

# Web Scraping Report

**Shreya Maher**  
College of Engineering  
Northeastern University  
Boston, MA  
[maher.sh@northeastern.edu](mailto:maher.sh@northeastern.edu)

## Abstract

The following involves scraping data from the website Coin Market Cap related to cryptocurrencies.

CoinMarketCap is a widely used website in the cryptocurrency domain that provides a comprehensive platform for tracking and analyzing various aspects of the cryptocurrency market. The website offers real-time data on a vast array of cryptocurrencies, including their names, current market prices, market capitalization (total value), trading volumes, circulating supply (amount in circulation), and historical price trends. Additionally, CoinMarketCap aggregates data on exchanges, trading pairs, market trends, and other relevant metrics.

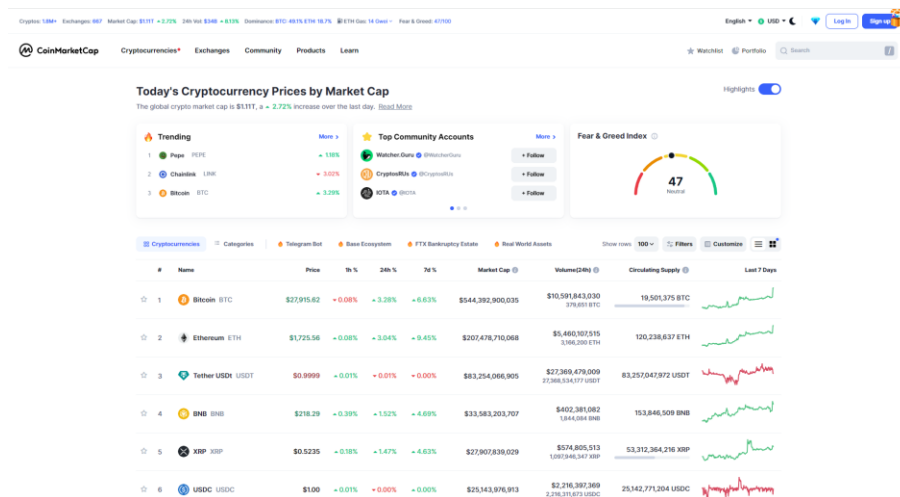


Figure 1: Website used to scrape data

## 1. Introduction:

### 1.1 Problem Definition:

The objective is to extract specific cryptocurrency data from the CoinMarketCap website, including essential metrics and associated images. The scope includes retrieving data for the top cryptocurrencies.

## 1.2 Motivation/Objective:

The motivation behind selecting this project was to strike a balance between simplicity and usefulness. Cryptocurrency analysis and market trends have gained immense popularity due to the rise of digital currencies. This project aimed to fetch data from CoinMarketCap, a reputable source for cryptocurrency information, and structure it for potential analysis. While the task isn't overly complex, the collected data can be leveraged for studying trends, market behavior, and conducting rudimentary analysis, providing a valuable foundation for further exploration into the cryptocurrency domain. This project sets the stage for understanding how to scrape real-time data, which is essential for anyone looking to delve into the field of cryptocurrency analysis and research.

## 2. Methodology:

### 2.1 Environment Setup:

- 2.1.1 Python environment with necessary libraries like Pandas, BeautifulSoup, requests, and IPython.

### 2.2 Tools Used:

- 2.2.1 Google Colab

## 3. Solution:

### 3.1 Handling of HTML and CSS:

- 3.1.1 Having utilized BeautifulSoup, I parsed HTML, extracted relevant information from specific HTML tags, and retained image tags. This facilitated data scraping and enabled the straightforward display of images and text using HTML.
- 3.1.2 Used concepts of 'tbody' and 'tr' to navigate through tabular data

### 3.2 Pagination or Dynamic Content:

- 3.2.1 Focusing on the top entries, pagination isn't necessary in this case. The script targets a static table of top cryptocurrencies.

### 3.3 Error Handling and Rate Limiting:

- 3.3.1 To mitigate potential errors, the code is restricted to scraping only the initial records and limiting the number of columns being extracted.

### 3.4 Data Storage and Format:

- 3.4.1 The extracted data is stored in a Pandas DataFrame, allowing for easy manipulation and analysis.

## 4. Challenges & Outlooks:

### 4.1 Dealing with Roadblocks:

- 4.1.1 The encountered challenges included handling HTML structure changes, understanding and navigating the HTML tree structure using elements such as tbody and tr, as well as scraping images and comprehending how to store and display them. These obstacles were overcome through careful analysis, studying relevant concepts, and adjusting the parsing logic accordingly.

## 4.2 Future Improvements:

- 4.2.1 To enhance the solution, we should implement additional error handling for improved robustness, pagination for comprehensive data retrieval, and the integration of a database for efficient data storage. This should involve incorporating a scraper capable of navigating through multiple pages on the website and extracting a larger volume of data.

## 5. Steps Followed:

- Import necessary libraries: Import the required libraries, including pandas, BeautifulSoup, requests, and IPython.display.
- Specify the URL: Define the URL of the website you want to scrape (in this case, CoinMarketCap).
- Send a request and parse HTML: Use requests to retrieve the HTML content of the specified URL and parse it using BeautifulSoup.
- Extract table data: Find the table body (tbody) and extract all table rows (trs).
- Iterate through rows: Iterate through the first 10 rows of the table.
- Extract data from each row: Extract specific data elements (name, price, market cap, volume, supply, and image source) from each row.
- Organize data: Store the extracted data for each row in a frame.
- Create an HTML image tag: Create an HTML <img> tag for each image URL to display the images in the DataFrame.
- Create a DataFrame: Create a DataFrame using pandas, using the collected data and specifying column names.
- Display the DataFrame with images: Use IPython.display to display the DataFrame with images using HTML.
- The resulting DataFrame will include columns for Name, Price, Market Cap, Volume, Supply, and Image, with data extracted from the website, and images displayed using HTML.

## 6. Output:

**Coin Market Data**  
This table displays data for the top 10 cryptocurrencies from CoinMarketCap.











	Name	Price	Volume	Supply	Chart
0	Bitcoin	\$27,837.69	\$10,370,977,304	19,501,375 BTC	
1	Ethereum	\$1,721.35	\$5,385,216,444	120,238,637 ETH	
2	Tether USDt	\$1.00	\$27,069,847,686	83,257,047,972 USDT	
3	BNB	\$217.53	\$396,838,891	153,846,509 BNB	
4	XRP	\$0.5221	\$568,286,997	53,312,364,216 XRP	
5	USDC	\$1.00	\$2,188,563,080	25,142,771,204 USDC	
6	Solana	\$23.70	\$655,346,744	412,985,815 SOL	
7	Cardano	\$0.2641	\$170,432,287	35,138,461,675 ADA	
8	Dogecoin	\$0.06298	\$168,594,621	141,239,166,384 DOGE	
9	TRON	\$0.0899	\$150,167,190	89,060,046,822 TRX	

Figure 2: Output of the data

- 117 **7. Learnings**  
118 a) Web Scraping Basics: How to retrieve and parse HTML content from a website  
119 using Python libraries like requests and BeautifulSoup.  
120 b) HTML Structure Navigation: Navigating and extracting specific data from HTML  
121 using BeautifulSoup, understanding the structure of HTML elements.  
122 c) Data Extraction and Processing: Extracting relevant data from HTML elements,  
123 organizing and processing the data for further use.  
124 d) Table Data Extraction: Extracting tabular data from HTML tables and formatting it  
125 for analysis.  
126 e) Image Scraping: Scraping image URLs from HTML and using HTML tags to  
127 display images.  
128 f) Data Presentation: Displaying scraped data, including images, in a formatted manner  
129 for effective visualization and comprehension.  
130

- 131 **8. References**  
132 • [Amazon Scraping](#)  
133 • [Data Scraping Series By Tim](#)  
134 • [Navigating Through HTML Tree](#)  
135 • [Data Scraping: Beautiful Soup](#)  
136  
137