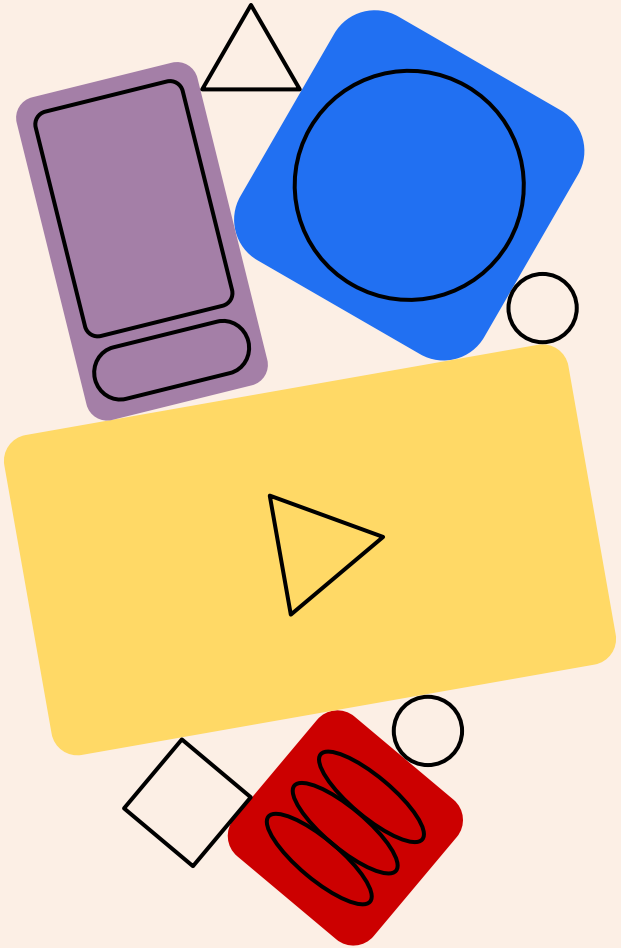
A quick crash course into the fundamentals of web scraping with python

DAA-005 Web Scraping



What is Web Scraping?

Definition

- Automated methods of extracting data from websites.
- Involves sending requests, parsing HTML, and extracting specific information

What is Web Scraping?

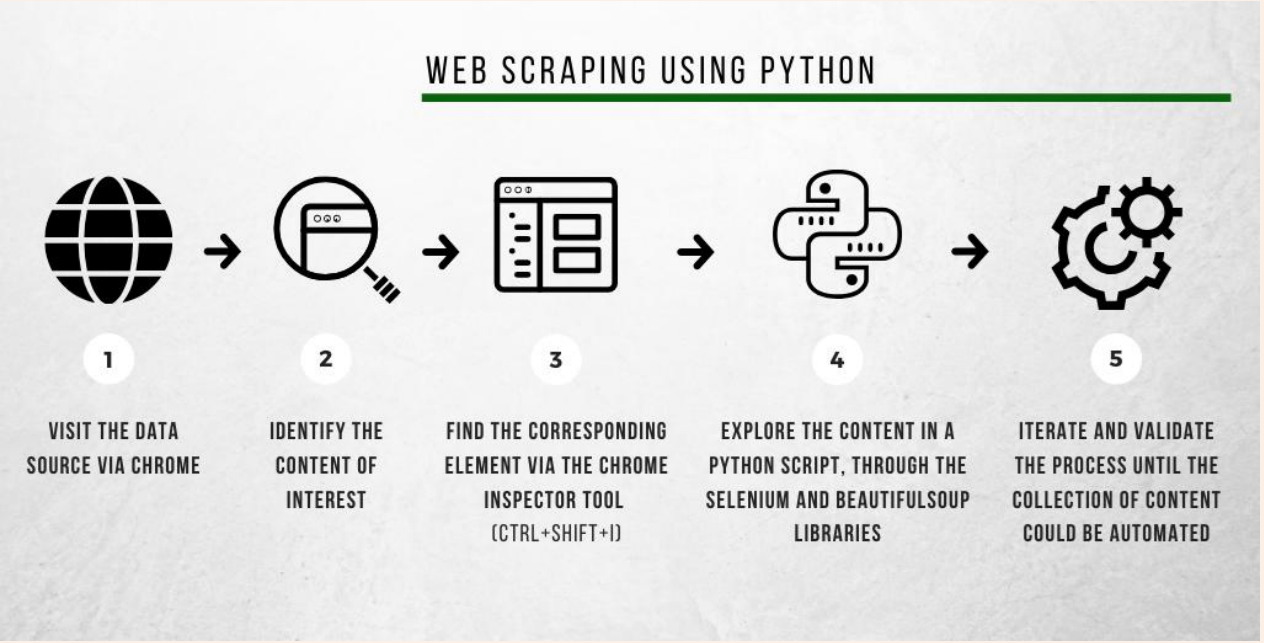
How it Works in General:

1. Send a request to a webpage
2. Parse its structure (HTML/CSS)
3. Extract the needed information

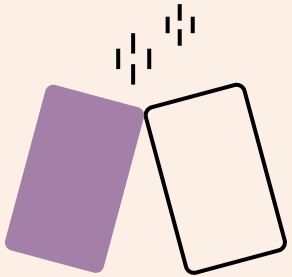
Key Idea:

- Converts unstructured web data into structured formats (CSV, JSON, Excel)

What is Web Scraping?

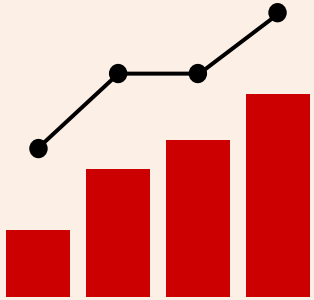


Why Web Scraping?



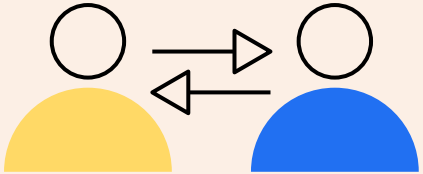
Fills Data Gaps

When APIs/structured datasets are unavailable for processing



Cost-Effective

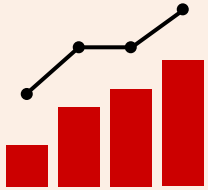
Extracts valuable insights without proprietary tools



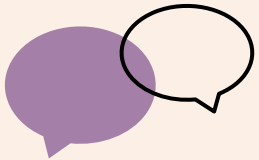
Flexible

Customizable for any public website

Practical use cases of Web Scraping in the Workforce



Monitoring E-Commerce Prices



Analyzing Public Sentiment from Social Media

Gather open data for machine learning

Where does Web Scraping belong in Data Science?

Where does Web Scraping belong?

Advanced Techniques	Deep Learning, Large Language Models (LLMs)
Machine Learning	Traditional ML, Model Evaluation
Statistical Analysis	Hypothesis Testing, Feature Engineering
Exploratory Data Analysis	Visualisation, Descriptive Satistics
Data Cleaning, Preprocessing	Data Wrangling, Normalisation & Scaling
Data Storage, Management	Data Lakes, Data Warehouses, ETL/ELT Pipelines
Data Collection	Web Scraping, APIs, Sensors/Logs

Use case with External Projects: FDM Singapore

1. Landing Page and Online Sentiment Analysis
    a. Campaign landing page
    b. Reddit (API)
2. HR firm Competitor Analysis
    a. Batch scraping from competing websites (Glassdoor, GradConnect, Indeed.com)

Introduction to BeautifulSoup

Introduction to BeautifulSoup

What is BeautifulSoup?

- A Python library for parsing HTML and XML
- Works well for static websites

Features

- Easy-to-use methods like .find() and .find_all()
- Handles messy HTML structures

Introduction to BeautifulSoup

Workflow

1. Fetch the webpage using requests
2. Parse HTML using BeautifulSoup
3. Locate and extract desired elements (e.g. titles, links, tables)

Let’s code!

Advanced Web Scraping Techniques

Scrapy

- Best for large-scale, automated crawling
- Efficient pipelines for handling and storing data

Selenium

- For interacting with dynamic, JavaScript-heavy sites
- Simulates user actions like clicking or logging in

Why use other methods over BeautifulSoup?

Decision-Making Factors

- Is the website static or dynamic?
- How large is the data?
- Is scalability important

Scenarios

- Scrape a single static page > BeautifulSoup
- Crawl an entire e-commerce site > Scrapy
- Scrape dynamic pages with interactions > Selenium

Web Scraping Ethics

Do’s

- Follow robots.txt
- Limit request rates to avoid overloading servers
- Use scraped data responsibly

Don’ts

- Scrape sensitive or private data
- Bypass security measures (CAPTCHA)

Web Scraping Ethics

eBay vs. Bidder's Edge

eBay sued Bidder's Edge in 2000 for scraping its auction data.

Facebook vs. Power Ventures

In 2009, Facebook won a lawsuit against Power Ventures for scraping user data and violating intellectual property rights.

Craigslist vs. 3taps

In 2013, Craigslist sued 3taps for web scraping and won $1 million in damages.

Meta vs. Bright Data

Meta sued Bright Data for scraping data from Facebook and Instagram, but the US District Court ruled in favor of Bright Data.

However, web scraping can be illegal in some situations, including:

- Scraping data from government computer systems or financial institutions
- Logging into websites or web pages to download data
- Breaching laws that apply to publicly available data
- Using the data in a way that's illegal

Some companies use web scraping in legitimate ways to gain data-driven insights. However, companies may not want to be scraped, and they can try to show that it damages their infrastructure or operations.