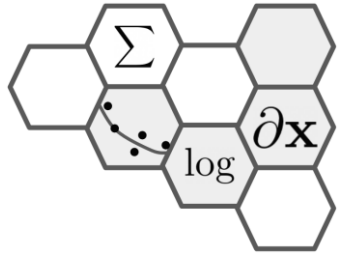
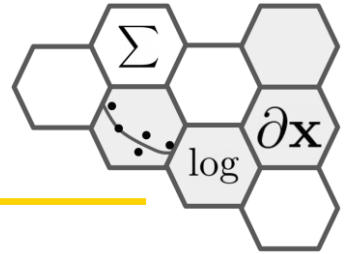


ML·DL Intermediate Course

numpy

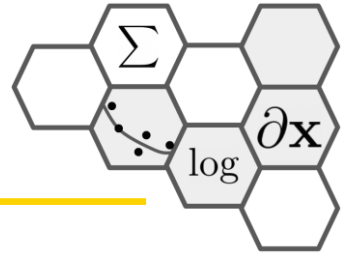
조준우
metamath@gmail.com

넘파이Numpy



- Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB®, you might find this tutorial useful to get started with Numpy. – cs231n

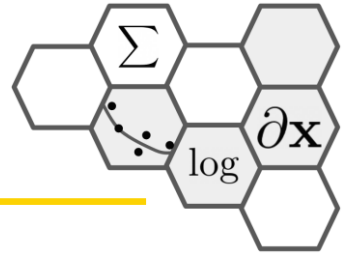
Import



- 약칭을 np로 쓰는 것이 거의 표준

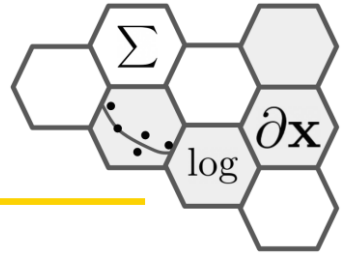
```
import numpy as np
```

ndarray



- numpy에서 제공하는 메인 객체인 배열
- 다차원 배열multidimensional array로써 1차원 배열(벡터), 2차원 배열(행렬), 3차원 배열(큐브), 4차원 배열(큐브가 여러 개 모인 것) 등등 계속 차원을 늘릴 수 있음

ndarray 생성



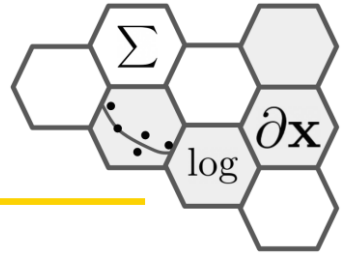
- 생성 : 행 우선 방식으로 [ ]로 적어주면 됨
- Python 리스트와 만드는 방식 동일

```
import numpy as np

# 값을 지정하여 생성
A = np.array([[1,2],[3,4]])
```

순차적으로 다음 행을 나열

가로(행)으로 먼저 숫자를 나열하고

ndarray 생성

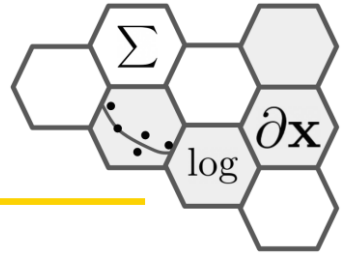


- 다양한 생성 방법

```
import numpy as np

# 랜덤 생성
E = np.random.rand(10)
F = np.random.rand(25).reshape(5,-1)
G = np.random.randint(1, 11, 9).reshape(3,3)
```

```
# 모든 요소가 0인 어레이
B = np.zeros(3,3)

# 모든 요소가 1인 어레이
C = np.ones(2,3)

# 대각 요소만 1인 어레이
D = np.eye(2)
```

Numpy 어레이 연산



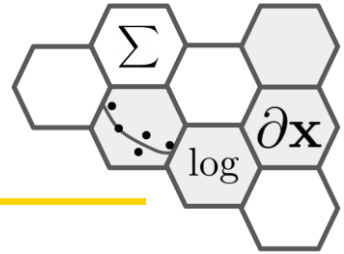
- 기본연산: 덧셈, 뺄셈, 곱셈, 나눗셈은 요소끼리 계산

- Numpy 어레이 곱 ≠ 행렬곱

```
c = np.ones((3, 3))
C = np.matrix(c)

print("numpy array multiplication")
print(c*c)

print("numpy dot function")
print(c.dot(c))

print("numpy matrix multiplication")
print(C*C)
```

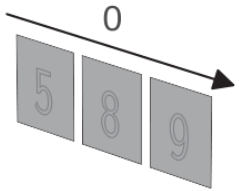
```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]

[[3. 3. 3.]
 [3. 3. 3.]
 [3. 3. 3.]]
```

축axis

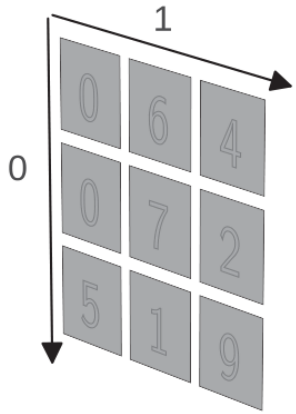


- 배열의 요소가 늘어선 방향
- 축에 번호를 붙여서 관리하는데 0부터 번호가 증가, 새롭게 생긴 axis가 0번

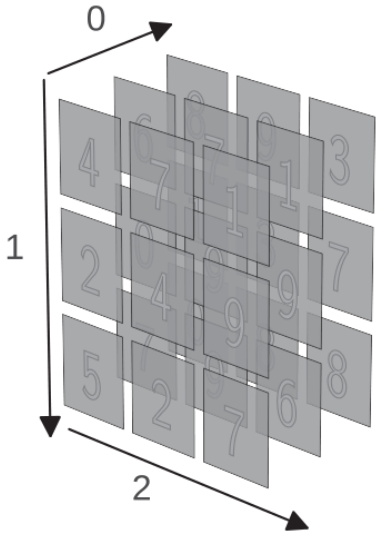
1D tensor: vector



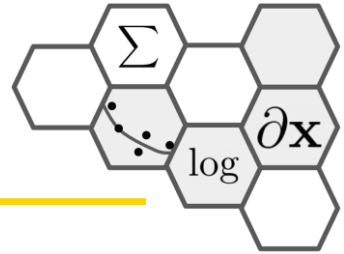
2D tensor: matrix



3D tensor



shape



- 배열의 모양

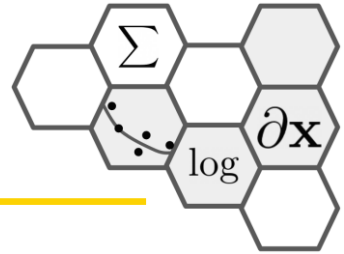
```
x = np.array([1,2,3])
print(x.shape)    # (3,)
```

기본적으로 숫자 3개가 나열 됨
1차원 어레이

```
x = np.array([[1,2,3], [1,2,3]])
print(x.shape)    # (2,3)
```

0번 축으로 2개, 1번 축으로 3개가 나열됨

reshape



- 기본 1차원 배열

```
x = np.array([1,2,3])
print(x.shape)  # (3,)
```

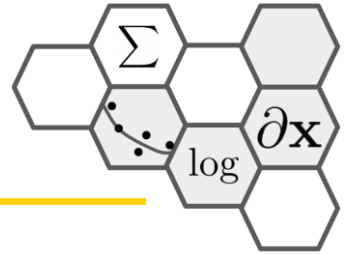
숫자가 가로로 나열되게

```
x_row_vector = x.reshape(1,3)
print(x_row_vector)
x_row_vector.shape
```

숫자가 세로로 나열되게

```
x_col_vector1 = x.reshape(3,1)
print(x_col_vector1)
print(x_col_vector1.shape)
```

reshape



- 2차원 이상에서 배열의 모양 변경

```
A = np.arange(12).reshape(4,3)
```

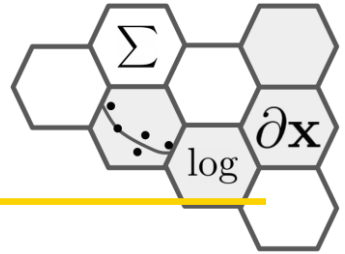
차원추가

```
A_ = A.reshape(-1, 2, 3)
# (2,2,3)
```

차원축소

```
A_.reshape(-1,3)
# (4,3)
```

자주쓰는 기능: arange



- Return evenly spaced values within a given interval.

start

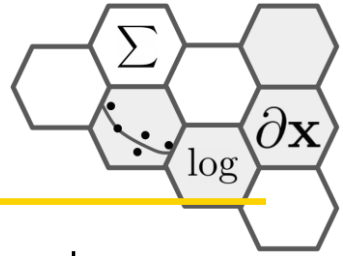
end

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

```
np.arange(1, 20, 2)
```

step

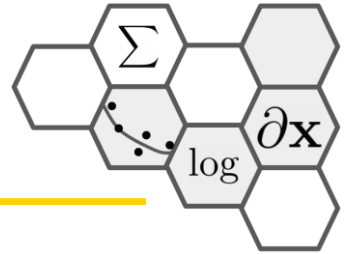
자주쓰는 기능: linspace



- Return evenly spaced numbers over a specified interval.

start
end
```
np.linspace(0, 10, 5)
```
number

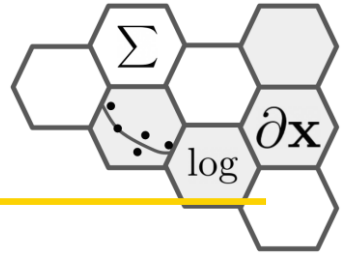
인덱싱indexing



- 배열의 요소에 접근하기 위해서는 인덱스를 사용
- 인덱스는 0부터 시작, 끝 인덱스 포함 안됨
- 2축 이상에 대한 인덱싱
- 3행 3열 요소 A[2,2]
- [start:end:stride]
  - A[:, 0::2]: 행은 모든 행 ->:, 열은 start 0번부터, end 생략, stride는 2

```
import numpy as np

A = np.arange(30).reshape(5,6)
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],
           [24, 25, 26, 27, 28, 29]])

      array([[ 0,  2,  4],
             [ 6,  8, 10],
             [12, 14, 16],
             [18, 20, 22],
             [24, 26, 28]])
```

어레이 인덱싱array indexing



- 인덱싱할 숫자를 임의의 어레이로 전달하는 방법
- 숫자를 지정할 자리에 숫자 대신 숫자 여러 개를 포함한 어레이를 전달하면 해당되는 요소 여러 개가 한번에 추출

```
np.array([A[0,1], A[1,2], A[2,3], A[3,1], A[4,0]])
```

→ 추출할 모든 요소를 다 지정

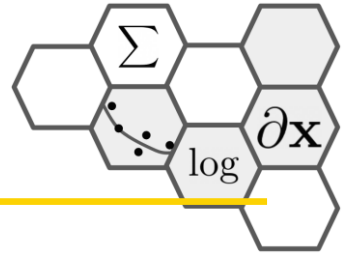
```
#>>> array([ 1,  8, 15, 19, 24])
```

```
A[[0,1,2,3,4], [1,2,3,1,0]]
```

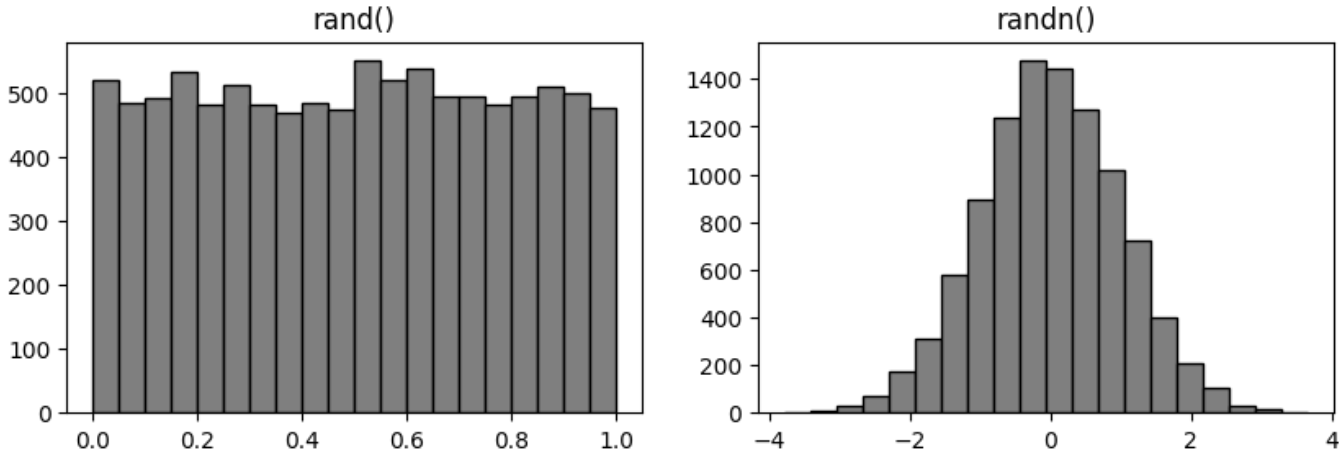
→ 추출할 모든 인덱스를 행과 열로 묶어 어레이로 지정

```
#>>> array([ 1,  8, 15, 19, 24])
```

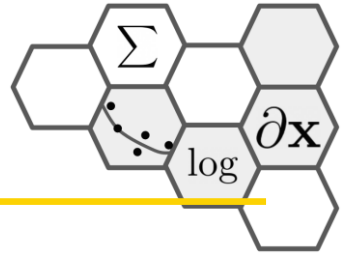
자주쓰는 기능: rand



- `rand(): np.random.rand(10000)`
  - Create an array of the given shape and populate it with random samples from a uniform distribution over [0, 1).
- `randn(): np.randon.randn(10000)`
  - Return a sample (or samples) from the “standard normal” distribution.



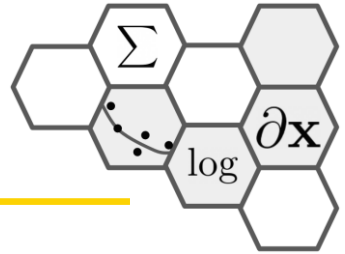
자주쓰는 기능: randint, choice



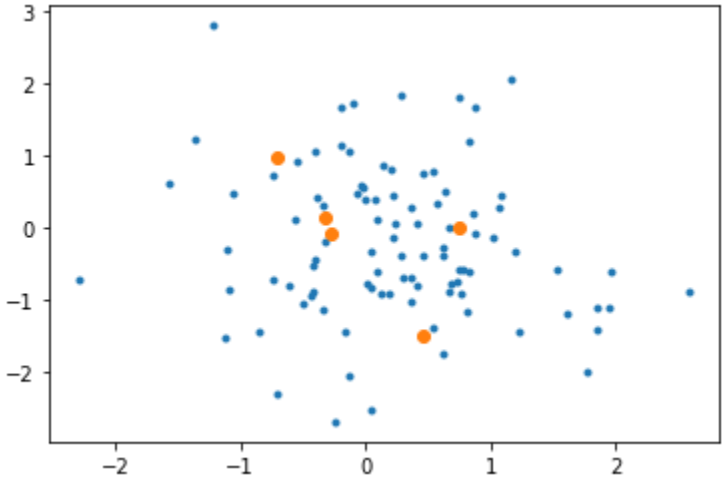
- `randint(): np.random.randint(0,10,5)`
  - Return random integers from the “discrete uniform” distribution of the specified dtype in the “half-open” interval [low, high). If high is None (the default), then results are from [0, low).

- `choice(): np.random.choice(np.random.randint(0,10,5), 3)`
  - Generates a random sample from a given 1-D array.

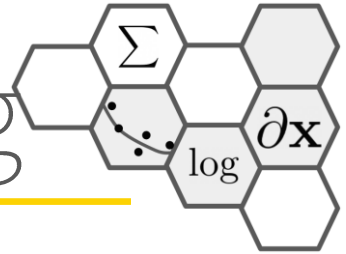
어레이 인덱싱 예제: 임의점 선택



- `X = np.random.randn(100, 2) # N,D`
- `# 임의의 점 5개 선택하기, np.random.choice, replace=False: 비복원 추출`
- `selected_idx = ...`



불린 인덱싱Boolean indexing



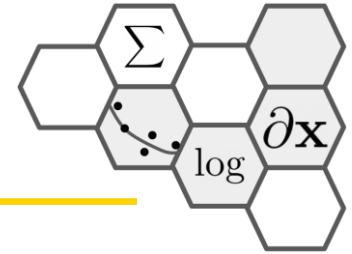
- Boolean 형 어레이를 전달

```
A > 20

#>>> array([[False, False, False, False, False, False],
           [False, False, False, False, False, False],
           [False, False, False, False, False, False],
           [False, False, False,  True,  True,  True],
           [ True,  True,  True,  True,  True,  True]])
```

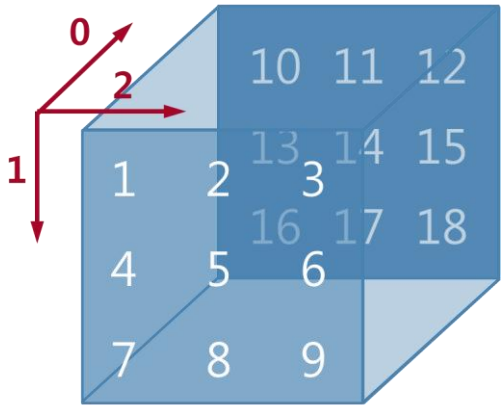
```
A[A > 20]

#>>> array([21, 22, 23, 24, 25, 26, 27, 28, 29])
```

20보다 큰 요소만 추출

전치transpose



- transpose(axis) : 행렬의 전치와 같은 역할, 3개축 이상에서도 같은 논리로 동작



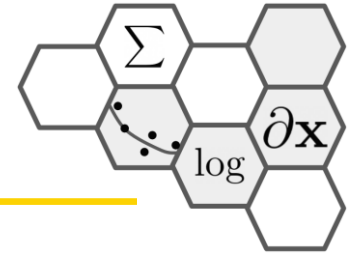

```
B = A[0,:]
print(B)
```

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

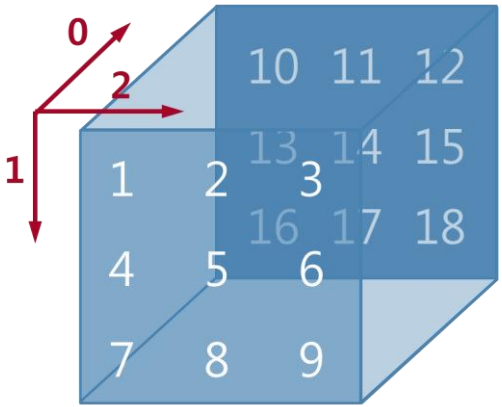
```
print(B.T)
```

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

전치transpose

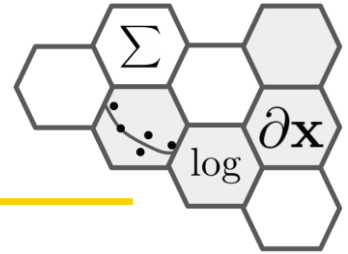


- transpose(axis) : 행렬의 전치와 같은 역할, 3개축 이상에서도 같은 논리로 동작

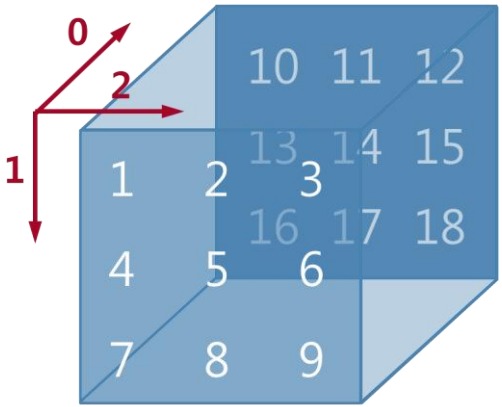



```
print(A.transpose(2, 0, 1))

[[[ 1  4  7]
  [10 13 16]]

 [[ 2  5  8]
  [11 14 17]]

 [[ 3  6  9]
  [12 15 18]]]
```

전치transpose



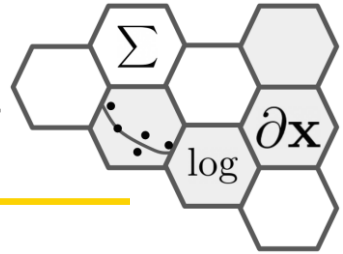
- transpose(axis) : 행렬의 전치와 같은 역할, 3개축 이상에서도 같은 논리로 동작



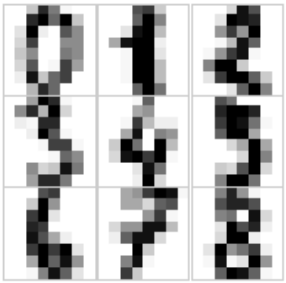

```
print(?)
```

```
[[[ 1, 10],
  [ 2, 11],
  [ 3, 12]],

 [[ 4, 13],
  [ 5, 14],
  [ 6, 15]],

 [[ 7, 16],
  [ 8, 17],
  [ 9, 18]]]
```

Matplotlib 예제 파일에서…

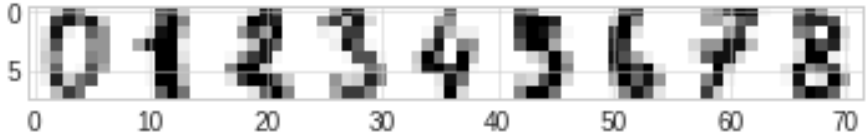


- transpose와 reshape을 이용한 이미지 모양 변형



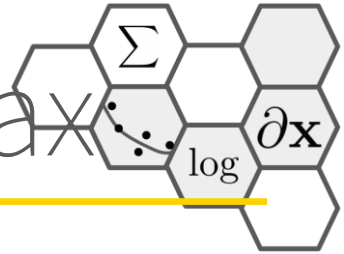
(9,8,8)

transpose & reshape



(8,72)

자주쓰는 기능: max, argmax



- `max()`
  - Return the maximum along a given axis.
- `argmax()`
  - Returns the indices of the maximum values along an axis.

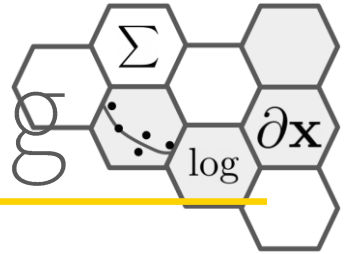
```
[[0.1891 0.1962 0.5885 0.6224 0.0469]    [0.6224
 [0.8457 0.8005 0.1416 0.9693 0.3967]]    0.9693]

[0.8457 0.8005 0.5885 0.9693 0.3967]
```

```
np.max(x)
np.argmax(x)
```

```
np.max(x, axis=0)
np.argmax(x, axis=0)
```

```
np.max(x, axis=1)
np.argmax(x, axis=1)
```

브로드캐스팅broadcasting



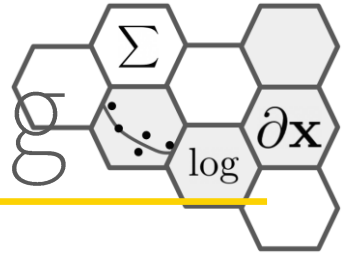
- Numpy에서 shape가 다른 배열 간에도 산술 연산이 가능하게 하는 메커니즘

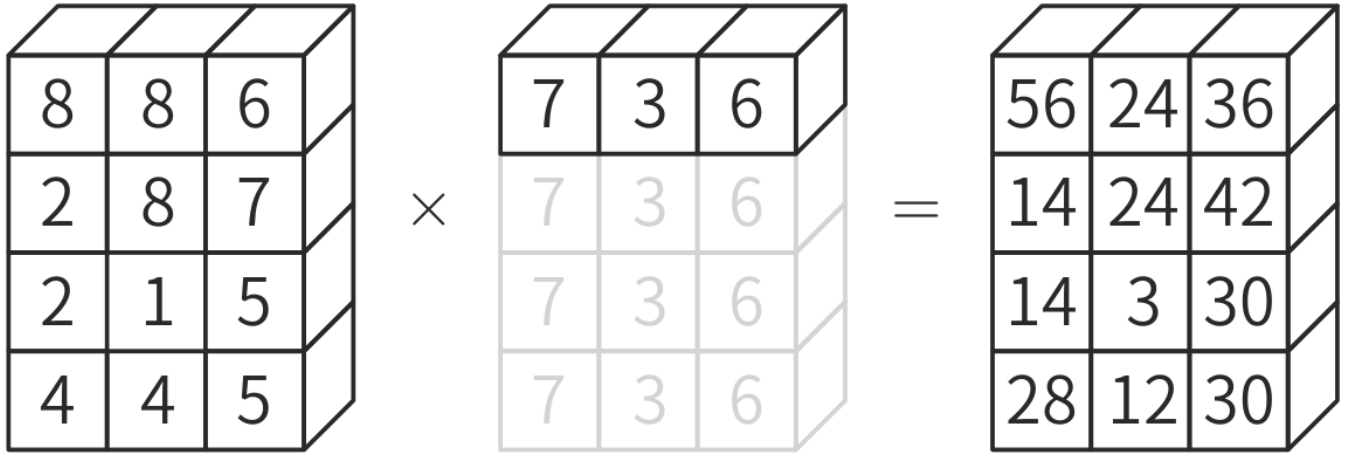
(3,)

(3,)　　(1,)

(3,)　　(3,)　　(3,)

브로드캐스팅broadcasting



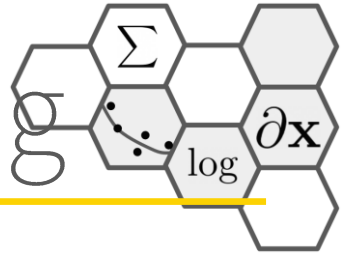


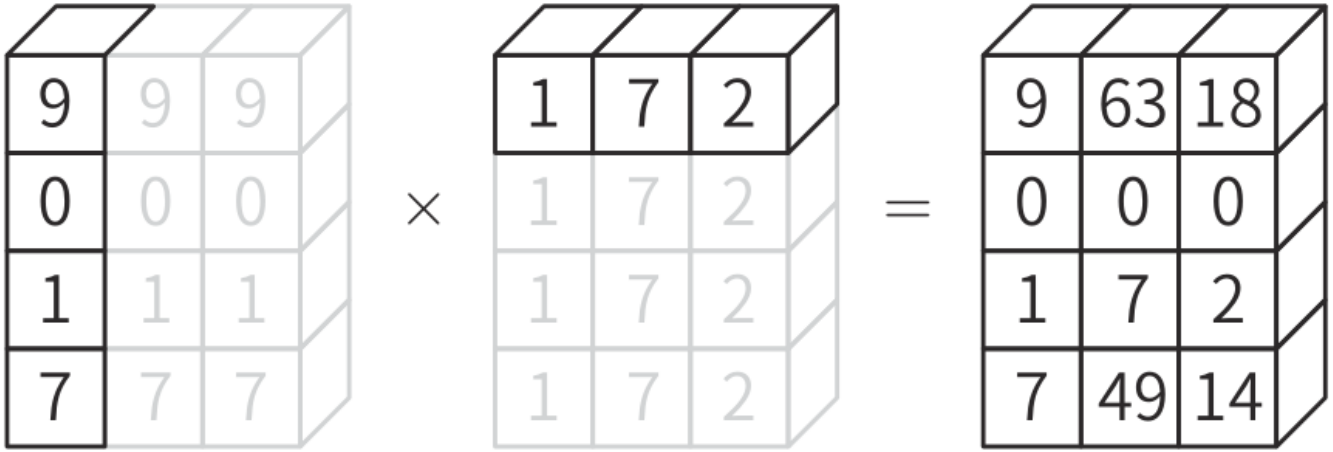
(4,3)
(4,3)
(4,3)

(3,)
(1,3)
(4,3)

(4,3)

브로드캐스팅broadcasting



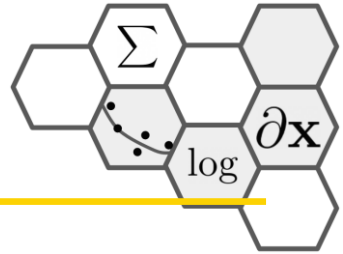


(4,1)
(4,1)
(4,3)

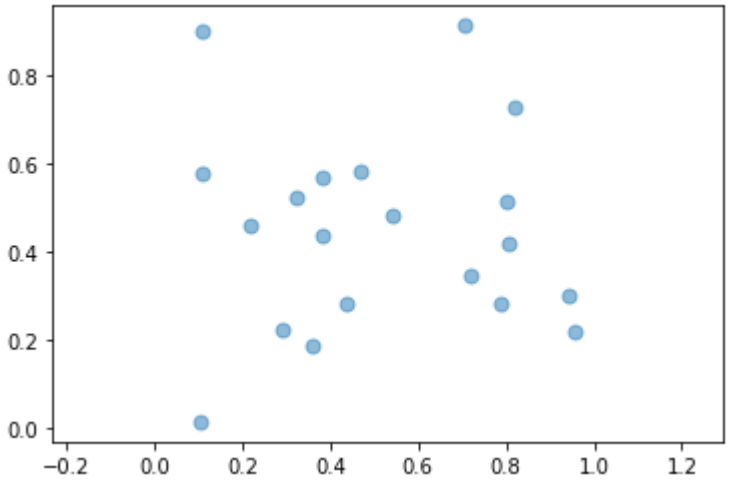
(3,)
(1,3)
(4,3)

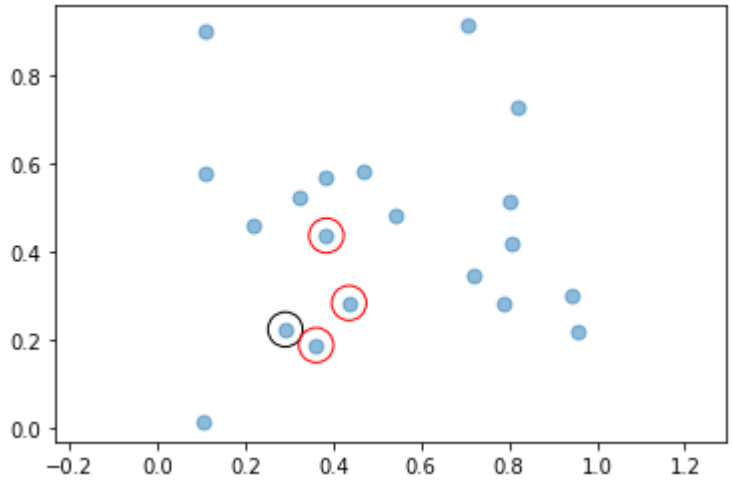
(4,3)

브로드캐스팅 예제: kNN

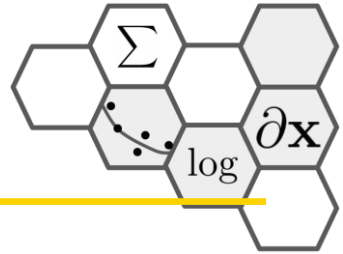



```
X = np.random.rand(20, 2)
fig = plt.figure()
ax = plt.axes()
ax.scatter(X[:,0], X[:,1], s=50, alpha=0.5)
ax.axis('equal')
plt.show()
```





자주쓰는 기능: concatenate



- `concatenate(): np.concatenate((a1,a2,...),axis=0)`
  - Join a sequence of arrays along an existing axis.

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

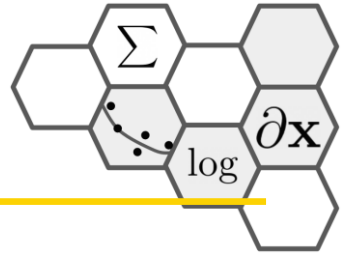
```
np.concatenate((a, b))
```

```
[[1 2]
 [3 4]
 [5 6]]
```

```
np.concatenate((a, b.T), axis=1)
```

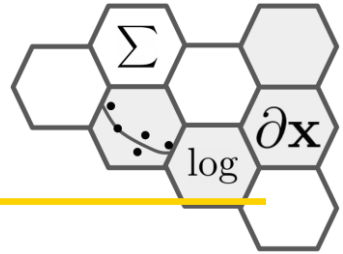
```
[[1 2 5]
 [3 4 6]]
```

자주쓰는 기능: stack



- `stack()`
  - Join a sequence of arrays along a new axis.
- `hstack()`
  - Stack arrays in sequence horizontally (column wise).
- `vstack()`
  - Stack arrays in sequence vertically (row wise).

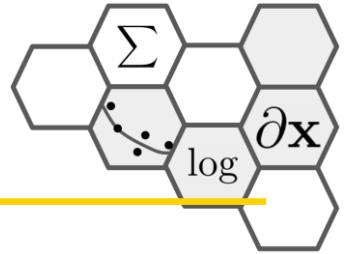
자주쓰는 기능: squeeze



- squeeze()
  - Remove axes of length one from an array.

```
x = np.array([[[0], [1], [2]]])
x.shape  # (1,3,1)

np.squeeze(x).shape             # (3,)
np.squeeze(x, axis=0).shape     # (3,1)
# np.squeeze(x, axis=1).shape   # error
np.squeeze(x, axis=2).shape     # (1,3)
```

자주쓰는 기능: where



- where()
  - Return elements chosen from x or y depending on condition.

```
np.where(condition, [x, y])

a = np.array([[0, 1, 2],
              [0, 2, 4],
              [0, 3, 6]])

np.where(a < 4, a, -1)  # -1 is broadcast
array([[ 0,  1,  2],
       [ 0,  2, -1],
       [ 0,  3, -1]])
```