**Onepicture of web Scrapping**

**<p>**

**<h2>What is Web Scrapping?<h2>**

In today's highly digitized environment, it is becoming increasingly crucial to identify and utilize various methods for accessing and utilizing the vast amounts of data available. Web scraping refers to the practice of extracting information from websites and converting it into a format that can be leveraged for insightful analysis.

**<h3>How does it works?<h3>**

Websites are designed for human consumption, not for computer processing, making it challenging for machines to understand what information is being sought. The URL is utilized to extract the HTML tag content of a website, which contains the design, content, and data being sought.

More advanced websites are constructed using CSS and various Java frameworks, making the extraction of data more complex. Web scraping involves harvesting all the data on the website and then using programming techniques to filter and identify the relevant information.

After data collection, it is frequently converted into a CSV or Excel format for further analysis.

**<h3> Ways of Scrapping <h3>**

Web scraping can be performed using various programming language packages, such as Python or R. Additionally, programmers may also utilize available APIs to extract data in a more organized and structured format

**</p>**

**<h1>In Our project</h1>**

We have used the libraires of python to scrap the data from website[**https://www.worldometers.info/gdp/gdp-by-country/**](https://www.worldometers.info/gdp/gdp-by-country/)( Worldometers )

<h2>Objective of Scrapping:-<h2>

Our primary goal is to gather the Gross Domestic Product (GDP) information for all countries available on a website and conduct a comparative analysis to assess the global GDP status and determine each country's relative position. To accomplish this, we will utilize web scraping techniques such as BeautifulSoup or Scrapy to extract the necessary data and store it in a well-organized format, such as a CSV file or a database. Subsequently, we will employ data analysis and visualization tools, such as Pandas, Matplotlib, or Seaborn, to perform a comprehensive comparison of the GDP data and present the results through informative visualizations.

**Picture 1 W1**

The packages that we have used are Pandas, NumPy, requests and bs4.

**Pandas**: - Is the package used for data management and manipulation

**NumPy**: - Is the package used for numeric operations

**Request**: - Is the package that is used to send HHTP request by using python on to the host to get the website data, it has different methods like **get ()**, **post ()**, **request ()**

**Bs4**: - Is the python package used to beautify the data

**Picture 2 W2**

By using the function **request.get(URL)** we are sending the request to the websit host for the data and in return, we are getting the data that we are saving in the carriable name “req”. in this syntax request is the package and get() is one of its function.

**req.status\_code :-**if every thing goes correctly and we get the data in return the output of the code req.status\_code will be 200, and now we can use **req.text** to see the data that we have received. In this case, we can say req is the variable holding the object of the bs4 and text is its attribute not function.

**Photo 3 W3**

As you see the result of **req.text**is the data we got from the website it is unclear and unorganized, to organize this data beautifully we use the bs4 package function **BS(req.text,'html.parser')** and save it in the variable soup. The BS() beautify the HTML code we received.

**print(soup.prettify()[:2000])**shows the output of 2000 rows

**Photo 4 W4**

The **soup.title**gives the title tag and its content but as we only need the text inside the tag we add **soup.title.text**in the output we will get the Content of the title tag which is**'GDP by Country - Worldometer'.**

**find() function is used to find the tag in the HTML script but it will only find the 1st tag it sees to find all the tags we use find\_all() function which will give you all the tags on the webpage you ask to search for.**

By giving the command **soup.find\_all("table")**we find all the tabes tags in the webpage and save it in**all\_table**variable**.**After storing the tags we want to see how many table tags we got,**print(len(all\_table))**gives the number of tags we got**.**

The **type1()**function is used to see the type of data we are having, **type(all\_table[0].text)**the data in the 0-index table in the variable **all\_table** is a String type

**Photo 5 w6**

**from IPython.display import HTML,**We are using this library to show the data in HTML table design which is better presentable by using function HTML**(table [0]. prettify ())**

**<h2>Transferring the data to CSV <h2>**

To use the data efficiently we need to transform the data into CSV format, so that we can use it to generate more insides and make more analysis

**Photo 6 w7**

So now we will pick the data row-wise and store it into a list by using list comprehension.

**rows=[row for row in table[0].find\_all('tr')]**from table one we are finding all tr tags and putting them in the row variable and saving it in the list named **rows.**

**print(rows[1].text)**we see the data in the 1 row, as we know that rows are a list and to see the data we need to use the indexing concept if we will write **print(rows[0].text)**we will be getting the names of columns.

**Photo 7 W8**

**col\_heads=[col.get\_text().strip() for col in rows[0].find\_all('th')] –**By find\_all(‘th’) we find the name of all table headings and sending in the carriable **col,**before saving the data into a list format in **col\_heads** we will split the names by using**.strip(). As we see the output a list is created with the names of columns**

**photo 8 w9**

Now we know of splitting the data of our rows and column name in different variables by using the slicing concept, **data\_rows = rows[1**:] from row 1 to end will be saved in**data\_rows**

**Photo 9 w11**

Now as we see we separated the data and the column's name from each other but before converting it into a data frame we need to split the data of **td** into a different list and transpose it.

**value\_rows = [[col.get\_text().strip() for col in row.find\_all('td')] for row in data\_rows]**

in the loop first, it picks the data from**data\_rows(tr)**one by one in a variable**row**and sends it to the inner loop where on the bases of the**tr**tag the data is separated and then striped and a new list variable is created as**value\_rows.**

**Photo 9 w12**

Nowto transpose the data we need to convert it onto an array and for that, we will have to use the **NumPy**library. Why do we need to Transpose? As the data in the list will be converted as the data in the column we what that every list to contain similar data, for example, there should be a list of countries' names.

By using the function of NumPy array() **“np.array()”** we convert the list into an array and we the top 5 records **np\_value\_rows[:5]**

**Photo 10 w13**

**.T**transpose the data is the same as pivoting table in **MS Excel,**as the result of command**np\_value\_rows[:5].T**we see that all the similar data is in a single list.

**Photo 10 w14**

We zip up the data together of list by **series\_list=list(zip(\*value\_rows))**

**Photo 11 w15**

We zip up the column name in **dict format** together

By using the Pandas library we convert the data into Dataframe **df= pd.DataFrame(data)** which is a structured format to be converted into CSV, df.to\_csv("GDP.csv")

**Photo 12 w16**

We read the CSV data which is saved in our computer with the help of Pandas library

**df= pd.read\_csv().**

But as we see this data is unworthy to be analyzed as it is uncleaned, we will process and clean the data in our next project **<a>** Data Cleaning and Analysis**</a>**