데이터 형태

변수 : 문자, 숫자

```
val1=3
val2=-6.8
str1='stay hunger'
str2='운명이다'
```

```
val1+val2, str1+' -Steve Jobs-'
```

```
(-3.8, 'stay hunger -Steve Jobs-')
```

list

```
list_val=range(9)
list_str=['wolfpack','hnu','stat']
```

```
list_str[1], list_val[1]
```

```
('hnu', 1)
```

Dictionary{}

```
dict_eg={7933001:'wolfpack',3725:'hnu'}
dict_eg[7933001], len(dict_eg)
```

```
('wolfpack', 2)
```

행렬

```
import numpy as np
A=np.array([[1,2],[3,4],[5,6]])
B=np.array([[1,2,3],[4,5,6]])
v=np.array([[7],[8],[9]])
u=np.array([[10,11,12]])
z=np.array((14,15,16))
print('행렬 A : \n',A,'\n 행렬 B : \n',B,'\n 벡터 v : \n',v,'\n 벡터 u : \n',u,'\n array z : \n',z
```

```
행렬 A :
 [[1 2]
 [3 4]
 [5 6]]
 행렬 B :
 [[1 2 3]
 [4 5 6]]
 벡터 v :
 [[7]
 [8]
 [9]]
 벡터 u :
 [[10 11 12]]
 array z :
 [14 15 16]
```


행렬 인덱스

```
A[:,1], A[0,0],A[:,:2]
```

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

행렬차수

- shape : 행과 열 차수, ndim : array 차원개수 size : 행과 열의 차수

```
A.shape,A.ndim,A.size,v.shape,v.ndim,u.ndim,v.size,z.shape,z.ndim,z.size
```

```
((3, 2), 2, 6, (3, 1), 2, 2, 3, (3,), 1, 3)
```

전치

```
print(A.T,'\n ******* \n',B.T,'\n ******* \n',v.T,'\n ******* \n',z.T)
```

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



```
 *******
 [14 15 16]
```

행렬 쌓기

- 가로 쌓기 : 열의 차수가 같으면 conformable (연산 가능)
- 세로 쌓기 : 행의 차수가 같으면 연산가능

```
print(np.hstack([A,v]),'\n ******* \n',np.vstack([B,v.T]))
```

```
[[1 2 7]
```

```
1  np.hstack([A,u])
```

```
  File "<ipython-input-11-b2ed34491b97>", line 1
    np.hstack([A,u]
                  ^
SyntaxError: unexpected EOF while parsing
```

SEARCH STACK OVERFLOW

더하기/빼기

> 연산가능 conformable : (행렬) 차수 동일

> 단 열벡터인 경우에는 행의 수가 같으면 행의 열마다 더해진다

Saved successfully!

```
array([[ 2,  4],
       [ 6,  8],
       [10, 12]])ERROR! Session/line number was not unique in database. History logging moved to new session 59
```

```
1  A+v
```

```
array([[ 8,  9],
       [11, 12],
       [14, 15]])
```

```
1  A+u
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-c64203dbbe55> in <module>()
----> 1 A+u

ValueError: operands could not be broadcast together with shapes (3,2) (1,3)
```

SEARCH STACK OVERFLOW

곱하기

> 행과 열의 곱으로 내적곱이다. 행렬의 행차수와 열벡터의 차수가 동일해야 한다.

> A@B 행렬의 곱 : A행렬의 열차수와 B 행렬의 행차수가 동일해야 한다. <=> A.dot(B)

Saved successfully!

```
array([[ 7, 11],
       [24, 32],
       [45, 54]])
```

```
1 A@v
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-15-514454d627ab> in <module>()
----> 1 A@v

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,
```

SEARCH STACK OVERFLOW

```
1 B@v
```

```
array([[ 50],
       [122]])
```

```
B.dot(v)
```

```
array([[ 50],
       [122]])
```

행렬식, 역행렬 Ax=b

> x+y+z=1

> 2x+y+2z=4

> 4x+3y-4z=7


```
[4,3,-4]])
b=np.array([1,4,7])
```

```
A
```

```
array([[ 1,  1,  1],
       [ 2,  1,  2],
       [ 4,  3, -4]])
```

** 행렬식 **

```
import numpy.linalg as la
la.det(A)
```

```
7.999999999999998
```

```
la.det(np.array([[1,1,1],[2,1,2],[4,3,4]]))
```

```
0.0
```

랭크 rank

```
la.matrix_rank(A)
```

```
3
```

```
la.matrix_rank(np.array([[1,1,1],[2,2,2]]))
```

```
1
```

역행렬


```
array([[-1.25 ,  0.875, -0.125],
       [ 2.   , -1.   , -0.   ],
       [ 0.25 ,  0.125, -0.125]])
```

** 역행렬(A)@A = 항등행렬

```
la.inv(A)@A
```

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

** 방정식 해 A^(-1)*b

```
la.inv(A).dot(b)  #solution of equation
```

```
array([ 3.125, -2.   , -0.125])
```

```
la.inv(A)@b
```

```
array([ 3.125, -2.   , -0.125])
```

** 방정식 해 확인 **

```
A@la.inv(A)@b
```

```
array([1., 4., 7.])
```

eigen values, vector


```
y([[0,1],[-2,-3]]))
```

```
[-1. -2.]
 [[ 0.70710678 -0.4472136 ]
 [-0.70710678  0.89442719]]
```

SVD singular decomposition

> A=U(Sigma)V', U, V orthogonal matrix

```
U,Sigma,Vt=la.svd(np.array([[0,1],[-2,-3]]))
print(U,'\n',Sigma,'\n',Vt)
```

Special Matrix

```
1  np.zeros((2,3))
```

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

```
1  np.ones((2,3))
```

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

```
1  np.eye(3)
```

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

Saved successfully!

```
1  np.full((2,3),3)
```

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

```
1  np.random.random((2,3))
```

```
array([[0.859799  , 0.37413985, 0.28712814],
       [0.02358198, 0.31388947, 0.17501232]])
```

Mean and Vriance

1. one' vector_x /n
2. (vector_x - vector_mu)'(vector_x - vector_mu) / (n-1)

```
1  import numpy as np
```

```
import numpy as np
x=np.array(range(0,10))
print('mean=',x.mean(),'var=',x.var())
```

```
mean= 4.5 var= 8.25
```

```
xbar=np.ones(x.shape)@x.T/x.shape
xbar
```

```
array([4.5])
```

```
var_x=(x.T-np.full(x.shape,xbar))@(x.T-np.full(x.shape,xbar)).T / x.shape
var_x
```

```
array([8.25])
```

Saved successfully!