BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

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 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".

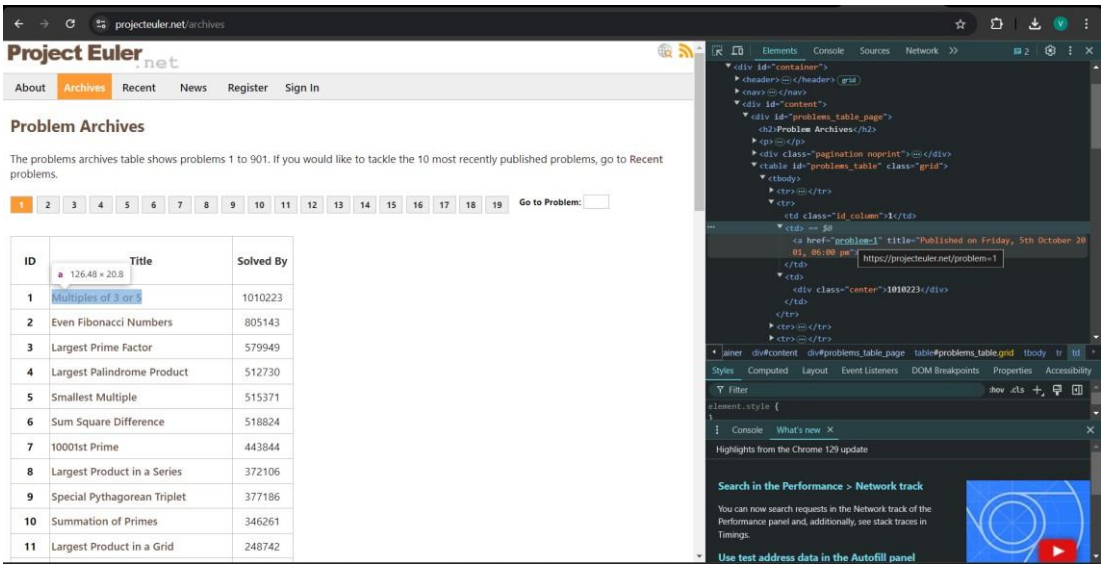
	Advantages: CSS selectors are generally easier to use and understand, making them suitable for straightforward data extraction tasks.
Code	• **euler.py:** *import* requests *from* bs4 *import* BeautifulSoup *import* sqlite3 *import* matplotlib.pyplot *as* plt *# type: ignore* *import* os print("Current 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.")
```

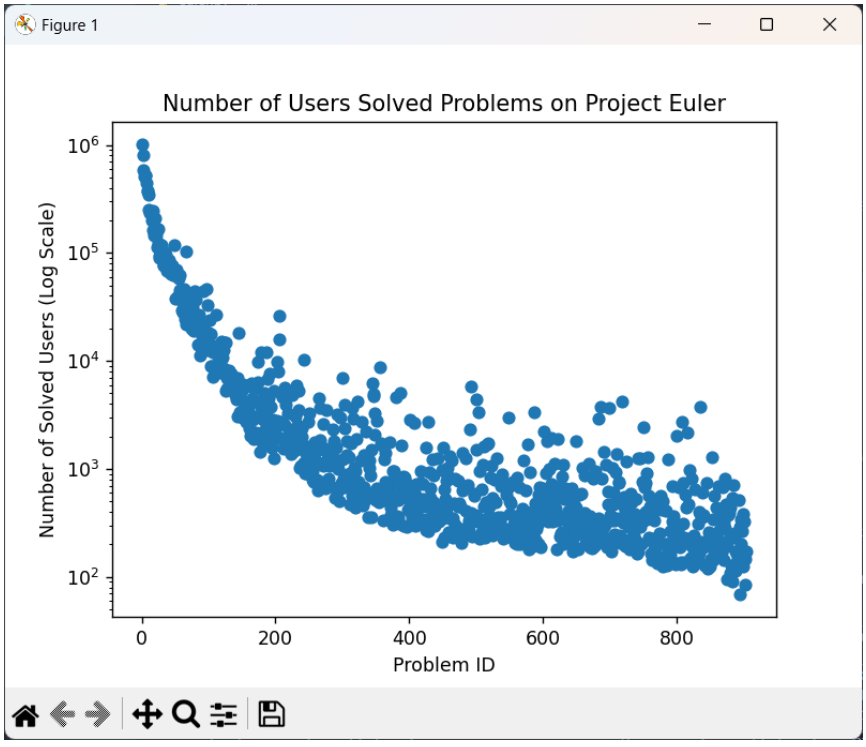
- **newProjectEuler.py:**

```
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>Multiples of 3 or 5</td><td> 1010203 </td></tr> **Database Used -** SQLite3 **Visualization –** Python-Matplotlib

```
PS D:\LEO\crawler_citl> python euler.py
Current working directory: D:\LEO\crawler_citl
Fetching data from: https://projecteuler.net/archives;page=1
Fetching data from: https://projecteuler.net/archives;page=2
Fetching data from: https://projecteuler.net/archives;page=3
Fetching data from: https://projecteuler.net/archives;page=4
Fetching data from: https://projecteuler.net/archives;page=5
Fetching data from: https://projecteuler.net/archives;page=6
Fetching data from: https://projecteuler.net/archives;page=7
Fetching data from: https://projecteuler.net/archives;page=8
Fetching data from: https://projecteuler.net/archives;page=9
Fetching data from: https://projecteuler.net/archives;page=10
Fetching data from: https://projecteuler.net/archives;page=11
Fetching data from: https://projecteuler.net/archives;page=12
Fetching data from: https://projecteuler.net/archives;page=13
Fetching data from: https://projecteuler.net/archives;page=14
Fetching data from: https://projecteuler.net/archives;page=15
Fetching data from: https://projecteuler.net/archives;page=16
Fetching data from: https://projecteuler.net/archives;page=17
Fetching data from: https://projecteuler.net/archives;page=18
Fetching data from: https://projecteuler.net/archives;page=19
Total problems extracted: 904
[(1, 'Multiples of 3 or 5', 1012319), (2, 'Even Fibonacci Numbers', 806701),
Difference', 519776), (7, '$10\\,001$st Prime', 444637), (8, 'Largest Product
a Grid', 249141), (12, 'Highly Divisible Triangular Number', 235731), (13, 'L
 'Number Letter Counts', 162183), (18, 'Maximum Path Sum I', 155302), (19, 'C
3, 'Non-Abundant Sums', 112375), (24, 'Lexicographic Permutations', 123083),
0  46 mins  Add Logs  Improve Code  Version Control  Share Code Link  Search Error
```

- **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.