# NumPy - Array Manipulation

In [1]:

```
import numpy as np
```

In [2]:

```
arr = np.random.randint(1,10 ,(3,4))
```

In [3]:

```
arr
```

Out[3]:

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

In [4]:

```
arr.reshape(6,2)
```

Out[4]:

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

In [5]:

```
arr.reshape(2,6)
```

Out[5]:

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

In [6]:

```
arr
```

Out[6]:

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

In [7]:

```
arr.T
```

Out[7]:

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

In [8]:

```
arr.flatten()
```

Out[8]:

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

In [9]:

```
arr1 = np.array([1,2,3,3,4])
```

In [11]:

```
arr1.ndim
```

Out[11]:

```
1
```

In [12]:

```
np.expand_dims(arr1, axis=1)
```

Out[12]:

```
array([[1],
       [2],
       [3],
       [3],
       [4]])
```

In [13]:

```
np.expand_dims(arr1, axis=0)
```

Out[13]:

```
array([[1, 2, 3, 3, 4]])
```

In [14]:

```
arr
```

Out[14]:

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

In [15]:

```
np.squeeze(arr)
```

Out[15]:

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

In [17]:

```
data = np.array([[1],[2],[3]])
```

In [18]:

```
data
```

Out[18]:

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

In [19]:

```
np.squeeze(data)
```

Out[19]:

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

In [20]:

```
arr1
```

Out[20]:

```
array([1, 2, 3, 3, 4])
```

In [21]:

```
np.repeat(arr1, 3)
```

Out[21]:

```
array([1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4])
```

In [22]:

```
np.roll(arr1 ,3)
```

Out[22]:

```
array([3, 3, 4, 1, 2])
```

In [23]:

```
np.diag(arr1)
```

Out[23]:

```
array([[1, 0, 0, 0, 0],
       [0, 2, 0, 0, 0],
       [0, 0, 3, 0, 0],
       [0, 0, 0, 3, 0],
       [0, 0, 0, 0, 4]])
```

# NumPy - Binary Operators

In [24]:

```
arr1 = np.random.randint(1,10 , (3,4))
```

In [25]:

```
arr2 = np.random.randint(1,10 , (3,4))
```

In [26]:

```
arr1
```

Out[26]:

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

In [27]:

```
arr2
```

Out[27]:

```
array([[1, 3, 3, 3],
       [9, 4, 7, 5],
       [5, 5, 8, 4]])
```

In [28]:

```
arr1 + arr2
```

Out[28]:

```
array([[ 3,  9,  7,  4],
       [17,  6,  8, 13],
       [14, 12, 13, 11]])
```

In [29]:

```
arr1*arr2
```

Out[29]:

```
array([[ 2, 18, 12,  3],
       [72,  8,  7, 40],
       [45, 35, 40, 28]])
```

In [30]:

```
arr1/arr2
```

Out[30]:

```
array([[2.        , 2.        , 1.33333333, 0.33333333],
       [0.88888889, 0.5       , 0.14285714, 1.6       ],
       [1.8       , 1.4       , 0.625     , 1.75      ]])
```

In [31]:

```
arr1-arr2
```

Out[31]:

```
array([[ 1,  3,  1, -2],
       [-1, -2, -6,  3],
       [ 4,  2, -3,  3]])
```

In [32]:

```
arr1%arr2
```

Out[32]:

```
array([[0, 0, 1, 1],
       [8, 2, 1, 3],
       [4, 2, 5, 3]])
```

In [33]:

```
arr1**arr2
```

Out[33]:

```
array([[        2,       216,        64,         1],
       [134217728,        16,         1,     32768],
       [    59049,     16807,    390625,      2401]])
```

In [34]:

```
arr1
```

Out[34]:

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

In [35]:

```
arr2
```

Out[35]:

```
array([[1, 3, 3, 3],
       [9, 4, 7, 5],
       [5, 5, 8, 4]])
```

In [36]:

```
arr1 & arr2
```

Out[36]:

```
array([[0, 2, 0, 1],
       [8, 0, 1, 0],
       [1, 5, 0, 4]])
```

In [37]:

```
~arr1
```

Out[37]:

```
array([[ -3,  -7,  -5,  -2],
       [ -9,  -3,  -2,  -9],
       [-10,  -8,  -6,  -8]])
```

In [38]:

```
arr1|arr2
```

Out[38]:

```
array([[ 3,  7,  7,  3],
       [ 9,  6,  7, 13],
       [13,  7, 13,  7]])
```

In [39]:

```
arr1>arr2
```

Out[39]:

```
array([[ True,  True,  True, False],
       [False, False, False,  True],
       [ True,  True, False,  True]])
```

# NumPy - String Functions.

In [41]:

```
np.array(["mujahid" , "raza"])
```

Out[41]:

```
array(['mujahid', 'raza'], dtype='<U7')
```

In [42]:

```
arr = np.array(["mujahid" , "raza"])
```

In [43]:

```
arr
```

Out[43]:

```
array(['mujahid', 'raza'], dtype='<U7')
```

In [44]:

```
np.char.upper(arr)
```

Out[44]:

```
array(['MUJAHID', 'RAZA'], dtype='<U7')
```

In [47]:

```
np.char.title(arr)
```

Out[47]:

```
array(['Mujahid', 'Raza'], dtype='<U7')
```

In [48]:

```
np.char.capitalize(arr)
```

Out[48]:

```
array(['Mujahid', 'Raza'], dtype='<U7')
```

# NumPy - Mathematical Functions

In [49]:

```
arr1
```

Out[49]:

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

In [50]:

```
np.sin(arr1)
```

Out[50]:

```
array([[ 0.90929743, -0.2794155 , -0.7568025 ,  0.84147098],
       [ 0.98935825,  0.90929743,  0.84147098,  0.98935825],
       [ 0.41211849,  0.6569866 , -0.95892427,  0.6569866 ]])
```

In [51]:

```
np.cos(arr1)
```

Out[51]:

```
array([[-0.41614684,  0.96017029, -0.65364362,  0.54030231],
       [-0.14550003, -0.41614684,  0.54030231, -0.14550003],
       [-0.91113026,  0.75390225,  0.28366219,  0.75390225]])
```

In [52]:

```
np.tan(arr1)
```

Out[52]:

```
array([[-2.18503986, -0.29100619,  1.15782128,  1.55740772],
       [-6.79971146, -2.18503986,  1.55740772, -6.79971146],
       [-0.45231566,  0.87144798, -3.38051501,  0.87144798]])
```

In [53]:

```
np.tanh(arr1)
```

Out[53]:

```
array([[0.96402758, 0.99998771, 0.9993293 , 0.76159416],
       [0.99999977, 0.96402758, 0.76159416, 0.99999977],
       [0.99999997, 0.99999834, 0.9999092 , 0.99999834]])
```

In [54]:

```
np.log10(arr1)
```

Out[54]:

```
array([[0.30103   , 0.77815125, 0.60205999, 0.        ],
       [0.90308999, 0.30103   , 0.        , 0.90308999],
       [0.95424251, 0.84509804, 0.69897   , 0.84509804]])
```

In [55]:

```
np.exp(arr1)
```

Out[55]:

```
array([[7.38905610e+00, 4.03428793e+02, 5.45981500e+01, 2.71828183e+00],
       [2.98095799e+03, 7.38905610e+00, 2.71828183e+00, 2.98095799e+03],
       [8.10308393e+03, 1.09663316e+03, 1.48413159e+02, 1.09663316e+03]])
```

In [56]:

```
np.sqrt(arr1)
```

Out[56]:

```
array([[1.41421356, 2.44948974, 2.        , 1.        ],
       [2.82842712, 1.41421356, 1.        , 2.82842712],
       [3.        , 2.64575131, 2.23606798, 2.64575131]])
```

In [58]:

```
np.power(arr1,2)
```

Out[58]:

```
array([[ 4, 36, 16,  1],
       [64,  4,  1, 64],
       [81, 49, 25, 49]])
```

In [59]:

```
np.mean(arr1)
```

Out[59]:

```
5.0
```

In [61]:

```
np.median(arr1)
```

Out[61]:

```
5.5
```

In [62]:

```
np.std(arr1)
```

Out[62]:

```
2.798809270624444
```

In [63]:

```
np.min(arr1)
```

Out[63]:

```
1
```

In [64]:

```
np.max(arr1)
```

Out[64]:

```
9
```

## Sort, Search & Counting Functions

In [1]:

```
import numpy as np
```

In [3]:

```
arr = np.array([4,3,4,5,6,76,7,7,678,7,9,4])
```

In [4]:

```
arr
```

Out[4]:

```
array([  4,   3,   4,   5,   6,  76,   7,   7, 678,   7,   9,   4])
```

In [5]:

```
np.sort(arr)
```

Out[5]:

```
array([  3,   4,   4,   4,   5,   6,   7,   7,   7,   9,  76, 678])
```

In [6]:

```
np.searchsorted(arr, 32)
```

Out[6]:

```
12
```

In [7]:

```
arr1 = np.array([0,343,565,0,0,0,0,0,0])
```

In [8]:

```
np.count_nonzero(arr1)
```

Out[8]:

```
2
```

In [9]:

```
arr
```

Out[9]:

```
array([  4,   3,   4,   5,   6,  76,   7,   7, 678,   7,   9,   4])
```

In [10]:

```
np.where(arr>6)
```

Out[10]:

```
(array([ 5,  6,  7,  8,  9, 10]),)
```

In [12]:

```
np.extract(arr>3 , arr)
```

Out[12]:

```
array([  4,   4,   5,   6,  76,   7,   7, 678,   7,   9,   4])
```

## NumPy - Byte Swapping

In [14]:

```
arr.byteswap()
```

Out[14]:

```
array([ 288230376151711744,   216172782113783808,   288230376151711744,
        360287970189639680,   432345564227567616,  5476377146882523136,
        504403158265495552,   504403158265495552, -6484620513460092928,
        504403158265495552,   648518346341351424,   288230376151711744])
```

# NumPy - Copies & Views

In [15]:

```
arr
```

Out[15]:

```
array([  4,   3,   4,   5,   6,  76,   7,   7, 678,   7,   9,   4])
```

In [16]:

```
a = np.copy(arr)
```

In [17]:

```
b = arr.view()
```

In [18]:

```
b
```

Out[18]:

```
array([  4,   3,   4,   5,   6,  76,   7,   7, 678,   7,   9,   4])
```

In [19]:

```
a
```

Out[19]:

```
array([  4,   3,   4,   5,   6,  76,   7,   7, 678,   7,   9,   4])
```

In [20]:

```
b[0] = 45
```

In [21]:

```
b
```

Out[21]:

```
array([ 45,   3,   4,   5,   6,  76,   7,   7, 678,   7,   9,   4])
```

In [22]:

```
a
```

Out[22]:

```
array([  4,   3,   4,   5,   6,  76,   7,   7, 678,   7,   9,   4])
```

In [23]:

```
arr
```

Out[23]:

```
array([ 45,   3,   4,   5,   6,  76,   7,   7, 678,   7,   9,   4])
```

# NumPy - Matrix Library

In [26]:

```
import numpy.matlib as nm
```

In [27]:

```
nm.zeros(5)
```

Out[27]:

```
matrix([[0., 0., 0., 0., 0.]])
```

In [28]:

```
nm.ones((3,4))
```

Out[28]:

```
matrix([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
```

In [29]:

```
nm.eye(5)
```

Out[29]:

```
matrix([[1., 0., 0., 0., 0.],
        [0., 1., 0., 0., 0.],
        [0., 0., 1., 0., 0.],
        [0., 0., 0., 1., 0.],
        [0., 0., 0., 0., 1.]])
```

# NumPy - Linear Algebra

In [31]:

```
arr1 = np.random.randint([[2,3] , [4,5]])
```

In [34]:

```
arr2 = np.random.randint([[5,3] , [2,5]])
```

In [35]:

```
np.dot(arr1,arr2)
```

Out[35]:

```
array([[0, 2],
       [1, 5]])
```

In [36]:

```
arr1@arr2
```

Out[36]:

```
array([[0, 2],
       [1, 5]])
```

In [37]:

```
np.cross(arr1,arr2)
```

Out[37]:

```
array([-2,  1])
```

In [ ]:

```

```