

[Open in app](#)[Get started](#)

You have **1** free member-only story left this month. [Sign up for Medium and get an extra one](#)



Tommy Chan

[Follow](#)

Jul 1, 2020 · 6 min read ★ · [Listen](#)



Save



# Web Scraping Crypto Prices With Python

This is the most beautiful soup



Illustrated by Bryan Feng (Author)

Recently I was looking for some data from a website to do my own analysis on



[Open in app](#)[Get started](#)

As always, if you want to do good analysis, you need good data. For cryptocurrencies, look no further than [CoinMarketCap](#).

## You don't always need APIs

Traditionally we'd like to use an API to fetch data, but we all know that APIs are designed with limitations for the free user. The free plan of most APIs are just there to tease you enough for you to pull out your wallet and spend money on a paid plan. Personally, I'm cheap so I'm looking to stay in the free realm.

What I first did was go through CoinMarketCap's website and look for their API and [pricing plans](#) to see if the free plan could get the job done.

This is what I see:

The screenshot shows the 'BASIC' plan for 'Basic personal use'. The word 'Free' is prominently displayed in the center, with 'No subscription required' underneath it. Below this is a button that says 'GET FREE API KEY'. Further down, the plan details are listed: '9 market data endpoints' with a help icon, '10K call credits /mo', and 'No historical data' which is highlighted with a yellow background. At the bottom, it says 'Personal use'.

**BASIC**  
Basic personal use

**Free**  
No subscription required

GET FREE API KEY

9 market data endpoints ?  
10K call credits /mo  
No historical data  
Personal use



[Open in app](#)[Get started](#)

The free API is not going to give us what we want but don't fret, we're going to use a very VERY handy workaround. Assuming you have python installed on your machine, here's what we're going to do:

```
pip install beautifulsoup4
```

We're installing BeautifulSoup, a popular tool used to web scrape via python. It's a very easy and simple to use tool and I'll show you exactly how to use it to do something that CoinMarketCap's free API won't do for us.

Though we're making a Beautiful Soup, we're going to need some other key ingredients, and I'll show you why as we go along.

```
from bs4 import BeautifulSoup
import requests
import pandas as pd
import json
import time
```

ingredients to our soup

Now we're going to want to play around and do some exploration. We'll do a simple get request of the main page into the BeautifulSoup object.

```
cmc = requests.get('https://coinmarketcap.com/')
soup = BeautifulSoup(cmc.content, 'html.parser')
```




[Open in app](#)
[Get started](#)


```
print(soup.title)

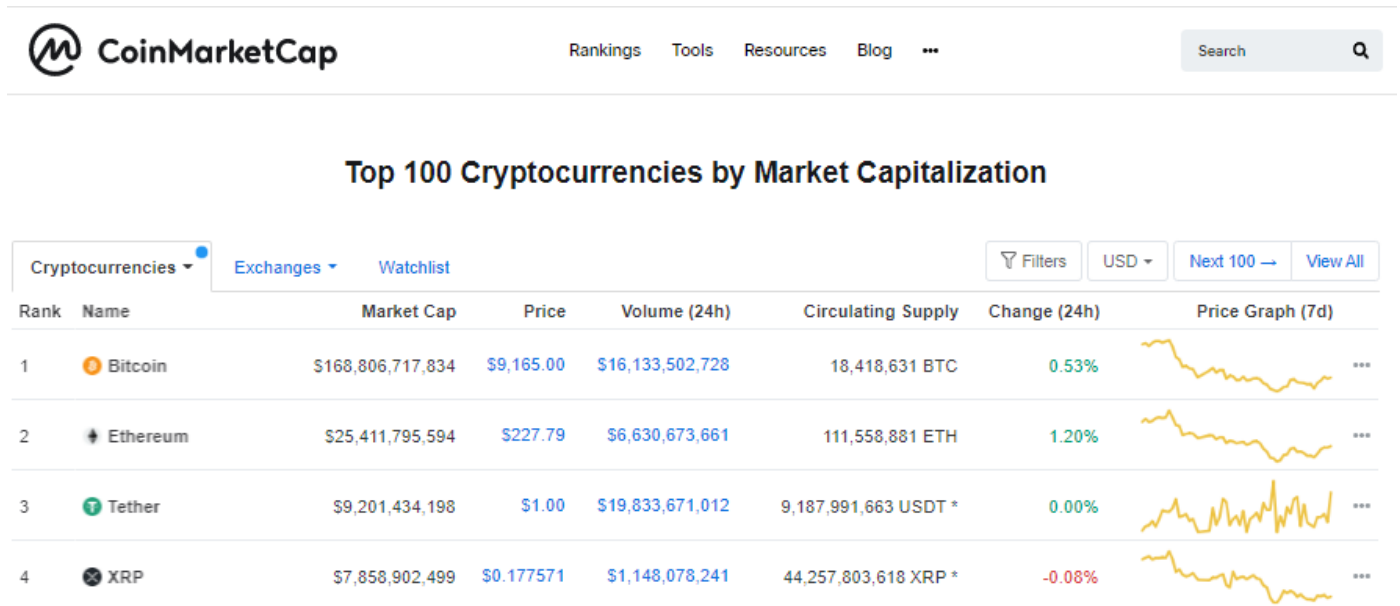
#<title class="next-head">Cryptocurrency Market Capitalizations | CoinMarketCap</title>
```

## Fetching Data

Now that we know how to get data from a website, it's time to identify what we need and how to extract it. My intention is to get historical data for each coin. I first need to understand where exactly this is in the website and object so I can code accordingly.

## Scouting

We'll start at the main page. I get a table with a list of coins, their market cap, price, volume, supply, 24h change, and a price graph. I instantly know that I'm going to want to get the list of coins from here.



The screenshot shows the CoinMarketCap website interface. At the top, there's a navigation bar with the CoinMarketCap logo, links for Rankings, Tools, Resources, and Blog, and a search bar. Below the navigation bar, the main heading is "Top 100 Cryptocurrencies by Market Capitalization". Under this heading, there are tabs for "Cryptocurrencies" (selected), "Exchanges", and "Watchlist". To the right of these tabs are buttons for "Filters", "USD" (currency selector), "Next 100" (pagination), and "View All". The table below lists the top 4 cryptocurrencies: Bitcoin, Ethereum, Tether, and XRP. Each row includes columns for Rank, Name, Market Cap, Price, Volume (24h), Circulating Supply, Change (24h), and Price Graph (7d).

Rank	Name	Market Cap	Price	Volume (24h)	Circulating Supply	Change (24h)	Price Graph (7d)
1	Bitcoin	\$168,806,717,834	\$9,165.00	\$16,133,502,728	18,418,631 BTC	0.53%	
2	Ethereum	\$25,411,795,594	\$227.79	\$6,630,673,661	111,558,881 ETH	1.20%	
3	Tether	\$9,201,434,198	\$1.00	\$19,833,671,012	9,187,991,663 USDT *	0.00%	
4	XRP	\$7,858,902,499	\$0.177571	\$1,148,078,241	44,257,803,618 XRP *	-0.08%	

I'll look for historical data by clicking into Bitcoin. In bitcoin's page I see a tab for historical data with a table of what I like. I understand what I want now and we're done with scouting and moving on to building out the scraping process.





Open in app

Get started

CoinMarketCap

Rankings Tools Resources Blog ...

Search

Bitcoin (BTC)

\$9,122.16 USD (0.10%)

1.00000000 BTC (0.00%)

Buy

Exchange

Gamble

Spin to Win

SPONSORED

Share

Watch

Rank 1

Website

Explorer ( 2 3 4 5 )

Message Board

Source Code

Technical Documentation

Coin Mineable PoW SHA-256 Store of Value

Privacy State channels

Market Cap	Volume (24h)	Circulating Supply	Max Supply
\$168,018,499,704 USD 18,418,712 BTC	\$16,191,061,525 USD 1,774,915 BTC	18,418,712 BTC	21,000,000 BTC

Charts

Market Pairs

Social

Tools

Historical Data

Ratings

On-Chain Analysis

News

## Historical data for Bitcoin

Currency in USD

May 30, 2020

Jun 30, 2020

Date	Open*	High	Low	Close**	Volume	Market Cap
Jun 29, 2020	9,140.03	9,237.57	9,041.88	9,190.85	16,460,547,078	169,280,659,246

## Parsing the website

We want to understand how the website is set up, so we're going to print the soup object from above, we can do a simple `print(soup)` but that's going to make everything mashed together and hard to identify so instead we'll do this:

```
print(soup.prettify())
```

This returns an organized version of CoinMarketCap and I'm able to eyeball and locate the data that I want.

&lt;/div&gt;



[Open in app](#)[Get started](#)

I know that bitcoin's page is formatted as

<https://coinmarketcap.com/currencies/bitcoin/>

This matches the key '**slug**' so at a minimum, I know that I'll need to save slug. I'll also save **id** as well just in case. I also see that this is JSON data within a script tag so we'll need to isolate it.

```
data = soup.find('script', id="__NEXT_DATA__", type="application/json")
coins = {}

#using data.contents[0] to remove script tags
coin_data = json.loads(data.contents[0])
listings = coin_data['props']['initialState']['cryptocurrency']['listingLatest']['data']

for i in listings:
    coins[str(i['id'])] = i['slug']
```

o( ^ III ^ ) ⊃ Hehehe...

Viola, we now have a dictionary of coin IDs and slugs we can use to scrape historical data.

## Fetching Historical Data

Now that we have our list of coin slugs, we can go ahead and drill into the historical data table for each page.

We'll need the historical data page, so I'll again refer back to bitcoin as a template for what I'm going to do.





Open in app

Get started

**Bitcoin** (BTC)

\$9,131.01 USD (-0.43%)

1.00000000 BTC (0.00%)

Buy ▾

Exchange ▾

Gamble ▾

Spin to Win ▾

SPONSORED

Share

★ Watch

Rank 1

Website

Explorer ( 2 3 4 5 )

Message Board

Source Code

Technical Documentation

Coin Mineable PoW SHA-256 Store of Value

Privacy State channels

Market Cap	Volume (24h)	Circulating Supply	Max Supply
\$168,187,637,878 USD 18,419,381 BTC	\$15,862,233,469 USD 1,737,182 BTC	18,419,381 BTC	21,000,000 BTC

Charts

Market Pairs

Social

Tools

Historical Data

Ratings

On-Chain Analysis

News

## Historical data for Bitcoin

Currency in USD

Jan 01, 2020

Jul 01, 2020

Date	Open*	High	Low	Close**	Volume	Market Cap
Jun 30, 2020	9,185.58	9,217.84	9,084.84	9,137.99	15,735,797,744	168,315,606,321
Jun 29, 2020	9,140.03	9,237.57	9,041.88	9,190.85	16,460,547,078	169,280,659,246
Jun 28, 2020	9,048.46	9,197.55	8,975.53	9,143.58	14,560,870,760	168,401,806,137

This is our table and the link we want to reference below.

<https://coinmarketcap.com/currencies/bitcoin/historical-data/?start=20200101&end=20200630>

As you can see, the template for viewing these tables for any coin by date goes:

`https://coinmarketcap.com/currencies/[slug]/historical-data/?start=[YYYYMMDD]&end=[YYYYMMDD]`

We'll code accordingly below to take a look at the underlying JSON object within the data.





Open in app

Get started

```

for i in coins:
    page = requests.get(f'https://coinmarketcap.com/currencies/{coins[i]}/historical-data/?start=20200101&end=20200630')
    soup = BeautifulSoup(page.content, 'html.parser')
    data = soup.find('script', id="__NEXT_DATA__", type="application/json")
    historical_data = json.loads(data.contents[0])

```

O(n\_n)O

In my example let's say I want to understand the volume and market cap each day. I find them nested accordingly:

```

quotes = historical_data['props']['initialState']['cryptocurrency']['ohlcvHistorical'][i]['quotes']

```

EDIT:

I also want to log the info:

```

info = historical_data['props']['initialState']['cryptocurrency']
      ['ohlcvHistorical'][i]

```

## Putting everything together with pandas

We know how to get the data from the web, now it's time to actually store it for future analysis. To get this done, we'll use the famous pandas package and store this in a data frame.

I'll first create a template of the things I want to store:





[Open in app](#)[Get started](#)

```
market_cap = []  
volume = []  
timestamp = []  
name = []  
symbol = []  
slug = []
```

Next, we'll be sending data from BeautifulSoup to these arrays and then into our pandas data frame.

```
for j in quotes:  
    market_cap.append(j['quote']['USD']['market_cap'])  
    volume.append(j['quote']['USD']['volume'])  
    timestamp.append(j['quote']['USD']['timestamp'])  
    name.append(info['name'])  
    symbol.append(info['symbol'])  
    slug.append(coins[i])
```

```
df = pd.DataFrame(columns = ['marketcap', 'volume', 'timestamp', 'name', 'symbol', 'slug'])  
  
df['marketcap'] = market_cap  
  
df['volume'] = volume  
  
df['timestamp'] = timestamp  
  
df['name'] = name  
  
df['symbol'] = symbol
```



[Open in app](#)[Get started](#)

Last but not least we want to save this to a csv file so we can bring our data anywhere.

```
df.to_csv('criptoes.csv', index = False)
```

Hooray, you've now learned how to pull historical crypto data from coin market cap, the main website everyone goes to for up to date crypto prices.

For more tutorials like this, please follow me on Medium at [Bryan Feng](#).

If you would like to see what I will do with this data in my own analysis, please follow me on substack at <https://bryology.substack.com/>

[About](#) [Help](#) [Terms](#) [Privacy](#)

Get the Medium app

