

Project 4: Web Scraping with BeautifulSoup



CS3753/CS5163: Data Science Summer 2023

Instructor: Dr. Mohammad Imran Chowdhury

Total Points: 80

Due: 07/25/2023 11:59 PM

In this project, you'll gain hands-on experience in Web Scraping with BeautifulSoup. To achieve this real-life experience on Web Scraping, we will be taking a webpage from Wikipedia with the URL (https://en.wikipedia.org/wiki/List_of_states_and_territories_of_the_United_States) as an example. This page contains the list of states in the U.S., their population, and other details. We will try to get the **names of the states** and the **population columns of the table**.

Flag, name and postal abbreviation ^[8]	Cities		Ratification or admission ^[4]	Population ^[10]	Total area ^[11]		Land area ^[11]		Water area ^[11]		Number of Reprs.	
	Capital	Largest ^[12]			mi ²	km ²	mi ²	km ²	mi ²	km ²		
 Alabama	AL	Montgomery	Huntsville	Dec 14, 1819	5,024,279	52,420	135,767	50,645	131,171	1,775	4,597	7
 Alaska	AK	Juneau	Anchorage	Jan 3, 1959	733,391	665,384	1,723,337	570,641	1,477,953	94,743	245,384	1
 Arizona	AZ	Phoenix		Feb 14, 1912	7,151,502	113,990	295,234	113,594	294,207	396	1,026	9
 Arkansas	AR	Little Rock		Jun 15, 1836	3,011,524	53,179	137,732	52,035	134,771	1,143	2,961	4
 California	CA	Sacramento	Los Angeles	Sep 9, 1850	39,538,223	163,695	423,967	155,779	403,466	7,916	20,501	52
 Colorado	CO	Denver		Aug 1, 1876	5,773,714	104,094	269,601	103,642	268,431	452	1,170	8
 Connecticut	CT	Hartford	Bridgeport	Jan 9, 1788	3,605,944	5,543	14,357	4,842	12,542	701	1,816	5
 Delaware	DE	Dover	Wilmington	Dec 7, 1787	989,948	2,489	6,446	1,949	5,047	540	1,399	1
 Florida	FL	Tallahassee	Jacksonville	Mar 3, 1845	21,538,187	65,758	170,312	53,625	138,887	12,133	31,424	28
 Georgia	GA	Atlanta		Jan 2, 1788	10,711,908	59,425	153,910	57,513	148,959	1,912	4,951	14
 Hawaii	HI	Honolulu		Aug 21, 1959	1,455,271	10,932	28,313	6,423	16,635	4,509	11,678	2
 Idaho	ID	Boise		Jul 3, 1890	1,839,106	83,569	216,443	82,643	214,045	926	2,398	2
 Illinois	IL	Springfield	Chicago	Dec 3, 1818	12,812,508	57,914	149,995	55,519	143,793	2,395	6,202	17
 Indiana	IN	Indianapolis		Dec 11, 1816	6,785,528	36,420	94,326	35,826	92,789	593	1,537	9
 Iowa	IA	Des Moines		Dec 28, 1846	3,190,369	56,273	145,746	55,857	144,669	416	1,077	4
Kansas	KS	Topeka	Wichita	Jan 29, 1861	2,937,880	82,278	213,100	81,759	211,754	520	1,346	4

Task 1 (15 points): The initial step is to identify the text or area of the webpage which is to be scraped. To find that, simply select the area of the page, right-click, and then click on inspect. You can see that the element I am looking for is in the table with the class name “**wikitable sortable plainrowheaders**” and it is a string of a <a> tag that is nested inside the <th> tag.

```
<table class="wikitable sortable plainrowheaders" style="text-align: center;">
  <caption>States of the United States of America </caption>
  <tbody>
    <tr></tr>
    <tr></tr>
    <tr>
      <th scope="row">Alabama
        <span class="flagicon"></span>
        <a href="/wiki/Alabama" title="Alabama">Alabama</a>
      </th>
      <td>AL </td>
      <td></td>
      <td></td>
      <td></td>
      <td style="text-align:right;" data-sort-value="701135767358042980">52,420 </td>
      <td style="text-align:right;" data-sort-value="701135767358042980">135,767 </td>
      <td style="text-align:right;" data-sort-value="701131170802544043">50,645 </td>
      <td style="text-align:right;" data-sort-value="701131170802544043">131,171 </td>
      <td style="text-align:right;" data-sort-value="7009459655549893771">1,775 </td>
      <td style="text-align:right;" data-sort-value="7009459655549893771">4,597 </td>
    </tr>
  </tbody>
</table>
```

In the next step, we will use a get request by passing the URL of the webpage that is to be parsed as shown in class. Further, we create a BeautifulSoup object with “**html.parser**”. (15 points)

Task 2 (25 points): Then, we use the BeautifulSoup object created above and collect the required table data by using the class name:

This requires you to use the **find()** method. Such as **scraped_doc.find(“table”, class_ = “wikitable sortable plainrowheaders”)**

and we then extract all the <th> tags in our table and finally get the text inside the <a> tags.

This requires you to use the **find_all('th')** and **find_all('a')** methods respectively. Once you print the text inside the <a> tag then the output should be as follows: (15 points)

```
for i in a_links:
    names.append(i.string)
print(names)
```

```
['postal abbreviation', '[8]', '[A]', '[10]', '[11]', '[11]', '[11]', None, '[12]', 'Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California', 'Colorado', 'Connecticut', 'Delaware', 'Florida', 'Georgia', 'Hawaii', 'Idaho', 'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', '[B]', 'Louisiana', 'Maine', 'Maryland', 'Massachusetts', '[B]', 'Michigan', 'Minnesota', 'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada', 'New Hampshire', 'New Jersey', 'New Mexico', 'New York', 'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon', 'Pennsylvania', '[B]', 'Rhode Island', 'South Carolina', 'South Dakota', 'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', '[B]', 'Washington', 'West Virginia', 'Wisconsin', 'Wyoming']
```

In the above result, you can observe that our list starts from index 9, and also we have a few [B] in between. We will prepare the final list by removing the unwanted strings and the final list should be as follows: (10 points)

```
print(states, len(states))
```

```
['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California', 'Colorado', 'Connecticut', 'Delaware', 'Florida', 'Georgia', 'Hawaii', 'Idaho', 'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana', 'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota', 'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada', 'New Hampshire', 'New Jersey', 'New Mexico', 'New York', 'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon', 'Pennsylvania', 'Rhode Island', 'South Carolina', 'South Dakota', 'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington', 'West Virginia', 'Wisconsin', 'Wyoming']
50
```

Task 3 (25 points): In a similar way, **we will now try to scrape the population columns** from the same table. When I inspect the column element, I can find that it is contained inside the <div> tags as shown below:

```
▼<tr>
  ▶<th scope="row">...</th>
    <td>AK </td>
  ▶<td>...</td>
  ▶<td>...</td>
  ▶<td>...</td>
  ▼<td>
    <div style="float:right;">733,391</div> == $0
  </td>
  <td style="text-align:right;" data-sort-value="7012172333675240733">665,384 </td>
  <td style="text-align:right;" data-sort-value="7012172333675240733">1,723,337 </td>
  <td style="text-align:right;" data-sort-value="7012147795327577084">570,641 </td>
  <td style="text-align:right;" data-sort-value="7012147795327577084">1,477,953 </td>
  <td style="text-align:right;" data-sort-value="7011245383502536374">94,743 </td>
  <td style="text-align:right;" data-sort-value="7011245383502536374">245,384 </td>
  ▶<td>...</td>
</tr>
▶<tr>...</tr>
▶<tr>...</tr>
▶<tr>...</tr>
▶</table> </table>
```

After scraping the population columns from the same table of the scraped page the output should be as follows: **(15 points)**

```
[ '5,024,279', '7', '733,391', '1', '7,151,502', '9', '3,011,524', '4', '39,538,223', '52', '5,773,714', '8', '3,605,944', '5', '989,948', '1', '21,538,187', '28', '10,711,908', '14', '1,455,271', '2', '1,839,106', '2', '12,812,508', '17', '6,785,528', '9', '3,190,369', '4', '2,937,880', '4', '4,505,836', '6', '4,657,757', '6', '1,362,359', '2', '6,177,224', '8', '7,029,917', '9', '10,077,331', '13', '5,706,494', '8', '2,961,279', '4', '6,154,913', '8', '1,084,225', '2', '1,961,504', '3', '3,104,614', '4', '1,377,529', '2', '9,288,994', '12', '2,117,522', '3', '20,201,249', '26', '10,439,388', '14', '779,094', '1', '11,799,448', '15', '3,959,353', '5', '4,237,256', '6', '13,002,700', '17', '1,097,379', '2', '5,118,425', '7', '886,667', '1', '6,910,840', '9', '29,145,505', '38', '3,271,616', '4', '643,077', '1', '8,631,393', '11', '7,705,281', '10', '1,793,716', '2', '5,893,718', '8', '576,851', '1']
```

Similarly, as before in Task2, here now, we will remove the unwanted strings in between. And the final result goes here: **(10 points)**

```
[ '5,024,279', '733,391', '7,151,502', '3,011,524', '39,538,223', '5,773,714', '3,605,944', '989,948', '21,538,187', '10,711,908', '1,455,271', '1,839,106', '12,812,508', '6,785,528', '3,190,369', '2,937,880', '4,505,836', '4,657,757', '1,362,359', '6,177,224', '7,029,917', '10,077,331', '5,706,494', '2,961,279', '6,154,913', '1,084,225', '1,961,504', '3,104,614', '1,377,529', '9,288,994', '2,117,522', '20,201,249', '10,439,388', '779,094', '11,799,448', '3,959,353', '4,237,256', '13,002,700', '1,097,379', '5,118,425', '886,667', '6,910,840', '29,145,505', '3,271,616', '643,077', '8,631,393', '7,705,281', '1,793,716', '5,893,718', '576,851']
```

Task 4 (15 points): Writing Data To CSV.

To do this, you can use the Pandas Dataframe object to save the data i.e., states column and population column. When you print the **df object** then the output should as follows: **(10 points)**

```
print(df)
```

	state	population			
0	Alabama	5,024,279	25	Montana	1,084,225
1	Alaska	733,391	26	Nebraska	1,961,504
2	Arizona	7,151,502	27	Nevada	3,104,614
3	Arkansas	3,011,524	28	New Hampshire	1,377,529
4	California	39,538,223	29	New Jersey	9,288,994
5	Colorado	5,773,714	30	New Mexico	2,117,522
6	Connecticut	3,605,944	31	New York	20,201,249
7	Delaware	989,948	32	North Carolina	10,439,388
8	Florida	21,538,187	33	North Dakota	779,094
9	Georgia	10,711,908	34	Ohio	11,799,448
10	Hawaii	1,455,271	35	Oklahoma	3,959,353
11	Idaho	1,839,106	36	Oregon	4,237,256
12	Illinois	12,812,508	37	Pennsylvania	13,002,700
13	Indiana	6,785,528	38	Rhode Island	1,097,379
14	Iowa	3,190,369	39	South Carolina	5,118,425
15	Kansas	2,937,880	40	South Dakota	886,667
16	Kentucky	4,505,836	41	Tennessee	6,910,840
17	Louisiana	4,657,757	42	Texas	29,145,505
18	Maine	1,362,359	43	Utah	3,271,616
19	Maryland	6,177,224	44	Vermont	643,077
20	Massachusetts	7,029,917	45	Virginia	8,631,393
21	Michigan	10,077,331	46	Washington	7,705,281
22	Minnesota	5,706,494	47	West Virginia	1,793,716
23	Mississippi	2,961,279	48	Wisconsin	5,893,718
24	Missouri	6,154,913	49	Wyoming	576,851

We then write the data frame to the CSV file using the line of code below. **(5 points)**

```
df.to_csv('us_info.csv')
```

The submission grading rubric is as follows (points out of 80 total):

Project element	Points
Task 1	15
Task 2	25
Task 3	25
Task 4	15

Submission Instructions: Create a compressed file (.zip or .tar.gz files are accepted) with your all source files such as .ipynb files and data files. Generally speaking, to complete Task1 through Task4, you just need one .ipynb file. But it's better to submit everything as a compressed zip file. Submit the compressed zip file to the "Project Submissions" area on the Blackboard course website.

Late submission policy: As described in the syllabus, any late submission will be penalized with 10% off after each 24 hours late. For example, an assignment worth 100 points turned in 2 days late will receive a 20 point penalty. Assignments turned in 5 or more days after the due date will receive a grade of 0.