NumPy

NumPy stands for numeric python and it is the library for numeric and scientific computing.

It consists of multidimentional array objects and a collection of routines for processing those arrays.

Creating NumPy array

1.Single dimebtional array

In [11]:
```
import numpy as np
l1 = [1,2,3,4]
n1 = np.array(l1)
n1
```

Out[11]:
```
array([1, 2, 3, 4])
```

In [21]:
```
#in a single command
n1_1 = np.array([1,2,3,4])
n1_1
```

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

In [31]:
```
type(n1)
```

Out[31]:
```
numpy.ndarray
```

In [41]:
```
type(n1_1)
```

Out[41]:
```
numpy.ndarray
```

2.Multidimentional array

In [51]:
```
import numpy as np
n2 = np.array([[10,20,30,40],[40,50,60,70]])
n2
```

Out[51]:
```
array([[10, 20, 30, 40],
       [40, 50, 60, 70]])
```

In [61]:
```
type(n2)
```

Out[61]:
```
numpy.ndarray
```

Initializing NumPy Array - with 'zeros'

import numpy as np #This command is not requeried for each new arry creation. Once created within a session, it works for all new array.

In [71]:
```
import numpy as np
n1 = np.zeros((1,2)) #1,2 - 1 row, 2 column
n1
```

Out[71]:
```
array([[0., 0.]])
```

In [81]:
```
n2 = np.zeros((5,5))
n2
```

Out[81]:
```
array([[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.]])
```

Initializing NumPy Array - with 'full'

In [91]:
```
import numpy as np
n1 = np.full((2,2),10) #((2,2),10)- 2 row, 2 column of number 10
n1
```

Out[91]:
```
array([[10, 10],
       [10, 10]])
```

In [101]:
```
n2 = np.full((4,4),6)
n2
```

Out[101]:
```
array([[6, 6, 6, 6],
       [6, 6, 6, 6],
       [6, 6, 6, 6],
       [6, 6, 6, 6]])
```

Initializing NumPy Array - with 'arange'

n1 = np.arange(10,20) #(10,20)- means from 10 to 20 where 20 is last index and is exculsive. n1

In [111]:
```
n2 = np.arange(10,50,5) #(10,50,5)- means from 10 to 50 with interval of 5.
n2
```

Out[111]:
```
array([10, 15, 20, 25, 30, 35, 40, 45])
```

Initializing NumPy Array - with 'random'

In [121]:
```
n1 = np.random.randint(1,100,5)
#randint is a method for random integer
#(1,100,5) means between 1 to 100 any 5 random number
n1
#n1 will cange every time the code is run.
```

Out[121]:
```
array([74, 96, 78, 67, 47])
```

NumPy Shape - check shape of array with 'shape'

In [131]:
```
n1
```

Out[131]:
```
array([74, 96, 78, 67, 47])
```

In [141]:
```
n1.shape
```

Out[141]:
```
(5,)
```

In [151]:
```
#Above resuls means 5 row and no column
```

In [161]:
```
n1 = np.array([[1,2,3,4],[2,3,4,5]])
n1
```

Out[161]:
```
array([[1, 2, 3, 4],
       [2, 3, 4, 5]])
```

In [171]:
```
n1.shape
```

Out[171]:
```
(2, 4)
```

In [181]:
```
#Above result means 2 row 4 column
```

In [191]:
```
# can change the shape - mean rows = column and column = rows
n1.shape = (4,2)
n1
```

Out[191]:
```
array([[1, 2],
       [3, 4],
       [2, 3],
       [4, 5]])
```

In [201]:
```
n1.shape
```

Out[201]:
```
(4, 2)
```

Stacking (joining) NumPy Arrays

vstack() - for vertical stacking

In [211]:
```
n1 = np.array([1,2,3,4])
n2 = np.array([9,8,7,6])
np.vstack((n1,n2))
```

Out[211]:
```
array([[1, 2, 3, 4],
       [9, 8, 7, 6]])
```

In [221]:
```
np.vstack((n2,n1))
```

Out[221]:
```
array([[9, 8, 7, 6],
       [1, 2, 3, 4]])
```

hstack() - for horizontal stacking

In [231]:
```
n1 = np.array([10,20,30,40])
n2 = np.array([1,2,3,4])
np.hstack((n1,n2))
```

Out[231]:
```
array([10, 20, 30, 40,  1,  2,  3,  4])
```

In [241]:
```
np.hstack((n2,n1))
```

Out[241]:
```
array([ 1,  2,  3,  4, 10, 20, 30, 40])
```

colunm_stack() as the name suggests😀

In [251]:
```
np.column_stack((n1,n2))
```

Out[251]:
```
array([[10,  1],
       [20,  2],
       [30,  3],
       [40,  4]])
```

In [261]:
```
np.column_stack((n2,n1))
```

Out[261]:
```
array([[ 1, 10],
       [ 2, 20],
       [ 3, 30],
       [ 4, 40]])
```

Intersection and Difference

In [271]:
```
n1 = np.array([10,20,30,40,50])
n2 = np.array([90,80,10,70,50])
np.intersect1d(n1,n2) #intersect1d is the method
```

Out[271]:
```
array([10, 50])
```

In [281]:
```
np.setdiff1d(n1,n2) #setdiff1d is the method - elements from n1 which are not present in n2
```

Out[281]:
```
array([20, 30, 40])
```

In [291]:
```
np.setdiff1d(n2,n1) #elements from n2 which are not present in n1
```

Out[291]:
```
array([70, 80, 90])
```

Addition of arrays

In [301]:
```
n1 = np.array([10,20])
n2 = np.array([30,40])
np.sum([n1,n2])
```

Out[301]:
```
100
```

In [311]:
```
#Add columns
np.sum(n1,n2),axis=0)
```

Out[311]:
```
array([40, 60])
```

In [321]:
```
#Add Rows
np.sum([n1,n2],axis=1)
```

Out[321]:
```
array([30, 70])
```

Basic Mathematics

In [331]:
```
#addition
n1 = np.array([10,20,30])
n1 = n1 + 1
n1
```

Out[331]:
```
array([11, 21, 31])
```

In [341]:
```
#Substraction
n1 = n1 - 1
n1
```

Out[341]:
```
array([10, 20, 30])
```

In [351]:
```
#Multiplication
n1 = n1*4
n1
```

Out[351]:
```
array([ 40,  80, 120])
```

In [361]:
```
#Division
n1 = n1/2
n1
```

Out[361]:
```
array([20., 40., 60.])
```

Basic Statistics

In [371]:
```
#Mean
n1 = np.array([10,20,30,40,50,60])
np.mean(n1)
```

Out[371]:
```
35.0
```

In [381]:
```
#Median
import numpy as np
n1 = np.array([10,20,30,40,50])
np.median(n1)
```

Out[381]:
```
30.0
```

In [391]:
```
#Standard Deviation
np.std(n1)
```

Out[391]:
```
14.142135623730951
```

Save and Load

In [401]:
```
n1 = np.array([10,20,30,40,50])
```

In [491]:
```
np.save('myarray',n1)
```

In [501]:
```
new_n1 = np.load('myarray.npy')
new_n1
```

Out[501]:
```
array([10, 20, 30, 40, 50])
```

In [ ]:
```

```