31-05-2023

In [1]:
```
pip install numpy
```

```
Defaulting to user installation because normal site-packages is not writeableNote:
you may need to restart the kernel to use updated packages.

Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages
(1.21.5)
```

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

In [3]:
```
x=[1,2,3]
x
```

Out[3]:
```
[1, 2, 3]
```

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

Out[4]:
```
list
```

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

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

In [6]:
```
type(x)
```

Out[6]:
```
numpy.ndarray
```

1 D Array

In [7]:
```
a=np.array([1,2,3,4,5,6])
a
```

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

convert list to array

In [8]:
```
a=[1,2,3,4]
arr=np.array(a)
arr
```

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

In [9]:
```
#take user input

a=[]
cnt=1
for i in range(int(input("How many elements you want?"))):
    val=eval(input(f"enter the value {cnt}:"))    #eval is function in python - use
    a.append(val)
    cnt+=1
```

```
b=np.array(a)
b
```

```
How many elements you want?4
enter the value 1:11
enter the value 2:12
enter the value 3:13
enter the value 4:14
```

Out[9]:
```
array([11, 12, 13, 14])
```

2 D Array

In [10]:
```
a=np.array([[1,2,3],[4,5,6]])
a
```

Out[10]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [11]:
```
a.shape    #to find how many rows and columns in my data
```

Out[11]:
```
(2, 3)
```

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

Out[12]:
```
numpy.ndarray
```

In [13]:
```
#to check dimensions

print(np.ndim(a))
```

```
2
```

In [14]:
```
np.ndim(a)
```

Out[14]:
```
2
```

In [15]:
```
#checking datatype values that inside the array
```

In [16]:
```
a.dtype
```

Out[16]:
```
dtype('int32')
```

3 D Array (matrices,rows,columns)

In [17]:
```
n=np.array([[[1,2,3],[4,5,6]],[[7,5,8],[1,4,8]]])
n
```

Out[17]:
```
array([[[1, 2, 3],
        [4, 5, 6]],

       [[7, 5, 8],
        [1, 4, 8]]])
```

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

Out[19]:
```
3
```

In [20]:
```
n.shape
```

Out[20]:
```
(2, 2, 3)
```

In [22]:
```
arr=np.array([[[1,2,3],[4,5,6]],[[7,5,8],[1,4,8]],[[100,200,300],[400,500,600]]])
arr
```

Out[22]:
```
array([[[  1,   2,   3],
        [  4,   5,   6]],

       [[  7,   5,   8],
        [  1,   4,   8]],

       [[100, 200, 300],
        [400, 500, 600]]])
```

In [23]:
```
arr.shape
```

Out[23]:
```
(3, 2, 3)
```

Attributes of Numpy

In [24]:
```
x=np.array([[10,20,30],[40,50,60]])
x
```

Out[24]:
```
array([[10, 20, 30],
       [40, 50, 60]])
```

In [25]:
```
x.shape
```

Out[25]:
```
(2, 3)
```

In [26]:
```
#to count total no of elements in array
x.size
```

Out[26]:
```
6
```

In [29]:
```
#to change shape of array
x.reshape(3,2)
```

Out[29]:
```
array([[10, 20],
       [30, 40],
       [50, 60]])
```

In [31]:
```
#Transpose
                        #transpose = change rows to columns and columns to rows
x.T
```

Out[31]:
```
array([[10, 40],
       [20, 50],
       [30, 60]])
```

01-06-2023

Joining of Array

In [2]:
```
import numpy as np
np.arange(1,7)
```

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

In [3]:
```
np.arange(5)
```

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

In [4]:
```
np.arange(0,8)
```

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

In [5]:
```
np.arange(-7)
```

Out[5]:
```
array([], dtype=int32)
```

In [6]:
```
np.arange(2,8).reshape(2,3)
```

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

In [9]:
```
np.arange(2,5)
```

Out[9]:
```
array([2, 3, 4])
```

In [11]:
```
np.arange(3,9).reshape(3,2).T
```

Out[11]:
```
array([[3, 5, 7],
       [4, 6, 8]])
```

In [12]:
```
np.arange(1,7).reshape(2,3).T.shape
```

Out[12]:
```
(3, 2)
```

In [13]:
```
a=np.arange(1,7).reshape(2,3)
b=np.arange(7,13).reshape(2,3)
```

In [14]:
```
a
```

Out[14]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [15]:
```
b
```

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

Concatenate

In [16]:
```
np.concatenate((a,b))              #concat a and b
```

Out[16]:
```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12]])
```

In [17]:
```
np.arange(1,13,2)
```

Out[17]:
```
array([ 1,  3,  5,  7,  9, 11])
```

In [18]:
```
np.arange(1,5,3)
```

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

hstack and vstack and stack

H = Horizantal

V = Vertical

In [27]:
```
#stack = storage of array vertically or horizontally
```

In [28]:
```
np.hstack((a,b))
```

Out[28]:
```
array([[ 1,  2,  3,  7,  8,  9],
       [ 4,  5,  6, 10, 11, 12]])
```

In [29]:
```
np.vstack((a,b))
```

Out[29]:
```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12]])
```

In [30]:
```
np.hstack((b,a))
```

Out[30]:
```
array([[ 7,  8,  9,  1,  2,  3],
       [10, 11, 12,  4,  5,  6]])
```

In [31]:
```
np.vstack((b,a))
```

Out[31]:
```
array([[ 7,  8,  9],
       [10, 11, 12],
       [ 1,  2,  3],
       [ 4,  5,  6]])
```

In [33]:
```
#stack.............it is use to convert 2D array into 3D array
```

In [36]:
```
np.stack((a,b))
```

Out[36]:
```
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

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

In [37]:
```
np.stack((a,b)).shape
```

Out[37]:
```
(2, 2, 3)
```

In [40]:
```
np.concatenate((a,b),axis=1,dtype="float")    #axis 1 is horizontal axis    it is
```

Out[40]:
```
array([[ 1.,  2.,  3.,  7.,  8.,  9.],
       [ 4.,  5.,  6., 10., 11., 12.]])
```

In [42]:
```
np.concatenate((a,b),axis=0,dtype="float")     #axis 0 is vertically downwordsvf a
                                               # axis = 0 is v stack
```

Out[42]:
```
array([[ 1.,  2.,  3.],
       [ 4.,  5.,  6.],
       [ 7.,  8.,  9.],
       [10., 11., 12.]])
```

In [43]:
```
#ravel

# is is used to convert 2D aaray into 1D array

a
```

Out[43]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [44]:
```
np.ravel(a)
```

Out[44]:
```
array([1, 2, 3, 4, 5, 6])
```

Append

In [45]:
```
# append(array,list of element)
# this is used to add elements in existing array
```

In [46]:
```
c=np.arange(1,7)
c
```

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

In [47]:
```
np.append(c,[7,8,9])
```

Out[47]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [48]:
```
np.append(c,[100,200,300,400,500])
```

Out[48]:
```
array([  1,   2,   3,   4,   5,   6, 100, 200, 300, 400, 500])
```

In [50]:
```
s=np.append(c,[100,200,300,400,500])
s
```

Out[50]:
```
array([  1,   2,   3,   4,   5,   6, 100, 200, 300, 400, 500])
```

In [51]:
```
a
```

Out[51]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [52]:
```
np.append(a,[10,20,30,40])
```

Out[52]:
```
array([ 1,  2,  3,  4,  5,  6, 10, 20, 30, 40])
```

In [54]:
```
np.append(a,[[10,11,12]],axis=0)
```

Out[54]:
```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [10, 11, 12]])
```

Insert

In [55]:
```
# This method is used to insert elements in specific position
#sequence = array name-indexing-value
```

In [56]:
```
d=np.arange(2,6)
d
```

Out[56]:
```
array([2, 3, 4, 5])
```

In [57]:
```
np.insert(d,0,100)
```

Out[57]:
```
array([100,   2,   3,   4,   5])
```

In [62]:
```
np.insert(d,-1,200)
```

Out[62]:
```
array([  2,   3,   4, 200,   5])
```

In [63]:
```
np.insert(d,-2,20)
```

Out[63]:
```
array([ 2,  3, 20,  4,  5])
```

05-06-2023

Delete

In [70]:
```
#delete (array,list of element index)
#it is used to delete elements from array
```

In [71]:
```
import numpy as np
```

In [72]:
```
a=np.arange(1,6)
a
```

Out[72]:
```
array([1, 2, 3, 4, 5])
```

In [73]:
```
np.delete(a,[0,1])
```

Out[73]:
```
array([3, 4, 5])
```

In [74]:
```
np.delete(a,[2])
```

Out[74]:
```
array([1, 2, 4, 5])
```

In [75]:
```
np.delete(a,[0])
```

Out[75]:
```
array([2, 3, 4, 5])
```

In [76]:
```
np.delete(a,[-1])
```

Out[76]:
```
array([1, 2, 3, 4])
```

In [77]:
```
b=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
b
```

Out[77]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [78]:
```
np.delete(b,1,axis=0)
```

Out[78]:
```
array([[ 1,  2,  3,  4],
       [ 9, 10, 11, 12]])
```

In [79]:
```
np.delete(b,0,axis=1)
```

Out[79]:
```
array([[ 2,  3,  4],
       [ 6,  7,  8],
       [10, 11, 12]])
```

In [80]:
```
np.delete(b,[0,3],axis=1)
```

Out[80]:
```
array([[ 2,  3],
       [ 6,  7],
       [10, 11]])
```

Empty

In [81]:
```
#empty(shape,dtype)  shows random values
#it is creates initialized array of specified shape
```

In [82]:
```
np.empty([3,2])
```

Out[82]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.]])
```

In [83]:
```
np.empty([2,2])
```

Out[83]:
```
array([[1., 1.],
       [1., 1.]])
```

In [84]:
```
np.empty([3,2],dtype="str")
```

Out[84]:
```
array([['', ''],
       ['', ''],
       ['', '']], dtype='<U1')
```

In [85]:
```
np.empty([3,2],dtype="int").T
```

Out[85]:
```
array([[-1006689664,           0,           1],
       [        512,           0,         512]])
```

Zeros

In [1]:
```
# it returns new array of specialized size and type.
```

In [5]:
```
import numpy as np
np.zeros([3,2])
```

Out[5]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.]])
```

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

Ones

In [89]:
```
#it returns new array of specified size and type filled with ones
```

In [90]:
```
np.ones([2,3])
```

Out[90]:
```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [91]:
```
np.ones([3,3],dtype="int")
```

Out[91]:
```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]])
```

In [92]:
```
np.ones([3,3],dtype="str")
```

Out[92]:
```
array([['1', '1', '1'],
       ['1', '1', '1'],
       ['1', '1', '1']], dtype='<U1')
```

In [93]:
```
x=np.ones([3,3],dtype="str")
x
```

Out[93]:
```
array([['1', '1', '1'],
       ['1', '1', '1'],
       ['1', '1', '1']], dtype='<U1')
```

In [94]:
```
np.ravel((x))
```

Out[94]:
```
array(['1', '1', '1', '1', '1', '1', '1', '1', '1'], dtype='<U1')
```

In [95]:
```
np.ones(5,dtype="int")
```

Out[95]:
```
array([1, 1, 1, 1, 1])
```

In [96]:
```
np.ones(7,dtype="int")*1
```

Out[96]:
```
array([1, 1, 1, 1, 1, 1, 1])
```

In [97]:
```
np.ones([5,5],dtype="int")*2
```

Out[97]:
```
array([[2, 2, 2, 2, 2],
       [2, 2, 2, 2, 2],
       [2, 2, 2, 2, 2],
       [2, 2, 2, 2, 2],
       [2, 2, 2, 2, 2]])
```

Eye

In [98]:
```
# it is identity matrix....
# diagnally create 0 and 1

# 1 = diagnal position
# 0 = Non diagnal position
```

In [99]:
```
np.eye(2)
```

Out[99]:
```
array([[1., 0.],
       [0., 1.]])
```

In [100…
```
np.eye(1)
```

Out[100]:
```
array([[1.]])
```

In [101…
```
np.eye(3)
```

Out[101]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [102…
```
np.eye(4,dtype="int")*5
```

Out[102]:
```
array([[5, 0, 0, 0],
       [0, 5, 0, 0],
       [0, 0, 5, 0],
       [0, 0, 0, 5]])
```

In [103…
```
np.eye(4)/2
```

Out[103]:
```
array([[0.5, 0. , 0. , 0. ],
       [0. , 0.5, 0. , 0. ],
       [0. , 0. , 0.5, 0. ],
       [0. , 0. , 0. , 0.5]])
```

In [104…
```
np.eye(4)+3
```

Out[104]:
```
array([[4., 3., 3., 3.],
       [3., 4., 3., 3.],
       [3., 3., 4., 3.],
       [3., 3., 3., 4.]])
```

In [105…
```
np.eye(5,5)+2
```

Out[105]:
```
array([[3., 2., 2., 2., 2.],
       [2., 3., 2., 2., 2.],
       [2., 2., 3., 2., 2.],
       [2., 2., 2., 3., 2.],
       [2., 2., 2., 2., 3.]])
```

In [106…
```
np.eye(4,5,k=-1)
```

Out[106]:
```
array([[0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.]])
```

In [107…
```
np.eye(4,5,k=1)
```

Out[107]:
```
array([[0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

Full

In [108…
```
#this function is used to initialize the elements in given values
#np.full(shape,values)
```

In [109…
```
a=np.full([2,2],100)
a
```

Out[109]:
```
array([[100, 100],
       [100, 100]])
```

In [110…
```
a=np.full([5,4],25)
a
```

Out[110]:
```
array([[25, 25, 25, 25],
       [25, 25, 25, 25],
       [25, 25, 25, 25],
       [25, 25, 25, 25],
       [25, 25, 25, 25]])
```

In [111…
```
np.full(9,11)
```

Out[111]:
```
array([11, 11, 11, 11, 11, 11, 11, 11, 11])
```

In [112…
```
np.full(12,100).reshape(3,4)
```

Out[112]:
```
array([[100, 100, 100, 100],
       [100, 100, 100, 100],
       [100, 100, 100, 100]])
```

In [113…
```
np.full(12,100).reshape(3,4).T
```

Out[113]:
```
array([[100, 100, 100],
       [100, 100, 100],
       [100, 100, 100],
       [100, 100, 100]])
```

In [114…
```
np.full([3,3],12)
```

Out[114]:
```
array([[12, 12, 12],
       [12, 12, 12],
       [12, 12, 12]])
```

In [115…
```
np.diag([1,2,3,4,5])     # diag = diagnally
```

Out[115]:
```
array([[1, 0, 0, 0, 0],
       [0, 2, 0, 0, 0],
       [0, 0, 3, 0, 0],
       [0, 0, 0, 4, 0],
       [0, 0, 0, 0, 5]])
```

In [116…
```
a=np.diag(np.arange(5))
a
```

Out[116]:
```
array([[0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 0, 2, 0, 0],
       [0, 0, 0, 3, 0],
       [0, 0, 0, 0, 4]])
```

In [117…
```
np.diag(a)
```

Out[117]:
```
array([0, 1, 2, 3, 4])
```

In [120…
```
np.diag([1,2,3,4]).reshape(16)
```

Out[120]:
```
array([1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 3, 0, 0, 0, 0, 4])
```

In [121…
```
np.diag(np.arange(1,16))
```

Out[121]:
```
array([[ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  3,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  4,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  5,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  6,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  7,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  8,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  9,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 10,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 11,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 12,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 13,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 14,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 15]])
```

In [123…
```
np.diag(np.arange(1,16)).reshape(225)
```

Out[123]:
```
array([ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  2,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  3,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  4,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  5,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  6,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  7,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  8,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  0,  9,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0, 10,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0, 11,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0, 12,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0, 13,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0, 14,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0, 15])
```

Randint

In [126…
```
#this functions is used to generate random number between given range
#randint(min,max,total values)
# randint shows only integer values
```

In [125…
```
np.random.randint(1,10,10)        # 1=min valvue   10=is max value   10=1 to 10 va
```

Out[125]:
```
array([4, 8, 8, 7, 3, 5, 7, 2, 3, 6])
```

In [128…
```
np.random.randint(-20,10,10)
```

Out[128]:
```
array([  7, -18, -16, -18,  -9,   7,  -4,  -6, -16,   3])
```

06-06-2023

In [6]:
```
import numpy as np
```

In [7]:
```
#rand()   ..... it shows float values
```

In [8]:
```
np.random.rand(2,3)
```

Out[8]:
```
array([[0.88481624, 0.82975491, 0.95976816],
       [0.78625268, 0.22271994, 0.12298761]])
```

In [9]:
```
np.random.rand(2,3).T
```

Out[9]:
```
array([[0.46058305, 0.60643586],
       [0.31247093, 0.03162801],
       [0.68093614, 0.86040396]])
```

In [11]:
```
#randn()..... it shows random values nearest to zero
# it will generate random valuies closed to zero.....either positive and negative
```

In [12]:
```
np.random.randn(5)
```

Out[12]:
```
array([-0.89878313, -1.90648864, -0.50218362, -0.22282846, -0.78201732])
```

In [13]:
```
np.random.randn(3)
```

Out[13]:
```
array([ 0.39563398, -0.04282673,  0.30957112])
```

In [14]:
```
np.random.randn(2,3)
```

Out[14]:
```
array([[ 1.74351899,  0.56420552,  1.79648883],
       [-1.48167771, -0.34665445,  0.71620033]])
```

In [15]:
```
np.random.randn(2,3).T
```

Out[15]:
```
array([[ 1.43679856,  0.40612477],
       [-0.4789441 ,  0.18124139],
       [ 1.42103165,  0.21447565]])
```

In [16]:
```
np.random.seed(0)         #.................to fix random values
np.random.randint(1,100,10)
```

Out[16]:
```
array([45, 48, 65, 68, 68, 10, 84, 22, 37, 88])
```

In [18]:
```
np.random.seed(0)            # we have to add arguments in seed function thats why
np.random.randint(1,100,11)
```

Out[18]:
```
array([45, 48, 65, 68, 68, 10, 84, 22, 37, 88, 71])
```

In [19]:
```
np.random.randint(1,100,11)
```

Out[19]:
```
array([89, 89, 13, 59, 66, 40, 88, 47, 89, 82, 38])
```

In [20]:
```
np.random.seed(2)
np.random.randint(1,100,11)
```

Out[20]:
```
array([41, 16, 73, 23, 44, 83, 76,  8, 35, 50, 96])
```

In [21]:
```
np.random.seed(2)
np.random.randint(1,100,11)
```

Out[21]:
```
array([41, 16, 73, 23, 44, 83, 76,  8, 35, 50, 96])
```

In [24]:
```
np.random.seed(111)
np.random.randint(1,11,10)
```

Out[24]:
```
array([ 5,  5,  5,  7,  4, 10,  3,  7,  3,  9])
```

In [25]:
```
np.random.seed(111)
np.random.randint(1,11,11)
```

Out[25]:
```
array([ 5,  5,  5,  7,  4, 10,  3,  7,  3,  9,  8])
```

Linspace

In [34]:
```
#it will return evenly spaced numbers over a specified interval.
#return values are calculated over start and interval[start,stop]
#parameters
#Linespace(start,stop,num,endpoint,retstep)
#bydefault it will returns 50 values
# endpoint bydefault is true
```

In [35]:
```
a=np.linspace(10,20)
a
```

Out[35]:
```
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.        ])
```

In [36]:
```
a.size
```

Out[36]:
```
50
```

In [37]:
```
np.linspace(10,20,num=15)      # if i have only 15 values then i use num=
```

Out[37]:
```
array([10.        , 10.71428571, 11.42857143, 12.14285714, 12.85714286,
       13.57142857, 14.28571429, 15.        , 15.71428571, 16.42857143,
       17.14285714, 17.85714286, 18.57142857, 19.28571429, 20.        ])
```

In [38]:
```
a.size
```

Out[38]:
```
50
```

In [40]:
```
c=np.linspace(10,20,num=20,endpoint=False)
c
```

Out[40]:
```
array([10. , 10.5, 11. , 11.5, 12. , 12.5, 13. , 13.5, 14. , 14.5, 15. ,
       15.5, 16. , 16.5, 17. , 17.5, 18. , 18.5, 19. , 19.5])
```

In [42]:
```
np.linspace(10,20,num=20,endpoint=False,retstep=True)
```

Out[42]:
```
(array([10. , 10.5, 11. , 11.5, 12. , 12.5, 13. , 13.5, 14. , 14.5, 15. ,
        15.5, 16. , 16.5, 17. , 17.5, 18. , 18.5, 19. , 19.5]),
 0.5)
```

Logspace

In [44]:
```
np.logspace(10,20,num=30)
```

Out[44]:
```
array([1.00000000e+10, 2.21221629e+10, 4.89390092e+10, 1.08263673e+11,
       2.39502662e+11, 5.29831691e+11, 1.17210230e+12, 2.59294380e+12,
       5.73615251e+12, 1.26896100e+13, 2.80721620e+13, 6.21016942e+13,
       1.37382380e+14, 3.03919538e+14, 6.72335754e+14, 1.48735211e+15,
       3.29034456e+15, 7.27895384e+15, 1.61026203e+16, 3.56224789e+16,
       7.88046282e+16, 1.74332882e+17, 3.85662042e+17, 8.53167852e+17,
       1.88739182e+18, 4.17531894e+18, 9.23670857e+18, 2.04335972e+19,
       4.52035366e+19, 1.00000000e+20])
```

In [45]:
```
#lets change base
```

In [48]:
```
np.logspace(10,20,num=20,base=3)
```

Out[48]:
```
array([5.90490000e+04, 1.05275908e+05, 1.87691864e+05, 3.34627706e+05,
       5.96593265e+05, 1.06364033e+06, 1.89631834e+06, 3.38086396e+06,
       6.02759614e+06, 1.07463405e+07, 1.91591857e+07, 3.41580835e+07,
       6.08989697e+07, 1.08574139e+08, 1.93572137e+08, 3.45111392e+08,
       6.15284174e+08, 1.09696354e+09, 1.95572886e+09, 3.48678440e+09])
```

Slicing

1D Array

In [49]:
```
a=np.array([10,20,30,40,50])
a
```

Out[49]:
```
array([10, 20, 30, 40, 50])
```

In [50]:
```
a[::]
```

Out[50]:
```
array([10, 20, 30, 40, 50])
```

In [52]:
```
a[0:4]
```

Out[52]:
```
array([10, 20, 30, 40])
```

In [53]:
```
a[0:1]
```

Out[53]:
```
array([10])
```

In [54]:
```
a[::-1]
```

Out[54]:
```
array([50, 40, 30, 20, 10])
```

In [56]:
```
a[0:5:2]
```

Out[56]:
```
array([10, 30, 50])
```

In [59]:
```
a[-3::-1]
```

Out[59]:
```
array([30, 20, 10])
```

2D Array

In [61]:
```
b=np.array([[10,20,30],[40,50,60]])
b
```

Out[61]:
```
array([[10, 20, 30],
       [40, 50, 60]])
```

In [62]:
```
b[1,1]
```

Out[62]:
```
50
```

In [65]:
```
b[1,0]
```

Out[65]:
```
40
```

In [66]:
```
b[1,2]
```

Out[66]:
```
60
```

In [71]:
```
b[0,2]
```

Out[71]:
```
30
```

In [73]:
```
b[1:2,1:2]
```

Out[73]:
```
array([[50]])
```

In [74]:
```
b[0:2,0:2]
```

Out[74]:
```
array([[10, 20],
       [40, 50]])
```

In [76]:
```
b[0:1,0:2]
```

Out[76]:
```
array([[10, 20]])
```

In [77]:
```
a=np.array([[1,2,3,4,5],[6,7,8,9,10]])
a
```

Out[77]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [78]:
```
a[0:2,1:4]
```

Out[78]:
```
array([[2, 3, 4],
       [7, 8, 9]])
```

Arithmatic Operator

In [79]:
```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
b=np.array([[1,1,1],[1,1,1],[1,1,1]])
```

In [80]:
```
a
```

Out[80]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [81]:
```
b
```

Out[81]:
```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]])
```

Addition

In [82]:
```
np.add(a,b)
```

Out[82]:
```
array([[ 2,  3,  4],
       [ 5,  6,  7],
       [ 8,  9, 10]])
```

In [85]:
```
np.subtract(a,b)
```

Out[85]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [88]:
```
np.divide(a,b)
```

Out[88]:
```
array([[1., 2., 3.],
       [4., 5., 6.],
       [7., 8., 9.]])
```

In [87]:
```
np.mod(a,b)
```

Out[87]:
```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]], dtype=int32)
```

Numpy Array Selection

In [89]:
```
#this is called process masking
```

In [91]:
```
a=np.array([1,2,3,4,5,6,7,8,9,10])
a
```

Out[91]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [92]:
```
a>4
```

Out[92]:
```
array([False, False, False, False,  True,  True,  True,  True,  True,
        True])
```

In [94]:
```
mask=a>4
mask
```

Out[94]:
```
array([False, False, False, False,  True,  True,  True,  True,  True,
        True])
```

In [95]:
```
a[mask]
```

Out[95]:
```
array([ 5,  6,  7,  8,  9, 10])
```

In [98]:
```
a[a>=4].reshape(1,7)
```

Out[98]:
```
array([[ 4,  5,  6,  7,  8,  9, 10]])
```

In [99]:
```
np.min(a)
```

Out[99]:
```
1
```

In [100…
```
np.max(a)
```

Out[100]:
```
10
```

In [101…
```
np.sum(a)
```

Out[101]:
```
55
```

In [ ]: