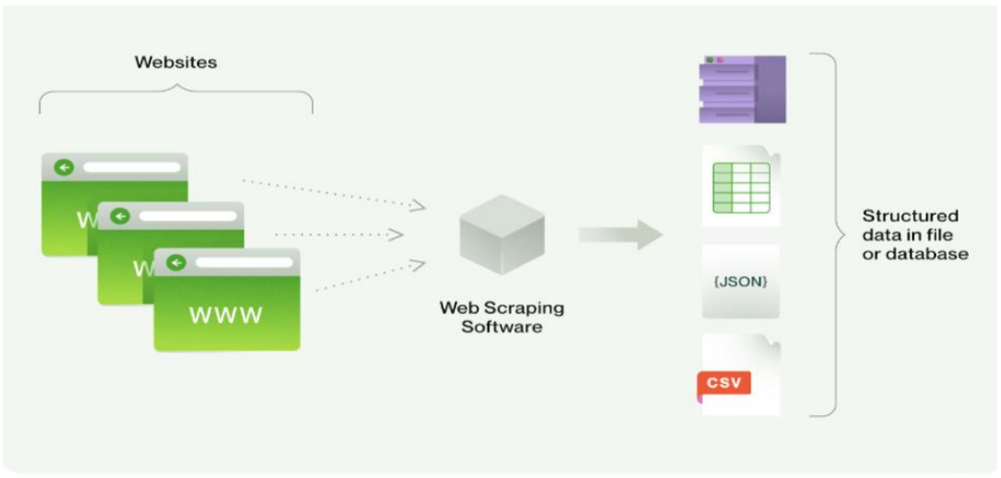
Principles of Data Visualization Assignment – 1

1. What is Web Scraping, explain with an example.

- Web scraping is an automatic method to obtain large amounts of data from websites.
- Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications.
- There are many different ways to perform web scraping to obtain data from websites. These include using online services, particular API's or even creating your code for web scraping from scratch.
- Web scraping requires two parts, namely the **crawler** and the **scraper**.
- The crawler is an artificial intelligence algorithm that browses the web to search for the particular data required by following the links across the internet.
- The scraper, on the other hand, is a specific tool created to extract data from the website.



Example: Product Prices

Imagine you're interested in a specific product and want to monitor its price on an e-commerce website. We can use web scraping to automatically extract the product's price and track any changes over time. By doing so, we can make informed purchasing decisions based on historical price data.

2. What are the challenges in performing web scraping.

1) **Website Structure Changes:**
    - **Unpredictable Updates:** Websites can modify their structure without notice, causing your scraping code to break and require immediate adjustments.
    - **Maintenance Overhead:** Continuous monitoring and updating of scraping scripts are necessary to keep up with evolving website layouts.
2) **Dynamic Content:**
    - **JavaScript Rendering:** Websites that heavily rely on JavaScript to load content may require the use of headless browsers or API requests to access the dynamically generated data.
    - **Increased Complexity:** Scraping dynamic content often involves additional steps to simulate user interactions and retrieve the data after it has been dynamically loaded.
3) **Ethical and Legal Concerns:**
    - **Terms of Use Violations:** Scraping without the website owner's permission can lead to legal action or being blocked from accessing the site.
    - **Data Privacy:** Scraping personal or sensitive data without proper consent can result in ethical and legal consequences, especially in jurisdictions with strict data protection laws.
4) **Rate Limiting and IP Blocking:**
    - **IP Blocking:** Excessive requests from the same IP address can lead to temporary or permanent IP blocking by the website's server.
    - **Mitigation Strategies:** Implementing delays between requests, rotating IP addresses, or using proxy servers can help avoid rate limiting and blocks.
5) **Data Quality and Cleaning:**
    - **Inconsistent Formatting:** Scraped data might come in various formats, requiring data cleaning to make it usable and uniform.
    - **Irrelevant Information:** Websites often contain ads, navigation links, or unrelated content that needs to be filtered out from the scraped data.

3. Describe the basic architecture for performing web scraping.

1) **URL Request:**

   - Choose target website and specific pages for scraping.
   - Send HTTP requests to retrieve HTML content.

2) **HTML Parsing:**

   - Use parsing libraries (e.g., BeautifulSoup) to parse HTML.
   - Identify relevant HTML elements, tags, classes.

3) **Data Extraction:**

   - Extract desired data from parsed HTML.
   - Employ regular expressions or methods for extraction.

4) **Data Transformation:**

   - Clean extracted data (remove artifacts, format issues).
   - Convert data to appropriate formats (text to numbers, dates).

5) **Pagination Handling:**

   - Set up iteration for multiple pages.
   - Modify URLs or follow next-page links.

6) **Error Handling:**

   - Implement error handling for loading or server errors.
   - Include retry logic for temporary issues.

7) **Headers and User Agents:**

   - Include headers, user agent info in requests.

8) **Rate Limiting:**

   - Implement rate limiting, throttling to avoid overloading.

9) **Ethical Considerations:**

- Respect robots.txt, terms of use.
- Comply with website's guidelines.

10) **Monitoring and Maintenance:**

- Regularly check and update scraping scripts.
- Adapt to changes in website structure.

4. Describe various techniques of web scraping.

1) **HTTP Requests:**

- Web scraping begins with sending HTTP requests to a target website's servers.
- The two most common HTTP methods are GET (retrieve data) and POST (send data).
- Libraries like **requests** in Python facilitate sending these requests.

2) **HTML Parsing:**

- Websites use HTML (Hypertext Markup Language) to structure content.
- Parsers like BeautifulSoup and lxml in Python help parse the HTML and extract relevant data.

3) **API Integration:**

- Some websites offer APIs (Application Programming Interfaces) to access data in a structured format.
- APIs provide a more controlled and efficient way of extracting data compared to traditional scraping.

4) **Headless Browsers:**

- A headless browser is a web browser without a graphical user interface.

- Tools like Puppeteer and Selenium allow interaction with websites in a headless mode, enabling dynamic content scraping.

5) **Regular Expressions:**

- Regular expressions (regex) are powerful for pattern matching in text.
- They can be used to extract specific information from raw HTML or text data.

6) **Captcha Handling:**

- Some websites use Captcha to prevent automated scraping.
- Captcha solving services or browser automation techniques are employed to bypass them.

7) **IP Rotation and Proxies:**

- To avoid IP bans or rate limitations, rotating IP addresses and using proxies is common.
- Proxies route requests through different IP addresses, making scraping less detectable.

8) **User-Agent Spoofing:**

- Websites often identify user agents (browser details) to distinguish between human and automated traffic.
- Spoofing user agents helps mimic different browsers and devices.