Importing Numpy and creating Arrays

In [2]:
```
#importing numpy library
import numpy as np
np.__version__
```

Out[2]:
```
'1.19.4'
```

In [25]:
```
#function 01 creating an array 1-D (np.array)                    (np.array)
a = np.array([1,2,3,4,5])
a
```

Out[25]:
```
array([1, 2, 3, 4, 5])
```

In [26]:
```
#2-D array (np.array 2-D)
a = np.array([[1,2,3,4,5],[6,7,8,9,10]], dtype = int)
a
```

Out[26]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [27]:
```
#function 02                                                    (np.arange)
a = np.arange(1,10,2)
a
```

Out[27]:
```
array([1, 3, 5, 7, 9])
```

In [28]:
```
#function 03 array of 6 with divided values from (0-50)          (np.linspace)
a = np.linspace(0,50,6)
a
```

Out[28]:
```
array([ 0., 10., 20., 30., 40., 50.])
```

In [29]:
```
#function 04 array with zeros                                   (np.zeros)
a = np.zeros(5, dtype=int)
a
```

Out[29]:
```
array([0, 0, 0, 0, 0])
```

In [30]:
```
#function 05 array with ones                                    (np.ones)
a = np.ones(5)
a
```

Out[30]:
```
array([1., 1., 1., 1., 1.])
```

In [31]:
```
#function 06 3x3 array with all values one                    (np.full)
a = np.full((3,3),10)
a
```

Out[31]:
```
array([[10, 10, 10],
       [10, 10, 10],
       [10, 10, 10]])
```

In [21]:
```
#function 07 3x4 array of random floating values b/w 0-100          (np.rando
m.rand)
a = np.random.rand(3,4)*50
a
# we can't write (3,4)*50 as ((3,4),50) bcz  a tuple can not be in integar for
m
```

Out[21]:
```
array([[22.98069764, 41.24821595, 30.13927941, 45.19732362],
       [ 9.06281993, 30.83591635, 36.2907192 , 16.1105286 ],
       [ 4.33561946, 36.28487862,  8.70964636,  2.22515625]])
```

In [22]:
```
#function 08 3x3 array of random int values b/w 0-50             (np.random.r
andint)
a = np.random.randint(60, size=(3,3))
a
```

Out[22]:
```
array([[33, 20, 18],
       [11, 59, 16],
       [56, 55, 40]])
```

In [24]:
```
#function 09 4x4 diagnol array                                   (np.eye)
arr = np.eye(4)
arr
```

Out[24]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

Functions of Numpy Arrays

In [201]:
```
#function 10 reshape                              (np.reshape)
a = np.arange(1,7).reshape(3,2)
a
```

Out[201]:
```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [54]:
```
#function 11 size of an array                    (np.size)
a = np.array(["lahore","Karachi","Islamabad"])
a = a.size
a
```

Out[54]: 3

In [161]:
```
#size of an Array
a.size
```

Out[161]: 6

In [202]:
```
#function 12 shape of an array                    (np.shape)
a = np.array(["lahore","Karachi","Islamabad"])
a=np.shape(a)
a
```

Out[202]: (3,)

In [203]:
```
#function 13 dimension of an array                (np.ndim)
a = np.array(["lahore","Karachi","Islamabad"])
a= np.ndim(a)
a
```

Out[203]: 1

In [61]:
```
#function 14 data type of an array                (np.dtype)
a = np.array([1,2,3,4,5,6,7,8,9,10])
a = a.dtype
a
```

Out[61]: dtype('int64')

In [65]:
```
#function 15 converting data type of an array     (a.astype)     a= variable
a = np.array([1,2,3,4,5])
a = a.astype(float)
a
```

Out[65]: array([1., 2., 3., 4., 5.])

In [204]:
```
#function 15 converting data type of an array     (.astype)
a = np.array([1,2,3,4,5]).astype(float)
a
```

Out[204]: array([1., 2., 3., 4., 5.])

In [66]:
```
#function 16 taking transpose of 3x2 to 2x3 array          (a.T)          a=va
riable
a = np.array([1,2,3,4,5,6])
a = a.reshape((3,2))
a = a.T
a
```

Out[66]:
```
array([[1, 3, 5],
       [2, 4, 6]])
```

In [67]:
```
#function 17 resizing an array                              (np.resize)
a = np.array([1,2,3,4,5,6])
a = np.resize(a,(3,3))
a
```

Out[67]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [1, 2, 3]])
```

In [68]:
```
#function 18 converting 2D into 1D array                (a.flatten)
 a=variable
a = np.array([(1,2,3),(4,5,6)])
a = a.flatten()
a
```

Out[68]:
```
array([1, 2, 3, 4, 5, 6])
```

In [69]:
```
#function 19 converting array to a list                  (a.tolist)
a-variable
a = np.array([(1,2,3),(4,5,6)])
a= a.tolist()
a
```

Out[69]:
```
[[1, 2, 3], [4, 5, 6]]
```

In [72]:
```
#function 20 sorting an array                           (np.sort)
a = np.array((2,1,4,3))
a = np.sort(a)
a
```

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

In [81]:
```
#sorting rows of an array                           (np.sort for rows)
a = np.array([(3,1,4),(0,2,5)])
a = np.sort(a,axis=1)
a
```

Out[81]:
```
array([[1, 3, 4],
       [0, 2, 5]])
```

In [80]:
```
#sorting columns of an array                    (np.sort for column)
a = np.array([(3,1,4),(0,2,5)])
a = np.sort(a, axis= 0)
a
```

Out[80]:
```
array([[0, 1, 4],
       [3, 2, 5]])
```

In [82]:
```
#sorting columns of an array                    (np.sort for column)
a = np.array([(3,1,4),(0,2,5)])
a = np.sort(a)
a
```

Out[82]:
```
array([[1, 3, 4],
       [0, 2, 5]])
```

Adding and REmoving Elements

In [84]:
```
#function 21 adding element to an array                (np.append)
a = np.array(["lahore","Karachi","Islamabad","Swat","Naran Kaghan"])
a = np.append(a,"GilGit")
a
```

Out[84]:
```
array(['lahore', 'Karachi', 'Islamabad', 'Swat', 'Naran Kaghan', 'GilGit'],
      dtype='<U12')
```

In [85]:
```
#function 22 inserting element to an array at specific index           (np.in
sert)
a = np.array(["lahore","Karachi","Islamabad","Swat","Naran Kaghan"])
a = np.insert(a,2 ,"Muree")
a
```

Out[85]:
```
array(['lahore', 'Karachi', 'Muree', 'Islamabad', 'Swat', 'Naran Kaghan'],
      dtype='<U12')
```

In [89]:
```
#function 23 deleting element from an array             (np.delete)
a = np.array(["lahore","Karachi","Islamabad","Peshawar"])
a = np.delete(a,1)
a
```

Out[89]:
```
array(['lahore', 'Islamabad', 'Peshawar'], dtype='<U9')
```

Concatenating and splitting arrays

In [95]:
```
#function 24 combining arrays                          (np.concatenate)
a1 = np.array([(1,2,3,1,2),(4,5,6,1,2)])
a2 = np.array([(3,1,4,2,2),(0,2,4,4,5)])
a3 = np.concatenate((a1,a2))
a3
```

Out[95]:
```
array([[1, 2, 3, 1, 2],
       [4, 5, 6, 1, 2],
       [3, 1, 4, 2, 2],
       [0, 2, 4, 4, 5]])
```

In [100]:
```
#function 25 splitting an array                      (np.array_split)
a1 = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
a2 = np.array_split(a1,4)
a2
```

Out[100]:
```
[array([1, 2, 3]), array([4, 5, 6]), array([7, 8, 9]), array([10, 11, 12])]
```

In [103]:
```
#function 26 splitting array at 2nd index              (np.hsplit)
a1 = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
a2 = np.hsplit(a1, 2)
a2
```

Out[103]:
```
[array([1, 2, 3, 4, 5, 6]), array([ 7,  8,  9, 10, 11, 12])]
```

Selecting Elements from Arrays

In [114]:
```
#function 27 selecting element at specific index of an array          (variable[index])
a = np.array([1,2,3,4,5,6,7])
a = a[5]
a
```

Out[114]:
```
6
```

In [120]:
```
#selecting element from an 2-D array                      (variable[])
a = np.array([[1,2,3,4,5,1,2,3,4,5],[6,7,8,9,10,6,7,8,9,10]])
a = a[1,2]
a
```

Out[120]:
```
8
```

In [123]:
```
#function 28 slicing elements from an array   (variable[starting index : ending index])
a = np.array([1,2,3,4,5,6,7,8,9,10])
a = a[0:5]
a
```

Out[123]:
```
array([1, 2, 3, 4, 5])
```

Max and Min Functions

In [124]:
```
#function 29 getting maximum value                (np.max)
a = np.array([10,20,30,40,50])
a = np.max(a)
a
```

Out[124]: 50

In [126]:
```
#function 30 getting minimum value                (np.min)
a = np.array([10,20,30,40,50])
a = np.min(a)
a
```

Out[126]: 10

Mathematical Operations[addition(+),subtraction(-),multiplication(*),divi

In [128]:
```
#function 31 addition of two arrays                    (np.add)
a1 = np.array([1,3,5,7,9,11])
a2 = np.copy(a1)
a3 = np.add(a1,a2)
a3
```

Out[128]: array([2, 6, 10, 14, 18, 22])

In [130]:
```
#addition of specific value in array                    (np.add)
a1 = np.array([1,2,3,4,5,6,7,8,9,10])
a  = np.add(a1,3)
a
```

Out[130]: array([4, 5, 6, 7, 8, 9, 10, 11, 12, 13])

In [132]:
```
#fuction 32 subtraction of two arrays                    (np.subtract)
a1 = np.array([10,20,30,40,50,60])
a2 = np.array([5,15,20,30,40,50])
a3 = np.subtract(a1,a2)
a3
```

Out[132]: array([5, 5, 10, 10, 10, 10])

In [140]:
```
#fuction 33 multiplication of two arrays                (np.multiply)
a1 = np.array([1,2,3,4,5,6,7,8])
a2 = np.copy(a1)
a3 = np.multiply(a1,a2)
a3
```

Out[140]:
```
array([ 1,  4,  9, 16, 25, 36, 49, 64])
```

In [143]:
```
#fuction 34 divide of two arrays                                (np.divide)
a1 = np.array([10,20,30,40,50,60])
a2 = np.array([5,10,15,20,25,30])
a3 = np.divide(a2,a1)
a3
```

Out[143]:
```
array([0.5, 0.5, 0.5, 0.5, 0.5, 0.5])
```

In [150]:
```
#function 35 taking square root of each element in an array          (np.sqr
t)
a = np.array([1,4,9,16,25,36,49,64,81,100])
a = np.sqrt(a)
a
```

Out[150]:
```
array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

In [154]:
```
#fuction 36 taking square of each element in an array               (np.squar
e)
a = np.array([1,4,9,16,25,36,49,64,81,100])
a = np.square(a)
a
```

Out[154]:
```
array([    1,    16,    81,   256,   625,  1296,  2401,  4096,  6561,
       10000])
```

In [155]:
```
#function 37 assigning power to each element in an array            (np.power)
a1 = np.array([1,2,3,4,5,6,7,8,9,10])
a2 = np.array([11,12,13,14,15,16,17,18,19,20])
a3 = np.power(a1,a2)
a3
```

Out[155]:
```
array([                  1,                4096,             1594323,
                 268435456,          30517578125,        2821109907456,
              232630513987207,    18014398509481984, 1350851717672992089,
       7766279631452241920])
```

In [156]:
```
#function 38 equivalence of arrays                                  (np.array_equa
l)
a1 = np.array([1,2,3,4,5])
a2 = np.array([6,7,8,9,10])
a3 = np.array_equal(arr1,arr2)
a3
```

Out[156]:
```
False
```

In [157]:
```
#function 39 exponent of elements in an array                    (np.exp)
a = np.array([1,2,3,4,5])
a = np.exp(a)
a
```

Out[157]:
```
array([  2.71828183,   7.3890561 ,  20.08553692,  54.59815003,
       148.4131591 ])
```

Cos,Sin and Tan Functions

In [158]:
```
#function 40 sine function                                       (np.sin)
a = np.array([0,30,45,60,90,120,])
a = np.sin(a)
a
```

Out[158]:
```
array([ 0.        , -0.98803162,  0.85090352, -0.30481062,  0.89399666,
        0.58061118])
```

In [159]:
```
#function 41 cosine function                                     (np.cos)
a = np.array([0,30,45,60,90,120,])
a = np.sin(a)
a
```

Out[159]:
```
array([ 0.        , -0.98803162,  0.85090352, -0.30481062,  0.89399666,
        0.58061118])
```

In [160]:
```
#function 42 tan function                                        (np.tan)
a = np.array([0,30,45,60,90,120,])
a = np.sin(a)
a
```

Out[160]:
```
array([ 0.        , -0.98803162,  0.85090352, -0.30481062,  0.89399666,
        0.58061118])
```

log function

In [165]:
```
#function 42 log function                                        (np.log)
a = np.array([1,3,5,7,9])
a = np.log(arr)
a
```

Out[165]:
```
array([0.        , 1.09861229, 1.60943791, 1.94591015, 2.19722458])
```

In [168]:
```
#function 43 absolute values of elements                         (np.abs)
a = np.array([1,2.11111,3.22222,4.33333,5.44444])
a = np.abs(a)
a
```

Out[168]:
```
array([1.     , 2.11111, 3.22222, 4.33333, 5.44444])
```

In [169]:
```
#function 43 absolute values of elements                    (np.abs)
a = np.array([1,2,3,4,5])
a = np.abs(a)
a
```

Out[169]:
```
array([1, 2, 3, 4, 5])
```

In [171]:
```
#function 44 round-of function                          (np.round)    we can
 add value upto we eant to get rounded value
a = np.array([1.24354,2.34567,3.45341,4.14352,5.24315])
a = np.round(a,3)
a
```

Out[171]:
```
array([1.244, 2.346, 3.453, 4.144, 5.243])
```

In [176]:
```
#function 45 "round-up" to the nearest int value                (np.ceil)
a = np.array([1.24354,2.34567,3.45341,4.14352,5.24315])
a = np.ceil(a)
a
```

Out[176]:
```
array([2., 3., 4., 5., 6.])
```

In [177]:
```
#function 46 "round-down"to the nearest int value               (np.ceil)
a = np.array([1.24354,2.34567,3.45341,4.14352,5.24315])
a = np.floor(a)
a
```

Out[177]:
```
array([1., 2., 3., 4., 5.])
```

statistical operations

In [183]:
```
#function 47 sum of matrices along "coulmns"                  (a.sum)
a=variable
a = np.array([[1,3,5],[2,4,6]])
a = a.sum(axis=0)
a
```

Out[183]:
```
array([ 3,  7, 11])
```

In [185]:
```
#function 47 sum of matrices along "row"                      (a.sum)
a=variable
a = np.array([[1,3,5],[2,4,6]])
a = a.sum(axis=1)
a
```

Out[185]:
```
array([ 9, 12])
```

In [186]:
```
#function 47 sum of whole matrix                                    (a.sum)
a=variable
a = np.array([[1,3,5],[2,4,6]])
a = a.sum()
a
```

Out[186]: 21

In [189]:
```
#function 48 mean of values                (along columns)
(np.mean)
a = np.array([[1,2,3],[4,5,6]])
a = np.mean(a,axis=0)
a
```

Out[189]: array([2.5, 3.5, 4.5])

In [190]:
```
#function 48 mean of values
(np.mean)
a = np.array([[1,2,3],[4,5,6]])
a = np.mean(a,axis=1)
a
```

Out[190]: array([2., 5.])

In [193]:
```
#function 48 median of values                (along columns)
(np.median)
a = np.array([[1,2,3],[4,5,6]])
a = np.median(a,axis=0)
a
```

Out[193]: array([2.5, 3.5, 4.5])

In [195]:
```
#function 50 variance of values                                      (n
p.var)
a = np.array([1,2,3,4,5,6,7,8,9,10])
a = np.var(a)
a
```

Out[195]: 8.25

In [3]:
```
#function 51 standard deviation of values                            (n
p.std)
a = np.array([1,2,3,4,5,6,7,8,9,10])
a = np.std(a)
a
```

Out[3]: 2.8722813232690143

In [ ]:
```

```