Numpy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

Why Use NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

Operations using NumPy

Using NumPy, a developer can perform the following operations

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation

Every item in a ndarray takes the same size as the block in the memory. Each element in ndarray is an object of the data-type object (called **dtype**).

Arrays
Creating array with numpy
NumPy is used to work with arrays. The array object in NumPy is called ndarray. We can create a NumPy ndarray object by using the array() function.

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

Output
```
[12345]
```

Creating 1-Dimentional, 2-Dimentional and 3-Dimentional Array
- An array that has 0-D arrays as its elements is called uni-dimensional or 1-D array. These are the most common and basic arrays.
- An array that has 1-D arrays as its elements is called a 2-D array. These are often used to represent matrix or 2nd order tensors.
- 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 for Creating 1-D, 2-D, 3-D array
```
import numpy as np
arr1 = np.array([1, 2, 3, 4, 5])  // One dimentional array
arr2 = np.array([[1, 2, 3], [4, 5, 6]]) //Two dimentional array
arr3 = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]]) //Three dimentional array
print(“1-D array\n”, arr1)
print(“2-D array \n”, arr2)
print(“3-D array \n”, arr3)
```

Output
```
1-D array
[1,2,3,4,5]
2-D array
[1,2,3]
[4.5.6]
3-D array
[1,2,3]
[4,5,6]
[1,2,3]
[4,5,6]
```

Check Number of Dimensions

NumPy Arrays provides the **ndim** attribute that returns an integer that tells us how many dimensions the array have.

Example

```
import numpy as np
arr1 = np.array([1, 2, 3, 4, 5])  // One dimentional array
arr2 = np.array([[1, 2, 3], [4, 5, 6]]) //Two dimentional array
arr3 = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]]) //Three dimentional array
print(arr1.dim)
print(arr2.dim)
print(arr3.dim)
```

Output

```
1
2
3
```

Data Types in NumPy

NumPy has several datatypes, 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, normally either int64 or int32
b – boolean, true or false
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)

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

Datatype Example

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

Output

```
int64
```

Example-2

```
import numpy as np
arr = np.array([‘apple’, ‘banana’, ‘Mango’, ‘Cherry’])
print(arr.dtype)
```

Output

```
<U6
```

Slicing arrays

- Slicing in python means taking elements from one given index to another given index.
- We pass slice instead of index like this: [start:end].
- We can also define the step, like this: [start:end:step].
- If we don't pass start its considered 0
- If we don't pass end its considered length of array in that dimension
- If we don't pass step its considered 1

Example 1

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

Output

```
[2 3 4 5]
```

Example 2

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

Output

```
[5 6 7]
```

Example -3

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

Output

```
[1 2 3 4]
```

Shape of an Array
The shape of an array is the number of elements in each dimension.

Get the Shape of an Array
NumPy arrays have an attribute called **shape** that returns a tuple with each index having the number of corresponding elements.

Example
Print the shape of a 2-D array:

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

The example above returns (2, 4), which means that the array has 2 dimensions, where the first dimension has 2 elements and the second has 4.

Scipy

SciPy is a free and open-source Python library used for scientific computing and technical computing.
It is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python.
It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

Why use SciPy

- SciPy contains varieties of sub packages which help to solve the most common issue related to Scientific Computation.
- SciPy package in Python is the most used Scientific library only second to GNU Scientific Library for C/C++ or Matlab's.
- Easy to use and understand as well as fast computational power.
- It can operate on an array of NumPy library.

Numpy VS SciPy
Numpy:
Numpy is written in C and use for mathematical or numeric calculation.
It is faster than other Python Libraries
Numpy is the most useful library for Data Science to perform basic calculations.
Numpy contains nothing but array data type which performs the most basic operation like sorting, shaping, indexing, etc.

SciPy:
SciPy is built in top of the NumPy
SciPy module in Python is a fully-featured version of Linear Algebra while Numpy contains

only a few features.
Most new Data Science features are available in Scipy rather than Numpy.

Sub-packages of SciPy:

- File input/output – scipy.io
- Special Function – scipy.special
- Linear Algebra Operation – scipy.linalg
- Interpolation – scipy.interpolate
- Optimization and fit – scipy.optimize
- Statistics and random numbers – scipy.stats
- Numerical Integration – scipy.integrate
- Fast Fourier transforms – scipy.fftpack
- Signal Processing – scipy.signal
- Image manipulation – scipy.ndimage

File Input / Output package:
Scipy, I/O package, has a wide range of functions for work with different files format which are Matlab, Arff, Wave, Matrix Market, IDL, NetCDF, TXT, CSV and binary format.

Special Function package
scipy.special package contains numerous functions of mathematical physics.
SciPy special function includes Cubic Root, Exponential, Log sum Exponential, Lambert, Permutation and Combinations, Gamma, Bessel, hypergeometric, Kelvin, beta, parabolic cylinder, Relative Error Exponential, etc..

Linear Algebra with SciPy
Linear Algebra of SciPy is an implementation of BLAS and ATLAS LAPACK libraries.
Performance of Linear Algebra is very fast compared to BLAS and LAPACK.
Linear algebra routine accepts two-dimensional array object and output is also a two-dimensional array.
Inverse Matrix , Eigenvalues and Eigenvector

Discrete Fourier Transform – scipy.fftpack
DFT is a mathematical technique which is used in converting spatial data into frequency data.
FFT (Fast Fourier Transformation) is an algorithm for computing DFT
FFT is applied to a multidimensional array.
Frequency defines the number of signal or wavelength in particular time period.

Optimization and Fit in SciPy – scipy.optimize
Optimization provides a useful algorithm for minimization of curve fitting, multidimensional or scalar and root fitting.

Integration with Scipy – Numerical Integration

When we integrate any function where analytically integrate is not possible, we need to turn for numerical integration.

SciPy provides functionality to integrate function with numerical integration.

scipy.integrate library has single integration, double, triple, multiple, Gaussian quadrate, Romberg, Trapezoidal and Simpson’s rules.