Bit Project

Introduction to Data Visualization and Graphs with Matplotlib

Module 3

Recap

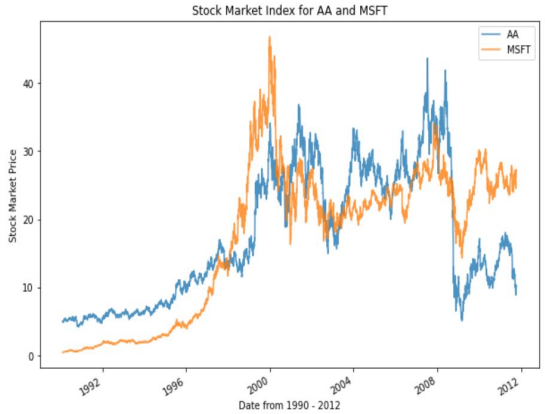
- Pandas
- Loading data into dataframe
- Cleaning data
- Indexing dataframes
- Aggregate functions

Why Visualize?

What is visualization?

- **Graphic** representation of data
- Helps us to:
  - **Explore** data
  - **Communicate** ideas

	AA	AAPL	GE	IBM	JNJ	MSFT	PEP	SPX	XOM
1990-02-01	4.98	7.86	2.87	16.79	4.27	0.51	6.04	328.79	6.12
1990-02-02	5.04	8.00	2.87	16.89	4.37	0.51	6.09	330.92	6.24
1990-02-05	5.07	8.18	2.87	17.32	4.34	0.51	6.05	331.85	6.25
1990-02-06	5.01	8.12	2.88	17.56	4.32	0.51	6.15	329.66	6.23
1990-02-07	5.04	7.77	2.91	17.93	4.38	0.51	6.17	333.75	6.33
...	...	...	...	...	...	...	...	...	...
2011-10-10	10.09	388.81	16.14	186.62	64.43	26.94	61.87	1194.89	76.28
2011-10-11	10.30	400.29	16.14	185.00	63.96	27.00	60.95	1195.54	76.27
2011-10-12	10.05	402.19	16.40	186.12	64.33	26.96	62.70	1207.25	77.16
2011-10-13	10.10	408.43	16.22	186.82	64.23	27.18	62.36	1203.66	76.37
2011-10-14	10.26	422.00	16.60	190.53	64.72	27.27	62.24	1224.58	78.11



Why visualize?

- Without visualization
  - Data is **less meaningful**
  - Data is **messy**
  - **Hard** to understand
- With visualization
  - **Patterns** become visible
  - **Meaningful** analysis

Without visualization

	AA	AAPL	GE	IBM	JNJ	MSFT	PEP	SPX	XOM
1990-02-01	4.98	7.86	2.87	16.79	4.27	0.51	6.04	328.79	6.12
1990-02-02	5.04	8.00	2.87	16.89	4.37	0.51	6.09	330.92	6.24
1990-02-05	5.07	8.18	2.87	17.32	4.34	0.51	6.05	331.85	6.25
1990-02-06	5.01	8.12	2.88	17.56	4.32	0.51	6.15	329.66	6.23
1990-02-07	5.04	7.77	2.91	17.93	4.38	0.51	6.17	333.75	6.33
...	...	...	...	...	...	...	...	...	...
2011-10-10	10.09	388.81	16.14	186.62	64.43	26.94	61.87	1194.89	76.28
2011-10-11	10.30	400.29	16.14	185.00	63.96	27.00	60.95	1195.54	76.27
2011-10-12	10.05	402.19	16.40						
2011-10-13	10.10	408.43	16.22						
2011-10-14	10.26	422.00	16.60						

THIS IS DIFFICULT TO MAKE SENSE OF!

With visualization

THIS IS EASIER TO MAKE SENSE OF!

About this Module

Goals

- **Matplotlib**
  - Understanding pyplot.plot
- **Figures and subplots**
  - Colors, Markers, and Line Styles
  - Ticks, Labels and Legends
- **Relational** graphs
  - Bar plots
  - Histograms
  - Line plots
  - Scatter plots
- Mapping **longitude and latitude**

Stock Market Index Dataset

- Stock prices of companies vs. Time

	AA (American Airlines)	AAPL (Apple)	GE (General Electric)	IBM	JNJ (Johnson & Johnson)	MSFT (Microsoft)	PEP (PepsiCo)	SPX (S&P 500)	XOM (ExxonMobil)
1990-02-01	4.98	7.86	2.87	16.79	4.27	0.51	6.04	328.79	6.12
1990-02-02	5.04	8.00	2.87	16.89	4.37	0.51	6.09	330.92	6.24
1990-02-05	5.07	8.18	2.87	17.32	4.34	0.51	6.05	331.85	6.25
1990-02-06	5.01	8.12	2.88	17.56	4.32	0.51	6.15	329.66	6.23
1990-02-07	5.04	7.77	2.91	17.93	4.38	0.51	6.17	333.75	6.33
...	...	...	...	...	...	...	...	...	...

California Housing Dataset

- 1990 California Census
- Info about houses in a CA district

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0
...	...	...	...	...	...	...

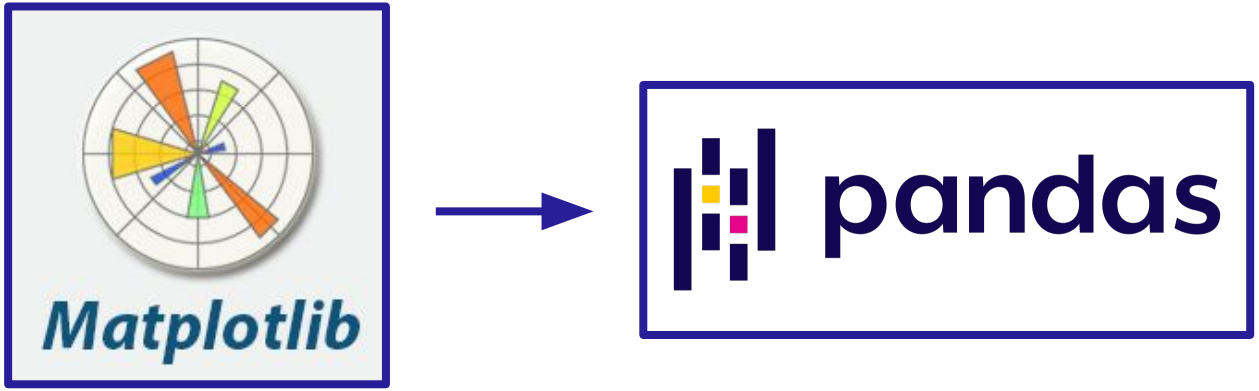
Plotting with Pandas and Matplotlib

What is Matplotlib?

- **Data visualization** library
- Works hand in hand with **Pandas**
- **Pyplot** subpackage
  - Focuses mainly on plotting
  - Designed to be like MATLAB
  - Other subpackages for other visuals
  - ‘matplotlib.pyplot’

Matplotlib and Pandas

- Pandas plotting implicitly uses Matplotlib
  - ‘df.plot()’
- **Simpler** methods and implementation
- Matplotlib is more **precise**



Importing Libraries

```
import pandas as pd

import matplotlib.pyplot as plt

import datetime
```

Loading the .csv

```
url = "github.com..."

df = pd.read_csv(url,

                 index_col = 0,

                 parse_dates = True)
```

Loading the .csv

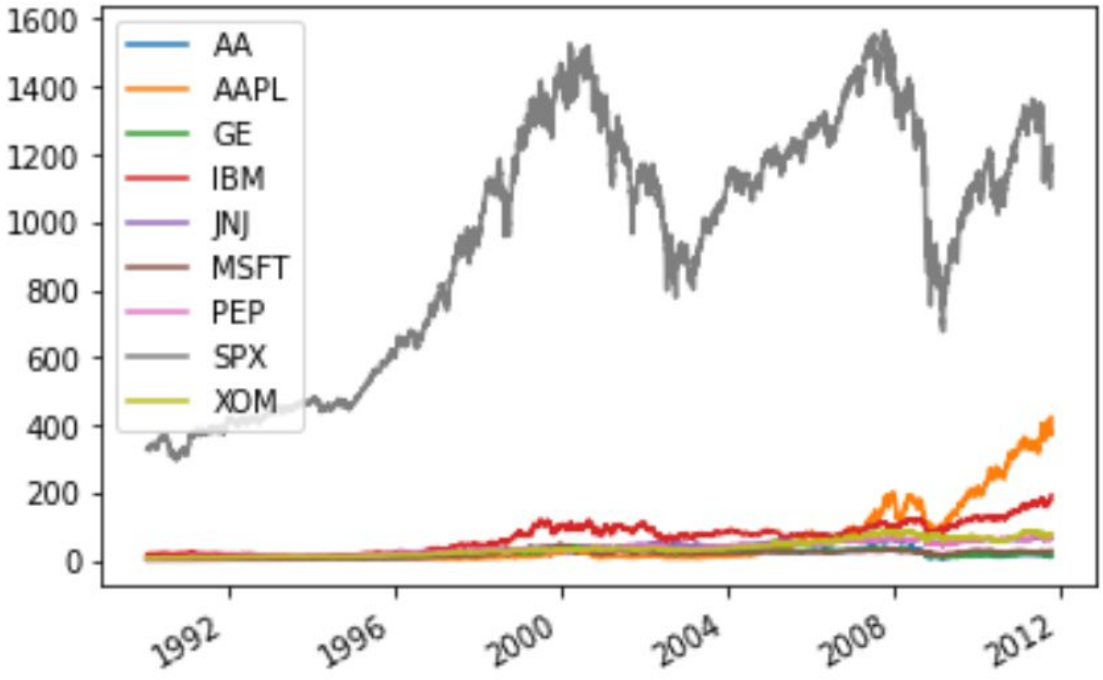
df

	AA	AAPL	GE	IBM	JNJ	MSFT	PEP	SPX	XOM
1990-02-01	4.98	7.86	2.87	16.79	4.27	0.51	6.04	328.79	6.12
1990-02-02	5.04	8.00	2.87	16.89	4.37	0.51	6.09	330.92	6.24
1990-02-05	5.07	8.18	2.87	17.32	4.34	0.51	6.05	331.85	6.25
1990-02-06	5.01	8.12	2.88	17.56	4.32	0.51	6.15	329.66	6.23
1990-02-07	5.04	7.77	2.91	17.93	4.38	0.51	6.17	333.75	6.33
...	...	...	...	...	...	...	...	...	...

Using df.plot()

```
df.plot()
```

Result


1600
1400
1200
1000
800
600
400
200
0
AA
AAPL
GE
IBM
JNJ
MSFT
PEP
SPX
XOM
1992
1996
2000
2004
2008
2012

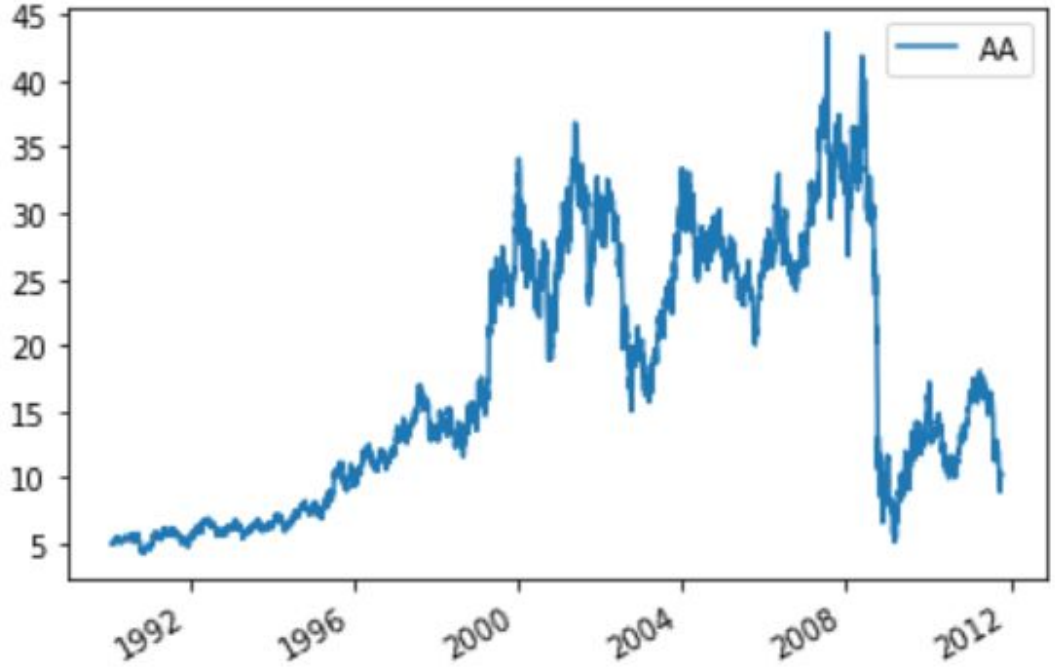
Using df.plot()

```
# plotting column 'AA'

df.plot(y = 'AA')
```

With Matplotlib

```
plt.plot(df['AA']) # plots column 'AA'

plt.legend(['AA']) # specifies the legend

plt.show() # displays the plot
```

Result

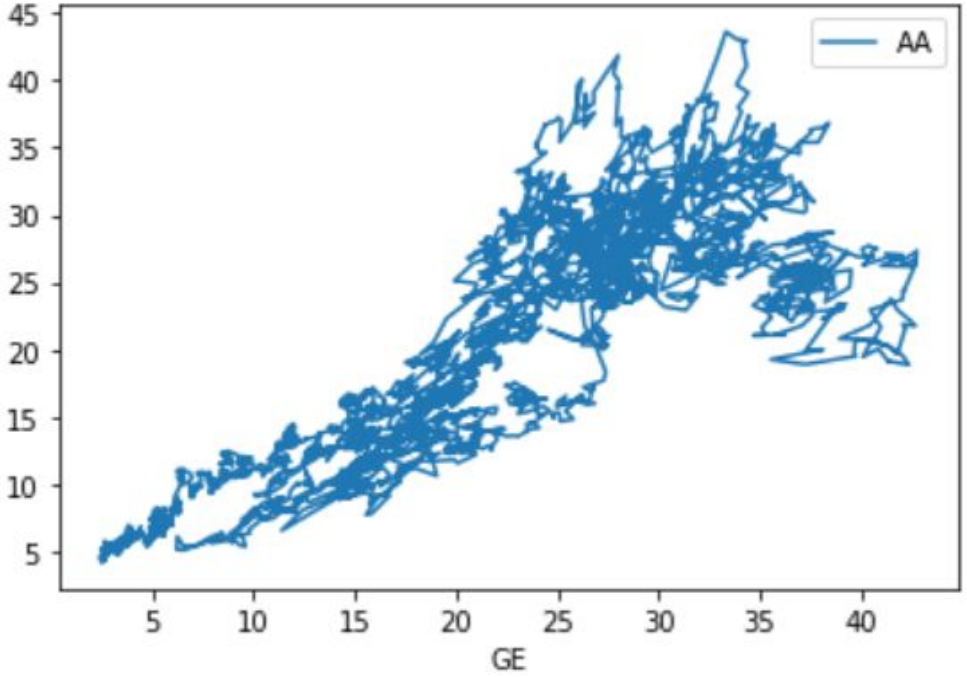


x-axis is the index

Using df.plot()

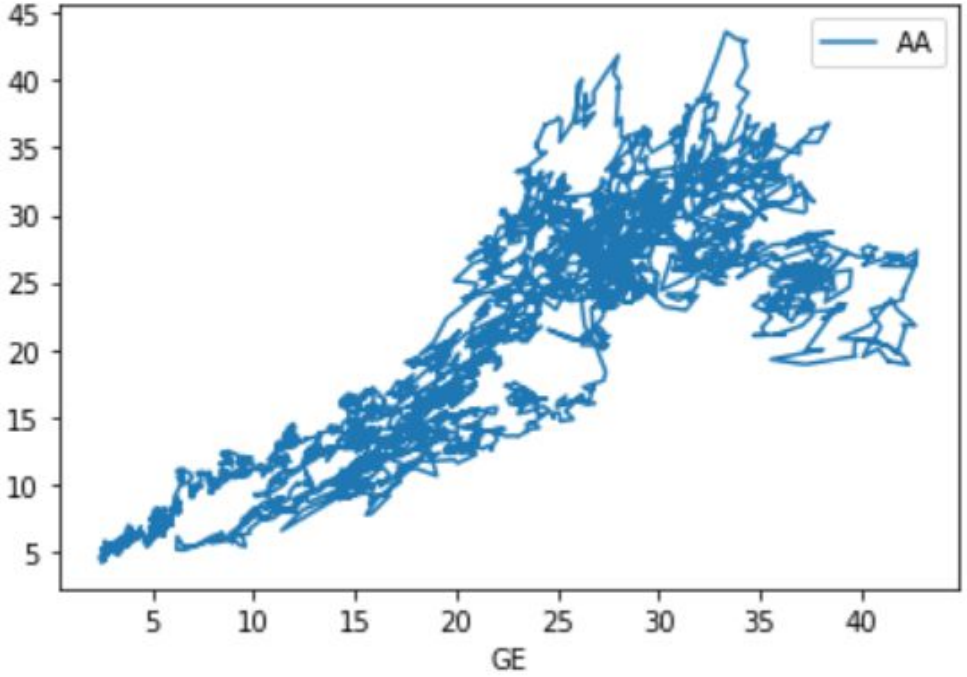
```
df.plot(x = 'GE', # new x-axis column
        y = 'AA')
```

Result



Why does this look... terrible?

Result



Plot *continuous* variables for x-axis

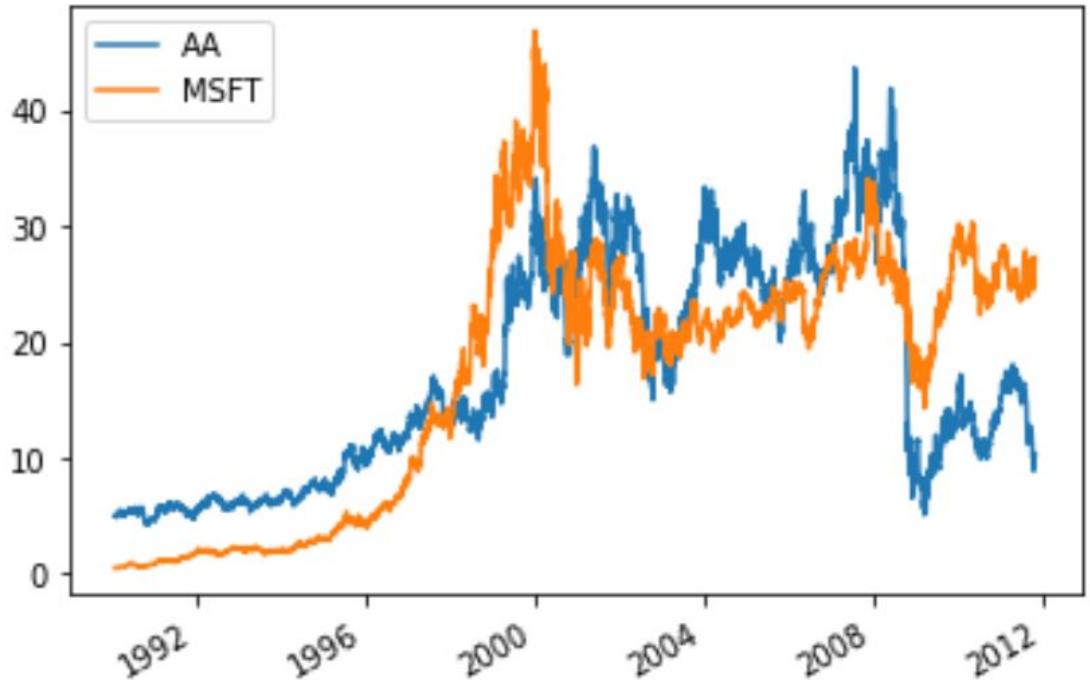
Using df.plot()

```
# plotting columns 'AA' and 'MSFT'

df.plot(y = ['AA', 'MSFT'])
```

With Matplotlib

```
plt.plot(df['AA']) # plots the 'AA' column

plt.plot(df['MSFT']) # plots the 'MSFT' column

plt.legend(['AA', 'MSFT']) # specifies legend

plt.show() # displays the plot
```

Result



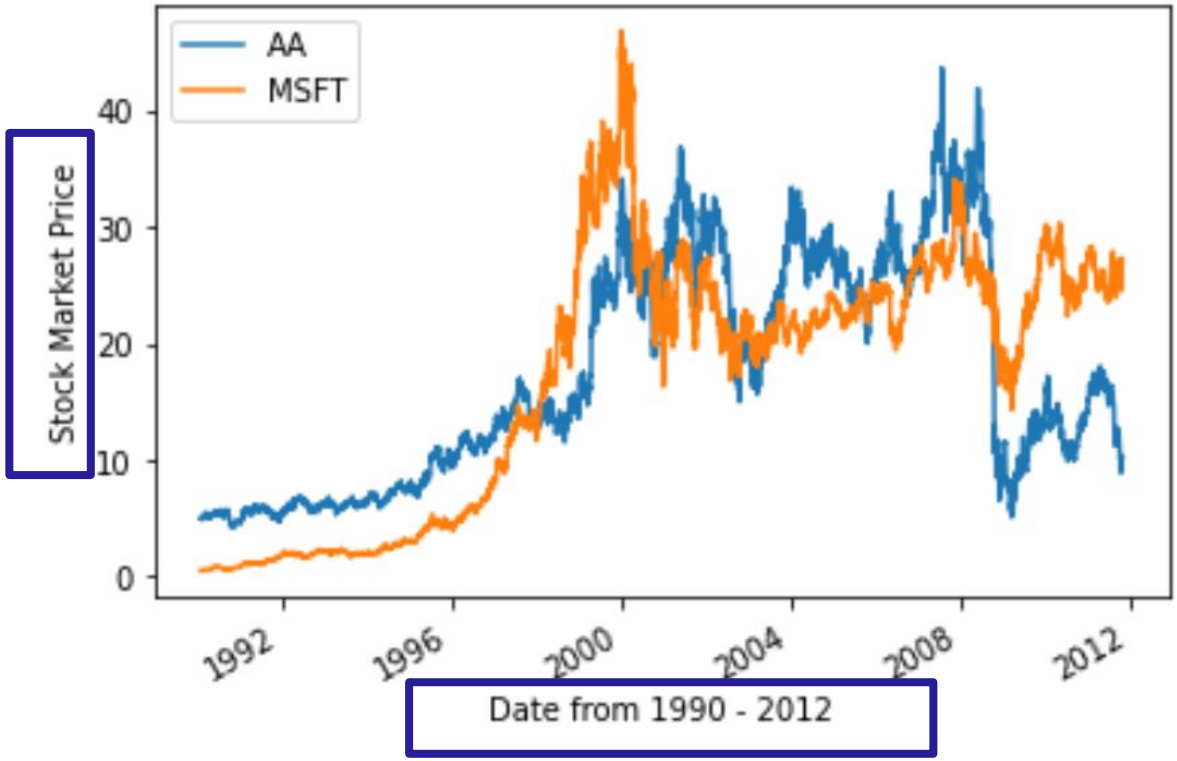
Using df.plot()

```
df.plot(y = ['AA', 'MSFT'],
        xlabel = 'Date from 1990 - 2012',
        ylabel = 'Stock Market Price')
```

With Matplotlib

```
plt.plot(df['AA']) # plots the first column

plt.plot(df['MSFT']) # plots the second column

plt.xlabel('Date from 1990 - 2012') # sets the x-axis label

plt.ylabel('Stock Market Price') # sets the y-axis label

plt.legend(['AA', 'MSFT']) # specifies the legend

plt.show() # displays the plot
```

Result


AA
MSFT
Stock Market Price
40
30
20
10
0
1992
1996
2000
2004
2008
2012
Date from 1990 - 2012

Now you try!

Using df.plot()

```
df.plot(y = ['AA', 'MSFT'],
        xlabel = 'Date from 1990 - 2012',
        ylabel = 'Stock Market Price',
        title = 'Stock Market Index for AA and MSFT')
```

With Matplotlib

```
plt.plot(df['AA']) # plots the first column

plt.plot(df['MSFT']) # plots the second column

...,

plt.title('Stock Market Index for AA and MSFT') # sets the title

plt.show() # displays the plot
```

Result


Stock Market Index for AA and MSFT
AA
MSFT
Stock Market Price
0
10
20
30
40
1992
1996
2000
2004
2008
2012
Date from 1990 - 2012

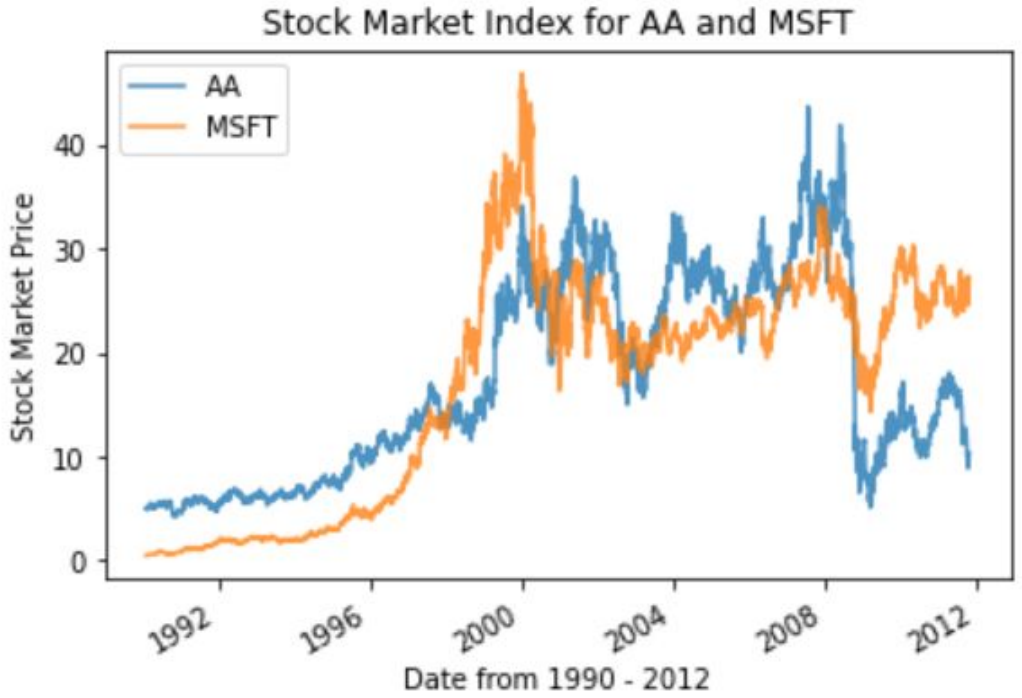
Using df.plot()

```
df.plot(y = ['AA', 'MSFT'],
        ...,
        alpha = 0.8)
```

With Matplotlib

```
plt.plot(df['AA'], alpha = 0.8)
    # plots the first column, sets the transparency
plt.plot(df['MSFT'], alpha = 0.8)
    # plots the second column, sets the transparency
...,
plt.show() # displays the plot
```

Result


Stock Market Index for AA and MSFT
AA
MSFT
Stock Market Price
0
10
20
30
40
1992
1996
2000
2004
2008
2012
Date from 1990 - 2012

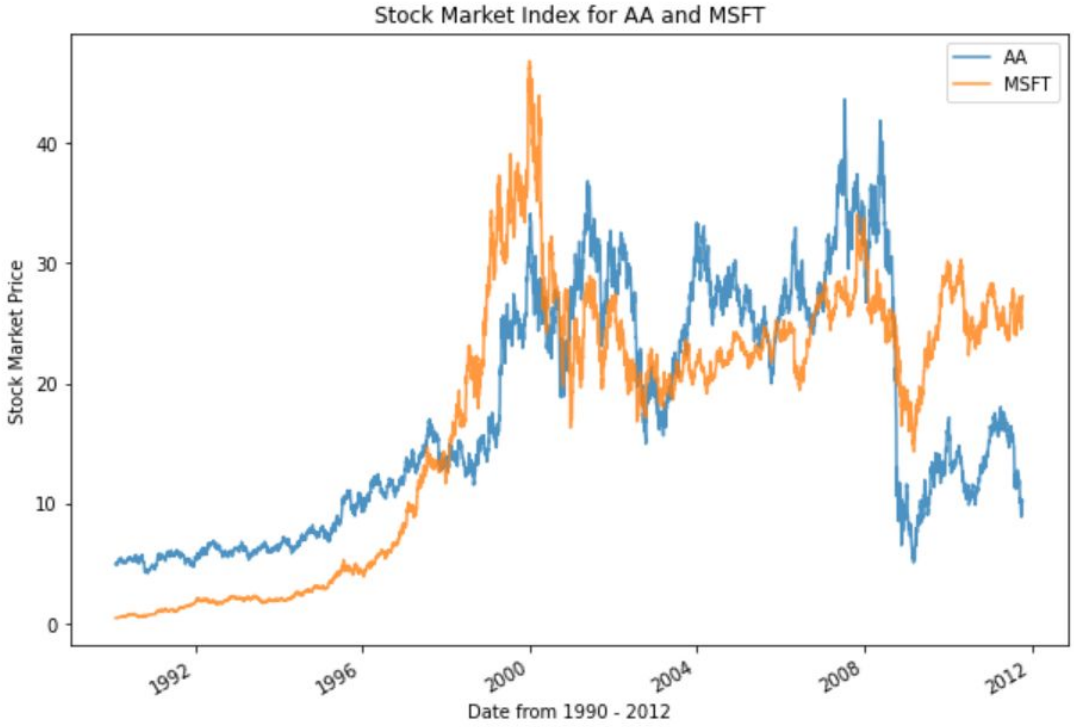
Using df.plot()

```
df.plot(y = ['AA', 'MSFT'],

        ...,

        figsize = (10,7))
```

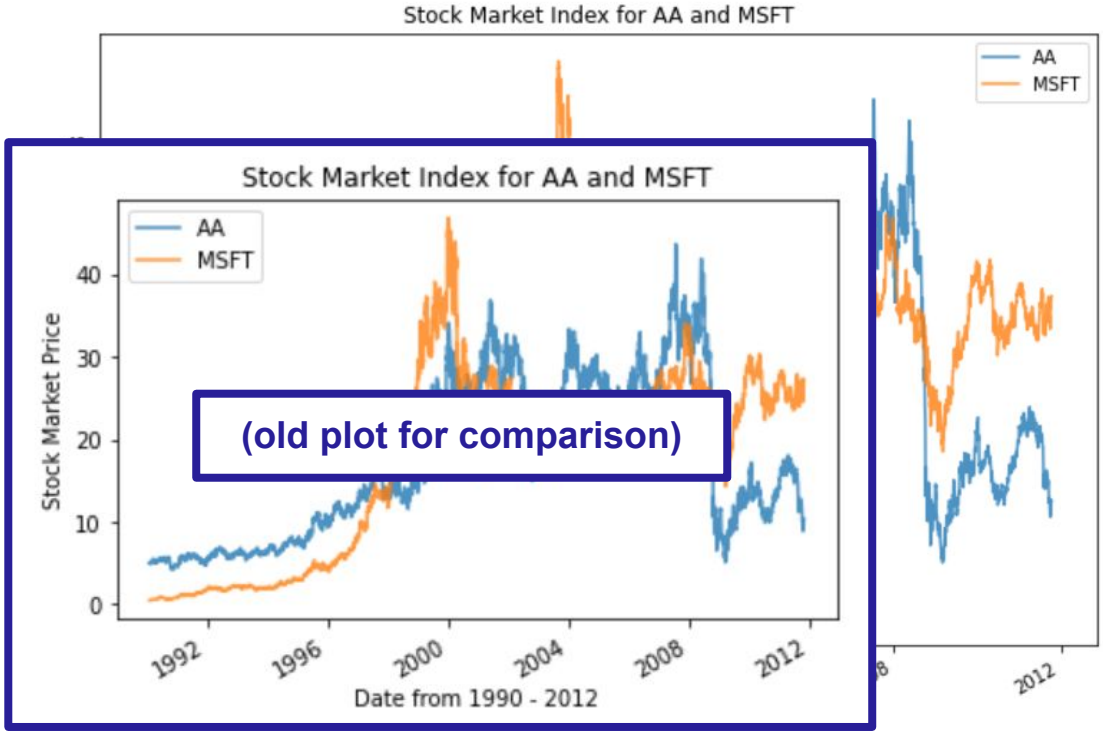
With Matplotlib

```
fig = plt.figure(figsize = (10, 7)) # sets the figure size

plt.plot(df['AA'], alpha = 0.8) # plots the first column

plt.plot(df['MSFT'], alpha = 0.8) # plots the second column

...,

plt.show() # displays the plot
```

Result


Stock Market Index for AA and MSFT
AA
MSFT
Stock Market Price
0
10
20
30
40
1992
1996
2000
2004
2008
2012
Date from 1990 - 2012

Result


Stock Market Index for AA and MSFT
AA
MSFT
Stock Market Price
0
10
20
30
40
1992
1996
2000
2004
2008
2012
Date from 1990 - 2012
(old plot for comparison)

Style customization

```
df.plot(x, y, 'ro-')

# is equivalent to

df.plot(x, y,

        color = 'r',

        marker = 'o',

        linestyle = '-')
```

- Optional **style argument**
- Marker marks **data points**
- (List of parameter values are on Colab)

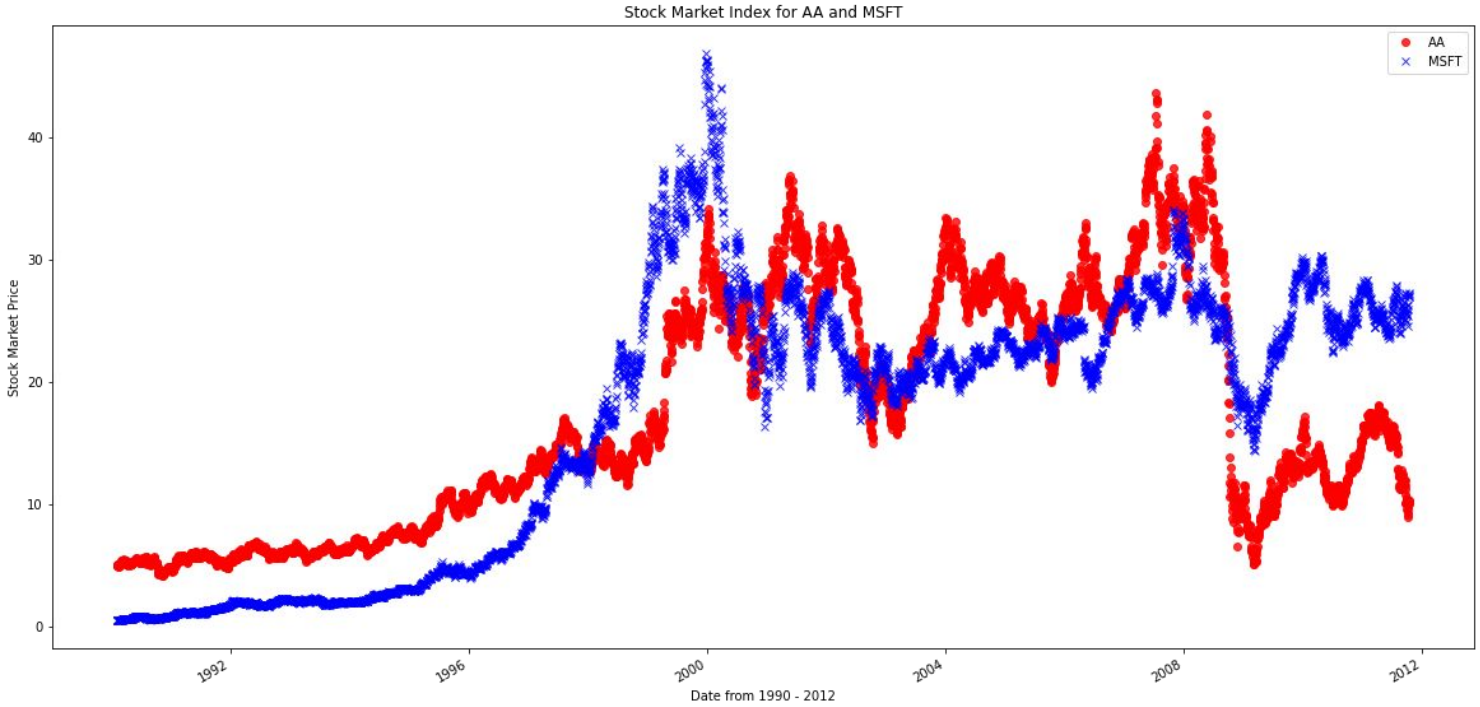
Using df.plot()

```
df.plot(y = ['AA', 'MSFT'],
        ...,
        style = ['ro-', 'bx--'])
# 'ro-' for 'AA' column
# 'bx--' for 'MSFT' column
```

With Matplotlib

```
plt.plot(df['AA'], 'ro', alpha = 0.8)
# specifies style for 'AA' column
plt.plot(df['MSFT'], 'bx', alpha = 0.8)
# specifies style for 'MSFT' column
...,
plt.show() # displays the plot
```

Result

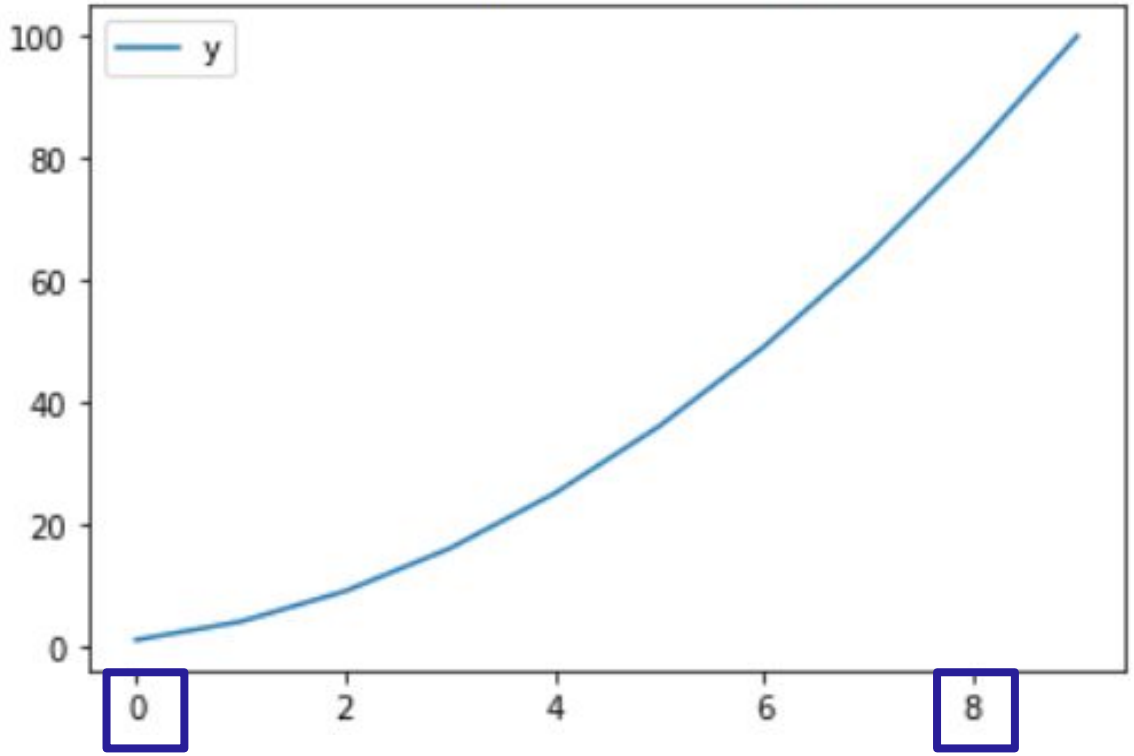

Stock Market Index for AA and MSFT
AA
MSFT
Stock Market Price
0
10
20
30
40
1992
1996
2000
2004
2008
2012
Date from 1990 - 2012

Now you try!

x-lim example

```
test_df = pd.DataFrame(data = {'y' : [0, 1, ...]})

test_df.plot()
```

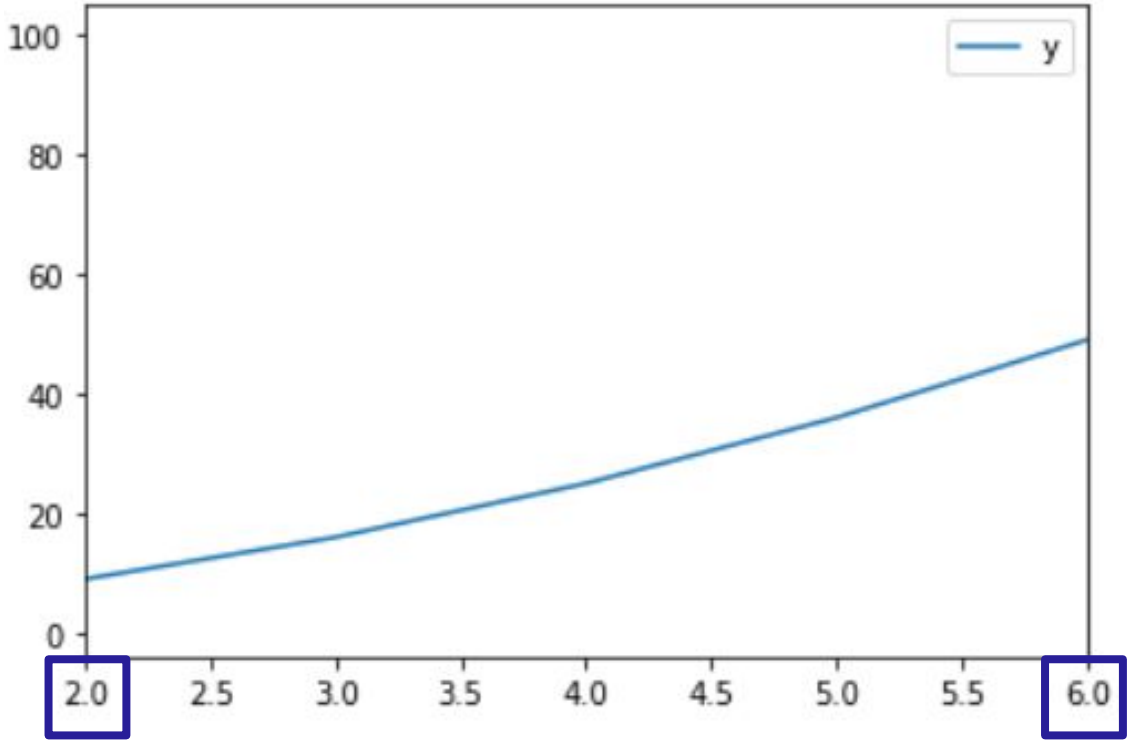
x-lim example



x-lim example

```
test_df = pd.DataFrame(data = {'y' : [0, 1, ...]})

test_df.plot(xlim = (2,6)) # sets the start and end value
```

x-lim example



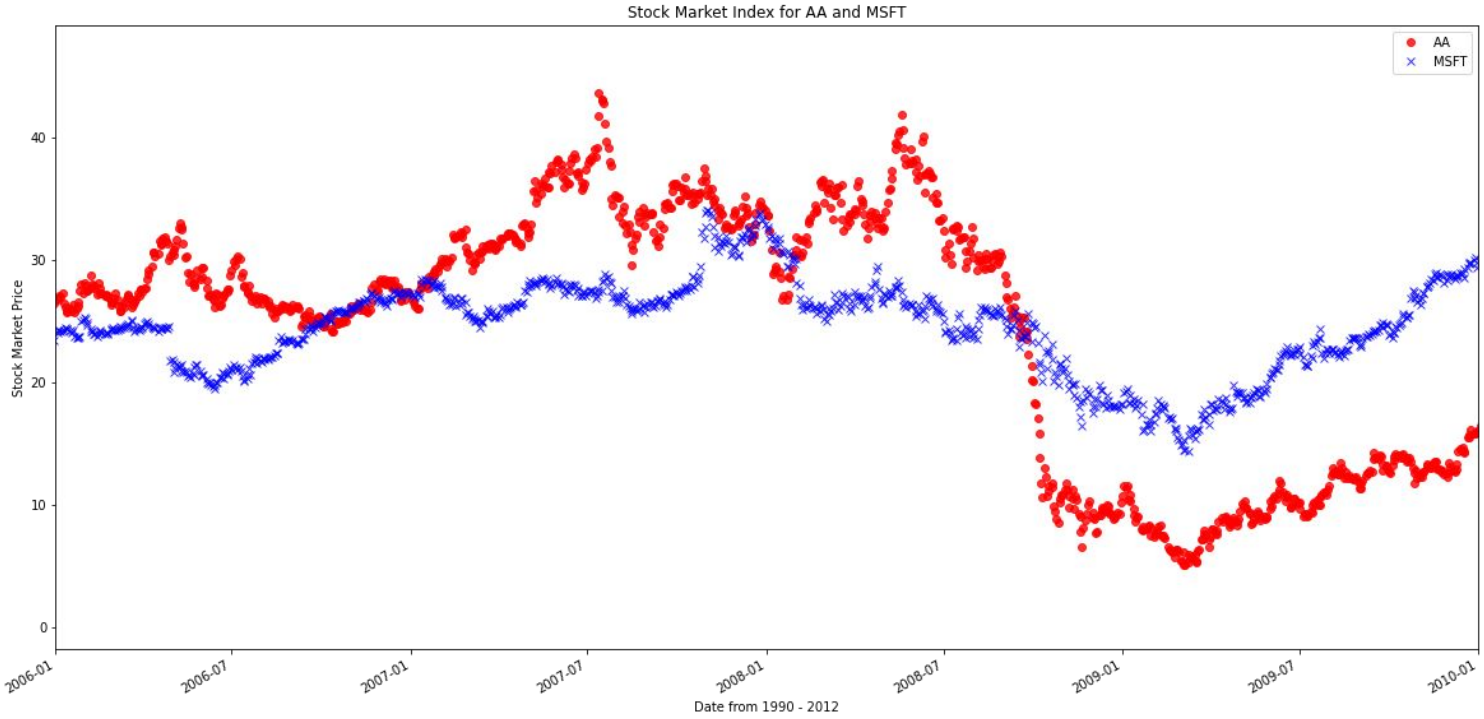
Using df.plot()

```
df.plot(y = ['AA', 'MSFT'],
        ...,
        xlim = (datetime.date(2006, 1, 1),
                datetime.date(2010, 1, 1))
```

With Matplotlib

```
plt.plot(df['AA'], 'ro', alpha = 0.8)

plt.plot(df['MSFT'], 'bx', alpha = 0.8)

plt.xlim(datetime.date(2006, 1, 1), datetime.date(2021, 1, 1))

# set the limit for the x-axis

# note that the argument is not a tuple

...,

plt.show() # displays the plot
```

Result


Stock Market Index for AA and MSFT
AA
MSFT
Stock Market Price
0
10
20
30
40
2006-01
2006-07
2007-01
2007-07
2008-01
2008-07
2009-01
2009-07
2010-01
Date from 1990 - 2012

Now you try!

Other Graphs

- More than just line graphs
- Use the '**kind**' parameter
- Pick carefully!
  - Have a **reason** why you chose a graph type

string	type of plot
'line'	line plot (default)
'bar'	vertical bar plot
'barh'	horizontal bar plot
'hist'	histogram
'box'	boxplot
'kde'	Kernel Density Estimation plot
'density'	same as 'kde'
'area'	area plot
'pie'	pie plot
'scatter'	scatter plot
'hexbin'	hexbin plot

Histogram

- **Counts** of value in columns
  - Value **frequency**
- Counts within a **bin** or range

```
[1, 2, 2, 3, 4, 1, 4]

# how many time does

# each number occur?
```

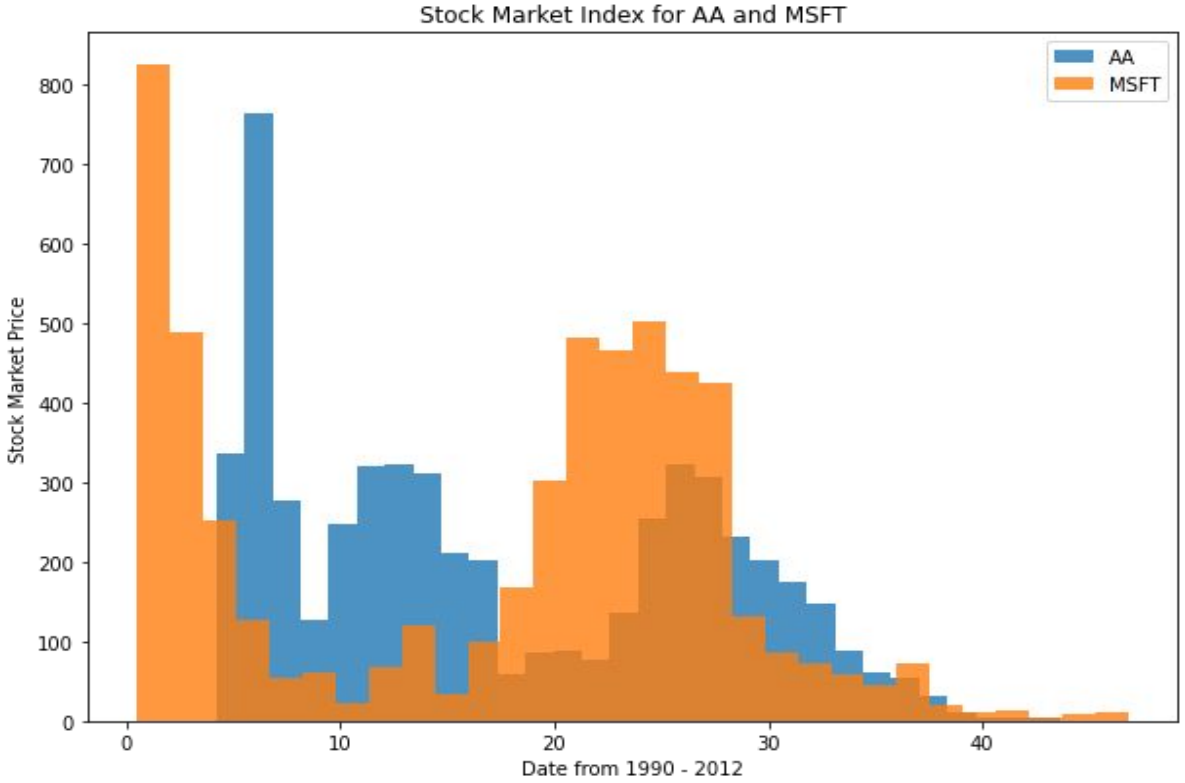
Using df.plot()

```
df.plot(kind = 'hist',
        bins = 30,
        y = ['IBM', 'AA'],
        ...)
```

With Matplotlib

```
plt.hist(df['AA'], bins = 30, alpha = 0.8)

plt.hist(df['MSFT'], bins = 30, alpha = 0.8)

...,

plt.show() # displays the plot
```

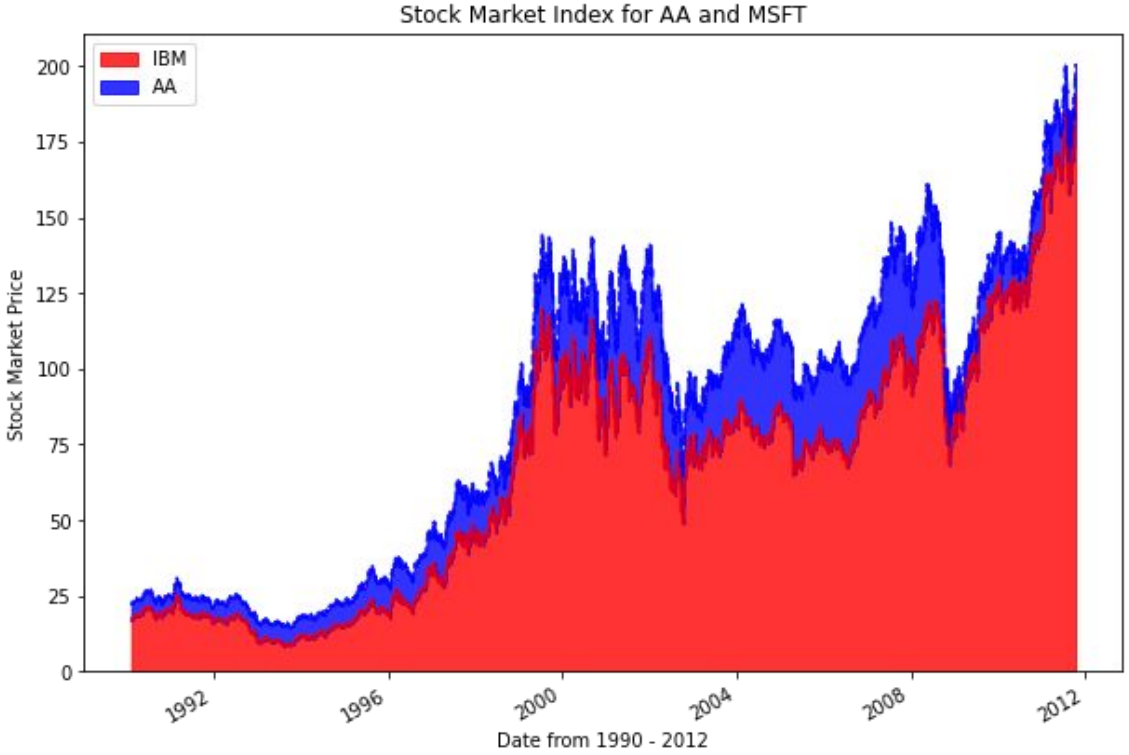
Result


Stock Market Index for AA and MSFT
AA
MSFT
Stock Market Price
0
100
200
300
400
500
600
700
800
0
10
20
30
40
Date from 1990 - 2012

Using df.plot()

```
df.plot(kind = 'area',
        y = ['IBM', 'AA'],
        style = ['r-', 'b--'],
        ...)
```

Result

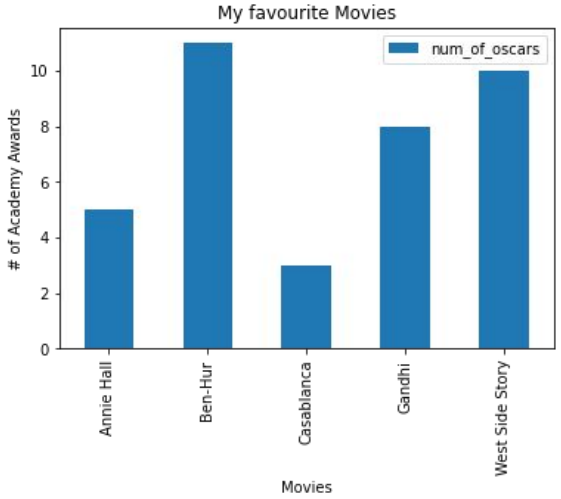

Stock Market Index for AA and MSFT
IBM
AA
Stock Market Price
200
175
150
125
100
75
50
25
0
1992
1996
2000
2004
2008
2012
Date from 1990 - 2012

Bar Charts

- Quantity varies across **categorical data**
  - Usually strings

df_for_bar_plot

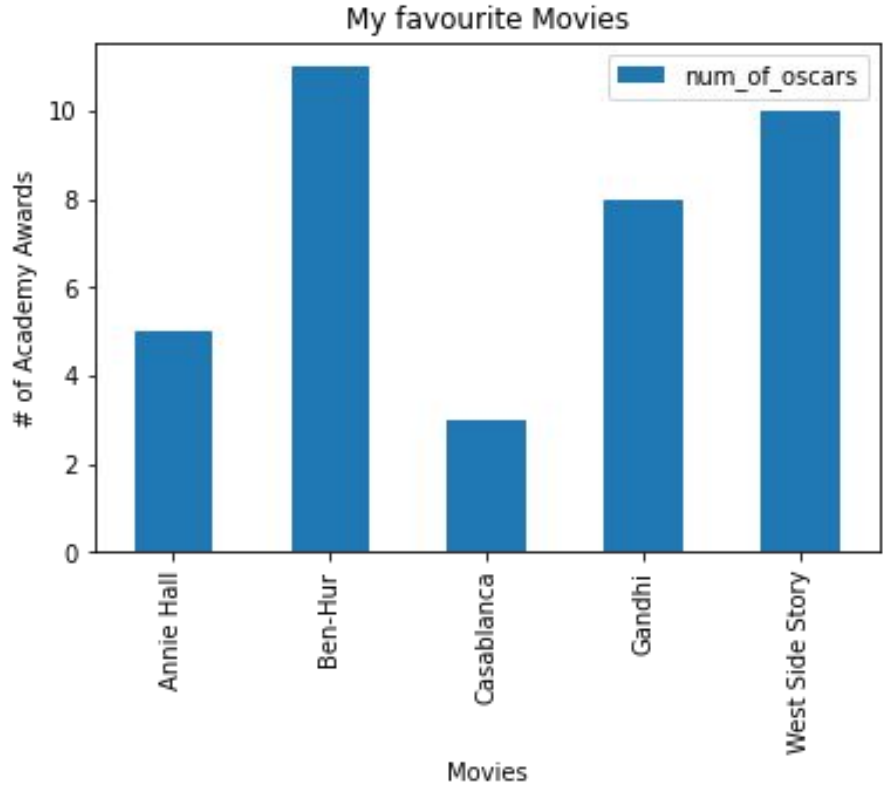
	movies	num_of_oscars
0	Annie Hall	5
1	Ben-Hur	11
2	Casablanca	3
3	Gandhi	8
4	West Side Story	10



Using df.plot()

```
df_for_bar_plot.plot(kind = 'bar',
                     x = 'movies',
                     y = 'num_of_oscars',
                     title = 'My favorite Movie',
                     ylabel = '# of Academy Awards',
                     xlabel = 'Movies')
```

Result


My favourite Movies
num_of_oscars
# of Academy Awards
0
2
4
6
8
10
Annie Hall
Ben-Hur
Casablanca
Gandhi
West Side Story
Movies

Now you try!

Switching Datasets!

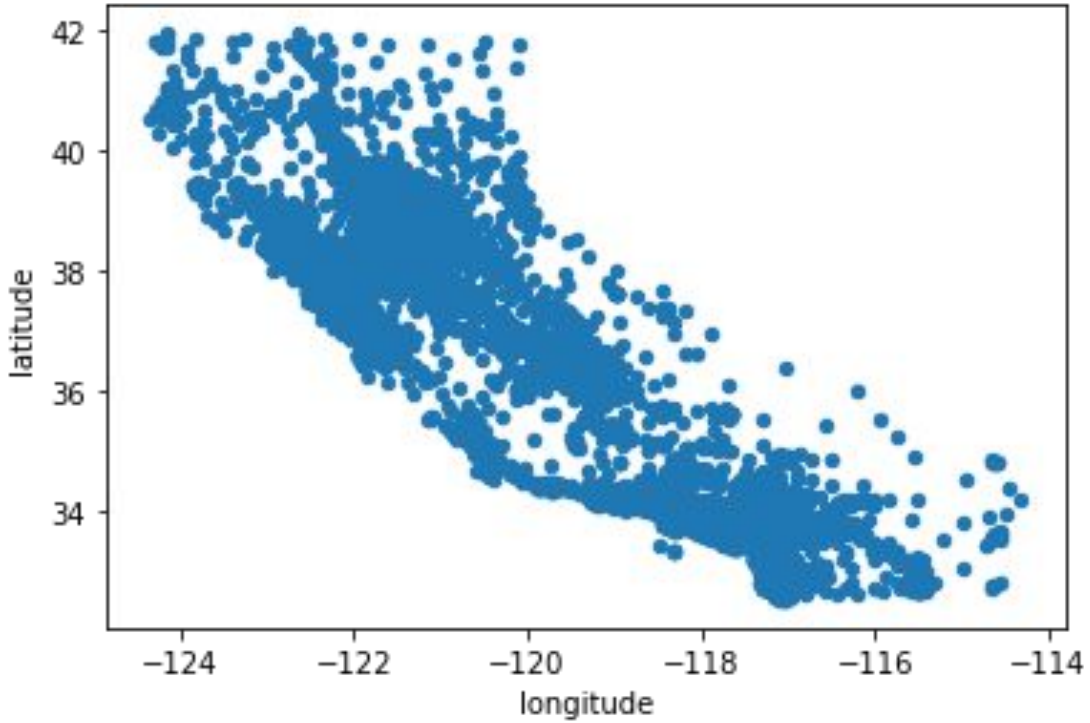
- df is now the California Housing Dataset

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0
...	...	...	...	...	...	...

Using df.plot()

```
df.plot(kind = 'scatter',
        x = 'longitude',
        y = 'latitude')
```

Result



Using df.plot()

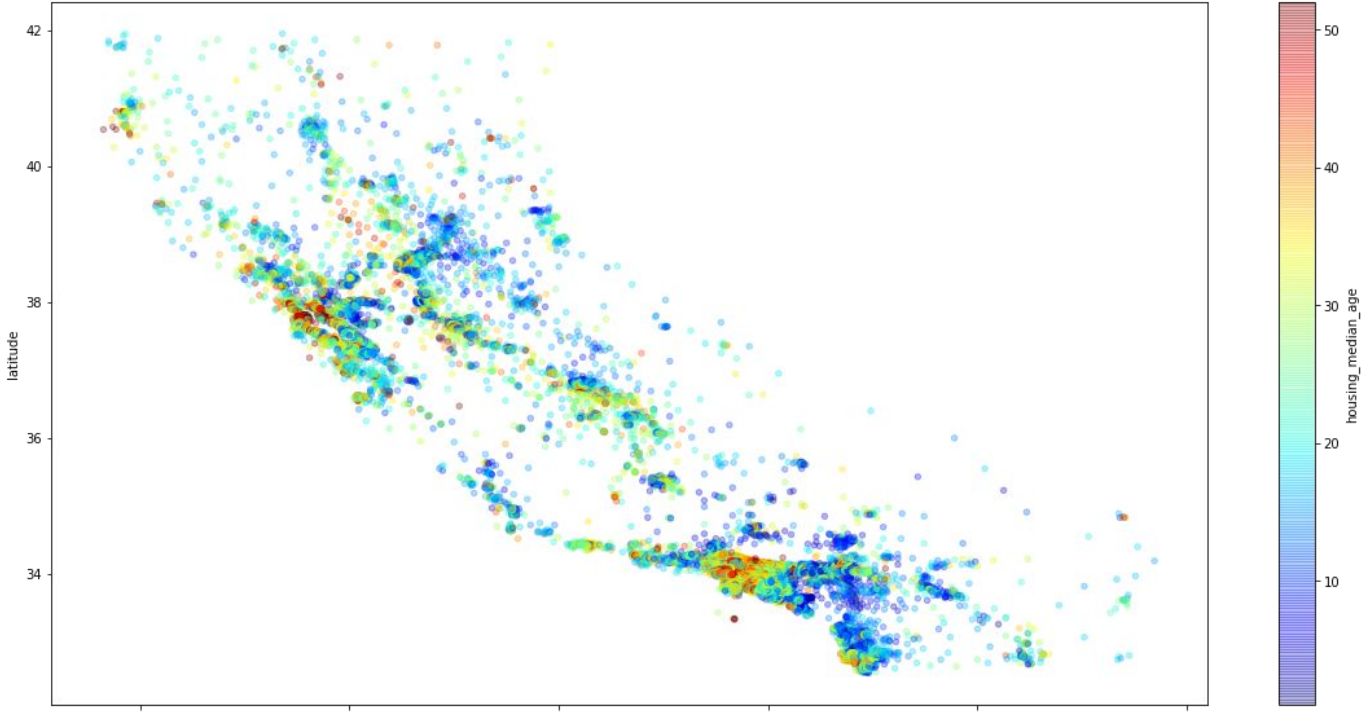
```
df.plot(kind = 'scatter',
        x = 'longitude',
        y = 'latitude',
        alpha = 0.3,
        ...,
        c = 'housing_median_age',
        cmap = plt.get_cmap('jet'),
        colorbar = True)
```

Using df.plot()

```
df.plot(kind = 'scatter
        x = 'longitude'
        y = 'latitude',
        alpha = 0.3,
        ...,
        c = 'housing_median_age',
        cmap = plt.get_cmap('jet'),
        colorbar = True)
```

```
# point color depends on
# 'housing_median_age' value
```

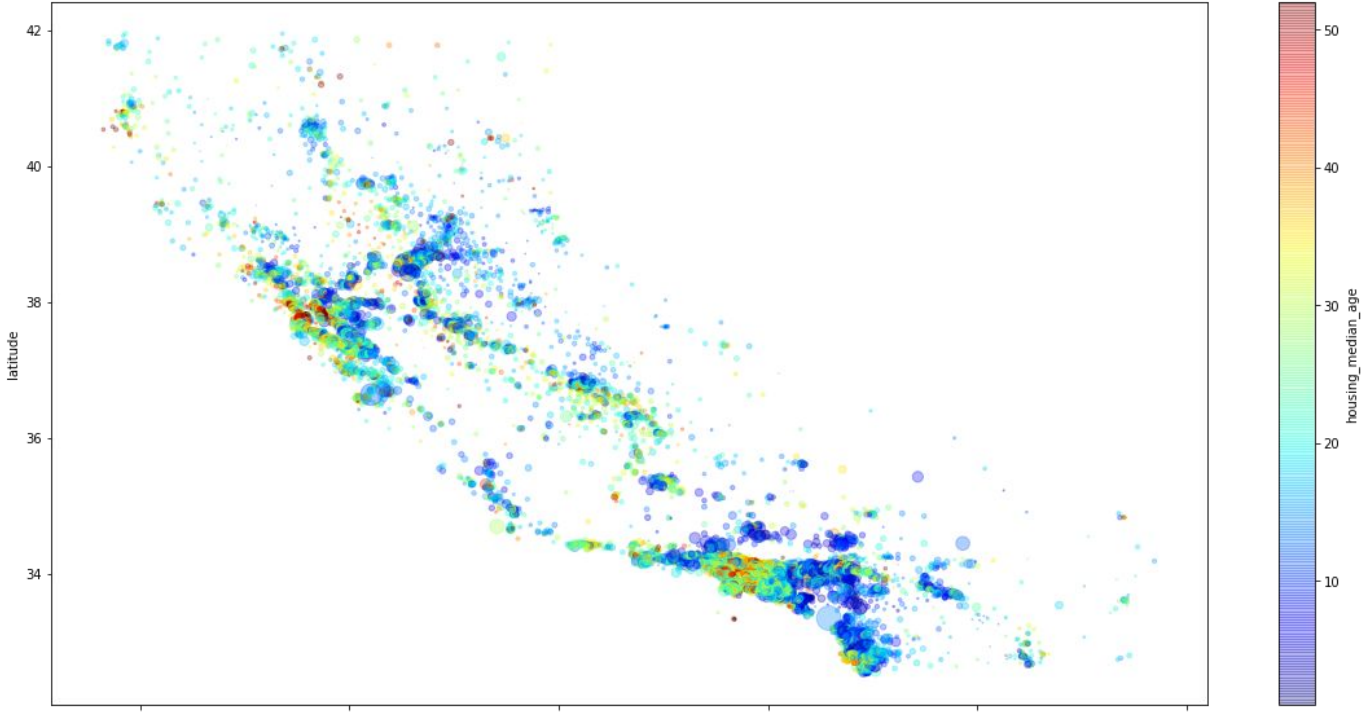
Result



Using df.plot()

```
df.plot(kind = 'scatter',
        x = 'longitude',
        y = 'latitude',
        ...,
        s = df['population'] / 100)
```

Result



Recap

- **Matplotlib**
  - Understanding pyplot.plot
- **Figures and subplots**
  - Colors, Markers, and Line Styles
  - Ticks, Labels and Legends
- **Relational** graphs
  - Bar plots
  - Histograms
  - Line plots
  - Scatter plots
- Mapping **longitude and latitude**

Follow Us!

@bitprojectorg

@bitprj

@BitProject

@BitProject

join.bitproject.org