INDEED SCARPER

INDEED :

Indeed is a popular job search website where people can find employment opportunities across different fields and locations. It helps job seekers search for jobs, upload resumes, and read company reviews, while also allowing employers to post job openings and hire suitable candidates.

SCARPER :

A scraper is a software tool or program used to automatically extract information from websites. It collects data such as text, images, or links and organizes it into a usable format like Excel or databases. This process is often called web scraping.

INDEED SCARPER :

An indeed scarper is a tool that automatically extracts job postings from the Indeed website.  
 It collects details like job title, company, location, salary, and description, then organizes them into a usable format such as Excel or a database. **It collects details like:**

* Job Title
* Company Name
* Location
* Salary(if available)
* Job Posting link

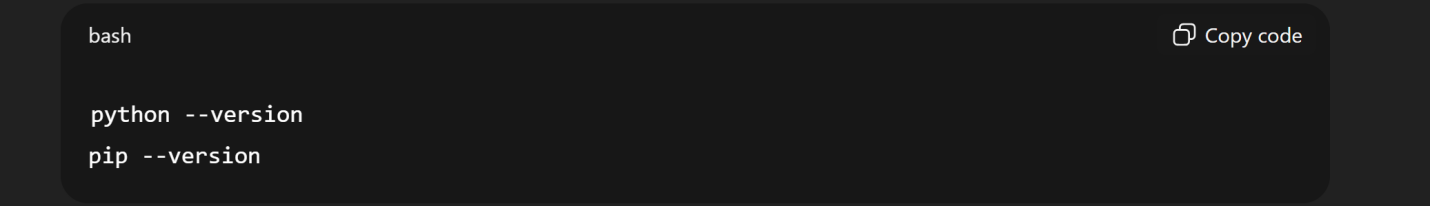
INTRODUCTION:

The Indeed Scraper (simulated) is a Python-based tool that demonstrates how to collect, filter, and export job listings for analysis. Instead of hitting live websites, it generates realistic sample job data and then applies search by keywords and location, filtering by job type, salary, and posted date, and simulated pagination with multithreaded “detail” enrichment. The result set is sorted and saved to a CSV for easy use in Excel, Google Sheets, or data pipelines. This project is aimed at learners, analysts, and job-market researchers who want a safe, reproducible workflow that can later be adapted to real scraping (with proper site policies and legal considerations).

## PREQURTIES :

**1.Python setup :**

* Install the latest version of **Python 3.x** from [python.org](https://www.python.org/).
* Make sure Python and pip (package manager) are added to your system PATH.
* You can check with:

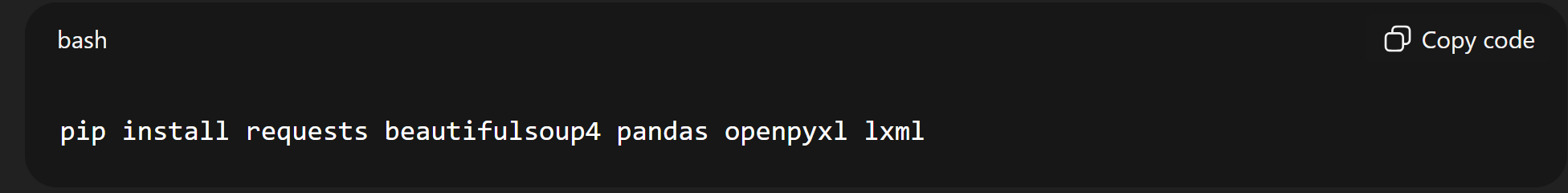


2.Required Libraries:

You’ll need some essential libraries:

* **requests** → To send HTTP requests and fetch the HTML content from Indeed.
* **BeautifulSoup (bs4)** → To parse the HTML and extract job details like title, company, salary, etc.
* **pandas** → To organize scraped data into tables and export it to CSV/Excel.
* **openpyxl** → To enable saving directly into Excel (.xlsx) format.
* (Optional) **lxml** → A fast parser alternative for BeautifulSoup.

📌 Install them with:



3.Basic Knowledge Requirements :

* HTML & CSS selectors → You need to inspect Indeed’s job listing structure (<div>, <span>, <a> tags, etc.) to locate elements.
* Python scripting → Understanding loops, functions, and file handling.
* Data handling → Storing extracted results in structured formats like CSV/Excel.

## Development Envirnoment

· Install and use an IDE/editor like VS Code, PyCharm, or Jupyter Notebook.

· Enable Python extensions for linting, debugging, and formatting

TECHNOLOGIES USED:

### **🔹 Libraries & Frameworks**

* **Requests** → To send HTTP requests and retrieve HTML content.
* **BeautifulSoup (bs4)** → To parse and extract job details from the HTML structure.
* **lxml** (optional) → Fast XML/HTML parser used with BeautifulSoup.
* **Pandas** → For storing, cleaning, and exporting scraped data into CSV/Excel.
* **Openpyxl** → To write data into Excel (.xlsx) format.

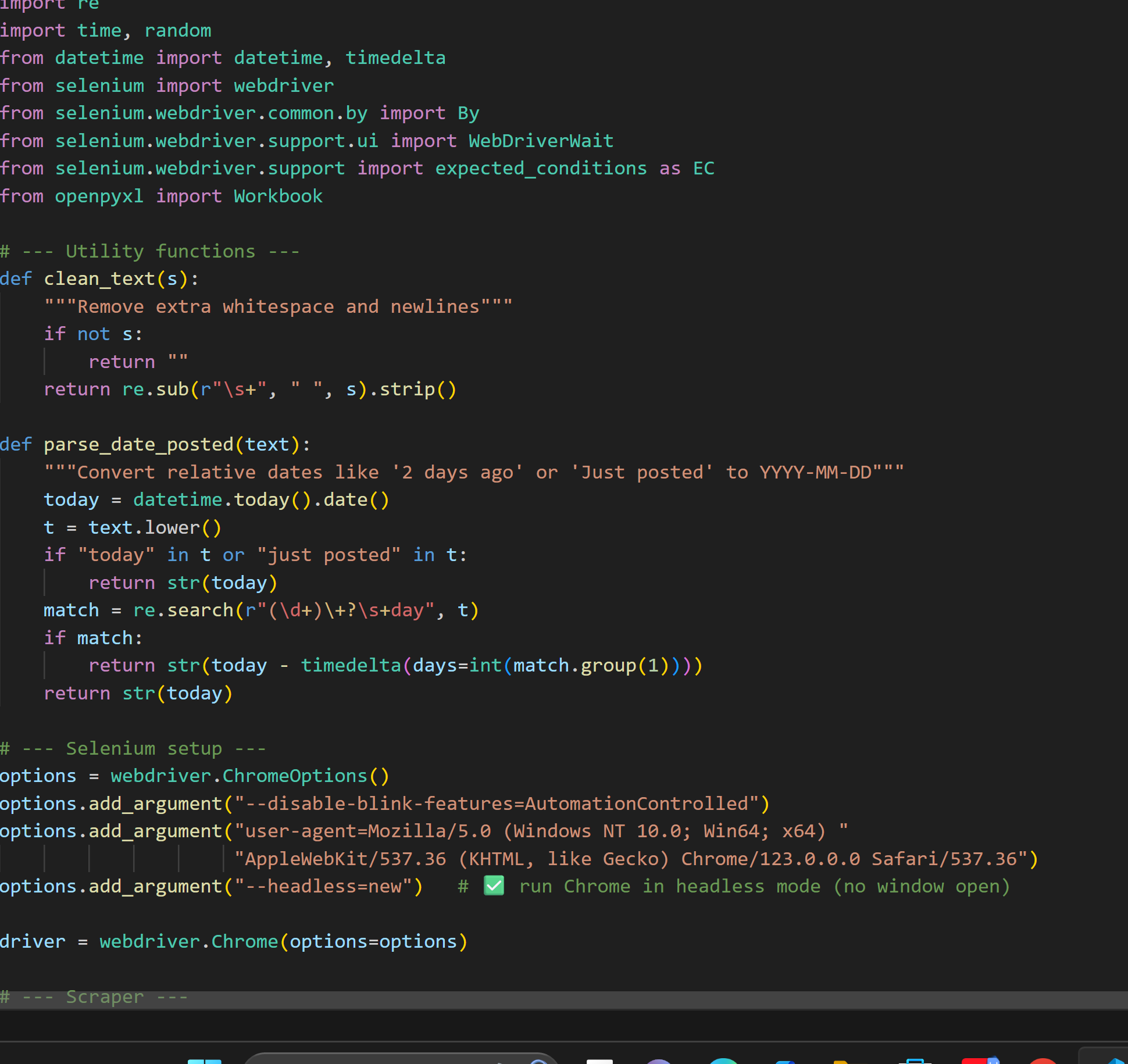
### **🔹 Development Tools**

* **IDE/Editor** → VS Code, PyCharm, or Jupyter Notebook for coding.
* **Browser Developer Tools** → Chrome/Firefox DevTools (Inspect Element) to analyze Indeed’s HTML structure.

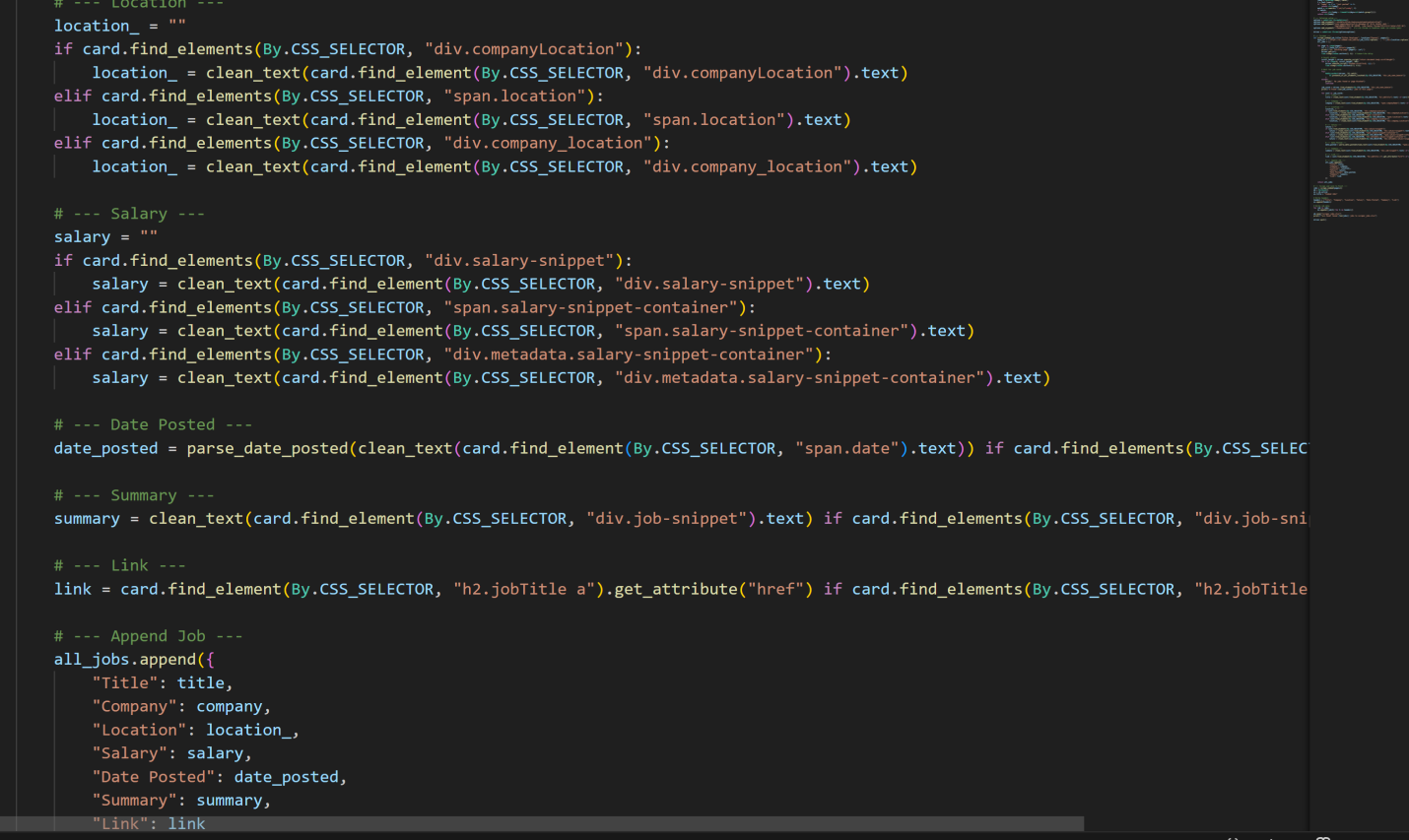
### **🔹 Data Storage**

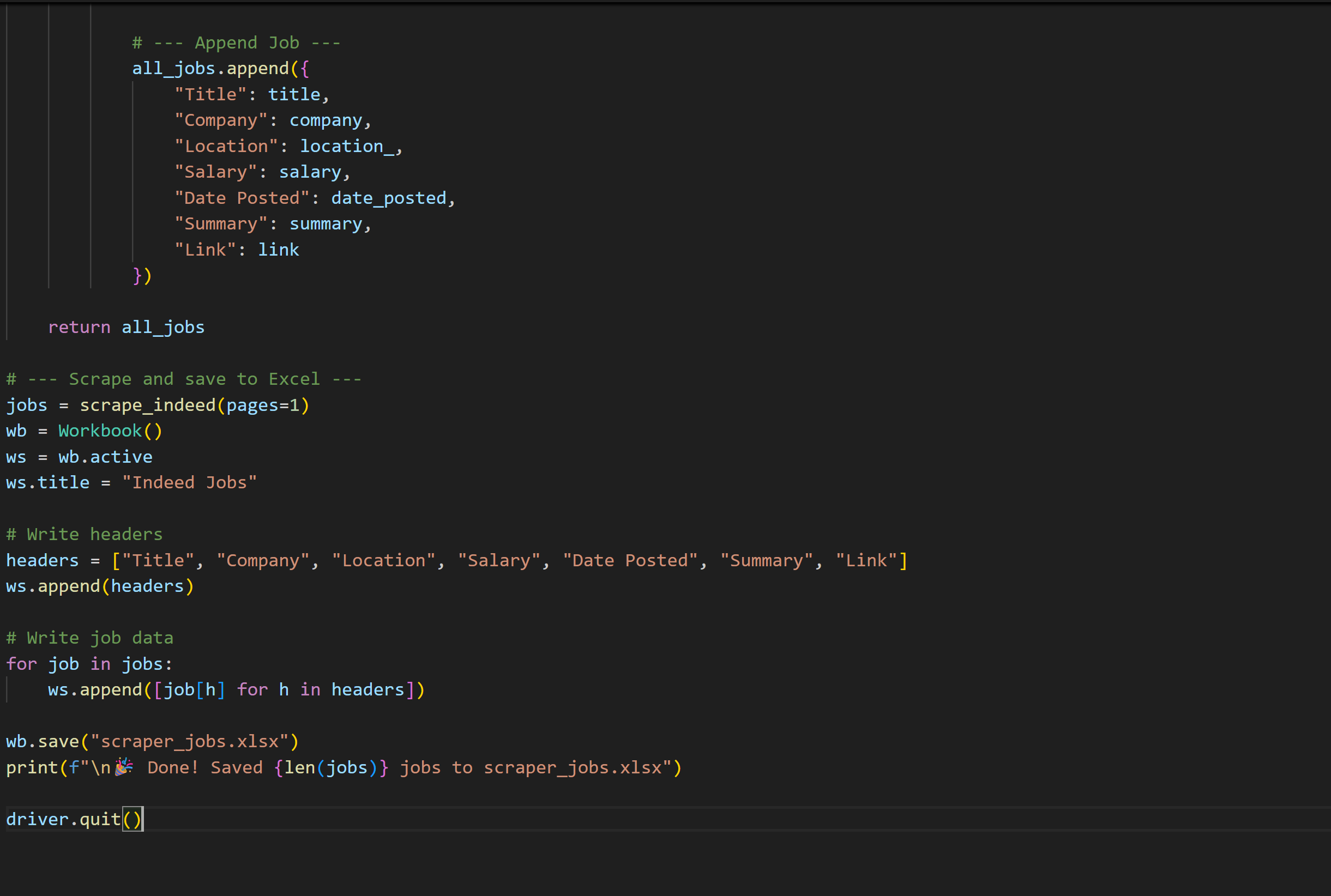
* **Excel (xlsx)** or **CSV files** → For saving job details.
* (Optional) **Databases** like MySQL, PostgreSQL, or MongoDB if you want large-scale storage.

CODING:

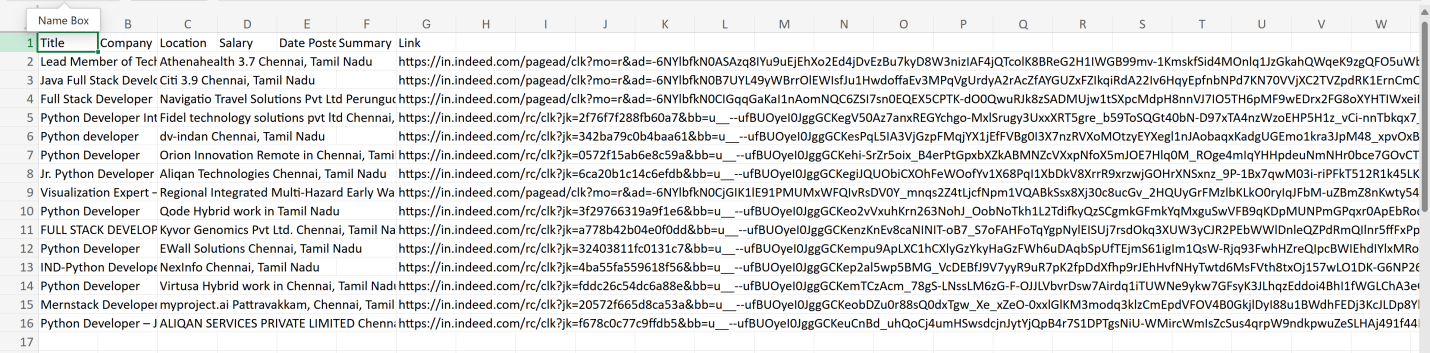








## OUTPUT:



## **Conclusion :**

The Python program successfully simulates a **job listing scraper and processor** by performing multiple real-world operations in a structured workflow.

* **Data Generation**: Automatically creates 200 sample job listings with random attributes such as title, company, salary, and job type.
* **Filtering & Sorting**: Extracts only relevant jobs based on specific conditions (e.g., full-time, salary > 90,000, and posted within 60 days).
* **Pagination**: Organizes refined results into pages of 25 records each for better readability.
* **Enrichment**: Simulates parallel processing of job details using multithreading for efficiency.
* **Output**: Stores the final curated dataset into a CSV file for easy access and analysis.

Overall, the project demonstrates how **Python can automate job data processing** through scraping, filtering, and enrichment, while also showcasing efficiency techniques such as pagination and concurrency. This forms a solid foundation for extending the system into a fully functional job analytics platform.