**Demo Video Script: Parsing CNBC Data**

**Introduction**

“This demo shows how two Python scripts work together to get market data and news from CNBC. The scripts are:

1. web\_scraper.py – It saves the webpage as an HTML file.
2. data\_filter.py – It reads the HTML and pulls out useful data into CSV files.”

**Part 1: web\_scraper.py**

**Purpose**  
“We use this script to load the CNBC webpage, make sure it’s fully loaded, and save it as HTML.”

**Steps**:

1. **Set up Selenium**:
   * “We use a headless Chrome browser to avoid opening a visible browser window.”
   * “We set the path to chromedriver and add options like --headless.”
2. **Open and Wait**:
   * “The script opens the CNBC page and waits a few seconds for everything to load, including JavaScript.”
3. **Save the HTML**:
   * “We get the final HTML from the browser using driver.page\_source.”
   * “The script writes the HTML file to data/raw\_data/web\_data.html for later use.”

**Result**  
“We now have a saved copy of the webpage with all its data.”

**Part 2: data\_filter.py**

“We use this script to pull specific data from the HTML file: market data and news.”

1. **Market Data**

**Goal**: Collect the **symbol**, **position**, and **percent change** for each stock or index.

**Steps**:

1. Find the market data section:
   * “The HTML has a part with id="market-data-scroll-container". This is where all market cards are located.”
   * “We use soup.find("div", id="market-data-scroll-container") to find it.”
2. Look inside each market card:
   * “Each card has class="MarketCard-container". We use find\_all to get all cards.”
3. Pull out details:
   * **Symbol**: find("span", class\_="MarketCard-symbol").text (e.g., "AAPL" for Apple).
   * **Position**: find("span", class\_="MarketCard-stockPosition").text (e.g., "Tech").
   * **Percent Change**: find("span", class\_="MarketCard-changesPct").text (e.g., "+1.23%").

**Result**  
“We save this data into a CSV file for easy analysis.”

1. **Latest News**

**Goal**: Collect the **time**, **headline**, and **link** for each news article.

**Steps**:

1. Find the news section:
   * “The HTML has a list with class="LatestNews-list". It holds all the news items.”
   * “We use soup.find("ul", class\_="LatestNews-list") to find it.”
2. Look inside each news item:
   * “Each item has class="LatestNews-item". We use find\_all to get them.”
3. Pull out details:
   * **Time**: find("time", class\_="LatestNews-timestamp").text (e.g., "2 hours ago").
   * **Headline**: find("a", class\_="LatestNews-headline").text (e.g., "Stock market rises today").
   * **Link**: find("a", class\_="LatestNews-headline")["href"] (the URL of the article).

**Result**  
“We save these details into another CSV file for reference.”

**Conclusion**

“Our two scripts work together to collect data.

1. web\_scraper.py saves the webpage.
2. data\_filter.py pulls out symbols, positions, changes, news times, headlines, and links.

By using the right HTML class names and IDs, we turn raw data into clean CSV files ready for analysis.”