|  |  |
| --- | --- |
| **Name** | Manish Shashikant Jadhav |
| **UID** | 2023301005 |
| **Subject** | CITL (Cloud and Internet Technology Lab) |
| **Experiment No.** | 8 |
| **Aim** | Demonstrate the behavior of Web Crawlers/ spiders (use XPATH,CSS PATH),extract information and store it in the database. |
| **Theory** | **1. Introduction to Web Crawling**  **Web Crawlers/Spiders**:   * **Definition**: Web crawlers, also known as spiders or bots, are automated programs that browse the internet systematically to index content and gather information. * **Functionality**: They navigate through web pages by following hyperlinks and retrieving content, which can be stored and processed for various applications, such as search engines, data mining, and analytics. * **Use Cases**: Common uses include indexing for search engines (like Google), gathering data for research, monitoring changes in websites, and scraping data for analysis.   **2. Web Scraping Techniques Data Extraction**:   * Web scraping involves extracting data from web pages. This can be   achieved using various techniques, with two common methods being **XPATH**  and **CSS selectors**.  **XPATH**:   * **Definition**: XPATH is a query language used to select nodes from an XML document. It can also be used to navigate HTML documents. * **Syntax**: XPATH uses a path-like syntax to specify the location of elements in a document. For example, //div[@class='example'] selects all <div> elements with a class of "example". * **Advantages**: XPATH is powerful for complex queries and allows for precise element selection, including attributes and text content.   **CSS Selectors**:   * **Definition**: CSS selectors are used to select elements in HTML based on their attributes, types, classes, and IDs. * **Syntax**: For example, .example selects all elements with the class "example", and #uniqueID selects the element with the ID "uniqueID".   **Advantages**: CSS selectors are generally easier to use and understand, making them suitable for straightforward data extraction tasks. |
| **Code** | * **euler.py:**   *import* requests  *from* bs4 *import* BeautifulSoup  *import* sqlite3  *import* matplotlib.pyplot *as* plt *# type: ignore*  *import* os  print("Current working directory:", os.getcwd())  *# Set up the database*  conn = sqlite3.connect('newProjectEuler.db')  c = conn.cursor()  c.execute('CREATE TABLE IF NOT EXISTS problems (id INTEGER PRIMARY KEY, title TEXT, solved\_count INTEGER)')  *# Iterate through all pages*  all\_problems = []  *for* page *in* range(1, 20):      url = f'https://projecteuler.net/archives;page={page}'      print(f"Fetching data from: {url}")      response = requests.get(url)  *# Check for a successful response*  *if* response.status\_code != 200:          print(f"Failed to retrieve data from {url}, status code: {response.status\_code}")  *continue*      soup = BeautifulSoup(response.content, 'html.parser')  *# Extract information from the current page*      page\_problems = []  *for* row *in* soup.select('tr'):          id\_column = row.select\_one('td.id\_column')          title\_column = row.select\_one('td:nth-of-type(2) a')          solved\_count\_column = row.select\_one('td:nth-of-type(3) div.center')  *if* id\_column and title\_column and solved\_count\_column:              problem\_id = int(id\_column.text.strip())              title = title\_column.text.strip()              solved\_count = int(solved\_count\_column.text.strip().replace(',', ''))              page\_problems.append((problem\_id, title, solved\_count))  *# Append the current page's problems to the total list*      all\_problems.extend(page\_problems)  *# Insert the extracted data into the database*      c.executemany('INSERT OR IGNORE INTO problems (id, title, solved\_count) VALUES (?, ?, ?)', page\_problems)      conn.commit()  *# Print the total number of problems extracted*  print(f"Total problems extracted: {len(all\_problems)}")  print(all\_problems)  *# Query the data for plotting*  c.execute('SELECT id, solved\_count FROM problems')  data = c.fetchall()  *# Prepare data for plotting*  *if* data:      ids, solved\_counts = zip(\*data)    *# Plotting the data*      plt.scatter(ids, solved\_counts)      plt.xscale('linear')      plt.yscale('log') *# Use a log scale for the y-axis*      plt.xlabel('Problem ID')      plt.ylabel('Number of Solved Users (Log Scale)')      plt.title('Number of Users Solved Problems on Project Euler')      plt.grid(False)      plt.show()  *else*:      print("No data available for plotting.")  *# Find the problems solved the most and least*  *if* data:      most\_solved = max(data, *key*=lambda *x*: *x*[1])      least\_solved = min(data, *key*=lambda *x*: *x*[1])      print(f"Problem with ID {most\_solved[0]} has been solved the most with {most\_solved[1]} solutions.")      print(f"Problem with ID {least\_solved[0]} has been solved the least with {least\_solved[1]} solutions.")  *# Close the database connection*  conn.close()  print("Data extraction and storage completed.")   * **newProjectEuler.py:**   *import* sqlite3  *# Connect to the database*  conn = sqlite3.connect('newProjectEuler.db')  c = conn.cursor()  *# Fetch all rows from the 'problems' table*  c.execute('SELECT \* FROM problems')  rows = c.fetchall()  *# Check if there is any data and print it*  *if* rows:      print("Data Stored in Database:")  *for* row *in* rows:          print(row)  *else*:      print("No data found in Database.")  *# Close the database connection*  conn.close() |
| **Output** | **Webpage: https://projecteuler.net/archives**    **CSS Selector:**  <tr><td class="id\_column">1</td><td><a href="problem=1" title="Published on Friday, 5th October 2001, 06:00 pm">Multiples of 3 or 5</a></td><td><div class="center">1010203</div></td></tr>  **Database Used -** SQLite3  **Visualization –** Python-Matplotlib     * **Scatter Plot:** |
| **Conclusion** | Hence by completing this experiment I got to know how to Demonstrate the behavior of Web Crawlers/ spiders (use XPATH,CSS PATH),extract information and store it in the database. |