NumPy Functions :

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

arange()

In [ ]:
```
#np.arange(START, END, STEP)
ar_1d = np.arange(1, 25, 2)
print(ar_1d)
```

```
[ 1  3  5  7  9 11 13 15 17 19 21 23]
```

In [ ]:
```
even_ar = np.arange(1, 13, 2)
even_ar
```

Out[ ]:
```
array([ 1,  3,  5,  7,  9, 11])
```

linspace()

In [ ]:
```
## we use linspace function when neeed values with equal gap
#np.linspace(start, end, how many values we need in start-end range)
np.linspace(1, 10, num=10)
```

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

reshape()

In [ ]:
```
## when we need to convert 1D arary into 2-3D array
```

In [ ]:
```
ar_2d = ar_1d.reshape(3,4)
ar_2d
```

Out[ ]:
```
array([[ 1,  3,  5,  7],
       [ 9, 11, 13, 15],
       [17, 19, 21, 23]])
```

In [ ]:
```
## conversion of 1D to 3D
#Array name.reshape(3D,Rows,Column)
ar_3d = ar_1d.reshape(2,3,2)
ar_3d
```

Out[ ]:
```
array([[[ 1,  3],
        [ 5,  7],
        [ 9, 11]],

       [[13, 15],
        [17, 19],
        [21, 23]]])
```

In [ ]:
```
arr=np.arange(1,13).reshape(4,3)
print(ar)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

Ravel()

In [ ]:
```
#Ravel Function convert Multi-D to 1D
```

In [ ]:
```
ar.ravel()
```

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

Flatten()

In [ ]:
```
ar.flatten()
```

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

In [ ]:
```
#'C' means to flatten in row-major (C-style) order.
ar.flatten(order= 'C')
```

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

In [ ]:
```
# 'F' means to flatten in column-major (Fortran- style) order.
ar.flatten(order='F')
```

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

In [ ]:
```
# 'A' means to flatten in column-major order if a is Fortran contiguous in memory, row-major order otherwise.
ar.flatten(order='A')
```

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

In [ ]:
```
# 'K' means to flatten a in the order the elements occur in memory. The default is 'C'.
ar.flatten(order='K')
```

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

Transpose()

In [ ]:
```
# convert Row into Column
```

In [ ]:
```
ar
```

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

In [ ]:
```
ar.transpose()
```

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

In [ ]:
```
## short key for transpose
ar.T
```

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

Concatenate & Sort( )

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

In [ ]:
```
a= np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14])
```

In [ ]:
```
b=np.array([20,30,40,50,60,70,80])
```

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

Out[ ]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 20, 30, 40,
       50, 60, 70, 80])
```

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

Out[ ]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 20, 30, 40,
       50, 60, 70, 80])
```

2-D Arrays

In [ ]:
```
d2 = np.array([[5,67,20],
               [10,20,30],
               [30,40,50]])
d2
```

Out[ ]:
```
array([[ 5, 67, 20],
       [10, 20, 30],
       [30, 40, 50]])
```

In [ ]:
```
d2.ndim
```

Out[ ]:
```
2
```

In [ ]:
```
d3=np.array([[[1,2,3],[1,2,3],[1,2,3]],
             [[1,2,3],[1,2,3],[1,2,3]],[[1,2,3],[1,2,3],[1,2,3]]])
d3
```

Out[ ]:
```
array([[[1, 2, 3],
        [1, 2, 3],
        [1, 2, 3]],

       [[1, 2, 3],
        [1, 2, 3],
        [1, 2, 3]],

       [[1, 2, 3],
        [1, 2, 3],
        [1, 2, 3]]])
```

In [ ]:
```
d3.ndim
```

Out[ ]:
```
3
```

In [ ]:
```
d3.size
```

Out[ ]:
```
27
```

In [ ]:
```
d3.shape
```

Out[ ]:
```
(3, 3, 3)
```

In [ ]:
```
d3.sort()
d3
```

Out[ ]:
```
array([[[1, 2, 3],
        [1, 2, 3],
        [1, 2, 3]],

       [[1, 2, 3],
        [1, 2, 3],
        [1, 2, 3]],

       [[1, 2, 3],
        [1, 2, 3],
        [1, 2, 3]]])
```

In [ ]:
```
new=np.arange(9)
new
```

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

In [ ]:
```
new.reshape(3,3)
```

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

In [ ]:
```
np.reshape(new, newshape=(1,9),order="F")
```

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

Reshape with New Axis

In [ ]:
```
a=np.arange(1,10)
a
```

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

In [ ]:
```
# Row Wise
b=a[np.newaxis, :]
b
```

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

In [ ]:
```
# Column Wise
c=a[:,np.newaxis]
c
```

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

In [ ]:
```
a[2:5]
```

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

In [ ]:
```

```