**WEB SCRAPPING**

* By Sangini Phansekar

**Task1**:- Setting up Web Scrapping Tools

**Task Overview** :

Set up web scraping tools—Beautifulsoup, Scrapy, and Selenium—and ensure they are properly configured for extracting data from websites.

**Task Deliverable: -**

1. Three separate test scripts demonstrating functionality for each tool.
2. Setup guide and instructions for running the scripts.

Task Execution :-

Step 1 : Checking with python version (if not having python go with step 1 otherwise proceed with further.

**Python Installation**

1. Installed Python from the website.

<https://www.python.org/downloads/>

1. Verified installation of Python



1. Verified version

python –version

**Tool 1** -

**Beautifulsoup :- BeautifulSoup is a Python library used for parsing and extracting data from HTML and XML files. It is commonly used for web scraping tasks due to its simplicity and ease of use. BeautifulSoup works with the HTML or XML structure of web pages and helps extract the required data using selectors like tags, attributes, and text.**

**Step 1 :- Installing Required Libraries**

Command- **pip install requests beautifulsoup4**

*  **requests**: To fetch the HTML content of a webpage.
*  **BeautifulSoup** (from bs4): To parse and navigate through the HTML.

Here's a complete example fetching quotes from the "Quotes to Scrape" website.

import requests

from bs4 import BeautifulSoup

# URL of the Quotes to Scrape website

url = 'https://quotes.toscrape.com/'

# Send HTTP GET request to the website

response = requests.get(url)

# Check if the request was successful (status code 200)

if response.status\_code == 200:

print(f"Successfully fetched {url}")

else:

print(f"Failed to fetch {url}, Status code: {response.status\_code}")

exit()

# Parse the HTML content of the page

soup = BeautifulSoup(response.text, 'html.parser')

# Find all the quotes on the page

quotes = soup.find\_all('div', class\_='quote')

# Loop through each quote and extract the necessary information

for quote in quotes:

# Extract the quote text

quote\_text = quote.find('span', class\_='text').text

# Extract the author of the quote

author = quote.find('small', class\_='author').text

# Extract the tags associated with the quote (if any)

tags = [tag.text for tag in quote.find\_all('a', class\_='tag')]

# Print the quote, author, and tags

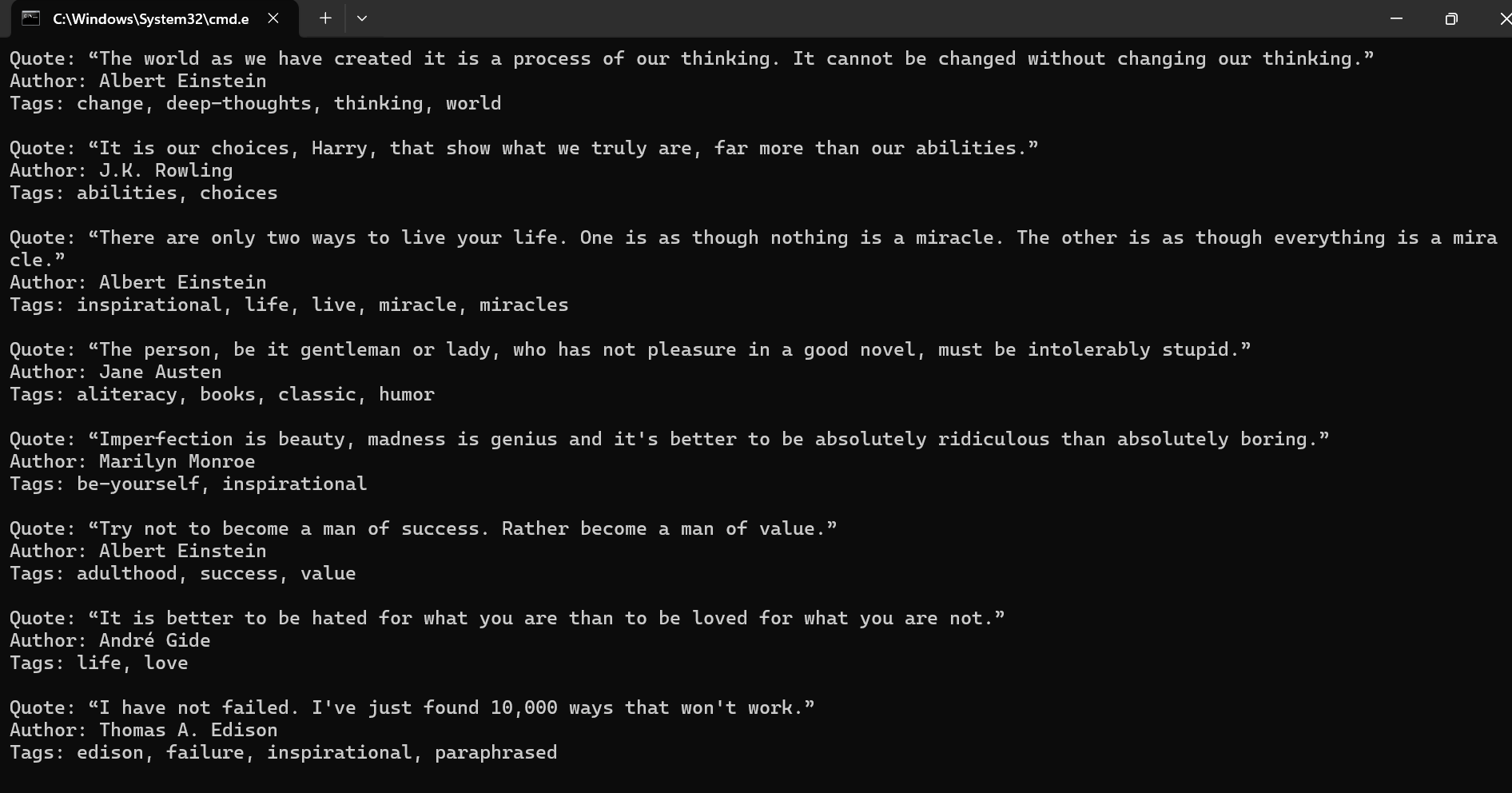
print(f"Quote: {quote\_text}")

print(f"Author: {author}")

print(f"Tags: {', '.join(tags)}\n")

* Saved above code in quotestobeautifulsoup.py file
* Run following command to execute

**Python quotestobeautifulsoup.py**



**Tool 2 :-**

**Scrapy :-** Scrapy is an **open-source and fast high-level web scraping framework** written in Python. It is designed for extracting data from websites and can also be used for other purposes like web crawling and automated testing. Scrapy handles many of the tedious tasks of scraping, such as request handling, response parsing, and following links.

**Step 1 :-**

* Go to the location (e.g. D:/WebNeuralInfotech/webscrapping/scrapy) open cmd and run command for installing scrapy library

**pip install scrapy**

* To show scrapy

**pip show scrapy**

**Step 2** :-

**Steps to Scrape Books**

**python -m scrapy startproject books\_scraper**

(Run Scrapy Using Python Explicitly (Optional): If adding to PATH is not possible, you can invoke Scrapy using Python directly)

(Otherwise **scrapy startproject books\_scraper**)

Step 3:-

Go into the spinders folder and make a python file books\_scrapy.py with the following code

import scrapy

class BooksSpider(scrapy.Spider):

name = "books"

start\_urls = ['http://books.toscrape.com']

def parse(self, response):

# Extract book details

for book in response.css('article.product\_pod'):

yield {

'title': book.css('h3 a::attr(title)').get(),

'price': book.css('.price\_color::text').get(),

'availability': book.css('.availability::text').re\_first('\S+'),

}

# Follow pagination links

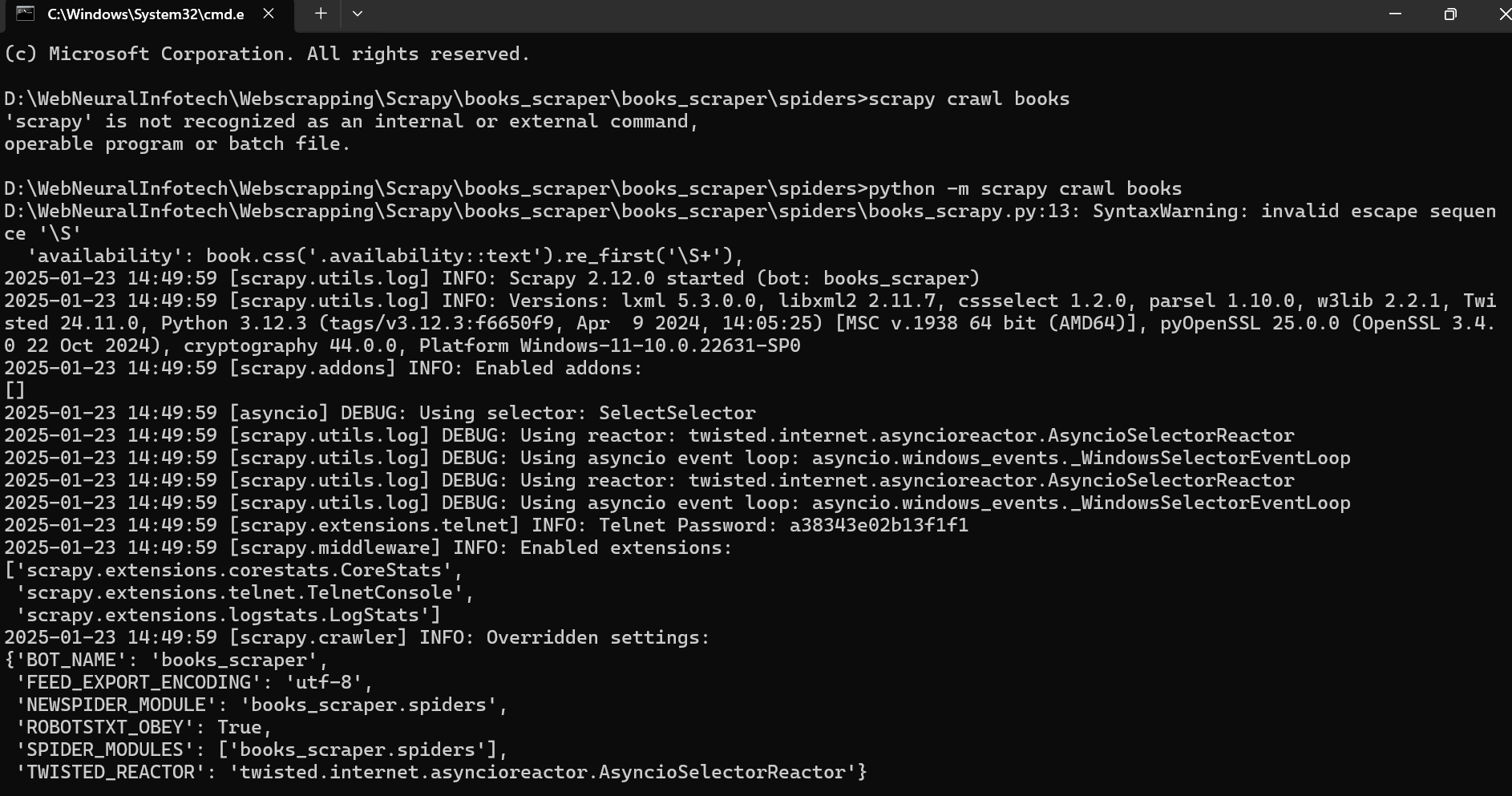
next\_page = response.css('li.next a::attr(href)').get()

if next\_page:

yield response.follow(next\_page, self.parse)

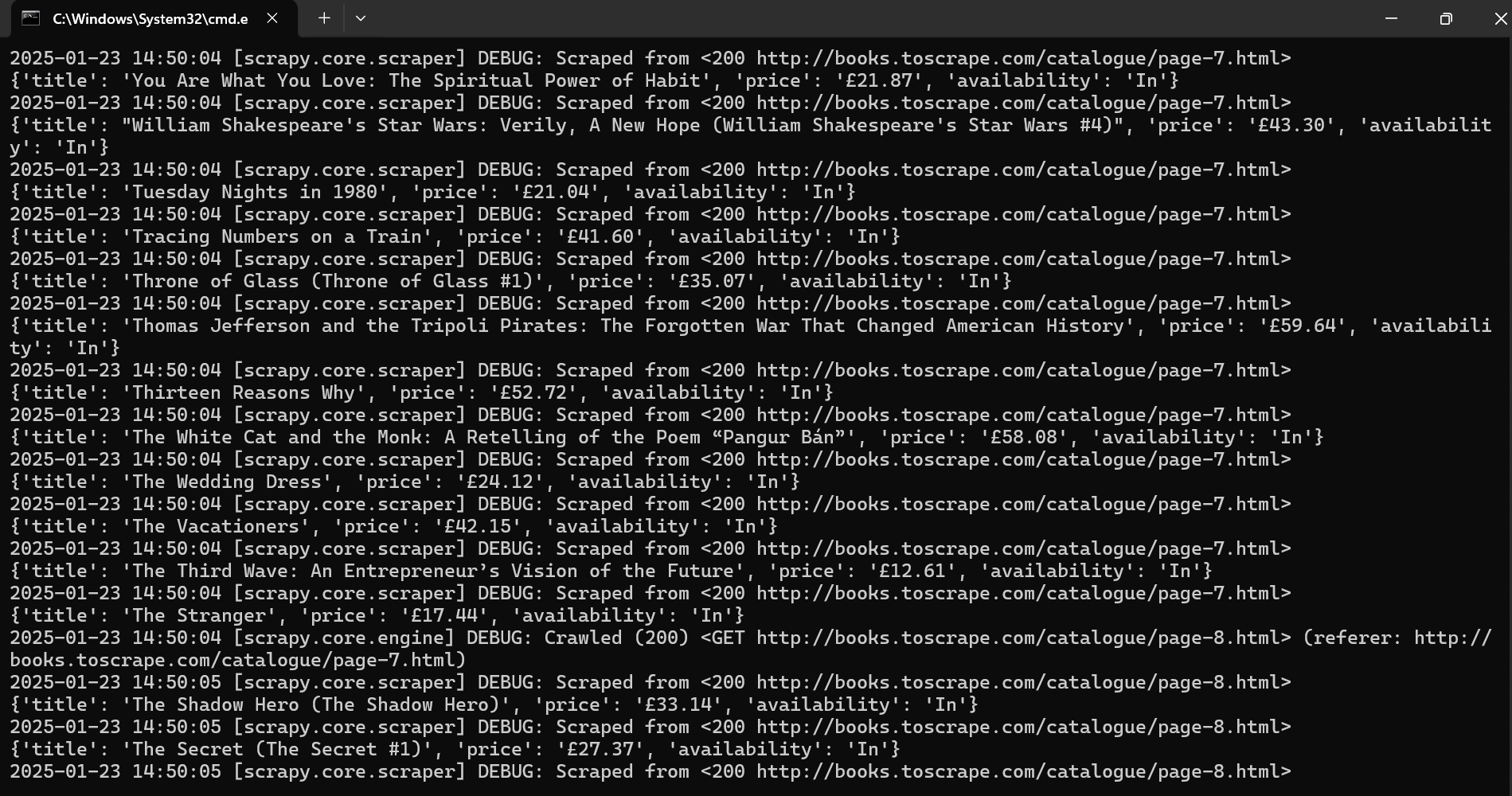
Step 3 :- Go to the spiders folder open command prompt to run

**python -m scrapy crawl books**

****

A screenshot of a computer program

Description automatically generated



Tool 3:

**Selenium:-** Selenium is a **powerful Python library** primarily used for **automating browsers**. It's often employed for testing web applications and scraping websites that use **JavaScript** to render dynamic content, which cannot be handled by tools like BeautifulSoup or Scrapy alone.

**Step 1:-**

Installing required Libraries

1. Install selenium for scrapping

**pip install selenium**

1. To check version

**pip show selenium**

**Step 2 :-**

1. Download Browser Drivers

For Chrome: Downloaded Chrome Driver from chromedriver.chromium.org

(Note:- run chromedriver.exe or save chromedriver.exe to the required forder)

run following commands in command prompt

1. To install webdriver

**pip install webdriver-manager**

1. Check installation

**pip show webdriver-manager**

**Step 3:-**

**To scrape and interact with the website Practice Test Automation (**[**https://practicetestautomation.com/practice-test-login/**](https://practicetestautomation.com/practice-test-login/)**)**

**Python Script :-**

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

import time

# Set up the Selenium WebDriver (e.g., using ChromeDriver)

driver = webdriver.Chrome() # Ensure you have the correct WebDriver installed

try:

# Open the Wikipedia homepage

driver.get("https://www.wikipedia.org/")

time.sleep(2) # Wait for the page to load

# Find the search input field

search\_box = driver.find\_element(By.ID, "searchInput")

# Enter the search term and press Enter

search\_term = "Web scraping"

search\_box.send\_keys(search\_term)

search\_box.send\_keys(Keys.RETURN)

time.sleep(3) # Wait for the results page to load

# Scrape the first paragraph of the article

first\_paragraph = driver.find\_element(By.XPATH, "//p[not(@class)]").text

print("First Paragraph:\n", first\_paragraph)

# Optionally, scrape the title of the page

title = driver.find\_element(By.ID, "firstHeading").text

print("\nPage Title:", title)

finally:

# Close the browser

driver.quit()

**Execution Command :-**

**python Wikipedia\_selenium.py**

A screenshot of a browser

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Scrape Snapdeal data as follows –

import requests

from bs4 import BeautifulSoup

import csv

def scrape\_snapdeal():

base\_url = "https://www.snapdeal.com"

search\_url = f"{base\_url}/search?keyword=electronics"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36"

}

response = requests.get(search\_url, headers=headers)

if response.status\_code != 200:

print("Failed to fetch the webpage.")

return

soup = BeautifulSoup(response.content, 'html.parser')

products = soup.find\_all('div', class\_='product-tuple-listing')

data = []

for product in products:

try:

product\_name = product.find('p', class\_='product-title').text.strip()

except AttributeError:

product\_name = "N/A"

try:

price = product.find('span', class\_='product-price').text.strip()

except AttributeError:

price = "N/A"

try:

rating\_div = product.find('div', class\_='filled-stars')

if rating\_div and 'style' in rating\_div.attrs:

rating = rating\_div['style'].split(':')[1].strip()

else:

rating = "N/A"

except (AttributeError, KeyError, IndexError):

rating = "N/A"

try:

reviews = product.find('div', class\_='product-review').text.strip()

except AttributeError:

reviews = "N/A"

try:

category = "Electronics" # Static category based on search query

except AttributeError:

category = "N/A"

try:

seller\_info = product.find('span', class\_='product-seller-name').text.strip()

except AttributeError:

seller\_info = "N/A"

try:

discount = product.find('span', class\_='product-discount').text.strip()

except AttributeError:

discount = "N/A"

try:

stock\_status = "In Stock" if "Add to Cart" in product.text else "Out of Stock"

except AttributeError:

stock\_status = "N/A"

data.append({

'Product Name': product\_name,

'Price': price,

'Rating': rating,

'Reviews': reviews,

'Category': category,

'Seller Info': seller\_info,

'Discount': discount,

'Stock Status': stock\_status

})

with open('snapdeal\_products.csv', mode='w', newline='', encoding='utf-8') as file:

writer = csv.DictWriter(file, fieldnames=data[0].keys())

writer.writeheader()

writer.writerows(data)

print("Data saved to snapdeal\_products.csv")

if \_\_name\_\_ == "\_\_main\_\_":

scrape\_snapdeal()

1. **Prepared CSV file with this as snapdeal\_products.csv**