

Github links: <https://github.com/tianqi0301/DSCI560>

1. Installation and Setup

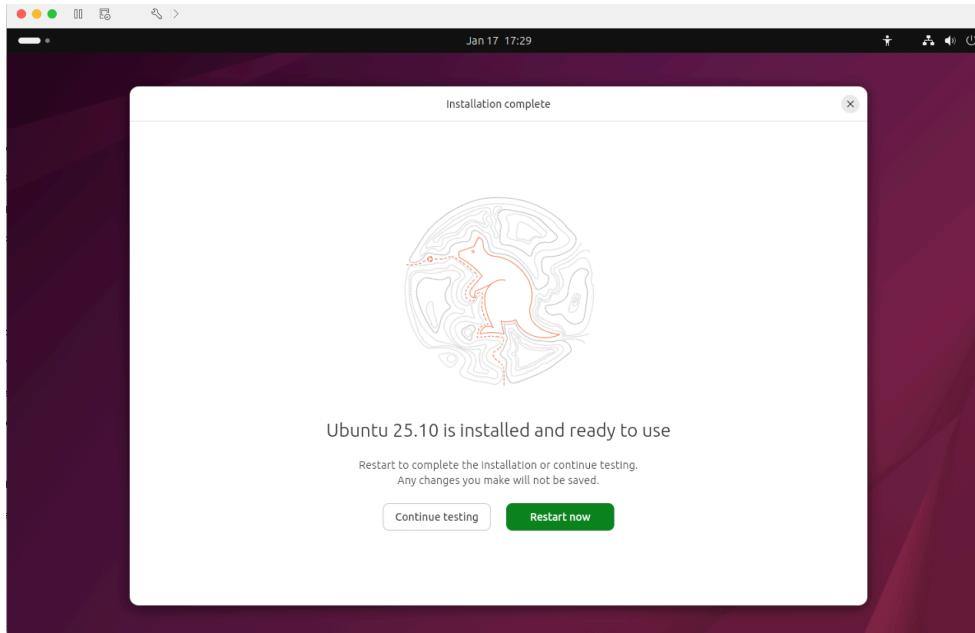
1.1. Register at Broadcom and then download and install VMware

I registered for a Broadcom account and downloaded VMware Fusion.

The screenshot shows the Broadcom My Profile interface. On the left, there's a sidebar with 'Basic Information' (Tianqi Qiu) and an 'Email' link. The main area has 'Personal Details' with fields for First Name (Tianqi), Last Name (Qiu), and Email (tianqi@usc.edu). There are 'Edit' and 'Delete Support Profile' buttons. Below this is a 'VIRTUAL MACHINES' section with a table header for 'Name' and 'Status'. The bottom of the screen shows a navigation bar with icons for home, search, and help.

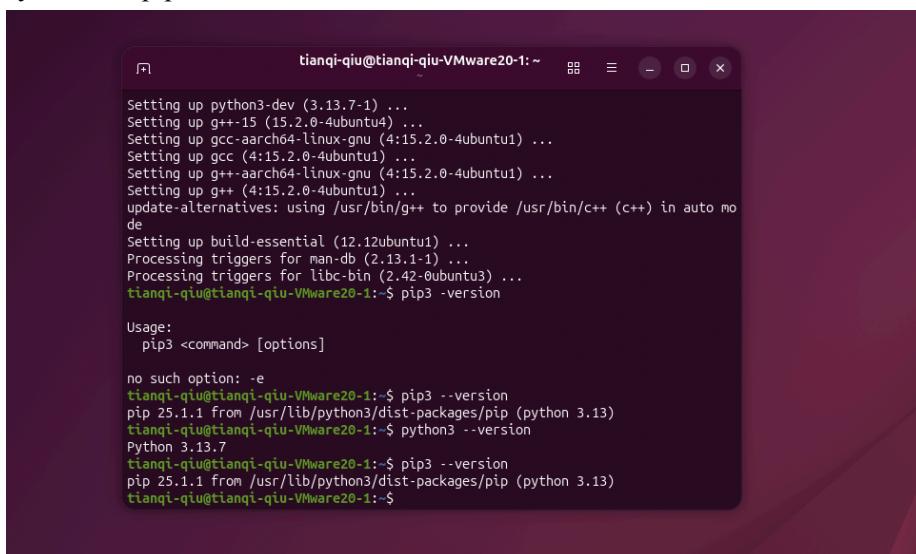
1.2. Download Ubuntu ISO Image

Ubuntu installed



1.3. Install Python on Linux

Python and pip3 installed



```
tianqi-qiu@tianqi-qiu-VirtualBox: ~
Setting up python3-dev (3.13.7-1) ...
Setting up g++-15 (15.2.0-4ubuntu4) ...
Setting up gcc-aarch64-linux-gnu (4:15.2.0-4ubuntu1) ...
Setting up gcc (4:15.2.0-4ubuntu1) ...
Setting up g++-aarch64-linux-gnu (4:15.2.0-4ubuntu1) ...
Setting up g++ (4:15.2.0-4ubuntu1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.12ubuntu1) ...
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for libc-bin (2.42-0ubuntu3) ...
tianqi-qiu@tianqi-qiu-VirtualBox: ~$ pip3 -version
Usage:
  pip3 <command> [options]
no such option: -e
tianqi-qiu@tianqi-qiu-VirtualBox: ~$ pip3 --version
pip 25.1.1 from /usr/lib/python3/dist-packages/pip (python 3.13)
tianqi-qiu@tianqi-qiu-VirtualBox: ~$ python3 --version
Python 3.13.7
tianqi-qiu@tianqi-qiu-VirtualBox: ~$ pip3 --version
pip 25.1.1 from /usr/lib/python3/dist-packages/pip (python 3.13)
tianqi-qiu@tianqi-qiu-VirtualBox: ~$
```

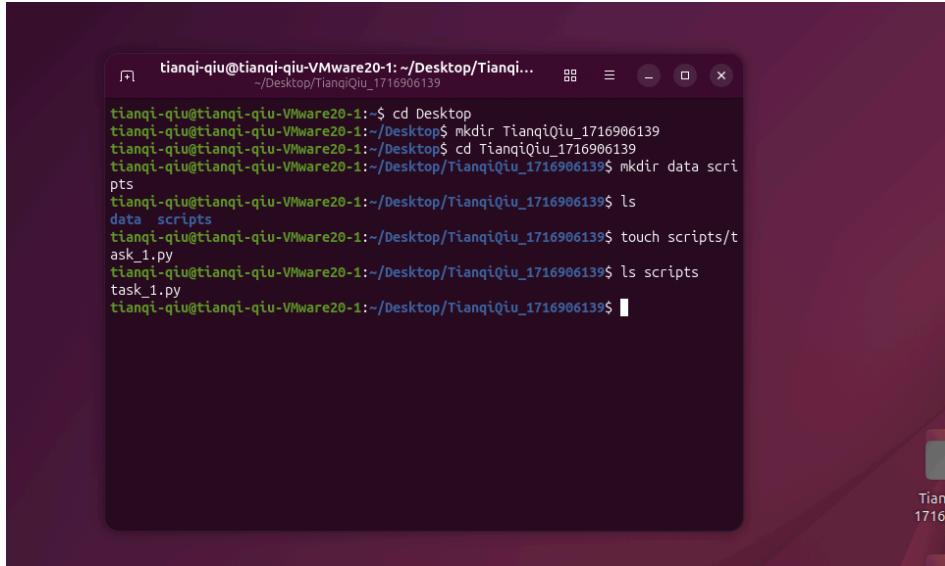
1.4. Tutorials

Go over python and linux tutorials

2. Get Familiar with Linux and Python

2.1. Playing around with Linux Terminal

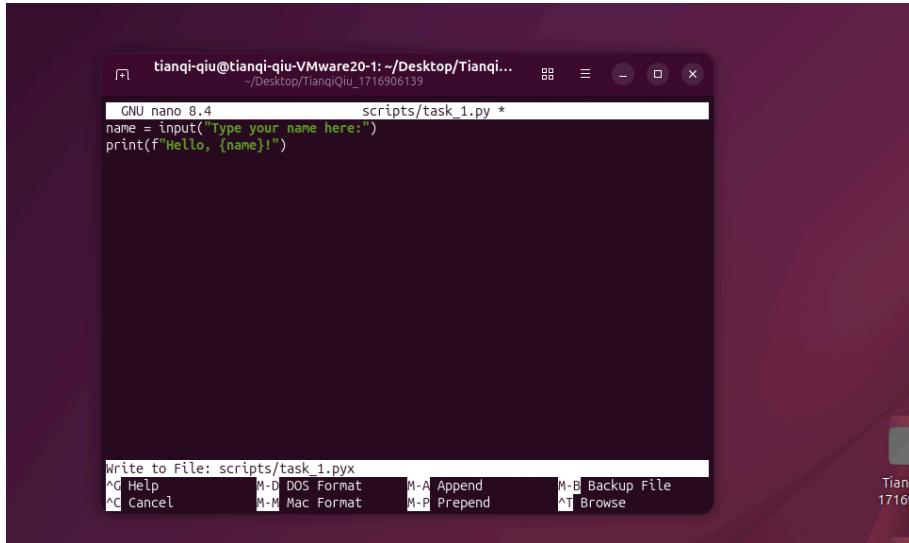
Folders and file created



```
tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop/Tianqi... /Desktop/TianqiQiu_1716906139
tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop$ mkdir TianqiQiu_1716906139
tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop$ cd TianqiQiu_1716906139
tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop/TianqiQiu_1716906139$ mkdir data scripts
tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop/TianqiQiu_1716906139$ ls
data scripts
tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop/TianqiQiu_1716906139$ touch scripts/task_1.py
tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop/TianqiQiu_1716906139$ ls scripts
task_1.py
tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop/TianqiQiu_1716906139$
```

2.2. A basic Python Script

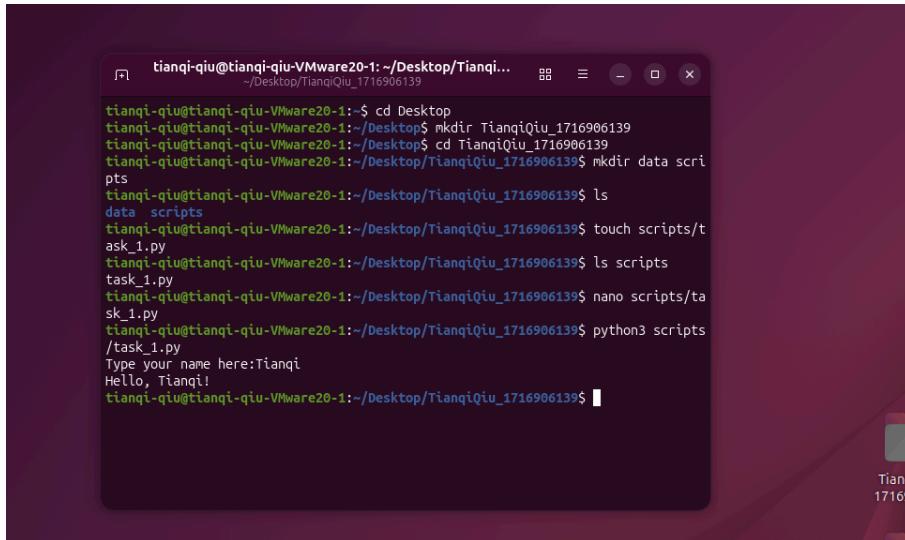
task1.py



```
GNU nano 8.4          scripts/task_1.py *
name = input("Type your name here:")
print(f"Hello, {name}!")

Write to File: scripts/task_1.pyx
^C Help           M-D DOS Format      M-A Append      M-B Backup File
^C Cancel         M-M Mac Format      M-P Prepend    ^T Browse
```

Hello, name!



A screenshot of a terminal window titled "tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop/Tianqi...". The terminal shows the user navigating to their Desktop, creating a directory named "TianqiQiu_1716906139", changing into it, creating a "data" subdirectory, and then creating a "scripts" directory. Inside "scripts", they touch a file named "task_1.py" and then list the contents of the directory. They then nano edit "task_1.py" and run it with python3. Finally, they type "Hello, Tianqi!" and press enter.

2.3. Python Web-scraping Task

scraper.py

```
import logging
import os
import shutil

from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

logging.basicConfig(level=logging.INFO)

BASE_URL = "https://www.cnbc.com/world/?region=world"

class Path:
    file_dir = os.path.dirname(__file__)
    root_dir = os.path.dirname(file_dir)
    data_dir = os.path.join(root_dir, "data")

def find_chromium_binary():
    return shutil.which("chromium-browser") or shutil.which("chromium")

def initialize_driver():
    chromium_path = find_chromium_binary()
    if not chromium_path:
        raise RuntimeError("Chromium browser not found")

    options = Options()
    options.binary_location = chromium_path
    options.add_argument("--headless=new")
    options.add_argument("--no-sandbox")
    options.add_argument("--disable-dev-shm-usage")

    service = Service("/usr/bin/chromedriver")
    return webdriver.Chrome(service=service, options=options)
```

```

def save_html_page(page: BeautifulSoup, save_to: str):
    market_banner = page.find("div", class_="MarketsBanner-marketData")
    latest_news = page.find("ul", class_="LatestNews-list")

    with open(save_to, "w", encoding="utf-8") as f:
        if market_banner:
            f.write(market_banner.prettify())
            f.write("\n\n")
        if latest_news:
            f.write(latest_news.prettify())


if __name__ == "__main__":
    driver = None
    try:
        logging.info("Initializing Chromium WebDriver...")
        driver = initialize_driver()

        logging.info("Opening CNBC webpage...")
        driver.get(BASE_URL)

        logging.info("Waiting for Market Cards to load...")
        WebDriverWait(driver, 60).until(
            EC.presence_of_element_located((By.CLASS_NAME, "MarketCard-row"))
        )

        soup = BeautifulSoup(driver.page_source, "html.parser")

        output_path = os.path.join(
            Path.data_dir, "raw_data", "web_data.html"
        )

        save_html_page(soup, output_path)
        logging.info("Saved HTML to %s", output_path)

        with open(output_path, "r", encoding="utf-8") as f:
            for _ in range(10):
                print(f.readline().strip())

    except Exception as e:
        logging.error("Unable to fetch data from page: %s", e)

    finally:
        if driver:
            driver.quit()

```

first 10 lines

```

(venv) tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop/TianqiQiu_1716906139/scripts$ head -n 10 ..../data/raw_data/web_data.html
<div class="MarketsBanner-marketData" id="market-data-scroll-container">
  <a class="MarketCard-container MarketCard-down" href="//www.cnbc.com/quotes/.STOXX">
    <div class="MarketCard-row">
      <span class="MarketCard-symbol">
        STOXX600*
      </span>
      <span class="MarketCard-stockPosition">
        614.38
      </span>
    </div>
(venv) tianqi-qiu@tianqi-qiu-VMware20-1:~/Desktop/TianqiQiu_1716906139/scripts$ 

```

2.4. Data Filtering Task

data_filter.py

```
GNU nano 8.4
import logging
import os

import pandas as pd
from bs4 import BeautifulSoup
from pydantic import BaseModel

logging.basicConfig(level=logging.INFO)

class Path:
    file_dir = os.path.dirname(__file__)
    root_dir = os.path.dirname(file_dir)

    data_dir = os.path.join(root_dir, "data")
    processed_data_dir = os.path.join(data_dir, "processed_data")

def read_parse_raw_data(path):
    logging.info("Reading and parsing the raw HTML file...")
    with open(path, "r", encoding="utf-8") as f:
        html_content = f.read()
    return BeautifulSoup(html_content, "html.parser")

class NewsItem(BaseModel):
    timestamp: str
    title: str
    link: str

class MarketCard(BaseModel):
    symbol: str
    stock_position: float
    change_pts: float

def filter_latest_news(page):
    logging.info("Filtering Latest News fields...")

    latest_news = page.find("ul", class_="LatestNews-list")
    latest_news = latest_news.find_all("li", class_="LatestNews-item")

    news_items = []
    for news_item in latest_news:
        headline = news_item.find("a", class_="LatestNews-headline")
        timestamp = news_item.find("time", class_="LatestNews-timestamp")

        if headline and timestamp:
            item = NewsItem(
                timestamp=timestamp.text.strip(),
                title=headline.get("title").strip(),
                link=headline.get("href").strip(),
            )
            news_items.append(item.model_dump())
    logging.info("Finished filtering Latest News.")
    return news_items
```

```
def filter_market_banner(page):
    logging.info("Filtering Market Banner fields...")

    market_banner = page.find("div", class_="MarketsBanner-marketData")
    market_cards = market_banner.find_all("a", class_="MarketCard-container")

    cards = []
    for card in market_cards:
        symbol = card.find("span", class_="MarketCard-symbol")
        stock_position = card.find("span", class_="MarketCard-stockPosition")
        change_pct = card.find("span", class_="MarketCard-changesPct")

        if symbol and stock_position and change_pct:
            item = MarketCard(
                symbol=symbol.text.strip(),
                stock_position=float(stock_position.text.strip().replace(",", "")),
                change_pts=float(
                    change_pct.text.strip()
                    .replace("%", "")
                    .replace(",", ""))
            ),
            cards.append(item.model_dump())
    logging.info("Finished filtering Market Banner.")
    return cards

if __name__ == "__main__":
    EXTRACTED_HTML_PATH = os.path.join(Path.data_dir, "raw_data", "web_data.html")
    soup = read_parse_raw_data(EXTRACTED_HTML_PATH)

    logging.info("Processing Latest News data...")
    latest_news = filter_latest_news(soup)
    logging.info("Storing Latest News data to CSV...")
    pd.DataFrame(latest_news).to_csv(
        os.path.join(Path.processed_data_dir, "news_data.csv"), index=False
    )
    logging.info("Latest News CSV created.")

    logging.info("Processing Market data...")
    market_banners = filter_market_banner(soup)
    logging.info("Storing Market data to CSV...")
    pd.DataFrame(market_banners).to_csv(
        os.path.join(Path.processed_data_dir, "market_data.csv"), index=False
    )
    logging.info("Market data CSV created.")

    logging.info("Data filtering task completed successfully.")
```

Run successfully with appropriate messages to the console

```
(venv) tianqi-qiu@tianqi-qiu-VHware20-1:~/Desktop/TianqiQiu_1716906139/scripts$ python data_filter.py
INFO:root:Reading and parsing the raw HTML file...
INFO:root:Processing Latest News data...
INFO:root:Filtering Latest News fields...
INFO:root:Finished filtering Latest News.
INFO:root:Storing Latest News data to CSV...
INFO:root:Latest News CSV created.
INFO:root:Processing Market data...
INFO:root:Filtering Market Banner fields...
INFO:root:Finished filtering Market Banner.
INFO:root:Storing Market data to CSV...
INFO:root:Market data CSV created.
INFO:root:Data filtering task completed successfully.
```

First 5 lines of filtered news data and market data

```
(venv) tianqi-qtu@tianqi-qiu-VMware20-1:~/Desktop/tianqi_qiu_1716906139/scripts$ head -n 5 ./data/processed_data/news_data.csv
timestamp,title,link
10 Hours Ago,Week in review: Stocks battled a flood of news and we booked some profits,https://www.cnbc.com/2026/01/17/week-in-review-stocks-battled-a-flood-of-news-and-we-booked-some-profits.html
10 Hours Ago,Trump threatens to sue JPMorgan Chase for 'debunking' him,https://www.cnbc.com/2026/01/17/trump-jpmorgan-chase-debunking.html
13 Hours Ago,Trump: NATO members to face tariffs up to 25% until a Greenland deal is struck,https://www.cnbc.com/2026/01/17/trump-greenland-tariffs-nato.html
14 Hours Ago,Led by Texas, states race to prove they can put bitcoin on public balance sheet',https://www.cnbc.com/2026/01/17/texas-us-states-budgets-bitcoin-crypto-strategic-reserve.html
(venv) tianqi-qtu@tianqi-qiu-VMware20-1:~/Desktop/tianqi_qiu_1716906139/scripts$ head -n 5 ./data/processed_data/market_data.csv
symbol,open,close,high,low,change_pts
STOXX600*,614.38,-0.03
DAX*,75297.13,-0.22
FTSE*,18235.29,-0.04
CAC*,8258.94,-0.65
```

3. Team Formation

I formed a team with Herun Kan and Shida Yan.