Peeking at data with head, tail, and describe

INTERMEDIATE PYTHON FOR FINANCE

Kennedy Behrman
Data Engineer, Author, Founder

Understanding your data

- Data is loaded correctly
- Understand the data's shape

First look at data

```
aapl
```

First look at data

```
aapl
```

Date
03/27/2020
03/26/2020
03/25/2020
03/24/2020

First look at data

```
aapl
```

	Price
Date	
03/27/2020	247.74
03/26/2020	258.44
03/25/2020	245.52
03/24/2020	246.88

First look at data

```
aapl
```

	Price	Volume
Date		
03/27/2020	247.74	51054150
03/26/2020	258.44	63140170
03/25/2020	245.52	75900510
03/24/2020	246.88	71882770

First look at data

```
aapl
```

	Price	Volume	Trend
Date			
03/27/2020	247.74	51054150	Down
03/26/2020	258.44	63140170	Up
03/25/2020	245.52	75900510	Down
03/24/2020	246.88	71882770	Up

Head

```
aapl.head()
```

```
             Price    Volumne    Trend
Date
03/27/2020   247.74   51054150   Down
03/26/2020   258.44   63140170   Up
03/25/2020   245.52   75900510   Down
03/24/2020   246.88   71882770   Up
03/23/2020   224.37   84188210   Down
```

Head

```
aapl.head()
```

Head

```
aapl.head(3)
```

~~~
```out
              Price     Volumne     Trend
Date
03/27/2020    247.74    51054150    Down
03/26/2020    258.44    63140170    Up
03/25/2020    245.52    75900510    Down
~~~

Tail

```
aapl.tail()
```

```
            Price     Volumne    Trend
Date
03/05/2020  292.92    46893220   Down
03/04/2020  302.74    54794570   Up
03/03/2020  289.32    79868850   Down
03/02/2020  298.81    85349340   Up
02/28/2020  273.36    106721200  Down
```

Describe

```
aapl.describe()
```

```
       Price       Volume
count  21.000000   2.100000e+01
mean   263.715714  7.551468e+07
std    23.360598   1.669757e+07
min    224.370000  4.689322e+07
25%    246.670000  6.409497e+07
50%    258.440000  7.505841e+07
75%    285.340000  8.418821e+07
max    302.740000  1.067212e+08
```

Include

describe() takes three option arguments, include, exclude, and percentiles.

```
aapl.describe(include='object')
```

object-type columns

```
        Trend
count   21
unique  2
top     Down
freq    14
```

highest count

Include

```
aapl.describe(include='all')
```

```
         Price       Volumne         Trend
count    21.000000   2.100000e+01    21
unique   NaN         NaN             2
top      NaN         NaN             Down
freq     NaN         NaN             14
mean     263.715714  7.551468e+07    NaN
std      23.360598   1.669757e+07    NaN
min      224.370000  4.689322e+07    NaN
25%      246.670000  6.409497e+07    NaN
```

```
aapl.describe(include=['float', 'object'])
```

float, int, object

```
        Price       Trend
count   21.000000   21
unique  NaN         2
top     NaN         Down
freq    NaN         14
mean    263.715714  NaN
std     23.360598   NaN
min     224.370000  NaN
25%     246.670000  NaN
50%     258.440000  NaN
75%     285.340000  NaN
max     302.740000  NaN
```

Percentiles

```
aapl.describe(percentiles=[.1, .5, .9])
```

```
      Price       Volumne
count 21.000000   2.100000e+01
mean  263.715714  7.551468e+07
std   23.360598   1.669757e+07
min   224.370000  4.689322e+07
10%   242.210000  5.479457e+07
50%   258.440000  7.505841e+07
90%   292.920000  1.004233e+08
max   302.740000  1.067212e+08
```

Exclude

```
aapl.describe(exclude='float')
```

```
        Volumne       Trend
count   2.100000e+01  21
unique NaN            2
top     NaN           Down
freq    NaN           14
mean    7.551468e+07  NaN
std     1.669757e+07  NaN
min     4.689322e+07  NaN
25%     6.409497e+07  NaN
```

Let's practice!

INTERMEDIATE PYTHON FOR FINANCE

Filtering data

INTERMEDIATE PYTHON FOR FINANCE

Kennedy Behrman
Data Engineer, Author, Founder

Introducing the data

```
prices.head()
```

Introducing the data

```
prices.head()
```

	Date	Symbol	High
0	2020-04-03	AAPL	245.70
1	2020-04-02	AAPL	245.15
2	2020-04-01	AAPL	248.72
3	2020-03-31	AAPL	262.49
4	2020-03-30	AAPL	255.52

Introducing the data

```
prices.describe()
```

Introducing the data

```
prices.describe()
```

	High
count	378.000000
mean	881.593138
std	720.771922
min	227.490000
max	2185.950000

Introducing the data

```
prices.describe(include='object')
```

	Symbol
count	378
unique	3
top	AMZN
freq	126

Comparison operators

`<` `<=` `>` `>=` `==` `!=`

Column comparison

```
prices.High > 2160
```

Column comparison

```
prices.High > 2160
```

```
0       False
1       False
2       False
3       False
4       False
          ...
374     False
375     False
376     False
377     False
```

Column comparison

```
prices.Symbol == 'AAPL'
```

Column comparison

```
prices.Symbol == 'AAPL'
```

```
0       True
1       True
2       True
3       True
4       True
       ...
374    False
375    False
376    False
377    False
```

Masking by symbol

```
mask_symbol = prices.Symbol == 'AAPL'
aapl = prices.loc[mask_symbol]
```

Remember that when we pass a sequence of boolean values to a DataFrame’s loc[] operator, a new DataFrame is returned containing only the rows matching the True.

Masking by symbol

```
mask_symbol = prices.Symbol == 'AAPL'
aapl = prices.loc[mask_symbol]
aapl.describe(include='object')
```

	Symbol
count	126
unique	1
top	AAPL
freq	126

Masking by price

```
mask_high = prices.High > 2160
big_price = prices.loc[mask_high]
```

Masking by price

```
big_price.describe()
```

	High
count	6.000000
mean	2177.406567
std	7.999334
min	2166.070000
max	2185.95000

Pandas boolean operators

- And `&`
- Or `|`
- Not `~`

Combining conditions

```
mask_prices = prices['Symbol'] != 'AMZN'
```

```
mask_date = historical_highs['Date'] > datetime(2020, 4, 1)
```

```
mask_amzn = mask_prices & mask_date
```

```
prices.loc[mask_amzn]
```

Combining conditions

	Date	Symbol	High
0	2020-04-03	AAPL	245.7000
1	2020-04-02	AAPL	245.1500
252	2020-04-03	TSLA	515.4900
253	2020-04-02	TSLA	494.2599

Let's practice!

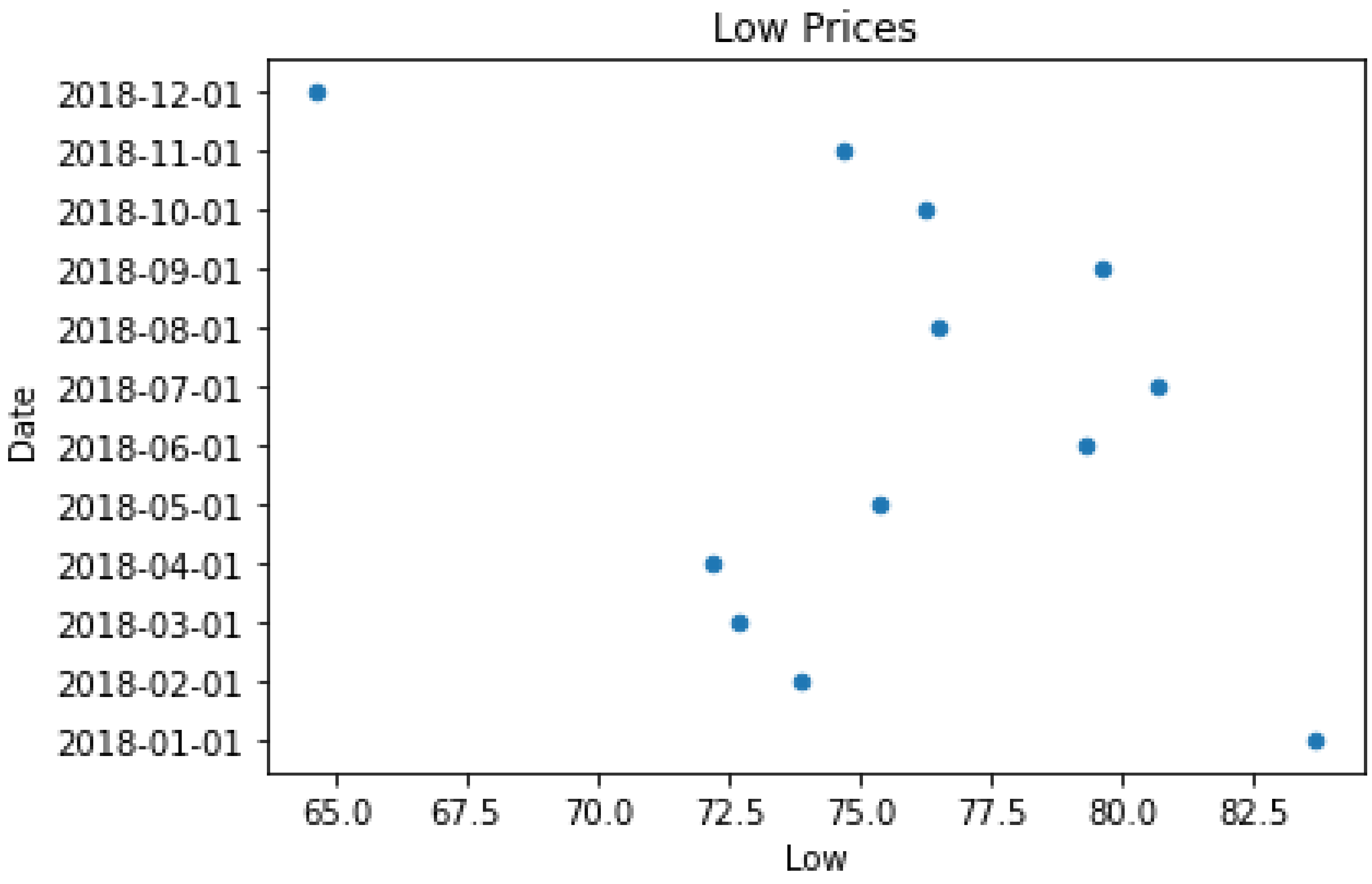
INTERMEDIATE PYTHON FOR FINANCE

Plotting data

INTERMEDIATE PYTHON FOR FINANCE

Kennedy Behrman
Data Engineer, Author, Founder

Look at your data



```
exxon.head()
```

Introducing the data

```
exxon.head()
```

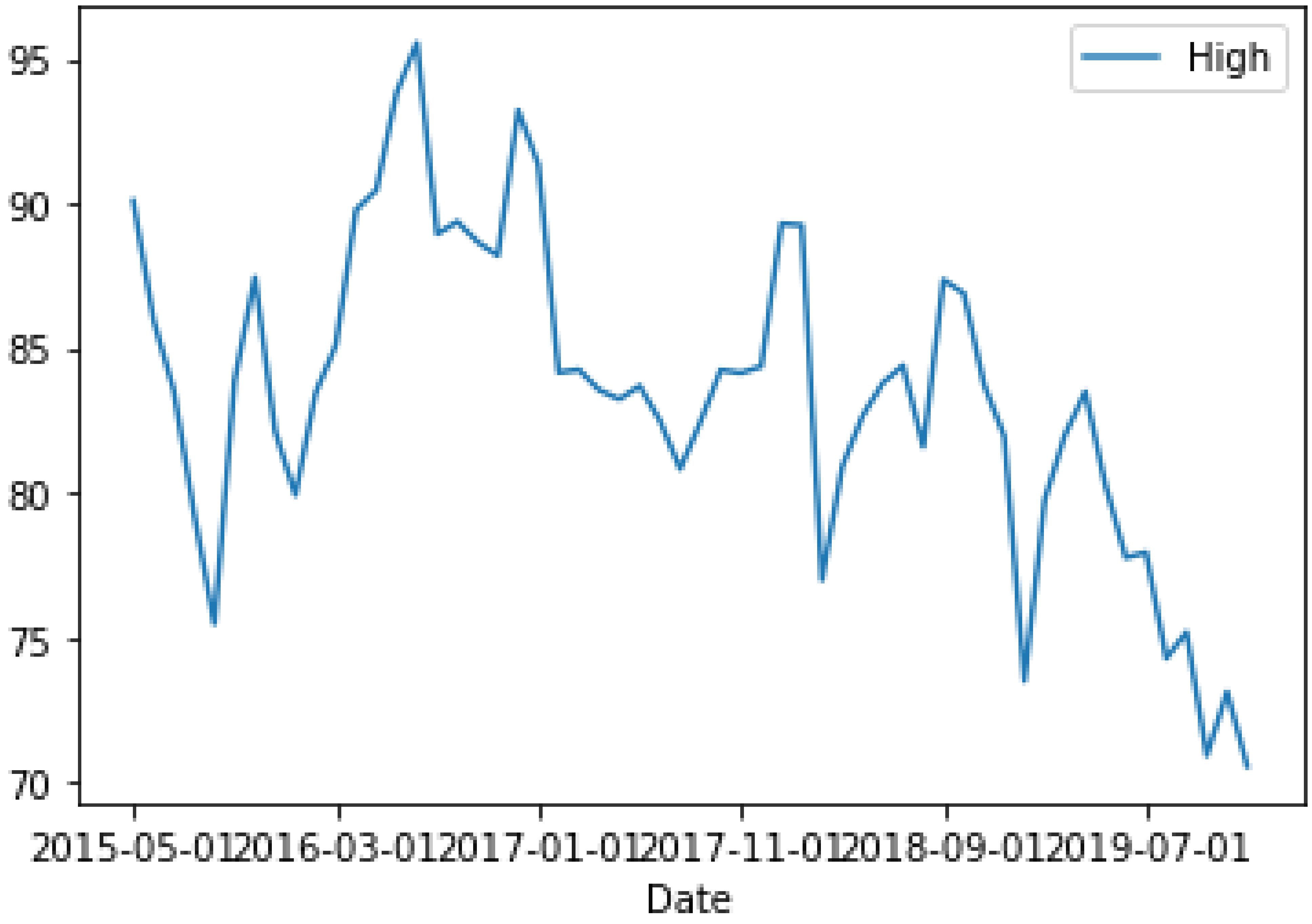
```
    Date        High       Volume     Month
0   2015-05-01  90.089996  198924100  May
1   2015-06-01  85.970001  238808600  Jun
2   2015-07-01  83.529999  274029000  Jul
3   2015-08-01  79.290001  387523600  Aug
4   2015-09-01  75.470001  316644500  Sep
```

Matplotlib

```
my_dataframe.plot()
```

Line plot

```
exxon.plot(x='Date',
           y='High' )
```

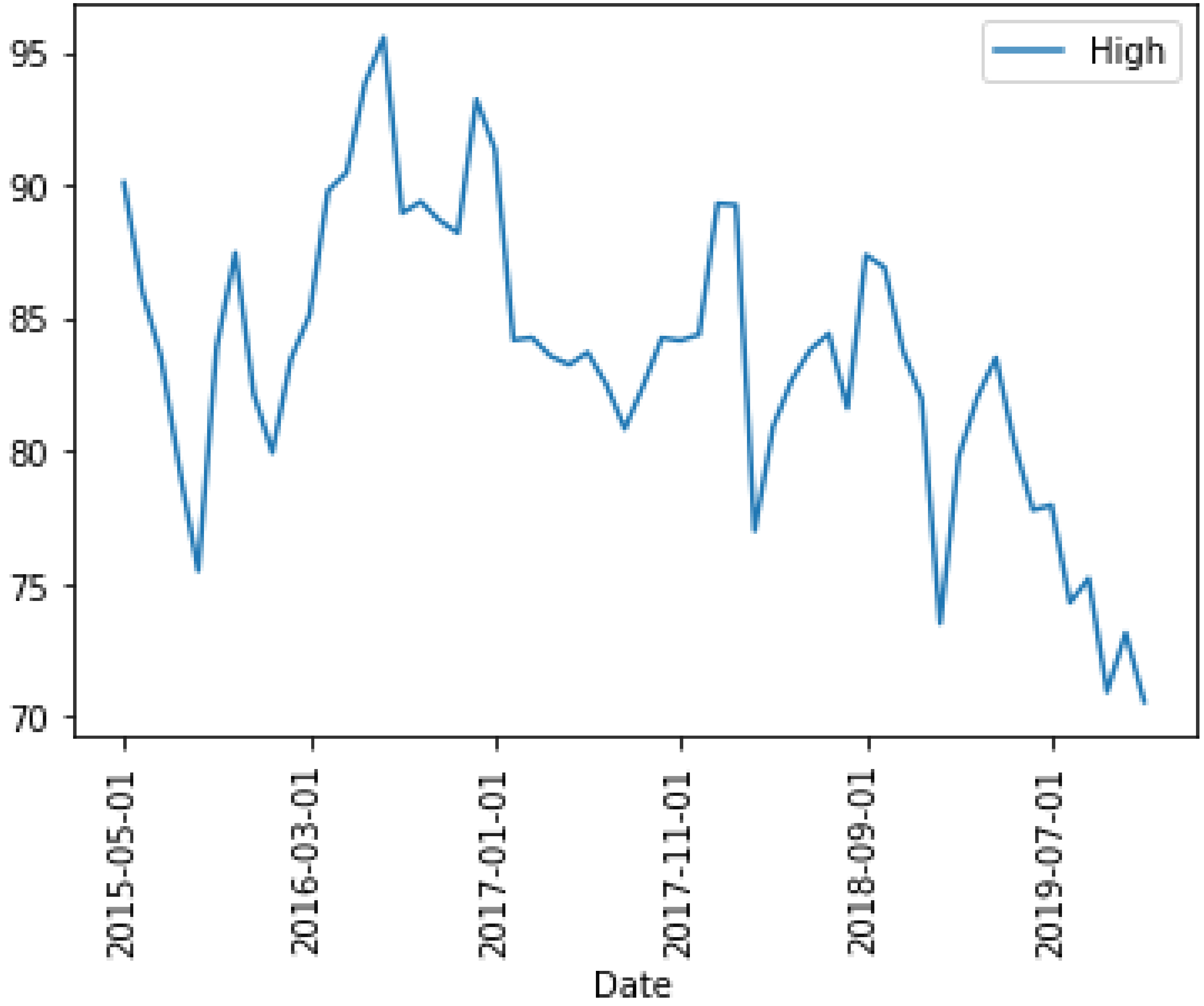


High
95
90
85
80
75
70
2015-05-01
2016-03-01
2017-01-01
2017-11-01
2018-09-01
2019-07-01
Date

Rotate

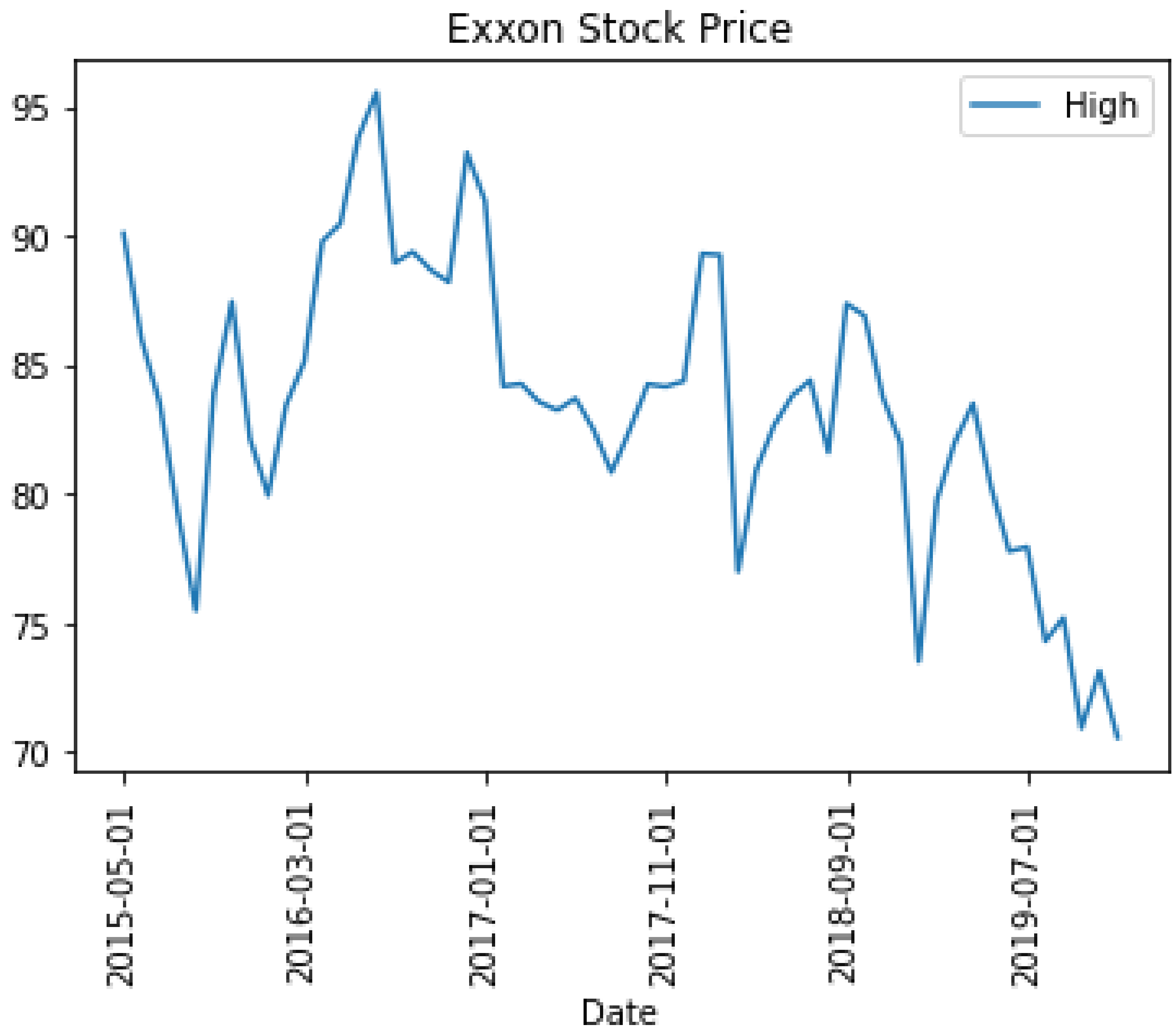
```
exxon.plot(x='Date',
           y='High',
           rot=90 )
```

the rotation of the labels


95
90
85
80
75
70
High
2015-05-01
2016-03-01
2017-01-01
2017-11-01
2018-09-01
2019-07-01
Date

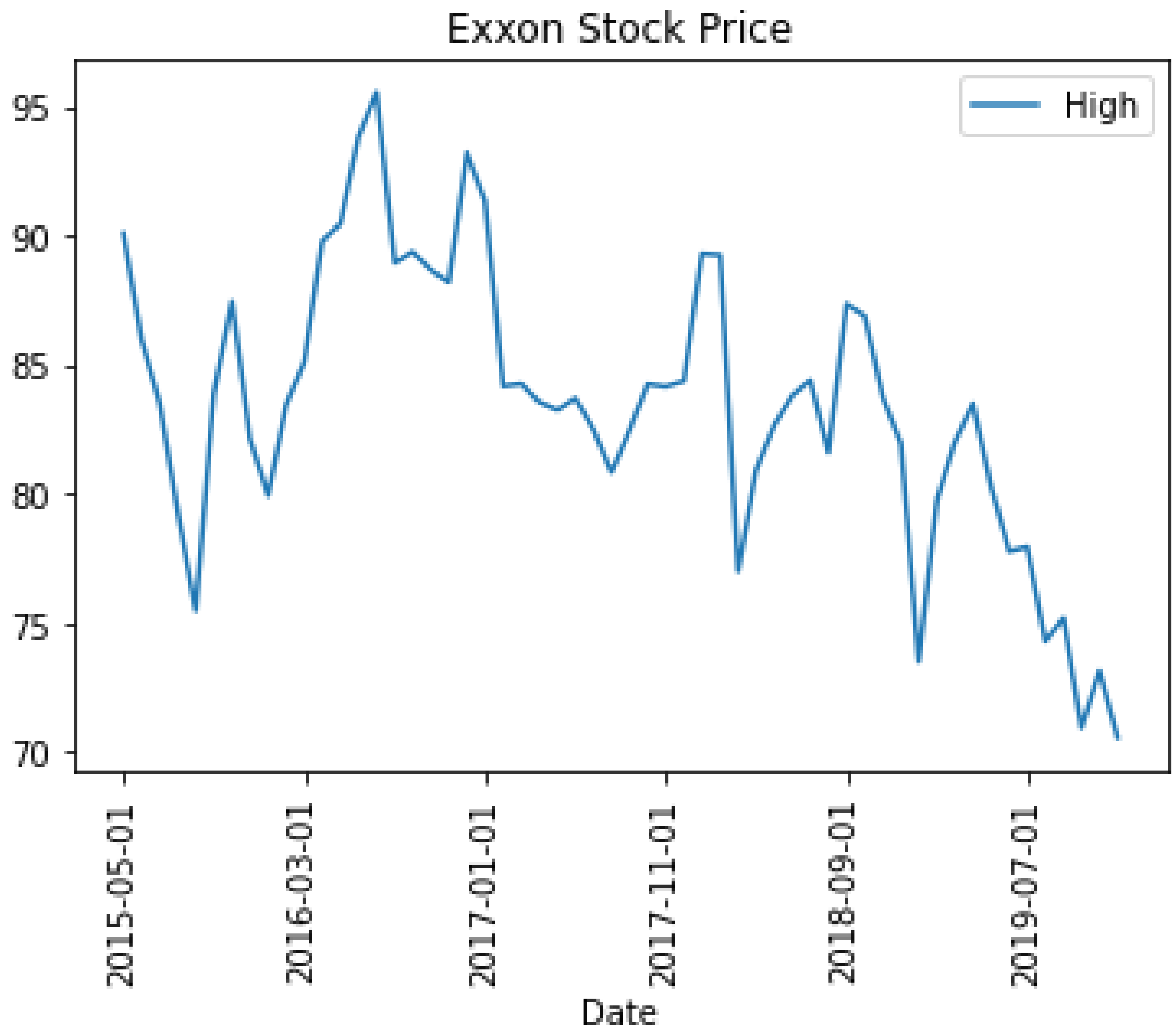
Title

```
exxon.plot(x='Date',
           y='High',
           rot=90,
           title='Exxon Stock Price')
```


Exxon Stock Price
High
95
90
85
80
75
70
2015-05-01
2016-03-01
2017-01-01
2017-11-01
2018-09-01
2019-07-01
Date

Index

```
exxon.set_index('Date', inplace=True)
exxon.plot(y='High',
           rot=90,
           title='Exxon Stock Price')
```

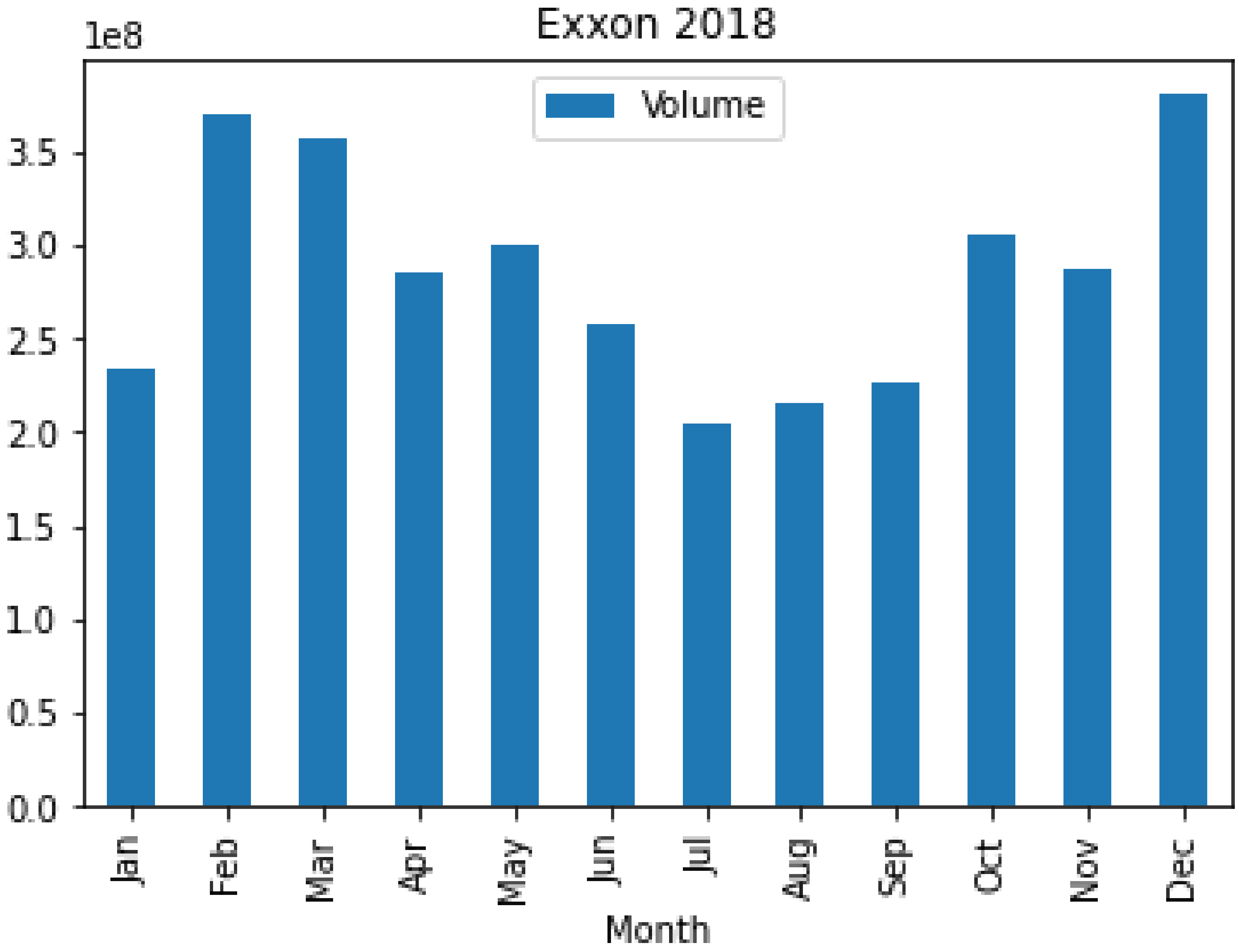

Exxon Stock Price
High
95
90
85
80
75
70
2015-05-01
2016-03-01
2017-01-01
2017-11-01
2018-09-01
2019-07-01
Date

Plot types

- `line`
- `bar`
- `barh`
- `hist`
- `box`
- `kde`
- `density`
- `area`
- `pie`
- `scatter`
- `hexbin`

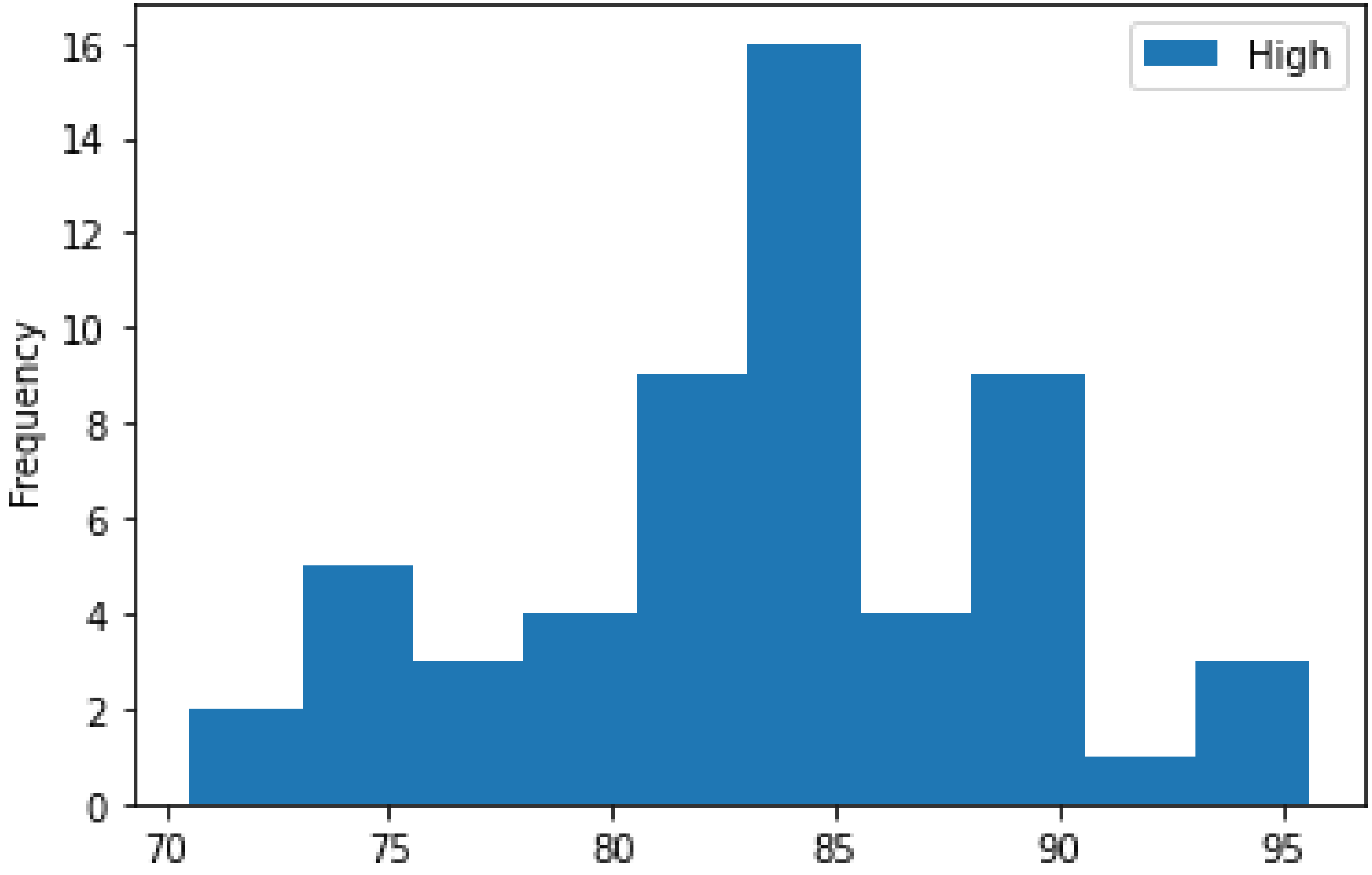
Bar

```
exxon2018.plot(x='Month',
               y='Volume',
               kind='bar',
               title='Exxon 2018')
```


Exxon 2018
1e8
Volume
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month

Hist

```
exxon.plot(y='High',kind='hist')
```



High
Frequency
0
2
4
6
8
10
12
14
16
70
75
80
85
90
95

Let's practice!

Wrapping up

INTERMEDIATE PYTHON FOR FINANCE

Kennedy Behrman
Data Engineer, Author, Founder

Chapter 1

- Representing time

`datetime`

- Mapping data

`dict()`

Chapter 2

- Comparison operators

```
< <= > >=
```

- Equality operators

```
== !=
```

- Boolean operators

```
and or not
```

- If statements

```
if a < b:
    print(a)
```

- Loops

```
while a < b:
      a = a + 1
```

```
for a in c:
    print(a)
```

Chapter 3

- Creating a DataFrame

```
DataFrame(data=data)
pd.read_csv('/data.csv')
```

- Accessing data

```
stocks.loc['a', 'Values']
stocks.iloc[2:22, 12]
```

- Aggregating, summarizing

```
stocks.mean()
stocks.median()
```

- Extending, manipulating

```
pce['PCESV'] = pcesv
gdp.apply(np.sum, axis=1)
```

Chapter 4

- Peeking

```
aapl.head()
aapl.tail()
aapl.describe()
```

- Filtering

```
mask = prices.High > 216
prices.loc[mask]
```

- Plotting

```
exxon.plot(x='Date',
           y='High' )
```

Congratulations!

INTERMEDIATE PYTHON FOR FINANCE