25-11-2020

LECTURE-14

KANAV BANSAL

NUMPY:

NumPy stands for Numerical Python.

↳ It is a python library used for working with arrays.

↳ It also has functions for working in domain of linear algebra, fourier transform, and matrices.

↳ It was created in 2005 by Travis Oliphant.

↳ It is an open source project and can be used freely.

↳ Most powerful numerical processing library in Python. Array Oriented computing.

↳ Provides extensions package to python for multi dimensional array.

↳ Very Efficient.

↳ Scientific computation.

Creating a simple array in numpy:

I/p - import numpy as np

```
arr = np.array([1, 2, 3, 4])
Print(arr)
Print(type(arr))
```

O/p - [1 2 3 4]

<class 'numpy.ndarray'

NOTE:

the array object in Numpy is called as ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

I/p -

```
l = [1, 2, 3, 4, 5, 6, 7]
Print(type(l))
arr = np.array([l])
Print(type(arr))
Print("List : ", l)
Print("NP Array : ", arr)
```

O/P - <class 'list'>

<class 'numpy.ndarray'>

List : [1, 2, 3, 4, 5, 6, 7]

NP Array : [[1 2 3 4 5 6 7]]

I/P - arr = np.arange(10)

Print(arr)

Print(type(arr))

O/P - [0 1 2 3 4 5 6 7 8 9]

<class 'numpy.ndarray'>

I/P - %%time

lst = list(range(1000000));

for i in range(1000000):

lst[i] *= lst[i]

O/P - Wall time : 448 ms

I/P - %%time

arr = np.arange(1000000)

arr = arr * arr

O/p - Wall time : 6.95 ms.

I/p - arr = np.array([1, 2, 3, 4])

```
Print(arr.ndim) # print number of
                  dimensions

# Print shape

Print(arr.shape)

# Print length

Print(len(arr))

# Print Data type

Print(arr.dtype)

# Print item size in byte of each element

Print(arr.itemsize)
```

O/p - 1

```
(4, )

4

int32

4
```

I/p -
```
arr = np.array([[1,2,3],[4,5,6]])
Print(arr)
Print(arr.ndim)
Print(arr.shape)
Print(len(arr))
Print(arr.dtype)
Print(arr.itemsize)
# Print diagonal elements
Print(np.diag(arr))
```

O/p -
```
[[1 2 3]
 [4 5 6]]
2
(2, 3)
2
int32
4
[1 5]
```

Functions for creating arrays:


```
I/p - arr = np.ones((3,3))
      Print(arr)

O/p - [[1. 1. 1.]
       [1. 1. 1.]
       [1. 1. 1.]]
```

1. → its a floating type

```
I/p - arr = np.zeros((3,3))
      print(arr)

O/p - [[0. 0. 0.]
       [0. 0. 0.]
       [0. 0. 0.]]
```

```
I/p - arr = np.eye(3)
      print(arr)

O/p - [[1. 0. 0.]
       [0. 1. 0.]
       [0. 0. 1.]]
```

eye = Identity matrix

```
I/p - arr = np.eye(3,2)
      print(arr)

O/p - [[1. 0.]
       [0. 1.]
       [0. 0.]]
```

I/P - arr = np.arange(10)

Print(arr)

O/P - [0 1 2 3 4 5 6 7 8 9]

I/P - arr = np.arange(1,10)

Print(arr)

O/P - [1, 2 3 4 5 6 7 8 9]

I/P - arr = np.arange(1, 10, 2)

Print(arr)

O/P - [1 3 5 7 9]

I/P - arr = np.zeros((3,3))

Print(arr)

Print(arr.dtype)

Print(arr.itemsize)

O/P - [[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]]

float64

8