**WEB SCRAPPING USING PYTHON(BEAUTIFUL SOUP)**

**\*\*Web Scraping Using Python (BeautifulSoup) – Step-by-Step with Explanations\*\***

**✅ Step 1: Choose a Website to Scrape**

We selected the following Wikipedia page to extract data:

<https://en.wikipedia.org/wiki/List_of_Fortune_500_computer_software_and_information_companies>

**Explanation**:  
This page contains a table listing Fortune 500 software and information companies. It is ideal for scraping because it uses a standard HTML table layout.

**✅ Step 2: Install Required Libraries**

!pip install beautifulsoup4

!pip install pandas

!pip install requests

**Explanation**:  
We install the three essential libraries:

* beautifulsoup4: For parsing HTML.
* pandas: For working with tabular data.
* requests: For sending HTTP requests to fetch website content.

You only need to install these once in your environment.

**✅ Step 3: Import Libraries**

from bs4 import BeautifulSoup

import pandas as pd

import requests

**Explanation**:  
We import:

* BeautifulSoup from bs4 to parse and search HTML content.
* pandas for creating and manipulating tables.
* requests to download HTML content from the web.

**✅ Step 4: Define the Target URL**

url = "https://en.wikipedia.org/wiki/List\_of\_Fortune\_500\_computer\_software\_and\_information\_companies"

**Explanation**:  
We store the target website’s URL in a variable named url for later use in sending a request.

**✅ Step 5: Fetch Webpage Content**

response = requests.get(url)

**Explanation**:  
We send an HTTP GET request to the webpage and store the server’s response (including the HTML code) in the response variable.

**✅ Step 6: Check Response Status**

response

**Explanation**:  
This shows whether the request was successful. If it prints <Response [200]>, it means the page loaded correctly and is ready to be parsed.

**✅ Step 7: Parse the HTML Content**

soup = BeautifulSoup(response.content, 'html.parser')

**Explanation**:  
We pass the raw HTML content from the response into BeautifulSoup using the built-in HTML parser. The result is a structured soup object we can navigate easily.

**✅ Step 8: Locate the Table**

table = soup.find('table', class\_='wikitable')

**Explanation**:  
We search for the first <table> element with the class wikitable, which contains the company data we want to extract.

**✅ Step 9: Extract Column Headers**

header = []

for th in table.find\_all('th'):

header.append(th.text.strip())

**Explanation**:  
We loop through each <th> (table header cell), extract and clean the text using .strip(), and add each column name to the header list.

**✅ Step 10: Extract Table Rows**

data = []

for tr in table.find\_all('tr')[1:]: # Skip the header row

row = []

for td in tr.find\_all('td'):

row.append(td.text.strip())

data.append(row)

**Explanation**:  
We skip the header row and loop through each remaining row (<tr>). Then we collect all data cells (<td>), clean the text, and store each row in the data list.

**✅ Step 11: Convert to DataFrame**

df = pd.DataFrame(data=data, columns=header)

**Explanation**:  
We use pandas to convert the data list and header list into a DataFrame, which organizes everything into a neat, readable table.

**✅ Step 12: View the Data**

df.head()

**Explanation**:  
This shows the first five rows of the DataFrame so we can verify that the data was scraped and structured properly.