**📌 Introduction**

In web development and data science, **web crawling** and **web scraping** are two related yet distinct techniques used to gather data from the internet. While they often go hand-in-hand, their purposes and implementations differ.

This document explains both concepts in detail, particularly how they can be implemented using **Python**, and provides working examples to demonstrate the difference.

**🕸️ 1. Web Crawling in Python**

**✅ Definition**

**Web Crawling** is the process of **automatically navigating** through a website by following hyperlinks to discover pages. It is used primarily to **collect URLs** of web pages for further processing or indexing (like search engines do).

**⚙️ Use Cases**

* Search engine indexing (e.g., Googlebot)
* SEO analysis tools
* Detecting broken links on websites
* Content discovery

**🧰 Python Libraries for Crawling**

* requests – to fetch pages
* BeautifulSoup – to parse HTML and extract links
* Scrapy – powerful web crawling framework

**🧹 2. Web Scraping in Python**

**✅ Definition**

**Web Scraping** is the process of **extracting specific data** from a web page, such as product names, prices, or reviews. It focuses on **parsing HTML content** and pulling out relevant information.

**⚙️ Use Cases**

* Price comparison tools
* Job aggregator websites
* Research and data collection
* News aggregation

**🧰 Python Libraries for Scraping**

* requests – to download the web page
* BeautifulSoup – to parse HTML and extract data
* Selenium – for dynamic JavaScript-rendered sites
* lxml – fast XML/HTML parsing
* Playwright – headless browser automation

**🧠 Crawling vs Scraping: Summary Table**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Crawling** | **Scraping** |
| Purpose | Discover pages and links | Extract structured data |
| Output | List of URLs or site structure | Structured data (CSV, JSON, etc.) |
| Typical Use | Search engines, link indexers | Data collectors, aggregators |
| HTML Parsing | Only to extract links | Required to extract data |
| Python Libraries | requests, BeautifulSoup, Scrapy | requests, BeautifulSoup, Selenium |
| Example Output | ["/about", "/contact"] | {"title": "Product", "price": "$10"} |