Web Scraping

Web Scraping is the process of automatically extracting information from Websites

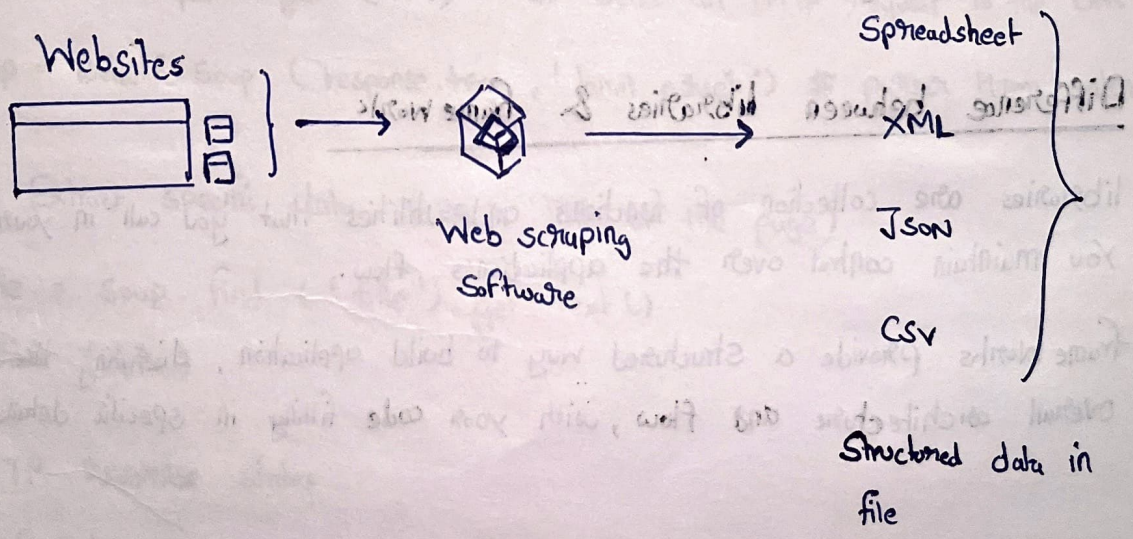
Key Concepts

1) HTML Structure :- Structure of web pages (HTML). You need to know how to navigate the tags to extract the information.

2) HTTP Requests :- Web scraping involves making HTTP requests to the server to fetch the web pages.

3) Parsing HTML :- After fetching the HTML, need to parse it to extract the relevant data. This is done using libraries in programming lang.

How web Scraping Works



Scraping refers to the process of extracting data from the website

Tools and libraries

1) **Beautiful Soup** :- A python library for parsing HTML and XML documents It creates a parse tree for parsing HTML documents making it easy to extract data.

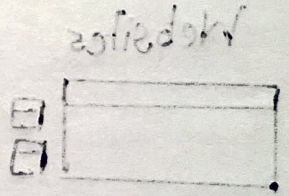
Parsing means process of analyzing code or data to determine its grammatical structure according to given sets of rules. and translates it into more manageable structure.

2) **Requests** :- A python library for making HTTP requests. It simplifies sending HTTP requests to the server.

3) **Selenium** :- A browser automation tool. It is useful for scraping dynamic websites where the content is loaded via javascript

4) **Scrapy** :- An open-source and collaborative web crawling framework for Python. It's a more advanced tool for large-scale web scraping projects.

Difference between libraries & frame work



- libraries are collection of functions and utilities that you call in your code. You maintain control over the application's flow.
- frame works provide a structured way to build application, dictating the overall architecture and flow, with your code filling in specific details.

Application of Web Scraping

- Retail & manufacturing
- financial Research
- Data science
- Marketing and Sales

Getting Started

- Install Required libraries
  - pip install requests beautifulsoup4

Simple Scraper

```
import requests
from bs4 import BeautifulSoup

url = "http://example.com"

response = requests.get(url)  # Send an HTTP request to the URL
soup = BeautifulSoup(response.text, 'html.parser') # parser HTML content

# Extract specific data (eg:- Title of the page)
title = soup.find('title').get_text()
print(title)
```

HTTP response status

1) Informational responses (100 - 199)
2) Successful responses (200 - 299)
3) Redirection messages (300 - 399)
4) Client error responses (400 - 499)
5) Server error responses (500 - 599)

Installing a parser

Beautiful Soup supports the HTML parser included in Python's standard library, but it also supports a number of third-party Python parsers.

Parser	
Python's html.parser	
lxml's HTML parser	
lxml's XML parser	

Beautiful Soup, a Python library used for web scraping

'find()'

'find_all()'

'find all()'

are method for searching the parsed HTML

CODE

```
<body>
<a href = " " > link 1 </a>
<a href = " " > link 2 </a>
<a href = " " > link 3 </a>
" "
Soup = BeautifulSoup (html_doc, 'html.parser')
tag = Soup.find ('a')
print (tag)
```

find()

- Searches for the first tag that matches the specified criteria
- Return a single 'Tag' object if a match is found, otherwise return None

- find_all()

```
all_tags = soup.find_all ('a')
for tag in all_tags:
    print (tag)
```

- Searches for all tags that match the specified criteria
- Returns a list of 'tags' objects if matches are found, otherwise returns an empty list.

Web Scraping E-Commerce Product information Using Beautiful Soup and Pandas

- Beautiful Soup helps in parsing HTML content fetched from a webpage.
- Extract Data : You can easily navigate and extract specific data from the parsed HTML.

Pandas

Store Data : Store the extracted data in Dataframe

Clean Data : Clean and transform the Data as needed

Analyze Data : Perform Analysis, filtering, and aggregation on the Data

Export Data : Export the cleaned data to formats like CSV or Excel

A Dataframe is a two-Dimensional, size-mutable, and potentially heterogeneous tabular data structure in pandas

CODE

```
import requests
import pandas as pd
from bs4 import BeautifulSoup

url = "http://www.amazon.com--/"

r = requests.get(url)

Soup = BeautifulSoup(r.text, "lxml")

names = soup.find_all("a", class_ = "title")
    print(names) # will give the HTML structure of all tags
                 with a

product_name = []

for i in names:
    name = i.text
    product_name.append(name)

print(product_name)   # will only print the product_name

prices = soup.find_all("h4", class_ = "prices-right")

prices_list = []

for i in prices:
    price = i.text
    prices_list.append(price)

print(price_list)   # will print the prices
```

Now we will use pandas to create Data frame

```
df = pd.Dataframe ({"Product Name": product_name, "Price": prices_list})
print (df)   # This will print product Name and prices in tabular form
             tabular form
df.to_csv ("products_Details.csv")

df.to_excel ("products_Details Excel.csv")
                    Name of file
```

#

To only find the specific box we use index.

Here [3] specifies the index

```
box = soup.find_all ("div", class_ = "...") [3]
```

Specifies the index of box which we want (Information)

Important

- If a website don't allow us to extract their HTML Page, then the response is <response [403]>
- If allow scraping then response will be <response [200]>