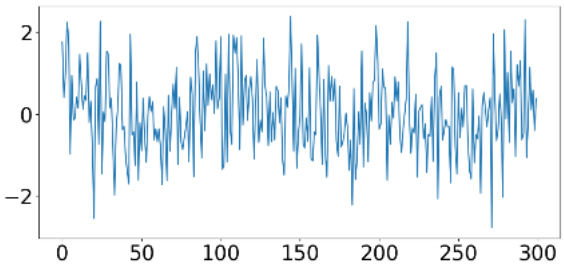
ML Day14 (Matplotlib)

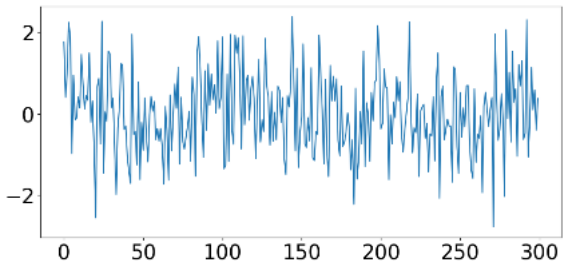
▼ ax.plot(y) (1)

```
loc=0 평균, scale=1 표준편차, size=(300,) 갯수

import matplotlib.pyplot as plt
import numpy as np

np.random.seed(0)

y_data = np.random.normal(loc=0,  scale=1, size=(300,))  # loc : 평균, scale : 표준편차

fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(y_data)

fig.tight_layout(pad=3)                                  # pad : 여백 크기 조정
ax.tick_params(labelsize=25)
plt.show()
```

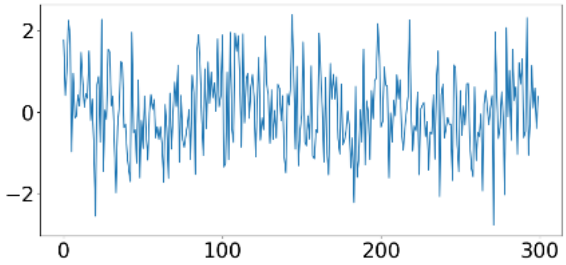


▼ ax.plot(y) (2)

```
np.random.seed(0)

y_data = np.random.normal(loc=0, scale=1, size=(300,))

fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(y_data)

fig.tight_layout(pad=3)
ax.tick_params(labelsize=25)

x_ticks = np.arange(301, step=50)     # np.arange(시작점(생략 시0), 끝점(생략 시 미포함), step(생략 시1)
ax.set_xticks(x_ticks)
plt.show()
```

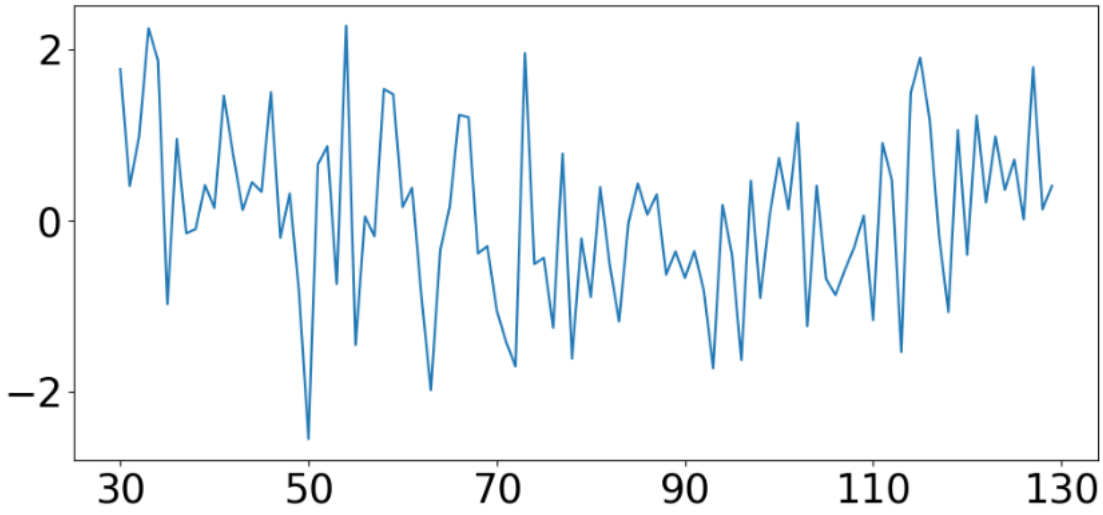



```
np.random.seed(0)

y_data = np.random.normal(loc=0, scale=1, size=(300,))

fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(y_data)

fig.tight_layout(pad=3)
ax.tick_params(labelsize=25)

x_ticks = np.arange(301, step=100)
ax.set_xticks(x_ticks)
plt.show()
```



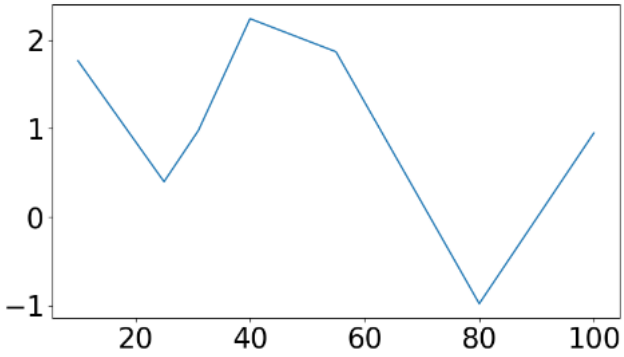
▼ ax.plot(x, y) (1)

```
np.random.seed(0)

n_data = 100
s_idx = 30
x_data = np.arange(s_idx, s_idx + n_data)      # arange - 특정 수열을 만들 때(start, end, step)
y_data = np.random.normal(0, 1, (n_data, ))

fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(x_data, y_data)

fig.tight_layout(pad=3)                        # 상하좌우 여백을 하나의 수치로 조정
x_ticks = np.arange(s_idx, s_idx + n_data + 1, 20)
ax.set_xticks(x_ticks)

ax.tick_params(labelsize=25)
plt.show()
```



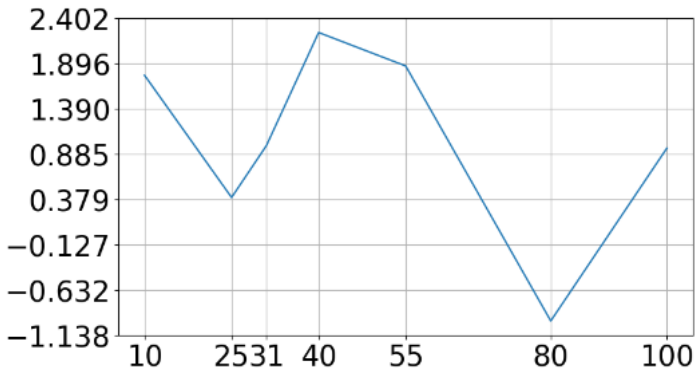
▼ **ax.plot(x, y) (2)**

```
np.random.seed(0)

x_data = np.array([10, 25, 31, 40, 55, 80, 100])
y_data = np.random.normal(0, 1, (7, ))

fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(x_data, y_data)

fig.subplots_adjust(left=0.2)  # figure의 상하좌우 여백 미세조정
ax.tick_params(labelsize=25)
plt.show()
```



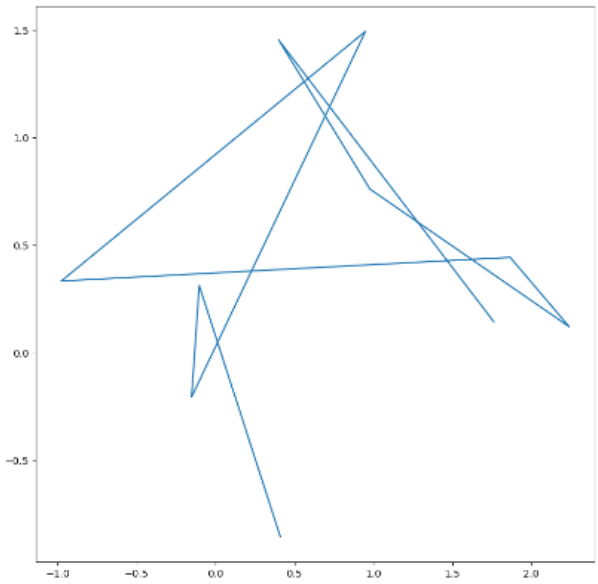

```
np.random.seed(0)

x_data = np.array([10, 25, 31, 40, 55, 80, 100])
y_data = np.random.normal(0, 1, (7, ))

fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(x_data, y_data)

fig.subplots_adjust(left=0.2)               # figure의 상하좌우 여백 미세조정
ax.tick_params(labelsize=25)

ax.set_xticks(x_data)
ylim = ax.get_ylim()                        # get을 이용해 ylim()의 (최소값, 최대값) 형태로 ylim 변수에 대입
yticks = np.linspace(ylim[0], ylim[1], 8)   # 최소값에서 최대값 범위 사이 8개의 값을 yticks으로
ax.set_yticks(yticks)

ax.grid()                                   # 격자 생성
```

```
plt.show()
```



▼ **ax.plot(x,y) (3)**

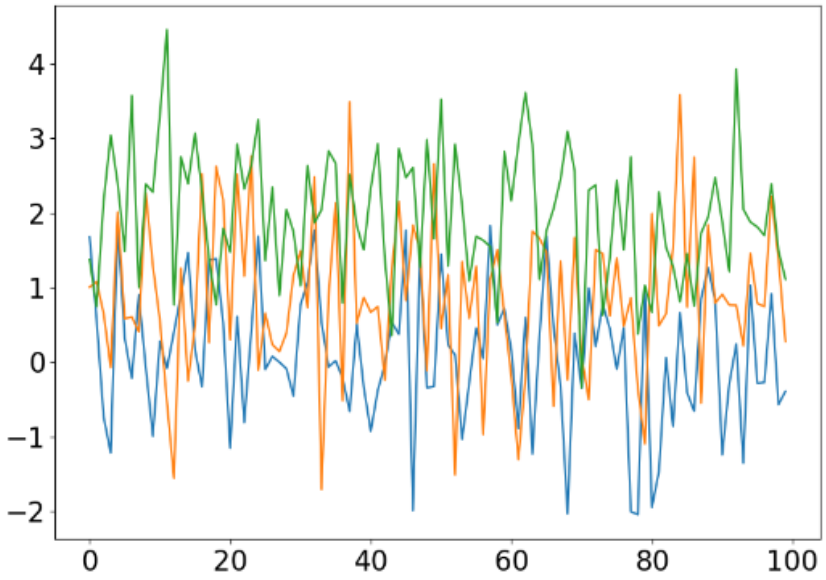
```
np.random.seed(0)

x_data = np.random.normal(0, 1, (10, ))
y_data = np.random.normal(0, 1, (10, ))

fig, ax = plt.subplots(figsize=(10, 10))
ax.plot(x_data, y_data)
plt.show()
```



▼ **Several Line Plots on One Ax (1)**

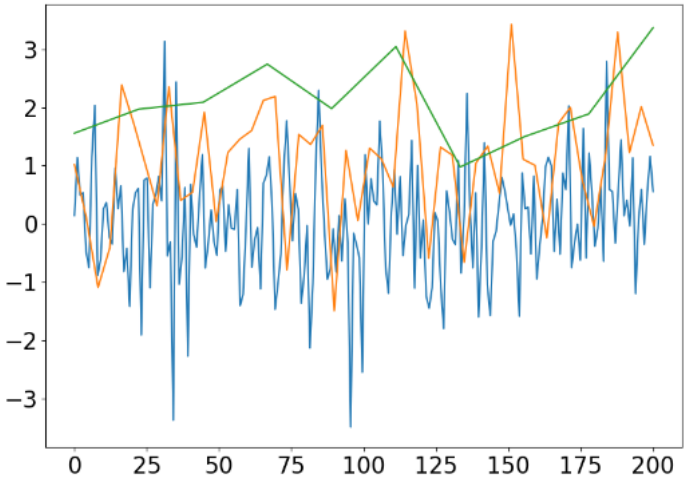
```
n_data = 100

random_noise1 = np.random.normal(0, 1, (n_data,))    # 각각의 평균을 달리하여 변수 지정
random_noise2 = np.random.normal(1, 1, (n_data,))
random_noise3 = np.random.normal(2, 1, (n_data,))

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(random_noise1)
```

```
ax.plot(random_noise2)
ax.plot(random_noise3)

ax.tick_params(labelsize=20)
plt.show()
```



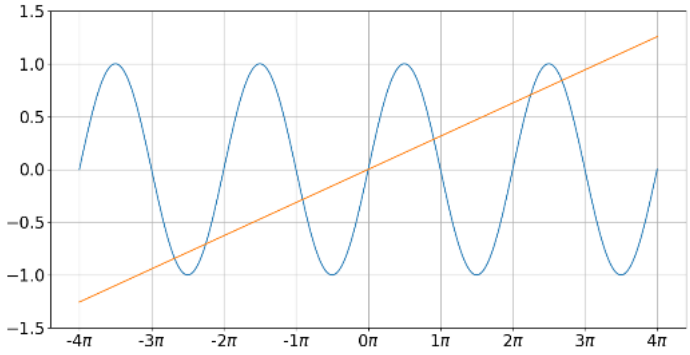
▼ Several Line Plots on One Ax (2)

```
n_data1, n_data2, n_data3 = 200, 50, 10     # n_data의 갯수를 달리함

x_data1 = np.linspace(0, 200, n_data1)
x_data2 = np.linspace(0, 200, n_data2)
x_data3 = np.linspace(0, 200, n_data3)

random_noise1 = np.random.normal(0, 1, (n_data1,))
random_noise2 = np.random.normal(1, 1, (n_data2,))
random_noise3 = np.random.normal(2, 1, (n_data3,))

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(x_data1, random_noise1)
ax.plot(x_data2, random_noise2)
ax.plot(x_data3, random_noise3)

ax.tick_params(labelsize=20)
plt.show()
```



▼ Several Line Plots on One Ax (3)

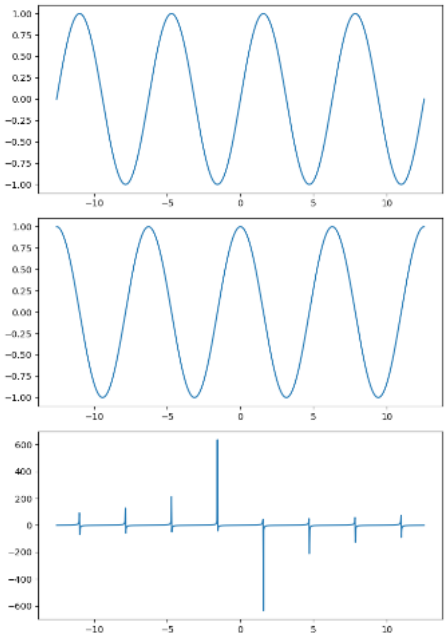
```
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 300)
sin = np.sin(t)
linear = 0.1*t

fig, ax = plt.subplots(figsize=(14, 7))
ax.plot(t, sin)
ax.plot(t, linear)

ax.set_ylim([-1.5, 1.5])

x_ticks = np.arange(-4*PI, 4*PI+0.1, PI)
x_ticklabels = [str(i) + r'$\pi$'            # x_ticklabels명 latex문자 -> r'$\문자$'
                for i in range(-4, 5)]
ax.set_xticks(x_ticks)
ax.set_xticklabels(x_ticklabels)

ax.tick_params(labelsize=20)
ax.grid()
plt.show()
```



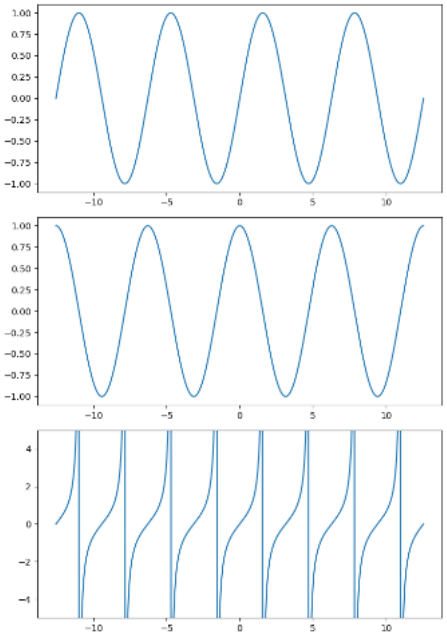
▼ Several Line Plots on Different Axes (1)

```
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 1000)
sin = np.sin(t)
cos = np.cos(t)
tan = np.tan(t)

fig, axes = plt.subplots(3, 1,               # (3, 1)의 plot을 만든다.
```

```
                        figsize=(7, 10))

axes[0].plot(t, sin)                        # index로 각각의 그래프의 axes 생성
axes[1].plot(t, cos)
axes[2].plot(t, tan)

fig.tight_layout()
plt.show()
```



```
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 1000)
sin = np.sin(t)
cos = np.cos(t)
tan = np.tan(t)

fig, axes = plt.subplots(3, 1,
                         figsize=(7, 10))

axes[0].plot(t, sin)
axes[1].plot(t, cos)
axes[2].plot(t, tan)              # matplotlib의 plot의 특성 상 별도로 code를 구현해야
                                  # 제대로 된 tan그래프를 그릴 수 있다.
fig.tight_layout()
axes[2].set_ylim([-5, 5])         # 위 코드와의 차이 -> tan그래프의 lim설정
plt.show()
```

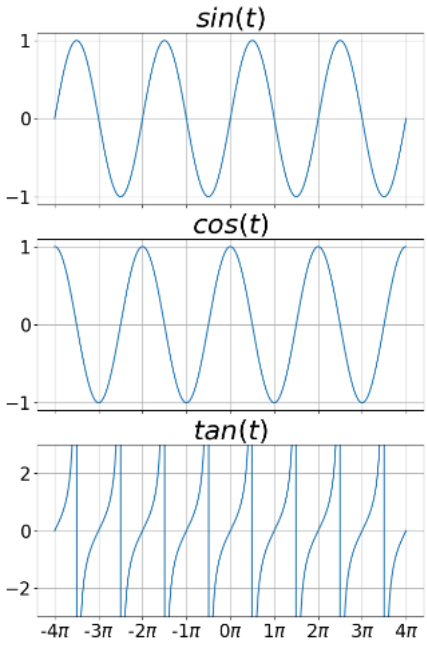


—> tan그래프는 사실 이어짐없이 위, 아래로 발산되는 그래프이나 matplotlib.plot(점들을 이어주는) 특성 상 발산되는 그래프를 제대로 그려줄 수 없는 한계가 있다. → bool index를 활용하여 발산그래프 표현 가능

▼ Several Line Plots on Different Axes (2)

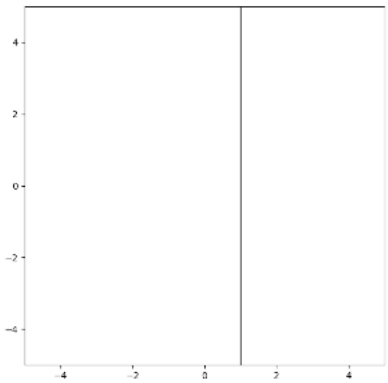
```
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 1000).reshape(1, -1)     # vector를 reshape(1, -1)하여 행렬로 치환 / (1, -1) -1은 앞 수에 맞춰 자동으로 reshap
sin = np.sin(t)
cos = np.cos(t)
tan = np.tan(t)
data = np.vstack((sin, cos, tan))     # vstack(vertical stack) - 수직으로 데이터를 쌓아준다.

title_list = [r'$sin(t)$', r'$cos(t)$', r'$tan(t)$']
x_ticks = np.arange(-4*PI, 4*PI+PI, PI)
x_ticklabels = [str(i) + r'$\pi$' for i in range(-4, 5)]

fig, axes = plt.subplots(3, 1,            # 행렬 형태인 (3, 1) 모양으로 그래프를 그린다.
                         figsize=(7, 10),
                         sharex=True)     # sharex=True 세개의 그래프가 하나의 x축을 공유한다.

for ax_idx, ax in enumerate(axes.flat):   # for문을 이용하여 각각의 axe에 접근하여 그래프를 그린다.
    ax.plot(t.flatten(), data[ax_idx])    # flat / flatten : array들을 vector로 변환하여 for문을 돌릴 수 있도록 한줄로 만든다.
    ax.set_title(title_list[ax_idx],
                 fontsize=30)
    ax.tick_params(labelsize=20)
    ax.grid()
    if ax_idx == 2:
        ax.set_ylim([-3, 3])

fig.subplots_adjust(left=0.1, right=0.95,
                    bottom=0.05, top=0.95)
axes[-1].set_xticks(x_ticks)               # -1 index인 마지막 그래프의 xticks의 lables등을 설정한다.(sharex=True)
axes[-1].set_xticklabels(x_ticklabels)
plt.show()
```

→ 이 code에선 위 plot을 만들때 (3, 1)형태로 만들었기 때문에 flat을 쓰지 않아도 for문을 돌릴 수 있지만, 만약 (3, 2)처럼 여러 겹의 그래프로 만들었을 경우 flat을 해주어야 for문이 작동한다.

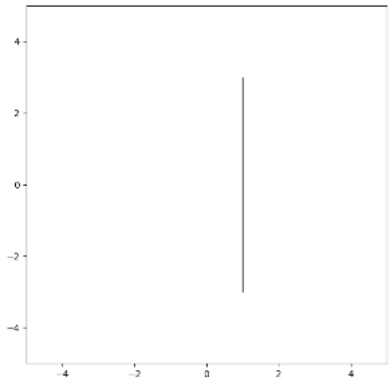


▼ **ax.axvline** and ax.axhline (점근선, 평균 등을 그래프 상에서 표현하고 싶을 때 사용된다.)

```
fig, ax = plt.subplots(figsize=(7, 7))

ax.set_xlim([-5, 5])
ax.set_ylim([-5, 5])

ax.axvline(x=1,
           color='black',
           linewidth=1)
plt.show()
```

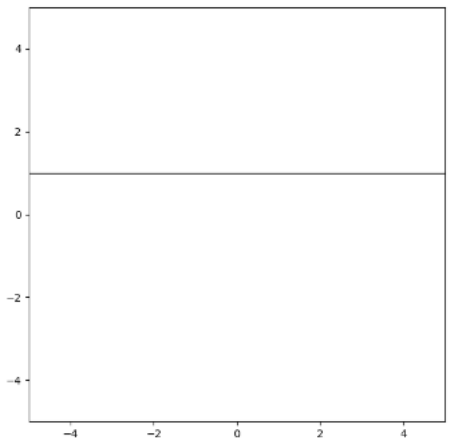


```
fig, ax = plt.subplots(figsize=(7, 7))

ax.set_xlim([-5, 5])
ax.set_ylim([-5, 5])

ax.axvline(x=1,
           ymax=0.8, ymin=0.2,      # ymax를 plot의 0.8비율로, ymin을 plot의 0.2비율로
           color='black',
           linewidth=1)
plt.show()
```

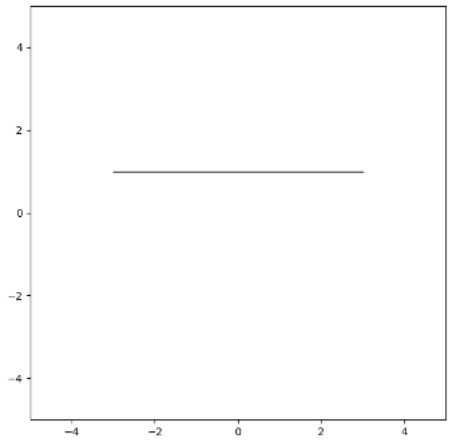


▼ ax.axvline and **ax.axhline**

```
fig, ax = plt.subplots(figsize=(7, 7))

ax.set_xlim([-5, 5])
ax.set_ylim([-5, 5])

ax.axhline(y=1,
           color='black',
           linewidth=1)
plt.show()
```

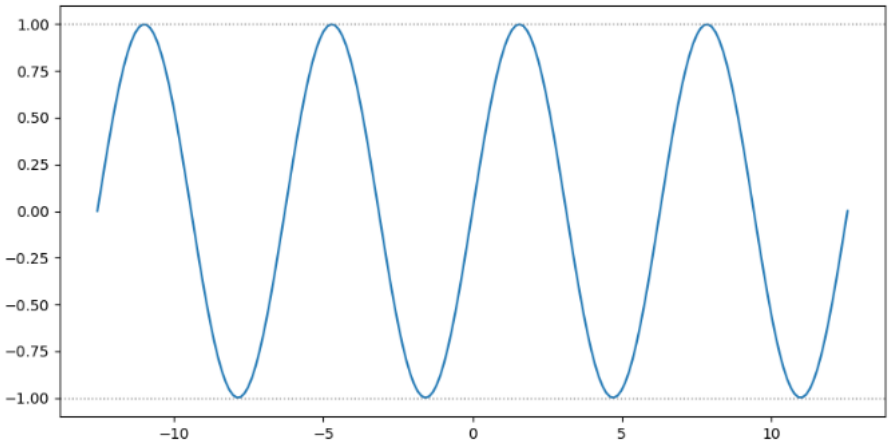


```
fig, ax = plt.subplots(figsize=(7, 7))

ax.set_xlim([-5, 5])
ax.set_ylim([-5, 5])

ax.axhline(y=1,
           xmax=0.8, xmin=0.2,
           color='black',
           linewidth=1)
plt.show()
```



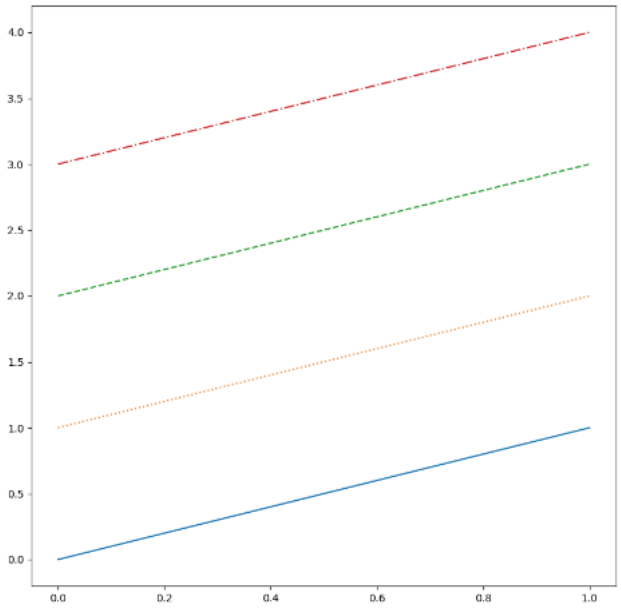
▼ ax.axvline and ax.axhline

```
x = np.linspace(-4*np.pi, 4*np.pi, 200)
sin = np.sin(x)

fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(x, sin)
ax.axhline(y=1, ls=':', lw=1, color='gray')     # ls : linestyle, lw : line width
ax.axhline(y=-1, ls=':', lw=1, color='gray')

plt.show()
```



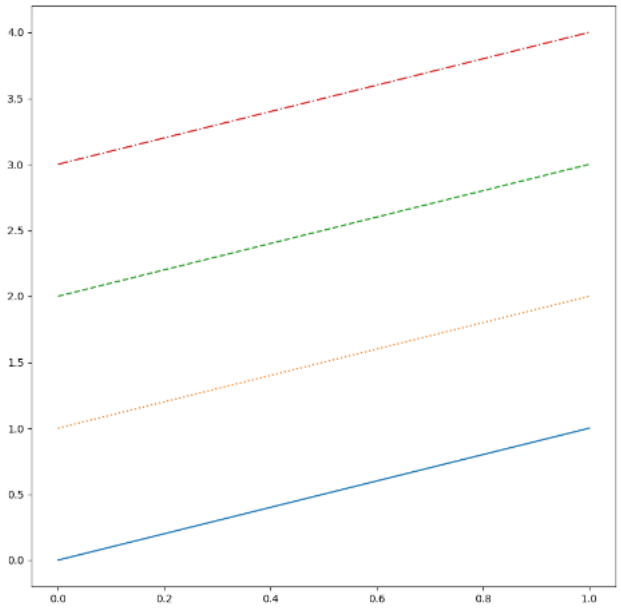
▼ Line Styles (1)

```
x_data = np.array([0, 1])
y_data = x_data

fig, ax = plt.subplots(figsize=(10, 10))

ax.plot(x_data, y_data)

ax.plot(x_data, y_data+1,
        linestyle='dotted')     # linestyle = ls로 축약가능

ax.plot(x_data, y_data+2,
        linestyle='dashed')

ax.plot(x_data, y_data+3,
        linestyle='dashdot')
plt.show()
```



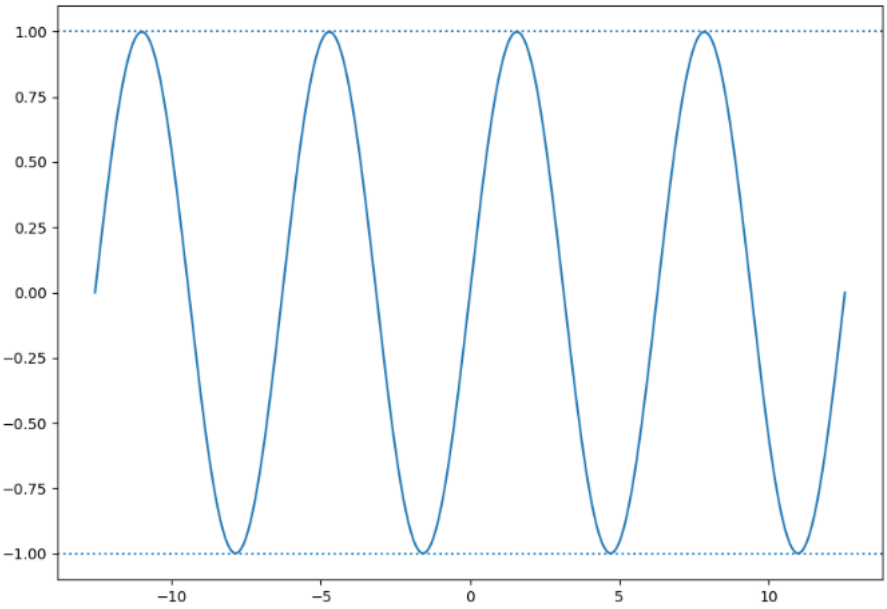
▼ **Line Style (2)**

```
x_data = np.array([0, 1])
y_data = x_data

fig, ax = plt.subplots(figsize=(10, 10))

ax.plot(x_data, y_data)
ax.plot(x_data, y_data+1,
        ls=':')
ax.plot(x_data, y_data+2,
        ls='--')
ax.plot(x_data, y_data+3,
        ls='-.')
plt.show()
```



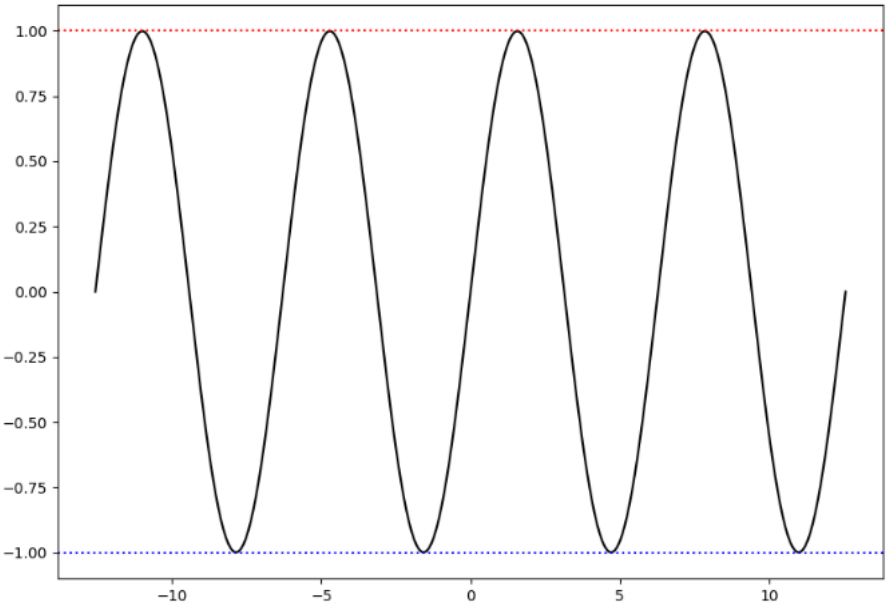
▼ Line Style (3)

```
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 300)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin)

ax.axhline(y=1,
           linestyle=':')
ax.axhline(y=-1,
           linestyle=':')
plt.show()
```

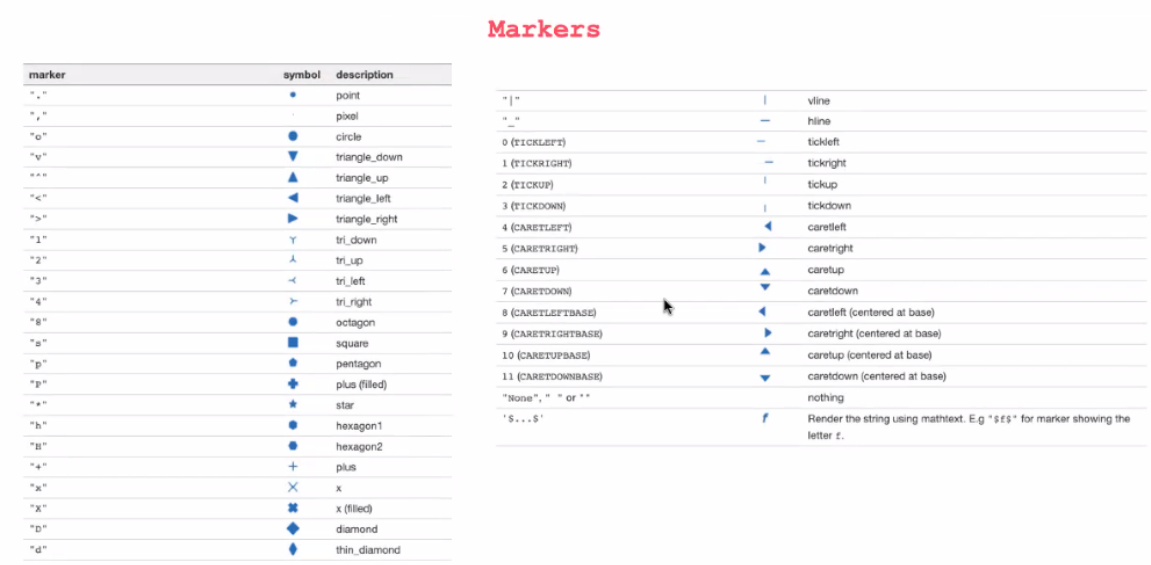


▼ Line Style (4)

```
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 300)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin,
        color='black')
ax.axhline(y=1,
           linestyle=':',
           color='red')
ax.axhline(y=-1,
           linestyle=":",
           color='blue')
plt.show()
```

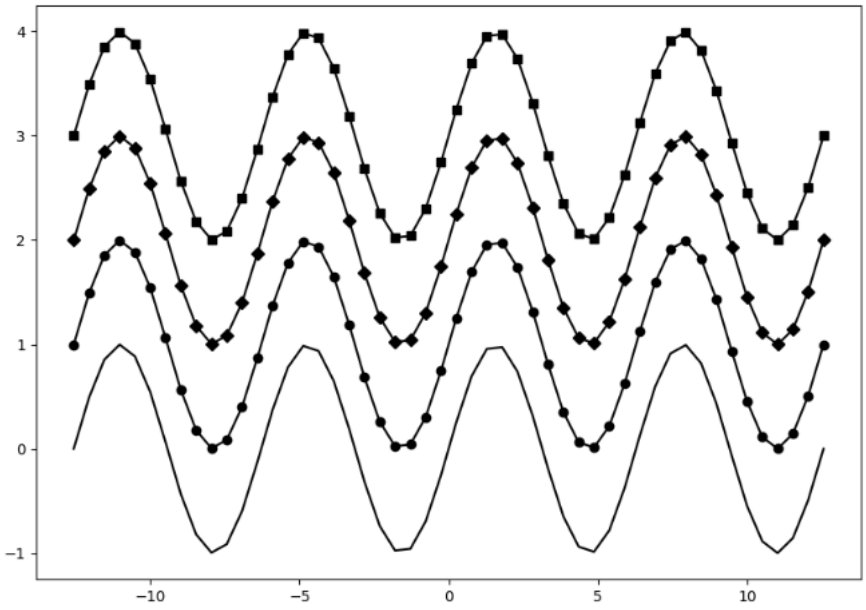


▼ **Markers**



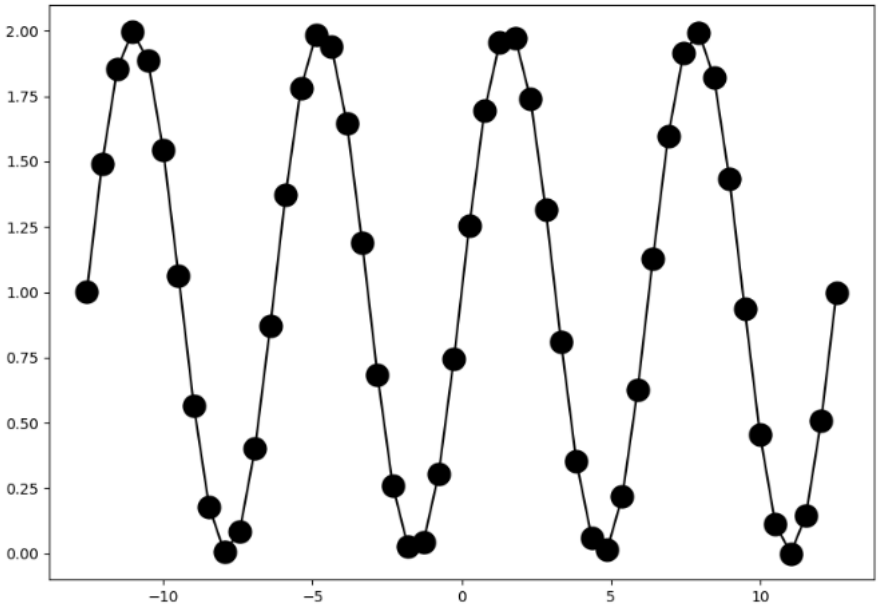

```
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 50)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin,
        color='black')
ax.plot(t, sin+1,
        marker='o',        # marker = '', parameter를 지정하여 marker를 쓰면 그래프 선에 marker표현이 가능
        color='black')
ax.plot(t, sin+2,
        marker='D',
        color='black')
ax.plot(t, sin+3,
        marker='s',
        color='black')
plt.show()
```

→ plot을 생성할 때 parameter(marker)를 지정하지 않고 ‘o’를 쓰면 선을 잇지 않고 scatter처럼 표현이 가능하다.



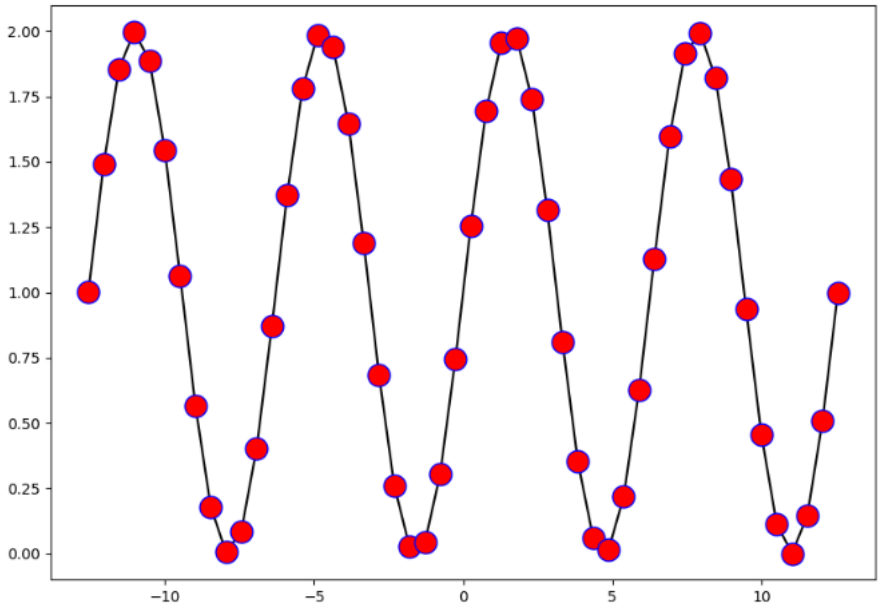
▼ **Customizing Markers (1)**

```
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 50)
sin=np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin+1,
        marker='o',
        color='black',
        markersize=15)      # marker size 조정
plt.show()
```



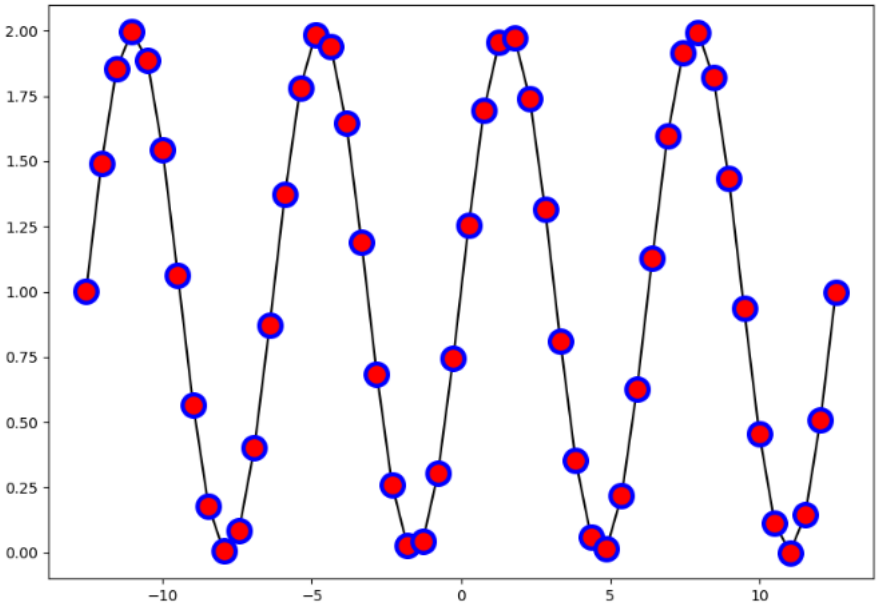
▼ Customizing Markers (2)

```
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 50)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin+1,
        marker='o',
        color='black',
        markersize=15,
        markerfacecolor='r',    # markerfacecolor : marker 색 지정
        markeredgecolor='b')    # markeredgecolor : marker 테두리 색 지정
plt.show()
```



▼ Customizing Markers (3)

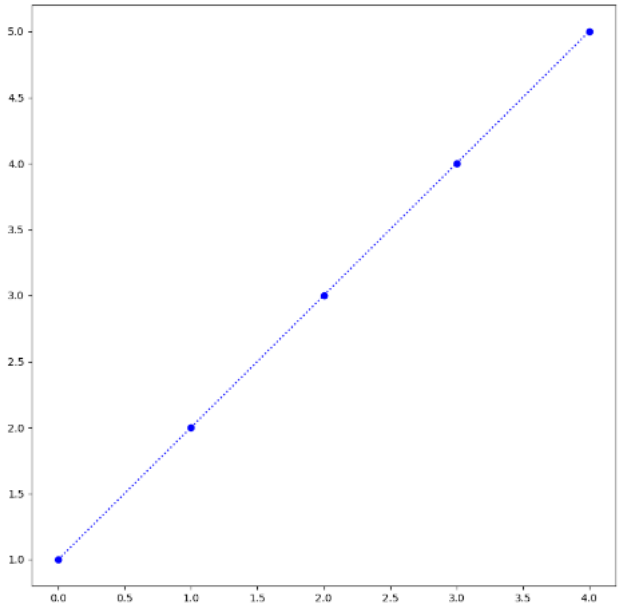
```
PI = np.pi
t = np.linspace(-4*PI, 4*PI, 50)
sin = np.sin(t)

fig, ax = plt.subplots(figsize=(10, 7))

ax.plot(t, sin+1,
        marker='o',
        color='black',
        markersize=15,
        markerfacecolor='r',
        markeredgecolor='b',
        markeredgewidth=3)        # marker 테두리 너비 조정
plt.show()
```



▼ **fmt Argument**

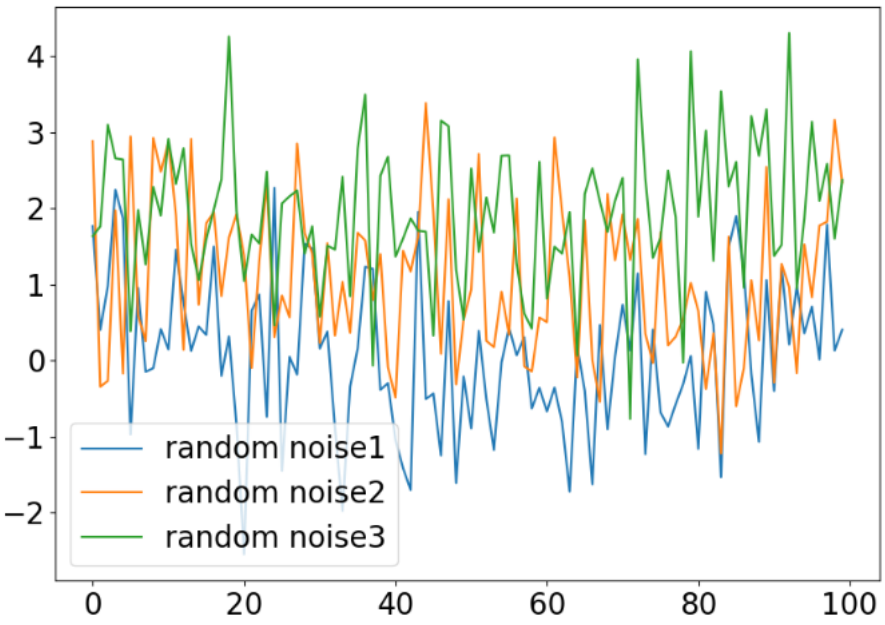
```
x_data = np.array([1, 2, 3, 4, 5])
y_data = x_data
fig, ax = plt.subplots(figsize=(10, 10))

ax.plot(x_data, y_data,
        linestyle=':',
        marker='o',
        color='b')
# ax.plot(x_data, y_data, 'bo:')
plt.show()
```

→ax.plot(x_data, y_data, ':ob')← 위 plot 생성 코드를 한 줄로 쓰기 가능 (marker, color 등 parameter가 겹치지 않기 때문에 순서를 바꿔써도 적용된다.)



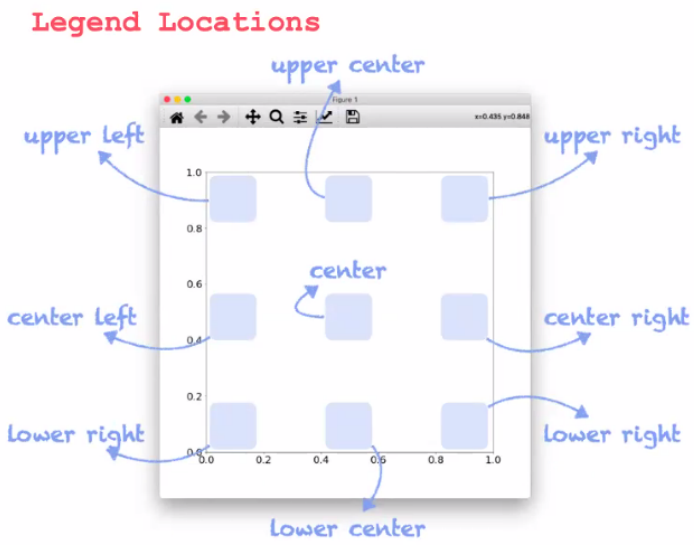
▼ **Basic Usage of Legend**

```
np.random.seed(0)

n_data = 100
random_noise1 = np.random.normal(0, 1, (n_data,))
random_noise2 = np.random.normal(1, 1, (n_data,))
random_noise3 = np.random.normal(2, 1, (n_data,))

fig, ax = plt.subplots(figsize=(10, 7))
ax.tick_params(labelsize=20)

ax.plot(random_noise1,
        label='random noise1')       # 각각의 plot의 label을 해주어야 legend가 생성될 수 있다.
ax.plot(random_noise2,
        label='random noise2')
ax.plot(random_noise3,
        label='random noise3')
ax.legend(fontsize=20)
plt.show()
```



▼ **Legend Locations**

Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10




```
np.random.seed(0)

n_data = 100
random_noise1 = np.random.normal(0, 1, (n_data,))
random_noise2 = np.random.normal(1, 1, (n_data,))
random_noise3 = np.random.normal(2, 1, (n_data,))

fig, ax = plt.subplots(figsize=(10, 7))
ax.tick_params(labelsize=20)

ax.plot(random_noise1,
        label='random noise1')
ax.plot(random_noise2,
        label='random noise2')
ax.plot(random_noise3,
        label='random noise3')
ax.legend(fontsize=20,
          loc='lower right')
plt.show()
```

→ loc : legend의 위치 조정 parameter(upper, center, lower / right, center, left)