Cryptowatch Data Visualizations

Part 1: Working with code

In [1]:

```
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline
```

numpy the most popular Python library for array manipulation and numeric computing matplotlib the most popular visualization library in the Python ecosystem.

In [2]:

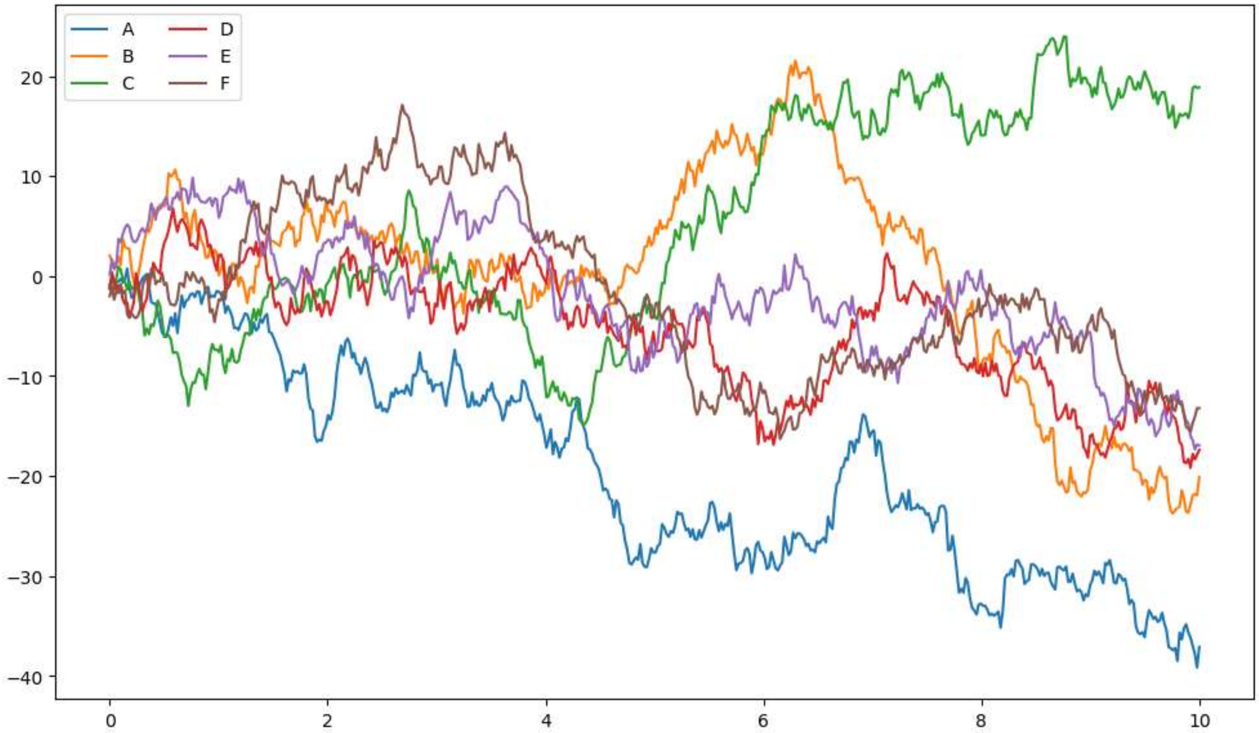
```
x = np.linspace(0, 10, 500)
y = np.cumsum(np.random.randn(500, 6), 0)
```

In [3]:

```
plt.figure(figsize=(12, 7))
plt.plot(x, y)
plt.legend('ABCDEF', ncol=2, loc='upper left')
```

Out[3]:

```
<matplotlib.legend.Legend at 0x2c2aace2410>
```



Part 2: Interacting with data¶

Notebooks.ai and Jupyter Lab make it really simple to intereact with files in your local storage. These files are securely stored in the cloud and you can access them from anywhere in the world.

To show you the full potential of Notebooks.ai, we're going to pull cryptocurrencies prices from a public API and download them as Excel files, pretty fancy . I need to import two libraries first: requests (to pull data from the web) and pandas to process it.

In [4]:

```
import  requests
import pandas as pd
```

I have a predefined function that simplifies the process of importing data from Cryptowatch (for reference, check their docs).

In [5]:

```
def get_historic_price(symbol, exchange='bitfinex', after='2018-09-01'):
    url = 'https://api.cryptowat.ch/markets/{exchange}/{symbol}usd/ohlc'.format(
        symbol=symbol, exchange=exchange)
    resp = requests.get(url, params={
        'periods': '3600',
        'after': str(int(pd.Timestamp(after).timestamp()))
    })
    resp.raise_for_status()
    data = resp.json()
    df = pd.DataFrame(data['result']['3600'], columns=[
        'CloseTime', 'OpenPrice', 'HighPrice', 'LowPrice', 'ClosePrice', 'Volume', 'NA'
    ])
    df['CloseTime'] = pd.to_datetime(df['CloseTime'], unit='s')
    df.set_index('CloseTime', inplace=True)
    return df
```

I will now pull data from Bitcoin and Ether, two of the most popular cryptocurrencies, for the last 7 days:

In [6]:

```
last_week = (pd.Timestamp.now() - pd.offsets.Day(7))
last_week
```

Out[6]:

```
Timestamp('2023-08-30 12:02:01.394501')
```

In [7]:

```
btc = get_historic_price('btc', 'bitstamp', after=last_week)
```

In [8]:

```
eth = get_historic_price('eth', 'bitstamp', after=last_week)
```

Bitcoin:

In [9]:

```
btc.head()
```

Out[9]:

	OpenPrice	HighPrice	LowPrice	ClosePrice	Volume	NA
CloseTime						
2023-08-30 13:00:00	27337	27416	27295	27350	123.875569	3.389334e+06
2023-08-30 14:00:00	27344	27452	27187	27365	234.699017	6.413749e+06
2023-08-30 15:00:00	27370	27418	27019	27108	213.444789	5.810830e+06
2023-08-30 16:00:00	27113	27284	27006	27204	295.054951	8.013086e+06
2023-08-30 17:00:00	27197	27212	27045	27152	127.199367	3.447892e+06

In [10]:

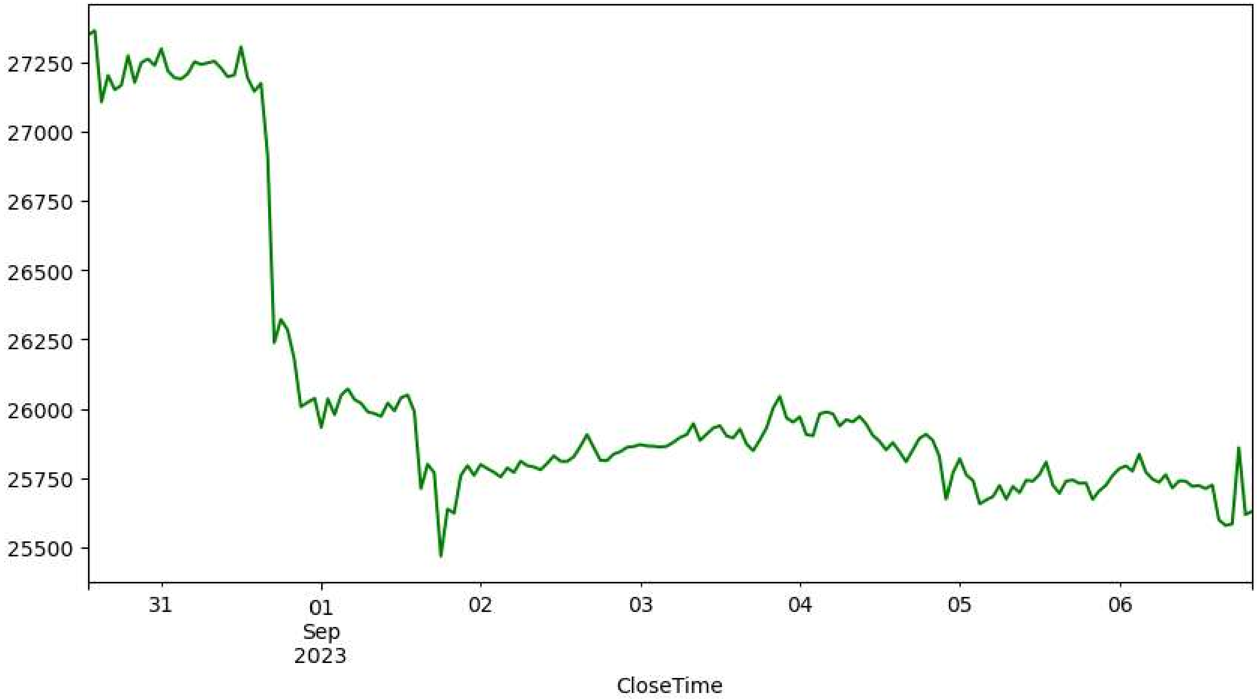
```
import datetime
```

In [18]:

```
btc['ClosePrice'].plot(figsize=(10, 5),color='green')
```

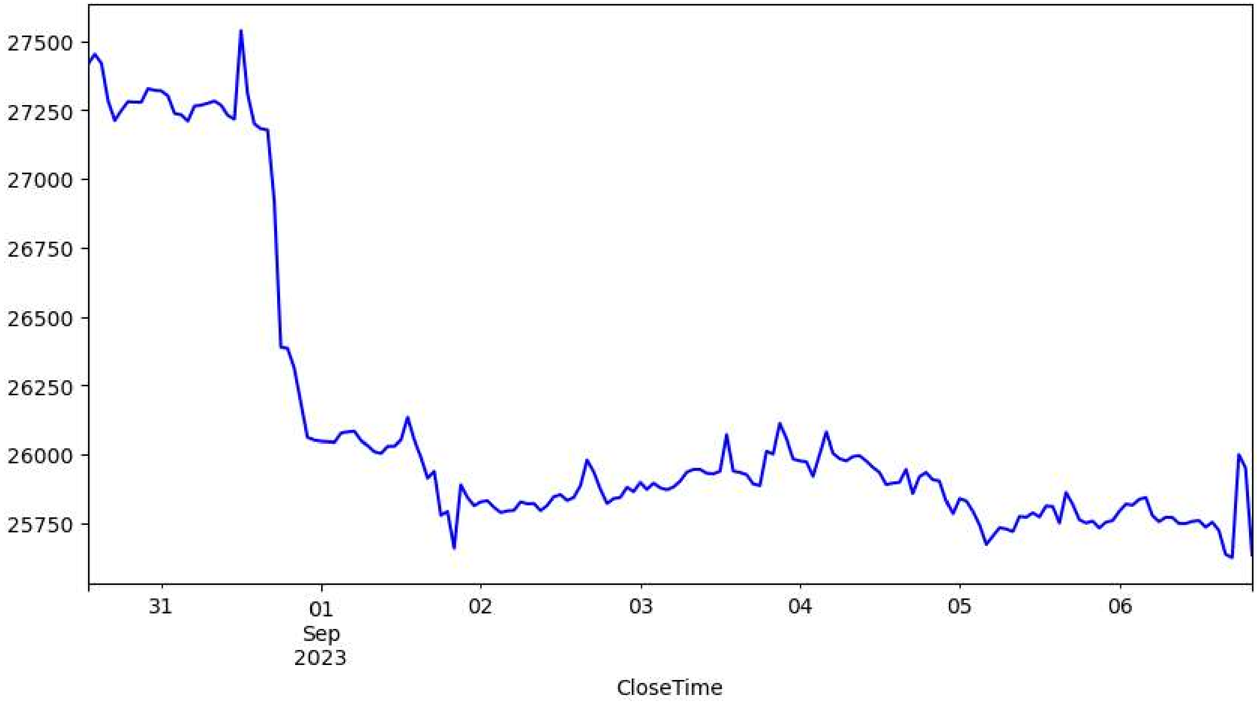
Out[18]:

```
<Axes: xlabel='CloseTime'>
```



In [20]:
```
btc['HighPrice'].plot(figsize=(10,5),color='blue')
```

Out[20]:
```
<Axes: xlabel='CloseTime'>
```



Ether:

In [21]:
```
eth.head()
```

Out[21]:

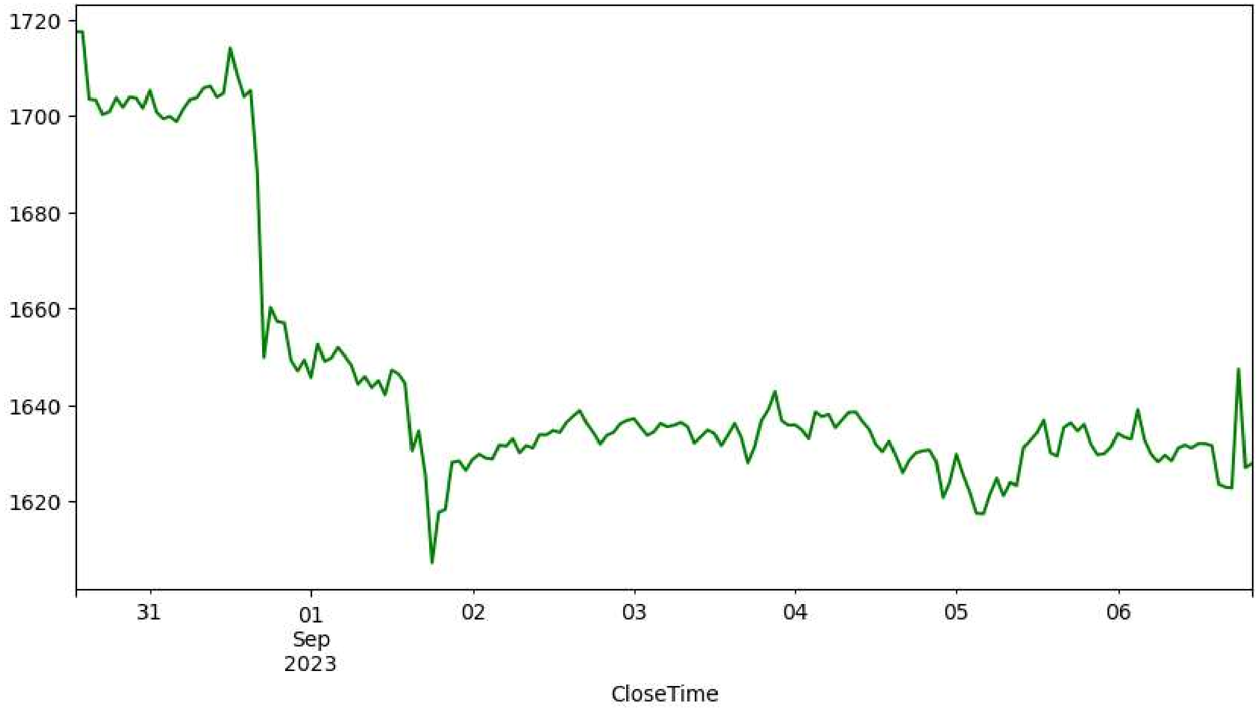
	OpenPrice	HighPrice	LowPrice	ClosePrice	Volume	NA
CloseTime						
2023-08-30 13:00:00	1715.9	1720.4	1713.2	1717.5	129.136666	2.217100e+05
2023-08-30 14:00:00	1717.3	1722.1	1705.0	1717.5	340.286165	5.831756e+05
2023-08-30 15:00:00	1718.3	1719.8	1696.8	1703.5	915.503452	1.561867e+06
2023-08-30 16:00:00	1702.6	1708.2	1698.1	1703.3	205.466617	3.499495e+05
2023-08-30 17:00:00	1702.8	1704.3	1696.5	1700.3	202.990096	3.451329e+05

In [22]:

```
eth['ClosePrice'].plot(figsize=(10, 5),color='green')
```

Out[22]:

```
<Axes: xlabel='CloseTime'>
```



As you can see, we're able to pull data from the internet with just a few lines, create a DataFrame and plot it all within Jupyter Lab.

Bonus: Dynamic plots with Bokeh

We've also included Bokeh as part of this main distribution. Bokeh is a plotting library that generates interactive plots, that can be manipulated right within your browser.

We first need to import the libraries:

In [23]:

```
from bokeh.plotting import figure, output_file, show
from bokeh.io import output_notebook
```

In [24]:

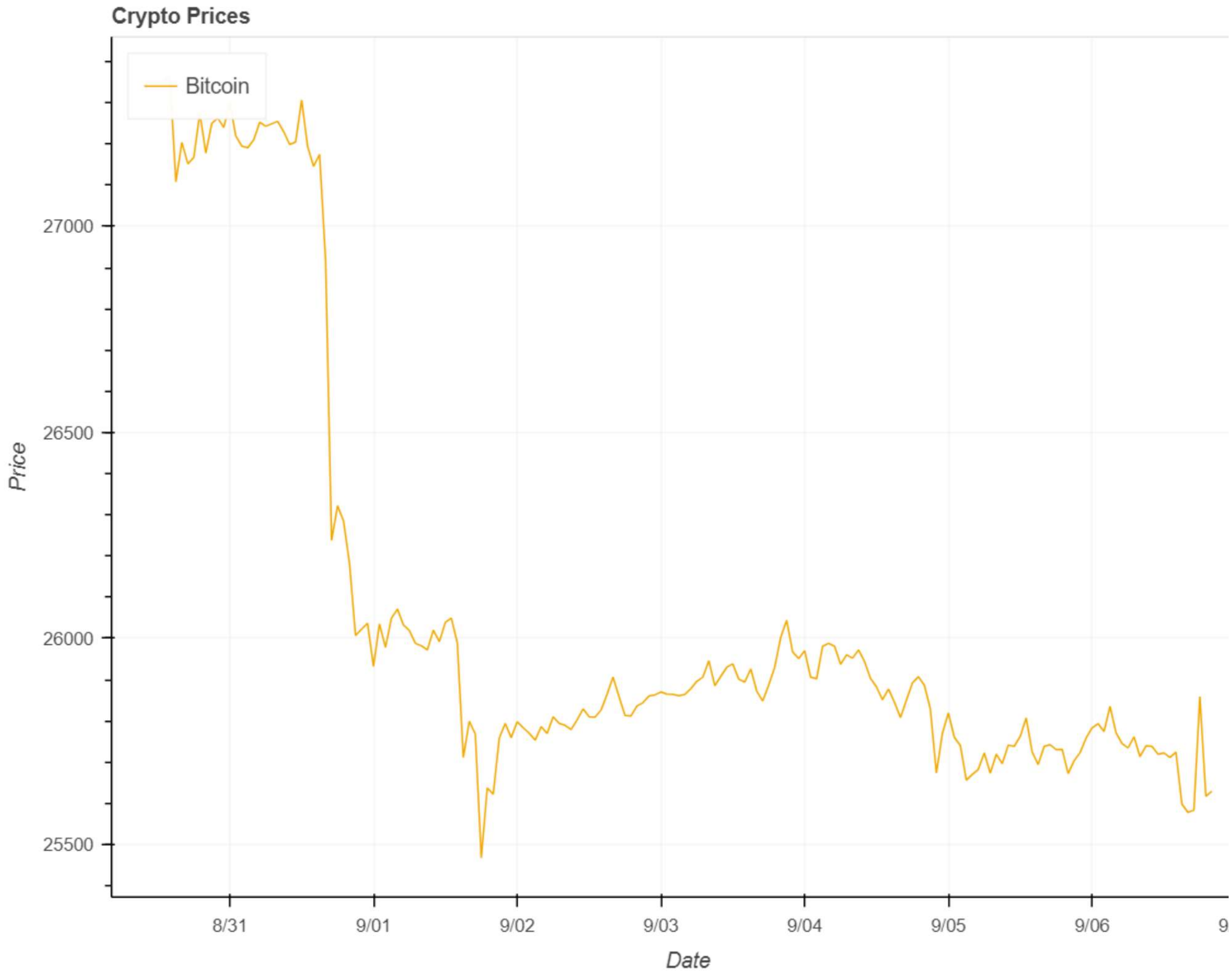
```
output_notebook()
```

BokehJS 3.2.1 successfully loaded.

In [34]:

```
p1 = figure(x_axis_type="datetime", title="Crypto Prices", width=800)
p1.grid.grid_line_alpha=0.3
```

```
p1.xaxis.axis_label = 'Date'
p1.yaxis.axis_label = 'Price'
p1.line(btc.index,btc['ClosePrice'], color='#f2a900', legend_label='Bitcoin')
#p1.line(eth.index, eth['ClosePrice'], color='#A6CEE3', legend='Ether')
p1.legend.location = "top_left"
show(p1)
```


Crypto Prices

☝️ as you can see, the plot is interactive. Try zomming in and out, and scrolling in the plot.

Part 3: Exporting to Excel

We're now ready to generate an Excel file from the downloaded prices. Working with Excel and other formats (like CSV or JSON) is extremely simple in Jupyter Lab (thanks to pandas and Python). Our first step will be to create an "Excel writer", a component from the pandas package:

In [35]:

```
writer = pd.ExcelWriter('cryptos.xlsx')
```

We'll now write both our Bitcoin and Ether data as separate sheets:

In [36]:

```
btc.to_excel(writer, sheet_name='Bitcoin')
```

In [37]:

```
eth.to_excel(writer, sheet_name='Ether')
```

save the files

In [38]:

```
writer.save()
```

```
C:\Users\ZJU\AppData\Local\Temp\ipykernel_19552\934276808.py:1: FutureWarning: save is not part of the p
ublic API, usage can give unexpected results and will be removed in a future version
  writer.save()
```

Help for this File execution , emails me turatsinzecmu2020@gmail.com

In [ ]:

```

```