

파이썬 라이브러리

NumPy

Pandas

Matplotlib

2017010698
수학과 오서영

NumPy

numpy는 수치해석용 파이썬 패키지로 numerical python의 줄임말
 벡터와 행렬을 사용하는 선형대수 계산 사용



행렬 : 직사각형 형태로 수가 배열된 것

행 : 행렬의 가로줄

열 : 행렬의 세로줄

m X n 행렬 : m개의 행과 n개의 열로 이루어진 행렬

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

행 \Rightarrow 2 x 3
 열

```
In [1]: import numpy as np
```

```
In [2]: array = np.array([1, 2, 3])  
print(array.size)  배열 요소 개수  
print(array.dtype)  요소 타입  
print(array[1])
```

```
3  
int32  
2
```

```
In [4]: A = np.arange(5)  0 ~ 4 배열  
print(A)
```

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

```
In [5]: B = np.zeros((4,3))  영행렬  
print(B)
```

```
[[0. 0. 0.]  
 [0. 0. 0.]  
 [0. 0. 0.]  
 [0. 0. 0.]]
```

```
In [6]: C = np.random.randint(0, 10, (3, 3))  0 ~ 9 숫자  
print(C)
```

```
[[2 2 7]  
 [5 2 7]  
 [2 3 2]]
```

```
In [10]: array1 = np.array([1, 2, 3, 4])
print(array1.shape)
```

(4,) ← 1차원 배열 (4개의 요소)

```
In [11]: array2 = array1.reshape((2, 2))
print(array2.shape)
print(array2)
```

(2, 2) ← 2차원 배열 (2행 2열) $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
[[1 2]
[3 4]]

```
In [12]: a = np.array([[1,2],[3,4]])
b = np.array([[10,20],[30,40]])
```

$$a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad b = \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$$

```
In [19]: c = a + b
c
```

$$c = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 11 & 22 \\ 33 & 44 \end{bmatrix}$$

```
Out [19]: array([[11, 22],
                [33, 44]])
```

```
In [21]: d = b - a
d
```

a와 b는
동일한 크기의 행렬

```
Out [21]: array([[ 9, 18],
                [27, 36]])
```

```
In [22]: e = a * b
e
```

```
Out [22]: array([[ 10,  40],
                [ 90, 160]])
```

$$e = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 10 & 40 \\ 90 & 160 \end{bmatrix}$$

요소끼리

```
In [23]: f = a @ b
f
```

```
Out [23]: array([[ 70, 100],
                [150, 220]])
```

행렬 곱

: 행 성분과 열 성분을 곱하여 더함

$$f = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 1 \cdot 10 + 2 \cdot 30 & 1 \cdot 20 + 2 \cdot 40 \\ 3 \cdot 10 + 4 \cdot 30 & 3 \cdot 20 + 4 \cdot 40 \end{bmatrix}$$

```
In [35]: s = np.ones((4,3))
s
```

```
Out [35]: array([[1., 1., 1.],
                [1., 1., 1.],
                [1., 1., 1.],
                [1., 1., 1.]])
```

```
In [36]: t = np.full((2,2),10)
t
```

```
Out [36]: array([[10, 10],
                [10, 10]])
```

```
In [ ]: s @ t
```

$$S_{4 \times \underline{3}} \times t_{\underline{2} \times 2}$$

```
In [38]: t = np.full((3,2),10)
s @ t
```

```
Out [38]: array([[30., 30.],
                [30., 30.],
                [30., 30.],
                [30., 30.]])
```

```
In [39]: np.arange(0,10) 0 ~ 9
```

```
Out [39]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [40]: np.arange(0,10,2) 0 ~ 9 - 2step
```

```
Out [40]: array([0, 2, 4, 6, 8])
```

```
In [41]: np.arange(0,10).reshape(2,5) 2행 5열
```

```
Out [41]: array([[0, 1, 2, 3, 4],  
                [5, 6, 7, 8, 9]])
```

```
In [42]: np.arange(0,10).reshape(3,3)
```

ValueError Traceback (most recent call last)

<ipython-input-42-3e4e3fa4eda0> in <module>

----> 1 np.arange(0,10).reshape(3,3)

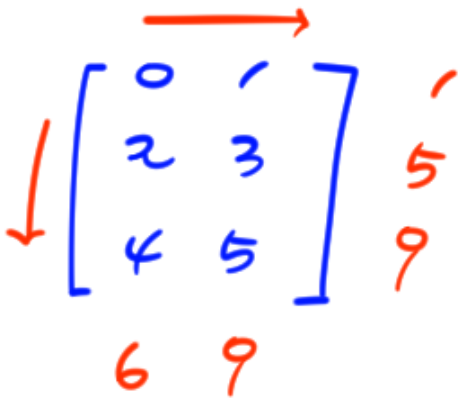
ValueError: cannot reshape array of size 10 into shape (3,3)

In [43]: `np.arange(0,6).reshape(3,2).sum(axis=0)` ↓

Out [43]: `array([6, 9])`

In [44]: `np.arange(0,6).reshape(3,2).sum(axis=1)` →

Out [44]: `array([1, 5, 9])`



In [49]: `a`

Out [49]: `array([[1, 2],
[3, 4]])`

In [54]: `a[0,0]`

Out [54]: `1`

In [55]: `a[1,0]`

Out [55]: `3`

$$\begin{array}{cc} [0,0] & [0,1] \\ \left[\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right] \\ [1,0] & [1,1] \end{array}$$

데이터를 효과적으로 처리하고 보여주는 라이브러리
Numpy와 함께 사용된다
데이터는 Series나 표(table)의 형태

- **Series**
: 인덱스와 값으로 구성

○ : '사과'

ㅅ : '딸기'

```
In [1]: import pandas as pd
```

```
In [6]: a = pd.Series([4, 5, -2, 8])  
print(a)
```

```
0    4  
1    5  
2   -2  
3    8  
dtype: int64
```

```
In [7]: a.values ?k
```

```
Out [7]: array([ 4,  5, -2,  8], dtype=int64)
```

```
In [8]: a.index
```

```
Out [8]: RangeIndex(start=0, stop=4, step=1) 0 ~ 3
```

```
In [12]: array1 = pd.Series( [ '개' , '고양이' , '토끼' ] , index = ['a' , 'b' , 'c'] )  
print(array1)
```

```
a    개  
b   고양이  
c    토끼  
dtype: object
```

```
In [5]: print(array1['b'])
```

```
고양이
```

```
In [9]: print(array1[1])
```

```
고양이
```

```
In [11]: array2 = pd.Series( [ 'dog' , 'cat' , 'rabbit' ] , index = [ 'a' , 'b' , 'c' ] )
```

```
In [13]: summary = pd.DataFrame({  
    'kor' : array1,  
    'eng' : array2  
})  
print(summary)
```

data를 index 기준으로 묶어준다

	kor	eng
a	개	dog
b	고양이	cat
c	토끼	rabbit

index	kor	eng
a	개	dog
b	고양이	cat
c	토끼	rabbit

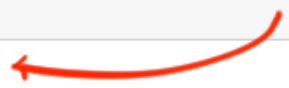
```
In [14]: array3 = pd.Series([ 1, 2, 3 ], index = [ 'a', 'b', 'c' ])
array4 = pd.Series([ 4, 5, 6 ], index = [ 'a', 'b', 'c' ])
summary = pd.DataFrame({
    'kor' : array1,
    'eng' : array2,
    'num1' : array3,
    'num2' : array4
})
print(summary)
```

	kor	eng	num1	num2
a	개	dog	1	4
b	고양이	cat	2	5
c	토끼	rabbit	3	6

```
In [21]: sum = summary[ 'num1' ] + summary[ 'num2' ]
summary[ 'sum' ] = sum
print(summary)
```

행계를 새로운 시리즈로 추가

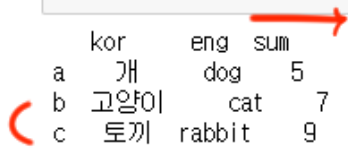
	kor	eng	num1	num2	sum
a	개	dog	1	4	5
b	고양이	cat	2	5	7
c	토끼	rabbit	3	6	9



```
In [54]: del summary['num1']
```

```
In [56]: del summary['num2']
```

```
In [57]: print(summary)
```



	kor	eng	sum
a	개	dog	5
b	고양이	cat	7
c	토끼	rabbit	9

```
In [58]: print( summary.loc[ 'b' : 'c' , 'sum' : ])
```

인덱싱

	sum
b	7
c	9

```
In [59]: print( summary.iloc[ 1:3 , 2 : ] )
```

	sum
b	7
c	9

In [69]:

```
import numpy as np
s = pd.Series(np.random.randint(0,6,10))
print(s)
```

0 ~ 5 44

```
0    0
1    5
2    2
3    5
4    2
5    0
6    5
7    0
8    4
9    1
dtype: int32
```

In [70]:

```
s.sort_values(ascending = True)
```

value를 기준으로
오름차순 정렬

Out [70]:

```
0    0
5    0
7    0
9    1
2    2
4    2
8    4
1    5
3    5
6    5
dtype: int32
```

=

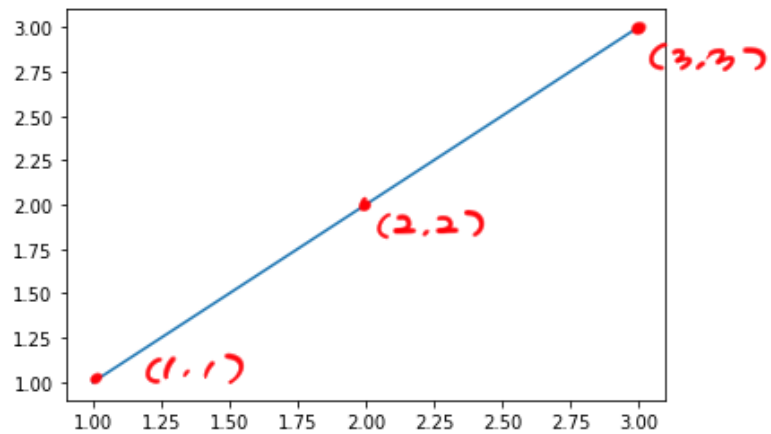
3

Matplotlib

(다양한 데이터를 시각화
그래프 그리기


```
In [1]: import matplotlib.pyplot as plt
```

```
In [3]: x = [1,2,3]  
y = [1,2,3]  
plt.plot(x, y)  
plt.show()
```



In [23]:

```
import numpy as np
```

```
x = np.linspace(0, np.pi * 10, 500)
```

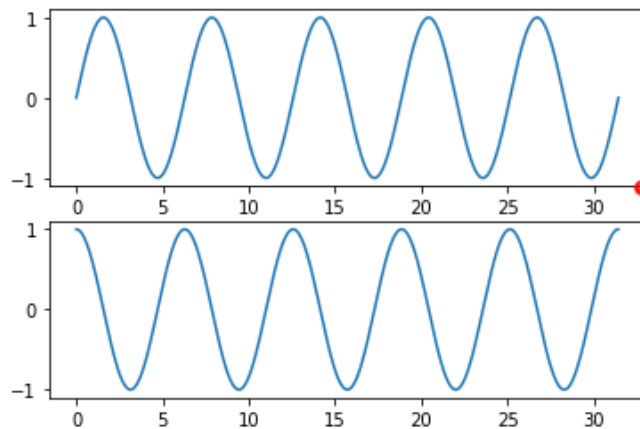
```
fig, axes = plt.subplots(2, 1)
```

```
axes[0].plot(x, np.sin(x))
```

```
axes[1].plot(x, np.cos(x))
```

```
plt.show()
```

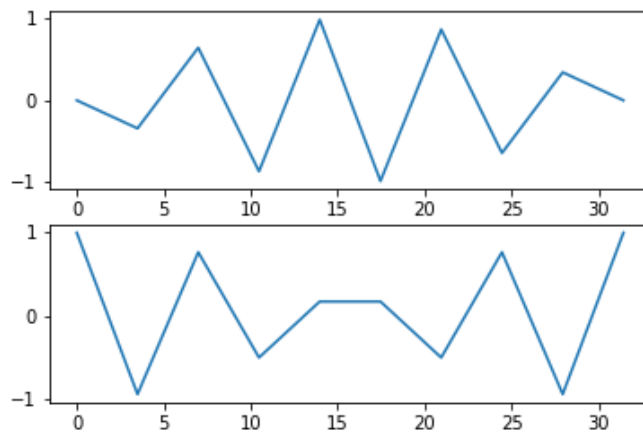
→ fig : 



np.pi * 10

```
In [24]: import numpy as np
```

```
x = np.linspace(0, np.pi * 10, 10)
fig, axes = plt.subplots(2, 1)
axes[0].plot(x, np.sin(x))
axes[1].plot(x, np.cos(x))
plt.show()
```

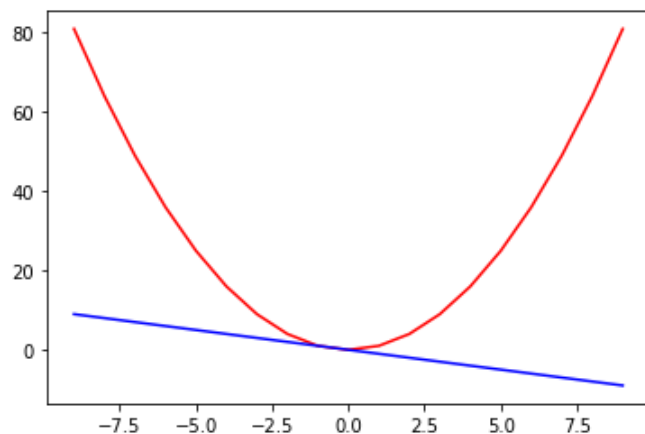


```
In [25]: x
```

```
Out [25]: array([ 0.          ,  3.4906585 ,  6.98131701, 10.47197551, 13.96263402,
        17.45329252, 20.94395102, 24.43460953, 27.92526803, 31.41592654])
```

$\pi \cdot 10$

```
In [28]: x = np.arange(-9,10)
y1 = x**2
y2 = -x
plt.plot(x,y1,color = 'red')
plt.plot(x,y2,color = 'blue')
plt.show()
```



```
In [29]: x
```

```
Out [29]: array([-9, -8, -7, -6, -5, -4, -3, -2, -1,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9])
```

```
In [32]: x = np.arange(-9,10)
plt.bar(x,x**2)
plt.title("My graph")
plt.xlabel("X")
plt.ylabel("X**2")
plt.show()
```

