영상처리프로그래밍 : NumPy (3)

한림대학교 소프트웨어융합대학
박섭형

2022년 1학기

배울 내용

NumPy 다차원 배열의 indexing

- 기본 인덱싱
- 고급 인덱싱
- 필드 엑세스

NumPy 다차원 배열의 연산

NumPy 함수

기본 인덱싱과 슬라이싱

- $0 \leq n_i < d_i$
  - i: axis 번호
  - n_i: axis i의 인덱스
  - d_i: axis i의 원소의 갯수
- $n_i < 0$인 경우는 $n_i + d_i$
- 기본 슬라이싱 문법 : start: end : step

[1]:
```
import numpy as np
a = np.arange(48).reshape(8,6)
a
```

[1]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16, 17],
```

```
       [18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35],
       [36, 37, 38, 39, 40, 41],
       [42, 43, 44, 45, 46, 47]])
```

[2]:
```
a[(6,3)]
```

[2]:
```
39
```

[3]:
```
a[6,3]
```

[3]:
```
39
```

[4]:
```
a[6][3]
```

[4]:
```
39
```

[5]:
```
a[(-2,-3)]
```

[5]:
```
39
```

[6]:
```
a[-2,-3]
```

[6]:
```
39
```

[7]:
```
a[-2][3]
```

[7]:
```
39
```

[8]:
```
a
```

[8]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35],
       [36, 37, 38, 39, 40, 41],
       [42, 43, 44, 45, 46, 47]])
```

[9]:
```
a[:5,2]
```

[9]:
```
array([ 2,  8, 14, 20, 26])
```

[10]:
```
a[:5:2,2]
```

[10]:
```
array([ 2, 14, 26])
```

[11]:
```
a[1,:]
```

[11]:
```
array([ 6,  7,  8,  9, 10, 11])
```

[12]:
```
a[1,1:3]
```

[12]:
```
array([7, 8])
```

[13]:
```
a
```

[13]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35],
       [36, 37, 38, 39, 40, 41],
       [42, 43, 44, 45, 46, 47]])
```

[14]:
```
a[([0, 2], [1, 3])]
```

[14]:
```
array([ 1, 15])
```

[15]:
```
a[[0, 1], [1, 3]]
```

[15]:
```
array([1, 9])
```

[16]:
```
a[(0, 1), (1, 3)]
```

[16]:
```
array([1, 9])
```

- 배열 a에서 0, 5, 42, 47을 원소로 갖는 배열 만들기

[17]:
```
a[[0, 0, 7, 7], [0, 5, 0, 5]]
```

[17]:
```
array([ 0,  5, 42, 47])
```

[18]:
```
a[[[0, 0], [7, 7]], [[0, 5], [0, 5]]]
```

[18]:
```
array([[ 0,  5],
       [42, 47]])
```

[19]:
```
a
```

[19]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35],
       [36, 37, 38, 39, 40, 41],
       [42, 43, 44, 45, 46, 47]])
```

[20]:
```
a[[[0, 1, 3], [5, 7, 6]], [[0, 2, 3], [2, 5, 0]]]
```

[20]:
```
array([[ 0,  8, 21],
       [32, 47, 36]])
```

[21]:
```
a
```

[21]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35],
       [36, 37, 38, 39, 40, 41],
       [42, 43, 44, 45, 46, 47]])
```

[22]:
```
a[[0, 3, 1], ::2]
```

[22]:
```
array([[ 0,  2,  4],
       [18, 20, 22],
       [ 6,  8, 10]])
```

NumPy 다차원 배열의 부울 배열 인덱싱

[23]:
```
a = np.arange(0, 10, 2)
a
```

[23]:
```
array([0, 2, 4, 6, 8])
```

[24]:
```
idx = np.array([True, True, False, True, False])
a[idx]
```

[24]:
```
array([0, 2, 6])
```

[25]:
```
1 < a
```

[25]:
```
array([False,  True,  True,  True,  True])
```

[26]:
```
a < 3
```

[26]:
```
array([ True,  True, False, False, False])
```

[27]:
```
(1 < a) & (a < 3)
```

[27]:
```
array([False,  True, False, False, False])
```

[28]:
```
index = (1 < a) & (a < 3)
index
```

[28]:
```
array([False,  True, False, False, False])
```

[29]:
```
a[index]
```

[29]:
```
array([2])
```

[30]:
```
import matplotlib.pyplot as plt
from scipy import misc
face = misc.face()
```

[31]:
```
plt.imshow(face)
plt.show()
```

[32]:
```
face.shape
```

[32]:
```
(768, 1024, 3)
```

타원을 이용해서 raccoon의 얼굴 부분만 남기고 검은색으로 변경하기

- 타원의 방정식

$$\frac{(x-x_c)^2}{a^2} + \frac{(y-y_c)^2}{b^2} = 1$$

- 타원의 외부

$$\frac{(x-x_c)^2}{a^2} + \frac{(y-y_c)^2}{b^2} > 1$$

- 영상에서 왼쪽 위 모서리의 좌표가 $(0,0)$이고, 세로축의 아래 방향을 x축의 양의 방향, 가로축의 오른쪽 방향을 y축의 양의 방향으로 정의한다.

[33]:
```
np.ogrid[0:5, 0:3]
```

[33]:
```
[array([[0],
        [1],
        [2],
        [3],
```

```
        [4]]),
 array([[0, 1, 2]])]
```

[34]:
```
X, Y = np.ogrid[0:face.shape[0], 0:face.shape[1]]
X.shape, Y.shape
```

[34]:
```
((768, 1), (1, 1024))
```

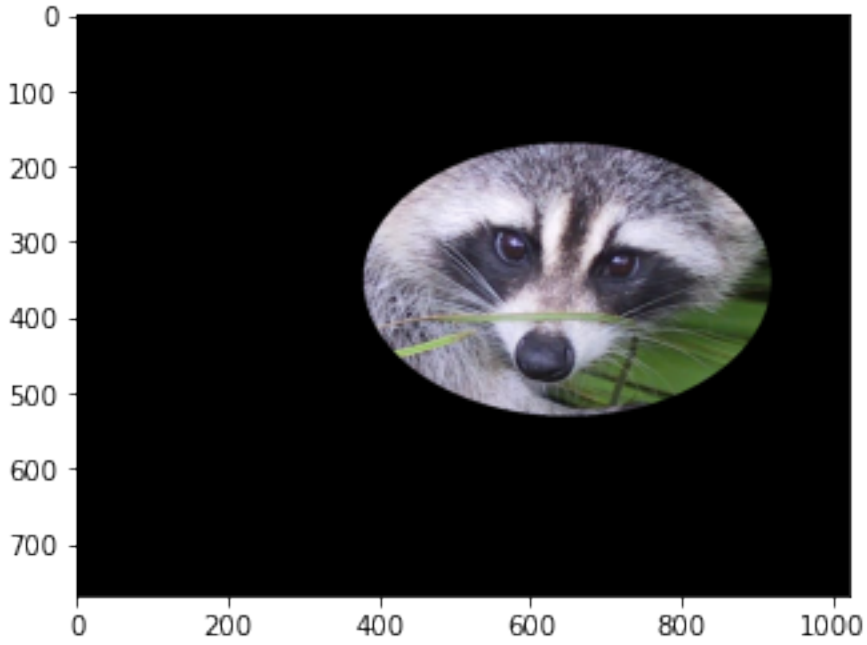
[35]:
```
face = misc.face()
center = np.array([350,650])
a = 180
b = 270

idx = (X-center[0])**2/a**2 + (Y-center[1])**2 / b**2 > 0.98
idx &= (X-center[0])**2/a**2 + (Y-center[1])**2 / b**2 < 1.02

face_with_ellipse = face.copy()
face_with_ellipse[idx] = (255, 255, 255)
plt.imshow(face_with_ellipse)
plt.show()
```

[36]:
```
center = np.array([350,650])
a = 180
b = 270

idx = (X-center[0])**2/a**2 + (Y-center[1])**2 / b**2 > 1
cropped = face.copy()
cropped[idx] = (0, 0, 0)
plt.imshow(cropped)
plt.show()
```



마름모를 이용해서 raccoon의 얼굴 부분만 남기고 검은색으로 변경하기

- 네 꼭지점의 좌표 : $A(350, 380), B(170, 650), C(350, 920), D(530, 650)$
- 두 점 (x_1, y_1)과 (x_2, y_2)를 잇는 직선의 방정식

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

또는

$$y - y_2 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_2)$$

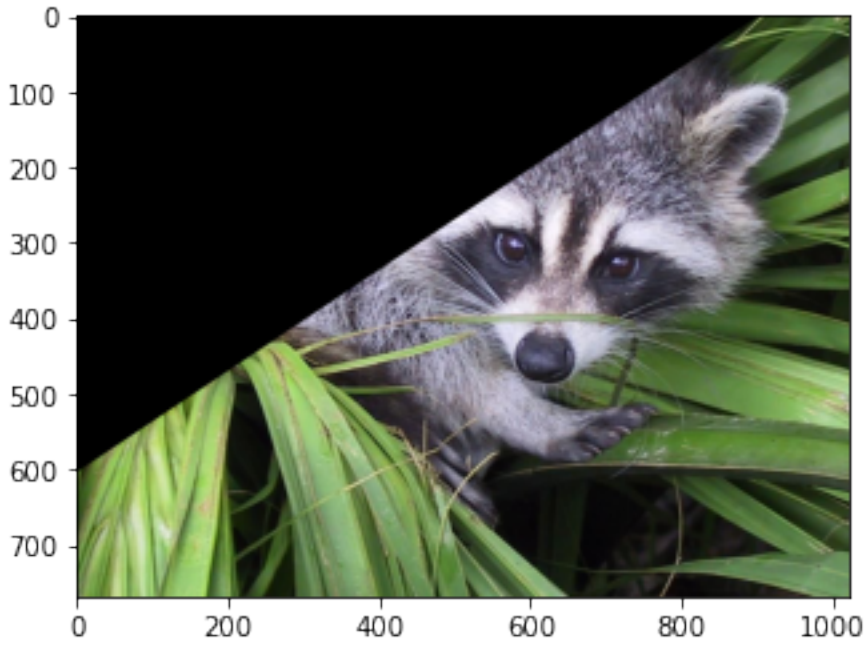
- 두 점 (x_1, y_1)과 (x_2, y_2)를 잇는 직선의 윗부분

$$y - y_1 > \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

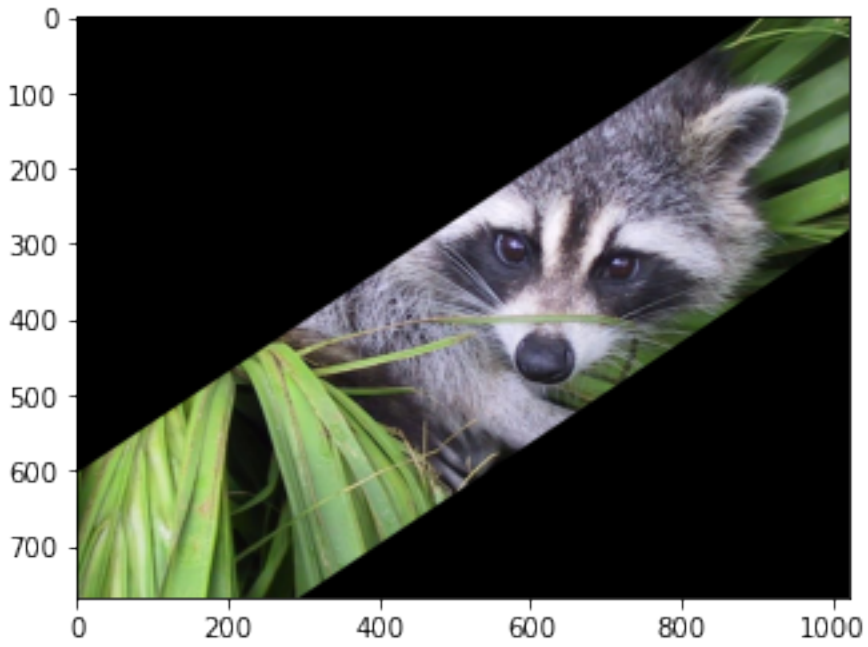
또는

$$y - y_2 > \frac{y_2 - y_1}{x_2 - x_1}(x - x_2)$$

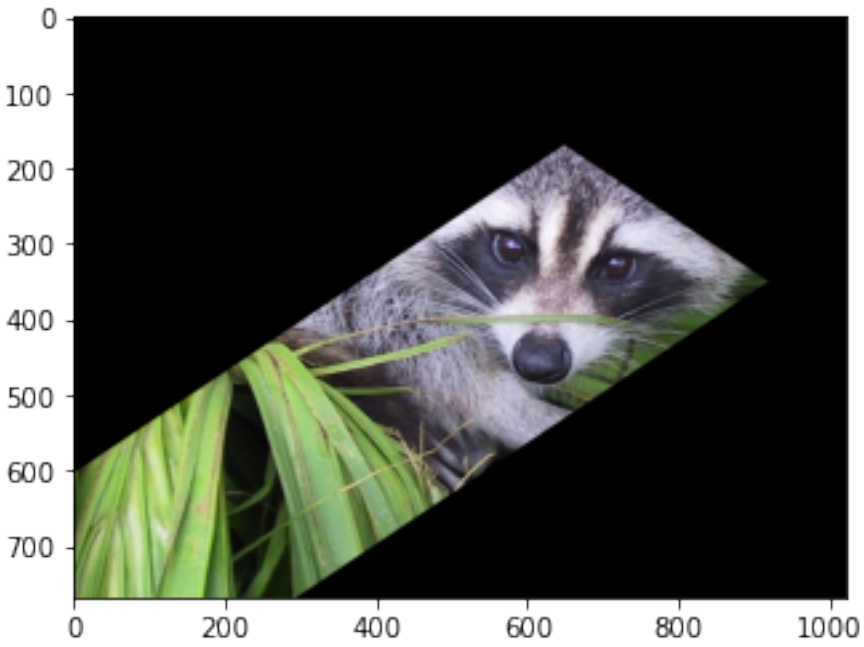
[37]:
```
A = np.array([350, 380])
B = np.array([170, 650])
C = np.array([350, 920])
D = np.array([530, 650])
# 직선 AB 윗 부분
slopeAB = (B-A)[1] / (B-A)[0]
idx1 = Y - A[1] < slopeAB * (X - A[0])
img1 = face.copy()
img1[idx1] = (0,0,0)
plt.imshow(img1)
plt.show()
```



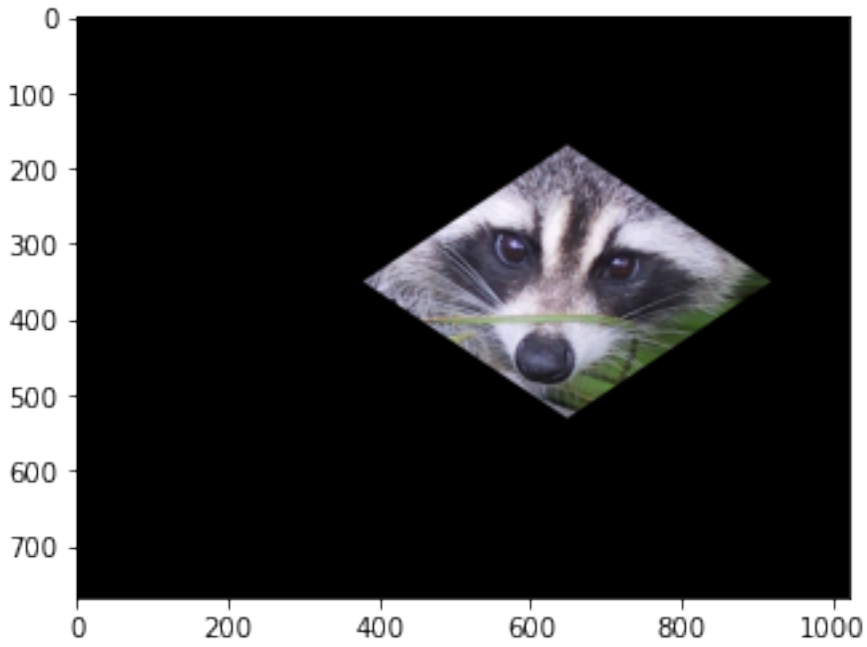
[38]:
```
# 직선 CD 윗부분
idx2 = Y - C[1] > slopeAB * (X - C[0])
img1[idx2] = (0,0,0)
plt.imshow(img1)
plt.show()
```



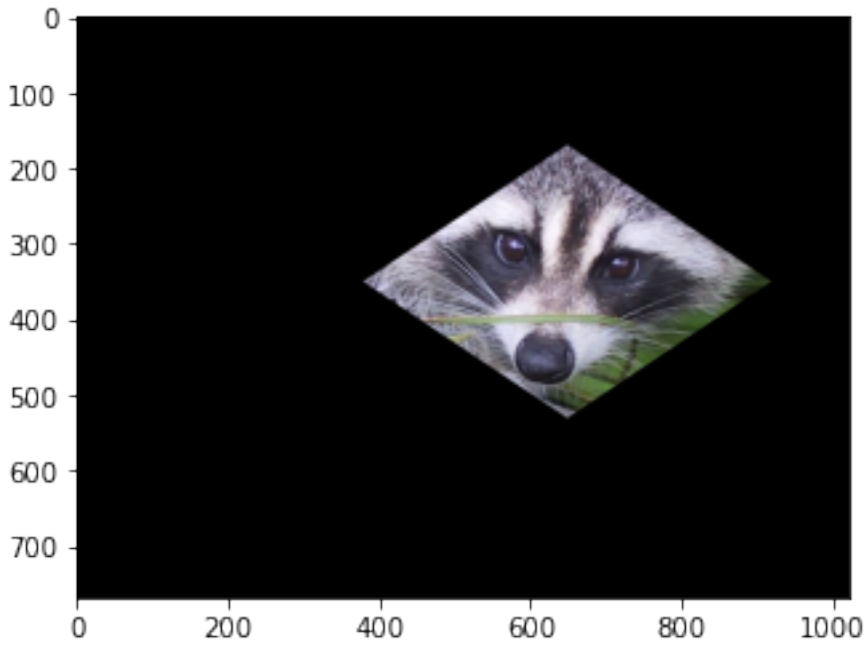
[39]:
```
# 직선 BC 윗부분
slopeBC = (C-B)[1] / (C-B)[0]
idx3 = Y - C[1] > slopeBC * (X - C[0])
img1[idx3] = (0,0,0)
plt.imshow(img1)
plt.show()
```



[40]:
```
# 직선 DA 아랫부분
idx4 = Y - A[1] < slopeBC * (X - A[0])
img1[idx4] = (0,0,0)
plt.imshow(img1)
plt.show()
```



[41]:
```
face = misc.face()
A = np.array([350, 380])
B = np.array([170, 650])
C = np.array([350, 920])
D = np.array([530, 650])
# 직선 AB 윗 부분
slopeAB = (B-A)[1] / (B-A)[0]
idx1 = Y - A[1] < slopeAB * (X - A[0])
idx2 = Y - C[1] > slopeAB * (X - C[0])
slopeBC = (C-B)[1] / (C-B)[0]
idx3 = Y - C[1] > slopeBC * (X - C[0])
idx4 = Y - A[1] < slopeBC * (X - A[0])
idx = idx1 & idx2 & idx3 & idx4
img2 = face.copy()
img2[idx] = (0,0,0)
plt.imshow(img1)
plt.show()
```



NumPy 다차원 배열을 이용한 행렬 연산

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 5 \end{bmatrix}, ; ; \mathbf{B} = \begin{bmatrix} -1 & 3 & 5 \\ 1 & 4 & 2 \end{bmatrix}$$

두 행렬의 덧셈과 뺄셈은 각각 다음과 같다.

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} 1-1 & 2+3 & 3+5 \\ 3+1 & 2+4 & 5+2 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 8 \\ 4 & 6 & 7 \end{bmatrix},$$

$$\mathbf{A} - \mathbf{B} = \begin{bmatrix} 1-(-1) & 2-3 & 3-5 \\ 3-1 & 2-4 & 5-2 \end{bmatrix} = \begin{bmatrix} 2 & -1 & -2 \\ 2 & -2 & 3 \end{bmatrix}.$$

NumPy ndarray를 이용한 연산

[42]:
```
a = np.array([1, 2, 3, 3, 2, 5]).reshape(2,3)
b = np.array([[-1,3,5],[1,4,2]])
c = a + b
print(a)
print(b)
print(c)
```

```
[[1 2 3]
 [3 2 5]]
[[-1  3  5]
 [ 1  4  2]]
[[0 5 8]
 [4 6 7]]
```

[43]:
```
np.add(a,b)
```

[43]:
```
array([[0, 5, 8],
       [4, 6, 7]])
```

[44]:
```
a - b
```

[44]:
```
array([[ 2, -1, -2],
       [ 2, -2,  3]])
```

[45]:
```
np.subtract(a,b)
```

[45]:
```
array([[ 2, -1, -2],
       [ 2, -2,  3]])
```

NumPy 다차원 배열의 원소 단위 4칙 연산

[46]:
```
a * b
```

[46]:
```
array([[-1,  6, 15],
       [ 3,  8, 10]])
```

[47]:
```
np.multiply(a,b)
```

[47]:
```
array([[-1,  6, 15],
       [ 3,  8, 10]])
```

[48]:
```
a / b
```

[48]:
```
array([[-1.        ,  0.66666667,  0.6       ],
       [ 3.        ,  0.5       ,  2.5       ]])
```

[49]:
```
np.divide(a,b)
```

[49]:
```
array([[-1.        ,  0.66666667,  0.6       ],
       [ 3.        ,  0.5       ,  2.5       ]])
```

[50]:
```
np.true_divide(a,b)
```

[50]:
```
array([[-1.        ,  0.66666667,  0.6       ],
       [ 3.        ,  0.5       ,  2.5       ]])
```

[51]:
```
a // b
```

[51]:
```
array([[-1,  0,  0],
       [ 3,  0,  2]], dtype=int32)
```

[52]:
```
np.floor_divide(a,b)
```

[52]:
```
array([[-1,  0,  0],
       [ 3,  0,  2]], dtype=int32)
```

[53]:
```
a % b
```

[53]:
```
array([[0, 2, 3],
       [0, 2, 1]], dtype=int32)
```

[54]:
```
np.mod(a,b)
```

[54]:
```
array([[0, 2, 3],
       [0, 2, 1]], dtype=int32)
```

[55]:
```
np.divmod(a,b)
```

[55]:
```
(array([[-1,  0,  0],
        [ 3,  0,  2]], dtype=int32),
 array([[0, 2, 3],
        [0, 2, 1]], dtype=int32))
```

[56]:
```
a**2
```

[56]:
```
array([[ 1,  4,  9],
       [ 9,  4, 25]], dtype=int32)
```

[57]:
```
pow(a,2)
```

[57]:
```
array([[ 1,  4,  9],
       [ 9,  4, 25]], dtype=int32)
```

[58]:
```
np.power(a,2)
```

[58]:
```
array([[ 1,  4,  9],
       [ 9,  4, 25]], dtype=int32)
```

[59]:
```
np.sqrt(b)
```

```
C:\Users\Public\Documents\ESTsoft\CreatorTemp/ipykernel_43000/3240365511.py:1:
RuntimeWarning: invalid value encountered in sqrt
  np.sqrt(b)
```

[59]:
```
array([[       nan, 1.73205081, 2.23606798],
       [1.        , 2.        , 1.41421356]])
```

[60]:
```
b ** 0.5
```

```
C:\Users\Public\Documents\ESTsoft\CreatorTemp/ipykernel_43000/4166513202.py:1:
RuntimeWarning: invalid value encountered in power
  b ** 0.5
```

[60]:
```
array([[       nan, 1.73205081, 2.23606798],
       [1.        , 2.        , 1.41421356]])
```

[61]:
```
1 / a
```

[61]:
```
array([[1.        , 0.5       , 0.33333333],
       [0.33333333, 0.5       , 0.2       ]])
```

[62]:
```
pow(a,b)
```

```
      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       C:\Users\Public\Documents\ESTsoft\CreatorTemp/ipykernel_43000/2156627119.
↪py in <module>
   ----> 1 pow(a,b)
```

```
ValueError: Integers to negative integer powers are not allowed.
```

[ ]:
```
a = np.array([1,2,3])
b = np.array([2,3,4])
a ** b
```

[63]:
```
np.power(a,b)
```

```
␣
↪---------------------------------------------------------------------------

ValueError                                Traceback (most recent call␣
↪last)

C:\Users\Public\Documents\ESTsoft\CreatorTemp/ipykernel_43000/3436113933.
↪py in <module>
----> 1 np.power(a,b)

ValueError: Integers to negative integer powers are not allowed.
```

[64]:
```
2 << a
```

[64]:
```
array([[ 4,  8, 16],
       [16,  8, 64]], dtype=int32)
```

[65]:
```
a << 2
```

[65]:
```
array([[ 4,  8, 12],
       [12,  8, 20]], dtype=int32)
```

NumPy의 수학 함수: 삼각 함수

[66]:
```
import numpy as np
x = np.array([0, 30, 45, 60, 90])
x = np.radians(x)
y = np.sin(x)
z = np.cos(x)
print(y)
print(z)
```

```
[0.         0.5        0.70710678 0.8660254  1.        ]
[1.00000000e+00 8.66025404e-01 7.07106781e-01 5.00000000e-01
 6.12323400e-17]
```

[67]:
```
y = np.sin(x)
z = np.cos(x)
print(y)
print(z)
with np.printoptions(precision=3):
    print(y)
    print(z)
with np.printoptions(precision=3, suppress=True):
    print(y)
    print(z)
```

```
[0.         0.5        0.70710678 0.8660254  1.        ]
[1.00000000e+00 8.66025404e-01 7.07106781e-01 5.00000000e-01
 6.12323400e-17]
[0.    0.5   0.707 0.866 1.   ]
[1.000e+00 8.660e-01 7.071e-01 5.000e-01 6.123e-17]
[0.    0.5   0.707 0.866 1.   ]
[1.    0.866 0.707 0.5   0.   ]
```