numpy practice session

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

In [3]:
```
# import this library in j.notebook
import numpy as np
```

creating an array Using numpy

In [5]:
```
# one D-array
import numpy as np
food = np.array(["pakora", "samosa", "raita"])
food
```

Out[5]:
```
array(['pakora', 'samosa', 'raita'], dtype='<U6')
```

In [6]:
```
price= np.array([5,5,5])
price
```

Out[6]:
```
array([5, 5, 5])
```

In [7]:
```
type(price)
```

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

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

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

In [9]:
```
len(price)
```

Out[9]:
```
3
```

In [10]:
```
price[2]
# its 0,1,2
# so it is 3 coloum of array
```

Out[10]:
```
5
```

In [11]:
```
price[0:]
```

Out[11]:
```
array([5, 5, 5])
```

In [12]:
```
food[1]
```

Out[12]:
```
'samosa'
```

In [13]:
```
# funcation of array
price.mean()
```

Out[13]:
```
5.0
```

In [14]:
```
# zero
np.zeros(6)
```

Out[14]:
```
array([0., 0., 0., 0., 0., 0.])
```

In [15]:
```
# ones
np.ones(5)
```

Out[15]:
```
array([1., 1., 1., 1., 1.])
```

In [17]:
```
#empty
np.empty(5)
```

Out[17]:
```
array([1., 1., 1., 1., 1.])
```

In [18]:
```
# range
np.arange(10)
```

Out[18]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [19]:
```
#specify
np.arange(2,20)
```

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

In [21]:
```
#specfic intervel
np.arange(2,20,2)
```

Out[21]:
```
array([ 2,  4,  6,  8, 10, 12, 14, 16, 18])
```

In [22]:
```
# table
np.arange(1,50,2)
```

Out[22]:
```
array([ 1,  3,  5,  7,  9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
       35, 37, 39, 41, 43, 45, 47, 49])
```

In [29]:
```
#line space
np.linspace(1,10,num=5)
# same result output of all
```

Out[29]:
```
array([ 1.  ,  3.25,  5.5 ,  7.75, 10.  ])
```

In [31]:
```
#specific your data type
np.ones(50, dtype=np.int64)
```

Out[31]:
```
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 [32]:
```
#specific your data type
np.ones(50, dtype=np.float64)
```

Out[32]:
```
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.])
```

array funcations

In [35]:
```
# array one
a= np.array([10,12,15,2,4,6,100,320,0.5,10.3])
a
```

Out[35]:
```
array([ 10. ,  12. ,  15. ,   2. ,   4. ,   6. , 100. , 320. ,   0.5,
       10.3])
```

In [36]:
```
a.sort()
a
```

Out[36]:
```
array([  0.5,   2. ,   4. ,   6. ,  10. ,  10.3,  12. ,  15. , 100. ,
       320. ])
```

In [41]:
```
# array two
b= np.array([10.2,3.4,53.6,91.6])
b
```

Out[41]:
```
array([10.2,  3.4, 53.6, 91.6])
```

In [43]:
```
np.concatenate((a,b))
# concatenate mean adding two array or more then two array
```

Out[43]:
```
array([  0.5,   2. ,   4. ,   6. ,  10. ,  10.3,  12. ,  15. , 100. ,
       320. ,  10.2,   3.4,  53.6,  91.6])
```

In [46]:
```
c=np.concatenate((a,b))
c
c.sort()
c
```

Out[46]:
```
array([  0.5,   2. ,   3.4,   4. ,   6. ,  10. ,  10.2,  10.3,  12. ,
        15. ,  53.6,  91.6, 100. , 320. ])
```

2-Da array

In [53]:
```
x =np.array([[1,2],[5,4]])
x
```

Out[53]:
```
array([[1, 2],
       [5, 4]])
```

In [54]:
```
y = np.array([[6,7],[7,8]])
```

In [ ]:
```
y
```

Out[54]:
```
array([[6, 7],
       [7, 8]])
```

In [56]:
```
#adding arrays
np.concatenate((x,y),axis=0)
```

Out[56]:
```
array([[1, 2],
       [5, 4],
       [6, 7],
       [7, 8]])
```

In [60]:
```
np.concatenate((x,y),axis=1)
```

Out[60]:
```
array([[1, 2, 6, 7],
       [5, 4, 7, 8]])
```

In [72]:
```
a=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]]])
a
```

Out[72]:
```
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]]])
```

In [73]:
```
# din find demintion number
a.ndim
```

Out[73]:
```
3
```

In [74]:
```
b=np.array([[5,6,7],
            [8,9,10],
           [10,11,12]])
```

In [75]:
```
b.ndim
```

Out[75]:
```
2
```

In [76]:
```
# array size (size mean number of elements)
a.size
```

Out[76]:
```
24
```

In [77]:
```
# shape for array of element matrix
a.shape
```

Out[77]:
```
(3, 2, 4)
```

In [79]:
```
a = np.arange(9)  #3*3
a
```

Out[79]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

In [80]:
```
# reshape of array or devepl array
b=a.reshape(3,3) #3*3=9
b
```

Out[80]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [81]:
```
# reshape
np.reshape(a,newshape=(1,9),order='c')
```

Out[81]:
```
array([[0, 1, 2, 3, 4, 5, 6, 7, 8]])
```

In [82]:
```
# convert ID To 2D array
a=np.array([1,2,3,4,5,6,7,8,9])
a
```

Out[82]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [83]:
```
a.shape
```

Out[83]:
```
(9,)
```

In [84]:
```
# row wise 2D convertion
b=a[np.newaxis, :]
b
```

Out[84]:
```
array([[1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

In [85]:
```
b.shape
# now it is converted to 2D array
```

Out[85]:
```
(1, 9)
```

In [87]:
```
# colume wise 2D convertion
c=a[:, np.newaxis]
c
```

Out[87]:
```
array([[1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
```

```
       [8],
       [9]])
```

In [88]:
```
a
```

Out[88]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [90]:
```
#silcing
a[2:9]
```

Out[90]:
```
array([3, 4, 5, 6, 7, 8, 9])
```

In [93]:
```
  # a multiple to six 6
a*6
```

Out[93]:
```
array([ 6, 12, 18, 24, 30, 36, 42, 48, 54])
```

In [95]:
```
# sum of array
a.sum()
```

Out[95]:
```
45
```

In [97]:
```
# also mean
a.mean()
```

Out[97]:
```
5.0
```

In [ ]:
```

```