NumPy Introduction

What is NumPy?

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
- NumPy stands for Numerical Python.

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.

Installation of NumPy

- If you have Python and PIP already installed on a system, then installation of NumPy is very easy.

Install it using this command:

```
C:\Users>pip install numpy
```

Import NumPy

- Once NumPy is installed, import it in your applications by adding the import keyword:
- Import numpy
- Now NumPy is imported and ready to use.

```
Importnumpy
Arr = numpy.array([1, 2, 3, 4, 5])
Print(arr)
```

OutPut: [1,2,3,4,5]

NumPy Creating Arrays

- Create a NumPyndarray Object
- NumPy is used to work with arrays. The array object in NumPy is called ndarray.
- We can create a NumPyndarray object by using the array() function.

```
Importnumpy as np
Arr = np.array([1, 2, 3, 4, 5])
Print(arr)
Print(type(arr))
```

OutPut:
[1 2 3 4 5]
<class 'numpy.ndarray'>

NumPy Array Indexing

- Array indexing is the same as accessing an array element.
- You can access an array element by referring to its index number.
- The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1 etc.

Example:

Get the first element from the following array:

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

```
OutPut: 1
```

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

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

```
OutPut: [2 3 4 5]
```

NumPy Data Types

Data Types in Python

By default Python have these data types:

- `strings` - used to represent text data, the text is given under quote marks. e.g. "ABCD"
- `integer` - used to represent integer numbers. e.g. -1, -2, -3
- `float` - used to represent real numbers. e.g. 1.2, 42.42
- `boolean` - used to represent True or False.
- `complex` - used to represent complex numbers. e.g. 1.0 + 2.0j, 1.5 + 2.5j

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)

NumPy Array Copy vs View

The Difference Between Copy and View

- The main difference between a copy and a view of an array is that the copy is a new array, and the view is just a view of the original array.
- The copy *owns* the data and any changes made to the copy will not affect original array, and any changes made to the original array will not affect the copy.

- The view *does not own* the data and any changes made to the view will affect the original array, and any changes made to the original array will affect the view.

Random Numbers in NumPy

What is a Random Number?

- Random number does NOT mean a different number every time. Random means something that can not be predicted logically.

Pseudo Random and True Random.

- Computers work on programs, and programs are definitive set of instructions. So it means there must be some algorithm to generate a random number as well.
- If there is a program to generate random number it can be predicted, thus it is not truly random.
- Random numbers generated through a generation algorithm are called *pseudo random*.
- Can we make truly random numbers?
- Yes. In order to generate a truly random number on our computers we need to get the random data from some outside source. This outside source is generally our keystrokes, mouse movements, data on network etc.

- We do not need truly random numbers, unless it is related to security (e.g. encryption keys) or the basis of application is the randomness (e.g. Digital roulette wheels).
- In this tutorial we will be using pseudo random numbers.

Generate Random Number

- NumPy offers the `random` module to work with random numbers.

Example:

Generate a random integer from 0 to 100:

```
from numpy import random
x = random.randint(100)
print(x)
```

```
OutPut: 18
```

Generate Random Array

- In NumPy we work with arrays, and you can use the two methods from the above examples to make random arrays.

Integers

The `randint()` method takes a `size` parameter where you can specify the shape of an array.

Example

Generate a 1-D array containing 5 random integers from 0 to 100:

```
from numpy import random
x=random.randint(100, size=(5))
print(x)
```

OutPut: [70 62 59 33 82]