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

In [3]:
```
np.__version__
```

Out[3]:
```
'1.26.4'
```

In [5]:
```
import sys
sys.version
```

Out[5]:
```
'3.12.4 | packaged by Anaconda, Inc. | (main, Jun 18 2024, 15:03:56) [MSC v.1929 6
4 bit (AMD64)]'
```

creating list

In [8]:
```
my_list=[0,1,2,3,4,5]
my_list
```

Out[8]:
```
[0, 1, 2, 3, 4, 5]
```

In [12]:
```
type(my_list)
```

Out[12]:
```
list
```

In [14]:
```
arr=np.array(my_list)
arr
```

Out[14]:
```
array([0, 1, 2, 3, 4, 5])
```

In [16]:
```
type(arr)
```

Out[16]:
```
numpy.ndarray
```

In [18]:
```
np.arange(15)
```

Out[18]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [20]:
```
np.arange(3,0)
```

Out[20]:
```
array([], dtype=int32)
```

In [22]:
```
np.arange(10)
```

Out[22]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [24]:
```
np.arange(0,5)
```

Out[24]:
```
array([0, 1, 2, 3, 4])
```

In [26]:
```
np.arange(10,20)
```

Out[26]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

In [28]:
```
np.arange(20,10)
```

Out[28]:
```
array([], dtype=int32)
```

In [30]:
```
np.arange(-20,10)
```

Out[30]:
```
array([-20, -19, -18, -17, -16, -15, -14, -13, -12, -11, -10,  -9,  -8,
        -7,  -6,  -5,  -4,  -3,  -2,  -1,   0,   1,   2,   3,   4,   5,
         6,   7,   8,   9])
```

In [32]:
```
np.arange(-16,10)
```

Out[32]:
```
array([-16, -15, -14, -13, -12, -11, -10,  -9,  -8,  -7,  -6,  -5,  -4,
        -3,  -2,  -1,   0,   1,   2,   3,   4,   5,   6,   7,   8,   9])
```

In [34]:
```
np.arange(-20,-10)
```

Out[34]:
```
array([-20, -19, -18, -17, -16, -15, -14, -13, -12, -11])
```

In [36]:
```
ar=np.arange(-30,20)
```

In [40]:
```
ar
```

Out[40]:
```
array([-30, -29, -28, -27, -26, -25, -24, -23, -22, -21, -20, -19, -18,
       -17, -16, -15, -14, -13, -12, -11, -10,  -9,  -8,  -7,  -6,  -5,
        -4,  -3,  -2,  -1,   0,   1,   2,   3,   4,   5,   6,   7,   8,
         9,  10,  11,  12,  13,  14,  15,  16,  17,  18,  19])
```

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

Out[42]:
```
array([], dtype=int32)
```

In [44]:
```
np.arange()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[44], line 1
----> 1 np.arange()

TypeError: arange() requires stop to be specified.
```

In [48]:
```
np.arange(10,30,5)
```

Out[48]:
```
array([10, 15, 20, 25])
```

In [50]:
```
np.arange(0,10,3)
```

Out[50]:
```
array([0, 3, 6, 9])
```

In [52]:
```
np.arange(10,20,30,5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[52], line 1
----> 1 np.arange(10,20,30,5)

TypeError: Cannot interpret '5' as a data type
```

In [54]:
```
np.zeros(3)
```

Out[54]:
```
array([0., 0., 0.])
```

In [56]:
```
np.zeros(6)
```

Out[56]:
```
array([0., 0., 0., 0., 0., 0.])
```

In [58]:
```
np.zeros(3,dtype=int)
```

Out[58]:
```
array([0, 0, 0])
```

In [60]:
```
np.zeros((2,2),dtype=int)
```

Out[60]:
```
array([[0, 0],
       [0, 0]])
```

In [64]:
```
zero=np.zeros([2,2])
print(zero)
print('###')
print(type(zero))
```

```
[[0. 0.]
 [0. 0.]]
###
<class 'numpy.ndarray'>
```

In [66]:
```
np.zeros((2,10))
```

Out[66]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

In [68]:
```
n=(6,7)
n1=(6,8)
print(np.zeros(n))
```

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

In [70]:
```
print(np.zeros(n1))
```

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

In [72]:
```
np.ones(3)
```

Out[72]:
```
array([1., 1., 1.])
```

In [74]:
```
np.ones(4,dtype=int)
```

Out[74]:
```
array([1, 1, 1, 1])
```

In [78]:
```
range(5)
```

Out[78]:
```
range(0, 5)
```

In [80]:
```
list(range(0,5))
```

Out[80]:
```
[0, 1, 2, 3, 4]
```

In [82]:
```
list(range(1,10,3))
```

Out[82]:
```
[1, 4, 7]
```

In [86]:
```
rand(2,3)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[86], line 1
----> 1 rand(2,3)

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

In [88]:
```
rand(3,2)
random.rand(3,2)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[88], line 1
----> 1 rand(3,2)
      2 random.rand(3,2)

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

In [90]:
```
np.random.rand(5)
```

Out[90]:
```
array([0.89403475, 0.47050074, 0.62868235, 0.82269941, 0.08842443])
```

In [92]:
```
np.random.rand(2,4)
```

Out[92]:
```
array([[0.97852534, 0.53343221, 0.2942976 , 0.5429143 ],
       [0.02882302, 0.4167748 , 0.00572514, 0.87710436]])
```

In [94]:
```
np.random.randint(2,4)
```

Out[94]:
```
2
```

In [98]:
```
np.random.randint(2,20)
```

Out[98]:
```
17
```

In [100…
```
np.random.randint(0,1)
```

Out[100…
```
0
```

In [102…
```
np.random.randint(10,20,3)
```

Out[102…
```
array([15, 16, 14])
```

In [104…
```
np.random.randint(10,20,(10,10))
```

Out[104…
```
array([[12, 17, 11, 13, 13, 10, 18, 17, 17, 14],
       [14, 15, 18, 11, 19, 17, 15, 12, 12, 10],
       [15, 16, 17, 12, 11, 10, 10, 11, 19, 18],
       [11, 17, 14, 14, 16, 19, 10, 13, 10, 19],
       [14, 13, 11, 15, 16, 11, 12, 17, 15, 14],
       [19, 15, 16, 12, 17, 10, 15, 16, 16, 16],
       [13, 12, 17, 19, 19, 10, 10, 12, 19, 17],
       [10, 11, 14, 12, 18, 15, 10, 11, 14, 19],
       [15, 15, 16, 11, 15, 11, 13, 13, 10, 10],
       [13, 12, 17, 16, 11, 11, 16, 18, 13, 13]])
```

In [110…
```
new=np.array([[45,34,22,2],[24,55,3,22]])
print(new)
```

```
[[45 34 22  2]
 [24 55  3 22]]
```

In [112…
```
new1=np.array([[35,26,40],[29,10,34]])
print(new1)
```

```
[[35 26 40]
 [29 10 34]]
```

In [114…
```
np.array([[1,2,3],[4,5,6],[7,8,9],[10,12,13]])
```

Out[114…
```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 12, 13]])
```

dtype-desired data type

In [117…
```
np.array([1,2,3],dtype=float)
```

Out[117…
```
array([1., 2., 3.])
```

In [119…
```
np.array([4,5,6],dtype=int)
```

Out[119…
```
array([4, 5, 6])
```

In [121…
```
np.array([10,20,30,40],dtype=bool)
```

Out[121…
```
array([ True,  True,  True,  True])
```

In [123…
```
np.array([0,1,3,4],dtype=bool)
```

Out[123…
```
array([False,  True,  True,  True])
```

In [125…
```
np.array([11,34,56],dtype=complex)
```

Out[125…
```
array([11.+0.j, 34.+0.j, 56.+0.j])
```

In [127…
```
np.array([12,43,56],dtype=int)
```

Out[127…
```
array([12, 43, 56])
```

arange and reshape

In [130…
```
np.arange(1,19)
```

Out[130…
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18])
```

In [132…
```
np.arange(10,23,2)
```

Out[132…
```
array([10, 12, 14, 16, 18, 20, 22])
```

In [134…
```
np.arange(20,50,3,2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[134], line 1
----> 1 np.arange(20,50,3,2)

TypeError: Cannot interpret '2' as a data type
```

In [136…
```
np.arange(20,30).reshape(5,2)
```

Out[136…
```
array([[20, 21],
       [22, 23],
       [24, 25],
       [26, 27],
       [28, 29]])
```

In [138…
```
np.arange(12,18).reshape(2,3)
```

Out[138…
```
array([[12, 13, 14],
       [15, 16, 17]])
```

In [140…
```
np.arange(23,48).reshape(5,5)
```

Out[140…
```
array([[23, 24, 25, 26, 27],
       [28, 29, 30, 31, 32],
       [33, 34, 35, 36, 37],
       [38, 39, 40, 41, 42],
       [43, 44, 45, 46, 47]])
```

ones and zeros

In [143…
```
np.ones(2)
```

Out[143…
```
array([1., 1.])
```

In [145…
```
p=np.ones(2,dtype=int)
```

In [147…
```
p
```

Out[147…
```
array([1, 1])
```

In [149…
```
np.zeros((2,3))
```

Out[149…
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [151…
```
np.zeros(6)
```

Out[151…
```
array([0., 0., 0., 0., 0., 0.])
```

In [157…
```
s=np.ones((4,6),dtype=bool)
s
```

Out[157…
```
array([[ True,  True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True,  True]])
```

In [159…
```
np.ones((3,2),dtype=int)
```

Out[159…
```
array([[1, 1],
       [1, 1],
       [1, 1]])
```

In [161…
```
np.random.rand(2,4)
```

Out[161…
```
array([[0.15394193, 0.99796893, 0.46821237, 0.48380895],
       [0.67161068, 0.39526297, 0.13162025, 0.7222622 ]])
```

In [163…

```
np.random.rand(10,20,4)
```

Out[163…

```
array([[[7.96429158e-01, 4.25945147e-01, 3.64630503e-01, 5.65377158e-01],
        [1.73301490e-01, 3.97228671e-01, 9.47933796e-01, 7.91273818e-01],
        [1.94309001e-01, 6.56133513e-01, 3.84338528e-01, 9.88799242e-01],
        [5.39664101e-01, 6.03488747e-01, 8.28487563e-01, 7.17718371e-01],
        [8.56283173e-01, 9.09822142e-02, 6.91415842e-01, 9.58031424e-01],
        [1.88834323e-01, 2.05343697e-02, 8.35545118e-01, 2.26744915e-01],
        [8.84382658e-01, 8.32689579e-01, 7.45377754e-01, 5.44922937e-01],
        [7.39286993e-01, 4.56278497e-01, 4.54772197e-01, 9.33151601e-01],
        [9.75673427e-01, 3.16681757e-01, 2.30597784e-01, 6.98468472e-01],
        [9.20199742e-01, 4.61197033e-01, 8.78289555e-01, 4.29538524e-01],
        [3.57658776e-01, 5.35766248e-01, 6.99233491e-01, 1.27932018e-01],
        [7.11591100e-02, 1.95151908e-01, 9.90929094e-01, 2.81734592e-01],
        [3.83471930e-01, 4.62619014e-01, 5.23522584e-01, 8.83130975e-01],
        [3.59150502e-01, 5.07311686e-01, 5.85104617e-01, 2.55201082e-01],
        [9.47285505e-01, 8.22906503e-01, 4.14399910e-01, 1.25208731e-01],
        [3.44808317e-01, 6.61568572e-01, 2.66111675e-01, 2.49113032e-01],
        [5.70315853e-01, 6.78502507e-01, 1.61180397e-01, 5.80104298e-01],
        [2.22266397e-01, 1.86915138e-01, 3.79761774e-01, 7.33652930e-01],
        [9.98168560e-01, 4.81509903e-01, 3.98387863e-01, 6.53373447e-01],
        [1.03750553e-01, 2.25756501e-01, 7.19900662e-01, 3.23269348e-01]],

       [[4.22558690e-01, 2.86379087e-01, 1.95606163e-01, 7.35034001e-01],
        [8.87489325e-01, 3.94284260e-01, 4.07660360e-01, 8.42321149e-01],
        [7.10036938e-01, 4.45479540e-01, 7.60579595e-02, 5.86819586e-01],
        [2.06200995e-01, 7.67685172e-01, 3.46889225e-01, 3.73551115e-01],
        [2.63304212e-01, 5.30838066e-01, 6.34849320e-01, 8.51877274e-01],
        [1.77328728e-01, 6.34821134e-02, 9.20455046e-01, 3.61898733e-01],
        [4.13565648e-01, 5.69950340e-01, 3.91544515e-01, 4.51454780e-01],
        [7.18837966e-02, 8.14653689e-01, 2.74561808e-01, 2.69413023e-01],
        [2.10800101e-01, 4.15914999e-01, 4.89515079e-01, 2.52443058e-01],
        [4.88569742e-01, 9.55910419e-01, 6.55495290e-01, 3.94419123e-01],
        [2.92495225e-01, 8.46712111e-02, 4.49379005e-01, 5.03909399e-02],
        [3.16687959e-01, 4.52379358e-01, 3.29145212e-01, 6.09784847e-01],
        [7.13425664e-01, 8.62090940e-01, 7.41646171e-01, 9.03985773e-01],
        [2.15171906e-01, 1.80129725e-01, 7.46735612e-01, 3.73860038e-01],
        [8.41100248e-01, 5.26151158e-01, 1.67842365e-01, 6.37683099e-01],
        [5.19181418e-01, 8.85541943e-01, 9.99966894e-02, 5.14153463e-01],
        [8.06251626e-01, 1.57162868e-01, 7.37710770e-03, 7.72524869e-01],
        [9.55134515e-01, 2.45258161e-01, 8.09377447e-01, 2.35812472e-01],
        [4.49795407e-01, 7.64757882e-01, 4.78612405e-01, 7.67156990e-01],
        [3.53253920e-01, 4.97497770e-01, 1.84777791e-02, 5.31911368e-01]],

       [[1.40631913e-01, 2.84492090e-01, 9.03580511e-01, 8.92929008e-01],
        [9.55541333e-01, 4.86083455e-01, 9.35906438e-01, 1.21816034e-01],
        [9.91704517e-01, 6.48775715e-01, 2.44732187e-01, 2.89023081e-01],
        [2.30173194e-01, 7.14564490e-01, 8.21265621e-01, 4.89813559e-01],
        [8.36284391e-01, 9.54960836e-01, 6.66203169e-03, 4.74641175e-01],
        [1.03770494e-01, 8.80587445e-01, 2.14054822e-02, 2.58806972e-01],
        [7.27065215e-01, 2.40853310e-01, 5.87239033e-01, 6.60707523e-01],
        [6.47425428e-01, 1.77884259e-01, 7.92801254e-01, 3.22554656e-01],
        [1.27491250e-01, 4.83670987e-01, 8.45660650e-01, 1.12129965e-01],
        [8.04953781e-01, 4.59783445e-01, 5.85055022e-01, 2.19356718e-01],
        [7.30295709e-01, 1.05250489e-01, 6.61024211e-01, 7.10197133e-01],
        [6.68374032e-01, 8.55020326e-01, 8.21243392e-01, 9.66931958e-02],
        [1.68024456e-01, 9.12859240e-01, 2.32723085e-01, 6.92639755e-02],
        [3.05491399e-01, 1.42273562e-01, 5.26478534e-01, 8.67279402e-01],
```

```
 [1.83590550e-01, 6.10849405e-01, 9.80425621e-01, 4.02819317e-01],
 [4.10929485e-01, 6.35062272e-01, 1.56368607e-02, 1.08295362e-01],
 [5.15571424e-01, 2.90632109e-01, 8.55118192e-01, 8.47469372e-01],
 [5.95511542e-01, 1.94314888e-01, 5.78948482e-01, 5.86762549e-01],
 [3.50196673e-02, 7.29097315e-01, 5.55535399e-01, 6.33053586e-01],
 [3.13277009e-02, 8.71423018e-01, 9.32069816e-01, 5.02821928e-02]],

[[1.23198615e-01, 9.04170850e-01, 3.51720460e-01, 3.10396485e-01],
 [1.21926520e-01, 5.02954893e-01, 2.33255417e-01, 2.74667753e-01],
 [2.58033399e-02, 4.63030733e-02, 2.46166168e-01, 6.63447101e-01],
 [1.57803362e-01, 1.28900766e-01, 6.67133430e-01, 2.41941547e-01],
 [5.68156940e-01, 1.03610567e-01, 7.20187428e-01, 6.65634931e-01],
 [1.56418019e-01, 3.08501598e-01, 6.07943530e-01, 8.98703119e-01],
 [5.15685523e-01, 4.89795837e-01, 7.16497897e-01, 5.35422410e-01],
 [7.69506465e-01, 6.98359757e-01, 1.52337662e-01, 8.81079735e-01],
 [6.89463480e-01, 9.77698340e-01, 6.35091638e-01, 2.25926278e-01],
 [3.76194017e-01, 7.02367961e-01, 9.00384135e-01, 6.19659224e-01],
 [1.32355067e-01, 1.43130088e-01, 7.03547662e-01, 8.47241325e-02],
 [9.36755588e-01, 4.73965587e-01, 5.46806463e-01, 8.56028466e-01],
 [1.98027186e-01, 5.52859178e-01, 9.98744661e-01, 5.01224834e-01],
 [9.70886384e-01, 7.96873603e-01, 3.35085752e-02, 8.16569123e-01],
 [9.39075221e-01, 4.59733295e-01, 1.73486443e-01, 2.07934126e-01],
 [3.67710784e-01, 2.05860780e-01, 9.45177432e-01, 7.10442060e-01],
 [2.57957609e-01, 7.83027745e-01, 1.03450725e-01, 3.16229415e-01],
 [9.28052467e-02, 5.50059024e-01, 8.74664810e-01, 4.21732272e-01],
 [3.96541636e-01, 8.21476688e-01, 8.95049140e-01, 1.89510519e-01],
 [9.23530053e-01, 2.11387447e-01, 1.50311612e-01, 9.56452748e-01]],

[[1.61541980e-01, 3.10915162e-01, 5.54130710e-01, 8.60379840e-01],
 [7.73872215e-01, 4.06372720e-01, 1.10886515e-01, 8.32276349e-01],
 [8.61125415e-01, 5.51340093e-01, 5.06481475e-01, 1.54025080e-01],
 [7.11632788e-02, 4.36354006e-01, 9.38463708e-01, 7.61540273e-01],
 [1.61114039e-01, 4.70359539e-01, 9.60497083e-02, 6.95737406e-01],
 [7.38287985e-01, 7.03022768e-02, 9.95854530e-03, 7.82634398e-01],
 [6.10136986e-01, 6.14067347e-01, 5.21717238e-01, 3.43448886e-01],
 [6.55019874e-02, 1.05923807e-01, 2.10206277e-01, 8.64405578e-01],
 [6.34441742e-01, 3.24146208e-01, 9.73842321e-01, 3.61872251e-01],
 [6.54436882e-01, 8.53000543e-01, 3.26792885e-01, 7.81549219e-01],
 [9.72736083e-02, 1.62301999e-01, 1.78433769e-01, 4.70511800e-01],
 [4.12242796e-01, 2.11323612e-01, 9.87971556e-01, 9.89530121e-01],
 [2.57834499e-01, 7.01059162e-01, 6.04004812e-01, 3.88635659e-01],
 [7.31355443e-01, 8.20187488e-01, 5.73279000e-01, 2.44661838e-01],
 [8.75807311e-01, 5.91191947e-01, 4.37461335e-01, 3.31704799e-01],
 [9.63492647e-01, 4.55523948e-01, 5.58339256e-01, 7.78082801e-01],
 [5.20730648e-01, 6.71916458e-01, 3.54385855e-01, 4.73383848e-01],
 [9.34345046e-01, 4.94358220e-01, 7.09039422e-01, 2.14988911e-01],
 [7.19298099e-01, 9.54592887e-01, 8.80027843e-01, 5.66213828e-01],
 [7.47983016e-01, 5.32050365e-01, 5.25465055e-01, 3.80920504e-01]],

[[5.02826230e-01, 4.37491158e-02, 8.60295678e-01, 2.47053705e-01],
 [3.10672619e-01, 1.09377266e-01, 7.93718976e-01, 1.19825596e-01],
 [3.32213377e-01, 2.10096996e-02, 8.04767446e-01, 5.90436854e-01],
 [4.49516046e-01, 2.49965094e-01, 9.87144601e-01, 1.28039193e-01],
 [4.95359168e-01, 7.74581474e-01, 8.96262300e-03, 7.31721143e-01],
 [4.89253196e-01, 5.85644306e-01, 8.40098984e-01, 4.70830342e-04],
 [3.04243684e-01, 5.28714304e-01, 1.45589415e-01, 3.98920573e-01],
```

```
 [5.11018969e-01, 7.21009876e-01, 2.45417233e-01, 8.07591450e-01],
 [2.18977606e-01, 1.04265390e-02, 5.02835917e-01, 1.88680854e-01],
 [6.25891560e-01, 9.97356178e-01, 3.07292056e-01, 4.76830045e-01],
 [2.98805207e-01, 1.68312941e-01, 5.46229407e-01, 5.22880282e-01],
 [7.55792365e-01, 8.33970005e-02, 7.14174958e-01, 7.75868053e-01],
 [3.45887185e-01, 7.36449029e-01, 7.96291406e-01, 7.55833762e-01],
 [8.82770877e-01, 2.42272078e-01, 8.88583395e-02, 8.77674874e-01],
 [5.12971842e-01, 5.76511673e-01, 8.96704676e-01, 5.71798508e-01],
 [8.96835426e-01, 5.66908463e-01, 5.70505151e-01, 4.49883281e-01],
 [3.77109080e-01, 2.12259116e-01, 5.71074113e-02, 2.44351751e-01],
 [8.19029577e-01, 6.05218171e-01, 7.61694726e-01, 6.95472620e-02],
 [2.52687038e-01, 6.15864285e-01, 6.96986698e-01, 9.92102131e-01],
 [4.86995203e-01, 7.17283317e-01, 2.77780529e-01, 4.25370169e-02]],

[[9.33641934e-01, 7.83378822e-01, 8.39415148e-01, 9.64435480e-01],
 [7.66803170e-01, 1.57978971e-01, 2.37584025e-02, 4.83111272e-01],
 [2.99068308e-01, 3.41715931e-01, 4.11616358e-01, 2.80389741e-01],
 [5.18480148e-01, 5.76311992e-01, 2.07378828e-01, 5.75868647e-01],
 [2.30227905e-01, 8.57298091e-01, 5.49219804e-02, 4.07690712e-01],
 [3.24258044e-01, 2.96621848e-01, 3.53727507e-01, 5.77228819e-01],
 [6.20444374e-01, 3.82483940e-01, 5.07812402e-01, 8.82602136e-01],
 [1.68560722e-01, 4.65630876e-01, 1.48909616e-01, 5.59224999e-01],
 [2.17227590e-01, 6.39004292e-01, 4.90514095e-01, 1.84561144e-01],
 [2.58465302e-01, 8.50246120e-01, 3.85089148e-02, 1.50948537e-01],
 [3.55556915e-01, 8.68483028e-01, 8.18417124e-01, 8.10303840e-02],
 [4.62339366e-01, 6.13981971e-01, 3.02287960e-01, 6.25812939e-01],
 [4.10403299e-01, 2.24744235e-01, 7.73940568e-01, 3.06345289e-01],
 [9.82181010e-01, 8.52429397e-01, 2.57254782e-01, 4.48043986e-01],
 [7.45667776e-01, 6.18117606e-01, 2.55440997e-01, 6.45350975e-01],
 [4.42338216e-01, 7.33591507e-01, 1.98254027e-01, 8.97261407e-01],
 [8.92774624e-01, 7.32943152e-01, 8.69957652e-01, 7.92583503e-01],
 [1.33342708e-01, 3.14162445e-01, 5.96529132e-01, 1.97395299e-01],
 [8.87663449e-01, 1.36922413e-01, 6.21283614e-01, 5.14435182e-01],
 [6.49603758e-01, 4.84801887e-01, 1.07924304e-01, 3.74817474e-01]],

[[4.92526109e-01, 8.73709066e-01, 8.35623213e-01, 1.80786997e-01],
 [4.81131999e-01, 7.21274796e-01, 3.16074634e-03, 2.77065073e-01],
 [9.45597382e-01, 2.49349476e-01, 5.71802610e-01, 2.57444334e-03],
 [6.69202922e-01, 4.73769390e-01, 9.26750414e-01, 5.67632795e-01],
 [2.08831455e-01, 3.01899518e-02, 6.34460949e-01, 5.29215610e-01],
 [7.56280354e-01, 9.61882097e-01, 4.52173001e-01, 5.28791845e-01],
 [9.01599953e-01, 2.36737467e-01, 9.26425408e-01, 2.65715845e-01],
 [2.50415213e-01, 9.29333009e-01, 1.79324537e-01, 6.24542981e-01],
 [2.55285110e-01, 8.60741148e-01, 9.29944252e-01, 3.59684702e-01],
 [6.90976874e-01, 4.01945313e-01, 8.76278757e-01, 1.06875511e-01],
 [4.25149620e-01, 7.03871110e-01, 3.33124480e-01, 8.06107041e-01],
 [8.46214952e-02, 6.03971083e-01, 7.65705726e-01, 3.05558640e-01],
 [8.31690713e-01, 8.55067173e-02, 8.50980694e-01, 3.84151259e-01],
 [5.23444443e-01, 9.51895705e-01, 1.35639156e-01, 9.65126835e-01],
 [4.00587597e-01, 8.68895464e-01, 6.27423200e-01, 2.26352473e-01],
 [7.98443107e-01, 5.20312856e-01, 5.39855804e-01, 3.01205855e-01],
 [9.59396931e-01, 8.47526532e-01, 8.05997492e-01, 7.36792654e-01],
 [8.31805989e-01, 5.71135356e-01, 1.31982155e-01, 1.94998631e-01],
 [3.13007322e-01, 2.13181050e-01, 3.77030657e-01, 8.32734162e-01],
 [6.80808631e-02, 5.85812487e-01, 5.55356966e-01, 7.12377544e-02]],
```

```
[[8.52907745e-01, 4.13099266e-01, 6.70622712e-01, 7.76384817e-01],
 [5.47257106e-01, 4.14037161e-01, 5.47525636e-01, 2.78264925e-01],
 [8.01683377e-03, 9.89203518e-01, 2.41246150e-01, 3.37574337e-01],
 [3.30620260e-01, 1.03750345e-01, 9.76945357e-01, 7.13831407e-02],
 [8.07397400e-01, 9.56838436e-01, 7.80173703e-01, 5.21591428e-01],
 [7.54675348e-01, 9.20680078e-01, 7.98413177e-01, 5.37740097e-01],
 [4.66905868e-01, 4.21853324e-01, 7.43197588e-01, 8.95714087e-01],
 [7.44273665e-01, 9.17111553e-01, 6.45982719e-01, 3.75143213e-01],
 [5.37497507e-01, 5.44510390e-01, 5.76147667e-02, 6.46829745e-01],
 [2.55132606e-01, 6.19884676e-02, 8.37688359e-01, 4.04452160e-01],
 [2.99920488e-01, 4.20676606e-01, 3.93041458e-01, 2.55286972e-01],
 [4.57187716e-01, 9.89721266e-02, 1.50798175e-01, 4.50114172e-01],
 [3.21705943e-01, 5.88196414e-02, 7.67318792e-02, 8.46325200e-01],
 [2.14514661e-01, 1.29303220e-01, 6.40745675e-02, 2.07835367e-01],
 [5.24574870e-01, 1.66834879e-01, 7.78281101e-01, 8.31305594e-01],
 [6.79557881e-01, 2.20703745e-01, 3.17573096e-01, 7.79307136e-01],
 [3.42605762e-02, 2.12522058e-01, 4.21301282e-01, 5.70332427e-01],
 [4.12866213e-01, 6.43836738e-01, 5.87112414e-01, 7.32029460e-01],
 [8.21218444e-01, 6.11431269e-01, 2.56885949e-02, 5.90366818e-02],
 [7.30642182e-01, 9.71930575e-01, 1.21952158e-01, 3.99731972e-01]],

[[1.27492853e-01, 1.28102603e-01, 9.44481465e-01, 9.05867247e-02],
 [4.05393316e-01, 5.50599479e-01, 5.83614674e-01, 2.00635056e-01],
 [8.28087216e-02, 7.22432059e-01, 7.54844052e-01, 5.88994314e-01],
 [1.57539072e-01, 4.69828972e-01, 8.56589713e-01, 1.92530278e-01],
 [9.10421465e-01, 5.92367964e-01, 5.75569309e-01, 4.17905244e-02],
 [6.66296680e-01, 1.22126752e-01, 7.49530469e-01, 3.21345918e-01],
 [7.11333044e-01, 4.79145825e-01, 5.62681472e-01, 4.61208501e-01],
 [6.76965907e-01, 5.45091362e-02, 3.29536773e-01, 7.00404737e-01],
 [3.56637985e-01, 6.03895437e-03, 7.10192860e-01, 9.45708155e-01],
 [3.30428601e-02, 6.54571286e-01, 2.17621803e-02, 3.08396176e-01],
 [2.69897553e-02, 4.20719360e-01, 2.29252046e-01, 5.07638687e-01],
 [5.47407086e-01, 4.35345776e-01, 3.37355411e-01, 9.82175766e-01],
 [1.78496914e-01, 7.71082992e-01, 2.95437924e-01, 4.66200544e-01],
 [4.77971601e-01, 3.35974305e-01, 4.08194088e-01, 4.57314042e-01],
 [5.74984451e-02, 4.68822829e-01, 6.62559582e-01, 5.45491042e-01],
 [6.73445740e-01, 2.04540220e-01, 2.76617387e-01, 5.11633149e-01],
 [9.99951203e-01, 7.44049345e-01, 2.38292340e-01, 3.32389499e-01],
 [3.09013273e-02, 6.63855269e-02, 1.77469003e-01, 6.05886043e-01],
 [5.39888425e-01, 7.47579597e-01, 1.13532718e-01, 4.10642959e-01],
 [9.58978483e-01, 8.52953553e-01, 2.44312199e-01, 3.40480077e-01]]])
```

In [165…

```
np.random.rand(5,10,2)
```

Out[165…

```
array([[[0.38196376, 0.45476226],
        [0.22050507, 0.00430697],
        [0.55795112, 0.92279371],
        [0.80434398, 0.82190316],
        [0.05413666, 0.12282293],
        [0.10675949, 0.75911569],
        [0.19184745, 0.28968224],
        [0.93443283, 0.09943633],
        [0.86505461, 0.14545757],
        [0.64050887, 0.15564891]],

       [[0.28065866, 0.65012512],
        [0.17843499, 0.94272897],
        [0.34656648, 0.10530978],
        [0.42683735, 0.12867929],
        [0.6857879 , 0.84342297],
        [0.67281951, 0.04075182],
        [0.48297866, 0.06978274],
        [0.68485568, 0.63689911],
        [0.89743254, 0.1339945 ],
        [0.43020616, 0.36887652]],

       [[0.34559226, 0.42082468],
        [0.01043233, 0.61095376],
        [0.48334932, 0.53343097],
        [0.79907358, 0.9534503 ],
        [0.5077404 , 0.49842857],
        [0.27600303, 0.62113306],
        [0.89722991, 0.45567341],
        [0.78077146, 0.25083324],
        [0.70664077, 0.24009685],
        [0.09359468, 0.69505817]],

       [[0.80521427, 0.72615032],
        [0.96731359, 0.05462001],
        [0.0510273 , 0.43228725],
        [0.12454155, 0.58319538],
        [0.08960619, 0.89956407],
        [0.20815173, 0.25831946],
        [0.99747126, 0.02510022],
        [0.58802091, 0.26851062],
        [0.76930637, 0.37106799],
        [0.02919376, 0.90797448]],

       [[0.78676752, 0.12273713],
        [0.4220999 , 0.42589863],
        [0.95098406, 0.9781636 ],
        [0.01560849, 0.81480549],
        [0.37578226, 0.54959959],
        [0.48981459, 0.4523072 ],
        [0.63998178, 0.13890299],
        [0.27419195, 0.58558447],
        [0.45611427, 0.06131985],
        [0.59009324, 0.11630039]]])
```

linespace

In [168…
```
np.linspace(2,8,2)
```

Out[168…
```
array([2., 8.])
```

In [170…
```
np.linspace(2,10,2)
```

Out[170…
```
array([ 2., 10.])
```

In [172…
```
np.linspace(10,20,5)
```

Out[172…
```
array([10. , 12.5, 15. , 17.5, 20. ])
```

In [174…
```
np.linspace(20,40,5)
```

Out[174…
```
array([20., 25., 30., 35., 40.])
```

In [176…
```
np.linspace(10,20)
```

Out[176…
```
array([10.        , 10.20408163, 10.40816327, 10.6122449 , 10.81632653,
       11.02040816, 11.2244898 , 11.42857143, 11.63265306, 11.83673469,
       12.04081633, 12.24489796, 12.44897959, 12.65306122, 12.85714286,
       13.06122449, 13.26530612, 13.46938776, 13.67346939, 13.87755102,
       14.08163265, 14.28571429, 14.48979592, 14.69387755, 14.89795918,
       15.10204082, 15.30612245, 15.51020408, 15.71428571, 15.91836735,
       16.12244898, 16.32653061, 16.53061224, 16.73469388, 16.93877551,
       17.14285714, 17.34693878, 17.55102041, 17.75510204, 17.95918367,
       18.16326531, 18.36734694, 18.57142857, 18.7755102 , 18.97959184,
       19.18367347, 19.3877551 , 19.59183673, 19.79591837, 20.        ])
```

identity matrix

In [179…
```
np.identity(5)
```

Out[179…
```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

In [181…
```
np.identity(3)
```

Out[181…
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [183…
```
np.identity(4,dtype=bool)
```

Out[183…
```
array([[ True, False, False, False],
       [False,  True, False, False],
       [False, False,  True, False],
       [False, False, False,  True]])
```

In [185…
```
np.identity(4,dtype=int)
```

Out[185…
```
array([[1, 0, 0, 0],
       [0, 1, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]])
```

array attributes

In [188…
```
a1=np.array([1,2,3,4,5])
a1
```

Out[188…
```
array([1, 2, 3, 4, 5])
```

In [190…
```
a2=np.arange(1,10).reshape(3,3)
```

In [192…
```
a2
```

Out[192…
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [194…
```
a3=np.arange(2,20).reshape(6,3)
a3
```

Out[194…
```
array([[ 2,  3,  4],
       [ 5,  6,  7],
       [ 8,  9, 10],
       [11, 12, 13],
       [14, 15, 16],
       [17, 18, 19]])
```

In [196…
```
np.arange(10,20).reshape(2,5)
```

Out[196…
```
array([[10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

In [198…
```
np.arange(10).reshape(5,2)
```

Out[198…
```
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

ndim-no.of dimension

In [201…
```
a1.ndim
```

Out[201…
```
1
```

In [203…
```
a2.ndim
```

Out[203…
```
2
```

In [205…
```
a3.ndim
```

Out[205…
```
2
```

In [207…
```
a4=np.arange(27).reshape(3,3,3)
a4
```

Out[207…
```
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]]])
```

In [209…
```
a4.ndim
```

Out[209…
```
3
```

shape-no.of rows n columns

In [212…
```
a1.shape
```

Out[212…
```
(5,)
```

In [214…
```
a2.shape
```

Out[214…
```
(3, 3)
```

In [216…
```
a3.shape
```

Out[216…
```
(6, 3)
```

In [218…
```
a4.shape
```

Out[218…
```
(3, 3, 3)
```

size-no. of elements

In [221…
```
a1.size
```

Out[221…
```
5
```

In [223…
```
a2
```

Out[223…
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [225…
```
a2.size
```

Out[225…
```
9
```

In [227…
```
a3
```

Out[227…
```
array([[ 2,  3,  4],
       [ 5,  6,  7],
       [ 8,  9, 10],
       [11, 12, 13],
       [14, 15, 16],
       [17, 18, 19]])
```

In [229…
```
a3.size
```

Out[229…
```
18
```

In [231…
```
a4
```

Out[231…
```
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]]])
```

In [233…
```
a4.size
```

Out[233…
```
27
```

itemsize-no of bytws occupied

In [236…
```
a1.itemsize
```

Out[236…
```
4
```

In [238…
```
a2.itemsize
```

Out[238…
```
4
```

In [240…
```
a3.itemsize
```

Out[240…
```
4
```

In [242…
```
a4
```

Out[242…
```
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]]])
```

In [244…
```
a4.itemsize
```

Out[244…
```
4
```

dtype-which type of datatype

In [247…
```
a1.dtype
```

Out[247…
```
dtype('int32')
```

In [249…
```
print(a1.dtype)
```

```
int32
```

In [251…
```
a2.dtype
```

Out[251…
```
dtype('int32')
```

In [253…
```
s.dtype
```

Out[253…
```
dtype('bool')
```

changing datatype-astype

In [258…
```
x=np.array([1.0,2.3,4.6,5.5])
x
```

Out[258… array([1. , 2.3, 4.6, 5.5])

In [262…
```
x.astype(int)
```

Out[262… array([1, 2, 4, 5])

In [266…
```
x1=np.array([4.9,5.0,9.9])
x1
```

Out[266… array([4.9, 5. , 9.9])

In [268…
```
x1.astype(int)
```

Out[268… array([4, 5, 9])

In [270…
```
x2=np.array([1,2,4,5])
x2
```

Out[270… array([1, 2, 4, 5])

In [274…
```
x2.astype(float)
```

Out[274… array([1., 2., 4., 5.])

ARRAY OPERATIONS

In [277…
```
z1=np.arange(10,20).reshape(5,2)
z1
```

Out[277…
```
array([[10, 11],
       [12, 13],
       [14, 15],
       [16, 17],
       [18, 19]])
```

In [283…
```
z2=np.arange(12,20).reshape(4,2)
z2
```

Out[283…
```
array([[12, 13],
       [14, 15],
       [16, 17],
       [18, 19]])
```

In [285…
```
z1+4 #addition
```

Out[285…
```
array([[14, 15],
       [16, 17],
       [18, 19],
       [20, 21],
       [22, 23]])
```

In [287…
```
z1-2 #substraction
```

Out[287…
```
array([[ 8,  9],
       [10, 11],
       [12, 13],
       [14, 15],
       [16, 17]])
```

In [289…
```
z1*2 #multiplication
```

Out[289…
```
array([[20, 22],
       [24, 26],
       [28, 30],
       [32, 34],
       [36, 38]])
```

In [291…
```
z1/2 #division
```

Out[291…
```
array([[5. , 5.5],
       [6. , 6.5],
       [7. , 7.5],
       [8. , 8.5],
       [9. , 9.5]])
```

In [293…
```
z1%2 #module
```

Out[293…
```
array([[0, 1],
       [0, 1],
       [0, 1],
       [0, 1],
       [0, 1]], dtype=int32)
```

In [295…
```
z1**2 #power
```

Out[295…
```
array([[100, 121],
       [144, 169],
       [196, 225],
       [256, 289],
       [324, 361]])
```

In [297…
```
z1
```

Out[297…
```
array([[10, 11],
       [12, 13],
       [14, 15],
       [16, 17],
       [18, 19]])
```

Relational Operator

In [300…
```
z2
```

Out[300…
```
array([[12, 13],
       [14, 15],
       [16, 17],
       [18, 19]])
```

In [302…
```
z2>20
```

Out[302…
```
array([[False, False],
       [False, False],
       [False, False],
       [False, False]])
```

In [304…
```
z2<20
```

Out[304…
```
array([[ True,  True],
       [ True,  True],
       [ True,  True],
       [ True,  True]])
```

In [306…
```
z2>15
```

Out[306…
```
array([[False, False],
       [False, False],
       [ True,  True],
       [ True,  True]])
```

In [308…
```
z2!=20
```

Out[308…
```
array([[ True,  True],
       [ True,  True],
       [ True,  True],
       [ True,  True]])
```

In [310…
```
z2==15
```

Out[310…
```
array([[False, False],
       [False,  True],
       [False, False],
       [False, False]])
```

VECTOR OPERATION

In [313…
```
y1=np.arange(1,11).reshape(5,2) #y1ij=y2ij
y1
```

Out[313…
```
array([[ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8],
       [ 9, 10]])
```

In [315…
```
y2=np.arange(10,20).reshape(5,2)
y2
```

Out[315…
```
array([[10, 11],
       [12, 13],
       [14, 15],
       [16, 17],
       [18, 19]])
```

In [317…
```
y1+y2 # vector addition
```

Out[317…
```
array([[11, 13],
       [15, 17],
       [19, 21],
       [23, 25],
       [27, 29]])
```

In [319…
```
y1-2
```

Out[319…
```
array([[-1,  0],
       [ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8]])
```

In [321…
```
y2*2
```

Out[321…
```
array([[20, 22],
       [24, 26],
       [28, 30],
       [32, 34],
       [36, 38]])
```

In [323…
```
y1//2
```

Out[323…
```
array([[0, 1],
       [1, 2],
       [2, 3],
       [3, 4],
       [4, 5]], dtype=int32)
```

In [325…
```
y1/2
```

Out[325…
```
array([[0.5, 1. ],
       [1.5, 2. ],
       [2.5, 3. ],
       [3.5, 4. ],
       [4.5, 5. ]])
```

In [327…
```
y2**2
```

Out[327…
```
array([[100, 121],
       [144, 169],
       [196, 225],
       [256, 289],
       [324, 361]])
```

In [329…
```
y1-y2
```

Out[329…
```
array([[-9, -9],
       [-9, -9],
       [-9, -9],
       [-9, -9],
       [-9, -9]])
```

In [331…
```
y1*y2
```

Out[331…
```
array([[ 10,  22],
       [ 36,  52],
       [ 70,  90],
       [112, 136],
       [162, 190]])
```

In [333…
```
y1/y2
```

Out[333…
```
array([[0.1       , 0.18181818],
       [0.25      , 0.30769231],
       [0.35714286, 0.4       ],
       [0.4375    , 0.47058824],
       [0.5       , 0.52631579]])
```

In [335…
```
y1//y2
```

Out[335…
```
array([[0, 0],
       [0, 0],
       [0, 0],
       [0, 0],
       [0, 0]])
```

In [337…
```
y1**y2
```

Out[337…
```
array([[          1,        2048],
       [     531441,    67108864],
       [ 1808548329,  2033549312],
       [-1526366847,           0],
       [-1953380655, -1981284352]])
```

In [339…
```
y1*y2
```

Out[339…
```
array([[ 10,  22],
       [ 36,  52],
       [ 70,  90],
       [112, 136],
       [162, 190]])
```

ARRAY FUNCTIONS

In [342…
```
k=np.random.rand(3,3)
k
```

Out[342…
```
array([[0.14472802, 0.76659897, 0.67303248],
       [0.91057155, 0.23641063, 0.78498103],
       [0.31272743, 0.39516137, 0.25665455]])
```

In [344…
```
k=np.round(k*10)
k
```

Out[344…
```
array([[1., 8., 7.],
       [9., 2., 8.],
       [3., 4., 3.]])
```

In [346…
```
k.max
```

Out[346…
```
<function ndarray.max>
```

In [348…
```
k.max()
```

Out[348…
```
9.0
```

In [350…
```
k.min()
```

Out[350…
```
1.0
```

In [352…
```
k.mean()
```

Out[352…
```
5.0
```

In [354…
```
k.sum()
```

Out[354…
```
45.0
```

In [358…
```
k.argmax() #indices of max values
```

Out[358…
```
3
```

In [360…
```
k.argmin() #index of min values
```

Out[360…
```
0
```

In [362…
```
k.prod()
```

Out[362…
```
290304.0
```

In [ ]: