MALLA REDDY UNIVERSITY

UNIT-1

INTRODUCTION TO PYTHON

NumPy:

Total covering topics under this NumPy array

1. What is **NumPy? Why Use NumPy**

2.Installing NumPy & import NumPy

3. Differences between a Python list and a NumPy array

4. Uses of NumPy

5. . Basic array operations (*addition, subtraction, multiplication, division)*

6. What is an array?

6. How to create a basic array

(np.array(), np.zeros(), np.ones(), np.empty(), np.arange(), np.linspace(), dtype)

7. Commands of NumPy library related to Multidimensional arrays.

8 . Adding, removing, and sorting elements

8. How do you know the shape and size of an array (ndarray.ndim, ndarray.size, ndarray.shape)

9. reshape an array (reshape (row, column))

10. Indexing and slicing

11. 1D,2D and 3D arrays

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 Pytho**n.

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 is very easy.
- Arrays are very frequently used in **data science**, where speed and resources are very important.
- It consumes **less memory**.
- It is **fast** as compared to the python List.

NumPy Installation and Importing:

>>> check NumPy is alreardy installed or not by using following command

>>>import NumPy   # you get the prompt (>>>) with out having errors means NumPy is there. Otherwise install NumPy.

Install NumPy Package: [by using following pip command]

```
>>>  pip install numpy
```

Import NumPy by using following command

```
>>>  import numpy as np
```

Note: np is reference of Numpy library, once you import numpy , then execute all commands no need to import every time.

Differences between lists and NumPy arrays in python:

- There are several important differences between NumPy arrays and list, NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically).
- The main benefits of using NumPy arrays should be smaller memory consumption and better runtime behaviour then list

Size - Numpy data structures take up less space then lists

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

Functionality - NumPy have optimized functions such as linear algebra operations built in.

- **Comparing Memory use of NumPy & list**

```
import numpy as np
import time
import sys
# Creating a NumPy array with 10 elements
        array = np.arange(10)
#    array.itemsize : Size of one element
#    array.size : length of array
print("Size of NumPy array: ", array.size * array.itemsize)
# Creating a list with 10 elements
# Now I'll print the size of list
list = range(0, 10)
# Multiplying size of 1 element with length of the list
print("Size of list: ", sys.getsizeof(1)*len(list))
```

Output:

```
Size of NumPy array: 40
Size of list: 280
```

Converting list as a array*:*

Lists can be converted to arrays using the built-in functions in the Python numpy library.

numpy provides us with two functions to use when converting a list into an array:

1. numpy.array()
2. numpy.asarray()

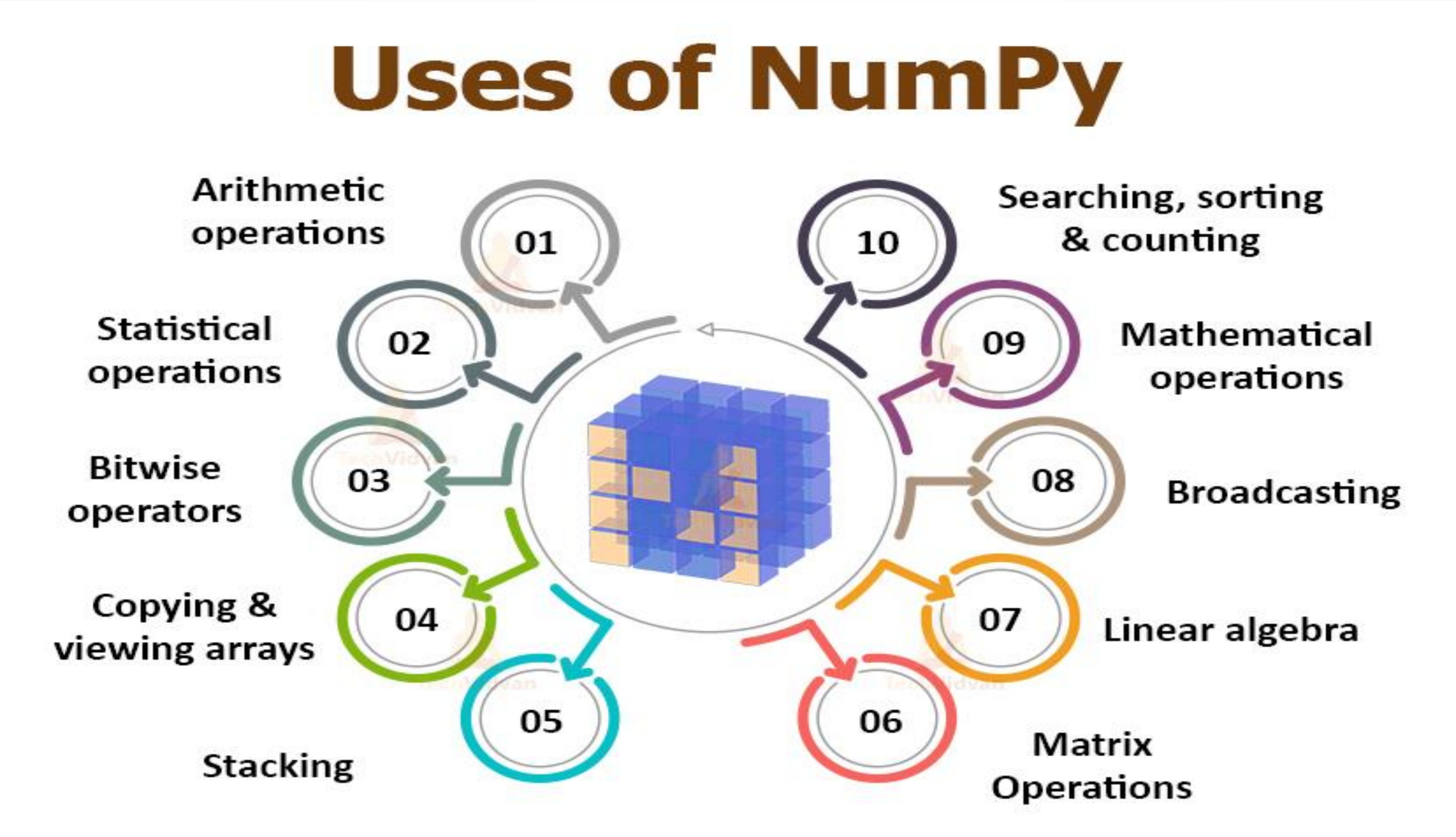
Using numpy. asarray() This function calls the numpy.array() function inside itself.

```
a=[10,20,30]
print(type(a))
# here 'a' is list
# 'a' is converted as a array by using asarray()
b=np.asarray(a,dtype=int)
print(b)
# all numpy arrays are ndarrays
print(type(b))
```

Output:

```
<class 'list'>
 [10 20 30]
<class 'numpy.ndarray'>
```

Uses of NumPy



Basic array operations (*addition, subtraction, multiplication, division) in NumPy array:*

- **<u>NumPy Addition operation</u>**

```
import numpy as np
import time

a1 = np.array([1, 2, 3])
a2 = np.array([4, 5, 6])

# To addition of  two arrays you can simply do it by
print("ADD a1 and a2 elements : ", a1 +  a2)

# To subtraction of  two arrays you can simply do it by
print("SUB a1 and a2 elements : ", a1 - a2)

# To multiplication of  two arrays you can simply do it by
print("MUL a1 and a2 elements : ", a1 * a2)
# To division of  two arrays you can simply do it by
print(“DIV a1 and a2 elements : ", a1 / a2)
```

Output:

```
ADD a1 and a2 elements : [5 7 9]
 SUB a1 and a2 elements : [-3 -3 -3]
MUL a1 and a2 elements : [ 4 10 18]
Division a1 and a2 elements [0.25 0.4 0.5 ]
```

What is an array:

An array is a central data structure of the NumPy library. An array is a grid of values and it contains information about the raw data, how to locate an element, and how to interpret an element. It has a grid of elements that can be indexed in various ways. The elements are all of the same type, referred to as the array dtype.

An array can be indexed by a tuple of nonnegative integers, by booleans, by another array, or by integers. The rank of the array is the number of dimensions. **The shape of the array is a tuple of integers giving the size of the array along each dimension.**

There are several ways to create basic NumPy arrays by using following functions.

np.array(), np.zeros(), np.ones(), np.empty(), np.arange(), np.linspace(), dtype

Creating a One-dimensional Array:

First, let’s create a one-dimensional array or an array with a rank 1. **arange** is a widely used function to quickly create an array. Passing a value 20 to the arange function creates an array with values ranging from 0 to 19.

```
import numpy as np
array = np.arange(20)
print(array)
```

Output: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19]

```
array. shape
```

Output:	(20,)

```
array[3]    #   output: 3
array[3]=25
print(array)
[ 0 1 2 25 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19]
```

Creating a Two-dimensional Array:

Let's talk about creating a two-dimensional array. If you only use the arange function, it will output a one-dimensional array. To make it a two-dimensional array, chain its output with the **reshape** function.

```
array = np.arange(20).reshape(4,5)
print(array)
```

Output:

```
[[ 0   1   2   3   4]
 [ 5   6   7   8   9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
array.shape
Output:  (4, 5)
```

```
print(array[3][4])    #    output: 19
array[3]=25
print(array)
[ 0 1 2 25 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19]
```

Creating a Three-dimensional Array:

To create a three-dimensional array, specify 3 parameters (subscript numbers) to the **reshape** function.

```
array = np.arange(27).reshape(3,3,3)

print(array)
```

Output:

```
[[[ 0 1 2]
 [ 3 4 5]
[ 6 7 8]]

[[ 9 10 11]
 [12 13 14]
 [15 16 17]]

 [[18 19 20]
[21 22 23]
[24 25 26]]]
```

```
print(array[3][4])   #   output: 19

array[3]=25

print(array)

[ 0 1 2 25 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19]
```

Creating a 1D ,2D & 3D Arrays using array():

```
# 1D array
import numpy as np
a=np.array([10,20,30])
print(a)
```

Output:

```
[10 20 30]
```

```
# 2D array
import numpy as np
a=np.array([[10,20,30],[1,2,3]])
print(a)
```

Output:

```
[[10 20 30]
[ 1 2 3]]
```

```
# 3D array
import numpy as np
a=np.array([[[10,20,30],[1,2,3]],[[10,60,30],[11,12,13]]])
print(a)
```

Output:

```
[[[10 20 30]
[ 1 2 3]]
[[10 60 30]
 [11 12 13]]]
```

```
# access 8th element from 3D array
print(a[1,0,1]) #   60
```

NumPy Array

- **1. Array of integers, floats and complex Numbers**

```
import numpy as np
 # array with integer
A = np.array([[1, 2, 3], [3, 4, 5]])
print(A)
 # array with float
A = np.array([[1.1, 2, 3], [3, 4, 5]])                  print(A)
  # Array of complex numbers
A = np.array([[1, 2, 3], [3, 4, 5]], dtype = complex)     print(A)
```

Creating a 1D ,2D & 3D Arrays using zeros():

```
# 1D array
import numpy as np
a=np.zeros(3)
print(a)
```

Output:

```
[0. 0. 0.]
```

```
# 2D array
import numpy as np
a=np.zeros([2,3])
print(a)
```

Output:

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

```
# 3D array
import numpy as np
a=np.zeros([2,3,3])
print(a)
```

Output:

```
[[[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]]
[[0. 0. 0.]
[0. 0. 0.]
 [0. 0. 0.]]]
```

Creating a 1D ,2D & 3D Arrays using ones():

# 1D array

```
import numpy as np
a=np.ones(3)
print(a)
```

Output:

```
[1. 1. 1.]
```

# 2D array

```
import numpy as np
a=np.ones([2,3])
print(a)
```

Output:

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

# 3D array

```
import numpy as np
a=np.ones([2,3,3])
print(a)
```

Output:

```
[[[1. 1. 1.]
[1. 1. 1.]
[1. 1. 1.]]
 [[1. 1. 1.]
 [1. 1. 1.]
[1. 1. 1.]]]
```

Creating a 1D ,2D & 3D Arrays using full():

# 1D array

```
import numpy as np
a=np.full([3],1)
# here 1 indicate filled element
print(a)
```

Output:

```
[1. 1. 1.]
```

# 2D array

```
import numpy as np
a=np.full([2,3],4)
print(a)
```

Output:

```
[[4 4 4]
[4 4 4]]
```

# 3D array

```
import numpy as np
a=np.full([2,3,3],5)
print(a)
```

Output:

```
[[[5 5 5]
 [5 5 5]
 [5 5 5]]
[[5 5 5]
[5 5 5]
 [5 5 5]]]
```

NumPy Array

Copying from One Array to Another

```
import numpy as np
a=np.full([2,3,2],5)
print(a)

b=np.full([2,3,2],3)
print(b)

np.copyto(b,a) #  copy content of a to b
print(b)
Output:
 [[[5 5]
 [5 5]
[5 5]]
[[5 5]
[5 5]
[5 5]]]
```

Output

# ‘a’ out put

```
[[[5 5]
[5 5]
[5 5]]
 [[5 5]
[5 5]
 [5 5]]]
```

# ‘b’ output

```
 [[[3 3]
[3 3]
 [3 3]]
[[3 3]
[3 3]
 [3 3]]]
```

NumPy Array

```
a = np.arange(27).reshape(3,3,3)
print(a)
# transpose of a given 3D array
print(a.transpose())
```

Output:

# ‘a’ output

```
[[[ 0 1 2]
[ 3 4 5]
 [ 6 7 8]]

 [[ 9 10 11]
 [12 13 14]
 [15 16 17]]

[[18 19 20]
[21 22 23]
[24 25 26]]]
```

# ‘Transpose’ output

```
[[[ 0 9 18]
[ 3 12 21]
[ 6 15 24]]

[[ 1 10 19]
 [ 4 13 22]
[ 7 16 25]]

 [[ 2 11 20]
 [ 5 14 23]
[ 8 17 26]]]
```

Commands of NumPy library related to Multidimensional arrays:

1. array()
2. Arange()
3. Zeros()
4. Ones()
5. Full()
6. Copy()
7. Transpose()

NumPy Array

Matrix Operations

- addition of two matrices, multiplication of two matrices and transpose of a matrix.
- We used nested lists before to write those programs.
- Let's see how we can do the same task using NumPy array.

- Addition of Two Matrices
- We use + operator to add corresponding elements of two NumPy matrices.

```
import numpy as np
 A = np.array([[2, 4], [5, -6]])
B = np.array([[9, -3], [3, 6]])
C = A + B      # element wise addition
print(C)
```

```
'''
Output:
[[11  1]
 [ 8  0]]
 '''
```

NumPy Array

- **Multiplication of Two Matrices**
- To multiply two matrices, we use **dot() method**. Learn more about how numpy.dot works.
- **Note:** * is used for array multiplication (multiplication of corresponding elements of two arrays) not matrix multiplication.

```
import numpy as np

A = np.array([[3, 6, 7], [5, -3, 0]])
B = np.array([[1, 1], [2, 1], [3, -3]])
C = A.dot(B)
print(C)
```

```
'''
Output:
[[ 36 -12]
 [ -1   2]]
'''
```

NumPy Array

- **Access matrix elements, rows and columns**
- **Access matrix elements**
- Similar like lists, we can access matrix elements using index. Let's start with a one-dimensional NumPy array.
-

```
import numpy as np
A = np.array([2, 4, 6, 8, 10])

print("A[0] =", A[0])     # First element
print("A[2] =", A[2])     # Third element
print("A[-1] =", A[-1])   # Last element
```

Output:

```
A[0] = 2
A[2] = 6
A[-1] = 10
```

NumPy Array

- Now, let's see how we can access elements of a **two-dimensional array** (which is basically a matrix).

```
import numpy as np
 A = np.array([[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]])

#  First element of first row
print("A[0][0] =", A[0][0])
 # Third element of second row
print("A[1][2] =", A[1][2])
 # Last element of last row
print("A[-1][-1] =", A[-1][-1])
```

When we run the program, the output will be:

```
A[0][0] = 1
A[1][2] = 9
A[-1][-1] = 19
```

NumPy Array

- **Access rows of a Matrix**

```
import numpy as np

A = np.array([[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]])

print("A[0] =", A[0]) # First Row
print("A[2] =", A[2]) # Third Row
print("A[-1] =", A[-1]) # Last Row (3rd row in this case)
```

When we run the program, the output will be:

```
A[0] = [1, 4, 5, 12]
A[2] = [-6, 7, 11, 19]
A[-1] = [-6, 7, 11, 19]
```

NumPy Array

- **Access columns of a Matrix**

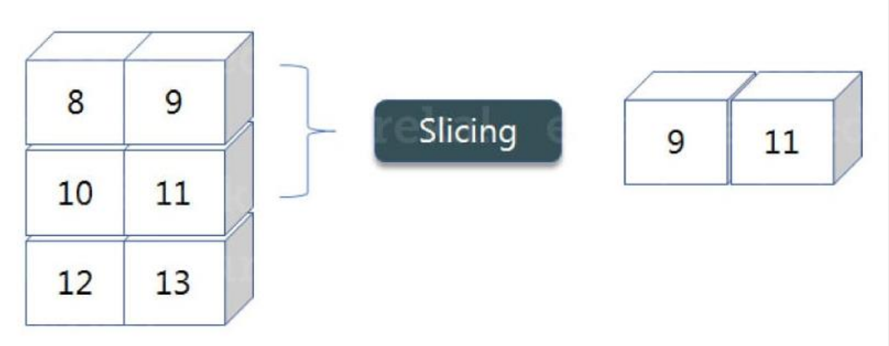
```
import numpy as np

A = np.array([[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]])

print("A[:,0] =",A[:,0]) # First Column
print("A[:,3] =", A[:,3]) # Fourth Column
print("A[:,-1] =", A[:,-1]) # Last Column (4th column in this case)
```

When we run the program, the output will be:

```
A[:,0] = [ 1 -5 -6]
A[:,3] = [12  0 19]
A[:,-1] = [12  0 19]
```

NumPy Array

- **Slicing of a Matrix**
- Slicing of a one-dimensional NumPy array is similar to a list.
- Let's take an example:



```
import numpy as np
letters = np.array([1, 3, 5, 7, 9, 7, 5])

# 3rd to 5th elements
print(letters[2:5])        # Output: [5, 7, 9]

# 1st to  _ elements
print(letters[:-5])        # Output: [1, 3]
```

```
# 6th to last elements
print(letters[5:])         # Output:[7, 5]

# 1st to last elements
print(letters[:])          # Output:[1, 3, 5, 7, 9, 7, 5]

# reversing a list
print(letters[::-1])       # Output:[5, 7, 9, 7, 5, 3, 1]
```

NumPy Array

- Now, let's see how we can slice a matrix.

```
import numpy as np

A = np.array([[1, 4, 5, 12, 14],
    [-5, 8, 9, 0, 17],
    [-6, 7, 11, 19, 21]])

print(A[:2, :4])  # two rows, four columns

''' Output:
[[ 1  4  5 12]
 [-5  8  9  0]]
'''
```

```
print(A[:1,])  # first row, all columns

''' Output:
[[ 1  4  5 12 14]]
'''
 print(A[:,2])  # all rows, second column

''' Output:
[ 5  9 11]
'''
 print(A[:, 2:5])  # all rows, third to fifth column

'''Output:
[[ 5 12 14]
 [ 9  0 17]
 [11 19 21]]
'’’
```

MALLA REDDY UNIVERSITY

www.mallareddyuniversity.com