What is a Web Scraper?

A \*\*web scraper\*\* is a tool or software designed to automatically extract data from websites.

How does a Web Scraper Work?

A web scraper works through a series of steps to extract data from websites. Here's a detailed breakdown of the process:

### Steps Involved in Web Scraping

1. \*\*Making an HTTP Request\*\*:

- The scraper sends an HTTP request to the target website's server to fetch the HTML content of the page. This is similar to what happens when you visit a website in your browser.

2. \*\*Fetching the HTML Content\*\*:

- The server responds with the HTML content of the requested page. This HTML contains the data that the scraper needs to extract.

3. \*\*Parsing the HTML\*\*:

- The scraper parses the HTML content to locate and extract the desired data. This involves navigating the Document Object Model (DOM) structure of the page. Tools like BeautifulSoup (for Python) or Cheerio (for JavaScript) are commonly used for this purpose.

4. \*\*Extracting Data\*\*:

- The scraper identifies the specific HTML elements (like tags, classes, or IDs) that contain the data. It then extracts this data, which can include text, images, links, or other elements.

5. \*\*Storing the Data\*\*:

- The extracted data is then stored in a structured format, such as a CSV file, a database, or a JSON file, for further analysis or use.

Why Selenium is the Used?

Selenium is one of the most popular tools for web automation testing, and there are several reasons for its widespread use:

### Key Reasons for Selenium's Popularity

1. \*\*Open Source\*\*:

- Selenium is free to use, which makes it accessible to a wide range of users, from individual developers to large enterprises¹².

2. \*\*Multi-Language Support\*\*:

- It supports multiple programming languages, including Java, Python, JavaScript, C#, Ruby, and Perl. This flexibility allows developers to write test scripts in the language they are most comfortable with²⁵.

3. \*\*Cross-Browser Compatibility\*\*:

- Selenium can automate browsers like Chrome, Firefox, Safari, Internet Explorer, and Edge. This means you can write one set of tests and run them across different browsers without modification²³.

4. \*\*Platform Independence\*\*:

- It works on various operating systems, including Windows, macOS, and Linux. This cross-platform capability ensures that tests can be run in different environments¹².

5. \*\*Community Support\*\*:

- Being open source, Selenium has a large and active community. This community continuously contributes to its development, provides support, and shares resources, making it easier for new users to get started¹⁴.

6. \*\*Integration with Other Tools\*\*:

- Selenium integrates well with other tools and frameworks, such as TestNG, JUnit, and Maven for test management, and Jenkins for continuous integration/continuous deployment (CI/CD). This makes it a versatile tool in the software development lifecycle²³.

7. \*\*Ease of Use\*\*:

- Selenium's user-friendly interface and extensive documentation make it relatively easy to learn and use, even for those who are new to automation testing¹².

These features collectively make Selenium a powerful and flexible tool for web automation testing, contributing to its popularity worldwide.

Selenium's combination of being open source, supporting multiple programming languages, offering cross-browser and cross-platform compatibility, and having strong community support makes it a go-to choice for many developers and testers around the world.

How does the code works?

To ensure abstraction the code asks for the search keyword the it puts it in a google search url using mozilla firefox browser

then it declares the selenium mozilla firefox driver location for the whole code

then an object was taken from the selenium library

a string that contains url with the search keyword was passed to it

then the selenium library can take control of the browser under the effect of Marionette bot

the bot injects the search string to the search box and hits the search button

after the web page fully loaded the bot starts extracting all of the HTML ‘a’ tags in it and put them in a list

then a new text file is created and opened under the name ‘Links.txt’

a for loop is opened to run into the list elements

then the ‘href’ HTML property value in the tag is extracted from each tag

then the values are injected to the file

after finishing the file is saved and closed

then another marionette bot starts

the bot extracts the tags list from the first bot

a for loop is opened to run into the list elements

then the ‘href’ HTML property value in the tag is extracted from each tag

then searches for each link

then it extracts the ‘title’ HTML tag value

then extracts the whole HTML code of the ‘body’ tag

a text file is opened under the name of the title tag value

the body tag code is injected into the text file

then the file is closed

after the for loop finishes the bots are closed and the ram is cleared

Why coding in this Mechanism?

First of all, the code is divided into two main pieces the main that collects the urls and worker that collects the data so each process starts and ends independently to ensure reasonable usage of the resources

Firefox was used due to its low usage of resources and faster execution

Selenium was used because of it’s ease of use and good interaction with the webpages using javascript and more convenience with webpages as it simulates the user usage for the web

Google was used for searching to ensure the reliability of the extracted data and links

The ‘By’ class was used from the selenium library to select the HTML tags easily

The get\_property function was used extract the tags values not the whole code

Some links got excluded from the process as they don’t contain any useful data

Some string characters from the title were also excluded so the OS can inject them to the text file name

The driver was put in an absolute global location on the machine so it can be found from any other location easily