numpy

In [2]:
```
import numpy as np
```

In [3]:
```
# create an 1d array
a=np.array([1,2,3,4])
a
```

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

In [5]:
```
# create an 2d array
b=np.array([[1,2,3,4],[5,6,7,8]])
b
```

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

In [6]:
```
# create an 3d array
c=np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
c
```

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

       [[ 7,  8,  9],
        [10, 11, 12]]])
```

In [8]:
```
# get dimension
print(a.ndim,"d")
print(b.ndim,"d")
print(c.ndim,"d")
```

```
1 d
2 d
3 d
```

In [9]:
```
# get shape
print(a.shape)
print(b.shape)
print(c.shape)
```

```
(4,)
(2, 4)
(2, 2, 3)
```

In [10]:
```
# get type
print(a.dtype)
print(b.dtype)
print(c.dtype)
```

```
int64
int64
int64
```

In [11]:

```
# get itemsize
print(a.itemsize)
print(b.itemsize)
print(c.itemsize)
```

```
8
8
8
```

In [14]:

```
# get size
print(a.size)
print(b.size)
print(c.size)
```

```
4
8
12
```

In [16]:

```
# get total size = size*itemsize
print(a.nbytes)
print(b.nbytes)
print(c.nbytes)
```

```
32
64
96
```

In [17]:

```
# get a specific element from row
print(a[2])
print(b[1,2])
print(c[1,1,2])
```

```
3
7
12
```

In [20]:

```
# get a specific row
print(a[:])
print(b[1,:])
print(c[0,0,:])
```

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

In [21]:

```
# get a specific column
print(a[2])
print(b[:,2])
print(c[1,:,1])
```

```
3
[3 7]
[ 8 11]
```

In [23]:

```
# getting a little more fancy[s.index:e.index:st.index]
print(a[0:3:2])
print(b[0,0:3:2])
print(c[1,0:3:2,0:3:2])
```

```
[1 3]
[1 3]
[[7 9]]
```

In [29]:

```
# all 0's matrix
print(np.zeros(4),"1d,")
print(np.zeros((2,3)),"2d,")
print(np.zeros((2,3,4)),"3d,")
```

```
[0. 0. 0. 0.] 1d,
[[0. 0. 0.]
 [0. 0. 0.]] 2d,
[[[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]] 3d,
```

In [30]:

```
# all 1's matrix
print(np.ones(4),"1d,")
print(np.ones((3,4)),"2d,")
print(np.ones((2,3,4)),"3d,")
```

```
[1. 1. 1. 1.] 1d,
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]] 2d,
[[[1. 1. 1. 1.]
  [1. 1. 1. 1.]
  [1. 1. 1. 1.]]

 [[1. 1. 1. 1.]
  [1. 1. 1. 1.]
  [1. 1. 1. 1.]]] 3d,
```

In [32]:

```
# full method
print(np.full(1,2),"1d,")
print(np.full((2,3),2),"2d,")
print(np.full((2,2,3),4),"3d,")
```

```
[2] 1d,
[[2 2 2]
 [2 2 2]] 2d,
[[[4 4 4]
  [4 4 4]]

 [[4 4 4]
  [4 4 4]]] 3d,
```

In [33]:

```
# full_like method here we change all present matrix value in a specific value
print(np.full_like(a.shape,22))
print(np.full_like(b.shape,33))
print(np.full_like(c.shape,44))
```

```
[22]
[33 33]
[44 44 44]
```

In [34]:

```
# random decimal numbers
print(np.random.rand(2,2,4))
```

```
[[[0.80296114 0.09487294 0.51161776 0.6819363 ]
  [0.91160124 0.73574521 0.1738925  0.48441694]]

 [[0.8245945  0.52624572 0.96013424 0.91459766]
  [0.86113075 0.69536227 0.96944071 0.85648539]]]
```

In [35]:

```
# random decimal no. replace a full matrix
print(np.random.random_sample(b.shape))
```

```
[[0.37222497 0.84462526 0.4078412  0.13518357]
 [0.85381057 0.59637908 0.18902396 0.57840766]]
```

In [43]:

```
# random integer value
np.random.randint(2,8,size=(2,3,3))
```

Out[43]:

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

       [[7, 7, 5],
        [7, 7, 3],
        [2, 7, 7]]])
```

In [49]:

```
# create a identity matrix
np.identity(4)
```

Out[49]:

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

In [61]:

```
# repeat a array in row's side
np.repeat(a,3,axis=0)
```

Out[61]:

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

In [64]:

```
# repeat a array in column's side
np.repeat(b,4,axis=0)
```

Out[64]:

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

In [69]:

```
# copy   a array
d=a.copy()
d
```

Out[69]:

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

In [71]:

```
# mathematics
print(a+2,",")
print(b+2,",")
print(c+2,",")
print(a**2,",")
```

```
[3 4 5 6] ,
[[ 3  4  5  6]
 [ 7  8  9 10]] ,
[[[ 3  4  5]
  [ 6  7  8]]

 [[ 9 10 11]
  [12 13 14]]] ,
[ 1  4  9 16] ,
```

In [73]:

```
# take trignomatry
print(np.sin(a),",")
print(np.cos(b),",")
print(np.tan(c),",")
```

```
[ 0.84147098  0.90929743  0.14112001 -0.7568025 ] ,
[[ 0.54030231 -0.41614684 -0.9899925  -0.65364362]
 [ 0.28366219  0.96017029  0.75390225 -0.14550003]] ,
[[[ 1.55740772e+00 -2.18503986e+00 -1.42546543e-01]
  [ 1.15782128e+00 -3.38051501e+00 -2.91006191e-01]]

 [[ 8.71447983e-01 -6.79971146e+00 -4.52315659e-01]
  [ 6.48360827e-01 -2.25950846e+02 -6.35859929e-01]]] ,
```

In [74]:

```
# maltiply of 2 matrix
e=np.ones((2,3))
f=np.full((3,2),3)
np.matmul(e,f)
```

Out[74]:

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

In [77]:

```
# find determinant
np.linalg.det(np.full((2,2),3))
```

Out[77]:

```
0.0
```

In [78]:

```
# statistics
print(np.mean(a))
print(np.max(b))
print(np.median(c))
```

```
2.5
8
6.5
```

In [79]:

```
# reshape of a given matrix
b.reshape(2,2,2)
```

Out[79]:

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

       [[5, 6],
        [7, 8]]])
```

In [81]:

```
# add vertically of 2 or more matrix
np.vstack([np.array([1,2,3,4]),np.array([5,6,7,8])])
```

Out[81]:

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

In [82]:

```
# add horizontal of 2 or more matrix
np.hstack([np.array([1,2,3,4]),np.array([5,6,7,8])])
```

Out[82]:

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

In [85]:

```
# load data from file
g=np.genfromtxt("Text.csv",delimiter=",")
g
```

Out[85]:

```
array([[  nan,   nan,   nan, ...,   nan,   nan,   nan],
       [16.99,  1.01,   nan, ...,   nan,   nan,  2.  ],
       [10.34,  1.66,   nan, ...,   nan,   nan,  3.  ],
       ...,
       [22.67,  2.  ,   nan, ...,   nan,   nan,  2.  ],
       [17.82,  1.75,   nan, ...,   nan,   nan,  2.  ],
       [18.78,  3.  ,   nan, ...,   nan,   nan,  2.  ]])
```

In [88]:

```
g.shape
```

Out[88]:

```
(245, 7)
```

In [89]:
```
g[g>12]
```

```
/home/sunil_/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid value encountered in greater
  """Entry point for launching an IPython kernel.
```

Out[89]:
```
array([16.99, 21.01, 23.68, 24.59, 25.29, 26.88, 15.04, 14.78, 35.26,
       15.42, 18.43, 14.83, 21.58, 16.29, 16.97, 20.65, 17.92, 20.29,
       15.77, 39.42, 19.82, 17.81, 13.37, 12.69, 21.7 , 19.65, 18.35,
       15.06, 20.69, 17.78, 24.06, 16.31, 16.93, 18.69, 31.27, 16.04,
       17.46, 13.94, 30.4 , 18.29, 22.23, 32.4 , 28.55, 18.04, 12.54,
       34.81, 25.56, 19.49, 38.01, 26.41, 48.27, 20.29, 13.81, 18.29,
       17.59, 20.08, 16.45, 20.23, 15.01, 12.02, 17.07, 26.86, 25.28,
       14.73, 17.92, 27.2 , 22.76, 17.29, 19.44, 16.66, 32.68, 15.98,
       34.83, 13.03, 18.28, 24.71, 21.16, 28.97, 22.49, 16.32, 22.75,
       40.17, 27.28, 12.03, 21.01, 12.46, 15.38, 44.3 , 22.42, 20.92,
       15.36, 20.49, 25.21, 18.24, 14.31, 14.  , 38.07, 23.95, 25.71,
       17.31, 29.93, 12.43, 24.08, 13.42, 14.26, 15.95, 12.48, 29.8 ,
       14.52, 22.82, 19.08, 20.27, 12.26, 18.26, 14.15, 16.  , 13.16,
       17.47, 34.3 , 41.19, 27.05, 16.43, 18.64, 14.07, 13.13, 17.26,
       24.55, 19.77, 29.85, 48.17, 25.  , 13.39, 16.49, 21.5 , 12.66,
       16.21, 13.81, 17.51, 24.52, 20.76, 31.71, 50.81, 15.81, 31.85,
       16.82, 32.9 , 17.89, 14.48, 34.63, 34.65, 23.33, 45.35, 23.17,
       40.55, 20.69, 20.9 , 30.46, 18.15, 23.1 , 15.69, 19.81, 28.44,
       15.48, 16.58, 43.11, 13.  , 13.51, 18.71, 12.74, 13.  , 16.4 ,
       20.53, 16.47, 26.59, 38.73, 24.27, 12.76, 30.06, 25.89, 48.33,
       13.27, 28.17, 12.9 , 28.15, 30.14, 12.16, 13.42, 15.98, 13.42,
       16.27, 20.45, 13.28, 22.12, 24.01, 15.69, 15.53, 12.6 , 32.83,
       35.83, 29.03, 27.18, 22.67, 17.82, 18.78])
```

In [90]:
```
((g>12) & (g<20))
```

```
/home/sunil_/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid value encountered in greater
  """Entry point for launching an IPython kernel.
/home/sunil_/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid value encountered in less
  """Entry point for launching an IPython kernel.
```

Out[90]:
```
array([[False, False, False, ..., False, False, False],
       [ True, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       ...,
       [False, False, False, ..., False, False, False],
       [ True, False, False, ..., False, False, False],
       [ True, False, False, ..., False, False, False]])
```

In [91]:
```
# find maximum value's index
a.argmax()
```

Out[91]:
```
3
```

In [92]:
```
a.argmin()
```

Out[92]:
```
0
```

In [94]:
```
# all function
np.all(a==b)
```

Out[94]:
```
False
```

In [98]:

```
# transpose
b.T
```

Out[98]:

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

In [100]:

```
# convert all matrix in 1d array
b.ravel()
```

Out[100]:

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

In [111]:

```
# sorting
b.sort()
b
```

Out[111]:

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

In [113]:

```
b.argsort()
```

Out[113]:

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

In [116]:

```
a = np.array([[0, 1], [2, 3]], order='C')
a.resize((2, 1))
a
```

Out[116]:

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

In [ ]: