

matplotlib

Visaulization

**Director of TEAMLAB
Sungchul Choi**



matplotlib

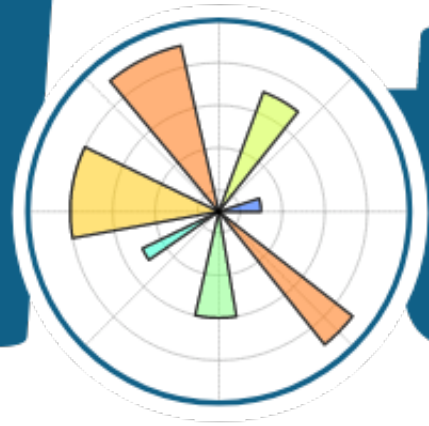
**우리의 데이터는
어떻게 생겼을까?**

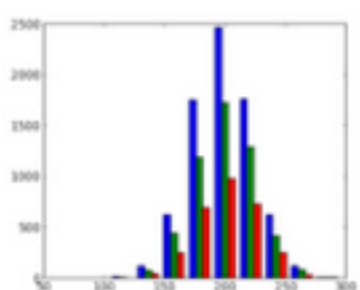
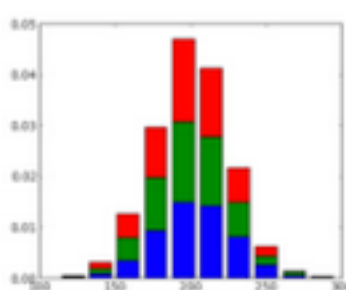
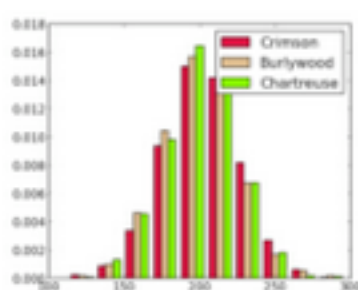
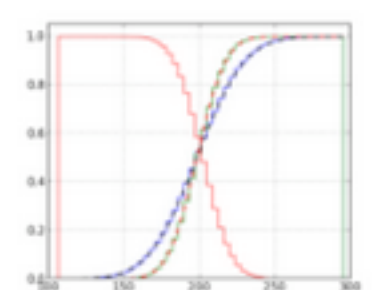
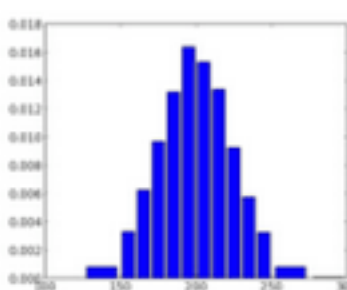
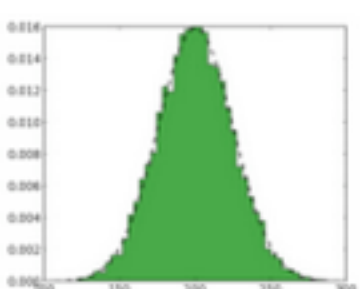
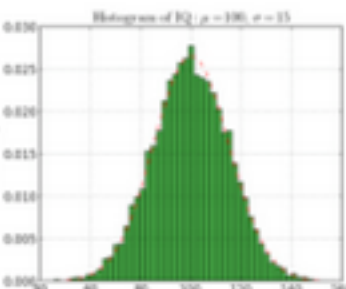
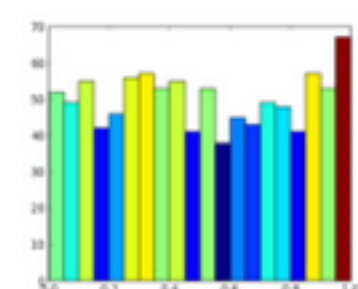
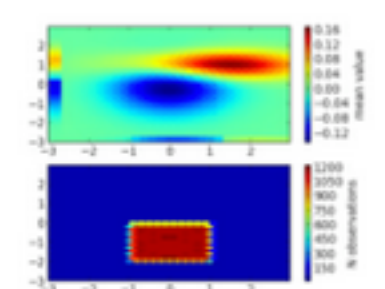
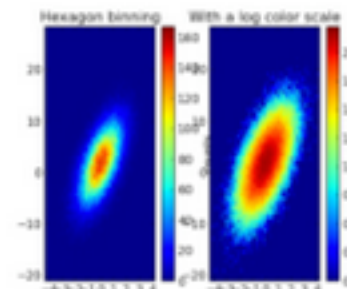
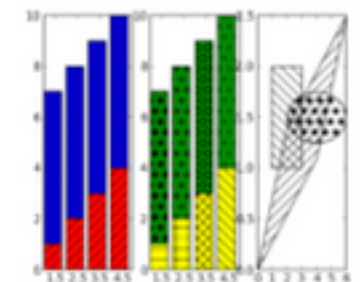
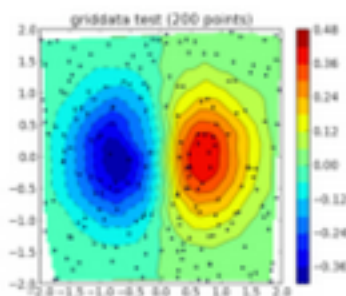
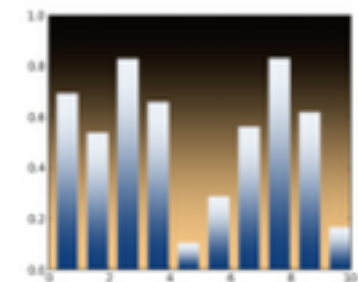
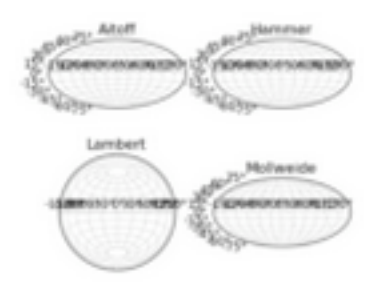
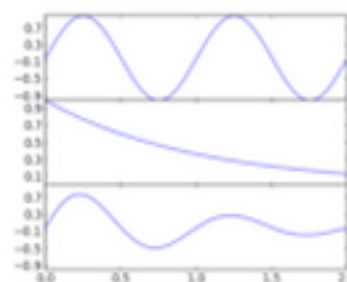
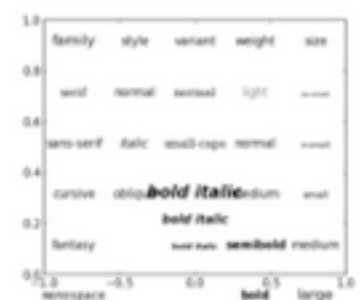
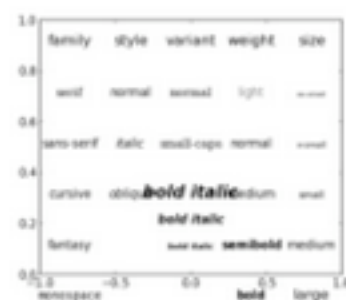
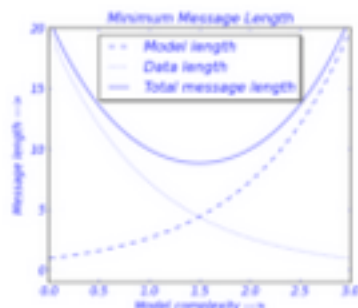
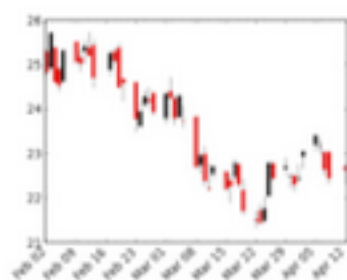
Visualization

데이터 시각화

파이썬의 대표적인 시각화 도구

matplotlib





**다양한 graph 지원
Pandas 연동!**

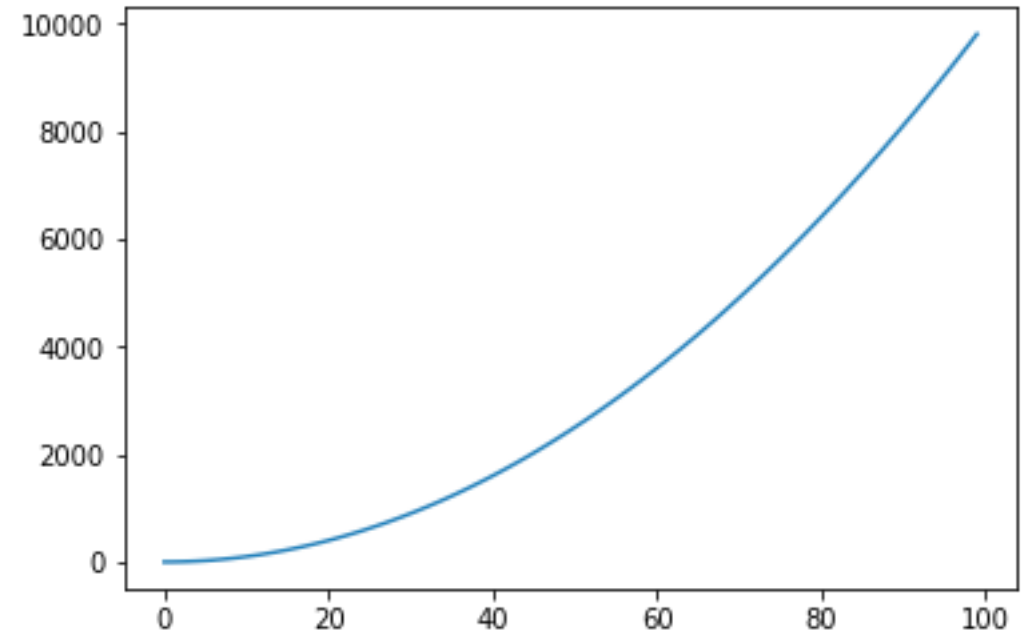
기본 사용법

matplotlib

- pyplot 객체를 사용하여 데이터를 표시
- Pyplot 객체에 그래프들을 쌓은 다음 show로 flush

```
import matplotlib.pyplot as plt

X = range(100)
Y = [value**2 for value in X]
plt.plot(X, Y)
plt.show()
```



matplotlib

- 최대 단점 argument를 kwargs 받음,
- 고정된 argument가 없어서 alt+tab으로 확인이 어려움

Signature: `plt.plot(*args, **kwargs)`

Docstring:

Plot lines and/or markers to the

:class:`~matplotlib.axes.Axes`. `*args*` is a variable length argument, allowing for multiple `*x*`, `*y*` pairs with an optional format string. For example, each of the following is legal::

```
plot(x, y)           # plot x and y using default line style and color
plot(x, y, 'bo')     # plot x and y using blue circle markers
plot(y)              # plot y using x as index array 0..N-1
plot(y, 'r+')         # ditto, but with red plusses
```

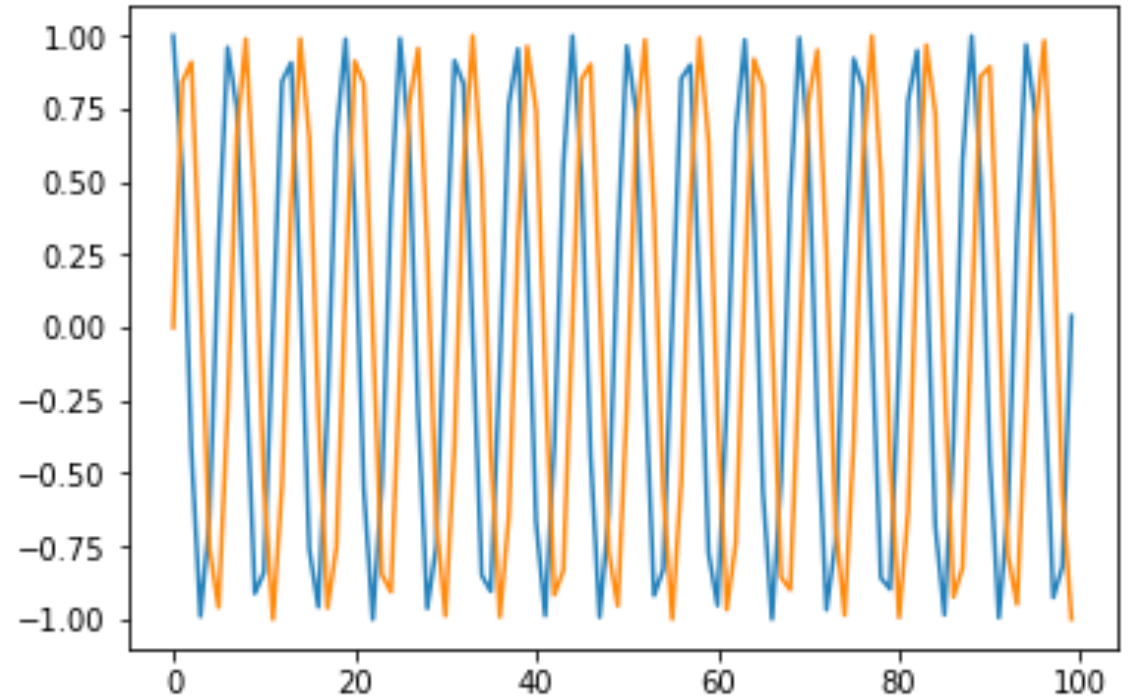
matplotlib

- Graph는 원래 figure 객체에 생성됨
- pyplot 객체 사용시, 기본 figure에 그래프가 그려짐

```
X_1 = range(100)
Y_1 = [np.cos(value) for value in X]

X_2 = range(100)
Y_2 = [np.sin(value) for value in X]

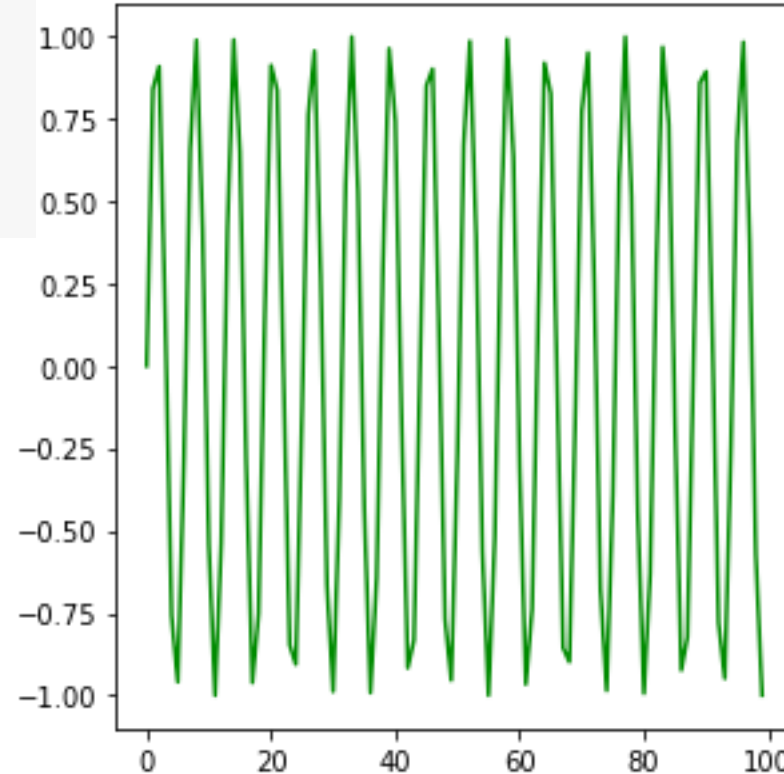
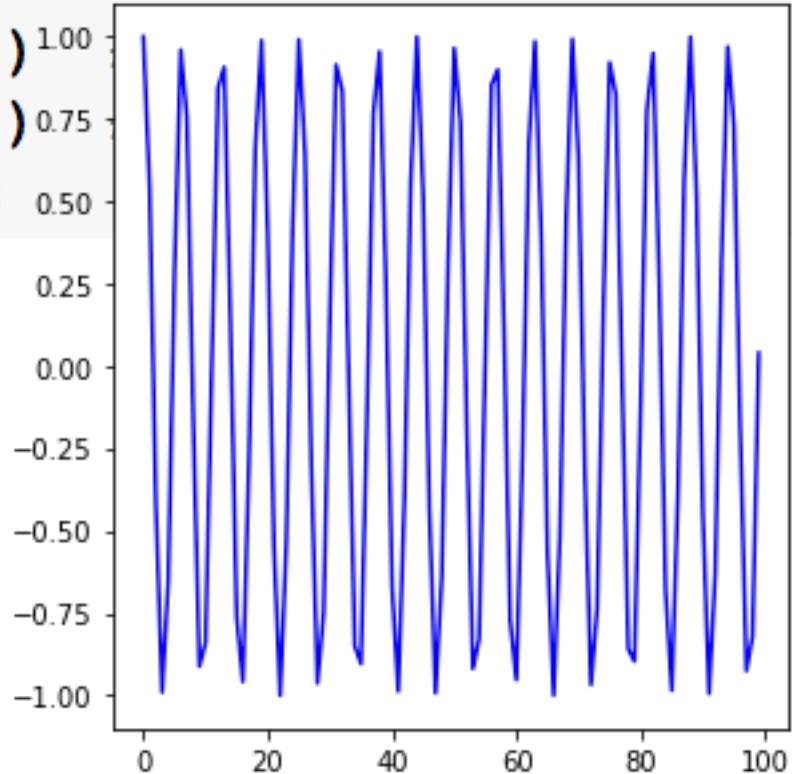
plt.plot(X_1, Y_1)
plt.plot(X_2, Y_2)
plt.show()
```



matplotlib

```
fig = plt.figure() # figure 반환  
fig.set_size_inches(10,5) # 크기지정  
ax_1 = fig.add_subplot(1,2,1) # 두개의 plot 생성  
ax_2 = fig.add_subplot(1,2,2) # 두개의 plot 생성
```

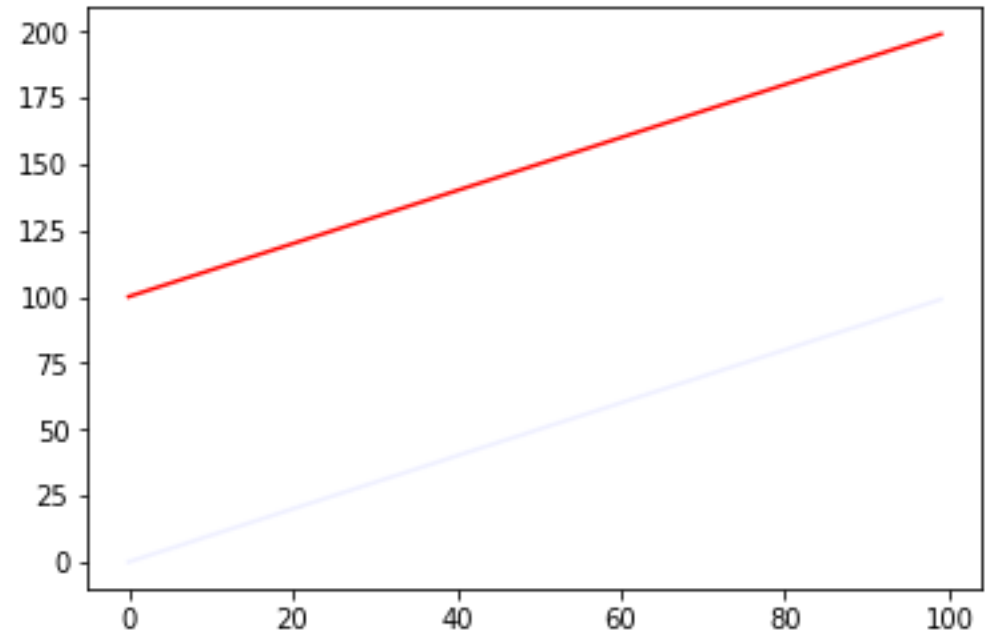
```
ax_1.plot(X_1, Y_1, c="b")  
ax_2.plot(X_2, Y_2, c="g")  
plt.show() # show & flush
```



Set color

- Color 속성을 사용
- Float → 흑백, rgb color, predefined color 사용

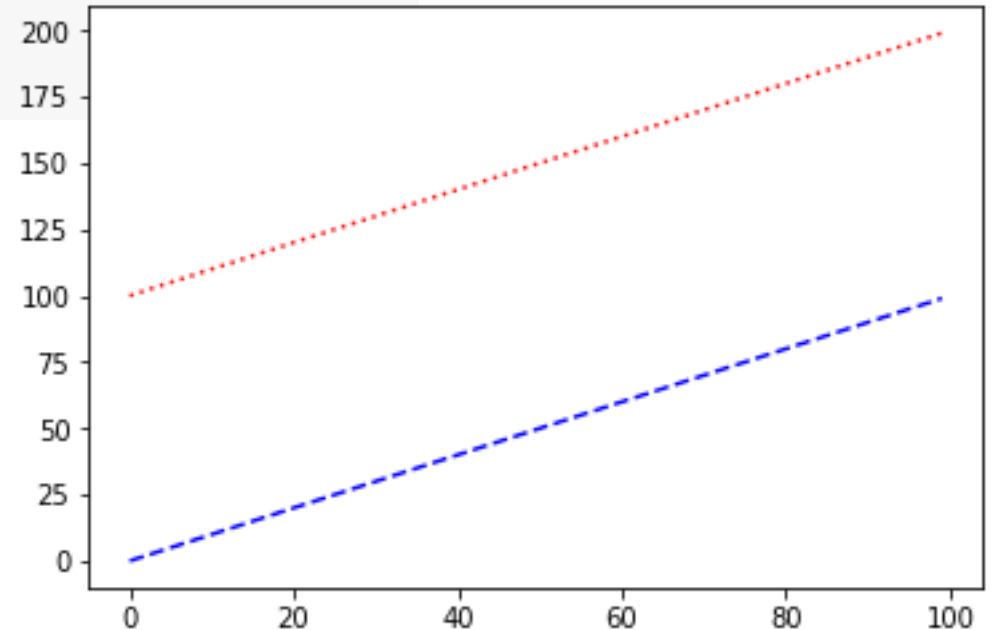
```
plt.plot(X_1, Y_1, color="#eeefff")  
plt.plot(X_2, Y_2, color="r")  
  
plt.show()
```



Set linestyle

- ls 또는 linestyle 속성 사용

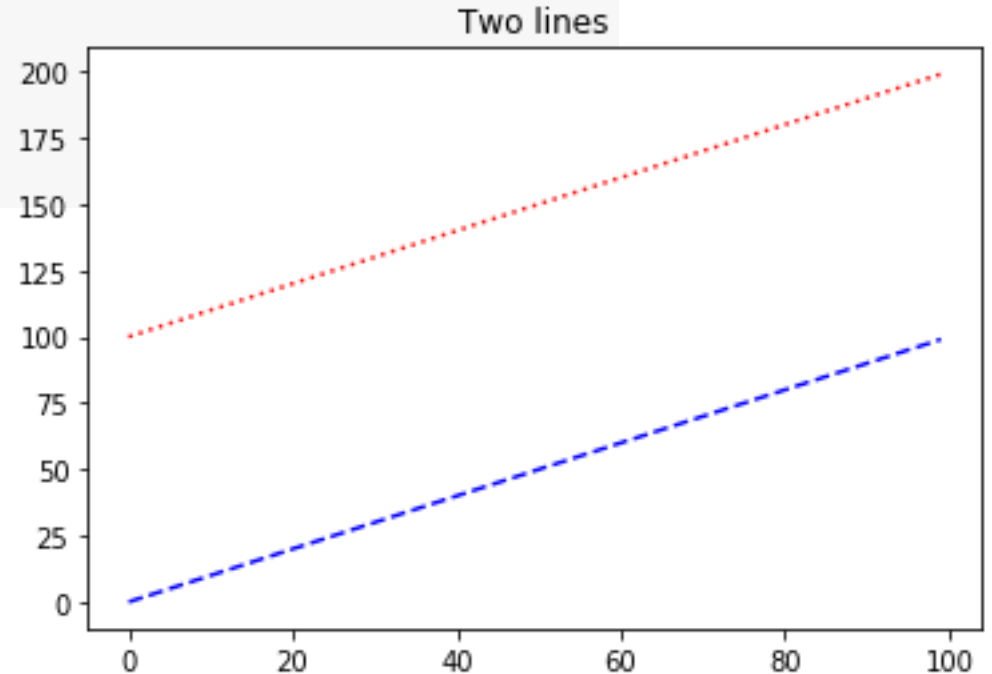
```
plt.plot(X_1, Y_1, c="b", linestyle="dashed")  
plt.plot(X_2, Y_2, c="r", ls="dotted")  
  
plt.show()
```



Set title

- Pyplot에 title함수 사용, figure의 subplot별 입력 가능

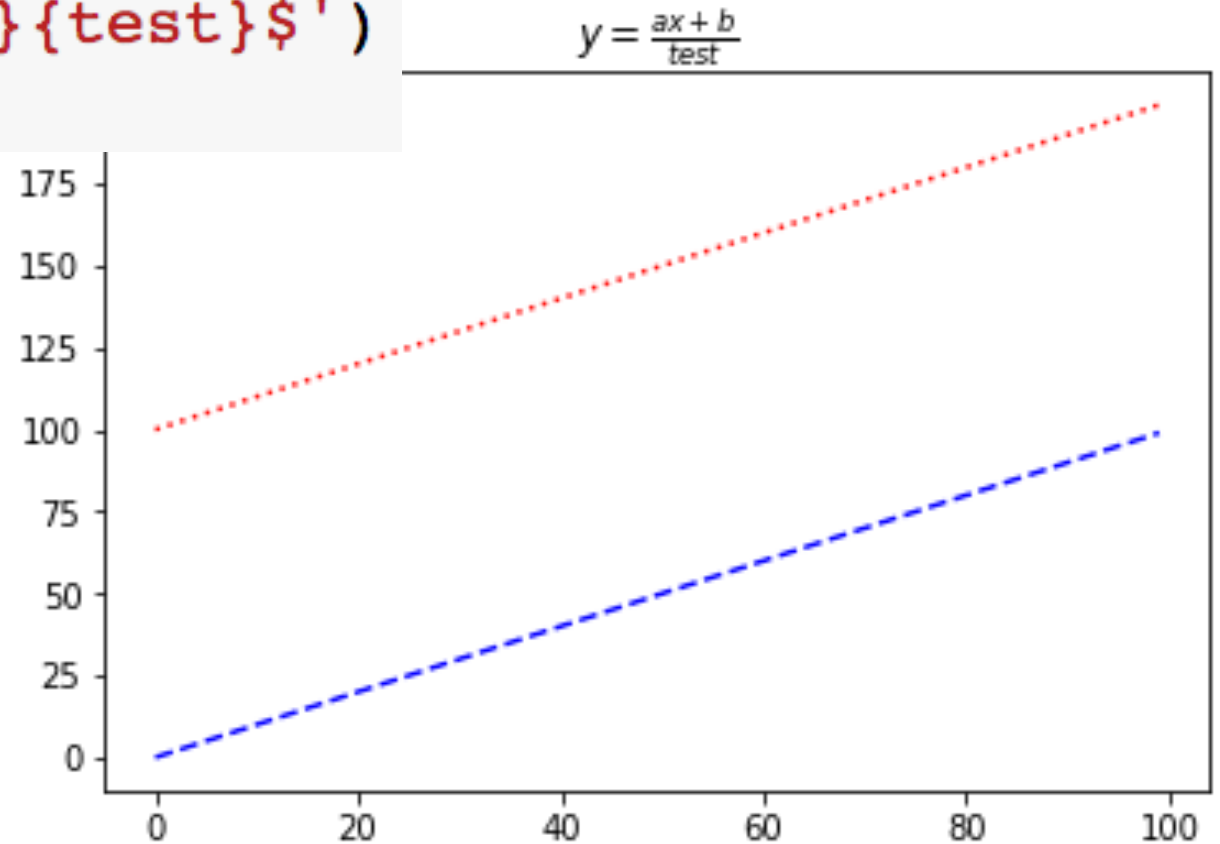
```
plt.plot(X_1, Y_1, color="b", linestyle="dashed")  
plt.plot(X_2, Y_2, color="r", linestyle="dotted")  
  
plt.title("Two lines")  
plt.show()
```



Set title

- Latex 타입의 표현도 가능 (수식 표현 가능)

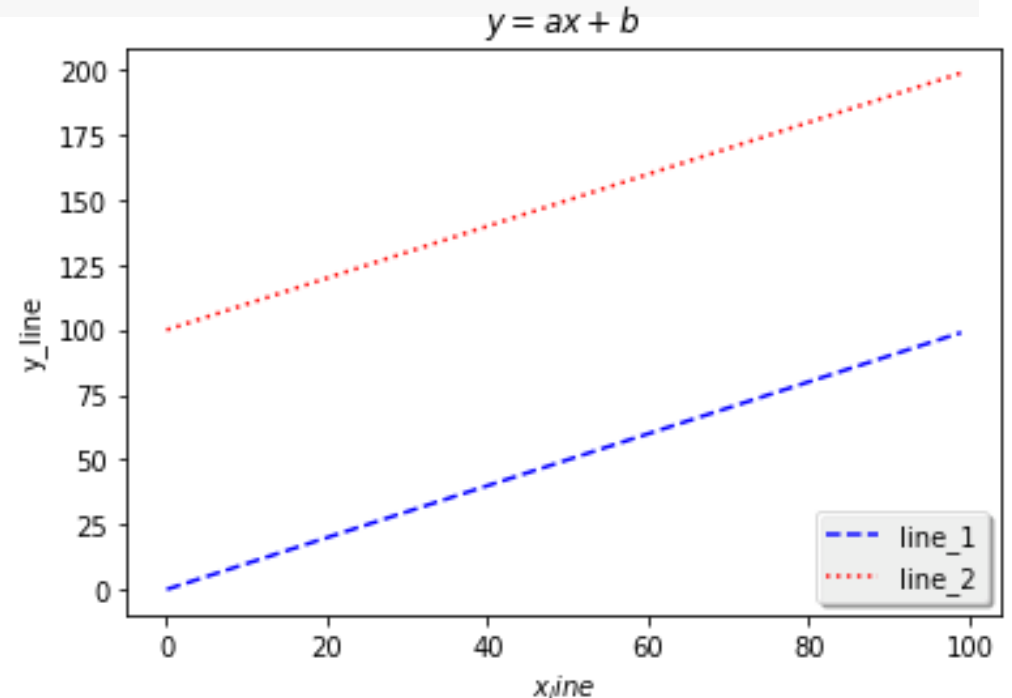
```
plt.title('$y = \frac{ax + b}{test}$')  
plt.show()
```



Set legend

- Legend 함수로 범례를 표시함, loc 위치등 속성 지정

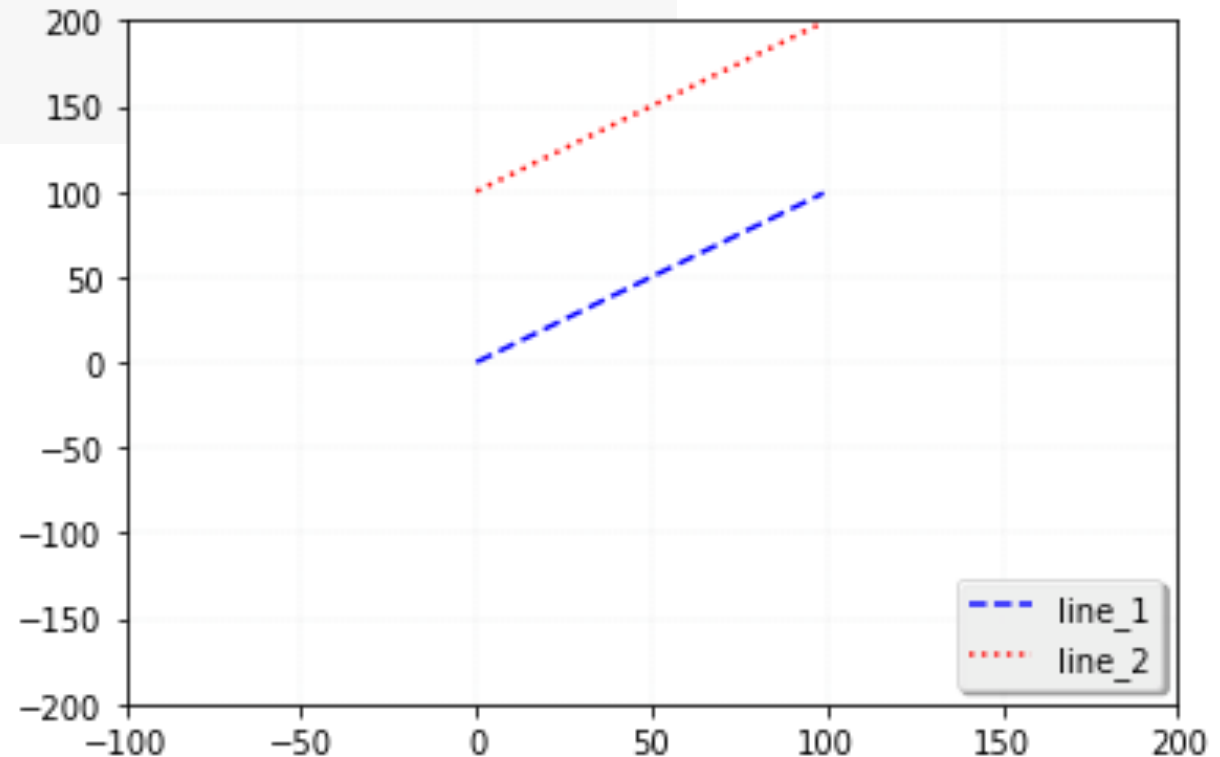
```
plt.plot(X_1, Y_1, color="b", linestyle="dashed", label='line_1')  
plt.plot(X_2, Y_2, color="r", linestyle="dotted", label='line_2')  
plt.legend(shadow=True, fancybox=True, loc="lower right")
```



Set grid & xylin

- Graph 보조선을 긋는 grid와 xy축 범위 한계를 지정

```
plt.grid(True, lw=0.4, ls="--", c=".90")  
plt.xlim(-100, 200)  
plt.ylim(-200, 200)
```



Matplotlib Graph

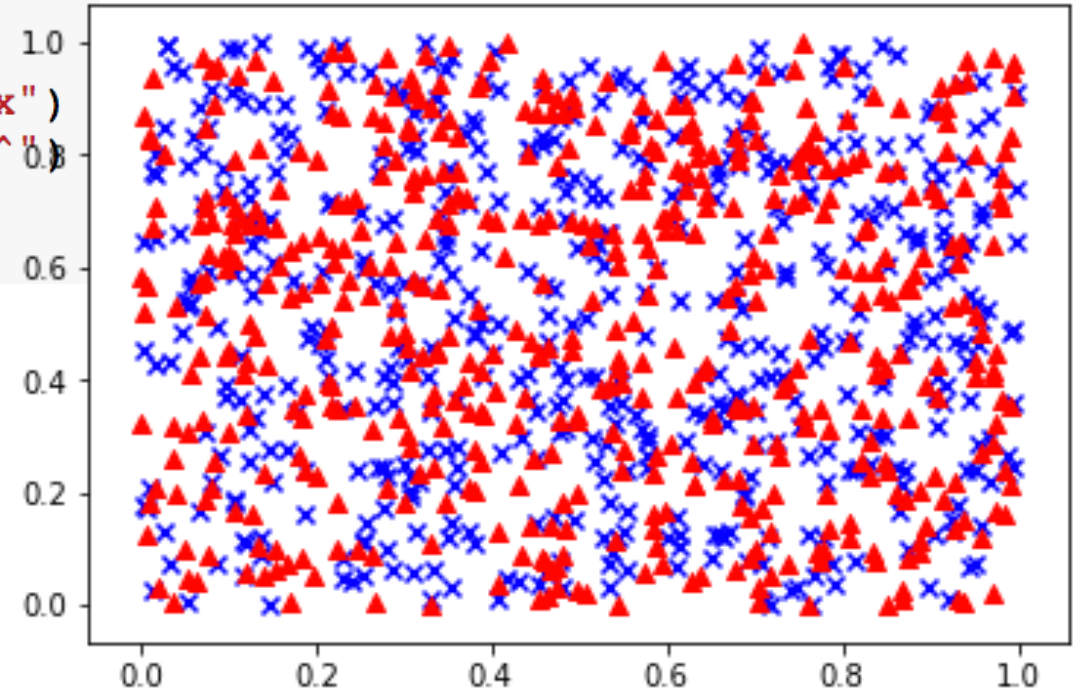
Scatter

- scatter 함수 사용, marker: scatter 모양지정

```
data_1 = np.random.rand(512, 2)
data_2 = np.random.rand(512, 2)

plt.scatter(data_1[:,0], data_1[:,1], c="b", marker="x")
plt.scatter(data_2[:,0], data_2[:,1], c="r", marker="^")

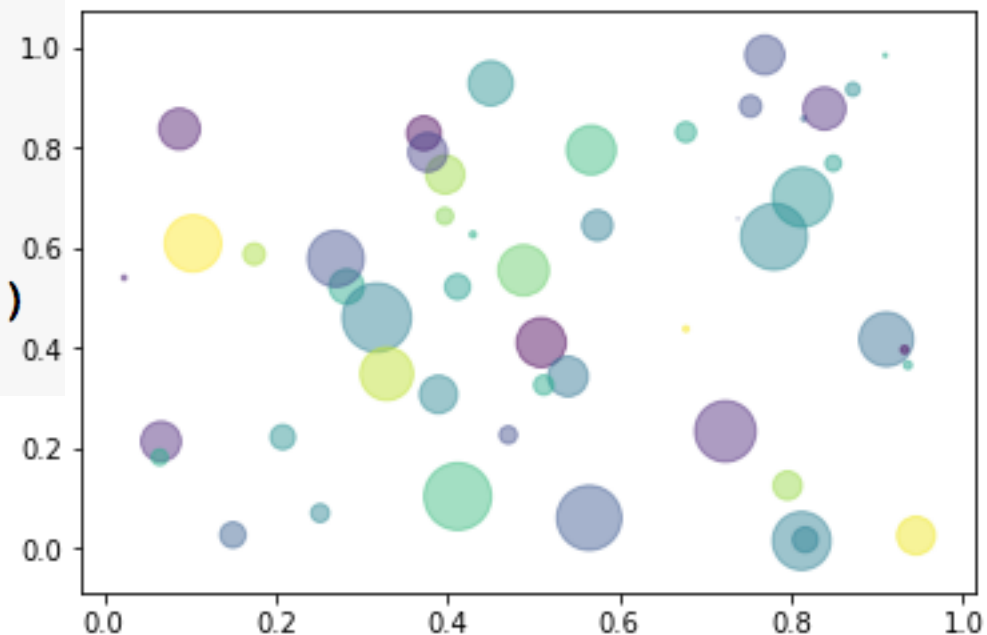
plt.show()
```



Scatter

- **s**: 데이터의 크기를 지정, 데이터의 크기비교가능

```
N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = np.pi * (15 * np.random.rand(N))**2
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```

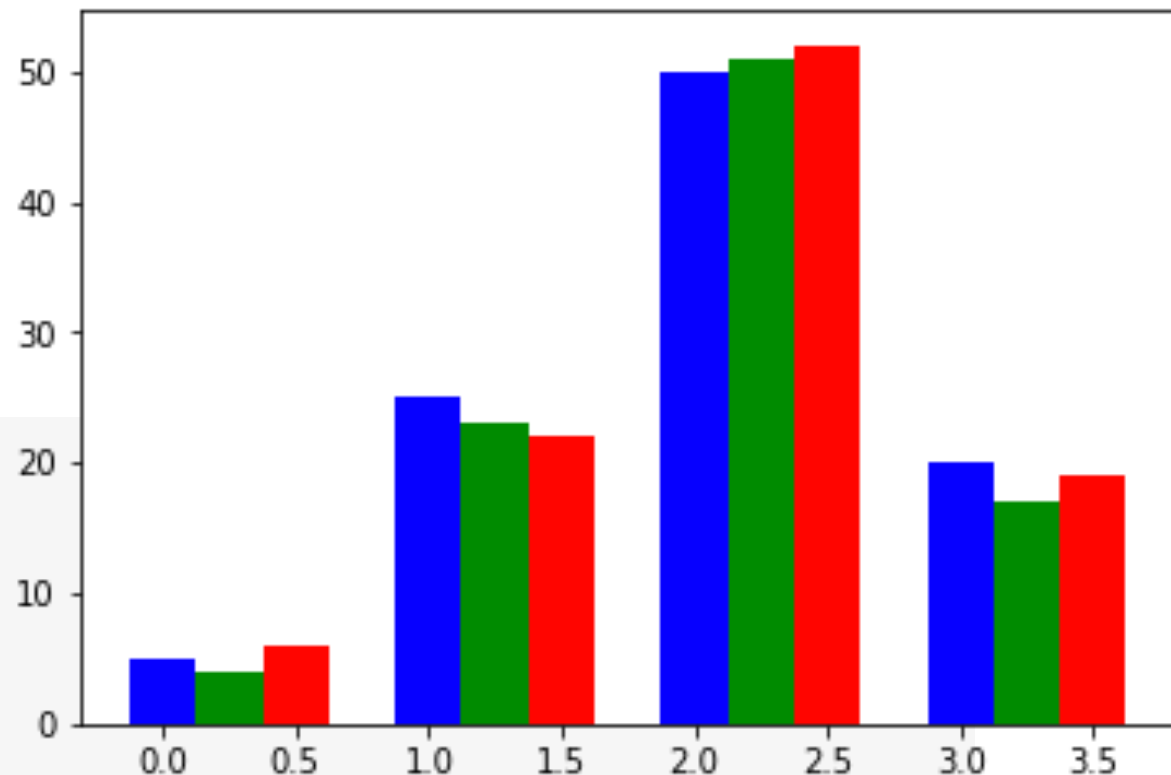


Bar chart

- Bar 함수 사용

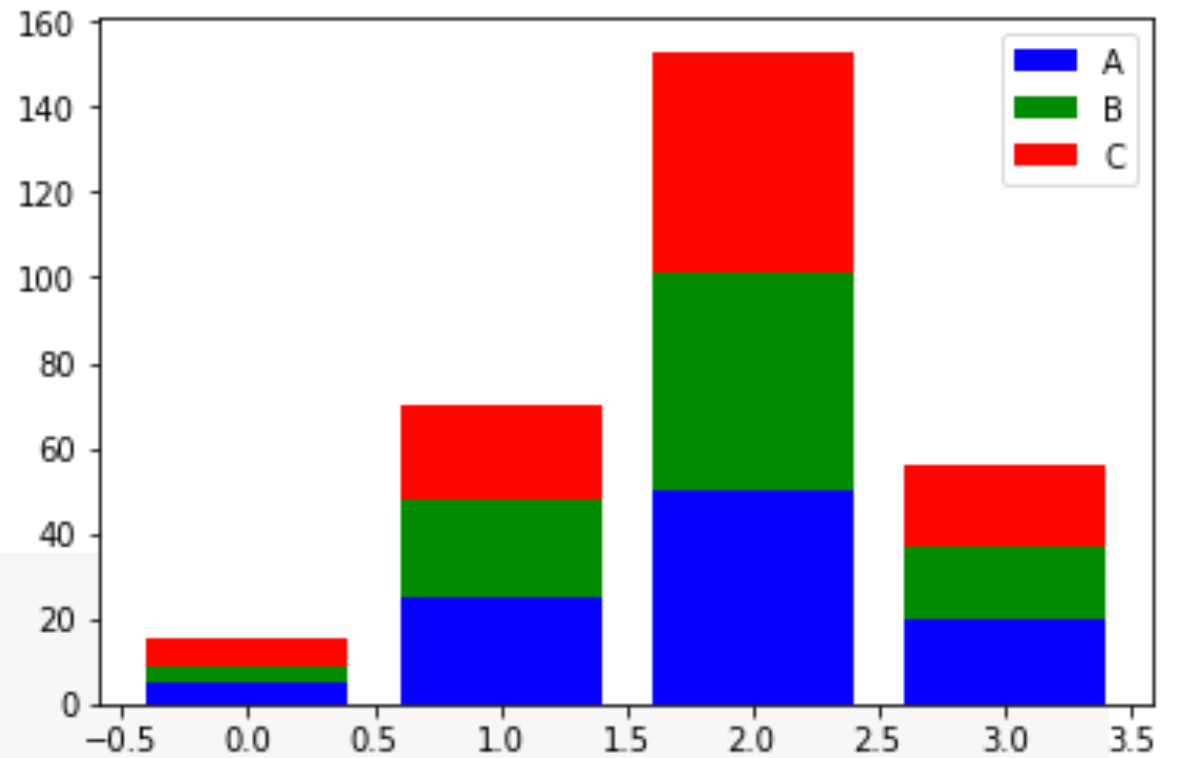
```
data = [[5., 25., 50., 20.],  
        [4., 23., 51., 17],  
        [6., 22., 52., 19]]
```

```
X = np.arange(4)  
plt.bar(X + 0.00, data[0], color = 'b', width = 0.25)  
plt.bar(X + 0.25, data[1], color = 'g', width = 0.25)  
plt.bar(X + 0.50, data[2], color = 'r', width = 0.25)  
plt.xticks(X+0.25, ("A", "B", "C", "D"))  
plt.show()
```



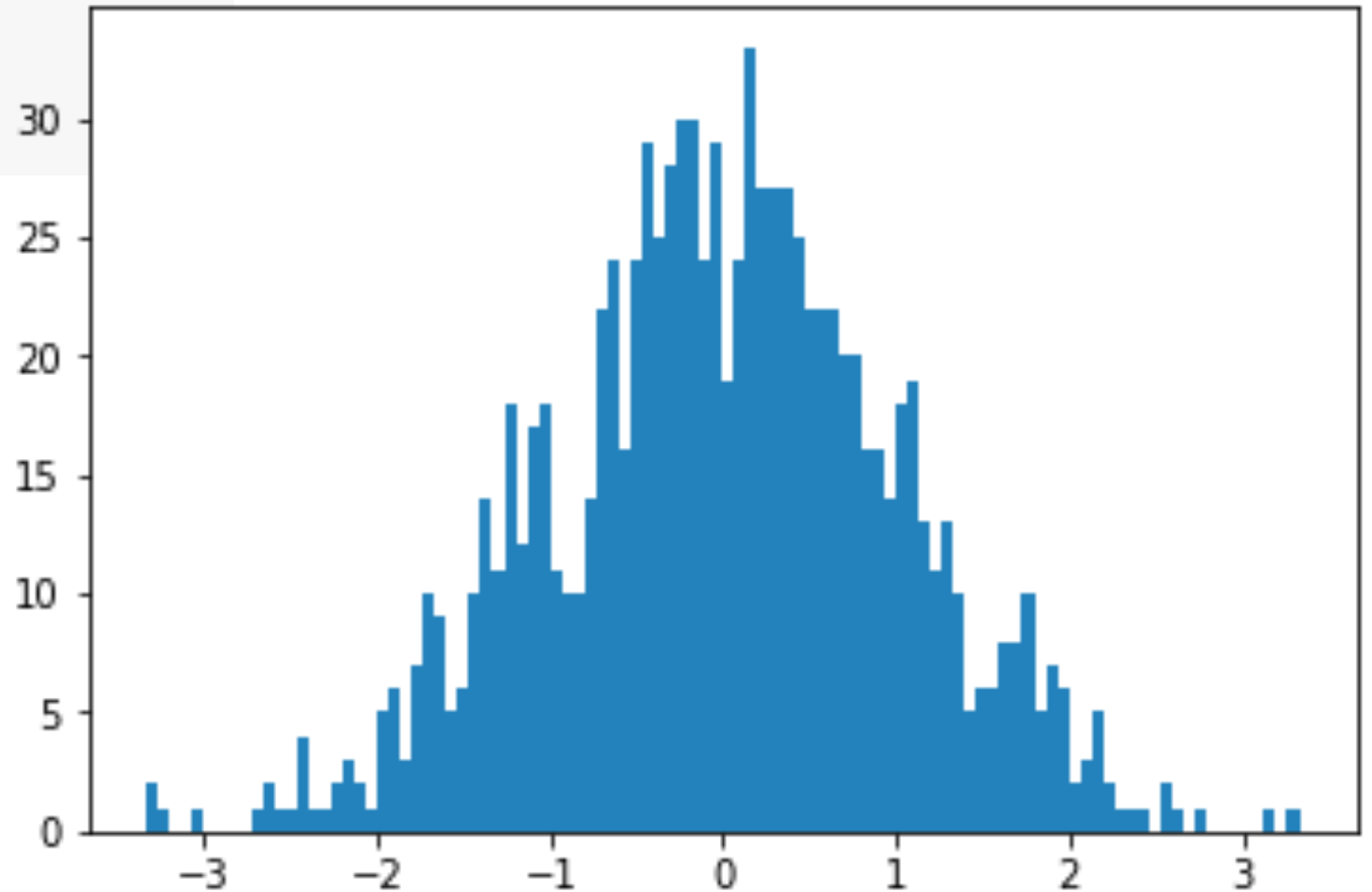
Bar chart

```
color_list = ['b', 'g', 'r']
data_label = ["A", "B", "C"]
X = np.arange(data.shape[1])
for i in range(data.shape[0]):
    plt.bar(X, data[i], bottom = np.sum(data[:i], axis=0),
            color = color_list[i], label=data_label[i])
plt.legend()
plt.show()
```



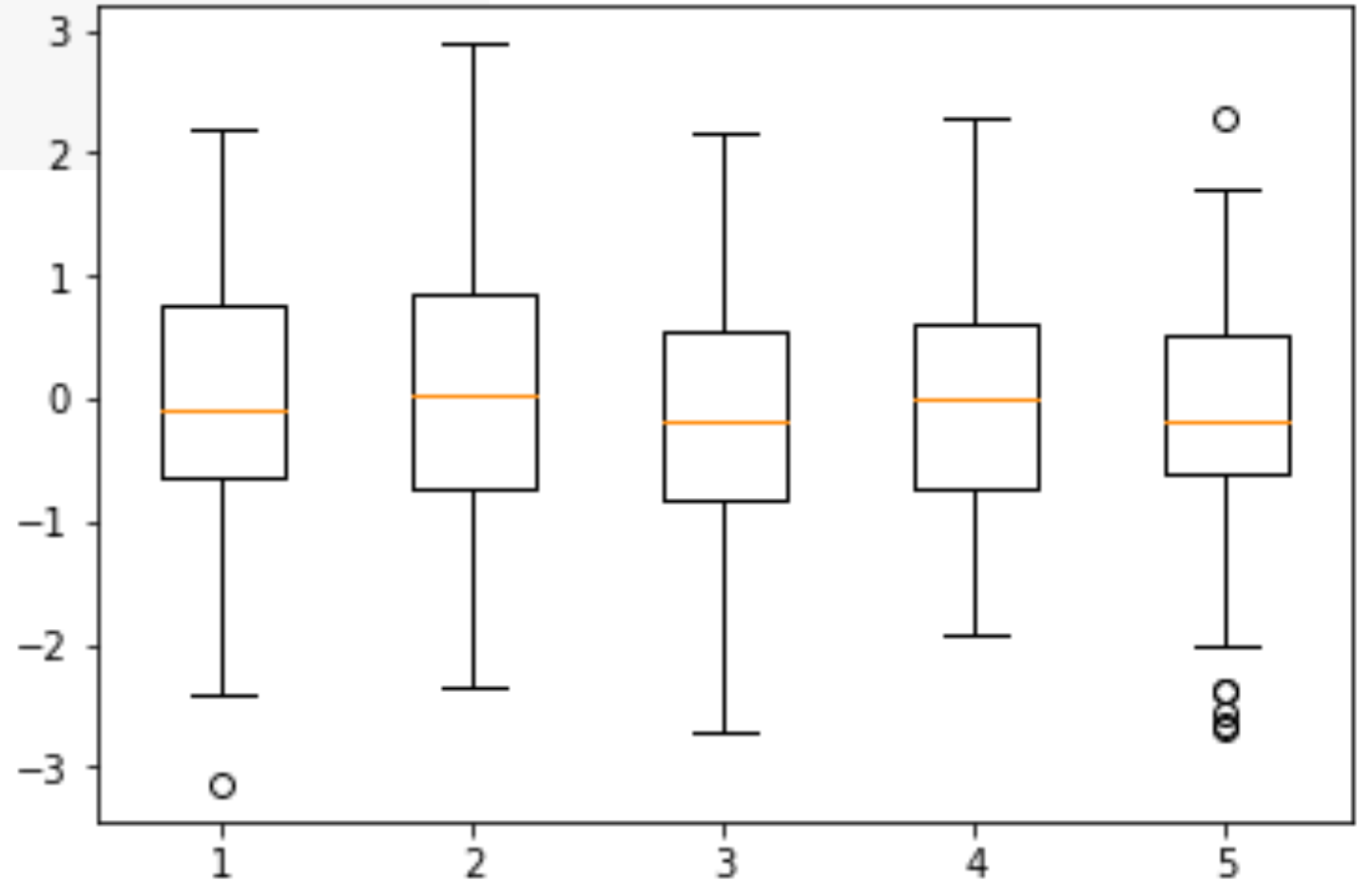
histogram

```
X = np.random.randn(1000)  
plt.hist(X, bins=100)  
plt.show()
```



boxplot

```
data = np.random.randn(100,5)  
plt.boxplot(data)  
plt.show()
```



matplotlib
with pandas

pandas matplotlib

- Pandas 0.7 버전이상 부터 matplotlib를 사용한 그래프 지원
- Dataframe, series별로 그래프 작성 가능

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

Boston House Price Dataset

- 머신 러닝 등 데이터 분석을 처음 배울 때,
가장 대표적으로 사용하는 **Example Dataset**
- 1978년에 발표된 데이터로, 미국 인구통계 조사 결과
미국 보스턴 지역의 주택 가격에 영향 요소들을 정리함

<http://lib.stat.cmu.edu/datasets/boston>

Boston House Price Dataset

X 변수
13개

Y 변수

[01] CRIM	자치시(town) 별 1인당 범죄율
[02] ZN	25,000 평방피트를 초과하는 거주지역의 비율
[03] INDUS	비소매상업지역이 점유하고 있는 토지의 비율
[04] CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
[05] NOX	10ppm 당 농축 일산화질소
[06] RM	주택 1가구당 평균 방의 개수
[07] AGE	1940년 이전에 건축된 소유주택의 비율
[08] DIS	5개의 보스턴 직업센터까지의 접근성 지수
[09] RAD	방사형 도로까지의 접근성 지수
[10] TAX	10,000 달러 당 재산세율
[11] PTRATIO	자치시(town)별 학생/교사 비율
[12] B	$1000(Bk - 0.63)^2$, 여기서 Bk는 자치시별 흑인의 비율을 말함.
[13] LSTAT	모집단의 하위계층의 비율(%)
[14] MEDV	본인 소유의 주택가격(중양값) (단위: \$1,000)

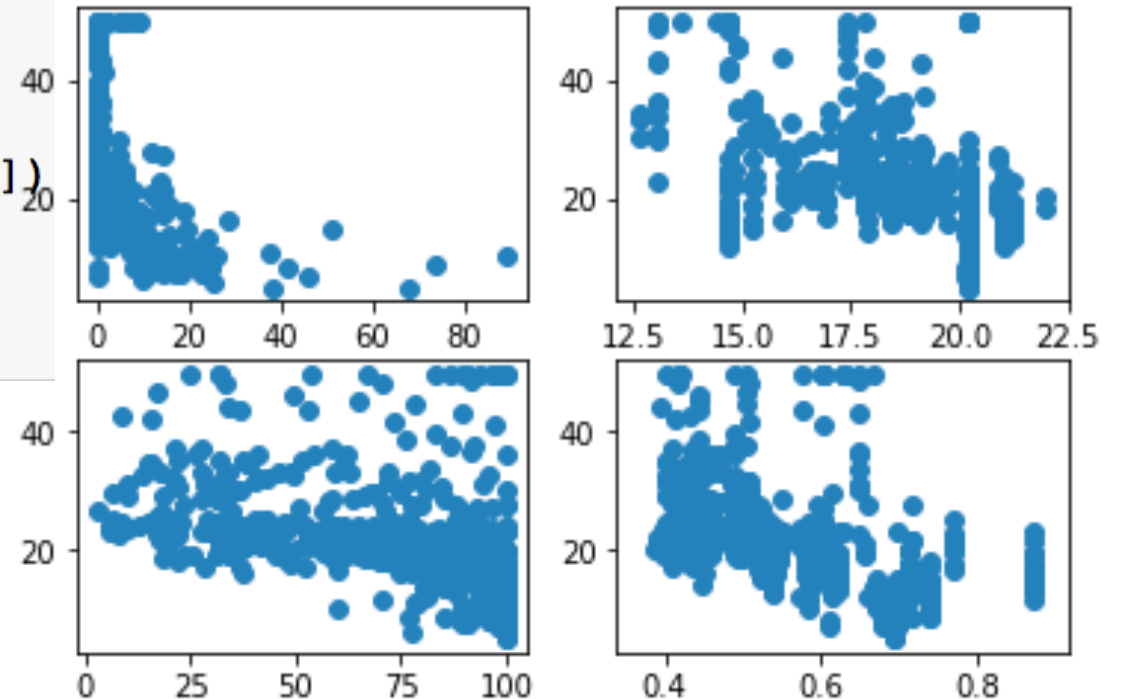
<http://www.dator.co.kr/ctg258/textyle/1721307>

<http://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html>

pandas matplotlib

- 데이터간의 상관관계를 볼 때 scatter graph 사용 가능

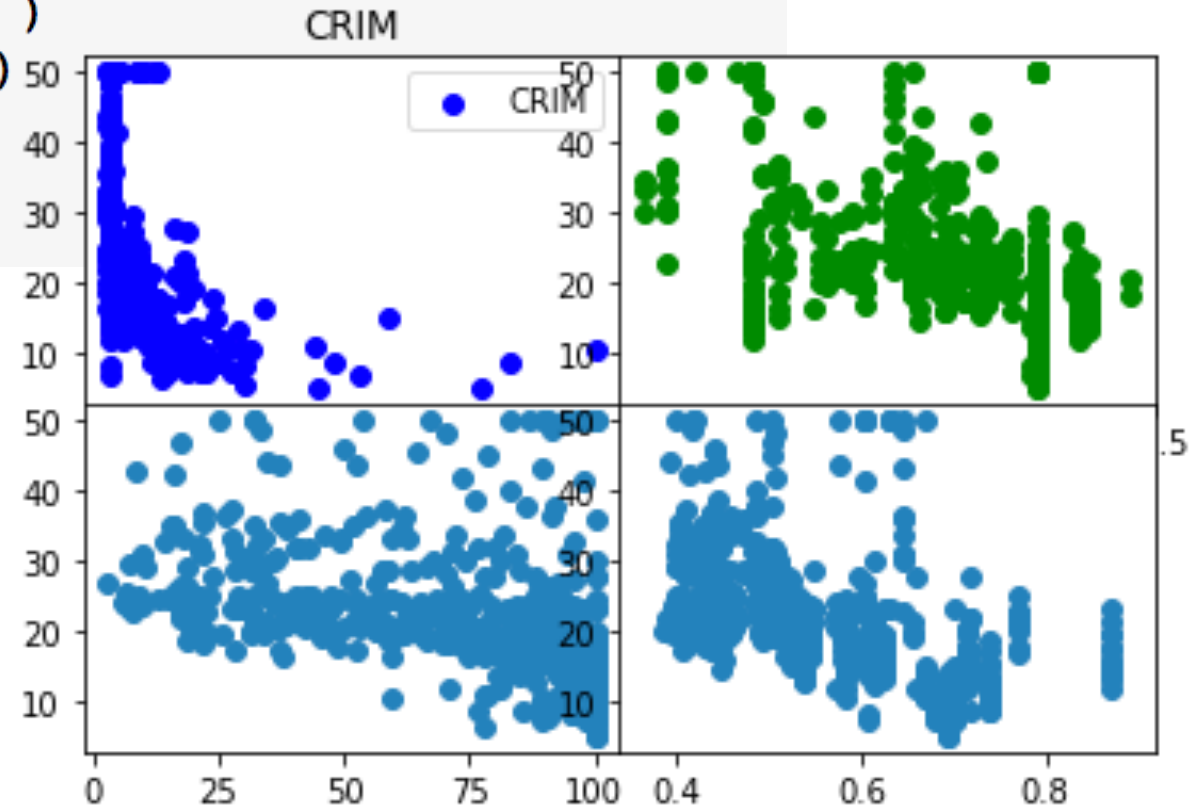
```
fig = plt.figure()
ax = []
for i in range(1,5):
    ax.append(fig.add_subplot(2,2,i))
ax[0].scatter(df_data["CRIM"], df_data["MEDV"])
ax[1].scatter(df_data["PTRATIO"], df_data["MEDV"])
ax[2].scatter(df_data["AGE"], df_data["MEDV"])
ax[3].scatter(df_data["NOX"], df_data["MEDV"])
plt.show()
```



pandas matplotlib

- matplotlib의 꾸미기 함수 그대로 사용함

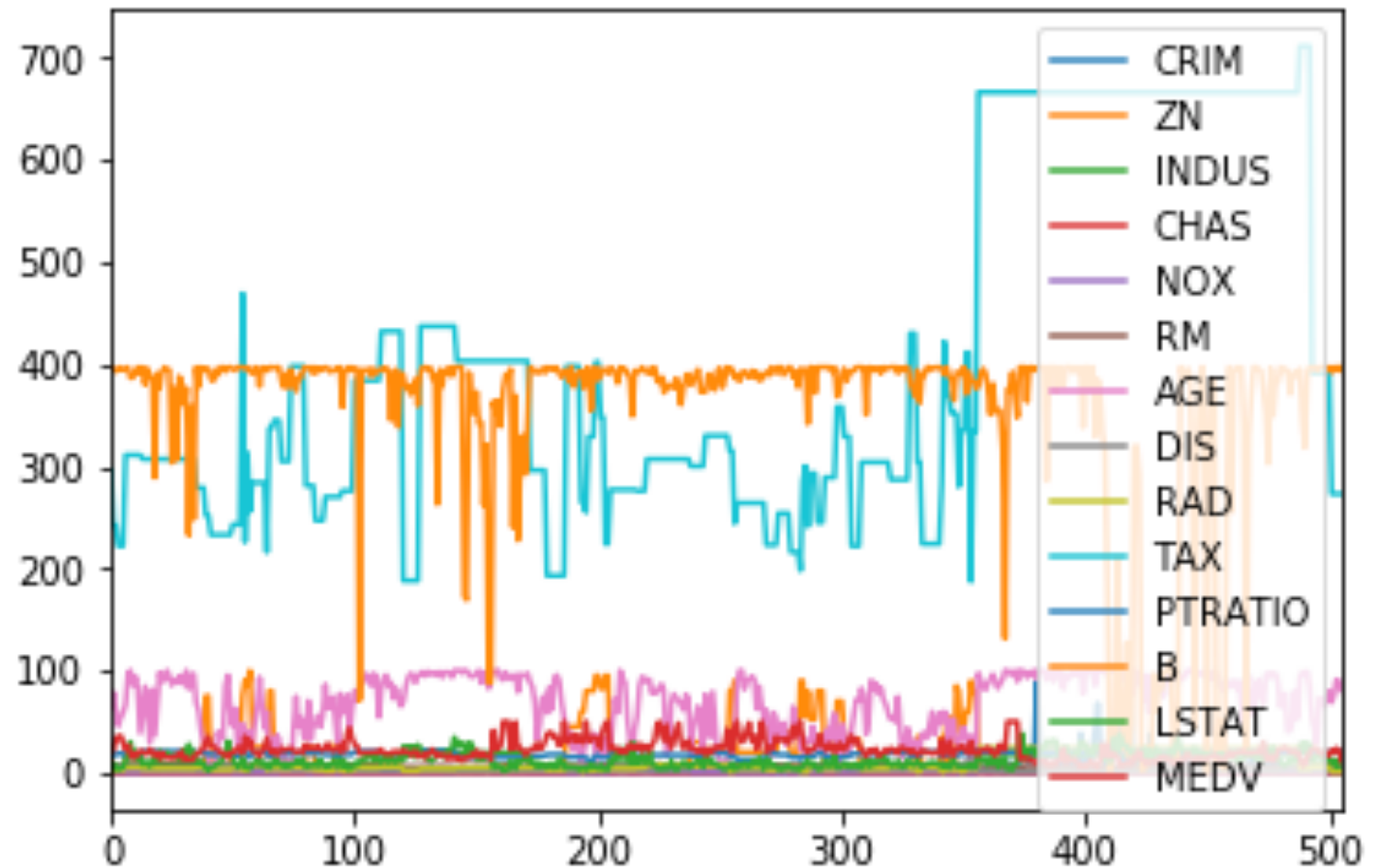
```
ax[0].scatter(df_data["CRIM"], df_data["MEDV"], color="b", label="CRIM")
ax[1].scatter(df_data["PTRATIO"], df_data["MEDV"], color="g" )
ax[2].scatter(df_data["AGE"], df_data["MEDV"] )
ax[3].scatter(df_data["NOX"], df_data["MEDV"])
plt.subplots_adjust(wspace=0, hspace=0)
ax[0].legend()
ax[0].set_title("CRIM")
```



pandas matplotlib

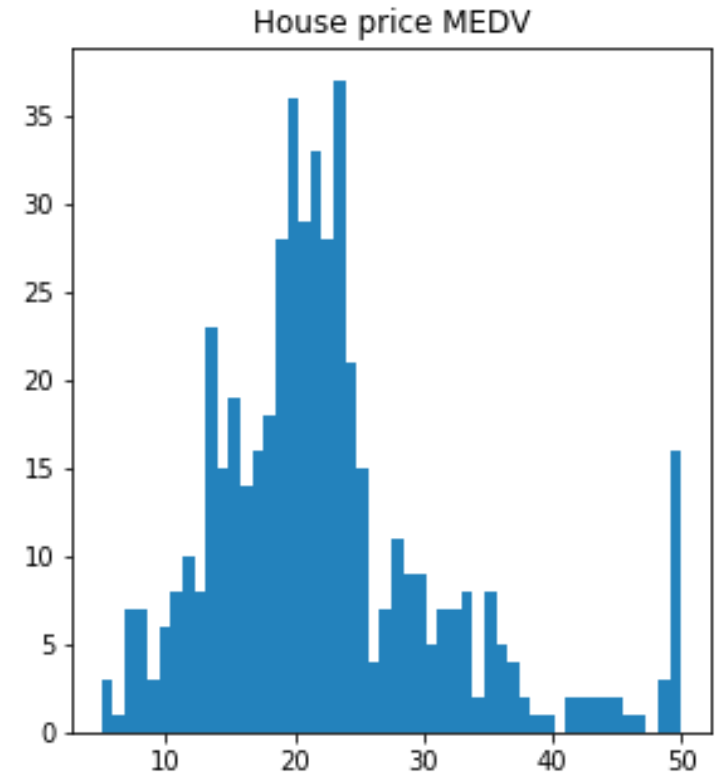
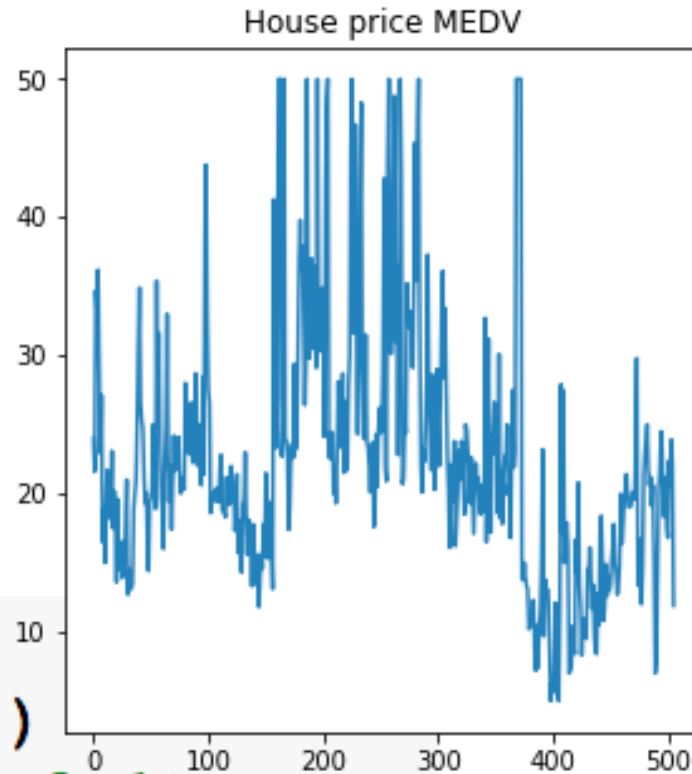
- plot 함수를 사용하면 전체 데이터의 graph를 그림

```
df_data.plot()
```



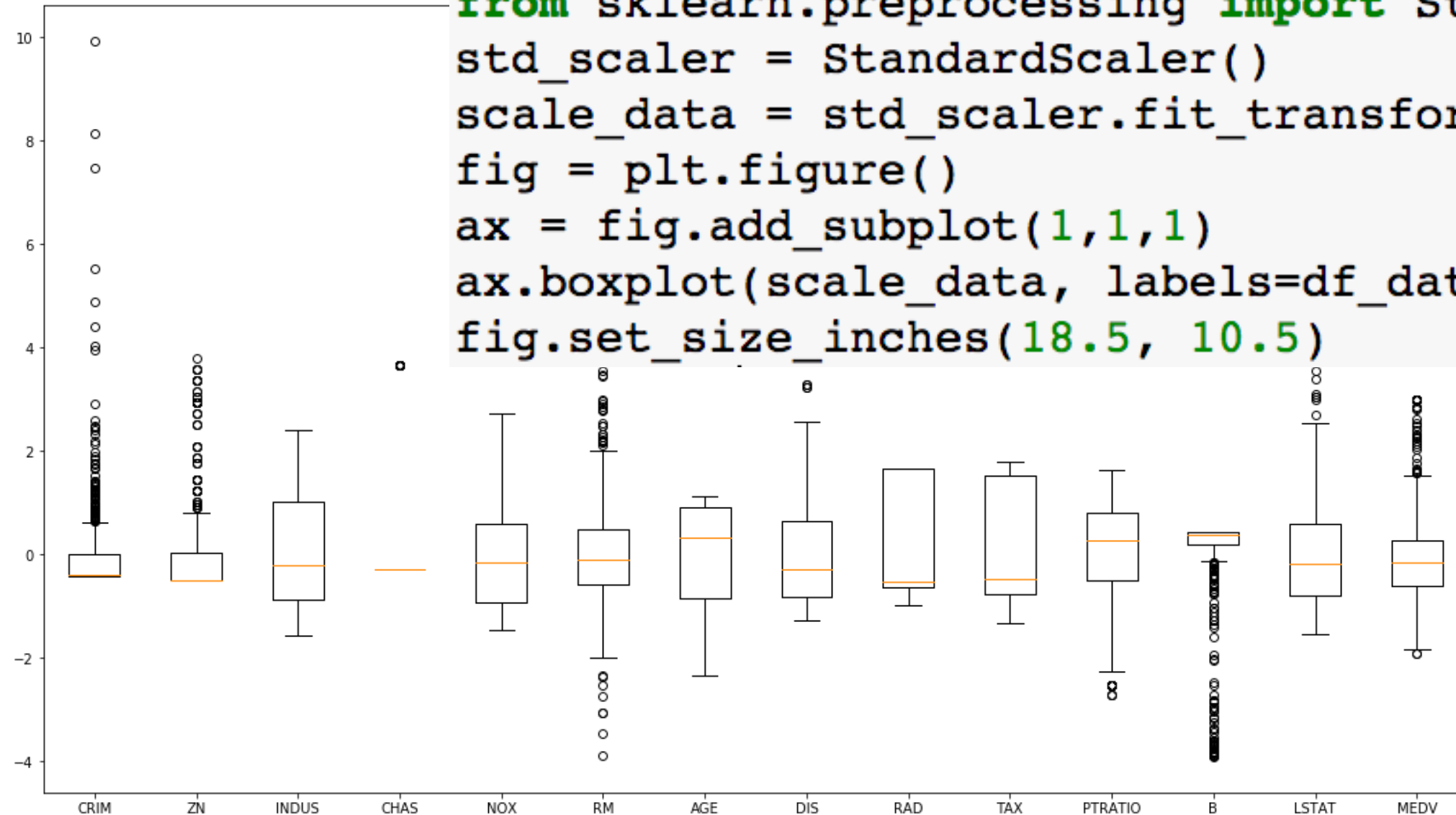
pandas matplotlib

```
fig = plt.figure()
fig.set_size_inches(10,5)
ax_1 = fig.add_subplot(1,2,1)
ax_2 = fig.add_subplot(1,2,2)
ax_1.plot(df_data["MEDV"])
ax_2.hist(df_data["MEDV"], bins=50)
ax_1.set_title("House price MEDV")
ax_2.set_title("House price MEDV")
```



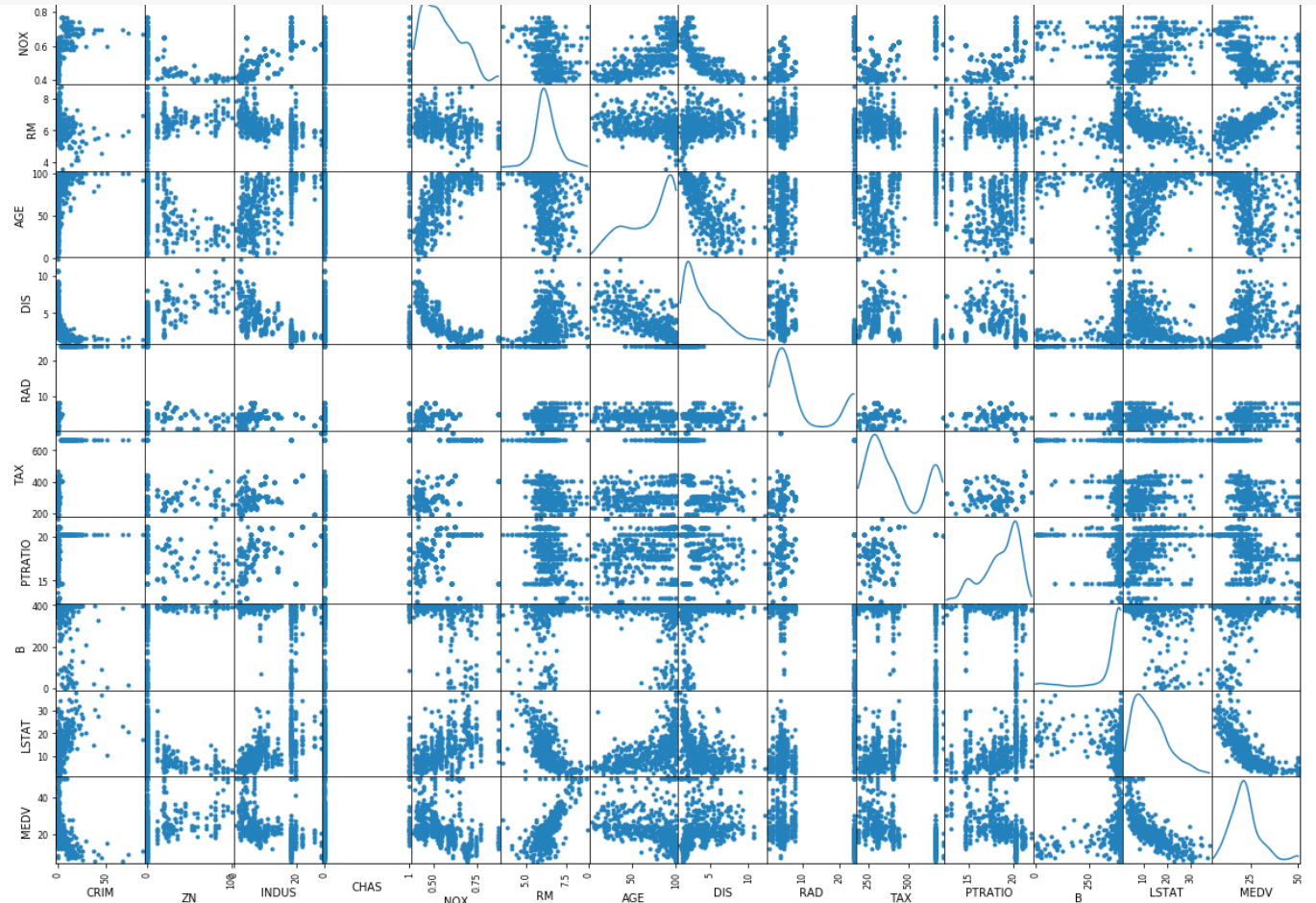
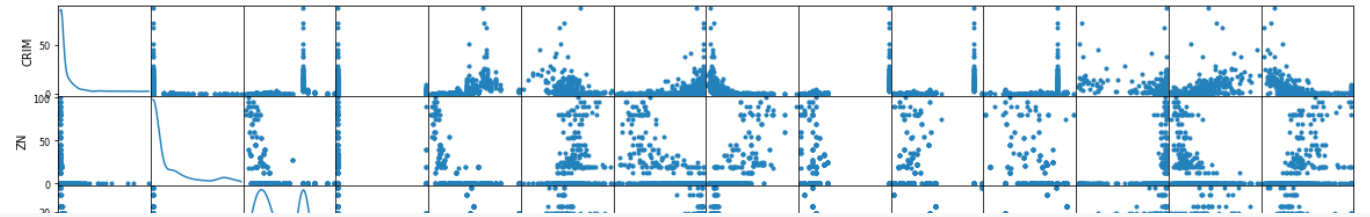
Scaled boxplot

```
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
scale_data = std_scaler.fit_transform(df_data)
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.boxplot(scale_data, labels=df_data.columns)
fig.set_size_inches(18.5, 10.5)
```



Scatter matrix

```
pd.scatter_matrix(df_data, diagonal="kde", alpha=1, figsize=(20, 20))  
plt.show()
```



Scatter matrix

```
fig = plt.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(corr_data, vmin=-1, vmax=1, interpolation='nearest')
fig.colorbar(cax)
fig.set_size_inches(10,10)
ticks= np.arange(0,14,1)
ax.set_xticks(ticks)
ax.set_yticks(ticks)

ax.set_xticklabels(df_data.columns)
ax.set_yticklabels(df_data.columns)
```

