In [4]:

```
a=[1,3,5]
b=[2,4,6]
c= a+b
c
```

Out[4]:

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

리스트 안에서는 수학적 계산 x

In [6]:

```
import numpy
```

In [7]:

```
A = numpy.array (a)
B= numpy.array (b)
```

```
array makes calculation possible but numpy 먼저 import 해야함
```

In [8]:

```
A+B
```

Out[8]:

```
array([ 3,  7, 11])
```

In [9]:

```
type (A)
```

Out[9]:

```
numpy.ndarray
```

In [10]:

```
import numpy as np
```

In [11]:

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

Out[11]:

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

In [12]:

```
X.shape
```

Out[12]:

```
(2, 3)
```

shape = 2 by 3 matrix

In [ ]:

```

```

In [ ]:

```

```

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import pyplot as plt is the same this as above
```

In [1]:

```
np.empty([2,3], dtype='int')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-3d3be3b5e906> in <module>
----> 1 np.empty([2,3], dtype='int')

NameError: name 'np' is not defined
```

dtype = data type

In [7]:

```
np.zeros([2,3])
```

Out[7]:

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

2 by 3의 matrix을 0으로 채움

In [ ]:

```
[[0,0,0],[0,0,0]]
```

계산을 못하니까 쓸모가 없음

In [11]:

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

Out[11]:

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

array 사용

In [12]:

```
np.ones([2,3], dtype = 'float64')
```

Out[12]:

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

ones 라는 함수 float 64는 64 소수자리까지 (정확하지만 데이터 많이 차지)

In [13]:

```
np.ones([2,3], dtype = 'int')
```

Out[13]:

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

In [14]:

```
np.arange(5)
```

Out[14]:

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

float loop 때 range랑 비슷함

In [15]:

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

Out[15]:

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

In [17]:

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

Out[17]:

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

0부터 10까지 2개 차이로

In [16]:

```
np.arange(0,10,2, dtype='float64')
```

Out[16]:

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

In [18]:

```
np.linspace (0,10,6)
```

Out[18]:

```
array([ 0.,  2.,  4.,  6.,  8., 10.])
```

0부터 10까지 6개로 나눔

In [20]:

```
x= np.array ([[1,2],[4,5],[8,9]])
x
```

Out[20]:

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

대괄호 2개 = 2차원 대괄호 3개 = 3차원

In [25]:

```
x= np.array ([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])
x
```

Out[25]:

```
array([[[1, 2],
        [4, 5],
        [8, 9]],

       [[1, 2],
        [4, 5],
        [8, 9]]])
```

In [26]:

```
x. ndim
```

Out[26]:

```
3
```

몇차원인지 알려줌

In [27]:

```
x.shape
```

Out[27]:

```
(2, 3, 2)
```

In [28]:

```
x.dtype
```

Out[28]:

```
dtype('int32')
```

In [29]:

```
x.astype (np.float64)
```

Out[29]:

```
array([[[1., 2.],
        [4., 5.],
        [8., 9.]],

       [[1., 2.],
        [4., 5.],
        [8., 9.]]])
```

In [30]:

```
np.zeros_like(x)
```

Out[30]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0],
        [0, 0]]])
```

생긴건 똑같이 숫자를 0으로 바꿈

In [31]:

```
x*0
```

Out[31]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0],
        [0, 0]]])
```

In [32]:

```
data = np.random.normal(0,1, 100)
print(data)
```

```
[ 0.5948894   0.62901742 -2.43481232 -0.64520163 -2.25552614  0.19489435
 -0.78682143 -1.25504291 -0.27158939 -0.58122381 -0.3975076   0.19090601
 -0.7051918   0.85886849 -0.1332941   1.01240101 -0.37015608 -0.3629753
  0.62532226 -0.55471334  0.51391598 -0.46922885 -0.13815178 -0.95075819
  0.28241838  0.76452934  1.1830317  -1.68797608 -0.27279998  1.34480936
  0.38829424  1.41437356  0.21335677  1.51224885  0.24974161 -0.25661151
  0.16083075 -0.21285222 -0.06453529  0.28284785  0.19434011 -0.58968835
 -1.84297084  1.38741169 -0.0083715  -0.50905944  1.47782995 -1.34516351
  0.77349661  1.09383414  1.09127973 -1.11432754  0.94883338 -0.23355717
 -0.39135684 -1.81958121  0.23487778  0.82448781 -0.87749371 -0.73695733
 -0.12413642  2.71239267  0.89010278  1.174496   -0.66606796  0.00658702
 -0.50367648 -1.21744002 -0.59010186 -0.58975187 -2.62121707  1.21332786
  0.25443795 -0.64268828  2.12788179  0.94568845  1.24823598 -1.05314019
  1.36540336  0.22325166 -0.4619426   0.78764955 -1.13887395 -0.57905073
  0.46988864  0.44221884 -0.80193795  1.03495679  2.23371987 -0.58567336
 -0.06988392  0.24613168 -1.43981342 -2.03215962  0.95582361 -0.27585455
  0.34924642 -0.40700961  0.14194833  0.0361661 ]
```

normal = normal distribution 만들어줌 0 = mean 1= standard deviation 100은 100개의 data 만들어라

In [34]:

```
data.ndim
```

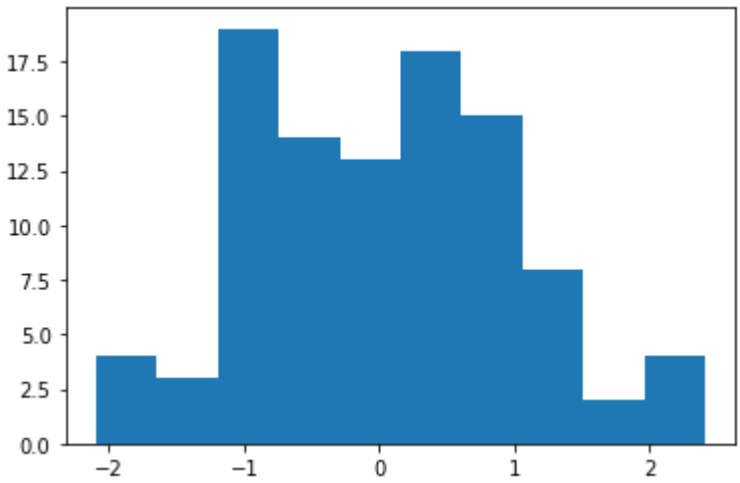
Out[34]:

1

In [35]:

```
data = np.random.normal(0,1, 100)
print(data)
plt.hist(data, bins=10)
plt.show()
```

```
[-0.29052981  0.3142496  -0.30092923  0.23294338 -0.0096937  -1.75264288
 -1.09379186  1.22418643  0.09211239  0.65077238 -0.18112944  0.59968089
  1.02008837  2.41507312  0.24426123 -0.61227634  2.210503   -0.42432102
  0.41848103  0.17453703 -0.86170595  1.58677576  0.8008617  -2.09665586
  1.36270504 -0.7862358   1.06420937 -0.05100987 -0.14991648 -0.75866774
 -1.12325    -0.86547804 -0.66849136 -0.75090565  0.71276998  0.17085606
  1.82536286 -1.32038878  0.75468933  0.27739382  1.15321934 -0.72314527
  1.08439659  1.48315776  2.13391077 -0.49218686  0.43490225 -0.09114682
 -0.80881791  0.80532131  0.73983968 -0.45784457 -1.62985832  0.38751317
 -0.53320662 -0.80517678  0.26518977 -1.11002782 -0.67592721  0.15333034
 -0.95616868  0.51447594  1.12573406  0.45444624 -0.85809141  0.84831613
  2.33190843  0.03833018 -1.08656145  0.62538228 -0.50315794  0.6696119
  0.54005942 -1.01313403  0.82195499 -0.77879309 -0.3755143  -0.97419932
  0.05934476  0.54322135  0.29534863 -0.2355785   0.11673703 -0.93300181
  1.06537317 -0.64259085 -0.63545322 -0.98052767 -1.55429745  0.81891179
  0.67071758 -2.04890039  0.70647498  0.68675609 -0.87092778 -0.65607294
  0.51223153 -0.04768571 -1.71196897  0.57827161]
```



bins =10 은 graph 에서 10개로 나눔 위에 graph의 값을 다 더하면 100개

In [37]:

```
X= np.ones ([2,3,4])
X
```

Out[37]:

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

       [[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]]])
```

2,3,4 = 3차원 2,3,4,5 = 4차원

In [38]:

```
Y = X.reshape (-1,3,2)
Y
```

Out[38]:

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

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]]])
```

reshape = shape바꾸는 것 -1은 뭔지 모르겠다 너가 알아서 해라 but 4써도 똑같이 나옴

In [42]:

```
np.allclose(X.reshape(-1,3,2),Y)
```

Out[42]:

```
True
```

X reshape 한거랑 y가 똑같은가

In [47]:

```
a = np.random.randint(0, 10, [2, 3])
b = np.random.random([2, 3])
np.savez("test", a, b)
```

randint = 0~10에서 숫자 골라서 2 3 matrix 만듬 np.savez 파일로 저장

In [43]:

```
ls -al test*
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: AE98-5A30

 C:₩Users₩ykim3₩OneDrive₩문서₩영어음성학 디렉터리


 C:₩Users₩ykim3₩OneDrive₩문서₩영어음성학 디렉터리

파일을 찾을 수 없습니다.
```

In [48]:

```
who
```

```
X       Y       a       b       data    np      plt     x
```

who 는 지금 어떤 variable 이 available 한지

In [54]:

```
del a,b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-54-a9b92be7739b> in <module>
----> 1 del a,b

NameError: name 'a' is not defined
```

In [55]:

```
who
```

```
X       Y       data    np      plt     x
```

In [56]:

```
npzfiles = np.load("test.npz")
npzfiles.files
```

Out[56]:

```
['arr_0', 'arr_1']
```

In [57]:

```
npzfiles['arr_0']
```

Out[57]:

```
array([[1, 6, 6],
       [2, 2, 8]])
```

npzfiles은 저장된 파일불러오기

skiprows 첫번째 row skip because it is title

In [58]:

```
arr= np.random.random([5,2,3])
```

In [59]:

```
print(type(arr))
print(len(arr))
print(arr.shape)
print(arr.ndim)
print(arr.size)
print(arr.dtype)
```

```
<class 'numpy.ndarray'>
5
(5, 2, 3)
3
30
float64
```

In [62]:

```
a=np.arange (1,5)
b=np.arange (9,5,-1)
```

In [63]:

```
print (a-b)
print (a*b)
```

```
[-8 -6 -4 -2]
[ 9 16 21 24]
```

In [64]:

```
a = np.arange(1, 10).reshape(3,3)
b = np.arange(9, 0, -1).reshape(3,3)
print(a)
print(b)
```

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

In [65]:

```
a==b
```

Out[65]:

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

In [66]:

```
a>b
```

Out[66]:

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

비교할때 dimention이랑 shape 같아야함

In [67]:

```
a
```

Out[67]:

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

In [69]:

```
a.sum()
```

Out[69]:

45

In [70]:

```
np.sum(a)
```

Out[70]:

45

In [71]:

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

Out[71]:

```
array([12, 15, 18])
```

axis = 몇번째 차원에서 실행

In [72]:

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

Out[72]:

```
array([ 6, 15, 24])
```

In [73]:

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

Out[73]:

```
array([ 6, 15, 24])
```

In [74]:

```
a = np.arange(1, 25).reshape(4, 6)
a
```

Out[74]:

```
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]])
```

In [75]:

```
a+100
```

Out[75]:

```
array([[101, 102, 103, 104, 105, 106],
       [107, 108, 109, 110, 111, 112],
       [113, 114, 115, 116, 117, 118],
       [119, 120, 121, 122, 123, 124]])
```

In [76]:

```
b= np.arange(6)
b
```

Out[76]:

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

In [77]:

```
a+b
```

Out[77]:

```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```

In [ ]:

```

```

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import pyplot as plt is the same this as above
```

In [6]:

```
np.empty([2,3], dtype='int')
```

Out[6]:

```
array([[-1662389008,          367,            0],
       [          0,       131074,    538970682]])
```

dtype = data type

In [7]:

```
np.zeros([2,3])
```

Out[7]:

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

2 by 3의 matrix을 0으로 채움

In [ ]:

```
[[0,0,0],[0,0,0]]
```

계산을 못하니까 쓸모가 없음

In [11]:

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

Out[11]:

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

array 사용

In [12]:

```
np.ones([2,3], dtype = 'float64')
```

Out[12]:

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

ones 라는 함수 float 64는 64 소수자리까지 (정확하지만 데이터 많이 차지)

In [13]:

```
np.ones([2,3], dtype = 'int')
```

Out[13]:

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

In [14]:

```
np.arange(5)
```

Out[14]:

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

float loop 때 range랑 비슷함

In [15]:

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

Out[15]:

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

In [17]:

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

Out[17]:

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

0부터 10까지 2개 차이로

In [16]:

```
np.arange(0,10,2, dtype='float64')
```

Out[16]:

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

In [18]:

```
np.linspace (0,10,6)
```

Out[18]:

```
array([ 0.,  2.,  4.,  6.,  8., 10.])
```

0부터 10까지 6개로 나눔

In [20]:

```
x= np.array ([[1,2],[4,5],[8,9]])
x
```

Out[20]:

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

대괄호 2개 = 2차원 대괄호 3개 = 3차원

In [25]:

```
x= np.array ([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])
x
```

Out[25]:

```
array([[[1, 2],
        [4, 5],
        [8, 9]],

       [[1, 2],
        [4, 5],
        [8, 9]]])
```

In [26]:

```
x. ndim
```

Out[26]:

```
3
```

몇차원인지 알려줌

In [27]:

```
x.shape
```

Out[27]:

```
(2, 3, 2)
```

In [28]:

```
x.dtype
```

Out[28]:

```
dtype('int32')
```

In [29]:

```
x.astype (np.float64)
```

Out[29]:

```
array([[[1., 2.],
        [4., 5.],
        [8., 9.]],

       [[1., 2.],
        [4., 5.],
        [8., 9.]]])
```

In [30]:

```
np.zeros_like(x)
```

Out[30]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0],
        [0, 0]]])
```

생긴건 똑같이 숫자를 0으로 바꿈

In [31]:

```
x*0
```

Out[31]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0],
        [0, 0]]])
```

In [32]:

```
data = np.random.normal(0,1, 100)
print(data)
```

```
[ 0.5948894   0.62901742 -2.43481232 -0.64520163 -2.25552614  0.19489435
 -0.78682143 -1.25504291 -0.27158939 -0.58122381 -0.3975076   0.19090601
 -0.7051918   0.85886849 -0.1332941   1.01240101 -0.37015608 -0.3629753
  0.62532226 -0.55471334  0.51391598 -0.46922885 -0.13815178 -0.95075819
  0.28241838  0.76452934  1.1830317  -1.68797608 -0.27279998  1.34480936
  0.38829424  1.41437356  0.21335677  1.51224885  0.24974161 -0.25661151
  0.16083075 -0.21285222 -0.06453529  0.28284785  0.19434011 -0.58968835
 -1.84297084  1.38741169 -0.0083715  -0.50905944  1.47782995 -1.34516351
  0.77349661  1.09383414  1.09127973 -1.11432754  0.94883338 -0.23355717
 -0.39135684 -1.81958121  0.23487778  0.82448781 -0.87749371 -0.73695733
 -0.12413642  2.71239267  0.89010278  1.174496   -0.66606796  0.00658702
 -0.50367648 -1.21744002 -0.59010186 -0.58975187 -2.62121707  1.21332786
  0.25443795 -0.64268828  2.12788179  0.94568845  1.24823598 -1.05314019
  1.36540336  0.22325166 -0.4619426   0.78764955 -1.13887395 -0.57905073
  0.46988864  0.44221884 -0.80193795  1.03495679  2.23371987 -0.58567336
 -0.06988392  0.24613168 -1.43981342 -2.03215962  0.95582361 -0.27585455
  0.34924642 -0.40700961  0.14194833  0.0361661 ]
```

normal = normal distribution 만들어줌 0 = mean 1= standard deviation 100은 100개의 data 만들어라

In [34]:

```
data.ndim
```

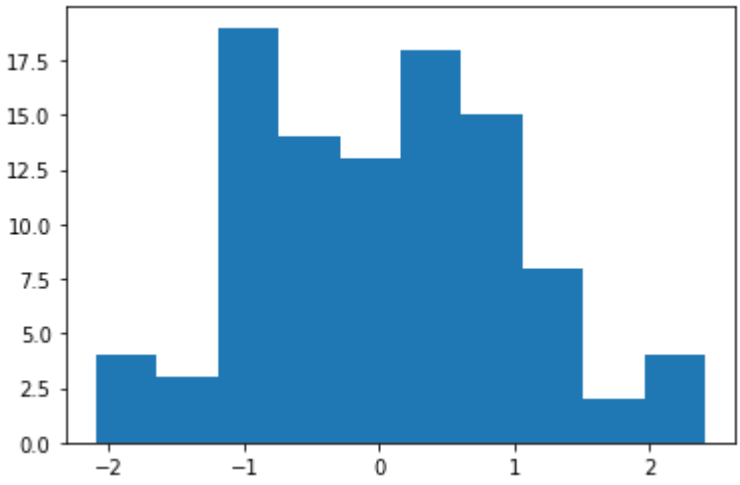
Out[34]:

```
1
```

In [35]:

```
data = np.random.normal(0,1, 100)
print(data)
plt.hist(data, bins=10)
plt.show()
```

```
[-0.29052981  0.3142496  -0.30092923  0.23294338 -0.0096937  -1.75264288
 -1.09379186  1.22418643  0.09211239  0.65077238 -0.18112944  0.59968089
  1.02008837  2.41507312  0.24426123 -0.61227634  2.210503   -0.42432102
  0.41848103  0.17453703 -0.86170595  1.58677576  0.8008617  -2.09665586
  1.36270504 -0.7862358   1.06420937 -0.05100987 -0.14991648 -0.75866774
 -1.12325    -0.86547804 -0.66849136 -0.75090565  0.71276998  0.17085606
  1.82536286 -1.32038878  0.75468933  0.27739382  1.15321934 -0.72314527
  1.08439659  1.48315776  2.13391077 -0.49218686  0.43490225 -0.09114682
 -0.80881791  0.80532131  0.73983968 -0.45784457 -1.62985832  0.38751317
 -0.53320662 -0.80517678  0.26518977 -1.11002782 -0.67592721  0.15333034
 -0.95616868  0.51447594  1.12573406  0.45444624 -0.85809141  0.84831613
  2.33190843  0.03833018 -1.08656145  0.62538228 -0.50315794  0.6696119
  0.54005942 -1.01313403  0.82195499 -0.77879309 -0.3755143  -0.97419932
  0.05934476  0.54322135  0.29534863 -0.2355785   0.11673703 -0.93300181
  1.06537317 -0.64259085 -0.63545322 -0.98052767 -1.55429745  0.81891179
  0.67071758 -2.04890039  0.70647498  0.68675609 -0.87092778 -0.65607294
  0.51223153 -0.04768571 -1.71196897  0.57827161]
```



bins =10 은 graph 에서 10개로 나눔 위에 graph의 값을 다 더하면 100개

In [37]:

```
X= np.ones ([2,3,4])
X
```

Out[37]:

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

       [[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]]])
```

2,3,4 = 3차원 2,3,4,5 = 4차원

In [38]:

```
Y = X.reshape (-1,3,2)
Y
```

Out[38]:

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

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]]])
```

reshape = shape바꾸는 것 -1은 뭔지 모르겠다 너가 알아서 해라 but 4써도 똑같이 나옴

In [42]:

```
np.allclose(X.reshape(-1,3,2),Y)
```

Out[42]:

```
True
```

X reshape 한거랑 y가 똑같은가

In [47]:

```
a = np.random.randint(0, 10, [2, 3])
b = np.random.random([2, 3])
np.savez("test", a, b)
```

randint = 0~10에서 숫자 골라서 2 3 matrix 만듬 np.savez 파일로 저장

In [43]:

```
ls -al test*
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: AE98-5A30

 C:₩Users₩ykim3₩OneDrive₩문서₩영어음성학 디렉터리


 C:₩Users₩ykim3₩OneDrive₩문서₩영어음성학 디렉터리

파일을 찾을 수 없습니다.
```

In [48]:

```
who
```

```
X       Y       a       b       data    np      plt     x
```

who 는 지금 어떤 variable 이 available 한지

In [54]:

```
del a,b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-54-a9b92be7739b> in <module>
----> 1 del a,b

NameError: name 'a' is not defined
```

In [55]:

```
who
```

```
X       Y       data    np      plt     x
```

In [56]:

```
npzfiles = np.load("test.npz")
npzfiles.files
```

Out[56]:

```
['arr_0', 'arr_1']
```

In [57]:

```
npzfiles['arr_0']
```

Out[57]:

```
array([[1, 6, 6],
       [2, 2, 8]])
```

npzfiles은 저장된 파일불러오기

skiprows 첫번째 row skip because it is title

In [58]:

```
arr= np.random.random([5,2,3])
```

In [59]:

```
print(type(arr))
print(len(arr))
print(arr.shape)
print(arr.ndim)
print(arr.size)
print(arr.dtype)
```

```
<class 'numpy.ndarray'>
5
(5, 2, 3)
3
30
float64
```

In [62]:

```
a=np.arange (1,5)
b=np.arange (9,5,-1)
```

In [63]:

```
print (a-b)
print (a*b)
```

```
[-8 -6 -4 -2]
[ 9 16 21 24]
```

In [64]:

```
a = np.arange(1, 10).reshape(3,3)
b = np.arange(9, 0, -1).reshape(3,3)
print(a)
print(b)
```

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

In [65]:

```
a==b
```

Out[65]:

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

In [66]:

```
a>b
```

Out[66]:

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

비교할때 dimention이랑 shape 같아야함

In [67]:

```
a
```

Out[67]:

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

In [69]:

```
a.sum()
```

Out[69]:

45

In [70]:

```
np.sum(a)
```

Out[70]:

45

In [71]:

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

Out[71]:

```
array([12, 15, 18])
```

axis = 몇번째 차원에서 실행

In [72]:

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

Out[72]:

```
array([ 6, 15, 24])
```

In [73]:

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

Out[73]:

```
array([ 6, 15, 24])
```

In [74]:

```
a = np.arange(1, 25).reshape(4, 6)
a
```

Out[74]:

```
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]])
```

In [75]:

```
a+100
```

Out[75]:

```
array([[101, 102, 103, 104, 105, 106],
       [107, 108, 109, 110, 111, 112],
       [113, 114, 115, 116, 117, 118],
       [119, 120, 121, 122, 123, 124]])
```

In [76]:

```
b= np.arange(6)
b
```

Out[76]:

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

In [77]:

```
a+b
```

Out[77]:

```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```

In [ ]:

```

```

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import pyplot as plt is the same this as above
```

In [6]:

```
np.empty([2,3], dtype='int')
```

Out[6]:

```
array([[-1662389008,          367,            0],
       [          0,       131074,    538970682]])
```

dtype = data type

In [7]:

```
np.zeros([2,3])
```

Out[7]:

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

2 by 3의 matrix을 0으로 채움

In [ ]:

```
[[0,0,0],[0,0,0]]
```

계산을 못하니까 쓸모가 없음

In [11]:

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

Out[11]:

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

array 사용

In [12]:

```
np.ones([2,3], dtype = 'float64')
```

Out[12]:

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

ones 라는 함수 float 64는 64 소수자리까지 (정확하지만 데이터 많이 차지)

In [13]:

```
np.ones([2,3], dtype = 'int')
```

Out[13]:

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

In [14]:

```
np.arange(5)
```

Out[14]:

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

float loop 때 range랑 비슷함

In [15]:

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

Out[15]:

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

In [17]:

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

Out[17]:

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

0부터 10까지 2개 차이로

In [16]:

```
np.arange(0,10,2, dtype='float64')
```

Out[16]:

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

In [18]:

```
np.linspace (0,10,6)
```

Out[18]:

```
array([ 0.,  2.,  4.,  6.,  8., 10.])
```

0부터 10까지 6개로 나눔

In [20]:

```
x= np.array ([[1,2],[4,5],[8,9]])
x
```

Out[20]:

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

대괄호 2개 = 2차원 대괄호 3개 = 3차원

In [25]:

```
x= np.array ([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])
x
```

Out[25]:

```
array([[[1, 2],
        [4, 5],
        [8, 9]],

       [[1, 2],
        [4, 5],
        [8, 9]]])
```

In [26]:

```
x. ndim
```

Out[26]:

```
3
```

몇차원인지 알려줌

In [27]:

```
x.shape
```

Out[27]:

```
(2, 3, 2)
```

In [28]:

```
x.dtype
```

Out[28]:

```
dtype('int32')
```

In [29]:

```
x.astype (np.float64)
```

Out[29]:

```
array([[[1., 2.],
        [4., 5.],
        [8., 9.]],

       [[1., 2.],
        [4., 5.],
        [8., 9.]]])
```

In [30]:

```
np.zeros_like(x)
```

Out[30]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0],
        [0, 0]]])
```

생긴건 똑같이 숫자를 0으로 바꿈

In [31]:

```
x*0
```

Out[31]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0],
        [0, 0]]])
```

In [32]:

```
data = np.random.normal(0,1, 100)
print(data)
```

```
[ 0.5948894   0.62901742 -2.43481232 -0.64520163 -2.25552614  0.19489435
 -0.78682143 -1.25504291 -0.27158939 -0.58122381 -0.3975076   0.19090601
 -0.7051918   0.85886849 -0.1332941   1.01240101 -0.37015608 -0.3629753
  0.62532226 -0.55471334  0.51391598 -0.46922885 -0.13815178 -0.95075819
  0.28241838  0.76452934  1.1830317  -1.68797608 -0.27279998  1.34480936
  0.38829424  1.41437356  0.21335677  1.51224885  0.24974161 -0.25661151
  0.16083075 -0.21285222 -0.06453529  0.28284785  0.19434011 -0.58968835
 -1.84297084  1.38741169 -0.0083715  -0.50905944  1.47782995 -1.34516351
  0.77349661  1.09383414  1.09127973 -1.11432754  0.94883338 -0.23355717
 -0.39135684 -1.81958121  0.23487778  0.82448781 -0.87749371 -0.73695733
 -0.12413642  2.71239267  0.89010278  1.174496   -0.66606796  0.00658702
 -0.50367648 -1.21744002 -0.59010186 -0.58975187 -2.62121707  1.21332786
  0.25443795 -0.64268828  2.12788179  0.94568845  1.24823598 -1.05314019
  1.36540336  0.22325166 -0.4619426   0.78764955 -1.13887395 -0.57905073
  0.46988864  0.44221884 -0.80193795  1.03495679  2.23371987 -0.58567336
 -0.06988392  0.24613168 -1.43981342 -2.03215962  0.95582361 -0.27585455
  0.34924642 -0.40700961  0.14194833  0.0361661 ]
```

normal = normal distribution 만들어줌 0 = mean 1= standard deviation 100은 100개의 data 만들어라

In [34]:

```
data.ndim
```

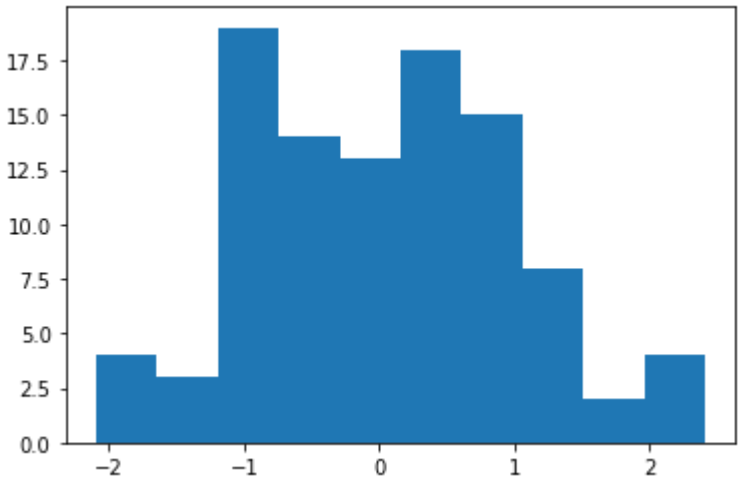
Out[34]:

```
1
```

In [35]:

```
data = np.random.normal(0,1, 100)
print(data)
plt.hist(data, bins=10)
plt.show()
```

```
[-0.29052981  0.3142496  -0.30092923  0.23294338 -0.0096937  -1.75264288
 -1.09379186  1.22418643  0.09211239  0.65077238 -0.18112944  0.59968089
  1.02008837  2.41507312  0.24426123 -0.61227634  2.210503   -0.42432102
  0.41848103  0.17453703 -0.86170595  1.58677576  0.8008617  -2.09665586
  1.36270504 -0.7862358   1.06420937 -0.05100987 -0.14991648 -0.75866774
 -1.12325    -0.86547804 -0.66849136 -0.75090565  0.71276998  0.17085606
  1.82536286 -1.32038878  0.75468933  0.27739382  1.15321934 -0.72314527
  1.08439659  1.48315776  2.13391077 -0.49218686  0.43490225 -0.09114682
 -0.80881791  0.80532131  0.73983968 -0.45784457 -1.62985832  0.38751317
 -0.53320662 -0.80517678  0.26518977 -1.11002782 -0.67592721  0.15333034
 -0.95616868  0.51447594  1.12573406  0.45444624 -0.85809141  0.84831613
  2.33190843  0.03833018 -1.08656145  0.62538228 -0.50315794  0.6696119
  0.54005942 -1.01313403  0.82195499 -0.77879309 -0.3755143  -0.97419932
  0.05934476  0.54322135  0.29534863 -0.2355785   0.11673703 -0.93300181
  1.06537317 -0.64259085 -0.63545322 -0.98052767 -1.55429745  0.81891179
  0.67071758 -2.04890039  0.70647498  0.68675609 -0.87092778 -0.65607294
  0.51223153 -0.04768571 -1.71196897  0.57827161]
```



bins =10 은 graph 에서 10개로 나눔 위에 graph의 값을 다 더하면 100개

In [37]:

```
X= np.ones ([2,3,4])
X
```

Out[37]:

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

       [[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]]])
```

2,3,4 = 3차원 2,3,4,5 = 4차원

In [38]:

```
Y = X.reshape (-1,3,2)
Y
```

Out[38]:

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

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]]])
```

reshape = shape바꾸는 것 -1은 뭔지 모르겠다 너가 알아서 해라 but 4써도 똑같이 나옴

In [42]:

```
np.allclose(X.reshape(-1,3,2),Y)
```

Out[42]:

```
True
```

X reshape 한거랑 y가 똑같은가

In [47]:

```
a = np.random.randint(0, 10, [2, 3])
b = np.random.random([2, 3])
np.savez("test", a, b)
```

randint = 0~10에서 숫자 골라서 2 3 matrix 만듬 np.savez 파일로 저장

In [43]:

```
ls -al test*
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: AE98-5A30

 C:₩Users₩ykim3₩OneDrive₩문서₩영어음성학 디렉터리


 C:₩Users₩ykim3₩OneDrive₩문서₩영어음성학 디렉터리

파일을 찾을 수 없습니다.
```

In [48]:

```
who
```

```
X       Y       a       b       data    np      plt     x
```

who 는 지금 어떤 variable 이 available 한지

In [54]:

```
del a,b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-54-a9b92be7739b> in <module>
----> 1 del a,b

NameError: name 'a' is not defined
```

In [55]:

```
who
```

```
X       Y       data    np      plt     x
```

In [56]:

```
npzfiles = np.load("test.npz")
npzfiles.files
```

Out[56]:

```
['arr_0', 'arr_1']
```

In [57]:

```
npzfiles['arr_0']
```

Out[57]:

```
array([[1, 6, 6],
       [2, 2, 8]])
```

npzfiles은 저장된 파일불러오기

skiprows 첫번째 row skip because it is title

In [58]:

```
arr= np.random.random([5,2,3])
```

In [59]:

```
print(type(arr))
print(len(arr))
print(arr.shape)
print(arr.ndim)
print(arr.size)
print(arr.dtype)
```

```
<class 'numpy.ndarray'>
5
(5, 2, 3)
3
30
float64
```

In [62]:

```
a=np.arange (1,5)
b=np.arange (9,5,-1)
```

In [63]:

```
print (a-b)
print (a*b)
```

```
[-8 -6 -4 -2]
[ 9 16 21 24]
```

In [64]:

```
a = np.arange(1, 10).reshape(3,3)
b = np.arange(9, 0, -1).reshape(3,3)
print(a)
print(b)
```

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

In [65]:

```
a==b
```

Out[65]:

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

In [66]:

```
a>b
```

Out[66]:

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

비교할때 dimention이랑 shape 같아야함

In [67]:

```
a
```

Out[67]:

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

In [69]:

```
a.sum()
```

Out[69]:

45

In [70]:

```
np.sum(a)
```

Out[70]:

45

In [71]:

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

Out[71]:

```
array([12, 15, 18])
```

axis = 몇번째 차원에서 실행

In [72]:

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

Out[72]:

```
array([ 6, 15, 24])
```

In [73]:

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

Out[73]:

```
array([ 6, 15, 24])
```

In [74]:

```
a = np.arange(1, 25).reshape(4, 6)
a
```

Out[74]:

```
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]])
```

In [75]:

```
a+100
```

Out[75]:

```
array([[101, 102, 103, 104, 105, 106],
       [107, 108, 109, 110, 111, 112],
       [113, 114, 115, 116, 117, 118],
       [119, 120, 121, 122, 123, 124]])
```

In [76]:

```
b= np.arange(6)
b
```

Out[76]:

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

In [77]:

```
a+b
```

Out[77]:

```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```

In [ ]:

```

```

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import pyplot as plt is the same this as above
```

In [6]:

```
np.empty([2,3], dtype='int')
```

Out[6]:

```
array([[-1662389008,          367,            0],
       [          0,       131074,    538970682]])
```

dtype = data type

In [7]:

```
np.zeros([2,3])
```

Out[7]:

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

2 by 3의 matrix을 0으로 채움

In [ ]:

```
[[0,0,0],[0,0,0]]
```

계산을 못하니까 쓸모가 없음

In [11]:

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

Out[11]:

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

array 사용

In [12]:

```
np.ones([2,3], dtype = 'float64')
```

Out[12]:

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

ones 라는 함수 float 64는 64 소수자리까지 (정확하지만 데이터 많이 차지)

In [13]:

```
np.ones([2,3], dtype = 'int')
```

Out[13]:

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

In [14]:

```
np.arange(5)
```

Out[14]:

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

float loop 때 range랑 비슷함

In [15]:

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

Out[15]:

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

In [17]:

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

Out[17]:

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

0부터 10까지 2개 차이로

In [16]:

```
np.arange(0,10,2, dtype='float64')
```

Out[16]:

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

In [18]:

```
np.linspace (0,10,6)
```

Out[18]:

```
array([ 0.,  2.,  4.,  6.,  8., 10.])
```

0부터 10까지 6개로 나눔

In [20]:

```
x= np.array ([[1,2],[4,5],[8,9]])
x
```

Out[20]:

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

대괄호 2개 = 2차원 대괄호 3개 = 3차원

In [25]:

```
x= np.array ([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])
x
```

Out[25]:

```
array([[[1, 2],
        [4, 5],
        [8, 9]],

       [[1, 2],
        [4, 5],
        [8, 9]]])
```

In [26]:

```
x. ndim
```

Out[26]:

```
3
```

몇차원인지 알려줌

In [27]:

```
x.shape
```

Out[27]:

```
(2, 3, 2)
```

In [28]:

```
x.dtype
```

Out[28]:

```
dtype('int32')
```

In [29]:

```
x.astype (np.float64)
```

Out[29]:

```
array([[[1., 2.],
        [4., 5.],
        [8., 9.]],

       [[1., 2.],
        [4., 5.],
        [8., 9.]]])
```

In [30]:

```
np.zeros_like(x)
```

Out[30]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0],
        [0, 0]]])
```

생긴건 똑같이 숫자를 0으로 바꿈

In [31]:

```
x*0
```

Out[31]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0],
        [0, 0]]])
```

In [32]:

```
data = np.random.normal(0,1, 100)
print(data)
```

```
[ 0.5948894   0.62901742 -2.43481232 -0.64520163 -2.25552614  0.19489435
 -0.78682143 -1.25504291 -0.27158939 -0.58122381 -0.3975076   0.19090601
 -0.7051918   0.85886849 -0.1332941   1.01240101 -0.37015608 -0.3629753
  0.62532226 -0.55471334  0.51391598 -0.46922885 -0.13815178 -0.95075819
  0.28241838  0.76452934  1.1830317  -1.68797608 -0.27279998  1.34480936
  0.38829424  1.41437356  0.21335677  1.51224885  0.24974161 -0.25661151
  0.16083075 -0.21285222 -0.06453529  0.28284785  0.19434011 -0.58968835
 -1.84297084  1.38741169 -0.0083715  -0.50905944  1.47782995 -1.34516351
  0.77349661  1.09383414  1.09127973 -1.11432754  0.94883338 -0.23355717
 -0.39135684 -1.81958121  0.23487778  0.82448781 -0.87749371 -0.73695733
 -0.12413642  2.71239267  0.89010278  1.174496   -0.66606796  0.00658702
 -0.50367648 -1.21744002 -0.59010186 -0.58975187 -2.62121707  1.21332786
  0.25443795 -0.64268828  2.12788179  0.94568845  1.24823598 -1.05314019
  1.36540336  0.22325166 -0.4619426   0.78764955 -1.13887395 -0.57905073
  0.46988864  0.44221884 -0.80193795  1.03495679  2.23371987 -0.58567336
 -0.06988392  0.24613168 -1.43981342 -2.03215962  0.95582361 -0.27585455
  0.34924642 -0.40700961  0.14194833  0.0361661 ]
```

normal = normal distribution 만들어줌 0 = mean 1= standard deviation 100은 100개의 data 만들어라

In [34]:

```
data.ndim
```

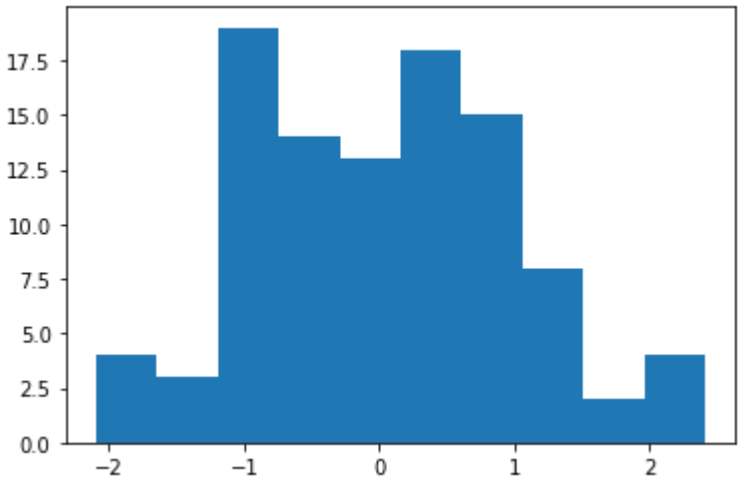
Out[34]:

```
1
```

In [35]:

```
data = np.random.normal(0,1, 100)
print(data)
plt.hist(data, bins=10)
plt.show()
```

```
[-0.29052981  0.3142496  -0.30092923  0.23294338 -0.0096937  -1.75264288
 -1.09379186  1.22418643  0.09211239  0.65077238 -0.18112944  0.59968089
  1.02008837  2.41507312  0.24426123 -0.61227634  2.210503   -0.42432102
  0.41848103  0.17453703 -0.86170595  1.58677576  0.8008617  -2.09665586
  1.36270504 -0.7862358   1.06420937 -0.05100987 -0.14991648 -0.75866774
 -1.12325    -0.86547804 -0.66849136 -0.75090565  0.71276998  0.17085606
  1.82536286 -1.32038878  0.75468933  0.27739382  1.15321934 -0.72314527
  1.08439659  1.48315776  2.13391077 -0.49218686  0.43490225 -0.09114682
 -0.80881791  0.80532131  0.73983968 -0.45784457 -1.62985832  0.38751317
 -0.53320662 -0.80517678  0.26518977 -1.11002782 -0.67592721  0.15333034
 -0.95616868  0.51447594  1.12573406  0.45444624 -0.85809141  0.84831613
  2.33190843  0.03833018 -1.08656145  0.62538228 -0.50315794  0.6696119
  0.54005942 -1.01313403  0.82195499 -0.77879309 -0.3755143  -0.97419932
  0.05934476  0.54322135  0.29534863 -0.2355785   0.11673703 -0.93300181
  1.06537317 -0.64259085 -0.63545322 -0.98052767 -1.55429745  0.81891179
  0.67071758 -2.04890039  0.70647498  0.68675609 -0.87092778 -0.65607294
  0.51223153 -0.04768571 -1.71196897  0.57827161]
```



bins =10 은 graph 에서 10개로 나눔 위에 graph의 값을 다 더하면 100개

In [37]:

```
X= np.ones ([2,3,4])
X
```

Out[37]:

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

       [[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]]])
```

2,3,4 = 3차원 2,3,4,5 = 4차원

In [38]:

```
Y = X.reshape (-1,3,2)
Y
```

Out[38]:

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

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]]])
```

reshape = shape바꾸는 것 -1은 뭔지 모르겠다 너가 알아서 해라 but 4써도 똑같이 나옴

In [42]:

```
np.allclose(X.reshape(-1,3,2),Y)
```

Out[42]:

```
True
```

X reshape 한거랑 y가 똑같은가

In [47]:

```
a = np.random.randint(0, 10, [2, 3])
b = np.random.random([2, 3])
np.savez("test", a, b)
```

randint = 0~10에서 숫자 골라서 2 3 matrix 만듬 np.savez 파일로 저장

In [43]:

```
ls -al test*
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: AE98-5A30

 C:₩Users₩ykim3₩OneDrive₩문서₩영어음성학 디렉터리


 C:₩Users₩ykim3₩OneDrive₩문서₩영어음성학 디렉터리

파일을 찾을 수 없습니다.
```

In [48]:

```
who
```

```
X       Y       a       b       data    np      plt     x
```

who 는 지금 어떤 variable 이 available 한지

In [54]:

```
del a,b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-54-a9b92be7739b> in <module>
----> 1 del a,b

NameError: name 'a' is not defined
```

In [55]:

```
who
```

```
X       Y       data    np      plt     x
```

In [56]:

```
npzfiles = np.load("test.npz")
npzfiles.files
```

Out[56]:

```
['arr_0', 'arr_1']
```

In [57]:

```
npzfiles['arr_0']
```

Out[57]:

```
array([[1, 6, 6],
       [2, 2, 8]])
```

npzfiles은 저장된 파일불러오기

skiprows 첫번째 row skip because it is title

In [58]:

```
arr= np.random.random([5,2,3])
```

In [59]:

```
print(type(arr))
print(len(arr))
print(arr.shape)
print(arr.ndim)
print(arr.size)
print(arr.dtype)
```

```
<class 'numpy.ndarray'>
5
(5, 2, 3)
3
30
float64
```

In [62]:

```
a=np.arange (1,5)
b=np.arange (9,5,-1)
```

In [63]:

```
print (a-b)
print (a*b)
```

```
[-8 -6 -4 -2]
[ 9 16 21 24]
```

In [64]:

```
a = np.arange(1, 10).reshape(3,3)
b = np.arange(9, 0, -1).reshape(3,3)
print(a)
print(b)
```

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

In [65]:

```
a==b
```

Out[65]:

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

In [66]:

```
a>b
```

Out[66]:

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

비교할때 dimention이랑 shape 같아야함

In [67]:

```
a
```

Out[67]:

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

In [69]:

```
a.sum()
```

Out[69]:

45

In [70]:

```
np.sum(a)
```

Out[70]:

45

In [71]:

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

Out[71]:

```
array([12, 15, 18])
```

axis = 몇번째 차원에서 실행

In [72]:

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

Out[72]:

```
array([ 6, 15, 24])
```

In [73]:

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

Out[73]:

```
array([ 6, 15, 24])
```

In [74]:

```
a = np.arange(1, 25).reshape(4, 6)
a
```

Out[74]:

```
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]])
```

In [75]:

```
a+100
```

Out[75]:

```
array([[101, 102, 103, 104, 105, 106],
       [107, 108, 109, 110, 111, 112],
       [113, 114, 115, 116, 117, 118],
       [119, 120, 121, 122, 123, 124]])
```

In [76]:

```
b= np.arange(6)
b
```

Out[76]:

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

In [77]:

```
a+b
```

Out[77]:

```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```

In [ ]:

```

```

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import pyplot as plt is the same this as above
```

In [6]:

```
np.empty([2,3], dtype='int')
```

Out[6]:

```
array([[-1662389008,          367,            0],
       [          0,       131074,    538970682]])
```

dtype = data type

In [7]:

```
np.zeros([2,3])
```

Out[7]:

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

2 by 3의 matrix을 0으로 채움

In [ ]:

```
[[0,0,0],[0,0,0]]
```

계산을 못하니까 쓸모가 없음

In [11]:

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

Out[11]:

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

array 사용

In [12]:

```
np.ones([2,3], dtype = 'float64')
```

Out[12]:

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

ones 라는 함수 float 64는 64 소수자리까지 (정확하지만 데이터 많이 차지)

In [13]:

```
np.ones([2,3], dtype = 'int')
```

Out[13]:

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

In [14]:

```
np.arange(5)
```

Out[14]:

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

float loop 때 range랑 비슷함

In [15]:

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

Out[15]:

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

In [17]:

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

Out[17]:

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

0부터 10까지 2개 차이로

In [16]:

```
np.arange(0,10,2, dtype='float64')
```

Out[16]:

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

In [18]:

```
np.linspace (0,10,6)
```

Out[18]:

```
array([ 0.,  2.,  4.,  6.,  8., 10.])
```

0부터 10까지 6개로 나눔

In [20]:

```
x= np.array ([[1,2],[4,5],[8,9]])
x
```

Out[20]:

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

대괄호 2개 = 2차원 대괄호 3개 = 3차원

In [25]:

```
x= np.array ([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])
x
```

Out[25]:

```
array([[[1, 2],
        [4, 5],
        [8, 9]],

       [[1, 2],
        [4, 5],
        [8, 9]]])
```

In [26]:

```
x. ndim
```

Out[26]:

```
3
```

몇차원인지 알려줌

In [27]:

```
x.shape
```

Out[27]:

```
(2, 3, 2)
```

In [28]:

```
x.dtype
```

Out[28]:

```
dtype('int32')
```

In [29]:

```
x.astype (np.float64)
```

Out[29]:

```
array([[[1., 2.],
        [4., 5.],
        [8., 9.]],

       [[1., 2.],
        [4., 5.],
        [8., 9.]]])
```

In [30]:

```
np.zeros_like(x)
```

Out[30]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0],
        [0, 0]]])
```

생긴건 똑같이 숫자를 0으로 바꿈

In [31]:

```
x*0
```

Out[31]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

       [[0, 0],
        [0, 0],
        [0, 0]]])
```

In [32]:

```
data = np.random.normal(0,1, 100)
print(data)
```

```
[ 0.5948894   0.62901742 -2.43481232 -0.64520163 -2.25552614  0.19489435
 -0.78682143 -1.25504291 -0.27158939 -0.58122381 -0.3975076   0.19090601
 -0.7051918   0.85886849 -0.1332941   1.01240101 -0.37015608 -0.3629753
  0.62532226 -0.55471334  0.51391598 -0.46922885 -0.13815178 -0.95075819
  0.28241838  0.76452934  1.1830317  -1.68797608 -0.27279998  1.34480936
  0.38829424  1.41437356  0.21335677  1.51224885  0.24974161 -0.25661151
  0.16083075 -0.21285222 -0.06453529  0.28284785  0.19434011 -0.58968835
 -1.84297084  1.38741169 -0.0083715  -0.50905944  1.47782995 -1.34516351
  0.77349661  1.09383414  1.09127973 -1.11432754  0.94883338 -0.23355717
 -0.39135684 -1.81958121  0.23487778  0.82448781 -0.87749371 -0.73695733
 -0.12413642  2.71239267  0.89010278  1.174496   -0.66606796  0.00658702
 -0.50367648 -1.21744002 -0.59010186 -0.58975187 -2.62121707  1.21332786
  0.25443795 -0.64268828  2.12788179  0.94568845  1.24823598 -1.05314019
  1.36540336  0.22325166 -0.4619426   0.78764955 -1.13887395 -0.57905073
  0.46988864  0.44221884 -0.80193795  1.03495679  2.23371987 -0.58567336
 -0.06988392  0.24613168 -1.43981342 -2.03215962  0.95582361 -0.27585455
  0.34924642 -0.40700961  0.14194833  0.0361661 ]
```

normal = normal distribution 만들어줌 0 = mean 1= standard deviation 100은 100개의 data 만들어라

In [34]:

```
data.ndim
```

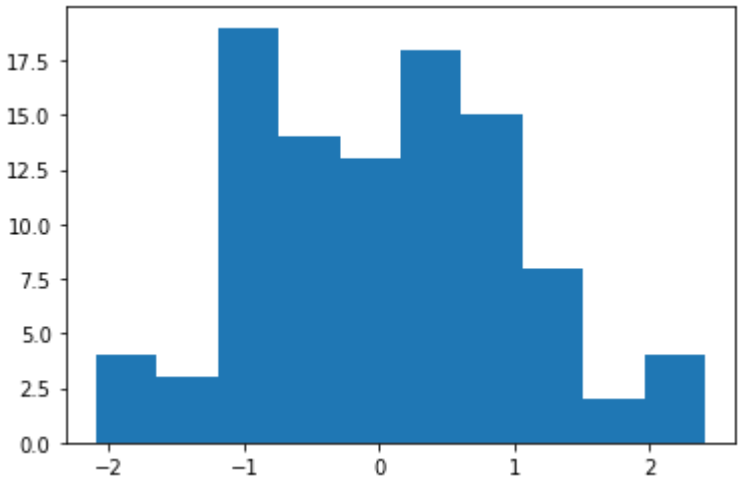
Out[34]:

```
1
```

In [35]:

```
data = np.random.normal(0,1, 100)
print(data)
plt.hist(data, bins=10)
plt.show()
```

```
[-0.29052981  0.3142496  -0.30092923  0.23294338 -0.0096937  -1.75264288
 -1.09379186  1.22418643  0.09211239  0.65077238 -0.18112944  0.59968089
  1.02008837  2.41507312  0.24426123 -0.61227634  2.210503   -0.42432102
  0.41848103  0.17453703 -0.86170595  1.58677576  0.8008617  -2.09665586
  1.36270504 -0.7862358   1.06420937 -0.05100987 -0.14991648 -0.75866774
 -1.12325    -0.86547804 -0.66849136 -0.75090565  0.71276998  0.17085606
  1.82536286 -1.32038878  0.75468933  0.27739382  1.15321934 -0.72314527
  1.08439659  1.48315776  2.13391077 -0.49218686  0.43490225 -0.09114682
 -0.80881791  0.80532131  0.73983968 -0.45784457 -1.62985832  0.38751317
 -0.53320662 -0.80517678  0.26518977 -1.11002782 -0.67592721  0.15333034
 -0.95616868  0.51447594  1.12573406  0.45444624 -0.85809141  0.84831613
  2.33190843  0.03833018 -1.08656145  0.62538228 -0.50315794  0.6696119
  0.54005942 -1.01313403  0.82195499 -0.77879309 -0.3755143  -0.97419932
  0.05934476  0.54322135  0.29534863 -0.2355785   0.11673703 -0.93300181
  1.06537317 -0.64259085 -0.63545322 -0.98052767 -1.55429745  0.81891179
  0.67071758 -2.04890039  0.70647498  0.68675609 -0.87092778 -0.65607294
  0.51223153 -0.04768571 -1.71196897  0.57827161]
```



bins =10 은 graph 에서 10개로 나눔 위에 graph의 값을 다 더하면 100개

In [37]:

```
X= np.ones ([2,3,4])
X
```

Out[37]:

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

       [[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]]])
```

2,3,4 = 3차원 2,3,4,5 = 4차원

In [38]:

```
Y = X.reshape (-1,3,2)
Y
```

Out[38]:

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

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]]])
```

reshape = shape바꾸는 것 -1은 뭔지 모르겠다 너가 알아서 해라 but 4써도 똑같이 나옴

In [42]:

```
np.allclose(X.reshape(-1,3,2),Y)
```

Out[42]:

```
True
```

X reshape 한거랑 y가 똑같은가

In [47]:

```
a = np.random.randint(0, 10, [2, 3])
b = np.random.random([2, 3])
np.savez("test", a, b)
```

randint = 0~10에서 숫자 골라서 2 3 matrix 만듬 np.savez 파일로 저장

In [43]:

```
ls -al test*
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: AE98-5A30

 C:₩Users₩ykim3₩OneDrive₩문서₩영어음성학 디렉터리


 C:₩Users₩ykim3₩OneDrive₩문서₩영어음성학 디렉터리

파일을 찾을 수 없습니다.
```

In [48]:

```
who
```

```
X       Y       a       b       data    np      plt     x
```

who 는 지금 어떤 variable 이 available 한지

In [54]:

```
del a,b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-54-a9b92be7739b> in <module>
----> 1 del a,b

NameError: name 'a' is not defined
```

In [55]:

```
who
```

```
X       Y       data    np      plt     x
```

In [56]:

```
npzfiles = np.load("test.npz")
npzfiles.files
```

Out[56]:

```
['arr_0', 'arr_1']
```

In [57]:

```
npzfiles['arr_0']
```

Out[57]:

```
array([[1, 6, 6],
       [2, 2, 8]])
```

npzfiles은 저장된 파일불러오기

skiprows 첫번째 row skip because it is title

In [58]:

```
arr= np.random.random([5,2,3])
```

In [59]:

```
print(type(arr))
print(len(arr))
print(arr.shape)
print(arr.ndim)
print(arr.size)
print(arr.dtype)
```

```
<class 'numpy.ndarray'>
5
(5, 2, 3)
3
30
float64
```

In [62]:

```
a=np.arange (1,5)
b=np.arange (9,5,-1)
```

In [63]:

```
print (a-b)
print (a*b)
```

```
[-8 -6 -4 -2]
[ 9 16 21 24]
```

In [64]:

```
a = np.arange(1, 10).reshape(3,3)
b = np.arange(9, 0, -1).reshape(3,3)
print(a)
print(b)
```

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

In [65]:

```
a==b
```

Out[65]:

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

In [66]:

```
a>b
```

Out[66]:

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

비교할때 dimention이랑 shape 같아야함

In [67]:

```
a
```

Out[67]:

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

In [69]:

```
a.sum()
```

Out[69]:

45

In [70]:

```
np.sum(a)
```

Out[70]:

45

In [71]:

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

Out[71]:

```
array([12, 15, 18])
```

axis = 몇번째 차원에서 실행

In [72]:

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

Out[72]:

```
array([ 6, 15, 24])
```

In [73]:

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

Out[73]:

```
array([ 6, 15, 24])
```

In [74]:

```
a = np.arange(1, 25).reshape(4, 6)
a
```

Out[74]:

```
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]])
```

In [75]:

```
a+100
```

Out[75]:

```
array([[101, 102, 103, 104, 105, 106],
       [107, 108, 109, 110, 111, 112],
       [113, 114, 115, 116, 117, 118],
       [119, 120, 121, 122, 123, 124]])
```

In [76]:

```
b= np.arange(6)
b
```

Out[76]:

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

In [77]:

```
a+b
```

Out[77]:

```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```

In [2]:

```
./nbconvert.py --format=pdf yourfile.ipynb
```

```
  File "<ipython-input-2-2849e3d77fc9>", line 1
    ./nbconvert.py --format=pdf yourfile.ipynb
    ^
SyntaxError: invalid syntax
```

In [ ]:

```

```