PIP install NUMPY

First one must install and import libary

In [1]:
```
import numpy as np
```

In [4]:
```
food =np.array(["Pakora" ,"Samosa" , "Raita" ])
food
```

Out[4]:
```
array(['Pakora', 'Samosa', 'Raita'], dtype='<U6')
```

In [5]:
```
price = np.array([5,5,5])
price
```

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

In [7]:
```
type(price) # here this type shows the dimension of arrary, in mathematics te
```

Out[7]:
```
numpy.ndarray
```

In [8]:
```
type(food)
```

Out[8]:
```
numpy.ndarray
```

In [10]:
```
# to check length of array we use following

len(food)
```

Out[10]:
```
3
```

In [11]:
```
price[2] # here we must remember index start from 0,1,2
```

Out[11]:
```
5
```

In [14]:
```
price[0:]
```

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

In [15]:
```
food[1] # to read samosa of array food, we must use index
```

Out[15]:
```
'Samosa'
```

In [16]:
```
price.mean() # we can do statistical operation of array using this
```

Out[16]: 5.0

MAKING of ARRAY METHODS

In [17]:
```
# Zeros this is first method that will prduce 5 zeros one dimensional arrary
np.zeros(5)
```

Out[17]: array([0., 0., 0., 0., 0.])

In [18]:
```
# for ones
np.ones(5)
```

Out[18]: array([1., 1., 1., 1., 1.])

In [19]:
```
#empty this empty will not produce and empty array but can give any array or
np.empty(5)
```

Out[19]: array([1., 1., 1., 1., 1.])

In [22]:
```
# RANGE
np.arange(10) # this gave range from zero to 10
```

Out[22]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [24]:
```
# Specifying the range
np.arange(2,20)
```

Out[24]:
```
array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
       19])
```

In [26]:
```
# making range with specific gap or interval ,
np.arange(2,20,2) # this 2 in last is interval
```

Out[26]: array([2, 4, 6, 8, 10, 12, 14, 16, 18])

In [28]:
```
# TABLE it will result in table of 5
np.arange(0,50,5)
```

Out[28]: array([0, 5, 10, 15, 20, 25, 30, 35, 40, 45])

In [31]:
```
# LINE SPACE it will distribute the number into equal parts of 5
np.linspace(1, 100, num=5)
```

Out[31]: array([1. , 25.75, 50.5 , 75.25, 100.])

In [34]:
```
# Specify data type , it gave 5 integer of ones, it can also give float if we
np.ones(50 , dtype=np.int64)
```

Out[34]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1], dtype=int64)
```

In [35]:
```
np.ones(50 , dtype=np.float64) # its now giving point values
```

Out[35]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Array functions

In [39]:
```
a = np.array([10,12,15,2,4,6,100,320,0.5,10.3]) # we can it convert all int t
print(a)
```

```
[ 10.   12.   15.    2.    4.    6.  100.  320.    0.5  10.3]
```

In [40]:
```
# sorting of array , it will sort in ascending order
a.sort()
print(a)
```

```
[  0.5   2.    4.    6.   10.   10.3  12.   15.  100.  320. ]
```

In [43]:
```
b = np.array([10.2,3.4,53.6,91.6,45.5])
print(b)
```

```
[10.2  3.4 53.6 91.6 45.5]
```

In [45]:
```
# we can combine or concatenate two arrarys
c=np.concatenate((a,b))
print(c)
```

```
[  0.5   2.    4.    6.   10.   10.3  12.   15.  100.  320.   10.2   3.4
  53.6  91.6  45.5]
```

In [46]:
```
c.sort()
print(c)
```

```
[  0.5   2.    3.4   4.    6.   10.   10.2  10.3  12.   15.   45.5  53.6
  91.6 100.  320. ]
```

2- D ARRARY

In [49]:
```
d = np.array([[1,2,3,4,5], [5,4,3,2,1]]) # arrary add easily in same dimesnio
print(d)
```

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

In [53]:
```
e = np.array([[6,7],[8,9]])
print(e)
```

```
[[6 7]
 [8 9]]
```

In [59]:
```
f = np.concatenate((d,e), axis=1) # we have to put axis as 1
print(f)
```

```
[[1 2 3 4 5 6 7]
 [5 4 3 2 1 8 9]]
```

In [63]:
```
g = np.array([[1,5], [6,10]])
h= np.array([[11,15] , [16,20]])
print(g)
print(h)
```

```
[[ 1  5]
 [ 6 10]]
[[11 15]
 [16 20]]
```

In [65]:
```
# now to concatenate in zero axis we the dimension must be same
i = np.concatenate((g,h), axis=0)
print(i) # in result we can see cleary it stack like follows frop top to botto
# from left to right axis = 1
```

```
[[ 1  5]
 [ 6 10]
 [11 15]
 [16 20]]
```

2 D Arrary

In [66]:
```
i.ndim # to check the dimension
```

Out[66]: 2

In [72]:
```
j = np.array([[[0,1,2,3],
               [4,5,6,7]]
              ,
                  [[0,1,2,3],
                   [4,5,6,7]]
               ,
                 [[0,1,2,3],
                 [4,5,6,7]]])
print(j)
```

```
[[[0 1 2 3]
  [4 5 6 7]]

 [[0 1 2 3]
  [4 5 6 7]]
```

```
 [[0 1 2 3]
```

In [73]:
```
j.ndim # to find the dimension , to make 3 dim arrary we make three 2 dim arra
```

Out[73]: 3

In [75]:
```
k=np.array([
             [5,6,7]
             ,
            [8,9,10]
            ,
            [10,11,12]
            ])
print(k)
```

```
[[ 5  6  7]
 [ 8  9 10]
 [10 11 12]]
```

In [76]:
```
k.ndim
```

Out[76]: 2

In [81]:
```
print(k.ndim)
print(k.shape)
```

```
2
(3, 3)
```

In [82]:
```
print(i.ndim)
print(i.shape)
```

```
2
(4, 2)
```

In [83]:
```
print(j.ndim)
print(j.shape)
```

```
3
(3, 2, 4)
```

just one arrary is one dimension arrary

array within arrary is two dimension arrary

an arrary with an array which is in another arrary is 3 dimension arrary

3d is actually triple stepdown index according

In [84]:
```
# to find size we can find size
```

In [89]:
```
k = np.arange(9) # this is 3*3
print(k)
```

```
[0 1 2 3 4 5 6 7 8]
```

In [91]:
```
# RESHAPE
l =k.reshape(3,3) # 3*3 = 9
print(l)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

In [94]:
```
# RESHAPE
np.reshape(k, newshape=(1,9), order='C')
```

Out[94]:
```
array([[0, 1, 2, 3, 4, 5, 6, 7, 8]])
```

Conversion of Arrays

In [96]:
```
# Convert 1-D array into 2D
m = np.array([1,2,3,4,5,6,7,8,9])
m.shape
```

Out[96]:
```
(9,)
```

In [101…
```
# Row wise 2 d conversion
n=m[np.newaxis, :]
print(n)
```

```
[[1 2 3 4 5 6 7 8 9]]
```

In [98]:
```
n.shape
```

Out[98]:
```
(1, 9)
```

In [99]:
```
n.ndim
```

Out[99]:
```
2
```

In [102…
```
# Colum wise 2 , d

o =m[ :, np.newaxis]
print(o)
```

```
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]
 [9]]
```

In [103…
```
# a[1:9] this will print array in index wise
# we can multiply, add, minu a number in the array
```

In [105…
```
o.mean()
```

Out[105… 5.0

In [106…
```
o.sum()
```

Out[106… 45

Generating 3d Array

In [107…
```
p=np.arange(24).reshape(2,3,4)
print(p)
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

In [ ]: