Amazon Scraper

A Python-Based Product Data Extraction and Analysis Tool

By Manisha P
Team 2

Cybernaut Intern

ABSTRACT :

The growing trend of online shopping has created the need for efficient tools to track product data in real-time. The Amazon Scraper is a Python-based solution designed to extract product details such as **name, price, rating, reviews, availability, and image URLs** from Amazon's website. The system allows for **price comparison, market research, competitive monitoring, and consumer trend analysis**. It uses **BeautifulSoup, Selenium, and Requests** to handle dynamic content and supports saving results in **CSV, Excel, or JSON formats**. Multi-threading and proxy support make the scraper scalable and resilient against blocking.

Project Description :

The **Amazon Scraper** is a Python-based data extraction tool that automatically collects product information from Amazon's e-commerce platform. It is designed for **price tracking, market research, and competitive analysis**.

The scraper extracts details such as:

- Product Name
- Price
- Rating & Reviews
- Availability
- Product Image URLs

It also supports **real-time tracking** and saving results in structured formats like **CSV, JSON, or Excel**. The tool handles **pagination**, enabling collection of data from multiple search result pages, and uses **proxies & multi-threading** to overcome Amazon's anti-scraping measures.

This project is useful for **buyers, researchers, and businesses** who want insights into product pricing trends, consumer behavior, and competitor strategies.

INTRODUCTION :

Amazon is one of the world's largest e-commerce platforms with millions of products across diverse categories. Manually monitoring product details is inefficient, error-prone, and impractical given frequent price changes and vast product listings.

Automated scraping solutions provide several benefits:

- **Real-time monitoring** of product prices and availability
- **Historical trend analysis** of pricing and reviews
- **Competitive research** for sellers and marketers
- **Data-driven decision-making** for buyers and businesses

The Amazon Scraper addresses these needs by leveraging **Python web scraping libraries** to extract structured data directly from Amazon search results and product pages.

EXISTING METHODS :

Current approaches for tracking Amazon product data include:

1. **Manual Monitoring** – Time-consuming and limited for large datasets.
2. **Amazon API (Product Advertising API)** – Provides product info but requires API keys and has request limits.
3. **Third-Party Tools** – Price trackers or research platforms exist, but they often require **subscriptions** and lack customization.

Limitations:

- Restricted API access
- No full control over scraping/filtering logic
- Limited storage and analysis options

PROPOSED SOLUTION :

The Amazon Scraper overcomes existing limitations by directly extracting product data from the Amazon website using Python.

Key Features:

1. **Product Search & Extraction** – Name, price, rating, reviews, availability
2. **Filtering & Sorting** – By price, ratings, or availability
3. **Pagination Handling** – Scraping across multiple search pages
4. **Real-Time Price Tracking** – Monitor product fluctuations
5. **Image URL Extraction** – For analysis and reporting
6. **Storage Support** – CSV, JSON, Excel formats
7. **Proxy & Multi-threading** – Prevent blocking and speed up scraping

Advantages:

- Fully automated, customizable scraping
- Works without API restrictions
- Flexible storage and export options

TECHNOLOGIES TO BE USED :

1. **Language:** Python
2. **Libraries:**
    - **BeautifulSoup** – Parse HTML content
    - **Requests** – Fetch web pages
    - **Selenium** – Handle dynamic/JS-heavy pages
    - **Pandas** – Data organization & storage
    - **Threading/Asyncio** – Speed up scraping
    - **Proxy Middleware** – Avoid IP blocking
3. **Browser Automation:** Selenium + ChromeDriver (headless mode)
4. **Data Storage:** CSV, Excel, JSON

METHODS :

1. **Initialization** – Import required libraries, set headers & proxies
2. **Access Amazon** – Send requests or load via Selenium
3. **Data Extraction** – Extract product name, price, rating, reviews, availability, image URL
4. **Pagination Handling** – Scrape multiple result pages
5. **Data Processing** – Store in Pandas DataFrame
6. **Export Results** – Save as CSV, Excel, or JSON
7. **Automation** – Schedule scrapers for periodic runs
8. **Proxy Handling** – Rotate IPs to avoid blocks

IMPLEMENTATION / FOLDER STRUCTURE :

The project folder contains the following files:

- **amazon.py** – Core scraping script
- **searchresults.py** – Handles search result pages
- **selectors.yaml** – Defines scraping selectors (HTML tags)
- **urls.txt** – Input product/category URLs
- **requirements.txt** – Python dependencies
- **output.jsonl** – Scraped output data
- **search_results_output.jsonl** – Stored results for analysis

amazon.py :

```
from selectorlib import Extractor
import requests
import json
from time import sleep


# Create an Extractor by reading from the YAML file
e = Extractor.from_yaml_file('selectors.yml')
```

```
def scrape(url):


    headers = {

        'dnt': '1',

        'upgrade-insecure-requests': '1',

        'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36',

        'accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9',

        'sec-fetch-site': 'same-origin',

        'sec-fetch-mode': 'navigate',

        'sec-fetch-user': '?1',

        'sec-fetch-dest': 'document',

        'referer': 'https://www.amazon.com/',

        'accept-language': 'en-GB,en-US;q=0.9,en;q=0.8',

    }


    # Download the page using requests

    print("Downloading %s"%url)

    r = requests.get(url, headers=headers)

    # Simple check to check if page was blocked (Usually 503)

    if r.status_code > 500:

        if "To discuss automated access to Amazon data please contact" in r.text:

            print("Page %s was blocked by Amazon. Please try using better
proxies\n"%url)

        else:

            print("Page %s must have been blocked by Amazon as the status code was
%d"%(url,r.status_code))

        return None

    # Pass the HTML of the page and create
```

```
    return e.extract(r.text)


# product_data = []
with open("urls.txt",'r') as urllist, open('output.jsonl','w') as outfile:
    for url in urllist.read().splitlines():
        data = scrape(url)
        if data:
            json.dump(data,outfile)
            outfile.write("\n")
            # sleep(5)
```

searchresults.py :

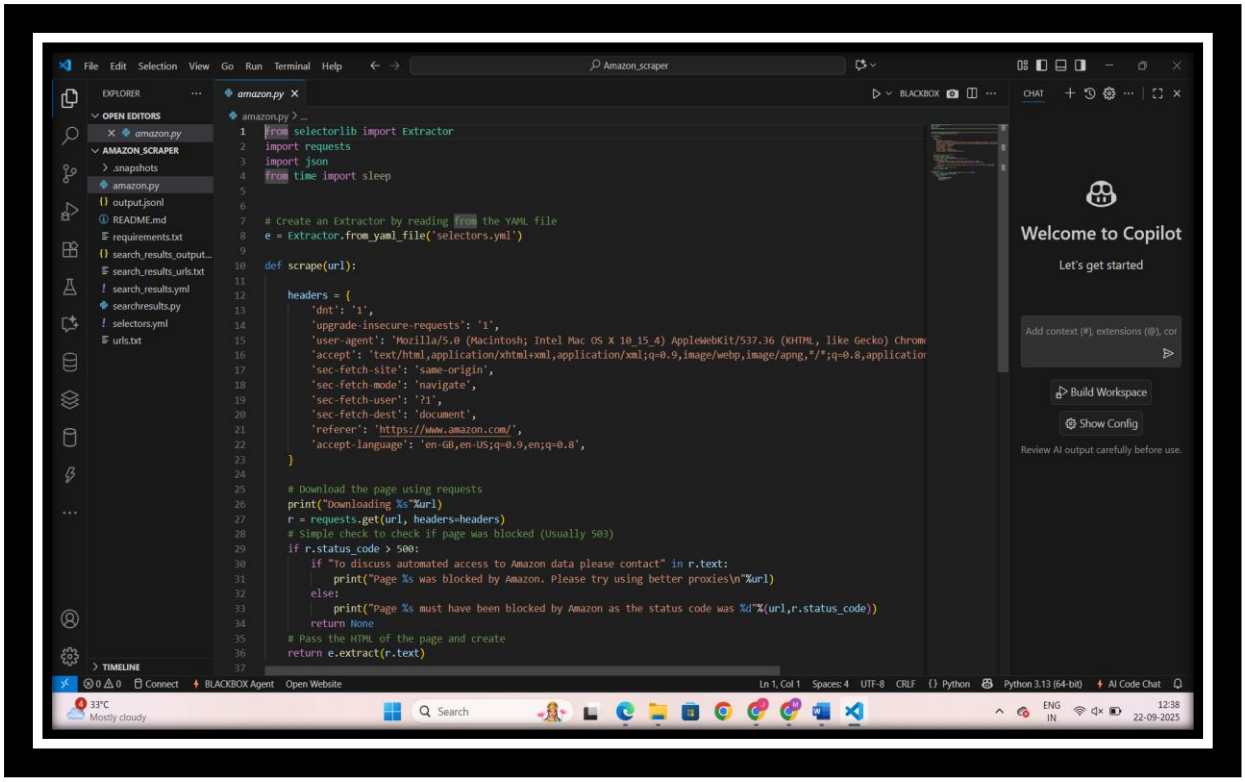
```
from selectorlib import Extractor
import requests
import json
from time import sleep


# Create an Extractor by reading from the YAML file
e = Extractor.from_yaml_file('search_results.yml')


def scrape(url):


    headers = {
        'dnt': '1',
        'upgrade-insecure-requests': '1',
        'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.61 Safari/537.36',
        'accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9',
        'sec-fetch-site': 'same-origin',
```

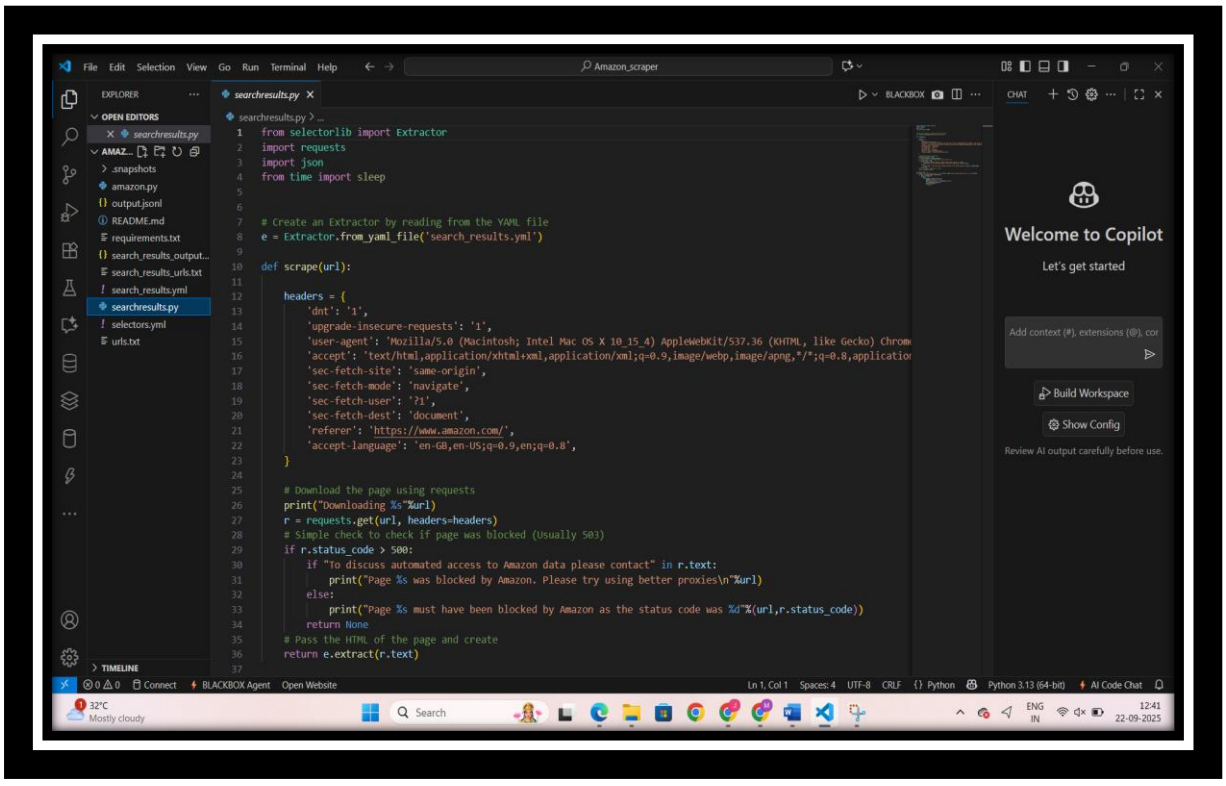
```
        'sec-fetch-mode': 'navigate',
        'sec-fetch-user': '?1',
        'sec-fetch-dest': 'document',
        'referer': 'https://www.amazon.com/',
        'accept-language': 'en-GB,en-US;q=0.9,en;q=0.8',
    }

    # Download the page using requests
    print("Downloading %s"%url)
    r = requests.get(url, headers=headers)
    # Simple check to check if page was blocked (Usually 503)
    if r.status_code > 500:
        if "To discuss automated access to Amazon data please contact" in r.text:
            print("Page %s was blocked by Amazon. Please try using better
proxies\n"%url)
        else:
            print("Page %s must have been blocked by Amazon as the status code was
%d"%(url,r.status_code))
        return None
    # Pass the HTML of the page and create
    return e.extract(r.text)

# product_data = []
with open("search_results_urls.txt",'r') as urllist,
open('search_results_output.jsonl','w') as outfile:
    for url in urllist.read().splitlines():
        data = scrape(url)
        if data:
            for product in data['products']:
                product['search_url'] = url
                print("Saving Product: %s"%product['title'])
```

```
                json.dump(product,outfile)
                outfile.write("\n")
                # sleep(5)
```

SAMPLE CODE SNIPPET :

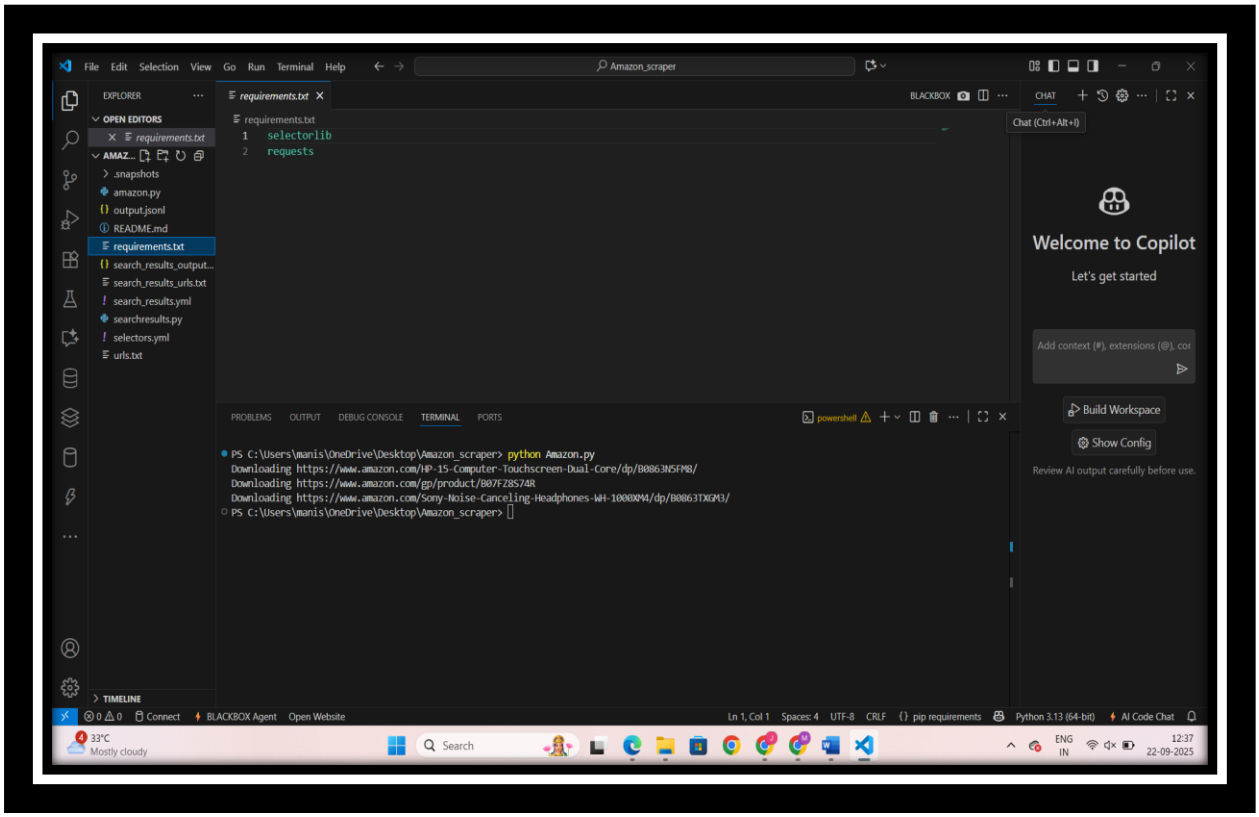
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

headers = {"User-Agent": "Mozilla/5.0"}
url = "https://www.amazon.in/s?k=laptops"
response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.content, "html.parser")

products = []
for item in soup.select('.s-main-slot .s-result-item'):
    name = item.h2.text if item.h2 else None
    price = item.select_one('.a-price .a-offscreen')
    price = price.text if price else None
    rating = item.select_one('.a-icon-alt')
    rating = rating.text if rating else None
    link = "https://www.amazon.in" + item.h2.a['href'] if item.h2 and item.h2.a else None

    products.append({"Name": name, "Price": price, "Rating": rating, "Link": link})

df = pd.DataFrame(products)
df.to_csv("amazon_products.csv", index=False)
```

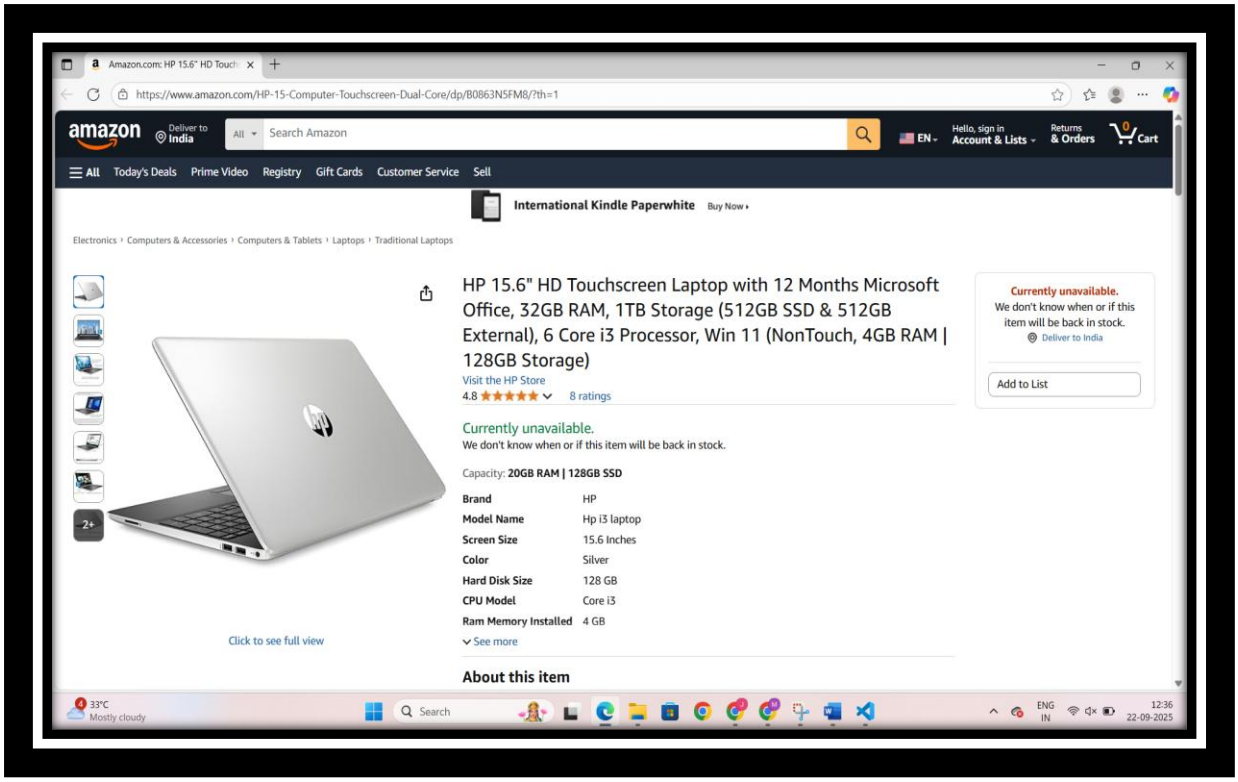

File Edit Selection View Go Run Terminal Help
Amazon_scraper
EXPLORER
OPEN EDITORS
amazon.py
AMAZON_SCRAPER
.snapshots
amazon.py
output.jsonl
README.md
requirements.txt
search_results_output...
search_results_urls.txt
search_results.yml
searchresults.py
selectors.yml
urls.txt
TIMELINE
amazon.py > ...
from selectorlib import Extractor
import requests
import json
from time import sleep
# Create an Extractor by reading from the YAML file
e = Extractor.from_yaml_file('selectors.yml')
def scrape(url):
headers = {
'dnt': '1',
'upgrade-insecure-requests': '1',
'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrom
'accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,applicatio
'sec-fetch-site': 'same-origin',
'sec-fetch-mode': 'navigate',
'sec-fetch-user': '?1',
'sec-fetch-dest': 'document',
'referer': 'https://www.amazon.com/',
'accept-language': 'en-GB,en-US;q=0.9,en;q=0.8',
}
# Download the page using requests
print("Downloading %s"%url)
r = requests.get(url, headers=headers)
# Simple check to check if page was blocked (Usually 503)
if r.status_code > 500:
if "To discuss automated access to Amazon data please contact" in r.text:
print("Page %s was blocked by Amazon. Please try using better proxies\n"%url)
else:
print("Page %s must have been blocked by Amazon as the status code was %d"%(url,r.status_code))
return None
Pass the HTML of the page and create
return e.extract(r.text)
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), cor
Build Workspace
Show Config
Review AI output carefully before use.
Connect
BLACKBOX Agent
Open Website
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python
Python 3.13 (64-bit)
AI Code Chat
33°C
Mostly cloudy
Search
ENG
IN
12:38
22-09-2025

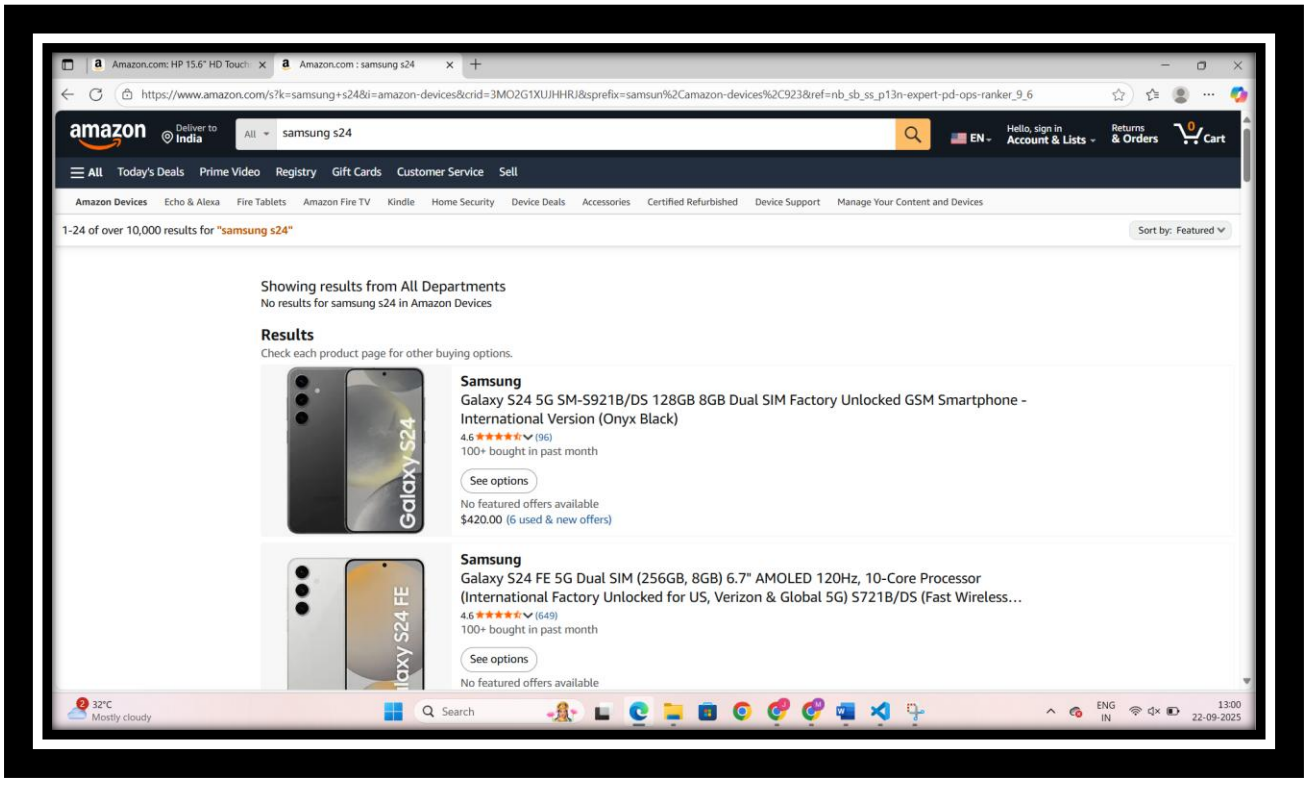

File Edit Selection View Go Run Terminal Help
Amazon_scraper
EXPLORER
OPEN EDITORS
searchresults.py
AMAZ...
.snapshots
amazon.py
output.jsonl
README.md
requirements.txt
search_results_output...
search_results_urls.txt
search_results.yml
searchresults.py
selectors.yml
urls.txt
TIMELINE
searchresults.py > ...
from selectorlib import Extractor
import requests
import json
from time import sleep
# Create an Extractor by reading from the YAML file
e = Extractor.from_yaml_file('search_results.yml')
def scrape(url):
headers = {
'dnt': '1',
'upgrade-insecure-requests': '1',
'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrom
'accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,applicatio
'sec-fetch-site': 'same-origin',
'sec-fetch-mode': 'navigate',
'sec-fetch-user': '?1',
'sec-fetch-dest': 'document',
'referer': 'https://www.amazon.com/',
'accept-language': 'en-GB,en-US;q=0.9,en;q=0.8',
}
# Download the page using requests
print("Downloading %s"%url)
r = requests.get(url, headers=headers)
# Simple check to check if page was blocked (Usually 503)
if r.status_code > 500:
if "To discuss automated access to Amazon data please contact" in r.text:
print("Page %s was blocked by Amazon. Please try using better proxies\n"%url)
else:
print("Page %s must have been blocked by Amazon as the status code was %d"%(url,r.status_code))
return None
Pass the HTML of the page and create
return e.extract(r.text)
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), cor
Build Workspace
Show Config
Review AI output carefully before use.
Connect
BLACKBOX Agent
Open Website
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
Python
Python 3.13 (64-bit)
AI Code Chat
32°C
Mostly cloudy
Search
ENG
IN
12:41
22-09-2025

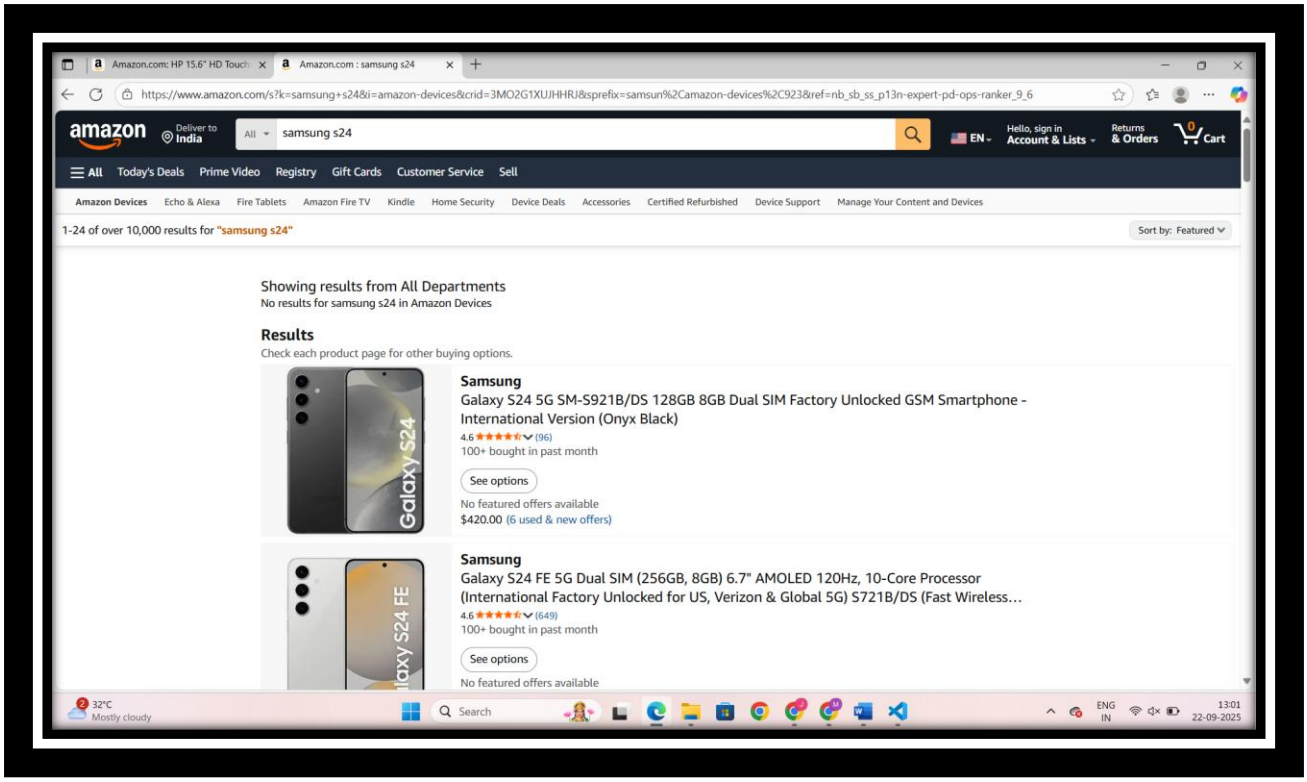
OUTPUT :

- CSV/JSON/Excel file containing:
  - Product Name
  - Price
  - Rating
  - Reviews
  - Availability
  - Image URL




Amazon.com: HP 15.6" HD Touch
https://www.amazon.com/HP-15-Computer-Touchscreen-Dual-Core/dp/B0863N5FM8/?th=1
amazon
Deliver to India
Search Amazon
EN
Hello, sign in Account & Lists
Returns & Orders
Cart
All
Today's Deals
Prime Video
Registry
Gift Cards
Customer Service
Sell
International Kindle Paperwhite
Buy Now
Electronics › Computers & Accessories › Computers & Tablets › Laptops › Traditional Laptops
HP 15.6" HD Touchscreen Laptop with 12 Months Microsoft Office, 32GB RAM, 1TB Storage (512GB SSD & 512GB External), 6 Core i3 Processor, Win 11 (NonTouch, 4GB RAM | 128GB Storage)
Visit the HP Store
4.8
8 ratings
Currently unavailable.
We don't know when or if this item will be back in stock.
Capacity: 20GB RAM | 128GB SSD
Brand HP
Model Name Hp i3 laptop
Screen Size 15.6 Inches
Color Silver
Hard Disk Size 128 GB
CPU Model Core i3
Ram Memory Installed 4 GB
See more
Currently unavailable.
We don't know when or if this item will be back in stock.
Deliver to India
Add to List
Click to see full view
About this item
33°C
Mostly cloudy
Search
ENG
IN
12:36
22-09-2025


Amazon.com: HP 15.6" HD Touch
Amazon.com : samsung s24
https://www.amazon.com/s?k=samsung+s24&i=amazon-devices&crid=3MO2G1XUJHHRJ&sprefix=samsun%2Camazon-devices%2C923&ref=nb_sb_ss_p13n-expert-pd-ops-ranker_9_6
amazon
Deliver to India
samsung s24
EN
Hello, sign in Account & Lists
Returns & Orders
Cart
All
Today's Deals
Prime Video
Registry
Gift Cards
Customer Service
Sell
Amazon Devices
Echo & Alexa
Fire Tablets
Amazon Fire TV
Kindle
Home Security
Device Deals
Accessories
Certified Refurbished
Device Support
Manage Your Content and Devices
1-24 of over 10,000 results for "samsung s24"
Sort by: Featured
Showing results from All Departments
No results for samsung s24 in Amazon Devices
Results
Check each product page for other buying options.
Samsung
Galaxy S24 5G SM-S921B/DS 128GB 8GB Dual SIM Factory Unlocked GSM Smartphone - International Version (Onyx Black)
4.6
(96)
100+ bought in past month
See options
No featured offers available
$420.00 (6 used & new offers)
Samsung
Galaxy S24 FE 5G Dual SIM (256GB, 8GB) 6.7" AMOLED 120Hz, 10-Core Processor (International Factory Unlocked for US, Verizon & Global 5G) S721B/DS (Fast Wireless…
4.6
(649)
100+ bought in past month
See options
No featured offers available
32°C
Mostly cloudy
Search
ENG
IN
13:00
22-09-2025


Amazon.com: HP 15.6" HD Touch
Amazon.com : samsung s24
https://www.amazon.com/s?k=samsung+s24&i=amazon-devices&crid=3MO2G1XUJHHRJ&sprefix=samsun%2Camazon-devices%2C923&ref=nb_sb_ss_p13n-expert-pd-ops-ranker_9_6
amazon
Deliver to India
All
samsung s24
EN
Hello, sign in Account & Lists
Returns & Orders
Cart
All
Today's Deals
Prime Video
Registry
Gift Cards
Customer Service
Sell
Amazon Devices
Echo & Alexa
Fire Tablets
Amazon Fire TV
Kindle
Home Security
Device Deals
Accessories
Certified Refurbished
Device Support
Manage Your Content and Devices
1-24 of over 10,000 results for "samsung s24"
Sort by: Featured
Showing results from All Departments
No results for samsung s24 in Amazon Devices
Results
Check each product page for other buying options.
Samsung
Galaxy S24 5G SM-S921B/DS 128GB 8GB Dual SIM Factory Unlocked GSM Smartphone - International Version (Onyx Black)
4.6 (96)
100+ bought in past month
See options
No featured offers available
$420.00 (6 used & new offers)
Samsung
Galaxy S24 FE 5G Dual SIM (256GB, 8GB) 6.7" AMOLED 120Hz, 10-Core Processor (International Factory Unlocked for US, Verizon & Global 5G) S721B/DS (Fast Wireless...
4.6 (649)
100+ bought in past month
See options
No featured offers available
32°C
Mostly cloudy
Search
ENG
IN
13:01
22-09-2025

FUTURE ENHANCEMENTS :

1. **GUI Dashboard** – Visual product trend analysis
2. **Database Integration** – MySQL/MongoDB for large-scale data
3. **Real-Time Alerts** – Notify users about price drops
4. **Multi-Website Support** – Flipkart, eBay, Walmart scraping
5. **Cloud Deployment** – Continuous scraping via AWS/Azure
6. **Machine Learning Integration** – Predict price trends

.

📌 FEATURES :

1. **Product Search & Extraction**
    - Extract product details such as names, prices, ratings, reviews, and availability.
2. **Filtering & Sorting**
    - Filter products by price, ratings, or availability for targeted analysis.
3. **Pagination Handling**
    - Scrape multiple result pages to build comprehensive product datasets.
4. **Real-Time Price Tracking**
    - Monitor product price fluctuations over time for best purchase decisions.
5. **Image URL Scraping**
    - Collect image URLs of products for reporting and dataset enrichment.
6. **Data Export**
    - Save results in CSV, JSON, or Excel formats for easy sharing and analysis.
7. **Multi-threading / Async Scraping**
    - Speed up data collection by running multiple requests simultaneously.
8. **Proxy Support**
    - Use proxy servers to bypass IP bans and ensure uninterrupted scraping.

CONCLUSION :

The Amazon Scraper demonstrates how **Python, BeautifulSoup, Selenium, and Pandas** can automate product data collection. It provides a customizable, extensible, and efficient way to monitor e-commerce product information. With enhancements like real-time alerts, dashboards, and ML-based forecasting, this scraper can evolve into a **comprehensive market research and consumer analysis tool**.

This system significantly reduces manual effort, enabling **real-time tracking** of e-commerce trends, **competitive analysis for sellers**, and **consumer behavior insights for researchers**. Its ability to store data in structured formats like CSV, JSON, and Excel makes it suitable for integration into **analytics pipelines, dashboards, and reporting systems**.

The scraper is **extensible and scalable**, with support for multi-threading, proxies, and pagination, making it adaptable for large-scale product monitoring across categories. In its current form, it provides immediate applications in **price comparison platforms, e-commerce market research, and academic studies**.

Looking ahead, the tool has the potential to evolve into a **comprehensive market intelligence solution** through enhancements such as **real-time alerts for price drops, visualization dashboards, cloud deployment for continuous monitoring, and machine learning models for predictive analysis**. With these improvements, the Amazon Scraper can serve as a **powerful decision-support system** for businesses, researchers, and individual consumers in the digital marketplace.

REFERENCES :

- Amazon – https://www.amazon.in
- BeautifulSoup Documentation – https://www.crummy.com/software/BeautifulSoup/
- Selenium Documentation – https://www.selenium.dev/documentation/
- Pandas Documentation – https://pandas.pydata.org/