# Introduction to NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python. 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.

## Installing & Updating the Package

NumPy comes pre-isntalled if you're using Anaconda. **pip install numpy**

To get the latest version: **pip install numpy --upgrade**

## Importing NumPy

```
import numpy as np
```

```
a = np.array(45)
print(a)
print(type(a))
print(a.ndim)
```

```
45
<class 'numpy.ndarray'>
0
```

```
b = np.array([1,2,3,4,5])
print(b)
print(type(b))
print(b.ndim)
```

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

```
c = np.array([[1,2,3,4],[5,6,7,8]])
print(c)
print("-------------")
print(c[0:,2])
print("-------------")
print(c.ndim)
```

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

```
-------------
2
```

```
d = np.array([[[1,2,3]],[[4,5,6]]])
print(d)
print("-------------")
print(type(d))
print("-------------")
print(d[1:,:2])
print("-------------")
print(d.ndim)
```

```
[[[1 2 3]]

 [[4 5 6]]]
-------------
<class 'numpy.ndarray'>
-------------
[[[4 5 6]]]
-------------
3
```

```
e = np.array([[[10,20,30,40],[50,60,70,80]],[[90,100,110,120],
[130,140,150,160]]])
print(e)
print("-------------")
print(type(e))
print("-------------")
print(e[1:,1:,1:3])
print("-------------")
print(e.ndim)
print("-------------")
print(e[0,1,2])
print("-------------")
print(e[0:2,1,1:3])
```

```
[[[ 10  20  30  40]
  [ 50  60  70  80]]

 [[ 90 100 110 120]
  [130 140 150 160]]]
-------------
<class 'numpy.ndarray'>
-------------
[[[140 150]]]
-------------
3
-------------
70
-------------
```

```
[[ 60  70]
 [140 150]]
```

```
f = np.array([1,2,3,4,5.5,'Hi',False])
f
```

```
array(['1', '2', '3', '4', '5.5', 'Hi', 'False'], dtype='<U32')
```

## Numpy ones

The numpy.ones() function returns a new array of given shape and type, with ones.

```
np.ones((5,4),dtype=np.int64)
```

```
array([[1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1]], dtype=int64)
```

## Numpy zeros

The numpy.zeros() function returns a new array of given shape and type, with zeros.

```
np.zeros((7,5),dtype=np.int64)
```

```
array([[0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0]], dtype=int64)
```

## Numpy full

numpy.full(shape, fill_value, dtype = None, order = ‘C’) : Return a new array with the same shape and type as a given array filled with a fill_value.

```
np.full((7,7),7,dtype=np.float32)
```

```
array([[7., 7., 7., 7., 7., 7., 7.],
       [7., 7., 7., 7., 7., 7., 7.],
       [7., 7., 7., 7., 7., 7., 7.],
       [7., 7., 7., 7., 7., 7., 7.],
       [7., 7., 7., 7., 7., 7., 7.],
       [7., 7., 7., 7., 7., 7., 7.],
       [7., 7., 7., 7., 7., 7., 7.]], dtype=float32)
```

```
np.full((7,7),7,dtype=np.int64)
```

```
array([[7, 7, 7, 7, 7, 7, 7],
       [7, 7, 7, 7, 7, 7, 7],
       [7, 7, 7, 7, 7, 7, 7],
       [7, 7, 7, 7, 7, 7, 7],
       [7, 7, 7, 7, 7, 7, 7],
       [7, 7, 7, 7, 7, 7, 7],
       [7, 7, 7, 7, 7, 7, 7]], dtype=int64)
```

## Numpy Identity

numpy.identity(n, dtype = None) : Return a identity matrix i.e. a square matrix with ones on the main diagonal.

```
np.identity(10,dtype=np.int64)
```

```
array([[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]], dtype=int64)
```

```
np.identity(3,dtype=np.float32)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]], dtype=float32)
```

## Numpy Random Randint

numpy.random.randint() is one of the function for doing random sampling in numpy. It returns an array of specified shape and fills it with random integers from low (inclusive) to high (exclusive)

```
np.random.randint(0,255,(3,4))
```

```
array([[ 95, 112, 161, 148],
       [166, 107, 232, 167],
       [ 62,   9, 254, 156]])
```

## Some python functions

np.ndim(): Return the number of dimension of an array. np.min(): Return the minimum of an array. np.max(): Return the maximum of an array. np.sum(): Return the sum of an array. np.mean(): Return the mean of an array.

```
a = np.array([(8,9,10),(11,12,13),(7,14,15)])
print("Matrix: ",a)
print("Minimum value using slicing: ",a[1].min()) #Getting min value
using slicing
print("Number of dimensions: ",np.ndim(a))
print("Minimum value: ",np.min(a))
print("Maximum value: ",np.max(a))
print("Sum value: ",np.sum(a))
print("Mean value: ",np.mean(a))
```

```
Matrix:  [[ 8  9 10]
 [11 12 13]
 [ 7 14 15]]
Minimum value using slicing:  11
Number of dimensions:  2
Minimum value:  7
Maximum value:  15
Sum value:  99
Mean value:  11.0
```

np.sqrt(): Return the non-negative square-root of an array. np.std(): Returns the standard deviation, a measure of the spread of a distribution, of the array elements. The standard deviation is computed for the flattened array by default, otherwise over the specified axis. np.log(): Return the natural logarithm of an array.

```
a = np.array([(1,0,3),(3,4,5)])
print("Square Root: ",np.sqrt(a))
print("------------------------------------------------------------")
print("Standard Deviation: ",np.std(a))
print("------------------------------------------------------------")
print("Natural Logrithm: ",np.log(a))
print("------------------------------------------------------------")

b = np.sqrt(a)
print(b.round(3)) # Round of to 3 decimal points
```

```
Square Root:  [[1.         0.         1.73205081]
 [1.73205081 2.         2.23606798]]
------------------------------------------------------------
Standard Deviation:  1.699673171197595
------------------------------------------------------------
Natural Logrithm:  [[0.               -inf 1.09861229]
 [1.09861229 1.38629436 1.60943791]]
```

```
------------------------------------------------------------
[[1.    0.    1.732]
 [1.732 2.    2.236]]
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_12240\168162492.py:6: 
RuntimeWarning: divide by zero encountered in log
  print("Natural Logrithm: ",np.log(a))
```

```
a = np.random.randint(0,30,(3,3))
print(a)
print(np.sqrt(a).round(3))
```

```
[[24 24 13]
 [18 18 29]
 [16  8 17]]
[[4.899 4.899 3.606]
 [4.243 4.243 5.385]
 [4.    2.828 4.123]]
```

np.floor(): Return the floor of the input. np.ceil(): Return the ceiling of the input.

```
a = np.array([-1.7, -1.5, -0.2, 0.2, 1.5, 1.7, 2.0])
print("Floor of an array: ",np.floor(a))
print("----------------------------------------------------")
print("Ceil of an array: ",np.ceil(a))
```

```
Floor of an array:  [-2. -2. -1.  0.  1.  1.  2.]
----------------------------------------------------
Ceil of an array:  [-1. -1. -0.  1.  2.  2.  2.]
```

```
x = np.array([(1,2),(3,4)])
y = np.array([(5,6),(3,4)])

print(x+y) # Addition of 2 matrix
print(x-y) # Substraction of 2 matrix
print(x*y) # Multiplication of 2 matrix
print(x/y) # Division of 2 matrix
print(x%y) # Modulo of 2 matrix
print(x@y) # Matrix Multiplication
```

```
[[6 8]
 [6 8]]
[[-4 -4]
 [ 0  0]]
[[ 5 12]
 [ 9 16]]
[[0.2        0.33333333]
 [1.         1.        ]]
[[1 2]
 [0 0]]
```

```
[[11 14]
 [27 34]]
```

## ravel()

It converts n dimension array to a 1D array

```
x = np.array([(1,2,3),(3,4,5),(6,7,8),(9,10,11)])
x

array([[ 1,  2,  3],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

y = x.ravel()
y

array([ 1,  2,  3,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

## np.reshape

Gives a new shape to an array without changing its data.

```
y = y.reshape(3,4)
y

array([[ 1,  2,  3,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])

np.reshape(y,(3,4))

array([[ 1,  2,  3,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

## np.transpose

It wil convert rows into columns and Columns into rows.

```
y = y.transpose()
y

array([[ 1,  2,  3,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])

y = np.transpose(y)
y
```

```
array([[ 1,  4,  8],
       [ 2,  5,  9],
       [ 3,  6, 10],
       [ 3,  7, 11]])
```

```
np.transpose(y)
```

```
array([[ 1,  2,  3,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

### np.shape

It will give shape of an array

```
np.shape(y)
```

```
(4, 3)
```

```
y.shape
```

```
(4, 3)
```

## NumPy Documentation

https://numpy.org/devdocs/ <-  A link to the NumPy documentation