27) Numpy

- Numpy is a library used for working with Array.
- Numpy stands for Numerical Python.
- Numpy library consisting of multidimensional Array object and collection of routines.
- Using Numpy mathematical and logical operations on array can be done.
- Numpy also has a functions working in domain of linear algebra, fourier Transform, and matrices.
- Numpy is an open source core Python library for scientific computation.
- Python is slower as compared to fortran and other languages to perform looping. To overcome this we use Numpy that converts homogeneous code into the compiled form.

(i) Features of Numpy:

(i) High-Performance.
(ii) Integrating code from C/C++.
(iii) Multidimensional containers
(iv) Broadcasting functions.
(v) Work with varied databases.
(vi) Additional Linear Algebra.

(1) High Performance N-dimensional Array object.

- It is homogeneous Array object.
- perform all operations on array elements.
- Array in Numpy can be onedimensional or Multidimensional.

(i) one dimensional Array:

- one dimensional Array can consisting of single row or column
- element of Array are the homogeneous nature.

(ii) Multidimensional Array:

- We have various rows and columns.
- each column as dimension, elements are ~~heterogeneous~~ homogeneous.

(ii) Integrating codes from c/c++ and fortran.
- integrates the functionalities in various programming languages. this helps to implement inter-platform functions.

(iii) Multidimensional containers.
- generic data types to the parameterized data types of Array.
- It can perform functions on the generic data types.
- parameter helps increase the diversity of Array.

(iv) Additional linear Algebra, fourier transform, random Number capabilities.
- capability to perform complex operations of element like linear algebra, fourier transform etc.
- We have seperate modules for each of complex functions.
  linalg → linear Algebra.
  fft → fourier Transform.
  matrix → matrices.
  matplotlib → plotting graphs.

(v) It consist of Broadcasting functions.
- is very useful work with Array of unexcen shape.
- some rules has for limitations and its implementation.
- for broadcasting one array need to be one dimensional.

(vi). Work with varied Databases.
- work with different data types.
- dtype function can be used.

- Numpy was created in 2005 by Travis Olphant.
- Numpy aims to provide an array of object up to 50x.
- The array object in Numpy is called ndarray.

(ii) install Numpy:

- pip install Numpy.
- import in application using import keywords.
- import numpy. (use alias np).

Example

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

Version

- version string stored under __version__ attribute.

①

(iii) Create a Numpy ndarray Object:

- Numpy used to work with arrays.
- Array object in Numpy is called as ndarray.
- can create a Numpy ndarray object using array() function.

```
arr = np.array([1,2,3,4,5])
print(arr).
print(type(arr)) ← to check type of array.
```

type()

- Built in Python function. (type of object).

-

- to creation ndarray, we can pass list, tuple, or any array like obj. into array() method, and it will be converted into an ndarray.

```
arr = np.array((1,2,3,4,5)).
```

- There are different types of array (multidimensional or single dimensional) are available in Python Numpy.

(i) 0D Array.
(ii) 1D Array.
(iii) 2D Array.
(iv) 3D Array.

(i) 0D Array:
- 0D Array, or scalars are the element in the array
- Each value of in array is 0D Array.

arr = np.array(42).

(ii) 1D Array:
- An array that has 0-D arrays as its elements is called uni-dimensional or 1-D array.
- one dimensional array is a type of linear array. (or single dimensional)
- Accessing elements involves a single subscript (Row or column)

Syntax: np.array([ ]).
Example: np.array([1,2,3,4,5]).

(iii) 2D-Array
- An array that has 1-D arrays as its elements is called a 2-D array.
- these are orders tensors or matrix
- Two dimensional array is an array within array
- It is an arrays of array.
- syntax np.array([[ ], [ ]])
Example np.array([[1,2,3], [4,5,6]]).

(iv) 3D-Array:
- An array that has 2-D arrays (matrices) as its elements is called 3D.
- used to represent 3rd order tensor.

Syntax: np.array([[[ ][ ]][[ ], [ ]]]).
Example: np.array([[[1,2,3], [4,5,6]], [[1,2,3], [4,5,6]]]).

Check Number of Dimensions:

- Numpy arrays provides the ndim attributes that returns integer.

```
a = np.array([1,2,3,4,5,6])
print(a.ndim).
```

Higher Dimensional Arrays

- An array can have any number of dimensions.
- Create any number of dimension using ndim arguments.

```
arr = np.array([1,2,3,4,5,6], ndim=5)
print(arr).
print("Number of Dimension", arr.ndim)
```

(2) Array Indexing:

- Array indexing is same as accessing an array element.
- You can access an array element to its index number.
- Indexes in Numpy array start with 0,

```
arr = np.array([1,2,3,4,5,6])
print(arr[0]).
```

(i) positive Indexing.
(ii) Negative Indexing.

1D-Array.
2D-Array.
3D-Array.

(i) 1D Array Indexing.

- Start with [0] indexing that means second element has index [1] et

```
arr = np.array[[1,2,3,4,5]])
print([arr[0])
```

(ii) 2D-Array Indexing

- To access 2D array we can use comma separated integers. represent a dimensions and index of elements.
- Access 2nd element of 1st dimensions.

```
arr = np.array([[1,2,3,4,5],[1,2,3,4,5]])
print(arr[0,1])
```

(iii) 3D-Array Indexing:

- To access elements from 3D-elements we can use comma separated, representing dimensions.

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

① Negative Indexing:

- Use Negative Indexing to access array from the end.

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

Dimension → 1, -1 ← element (last element).

③ Numpy Array slicing:

- slicing in python means taking elements from one given index to another given index.
- we pass index → [start : End]
- We can also pass step → [start : step : End].
- If we don't pass start considered as 0,
- If we don't pass End its considered length of array in that dimensions.
- If we don't pass step its considered 1.

```
arr = np.array([1,2,3,4,5,6,7,8,9])
print(arr[1:5])
```

(i) positive slicing.
(ii) Negative slicing.
(iii) step.

(i) positive slicing:

```
arr = np.array([1,2,3,4,5,6,7,8])
print(arr[1:5])

print(arr[4:])

print(arr[:4])
```

(ii) Negative Indexing:

- use minus operator to refer an index from the end.

```
arr = np.arr([1,2,3,4,5,6,7,8])
print(arr[-3:-1])
```

(iii) step:

- use step value to determine the step of slicing.

```
arr = np.array([1,2,3,4,5,6,7,8,9])
print(arr[1:5:2])
```

start step. End

```
print(arr[::4])
```

considered as start index[0] considered as step 1

(ii) 2D-Array slicing.

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

(3) Data Types in NumPy.

- By default following datatypes in NumPy.

(i) String - used to represent a text data, Ex. "Python".
(ii) Integer. - used to represent integers numbers Ex. -1, -2, 3
(iii) float - used to represent real Numbers Ex. 1.2, 42.42.
(iv) boolean. - used to represent True or False.
(v) complex. - used to represent complex numbers Ex. 1.0+2.0j.

- NumPy has some extra data types. with one character, like i for integers, u for unsigned integers. etc.

(i) i - integers.
(ii) b - boolean.
(iii) u - unsigned integers.
(iv) f - float
(v) c - complex float.
(vi) m - timedelta.
(vii) M - datetime.
(viii) O - object
(ix) S - string
(x) U - unicode string.
(xi) fixed chunk of memory for other type.

- checking data type of an Array.
- NumPy array object has a property called dtype.

```
print(arr.dtype).
```

- creating Arrays with defined data type
- we use array() function to create arrays().

```
arr = np.array([1, 2, 3, 4], dtype='S')
print(arr).
```

- If value Not be converted.

If type is given in which elements can't casted then NumPy will raised ValueError.

- converting Data Type on Arrays (Existing).
- to make copy using astype() method, astype() method create copy of array and allows you to specify data type as parameter.

```
arr = np.array([1.1, 2.1, 3.1])
```

④ Numpy Array copy vs View

- The main difference between copy and view of an array is that copy is a new Array, and view is just view of original Array.
- copy also known as deep copy.
- copy completly new Array and copy owns data.
- If any changed made in copy it does not affect original array.
- and when we changes on copy it does not affect copy data.

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

(ii) View

- View also known as shallow copy.
- Just view of original array and view does not own the data.
- When we make changes to the view it affects the original array.
- and changes made to original array that affects to view array.

```
arr = np.array([1,2,3,4,5])
x = arr.view()
arr[0] = 42
print(arr)
print(x)
```

- Check arrays owns its data.

```
arr = np.array([1,2,3,4,5])
x = arr.copy()
x = arr.view()

print(x.base)
print(y.base)
```

⑤ NumPy Array shape.

- shape of an array is the number of elements in each dimensions.
- NumPy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements.

```
arr = np.array([[1,2,3,4],[5,6,7,8]])
print(arr.shape).
```

⑥ NumPy Array Reshaping :

- Reshaping means changing shape of an array.
- shape of an array is the number of elements in each dimension.
- we can add or remove dimensions or change number of elements in each dimension.

(i) Reshape 1D to 2D Array.

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

(ii) Reshape 1D to 3D. Array.

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

- Returns COPY or View

```
arr = np.array([1,2,3,4,5,6,7,8])
print(arr.reshape(2,4).base).
```

⑦ Numpy Iterating Arrays.

- Iterating means going through elements one by one.
- If we iterate on a 1-D array it will go through each element one by one.

```
arr = np.array([1,2,3])
for x in arr:
    print(x).
```

(i) Iterating 2-D Arrays.

- In 2-D array it will go through all the rows.

```
arr = np.array([[1,2,3],[4,5,6]]).
for x in arr:
    print(x).
```

- If we iterate on n-D array it will go through n-1th dimension one by one.

(ii) Iterating 3D-Arrays.

- In a 3-D Array it will go through all the 2D-Arrays.

```
arr = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]
for x in arr:
    print(x).
```

(iii) Iterating Arrays using nditer().

- The function nditer() is helping function that can be used very basic to very advanced iterations. ~~It solves~~
- It solves some basic issues which we face in iteration

(iv) Iterating Array with Different Datatypes?

- We can use op_dtypes argument and pass it the expected datatype.
- Numpy does not change the data type of element in-place.
- order to enable it in nditer() we pass flags = ['buffered'].

```
arr = np.array([1,2,3])
for x in np.nditer(arr, flags['buffered'], [op_dtypes['S']]
    print(x).
```

(4) Iterating with different step size.

```
arr = np.array([[1,2,3,4],[5,6,7,8]])
for x in np.nditer(arr[:, ::2]):
    print(x)
```

Enumerated Iteration using ndenumerate()

- Enumeration means mentioning sequence number of something one by
- something we require corresponding index of element while iterating. ndenumerate() method can be used for those use case.

```
arr = np.array([1,2,3])
for idx, x in np.ndenumerate(arr):
    print(idx, x)
```

(8) Numpy Joining Array.

- Joining means putting contents of two or more arrays in a single arra
- In Numpy we join Arrays by axes.
- We pass a sequence of Arrays that we want to join to the concatenate(function, along with the axis.
- If axis is not explicitly passed, it is taken as 0.

```
arr1 = np.array([1,2,3,4])
arr2 = np.array([5,6,7,8])
arr = np.concatenate((arr1, arr2))
print(arr)
```

(i) Joining Array using stack functions.

- Stacking same as concatenation (stacking is done along new axis).
- We pass a sequence of array that we want to join the stack() metho along with axis.

```
arr = np.stack((arr1, arr2), axis=1)
```

Rows

```
arr = np.hstack((arr1, arr2), axis=1)
```

column.

```
arr = np.vstack((arr1, arr2), axis=1)
```

⑨ Splitting Numpy Arrays.
- Splitting is reverse operation of joining.
- joining means multiple arrays into one and splitting breaks one array into multiple.
- use array_split() for splitting arrays.

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

Note: The return value is an array containing three arrays.

⑩ Numpy Array Search:
- you can search an array for a certain value, and return the indexes that get match.
- Search an array, use the where() method.

```
arr = np.array([1, 2, 3, 4, 5])
x = np.where(arr == 4).
```

(i) search sorted()
- there is method called searchsorted() which performs a binary search in the array.
- return index where the specified value would be inserted to maintain search order.

```
searchsorted()
```

⑪ sorting Arrays:
- sorting means putting elements in an ordered sequence.
- order sequence is any sequence that has an order corresponding to elements
- Numpy ndarray object has a function called sort().

(12) NumPy Random.

- Random means that can not predicted logically.
- Random ~~different~~ numbers does not mean different numbers everytime.