CRYPTOCURRENCY PRICE TRACKER

# INTRODUCTION:

In the ever-changing world of cryptocurrency, real-time price tracking is essential for investors, analysts, and enthusiasts. This mini project focuses on building a Cryptocurrency Price Tracker using Python, Selenium, and Excel. The tool automatically scrapes the latest market data from CoinMarketCap, including coin names, prices, daily percentage changes, and market capitalization, and stores it in an Excel file for future analysis. The project demonstrates practical skills in web scraping, data storage in spreadsheet format, and automation, making it a valuable exercise for those interested in data-driven applications.

# .Prerequisites:

## Software:

* + Python 3.8+
  + Google Chrome browser
  + Microsoft Excel (or compatible spreadsheet software)

## Python Libraries:

pip install selenium pandas openpyxl webdriver-manager

## Basic Knowledge:

* + Python programming
  + Basics of Excel file handling using Pandas

# Step-by-Step Instructions:

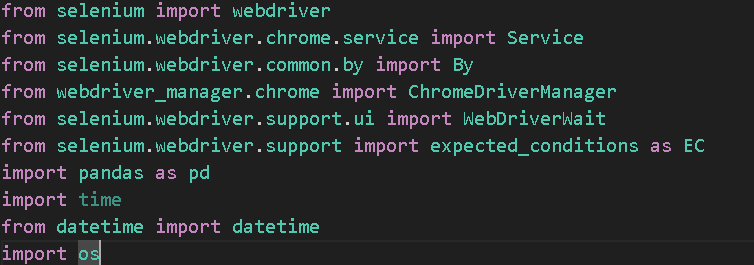
* **Step 1: Install required libraries:**

**pip install selenium pandas openpyxl webdriver-manager**

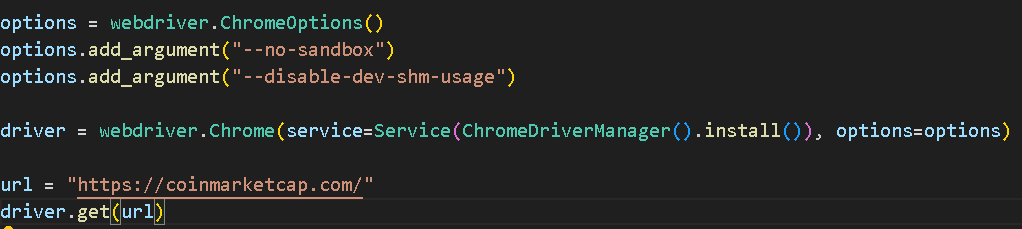
* **Step 2 — Create the script file:**

1. Inside your project folder, create a file named crypto\_tracker.py.
2. The script will perform three main tasks: (a) open the browser, (b) scrape the table rows, and (c) save results into an Excel file.

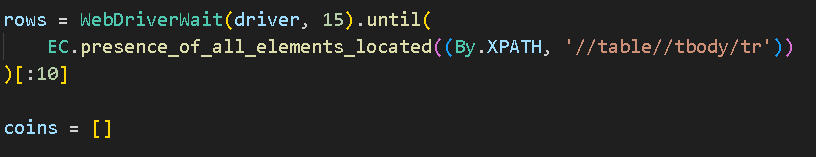
**Step 4: Implementing the Script:**

****

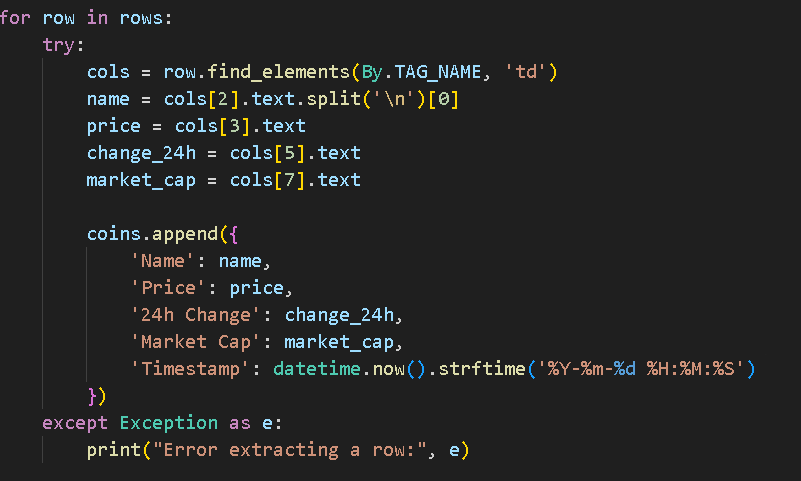
* **from selenium import webdriver** – Lets you control a web browser with Selenium.
* **from selenium.webdriver.chrome.service import Service** – Manages the ChromeDriver service that talks to Chrome.
* **from selenium.webdriver.common.by import By** – Provides ways to locate elements on a page (by XPATH, CSS, etc.).
* **from webdriver\_manager.chrome import ChromeDriverManager** – Automatically downloads and manages the correct ChromeDriver version.
* **from selenium.webdriver.support.ui import WebDriverWait** – Allows waiting until certain conditions (like elements loading) are met.
* **from selenium.webdriver.support import expected\_conditions as EC** – Predefined conditions to wait for, such as element visibility.
* **import pandas as pd** – For working with tabular data and saving to CSV/Excel.
* **import time** – Lets you add delays in execution.
* **from datetime import datetime** – To get the current date and time.
* **import os** – For handling file paths and checking if files exist.



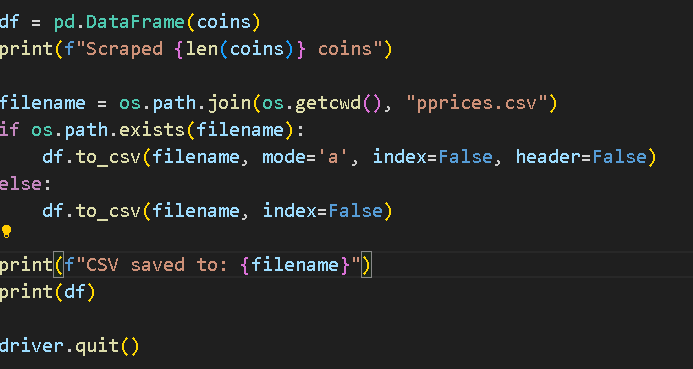
* **options = webdriver.ChromeOptions()** – Create Chrome settings.
* **options.add\_argument("--no-sandbox")** – Disable sandbox for compatibility.
* **options.add\_argument("--disable-dev-shm-usage")** – Avoid shared memory issues.
* **driver = webdriver.Chrome(...)** – Launch Chrome with these settings and auto-install driver.
* **url = "https://coinmarketcap.com/"** – Store target website link.
* **driver.get(url)** – Open the website in Chrome.



* **WebDriverWait(driver, 15)** – Waits up to **15 seconds** for a condition to be met.
* **EC.presence\_of\_all\_elements\_located(...)** – The condition: all table rows (<tr>) inside <tbody> of <table> must be present in the DOM.
* **(By.XPATH, '//table//tbody/tr')** – Locator: finds all cryptocurrency rows in the table.
* **[:10]** – Takes only the first **10 rows** (top 10 coins).
* **coins = []** – Creates an empty list to store the scraped coin data



* **for row in rows:** – Loops through each scraped table row.
* **cols = row.find\_elements(By.TAG\_NAME, 'td')** – Finds all table cells (<td>) in the row.
* **name = cols[2].text.split('\n')[0]** – Gets the coin name from the 3rd cell and removes extra text after a newline.
* **price = cols[3].text** – Gets the price from the 4th cell.
* **change\_24h = cols[5].text** – Gets the 24-hour change from the 6th cell.
* **market\_cap = cols[7].text** – Gets the market cap from the 8th cell.
* **coins.append({...})** – Adds this coin’s data as a dictionary to the coins list, including the current timestamp.
* **except Exception as e:** – If something goes wrong with a row, print the error instead of stopping the whole script.



* **df = pd.DataFrame(coins)** – Converts the coins list of dictionaries into a Pandas DataFrame for easier handling and saving.
* **print(f"Scraped {len(coins)} coins")** – Displays how many coins were scraped.
* **filename = os.path.join(os.getcwd(), "pprices.csv")** – Creates a file path for pprices.csv in the current working directory.
* **if os.path.exists(filename):** – Checks if the CSV already exists.
  + **If yes:** Append new rows (mode='a'), without writing the header again (header=False).
  + **If no:** Create the file and write data with headers.
* **print(f"CSV saved to: {filename}")** – Confirms where the file was saved.
* **print(df)** – Displays the DataFrame in the terminal.
* **driver.quit()** – Closes the browser and ends the Selenium session.

**FULL SCRIPT:**

from selenium import webdriver

from selenium.webdriver.chrome.service import Service from selenium.webdriver.common.by import By

from webdriver\_manager.chrome import ChromeDriverManager from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC import pandas as pd

import time

from datetime import datetime import os

options = webdriver.ChromeOptions() # options.add\_argument("--headless") options.add\_argument("--no-sandbox")

options.add\_argument("--disable-dev-shm-usage")

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)

url = "https://coinmarketcap.com/" driver.get(url)

# Wait for the table rows to appear

rows = WebDriverWait(driver, 15).until( EC.presence\_of\_all\_elements\_located((By.XPATH, '//table//tbody/tr'))

)[:10]

coins = []

for row in rows: try:

cols = row.find\_elements(By.TAG\_NAME, 'td') name = cols[2].text.split('\n')[0]

price = cols[3].text change\_24h = cols[5].text market\_cap = cols[7].text

coins.append({ 'Name': name, 'Price': price,

'24h Change': change\_24h, 'Market Cap': market\_cap,

'Timestamp': datetime.now().strftime('%Y-%m-%d %H:%M:%S')

})

except Exception as e:

print("Error extracting a row:", e)

df = pd.DataFrame(coins) print(f"Scraped {len(coins)} coins")

filename = os.path.join(os.getcwd(), "prices.csv") if os.path.exists(filename):

df.to\_csv(filename, mode='a', index=False, header=False) else:

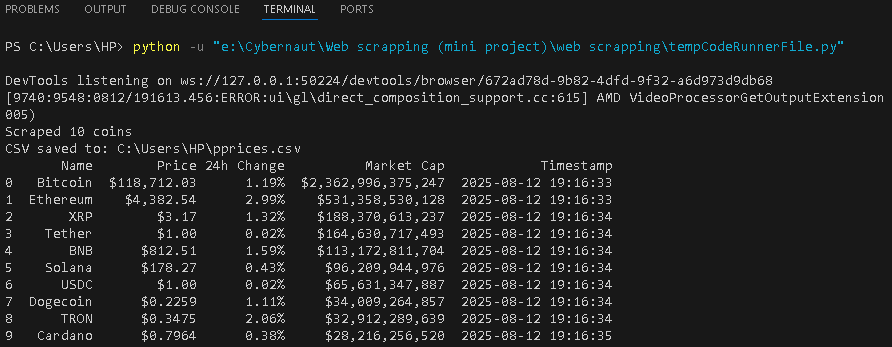
df.to\_csv(filename, index=False)

print(f"CSV saved to: {filename}") print(df)

driver.quit()

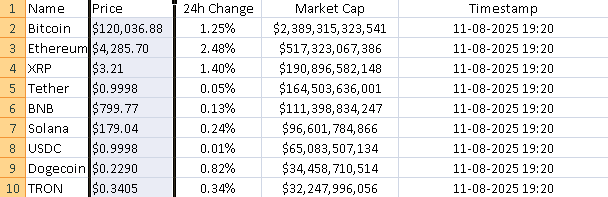
**Step 3: Running the Script**

python crypto\_tracker.py



**OUTPUT:**

* **Terminal:** Displays scraped coin data.
* **prices.csv:** Created/updated with new entries.



# Troubleshooting Guide:

|  |  |  |
| --- | --- | --- |
| **Issue** | **Possible Cause** | **Solution** |
| No data scraped | Table not loaded in time | Increase WebDriverWait  time. |
| CSV not created | Wrong working directory | Use os.getcwd() to confirm path. |
| ChromeDriver error | Version mismatch | Run pip install -- upgrade webdriver- manager |

**End of Document:**

This concludes the **Web Scraping Mini Project – Cryptocurrency Price Tracker**

documentation.

By following the steps outlined above, you can:

* **Scrape** live cryptocurrency data from CoinMarketCap
* **Store** it in CSV/Excel format for analysis
* **Automate** the process for regular updates You can extend the project by:
* Adding filters for specific coins or price ranges
* Integrating the data into a dashboard (Excel, Power BI, Tableau)
* Scheduling automated runs with Task Scheduler or cron jobs

**Project completed by:**

**Syed Ismail N**