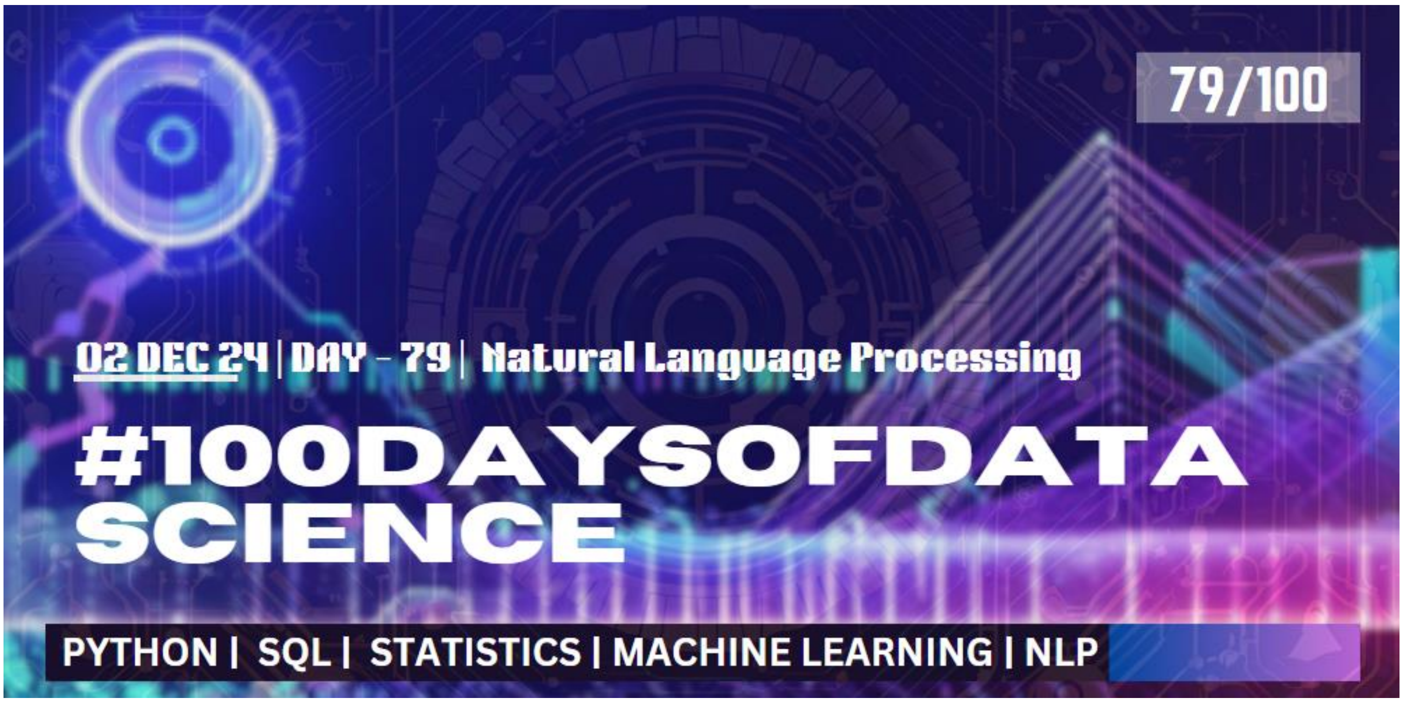

79/100
02 DEC 24 | DAY - 79 | Natural Language Processing
#100DAYSOFDATA SCIENCE
PYTHON | SQL | STATISTICS | MACHINE LEARNING | NLP

Web Scraping

What is Web Scraping?
Web scraping is the process of extracting data from websites. Using tools like **BeautifulSoup**, we can automate the retrieval and organization of structured data, transforming web pages into usable datasets for analysis or integration into applications.

Key Features of BeautifulSoup

1. **HTML Parsing**:
   BeautifulSoup enables seamless parsing of HTML and XML documents, allowing you to navigate, search, and modify the document structure with ease.
2. **Navigating HTML Trees**:
   Retrieve specific elements using tags, attributes, and classes. For instance, locate all <div> tags or extract data from <table> structures.
3. **Data Cleaning**:
   Simplifies the extraction of clean, usable data by removing unwanted elements like JavaScript, advertisements, or excessive formatting.
4. **Encoding Compatibility**:
   Handles various character encodings and broken HTML documents efficiently.

Applications of Web Scraping

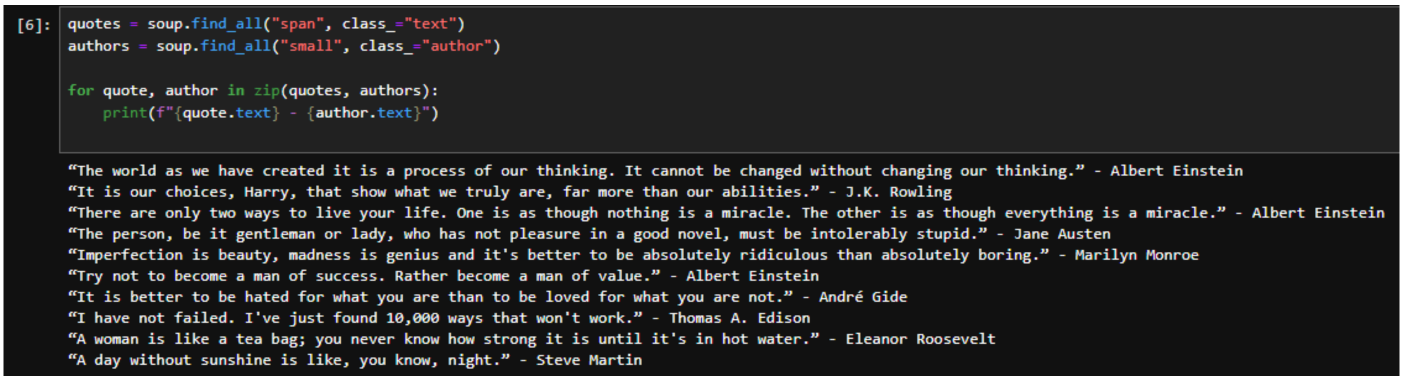
- **Market Research**: Scrape product prices, reviews, and competitor data from e-commerce sites.
- **Content Aggregation**: Gather articles, news, or blog posts for curated platforms.
- **Academic Research**: Extract data for datasets in fields like social sciences or economics.
- **Sentiment Analysis**: Collect social media or review data for natural language processing tasks.

Best Practices for Web Scraping

- **Follow Website Policies**: Always check the site's **robots.txt** to ensure compliance with their scraping rules.
- **Limit Request Rates**: Use delays to avoid overloading servers or getting blocked.
- **Handle Exceptions**: Build error-handling logic for missing or changed elements.

Commonly Used Methods in BeautifulSoup

1. **find() and find_all()**: Locate single or multiple elements by tag name.
   Example: soup.find('a') retrieves the first <a> tag.

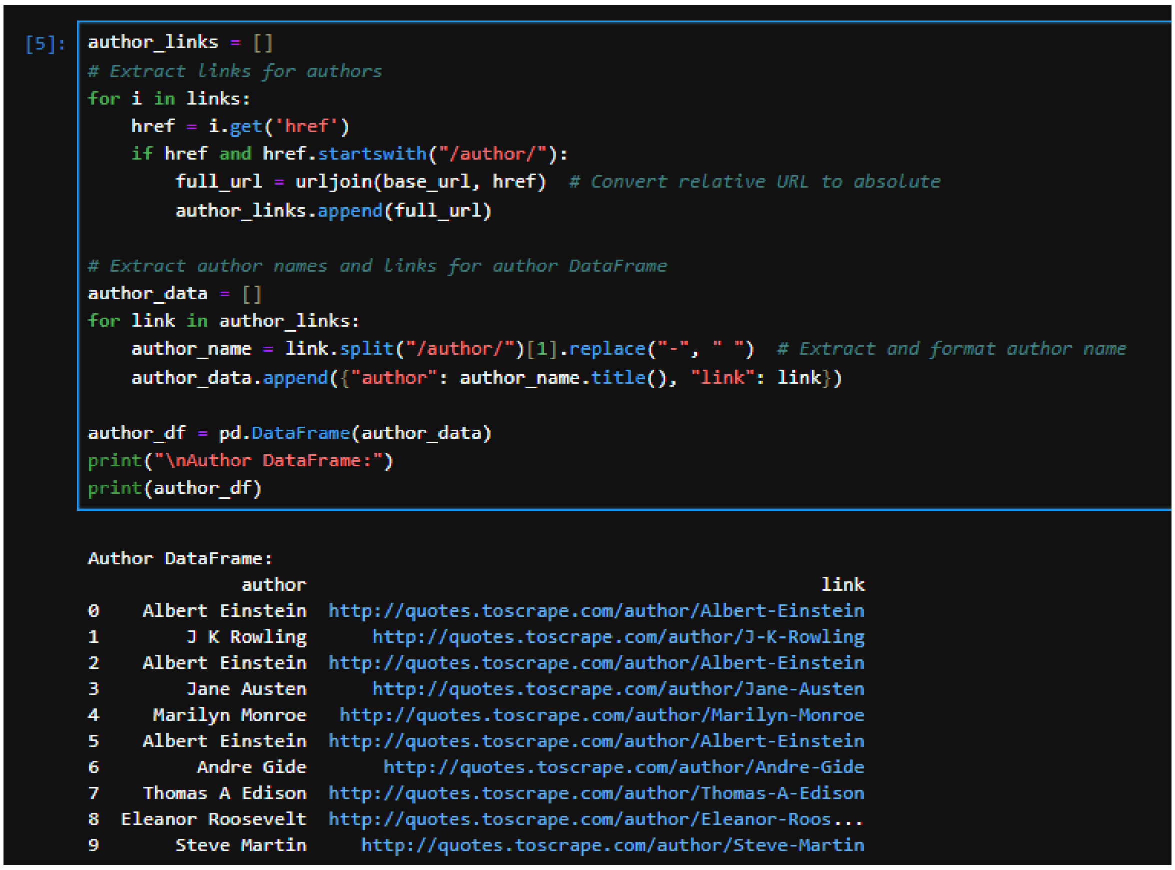


```
[6]: quotes = soup.find_all("span", class_="text")
     authors = soup.find_all("small", class_="author")

     for quote, author in zip(quotes, authors):
         print(f"{quote.text} - {author.text}")
```

```
“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.” - Albert Einstein
“It is our choices, Harry, that show what we truly are, far more than our abilities.” - J.K. Rowling
“There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle.” - Albert Einstein
“The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.” - Jane Austen
“Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring.” - Marilyn Monroe
“Try not to become a man of success. Rather become a man of value.” - Albert Einstein
“It is better to be hated for what you are than to be loved for what you are not.” - André Gide
“I have not failed. I've just found 10,000 ways that won't work.” - Thomas A. Edison
“A woman is like a tea bag; you never know how strong it is until it's in hot water.” - Eleanor Roosevelt
“A day without sunshine is like, you know, night.” - Steve Martin
```

2. **CSS Selectors**: Use select() to find elements by class or ID.
   Example: soup.select('.class-name') locates elements with a specific class.
3. **Attribute Access**: Extract attributes like href or src.
   Example: link['href'] retrieves the URL from an anchor tag.



```
[5]: author_links = []
     # Extract links for authors
     for i in links:
         href = i.get('href')
         if href and href.startswith("/author/"):
             full_url = urljoin(base_url, href)  # Convert relative URL to absolute
             author_links.append(full_url)

     # Extract author names and links for author DataFrame
     author_data = []
     for link in author_links:
         author_name = link.split("/author/")[1].replace("-", " ")  # Extract and format author name
         author_data.append({"author": author_name.title(), "link": link})

     author_df = pd.DataFrame(author_data)
     print("\nAuthor DataFrame:")
     print(author_df)
```

```
Author DataFrame:
              author                                            link
0    Albert Einstein  http://quotes.toscrape.com/author/Albert-Einstein
1         J K Rowling      http://quotes.toscrape.com/author/J-K-Rowling
2    Albert Einstein  http://quotes.toscrape.com/author/Albert-Einstein
3        Jane Austen      http://quotes.toscrape.com/author/Jane-Austen
4     Marilyn Monroe   http://quotes.toscrape.com/author/Marilyn-Monroe
5    Albert Einstein  http://quotes.toscrape.com/author/Albert-Einstein
6         Andre Gide       http://quotes.toscrape.com/author/Andre-Gide
7    Thomas A Edison  http://quotes.toscrape.com/author/Thomas-A-Edison
8  Eleanor Roosevelt  http://quotes.toscrape.com/author/Eleanor-Roos...
9       Steve Martin     http://quotes.toscrape.com/author/Steve-Martin
```

Common Libraries Used Alongside BeautifulSoup

1. **Requests**: For sending HTTP requests to retrieve web pages.
2. **Pandas**: For converting scraped data into structured formats like CSV or Excel.

When to Use BeautifulSoup

- **Ideal For**: Static web pages, lightweight tasks, and simpler scraping needs.
- **Avoid For**: Highly dynamic or JavaScript-heavy websites (use Selenium or Puppeteer instead).

BeautifulSoup provides an efficient, beginner-friendly way to start web scraping and extract valuable insights from the web. Mastering this tool opens doors to limitless possibilities in data analysis and automation. ✨