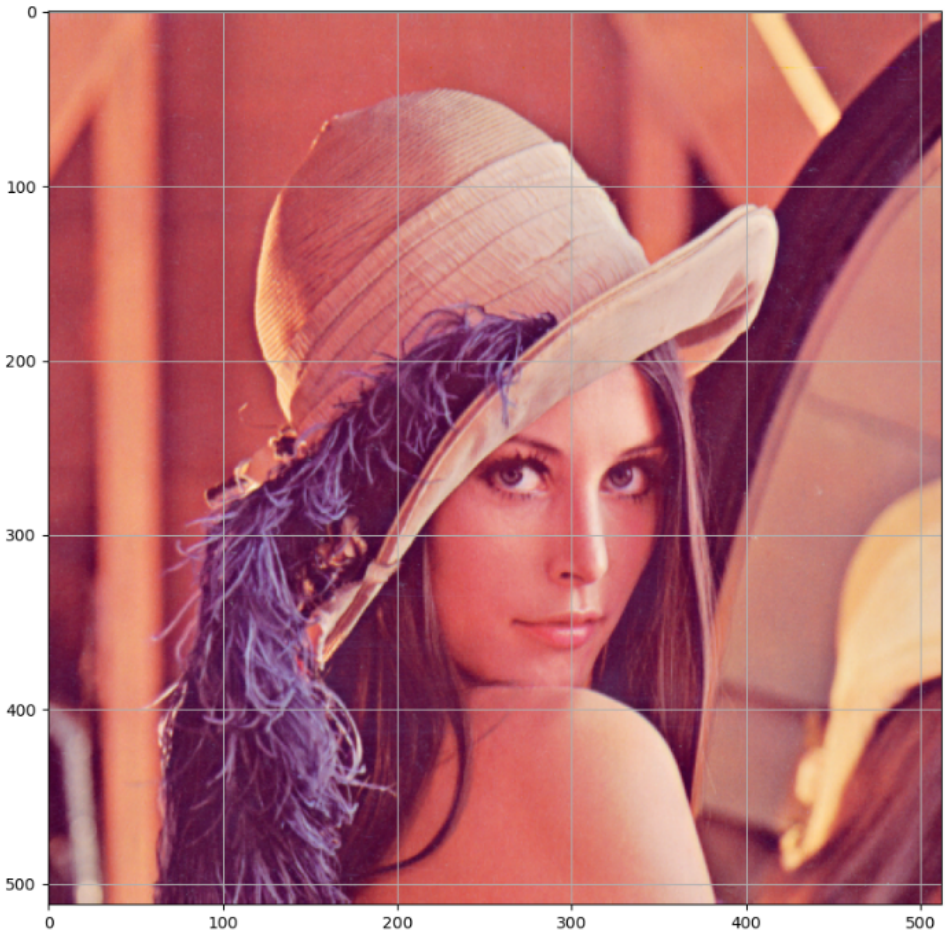
ML Day23 (Sobel Filtering)

▼ **ax.imread -** Image 파일 불러오기

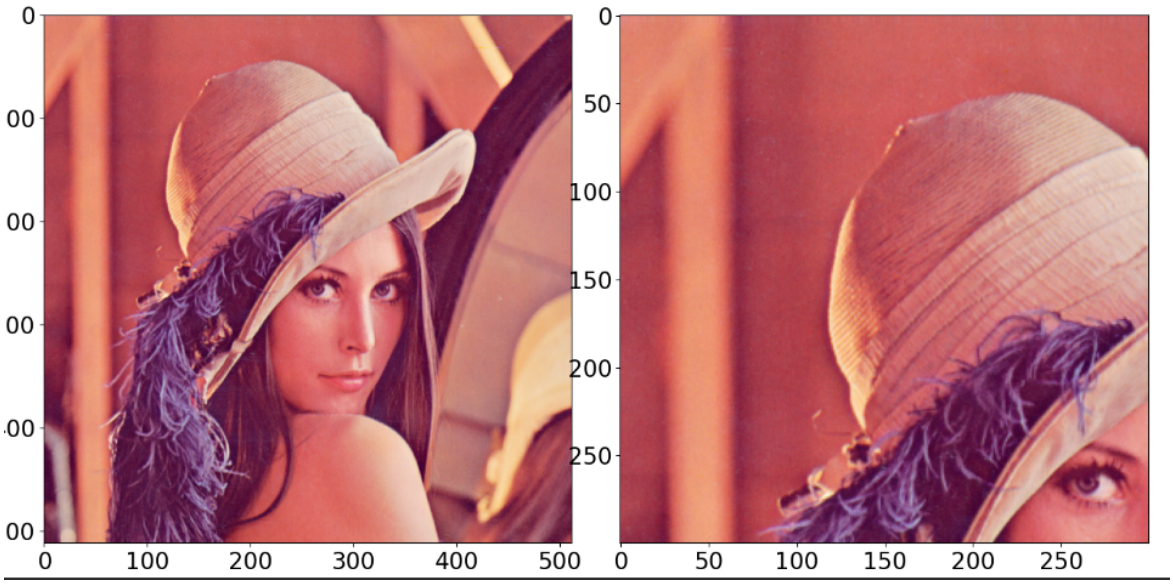
```
import matplotlib.pyplot as plt

img = plt.imread('./Lenna.png')
print(img.shape)

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

ax.grid()
plt.show()
```

```
(512, 512, 3)       # (height, width, rgbcolor수)
```



▼ **Image Coordinates -** 불러온 image파일을 원하는 크기만큼 slicing하여 나타내기

```
import matplotlib.pyplot as plt

img = plt.imread('./Lenna.png')
print(img.shape)

fig, axes = plt.subplots(1, 2, figsize=(14, 7))
fig.tight_layout()

axes[0].tick_params(labelsize=20)
axes[1].tick_params(labelsize=20)

axes[0].imshow(img)

img_cropped = img[:300, :300, :]       # 첫번째 이미지에서 [:300, :300, :]만큼 잘라냄   *[:300, :300, :] => height 300까지, width 300까지,

axes[1].imshow(img_cropped)
plt.show()
```



▼ **ax.imshow (1) -** 각 그림의 RGBcolor에 해당하는 R, G, B 값을 slicing하여 ax.imshow로 표현

```
import matplotlib.pyplot as plt

img = plt.imread('./Lenna.png')
print(img.shape)

r_img = img[:, :, 0]              # RGBcolor에서 0번 index값을 가지는 'r'값 slicing
g_img = img[:, :, 1]
b_img = img[:, :, 2]

print(r_img.shape)
print(g_img.shape)
print(b_img.shape)

fig, axes = plt.subplots(2, 2, figsize=(15, 15))

for ax in axes.flatten():
    ax.tick_params(labelleft=False,                         # labelleft, labelbottom의 값을 False로 주어 없앰
                   labelbottom=False)
    for spine_loc, spine in ax.spines.items():              # spine : 축을 커스터마이징 할 수 있는 객체
        spine.set_visible(False)

fig.tight_layout()

axes[0, 0].imshow(img)
axes[0, 1].imshow(r_img)
axes[1, 0].imshow(g_img)
axes[1, 1].imshow(b_img)
plt.show()
```

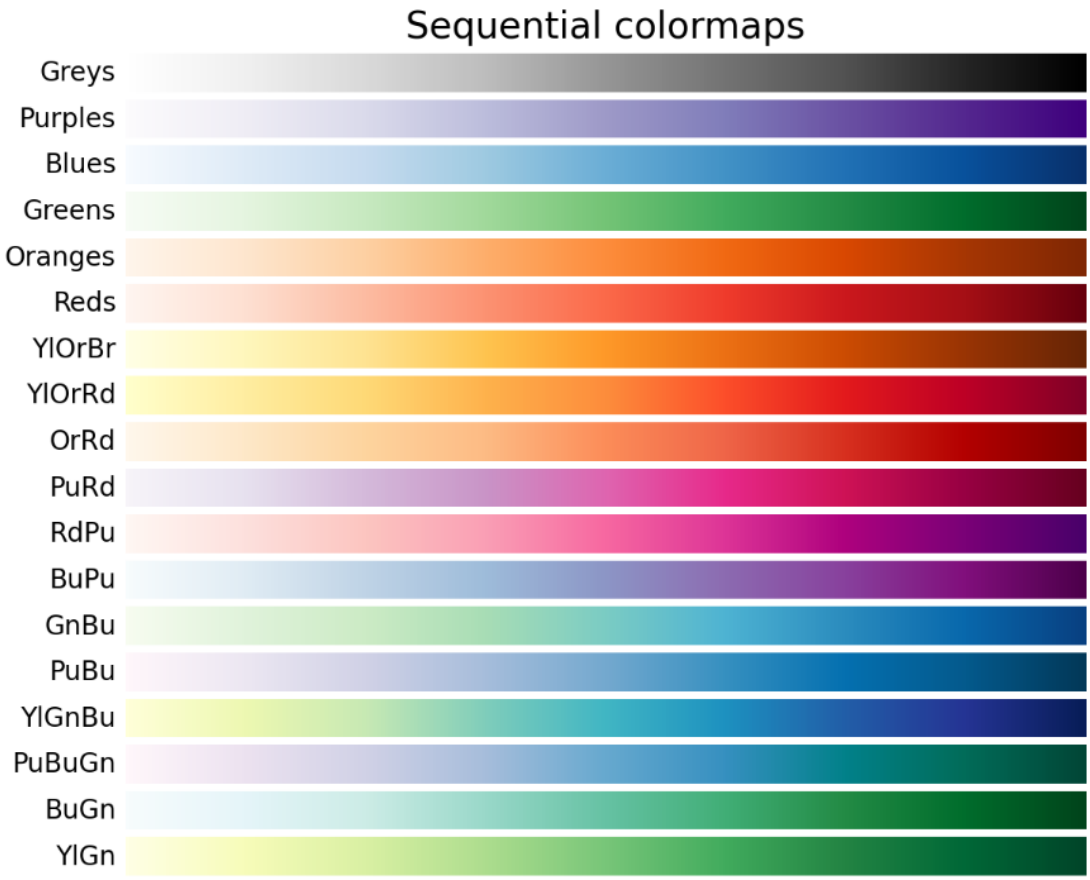
▼ **ax.imshow (2) -** cmap='gray'로 흑백 image 표현

```
import matplotlib.pyplot as plt

img = plt.imread('./Lenna.png')
print(img.shape)

r_img = img[:, :, 0]
g_img = img[:, :, 1]
b_img = img[:, :, 2]

print(r_img.shape)
print(g_img.shape)
print(b_img.shape)

fig, axes = plt.subplots(2, 2, figsize=(15, 15))
for ax in axes.flatten():
    ax.tick_params(labelleft=False,
                   labelbottom=False)
    for spine_loc, spine in ax.spines.items():
        spine.set_visible(False)
fig.tight_layout()

axes[0, 0].imshow(img)

axes[0, 1].imshow(r_img, cmap='gray')      # image 자체의 pixel 하나하나에 RGBcolor가 입혀져 있는 상태이기 때문에 흑백으로 image를 나타내도 명암구
axes[1, 0].imshow(g_img, cmap='gray')
```

```
axes[1, 1].imshow(b_img, cmap='gray')
plt.show()
```

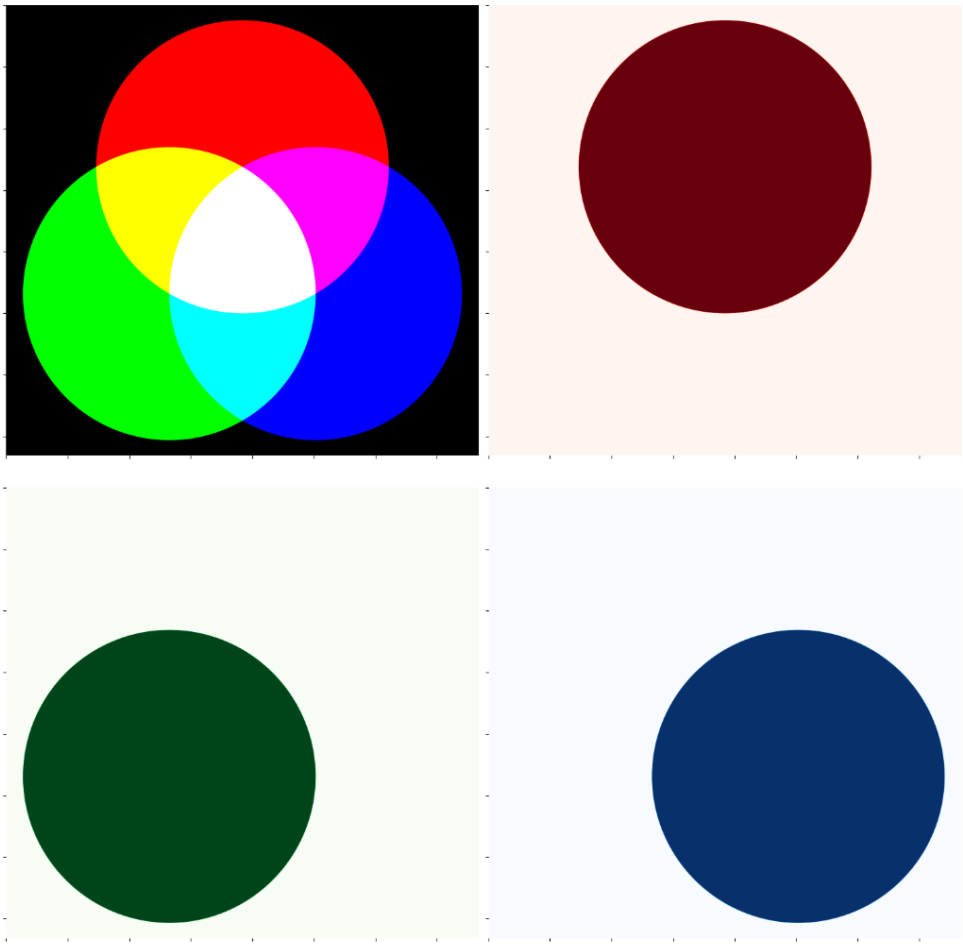
```
(512, 512, 3)
(512, 512)
(512, 512)
(512, 512)
```

▼ **ax.imshow (3) -** cmap='color_r'로 색상을 반전 표현




```
import matplotlib.pyplot as plt

img = plt.imread('./Lenna.png')
print(img.shape)

r_img = img[:, :, 0]
g_img = img[:, :, 1]
b_img = img[:, :, 2]

print(r_img.shape)
print(g_img.shape)
print(b_img.shape)

fig, axes = plt.subplots(2, 2, figsize=(15, 15))
for ax in axes.flatten():
    ax.tick_params(labelleft=False,
                   labelbottom=False)
    for spine_loc, spine in ax.spines.items():
        spine.set_visible(False)
fig.tight_layout()

axes[0, 0].imshow(img)

axes[0, 1].imshow(r_img, cmap='Reds_r')          # sequential한 'Reds'색상을 r(=reverse)로 반전 표현
axes[1, 0].imshow(g_img, cmap='Greens_r')
axes[1, 1].imshow(b_img, cmap='Blues_r')
plt.show()
```

```
import matplotlib.pyplot as plt

img = plt.imread('./RGB.png')
print(img.shape)

r_img = img[:, :, 0]
g_img = img[:, :, 1]
b_img = img[:, :, 2]

print(r_img.shape)
print(g_img.shape)
print(b_img.shape)

fig, axes = plt.subplots(2, 2, figsize=(15, 15))
for ax in axes.flatten():
    ax.tick_params(labelleft=False,
                   labelbottom=False)
    for spine_loc, spine in ax.spines.items():
        spine.set_visible(False)
fig.tight_layout()

axes[0, 0].imshow(img)

axes[0, 1].imshow(r_img, cmap='Reds')
axes[1, 0].imshow(g_img, cmap='Greens')
axes[1, 1].imshow(b_img, cmap='Blues')
plt.show()
```



Sobel Filtering

- Deep learning network 연산기법 중 CNN과 비슷함
- Edge detection
  - Edge pixels = 특정한 pixel 주변의 밝기 값이 급격하게 변하는 pixel
  - filter개념을 이용하여 변하는 window와 filter간의 내적을 구하고 내적 값이 가장 큰 경우 edge로 판단

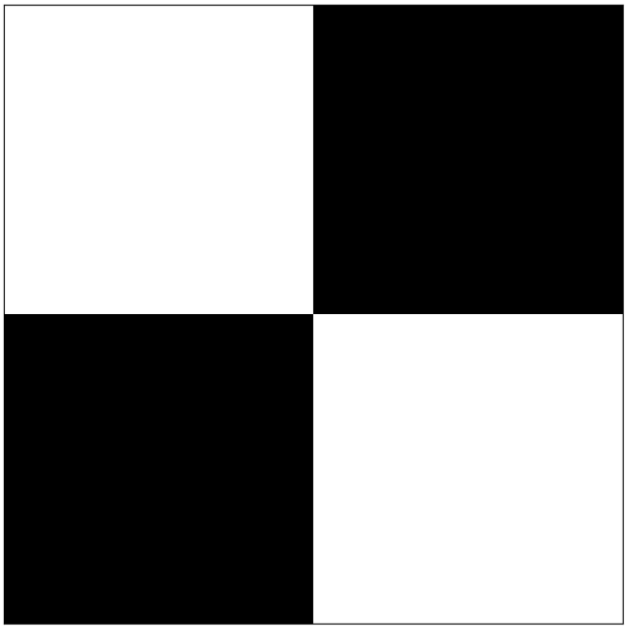
▼ ndarray review

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

tmp = np.ones(shape=(2, 3))   # np.ones: 입력된 shape의 array를 만들고 모두 1로 채워주는 함수 / (2, 3)의 shape을 가지고 모두 1로 채워져있는 행렬을
print(tmp, '\n')

tmp2 = 10 * tmp               # tmp에 들어있는 모든 원소에 10을 곱함 / ndarray에 scalar를 곱하면 원소 별 곱셈이 이루어짐
print(tmp2)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
[[10. 10. 10.]
 [10. 10. 10.]]
```

▼ Check Pattern Image (1)

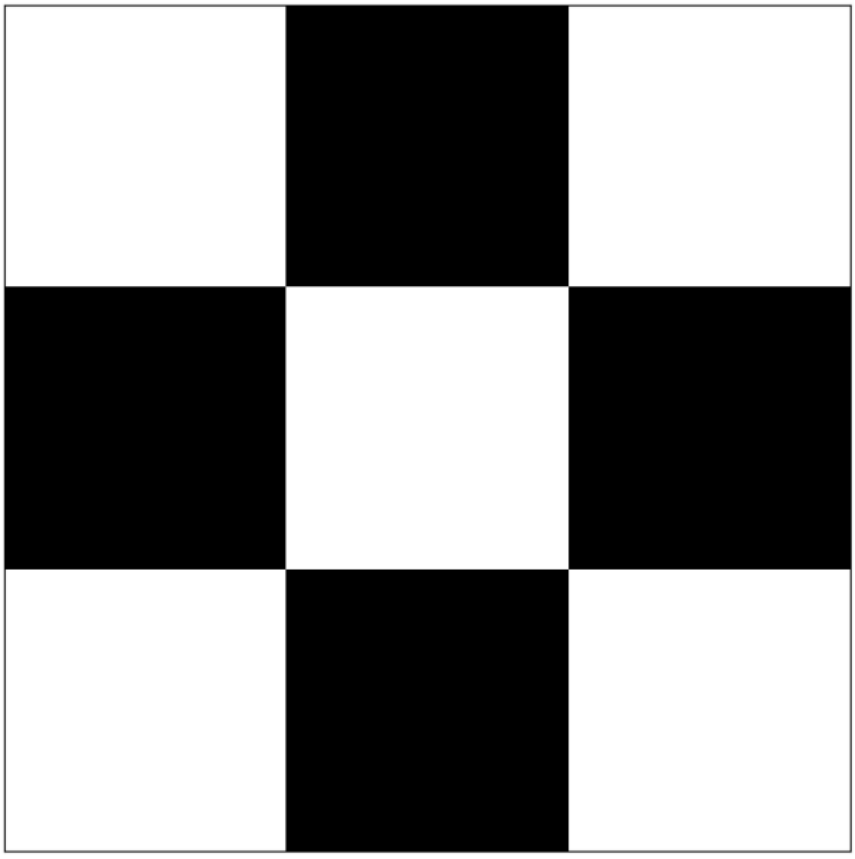
```
import numpy as np
import matplotlib.pyplot as plt

white_patch = 255*np.ones(shape=(10, 10))   # (10, 10)짜리 흰색 패치 만들기 => 255로 채우기
black_patch = 0*np.ones(shape=(10, 10))     # (10, 10)짜리 검은색 패치 만들기 => 0으로 채우기

img1 = np.hstack([white_patch, black_patch])   # [흰, 검]의 (10, 20) 이미지 만들기
img2 = np.hstack([black_patch, white_patch])   # [검, 흰]의 (10, 20) 이미지 만들기
img = np.vstack([img1, img2])

fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(img, cmap='gray')     # 이미지 띄우기, 흑백이미지를 만들 것이므로 colormap은 'gray'로

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```



▼ Check Pattern Image (2)

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

white_patch = 255*np.ones(shape=(10, 10))
black_patch = 0*np.ones(shape=(10, 10))


img1 = np.hstack([white_patch, black_patch, white_patch])
img2 = np.hstack([black_patch, white_patch, black_patch])
img3 = np.hstack([white_patch, black_patch, white_patch])
img = np.vstack([img1, img2, img3])

fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(img, cmap='gray')     # 이미지 띄우기, 흑백이미지를 만들 것이므로 colormap은 'gray'로 , cmap='gray'

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```

```
# 위에서 쓸데없는 img3 hstack을 줄임
import numpy as np
import matplotlib.pyplot as plt

white = 255*np.ones(shape=(10, 10))
black = 0*np.ones(shape=(10, 10))

img1 = np.hstack([white, black, white])
img2 = np.hstack([black, white, black])
img = np.vstack([img1, img2, img1])

fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(img, cmap='gray')

ax.tick_params(left=False, bottom=False,
               labelleft=False, labelbottom=False)
plt.show()
```



▼ np.repeat & np.tile

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

data = np.arange(5)
print(data)

# np.repeat => 원소별 반복
print("repeat:", np.repeat(data, repeats=3))

# np.tile => 전체 패턴 반복
print("tile:", np.tile(data, reps=3))
```

```
[0 1 2 3 4]
repeat: [0 0 0 1 1 1 2 2 2 3 3 3 4 4 4]
tile: [0 1 2 3 4 0 1 2 3 4 0 1 2 3 4]
```

▼ **np.repeat**

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

data = np.arange(6).reshape(2, 3)
print(data)

print("repeat(axis=0):\n",
      np.repeat(data, repeats=3, axis=0))       # axis=0, 행 방향으로

print("repeat(axis=1):\n",
      np.repeat(data, repeats=3, axis=1))       # axis=1, 열 방향으로

print("repeat(axis=0 and axis=1)\n",
      np.repeat(np.repeat(data, repeats=2, axis=0),  # axis=0 방향으로 2번 repeat 후 axis=1 방향으로 repeat 3
                repeats=3, axis=1))
```

```
[[0 1 2]
 [3 4 5]]
repeat(axis=0):
 [[0 1 2]
 [0 1 2]
 [0 1 2]
 [3 4 5]
 [3 4 5]
 [3 4 5]]
repeat(axis=1):
 [[0 0 0 1 1 1 2 2 2]
 [3 3 3 4 4 4 5 5 5]]
repeat(axis=0 and axis=1)
 [[0 0 0 1 1 1 2 2 2]
 [0 0 0 1 1 1 2 2 2]
 [0 0 0 1 1 1 2 2 2]
 [3 3 3 4 4 4 5 5 5]
 [3 3 3 4 4 4 5 5 5]
 [3 3 3 4 4 4 5 5 5]]
```

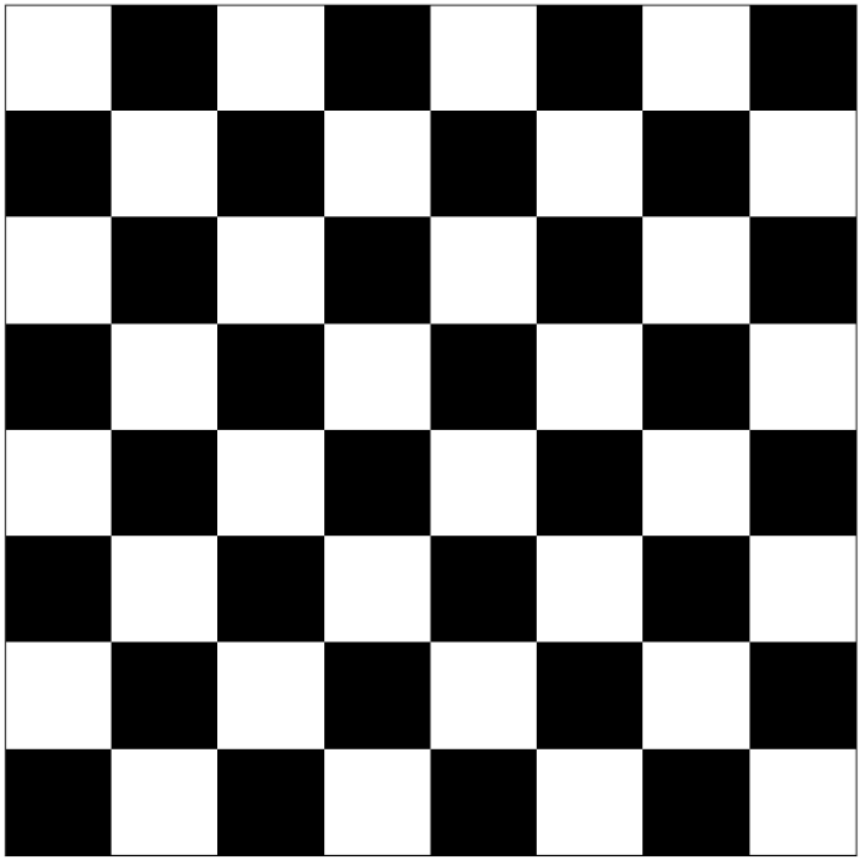
▼ **np.tile**

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

data = np.arange(6).reshape(2, 3)
print(data)

print('tile(axis=0):\n',
      np.tile(data, reps=[3, 1]))
print('tile(axis=1):\n',
      np.tile(data, reps=[1, 3]))
print('tile(axis=0 and axis=1):\n',
      np.tile(data, reps=[3, 3]))
```

```
[[0 1 2]
 [3 4 5]]
tile(axis=0):
 [[0 1 2]
 [3 4 5]
 [0 1 2]
 [3 4 5]
 [0 1 2]
 [3 4 5]]
tile(axis=1):
 [[0 1 2 0 1 2 0 1 2]
 [3 4 5 3 4 5 3 4 5]]
tile(axis=0 and axis=1):
 [[0 1 2 0 1 2 0 1 2]
 [3 4 5 3 4 5 3 4 5]
 [0 1 2 0 1 2 0 1 2]
 [3 4 5 3 4 5 3 4 5]
 [0 1 2 0 1 2 0 1 2]
 [3 4 5 3 4 5 3 4 5]]
```

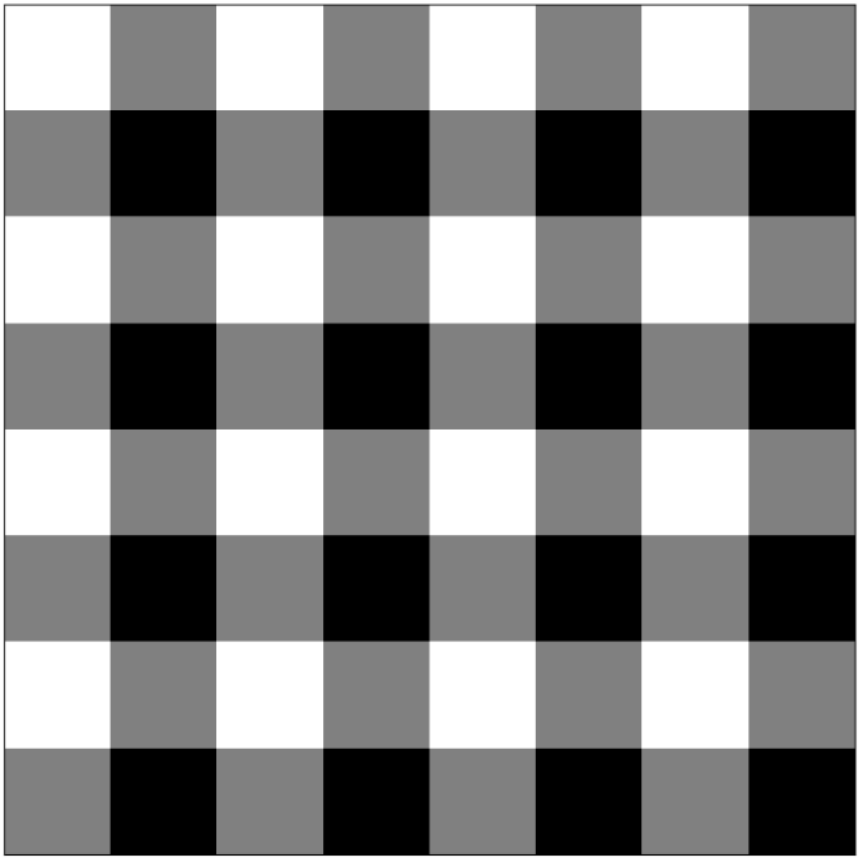
▼ **Check Pattern Image w/ np.tile (1)**

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

white = 255*np.ones(shape=(10, 10))
black = 0*np.ones(shape=(10, 10))


img1 = np.hstack([white, black])
img2 = np.hstack([black, white])
img = np.vstack([img1, img2])      # (2, 2) 기본 이미지 = [0][0]부터 흰, 검, 검, 흰 => 기본 이미지를 repeat 또는 tile

img_data = np.tile(img, reps=[4, 4])

fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(img_data, cmap='gray')

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```



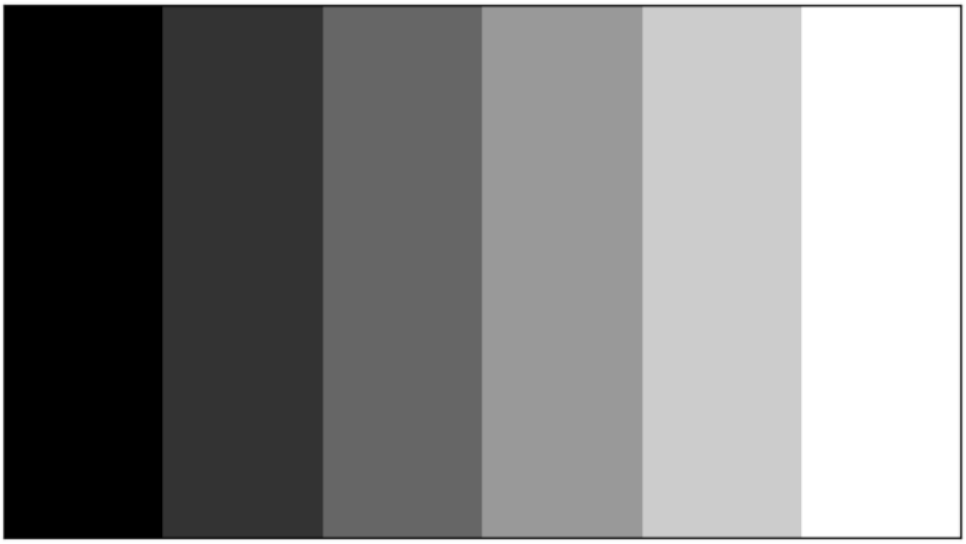
▼ **Check Pattern Image w/ np.tile (2)**

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

white = 255*np.ones(shape=(10, 10))
black = 0*np.ones(shape=(10, 10))
gray = 128*np.ones(shape=(10, 10))

img1 = np.hstack([white, gray])
img2 = np.hstack([gray, black])
img = np.vstack([img1, img2])

img_data = np.tile(img, reps=[4, 4])

fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(img_data, cmap='gray')

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```



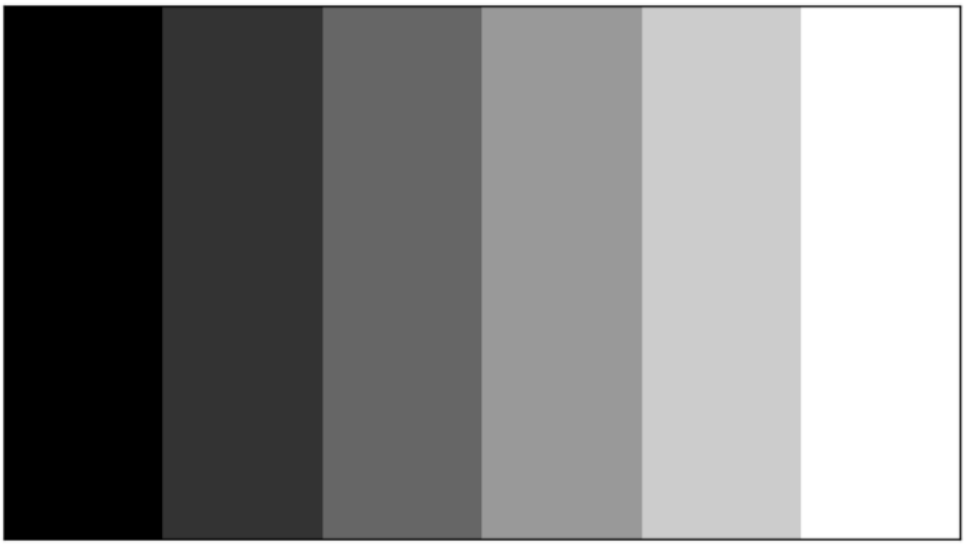
▼ **Grayscale Gradation Image w/ np.repeat (1)**

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

img = np.arange(0, 256, 50).reshape(1, -1)           # np.arange로 0부터 255까지(검정부터 흰색까지 뽑기 위해) 50step으로 img를 생성 후 (1, -1
img = img.repeat(100, axis=0).repeat(30, axis=1)     # axis=0 방향으로 100, axis=1 방향으로 30번 repeat

fig, ax = plt.subplots(figsize=(8, 4))
ax.imshow(img, cmap='gray')

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```



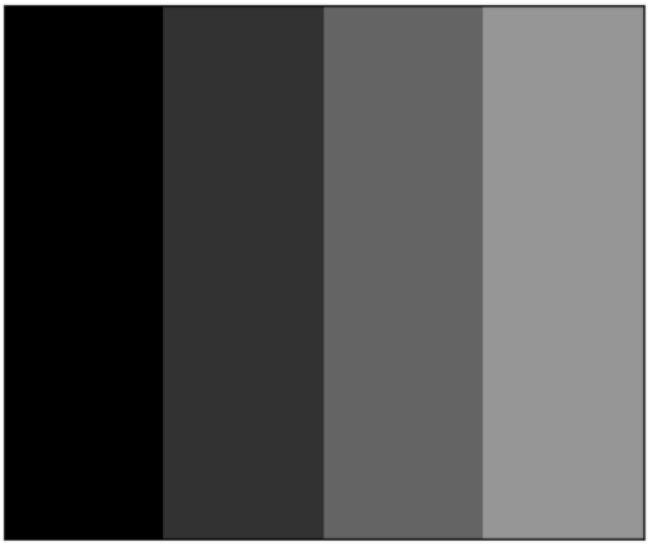
▼ **Grayscale Gradation Image w/ np.repeat (2)**

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

img = np.arange(0, 151, 50).reshape(1, -1)
img = img.repeat(100, axis=0).repeat(30, axis=1)

fig, ax = plt.subplots(figsize=(8, 4))
ax.imshow(img, cmap='gray')                # cmap='gray'는 위의 img값에 0 ~ 151까지의 색상 양 끝값을 자동으로 검정과 흰색으로 조정한다.

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```



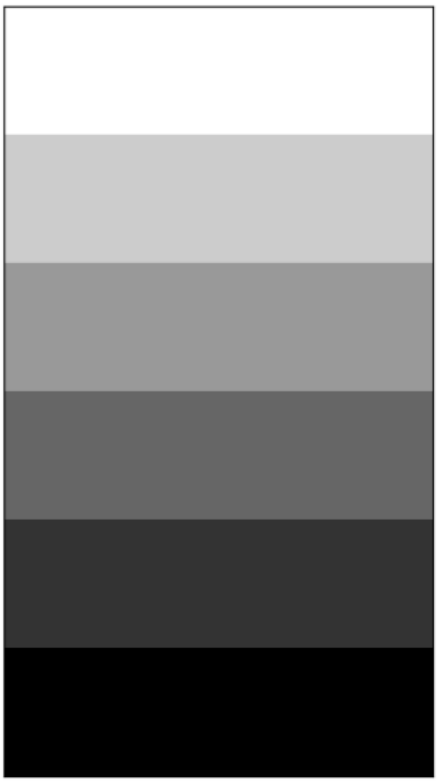
```
import numpy as np
import matplotlib.pyplot as plt

img = np.arange(0, 151, 50).reshape(1, -1)
print(img)
img = img.repeat(100, axis=0).repeat(30, axis=1)

fig, ax = plt.subplots(figsize=(8, 4))
ax.imshow(img, cmap='gray', vmax=255, vmin=0)                # 151인 max값을 vmax=255, vmin=0로 설정
ax.tick_params(left=False, labelleft=False,
```

```
               bottom=False, labelbottom=False)

plt.show()
```



▼ **Grayscale Gradation Image w/ np.repeat (3)**

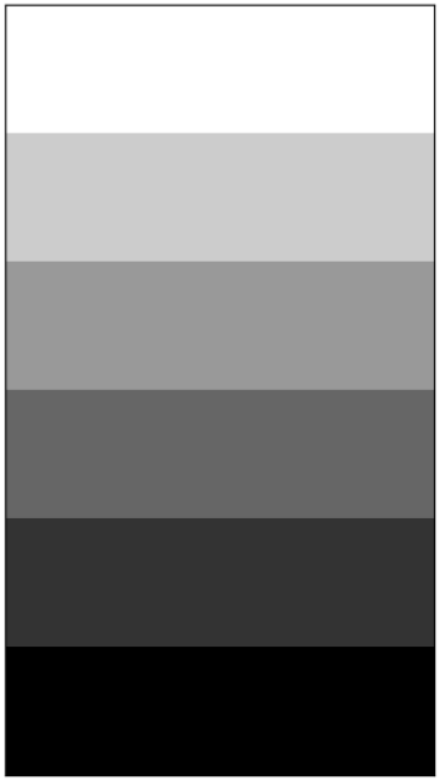
```
import numpy as np
import matplotlib.pyplot as plt

img = np.arange(256, 0, -50).reshape(-1, 1)       # 256부터 0까지 -50(거꾸로 50step씩)

img = img.repeat(30, axis=0).repeat(100, axis=1)

fig, ax = plt.subplots(figsize=(4, 8))
ax.imshow(img, cmap='gray')

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```



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

img = np.arange(0, 256, 50).reshape(-1, 1)
img = img.repeat(30, axis=0).repeat(100, axis=1)

fig, ax = plt.subplots(figsize=(4, 8))
ax.imshow(img, cmap='gray_r')                       # gray색을 reverse로

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```

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

img = np.arange(0, 256, 50).reshape(-1, 1)
img = np.tile(img, reps=[1, 3])                     # tile을 이용하여

fig, ax = plt.subplots(figsize=(4, 8))
ax.imshow(img, cmap='gray_r')

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```

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

img = np.arange(0, 256, 50)[::-1].reshape(-1, 1)    # list의 50step (-1)방향으로
img = img.repeat(30, axis=0).repeat(100, axis=1)

fig, ax = plt.subplots(figsize=(4, 8))
ax.imshow(img, cmap='gray')

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```



▼ **Grayscale Gradation Image w/ np.repeat (4)**

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

img1 = np.arange(0, 256, 1).reshape(1, -1)
img2 = np.arange(0, 256, 1)[::-1].reshape(1, -1)

img1_data = np.tile(img1, reps=[150, 1])                      # tile: 패턴 전체를 반복
img2_data = np.tile(img2, reps=[150, 1])

# img1_data = img1.repeat(1000, axis=0).repeat(100, axis=1)   # repeat: 원소 별 반복
# img2_data = img2.repeat(1000, axis=0).repeat(100, axis=1)
img = np.vstack([img1_data, img2_data])

fig, ax = plt.subplots(figsize=(4, 4))
ax.imshow(img, cmap='gray')

ax.tick_params(left=False, labelleft=False,
               bottom=False, labelbottom=False)

plt.show()
```