Web Scraping - BeautifulSoup:

Web Scraping:

- A web scraping tool refers to a software program crafted for the extraction, or 'web scraping,' of pertinent data from websites. In the process of collecting relevant information from websites, it is highly likely that you will employ a web scraper to extract specific datasets. This scraping tool, also known as a website scraper, plays a crucial role in the web scraping process by making HTTP requests to a target website and extracting data from its pages. It effectively parses content that is publicly accessible, visible to users, and rendered by the server in the form of HTML.

- Web scraping, also known as web data extraction, involves the automated collection of both structured and unstructured data from the internet. It serves various purposes, including price monitoring, price intelligence, news tracking, lead generation, and market research. Individuals and businesses leverage web scraping to extract valuable insights from publicly available web data, enabling them to make informed decisions. If you've ever manually copied and pasted information from a website, you've essentially performed a similar function as a web scraper. However, web scraping distinguishes itself by utilizing machine learning and intelligent automation to efficiently retrieve vast amounts of data from the vast expanse of the internet, eliminating the tedious manual extraction process. Whether you choose to employ a web scraper or collaborate with a web data extraction partner, understanding the basics of web scraping is crucial for successful data retrieval.

Requests

- In Python, the "requests" module is a popular library used for making HTTP requests. It provides a simple and convenient way to interact with web services, allowing users to send HTTP requests and handle the corresponding responses. The requests module supports various HTTP methods, such as GET, POST, and others, making it versatile for tasks like fetching data from web servers or interacting with web APIs.

```
!pip3 install requests
```

BeautifulSoup:

- Beautiful Soup, a Python library, is designed for extracting data from HTML, XML, and various markup languages. In scenarios where webpages present relevant data for research, like date or address information without a direct download option, Beautiful

Soup proves instrumental. This tool aids in retrieving specific content from webpages, eliminating HTML markup, and facilitating the saving of extracted information. Essentially, Beautiful Soup serves as a web scraping tool, assisting in the cleaning and parsing of documents obtained from the web.

Uses of Beautiful Soup:

- Beautiful Soup library proves valuable in isolating titles and links from webpages by efficiently extracting all text within HTML tags. Additionally, it facilitates the modification of HTML within the document under consideration.

Installing BeautifulSoup:

- The installation of Beautiful Soup is a breeze if you already have pip or another Python installer in place. In case pip is not installed, a brief tutorial on installing Python modules can guide you through the process. Once pip is set up, simply run the following command in the terminal to install Beautiful Soup.

```
!pip3 install beautifulsoup4
```

- To import BeautifulSoup

```
from bs4 import BeautifulSoup
```

- Moreover, it is necessary to install a "parser" for interpreting the HTML.

```
!pip3 install lxml
```

Introduction to the DOM:

- The Document Object Model (DOM) serves as the data representation of the elements forming the structure and content of a web document. This guide provides an introduction to the DOM, delving into how it internally represents an HTML document and exploring the use of APIs for creating web content and applications.

- The Document Object Model (DOM) functions as a programming interface for web documents, offering a representation of the page that allows programs to dynamically alter the document's structure, style, and content. Utilizing nodes and objects, the DOM enables interaction between programming languages and the web page.

- A web page exists as a document, viewable in the browser window, or accessible as an HTML source. Despite being the same document in both instances, the Document Object Model (DOM) representation enables manipulation. Serving as an object-oriented depiction of the web page, it becomes alterable through scripting languages like JavaScript.

HTML:

- HTML, an acronym for Hyper Text Markup Language, serves as the fundamental language for crafting web pages and web applications. Let's delve into the definition of Hypertext Markup Language and understand its role in the creation of web pages.

- HyperText refers to "Text within Text," where a text contains links within it, creating a hypertext. Clicking on a link that directs you to a new webpage signifies interacting with a hypertext. This concept enables the linking of two or more web pages (HTML documents) to each other.

- A markup language is a computer language employed to impose layout and formatting conventions on a text document. This language enhances interactivity and dynamism by transforming plain text into elements such as images, tables, links, and more.

- A web page, often authored in HTML and interpreted by a web browser, is a document accessible through a specific URL. Web pages can be categorized as either static or dynamic. Solely utilizing HTML, static web pages can be created.

Tags:

- Tags are used to represent HTML elements. These can be seen as keywords that define how a web browser will format and display the website content.
- The HTML tags are usually available in pairs, i.e. opening and closing (it's the same, with the tag name '/' at the beginning) tag.

Eg: <html> and </html> is a tag that comes in pairs and <hr> does not have a closing tag.
NOTE: There are also "self-closing" tags, whereby a br tag, for eg., will look like "
" instead of simply "
".

Some of the important tags are used in BeautifulSoup:

- **<a> (Anchor):** Useful for extracting hyperlinks, which can lead to data sources
- **<p> (Paragraph):** Commonly used for text content, including textual data that you may want to scrape.

- **<h1>, <h2>, <h3>,...<h6> (Headings):** Often used for titles and section headings that provide structure to web pages and data.
- **<ul>, <ol> (Unordered and Ordered Lists):** Useful for extracting structured lists of data.
- ** (List Item):** Elements within lists, which can contain valuable data.
- **<table>:** Used for presenting tabular data. Essential for scraping structured data tables.
- **<tr> (Table Row):** Rows within tables, containing data points.
- **<tr> (Table Row):** Rows within tables, containing data points.
- ** :** Frequently used as a container for various content, including data.
- **<form>:** Used for web forms, which may be a source of data.
- **<input>:** Found within forms and can hold data that can be extracted.
- **<textarea>:** Often used for multi-line text input, where textual data might be present.
- **<label>:** Linked to form elements, providing descriptions or labels for input fields.
- **<img>:** Useful for extracting image data or checking image attributes.
- **<iframe>:** May contain data or content embedded from other sources.
- **<script>:** Contains JavaScript code, which sometimes loads or manipulates data.
- **<input>:** Used for various input fields like text boxes, radio buttons, and checkboxes.
- **<select>:** Used for dropdown lists, which may contain selectable data.
- **<body>:** contains all the contents of an HTML document, such as title, headings, paragraphs, hyperlinks, tables, lists, etc.

Extra:

To get the list of all valid tags in HTML5, visit:
https://developer.mozilla.org/en-US/docs/Web/HTML/Element

Navigation of the structure in the DOM:

- Parents are the tags that encapsulate an element, forming a hierarchical structure within HTML. In the Document Object Model (DOM), these parent tags contain and surround child elements, creating a tree-like representation. Understanding the parent-child relationship is crucial in navigating and manipulating the structure of HTML documents using languages like JavaScript. It allows developers to access and modify content within specific sections of a webpage, providing flexibility in web development and design.

- Siblings are other tags that share the same parent within an HTML document. In the Document Object Model (DOM), siblings are elements that reside at the same hierarchical level, having a common parent. They are adjacent to each other in the structure of the HTML markup.

- Children refer to the tags contained within a specific HTML element. In the context of the Document Object Model (DOM), children are elements nested inside their parent tag. This relationship creates a hierarchical structure where the parent is the encompassing tag, and the children are the tags residing inside it.

What is Parsing in Web Scraping?

- In the realm of web scraping, parsing refers to the transformation of unstructured data into a more organized format, such as a parse tree, making it simpler to comprehend, utilize, and extract valuable information.

Parse the HTML:

- To initiate the HTML parsing process, generate a BeautifulSoup object and include the HTML to be parsed as a mandatory argument. The resulting soup object will represent a parsed rendition of the provided HTML.

```
import requests
from bs4 import BeautifulSoup
url  = 'https://en.wikipedia.org/wiki/Borders_of_India'
response = requests.get(url).text
soup = BeautifulSoup(response,'html.parser')
print(soup)
```

- This returns the parsed HTML and creates a BeautifulSoup object from the HTML response.

Leverage BeautifulSoup's Object Methods for Extracting Information from HTML.

- The BeautifulSoup library incorporates numerous built-in methods designed for extracting data from HTML. Employ methods such as soup.find() or soup.find_all() to retrieve specific elements from the parsed HTML.

```
import requests
from bs4 import BeautifulSoup
url  = 'https://en.wikipedia.org/wiki/Borders_of_India'
response = requests.get(url).text
soup = BeautifulSoup(response)
for i  in soup.findAll("h2"):
     print(i.text)
```

- We then retrieve any HTML element (the title tag in this case) from the BeautifulSoup object with the soup.findAll() method.

Limitations:

- While BeautifulSoup is a powerful tool, it comes with certain limitations and drawbacks. Notably, it lacks the capability to execute JavaScript or handle dynamic web pages that involve user interaction or AJAX requests. To address these functionalities, one might consider utilizing browser automation tools like Selenium or Scrapy.

- BeautifulSoup also faces limitations when dealing with intricate web scraping tasks, including tasks like crawling multiple pages, managing data storage in databases, and handling errors and retries. To tackle such scenarios, it's advisable to turn to a comprehensive web scraping framework like Scrapy or PySpider. BeautifulSoup excels in simpler, static web scraping tasks, primarily focused on parsing and extracting data from individual web pages.