In [15]:

```
#importing numpy
import numpy as np
#creating a basic array
a=np.array([23,56,12,90,55,4])
a
#datatype of array
a.dtype
```

Out[15]:

```
dtype('int32')
```

In [16]:

```
np.zeros(2)
```

Out[16]:

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

In [5]:

```
np.ones(3)
```

Out[5]:

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

In [6]:

```
np.empty(2)
```

Out[6]:

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

In [8]:

```
# create an array with a range of elements:
np.arange(5)
```

Out[8]:

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

In [9]:

```
# an array that contains a range of evenly spaced intervals
#specify the first number, last number, and the step size.
np.arange(2,8,2)
```

Out[9]:

```
array([2, 4, 6])
```

In [10]:

```
#np.linspace(start = 0, stop = 100, num = 5)
#create an array with values that are spaced linearly in a specified interval
np.linspace(0,10,num=5)
```

Out[10]:

```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

In [12]:

```
np.linspace(10,100,num=10)
```

Out[12]:

```
array([ 10.,  20.,  30.,  40.,  50.,  60.,  70.,  80.,  90., 100.])
```

In [14]:

```

```

Out[14]:

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

In [17]:

```
a=np.array([23,56,78,90,45,21,1])
#sorting an elements
sort=np.sort(a)
sort
```

Out[17]:

```
array([ 1, 21, 23, 45, 56, 78, 90])
```

In [18]:

```
b=np.array([2,4,5,6,33,43])
#concatinating arrays
np.concatenate((a,b))
```

Out[18]:

```
array([23, 56, 78, 90, 45, 21,  1,  2,  4,  5,  6, 33, 43])
```

In [21]:

```
np.concatenate((a,b),axis=0)
```

Out[21]:

```
array([23, 56, 78, 90, 45, 21,  1,  2,  4,  5,  6, 33, 43])
```

In [24]:

```
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6]])
np.concatenate((x,y),axis=0)
```

Out[24]:

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

In [30]:

```
#ndarray.ndim will tell you the number of axes, or dimensions, of the array.
array_example = 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]]])
array_example.ndim
```

Out[30]:

```
3
```

In [36]:

```
array1=np.array([[1,2,3,4],[5,6,7,8]])
array1.ndim
```

Out[36]:

```
2
```

In [37]:

```
#ndarray.size will tell you the total number of elements of the array.
array1.size
```

Out[37]:

```
8
```

In [38]:

```
# will display a tuple of integers that indicate the number of elements stored a
array1.shape
```

Out[38]:

```
(2, 4)
```

In [40]:
```
# a new shape to an array without changing the data
array1.reshape(4,2)
```

Out[40]:
```
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
```

In [44]:
```
np.reshape(array1, newshape=(1, 8), order='C')
```

Out[44]:
```
array([[1, 2, 3, 4, 5, 6, 7, 8]])
```

In [48]:
```
#Using np.newaxis will increase the dimensions of your array by one dimension whe
c=np.array([1,2,3,4,5])
c.shape
c.ndim
```

Out[48]:
```
1
```

In [49]:
```
c2=c[np.newaxis, :]
c2.shape
c2.ndim
```

Out[49]:
```
2
```

In [50]:
```
#to add an axis at index position 1 with
c=np.expand_dims(c2,axis=0)
c.shape
```

Out[50]:
```
(1, 1, 5)
```

In [2]:
```
#indexcing and slicing
import numpy as np
data=np.array([34,56,78,93])
data[0]
```

Out[2]:
```
34
```

In [3]:
```
data[0:2]
```

Out[3]:
```
array([78, 93])
```

In [4]:
```
data[1:]
```

Out[4]:
```
array([56, 78, 93])
```

In [5]:
```

data[-2:]
```

Out[5]:
```
array([78, 93])
```

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

Out[6]: dtype('int32')
```

```
In [7]: a.ndim

Out[7]: 2
```

```
In [8]: print(a[a<5])

        [1 2 3 4]
```

```
In [9]: print(a[a%2==0])

        [ 2  4  6  8 10 12]
```

```
In [11]: print(a[(a>2) & (a<11)])

         [ 3  4  5  6  7  8  9 10]
```

Type *Markdown* and LaTeX: α^2

```
In [12]: a.sum()

Out[12]: 78
```

```
In [13]: #operation between a vector and a scalar
         data=np.array([1.0,2.0])
         data*1.6

Out[13]: array([1.6, 3.2])
```

```
In [14]: a=np.array([1,2,3,4,5])
         a.sum()

Out[14]: 15
```

```
In [15]: a.min()

Out[15]: 1
```

```
In [16]: a.max()

Out[16]: 5
```

```
In [17]: a.mean()

Out[17]: 3.0
```

In [19]:

```
a.dtype
```

Out[19]:

```
dtype('int32')
```

In [20]:

```
#creating matrices
data=np.array([[1,2],[3,4],[5,6]])
data.ndim
```

Out[20]:

```
2
```

In [21]:

```
data
```

Out[21]:

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

In [22]:

```
data[0,1]
```

Out[22]:

```
2
```

In [25]:

```
data[1,0]
```

Out[25]:

```
3
```

In [26]:

```
data[2,1]
```

Out[26]:

```
6
```

In [27]:

```
data.max(axis=0)
```

Out[27]:

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

In [29]:

```
#reversing an array
reverse=np.flip(a)
```

In [ ]:

```

```