

# AI Programming

Lecture 21

# Review

- **NumPy array**
  - Attributes: shape, ndim,...
  - Methods: max, min, mean, flatten
  - Functions: append, random, zeros, ones, full, eye, arrange, linspace

# Review

- Append

```
a = np.array([1, 2, 3])
b = np.array([[4, 5, 6], [7, 8, 9]])
np.append(a, b)
```

```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
np.append(b, a)
```

```
array([4, 5, 6, 7, 8, 9, 1, 2, 3])
```

```
np.append([1, 2, 3], [4, 5, 6])
```

```
array([1, 2, 3, 4, 5, 6])
```

```
a = np.array([[1, 2, 3]])
b = np.array([[4, 5, 6], [7, 8, 9]])
np.append(a, b, axis=0)
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
np.append(a, b, axis=1)
```

```
np.append([[1, 2], [3, 4]], [[5, 6], [7, 8]], axis=0)
```

```
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
```

```
np.append([[1, 2], [3, 4]], [[5, 6], [7, 8]], axis=1)
```

```
array([[1, 2, 5, 6],
       [3, 4, 7, 8]])
```

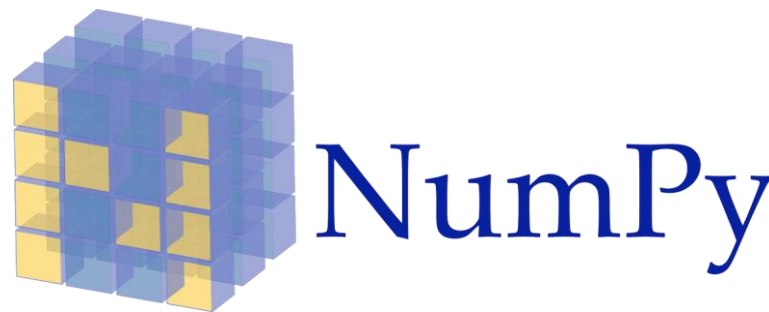
```
np.append([[1, 2], [3, 4]], [[5, 6]], axis=0)
```

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

```
np.append([[1, 2], [3, 4]], [5, 6], axis = 0)
```

# Preview

- NumPy ([link](#))
  - 5. ndarray의 재구성
  - 6. 다차원 배열의 축
  - 7. 배열의 인덱싱과 슬라이싱
  - 8. 2차원 배열의 인덱싱
  - 9. 2차원 배열의 슬라이싱



# 5 ndarray의 재구성

# Reshape

- Reshape

```
np.arange(0, 10).reshape(2, 5)
```

```
array([[0, 1, 2, 3, 4],  
       [5, 6, 7, 8, 9]])
```


```
np.arange(0, 10).reshape(5, 2)
```

```
array([[0, 1],  
       [2, 3],  
       [4, 5],  
       [6, 7],  
       [8, 9]])
```

```
np.arange(0, 10).reshape(2, 5)
```

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

(10,) : 1차원 배열(10개의 원소)



0	1	2	3	4
5	6	7	8	9

```
np.arange(0, 10).reshape(3, 3)
```

```
-----  
ValueError                                Traceback (most recent call last)  
<ipython-input-119-b2159be785fe> in <module>()  
----> 1 np.arange(0, 10).reshape(3, 3)
```

ValueError: cannot reshape array of size 10 into shape (3,3)

# Reshape

- 3D array (tensor)

```
np.arange(24).reshape(4, 3, 2)
```

```
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]]])
```

```
np.arange(24).reshape(4, 3, -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]]])
```

```
np.arange(24).reshape(4, -1, 2)
```

```
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]]])
```

# Reshape

- Reshape of multi-dimensional array

```
a = np.arange(16).reshape(4, 4)  
a.reshape(2, 8)
```

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

```
a.reshape(8, 2)
```

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

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p



a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



a	b	c	d	e	f	g	h
i	j	k	l	m	n	o	p



# Reshape

- **Transpose**

```
a = np.arange(6).reshape(2, 3)  
a
```

```
array([[0, 1, 2],  
       [3, 4, 5]])
```

```
np.transpose(a)
```

```
array([[0, 3],  
       [1, 4],  
       [2, 5]])
```

```
a.transpose()
```

```
array([[0, 3],  
       [1, 4],  
       [2, 5]])
```

```
a.T
```

```
array([[0, 3],  
       [1, 4],  
       [2, 5]])
```

# Reshape

## • Exercise



### LAB 11-5 : 배열의 재구성

1. `arange()` 함수를 사용하여 1에서 12까지의 원소를 가지는 1차원 배열 `a1`을 생성하여라. 그리고 이 `a1` 배열을 `reshape()` 메소드를 사용하여 2행 6열의 행렬로 재구성하여라.

```
a1 = [[ 1 2 3 4 5 6]
      [ 7 8 9 10 11 12]]
```

2. `arange()` 함수를 사용하여 1에서 30까지의 원소를 가지는 1차원 배열 `a2`을 생성하여라. 그리고 이 `a2` 배열을 `reshape()` 메소드를 사용하여 3행 10열의 행렬로 재구성하여라.

```
a2 = [[ 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]]
```

```
a1 = np.arange(1, 13).reshape(2, 6)
a1
```

```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12]])
```

```
a2 = np.arange(1, 31).reshape(3, 10)
a2
```

```
array([[ 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]])
```

# Reshape

- (Cont'd)

3. 문제 2에서 생성한 a2 배열을 reshape() 메소드를 사용하여 6행 5열의 행렬로 재구성한 행렬 a3을 생성하여라.

```
a3 = [[ 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]]
```

4. 문제 3에서 생성한 a3 배열을 transpose() 함수를 사용하여 다음과 같은 전치 행렬을 생성하여라.

```
a4 = [[ 1 6 11 16 21 26]
      [ 2 7 12 17 22 27]
      [ 3 8 13 18 23 28]
      [ 4 9 14 19 24 29]
      [ 5 10 15 20 25 30]]
```

```
a3 = a2.reshape(6, 5)
a3
```

```
array([[ 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]])
```

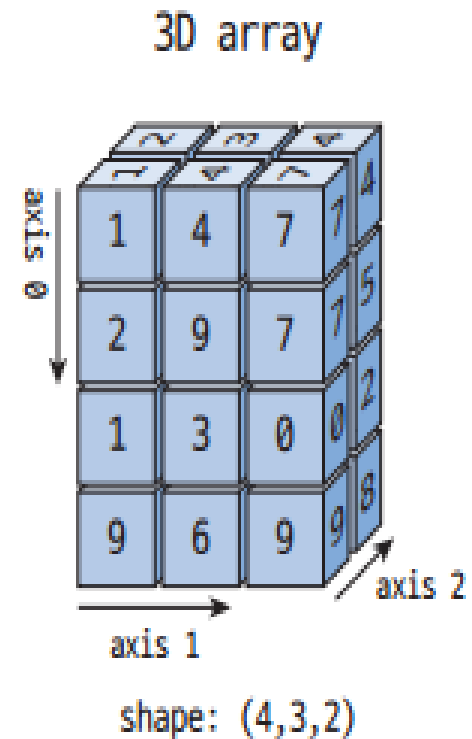
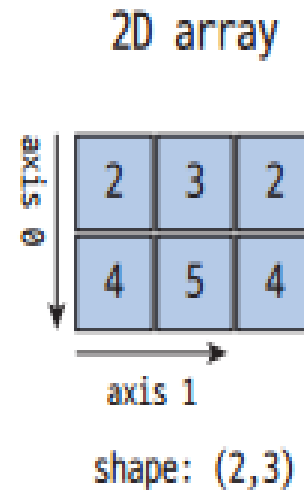
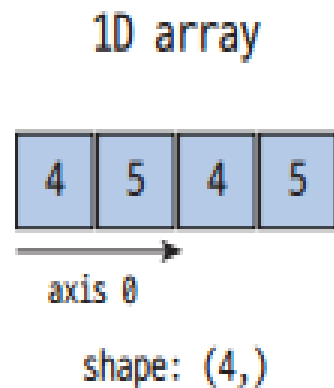
```
a4 = np.transpose(a3)
a4
```

```
array([[ 1,  6, 11, 16, 21, 26],
       [ 2,  7, 12, 17, 22, 27],
       [ 3,  8, 13, 18, 23, 28],
       [ 4,  9, 14, 19, 24, 29],
       [ 5, 10, 15, 20, 25, 30]])
```

## 6 다차원 배열의 축

# Axis

- Axis of multi-dimensional array



# Axis

## • Sum

```
a = np.arange(6).reshape(3, 2)
a
```

```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

```
a.sum()
```

```
15
```

```
a.sum(axis=0)
```

```
array([6, 9])
```

```
a.sum(axis=1)
```

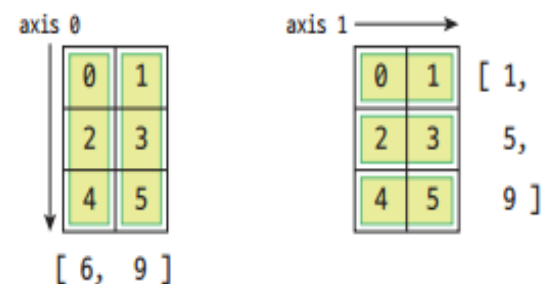
```
array([1, 5, 9])
```

```
a.sum(axis=0, keepdims=True)
```

```
array([[6, 9]])
```

```
a.sum(axis=1, keepdims=True)
```

```
array([[1],
       [5],
       [9]])
```



[그림 11-11] axis 0과 axis 1에 대하여 각각 sum() 함수를 수행한 결과

# Axis

```
a = np.arange(6).reshape(3, 2)
a
```

```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

- Max, min, mean

```
a.min(axis=0)
```

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

```
a.max(axis=0)
```

```
array([4, 5])
```

```
a.mean(axis=0)
```

```
array([2., 3.])
```

```
a.min(axis=1)
```

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

```
a.max(axis=1)
```

```
array([1, 3, 5])
```

```
a.mean(axis=1)
```

```
array([0.5, 2.5, 4.5])
```

```
a.min(axis=0, keepdims=True)
```

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

```
a.max(axis=0, keepdims=True)
```

```
array([[4, 5]])
```

```
a.mean(axis=0, keepdims=True)
```

```
array([[2., 3.]])
```

```
a.min(axis=1, keepdims=True)
```

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

```
a.max(axis=1, keepdims=True)
```

```
array([[1],
       [3],
       [5]])
```

```
a.mean(axis=1, keepdims=True)
```

```
array([[0.5],
       [2.5],
       [4.5]])
```

## 7 배열의 인덱싱과 슬라이싱



# Indexing and Slicing

- 1D array

```
a = np.array([1, 2, 3])  
print(a[0], a[1], a[2])
```

```
1 2 3
```

```
print(a[-1], a[-2], a[-3])
```

```
3 2 1
```

```
print(a[0:1], a[0:2], a[:])
```

```
[1] [1 2] [1 2 3]
```

```
b = np.arange(10)  
b[1:5]
```

```
[1 2 3 4]
```

```
b[1:]
```

```
[1 2 3 4 5 6 7 8 9]
```

```
b[::2]
```

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

```
b[::-1]
```

```
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

# Indexing and Slicing

- Exercise



## LAB 11-7 : 배열의 인덱싱과 슬라이싱

1. 1에서 10까지의 원소를 가지는 1차원 배열 `a`를 생성하여라. 이 `a`를 인덱싱하여 `[2, 4, 6, 8]`의 원소를 가진 배열 `b`를 생성하여 다음과 같이 출력하여라.

```
a = [ 1 2 3 4 5 6 7 8 9 10]
b = [2 4 6 8]
```

2. 문제 1에서 생성한 배열 `a`를 슬라이싱하여 다음과 같은 배열 `b`, `c`, `d`, `e`, `f`를 생성하여라.

```
b = [6 7 8 9 10]
c = [7 8 9 10]
d = [1 2 3]
e = [1 3 5 7 9]
f = [10 8 6 4 2]
```

```
a = np.arange(1, 11)
a[1:8:2]
```

```
array([2, 4, 6, 8])
```

```
a[5:]
```

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

```
a[6:]
```

```
array([ 7,  8,  9, 10])
```

```
a[0:3]
```

```
array([1, 2, 3])
```

```
a[::2]
```

```
array([1, 3, 5, 7, 9])
```

```
a[::-2]
```

```
array([10,  8,  6,  4,  2])
```

## 8 2차원 배열의 인덱싱

# Indexing

- 2D arrays

```
a = np.arange(6).reshape(3, 2)
a
```

```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

```
print(a[0, 0], a[0, 1], a[1, 0], a[1, 1], a[2, 0], a[2, 1])
```

```
0 1 2 3 4 5
```

```
a[0, :]
```

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

```
a[1]
```

```
array([2, 3])
```

axis = 0

행 방향 인덱스 i

0	1
2	3
4	5

열 방향 인덱스 j

axis = 1

a[0, 0]	a[0, 1]
a[1, 0]	a[1, 1]
a[2, 0]	a[2, 1]

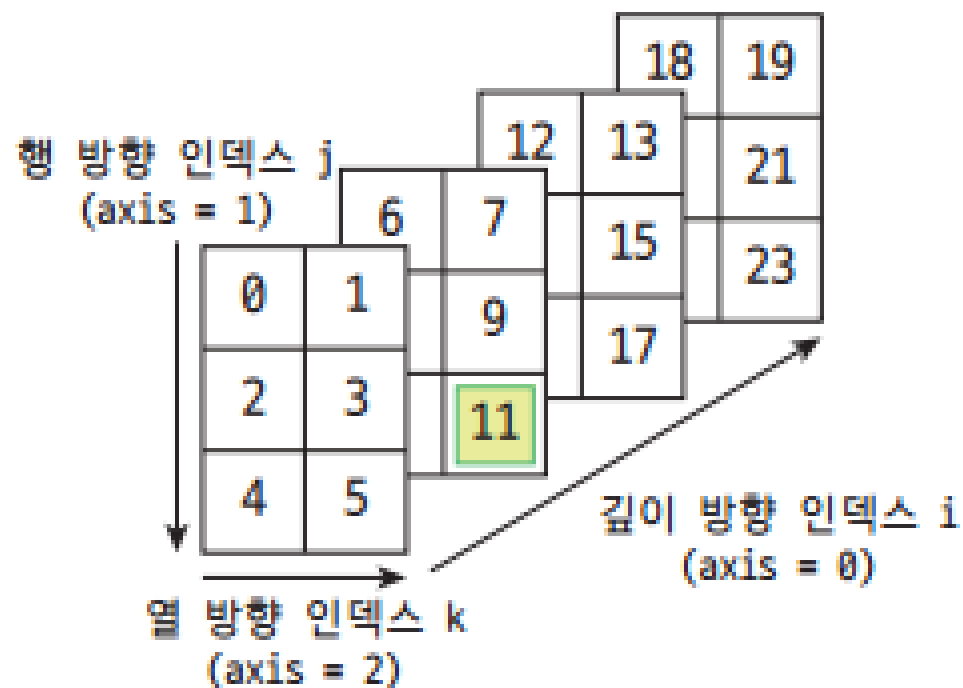
a[i, j] 인덱스의 위치

```
d = np.array([[1, 2], [3, 4]])
print(d[0, 0], d[0, 1], d[1, 0], d[1, 1])
print(d[:, 0])
print(d[1, :])
```

```
1 2 3 4
[1 3]
[3 4]
```

# Indexing

- 3D array



```
a = np.arange(24).reshape(4, 3, 2)
a
```

```
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]])
```

```
a[1, 2, 1]
```

# Indexing

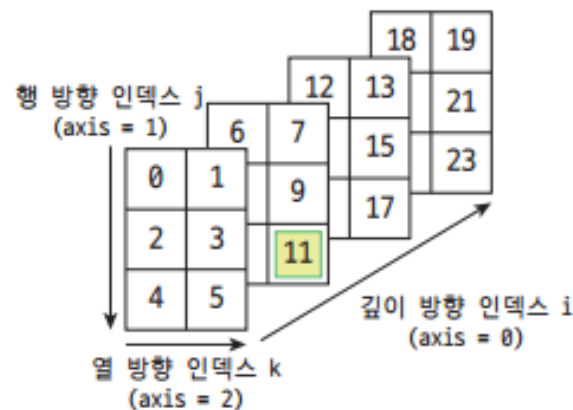
## • Exercise



### LAB 11-8 : 3차원 배열의 인덱싱

1. [그림 11-14]의 (4, 3, 2) 형태의 3차원 배열을 다음과 같이 인덱스와 원소 값이 나타나도록 출력하시오(힌트 : for 문과 `nindex(a, shape)` 함수를 사용하도록 한다).

index	element
(0, 0, 0)	0
(0, 0, 1)	1
(0, 1, 0)	2
(0, 1, 1)	3
...(생략)	
(3, 2, 0)	22
(3, 2, 1)	23



```
a = np.arange(24).reshape(4, 3, 2)
for i in range(a.shape[0]):
    for j in range(a.shape[1]):
        for k in range(a.shape[2]):
            print("{0:d}, {1:d}, {2:d}){3:7d}".format(i, j, k, a[i, j, k]))
```

```
(0, 0, 0) 0
(0, 0, 1) 1
(0, 1, 0) 2
(0, 1, 1) 3
(0, 2, 0) 4
(0, 2, 1) 5
(1, 0, 0) 6
(1, 0, 1) 7
(1, 1, 0) 8
(1, 1, 1) 9
(1, 2, 0) 10
(1, 2, 1) 11
(2, 0, 0) 12
(2, 0, 1) 13
(2, 1, 0) 14
(2, 1, 1) 15
(2, 2, 0) 16
(2, 2, 1) 17
(3, 0, 0) 18
(3, 0, 1) 19
(3, 1, 0) 20
(3, 1, 1) 21
(3, 2, 0) 22
(3, 2, 1) 23
```

## 9 2차원 배열의 슬라이싱

# Slicing

- 2D array

0	1	2
3	4	5
6	7	8

`a[0]`  
`a[0, :]`

0	1	2
3	4	5
6	7	8

`a[0, 0:2]`

0	1	2
3	4	5
6	7	8

`a[:2, :2]`

0	1	2
3	4	5
6	7	8

`a[1:, 1:]`

0	1	2
3	4	5
6	7	8

`a[1, 1:]`



# Slicing

- (Cont'd)

```
a = np.arange(9).reshape(3, 3)
print(a[0])
print(a[0, :])
print(a[:, 0])
```

```
[0 1 2]
[0 1 2]
[0 3 6]
```

```
print(a[0, 0:2])
print(a[0, :2])
```

```
[0 1]
[0 1]
```

```
print(a[0:2, 0:2])
print(a[:2, :2])
```

```
[[0 1]
 [3 4]]
[[0 1]
 [3 4]]
```

```
print(a[1:, 1:])
```

```
[[4 5]
 [7 8]]
```

```
print(a[1, 1:])
```

```
[4 5]
```

```
a[1, 1:].shape
```

```
(2,)
```

```
a[1:2, 1:].shape
```

```
(1, 2)
```

# Slicing

## • Exercise



### LAB 11-9 : 2차원 배열의 슬라이싱

1. 아래 그림과 같이 0에서 15까지의 연속적인 값을 원소로 가지는 4x4 크기의 2차원 배열 `a`를 생성하여라. 이 `a`에 대하여 인덱싱과 슬라이싱을 적용하여 다음과 같은 `b`, `c`, `d`, `e` 배열을 구하여라.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

`b`

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

`c`

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

`d`

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

`e`

```
b = [ 1 5 9 13]
c = [ 9 10 11]
d = [[ 0 1] [ 4 5]]
e = [[ 5 6] [ 9 10]]
```

```
a = np.arange(16).reshape(4, 4)
a
```

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

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

```
array([ 1,  5,  9, 13])
```

```
a[2, 1:]
```

```
array([ 9, 10, 11])
```

```
a[:2, :2]
```

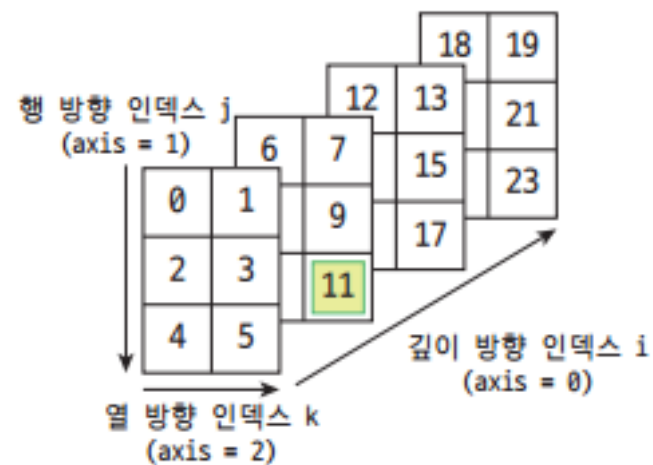
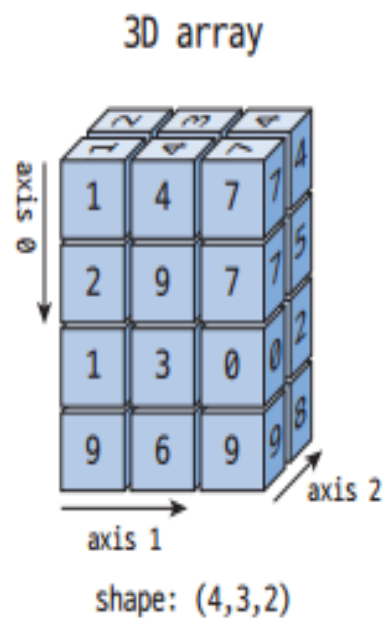
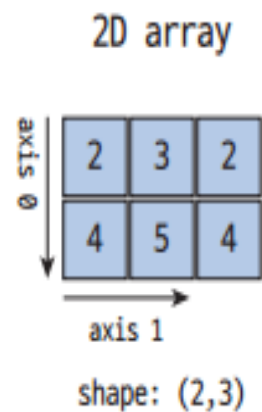
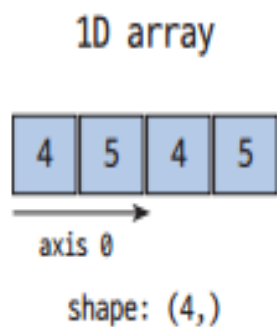
```
array([[0, 1],
       [4, 5]])
```

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

```
array([[ 5,  6],
       [ 9, 10]])
```

# Summary

- Axis



# Summary

- Slicing

0	1	2
3	4	5
6	7	8

`a[0]`  
`a[0, :]`

0	1	2
3	4	5
6	7	8

`a[0, 0:2]`

0	1	2
3	4	5
6	7	8

`a[:2, :2]`

0	1	2
3	4	5
6	7	8

`a[1:, 1:]`

0	1	2
3	4	5
6	7	8

`a[1, 1:]`

# Assignment 16

- Practice

```
n = int(input('n을 입력하시오 : '))
a = np.ones((n, n), dtype='int32')
print('a 행렬')
print(a)
b = a
b[1:-1, 1:-1] = 0
print('b 행렬')
print(b)
```

```
n을 입력하시오 : 3
a 행렬
[[1 1 1]
 [1 1 1]
 [1 1 1]]
b 행렬
[[1 1 1]
 [1 0 1]
 [1 1 1]]
```

```
n을 입력하시오 : 5
a 행렬
[[1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]
 [1 1 1 1 1]]
b 행렬
[[1 1 1 1 1]
 [1 0 0 0 1]
 [1 0 0 0 1]
 [1 0 0 0 1]
 [1 1 1 1 1]]
```

# Assignment 16

- Problem 1

n을 입력하시오 : 3

a 행렬

$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$

$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$

$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$

b 행렬

$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$

$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$

$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$

n을 입력하시오 : 5

a 행렬

$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

b 행렬

$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \end{bmatrix}$

$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \end{bmatrix}$

$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \end{bmatrix}$

$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$

# Assignment 16

- Problem 2

n을 입력하시오 : 5

```
[[1 0 0 0 0]
 [1 1 0 0 0]
 [1 1 1 0 0]
 [1 1 1 1 0]
 [1 1 1 1 1]]
```

n을 입력하시오 : 7

```
[[1 0 0 0 0 0 0]
 [1 1 0 0 0 0 0]
 [1 1 1 0 0 0 0]
 [1 1 1 1 0 0 0]
 [1 1 1 1 1 0 0]
 [1 1 1 1 1 1 0]
 [1 1 1 1 1 1 1]]
```