ML Day13 (Matplotlib)

<Named Colors>

▼ **Named Colors**

```
import matplotlib.pyplot as plt
import numpy as np
```

```
color_list = ['b', 'g', 'r', 'c', 'm', 'y']

fig, ax = plt.subplots(figsize=(5, 10))
ax.set_xlim([-1, 1])
ax.set_ylim([-1, len(color_list)])

for c_idx, c in enumerate(color_list):   # c_list의 index값과 c원소를 enumerate
    ax.text(0, c_idx,
            "color="+c,                  # "color=" + c(문자)
            fontsize=20,
```

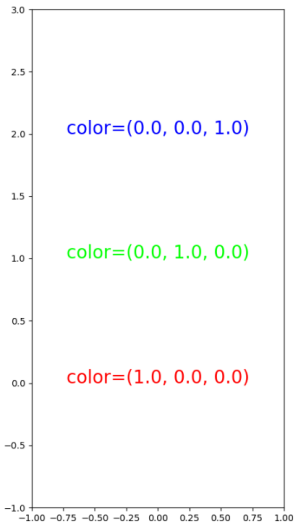
```
            ha='center',
            color=c)
plt.show()
```

▼ Named Colors(tab10 Colors)

```
color_list = ['tab:blue', 'tab:orange',
              'tab:green', 'tab:red',
              'tab:purple', 'tab:brown',
              'tab:pink', 'tab:gray',
              'tab:olive', 'tab:cyan']

fig, ax = plt.subplots(figsize=(5, 10))
ax.set_xlim([-1, 1])
ax.set_ylim([-1, len(color_list)])

for c_idx, c in enumerate(color_list):
    ax.text(0, c_idx,
            "color="+c,
            fontsize=20,
            ha='center',
            color=c)


plt.show()
```

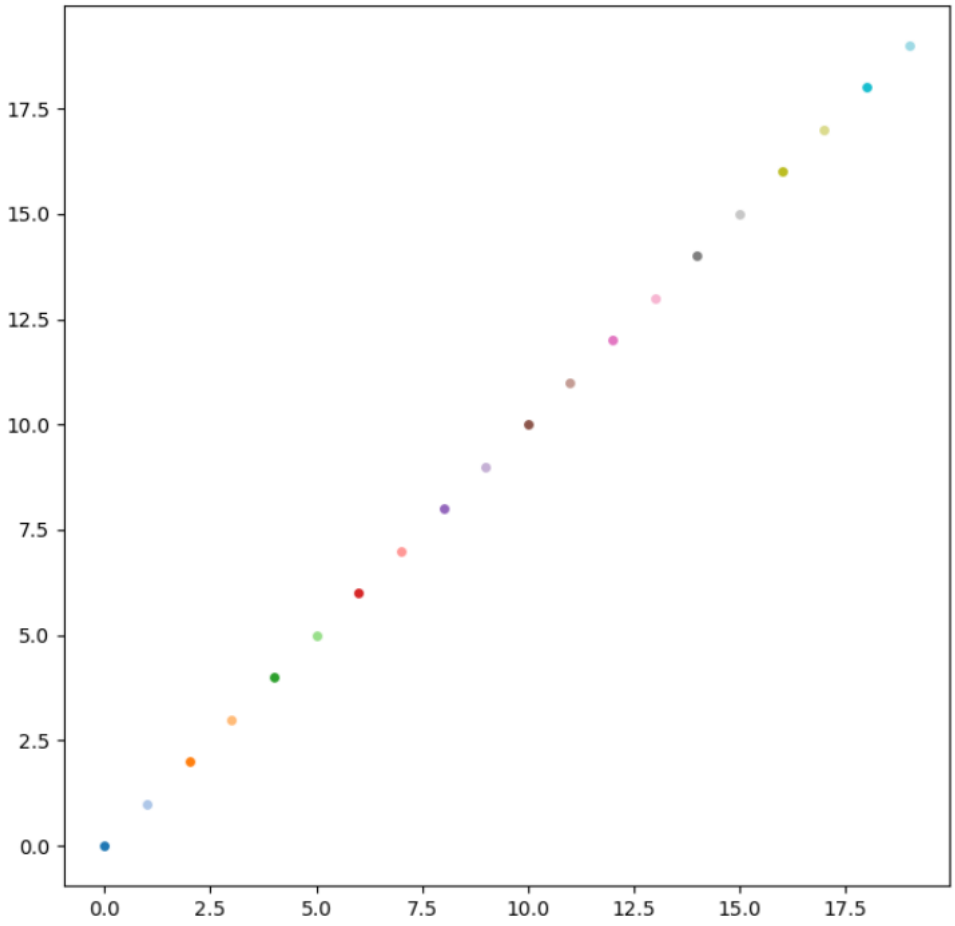
▼ **RGB Colors (0~255의 값들을 matplotlib에서는 0~1사이로 표현)**

```
color_list = [(1., 0., 0.),      # red
              (0., 1., 0.),      # green
              (0., 0., 1.)]      # blue

fig, ax = plt.subplots(figsize=(5, 10))
ax.set_xlim([-1, 1])
ax.set_ylim([-1, len(color_list)])

for c_idx, c in enumerate(color_list):   # for문으로 c_idx와 c를 enumerate
    ax.text(0, c_idx,
            f"color={c}",                # f문자열로 {}에 c가 들어간다.
            fontsize=20,
            ha='center',
            color=c)

plt.show()
```



▼ Discrete Colormaps(lut Argument)

```
import matplotlib.pyplot as plt
import matplotlib.cm as cm                  # color 모음을 사용할 수 있는 기능을 import

cmap = cm.get_cmap('tab20', lut=20)         # lut=20 -> 'tab20'에서 20개를 뽑아오겠다는 의미(look up table)
fig, ax = plt.subplots(figsize=(8, 8))      # cmap : indexing처럼 작동하는 함수
for i in range(20):                         # for문을 통해 20만큼
    ax.scatter(i, i, color=cmap(i), s=20)   # (i, i)좌표에 점 표현, scatter에서 s=20은 점의 size

plt.show()
```



▼ Continuous Colormaps (lut Argument) 1

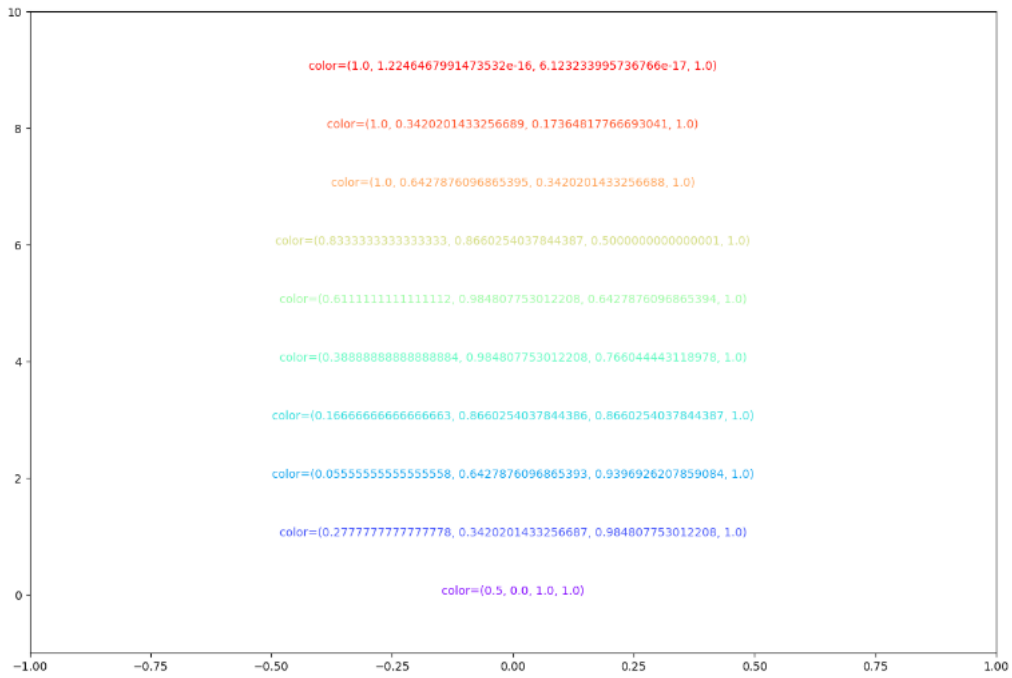
```
n_color = 10
cmap = cm.get_cmap('rainbow', lut=n_color)   # continuous한 rainbow의 색상들을 n_color(10개)만큼 분화해서 뽑아옴

for c_idx in range(n_color):
    print(cmap(c_idx))
```

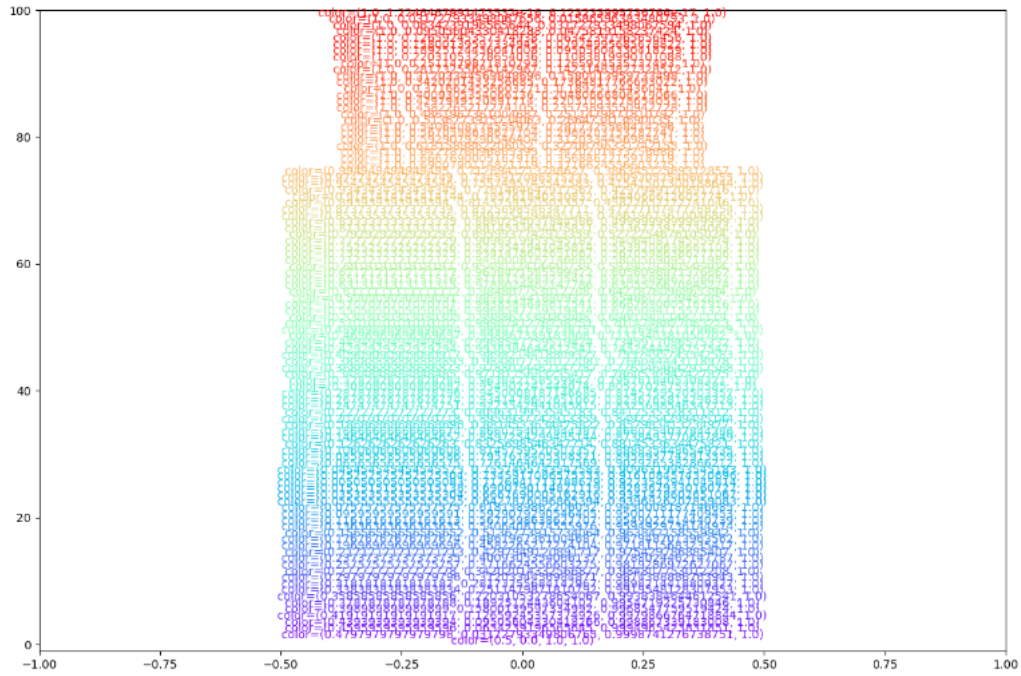
```
(0.5, 0.0, 1.0, 1.0)
(0.2777777777777778, 0.3420201433256687, 0.984807753012208, 1.0)
(0.05555555555555558, 0.6427876096865393, 0.9396926207859084, 1.0)
(0.16666666666666663, 0.8660254037844386, 0.8660254037844387, 1.0)
(0.38888888888888884, 0.984807753012208, 0.766044443118978, 1.0)
(0.6111111111111112, 0.984807753012208, 0.6427876096865394, 1.0)
(0.8333333333333333, 0.8660254037844387, 0.5000000000000001, 1.0)
(1.0, 0.6427876096865395, 0.3420201433256688, 1.0)
(1.0, 0.3420201433256689, 0.17364817766693041, 1.0)
(1.0, 1.2246467991473532e-16, 6.123233995736766e-17, 1.0)
```

→ cmap(c_idx) → (R, G, B, 투명도)

▼ Continuous Colormaps(lut Argument) 2

```
n_color = 10
cmap = cm.get_cmap('rainbow', lut=n_color)

fig, ax = plt.subplots(figsize=(15, 10))
ax.set_xlim([-1, 1])
ax.set_ylim([-1, n_color])

for c_idx in range(n_color):           # for문으로 n_color(10)만큼
    color = cmap(c_idx)                # color변수에 cmap(c_idx)를 대입

    ax.text(0, c_idx,                  # (0, c_idx)좌표에
            f"color={cmap(c_idx)}",    # 각 color에 해당하는 값(cmap(c_idx))을 문자열과
            fontsize=15,
            ha='center',
            color=color)               # 해당하는 color로 표현

plt.show()
```

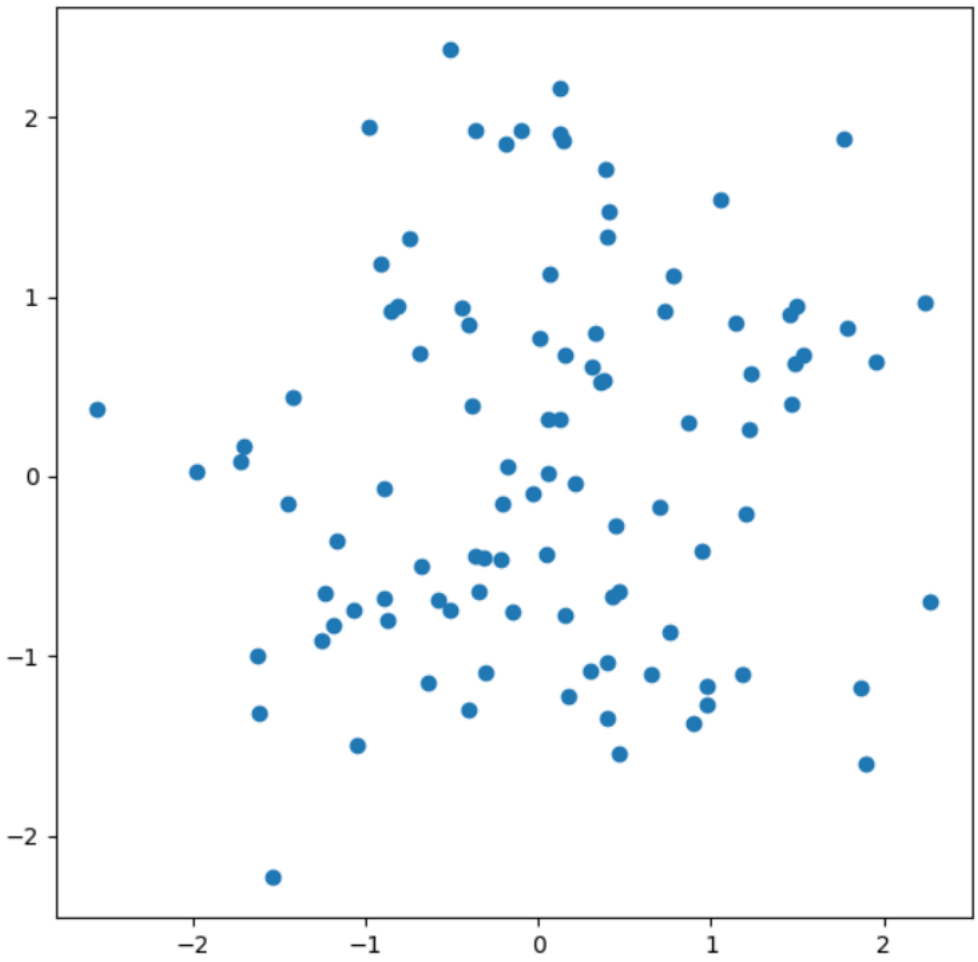




→ n_color =100

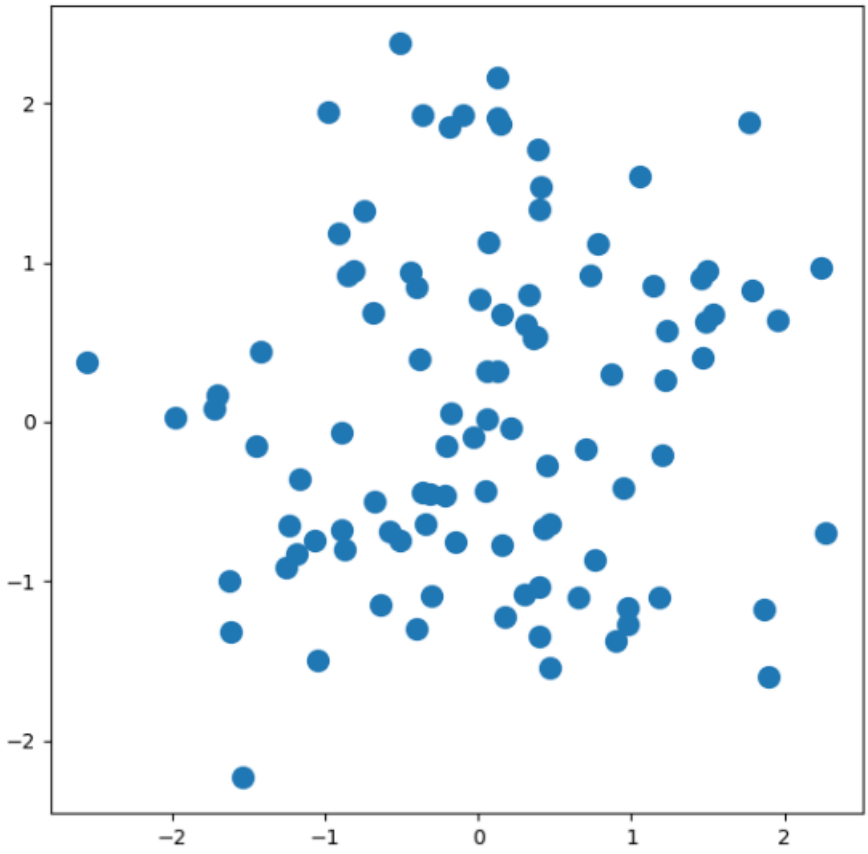
▼ ax.plot and ax.scatter (1)

```
import numpy as np

np.random.seed(0)            # 0에 id를 할당하여 np.random으로 계속 값이 바뀌는 것을 고정시킴

n_data = 100
x_data = np.random.normal(0, 1, (n_data,))   # np.random.normal : 정규 분포로 부터 임의의 값 추출(평균, 표준편차, 추출할 값 개수)
y_data = np.random.normal(0, 1, (n_data,))   # (n_data,) -> vector
fig, ax = plt.subplots(figsize=(7,7))

ax.scatter(x_data, y_data)
# ax.plot(x_data, y_data, 'o')                # plot으로 표현할 시 'o' - marker 모양 지정
                                              # marker종류('o', 'v', '^', '<', '>', 's', '*', 'h', 'H', 'D', 'P', 'X'))
plt.show()
```



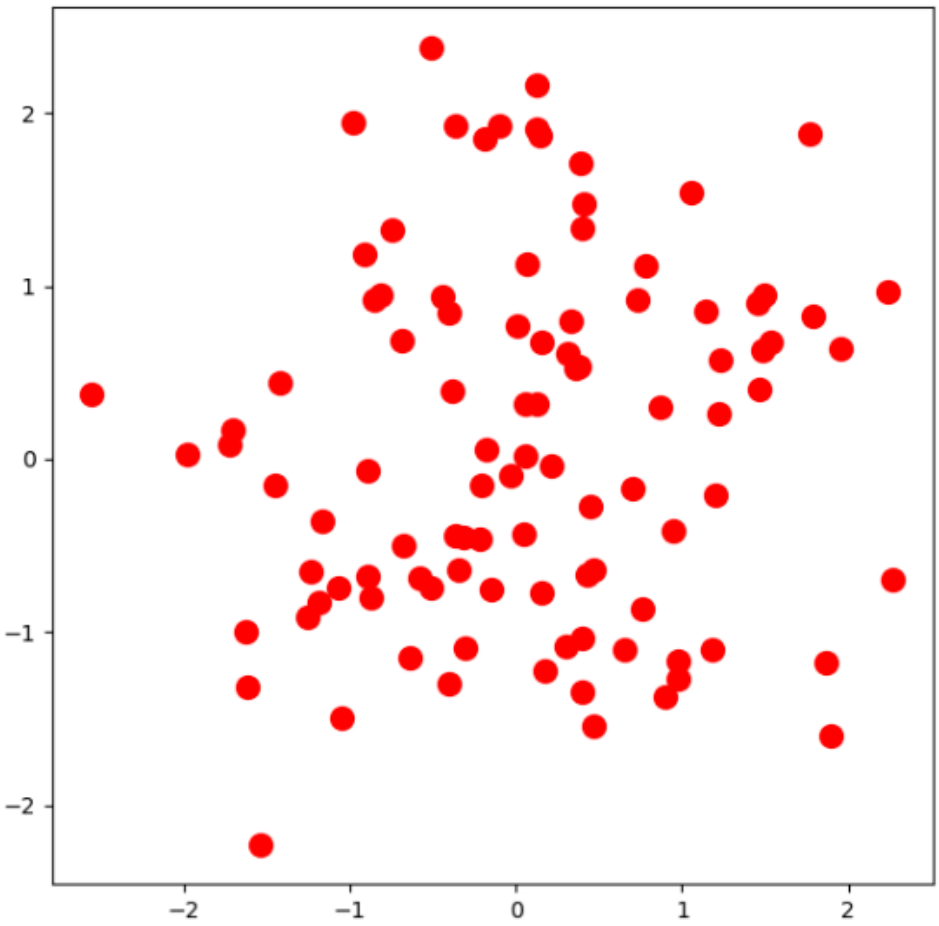
▼ **ax.plot and ax.scatter (2) - Maker size**

```
np.random.seed(0)

n_data = 100
x_data = np.random.normal(0, 1, (n_data))
y_data = np.random.normal(0, 1, (n_data))

fig, ax = plt.subplots(figsize=(7, 7))
ax.scatter(x_data, y_data,
           s=100)                              # scatter에서 점의 size
# ax.plot(x_data, y_data, 'o', markersize=10)  # plot에서 marker의 size
plt.show()
```



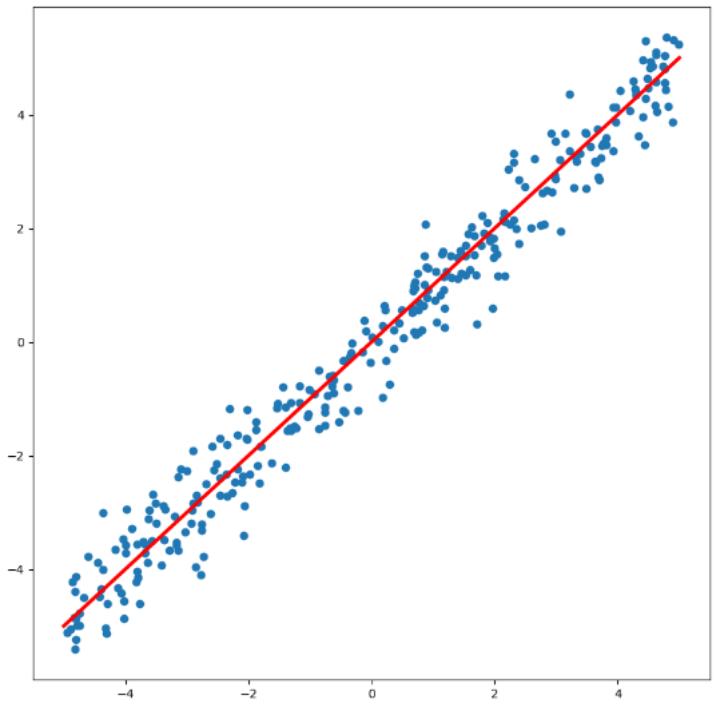
▼ **ax.plot and ax.scatter (3) - Color**

```
np.random.seed(0)

n_data = 100
x_data = np.random.normal(0, 1, (n_data,))
y_data = np.random.normal(0, 1, (n_data,))

fig, ax = plt.subplots(figsize=(7, 7))
ax.scatter(x_data, y_data,
           s=100,
           color='r')                       # marker의 color 지정

# ax.plot(x_data, y_data, 'o', color='red', markersize=10)
plt.show()
```



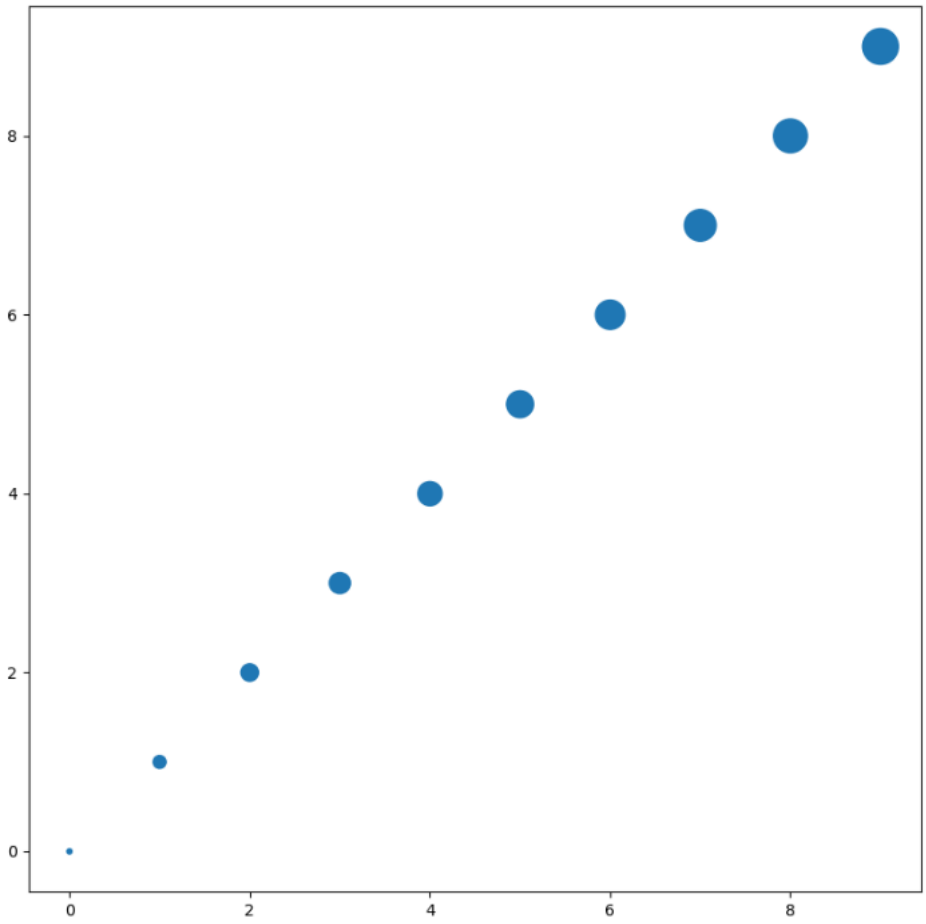
▼ **ax.plot and ax.scatter (4) - Linear regression**

```
np.random.seed(0)

x_min, x_max = -5, 5
n_data = 300

x_data = np.random.uniform(x_min, x_max, n_data)       # uniform(최소값, 최대값, 범위) -> 범위 안에서 발생할 확률이 동일하게 랜덤추출
y_data = x_data + 0.5*np.random.normal(0, 1, n_data)   # x_data에 0.5*만큼 정규분포를 따르는 noise를 주어 비교하기 위해

pred_x = np.linspace(x_min, x_max, 2)     # linspace(최소값, 최대값, 갯수) -> 최소값, 최대값 사이에 갯수를 동일한 간격으로 이어주는 선을 만듦
pred_y = pred_x

fig, ax = plt.subplots(figsize=(10, 10))
ax.scatter(x_data, y_data)
ax.plot(pred_x, pred_y,
        color='r',
        linewidth=3)
plt.show()
```



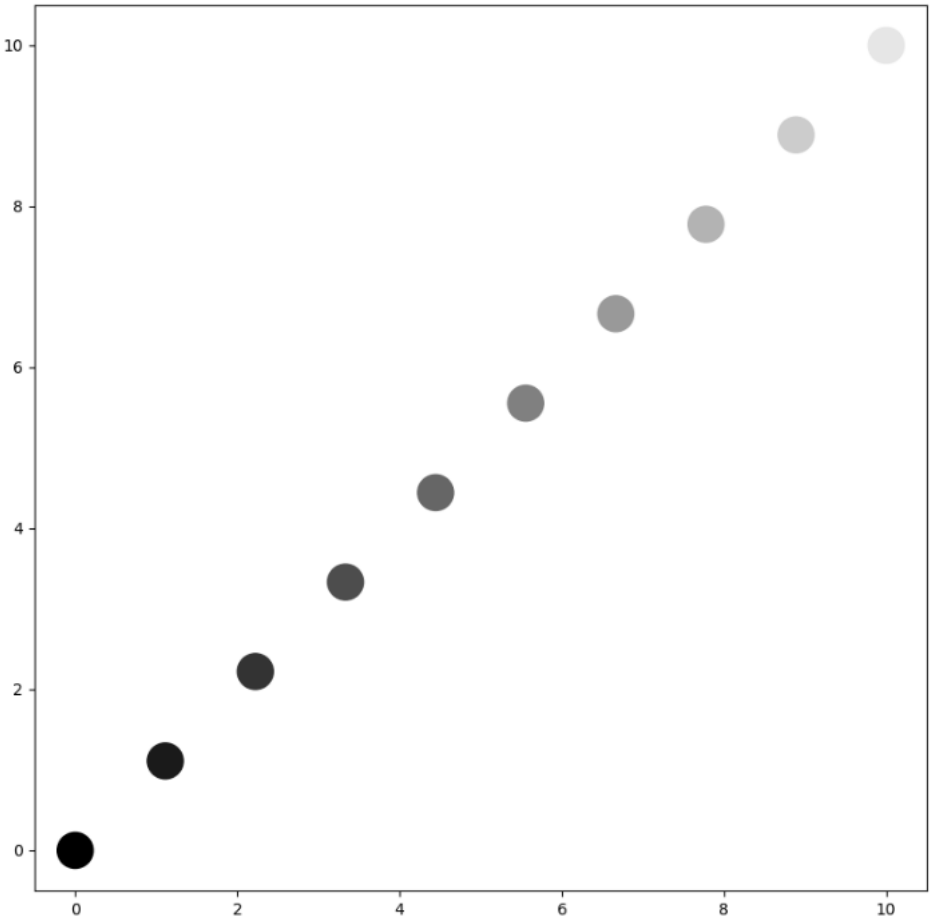
▼ **Size Array and Color Array (1)**

```
n_data = 10
x_data = np.linspace(0, 9, n_data)    # 0부터 9까지 n_data 만큼 동일한 간격으로 추출
y_data = np.linspace(0, 9, n_data)

s_arr = np.linspace(10, 500, n_data)

fig, ax = plt.subplots(figsize=(10, 10))
ax.scatter(x_data, y_data,            # 0 ~ 9까지 동일한 간격으로 x, y좌표에 scatter
           s=s_arr)                   # 크기는 s_arr에 따라
plt.show()
```



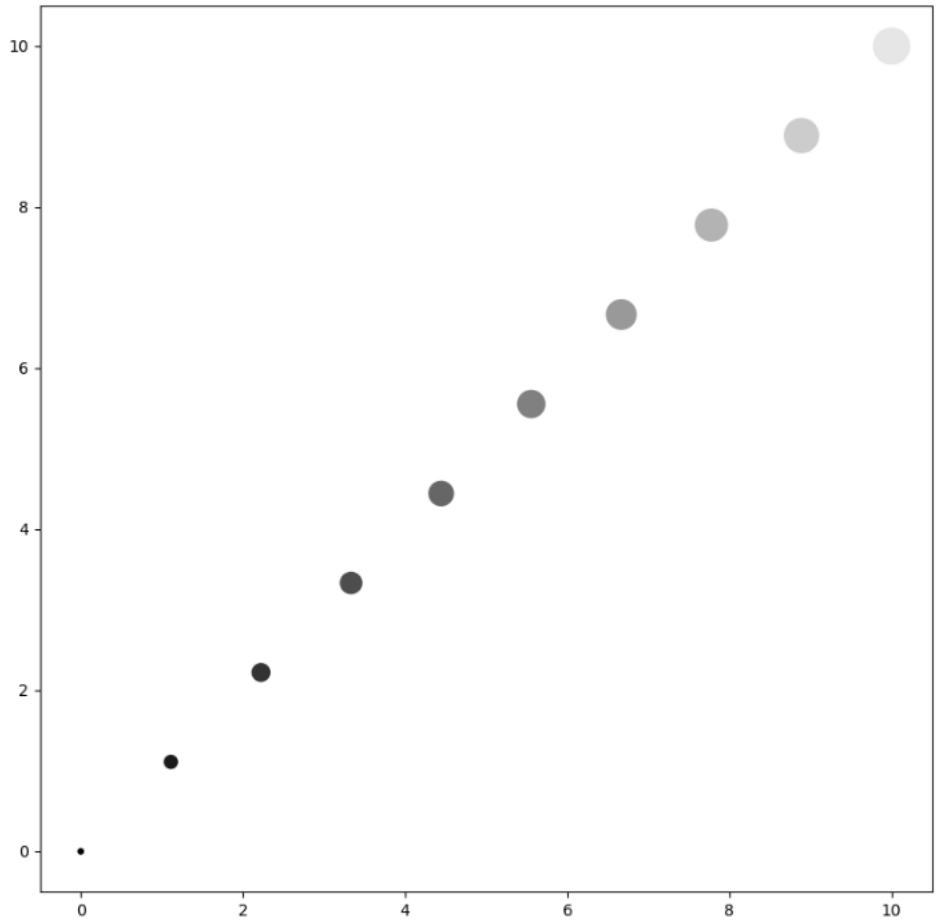
▼ **Size Array and Color Array (2)**

```
n_data = 10
x_data = np.linspace(0, 10, n_data)
y_data = np.linspace(0, 10, n_data)

c_arr = [(c/(n_data), c/(n_data), c/(n_data)) for c in range(n_data)]
        # RGB color -> c에 모두 같은 값(0부터)이 들어가기 때문에 무채색(검정)부터 for문 작동
fig, ax = plt.subplots(figsize=(10, 10))
ax.scatter(x_data, y_data,
           s=500,
           c=c_arr)      # c = color
plt.show()
```



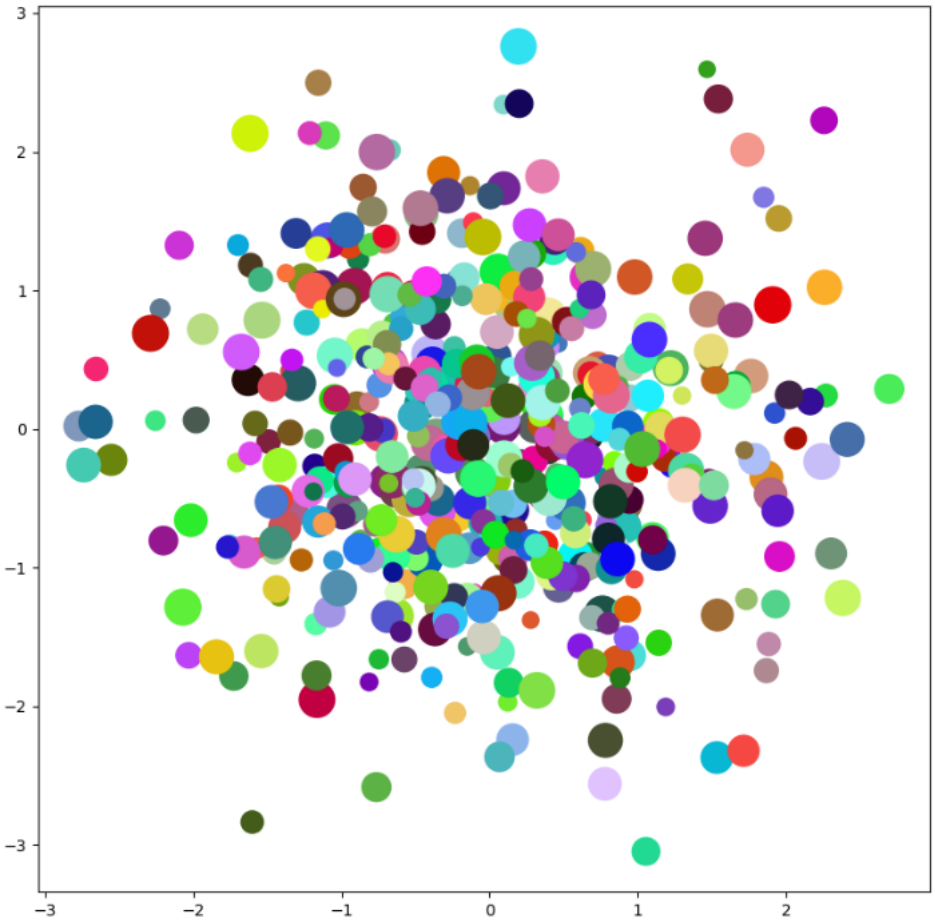
▼ **Size Array and Color Array (3)**

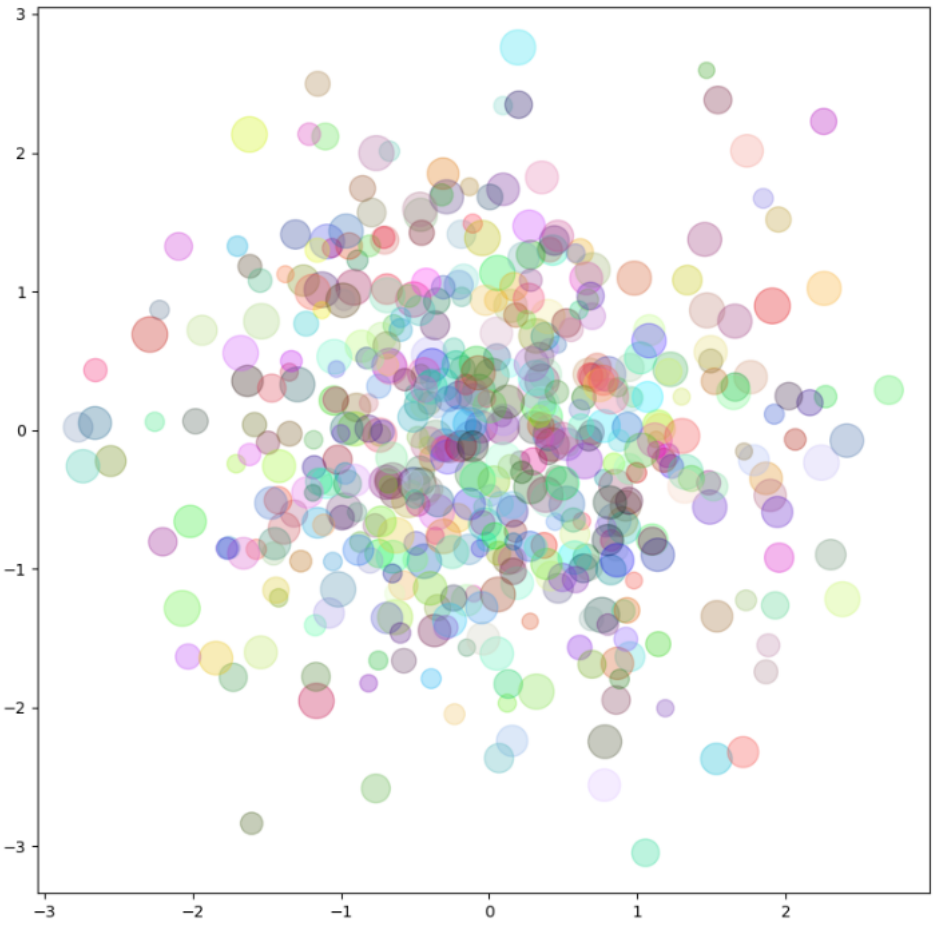
```
n_data = 10
x_data = np.linspace(0, 10, n_data)
y_data = np.linspace(0, 10, n_data)

s_arr = np.linspace(10, 500, n_data)              # 10 ~ 500사이를 n_data(10)갯수 만큼 동일한 간격으로 추출
c_arr = [(c/n_data, c/n_data, c/n_data) for c in range(n_data)]   # for문으로 n_data만큼 R,G,B color를 변수 c_arr에 대입

fig, ax = plt.subplots(figsize=(10, 10))
ax.scatter(x_data, y_data,
           s=s_arr,
           c=c_arr)
plt.show()
```



▼ **Size Array and Color Array (4)**

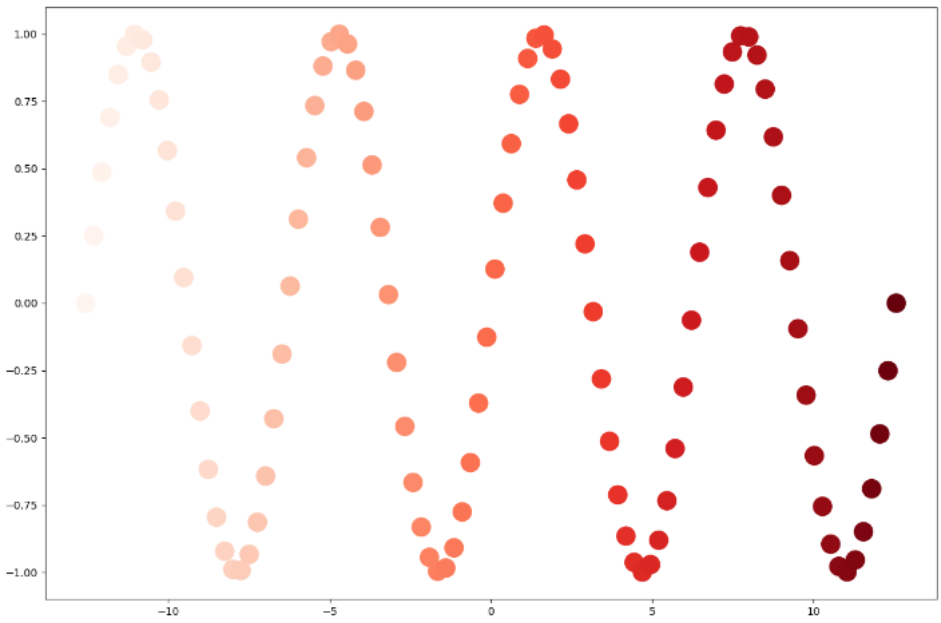
```
np.random.seed(0)

n_data = 500
x_data = np.random.normal(0, 1, size=(n_data, )) # 평균0, 표준편차1을 가지고 n_data vector만큼 추출
y_data = np.random.normal(0, 1, size=(n_data, ))

s_arr = np.random.uniform(100, 500, n_data)    # 100 ~ 500 사이의 수들에서 발생할 확률이 동일한 수들을 n_data만큼 추출
c_arr = [np.random.uniform(0, 1, 3)            # 0 ~ 1사이의 수들에서 발생할 확률이 동일한 수들을 3개씩 뽑아 3개의 원소를 가지는 RGB list값을 뽑는다
         for _ in range(n_data)]               # for문을 이용하여 n_data(500)만큼 돌려 RGB값을 500개를 뽑는다.

fig, ax = plt.subplots(figsize=(10, 10))
ax.scatter(x_data, y_data,
           s=s_arr,
           c=c_arr)

#ax.scatter(x_data, y_data,
 #          s=s_arr,
  #         c=c_arr,
   #        alpha=0.3)
plt.show()
```


3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2



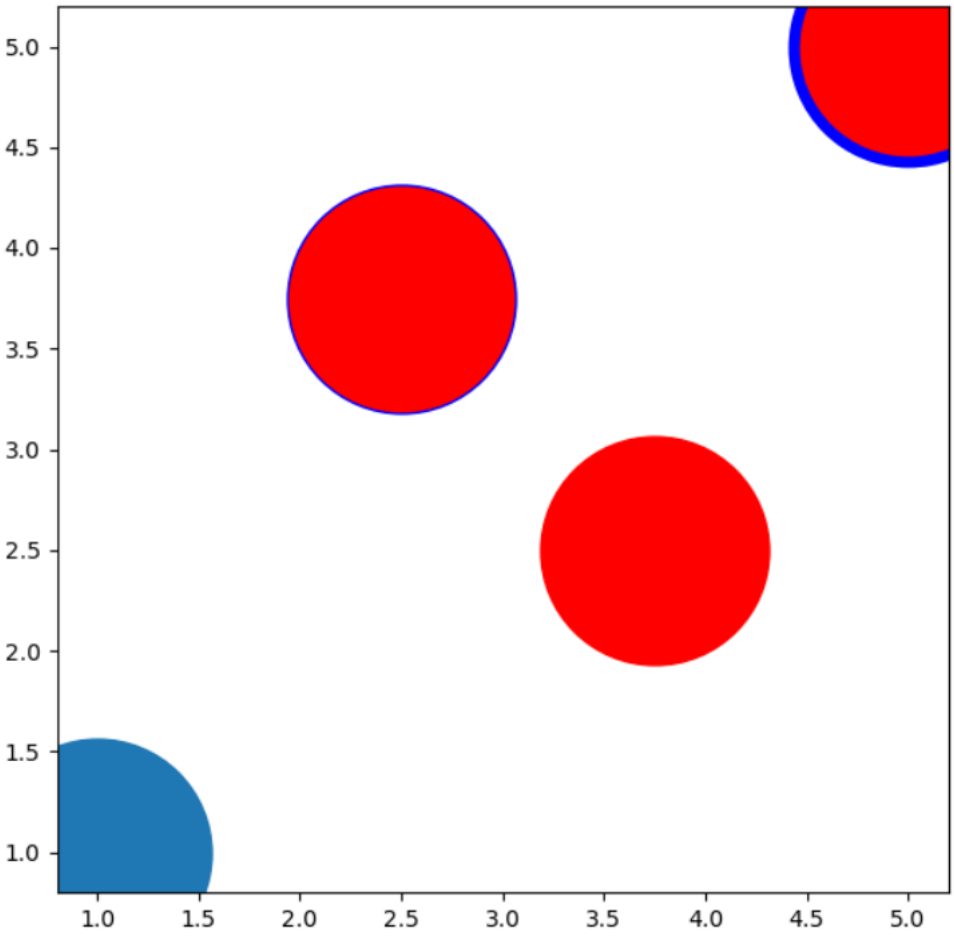
▼ **Color Array at c Argument**

```
PI = np.pi          # np.pi = 3.14
n_point = 100
t = np.linspace(-4*PI, 4*PI, n_point)
sin = np.sin(t)

cmap = cm.get_cmap('Reds', lut=n_point)     # continuous한 'Reds'의 색상들을 n_point(100개)만큼 뽑아 cmap 변수에 대입
c_arr = [cmap(c_idx) for c_idx in range(n_point)]  # n_point만큼 cmap의 c_idx 색상들을 차례대로 c_arr list에 대입

fig, ax = plt.subplots(figsize=(15, 10))
ax.scatter(t, sin,
           s=300,
           c=c_arr)

plt.show()
```



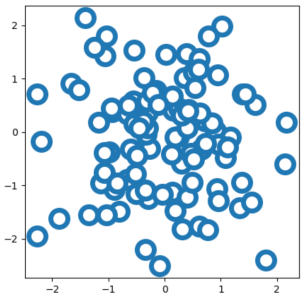
▼ **Advanced Markers (1)**

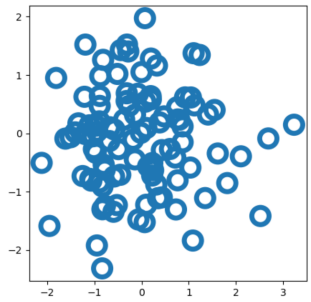
```
fig, ax = plt.subplots(figsize=(7, 7))

ax.scatter(1, 1,
           s=10000)
ax.scatter(2.5, 3.75,
           s=10000,
           facecolor='red',    # 원 내부 color
           edgecolor='b')      # 테두리 color
ax.scatter(3.75, 2.5,
           s=10000,
           facecolor='r')
ax.scatter(5, 5,
           s=10000,
           facecolor='red',
           edgecolor='b',
           linewidth=5)       # 테두리 두께

plt.show()
```



▼ **Advanced Markers (2)**

```
n_data = 100
x_data = np.random.normal(0, 1, (n_data, ))
y_data = np.random.normal(0, 1, (n_data, ))

fig, ax = plt.subplots(figsize=(5, 5))

ax.scatter(x_data, y_data,
           s=300,
           facecolor='white',          # facecolor='None' -> 투명
           edgecolor='tab:blue',
           linewidth=5)

plt.show()
```





→ facecolor= ‘None’