01-Pandas-Visualizations

September 28, 2024

1 Pandas Built-in Data Visualization

In this lecture we will learn about pandas built-in capabilities for data visualization! It's built-off of matplotlib, but it baked into pandas for easier usage!

There are 3 main things we will cover:

- Key Continuous Plots used often in the course
- Key Distribution Plots used often in the course
- Editing Plot Properties and Connecting plots with Matplotlib

NOTE: In the next lecture we cover time series plots, which are related to continuous plots, but have special characteristics due to the index being datetime data.

Let's take a look!

1.1 Imports

[27]:
```
import numpy as np
import pandas as pd
```

1.2 The Data

Let's read in a dataset from a .csv file:

[28]:
```
df = pd.read_csv('FB.csv')
```

[29]:
```
df.head()
```

[29]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2016-09-06	126.669998	129.940002	126.470001	129.729996	129.729996	26278400
1	2016-09-07	130.039993	131.979996	129.949997	131.050003	131.050003	27990800
2	2016-09-08	130.919998	131.080002	129.809998	130.270004	130.270004	15676600
3	2016-09-09	129.710007	129.949997	127.099998	127.099998	127.099998	
4	2016-09-12	125.959999	128.759995	125.750000	128.690002	128.690002	

```
3  27100700
4  21252800
```

[31]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       1258 non-null   object
 1   Open       1258 non-null   float64
 2   High       1258 non-null   float64
 3   Low        1258 non-null   float64
 4   Close      1258 non-null   float64
 5   Adj Close  1258 non-null   float64
 6   Volume     1258 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 68.9+ KB
```

2 Plot Types

There are several plot types built-in to pandas, most of them statistical plots by nature:

- df.plot.area
- df.plot.barh
- df.plot.density
- df.plot.hist
- df.plot.line
- df.plot.scatter
- df.plot.bar
- df.plot.box
- df.plot.hexbin
- df.plot.kde
- df.plot.pie

You can also just call df.plot(kind='hist') or replace that kind argument with any of the key terms shown in the list above (e.g. 'box','barh', etc..) ____

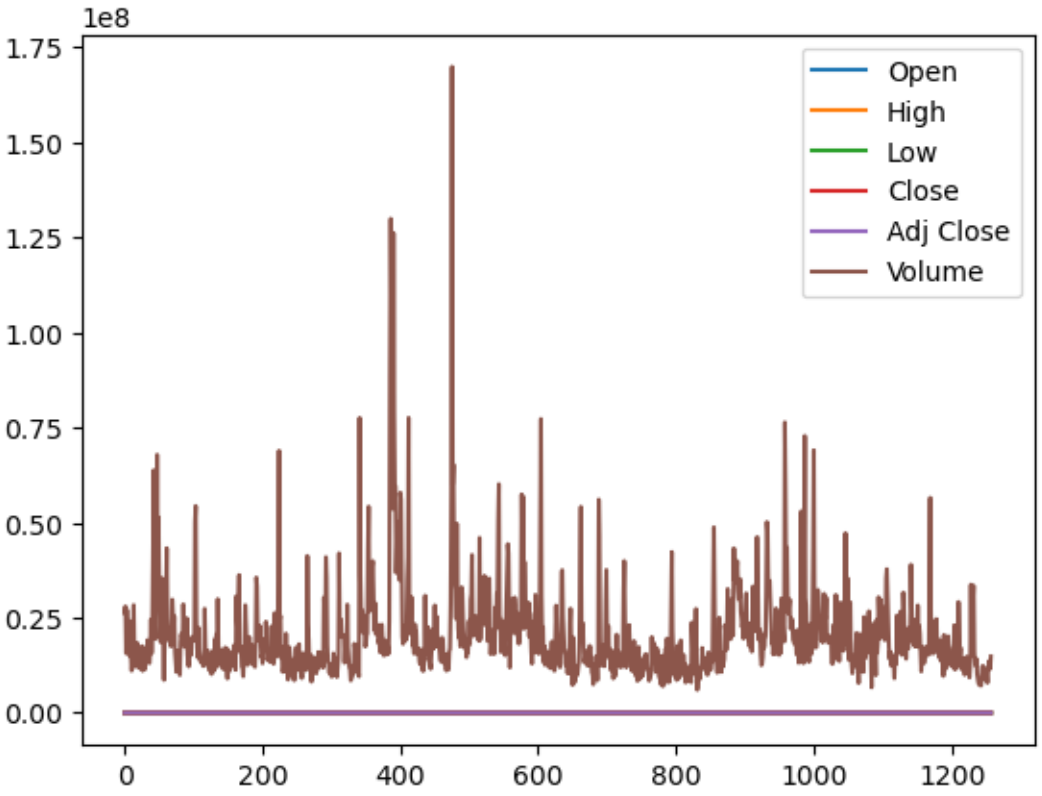
Let's start going through the main plot types we use.

3 Continuous Plots

3.1 Line Plot (The Default .plot() call)

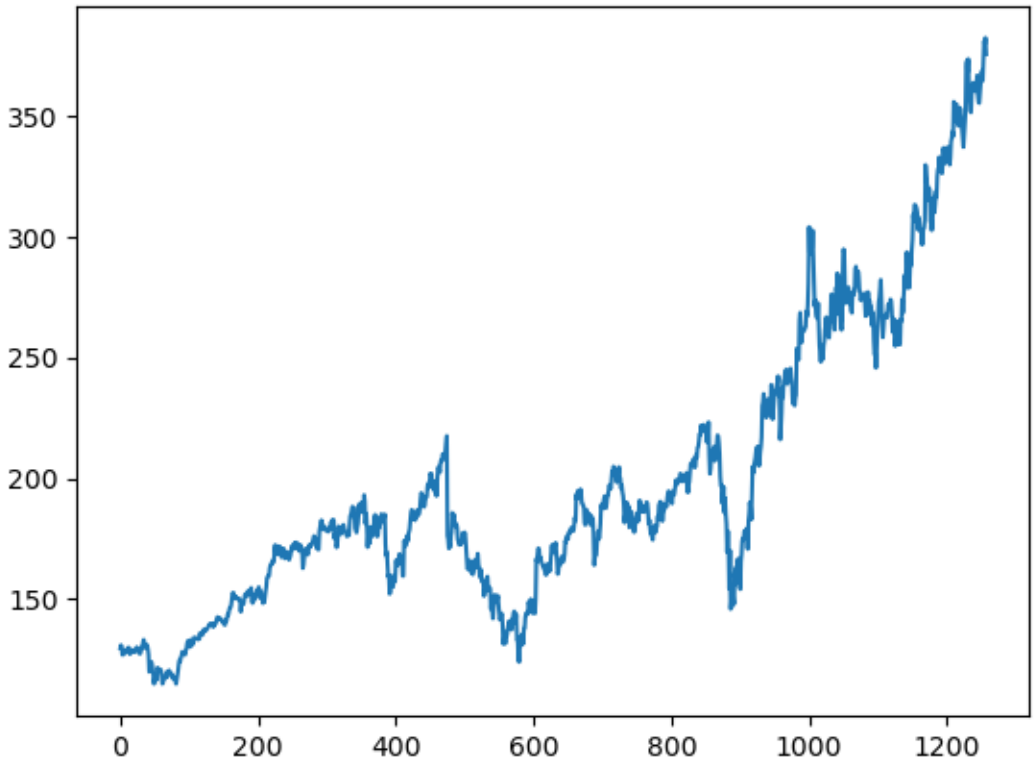
[32]:
```
df.plot() # could be an error if data doesnt make sense!
```

[32]: <Axes: >



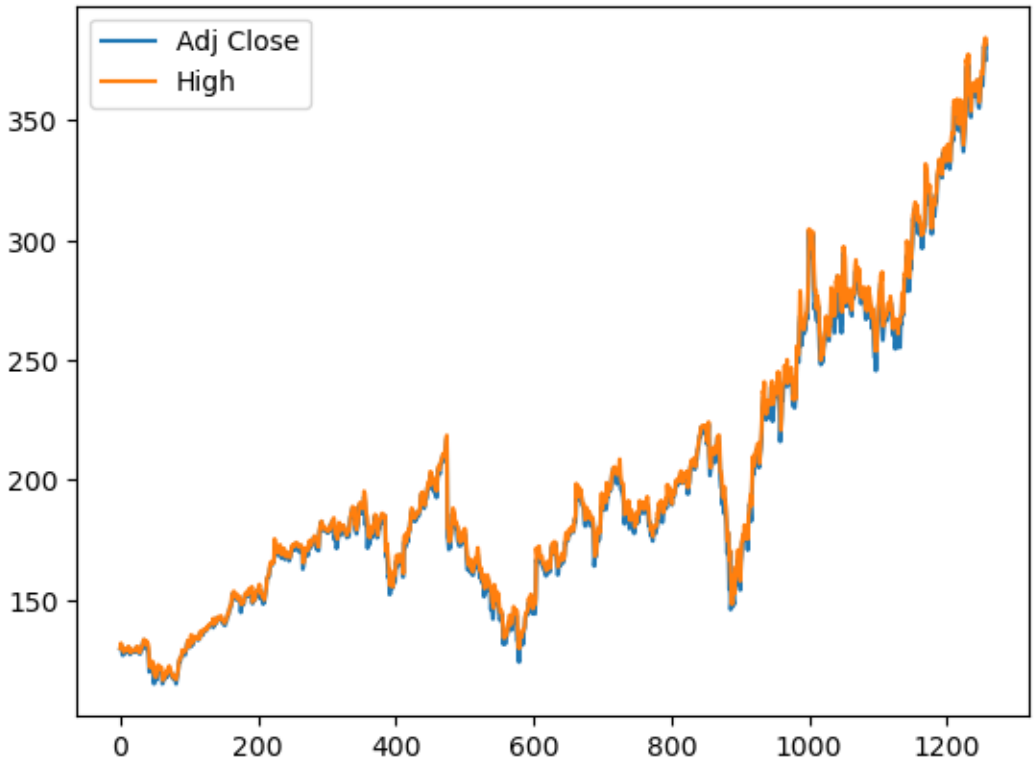
[33]:
```
# Notice the index of the df is still numerical!
# Also notice no legend if you only plot a series
df['Adj Close'].plot()
```

[33]: <Axes: >



[34]:
```
df[['Adj Close','High']].plot()
```

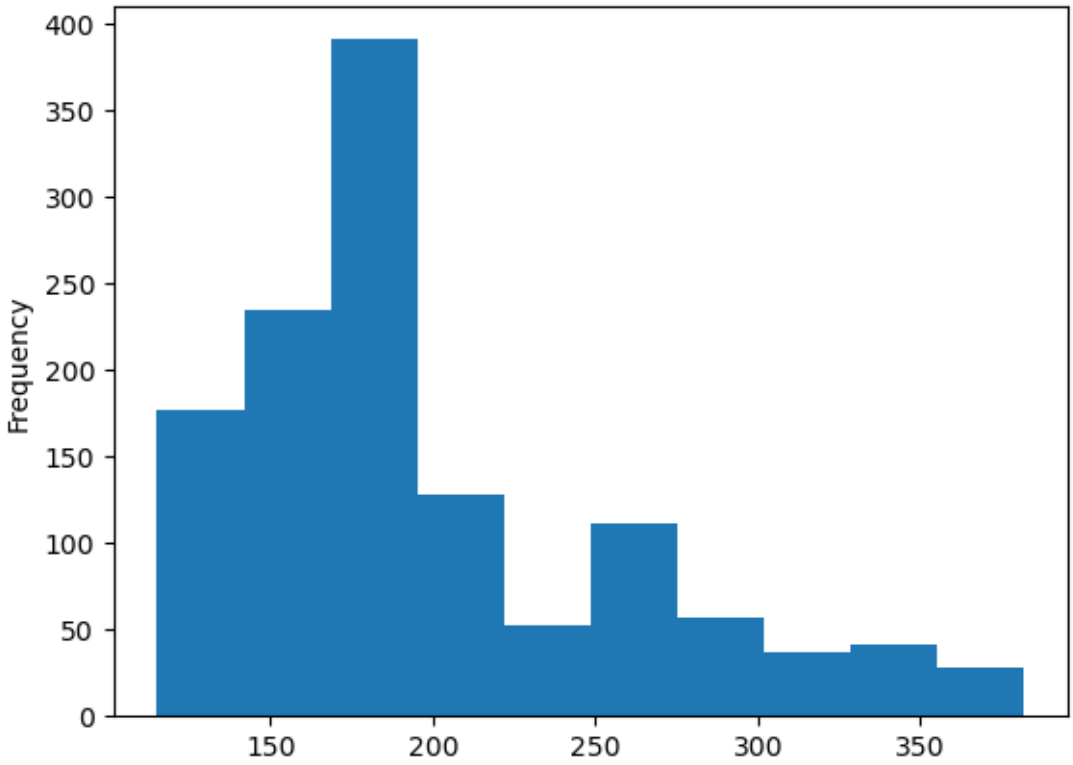
[34]: <Axes: >



3.2 Distribution Plots

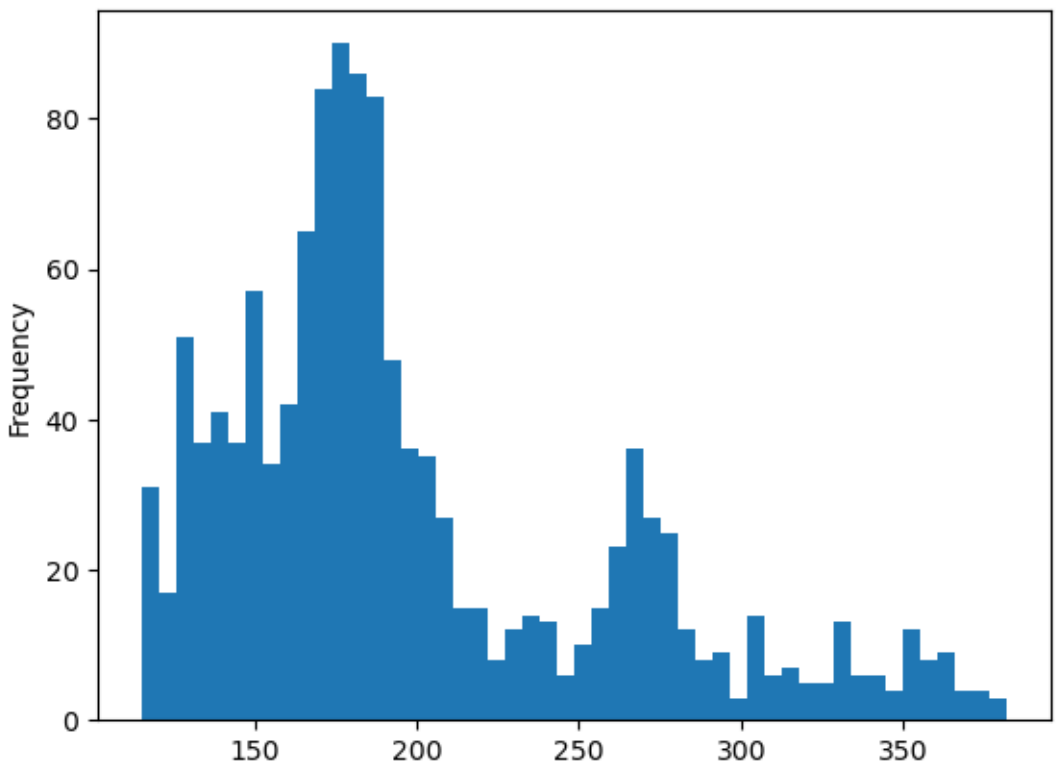
[35]:
```
df['Close'].plot(kind='hist')
```

[35]:
```
<Axes: ylabel='Frequency'>
```



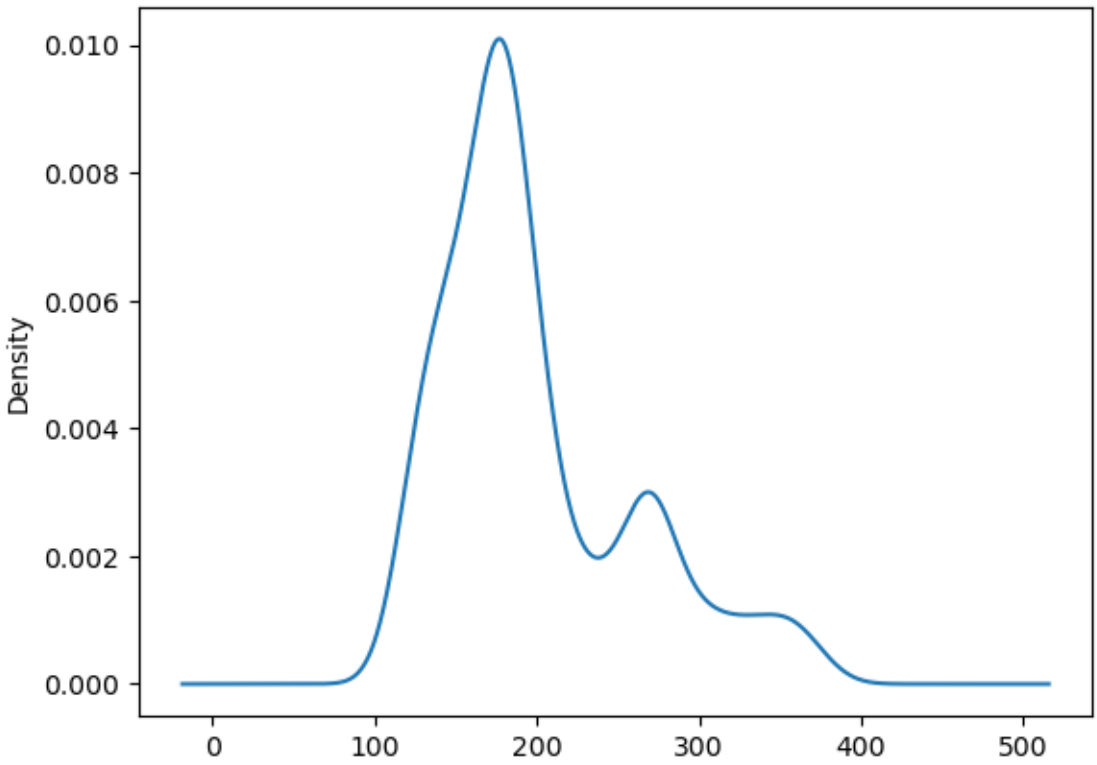
[38]:
```
df['Close'].plot(kind='hist',bins=50)
```

[38]: <Axes: ylabel='Frequency'>



[10]:
```
df['Close'].plot(kind='kde')
```

[10]: <Axes: ylabel='Density'>



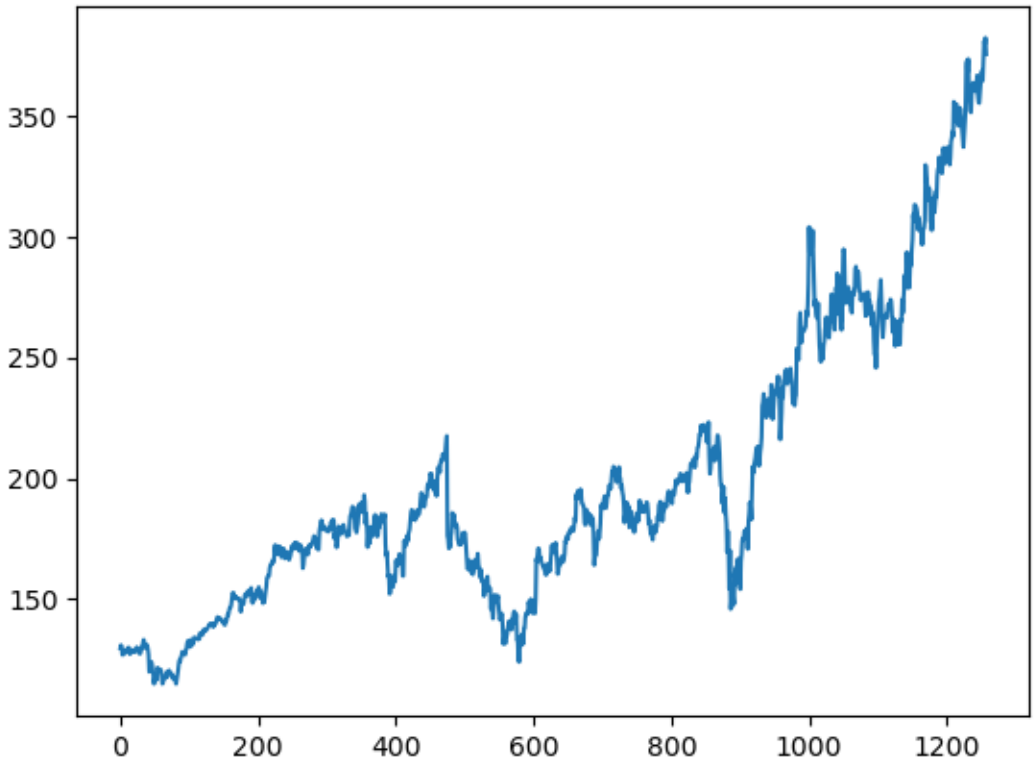
4 Editing Pandas Plots

4.1 Size and DPI

[39]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

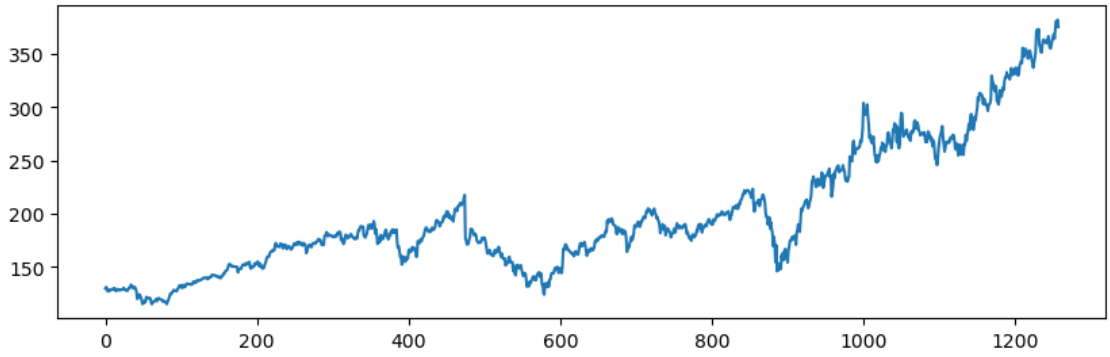
[40]:
```
df['Adj Close'].plot()
```

[40]: <Axes: >



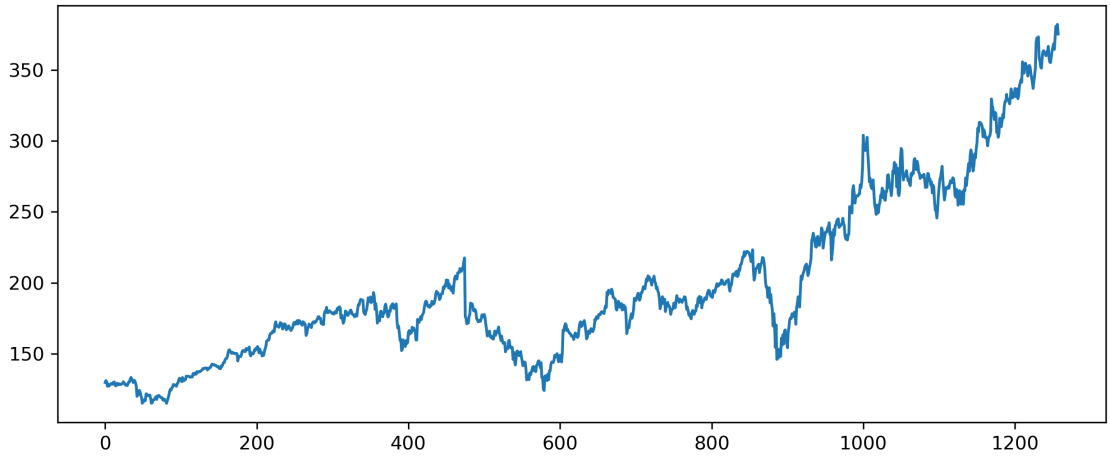
[41]:
```
df['Adj Close'].plot(figsize=(10,3))
```

[41]: <Axes: >



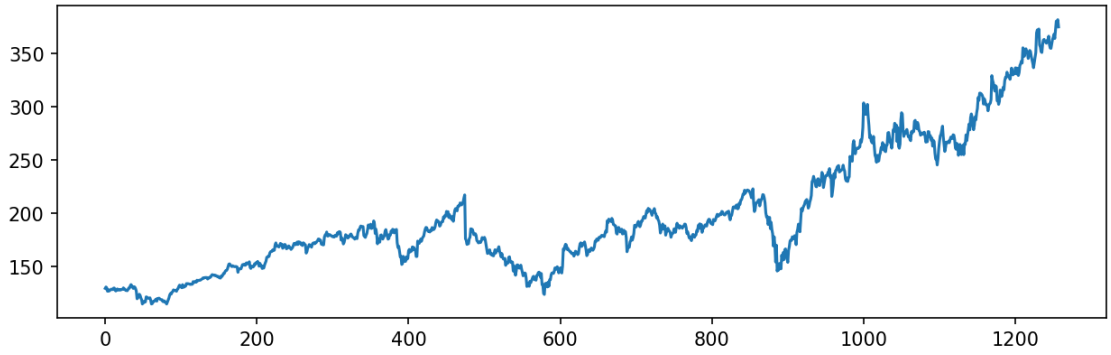
[44]:
```
# You should make sure these lines are in the same cell
plt.figure(figsize=(10,4),dpi=300)
df['Adj Close'].plot()
```

[44]: <Axes: >



[45]:
```
# You should make sure these lines are in the same cell
plt.figure(figsize=(10,3),dpi=150)
df['Adj Close'].plot()
```

[45]: <Axes: >

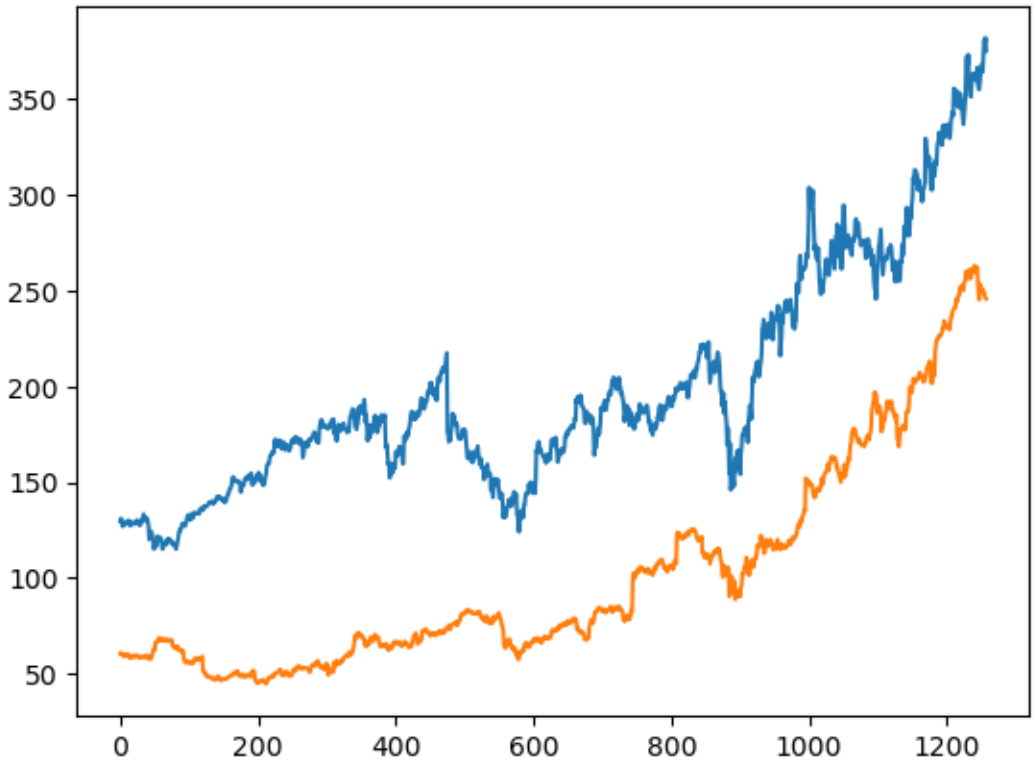


4.1.1 Connecting Plots with Matplotlib

[46]:
```
new_df = pd.read_csv('TGT.csv')
```

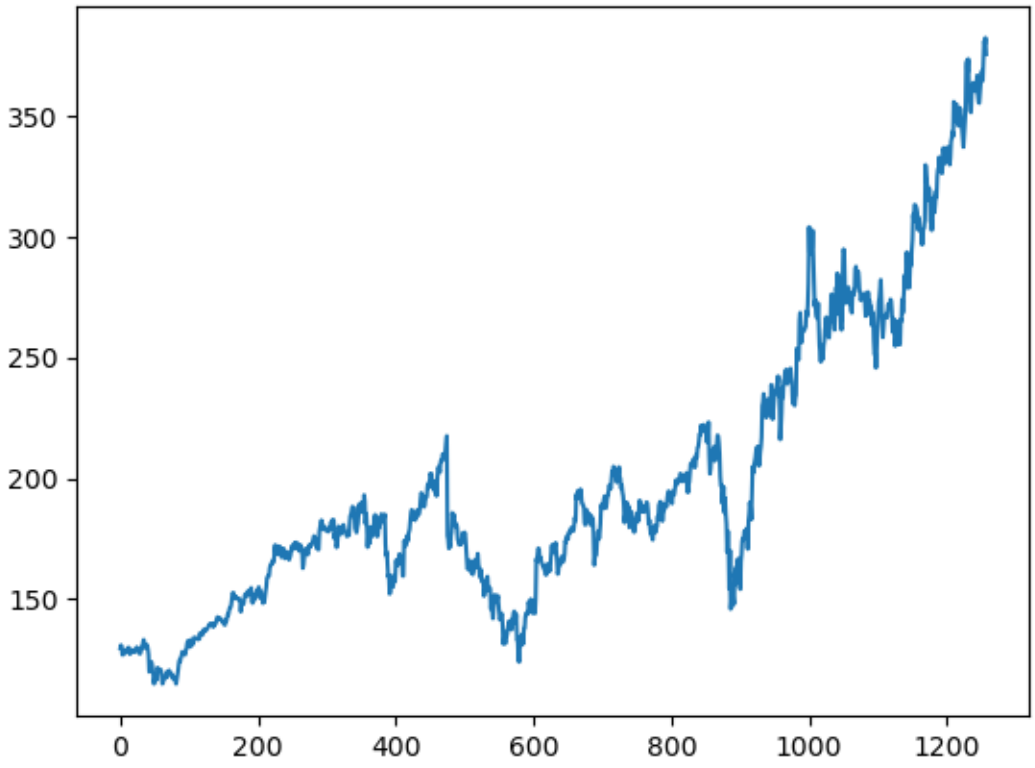
[47]:
```
df['Adj Close'].plot()
new_df['Adj Close'].plot()
```

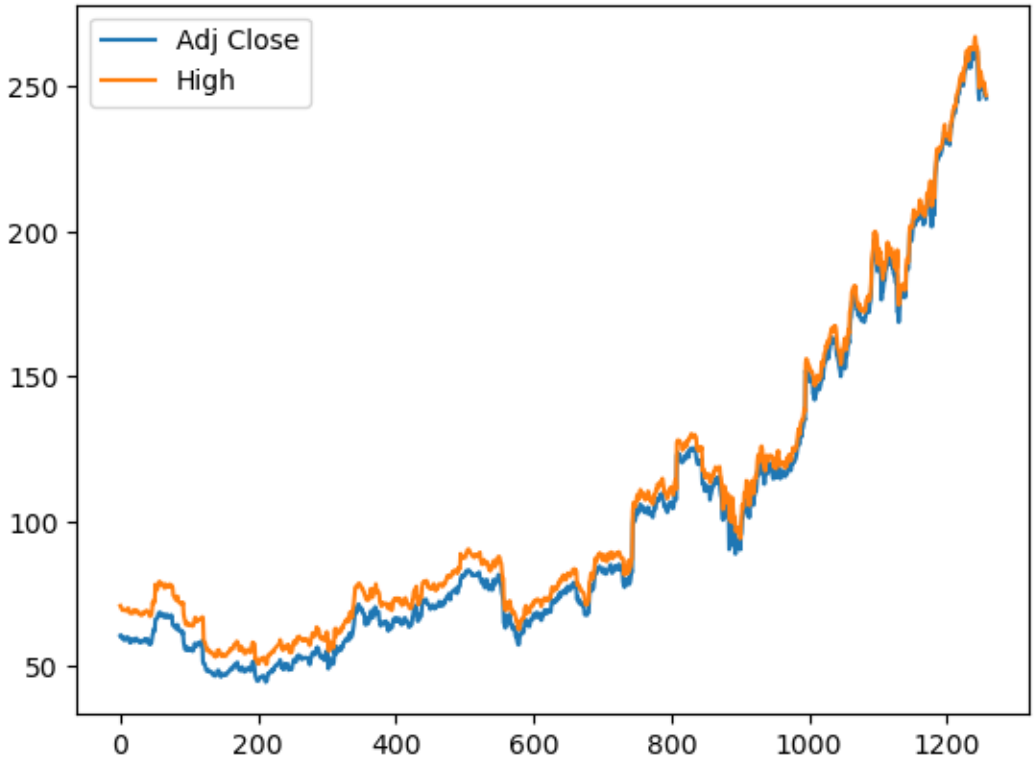
[47]: <Axes: >



[18]:
```
df['Adj Close'].plot()
new_df[['Adj Close','High']].plot()
```

[18]: <Axes: >

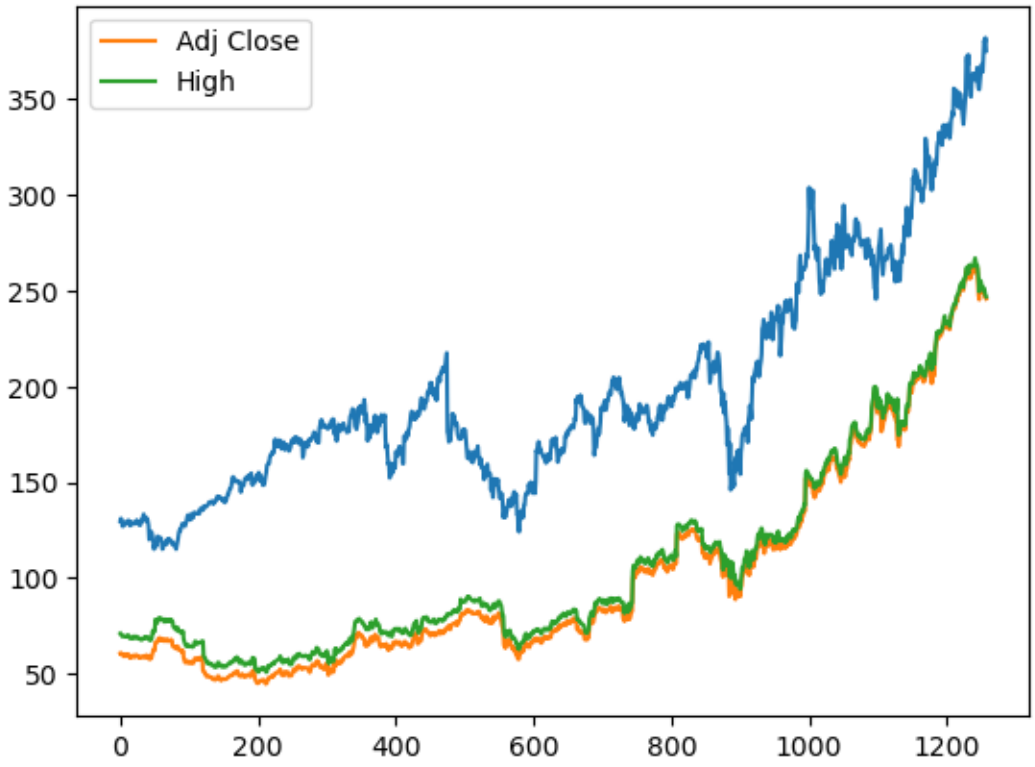

350
300
250
200
150
0
200
400
600
800
1000
1200


Adj Close
High
250
200
150
100
50
0
200
400
600
800
1000
1200

Connect them with the axes of the matplotlib figure created behind the scenes.

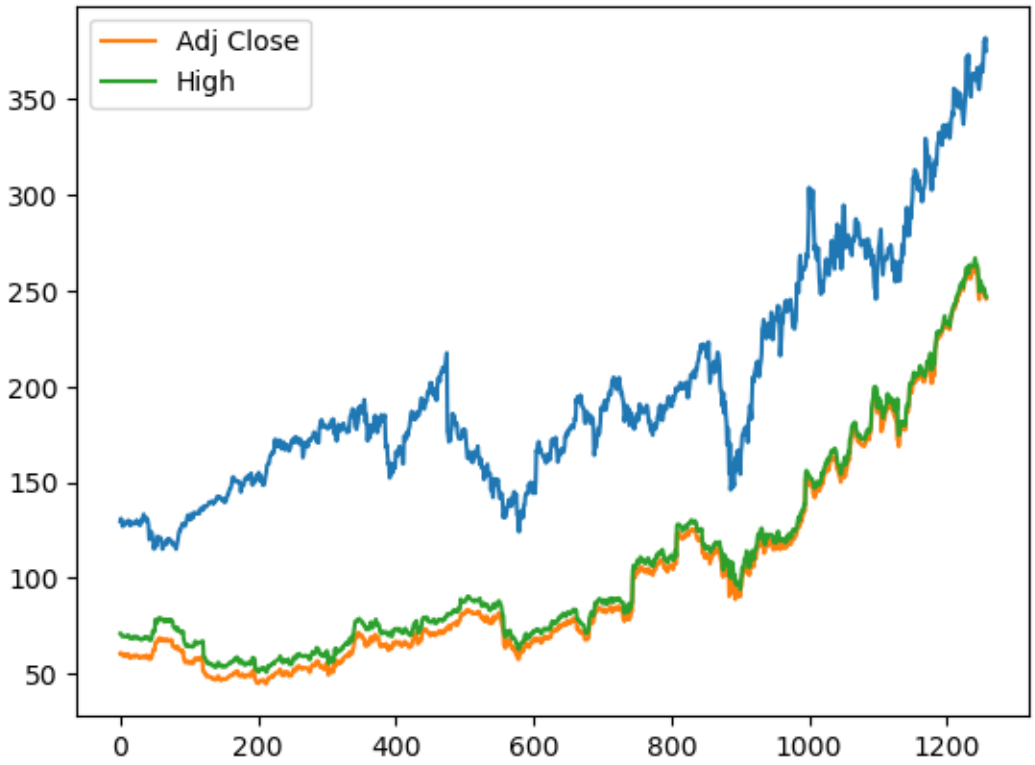
[52]:
```
fig,ax1 = plt.subplots()
df['Adj Close'].plot(ax=ax1)
new_df[['Adj Close','High']].plot(ax=ax1)
```

[52]: <Axes: >



[23]:
```
ax2 = df['Adj Close'].plot()
new_df[['Adj Close','High']].plot(ax=ax2)
```

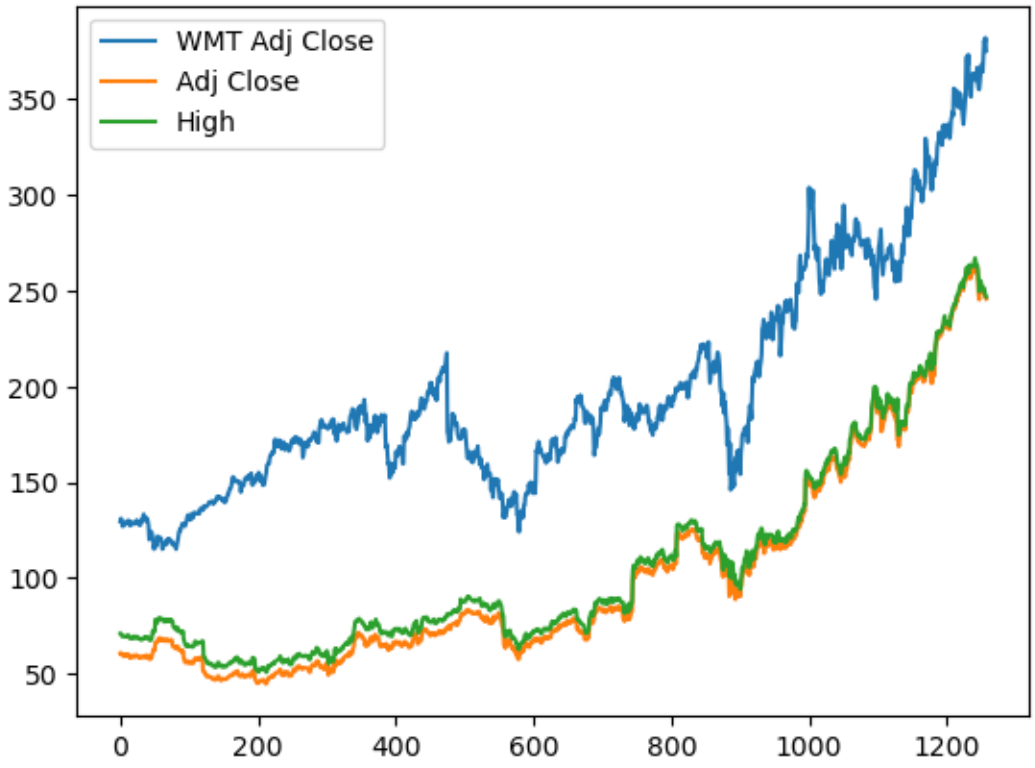
[23]: <Axes: >



4.1.2 Legends and Labels

[59]:
```
fig,ax = plt.subplots()
df['Adj Close'].plot(ax=ax,label='WMT Adj Close')
new_df[['Adj Close','High']].plot(ax=ax)
plt.legend()
```

[59]: <matplotlib.legend.Legend at 0x7fb41c0e6dd0>

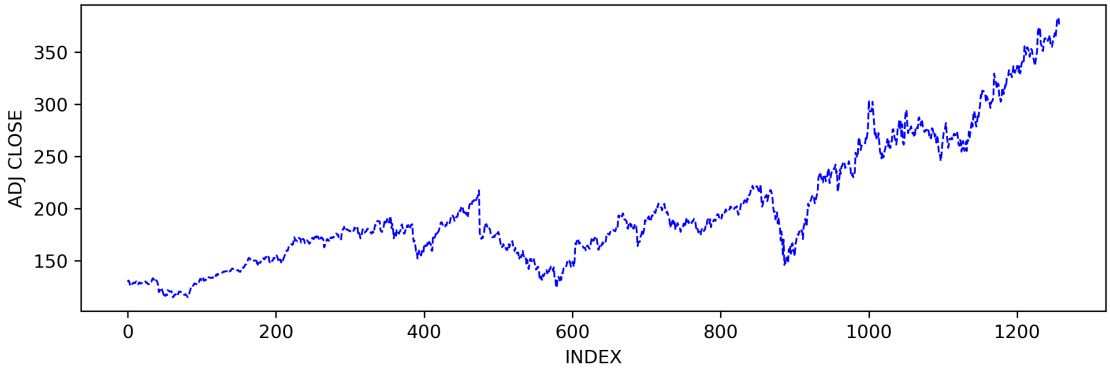


Remember, sometimes its easier to just combine everything together in pandas dataframes first, then plot from a single dataframe, rather than making major adjustments through matplotlib!

4.2 Styling

Many matplotlib styling calls are available inside the .plot() command, which is why you see ***args** in the documentation, which goes back into matplotlib plot type being called, for example:

[62]:
```
plt.figure(dpi=300)
df['Adj Close'].plot(figsize=(10,3),lw='1',ls='--',color='blue',
                     xlabel='INDEX',ylabel='ADJ CLOSE');
```



Keep in mind, it sometimes may not work for esoteric style calls, so you may have to revert back to matplotlib, think of this more as a tool for quickly making plots, rather than a tool for creating robust and unique plots.

4.2.1 Saving Pandas plots

[26]:
```
plt.figure(dpi=300)
df['Adj Close'].plot(figsize=(10,3),lw='2',ls='--',color='red',
                     xlabel='INDEX',ylabel='ADJ CLOSE');
plt.savefig('my_pandas_plot.png')
```