REST APIs

We are goint to use CoinGecko API to create a candlestick graphs for Bitcoin. We will use the API to get the price data for 30 days with 24 observation per day, 1 per hour. We will find the max, min, open, and close price per day meaning we will have 30 candlesticks and use that to generate the candlestick graph.

In [1]:
```
"""
!pip install pycoingecko
!pip install plotly
!pip install mplfinance
"""
```

```
Collecting pycoingecko
  Downloading pycoingecko-2.2.0-py3-none-any.whl (8.3 kB)
Requirement already satisfied: requests in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (from pycoi
ngecko) (2.25.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/gennaro/opt/anaconda3/lib/python3.8/site-package
s (from requests->pycoingecko) (1.26.4)
Requirement already satisfied: certifi>=2017.4.17 in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages
(from requests->pycoingecko) (2020.12.5)
Requirement already satisfied: chardet<5,>=3.0.2 in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (f
rom requests->pycoingecko) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (from r
equests->pycoingecko) (2.10)
Installing collected packages: pycoingecko
Successfully installed pycoingecko-2.2.0
Collecting plotly
  Downloading plotly-5.1.0-py2.py3-none-any.whl (20.6 MB)
     |████████████████████████████████| 20.6 MB 135 kB/s eta 0:00:01     |████████████████████████▍       | 1
7.1 MB 7.7 MB/s eta 0:00:01
Requirement already satisfied: six in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (from plotly)
(1.15.0)
Collecting tenacity>=6.2.0
  Downloading tenacity-8.0.1-py3-none-any.whl (24 kB)
Installing collected packages: tenacity, plotly
Successfully installed plotly-5.1.0 tenacity-8.0.1
Collecting mplfinance
  Downloading mplfinance-0.12.7a17-py3-none-any.whl (62 kB)
     |████████████████████████████████| 62 kB 544 kB/s eta 0:00:01
Requirement already satisfied: pandas in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (from mplfina
nce) (1.2.4)
Requirement already satisfied: matplotlib in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (from mpl
finance) (3.3.4)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (f
rom matplotlib->mplfinance) (1.3.1)
Requirement already satisfied: pillow>=6.2.0 in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib->mplfinance) (8.2.0)
Requirement already satisfied: python-dateutil>=2.1 in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages
(from matplotlib->mplfinance) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /Users/gennaro/opt/anaconda3/lib/pyt
hon3.8/site-packages (from matplotlib->mplfinance) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (from m
atplotlib->mplfinance) (0.10.0)
Requirement already satisfied: numpy>=1.15 in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (from ma
tplotlib->mplfinance) (1.20.1)
Requirement already satisfied: six in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (from cycler>=0.
10->matplotlib->mplfinance) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in /Users/gennaro/opt/anaconda3/lib/python3.8/site-packages (from p
andas->mplfinance) (2021.1)
Installing collected packages: mplfinance
Successfully installed mplfinance-0.12.7a17
```

In [2]:
```
import pandas as pd
import numpy as np
import plotly.graph_objects as go
from plotly.offline import plot
import matplotlib.pyplot as plt
import datetime
from pycoingecko import CoinGeckoAPI
from mplfinance.original_flavor import candlestick2_ohlc
```

Lets start off by getting the data we need. Using the `get_coin_market_chart_by_id(id, vs_currency, days)`. `id` is the name of the coin you want, `vs_currency` is the currency you want the price in, and `days` is how many days back from today you want.

In [3]:
```
cg = CoinGeckoAPI()

bitcoin_data = cg.get_coin_market_chart_by_id(id='bitcoin', vs_currency='usd', days=30)
```

In [4]:
```
type(bitcoin_data )
bitcoin_data.keys()
```

Out[4]:
```
dict_keys(['prices', 'market_caps', 'total_volumes'])
```

The response we get is in the form of a JSON which includes the price, market caps, and total volumes along with timestamps for each observation. We are focused on the prices so we will select that data.

In [5]:
```
bitcoin_price_data = bitcoin_data['prices']

bitcoin_price_data[0:5]
```

Out[5]:
```
[[1625655689521, 34684.643576587696],
 [1625659225480, 34896.269656399796],
 [1625662871205, 34831.016716574613],
 [1625666484260, 34856.454751721581],
 [1625670106643, 34645.808964225257]]
```

Finally lets turn this data into a Pandas DataFrame.

In [6]:
```
data = pd.DataFrame(bitcoin_price_data, columns=['TimeStamp', 'Price'])
```

Now that we have the DataFrame we will convert the timestamp to datetime and save it as a column called `Date`. We will map our `unix_to_datetime` to each timestamp and convert it to a readable datetime.

In [7]:
```
data['date'] = data['TimeStamp'].apply(lambda d: datetime.date.fromtimestamp(d/1000.0))
data.head()
```

Out[7]:

	TimeStamp	Price	date
0	1625655689521	34684.643577	2021-07-07
1	1625659225480	34896.269656	2021-07-07
2	1625662871205	34831.016716	2021-07-07
3	1625666484260	34856.454752	2021-07-07
4	1625670106643	34645.808964	2021-07-07

Using this modified dataset we can now group by the `Date` and find the min, max, open, and close for the candlesticks.

In [8]:
```
candlestick_data = data.groupby(data.date, as_index=False).agg({"Price": ['min', 'max', 'first', 'last']})
candlestick_data
```

Out[8]:

	date	Price			
		min	max	first	last
0	2021-07-07	34270.507113	34896.269656	34684.643577	34270.507113
1	2021-07-08	32419.564309	34246.718293	34246.718293	32855.186521
2	2021-07-09	32406.328806	34017.671084	32573.549339	34017.671084
3	2021-07-10	33407.343222	34247.224326	34062.786216	33518.692003
4	2021-07-11	33516.675249	34596.640048	33686.074362	34596.640048
5	2021-07-12	32882.912733	34596.993395	34461.454130	32992.757132
6	2021-07-13	32418.781521	33327.475007	33153.003562	32663.552887
7	2021-07-14	31856.382637	32971.734821	32572.967298	32971.734821
8	2021-07-15	31372.653494	33098.661964	33094.153414	31888.518884
9	2021-07-16	31174.142554	32196.821372	31850.536937	31808.979779
10	2021-07-17	31340.322628	31973.261936	31602.173425	31787.894581
11	2021-07-18	31262.148761	32239.857247	31704.688442	31836.846214
12	2021-07-19	30693.310473	31931.418112	31614.674468	30831.454281
13	2021-07-20	29612.195330	30974.897413	30965.831961	29835.161507
14	2021-07-21	29599.878053	32268.756320	29599.878053	31973.688491
15	2021-07-22	31920.429443	32540.290282	32119.049551	32269.562945
16	2021-07-23	32180.828599	32858.577162	32265.141095	32674.785027
17	2021-07-24	33367.693486	34458.753006	33367.693486	34164.333378
18	2021-07-25	33984.608998	34749.942488	34055.867021	34588.613629
19	2021-07-26	34624.035758	39841.370269	34624.035758	37230.717875
20	2021-07-27	36547.977268	38693.712529	37413.736771	38267.710998
21	2021-07-28	38457.399449	40840.142596	39265.542289	40187.294589
22	2021-07-29	39476.953676	40616.613873	39752.003988	39564.840254
23	2021-07-30	38681.764211	41780.304209	39838.622125	41780.304209
24	2021-07-31	41192.332176	41964.840673	41314.594012	41906.233844
25	2021-08-01	40998.705035	42628.499544	41874.177040	41153.104589
26	2021-08-02	38874.343392	40570.458060	40302.129545	39483.899740
27	2021-08-03	38082.362403	39759.769303	39459.971477	38419.854982
28	2021-08-04	37729.027675	39877.209001	38406.356156	39877.209001
29	2021-08-05	37595.746824	40952.350810	39653.344554	40844.893800
30	2021-08-06	39871.571561	41135.481842	40927.234106	40767.903281

Finally we are now ready to use plotly to create our Candlestick Chart.

In [9]:
```
fig = go.Figure(data=[go.Candlestick(x=candlestick_data['date'],
                open=candlestick_data['Price']['first'],
                high=candlestick_data['Price']['max'],
                low=candlestick_data['Price']['min'],
                close=candlestick_data['Price']['last'])
                ])

fig.update_layout(xaxis_rangeslider_visible=False)

fig.show()
```