NumPy

```
[ ]  # Numpy: Used for Mathematical and logical calculations on Arrays.
     # Numpy is used for mathematical calculations
     # Numpy array is fast and efficient and provide many functions.
     # 1 D array: Vetor
     # 2 D array: Matrix
     # 3 D array: Tensor
```

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

# input 1-d array from user
num = int(input("enter the the size of array"))
list1 = []
for i in range(num):
    num1 = int(input("Enter number"))
    list1.append(num1)
myarr = np.array(list1)
print(myarr)
```

```
# 2d arary using numpy
a = [[1,2,3],[4,5,6],[6,4,3]]
myarr = np.array(a)
print(myarr)

# input 2-D from user
num = int(input("Enter the number of the elements:"))
list2 = []
for i in range(num):
    list1 = []
    for j in range(num):
        num1 = int(input(""))
        list1.append(num1)
    list2.append(list1)
print(list2)

myarr = np.array(list2)
print(myarr)
```

```
# myarr.ndim  myarr.shape  myarr.size

print("Total dimension:",myarr.ndim)      # <ArrayName>.ndim
print("Shape:",myarr.shape)               # <ArrayName>.shape
print("Size:",myarr.size)                 # <ArrayName>.size
```

NumPy

```
# Numpy functions in python
# A) zeros(): This function creates an arary(1d and 2d) and fill all the values with 0.
myarr = np.zeros(5)
print(myarr)
myarr= np.zeros((2,3))  # 2 rows and 3 column
print(myarr)


# B) ones(): This function creates an arary(1d and 2d) and fill all the values with 1.
myarr = np.ones((2))
print(myarr)


# C) eye(): This function creates an 2-d array with digonal element as 1 and rest with 0.
myarr = np.eye(3)  # automatic create an 3*3 matrix.
print(myarr)
myarr = np.eye(3,4)  # an eye matirx with 3 rows and 4 columns
print(myarr)
```

```
# D) diag(): This fucntion creates a 2-d array with all the diagonal elements as given values and rest as 0.
myarr = np.diag([1,2,3,4])
print(myarr)
# to get the diagonal elements in 2-d array we use as a fuction.
myarr = np.diag([1,2,3,4])
print(myarr)
print(np.diag(myarr))     # np.diag(<ArrayName>) return all digonal elements of given array.


# E) randint: This function is used to generate random numeber bewtween a given range.
# syntax : np.random.randint(min,max,total_values)
# UpperBound(max) not included.
myarr = np.random.randint(1,10,3)
print(myarr)
```

```
# F) rand: This functin is used to generate a random values between 0 and 1.
# syntax: np.random.rand(5)
myarr = np.random.rand(5)   # any 5 decimal values between 0 to 1.
print(myarr)


# G) randn: This function is used to generate a random values close to 0.
# this may generate positive or negative number as well.
# syntax: np.random.randn(<number_of_values>
arr = np.random.randn(5)  # numbers can be positive as well as -ve.
print(arr)
```

```
# myarr.reshape(<rownow>,<colnow>)
myarr = np.random.randint(1,50,12)
myarr2 = myarr.reshape(3,4)
print(myarr2)
myarr2 = myarr.reshape(4,-1)  # -ve number columns decide by python itself
print(myarr2)
myarr2 = myarr.reshape(-1,2) # assigned arguments required to be multiple of total numbers of elements.
print(myarr2)
```

NumPy

```
# np.random.seed()
myarr = np.array([1,234,354,65,7])

myarr = np.random.seed(34)      # fixed the random generated values.
myarr = np.random.randint(1,50,10)
print(myarr)
```

```
# indexing in 2-d array
# myarr[<row : col> , <row : col>]
myarr = np.array([[1,234,354],[65,7,45],[24,65,23]])
print(myarr)
print(myarr[2:3 , 2])
```

```
# view vs copy
# view: if we make any changes to sub array it get reflected to
# org array
myarr = np.array([1,2,3,50,3,5,35,5])
slice = myarr[1:6]
slice[1:] = 0
print(slice)
print(myarr)

# myarr.copy(): create the copy of org array
# copy: if we make any changes to sub array it doesn't get reflect
# org array
myarr = np.array([1,2,3,50,3,5,35,5])
slice = myarr[1:6].copy()
slice[1:] = 0
print(slice)
print(myarr)
```

```
# conditional statements

import numpy as np
myarr = np.array([1,2,3,50,3,5,35,5])
print(myarr>3)

# eq1
myarr = np.array([1,2,3,50,3,5,35,5])
print(myarr[myarr%2 == 1])
print(myarr)  # org array

# eq2
myarr = np.array([1,2,3,50,3,5,35,5])
myarr[myarr%2 == 0] = 0
print(myarr)
```

NumPy

```
# vstack(())  hstack(())      .sum([])
import numpy as np
n1 = np.array([1,2,3])
n2 = np.array([4,5,6])
arr= np.vstack((n1,n2))
print(arr)

n1 = np.array([1,2,3])
n2 = np.array([4,5,6])
arr = np.hstack((n1,n2))
print(arr)

n1 = np.array([10,20,30])
n2 = np.array([30,40,50])
arr = np.sum([n1,n2])        # all elements sum
print(arr)
arr = np.sum([n1,n2], axis = 0)        # col wise sum
print(arr)
arr = np.sum([n1,n2], axis = 1)        # row wise sum in single line
print(arr)
# operations on numpy
# 1) any of the addition subtraction multiplication power operations can be performed
# In numpy if we divide array with 0 it give infinity and not any error.

# 2) also we can add two arrays(1-D)
myarr1 = np.array([1,2,3,4])
myarr2 = np.array([5,6,7,8])
print(myarr1 + myarr2)

# 3) Same operations on 2d array
myarr1 = np.array([1,2,3,4]).reshape(2,2)
myarr2 = np.array([5,6,7,8]).reshape(2,2)
print(myarr1 + myarr2)

# eq1)
myarr1 = np.array([1,2,3,4]).reshape(2,2)
myarr2 = np.array([5,6,7,8]).reshape(2,2)
print(myarr1/myarr2)      # divisible operations may return float value.

# eq 2)
# normal multiplication can be done in the form of dot product
myarr1 = np.array([1,2,3,4]).reshape(2,2)
myarr2 = np.array([5,6,7,8]).reshape(2,2)
print(myarr1.dot(myarr2))
```

NumPy

```
# Functions in python part 1
arr=np.array([10,20,30,40,50])
np.max(arr) # returns the max number
np.argmin(arr) # position of min number
np.min(arr) # retruns the min number
np.argmax(arr) # position of max number
np.sqrt(arr) # square root of every number
np.sin(arr) # sin value of every number
np.cos(arr) # cos value of every number
# Functions in python part 2
arr=np.array([10,20,30,40,50])
np.min(arr, axis=1) # finds row wise min
np.max(arr, axis=0) # finds col wise max
np.sum(arr, axis=1) # finds row wise sum
np.cumsum(arr, axis=0)
np.cumsum(arr) # cumulative
# Functions in python part 3
arr=np.array([10,20,30,40,50])
np.random.shuffle(arr)
np.unique(arr)
```