Web Scraping Exercise

Web Scraping allows you to gather large volumes of data from diverse and real-time online sources. This data can be crucial for enriching your datasets, filling in gaps, and providing current information that enhances the quality and relevance of your analysis. Web scraping enables you to collect data that might not be readily available through traditional APIs or databases, offering a competitive edge by incorporating unique and comprehensive insights. Moreover, it automates the data collection process, saving time and resources while ensuring a scalable approach to continuously updating and maintaining your datasets.

Ethical web scraping involves respecting website terms of service, avoiding overloading servers, and ensuring that the collected data is used responsibly and in compliance with privacy laws and regulations.

Use Python, `requests`, `BeautifulSoup` and/or `pandas` to scrape web data:

Import Libraries

In [1]:
```
import requests
from bs4 import BeautifulSoup
```

Define the Target URL

In [2]:
```
url = "http://finance.yahoo.com/quote/AAPL"

#here, AAPL is the ticker symbol which can be replaced with other tickers
```

Send a Request to the Website

Do not forget to check the response status code

In [3]:
```
request = requests.get(url)
```

Parse the HTML Content

Use a library to access the HTMl content

In [4]:
```
soup = BeautifulSoup(request.content, 'html.parser')
soup.prettify()
```

Out[4]:
```
'Edge: Too Many Requests\n'
```

As we run into an "Too many requests" issue on the first try, it seems like we have to spoof the user agent to avoid bot detection.

In [5]:
```
import random

USER_AGENTS = [
    # Windows - Chrome
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
    # Windows - Firefox
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/
    # macOS - Safari
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 13_2_1) AppleWebKit/605.1.15 (KHTML,
    # macOS - Chrome
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
    # Linux - Chrome
    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
    # Linux - Firefox
    "Mozilla/5.0 (X11; Linux x86_64; rv:119.0) Gecko/20100101 Firefox/119.0",
    # Edge - Windows
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
    # Android - Chrome Mobile
    "Mozilla/5.0 (Linux; Android 12; Pixel 5) AppleWebKit/537.36 (KHTML, like Ge
    # iPhone - Safari
    "Mozilla/5.0 (iPhone; CPU iPhone OS 17_3 like Mac OS X) AppleWebKit/605.1.15
    # iPad - Safari
    "Mozilla/5.0 (iPad; CPU OS 16_6 like Mac OS X) AppleWebKit/605.1.15 (KHTML,
]

headers = {
    "User-Agent": random.choice(USER_AGENTS),
}

response = requests.get(url, headers=headers, timeout=10)
response.raise_for_status()
soup = BeautifulSoup(response.text, "html.parser")

#soup
```

Identify the Data to be Scraped

Write a couple of sentence on the data you want to scrape

I want to scrape the title (name of the company), which is contained in an h1 element of class "yf-xxeib":

<h1 class="yf-xxbei9">Apple Inc. (AAPL)</h1>

Extract Data

Find specific elements and extract text or attributes from elements (handle pagination if necessary)

In [6]:
```
for h1 in soup.find_all("h1"):
    title = h1.get_text(strip=True)
```

```
    print(title)
```

```
Yahoo Finance
Apple Inc. (AAPL)
```

It seems like there is another <h1> element for the page title in the same html, so we have to check for the classname as well to get the right one

In [7]:
```
title = soup.find("h1", class_="yf-xxbei9").get_text(strip=True)
title
```

Out[7]:
```
'Apple Inc. (AAPL)'
```

In [8]:
```
name = title.rsplit("(", 1)[0].strip()
name
```

Out[8]:
```
'Apple Inc.'
```

Looks good, now I want to scrape the names of all the companies in my dataset

In [9]:
```
import json
from time import sleep

TICKER_LIST = [
    "AAPL", "MSFT", "AMZN", "TSLA", "GOOG", "NVDA", "BRK-B", "JNJ",
    "V", "WMT", "XOM", "JPM", "O", "PG", "HD", "PFE", "MA", "UNH", "BAC",
    "PEP", "KO", "DIS", "CVX", "AVGO", "MRK", "LLY", "ABBV", "INTC", "T",
    "CSCO", "CMCSA", "MCD", "NKE", "ADBE", "CRM", "COST", "WFC", "ABT", "TXN",
    "AMGN", "QCOM", "UPS", "LOW", "IBM", "GE", "CAT", "DE", "ORCL", "BA",
    "MDT", "MS", "GS", "LMT", "VRTX", "ADI", "FDX", "ZTS", "SBUX", "DHR"
]
base_url = "http://finance.yahoo.com/quote/"


ticker_name_map = {}

for ticker in TICKER_LIST:
    sleep(1)

    url = base_url + ticker

    headers = {
        "User-Agent": random.choice(USER_AGENTS),
    }

    request = requests.get(url, headers=headers, timeout=10)
    soup = BeautifulSoup(request.text, "html.parser")
    title = soup.find("h1", class_="yf-xxbei9").get_text(strip=True)
    name = title.rsplit("(", 1)[0].strip()
    print(ticker + ": " +name)
    ticker_name_map[ticker] = name
```

AAPL: Apple Inc.
MSFT: Microsoft Corporation
AMZN: Amazon.com, Inc.
TSLA: Tesla, Inc.
GOOG: Alphabet Inc.
NVDA: NVIDIA Corporation
BRK-B: Berkshire Hathaway Inc.
JNJ: Johnson & Johnson
V: Visa Inc.
WMT: Walmart Inc.
XOM: Exxon Mobil Corporation
JPM: JPMorgan Chase & Co.
O: Realty Income Corporation
PG: The Procter & Gamble Company
HD: The Home Depot, Inc.
PFE: Pfizer Inc.
MA: Mastercard Incorporated
UNH: UnitedHealth Group Incorporated
BAC: Bank of America Corporation
PEP: PepsiCo, Inc.
KO: The Coca-Cola Company
DIS: The Walt Disney Company
CVX: Chevron Corporation
AVGO: Broadcom Inc.
MRK: Merck & Co., Inc.
LLY: Eli Lilly and Company
ABBV: AbbVie Inc.
INTC: Intel Corporation
T: AT&T Inc.
CSCO: Cisco Systems, Inc.
CMCSA: Comcast Corporation
MCD: McDonald's Corporation
NKE: NIKE, Inc.
ADBE: Adobe Inc.
CRM: Salesforce, Inc.
COST: Costco Wholesale Corporation
WFC: Wells Fargo & Company
ABT: Abbott Laboratories
TXN: Texas Instruments Incorporated
AMGN: Amgen Inc.
QCOM: QUALCOMM Incorporated
UPS: United Parcel Service, Inc.
LOW: Lowe's Companies, Inc.
IBM: International Business Machines Corporation
GE: GE Aerospace
CAT: Caterpillar Inc.
DE: Deere & Company
ORCL: Oracle Corporation
BA: The Boeing Company
MDT: Medtronic plc
MS: Morgan Stanley
GS: The Goldman Sachs Group, Inc.
LMT: Lockheed Martin Corporation
VRTX: Vertex Pharmaceuticals Incorporated
ADI: Analog Devices, Inc.
FDX: FedEx Corporation
ZTS: Zoetis Inc.
SBUX: Starbucks Corporation
DHR: Danaher Corporation

Store Data in a Structured Format

Give a brief overview of the data collected (e.g. count, fields, ...)

In [10]:
```
print(f"Number of Tickers in starting list: {len(TICKER_LIST)}")
print(f"Number of Company names: {len(ticker_name_map)}")
```

```
Number of Tickers in starting list: 59
Number of Company names: 59
```

The .json file looks like this (truncated):

{

"AAPL": "Apple Inc.",

"MSFT": "Microsoft Corporation",

"AMZN": "Amazon.com, Inc.",

"TSLA": "Tesla, Inc.",

"GOOG": "Alphabet Inc.",

"NVDA": "NVIDIA Corporation",

"BRK-B": "Berkshire Hathaway Inc.",

Save the Data

In [11]:
```
# Save to JSON file
with open("ticker_names.json", "w", encoding="utf-8") as f:
    json.dump(ticker_name_map, f, indent=2, ensure_ascii=False)

print("\nSaved to ticker_names.json")
```

```
Saved to ticker_names.json
```