Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

O-LEVEL (M3-R5.1)
Python Programming By

Mrs. Geeta
Python Instructor
Zed-King Group of Institutions
Kaithal

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

O-LEVEL (M3-R5.1)

Python Programming

Video Tutorial

O-LEVEL (M3-R5.1) Full Video Playlist Available on YouTube Channel **Zed-king Computer Institute Kaithal**

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Free Online Computer Classes On YouTube Channel

https://www.youtube.com/@zedkinginstitute

For Free PDF Notes

Our Telegram Channel

t.me/boost/Zedking_Institute

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

O-LEVEL (M3-R5.1)
CHAPTER-9

Numpy Basics

NumPy:-

Numpy is a library of python through which operations can be performed multidimensional arrays in python.

The full name of NumPy is Numerical Python.

Check PIP (preferred installer package (Program))package:-

1. Go to cmd
2. Type pip list command
3. Press enter
4. If screen show
5. It means pip is installed already
6. If pip is not installed
7. Then screen show
8. Pip is not recognized as an internal or external command

Download & install numPy:-

1. Go to cmd
2. Type pip install numpy
3. Press enter

```
C:\Users\Akhilesh>pip install numpy
Collecting numpy
  Obtaining dependency information for numpy from https://files.pythonhosted.org/packages/28/75/3b679b4171
87be72c971c9e718b1c17b8f8749240ddca8/numpy-1.26.2-cp312-cp312-win_amd64.whl.metadata
  Downloading numpy-1.26.2-cp312-cp312-win_amd64.whl.metadata (61 kB)
     ---------------------------------------- 61.2/61.2 kB 155.4 kB/s eta 0:00:00
Downloading numpy-1.26.2-cp312-cp312-win_amd64.whl (15.5 MB)
   ---------------------------------------- 15.5/15.5 MB 2.0 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.26.2

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

About Lesson

Check numPy Installation:-

1. Go to cmd
2. Type pip list command
3. Press enter

```
C:\Users\Akhilesh>pip list
Package  Version
-------- -------
asgiref  3.7.2
Django   4.2.5
numpy    1.26.2
pip      23.2.1
sqlparse 0.4.4
tzdata   2023.3

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

About Lesson

Creation Numpy Array (One Dimensional)

Creation Numpy Array (One Dimensional)

```
import numpy as np
a=[1,2,3,4,5]
myarr=np.array(a)
print(array)
```

Or

```
import numpy as np
myarr=np.array([1,2,3,4,5])
print(myarr)
```

Through User Input:

```
import numpy as np
a=[]
n=int(input("Enter size of the array"))
for i in range(n):
    val=int(input("Enter Number"))
    a.append(val)
myarr=np.array(a)
print(myarr)
```

About Lesson

Creation Numpy Array (Two Dimensional)

```
import numpy as np

a=[[1,2,3],[4,5,6],[7,8,9]]

myarr=np.array(a)

print(myarr)
```

```
import numpy as np

myarr=np.array([[1,2,3],[4,5,6],[7,8,9]])

print(myarr)
```

Array Attribute:-

Numpy array is a set of attributes that you can access like array size, shape, number of dimensions etc.

About Lesson

ndim:- it display the dimensions of array.

```
import numpy as np

arr=np.array([[1,2,3,4],[4,5,6,7],[8,9,10,11]])

print("Total no of dimension= ",arr.ndim)
```

Shape:- it return a tuple of integers indicating the size of the array.

```
import numpy as np

arr=np.array([[1,2,3,4],[4,5,6,7],[8,9,10,11]])

print("Shape of Array is= ",arr.shape)

output

Shape of Array is=  (3, 4)
```

Size:- it return the total number of elements in the NumPy array.

```
import numpy as np

arr=np.array([[1,2,3,4],[4,5,6,7],[8,9,10,11]])

print("size of Array is= ",arr.size)
```

output

```
size of Array is=  12
```

Dtype:- it return the type of elements in the array. i.e. int 64.

```
import numpy as np

arr=np.array([[1.5,2.6,3.4]])

print("data type is= ",arr.dtype)

output

data type is=  float64
```

itemsize:- it return the length of each element of array in bytes.

```
import numpy as np

arr=np.array([[1,2,3],[5,6,7]])

print("data type is= ",arr.itemsize, "bytes")

output

each item contain= 4 bytes
```

Reshape:- the reshape() function associated with the ndarray object is used to reshape the array. It accept the two parameters indicating the row and columns of the new shape of the array.

```
import numpy as np

a=np.array([[1,2,3],[5,6,7],[8,9,6],[4,6,8]])

print("Original Array is ")

print(a)

a=a.reshape(3,4)

print("reshaped array is ")

print(a)
```

About Lesson

Numpy functions

Zeros:- this function creates an array (either one dimensional or multi dimensional) and fill all the values with Zero (0).

```
import numpy as np
arr=np.zeros(5)
print(arr)
```

```
import numpy as np
arr=np.zeros((3,4))
print(arr)
```

Ones:- this function creates an array (either one dimensional or multi dimensional) and fill all the values with one (1).

```
import numpy as np
arr=np.ones((3,4))
print(arr)
```

Eye:- this function creates an array with all the diagonal elements as 1 and rest as 0.

```
import numpy as np
arr=np.eye(3,4)
print(arr)
```

```
import numpy as np
arr=np.eye(3)
print(arr)
```

Diag:- this function creates a two dimensional array with all the diagonal elements as the given value and rest as 0.

```
import numpy as np
arr=np.diag([1,2,3,4])
print(arr)
```

About Lesson

Randint:- This function is used to generate a random number between a given range.

Syntax:- **randint(min,max,total_values)**

```
import numpy as np
arr=np.random.randint(1,10,3)
print(arr)
```

rand:- This function is used to generate a random values between 0 to 1.

Syntax:- **rand(number of value)**

```
import numpy as np
arr=np.random.rand(2,3)
print(arr)
```

```
import numpy as np
arr=np.random.rand(5)
print(arr)
```

randn:- This function is used generate a random values close to 0 (zero). This may return positive or negative numbers as well.

Syntax:- **randn(number of value)**

```
import numpy as np

arr=np.random.randn(5)

print(arr)
```

seed:- we know that randint() function generates random numbers. Every time we run the program, new set of random number is generated. But what if we want to fix the generation of this random number. In this case we use seed, this generates fix set of random numbers.

```
import numpy as np

np.random.seed(10)

arr=np.random.randint(1,50,10)

print(arr)
```

Array From Numerical Ranges:-

Np.arange:- arrange(0 returns an ndarray object containing evenly spaced values within a given range.

Syntax:- numpy.arange(start,stop,step,dtype)

```
import numpy as np
arr=np.arange(1,10,2)
print(arr)
```

Output

```
[1 3 5 7 9]
```

Conditional Selection

```
import numpy as np
arr=np.arange(1,20,1)
b=arr>10
print(arr[b])
```

Output

```
[11 12 13 14 15 16 17 18 19]
```

Multiply Number Printing

```
import numpy as np
arr=np.arange(1,5)
print(arr*2)
```

Output

```
[2 4 6 8]
```

About Lesson

Even Number Printing

```
import numpy as np
arr=np.arange(1,20,1)
print(arr[arr%2==0])
```

Output

```
[ 2  4  6  8 10 12 14 16 18]
```

Exponent Number Printing

```
import numpy as np
arr=np.arange(1,5)
print(arr/0)
```

Output

```
[inf inf inf inf]
```

Note:- it numpy array divide by zero gives infinity and not any error

Exponent Number Printing

```
import numpy as np
arr=np.arange(1,5)
print(arr**2)
```

Output

```
[ 1  4  9 16]
```

About Lesson

Np.linspace:- linspace() function returns values between a given range and with a same gap between consecutive elements.

Syntax:– numpy.linspace(start,stop,num,endpoint,retstep,dtype)

```
import numpy as np

a=np.linspace(5,15,5)

print(a)
```

Output

```
[ 5.   7.5 10.  12.5 15. ]
```

```
import numpy as np

a=np.linspace(5,15,5,endpoint=False)

print(a)
```

Output

```
[ 5.  7.  9. 11. 13.]
```

Numpy.logspace:- logspace() function creates an array by using the numbers that are evenly separated on a log scale. Start and stop end points of the scale are indices of the base, usually 10.

Syntax:- numpy.logspace(start,stop,num,endpoint,base,dtype)

```
import numpy as np

a=np.logspace(5,15,3)

print(a)
```

Output

```
[1.e+05 1.e+10 1.e+15]
```

About Lesson

Arithmetic Operation on Array:-

Adding Two Array

```
import numpy as np

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

b=np.array([5,6,7,8])

c=a+b

print(c)
```

Output

```
[ 6  8 10 12]
```

Operation on Two Dimensional Array

```
import numpy as np

a=np.array([1,2,3,4]).reshape(2,2)

b=np.array([5,6,7,8]).reshape(2,2)

print("Reshape array is ")

print(a)

print(b)

print(a+b)

print(b-a)

print(a*b)

print(b/a)
```

Output

Multiply Two Matrix

```
import numpy as np

a=np.array([1,2,3,4]).reshape(2,2)

b=np.array([5,6,7,8]).reshape(2,2)

print(a.dot(b))
```

Output

```
[[19 22]

 [43 50]]
```

About Lesson

Numpy Functions:-

Min,max,argmin,argmax,sqrt,sin,cos

```
import numpy as np

a=np.array([10,20,30,40])

print(np.min(a))

print(np.max(a))

print(np.argmin(a))

print(np.argmax(a))

print(np.sqrt(a))

print(np.sin(a))

print(np.cos(a))
```

Shuffle Function:-

```
import numpy as np
a=np.array([10,20,30,40,20,50,30,40])
print(a)
np.random.shuffle(a)
print(a)
```

Unique Function:-

```
import numpy as np
a=np.array([10,20,30,40,20,50,30,40])
print(a)
print(np.unique(a))
```

Unique size function:-

```
import numpy as np
a=np.array([10,20,30,40,20,50,30,40])
print(a)
print(np.unique(a))
print(np.unique(a).size)
```

About Lesson

Joining & Spliting:-

Joining (concatenation) of arrays means combining multiple arrays into one and splitting of arrays means splitting one array into many.

#Concatenation Example

```
import numpy as np

a=np.array([10,20,30])

b=np.array([40,50,60])

c=np.concatenate([a,b])

print(a)

print(b)

print(c)
```

#Concatenation Example Using Stack

```
import numpy as np
a=np.array([[10,20,30],[15,25,35]])
b=np.array([[40,50,60],[45,55,65]])
print("Original Array is ")
print(a)
print(b)
print("Arrays Vertically Concatenationn",np.vstack((a,b)))
print("Arrays Horizontally concatenationn",np.hstack((a,b)))
```

Output:-

```
Original Array is
[[10 20 30]
 [15 25 35]]
[[40 50 60]
 [45 55 65]]
Arrays Vertically Concatenation
 [[10 20 30]
 [15 25 35]
 [40 50 60]
 [45 55 65]]
Arrays Horizontally concatenation
 [[10 20 30 40 50 60]
 [15 25 35 45 55 65]]
```

About Lesson

#spliting example

```
import numpy as np
a=[1,2,3,4,5,6]
print(a)
a1,a2,a3=np.split(a,[3,4])
print(a1,a2,a3)
```

Output

```
[1, 2, 3, 4, 5, 6]
[1 2 3] [4] [5 6]
```

#creating 1D array

```
import numpy as np
a=[ ]
size=int(input("Enter how many number "))
for i in range(size):
    val=int(input("enter value "))
    a.append(val)
arr=np.array(a)
for i in range(arr.size):
    print(arr[i])
print(type(arr))
```

#indexing 1D Array

```
import numpy as np

a=np.array([10,20,30,40,50])

print(a[0])

print(a[1])

print(a[-1])

print(a[-3])
```

Output:

```
10

20

50

30
```

Indexing of 2 dimensional array:-

- indexing/slicing using double bracket notation

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]
[2][0]	[2][1]	[2][2]

- indexing/slicing using single bracket notation

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]

#indexing 2D Array

```
import numpy as np

a=np.array([[10,20,30],[40,50,60],[70,80,90]])

print(a)

print(a[0,0])

print(a[1,0])

print(a[2,2])
```

Create 2 Dimensional Array Using User Input:-

```
#creating 2D array
import numpy as np
matrix=[]
row=int(input("Enter number of row"))
col=int(input("Enter number of column "))
for i in range(row):
    a=[]
    for j in range(col):
        val=int(input("enter value "))
        a.append(val)
    matrix.append(a)
arr=np.array(matrix)
for i in range(row):
    for j in range(col):
        print(arr[i][j],end=" ")
    print()
print(type(arr))
```

```
#creating 2D array using user input
import numpy as np
matrix=[]
row=int(input("Enter number of row"))
col=int(input("Enter number of column "))
for i in range(row):
    a=[]
    for j in range(col):
        val=int(input("enter value "))
        a.append(val)
    matrix.append(a)
for i in range(row):
    for j in range(col):
        print(matrix[i][j],end=" ")
    print()
```

Slicing in 1 Dimensional Array:-

Syntax:- arr_name[start:end:step]

arr_name[star_index:end_index+1]

arr_name[start_index:]

arr_name[:end_index+1]

arr_name[:]

Note:-Indexing can be positive as well Negative.

```
import numpy as np
a=[]
size=int(input("Enter size"))
for i in range(size):
    val=int(input("Enter value "))
    a.append(val)
arr=np.array(a)
for i in range(arr.size):
    print(arr[i])
arr1=arr[2:4]
print(arr1)
```

#Slicing in 1D array

```
import numpy as np

a=np.array([10,20,30,40,50,60,70,80,90])

print(a)

print(a[4:])

print(a[:4])

print(a[4:7])

print(a[:-1])

print(a[::-1])

print(a[-4::])

print(a[-2::-2])

print(a[::])
```

About Lesson

Slicing of 2 Dimensional Array:-

```
#Slicing in 2D array

import numpy as np

a=np.array([[10,20,30],[40,50,60],[70,80,90]])

print(a)

print(a[:2,::2])
```

Output:-

```
[[10 20 30]
 [40 50 60]
 [70 80 90]]
[[10 30]
 [40 60]]
```

3-D arrays

- An array that has 2-D arrays (matrices) as its elements is called 3-D array.
- These are often used to represent a 3rd order tensor.
- Example
- Create a 3-D array with two 2-D arrays, both containing two arrays with the values 1,2,3 and 4,5,6:
- Program:
- import numpy as np

  arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

  print(arr)

Access 3-D Arrays

- To access elements from 3-D arrays we can use comma separated integers representing the dimensions and the index of the element.
- Program: Access the third element of the second array of the first array:

```
import numpy as np
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
print(arr[0, 1, 2])
```

Output: arr[0, 1, 2] prints the value 6

Data Types in NumPy

- NumPy has some extra data types, and refer to data types with one character, like i for integers, u for unsigned integers etc.
- Below is a list of all data types in NumPy and the characters used to represent them
- i – integer
- b – boolean
- u – unsigned integer
- f – float
- c – complex float
- m – timedelta
- M – datetime
- O – object
- S – string
- U – unicode string
- V – fixed chunk of memory for other type (void)

Zed-King
GROUP OF INSTITUTIONS
ISO 9001 : 2015 Certified

Checking the Data Type of an Array

- The NumPy array object has a property called dtype that returns the data type of the array:

- **Example**

- Get the data type of an array object:

```
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr.dtype)
```

- Get the data type of an array containing strings:

```
import numpy as np
arr = np.array(['apple', 'banana', 'cherry'])
print(arr.dtype)
```

Activate

Check Number of Dimensions?

- NumPy Arrays provides the ndim attribute that returns an integer that tells us how many dimensions the array have.
- **Example**
- Check how many dimensions the arrays have:
- Program:

```
import numpy as np
a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

Difference Between List and Numpy

A common beginner question is what is the real difference here. The answer is performance. Numpy data structures perform better in:

Size:- Numpy data structures take up less space

Performance:- they have a need for speed and are faster than lists

Functionality:- SciPy and Numpy have optimized functions such as linear algebra operations built in.