Design Laboratory (CS69202)
Spring Semester 2025

Topic : Web Scrapping Date: January 15, 2025

Important Instructions:

1. Using Python for this assignment is mandatory
2. For sending GET requests and receiving data – use the python requests library or the urllib3 library
3. For processing structured data that you receive back: JSON library
4. For processing unstructured data: beautifulsoup4 library which is to be called bs4
5. For storing data sqlite3 library should be used
6. All errors should be handled properly.

Example Code

Finally here is an example code example.py . It's fairly well documented, so first run it, then read it and ask questions. Now you are all set to write some scrapers:

Problem 1: Collecting and Storing structured JSON data (Working with NASA's Near-Earth Object Web Service (NeoWs) API)

==========

Part 0: API Key Registration

Sign up for a free NASA API key here. Once registered, you will receive a unique API key through your email that you can include in your requests.

Part 1: Fetch and Store Data in SQLite Database

Write a Python program that fetches data from NASA's NeoWs API for near-Earth objects based on a user-specified **start date** and **end date**. Extracts the following details for each asteroid and Stores the extracted data in a local SQLite database named `asteroid_data.`

- Asteroid name
- Estimated diameter (minimum and maximum, in meters)
- Hazardous status (boolean)
- Close approach date
- Velocity (in km/sec).

Part 2: Write Queries to Analyze the Data

Answer the following queries based on the data stored in `asteroid_data`

- **Query 1**: Write an SQL query to find the top 5 fastest asteroids (by velocity) and display their name, velocity, and hazardous status.
- **Query 2**: Write an SQL query to find all hazardous asteroids that are estimated to have a maximum diameter greater than 600 meters.

Part 3: Visualisation of Data using Seaborn

1. Create a Python program to visualise the asteroid data stored in the `asteroid_data`. Use the asteroid_data table to create a **bar plot** of the **top 5 fastest asteroids** (by velocity). The x-axis should display the asteroid names, and the y-axis should show their velocities (in km/sec). Use Seaborn's barplot for visualisation, and add labels, a title, and gridlines for better readability.

2. Using the `asteroid_data` table, create a **scatter plot** that shows the relationship between the **velocity** of asteroids and their **estimated maximum diameter** (in meters). The x-axis should represent the **velocity** (in km/sec), and the y-axis should represent the **estimated maximum diameter**. Use **colour coding** to distinguish between hazardous and non-hazardous asteroids. Add appropriate labels, a title, and a legend for clarity. Annotate the points for the top 5 fastest asteroids with their names.

Problem 2: Collecting, storing and processing unstructured data （Working with Wikipedia page data)

==========

Your task is to collect information about the EPL(English Premier League) from its Wikipedia page and process the data using Python's urllib3 / requests (to get data) and BeautifulSoup library (for parsing/processing) and store the collected data in an SQLite database.

1. Collect the main page of EPL Wikipedia for this task, the page is here: https://en.wikipedia.org/wiki/Premier_League. Note that you might need to use headers for fetching this page.
2. Now create a database Create a SQLite database named EPL.db' and a table named 'EPL_season2425'' with the following columns:
    a. Name of club
    b. Current standing position
    c. First season in PL
    d. Season in the top division
    e. Season in PL
    f. No of the season in the current spell of EPL
    g. Top division title
    h. The most recent top-division title
3. Collect the information of the managers for each of the clubs from the same URL. Store the following fields in a separate table 'EPL_season2425_managers'
    a. Manager
    b. Nationality
    c. Name
    d. Appointed
    e. Time as a manager
4. Now answer the following queries
    a. Who is the earliest-appointed manager among all clubs, and when did their club last win the top division title?
    b. What is the nationality of the manager leading the club with the highest number of top-division titles?
    c. Which club has spent the most consecutive seasons in the top division of the EPL, and is their manager the earliest-appointed among all clubs? [if yes, print the year, otherwise name the manager who was appointed the earliest]

Problem 3: Using multiple processes to speed up

==========

Now we will convert the above code to be run by multiple processes for speed up.

5. Collect the main page of EPL(English Premier League) Wikipedia for this task, the page is here: https://en.wikipedia.org/wiki/Premier_League. Note that you might need to use headers for fetching this page.
6. Now create a database Create a SQLite database named EPL.db' and a table named 'EPL_season2425_clubdetails'' with the following columns:
    a. Name of club
    b. Club_URL
    c. Current standing position
    d. First season in PL
    e. Season in the top division
    f. Season in PL
    g. No of the season in the current spell of EPL
    h. Top division title
    i. The most recent top-division title
    j. For each club fetch the following from [https://en.wikipedia.org/wiki/{*club_name*}]
        i. Number of First-team squad players
        ii. Number of players out on loan
        iii. Number of managers since the start [including the present one]
        iv. Average winning percentage of the managers [excluding the current one]
    k. DONE_OR_NOT_DONE (a 1 or 0 variable signifying whether fetched or not respectively)
7. Parse the HTML from step 5 and extract the common details of the clubs as well as the details of the clubs from their respective pages.
8. Insert the URL for the individual club for each row (in Club_URL) and set DONE_OR_NOT_DONE as 0 for all.
9. Now the handler code will spawn three processes using the **os.system** call.
    a. Example of this call

```
import os
os.system("python3 scraper.py&")
```

    b. This will run "python3 scraper.py" in a separate process.
10. This is what scraper.py will do
    a. It will check the database for rows where the DONE_OR_NOT_DONE flag is 0.
    b. It will pick a row where DONE_OR_NOT_DONE is 0 (if no such row, scraper.py will exit).
    c. For the row chosen, scraper.py will first set the DONE_OR_NOT_DONE to 1.

    d. Then it will fetch the specific page using the URL in the club_URL column, parse the page and populate the database with the data

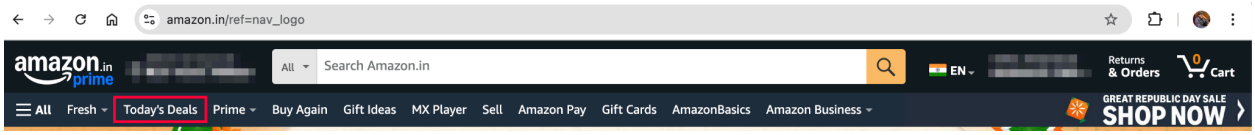
11. Write a checker.py code that can check the database and
    a. Report if all the database rows are populated, i.e., there is no DONE_OR_NOT_DONE which is set to 0 and no process is working (figure out how do you check that?)
    b. If all database rows are populated, then print answers to the following:
        i. Which club has the maximum number of players on loan?
        ii. Which club has the lowest average winning percentage of managers among the top 5 clubs that stayed in the top division for the maximum time?
        iii. Name the top 3 clubs that have the maximum First-team squad and stayed in all the years in top divisions since 1980.
    c. Finally, write a small text document documenting the percentage speed up in time you actually get by running multiple processes. What is your experiment set-up and results?

Problem 4: Automating Product Search and Scraping with Selenium

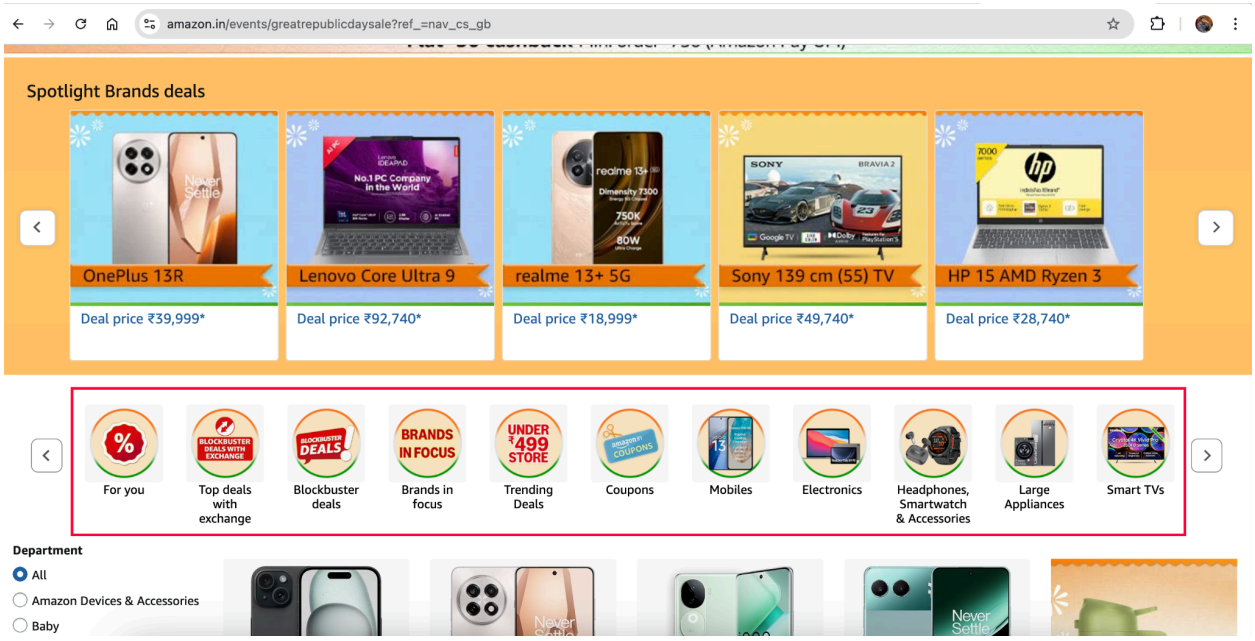
==========

The goal of this assignment is to build a Python script using Selenium to automate the process of searching for a product on an e-commerce website and extracting key details such as product name, price, ratings, reviews, and URLs.

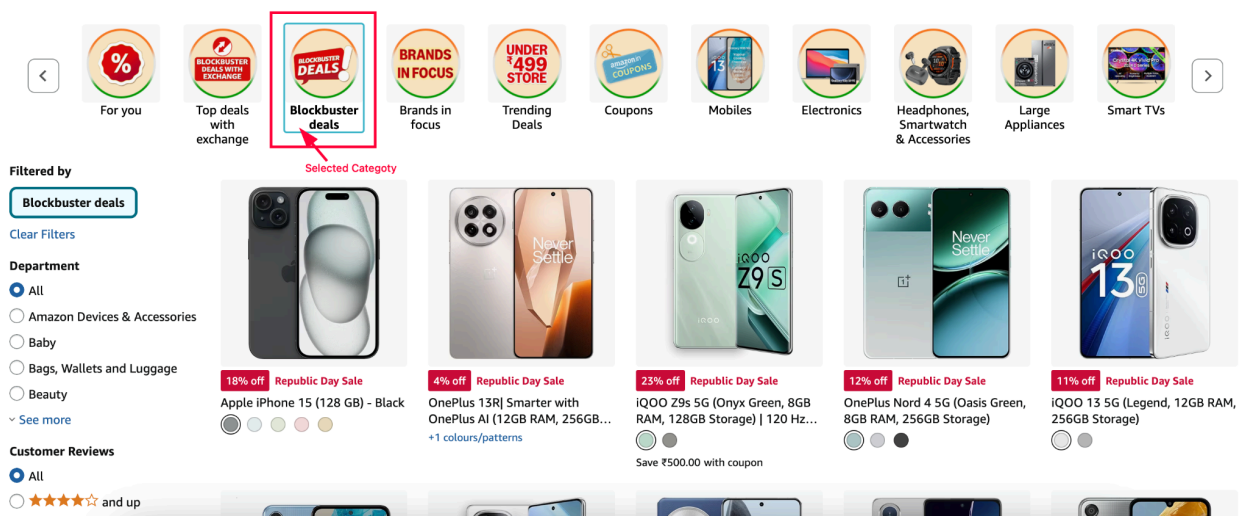
1. Write a Selenium script to navigate to an e-commerce website [https://www.amazon.in/]
2. Use the “Today’s Deal” component of the navbar and redirect there.



3. On the today’s deal page, go to the slider of categories



4.
5. For each selected category in this slider, multiple items appear.

6. 

7. Extract the following details for the first 10 products listed for all the categories:
    a. **Category Name**
    b. **Product Name**
    c. **Price**
    d. **Rating** (if available)
    e. **Number of Reviews** (if available)
    f. **Product URL**
8. Store the extracted details in a database file named `amazon.db` with appropriate tables and column headers
9. Implement error handling to manage missing elements or browser issues.