# **Polywood Crawler Documentation**

## **1. Overview**

* The **Polywood Crawler** is a Python-based web scraping tool developed to extract comprehensive product data from the Polywood website ([https://www.polywood.com](https://www.polywood.com/)). It programmatically navigates through each category and collection, scrapes paginated product links, and collects detailed information on every product page, including title, SKU, descriptions, dimensions, features, and images. The output is structured into a JSON file, useful for analytics, inventory management, and catalog generation\*
* **Website**:<https://www.polywood.com>
* **Data Extracted:**
* Product URLs
* Titles
* SKUs
* Main Category
* Collection
* Descriptions
* Features (text block)
* Weight & Dimensions (table as list of dicts)
* Images (all available image URLs)

## **2. Tools & Requirements**

* **Python Version:** 3.8 or above
* **Core Packages Used:**
* playwright==1.42.0 – Browser automation
* beautifulsoup4==4.12.3 – HTML parsing
* lxml==5.2.2 – Fast HTML/XML processing
* asyncio – Asynchronous flow control
* logging – For tracking execution logs
* **Browser:** Chromium (installed via Playwright)
* **OS Compatibility:** Windows, macOS, Linux

## **3. Installation Guide**

* pip install -r requirements.txt
* playwright install
* playwright --version # To verify installation

## **4. Execution Steps**

* **Run the crawler:** python polywood.py
* **Process Flow:**
  + **Category Navigation:** Scrapes all product categories and sub-collections.
  + **Product Link Extraction:**  From each collection URL, iterates paginated results and gathers product URLs.
  + **Product Page Scraping:** Each product page is opened and scraped for:
* Title, SKU, Description, Category/Collection
* Features
* Images (all product views)
* Weight & Dimensions table
* **Data Storage:**
* Data saved as structured JSON: output/output.json
* Deduplication handled via link checking
* **Execution Notes:**
* Logs progress in scraper.log
* Built-in retries on failed pages
* Headless mode is disabled by default for debugging

## **5. Authentication / Access**

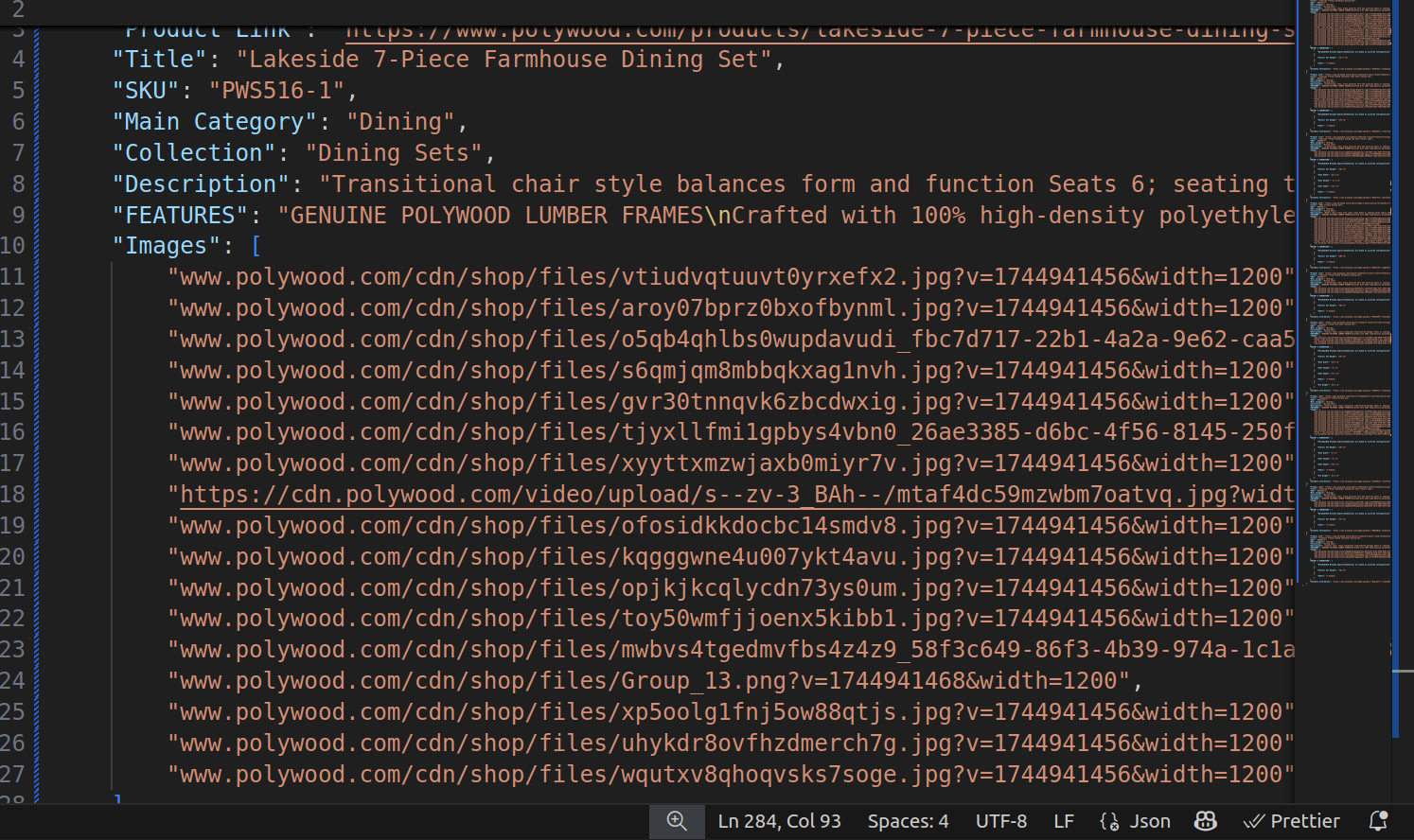
* **Login Required:** No
* **VPN Required:** No
* **Rate Limiting:** Can be added using asyncio.sleep()
* **Headless Mode:** Disabled by default (headless=False)

## **6. Dynamic Site Behavior Handling**

* Waits for full content loading via page.wait\_for\_timeout(5000)
* Handles missing content gracefully via try-except blocks
* Navigates all paginated results per category

## **7. Output Format**

* JSON saved as output/output.json with structure like:



## **8. Troubleshooting Tips**

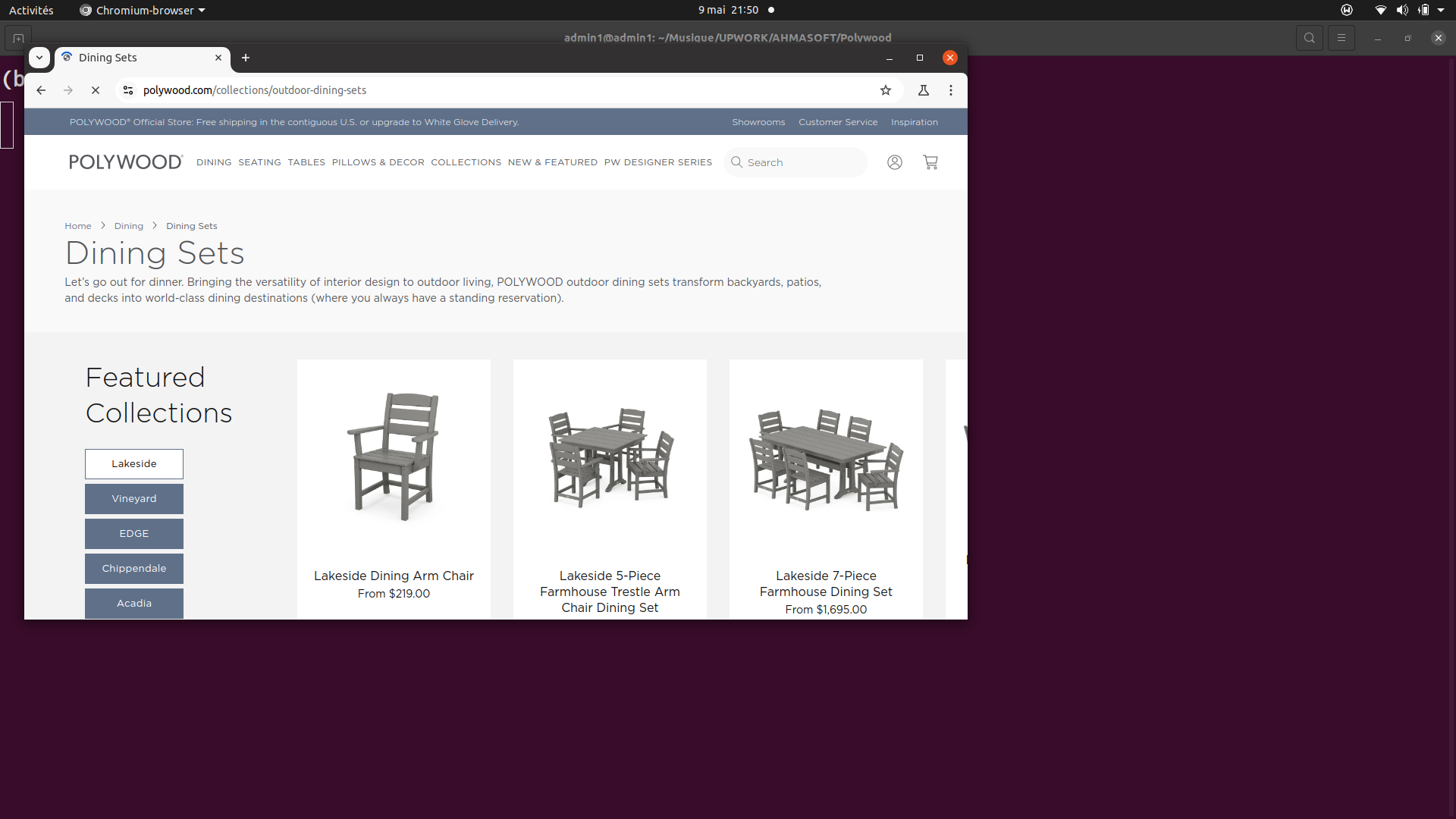
* **Playwright Errors**: Ensure Chromium is installed by running:  
  playwright install
* **Timeouts**: Increase wait times in page.wait\_for\_timeout() or check your internet connection.
* **Incomplete JSON Output**: Delete the output.json file and rerun the script. This often happens due to crashes or manual interruptions.
* **Missing Data or Fields**: Inspect the page manually. The site's structure might have changed. Update your BeautifulSoup selectors accordingly.

### **Debugging Options:**

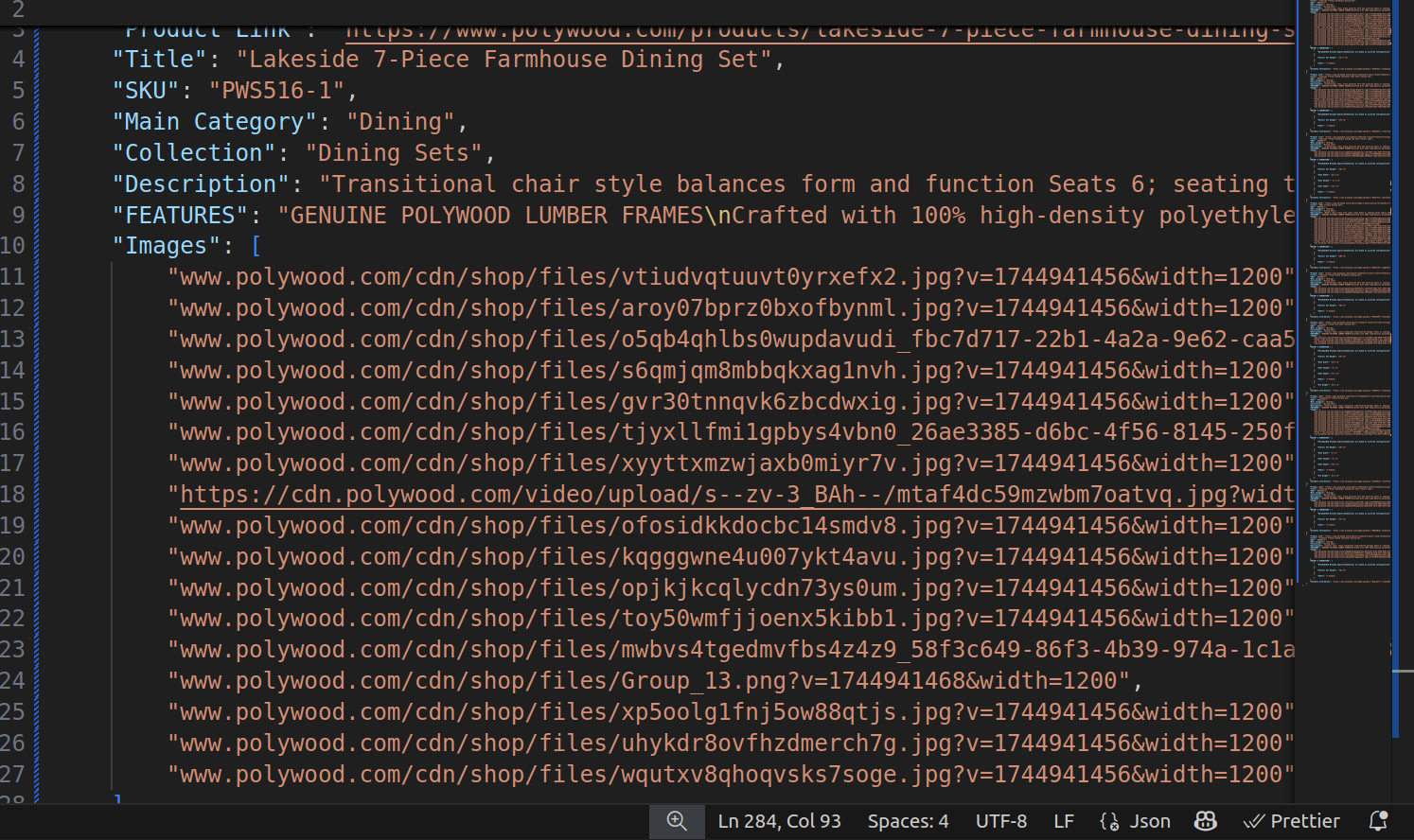
* **Enable Browser View**: Set headless=False in async\_playwright().start() to visually debug.
* **Manual URL Testing**: Paste problematic product URLs into your browser and inspect element structure.
* **Use Logs**: Check scraper.log for errors and scraping progress.
* **Run in Batches**: Modify your script to process only a few pages or products for quick testing.

## **9. Visuals & Screenshots (Optional)**

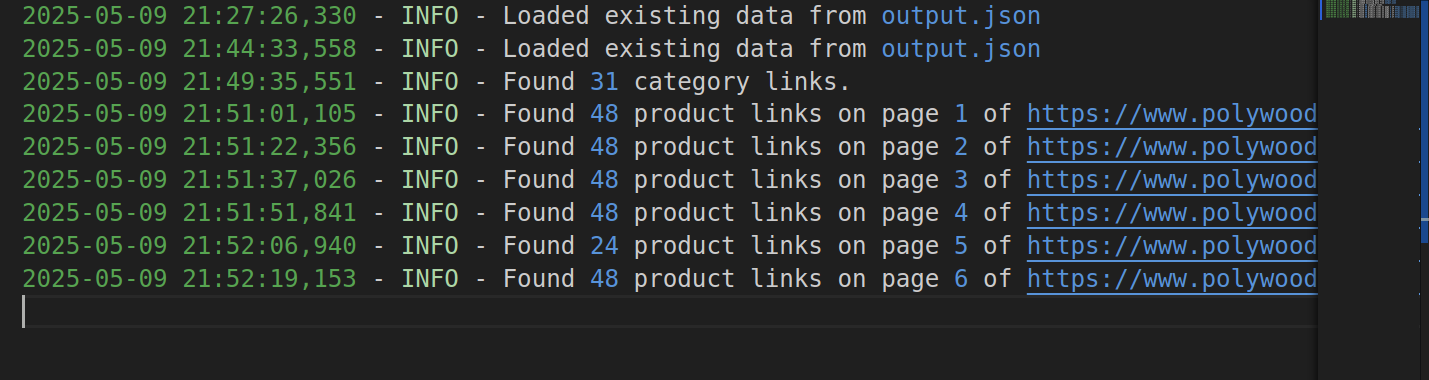
* Screenshot of browser during scrape:



* Sample JSON view in editor



* Scraper.log file example



## **10. Final Notes**

* **Author:** Ahmad S.
* **Last Updated:** May 9, 2025
* **Version:** 2.0
* **For Support:**
  + When reporting issues, please include:
* Console logs (scraper.log)
* Screenshot of the error
* Incomplete or corrupted JSON file
* The URL where the error occurred