Scraping Data from a Real Website | Web Scraping in Python
Advanced Web Scraping Tutorial! (w/ Python Beautiful Soup Library)

Week 1 Goals:

- Plan an analysis of financial reports: Use web scraping tools like BeautifulSoup to extract data from a company's financial reports. Use pandas to analyze key metrics like revenue, profit, and earnings per share. - 30 Points

- Draft and Start an outline with code for this project - 70 Points

Week 2 Goals:

- Finish analysis of financial reports using BeautifulSoup and pandas. Probably will use dataset regarding countries' GDPs - 100 Points

Financial Report Analysis Notes:

1. Import Required Libraries: Import necessary libraries such as BeautifulSoup for web scraping and pandas for data analysis.
2. Define Target Company and Reports: Specify the target company and the URLs of the financial reports to be scraped.
3. Web Scraping with BeautifulSoup: Use BeautifulSoup to scrape the HTML content of the financial reports.
4. Extract Key Financial Metrics: Identify and extract key metrics such as revenue, profit, and earnings per share from the scraped data.
5. Data Cleaning and Preparation: Clean and prepare the extracted data for analysis, handling missing values and formatting issues.
6. Analyze Key Metrics with Pandas: Use pandas to analyze the cleaned data and calculate key financial metrics.
7. Visualize Financial Data: Create visualizations to represent the financial metrics using libraries like Matplotlib or Seaborn.

Requests and BeautifulSoup - two of the most commonly used Python libraries for web scraping. Requests is responsible for accessing a webpage, while BeautifulSoup is responsible for parsing and extracting useful information from that page.

Requests Library (Overview)

The Requests library allows a Python program to communicate with websites over the internet. It sends HTTP requests (such as GET or POST) to a webpage and receives the webpage's response. The response can include the page's HTML content, its status code, information about the server, and sometimes data in formats like JSON.

When performing web scraping, the first step is typically to use Requests to retrieve the HTML source code of a webpage. The library simplifies this process by handling network communication, managing query parameters, headers, cookies, and error checking. A successful request usually returns a status code of 200, which means the webpage loaded correctly. If a different status code appears, it often indicates an issue such as a missing page or a blocked request.

Requests also allows you to customize how you appear to the website by adding headers, such as identifying yourself as a real browser. This can prevent the website from blocking your scraping attempt. The library can handle timeouts, redirects, and authentication, making it flexible for many scraping situations.

BeautifulSoup Library (Overview)

BeautifulSoup, part of the bs4 package, is a library used for parsing HTML or XML documents. After the HTML content has been retrieved with Requests, BeautifulSoup takes that raw HTML and converts it into a structured format that is easier to navigate and search. It essentially turns the messy, nested tags of a webpage into a readable tree structure.

BeautifulSoup allows you to locate specific elements on a webpage by searching for tags, classes, and IDs. For example, if you want to extract all headlines, all links, or specific information like prices or titles, BeautifulSoup makes it possible to search the HTML and isolate those parts. It can also extract text from within tags, remove HTML markup, and access attributes like hyperlinks or image sources.

Another key advantage of BeautifulSoup is its flexible searching system. It supports searching by tag names, filtering by attributes, and using CSS selector syntax to pinpoint elements. This makes it versatile and workable even when the HTML structure is complex or poorly formatted.

How They Work Together

In a typical web scraping workflow, Requests retrieves the raw webpage, and BeautifulSoup parses and extracts the necessary pieces of information. Together, they form a powerful combination for turning website content into structured data that can be used for analysis, research, automation, or storage.