Trainer: Mr. Srinivas Reddy, Ph: 91 9292005440 Mail: info@datahill.in

NumPy:

NumPy is an open source library available in Python that used in mathematical, statistical operations, scientific, engineering, and data science programming.

Operations using NumPy:

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.

How to Install & import NumPy:

pip install numpy

import numpy as np

To check your installed version of Numpy

print (np.__version__)

Python Numpy Array:

===================

NumPy arrays are a bit like Python lists, but still very much different at the same time.

```
a = [10,20,30,40]
print(a)
# np.array() is used to convert python list to a numpy array
b = np.array(a)
print(b)

a  = np.array([10,20,30,40)
print(a)

# We can perform mathematical operations like additions, subtraction,
division and multiplication on an array.
a+10

print(a.shape)      # Shape of Array
print(a.dtype)      # internal data type

b  = np.array([10.5,20.2,30.6])
print(b.dtype)

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

a = np.array([1,2,3], dtype = complex)
```

```
print(a)

# 2 dim array
c = np.array([(1,2,3),
              (4,5,6)])
print(c.shape)

# 3 dim array
d = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
print(d.shape)

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

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

# NumPy also provides a reshape function to resize an array.
a = np.array([[1,2,3],[4,5,6]])
b = a.reshape(3,2)
print(b)
```

```
# ndarray.ndim attribute returns the number of array dimensions.
a = np.array([1,2,3,4,5,6,7,8])
print(a)
print(a.ndim)              # 1 dimensional array

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

# numpy.empty creates an uninitialized array of specified shape and dtype
Syntax: numpy.empty(shape, dtype = float, order = 'C')
# 'C' for C-style row-major array, 'F' for FORTRAN style column-major array
np.empty([3,2], dtype = int)
# The elements in an array show random values as they are not initialized

# numpy.zeros Returns a new array of specified size, filled with zeros.
np.zeros((2,2))
np.zeros((2,2), dtype=np.int16)

# numpy.ones Returns a new array of specified size, filled with ones.
np.ones(5)
```

```
np.ones([2,2], dtype = int)

# numpy.asarray is used to convert Python sequence into ndarray
a = [1,2,3]
b = np.asarray(a)# convert list to ndarray
print(b)

b = np.asarray(a, dtype = float)      # dtype is set
print(b)

a = (1,2,3)
b = np.asarray(a)# ndarray from tuple
print(b)

a = [(1,2,3),(4,5)]
b = np.asarray(a)        # ndarray from list of tuples
print(b)

# numpy.arange is used to print the sequence of values
numpy.arange(start, stop,step)
np.arange(5)
np.arange(5, dtype = float)
np.arange(1, 11)
```

np.arange(10,20,2)

numpy.linspace is similar to arange(),instead of step size, works with length

Syntax: numpy.linspace(start, stop, num, endpoint, retstep, dtype)

np.linspace(10,20,5)

np.linspace(10,20, 5, endpoint = False)

np.linspace(1.0, 5.0, 10)

np.linspace(1.0, 5.0, 5, endpoint=False)

np.linspace(1,2,5, retstep = True)

Indexing & Slicing:

===================

a = np.arange(10)

slice(2,7,2)

a[2:7:2]

a[5] # slice single item

a[2:] # slice items starting from index

a[2:5]# slice items between indexes

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

a[1:] # slice items starting from index

a[...,1] # this returns array of items in the second column

a[1,...] # Now we will slice all items from the second row

```
a[...,1:]       # Now we will slice all items from column 1 onwards

x = np.array([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
x[1:4,1:3]          # slicing
y = x[1:4,[1,2]]    # using advanced index for column

a = np.array([1,2,3,4])
b = np.array([10,20,30,40])
print(a+b)
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print(a+b)   # b is broadcast to become compatible with a

a.T    # Transpose
```

Statistical Functions

```
np.min(a)
np.max(a)
np.mean(a)
np.median(a)
np.std(a)
```

Arithmetic operations

```
np.add(a,b)
```

np.subtract(a,b)

np.multiply(a,b)

np.divide(a,b)

np.power(a,2)

NumPy package contains numpy.linalg module that provides all the functionality required for linear algebra.

a = np.array([[1,2],[3,4]])

b = np.array([[11,12],[13,14]])

np.dot(a,b) # [[1*11+2*13, 1*12+2*14],[3*11+4*13, 3*12+4*14]]

np.matmul(a,b)

np.vdot(a,b) # 1*11 + 2*12 + 3*13 + 4*14

np.inner(a,b) # 1*11+2*12, 1*13+2*14

3*11+4*12, 3*13+4*14

np.linalg.det(a) # Determinant ad-bc

x = np.array([[1,2,3],[4,5,6],[7,8,9]])

np.linalg.det(x) # 1*(5*9 - 6*8) + 2*(4*9 - 6*7) + 3*(4*8 - 5*7)

Generate random numbers:

========================

Random numbers between [0,1) of shape 2,2

print(np.random.rand(2,2))

Normal distribution with mean=0 and variance=1 of shape 2,2

```
print(np.random.randn(2,2))

# Random integers between [0, 10) of shape 2,2
print(np.random.randint(0, 10, size=[2,2]))

# One random number between [0,1)
print(np.random.random())

# Random numbers between [0,1) of shape 2,2
print(np.random.random(size=[2,2]))

# Pick 10 items from a given list, with equal probability
print(np.random.choice(['a', 'e', 'i', 'o', 'u'], size=10))

# Pick 10 items from a given list with a predefined probability 'p'
print(np.random.choice(['a', 'e', 'i', 'o', 'u'], size=10, p=[0.3, .1, 0.1, 0.4, 0.1]))

# Create the random state
rn = np.random.RandomState(100)

# Create random numbers between [0,1) of shape 2,2
print(rn.rand(2,2))
```

```
# Set the random seed
np.random.seed(100)

# Create random numbers between [0,1) of shape 2,2
print(np.random.rand(2,2))

# Create random integers of size 10 between [0,10)
np.random.seed(100)
arr_rand = np.random.randint(0, 10, size=10)
print(arr_rand)

# Get the unique items and their counts
uniqs, counts = np.unique(arr_rand, return_counts=True)
print("Unique items : ", uniqs)
print("Counts       : ", counts)
```