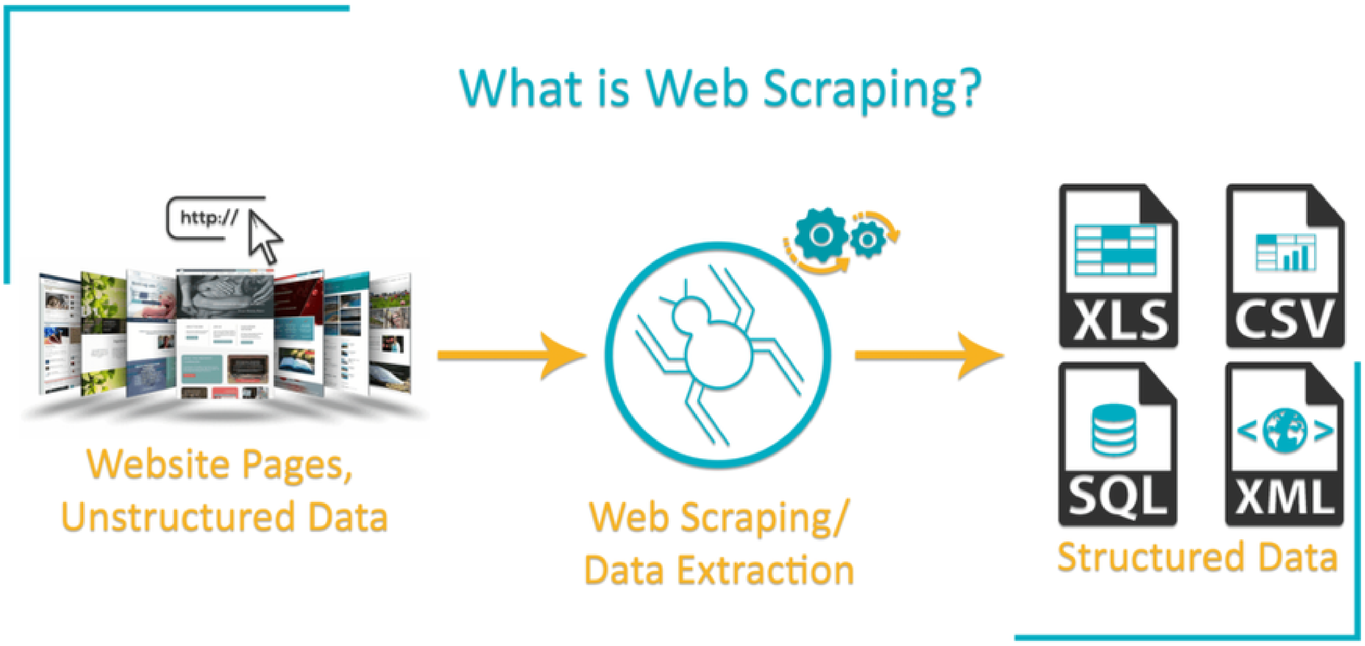
REPORT

WEB SCRAPING WITH PYTHON

Name	:	Priyanshu Tariyal
Semester	:	4th
Section	:	'O'
Class Roll No.	:	40
University Roll No.	:	2118963

Introduction

Web scraping is a technique used to extract data from websites, turning unstructured information into structured data that can be analyzed and utilized for various purposes. Python is a popular programming language for web scraping due to its ease of use, rich libraries, and strong community support. In this report, we will explore the fundamentals of web scraping with Python, including the tools, techniques, best practices, and potential challenges.


What is Web Scraping?
http://
Website Pages, Unstructured Data
Web Scraping/ Data Extraction
XLS
CSV
SQL
XML
Structured Data

Tools for Web Scraping with Python

1. Beautiful Soup

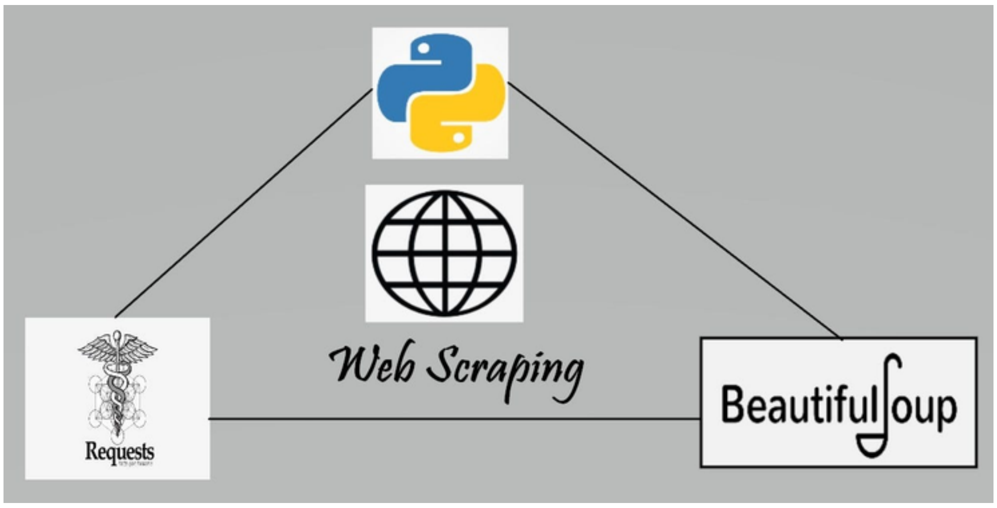
Beautiful Soup is a Python library that enables parsing HTML and XML documents, making it easy to navigate through the page's structure and extract data based on tags, attributes, and text content. It provides a simple interface for handling complex web pages and is widely used in the web scraping community.

2. Requests

The Requests library is an essential tool for making HTTP requests to web pages and retrieving their content. It simplifies the process of sending GET and POST requests, handling cookies, and managing headers.

3. Selenium

While Beautiful Soup and Requests work well for static websites, some websites heavily rely on JavaScript to load content dynamically. In such cases, Selenium comes to the rescue. Selenium is a web automation library that allows interaction with web pages through a browser, effectively emulating user behavior. It's particularly useful for scraping JavaScript-driven websites.



Web Scraping Techniques

1. Parsing HTML with Beautiful Soup

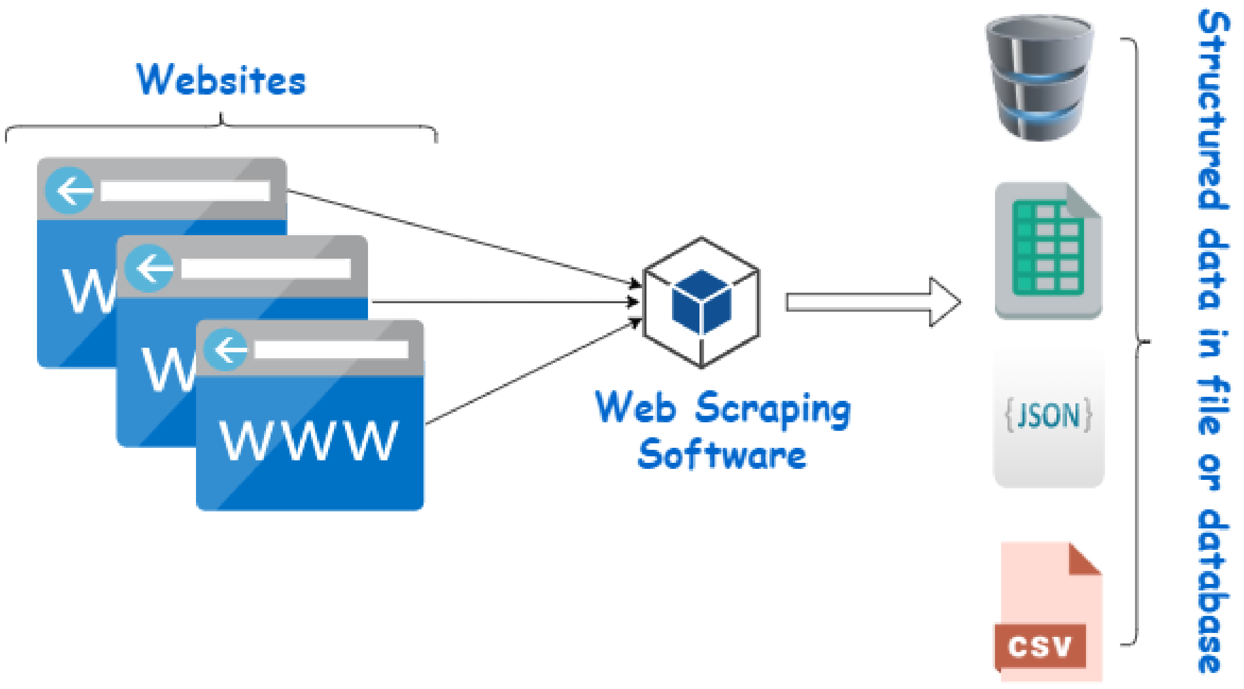
To extract data from a web page using Beautiful Soup, the typical workflow involves:

- Making an HTTP request to the target URL using the Requests library.
- Parsing the HTML content of the response using Beautiful Soup.
- Navigating through the HTML tree to locate and extract the desired data.

2. Handling Dynamic Content with Selenium

Selenium automates web browsers, such as Chrome or Firefox, to render and interact with web pages. The process usually involves:

- Installing the appropriate web driver (e.g., ChromeDriver, GeckoDriver) for the selected browser.
- Initializing the web driver and loading the target URL.
- Waiting for the page to load and render dynamic content (using explicit or implicit waits).
- Interacting with the page elements (e.g., clicking buttons) if necessary.
- Extracting the desired data from the loaded page.



Best Practices for Web Scraping

1. Respect Robots.txt

The "robots.txt" file is a standard used by websites to communicate with web crawlers, specifying which parts of the site are off-limits for scraping. It is essential to review this file and follow its guidelines to avoid legal and ethical issues.

2. Use Delays and Headers

To prevent overloading the target website's server and avoid being detected as a bot, it's best to add delays between successive requests. Additionally, some websites may require specific User-Agent headers to make your requests appear more like those of a regular browser.

3. Be Polite and Ethical

Web scraping should be done responsibly and ethically. Avoid scraping sensitive information, be mindful of the website's terms of service, and limit the frequency of your requests to avoid causing strain on the server.

SCRAPE PUBLIC DATA ONLY
Only extract data that is publicly available or you can get under the radar.
USE A TOOL
Prefer Using a tool for hassle free data extraction.
REVIEW YOUR CODE
Periodically check your code and make changes if needed.
TERMS & CONDITIONS
Always Check the terms and conditions of website you are extracting data from.
DON'T OVERLOAD SERVER
Make Sure Your Code isn't loading the target website's server.

Challenges and Limitations

1. Website Structure Changes

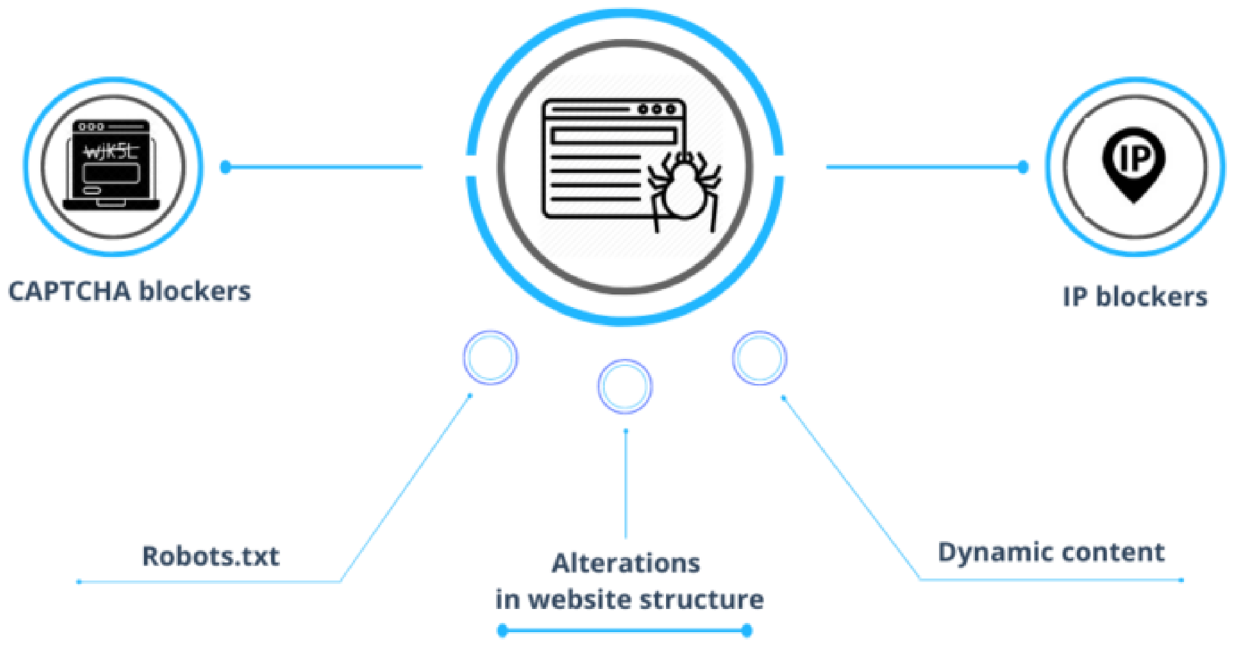
Web scraping heavily relies on the target website's structure. If the website undergoes significant changes, such as restructuring HTML elements or introducing new classes, your scraping code may break.

2. Anti-Scraping Measures

Some websites implement measures to detect and block web scraping activities. These may include CAPTCHAs, IP blocking, or even legal actions against scrapers.

3. Data Quality and Cleaning

Extracted data may not always be in a clean and usable format. Additional efforts may be required for data cleaning and validation.



Conclusion

Python provides a powerful set of tools and libraries for web scraping, making it accessible to both beginners and experienced developers. However, web scraping should be performed responsibly, adhering to legal and ethical guidelines, and respecting the target website's terms of service. With proper techniques and best practices, web scraping can be a valuable method for gathering data for research, analysis, and various other applications.