PDA data science - Yahoo Finance

Notebook 3: by Michael Ferrie, March 2022

Introduction to API's

For those who might be unfamiliar, API stands for Application Programming Interface. An API is actually an interface that has a set of functions. These set of functions allow programmers to acquire some specific features or the data of an application. Web API is an API as the name suggests, it can be accessed over the web using the HTTP protocol. It is a framework that helps you to create and develop HTTP based RESTFUL services. Once we can access an API programmatically we can create requests to send to it in order to return data that is useful to us. Many companies offer useful public API's and python allows us to connect to these.

Yahoo Finance

Yahoo Finance used to have their own official API, but this was decommissioned in May 2017, following wide-spread misuse of data. These days a range of unofficial APIs and libraries exist to access the same data, including yfinance.

The project yfinance was developed by Ran Aroussi and offers a replacement for the old Yahoo API and we can use it with Python to download market data from Yahoo! finance. Before getting started you should install the library with pip:

```
pip install yfinance
```

With yfinance we can obtain historical and real time data for a variety of financial markets and products, as shown on Yahoo Finance.

A stock ticker reports transaction and price data, these tickers, update continuously throughout the day. Have a look at the list of tickers.

OHLC is a financial term for Open Close Low High, this is the value of a stock when the market opens and closes and the lowest point and highest point in a particular duration.

Read through the notebook and run each of the examples then answer the questions at the end.

Getting a reply from the API

- First we will import the `yfinance` library, then using the Ticker module, we can search for information on Google, they have the abbreviation GOOG, run the following you should get a response from the API. The Ticker module, allows you to access ticker data in a more Pythonic way. Ticker.info provides us with a lot of data on a company, have a look.

In [1]:
```
# import library
import yfinance as yf

# assign data to a variable, then print result
goog = yf.Ticker("GOOG")
print(goog.info)
```

```
{'zip': '94043', 'sector': 'Communication Services', 'fullTimeEmployees': 163906, 'longB
usinessSummary': 'Alphabet Inc. provides various products and platforms in the United St
ates, Europe, the Middle East, Africa, the Asia-Pacific, Canada, and Latin America. It o
perates through Google Services, Google Cloud, and Other Bets segments. The Google Servi
ces segment offers products and services, including ads, Android, Chrome, hardware, Gmai
l, Google Drive, Google Maps, Google Photos, Google Play, Search, and YouTube. It is als
o involved in the sale of apps and in-app purchases and digital content in the Google Pl
ay store; and Fitbit wearable devices, Google Nest home products, Pixel phones, and othe
r devices, as well as in the provision of YouTube non-advertising services. The Google C
loud segment offers infrastructure, platform, and other services; Google Workspace that
include cloud-based collaboration tools for enterprises, such as Gmail, Docs, Drive, Cal
endar, and Meet; and other services for enterprise customers. The Other Bets segment sel
ls health technology and internet services. The company was founded in 1998 and is headq
uartered in Mountain View, California.', 'city': 'Mountain View', 'phone': '650 253 000
0', 'state': 'CA', 'country': 'United States', 'companyOfficers': [], 'website': 'http
s://www.abc.xyz', 'maxAge': 1, 'address1': '1600 Amphitheatre Parkway', 'industry': 'Int
ernet Content & Information', 'ebitdaMargins': 0.35452998, 'profitMargins': 0.27573, 'gr
ossMargins': 0.56929, 'operatingCashflow': 97468997632, 'revenueGrowth': 0.23, 'operatin
gMargins': 0.3047, 'ebitda': 95841001472, 'targetLowPrice': 2950, 'recommendationKey':
'strong_buy', 'grossProfits': 146698000000, 'freeCashflow': 52881248256, 'targetMedianPr
ice': 3150, 'currentPrice': 2186.26, 'earningsGrowth': -0.063, 'currentRatio': 2.871, 'r
eturnOnAssets': 0.15049, 'numberOfAnalystOpinions': 9, 'targetMeanPrice': 3168.89, 'debt
ToEquity': 11.26, 'returnOnEquity': 0.308, 'targetHighPrice': 3600, 'totalCash': 1339700
01920, 'totalDebt': 28601999360, 'totalRevenue': 270334001152, 'totalCashPerShare': 203.
447, 'financialCurrency': 'USD', 'revenuePerShare': 406.892, 'quickRatio': 2.738, 'recom
mendationMean': 1.5, 'exchange': 'NMS', 'shortName': 'Alphabet Inc.', 'longName': 'Alpha
bet Inc.', 'exchangeTimezoneName': 'America/New_York', 'exchangeTimezoneShortName': 'ED
T', 'isEsgPopulated': False, 'gmtOffSetMilliseconds': '-14400000', 'quoteType': 'EQUIT
Y', 'symbol': 'GOOG', 'messageBoardId': 'finmb_29096', 'market': 'us_market', 'annualHol
dingsTurnover': None, 'enterpriseToRevenue': 4.936, 'beta3Year': None, 'enterpriseToEbit
da': 13.922, '52WeekChange': -0.091582954, 'morningStarRiskRating': None, 'forwardEps':
132.65, 'revenueQuarterlyGrowth': None, 'sharesOutstanding': 313376000, 'fundInceptionDa
te': None, 'annualReportExpenseRatio': None, 'totalAssets': None, 'bookValue': 385.577,
'sharesShort': 1737779, 'sharesPercentSharesOut': 0.0026, 'fundFamily': None, 'lastFisca
lYearEnd': 1640908800, 'heldPercentInstitutions': 0.64886004, 'netIncomeToCommon': 74538
999808, 'trailingEps': 110.585, 'lastDividendValue': None, 'SandP52WeekChange': -0.07045
1856, 'priceToBook': 5.6700997, 'heldPercentInsiders': 0.00037999998, 'nextFiscalYearEn
d': 1703980800, 'yield': None, 'mostRecentQuarter': 1648684800, 'shortRatio': 1.24, 'sha
resShortPreviousMonthDate': 1648684800, 'floatShares': 555688291, 'beta': 1.132142, 'ent
erpriseValue': 1334283993088, 'priceHint': 2, 'threeYearAverageReturn': None, 'lastSplit
Date': 1430092800, 'lastSplitFactor': '10027455:10000000', 'legalType': None, 'lastDivid
endDate': None, 'morningStarOverallRating': None, 'earningsQuarterlyGrowth': -0.083, 'pr
iceToSalesTrailing12Months': 5.3151135, 'dateShortInterest': 1651190400, 'pegRatio': 1.1
5, 'ytdReturn': None, 'forwardPE': 16.481419, 'lastCapGain': None, 'shortPercentOfFloa
t': None, 'sharesShortPriorMonth': 1397898, 'impliedSharesOutstanding': 0, 'category': N
one, 'fiveYearAverageReturn': None, 'previousClose': 2214.91, 'regularMarketOpen': 2241.
71, 'twoHundredDayAverage': 2747.595, 'trailingAnnualDividendYield': 0, 'payoutRatio':
```

```
0, 'volume24Hr': None, 'regularMarketDayHigh': 2251, 'navPrice': None, 'averageDailyVolu
me10Day': 1564790, 'regularMarketPreviousClose': 2214.91, 'fiftyDayAverage': 2542.118,
'trailingAnnualDividendRate': 0, 'open': 2241.71, 'toCurrency': None, 'averageVolume10da
ys': 1564790, 'expireDate': None, 'algorithm': None, 'dividendRate': None, 'exDividendDa
te': None, 'circulatingSupply': None, 'startDate': None, 'regularMarketDayLow': 2127.46,
'currency': 'USD', 'trailingPE': 19.76995, 'regularMarketVolume': 1879301, 'lastMarket':
None, 'maxSupply': None, 'openInterest': None, 'marketCap': 1436855959552, 'volumeAllCur
rencies': None, 'strikePrice': None, 'averageVolume': 1456942, 'dayLow': 2127.46, 'ask':
0, 'askSize': 800, 'volume': 1879301, 'fiftyTwoWeekHigh': 3042, 'fromCurrency': None, 'f
iveYearAvgDividendYield': None, 'fiftyTwoWeekLow': 2127.46, 'bid': 0, 'tradeable': Fals
e, 'dividendYield': None, 'bidSize': 1000, 'dayHigh': 2251, 'regularMarketPrice': 2186.2
6, 'preMarketPrice': 2206, 'logo_url': 'https://logo.clearbit.com/abc.xyz', 'trailingPeg
Ratio': 0.7984}
```

Specifying date ranges

We can specify a date range for data to be returned, The full range of intervals available are:

1m, 2m, 5m, 15m, 30m, 60m, 90m, 1h, 1d, 5d, 1wk, 1mo, 3mo

In [2]:
```
# specify 3 days with '3d' of data from the API
data = yf.download(['GOOG'], period='3d')
print(data)

# the data comes from the API as a pandas dataframe
print(type(data))
```

```
[*********************100%***********************]  1 of 1 completed
                   Open         High          Low        Close    Adj Close  \
Date
2022-05-18  2304.750000  2313.913086  2242.840088  2248.020020  2248.020020
2022-05-19  2236.820068  2271.750000  2209.360107  2214.909912  2214.909912
2022-05-20  2241.709961  2251.000000  2127.459961  2186.260010  2186.260010

             Volume
Date
2022-05-18  1399100
2022-05-19  1459600
2022-05-20  1878100
<class 'pandas.core.frame.DataFrame'>
```

Access Microsoft finances

Here is how to get some data on Microsoft, we will explore some of the options of the library.All of the available options are here: https://pypi.org/project/yfinance/

In [3]:
```
msft = yf.Ticker("MSFT")

# get stock info
msft.info

# get historical market data
hist = msft.history(period="max")

# show major holders
msft.major_holders

# show cashflow
msft.cashflow

# print some of the data
```

```
print(msft.cashflow)
print(msft.major_holders)

# show the data type
print(type(msft.cashflow))
```

```
                                            2021-06-30    2020-06-30  \
Investments                               2.876000e+09  6.980000e+09   
Change To Liabilities                     7.431000e+09  5.230000e+09   
Total Cashflows From Investing Activities -2.757700e+10 -1.222300e+10   
Net Borrowings                           -3.750000e+09 -5.518000e+09   
Total Cash From Financing Activities     -4.848600e+10 -4.603100e+10   
Change To Operating Activities            1.160000e+09 -6.730000e+08   
Issuance Of Stock                         1.693000e+09  1.343000e+09   
Net Income                                6.127100e+10  4.428100e+10   
Change In Cash                            6.480000e+08  2.220000e+09   
Repurchase Of Stock                      -2.738500e+10 -2.296800e+10   
Effect Of Exchange Rate                  -2.900000e+07 -2.010000e+08   
Total Cash From Operating Activities      7.674000e+10  6.067500e+10   
Depreciation                              1.090000e+10  1.230000e+10   
Other Cashflows From Investing Activities -9.220000e+08 -1.241000e+09   
Dividends Paid                           -1.652100e+10 -1.513700e+10   
Change To Inventory                      -7.370000e+08  1.680000e+08   
Change To Account Receivables            -6.481000e+09 -2.577000e+09   
Other Cashflows From Financing Activities -2.523000e+09 -3.751000e+09   
Change To Netincome                       5.505000e+09  5.577000e+09   
Capital Expenditures                     -2.062200e+10 -1.544100e+10   

                                            2019-06-30    2018-06-30  
Investments                               5.400000e+08  6.557000e+09  
Change To Liabilities                     4.694000e+09  7.070000e+09  
Total Cashflows From Investing Activities -1.577300e+10 -6.061000e+09  
Net Borrowings                           -4.000000e+09 -1.020100e+10  
Total Cash From Financing Activities     -3.688700e+10 -3.359000e+10  
Change To Operating Activities           -1.542000e+09 -4.590000e+08  
Issuance Of Stock                         1.142000e+09  1.002000e+09  
Net Income                                3.924000e+10  1.657100e+10  
Change In Cash                           -5.900000e+08  4.283000e+09  
Repurchase Of Stock                      -1.954300e+10 -1.072100e+10  
Effect Of Exchange Rate                  -1.150000e+08  5.000000e+07  
Total Cash From Operating Activities      5.218500e+10  4.388400e+10  
Depreciation                              1.160000e+10  9.900000e+09  
Other Cashflows From Investing Activities -1.241000e+09 -9.800000e+07  
Dividends Paid                           -1.381100e+10 -1.269900e+10  
Change To Inventory                       5.970000e+08 -4.650000e+08  
Change To Account Receivables            -2.812000e+09 -3.862000e+09  
Other Cashflows From Financing Activities -6.750000e+08 -9.710000e+08  
Change To Netincome                      -2.521000e+09 -3.054000e+09  
Capital Expenditures                     -1.392500e+10 -1.163200e+10  
        0                                      1
0   0.08%        % of Shares Held by All Insider
1  71.64%       % of Shares Held by Institutions
2  71.69%        % of Float Held by Institutions
3    5925   Number of Institutions Holding Shares
<class 'pandas.core.frame.DataFrame'>
```

Multiple stocks

We can download data for one ticker using the Ticker object and multiple tickers using the download method.

In [4]:
```
# get data for google and meta for one month
df = yf.download(['GOOG','META'], period='1mo')
```

```
# use head to show only the top of the dataframe
df.head()
```

```
[*********************100%***********************]  2 of 2 completed
```

Out[4]:

	Adj Close		Close		High		Low		Ope
	GOOG	META	GOOG	META	GOOG	META	GOOG	META	GOOG	MET
Date										
2022-04-21	2498.750000	10.17	2498.750000	10.17	2606.149902	10.7950	2493.000000	10.1201	2587.000000	10.6
2022-04-22	2392.280029	9.95	2392.280029	9.95	2509.040039	10.2994	2382.810059	9.9100	2500.000000	10.1
2022-04-25	2465.000000	10.08	2465.000000	10.08	2465.560059	10.0900	2375.385010	9.7600	2388.590088	9.7
2022-04-26	2390.120117	9.63	2390.120117	9.63	2455.000000	10.0100	2383.237061	9.6200	2455.000000	10.0
2022-04-27	2300.409912	9.47	2300.409912	9.47	2350.000000	9.7500	2262.485107	9.4300	2287.459961	9.5

In [5]:

```
# Specify the date range and group by ticker
# Remember dates are in american middle endian
df = yf.download(['GOOG','META'], start='2022-01-01',
                 end='2022-01-31', group_by='ticker')
df.head()
```

```
[*********************100%***********************]  2 of 2 completed
```

Out[5]:

	GOOG									
	Open	High	Low	Close	Adj Close	Volume	Open	High	Low	Close
Date										
2022-01-03	2889.510010	2911.000000	2870.050049	2901.489990	2901.489990	1260700	15.36	15.360	15.04	15.25
2022-01-04	2911.010010	2932.199951	2876.322998	2888.330078	2888.330078	1146400	15.30	15.300	14.69	14.91
2022-01-05	2883.620117	2885.959961	2750.469971	2753.070068	2753.070068	2482100	14.79	14.860	14.24	14.24
2022-01-06	2749.949951	2793.719971	2735.270020	2751.020020	2751.020020	1452500	14.19	14.435	13.94	14.28
2022-01-07	2758.100098	2765.094971	2715.780029	2740.090088	2740.090088	970400	14.28	14.440	14.01	14.13

Bitcoin value in USD

Let's have a look at some Bitcoin data, this will pull the Bitcoin value for the first week in February 2022, we can set the interval to 1 hour so that we can see the value change.

In [6]:

```
df = yf.download(['BTC-USD'], start='2022-02-01', end='2022-02-07', interval='1h')
df.head()
print(df)
```

```
[*********************100%***********************]  1 of 1 completed
                                   Open          High           Low  \
2022-02-01 00:00:00+00:00  38481.765625  38545.562500  38336.601562
2022-02-01 01:00:00+00:00  38359.769531  38391.707031  38223.531250
```

```
2022-02-01 02:00:00+00:00  38266.632812  38489.617188  38256.949219
2022-02-01 03:00:00+00:00  38486.484375  38622.914062  38430.562500
2022-02-01 04:00:00+00:00  38617.210938  38639.039062  38532.636719
...                                 ...           ...           ...
2022-02-06 19:00:00+00:00  41517.476562  41707.207031  41517.476562
2022-02-06 20:00:00+00:00  41707.281250  41736.804688  41546.320312
2022-02-06 21:00:00+00:00  41603.175781  41701.269531  41568.250000
2022-02-06 22:00:00+00:00  41678.640625  41699.085938  41611.902344
2022-02-06 23:00:00+00:00  41599.914062  42500.785156  41599.914062

                                  Close     Adj Close      Volume
2022-02-01 00:00:00+00:00  38341.386719  38341.386719           0
2022-02-01 01:00:00+00:00  38265.773438  38265.773438           0
2022-02-01 02:00:00+00:00  38468.675781  38468.675781           0
2022-02-01 03:00:00+00:00  38615.339844  38615.339844   268036096
2022-02-01 04:00:00+00:00  38573.503906  38573.503906    40687616
...                                 ...           ...         ...
2022-02-06 19:00:00+00:00  41707.207031  41707.207031    18229248
2022-02-06 20:00:00+00:00  41594.679688  41594.679688           0
2022-02-06 21:00:00+00:00  41680.023438  41680.023438           0
2022-02-06 22:00:00+00:00  41611.902344  41611.902344           0
2022-02-06 23:00:00+00:00  42398.328125  42398.328125  1624065024

[144 rows x 6 columns]
```

Get some statistics from the data

Once we get the data into a pandas dataframe we can run many possible operations on it, have a look at the list. There are so many possibilities, the trick is being able to understand the documentation so you can use them. Here are some examples:

In [7]:
```
# Describe will give you an overview of the data
df.describe()
```

Out[7]:

	Open	High	Low	Close	Adj Close	Volume
count	144.000000	144.000000	144.000000	144.000000	144.000000	1.440000e+02
mean	39188.592041	39323.117811	39081.692546	39212.252360	39212.252360	1.878929e+08
std	1869.031587	1873.242475	1881.953414	1886.996552	1886.996552	4.539156e+08
min	36431.238281	36637.382812	36375.539062	36430.933594	36430.933594	0.000000e+00
25%	37497.721680	37647.432617	37425.184570	37487.948242	37487.948242	0.000000e+00
50%	38581.998047	38681.814453	38458.218750	38580.265625	38580.265625	0.000000e+00
75%	41469.354492	41568.612305	41401.530273	41465.515625	41465.515625	2.080177e+08
max	41742.320312	42500.785156	41617.992188	42398.328125	42398.328125	4.199303e+09

In [8]:
```
# Access a specific column in the data with []
print(df['High'].mean())
print(df['High'].min())
print(df['High'].max())
print(df['High'].median())
```

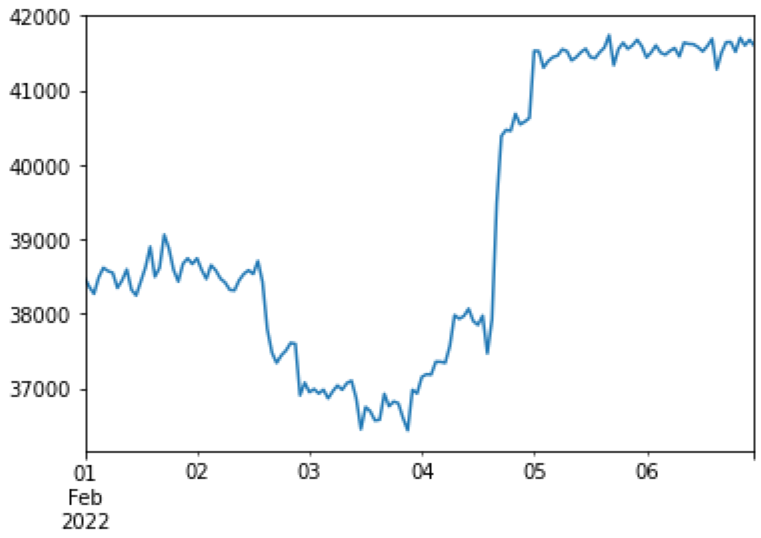
```
39323.11781141493
36637.3828125
42500.78515625
38681.814453125
```

Visualise the data

Pandas has a simple plot function and it will let us access specific columns in the dataframe, it understands the shape of the data and it is easy to work with. This will allow us to see the Bitcoin value in USD over the week that we have data for. Something obviously happened on the 4th of February to cause the spike. The best day to buy bitcoin that week would have been on the 3rd.

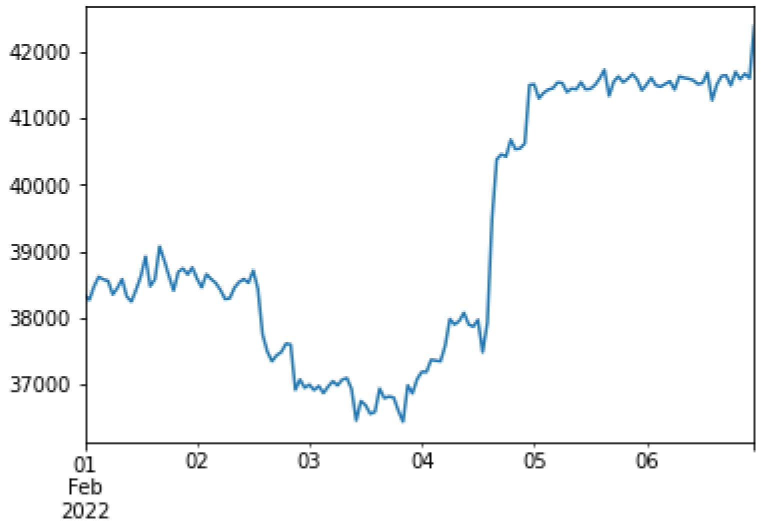
In [9]:
```
# Plot the open column
df["Open"].plot()
```

Out[9]:
```
<AxesSubplot:>
```



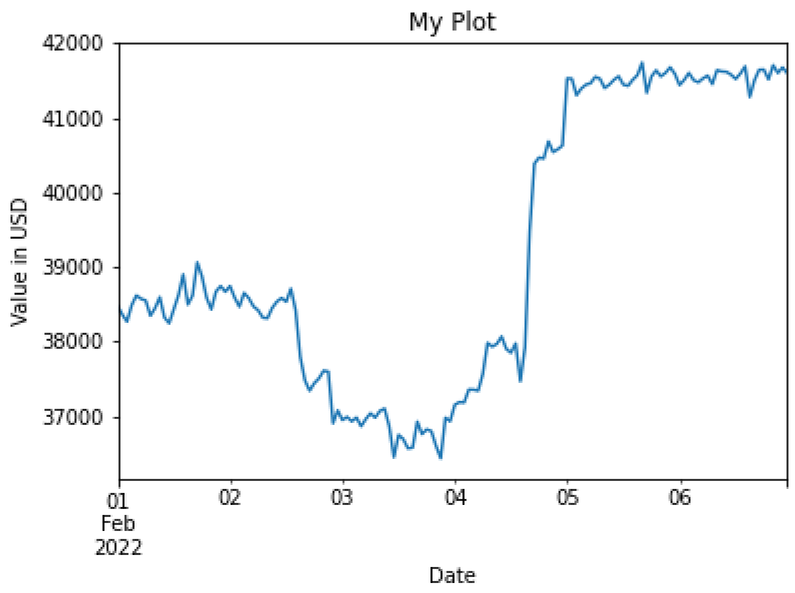
In [10]:
```
# Plot the close column
df["Close"].plot()
```

Out[10]:
```
<AxesSubplot:>
```



In [11]:
```
# Adding some extra details to the plot
df['Open'].plot(kind='line', title="My Plot", xlabel="Date", ylabel="Value in USD")
```

Out[11]:
```
<AxesSubplot:title={'center':'My Plot'}, xlabel='Date', ylabel='Value in USD'>
```



Multiple line Plots

Now if we want to compare two plots side by side we need another library, if you have not already, install Matplotlib, we need to use pyplot from this library for this next task.

```
pip install matplotlib
```

First we want to pass multiple columns in the data frame to the plot function, and assign them a colour, then we can add a title and some labels, and a legend.

In [12]:
```
# import pyplot
import matplotlib.pyplot as plt

# use the plot method on each of the columsn in the dataframe
df['Open'].plot(label='Open', color='green')
df['Close'].plot(label='Close', color='orange')
df['High'].plot(label='High', color='blue')
df['Low'].plot(label='Low', color='red')

# adding title to the plot
plt.title('Bitcoin in USD for Week 1 in February')

# adding labels to the axes, add a legend
plt.xlabel('Date')
plt.ylabel('Value in USD')
plt.legend()
```

Out[12]:
```
<matplotlib.legend.Legend at 0x7f8c0cd241f0>
```

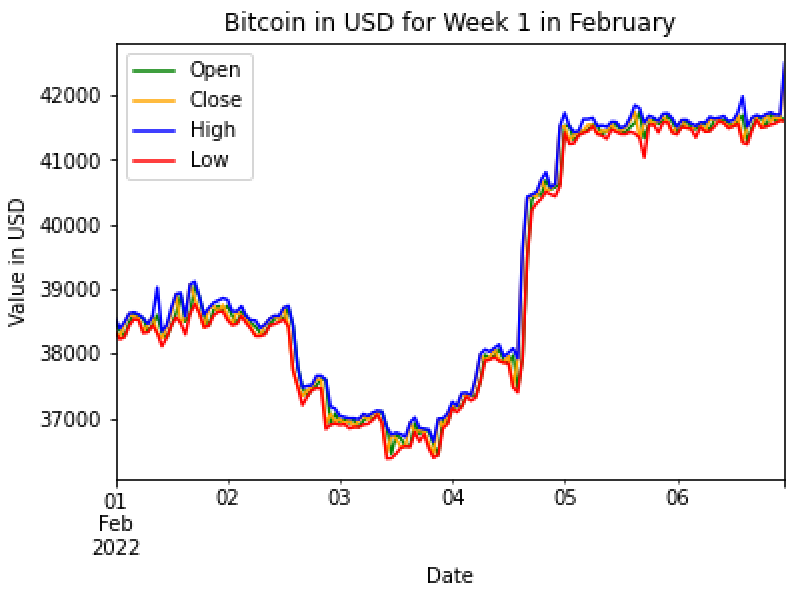


Figure size

Our multi line plot looks a bit squashed, here are some options to make it better, `figsize` lets us specify the size of the chart in **inches**, by default all charts come out as 6.4x4.8 inches. The chart will look better at a bigger size, let us also specify the quality of the image as 150 dpi. The plot will be easier to see if we increase this, but it might take slightly longer to render.

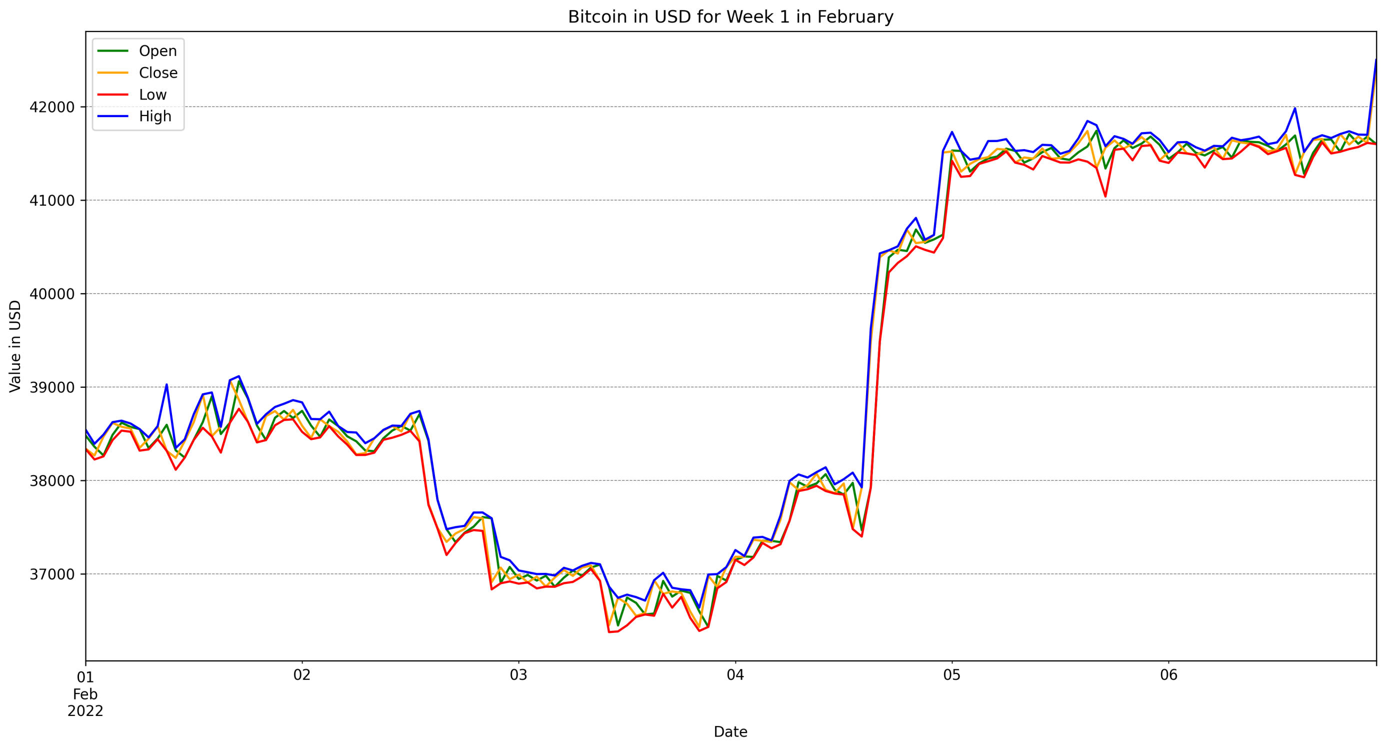
In [13]:
```
# set the plot size and image quality
plt.figure(figsize=(16, 8), dpi=300)

# use the plot method on each of the columsn in the dataframe
df['Open'].plot(label='Open', color='green')
df['Close'].plot(label='Close', color='orange')
df['Low'].plot(label='Low', color='red')
df['High'].plot(label='High', color='blue')


# adding title to the plot
plt.title('Bitcoin in USD for Week 1 in February')

# adding labels to the axes, add a legend, add gridlines
plt.xlabel('Date')
plt.ylabel('Value in USD')
plt.grid(color = 'grey', linestyle = '--', linewidth = 0.5)
plt.legend()
```

Out[13]:
```
<matplotlib.legend.Legend at 0x7f8c0cc87910>
```



Questions

Add your solution to the questions below

Plot the Bitcoin value in USD using the high column for the whole month of January 2021, set the interval to 1 hour?

In [14]:
```
# your answer below this line
```

Create a plot of the Bitcoin value in USD using the high column for the whole year of 2021, set the interval to 1 day?

In [15]:
```
# your answer below this line
```

Plot the Bitcoin value in USD using the close column for as far back as you can get data for, this could be a number of years?

In [16]:
```
# your answer below this line
```

Go to the Cryptocurency tickers list find the name of the Etherium USD ticker and then create a plot of the value of Etherium in USD using the high column for the whole year of 2021, set the interval to 1 day?

In [17]:
```
# your answer below this line
```

Go to the Most active tickers list and find the top two most active stocks. Create a plot that compares their high value for every day in 2021. The plot should have two lines in different colours, there should be a legend and axis labels.

In [18]:
```
# your answer below this line
```

Adapt your plot from the previous question to show the high value of the top 4 stocks in the list?

In [19]:
```
# your answer below this line
```

Go to the FIAT Currency tickers list and find the tickers for EUR/USD and GBP/USD, create a plot that shows the value from the close column for the last 3 years every day for each of the tickers.

In [20]:
```
# your answer below this line
```

Adapt the chart from the previous question to add additional lines for the AUD/USD CAD/USD and NZD/USD, give each line an appropriate colour and add a legend and axis labels, print this chart out in 150dpi and make it 16x8 inches. Add gridlines to the plot?

In [21]:
```
# your answer below this line
```