**Web Scraping Netmeds Data**

Project Overview

This project focuses on web scraping medicine-related data from Netmeds, an online pharmacy. The goal was to extract essential details such as tablet names, prescription requirements (Rx), discounts, tablet information, and other critical metadata. Initially, the script took approximately 15 hours to execute, but through code optimization and multithreading, execution time was significantly reduced.

Technologies Used

* **Python** for scripting
* **Requests** for making HTTP requests
* **BeautifulSoup** for HTML parsing
* **ThreadPoolExecutor** for parallel processing
* **JSON** for storing extracted data
* **Logging** for tracking errors and execution progress

Step-by-Step Process

1. Extracting Medicine Category Links

The first step involved extracting all category links that contained lists of medicines:

* Sent a request to the **Netmeds Prescriptions** page.
* Parsed the HTML using **BeautifulSoup**.
* Located medicine category sections and extracted **href** links.
* Stored the category links for further processing.

2. Extracting Medicine Links

Once category links were obtained:

* Iterated over each category link and fetched the webpage.
* Identified and extracted medicine URLs from product listings.
* Stored all valid medicine links.

3. Fetching Medicine Details

For each medicine URL:

* Sent an HTTP GET request to retrieve the medicine page.
* Extracted key information:
  + **Tablet name**
  + **Disease** (if mentioned)
  + **Rx Required** (Prescription Required or Not)
  + **Final Price**
  + **Discount**
  + **Country of Origin**
  + **Manufacturer**
  + **Tablet Info** (Dosage, Composition, etc.)
* Stored the extracted data in a structured format.

4. Optimizing Performance

Initially, the script took over **15 hours** to execute. Optimization strategies included:

1. Threading for Parallel Execution

🔹 **Concept Used:** threading.Lock(), ThreadPoolExecutor  
🔹 **Purpose:** Prevents race conditions when multiple threads access shared data.  
🔹 **Optimization Benefit:**

* Enables concurrent execution, improving performance when processing multiple links.
* The lock ensures that shared data (extracted\_data\_list, failed\_links) remains consistent.

2. Exception Handling for Robustness

🔹 **Concept Used:** try-except-finally  
🔹 **Purpose:** Handles errors gracefully without crashing the program.  
🔹 **Optimization Benefit:**

* Prevents program failures due to network issues (requests.exceptions.RequestException).
* Logs errors for debugging.

3. Logging for Debugging & Monitoring

🔹 **Concept Used:** logging module  
🔹 **Purpose:** Stores warnings, errors, and progress updates in a log file.  
🔹 **Optimization Benefit:**

* Helps monitor scraping progress without printing unnecessary details.
* Easier debugging when errors occur.

4. Web Scraping with BeautifulSoup

🔹 **Concept Used:** BeautifulSoup  
🔹 **Purpose:** Extracts required data from HTML content.  
🔹 **Optimization Benefit:**

* Efficiently locates and retrieves elements using **CSS class names**.
* Uses .get\_text(strip=True) to remove extra spaces.

5. Dictionary for Structured Data Storage

🔹 **Concept Used:** dict  
🔹 **Purpose:** Stores extracted data efficiently.  
🔹 **Optimization Benefit:**

* Organizes data into **key-value pairs** for structured storage.
* Facilitates easy conversion to **JSON format** for saving.

6. Periodic Data Saving (Checkpoints)

🔹 **Concept Used:** with open() (File Handling), json.dump()  
🔹 **Purpose:** Saves extracted data periodically to prevent loss in case of crashes.  
🔹 **Optimization Benefit:**

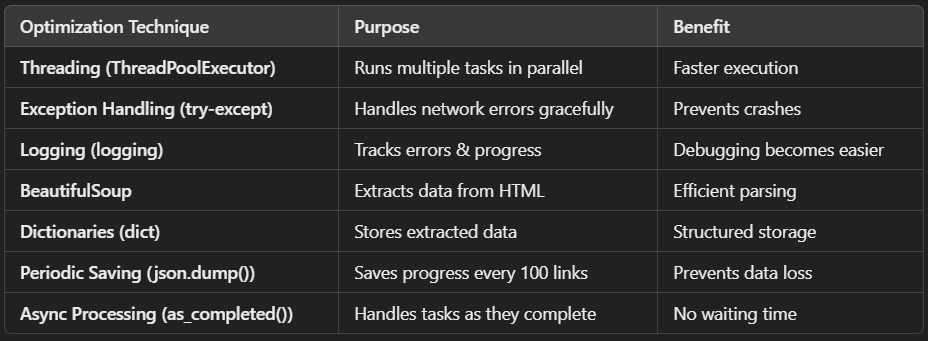
* Saves every **100 links** to ensure progress isn't lost.
* Uses a **lock** to prevent race conditions while writing files.

7. Asynchronous Processing of Future Results

🔹 **Concept Used:** as\_completed() from concurrent.futures  
🔹 **Purpose:** Processes completed tasks asynchronously instead of waiting.  
🔹 **Optimization Benefit:**

* **Reduces idle time** by handling completed tasks immediately.

Code Optimization Summary



**Results & Achievements**

* Successfully scraped **27,000+ medicines**.
* Reduced execution time from **15 hours to a few hours** with optimizations.
* Created a structured and reusable pipeline for scraping similar websites.

This project demonstrated the power of **web scraping, multithreading, and optimization** in large-scale data extraction. The extracted data can be further analyzed for insights into medicine pricing, availability, and discounts. 🚀