Numpy

Numpy installation

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

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (1.23.5)
Note: you may need to restart the kernel to use updated packages.

WARNING: There was an error checking the latest version of pip.
```

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

In [36]:
```
#create a one-dimensional array from a list
a=np.array([1,2,3,4,5])
a
```

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

In [37]:
```
#1D
a1=np.array([1,2,3,4,5])
a1
```

Out[37]:
```
array([1, 2, 3, 4, 5])
```

In [16]:
```
# ndim is used to identify dimension
a1.ndim
```

Out[16]:
```
1
```

In [14]:
```
#shape is used to identidy shape of the array
a1.shape
```

Out[14]:
```
(5,)
```

In [19]:
```
#2D
a2=np.array([[1,2,3,4,5],[6,7,8,9,0]])
a2
```

Out[19]:
```
array([[1, 2, 3, 4, 5],
       [6, 7, 8, 9, 0]])
```

In [17]:
```
# ndim is used to identify dimension
a2.ndim
```

Out[17]:
```
2
```

In [18]:
```
#shape is used to identidy shape of the array
a2.shape
```

Out[18]:
```
(2, 5)
```

In [21]:
```
#3D
a3=np.array([[[1,2,3,4,5]]])
a3
```

Out[21]:
```
array([[[1, 2, 3, 4, 5]]])
```

In [22]:
```
#dtype is used to idnetify data type of the array
a1.dtype
```

Out[22]:
```
dtype('int32')
```

In [23]:
```
a2.dtype
```

Out[23]:
```
dtype('int32')
```

In [24]:
```
a3.dtype
```

Out[24]:
```
dtype('int32')
```

In [26]:
```
#change the data type of the array
a1=np.array([1,2,3,4,5],dtype=float)
a1.dtype
```

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

In [32]:
```
#array creation using empty()
a4=np.empty((3,4), dtype=int)
a4
```

Out[32]:
```
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]])
```

In [33]:
```
##array creation using full()
a4=np.full((3,3), 55)
a4
```

Out[33]:
```
array([[55, 55, 55],
       [55, 55, 55],
       [55, 55, 55]])
```

In [35]:
```
#array creation using zeros()
a4=np.zeros((1,4))
a4
```

Out[35]:
```
array([[0., 0., 0., 0.]])
```

In [38]:
```
#array creation using ones()
a4=np.ones((1,5))
a4
```

Out[38]:
```
array([[1., 1., 1., 1., 1.]])
```

In [39]:
```
#array creation using linspace()
a4=np.linspace(0,100,5)
a4
```

Out[39]:
```
array([  0.,  25.,  50.,  75., 100.])
```

In [40]:
```
a4=np.linspace(0,50,5)
a4
```

Out[40]:
```
array([ 0. , 12.5, 25. , 37.5, 50. ])
```

In [42]:
```
#array creation using arange()
a4=np.arange(10,50,4)
a4
```

Out[42]:
```
array([10, 14, 18, 22, 26, 30, 34, 38, 42, 46])
```

In [43]:
```
#array creation using arange(),size is used to identify the size of the array
a4=np.arange(10,50,4).size
a4
```

Out[43]:
```
10
```

In [44]:
```
#array creation using arange(),dtype is used to indentify the data type of the array
a4=np.arange(10,50,4).dtype
a4
```

Out[44]:
```
dtype('int32')
```

In [45]:
```
#array creation using arange(),shape is used to indentify the shape  of the array
a4=np.arange(10,50,4).shape
a4
```

Out[45]:
```
(10,)
```

In [46]:
```
#array creation using arange(),reshape() is used to indentify the reshape the array
a4=np.arange(10,50,4)
a4.reshape(2,5)
```

Out[46]:
```
array([[10, 14, 18, 22, 26],
       [30, 34, 38, 42, 46]])
```

Indexing and accessing

In [47]:
```
s1=np.array([10,14,18,22])
s1
```

Out[47]:
```
array([10, 14, 18, 22])
```

In [48]:
```
#aceesing first element
#array creation using arange(),dtype is used to indentify the data type of the array
s1[0]
```

Out[48]:
```
10
```

In [49]:
```
#slicing
s1[0:3]
```

Out[49]:
```
array([10, 14, 18])
```

In [50]:
```
#sum() is used to calculate the sum of elements in the array
s1.sum()
```

Out[50]:
```
64
```

In [51]:
```
#mean() used to calculate the mean of the elemnts in the array
s1.mean()
```

Out[51]:
```
16.0
```

In [52]:
```
# std() used to calculate the standard deviation
s1.std()
```

Out[52]:
```
4.47213595499958
```

In [53]:
```
s2=np.array([[10,20,30,40,50],[60,70,80,90,100]])
s2
```

Out[53]:
```
array([[ 10,  20,  30,  40,  50],
       [ 60,  70,  80,  90, 100]])
```

In [56]:
```
s2[0,2]
# 0- 0th row and 2- 2nd column
```

Out[56]:
```
30
```

In [57]:
```
#slicing
s2[0:3,1:2]
```

Out[57]:
```
array([[20],
       [70]])
```

In [58]:
```
#add values based on column waise, axis=0 - column waise, axis=1 - row waise
s2.sum(axis=0)
```

Out[58]:
```
array([ 70,  90, 110, 130, 150])
```

In [64]:
```
#concatenate() used to concat 2 arrays, and the 2 arrays have the same dimension
s11=np.array([10,20,30,40])
s11
```

Out[64]:
```
array([10, 20, 30, 40])
```

In [65]:
```
s22=np.array([50,60,70,80])
s22
```

Out[65]:
```
array([50, 60, 70, 80])
```

In [66]:
```
s3=np.concatenate((s11, s22))
s3
```

Out[66]:
```
array([10, 20, 30, 40, 50, 60, 70, 80])
```

In [67]:
```
#array_split() used to split the array
s4=np.array_split(s3,2)
s4
```

Out[67]:
```
[array([10, 20, 30, 40]), array([50, 60, 70, 80])]
```

In [69]:
```
s5=np.array_split(s3,3)
s5
```

Out[69]:
```
[array([10, 20, 30]), array([40, 50, 60]), array([70, 80])]
```

In [72]:
```
#perform addition using + operatore
s4=np.array(s11 + s22)
s4
```

Out[72]:
```
array([ 60,  80, 100, 120])
```

In [73]:
```
#perform subtraction using - operatore
s4=np.array(s11 - s22)
s4
```

Out[73]:
```
array([-40, -40, -40, -40])
```

In [74]:
```
#perform multiplication using * operatore
s4=np.array(s11 * s22)
s4
```

Out[74]:
```
array([ 500, 1200, 2100, 3200])
```

In [75]:
```
#perform division using / operatore
s4=np.array(s11 / s22)
s4
```

Out[75]:
```
array([0.2       , 0.33333333, 0.42857143, 0.5       ])
```

In [77]:
```
s1.max()
```

Out[77]:
```
22
```

In [78]:
```
s1.min()
```

Out[78]:
```
10
```

In [79]:
```
s1.var()
```

Out[79]:
```
20.0
```

Accessing rows and column waise

In [80]:
```
s5=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
s5
```

Out[80]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [81]:
```
s5[0]
```

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

In [82]:
```
s5[0,1]
#accessing 0th row and 1st column
```

Out[82]:
```
2
```

In [83]:
```
s5[0][1]
#accessing 0th row and 1st column
```

Out[83]:
```
2
```

In [84]:
```
s5[:,1]
#accessing 1st column
```

Out[84]:
```
array([ 2,  6, 10])
```

In [86]:
```
s5[1,2:]
#accessing 1st row and 2nd column
```

Out[86]:
```
array([7, 8])
```

In [87]:
```
s5[:,-1]
#accessing last column
```

Out[87]:
```
array([ 4,  8, 12])
```

In [ ]:
```

```