In [2]:
```
# numpy stand for numeric python
#core libreary for numeric and scientific computing
#multi dimensional array


#create simple numpy

import numpy as np
n1=[1,2,3,4,5]     #single dimensional array
a=np.array(n1)
print(a)
```

```
[1 2 3 4 5]
```

In [3]:
```
n2=([[1,2,3,4],[5,6,7,8]])       #multi dimensional array
b=np.array(n2)
print(b)
```

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

In [7]:
```
import numpy as np
c=np.zeros((3,3))   #with zeros
print(c)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [11]:
```
import numpy as np
n4=np.full((4,4),10)    #with number
print(n4)
```

```
[[10 10 10 10]
 [10 10 10 10]
 [10 10 10 10]
 [10 10 10 10]]
```

In [12]:
```
np.arange(1,11)     #with range
```

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

In [19]:
```
np.arange(10,100,10)
```

Out[19]:
```
array([10, 20, 30, 40, 50, 60, 70, 80, 90])
```

In [43]:
```
np.random.randint(1,100,2)    #with random number
```

Out[43]:
```
array([60, 91])
```

In [47]:
```
# numpy shape
#how much row and column present

n2=np.array([[1,2,3,4],[5,6,7,8]])     #output(row ,column)
np.shape(n2)
```

Out[47]:
```
(2, 4)
```

In [79]:
```
#change shape
```

```
n2.shape=(4,2)
print(n2)
```

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

In [82]:
```
#joining array

n3=np.array([1,2,3])
n4=np.array([4,5,6])
np.vstack((n3,n4))       #vstack(vertical join)
```

Out[82]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [83]:
```
n3=np.array([1,2,3])
n4=np.array([4,5,6])
np.hstack((n3,n4))       #hstack(horizontal join)
```

Out[83]:
```
array([1, 2, 3, 4, 5, 6])
```

In [98]:
```
n3=np.array([1,2,3])
n4=np.array([4,5,6])
np.column_stack((n3,n4))       #column_stack(column join)
```

Out[98]:
```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

In [33]:
```
#splitting array

#1) hsplit()
x=np.arange(16).reshape((4,4))
[y,z]=np.hsplit(x,2)
y
```

Out[33]:
```
array([[ 0,  1],
       [ 4,  5],
       [ 8,  9],
       [12, 13]])
```

In [34]:
```
#2) vsplit
x=np.arange(16).reshape((4,4))
[y,z]=np.vsplit(x,2)
y
```

Out[34]:
```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

In [100…
```
#numpy intersection and differemce


n5=np.array([10,20,30,40,90,80])
n6=np.array([40,20,50,60,70,80])
np.intersect1d(n5,n6)        #common value
```

Out[100…
```
array([20, 40, 80])
```

In [102…
```
np.setdiff1d(n5,n6)    #present in n5 value ,not present in n6
```

Out[102… array([10, 30, 90])

In [106…
```
np.setdiff1d(n6,n5)      #present in n6 value ,not present in n5
```

Out[106… array([50, 60, 70])

In [112…
```
#numpy array mathematics

import numpy as np
n7=np.array([10,20,30,40])
n8=np.array([50,60,70,70])
np.sum([n7,n8])
```

Out[112… 350

In [15]:
```
#arithmetic operators


n9=np.array([1,2,3,4])
n9
n9+4                      #add  #sub  #mul #div
```

Out[15]: array([5, 6, 7, 8])

In [29]:
```
A=np.array([10,20,30])
B=np.array([40,50,60])
np.dot(A,B)               #matrix dot
```

Out[29]: 3200

In [37]:
```
#increment and decrement
c=np.array([1,2,3,4,5])
c+=4     # += # -+ # *=
c
```

Out[37]: array([5, 6, 7, 8, 9])

In [42]:
```
#universal function
#sqrt()
d=np.array([1,2,3,4,5])
np.sqrt(d)
```

Out[42]: array([1.   , 1.41421356, 1.73205081, 2.   , 2.23606798])

In [43]:
```
#Log()
e=np.array([1,2,3,4,5])
np.log(d)
```

Out[43]: array([0.   , 0.69314718, 1.09861229, 1.38629436, 1.60943791])

In [47]:
```
#trignometric (sin,cos,tan.....)
f=np.array([1,2,3,4,5])
np.sin(f)
```

Out[47]: array([0.54030231, -0.41614684, -0.9899925 , -0.65364362, 0.28366219])

In [9]:
```
#indexing
#slicing
#iterating
import numpy as np
g=np.array([10,11,12,13,14,15])
#g[4]      #index
g[::2]    #10-->12-->14     #slicing
```

Out[9]:
```
array([10, 12, 14])
```

In [12]:
```
h=np.array([10,20,30,40,50,60])
h[:4:3]
```

Out[12]:
```
array([10, 40])
```

In [66]:
```
# iterating an array
i1=[1,2,3,4,5]
for i in i1:
    print(i)
```

```
1
2
3
4
5
```

In [46]:
```
# condition and boolean arrays

i=np.array([10,20,30,40,50])
i
i<20
```

Out[46]:
```
array([ True, False, False, False, False])
```

In [71]:
```
#shape manipulation
#1)reshape()
#2)shape()
j=np.array([1.2,1.3,1.4,1.5,1.6,1.7])
j.reshape(3,2)
```

Out[71]:
```
array([[1.2, 1.3],
       [1.4, 1.5],
       [1.6, 1.7]])
```

In [69]:
```
#2) shape()
k=np.array([1.2,1.3,1.4,1.5,1.6,1.7])
k.shape=(3,2)
k
```

Out[69]:
```
array([[1.2, 1.3],
       [1.4, 1.5],
       [1.6, 1.7]])
```

In [75]:
```
k.ravel()
```

Out[75]:
```
array([1.2, 1.3, 1.4, 1.5, 1.6, 1.7])
```

In [77]:
```
k.transpose()
```

```
array([[1.2, 1.4, 1.6],
```

Out[77]:
```
       [1.3, 1.5, 1.7]])
```

In [38]:
```
#structured array


structured=np.array([(1,'First',0.5,1+2j),(2,'Second',1.3,2-2j),(3,'Third',0.8,1+3j)],d

structured
```

Out[38]:
```
array([(1, b'First', 0.5, 1.+2.j), (2, b'Second', 1.3, 2.-2.j),
       (3, b'Third', 0.8, 1.+3.j)],
      dtype=[('f0', '<i2'), ('f1', 'S6'), ('f2', '<f4'), ('f3', '<c8')])
```

In [47]:
```
# loding and saving data in binary files

#  for save data
name=({'nikhil':30,'kk':25,'Rl':26})
np.save('save_data',name)
```

In [70]:
```
# for load data(show)
load=np.load('save_data.npy')
load
```

In [ ]: