Web Scraping

What is web scraping?

Web scraping is the process of **automatically extracting data from websites**.
Instead of copying data by hand, you write a program (usually in Python) that:

1. Requests a web page
2. Reads its HTML
3. Pulls out the data you want (text, prices, titles, links, etc.)

Typical use cases

- 📊 Collecting product prices
- 📰 Scraping news headlines
- 📚 Gathering research data
- 🔍 Monitoring job postings
- 🤖 Feeding data into ML / analytics pipelines

How web scraping works (step by step)

1. **Send HTTP request** → get the web page
2. **Parse HTML** → understand page structure
3. **Extract data** → grab specific elements
4. **Store results** → CSV, database, JSON, etc.

Core Python tools for web scraping

1. `requests` – get the web page

Used to send HTTP requests.

```
import requests

response = requests.get("https://example.com")
html = response.text
```

2. `BeautifulSoup` – parse & extract HTML

BeautifulSoup is a Python library used for web scraping purposes to pull the data out of HTML and XML files. It creates a parse tree from page source code that can be used to extract data in a hierarchical and more readable manner.
Used to read and navigate HTML.

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(html, "html.parser")
title = soup.find("h1").text
print(title)
```

3. `lxml` (optional but faster)

Alternative HTML parser, very fast for large pages.

4. `Selenium` – for dynamic websites

Used when data is loaded with **JavaScript** (React, Vue, etc.).

- Simulates a real browser
- Slower but powerful

Simple example: scraping article titles

```
import requests
from bs4 import BeautifulSoup

url = "https://example.com"
response = requests.get(url)

soup = BeautifulSoup(response.text, "html.parser")

titles = soup.find_all("h2")

for t in titles:
    print(t.text)
```

Static vs Dynamic websites

TYPE	TOOL
Static HTML	`requests + BeautifulSoup` ✅
JavaScript-rendered	`Selenium`, `Playwright` ⚠️

Is web scraping legal?

It depends ⚠️
Always:

- Check **robots.txt**
- Read **Terms of Service**
- Avoid scraping private / sensitive data

- Use delays (don’t spam requests)

Ethical scraping = good scraping 👍

Common beginner mistakes

- ❌ Scraping without checking page structure
- ❌ Forgetting headers (User-Agent
- ❌ Scraping JS sites with `requests`
- ❌ Sending too many requests too fast

What I can help you with next

If you want, I can:

- Build a **real scraping project**
- Teach **Selenium scraping**
- Scrape **jobs, products, or news**
- Explain **anti-scraping & how sites block bots**
- Show how scraping connects to **REST APIs**