Published in Analytics Vidhya

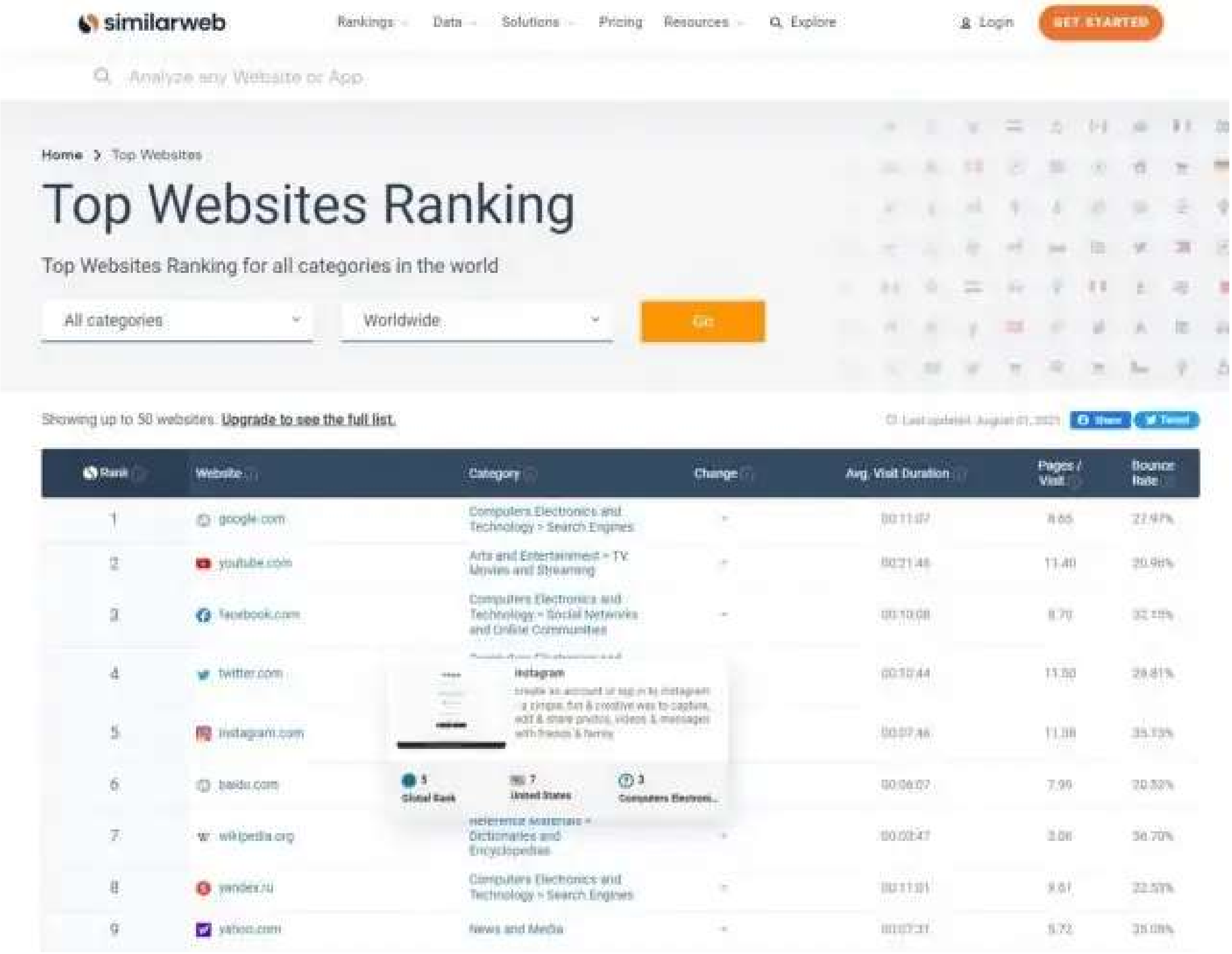
Vandany Lubis (Danny) Follow

Oct 4, 2021 · 6 min read · Listen

Save

How to Scrape Table from Website using Python

Previously, we have learned about the basics of scraping data on the sites, but did you know that we are also able to scrape data in a tabular form? If you often surfing the internet you probably notice that many websites containing data tables. For example flight schedule, product features, services differentiation, tv schedule, ranking, and many more.



Top Websites Ranking

Top Websites Ranking for all categories in the world

Showing up to 50 websites. Upgrade to see the full list.

Last updated: August 01, 2021

Rank	Website	Category	Change	Avg. Visit Duration	Pages / Visit	Bounce Rate
1	google.com	Computers Electronics and Technology > Search Engines	-	00:11:07	8.65	22.97%
2	youtube.com	Arts and Entertainment > TV Movies and Streaming	-	00:21:46	11.40	20.96%
3	facebook.com	Computers Electronics and Technology > Social Networks and Online Communities	-	00:10:08	8.70	32.11%
4	twitter.com			00:10:44	11.50	28.81%
5	instagram.com			00:07:46	11.08	35.13%
6	baidu.com			00:06:07	7.99	20.32%
7	wikipedia.org	Reference Materials > Dictionaries and Encyclopedias	-	00:03:47	3.06	56.70%
8	yandex.ru	Computers Electronics and Technology > Search Engines	-	00:11:01	9.61	22.57%
9	yahoo.com	News and Media	-	00:07:31	5.72	35.08%

Instagram

create an account or log in to instagram - a simple, fun & creative way to capture, edit & share photos, videos & messages with friends & family

5 Global Rank

7 United States

3 Computers Electroni...

Fig.1 Data Table from Similarweb.com

In some cases maybe it is easy to copy and paste the table into excel without doing coding, but as data scientists, we need to understand that in the future we will work with bigger data where copy-paste method is not really efficient. So now I will show you how to scrape a data table from a website.

STEP 1. INSTALLING LIBRARIES

First of all, we need these required libraries installed in our environment:

1. BeautifulSoup4
2. Requests
3. pandas
4. lxml

I recommend you to read the previous tutorial about how to scrape data from the website for beginners if you having trouble in this step.

STEP 2. IMPORT REQUIRED LIBRARIES

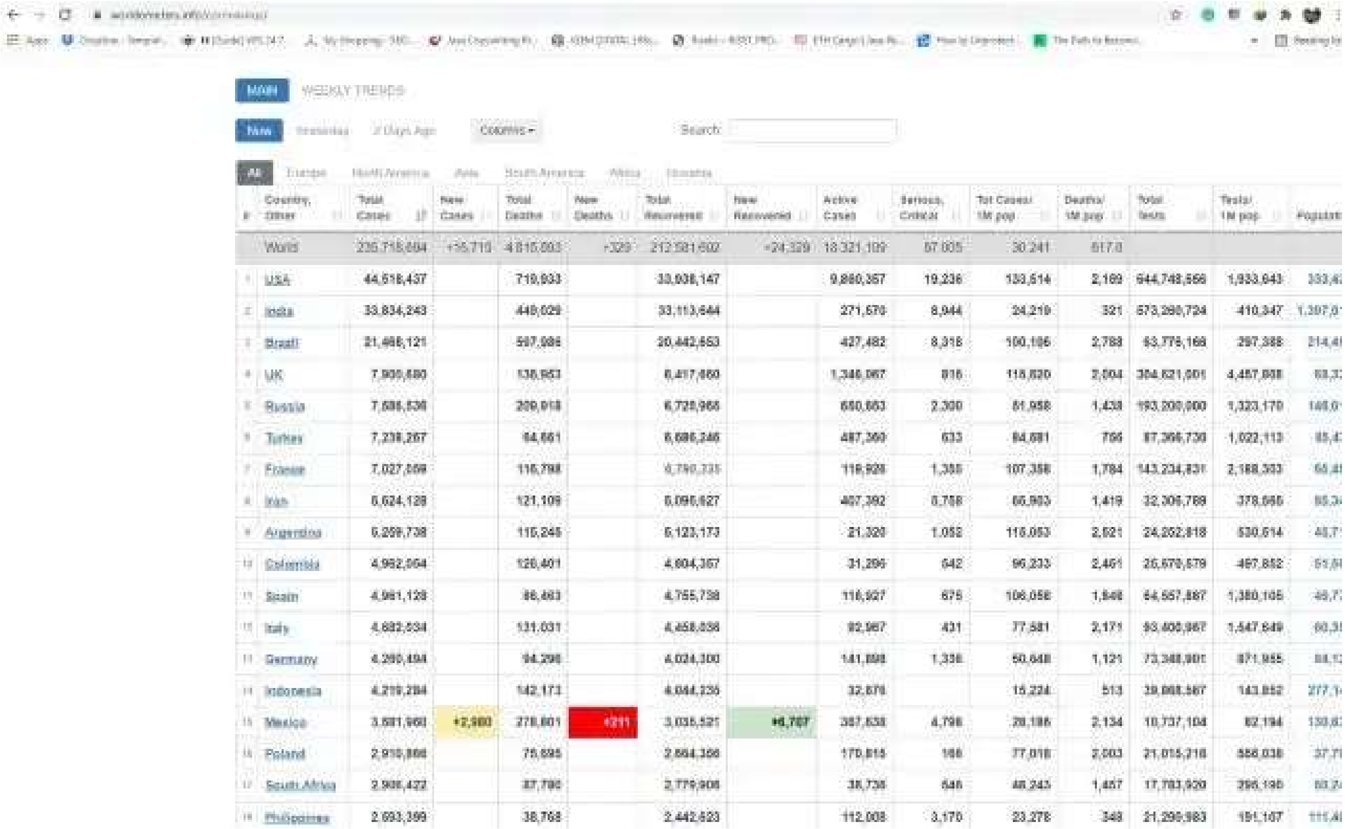
So after the important libraries are already installed the next thing we can do is open Spyder. The reason why I using Spyder is that it is more comfortable to use for projects, but it is okay if you have your own preferences.

Okay, once we open the Spyder the next thing we can do is importing the required library:

```
# Import libraries
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

STEP 3. SELECT PAGE

In this project, we will scrape the covid data table from Worldometers. Same as the previous tutorial this website is also considered easier to understand for beginners since it is made with HTML.



#	Country, Other	Total Cases	New Cases	Total Deaths	New Deaths	Total Recovered	New Recovered	Active Cases	Serious, Critical	Tot Cases/ 1M pop	Deaths/ 1M pop	Total Tests	Tests/ 1M pop	Populat
	World	235,718,694	+35,716	4,815,093	+329	212,581,602	+24,329	18,321,109	87,005	30,241	617.0			
1	USA	44,518,437		719,933		33,938,147		9,860,357	19,236	133,514	2,159	544,748,566	1,633,643	333,6
2	India	33,834,243		449,029		33,113,644		271,570	8,944	24,219	321	573,280,724	410,347	1,397,0
3	Brazil	21,468,121		597,986		20,442,653		427,482	8,318	100,186	2,788	63,776,166	297,388	214,4
4	UK	7,900,680		136,953		6,417,660		1,346,067	916	115,820	2,004	304,621,001	4,457,938	68,3
5	Russia	7,686,536		209,918		6,720,968		660,663	2,300	61,958	1,438	193,200,000	1,323,170	146,0
6	Turkey	7,238,267		64,661		6,686,246		487,360	633	84,681	756	87,366,730	1,022,113	85,4
7	France	7,027,059		116,798		6,790,735		119,928	1,355	107,358	1,784	143,234,831	2,188,333	65,4
8	Iran	5,624,128		121,109		5,096,627		407,392	5,768	65,903	1,419	32,306,769	378,565	85,3
9	Argentina	5,259,738		115,245		5,123,173		21,320	1,052	115,053	2,521	24,252,818	530,614	45,7
10	Colombia	4,952,004		126,401		4,604,357		31,296	642	96,233	2,461	25,670,579	497,852	51,6
11	Spain	4,961,128		86,463		4,755,738		118,927	675	106,058	1,848	54,557,887	1,380,105	46,7
12	Italy	4,682,534		131,031		4,458,036		92,967	431	77,581	2,171	93,400,967	1,547,649	60,3
13	Germany	4,260,494		94,298		4,024,300		141,896	1,336	50,648	1,121	73,348,901	871,955	84,1
14	Indonesia	4,219,284		142,173		4,044,235		32,876		15,224	513	39,068,567	143,852	277,1
15	Mexico	3,681,960	+2,980	278,801	+211	3,036,521	+6,707	367,638	4,798	28,186	2,134	10,737,104	82,194	130,6
16	Poland	2,910,866		75,695		2,664,356		170,815	166	77,016	2,003	21,015,216	556,038	37,7
17	South Africa	2,906,422		87,780		2,779,906		38,736	646	48,243	1,457	17,783,920	295,195	60,2
18	Philippines	2,693,399		38,768		2,442,623		112,008	3,170	23,278	348	21,290,983	191,107	111,4

Fig.2 COVID Data Table

STEP 4. REQUEST PERMISSION

After we select what page we want to scrape, now we can copy the page’s URL and use **requests** to ask permission from the hosting server that we want to fetch data from their site.

```
# Create an URL object
url = 'https://www.worldometers.info/coronavirus/'

# Create object page
page = requests.get(url)
```

If the output is **<Response [200]>** that means the server allows us to collect data from their website. After we have permission from the server now we need to parse the HTML code using **lxml** to make it more readable.

```
# parser-lxml = Change html to Python friendly format
# Obtain page's information
soup = BeautifulSoup(page.text, 'lxml')
soup
```

STEP 5. INSPECT TABLE ELEMENT

In the previous article, we have learned how to inspect every element in a website page so I assume you have understood how to work with it. To obtain the information we need to inspect the location of the table first.

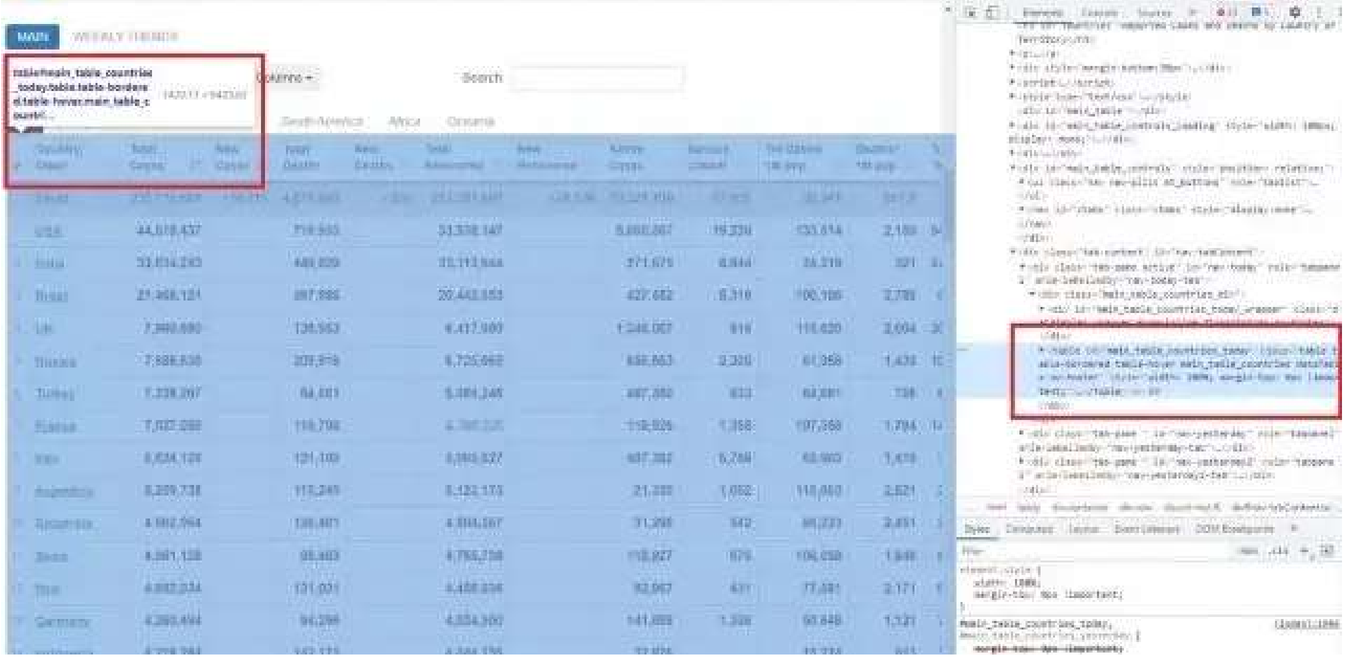


Fig.3 Table location

As we can see from the picture above that table is located under the **<table>** tag and **id = ‘main_table_countries_today’.** After we have found the location of the table now we can define the variable. In this section, we define the table as **‘table1’**.

```
# Obtain information from tag <table>
table1 = soup.find(‘table’, id=’main_table_countries_today’)
table1
```

STEP 6. CREATE A COLUMN LIST

After the **table1** has been created now the next thing we can do is inspecting the location of each column. If we look into each column we notice that they have the same characteristic.

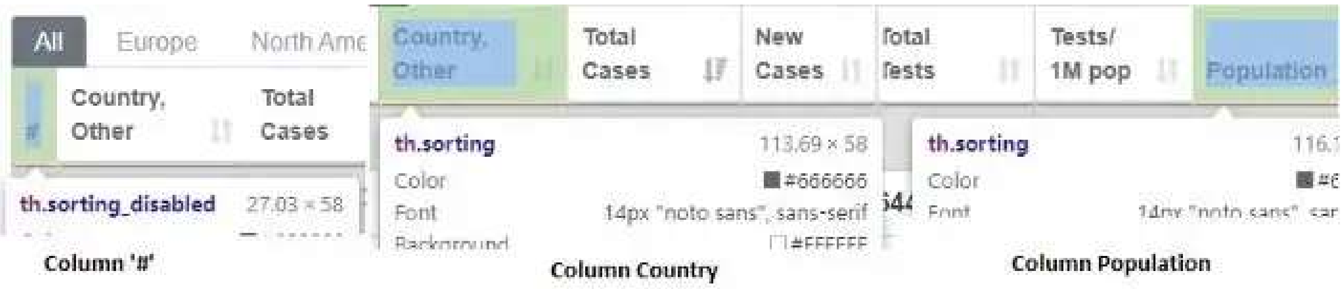


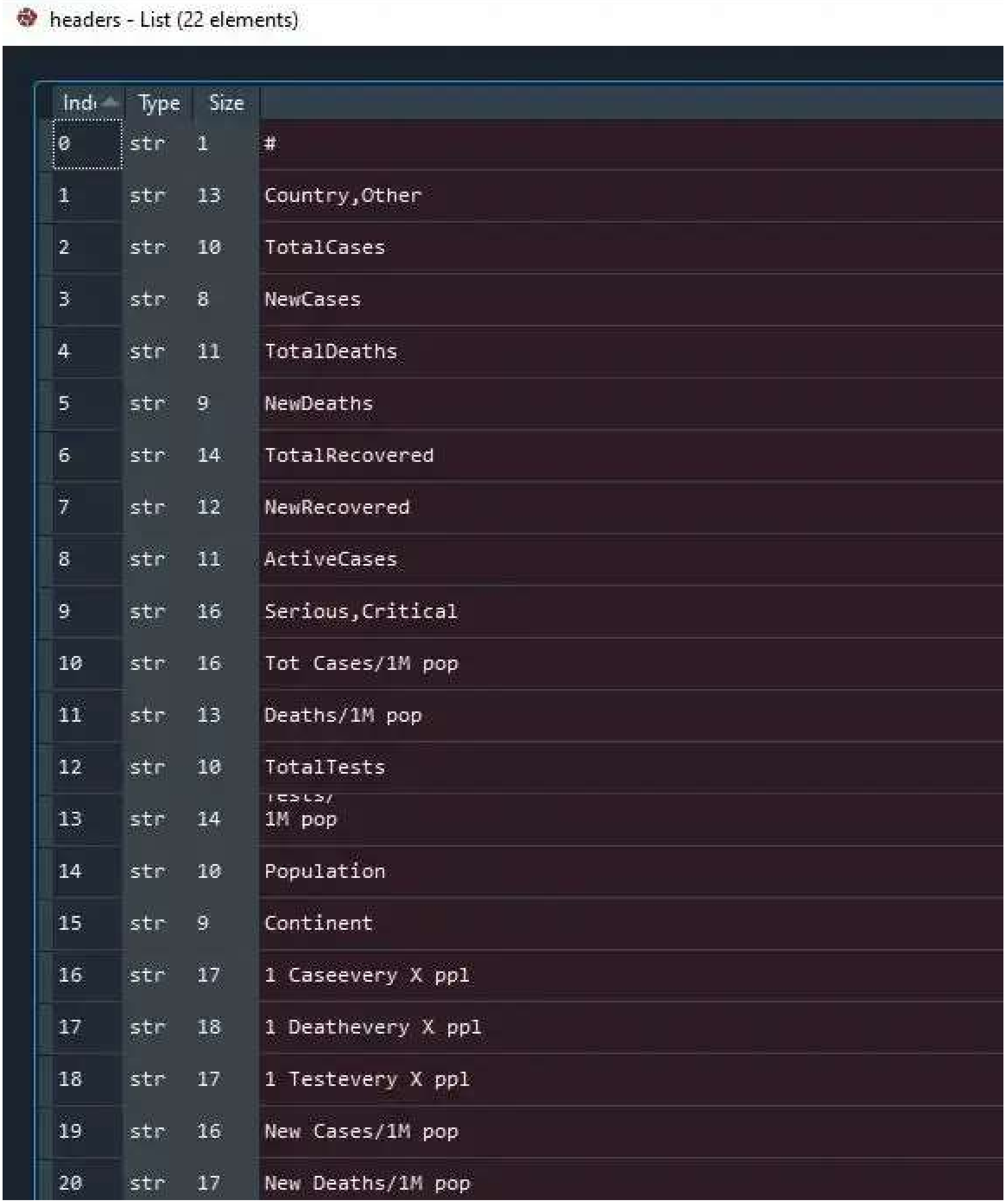
Fig.4 Columns

In the picture above we can see the characteristic of each column is they are located under the tag **<th>.**

After we found the tag of each column the next thing we can do is create a for loop to fill an empty list with each column. We define an empty list as **headers**

```
# Obtain every title of columns with tag <th>
headers = []
for i in table1.find_all(‘th’):
 title = i.text
 headers.append(title)
```

Output:



headers - List (22 elements)

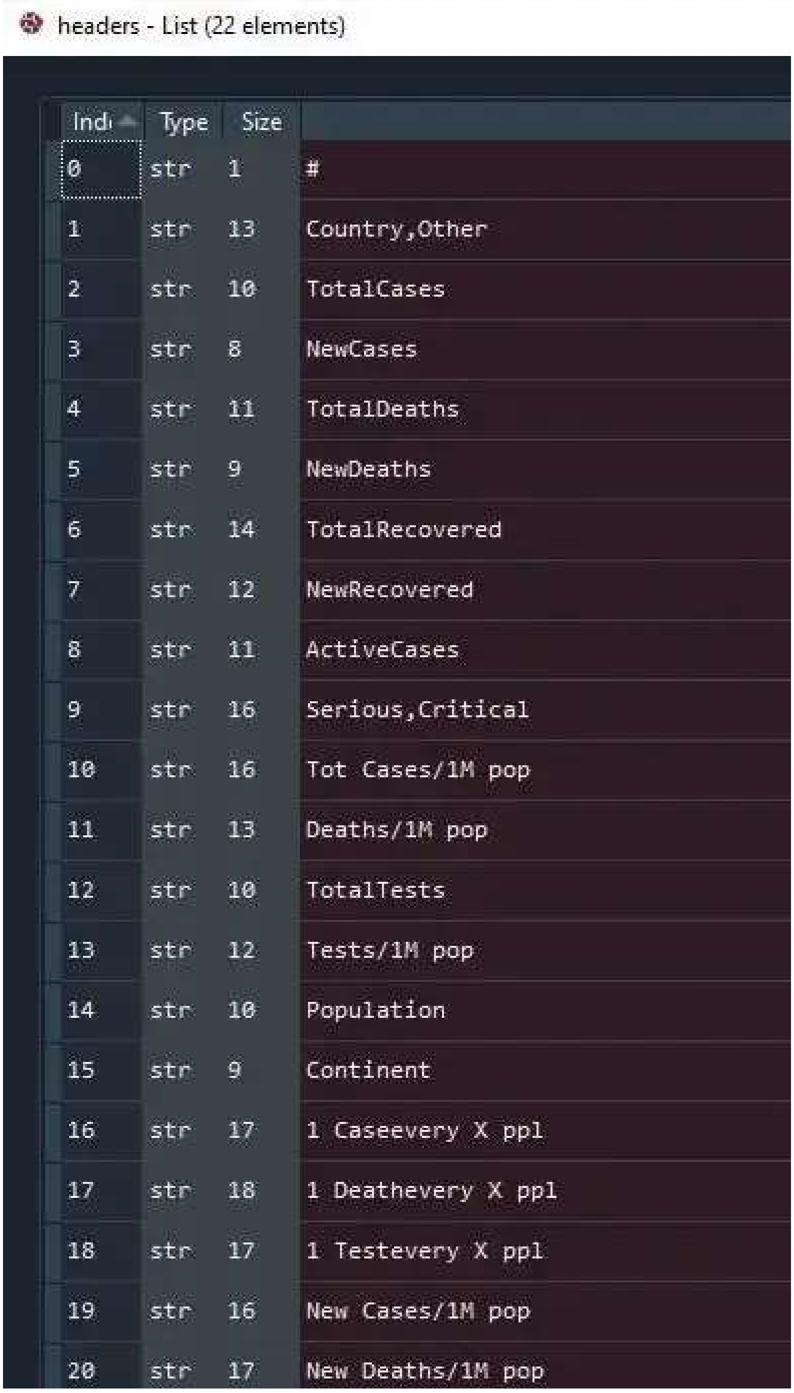
Ind	Type	Size	
0	str	1	#
1	str	13	Country,Other
2	str	10	TotalCases
3	str	8	NewCases
4	str	11	TotalDeaths
5	str	9	NewDeaths
6	str	14	TotalRecovered
7	str	12	NewRecovered
8	str	11	ActiveCases
9	str	16	Serious,Critical
10	str	16	Tot Cases/1M pop
11	str	13	Deaths/1M pop
12	str	10	TotalTests
13	str	14	Tests/ 1M pop
14	str	10	Population
15	str	9	Continent
16	str	17	1 Caseevery X ppl
17	str	18	1 Deathevery X ppl
18	str	17	1 Testevery X ppl
19	str	16	New Cases/1M pop
20	str	17	New Deaths/1M pop

Fig.5 Headers

After the list is successfully filled with columns, now we can check again. Let’s look at index 13 we find wrapped text. Wrapped text like this could be a problem when we want to make a data frame from it, so we need to convert it into one-line text.

```
# Convert wrapped text in column 13 into one line text
headers[13] = ‘Tests/1M pop’
```

Output:



headers - List (22 elements)

Indi	Type	Size	
0	str	1	#
1	str	13	Country,Other
2	str	10	TotalCases
3	str	8	NewCases
4	str	11	TotalDeaths
5	str	9	NewDeaths
6	str	14	TotalRecovered
7	str	12	NewRecovered
8	str	11	ActiveCases
9	str	16	Serious,Critical
10	str	16	Tot Cases/1M pop
11	str	13	Deaths/1M pop
12	str	10	TotalTests
13	str	12	Tests/1M pop
14	str	10	Population
15	str	9	Continent
16	str	17	1 Caseevery X ppl
17	str	18	1 Deathevery X ppl
18	str	17	1 Testevery X ppl
19	str	16	New Cases/1M pop
20	str	17	New Deaths/1M pop

Fig.6 Index 13 have been fixed

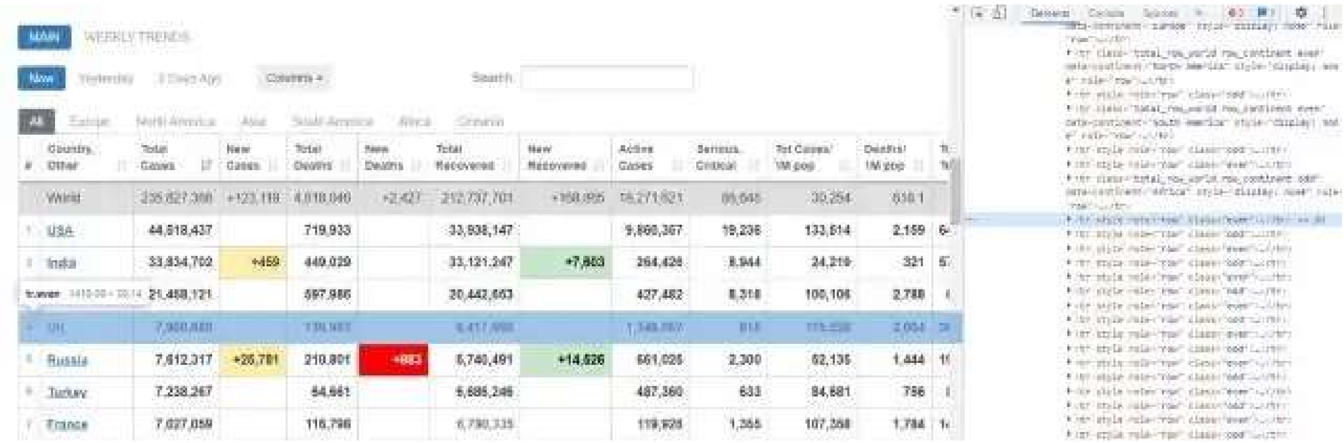
STEP 7. CREATE A DATA FRAME

After the list of columns is made the next thing we can do is create a dataframe. We define the dataframe as **mydata.**

```
# Create a dataframe
mydata = pd.DataFrame(columns = headers)
```

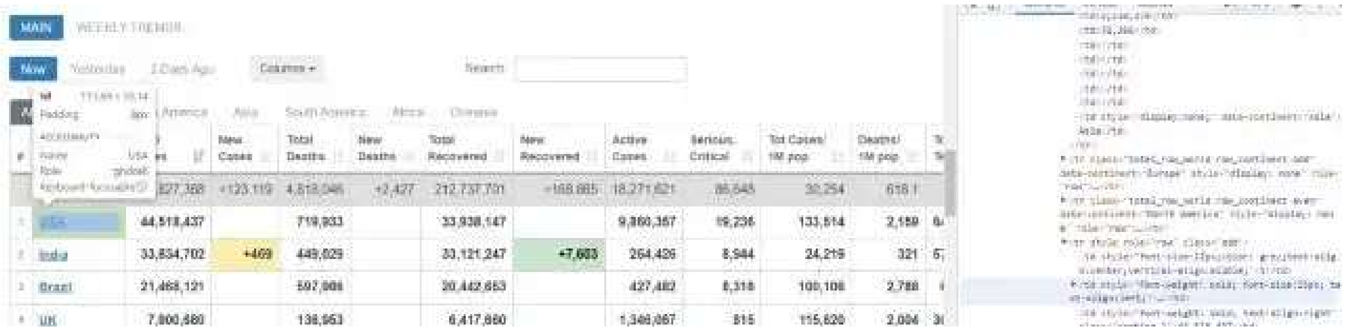
STEP 8. CREATE A FOR LOOP TO FILL DATAFRAME

After the dataframe is created now we can fill it with items in each column. Before we create a for loop, we need to identify the location of the row and item column first.



#	Country, Other	Total Cases	New Cases	Total Deaths	New Deaths	Total Recovered	New Recovered	Active Cases	Serious, Critical	Tot Cases/ 1M pop	Deaths/ 1M pop
	World	235,827,388	+123,119	4,818,046	+2,427	212,737,701	+168,885	18,271,621	86,645	30,254	618.1
1	USA	44,518,437		719,933		33,938,147		9,860,357	19,236	133,514	2,159
2	India	33,834,702	+459	449,029		33,121,247	+7,603	264,426	8,944	24,219	321
3		21,468,121		597,986		20,442,653		427,482	8,318	100,106	2,788
4	UK	7,800,680		136,953		6,417,660		1,346,067	815	115,620	2,004
5	Russia	7,612,317	+26,781	210,801	+883	6,740,491	+14,626	661,025	2,300	52,135	1,444
6	Turkey	7,238,267		64,661		6,686,246		487,360	633	84,681	756
7	France	7,027,059		116,796		6,790,335		119,928	1,355	107,358	1,784

Fig.7 Location of row



#	Country, Other	Total Cases	New Cases	Total Deaths	New Deaths	Total Recovered	New Recovered	Active Cases	Serious, Critical	Tot Cases/ 1M pop	Deaths/ 1M pop
		235,827,388	+123,119	4,818,046	+2,427	212,737,701	+168,885	18,271,621	86,645	30,254	618.1
1	USA	44,518,437		719,933		33,938,147		9,860,357	19,236	133,514	2,159
2	India	33,834,702	+459	449,029		33,121,247	+7,603	264,426	8,944	24,219	321
3	Brazil	21,468,121		597,986		20,442,653		427,482	8,318	100,106	2,788
4	UK	7,800,680		136,953		6,417,660		1,346,067	815	115,620	2,004

Fig.8 Location of item column

From the pictures above we can identify that the row is located under tag **<tr>** and items are located under tag **<td>** . This is applied to all rows and items within the table.

After we found the tags now we can create a **for loop**.

```
# Create a for loop to fill mydata
for j in table1.find_all(‘tr’)[1:]:
 row_data = j.find_all(‘td’)
 row = [i.text for i in row_data]
 length = len(mydata)
 mydata.loc[length] = row
```

Fig.9 Dataframe

STEP 9. CLEANING THE DATAFRAME

After the dataframe has been successfully created now we can drop and clear the unnecessary rows. In this section, we will drop index 0–6, 222–228, then resetting the index, and drop the ‘#’ column.

```
# Drop and clearing unnecessary rows
mydata.drop(mydata.index[0:7], inplace=True)
mydata.drop(mydata.index[222:229], inplace=True)
mydata.reset_index(inplace=True, drop=True)

# Drop “#” column
mydata.drop(‘#’, inplace=True, axis=1)
```

Output:

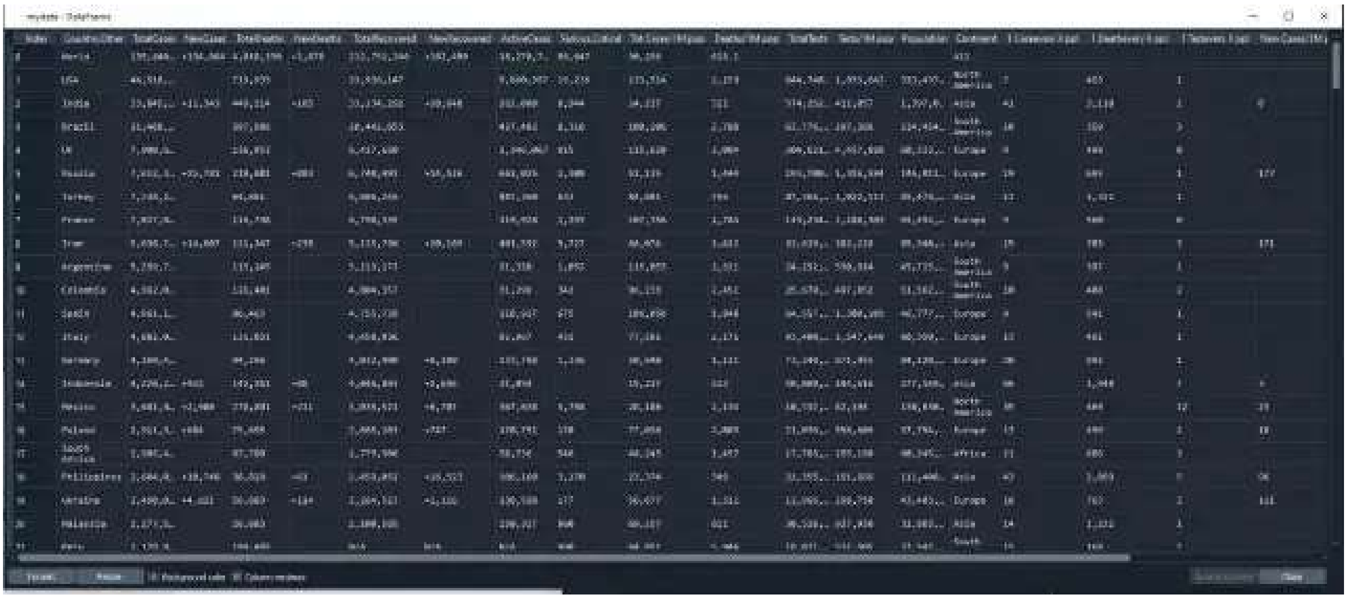


Fig.10 Final dataframe

STEP 10. EXPORT TO CSV AND TRY TO RUN IT

After the dataframe is finished so the next thing we can do is export it into CSV format.

```
# Export to csv
mydata.to_csv('covid_data.csv', index=False)

# Try to read csv
mydata2 = pd.read_csv('covid_data.csv')
```

Since the dataframe has been exported into CSV now we can use the data to build a data science project, put it into machine learning, data analysis project, and so on.

That's the tutorial I gave, hopefully, it will be useful for you guys especially for you who are learning web scraping. See you again in the next project.

Web Scraping Data Science Data Engineering Beautifulsoup

Sign up for Analytics Vidhya News Bytes

By Analytics Vidhya

Latest news from Analytics Vidhya on our Hackathons and some of our best articles! Take a look.

Your email

Get this newsletter

By signing up, you will create a Medium account if you don’t already have one. Review our Privacy Policy for more information about our privacy practices.

About Help Terms Privacy