NUPPY ARRAYS

In [1]:
```
#NumPy is a popular Python library for numerical computing that provides support for arrays
```

In [2]:
```
#creating numpy/n-d arrays

import numpy as np
```

In [3]:
```
arr1= np.array([1,2,3,4,5])
arr1
```

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

In [4]:
```
type(arr1)
```

Out[4]:
```
numpy.ndarray
```

In [5]:
```
arr2=np.array([[1,2,3,],[2,3,4]])
arr2
```

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

In [6]:
```
arr3 =np.zeros((2,3))
arr3
```

Out[6]:
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [7]:
```
arr4 = np.ones((3,3))
arr4
```

Out[7]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

In [8]:
```
arr5 =np.identity(5)
arr5
```

Out[8]:
```
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 [9]:
```
arr6 = np.arange(10)
arr6
```

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

In [10]:
```
arr7 = np.arange(5,16)
arr7
```

Out[10]:
```
array([ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [11]:
```
type(arr7)
```

Out[11]:
```
numpy.ndarray
```

In [12]:
```
arr7.shape
```

Out[12]:
```
(11,)
```

In [13]:
```
arr8 = np.linspace(10,20,10)
arr8
```

Out[13]:
```
array([10.        , 11.11111111, 12.22222222, 13.33333333, 14.44444444,
       15.55555556, 16.66666667, 17.77777778, 18.88888889, 20.        ])
```

In [14]:
```
arr9 = arr8.copy()
arr9
```

Out[14]:
```
array([10.        , 11.11111111, 12.22222222, 13.33333333, 14.44444444,
       15.55555556, 16.66666667, 17.77777778, 18.88888889, 20.        ])
```

In [15]:
```
arr10 = np.array([[[1,2],[3,4]],[[5,6],[7,8]]]) #3 dimensional matrix
arr10
```

Out[15]:
```
array([[[1, 2],
        [3, 4]],

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

In [16]:
```
arr10.shape
```

Out[16]:
```
(2, 2, 2)
```

In [17]:
```
arr10.ndim
```

Out[17]:
```
3
```

In [18]:
```
arr2
```

Out[18]:
```
array([[1, 2, 3],
       [2, 3, 4]])
```

In [19]:
```
arr2.ndim
```

Out[19]:
```
2
```

In [20]:
```
arr1
```

Out[20]:
```
array([1, 2, 3, 4, 5])
```

In [21]:
```
arr1.ndim
```

Out[21]:
```
1
```

In [22]:
```
#size
arr1.size #numeber of item
```

Out[22]:
```
5
```

In [23]:
```
arr10.size
```

Out[23]:
```
8
```

In [24]:
```
arr10.itemsize #4 bytes memory occupy
```

Out[24]:
```
4
```

In [25]:
```
arr9.itemsize #because it is float
```

Out[25]:
```
8
```

In [26]:
```
arr9.dtype
```

Out[26]:
```
dtype('float64')
```

In [27]:
```
arr10.dtype
```

Out[27]:
```
dtype('int32')
```

In [28]:
```
arr10.astype('float')
```

Out[28]:
```
array([[[1., 2.],
        [3., 4.]],

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

In [29]:
```
arr9.astype('int')
```

Out[29]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 20])
```

In [30]:
```
# numpy contain less memory

lista=range(100)
arr11=np.arange(100)
```

In [31]:
```
import sys
```

In [32]:
```
print(sys.getsizeof(87)*len(lista))
```

```
2800
```

In [33]:
```
print(arr11.itemsize*arr11.size)
```

```
400
```

In [34]:
```
# numpy arrays as faster in comparision to list

import time
```

In [35]:
```
x=range(10000000)
y=range(10000000,20000000)

start_time =time.time()

c=[(x,y) for x,y in zip(x,y)]

print(time.time()-start_time)
```

```
2.2911977767944336
```

In [36]:
```
a =np.arange(10000000)
b = np.arange(10000000,20000000)


start_time =time.time()

c=a+b

print(time.time() - start_time)
```

```
0.47530627250671387
```

In [37]:
```
# indexing slicing and iteration

arr12=np.arange(24)
arr12
```

Out[37]:
```
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])
```

In [38]:
```
arr12 = np.arange(24).reshape(6,4)  #reshape function is used for change the shape of func
arr12
```

Out[38]:
```
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]])
```

In [39]:
```
arr1
```

Out[39]:
```
array([1, 2, 3, 4, 5])
```

In [40]:
```
arr1[3]
```

Out[40]:
```
4
```

In [41]:
```
arr1[2:4]
```

Out[41]:
```
array([3, 4])
```

In [42]:
```
arr1[-1]
```

Out[42]:
```
5
```

In [43]:
```
arr12
```

Out[43]:
```
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]])
```

In [44]:
```
arr12[2]
```

Out[44]:
```
array([ 8,  9, 10, 11])
```

In [45]:
```
arr12[-1]
```

Out[45]:
```
array([20, 21, 22, 23])
```

In [46]:
```
arr12[:2]
```

Out[46]:
```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

In [47]:
```
arr12[:,1:2]
```

Out[47]:
```
array([[ 1],
       [ 5],
       [ 9],
       [13],
       [17],
       [21]])
```

In [48]:
```
arr12[:,1:3]
```

Out[48]:
```
array([[ 1,  2],
       [ 5,  6],
       [ 9, 10],
       [13, 14],
       [17, 18],
       [21, 22]])
```

In [49]:
```
arr12[2:4 ,1:3]
```

Out[49]:
```
array([[ 9, 10],
       [13, 14]])
```

In [50]:
```
# iteration

arr12
```

Out[50]:
```
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]])
```

In [51]:
```
for i in arr12:
    print(i)
```

```
[0 1 2 3]
[4 5 6 7]
[ 8  9 10 11]
[12 13 14 15]
[16 17 18 19]
[20 21 22 23]
```

In [52]:
```
for i in np.nditer(arr12):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

In [53]:
```
#Creating Arrays:

#numpy.array(): Create an array from a list or tuple.
#numpy.arange(): Create an array with evenly spaced values within a given interval.
#numpy.zeros(): Create an array filled with zeros.
#numpy.ones(): Create an array filled with ones.
#numpy.random.rand(): Create an array of random values from a uniform distribution.
```

In [54]:
```
#import numpy as np
```

In [55]:
```
arr1=np.array([1,2,3,4,5,6])
arr2=np.array([4,5,6,7,8,9])
```

In [56]:
```
arr1-arr2
```

Out[56]:
```
array([-3, -3, -3, -3, -3, -3])
```

In [57]:
```
arr1*arr2
```

Out[57]:
```
array([ 4, 10, 18, 28, 40, 54])
```

In [58]:
```
arr1*2
```

Out[58]:
```
array([ 2,  4,  6,  8, 10, 12])
```

In [59]:
```
arr2*3
```

Out[59]:
```
array([12, 15, 18, 21, 24, 27])
```

In [60]:
```
arr2>3
```

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

In [61]:
```
arr3=np.arange(6).reshape(2,3)
arr4=np.arange(6,12).reshape(3,2)
```

In [62]:
```
arr3.dot(arr4)
```

Out[62]:
```
array([[ 28,  31],
       [100, 112]])
```

In [63]:
```
arr1.dot(arr2)
```

Out[63]:
```
154
```

In [64]:
```
arr4
```

Out[64]:
```
array([[ 6,  7],
       [ 8,  9],
       [10, 11]])
```

In [65]:
```
arr4.max()
```

Out[65]:
```
11
```

In [66]:
```
arr4.min()
```

Out[66]:
```
6
```

In [67]:
```
arr4
```

Out[67]:
```
array([[ 6,  7],
       [ 8,  9],
       [10, 11]])
```

In [68]:
```
arr4.min(axis=0)
```

Out[68]:
```
array([6, 7])
```

In [69]:
```
arr4.max(axis=0)
```

Out[69]:
```
array([10, 11])
```

In [70]:
```
arr4.min(axis=1)
```

Out[70]:
```
array([ 6,  8, 10])
```

In [71]:
```
arr4.max(axis=1)
```

Out[71]:
```
array([ 7,  9, 11])
```

In [72]:
```
arr4.sum()
```

Out[72]:
```
51
```

In [73]:
```
arr4.sum(axis=0)
```

Out[73]:
```
array([24, 27])
```

In [74]:
```
arr4.mean()
```

Out[74]:
```
8.5
```

In [75]:
```
arr4.std()
```

Out[75]:
```
1.707825127659933
```

In [76]:
```
np.sin(arr4)
```

Out[76]:
```
array([[-0.2794155 ,  0.6569866 ],
       [ 0.98935825,  0.41211849],
       [-0.54402111, -0.99999021]])
```

In [77]:
```
np.median(arr4)
```

Out[77]:
```
8.5
```

In [78]:
```
np.exp(arr4)         #expoliate
```

Out[78]:
```
array([[  403.42879349,  1096.63315843],
       [ 2980.95798704,  8103.08392758],
       [22026.46579481, 59874.1417152 ]])
```

In [79]:
```
#reshaping numpy array

arr4
```

Out[79]:
```
array([[ 6,  7],
       [ 8,  9],
       [10, 11]])
```

In [80]:
```
arr4.ndim
```

Out[80]:
```
2
```

In [81]:
```
arr4.ravel()
```

Out[81]:
```
array([ 6,  7,  8,  9, 10, 11])
```

In [82]:
```
#transpose

arr4
```

Out[82]:
```
array([[ 6,  7],
       [ 8,  9],
       [10, 11]])
```

In [83]:
```
arr4.transpose()      #row is converted to column ,column converted to row
```

Out[83]:
```
array([[ 6,  8, 10],
       [ 7,  9, 11]])
```

In [84]:
```
#stacking

#stacking is stage that we combine two arrays

arr3
```

Out[84]:
```
array([[0, 1, 2],
       [3, 4, 5]])
```

In [85]:
```
arr5=np.arange(12,18).reshape(2,3)
```

In [86]:
```
arr5
```

Out[86]:
```
array([[12, 13, 14],
       [15, 16, 17]])
```

In [87]:
```
np.hstack((arr3,arr5))       #horizontal stacking
```

Out[87]:
```
array([[ 0,  1,  2, 12, 13, 14],
       [ 3,  4,  5, 15, 16, 17]])
```

In [88]:
```
np.vstack((arr3,arr5))        #vertical stacking
```

Out[88]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [12, 13, 14],
       [15, 16, 17]])
```

In [89]:
```
#spliting

np.hsplit(arr3,3)           #horizontal split
```

Out[89]:
```
[array([[0],
        [3]]),
 array([[1],
        [4]]),
 array([[2],
        [5]])]
```

In [90]:
```
np.vsplit(arr3,2)          #vertical split
```

Out[90]:
```
[array([[0, 1, 2]]), array([[3, 4, 5]])]
```

In [91]:
```
#slicing#indexing

arr8=np.arange(24).reshape(6,4)
```

In [92]:
```
arr8
```

Out[92]:
```
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]])
```

In [93]:
```
arr8[[0,2,4]]
```

Out[93]:
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11],
       [16, 17, 18, 19]])
```

In [94]:
```
arr8[[-1,-3,-4]]
```

Out[94]:
```
array([[20, 21, 22, 23],
       [12, 13, 14, 15],
       [ 8,  9, 10, 11]])
```

In [95]:
```
arr =np.random.randint(low=1,high=100,size=20).reshape(4,5)
arr
```

Out[95]:
```
array([[24, 72, 64, 15, 92],
       [93, 47, 37, 83, 81],
       [17, 51, 46, 61, 50],
       [50, 54, 36, 75, 33]])
```

In [96]:
```
arr[0]
```

Out[96]:
```
array([24, 72, 64, 15, 92])
```

In [97]:
```
arr[3]
```

Out[97]:
```
array([50, 54, 36, 75, 33])
```

In [98]:
```
arr>50
```

Out[98]:
```
array([[False,  True,  True, False,  True],
       [ True, False, False,  True,  True],
       [False,  True, False,  True, False],
       [False,  True, False,  True, False]])
```

In [99]:
```
#indexing using boolean array
arr[arr>50]
```

Out[99]:
```
array([72, 64, 92, 93, 83, 81, 51, 61, 54, 75])
```

In [100]:
```
arr[(arr>50) &(arr%2!=0)]
```

Out[100]:
```
array([93, 83, 81, 51, 61, 75])
```

In [101]:
```
arr[(arr>50) & (arr%2!=0)]=0
```

In [102]:
```
arr
```

Out[102]:
```
array([[24, 72, 64, 15, 92],
       [ 0, 47, 37,  0,  0],
       [17,  0, 46,  0, 50],
       [50, 54, 36,  0, 33]])
```

In [103]:
```
#ploting graph using numpy

x=np.linspace(-40,40,100)
```

In [104]:
```
x
```

Out[104]:
```
array([-40.        , -39.19191919, -38.38383838, -37.57575758,
       -36.76767677, -35.95959596, -35.15151515, -34.34343434,
       -33.53535354, -32.72727273, -31.91919192, -31.11111111,
       -30.3030303 , -29.49494949, -28.68686869, -27.87878788,
       -27.07070707, -26.26262626, -25.45454545, -24.64646465,
       -23.83838384, -23.03030303, -22.22222222, -21.41414141,
       -20.60606061, -19.7979798 , -18.98989899, -18.18181818,
       -17.37373737, -16.56565657, -15.75757576, -14.94949495,
       -14.14141414, -13.33333333, -12.52525253, -11.71717172,
       -10.90909091, -10.1010101 ,  -9.29292929,  -8.48484848,
        -7.67676768,  -6.86868687,  -6.06060606,  -5.25252525,
        -4.44444444,  -3.63636364,  -2.82828283,  -2.02020202,
        -1.21212121,  -0.4040404 ,   0.4040404 ,   1.21212121,
         2.02020202,   2.82828283,   3.63636364,   4.44444444,
         5.25252525,   6.06060606,   6.86868687,   7.67676768,
         8.48484848,   9.29292929,  10.1010101 ,  10.90909091,
        11.71717172,  12.52525253,  13.33333333,  14.14141414,
        14.94949495,  15.75757576,  16.56565657,  17.37373737,
        18.18181818,  18.98989899,  19.7979798 ,  20.60606061,
        21.41414141,  22.22222222,  23.03030303,  23.83838384,
        24.64646465,  25.45454545,  26.26262626,  27.07070707,
        27.87878788,  28.68686869,  29.49494949,  30.3030303 ,
        31.11111111,  31.91919192,  32.72727273,  33.53535354,
        34.34343434,  35.15151515,  35.95959596,  36.76767677,
        37.57575758,  38.38383838,  39.19191919,  40.        ])
```

In [105]:
```
x.size
```

Out[105]:
```
100
```

In [106]:
```
y=np.sin(x)
```

In [107]:
```
y
```

Out[107]:
```
array([-0.74511316, -0.9969604 , -0.63246122,  0.12304167,  0.80247705,
        0.98580059,  0.55967698, -0.21245326, -0.85323945, -0.96653119,
       -0.48228862,  0.30011711,  0.89698277,  0.93931073,  0.40093277,
       -0.38531209, -0.93334716, -0.90436313, -0.31627868,  0.46733734,
        0.96203346,  0.86197589,  0.22902277, -0.54551809, -0.9828057 ,
       -0.81249769, -0.13988282,  0.61921119,  0.995493  ,  0.75633557,
        0.04959214, -0.68781042, -0.99999098, -0.69395153,  0.0411065 ,
        0.75075145,  0.99626264,  0.62585878, -0.13146699, -0.8075165 ,
       -0.98433866, -0.55261747,  0.22074597,  0.85763861,  0.96431712,
        0.47483011, -0.30820902, -0.90070545, -0.93636273, -0.39313661,
        0.39313661,  0.93636273,  0.90070545,  0.30820902, -0.47483011,
       -0.96431712, -0.85763861, -0.22074597,  0.55261747,  0.98433866,
        0.8075165 ,  0.13146699, -0.62585878, -0.99626264, -0.75075145,
       -0.0411065 ,  0.69395153,  0.99999098,  0.68781042, -0.04959214,
       -0.75633557, -0.995493  , -0.61921119,  0.13988282,  0.81249769,
        0.9828057 ,  0.54551809, -0.22902277, -0.86197589, -0.96203346,
       -0.46733734,  0.31627868,  0.90436313,  0.93334716,  0.38531209,
       -0.40093277, -0.93931073, -0.89698277, -0.30011711,  0.48228862,
        0.96653119,  0.85323945,  0.21245326, -0.55967698, -0.98580059,
       -0.80247705, -0.12304167,  0.63246122,  0.9969604 ,  0.74511316])
```

In [108]:
```
y.size
```

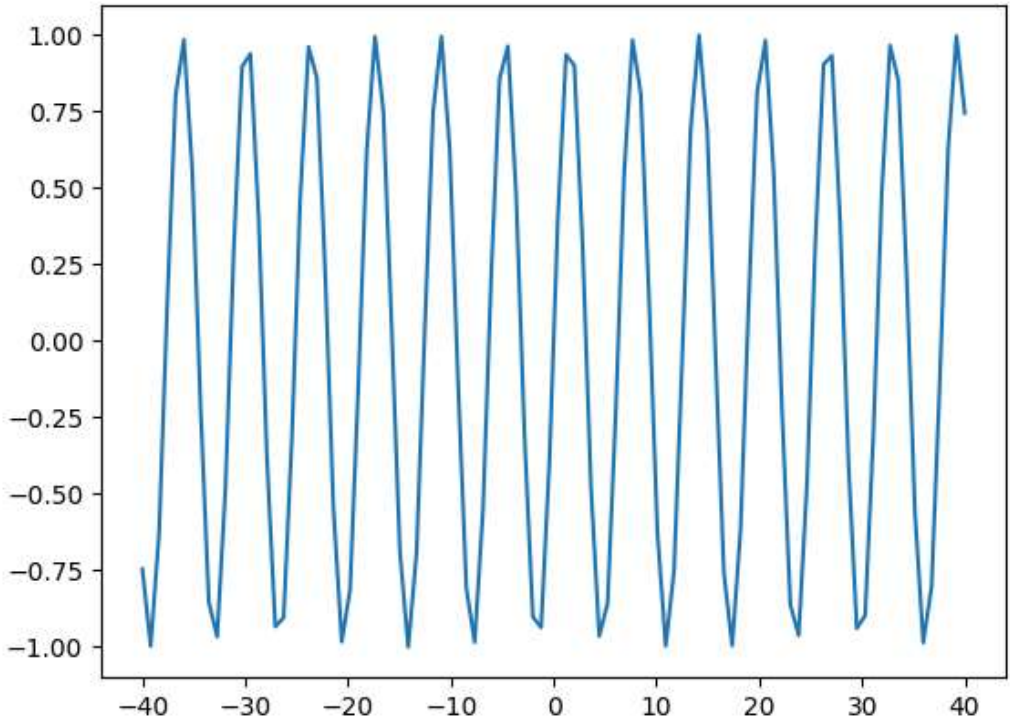
Out[108]:
```
100
```

In [109]:
```
import matplotlib.pyplot as plt

%matplotlib inline
```

In [110]:
```
plt.plot(x,y)
```

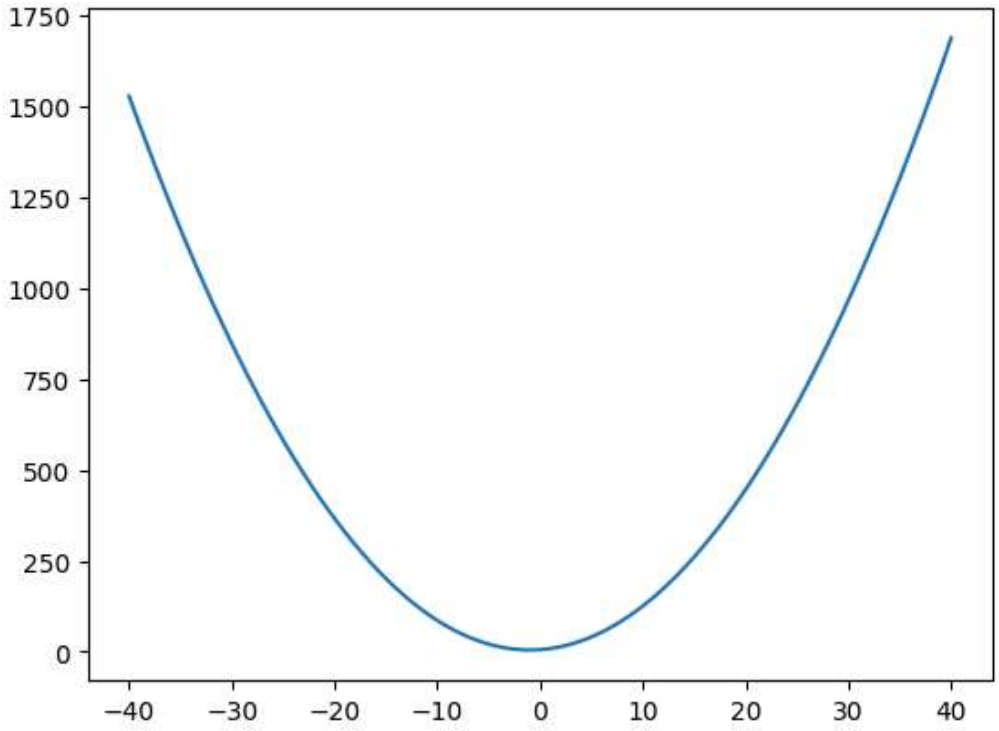
Out[110]:
```
[<matplotlib.lines.Line2D at 0x21019962110>]
```



In [111]:
```
y=x*x+2*x+6
```

In [112]:
```
plt.plot(x,y)
```

Out[112]:
```
[<matplotlib.lines.Line2D at 0x210199c7590>]
```



In [113]:
```
# brodcasting

# it allows arrays of different shapes to be combined together during arithmetic operations
#usually done on corresponding elements
#if two arrays are of exactly the same shape.then these operations are smoomthly performed
#if dimensions of two arrays are dissimilar , then the element-to-element operation are not
#however operations on arrays of non-similar shape is still possible in numpy,because of th
# the smaller array is broadcast to the size of the larger array so that they have compatal


#senario 1

a1=np.arange(8).reshape(2,4)
a2=np.arange(8,16).reshape(2,4)

print(a1)
print(a2)
```

```
[[0 1 2 3]
 [4 5 6 7]]
[[ 8  9 10 11]
 [12 13 14 15]]
```

In [114]:
```
a1+a2
```

Out[114]:
```
array([[ 8, 10, 12, 14],
       [16, 18, 20, 22]])
```

In [115]:
```
#senario 2

a3=np.arange(9).reshape(3,3)
a4=np.arange(3).reshape(1,3)

print(a3,a4)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]] [[0 1 2]]
```

In [116]:
```
a3+a4
```

Out[116]:
```
array([[ 0,  2,  4],
       [ 3,  5,  7],
       [ 6,  8, 10]])
```

In [117]:
```
#rules for broadcasting

#if x=m and y=n operation will take place

a1=np.arange(8).reshape(2,4)
a2=np.arange(8,16).reshape(2,4)

a1+a2
```

Out[117]:
```
array([[ 8, 10, 12, 14],
       [16, 18, 20, 22]])
```

In [118]:
```
# if x=1 and y=n then also operation will take place(same dimension)

a5 = np.arange(3).reshape(1,3)
a6 = np.arange(12).reshape(4,3)

print(a5)
print(a6)
```

```
[[0 1 2]]
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

In [119]:
```
a5+a6
```

Out[119]:
```
array([[ 0,  2,  4],
       [ 3,  5,  7],
       [ 6,  8, 10],
       [ 9, 11, 13]])
```

In [120]:
```
# if y=1 and x=m then also operation will take place , even if
#they are not of the same dimension

a7=np.arange(4).reshape(4,1)
a8=np.arange(12).reshape(4,3)

print(a7)
print(a8)
```

```
[[0]
 [1]
 [2]
 [3]]
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

In [121]:
```
a7+a8
```

Out[121]:
```
array([[ 0,  1,  2],
       [ 4,  5,  6],
       [ 8,  9, 10],
       [12, 13, 14]])
```

In [122]:
```
# if x=1 and y!=n then also operation will not take place

a9=np.arange(3).reshape(1,3)
a10=np.arange(16).reshape(4,4)

print(a9)
print(a10)
```

```
[[0 1 2]]
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

In [123]:
```
a9 + a10                  # value error
                          #operands could not be broadcast together with shapes (1,3) (4,4)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[123], line 1
----> 1 a9 + a10

ValueError: operands could not be broadcast together with shapes (1,3) (4,4)
```

In [124]:
```
#if x=1 and n=1 then y==m , operation to take place

a11=np.arange(3).reshape(1,3)
a12=np.arange(3).reshape(3,1)

print(a11)
print(a12)
```

```
[[0 1 2]]
[[0]
 [1]
 [2]]
```

In [125]:
```
a11+a12
```

Out[125]:
```
array([[0, 1, 2],
       [1, 2, 3],
       [2, 3, 4]])
```

In [126]:
```
# if x=1 and y=1 then the operation will take place no matter what

a13 = np.arange(1).reshape(1,1)
a14 = np.arange(20).reshape(4,5)

print(a13)
print(a14)
```

```
[[0]]
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
```

In [127]:
```
a13+a14
```

Out[127]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

In [128]:
```
# if they are of different dimensions

a15 =np.arange(4)
a16 =np.arange(20).reshape(5,4)

print(a15)
print(a16)
```

```
[0 1 2 3]
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]]
```

In [129]:
```
a15+a16
```

Out[129]:
```
array([[ 0,  2,  4,  6],
       [ 4,  6,  8, 10],
       [ 8, 10, 12, 14],
       [12, 14, 16, 18],
       [16, 18, 20, 22]])
```

In [131]:
```
# various functions in numpy

np.random.random()
```

Out[131]:
```
0.8717755450685758
```

In [134]:
```
np.random.seed(1)     # if we use seed function i getting same random value again and agai
np.random.random()
```

Out[134]:
```
0.417022004702574
```

In [140]:
```
np.random.uniform(3,10)
```

Out[140]:
```
4.303821479643696
```

In [141]:
```
np.random.uniform(1,100,10)
```

Out[141]:
```
array([35.21051198, 40.27997995, 54.34285667, 42.50025693, 68.83673054,
       21.24077272, 87.9336262 ,  3.71137173, 67.37628351, 42.31317543])
```

In [144]:
```
np.random.randint(1,10)
```

Out[144]:
```
8
```

In [148]:
```
np.random.randint(1,10,15).reshape(3,5)
```

Out[148]:
```
array([[7, 4, 8, 8, 5],
       [6, 4, 7, 9, 1],
       [3, 8, 8, 8, 4]])
```

In [152]:
```
a=np.random.randint(1,10,6)
a
```

Out[152]:
```
array([4, 1, 9, 7, 5, 6])
```

In [153]:
```
np.max(a)
```

Out[153]: 9

In [154]:
```
a[np.argmax(a)]
```

Out[154]: 9

In [155]:
```
a[np.argmin(a)]
```

Out[155]: 1

In [156]:
```
np.argmin(2)
```

Out[156]: 0

In [157]:
```
a=np.random.randint(1,10,6)
a
```

Out[157]: array([7, 3, 6, 8, 9, 5])

In [159]:
```
a[a%2==1]=-1
a
```

Out[159]: array([-1, -1, 6, 8, -1, -1])

In [162]:
```
a=np.random.randint(1,50,6)
a
```

Out[162]: array([33, 13, 2, 31, 42, 25])

In [163]:
```
np.where(a%2==1,-1,a)
```

Out[163]: array([-1, -1, 2, -1, 42, -1])

In [164]:
```
a
```

Out[164]: array([33, 13, 2, 31, 42, 25])

In [165]:
```
out=np.where(a%2==1,-1,a)
```

In [166]:
```
out
```

Out[166]: array([-1, -1, 2, -1, 42, -1])

In [168]:
```
a=np.random.randint(1,50,10)
a
```

Out[168]: array([35, 11, 33, 42, 19, 23, 7, 3, 8, 40])

In [169]:
```
a=np.sort(a)
a
```

Out[169]: array([3, 7, 8, 11, 19, 23, 33, 35, 40, 42])

In [170]:
```
np.percentile(a,25)
```

Out[170]: 8.75

In [171]:
```
np.percentile(a,50)
```

Out[171]: 21.0

In [173]:
```
np.percentile(a,99.8)
```

Out[173]: 41.964

In [ ]:
```

```