Assignment Summary

Scrape Products script

Import Necessary Libraries:

The code starts by importing the required libraries: requests for making web requests and BeautifulSoup for parsing HTML.

Function Definition - scrape_products:

The main function is defined, which takes a URL and headers as inputs for web scraping.
Web Page Retrieval:

It makes an HTTP GET request to the specified URL using the provided headers to simulate a web browser's request.

Parsing the Web Page:

The HTML content of the web page is parsed using BeautifulSoup for easier data extraction.

Data Lists Initialization:

Lists to store product information like URL, name, rating, number of reviews, and price are initialized.

Loop Through Product Elements:

The script loops through product elements on the page. Each product element corresponds to a product listing.

Extract Product Details:

Inside the loop, it extracts the following information for each product:
Product URL: The URL to the product's page on Amazon.
Product Name: The name or title of the product.
Product Rating: The star rating of the product, if available.

Number of Reviews: The count of reviews for the product, if available.
Product Price: The price of the product.

Saving Data to CSV:

The scraped data is saved to a CSV file named 'amazon_products.csv.' If the file is empty (no header row), a header row is written to the file.

Print Confirmation:

A message is printed to confirm that the data has been saved to the CSV file.

Scrape Product script

Main Function scrape_product:

This function scrapes product information from a list of Amazon URLs.
It can use a proxy list to make requests, which can help prevent IP blocking.

Initialization:

It starts by initializing the URL column, proxy list, and other necessary variables.

Data Retrieval:

For each URL in the dataset, the script makes an HTTP request using requests.get().
It cycles through a list of proxies if use_proxy is set to True.
The script collects product descriptions and details by parsing the HTML content.

Handling Exceptions:

It catches and handles exceptions if any errors occur while processing the URLs.

Data Storage:

The product descriptions and details are stored in separate lists (products_description_list and product_details_list).

Keys Combination:

The script combines keys from the collected data to determine what to update in the CSV file.

CSV File Update:

The script updates the CSV file with new product details and descriptions using the update_products function.
If the update is successful, it prints "Update successful"; otherwise, it prints "Update failed."

Helper Functions:

The code includes several helper functions for data cleaning and formatting:
get_products: Retrieves data from an existing CSV file.
update_products: Updates the CSV file with new data.
update_rows: Updates a row in the data with new information.
fetch_data: Extracts product descriptions and details from the web page.
clean_details: Cleans and formats product details.
clean_description: Cleans and formats product descriptions.

Main Script

import Statements:

The script imports two functions from separate modules (scrape_product and scrape_products).

main_scrap Function:

This is the main entry point for the script.
It takes a boolean argument use_proxy, which indicates whether to use proxies for web scraping.
A headers dictionary is defined, containing user-agent information for making HTTP requests.

Amazon URL Setup:

The url variable is set to an Amazon URL, and the script is set to scrape up to 200 pages.

Loop for Scraping Pages:

A loop runs from page 1 to the total number of pages (200). Inside the loop, the scrape_products function is called with the URL for the current page, and headers.

Scrape_product Function:

After scraping the search result pages, the scrape_product function is called. which is the main function responsible for scraping detailed product information.
The headers and use_proxy arguments are passed to this function.

Execution:

Finally, the script checks if it's being run as the main program (__name__ == "__main__") and calls the main_scrap function with use_proxy set to False. This means that, by default, proxies are not used.