Web Scraping With Python

Learn to perform web scraping with Python in order to gather data from multiple websites quickly, saving you both time, and effort.

Web scraping with **Python** allows you to **efficiently collect relevant data points**, providing you with the **tools** you need to **get the job done.**

6 reasons to use Python for Web Scraping

Python is one of the better-known coding languages, which makes it advantageous to many developers. It has many specific features that make it the preferred choice for **data collection** and **web scraping** including:

#1: Simplicity – Python is a clear, straight forward coding language that does not include excessive non-alphabetical characters, unlike some other coding languages. The simplicity makes it easier for developers to learn and understand than other languages.

#2: Large libraries – Python has a large number of libraries at its disposal (NumPy, Matlpotlib, Pandas, etc) which provides developers the ability to easily scrape and manipulate a wide variety of data sets.

#3: Timely typing – Python does not require developers to define or categorize the data types for variables. Instead, variables can be used directly whenever necessary, decreasing the possibility of confusion and saving time.

#4: Syntax is easily understood – Unlike other coding languages, Python syntax is very similar to reading English and therefore easy to understand. The indentations used in Python syntax can help developers discern different scopes and blocks in the code.

#5: Quick – Python allows developers to write simple code for complicated tasks. Developers do not want to spend an excessive amount of time writing code when the point of data scraping is to minimize unnecessary effort. Python allows you to do so.

#6: Familiarity – Python is one of the more commonly known coding languages. This creates a community of developers who can provide answers in case of any questions or road bumps that may come up throughout the process of writing the code.

How does Web Scraping with Python work

Once the code is written and run, a request for scraping is sent to your website of choice. If the request is approved, the server will send the desired data, allowing you to read the HTML or XML page. The code then automatically analyses the HTML or XML page, finds and parses the desired data.

The **5 basic steps** of web scraping with Python:

Step 1: Choose the URL from which you would like to scrape

Step 2: Read the page and find the data you would like to collect

Step 3: Write the code

Step 4: Run the code to extract the data

Step 5: Store the data in the necessary format

It is important to bear in mind that while certain sites allow web scraping freely, others may block you from doing so. In order to find out if a website blocks web scraping, you can check the website's "robot.txt" file. You can find this file by adding "/robots.txt" to the URL of the website you wish to scrape. For example, if you would like to scrape data from kayak.com, you would type www.kayak.com/robot.txt into the address bar.

Using Python Libraries for Web Scraping

Python can be applied to a variety of different uses, each of which coincides with a different Python library. For web scraping purposes, you will use the following libraries:

Selenium: This is a web testing library used to automate browser activity.

Beautiful Soup: This is a library used for parsing HTML and XML documents. This library creates "parse trees," allowing for easy data extraction.

Pandas: This is a library used for data manipulation and analysis. This library extracts and stores the data in your preferred format.

Inspecting the Site

Once you have chosen the website from which you would like to extract your desired data sets, your first step is locating the links to the files that you would like to download. There are many layers of "tags" or code on any given site and not all of this information is relevant to you. Inspecting the page allows us to figure out where the data you want to scrape is located.

To inspect the page, right-click on-site and then click ‘Inspect’ on the drop-down menu. Once you have clicked ‘Inspect’, you will see a box with raw code open.

Notice in the top left-hand corner of the code box, there is an ‘arrow’ symbol. By first clicking this arrow and then clicking on a specific component of the site, the code for what you have selected will be highlighted in the box. Once you have identified the relevant links, you can begin writing code in Python.

3 Steps to Writing Code in Python

Step 1: To start, you need to import the selenium library:

- From Selenium import webdriver

Step 2: Set the credentials and settings to run Selenium:

- Set the proxy credentials. In this case, we used **Bright Data’s Proxy Manager**
- The path to the driver which will run Chrome
- Set the options of Selenium to use the proxy
- Set the target URL you want to scrape

Note: You can send headers with the request to emulate more “human” behavior and avoid bot detection.

Step 3: Run your code. Selenium will open the target URL, store the page source to a variable, and then write it into a file called **“output1.html”**. After it’s done, the driver will close.

After extracting the data, you might want to store it in a specific format. This format varies depending on the purposes of your scraping activities. After changing the format, run the code again in its entirety. You can iterate through the data you scraped and extract the exact information you need.

Syllabus

- Building a web scraper: Python prepwork
- Getting to the libraries

- WebDrivers and browsers
- Finding a cozy place for our Python web scraper
- Importing and using libraries
- Picking a URL

- Scrape and Parse Text From Websites
  - Your First Web Scraper
  - Extract Text From HTML With String Methods
  - A Primer on Regular Expressions
  - Extract Text From HTML With Regular Expressions
  - Check Your Understanding

- Use an HTML Parser for Web Scraping in Python
  - Install Beautiful Soup
  - Requests

    > In order to get the HTML of the website, we need to make a *request* to get the content of the webpage. To learn more about requests in a general sense, you can check out this article . Python ha…

  - The BeautifulSoup Object

    > When we printed out all of that HTML from our request, it seemed pretty long and messy. How could we pull out the relevant information from that long string? BeautifulSoup is a Python library that…

  - Object Types

BeautifulSoup breaks the HTML page into several types of objects. #### Tags A Tag corresponds to an HTML Tag in the original document. These lines of code: soup = BeautifulSoup(‘ An example di…

- Navigating by Tags

  To navigate through a tree, we can call the tag names themselves. Imagine we have an HTML page that looks like this: World’s Best Chocolate Chip Cookies Ingredients 1 cup flour …

- Website Structure

  When we’re telling our Python script what HTML tags to grab, we need to know the structure of the website and what we’re looking for. Many browsers, including Chrome, Firefox, and Safari, have Dev…

- Find

  Beautiful Soup offers two methods for traversing the HTML tags on a webpage, .find() and .find_all(). Both methods can take just a tag name as a parameter but will return slightly different informa…

- Select for CSS Selectors

  Another way to capture your desired elements with the soup object is to use CSS selectors. The .select() method will take in all of the CSS selectors you normally use in a .css file! Search Resul…

- Reading Text

  When we use BeautifulSoup to select HTML elements, we often want to grab the text inside of the element, so that we can analyze it. We can use .get_text() to retrieve the text inside of whatever ta…

- Review

    Amazing! Now you know the basics of how to use BeautifulSoup to turn websites into data. If you take our Data Visualization or Data Manipulation courses, you can see how you might analyze this …

- selenium Library

- Interact With HTML Forms
    - Install MechanicalSoup
    - Create a Browser Object
    - Submit a Form With MechanicalSoup
    - Check Your Understanding

- Scrapy framework python

- Interact With Websites in Real Time

- Make an Instagram Bot