Web Scraping with Python (Indiameters)

What is web scraping?

Web scraping, **web harvesting**, or **web data extraction** is data scraping used for extracting data from websites. Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying, in which specific data is gathered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis.

The crawler:

A web crawler, which we generally call a “spider,” is an artificial intelligence that browses the internet to index and search for content by following links and exploring, like a person with too much time on their hands. In many projects you first “crawl” the web or one specific website to discover URLs which then you pass on to your scraper.

The Spider:

A web scraper is a specialized tool designed to accurately and quickly extract data from a web page. Web scrapers vary widely in design and complexity, depending on the project. An important part of every scraper is the data locators (or selectors) that are used to find the data that you want to extract from the HTML file - usually xpath, css selectors, regex or a combination of them is applied.

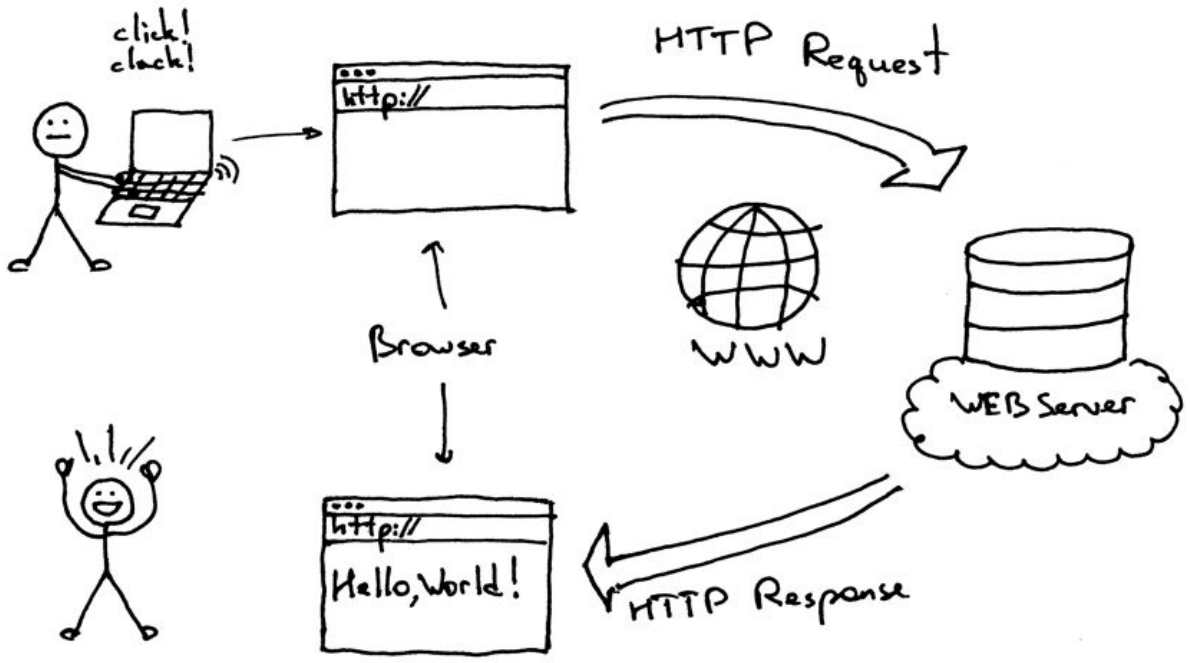
Web Scraping process:

1. Identify target website
2. Collect URLs of the pages where you want to extract data from
3. Make a request to these URLs to get the HTML of the page
4. Use locators to find the data in the HTML
5. Save the data in a JSON or CSV file or Excel or some other structured format

Libraries used in Web Scraping:

Requests:It is one of the most basic yet essential library for web scraping.‘Requests’ lets us make HTML requests to the website’s server for retrieving the data on its page. Getting the HTML content of a web page is the first and

foremost step of web scraping.



lxml: We know the requests library cannot parse the HTML retrieved from a web page. Therefore, we require lxml, a high performance, blazingly fast, production-quality HTML, and XML parsing Python library.

It combines the speed and power of Element trees with the simplicity of Python. It works well when we're aiming to scrape large datasets. The combination of requests and lxml is very common in web scraping. It also allows you to extract data from HTML using XPath and CSS selectors.

BeautifulSoup is perhaps the most widely used Python library for web scraping. It creates a parse tree for parsing HTML and XML documents. Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8.One of the primary reasons the Beautiful Soup library is so popular is that it is easier to work with and well suited for beginners. We can also combine Beautiful Soup with other parsers like **lxml**. But all this ease of use comes with a cost – it is slower than **lxml**. Even while using **lxml** as a parser, it is slower than pure **lxml**.

One major advantage of the Beautiful Soup library is that it works very well with poorly designed HTML and has a lot of functions. The combination of **Beautiful Soup** and **Requests** is quite common in the industry.

.

Selenium: There is a limitation to all the Python libraries we have discussed so far – we cannot easily scrape data from dynamically populated websites. It happens because sometimes the data present on the page is loaded through JavaScript. In simple words, if the page is not static, then the Python libraries mentioned earlier struggle to scrape the data from it.

That's where Selenium comes into play

Scrapy : Scrapy is not just a library; it is an entire web scraping framework **.**Scrapy provides spider bots that can crawl multiple websites and extract the data. With Scrapy, you can create your spider bots, host them on Scrapy Hub, or as an API. It allows you to create fully-functional spiders in a matter of a few minutes. You can also create pipelines using Scrapy.

The best thing about Scrapy is that it's asynchronous. It can make multiple HTTP requests simultaneously. This saves us a lot of time and increases our efficiency

Scraping Indian Cities population from the websites

Here we will be using **BeautifulSoup ,requests and pandas** libraries to scrape the data.And we will be using "html.parser" as parser to parse the the html content.
Note: always check the legality of website weather the website you are going to scrape allows you to scrape data from it.
You can check it by adding "/robots.txt"at the end of the URL of the website.

So lets scrape the cities population from th website:

URl of website:("https://www.worldometers.info/world-population/india-population/")
And we will save our scrape data in excel format

We will scrape below table:

Main Cities by Population in India

(includes boroughs, districts, urban agglomerations, etc.)

#	CITY NAME	POPULATION
1	**Mumbai**	12,691,836
2	**Delhi**	10,927,986
3	**Bengaluru**	5,104,047
4	**Kolkata**	4,631,392
5	**Chennai**	4,328,063
6	**Ahmedabad**	3,719,710
7	**Hyderabad**	3,597,816
8	**Pune**	2,935,744
9	**Surat**	2,894,504
10	**Kanpur**	2,823,249

(Below is the snap of the code with its proper explanation and output):

jupyter webscrapping population Last Checkpoint: 6 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [3]:
```
pip install bs4 # installing bs4 package
```

```
Requirement already satisfied: bs4 in c:\users\anand\anaconda3\lib\site-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in c:\users\anand\anaconda3\lib\site-packages (from bs4) (4.9.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\anand\anaconda3\lib\site-packages (from beautifulsoup4->bs4) (2.0.1)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
import bs4
from bs4 import BeautifulSoup
```

In [5]:
```
import requests # importing requests library
```

In [ ]:

In [6]:
```
import pandas as pd #importing pandas to create dataframes aand to save output in Excel file
```

In [7]:
```
url=("https://www.worldometers.info/world-population/india-population/")
```

In [8]:
```
r=requests.get(url) # sending rewuest to the url
htmlcontent=r.content # saving the content of url in htmlcontent varialble
soup=BeautifulSoup(htmlcontent,'html.parser') #making of soup by pretifying the html conent
```

```
In [47]: population=soup.findAll("table",attrs={'class':'table table-hover table-condensed table-list'}) # finding All the tables on thee
```

```
In [49]: type(population) # checking the type of the variable to check it is in tag format or not
```

```
Out[49]: bs4.element.ResultSet
```

```
In [50]: len(population) # as the variable was in list of the table we have to identify which table we want to scrape
```

```
Out[50]: 1
```

```
In [51]: population # printing the list of the table (there is only one table in list)
```

```
Out[51]: [<table class="table table-hover table-condensed table-list"> <thead> <tr style="background-color:#F8F8F8;"> <th style="widt
h:20px">#</th> <th style="font-weight: 100; font-size:16px; text-align:left; padding-top:5px; padding-bottom:5px">CITY NAME</
th> <th style="font-weight: 100; font-size:16px; text-align:left; padding-top:5px; padding-bottom:5px">POPULATION</th> </tr>
</thead> <tbody> <tr> <td style="color:#999; text-align:center; vertical-align:middle">1</td> <td style="font-weight: bold; f
ont-size:17px; text-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">Mumbai</td> <td style=" font-size:17p
x; text-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">12,691,836</td> </tr> <tr> <td style="color:#999;
text-align:center; vertical-align:middle">2</td> <td style="font-weight: bold; font-size:17px; text-align:left; padding-left:
10px; padding-top:5px; padding-bottom:5px">Delhi</td> <td style=" font-size:17px; text-align:left; padding-left:10px; padding
-top:5px; padding-bottom:5px">10,927,986</td> </tr> <tr> <td style="color:#999; text-align:center; vertical-align:middle">3</
td> <td style="font-weight: bold; font-size:17px; text-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">Be
ngaluru</td> <td style=" font-size:17px; text-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">5,104,047</
td> </tr> <tr> <td style="color:#999; text-align:center; vertical-align:middle">4</td> <td style="font-weight: bold; font-siz
e:17px; text-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">Kolkata</td> <td style=" font-size:17px; tex
t-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">4,631,392</td> </tr> <tr> <td style="color:#999; text-a
lign:center; vertical-align:middle">5</td> <td style="font-weight: bold; font-size:17px; text-align:left; padding-left:10px;
```

```
In [52]: population=population[0] # selecting the table which we wanted toscrape from the list of the tables
```

```
In [58]: population.tbody.get_text() #selecting the text of the body tag of the table
```

```
Out[58]: '  1 Mumbai 12,691,836   2 Delhi 10,927,986   3 Bengaluru 5,104,047   4 Kolkata 4,631,392   5 Chennai 4,328,063   6 Ahmedabad
3,719,710   7 Hyderabad 3,597,816   8 Pune 2,935,744   9 Surat 2,894,504   10 Kanpur 2,823,249   11 Jaipur 2,711,758   12 Navi
Mumbai 2,600,000   13 Lucknow 2,472,011   14 Nagpur 2,228,018   15 Indore 1,837,041   16 Patna 1,599,920   17 Bhopal 1,599,914
18 Ludhiana 1,545,368   19 Tirunelveli 1,435,844   20 Agra 1,430,055   21 Vadodara 1,409,476   22 Gorakhpur 1,324,570   23 Nash
ik 1,289,497   24 Pimpri 1,284,606   25 Kalyan 1,262,255   26 Thane 1,261,517   27 Meerut 1,223,184   28 Nowrangapur 1,220,946
29 Faridabad 1,220,229   30 Ghaziabad 1,199,191   31 Dombivli 1,193,000   32 Rajkot 1,177,362   33 Varanasi 1,164,404   34 Amri
tsar 1,092,450   35 Allahabad 1,073,438   36 Visakhapatnam 1,063,178   37 Teni 1,034,724   38 Jabalpur 1,030,168   39 Haora 1,0
27,672   40 Aurangabad 1,016,441   41 Shivaji Nagar 1,000,000   42 Solapur 997,281   43 Srinagar 975,857   44 Chandigarh 960,78
7   45 Coimbatore 959,823   46 Jodhpur 921,476   47 Madurai 909,908   48 Guwahati 899,094   49 Gwalior 882,458   50 Vijayawada
874,587   51 Mysore 868,313   52 Ranchi 846,454   53 Hubli 840,214   54 Jalandhar 785,178   55 Thiruvananthapuram 784,153   56
Salem 778,396   57 Tiruchirappalli 775,484   58 Kota 763,088   59 Bhubaneshwar 762,243   60 Aligarh 753,207   61 Bareilly 745,4
35   62 Moradabad 721,139   63 Bhiwandi 707,035   64 Raipur 679,995   65 Gorakhpur 674,246   66 Bhilai 625,138   67 Jamshedpur
616,338   68 Borivli 609,617   69 Cochin 604,696   70 Amravati 603,837  '
```

```
In [59]: type(population)
```

```
Out[59]: bs4.element.Tag
```

```
In [60]: anda=population.find_all("tr") # selecting all the rows  of thebody and saving it in variable
```

```
n [61]: anda
```

```
ut[61]: [<tr style="background-color:#F8F8F8;"> <th style="width:20px">#</th> <th style="font-weight: 100; font-size:16px; text-alig
n:left; padding-top:5px; padding-bottom:5px">CITY NAME</th> <th style="font-weight: 100; font-size:16px; text-align:left; pad
ding-top:5px; padding-bottom:5px">POPULATION</th> </tr>,
 <tr> <td style="color:#999; text-align:center; vertical-align:middle">1</td> <td style="font-weight: bold; font-size:17px; t
ext-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">Mumbai</td> <td style=" font-size:17px; text-align:le
ft; padding-left:10px; padding-top:5px; padding-bottom:5px">12,691,836</td> </tr>,
 <tr> <td style="color:#999; text-align:center; vertical-align:middle">2</td> <td style="font-weight: bold; font-size:17px; t
ext-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">Delhi</td> <td style=" font-size:17px; text-align:lef
t; padding-left:10px; padding-top:5px; padding-bottom:5px">10,927,986</td> </tr>,
 <tr> <td style="color:#999; text-align:center; vertical-align:middle">3</td> <td style="font-weight: bold; font-size:17px; t
ext-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">Bengaluru</td> <td style=" font-size:17px; text-alig
n:left; padding-left:10px; padding-top:5px; padding-bottom:5px">5,104,047</td> </tr>,
 <tr> <td style="color:#999; text-align:center; vertical-align:middle">4</td> <td style="font-weight: bold; font-size:17px; t
ext-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">Kolkata</td> <td style=" font-size:17px; text-align:l
eft; padding-left:10px; padding-top:5px; padding-bottom:5px">4,631,392</td> </tr>,
 <tr> <td style="color:#999; text-align:center; vertical-align:middle">5</td> <td style="font-weight: bold; font-size:17px; t
ext-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">Chennai</td> <td style=" font-size:17px; text-align:l
eft; padding-left:10px; padding-top:5px; padding-bottom:5px">4,328,063</td> </tr>,
 <tr> <td style="color:#999; text-align:center; vertical-align:middle">6</td> <td style="font-weight: bold; font-size:17px; t
ext-align:left; padding-left:10px; padding-top:5px; padding-bottom:5px">Ahmedabad</td> <td style=" font-size:17px; text-alig
```

In [64]:
```
City_Name=[]
for row in anda: # with the help of nested loop extracting all the rows of the cityname columns and saving it in list
    for cell in row.find_all("td",attrs={'style':'font-weight: bold; font-size:17px; text-align:left; padding-left:10px; padding-

        sa=cell.text
        City_Name.append(sa)
City_Name
```

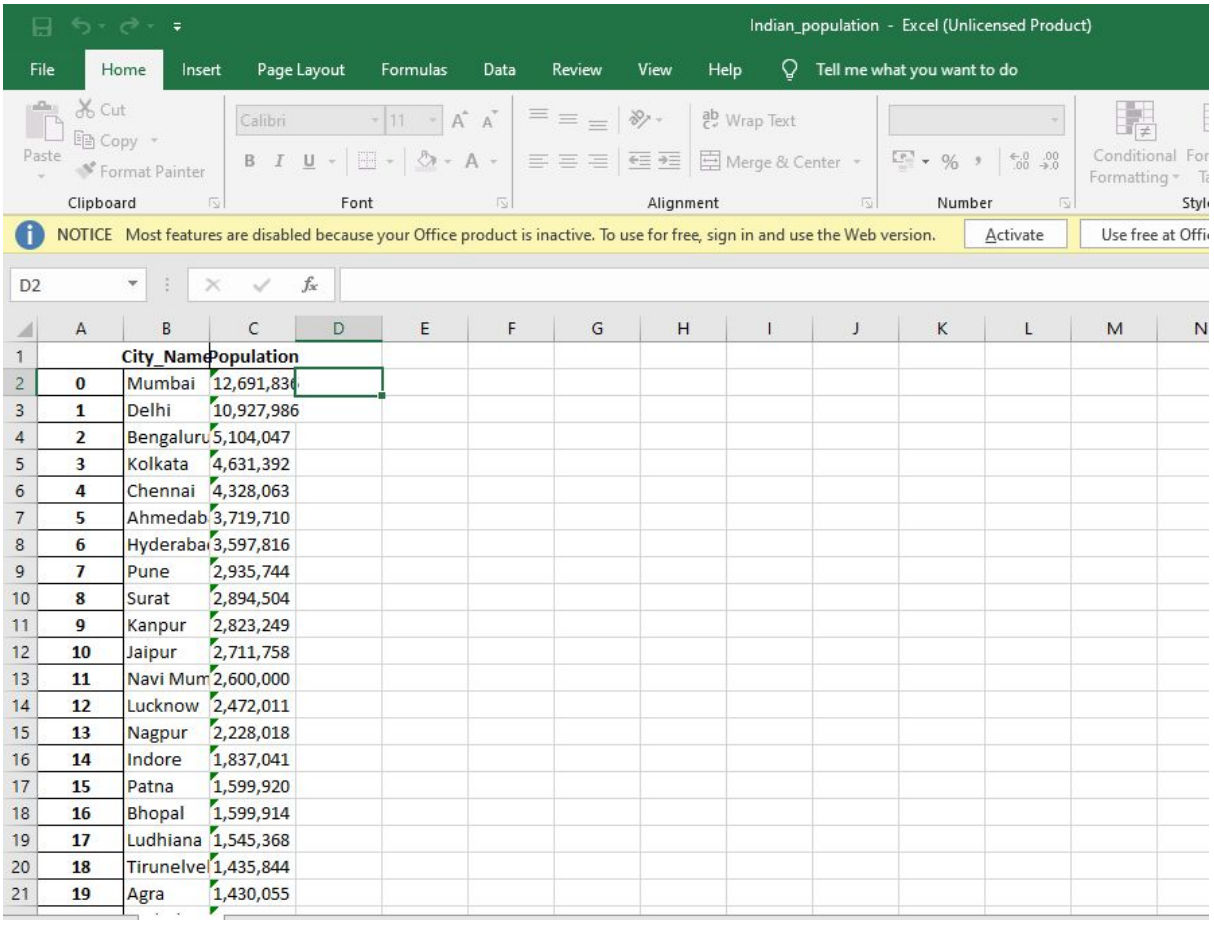
Out[64]:
```
['Mumbai',
 'Delhi',
 'Bengaluru',
 'Kolkata',
 'Chennai',
 'Ahmedabad',
 'Hyderabad',
 'Pune',
 'Surat',
 'Kanpur',
 'Jaipur',
 'Navi Mumbai',
 'Lucknow',
 'Nagpur',
 'Indore',
 'Patna',
 'Bhopal',
 'Ludhiana',
 'Tirunelveli',
 'Agra',
 'Vadodara',
 'Gorakhpur',
 'Nashik',
```


In [66]:
```
City_population=[]
for row in anda:  # with the help of nested loop extracting all the rows of the Population columns and saving it in list
    for cell in row.find_all("td",attrs={"style":" font-size:17px; text-align:left; padding-left:10px; padding-top:5px; padding-b
        sd=cell.text
        City_population.append(sd)

City_population
```

Out[66]:
```
['12,691,836',
 '10,927,986',
 '5,104,047',
 '4,631,392',
 '4,328,063',
 '3,719,710',
 '3,597,816',
 '2,935,744',
 '2,894,504',
 '2,823,249',
 '2,711,758',
 '2,600,000',
 '2,472,011',
 '2,228,018',
 '1,837,041',
 '1,599,920',
 '1,599,914',
 '1,545,368',
 '1,435,844',
 '1,430,055',
 '1,409,476',
 '1,324,570',
 '1,289,497',
```

```
In [75]: india_pop=[City_Name,City_population] #apppending list to the list

In [76]: india_pop=pd.DataFrame(india_pop).transpose() # forming the lists into dataFrame

In [78]: india_pop.columns=["City_Name","Population"] # adding columns name to the dataframe

In [79]: india_population=india_pop
         print(india_population)

     City_Name  Population
0       Mumbai  12,691,836
1        Delhi  10,927,986
2    Bengaluru   5,104,047
3      Kolkata   4,631,392
4      Chennai   4,328,063
..         ...         ...
65      Bhilai     625,138
66  Jamshedpur     616,338
67     Borivli     609,617
68      Cochin     604,696
69    Amravati     603,837

[70 rows x 2 columns]

In [82]: india_population.to_excel("Indian_population.xlsx") # saving the Dataframe in excel format
```

OUTPUT:



	A	B	C
1		City_Name	Population
2	0	Mumbai	12,691,836
3	1	Delhi	10,927,986
4	2	Bengaluru	5,104,047
5	3	Kolkata	4,631,392
6	4	Chennai	4,328,063
7	5	Ahmedab	3,719,710
8	6	Hyderaba	3,597,816
9	7	Pune	2,935,744
10	8	Surat	2,894,504
11	9	Kanpur	2,823,249
12	10	Jaipur	2,711,758
13	11	Navi Mum	2,600,000
14	12	Lucknow	2,472,011
15	13	Nagpur	2,228,018
16	14	Indore	1,837,041
17	15	Patna	1,599,920
18	16	Bhopal	1,599,914
19	17	Ludhiana	1,545,368
20	18	Tirunelvel	1,435,844
21	19	Agra	1,430,055

22	20	Vadodara	1,409,476
23	21	Gorakhpu	1,324,570
24	22	Nashik	1,289,497
25	23	Pimpri	1,284,606
26	24	Kalyan	1,262,255
27	25	Thane	1,261,517
28	26	Meerut	1,223,184
29	27	Nowranga	1,220,946
30	28	Faridabad	1,220,229
31	29	Ghaziabac	1,199,191
32	30	Dombivli	1,193,000
33	31	Rajkot	1,177,362
34	32	Varanasi	1,164,404
35	33	Amritsar	1,092,450
36	34	Allahabad	1,073,438
37	35	Visakhapa	1,063,178
38	36	Teni	1,034,724
39	37	Jabalpur	1,030,168
40	38	Haora	1,027,672
41	39	Aurangab	1,016,441
42	40	Shivaji Na	1,000,000
47		Guwahati	899,094
48		Gwalior	882,458
49		Vijayawac	874,587
50		Mysore	868,313
51		Ranchi	846,454
52		Hubli	840,214
53		Jalandhar	785,178
54		Thiruvana	784,153
55		Salem	778,396
56		Tiruchirap	775,484
57		Kota	763,088
58		Bhubanes	762,243
59		Aligarh	753,207
60		Bareilly	745,435
61		Moradaba	721,139
62		Bhiwandi	707,035
63		Raipur	679,995
64		Gorakhpu	674,246
65		Bhilai	625,138
66		Jamshedp	616,338
67		Borivli	609,617

Conclusion: Hence we successfully scrape the table date from the target URL and we can further use this data for data visualization and in chatbot .