AVBnr&utm_medium=Banner&utm_campaign=DSWeek)

Introduction

The need and importance of extracting data from the web is becoming increasingly loud and clear. Every few weeks, I find myself in a situation where we need to extract data from the web. For example, last week we were thinking of creating an index of hotness and sentiment about various data science courses available on the internet. This would not only require finding out new courses, but also scrape the web for their reviews and then summarizing them in a few metrics! This is one of the problems / products, whose efficacy depends more on web scrapping and information extraction (data collection) than the techniques used to summarize the data.

Analytics Vidhya
DataFest Closing Ceremony
Come & meet our Datavengers
Time: 9 PM to 10 PM (IST)
(Webinar) | 10 May 2017
DATAFEST 2017
Register Now

(https://datahack.analyticsvidhya.com/contest/datafest-closing-ceremony/)

Ways to extract information from web

There are several ways to extract information from the web. Use of API (https://en.wikipedia.org/wiki/Application_programming_interface)s being probably the best way to extract data from a website. Almost all large websites like Twitter, Facebook, Google, Twitter, StackOverflow provide APIs to access their data in a more structured manner. **If you can get what you need through an API, it is almost always preferred approach over web scrapping.** This is because if you are getting access to structured data from the provider, why would you want to create an engine to extract the same information.

Sadly, not all websites provide an API. Some do it because they do not want the readers to extract huge information in structured way, while others don’t provide APIs due to lack of technical knowledge. What do you do in these cases? Well, we need to scrape the website to fetch the information.

There might be a few other ways like RSS feeds, but they are limited in their use and hence I am not including them in the discussion here.

(https://www.analyticsvidhya.com/wp-content/uploads/2015/10/web-scraping.jpg)

What is Web Scraping?

Web scraping is a computer software technique of extracting information from websites. This technique mostly focuses on the transformation of unstructured data (HTML format) on the web into structured data (database or spreadsheet).

You can perform web scrapping in various ways, including use of Google Docs to almost every programming language. I would resort to Python because of its ease and rich eocsystem. It has a library known as 'BeautifulSoup' which assists this task. In this article, I'll show you the easiest way to learn web scraping using python programming.

For those of you, who need a non-programming way to extract information out of web pages, you can also look at import.io (https://import.io/) . It provides a GUI driven interface to perform all basic web scraping operations. The hackers can continue to read this article!

Analytics Vidhya
DataFest Closing Ceremony
Come & meet our Datavengers
Time: 9 PM to 10 PM (IST)
(Webinar) | 10 May 2017
DATAFEST 2017
Register Now

(https://datahack.analyticsvidhya.com/contest/datafest-closing-ceremony/)

Libraries required for web scraping

As we know, python is a open source programming language. You may find many libraries to perform one function. Hence, it is necessary to find the best to use library. I prefer **BeautifulSoup** (python library), since it is easy and intuitive to work on. Precisely, I'll use two Python modules for scraping data:

- **Urllib2**: It is a Python module which can be used for fetching URLs. It defines functions and classes to help with URL actions (basic and digest authentication, redirections, cookies, etc). For more detail refer to the documentation page (https://docs.python.org/2/library/urllib2.html).
- **BeautifulSoup:** It is an incredible tool for pulling out information from a webpage. You can use it to extract tables, lists, paragraph and you can also put filters to extract information from web pages. In this article, we will use latest version BeautifulSoup 4. You can look at the installation instruction in its documentation page (http://www.crummy.com/software/BeautifulSoup/bs4/doc/).

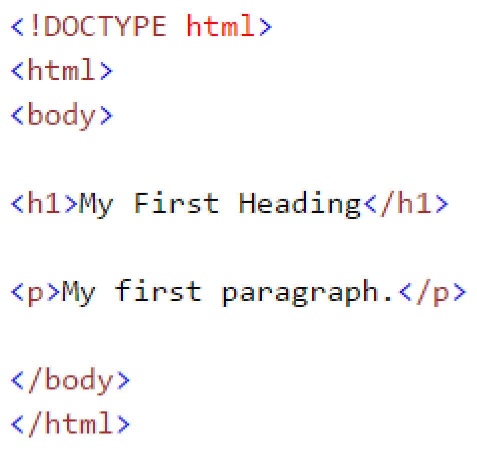
BeautifulSoup does not fetch the web page for us. That's why, I use urllib2 in combination with the BeautifulSoup library.

Python has several other options for HTML scraping in addition to BeatifulSoup. Here are some others:

- mechanize (http://wwwsearch.sourceforge.net/mechanize/)
- scrapemark (http://arshaw.com/scrapemark/)
- scrapy (http://scrapy.org/)

Basics – Get familiar with HTML (Tags)

While performing web scarping, we deal with html tags. Thus, we must have good understanding of them. If you already know basics of HTML, you can skip this section. Below is the basic syntax of HTML:



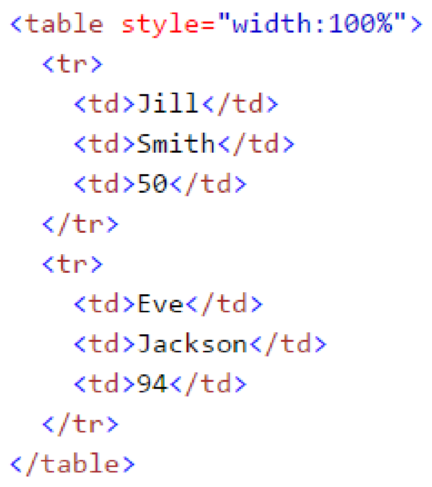
```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

(https://www.analyticsvidhya.com/wp-content/uploads/2015/10/HTML.png)This syntax has various tags as elaborated below:

1. **<!DOCTYPE html> :** HTML documents must start with a type declaration
2. HTML document is contained between **<html>** and **</html>**
3. The visible part of the HTML document is between **<body>** and **</body>**
4. HTML headings are defined with the **<h1>** to **<h6>** tags
5. HTML paragraphs are defined with the **<p>** tag

Other useful HTML tags are:

1. HTML links are defined with the **<a>** tag, "<a href="http://www.test.com">This is a link for test.com</a>"
2. HTML tables are defined with<Table>, row as <tr> and rows are divided into data as <td>



```
<table style="width:100%">
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

3. HTML list starts with <ul> (unordered) and <ol> (ordered). Each item of list starts with <li>

List

(https://www.analyticsvidhya.com/wp-content/uploads/2015/10/List.png)

If you are new to this HTML tags, I would also recommend you to refer HTML tutorial from W3schools (http://www.w3schools.com/html/). This will give you a clear understanding about HTML tags.

Scrapping a web Page using BeautifulSoup

Here, I am scraping data from a Wikipedia page (https://en.wikipedia.org/wiki/List_of_state_and_union_territory_capitals_in_India). Our final goal is to extract list of state, union territory capitals in India. And some basic detail like establishment, former capital and others form this wikipedia page (https://en.wikipedia.org/wiki/List_of_state_and_union_territory_capitals_in_India). Let's learn with doing this project step wise step:

1. **Import necessary libraries:**

```
#import the library used to query a website
import urllib2
```

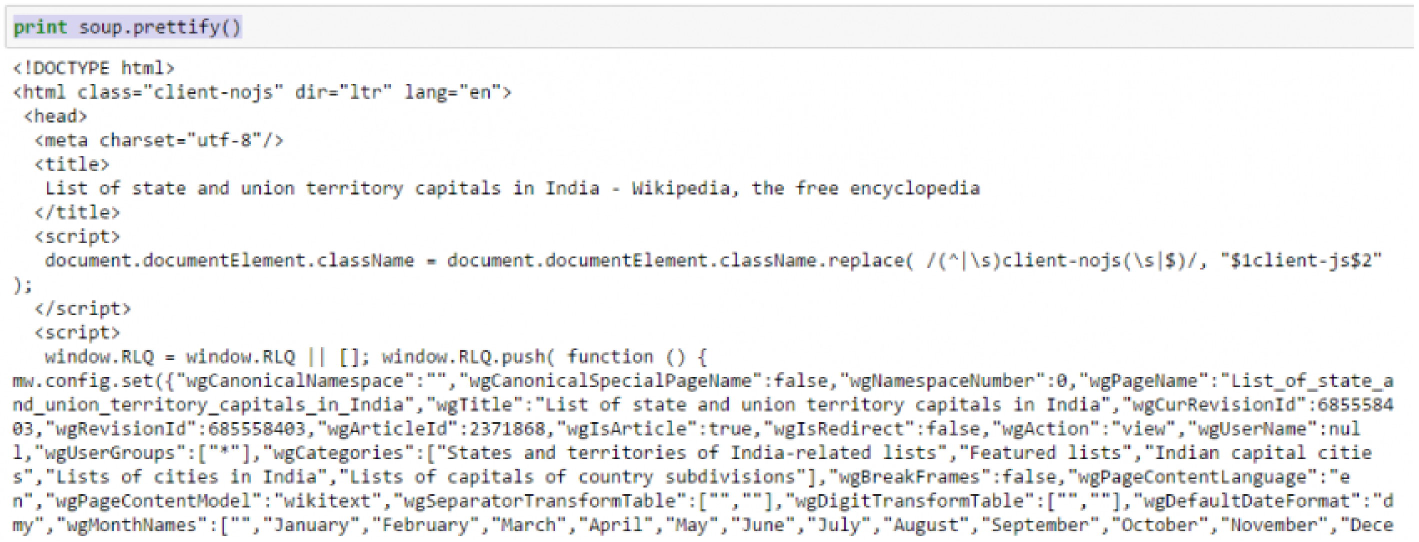
```
#specify the url
wiki = "https://en.wikipedia.org/wiki/List_of_state_and_union_territory_capitals_in_India"
```

```
#Query the website and return the html to the variable 'page'
page = urllib2.urlopen(wiki)
```

```
#import the Beautiful soup functions to parse the data returned from the website
from bs4 import BeautifulSoup
```

```
#Parse the html in the 'page' variable, and store it in Beautiful Soup format
soup = BeautifulSoup(page)
```

2. **Use function "prettify" to look at nested structure of HTML page**



```
print soup.prettify()
```

```
<!DOCTYPE html>
<html class="client-nojs" dir="ltr" lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   List of state and union territory capitals in India - Wikipedia, the free encyclopedia
  </title>
  <script>
   document.documentElement.className = document.documentElement.className.replace( /(^|\s)client-nojs(\s|$)/, "$1client-js$2"
);
  </script>
  <script>
   window.RLQ = window.RLQ || []; window.RLQ.push( function () {
mw.config.set({"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNumber":0,"wgPageName":"List_of_state_a
nd_union_territory_capitals_in_India","wgTitle":"List of state and union territory capitals in India","wgCurRevisionId":6855584
03,"wgRevisionId":685558403,"wgArticleId":2371868,"wgIsArticle":true,"wgIsRedirect":false,"wgAction":"view","wgUserName":nul
l,"wgUserGroups":["*"],"wgCategories":["States and territories of India-related lists","Featured lists","Indian capital citie
s","Lists of cities in India","Lists of capitals of country subdivisions"],"wgBreakFrames":false,"wgPageContentLanguage":"e
n","wgPageContentModel":"wikitext","wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"d
my","wgMonthNames":["","January","February","March","April","May","June","July","August","September","October","November","Dece
```

(https://www.analyticsvidhya.com/wp-content/uploads/2015/10/FirstP.png)Above, you can see that structure of the HTML tags. This will help you to know about different available tags and how can you play with these to extract information.

3. **Work with HTML tags**

a. **soup.<tag>:** Return content between opening and closing tag including tag.

```
In[30]:soup.title

Out[30]:<title>List of state and union territory capitals in India - Wikipedia, the free
encyclopedia</title>
```

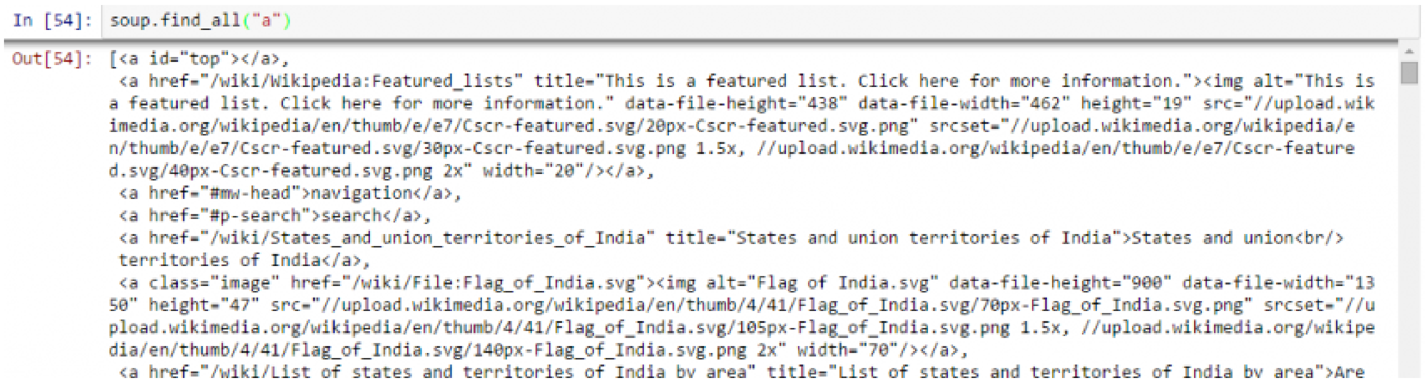
b. **soup.<tag>.string:** Return string within given tag

```
In [38]:soup.title.string

Out[38]:u'List of state and union territory capitals in India - Wikipedia, the free ency
clopedia'
```

c. **Find all the links within page's <a> tags::** We know that, we can tag a link using tag "<a>". So, we should go with option **soup.a** and it should return the links available in the web page. Let's do it.

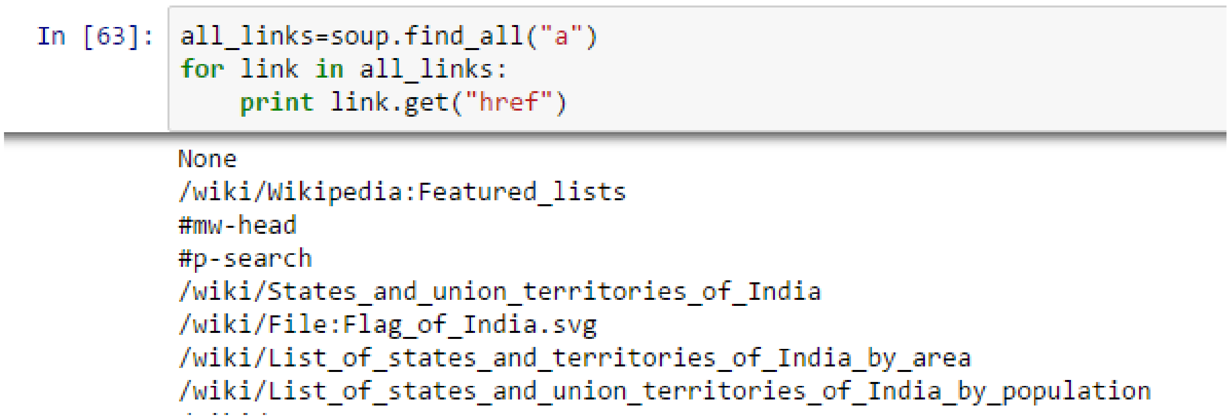
```
In [40]:soup.a

Out[40]:<a id="top"></a>
```

Above, you can see that, we have only one output. Now to extract all the links within <a>, we will use “**find_all().**



```
In [54]: soup.find_all("a")

Out[54]: [<a id="top"></a>,
          <a href="/wiki/Wikipedia:Featured_lists" title="This is a featured list. Click here for more information."><img alt="This is
         a featured list. Click here for more information." data-file-height="438" data-file-width="462" height="19" src="//upload.wik
         imedia.org/wikipedia/en/thumb/e/e7/Cscr-featured.svg/20px-Cscr-featured.svg.png" srcset="//upload.wikimedia.org/wikipedia/e
         n/thumb/e/e7/Cscr-featured.svg/30px-Cscr-featured.svg.png 1.5x, //upload.wikimedia.org/wikipedia/en/thumb/e/e7/Cscr-feature
         d.svg/40px-Cscr-featured.svg.png 2x" width="20"/></a>,
          <a href="#mw-head">navigation</a>,
          <a href="#p-search">search</a>,
          <a href="/wiki/States_and_union_territories_of_India" title="States and union territories of India">States and union<br/>
          territories of India</a>,
          <a class="image" href="/wiki/File:Flag_of_India.svg"><img alt="Flag of India.svg" data-file-height="900" data-file-width="13
         50" height="47" src="//upload.wikimedia.org/wikipedia/en/thumb/4/41/Flag_of_India.svg/70px-Flag_of_India.svg.png" srcset="//u
         pload.wikimedia.org/wikipedia/en/thumb/4/41/Flag_of_India.svg/105px-Flag_of_India.svg.png 1.5x, //upload.wikimedia.org/wikipe
         dia/en/thumb/4/41/Flag_of_India.svg/140px-Flag_of_India.svg.png 2x" width="70"/></a>,
          <a href="/wiki/List of states and territories of India by area" title="List of states and territories of India by area">Are
```

(https://www.analyticsvidhya.com/wp-content/uploads/2015/10/All_Links.png) Above, it is showing all links including titles, links and other information. Now to show only links, we need to iterate over each a tag and then return the link using attribute “href” with **get**.



```
In [63]: all_links=soup.find_all("a")
         for link in all_links:
             print link.get("href")

         None
         /wiki/Wikipedia:Featured_lists
         #mw-head
         #p-search
         /wiki/States_and_union_territories_of_India
         /wiki/File:Flag_of_India.svg
         /wiki/List_of_states_and_territories_of_India_by_area
         /wiki/List_of_states_and_union_territories_of_India_by_population
```

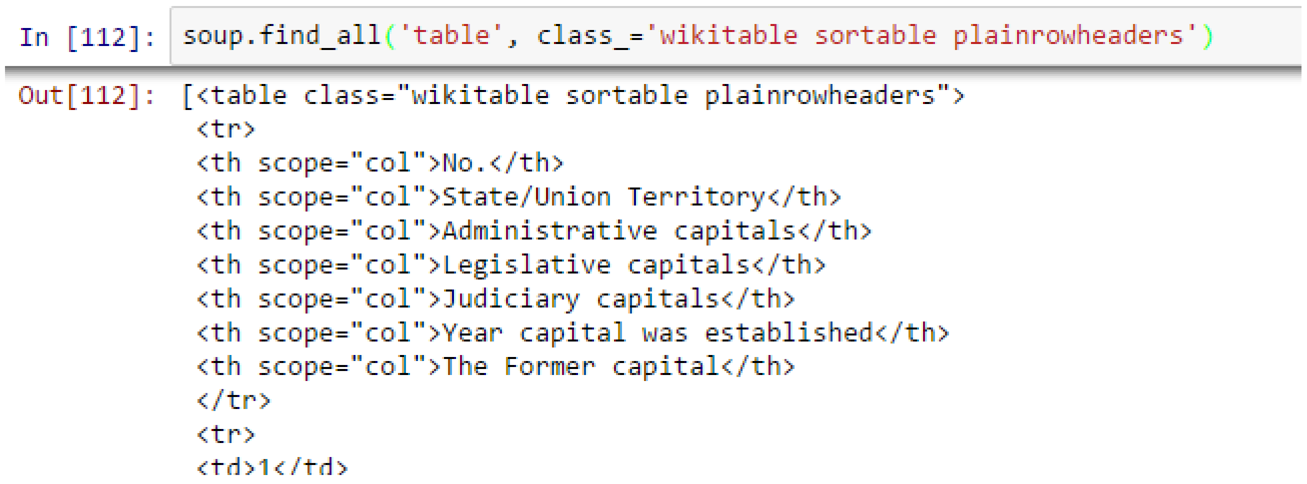
(https://www.analyticsvidhya.com/wp-content/uploads/2015/10/All_Links2.png)

4. **Find the right table:** As we are seeking a table to extract information about state capitals, we should identify the right table first. Let’s write the command to extract information within all **table** tags.

```
all_tables=soup.find_all('table')
```

Now to identify the right table, we will use attribute “class” of table and use it to filter the right table. In chrome, you can check the class name by right click on the required table of web page –> Inspect element –> Copy the class name OR go through the output of above command find the class name of right table.

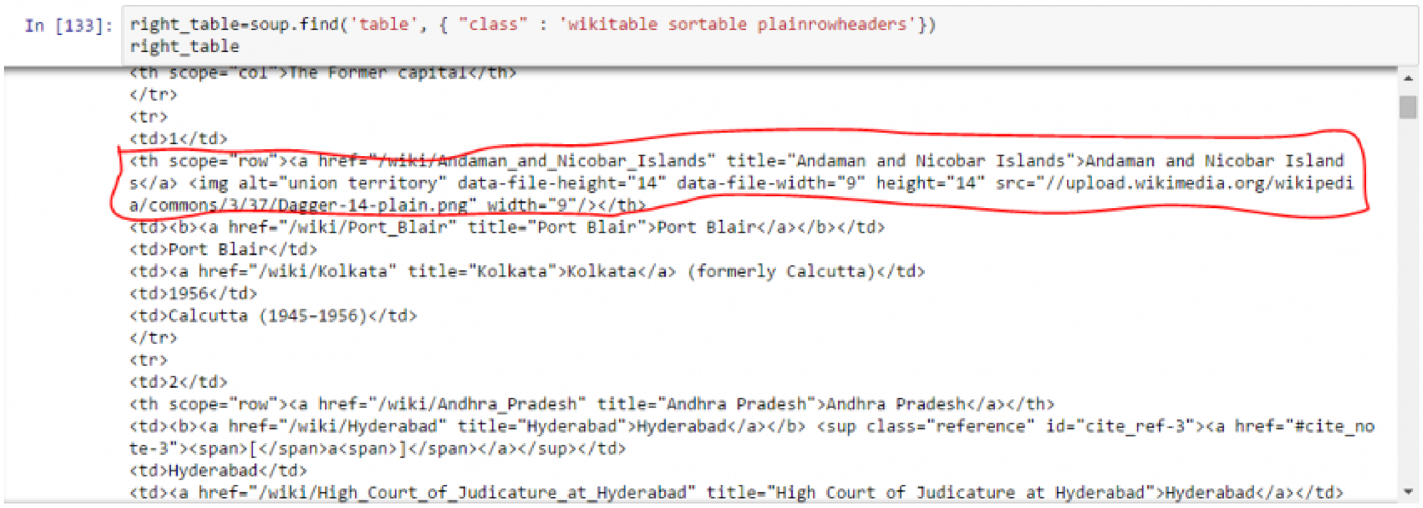
```
right_table=soup.find('table', class_='wikitable sortable plainrowheaders')
right_table
```



```
In [112]: soup.find_all('table', class_='wikitable sortable plainrowheaders')

Out[112]: [<table class="wikitable sortable plainrowheaders">
           <tr>
           <th scope="col">No.</th>
           <th scope="col">State/Union Territory</th>
           <th scope="col">Administrative capitals</th>
           <th scope="col">Legislative capitals</th>
           <th scope="col">Judiciary capitals</th>
           <th scope="col">Year capital was established</th>
           <th scope="col">The Former capital</th>
           </tr>
           <tr>
           <td>1</td>
```

(https://www.analyticsvidhya.com/wp-content/uploads/2015/10/righttable.png)Above, we are able to identify right table.

5. **Extract the information to DataFrame:** Here, we need to iterate through each row (tr) and then assign each element of tr (td) to a variable and append it to a list. Let's first look at the HTML structure of the table (I am not going to extract information for table heading <th>)



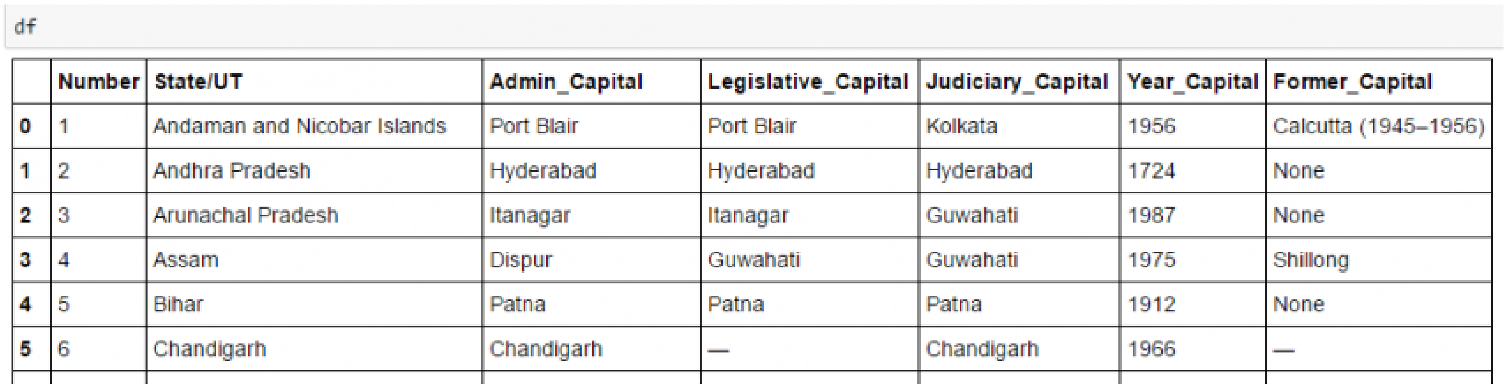
```
In [133]: right_table=soup.find('table', { "class" : 'wikitable sortable plainrowheaders'})
          right_table

          <th scope="col">The Former capital</th>
          </tr>
          <tr>
          <td>1</td>
          <th scope="row"><a href="/wiki/Andaman_and_Nicobar_Islands" title="Andaman and Nicobar Islands">Andaman and Nicobar Island
          s</a> <img alt="union territory" data-file-height="14" data-file-width="9" height="14" src="//upload.wikimedia.org/wikipedi
          a/commons/3/37/Dagger-14-plain.png" width="9"/></th>
          <td><b><a href="/wiki/Port_Blair" title="Port Blair">Port Blair</a></b></td>
          <td>Port Blair</td>
          <td><a href="/wiki/Kolkata" title="Kolkata">Kolkata</a> (formerly Calcutta)</td>
          <td>1956</td>
          <td>Calcutta (1945-1956)</td>
          </tr>
          <tr>
          <td>2</td>
          <th scope="row"><a href="/wiki/Andhra_Pradesh" title="Andhra Pradesh">Andhra Pradesh</a></th>
          <td><b><a href="/wiki/Hyderabad" title="Hyderabad">Hyderabad</a></b> <sup class="reference" id="cite_ref-3"><a href="#cite_no
          te-3"><span>[</span>a<span>]</span></a></sup></td>
          <td>Hyderabad</td>
          <td><a href="/wiki/High_Court_of_Judicature_at_Hyderabad" title="High Court of Judicature at Hyderabad">Hyderabad</a></td>
```

(https://www.analyticsvidhya.com/wp-content/uploads/2015/10/Structure.png)Above, you can notice that second element of <tr> is within tag <th> not <td> so we need to take care for this. Now to access value of each element, we will use "find(text=True)" option with each element. Let's look at the code:

```
#Generate lists

A=[]

B=[]

C=[]

D=[]

E=[]

F=[]

G=[]

for row in right_table.findAll("tr"):

    cells = row.findAll('td')

    states=row.findAll('th') #To store second column data

    if len(cells)==6: #Only extract table body not heading

        A.append(cells[0].find(text=True))

        B.append(states[0].find(text=True))

        C.append(cells[1].find(text=True))

        D.append(cells[2].find(text=True))

        E.append(cells[3].find(text=True))

        F.append(cells[4].find(text=True))

        G.append(cells[5].find(text=True))
```

```
#import pandas to convert list to data frame

import pandas as pd

df=pd.DataFrame(A,columns=['Number'])

df['State/UT']=B

df['Admin_Capital']=C

df['Legislative_Capital']=D

df['Judiciary_Capital']=E

df['Year_Capital']=F

df['Former_Capital']=G

df
```

Finally, we have data in dataframe:


```
df
```

	Number	State/UT	Admin_Capital	Legislative_Capital	Judiciary_Capital	Year_Capital	Former_Capital
0	1	Andaman and Nicobar Islands	Port Blair	Port Blair	Kolkata	1956	Calcutta (1945–1956)
1	2	Andhra Pradesh	Hyderabad	Hyderabad	Hyderabad	1724	None
2	3	Arunachal Pradesh	Itanagar	Itanagar	Guwahati	1987	None
3	4	Assam	Dispur	Guwahati	Guwahati	1975	Shillong
4	5	Bihar	Patna	Patna	Patna	1912	None
5	6	Chandigarh	Chandigarh	—	Chandigarh	1966	—

(https://www.analyticsvidhya.com/wp-content/uploads/2015/10/Output.png)

Similarly, you can perform various other types of web scraping using "**BeautifulSoup**". This will reduce your manual efforts to collect data from web pages. You can also look at the other attributes like .parent, .contents, .descendants and .next_sibling, .prev_sibling and various attributes to navigate using tag name. These will help you to scrap the web pages effectively.-

But, why can't I just use Regular Expressions?

Now, if you know regular expressions (https://www.analyticsvidhya.com/blog/2015/06/regular-expression-python/), you might be thinking that you can write code using regular expression which can do the same thing for you. I definitely had this question. In my experience with BeautifulSoup and Regular expressions to do same thing I found out:

- Code written in BeautifulSoup is usually more robust than the one written using regular expressions. Codes written with regular expressions need to be altered with any changes in pages. Even BeautifulSoup needs that in some cases, it is just that BeautifulSoup is relatively better.
- Regular expressions are much faster than BeautifulSoup, usually by a factor of 100 in giving the same outcome.

So, it boils down to speed vs. robustness of the code and there is no universal winner here. If the information you are looking for can be extracted with simple regex statements, you should go ahead and use them. For almost any complex work, I usually recommend BeautifulSoup more than regex.

End Note

In this article, we looked at web scraping methods using “BeautifulSoup” and “urllib2” in Python. We also looked at the basics of HTML and perform the web scraping step by step while solving a challenge. I’d recommend you to practice this and use it for collecting data from web pages.

Did you find this article helpful? Please share your opinions / thoughts in the comments section below.

If you like what you just read & want to continue your analytics learning, subscribe to our emails (http://feedburner.google.com/fb/a/mailverify?uri=analyticsvidhya), follow us on twitter (http://twitter.com/analyticsvidhya) or like our facebook page (http://facebook.com/analyticsvidhya).

RELATED