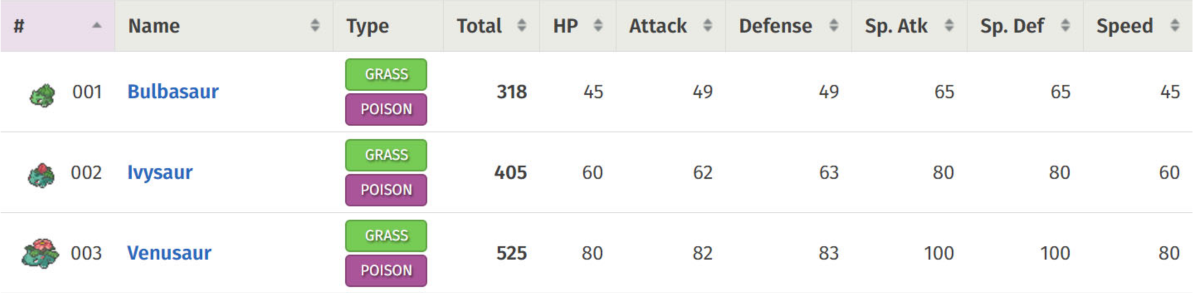
Academic Integrity: Each student is expected to submit his or her own original work. You may collaborate with your classmates on the concepts of the homework assignment, but you should not submit the same documentation for any part of the assignment. Submissions that contain significant similarities will be reported directly to the Office of Student Conduct.

Background

We will be using BeautifulSoup and Pandas to extract data on pokemon from **https://pokemondb.net** **(https://pokemondb.net/)** . The data will come both from the main database page, as well as the individual pokemon pages.

Exercise #1 - Scraping the pokedex

The pokedex table containing basic data about each pokemon is available at **https://pokemondb.net/pokedex/all** **(https://pokemondb.net/pokedex/all)**



#	Name	Type	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed
001	Bulbasaur	GRASS POISON	318	45	49	49	65	65	45
002	Ivysaur	GRASS POISON	405	60	62	63	80	80	60
003	Venusaur	GRASS POISON	525	80	82	83	100	100	80

Step 1.1: Use BeautifulSoup to extract all the table rows as a list. How many rows are there (including the header row)?

Step 1.2: Save the first row of the table (*bulbasaur*) as a variable. Using .find(), and .findall(), extract and print the following contents from that row. You may leave numbers as strings.

- The name of the pokemon
- The url to the pokemon's page
- The type or types (as a string separated by spaces, e.g. "Fire Flying")
- The total points
- In a single list (via appending): ID Number, HP, Attack, Defense, Sp. Atk, Sp. Def, Speed (Hint: If you look carefully at the class names for these columns, you will see why those values should be processed together.)

Step 1.3: Generalize step 1.2. Define a function that takes in a row of the pokedex table and returns it as a DataFrame with a single row. Create a single DataFrame by appending these rows. (Appending the data to a list or dictionary and then creating a DataFrame is also acceptable.) Make sure you skip the header row.

Exercise #2 - Cleaning the Pokedex

Step 2.1: Add column names to the DataFrame. Convert strings to numeric where appropriate. Make the ID number the first column in the DataFrame if it is not already.

Step 2.2: Notice that the pokemon types are not mutually exclusive. (Pokemon may have more than one type.) Create 18 dummy variables for each type of pokemon.

Step 2.3: Remove duplicate values of pokemon based on the URL. (See Charizard as an example.) Keep the first observation in the case of a duplicate. Print the number of rows in the deduplicated dataset.

Step 2.4: For the next exercise, we wish to create a sample of the pokemon. (Note: this sample is not a true random sample since the data is already sorted on ID number.) Add a dummy variable to the DataFrame called "sample" that tags every 4th pokemon to be included in the sample. For example, if the pokemon were [A, B, C, D, E, F, G, H, I], pokemon D and H would be in the sample. (Suggested Hint: Use row numbers/indices and modular arithmetic.)

Exercise #3 - Scraping Individual Pages

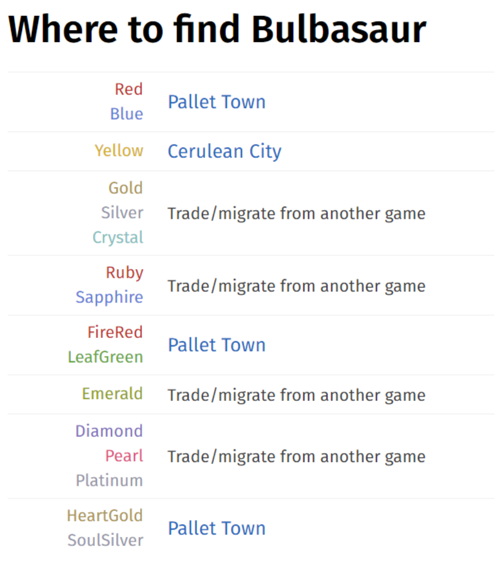
In steps 3.1-3.3, use the Bulbasaur page as an example **[https://pokemondb.net/pokedex/bulbasaur (https://pokemondb.net/pokedex/bulbasaur)](https://pokemondb.net/pokedex/bulbasaur)**

Step 3.1: Scrape the main image for Bulbasaur in a general way that could be applied to other pokemon pages by searching for the relevant tag and extracting the image URL. Display the image in your Jupyter notebook using code. (Take a look at last year's exam for an example of how to do this using the Image module from iPython.display.)

Step 3.2: Extract the location table. Use pd.read_html() to extract all of the tables from the Bulbasaur page. To do this, you must be a little "sneaky" because pokemondb does not accept the headers (browser info) passed by pandas. Use the code below which utilizes requests. Then, manually look through the returned tables until you find the table that contains the locations for Bulbasaur. (Hint: To save time, start at index 10 until you find it.)

```
tables = pd.read_html(requests.get(url, headers={'User-agent': 'Mozilla/5.0'}).text)
```

Show the desired DataFrame for the locations table in your notebook.



Where to find Bulbasaur

Red Blue	Pallet Town
Yellow	Cerulean City
Gold Silver Crystal	Trade/migrate from another game
Ruby Sapphire	Trade/migrate from another game
FireRed LeafGreen	Pallet Town
Emerald	Trade/migrate from another game
Diamond Pearl Platinum	Trade/migrate from another game
HeartGold SoulSilver	Pallet Town

Step 3.3: Transpose the DataFrame such that each column is a video game and each row/cell is the location where you find Bulbasaur in that game. (e.g. the column 'RedBlue' contains 'Pallet Town').

Step 3.4: Across all the pokemon pages, the location table is **almost*** always the second to last table on the page. (Use an index of -2.) Extract the location table and transpose it for all the pokemon in the sample (see step 2.4). Make sure you include wait time in your code or some other method to ensure that you are not blocked from the site after you request each page. Unfortunately, the location table is not in the same format for every page, so we will be extracting **only** the information for the **X and Y games** in the following steps.

- Check if the the column 'XY' is in the DataFrame. If so, create a new DataFrame with only the name or URL for the pokemon and the 'XY' column. Append that DataFrame to a list to concatenate with the other pokemon that have the XY location column.
- Create a single DataFrame that contains the name or URL of the pokemon and the XY location.
- Append all the sample pokemon and their XY locations to a single DataFrame.

You should get 141 sample pokemon that meet this criteria.

*One sample pokemon (Meltan) has a page that will not work using this method. You can either remove Meltan from your sample or wrap your scraping code in a try/except statement.

Exercise #4 - Analysis

Step 4.1: Use the full sample of pokemon from the pokedex DataFrame. Create a table that shows the average attack and defense for each type. Each type should be a row. Average attack and defense should be columns. Which type has the highest and lowest average attack? Average defense? (Note: There are multiple approaches you could use here given the categories are not mutually exclusive.)

Step 4.2: Join the pokedex data to the location DataFrame created in Step 3.4. (Exclude pokemon that are not in the sample.) For the locations in pokemon X/Y, calculate the average total points for each location. Which location has the highest average total point score?

HW #4 Rubric