Web scraping is a data extraction method that collects data only from websites. It is often used for data mining and gathering valuable insights from large websites. Web scraping is also useful for personal use. Python includes a nice library called BeautifulSoup that enables web scraping. In this article, we will extract current stock prices using web scraping and save them in an excel file using Python.

Required Modules

In this article, we'll look at how to work with the Requests, Beautiful Soup and Pandas Python packages to consume data from websites.

- Requests module allows you to integrate your Python programs with web services.
- Beautiful Soup module is designed to make screen scraping a snap. Using Python's interactive console and these two libraries, we'll walk through how to assemble a web page and work with the textual information available on it.
- Pandas module is designed to provide high-performance data manipulation in Python. It is used for data analysis that requires lots of processing, such as restructuring, cleaning or merging, etc.

Approach

- Initially, we are going to import our required libraries.
- Then we take the URL stored in our list.
- We will feed the URL to our soup object which will then extract relevant information from the given URL based on the class id we provide it.
- Store all the data in the Pandas Dataframe and save it to a CSV file.

Note: The HTML structure of stock data websites may change frequently.

- Before running this script, inspect the webpage and update the element IDs, classes or XPath selectors accordingly.

- Use browser developer tools (F12 - Inspect Element) to find the correct elements for price, change and volume.

🗒️ Imports and Headers

python
CopyEdit
```
import requests
```

- Purpose: Allows sending HTTP requests (GET, POST, etc.)
- If Missing: You cannot fetch webpage data. requests.get() will raise a NameError.

python
CopyEdit
```
from bs4 import BeautifulSoup as temp
```

- Purpose: Imports BeautifulSoup HTML parser and renames it to temp for convenience.
- If Missing: temp(...) will throw a NameError. HTML parsing will not be possible.

python
CopyEdit
```
import pandas as pd
```

- Purpose: Required to use DataFrame for tabular data and export to Excel.
- If Missing: pd.DataFrame() and df.to_excel() will not work.

python
CopyEdit
```
import time
```

- Purpose: Enables time.sleep() to pause between requests.
- If Missing: time.sleep() will raise a NameError, and your script may get blocked by the server due to too many rapid requests.

🌐 Headers for Web Request

python
CopyEdit
```
headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36'
}
```

- Purpose: Mimics a real browser so that websites don't block your bot.
- If Missing: Many modern websites (like Groww) may block or return incomplete data due to missing headers.

📌 URLs to Scrape

python
CopyEdit
```
urls = [
    'https://groww.in/us-stocks/nke',
    ...
]
```

- Purpose: Stores a list of Groww stock URLs to scrape.
- If Missing: The loop has no URLs to process — nothing will be scraped.

📦 Data Storage List

python
CopyEdit
```
all_data = []
```

- Purpose: Collects the scraped data in list format.
- If Missing: You can't store the data — later you'll get a NameError or an empty DataFrame.

🔁 Main Loop for Scraping

python
CopyEdit
```
for url in urls:
```

- Purpose: Loops through all URLs one by one.
- If Missing: You'd be scraping just one URL manually or not at all.

🌐 Web Request

python
CopyEdit
page = requests.get(url, headers=headers, timeout=10)

- Purpose: Sends GET request to the current URL.
- If Missing: You won't have the HTML content to parse, and soup = temp(...) will fail.

🧽 Parse HTML

python
CopyEdit
soup = temp(page.text, 'html.parser')

- Purpose: Converts raw HTML string into a searchable BeautifulSoup object.
- If Missing: You can't extract data using .find().

🔍 Extract Stock Info

Each of the following lines searches for specific HTML elements and extracts their text if found:
python
CopyEdit
company_tag = soup.find('h1', {'class': 'usph14Head displaySmall'})
company = company_tag.text if company_tag else 'N/A'

- Purpose: Extracts the company name.
- If Missing: You'll have no company name or get an error later when storing data.

python
CopyEdit
price_tag = soup.find('span', {'class': 'uht141Pri contentPrimary displayBase'})
price = price_tag.text if price_tag else 'N/A'

- Purpose: Extracts current stock price.
- If Missing: You'd skip a core piece of information — the price.

python
CopyEdit
change_tag = soup.find('div', {'class': 'uht141Day bodyBaseHeavy contentNegative'})
change = change_tag.text if change_tag else 'N/A'

- Purpose: Gets how much the stock price changed today.
- If Missing: Your data won't reflect daily movement (gain/loss).

python
CopyEdit
volume_tag = soup.find('table', {'class': 'tb10Table col l5'})
volume = volume_tag.find_all('td')[1].text if volume_tag else 'N/A'

- Purpose: Gets trading volume from the second <td> in the table.
- If Missing: You lose information on market activity level.

📥 Store in List

python
CopyEdit
all_data.append([company, price, change, volume])

- Purpose: Saves the extracted data for each stock in a list.
- If Missing: Your final DataFrame will be empty.

🛑 Error Handling

python
CopyEdit
except Exception as e:
    print(f"Error scraping {url}: {e}")

- Purpose: Prevents the script from crashing if something fails (like a network error or missing tag).
- If Missing: One error will stop the entire script.

🕑 Sleep Between Requests

python
CopyEdit
time.sleep(10)

- Purpose: Prevents rate-limiting by waiting 10 seconds between each request.
- If Missing: Server may block or throttle your IP after multiple fast requests.

🧮 Pandas Table Creation

python
CopyEdit
columns = ["company", "price", "change", "volume"]
df = pd.DataFrame(all_data, columns=columns)

- Purpose: Converts list of lists into a structured DataFrame.
- If Missing: You won't be able to work with or export data in tabular format.

💾 Save as Excel

python
CopyEdit
df.to_excel("stocks.xlsx", index=False)

- Purpose: Saves the data as an Excel file.
- If Missing: Your scraped data will not be stored — only in memory and lost after script ends.

✅ Summary of What Will Break If Omitted

Line of Code	What Breaks If Missing
import requests	Can't fetch pages
headers = {...}	Website may block you
urls = [...]	No stocks to scrape

time.sleep(10)	May get IP blocked
try-except	Whole script crashes on 1 error
append(...)	No data collected
pd.DataFrame(...)	Can't tabulate data
df.to_excel(...)	No saved output

Let me know if you'd like me to turn this into a Jupyter notebook or add progress logging or retry logic for failed URLs.