Lecture 2: **Web Scraping with Python -** Beautiful Soup

Introduction

- Beautiful Soup (bs4) is a Python library used for web scraping and parsing **HTML** and **XML** documents, making it easier to **extract data** from **web pages**.
- HTML document consists of tree of tags. Bs4 creates **parse tree** in a hierarchy to extract information seamlessly.
- Developed by **Leonard Richardson.**
- **Open-source** and widely used for extracting data from web pages.
- Beautiful Soup (bs4) is widely recognized for its **simplicity**
- Provides extensive **documentation** with helpful community

Why Beautiful Soup(bs4) ?

01

Necessity

The need for web Scraping

02

Data Extraction

Manual data extraction vs automation

03

Handeling Tasks

Bs4 simplifies web scraping tasks

04

Time & Cost

Saves time, effort and cost

Features of Beautiful Soup

Parsing

HTML and XML parsing

Navigation

Navigation and searching the parse tree

Analyse & research

- Built-in Parser options
- Modifying the parse tree

Encoding

Unicode, encodings and character conversion

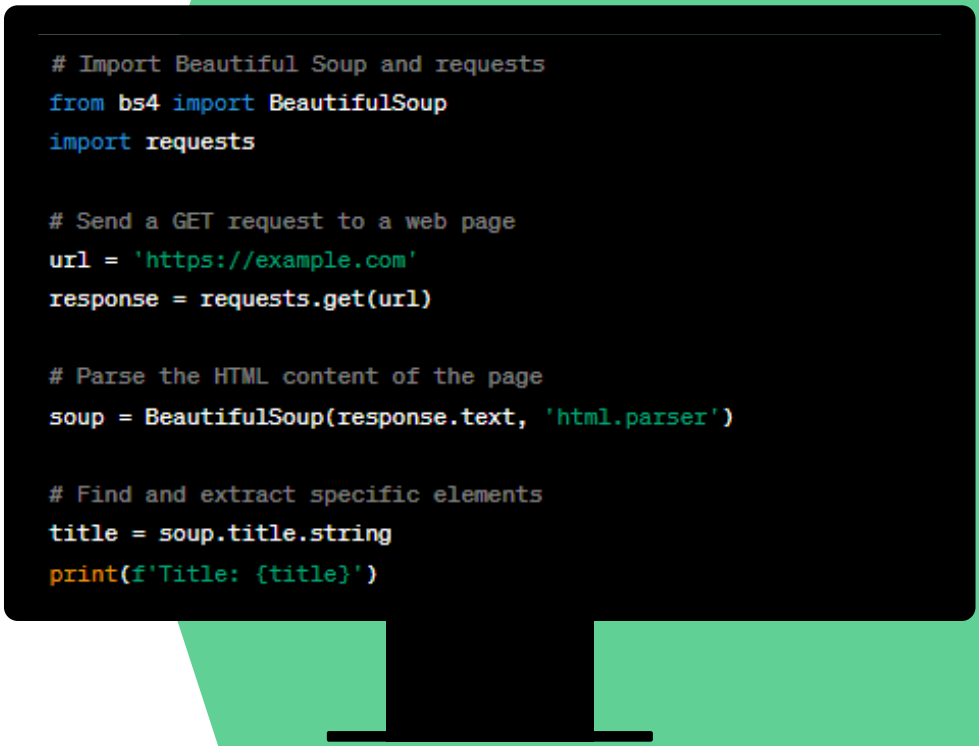
Advantages 

- User-friendly and easy to learn
- Handles poorly formatted HTML
- Supports popular parsers (HTML5lib, lxml, and more)
- Offers a tree-like structure for easy navigation
- Works well with other Python libraries (requests)

Disadvantages 

- Slower performance compared to alternatives (Scrapy). It is not suited for large scale applications.
- Limited JavaScript support (not suitable for dynamic websites) and to extract data via API.
- It does not support well asynchronous scraping with multiple requests simultaneously and has limited proxy support.
- May not handle very complex HTML structures efficiently and has limited extensions.

Basic Example/Syntax



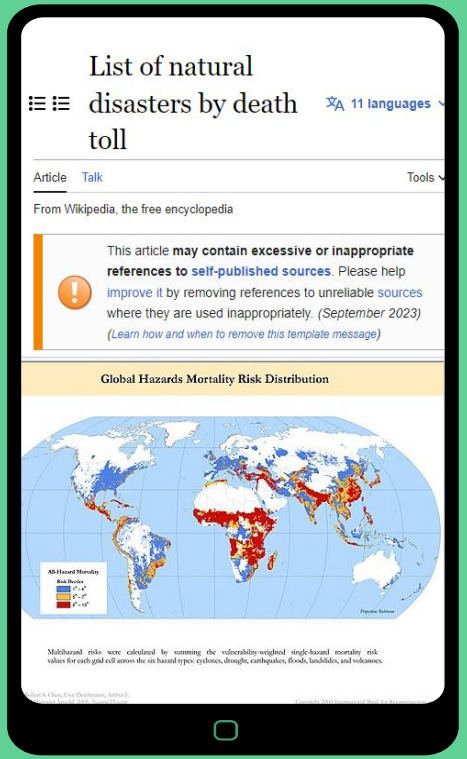
```python
# Import Beautiful Soup and requests
from bs4 import BeautifulSoup
import requests

# Send a GET request to a web page
url = 'https://example.com'
response = requests.get(url)

# Parse the HTML content of the page
soup = BeautifulSoup(response.text, 'html.parser')

# Find and extract specific elements
title = soup.title.string
print(f'Title: {title}')
```

Getting best of Web Scraping 

- **Practical applications:** data collection, research, automation.
- **Recommended prerequisites:** Python basics, HTML/CSS understanding.
- **Practice:** Start with simple websites, gradually work with complex pages.
- **Ethical considerations:** respect website terms of use, robots.txt, and copyright laws.

Quick tips for effective Web Scraping

- Respect website policies (**robots.txt**)
- Set **user-agent to mimic** a real user
- Use **try-except** for error handling
- Limit the rate of requests to **avoid overloading servers**
- Regularly check and **update the code** based on website evolution



Wikipedia Example

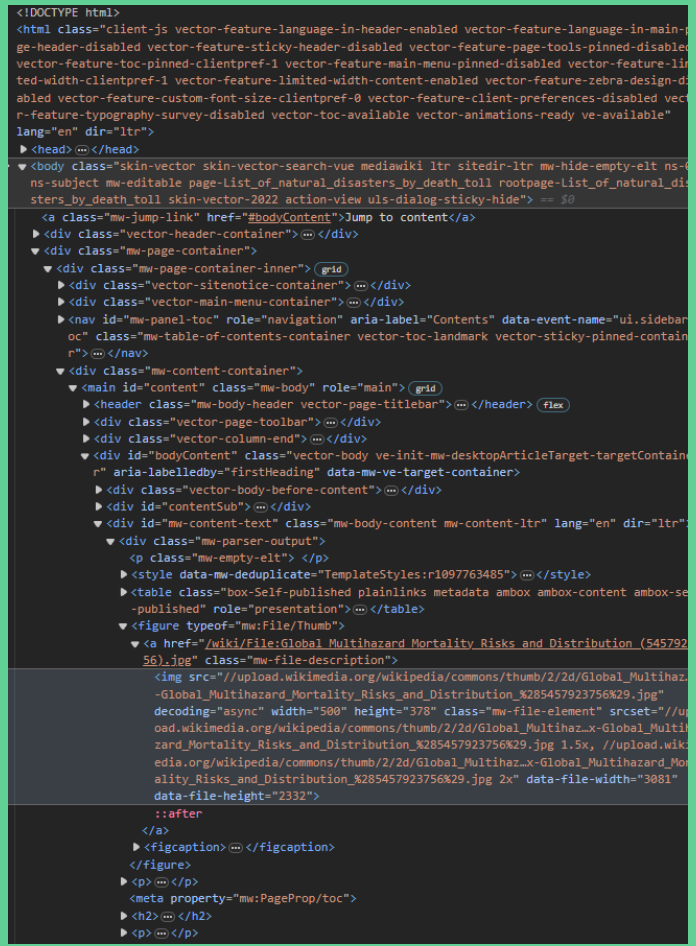
Visit Wikipedia: "https://en.wikipedia.org/wiki/List_of_natural_disasters_by_death_toll"

About Wikipedia

- Wikipedia is a free online encyclopedia.
- Created and edited by volunteers worldwide.
- Covers a vast range of topics.
- Launched in 2001 by Jimmy Wales and Larry Sange.

Structure Of Wikipedia

- Consists of articles on various subjects
- Articles are organized by categories
- Hyperlinked for easy navigation
- Allows users to edit and contribute



Wikipedia Project

- **Objective**: The goal of this project is to scrape data from the following **Wikipedia page** linked to **climate impact** "List of natural disasters" to extract information about the deadliest natural disasters by year, excluding epidemics and famines. We will focus on the data related to natural disasters that occurred in the 20th and 21st centuries. The extracted data will be organized and stored in a CSV file for further analysis or reference.

Project Steps

❑ **Web Scraping:**

- Utilize web scraping tools and libraries like Beautiful Soup and requests in Python for data extraction from the Wikipedia page.
- Focus on sections of the page containing information about natural disasters, excluding epidemics and famines.

❑ **Data Cleaning and Filtering:**

- After scraping, clean the data to eliminate unwanted characters and formatting.
- Filter natural disasters by year, excluding epidemics and famines, in the 20th and 21st centuries, potentially using regular expressions.

❑ **Data Organization:**

- Organize data into a structure, e.g., a list of dictionaries, with details such as year, natural disaster type, location, and death toll.

Project Steps

❑ **CSV File Creation:**

- Create a CSV file using Python CSV library or pandas.
- Write organized data frame to the CSV, ensuring proper headers for each column.

❑ **Data Analysis and Visualization:**

- Perform data analysis or create visualizations, generate insights to depict natural disaster over a period of time, impact by type and affected countries.

The extracted information can be used for educational purposes, awareness campaigns, or as a historical reference.

Benefits and applications

❖ **Research and Analysis:**
- Valuable resource for researchers and scholars studying natural disasters.
- Provides data for in-depth analysis of patterns and trends in recent history.

❖ **Historical Reference:**
- Serves as a historical reference for historians interested in the impact of natural disasters over time.

❖ **Disaster Management:**
- Supports disaster management professionals in understanding historical disasters.
- Enhances preparedness and response strategies by learning from past events.

❖ **Educational Materials:**
- Data can be used to create educational materials and resources.
- Helps educate students, the public, and future disaster management professionals.

Challenges of the Project

- **Page Structure Changes:**

- Wikipedia page structures are not static and may evolve over time.
- The scraping code might require regular updates to adapt to these changes, ensuring it remains effective in data extraction.

- **Data Accuracy and Consistency:**

- The presence of varying formatting on the Wikipedia page can pose challenges.
- Ensuring the accuracy and consistency of the extracted data may require additional effort in data cleaning and verification.

Sample Script

```
<td><span data-sort-value="000000001920-12-16-0000" style="white-space:nowrap">December 16, 1920</span>
</td></tr>
</tbody></table>
<h2><span class="mw-headline" id="Deadliest_natural_disasters_by_year_excluding_epidemics_and_famines">Deadliest natural disasters by year exclu
<h3><span class="mw-headline" id="20th_century">20th century</span><span class="mw-editsection"><span class="mw-editsection-bracket">[</span><a
<table class="wikitable sortable mw-collapsible" style="font-size:100%">

<tbody><tr>
<th>Year
</th>
<th data-sort-type="number">Death toll
</th>
<th>Event
</th>
<th>Countries affected
</th>
<th>Type
</th>
<th>Date
</th></tr>
<tr>
<td>1900
</td>
<td>6,000–8,000
</td>
<td><a href="/wiki/1900_Galveston_hurricane" title="1900 Galveston hurricane">1900 Galveston hurricane</a>
</td>
<td>United States
</td>
<td>Tropical cyclone
</td>
<td>September 9
</td></tr>
<tr>
<td>1901
</td>
<td>9,500
</td>
<td><a href="/wiki/1901_eastern_United_States_heat_wave" title="1901 eastern United States heat wave">1901 eastern United States heat wave</a>
</td>
<td>United States
</td>
<td>Heat wave
</td>
<td>June–July
</td></tr>
<tr>
<td>1902
</td>
<td>29,000
</td>
<td><a href="/wiki/1902_eruption_of_Mount_Pel%C3%A9e" title="1902 eruption of Mount Pelée">1902 eruption of Mount Pelée</a>
</td>
<td>Martinique
```

```python
response = requests.get(url, headers={'User-Agent': get_random_user_agent()})
response.raise_for_status()  # Check for any request errors

# Parse the HTML content of the page with BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Find all tables with the same class
tables = soup.find_all('table', {'class': 'wikitable sortable mw-collapsible'})
```

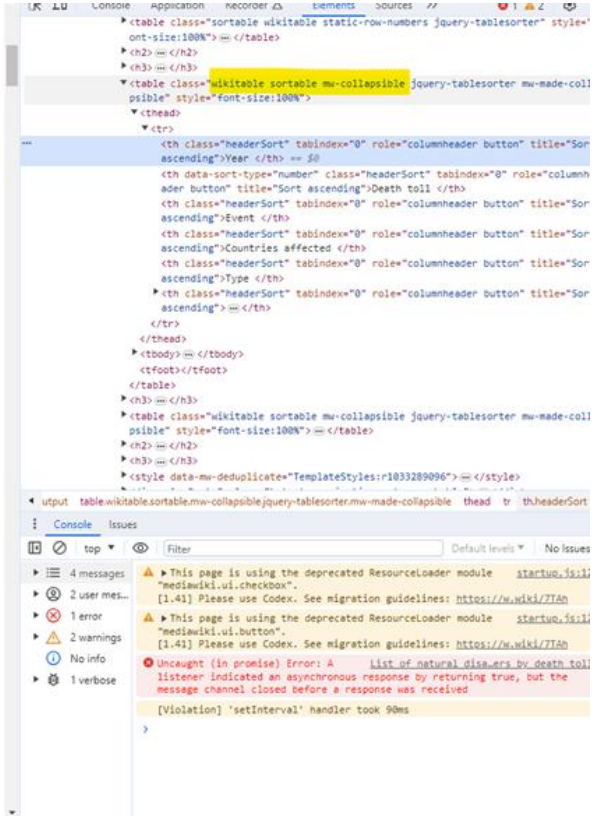
Sample Script

3	655,000[5]	1976 Tangshan earthquake		July 28, 1976
4	500,000[6][1]	1970 Bhola cyclone	East Pakistan (now Bangladesh)	November 13, 1970
5	316,000[7]	2010 Haiti earthquake	Haiti	January 12, 2010
6	300,000[8]	526 Antioch earthquake	Byzantine Empire (now Hatay/Turkey)	May 526
7	≈300,000[9][10]	1839 Coringa cyclone	Andhra Pradesh, India	November 25, 1839
8	≈300,000[11]	1737 Calcutta cyclone	Bengal, India	October 1737
9	≈300,000[12]	1139 Ganja earthquake	Seljuk Empire (present-day Azerbaijan)	September 30, 1139
10	273,400[13]	1920 Haiyuan earthquake	China	December 16, 1920

Deadliest natural disasters by year excluding epidemics and famines [edit]

20th century [edit]

Year	Death toll	Event	Countries affected	Type	Date [hide]
1900	6,000–8,000	1900 Galveston hurricane	United States	Tropical cyclone	September 9
1901	9,500	1901 eastern United States heat wave	United States	Heat wave	June–July
1902	29,000	1902 eruption of Mount Pelée	Martinique	Volcanic eruption	April–August
1903	3,500	1903 Manjzikert earthquake	Turkey	Earthquake	April 28
1904	400	1904 Sichuan earthquake	China		August 30
1905	20,000+	1905 Kangra earthquake	India		April 4
1906	15,000	1906 Hong Kong typhoon	China	Tropical cyclone	September 18
1907	12,000–15,000	1907 Qaratog earthquake	Uzbekistan	Earthquake	October 21
1908	75,000–82,000	1908 Messina earthquake	Italy		December 28
1909	6,000–8,000	1909 Borujerd earthquake	Iran		January 23
1910	2,450	1910 Costa Rica earthquakes	Costa Rica		May 4
1911	41,072[14]	1911 France heat wave	France	Heat wave	June–August
1912	50,000–220,000	1912 China typhoon	China	Tropical cyclone	August 29
1913	1,900	1913 Yunnan earthquake	China		December 21



Sample Script

```
try:
    response = requests.get(url, headers={'User-Agent': get_random_user_agent()})
    response.raise_for_status()  # Check for any request errors

    # Parse the HTML content of the page with BeautifulSoup
    soup = BeautifulSoup(response.content, 'html.parser')

    # Find all tables with the same class
    tables = soup.find_all('table', {'class': 'wikitable sortable mw-collapsible'})

    if len(tables) >= 2:
        # Initialize a list to store data
        disasters = []

        for table in tables:
            # Loop through the rows of each table
            for row in table.find_all('tr')[1:]:  # Skip the header row
                columns = row.find_all('td')
                if len(columns) >= 6:
                    year = columns[0].text.strip()
                    death_toll = columns[1].text.strip()
                    event = columns[2].text.strip()
                    countries_affected = columns[3].text.strip()
                    event_type = columns[4].text.strip()
                    date = columns[5].text.strip()

                    # Append the data to the list
                    disasters.append([year, death_toll, event, countries_affected, event_type, date])

        # Save the data as a CSV file with 'utf-8' encoding
        with open('natural_disasters.csv', 'w', newline='', encoding='utf-8') as csvfile:
            csv_writer = csv.writer(csvfile)
            csv_writer.writerow(['Year', 'Death Toll', 'Event', 'Countries Affected', 'Type', 'Date'])
            csv_writer.writerows(disasters)

        # print confirmation
        print("Data saved as 'natural_disasters.csv'")

        # Create a DataFrame from the scraped data
        df = pd.DataFrame(disasters, columns=['Year', 'Death Toll', 'Event', 'Countries Affected', 'Type', 'Date'])
        display(df.head()) # Displaying head of the dataframe
    else:
        print("Tables not found on the page. Check if the page structure has changed or if there are at least two tables with the same class.")

except requests.exceptions.RequestException as e:
    print("Request error:", e)

except Exception as e:
    print("An error occurred:", e)
```

✓ 1.0s

```
Data saved as 'natural_disasters.csv'
```

	Year	Death Toll	Event	Countries Affected	Type	Date
0	1900	6,000–8,000	1900 Galveston hurricane	United States	Tropical cyclone	September 9
1	1901	9,500	1901 eastern United States heat wave	United States	Heat wave	June–July
2	1902	29,000	1902 eruption of Mount Pelée	Martinique	Volcanic eruption	April–August
3	1903	3,500	1903 Manzikert earthquake	Turkey	Earthquake	April 28
4	1906	15,000	1906 Hong Kong typhoon	China	Tropical cyclone	September 18

Removing the Special Characters from the 'Death Toll' Column and choosing only lower Bound of the Range EX: 6000-9000 -> 6000

```
# Function to extract numeric values from the "Death Toll" column (retaining only the lower bound of the range)
def extract_numeric(death_toll):
    # Use regular expressions to remove non-numeric characters and retain only the lower bound of the range
    match = re.search(r'(\d+(?:,\d+)?)', str(death_toll))
    return int(match.group(1).replace(',', '')) if match else 0

# Apply the function to the "Death Toll" column
df['Death Toll'] = df['Death Toll'].apply(extract_numeric)

# Display the preprocessed DataFramed
display(df.head())
```

✓ 0.0s

	Year	Death Toll	Event	Countries Affected	Type	Date
0	1900	6000	1900 Galveston hurricane	United States	Tropical cyclone	September 9
1	1901	9500	1901 eastern United States heat wave	United States	Heat wave	June–July
2	1902	29000	1902 eruption of Mount Pelée	Martinique	Volcanic eruption	April–August
3	1903	3500	1903 Manzikert earthquake	Turkey	Earthquake	April 28
4	1906	15000	1906 Hong Kong typhoon	China	Tropical cyclone	September 18

Impact overview by country

Thanks