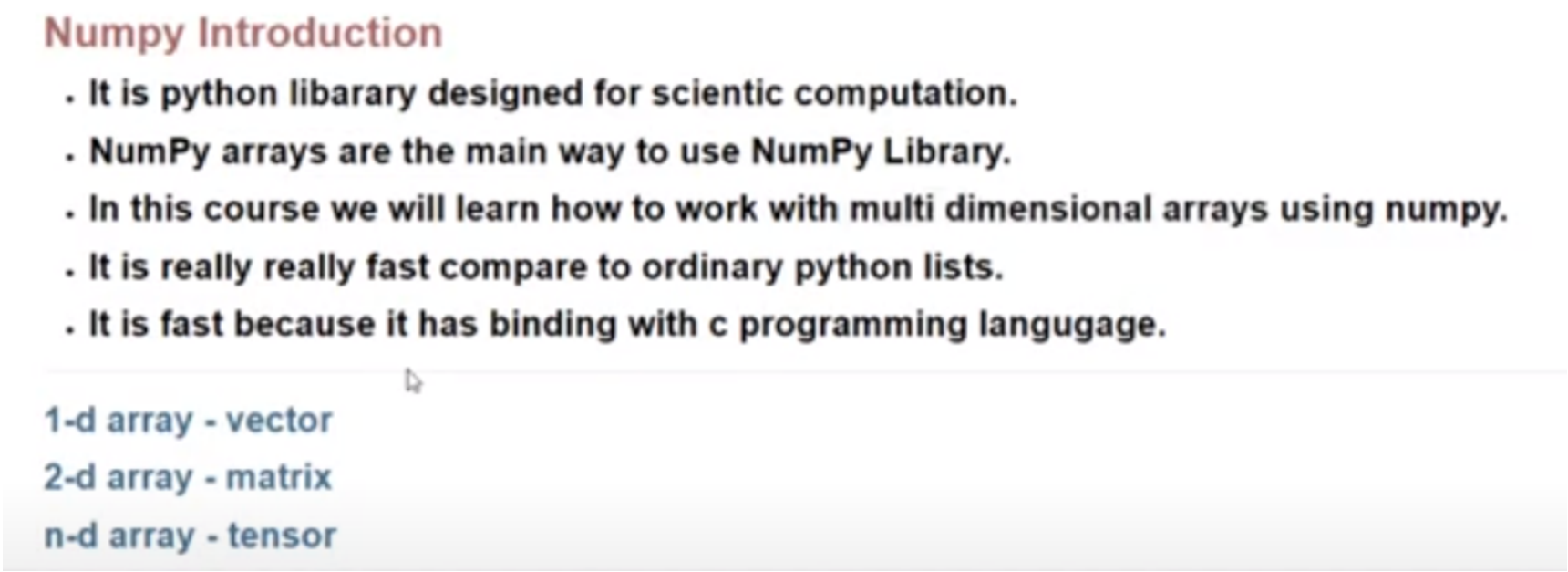
Introduction



Create Numpy Arrays

In [2]:
```
import numpy as np
```

In [24]:
```
a=np.array([1,2,3])
a
```

Out[24]:
```
array([1, 2, 3])
```

In [25]:
```
type(a)
```

Out[25]:
```
numpy.ndarray
```

In [13]:
```
a.ndim
```

Out[13]:
```
1
```

In [14]:
```
a.shape
```

Out[14]:
```
(3,)
```

In [26]:
```
b=np.array([[1,2,3],[4,5,6]])
b
```

Out[26]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [10]:
```
type(b)
```

Out[10]:
```
numpy.ndarray
```

In [11]:
```
b.ndim
```

Out[11]:
```
2
```

In [15]:
```
b.shape
```

Out[15]:
```
(2, 3)
```

In [16]:
```
b.dtype
```

Out[16]:
```
dtype('int32')
```

Matrix Creation

np.zeros

In [19]:
```
arr=np.zeros(5)
arr
```

Out[19]:
```
array([0., 0., 0., 0., 0.])
```

In [27]:
```
arr.dtype
```

Out[27]:
```
dtype('float64')
```

In [3]:
```
arr1=np.zeros((5,), dtype=int)
```

In [4]:
```
arr1
```

Out[4]:
```
array([0, 0, 0, 0, 0])
```

In [33]:
```
arr1.dtype
```

Out[33]:
```
dtype('int32')
```

In [22]:
```
arr=np.zeros((2,3))
arr
```

Out[22]:
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

np.ones

In [23]:
```
arr=np.ones(5)
print(arr)
```

```
[1. 1. 1. 1. 1.]
```

In [24]:
```
arr=np.ones((3,3))
print(arr)
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

In [25]:
```
print(1+np.ones((3,3)))
```

```
[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]
```

np.eye

In [27]:
```
arr=np.eye((2),dtype=int)
print(arr)
```

```
[[1 0]
 [0 1]]
```

In [29]:
```
arr=np.eye((3),dtype=int)
print(arr)
print(arr.shape)
```

```
[[1 0 0]
 [0 1 0]
 [0 0 1]]
(3, 3)
```

np.diag

In [47]:
```
arr=np.diag([1,2,3])
arr
```

Out[47]:
```
array([[1, 0, 0],
       [0, 2, 0],
       [0, 0, 3]])
```

In [ ]:
```
arr=np.diag([1,2,3])
arr
```

Generate Random Float

In [38]:
```
r=np.random.rand(2,2)#Random number generation (from Uniform distribution  from 0 to 1
r
```

Out[38]:
```
array([[0.70898963, 0.19630521],
       [0.28554925, 0.92244737]])
```

In [40]:
```
#Random number generation from Normal distribution with zero mean and standard deviation 1 i.e. standard normal")
r=np.random.randn(2,2)
r
```

Out[40]:
```
array([[-0.08396783,  0.77999775],
       [-0.15600502, -0.14903589]])
```

Generate Random Array

In [104]:
```
# Generate a 1-D array containing 5 random integers from 0 to 100:
#randint (low, high, # of samples to be drawn)
rand_arr=np.random.randint(10,size=(5))
rand_arr
```

Out[104]:
```
array([5, 1, 0, 2, 7])
```

In [109]:
```
# Generate a 2-D array with 3 rows, each row containing 5 random integers from 0 to 100:
rand_arr=np.random.randint(10,size=(2,3))
rand_arr
```

Out[109]:
```
array([[4, 0, 5],
       [3, 9, 1]])
```

In [110]:
```
rand_arr=np.random.rand(5)
rand_arr
```

Out[110]:
```
array([0.82710566, 0.85312883, 0.53820293, 0.66561322, 0.18900387])
```

In [173]:
```
min=10
max=20
rand_arr=min+((max-min)*np.random.rand())
print(rand_arr)
```

```
15.678671645203169
```

reshaping

In [183]:
```
arr=np.random.randint(1,50,10)
arr
```

Out[183]:
```
array([24, 21, 29, 43, 48, 14, 24, 21, 25, 48])
```

In [184]:
```
arr.shape
```

Out[184]:
```
(10,)
```

In [185]:
```
arr=arr.reshape(2,5)
arr
```

Out[185]:
```
array([[24, 21, 29, 43, 48],
       [14, 24, 21, 25, 48]])
```

In [186]:
```
arr.shape
```

Out[186]:
```
(2, 5)
```

ravel()

In [17]:
```
a=np.array([[1,2,3],[4,5,6]])
a
```

Out[17]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [16]:
```
a.ravel() # flattening into 1d array
```

Out[16]:
```
array([1, 2, 3, 4, 5, 6])
```

arange

In [6]:
```
# Return evenly spaced values within a given interval.
# A series of numbers from low to high
np.arange(1,10,2)
```

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

In [7]:
```
np.arange(1,10,3)
```

Out[7]:
```
array([1, 4, 7])
```

In [8]:
```
list(range(1,10,2))
```

Out[8]:
```
[1, 3, 5, 7, 9]
```

In [34]:
```
np.arange(10,0,-1)
```

Out[34]:
```
array([10,  9,  8,  7,  6,  5,  4,  3,  2,  1])
```

linspace

In [12]:
```
# arange uses a step size
# linspace uses the number of samples
```

In [15]:
```
np.linspace(1,10,5)
```

Out[15]:
```
array([ 1.  ,  3.25,  5.5 ,  7.75, 10.  ])
```

In [17]:
```
np.linspace(1,10,10,dtype=int)
```

Out[17]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

Assigning value

In [41]:
```
arr=np.diag([1,2,3])
arr
```

Out[41]:
```
array([[1, 0, 0],
       [0, 2, 0],
       [0, 0, 3]])
```

In [42]:
```
arr[2,1]=4
arr
```

Out[42]:
```
array([[1, 0, 0],
       [0, 2, 0],
       [0, 4, 3]])
```

Slicing

In [61]:
```
a=np.arange(10,20)
a
```

Out[61]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

In [62]:
```
a[1:8:2] #[startindex:endindex(exclusive):step]
```

Out[62]:
```
array([11, 13, 15, 17])
```

In [63]:
```
a[[0,2,4]] #indexing can be done with array of integers
```

Out[63]:
```
array([10, 12, 14])
```

In [ ]:
```
# combination of assignment and slicing
```

In [64]:
```
a=np.arange(10)
```

In [65]:
```
b=np.arange(5)
a[5:]=b[::-1]
a
```

Out[65]:
```
array([0, 1, 2, 3, 4, 4, 3, 2, 1, 0])
```

In [53]:
```
a=np.arange(10)
print(a)
b=a[::2]
print(b)
```

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

In [55]:
```
b[0]=10
b
```

Out[55]:
```
array([10,  2,  4,  6,  8])
```

In [56]:
```
a
```

Out[56]:
```
array([10,  1,  2,  3,  4,  5,  6,  7,  8,  9])
```

In [54]:
```
# a slicing operation creates a view on the original array, which is just a way of accessing array data.
# Thus the original array is not copied in memory.
np.shares_memory(a,b) # used to check if two arrays share the same memory block.
```

Out[54]:
```
True
```

In [57]:
```
a=np.arange(10)
c=a[::2].copy() # force copy
c
```

Out[57]:
```
array([0, 2, 4, 6, 8])
```

In [58]:
```
np.shares_memory(a,c)
```

Out[58]:
```
False
```

Operations on numpy

In [ ]:
```
# 1. Basic Operations with scalars(constant)
```

In [21]:
```
a=np.array([1,2,3,4])
a+1
```

Out[21]:
```
array([2, 3, 4, 5])
```

In [22]:
```
a**2
```

Out[22]:
```
array([ 1,  4,  9, 16], dtype=int32)
```

In [23]:
```
a=np.array([1,2,3,4]).reshape(2,2)
b=np.array([1,2,3,4]).reshape(2,2)
print('a=',a)
print()
print('b=',b)
```

```
a= [[1 2]
 [3 4]]

b= [[1 2]
 [3 4]]
```

In [24]:
```
a+b
```

Out[24]:
```
array([[2, 4],
       [6, 8]])
```

In [25]:
```
a-b
```

Out[25]:
```
array([[0, 0],
       [0, 0]])
```

In [26]:
```
a*b
```

Out[26]:
```
array([[ 1,  4],
       [ 9, 16]])
```

In [27]:
```
a.dot(b)
```

Out[27]:
```
array([[ 7, 10],
       [15, 22]])
```

In [36]:
```
a/b
```

Out[36]:
```
array([[1., 1.],
       [1., 1.]])
```

In [28]:
```
a
```

Out[28]:
```
array([[1, 2],
       [3, 4]])
```

Statistics

In [29]:
```
a.min()
```

Out[29]:
```
1
```

In [30]:
```
a.max()
```

Out[30]:
```
4
```

In [45]:
```
a.argmin() #index of minimum element
```

Out[45]:
```
0
```

In [46]:
```
a.argmax() #index of maximum element
```

Out[46]:
```
3
```

In [31]:
```
a.sum()
```

Out[31]:
```
10
```

In [32]:
```
a.sum(axis=0)
```

Out[32]:
```
array([4, 6])
```

In [33]:
```
a.sum(axis=1)
```

Out[33]:
```
array([3, 7])
```

In [37]:
```
np.sqrt(a*b)
```

Out[37]:
```
array([[1., 2.],
       [3., 4.]])
```

In [38]:
```
np.std(a)
```

Out[38]:
```
1.1180339887498895
```

In [39]:
```
a.mean()
```

Out[39]:
```
2.5
```

In [40]:
```
a.median()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-40-d799d54f1221> in <module>
----> 1 a.median()

AttributeError: 'numpy.ndarray' object has no attribute 'median'
```

In [41]:
```
np.median(a)
```

Out[41]:
```
2.5
```

In [42]:
```
a
```

Out[42]:
```
array([[1, 2],
       [3, 4]])
```

In [44]:
```
np.median(a,axis=0)
```

Out[44]:
```
array([2., 3.])
```

Transpose

In [48]:
```
a.T
```

Out[48]:
```
array([[1, 3],
       [2, 4]])
```