Lab 1 Solution

# Installation & Setup

* Development done locally in a local git repository on a windows machine.
* The code is synced with a remote git repository.

A screenshot of a computer

Description automatically generated

* A Linux Ubuntu instance is spin up using GitHub Codespace for the repository.

A screenshot of a computer

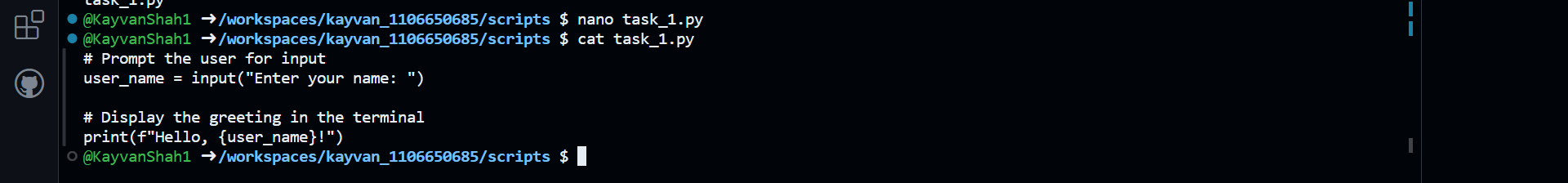
Description automatically generated

# Get Familiar with Linux and Python

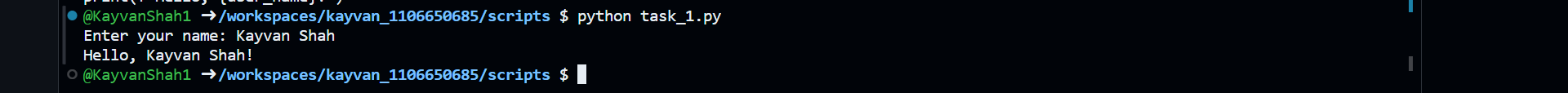
## Playing around with Linux Terminal

A screenshot of a computer

Description automatically generated



## A basic Python Script



## Web Scraping Task

* Create a virtual environment and install the dependencies using the ***requirements.txt***.
  + *venv* – Created a virtual environment using command
  + Linux has a built-in support for Chrome and using a library **chromedriver-py** we automatically manage the webdriver downloads and retrieval of the executable path while initializing the web driver.

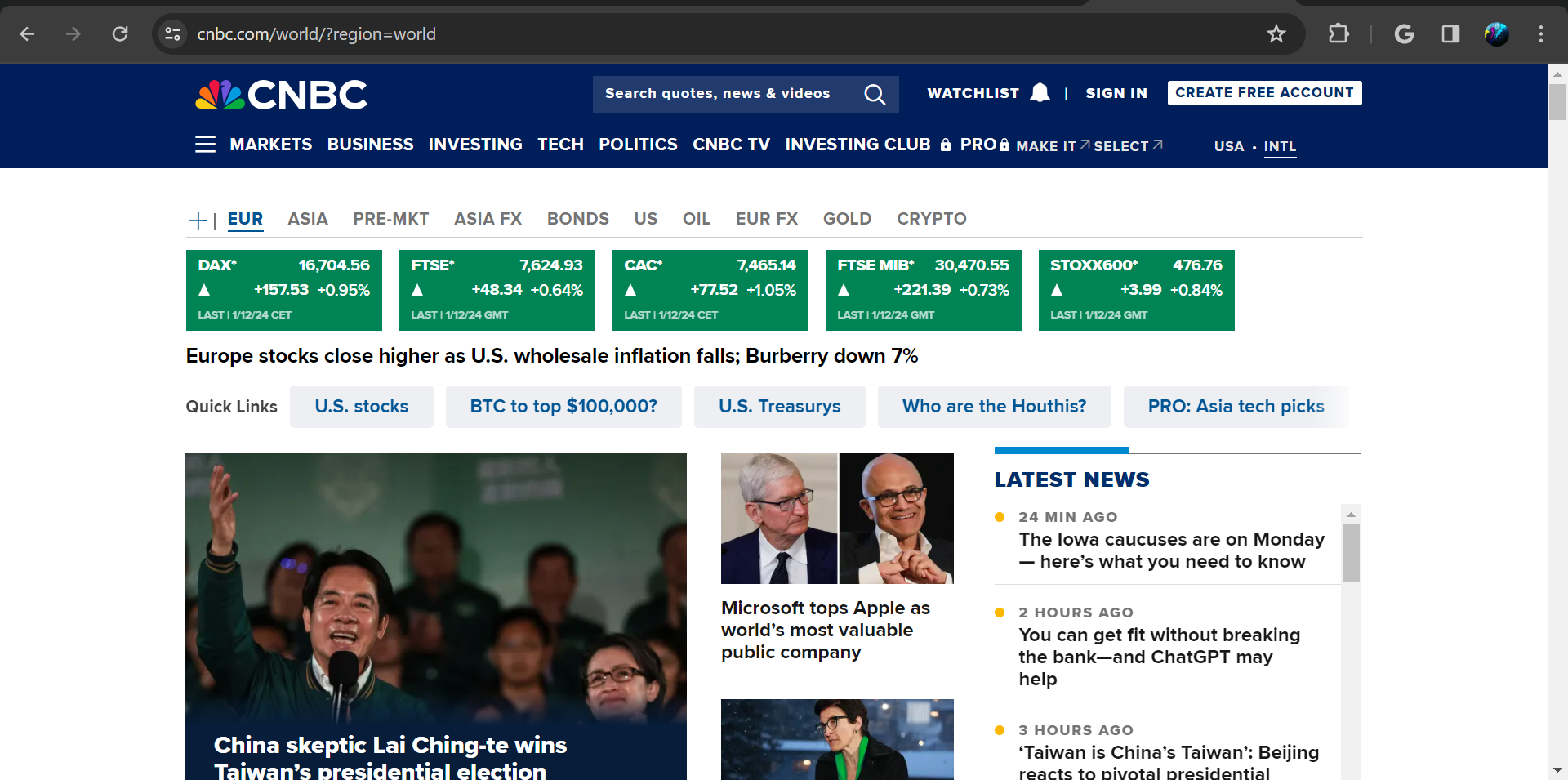
A screen shot of a computer

Description automatically generated

A computer screen shot of a program code

Description automatically generated

* A snapshot of the webpage from where the data is to be scraped.
  + The webpage loads a few fields dynamically. In this case the market banner is loaded using a JavaScript.
  + To capture the data from dynamically populated HTML elements we must use selenium with a web driver suited for the installed browser on the system.
  + While the latest news isn’t loaded dynamically hence, we can get the static snapshot of the HTML tree of the scrape that its section’s data.



* Scraping the raw data
  + This Python script is designed to scrape data from the CNBC website related to global markets. It begins by configuring logging for informative messages and defining file and directory paths.
  + The script utilizes the `pydantic\_settings` module to manage configuration settings, with a default base URL for CNBC. It then initializes a headless Chrome WebDriver using `chromedriver\_py` and configurable options.
  + The main functionality involves loading the CNBC webpage, waiting for the visibility of market card rows, and extracting HTML content for the market banner and the latest news panel using BeautifulSoup.
  + The extracted HTML is saved to a file in the "raw\_data" directory. As part of error handling, the script logs any exceptions that might occur during data fetching.
  + Finally, it prints the first ten lines of the saved HTML file to the standard output for quick verification.
  + Overall, this script demonstrates the use of web scraping tools like Selenium and BeautifulSoup, with logging providing insights into the different stages of the data retrieval process.

A black screen with white text

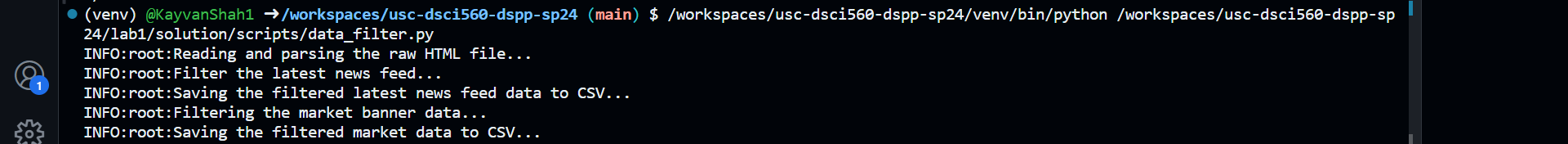
Description automatically generated

* Filtering the data to extract the fields of interest.

This Python script performs the following tasks:

1. Logging Setup: Configures logging to display INFO level messages.
2. Path and Directory Definitions: Defines the file and directory paths using the `Path` class.
3. HTML Parsing and Data Extraction:
   1. “read\_parse\_raw\_data”: Reads and parses an HTML file using BeautifulSoup.
   2. “NewsItem” and “MarketCard” Pydantic models are defined for structured data representation.
4. Data Filtering Functions:
   1. “filter\_latest\_news”: Extracts the latest news feed from the HTML, converts it into “NewsItem” Pydantic model instances, and appends them to a list.
   2. “filter\_market\_banner”: Filters market banner data from the HTML, converts it into “MarketCard” Pydantic model instances, and appends them to a list.
5. Main Execution:
   1. Reads and parses an HTML file (“web\_data.html”).
   2. Filters the latest news and market banner data using the defined functions.
   3. Logs the process at different stages (reading, filtering, and saving).
   4. Saves the filtered data into CSV files (“news\_data.csv” and “market\_data.csv”) in the "processed\_data" directory.

Overall, the script demonstrates the use of BeautifulSoup for HTML parsing, Pydantic for structured data representation, and Pandas for data manipulation. Logging is employed to provide information about the different stages of the process, making it easier to understand and debug the execution flow.



## Output Files

* Raw data output HTML file

A screenshot of a computer program

Description automatically generated

* Processed data CSV files
  + Market data
    - Has the following fields:
      * symbol – abbreviations for a stock ticker.
      * stock\_position – current/latest price of the stock.
      * change\_pts – most recent change in price observed.

A screenshot of a computer program

Description automatically generated

* + Latest news
    - Has the following fields:
      * timestamp – approximate time from the current time when the news was posted.
      * title – the headline of the news column
      * link – link to the official source of news on the website

A screen shot of a computer

Description automatically generated