teleuniv
innovative interactive immersive

Python: NumPy

teleuniv
innovative interactive immersive

Machine Learning

Session - 5

Previous sessions:

Data Types, Collections

Control Statements, Operators

This Session:

NumPy

Introduction

NumPy : Package for N-Dimensional Array (called ndarray)

Provides an array and supporting mathematical functions

Array of Multidimensional, uniform collection of elements

Array is characterized with Data type and shape

Supports up to 32 dimensions

Supports to create new data type like structures in C

teleuniv
innovative interactive immersive

NumPy Array Structure

Machine Learning

Describes memory with the attributes:

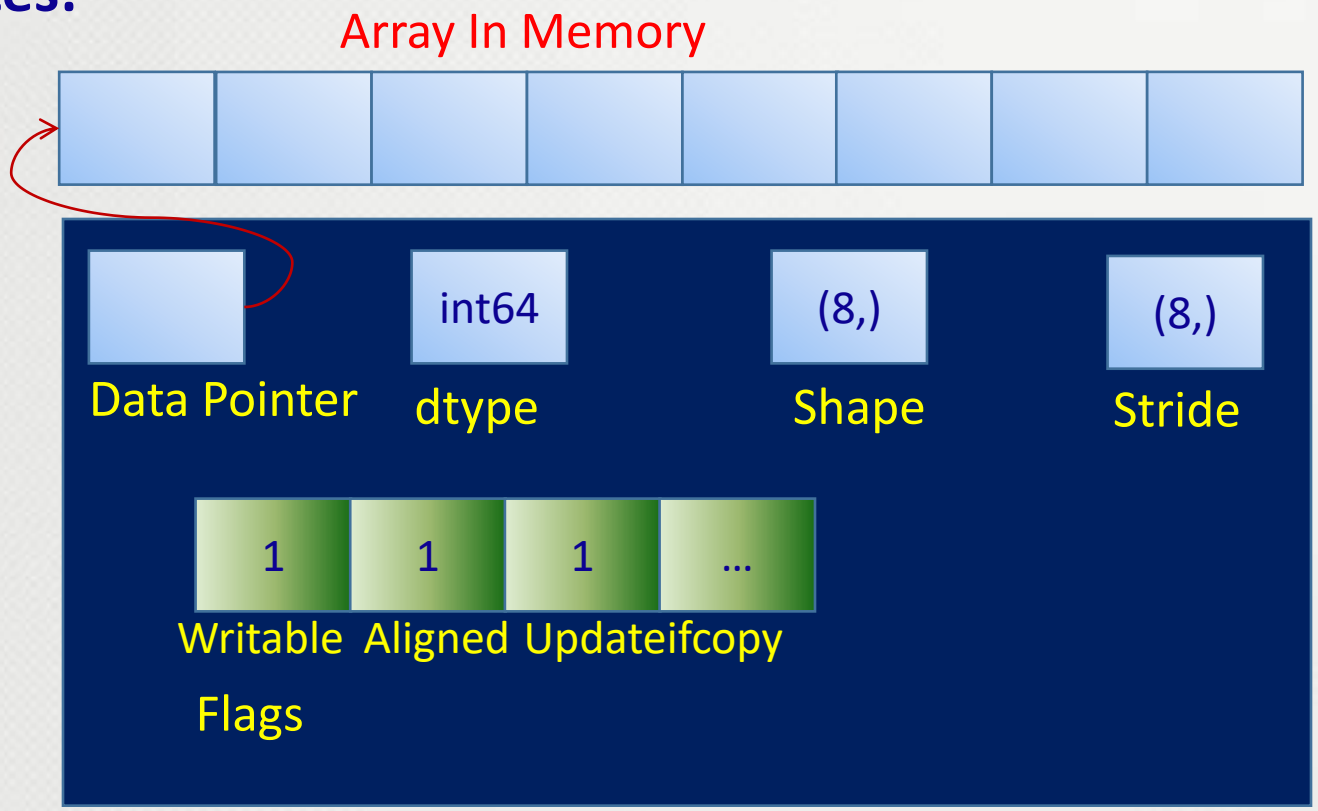
Data Pointer

Data type description

Shape

Strides

Flags



Provides various methods:

arange

reshape

sum

mean

min

max

var

Numpy Data Types

Type	Description
`bool`	Boolean (`True` or `False`) stored as a bit
`inti`	Platform integer (normally either `int32` or `int64`)
`int8`	Byte (-128 to 127)
`int16`	Integer (-32768 to 32767)
`int32`	Integer (-2 ** 31 to 2 ** 31 -1)
`int64`	Integer (-2 ** 63 to 2 ** 63 -1)
`uint8`	Unsigned integer (0 to 255)
`uint16`	Unsigned integer (0 to 65535)
`uint32`	Unsigned integer (0 to 2 ** 32 - 1)
`uint64`	Unsigned integer (0 to 2 ** 64 - 1)
`float16`	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
`float32`	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
`float64` or `float`	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa

teleuniv
innovative interactive immersive

Creating numpy Array

Machine Learning

Import the num py package

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

Creating numpy

There are various ways to create numpy array

General syntax: ***np.array(list of elements, dtype=datatype)***

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

```
In [10]: f=np.array([2,3,4,5],dtype=float)
         f
Out[10]: array([ 2.,  3.,  4.,  5.])
```

teleuniv
innovative interactive immersive

Creating numpy Array

Machine Learning

Using arange(): Generate a sequence of elements in given range and return ndarray.

arange([start,] stop[, step,], dtype=None)

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

        a=np.arange(10)
        a

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

```
In [9]: e=np.array([np.arange(4),np.arange(4)])
        e

Out[9]: array([[0, 1, 2, 3],
               [0, 1, 2, 3]])
```

Creating numpy Array

Using reshape() Gives a new shape to an array without changing its data.

```
In [4]: c=np.arange(9)
        d=c.reshape([3,3])
        d

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

Accessing Elements

Forward Index starts from 0

Backward index starts from -1

Slicing can be used

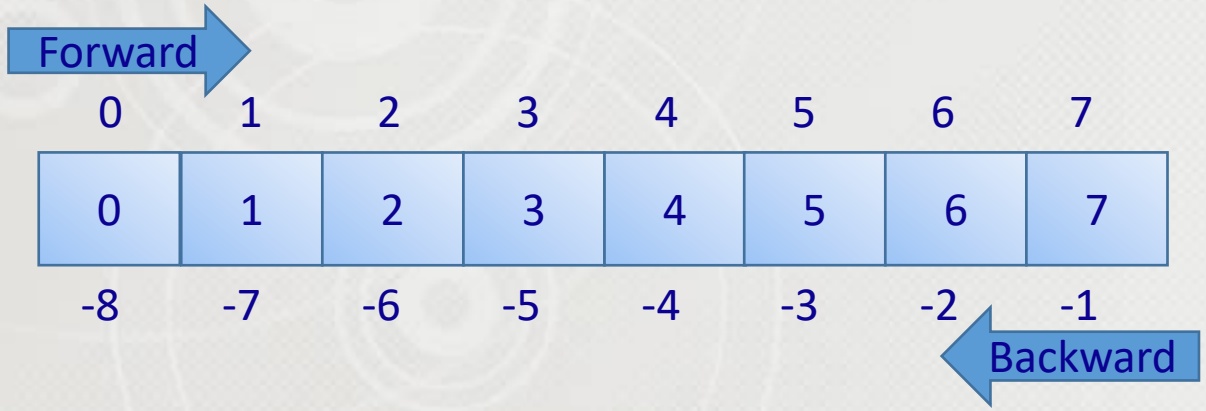
Slicing

a[1:4]

a[-7:-4]

a[-5:-8:-1]

Accessing Elements

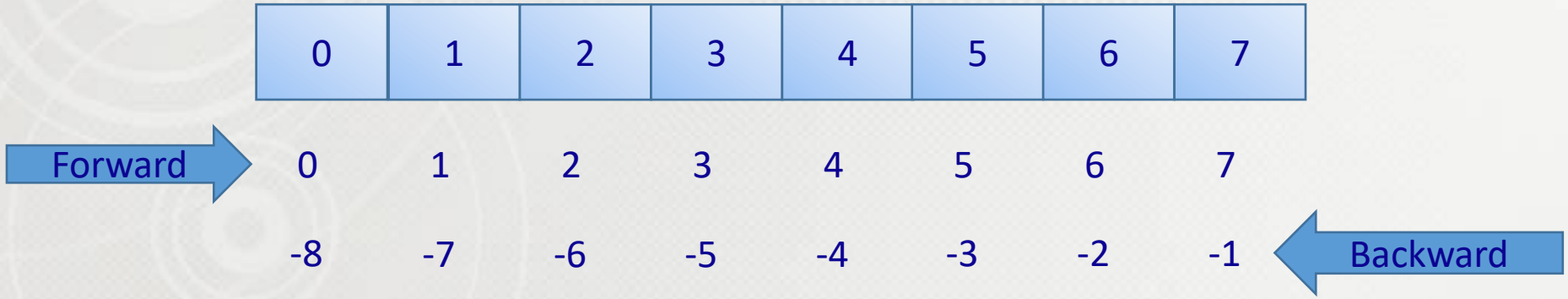



```
In [9]: from numpy import *
        a=arange(8)
        print(a)
        a[1:3]

        [0 1 2 3 4 5 6 7]

Out[9]: array([1, 2])
```

```
In [3]: a= arange(8)
        print(a)
        a[::-1]

        [0 1 2 3 4 5 6 7]

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

```
In [5]: a= arange(8)
        print(a)
        a[6:3:-1]

        [0 1 2 3 4 5 6 7]

Out[5]: array([6, 5, 4])
```

Accessing Elements




```
In [12]: a=arange(8)
         print(a)
         a[-7:-5]

         [0 1 2 3 4 5 6 7]

Out[12]: array([1, 2])
```

```
In [6]: a= arange(8)
        print(a)
        a[-3:-6:-1]

        [0 1 2 3 4 5 6 7]

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

Accessing Elements

Multi-Dimensional Arrays

```
In [16]: a=arange(16).reshape(4,4)
         print(a)
         a[1:3,1:3]
```

```
          0   1   2   3
     0 [[ 0   1   2   3] -4
     1  [ 4   5   6   7] -3
     2  [ 8   9  10  11] -2
     3  [12  13  14  15]]-1
         -4  -3  -2  -1
Out[16]: array([[ 5,  6],
                [ 9, 10]])
```

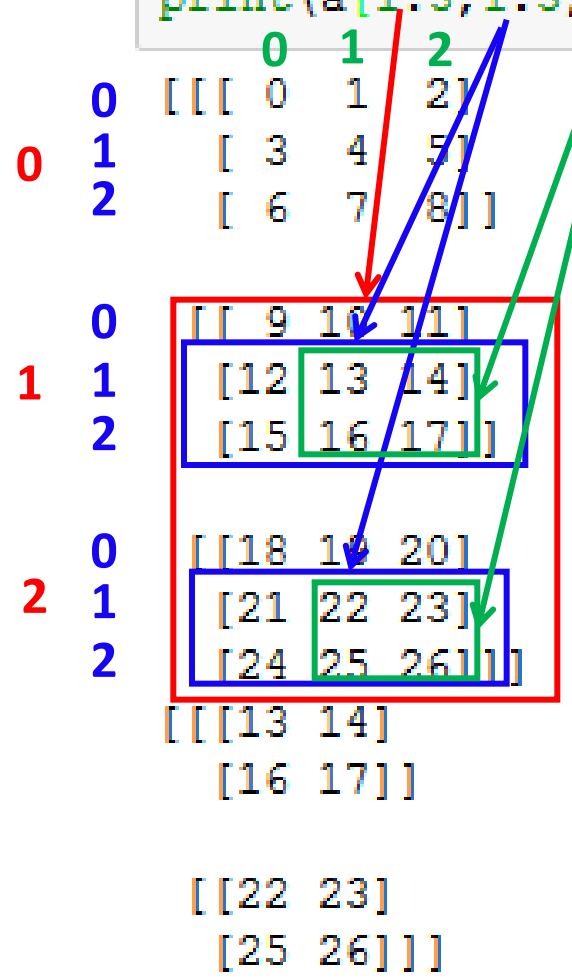
```
In [19]: a=arange(16).reshape(4,4)
         print(a)
         a[-3:-1,-3:-1]
```

```
          0   1   2   3
     0 [[ 0   1   2   3] -4
     1  [ 4   5   6   7] -3
     2  [ 8   9  10  11] -2
     3  [12  13  14  15]]-1
         -4  -3  -2  -1
Out[19]: array([[ 5,  6],
                [ 9, 10]])
```

Accessing Elements

Multi-Dimensional Arrays

```
In [38]: a=arange(27).reshape(3,3,3)
         print(a)
         print(a[1:3,1:3,1:3])
```



Attributes of numpy Array

shape: represents the shape of the array

Returns the shape as **tuple**

```
a=arange(12).reshape(4,3)
print(a)
print("\n\n")
print(a.shape)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]


(4, 3)
```

```
In [10]: a=arange(24).reshape(2,4,3)
         print(a)
         print("\n\n")
         print(a.shape)
```

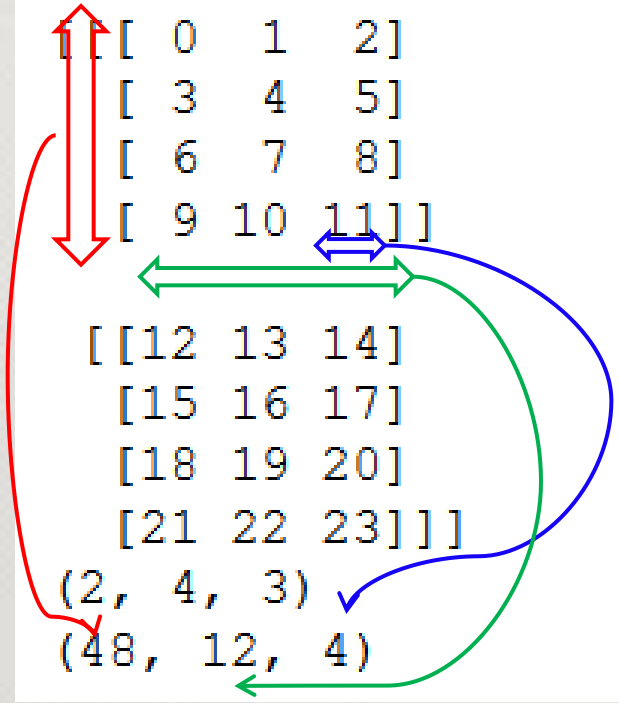
```
[[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]
  [ 9 10 11]]

 [[12 13 14]
  [15 16 17]
  [18 19 20]
  [21 22 23]]]


(2, 4, 3)
```

Attributes of numpy Array

stride: Tuple of bytes to step in each dimension when traversing an array.

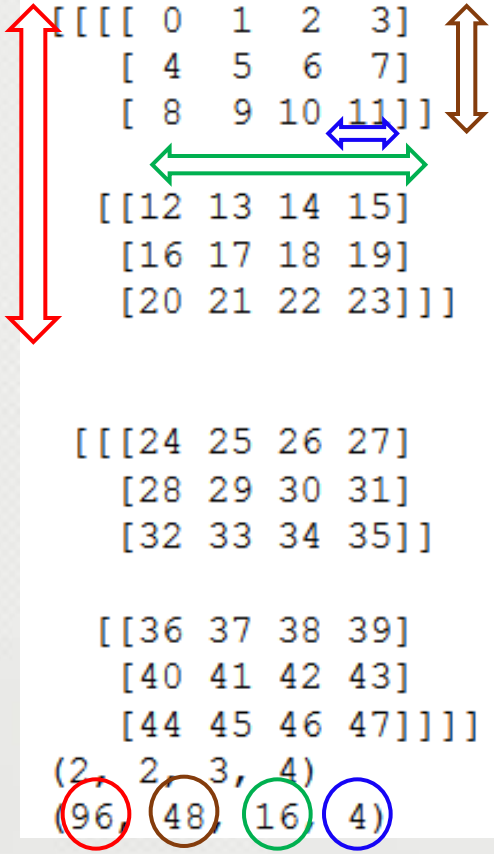
```
a=arange(24).reshape(2,4,3)
print(a)
print(a.shape)
print(a.strides)
```

```
[[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]
  [ 9 10 11]]

 [[12 13 14]
  [15 16 17]
  [18 19 20]
  [21 22 23]]]
(2, 4, 3)
(48, 12, 4)
```



```
a=arange(48).reshape(2,2,3,4)
print(a)
print(a.shape)
print(a.strides)
```

```
[[[[ 0  1  2  3]
   [ 4  5  6  7]
   [ 8  9 10 11]]

  [[12 13 14 15]
   [16 17 18 19]
   [20 21 22 23]]]


 [[[24 25 26 27]
   [28 29 30 31]
   [32 33 34 35]]

  [[36 37 38 39]
   [40 41 42 43]
   [44 45 46 47]]]]
(2, 2, 3, 4)
(96, 48, 16, 4)
```



teleuniv
innovative interactive immersive

Attributes of numpy Array

Machine Learning

ndim: number of dimensions of the array.

data: the memory address of the array

size: number of elements in the array

itemsize: size of each element in bytes

nbytes: size of the array in bytes.

T: transpose of the given array

dtype: Data type of the array

```
a=arange(16)
print("a.ndim -> ",a.ndim)
a=a.reshape(4,4)
print("a.ndim -> ",a.ndim)
print("a.data -> ",a.data)
print("a.size -> ",a.size)
print("a.itemsize -> ",a.itemsize)
print("a.nbytes -> ",a.nbytes)
print("a.T -> ",a.T)
print("a.dtype ->", a.dtype)
```

```
a.ndim ->  1
a.ndim ->  2
a.data ->  <memory at 0x000001D17B2148B8>
a.size ->  16
a.itemsize ->  4
a.nbytes ->  64
a.T ->  [[ 0  4  8 12]
 [ 1  5  9 13]
 [ 2  6 10 14]
 [ 3  7 11 15]]
a.dtype -> int32
```

Arithmetic Operations

Arithmetic operations will be performed on corresponding elements of arrays

```
a=array([[1,2],[4,5]])
b=array([[10,20],[40,50]])
print("......a.......")
print(a)
print("......b.......")
print(b)
print("......a+b.......")
c=a+b
print(c)
print("......a-b.......")
c=a-b
print(c)
print("......a*b.......")
c=a*b
print(c)
```

```
......a.......
[[1 2]
 [4 5]]
......b.......
[[10 20]
 [40 50]]
......a+b.......
[[11 22]
 [44 55]]
......a-b.......
[[ -9 -18]
 [-36 -45]]
......a*b.......
[[ 10  40]
 [160 250]]
```

Broadcasting Rules

Numpy verifies the dimensions are matched or not.
Smaller array will be repeated/broadcasted to match with large array.

```
a=array([[1,2],[4,5]])
b=array([[10,20]])
print(a)
print(b)
print(".....a+b.......")
c=a+b
print(c)
print(".....a-b.......")
c=a-b
print(c)
print(".....a*b.......")
c=a*b
print(c)
```

```
[[1 2]
 [4 5]]
[[10 20]]
.....a+b.......
[[11 22]
 [14 25]]
.....a-b.......
[[ -9 -18]
 [ -6 -15]]
.....a*b.......
[[ 10  40]
 [ 40 100]]
```

Methods on numpy Array

Machine Learning

Various methods defined on numpy.

```
a=array([[0,2,3],[4,5,6]])
print(a)
print("a.sum() -> ",a.sum())
print("a.sum(1) -> ",a.sum(1))
print("a.sum(0) -> ",a.sum(0))
print("a.max() ->",a.max())
print("a.max(1) ->",a.max(1))
print("a.min() ->", a.min())
print("a.mean() ->",a.mean())
print("a.var() ->", a.var())
print("a.all() ->", a.all())
print("a.any() ->", a.any())
```

```
[[0 2 3]
 [4 5 6]]
a.sum() ->  20
a.sum(1) ->  [ 5 15]
a.sum(0) ->  [4 7 9]
a.max() -> 6
a.max(1) -> [3 6]
a.min() -> 0
a.mean() -> 3.33333333333
a.var() -> 3.88888888889
a.all() -> False
a.any() -> True
```

Methods on numpy Array

Machine Learning

Various methods defined on numpy.

```
a=arange(24).reshape(2,4,3)
print(a)
print("a.sum()->",a.sum())
print("a.sum(0)->",a.sum(0))
print("a.sum(1)->",a.sum(1))
print("a.sum(2)->",a.sum(2))
```

```
[[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]
  [ 9 10 11]]

 [[12 13 14]
  [15 16 17]
  [18 19 20]
  [21 22 23]]]
```

a[:,:,:]

a[0,:,:] +
a[1,:,:]

a[:,0,:] +
a[:,1,:] +
a[:,2,:]+
a[:,3,:]

a[:,:,0] +
a[:,:,1] +
a[:,:,2]

```
a.sum()-> 276
a.sum(0)-> [[12 14 16]
 [18 20 22]
 [24 26 28]
 [30 32 34]]
a.sum(1)-> [[18 22 26]
 [66 70 74]]
a.sum(2)-> [[ 3 12 21 30]
 [39 48 57 66]]
```

teleuniv
innovative interactive immersive

Methods on numpy Array

Machine Learning

Various methods defined on numpy.

```
a=arange(9)
b=a.reshape(3,3)
print(b)
```

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

```
a=array([30,12,23,54,5,36])
print(a)
print(sort(a))
```

```
[30 12 23 54  5 36]
[ 5 12 23 30 36 54]
```

```
a=array([[1,2],[4,5]])
b=array([[1,0],[0,1]])
print(a)
print(b)
c=a.dot(b)
print(c)
```

```
[[1 2]
 [4 5]]
[[1 0]
 [0 1]]
[[1 2]
 [4 5]]
```

```
a=array([[0,2,3],[4,5,6]])
print(a)
print(a.tolist())
print(a.transpose())
```

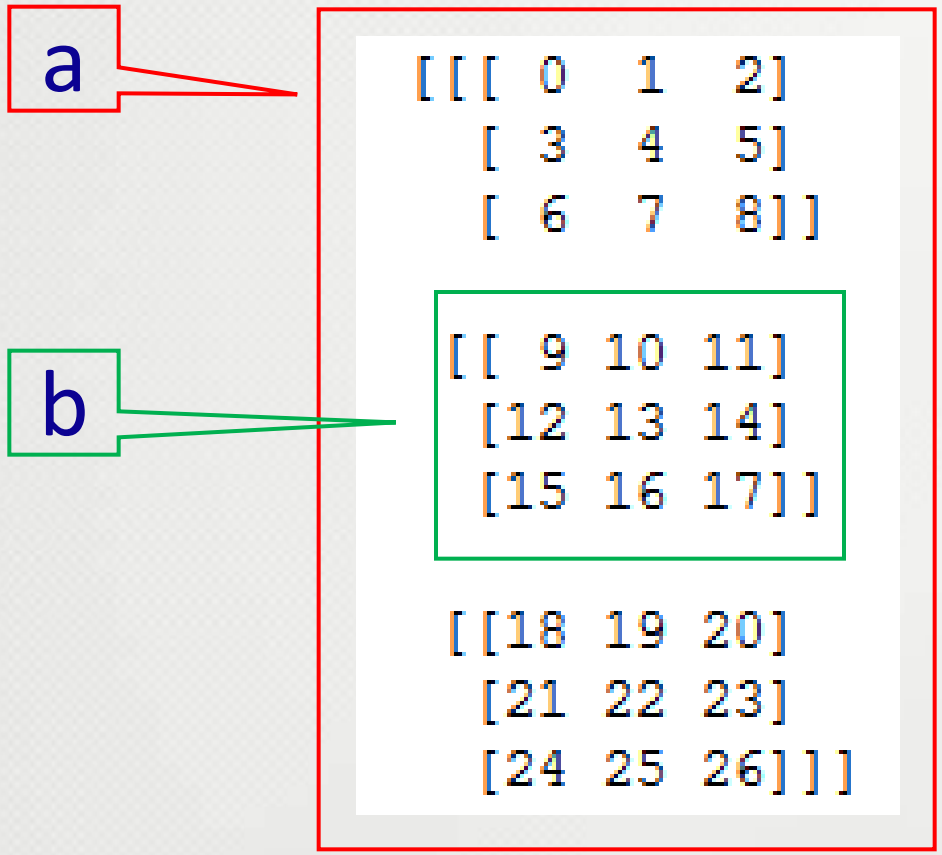
```
[[0 2 3]
 [4 5 6]]
[[0, 2, 3], [4, 5, 6]]
[[0 4]
 [2 5]
 [3 6]]
```

Creating views

We can generate view on same memory which was allocated for original array

```
In [42]: a=arange(27).reshape(3,3,3)
         print("Original Array")
         print(a)
         b=a[1,]
         print("View")
         print(b)
```

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

 [[ 9 10 11]
  [12 13 14]
  [15 16 17]]

 [[18 19 20]
  [21 22 23]
  [24 25 26]]]
View
[[ 9 10 11]
 [12 13 14]
 [15 16 17]]
```



Creating views

We can generate view on same memory which was allocated for original array

```
a=arange(27).reshape(3,3,3)
b=a[1,]
b[1,1]=0
print("View")
print(b)
print("Original Array")
print(a)
```

```
View
[[ 9 10 11]
 [12  0 14]
 [15 16 17]]
Original Array
[[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]]

 [[ 9 10 11]
  [12  0 14]
  [15 16 17]]

 [[18 19 20]
  [21 22 23]
  [24 25 26]]]
```

teleuniv
innovative interactive immersive

Creating views

In [45]:
```
a=arange(27).reshape(3,3,3)
print("Original Array")
print(a)
b=(a[1:3,1:3,1:3])
print("......View.....")
print(b)
```

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

 [[ 9 10 11]
  [12 13 14]
  [15 16 17]]

 [[18 19 20]
  [21 22 23]
  [24 25 26]]]
......View.....
[[[13 14]
  [16 17]]

 [[22 23]
  [25 26]]]
```

In [46]:
```
a=arange(27).reshape(3,3,3)
print("Original Array")
print(a)
b=(a[::2,::2,::2])
print("......View.....")
print(b)
```

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

 [[ 9 10 11]
  [12 13 14]
  [15 16 17]]

 [[18 19 20]
  [21 22 23]
  [24 25 26]]]
......View.....
[[[ 0  2]
  [ 6  8]]

 [[18 20]
  [24 26]]]
```

teleuniv
innovative interactive immersive

User Defined Data Types

Machine Learning

Allows to create user defined data type

dtype([(item_name1, data_type1),(item_name2, data_type2)])

```
import numpy as np
mytype=np.dtype([('rno',np.int32),('name',np.str_,30),('m1',np.int32),('m2',np.int32),('m3',np.int32)])
print(mytype)
student=[(1,'rama',45,54,45),(2,'krishna',54,43,43)]
std=np.array(student,dtype=mytype)
std
```

```
[('rno', '<i4'), ('name', '<U30'), ('m1', '<i4'), ('m2', '<i4'), ('m3', '<i4')]

array([(1, 'rama', 45, 54, 45), (2, 'krishna', 54, 43, 43)],
      dtype=[('rno', '<i4'), ('name', '<U30'), ('m1', '<i4'), ('m2', '<i4'), ('m3', '<i4')])
```

Conclusion

Discussed about ...

- Numpy – creation – operations – methods - broadcasting

Next Session .....

Image and Audio Handling


THANK
YOU