NUMPY ARRAY

Sr.No.	Syntax/command	Output	Remarks
1	import numpy as np		To activate numpy from library
2	np.array(l)	array([1, 2, 3, 4, 5, 6])	To convert list l to array

3	l1 = [4,5,6,7,"sudh', 45.56",True]
	np.array(l1)
		array(['4', '5', '6', '7', "sudh', 45.56", 'True'], dtype='<U12')	While converting list with all different kinds of values, array will change all values to string format. U12 in internal representation for numpy array.

4	np.array([[1,2,3], [4,5,6]])	array([[1, 2, 3], [4, 5, 6]])	List inside a list will give 2 dimensional array
5	np.array([[[1,2,3], [4,5,6], [7,8,9]]])	array([[[1, 2, 3], [4, 5, 6], [7, 8, 9]]])	List inside a list will give 3 dimensional array
6	a3.ndim	3'	Will indicate dimension of the array.
7	a3.size	9	Will give the number of elements in an array. A 3x3 array will yield value 9.
8	a1.shape	(2, 3)	Will indicate matrix size of the array. (2, 3) means 2x3 matrix.
9	a3.shape	(1, 3, 3)	Means we have 1 nos 3x3 matrix.
10	np.random.randint(2,50)	11	Will indicate a random integer from 2-50
11	np.random.randint(2,50,(3,4))	array([[49, 37, 42, 9], [24, 38, 5, 21], [9, 8, 25, 9]])	Will generate a 3x4 matrix with random values ranging from 2-50
12	np.random.rand(2,3)		Generates data with random mean and random standard deviation.
13	a4 = np.random.randn(3,4)		Generates random data of normal distribution with mean = 0 and standard deviation = 1. Avg of each of the column in the matrix will be zero
14	a4.reshape(2,6)		It will crop or reshape the 3x4 matrix to 2x6 matrix and to execute the command total no of elements should be same.
15	a4.reshape(12,-1)		Here -1 will auto calcutate value in its place based on matrix size and total elements. Any negative value can be used.

16	a4.reshape(2,3,2)		Will convert 2 dim array to 3 dim provided total no of elemnts are same in both cases i.e 2 nos of 3x2 matrix. (2*3*2=12)
17	a4.reshape(1,1,1,1,1,1,2,3,2)		The no of 1's within the bracket will increase the dimensions of the arrary i.e it will add the same no of square brackets as the no of 1's. Rest is 2*3x2 matrix
18	a1[2:6]		**Slicing operation** : It will extract data between index 2 to 6 (excluding upper limit) from the array.
19	a1[::-1]		It will reveerse the entire array.
20	a2[:,1:] or a2[:,[1,2]] or a2[[0,1] , 1:]		To extract rows and columns in a matrix array. The first part indicates rows and then the columns.
21	a5[a5>40]		Will give array data of all elements greater than 40
22	a5[0,1] = 99		Will change array element at 0,1 to 99
23	a6 * a7		Will execute multiplication element wise based on the coordinates.
24	a6 @ a7		**Matrix multiplication**
25	a6 + 100		It will add 100 to each and every element of a6
26	a6 * 2		It will multiply 2 to each and every element of a6
27	a6/0		It will divide 0 to each and every element of a7. It will give warning but still give the inf as ans.
28	a6**3		It will give power of 3 to each and every element of a6
29	np.zeros((4,4))		It will give array with all elements 0
30	np.ones((4,4))		It will give array with all elements 1
31	a9 + np.array([1,2,3,4])		**Broadcasting Operation**: Will add the array row wise to the rows of a9 array

32	np.array([[1,2,3,4]]).T + a9		T - transpose will switch the array vertically or horizontally provided you give it two dimension i.e additional brackets. And will carry out addition operation column wise now. **(Broadcasting operation)**
33	a6.T		Will interchange rows to columns and columns to rows.
34	np.sqrt(a5)		Will give square root of each and every element of a5
35	np.exp(a5)		Will give exponential of each and every element of a5
36	np.log10(a5)		Will give log value of each and every element of a5
37	list (range(0,10 ,2))	[0, 2, 4, 6, 8]	Perform list operation indicating values between 0 to 10 with a jump of 2. **Decimal values cannot be used for range function**
38	np.arange(10)	array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])	Similar to range function except that arange returns an array of range values
39	np.arange(0,10,2.5)	array([0. , 2.5, 5. , 7.5])	Used when jump size or values are in decimals.
40	np.linspace(2,3,num=50)		Will give array of 50 elements with values between range 2-3
41	np.linspace(2,3,num=50,retstep=True)	array([sample], step)	**Retstep command:** If True, return (`samples`, `step`), where `step` is the spacing between samples.
42	np.logspace(2,3,num=4,base=10)	array([100. , 215.443469 , 464.15888336, 1000.])	Will give log values to the base 10 between numbers 2 -3
43	np.eye(5)	array([[1., 0., 0., 0., 0.], [0., 1., 0., 0., 0.], [0., 0., 1., 0., 0.], [0., 0., 0., 1., 0.], [0., 0., 0., 0., 1.]])	Will give identity matrix of size 5x5 with diagonal elements as one and all other elements zero
43	a = np.array([3,4,5,6] , ndmin= 3)	array([[[3, 4, 5, 6]]])	It will return 3 dimensional array
43	arr.ndim	2	It will return dimension of the array
43	np.random.randint(1,10,(4,4,2))		It will generate 4 nos 4*2 array with random values between 1-10
43	print(np.__version__)		Checking NumPy Version

43	c =
	np.array([34,44,5,23,11,89,9])
	select = [0,1,2,3]
	d = c[select]
	d
		array([34, 44, 5, 23])	We can use the list as an argument in the brackets. The output is the elements corresponding to the particular indexes:

43	c =
	np.array([34,44,5,23,11,89,9])
	select = range(4)
	c[select] = 1000
	c
			It will take select list as argument and replace those indexes with number 1000.

43	u = np.array([1, 0])
	v = np.array([0, 1])
	z = np.add(u, v)
	z
		array([1, 1])	**Array addition**

43	a = np.array([10, 20, 30])
	b = np.array([5, 10, 15])
	c = np.subtract(a, b)
	c
		array([5, 10, 15])	**Array subtraction**

43	x = np.array([1, 2])
	y = np.array([2, 1])
	z = np.multiply(x, y)
	z
		array([2, 2])	**Array multiplication**

43	a = np.array([10, 20, 30])
	b = np.array([2, 10, 5])
	c = np.divide(a, b)
	c
		array([5., 2., 6.])	**Array division**

43	X = np.array([1, 2])
	Y = np.array([3, 2])
	np.dot(X, Y)
		7	**Dot product = 1*3+2*2=7**

43	u = np.array([1, 2, 3, -1]) u + 1	array([2, 3, 4, 0])	Adding Constant to a Numpy Array

43	np.pi
	x = np.array([0, np.pi/2 , np.pi])
	y = np.sin(x)
	y
		array([0.0000000e+00,
		1.0000000e+00,
		1.2246468e-16])
			Using pie values in numpy

43	np.linspace(-2, 2, num=5)	array([-2., -1., 0., 1., 2.])	A numpy array within [-2, 2] and 5 elements

43	arr1 = np.array([1, 2, 3])
	print(arr1)
	for x in arr1:
	  print(x)
		[1 2 3]
		1
		2
		3
			Iterating 1-D Arrays

43	A[1, 2]	Same ans	Extracting values from the array
43	A[1][2]		

43	X = np.array([[1, 0], [0, 1]])
	Y = np.array([[2, 1], [1, 2]])
	Z = X + Y
	Z
		array([[3, 1],
		[1, 3]])
			Matrix addition

43	Y = np.array([[2, 1], [1, 2]])
	Z = 2 * Y
	Z
		array([[4, 2],
		[2, 4]])
			Multiplying a matrix by a scaler

43	Y = np.array([[2, 1], [1, 2]])
	X = np.array([[1, 0], [0, 1]])
	Z = X * Y
	Z
		array([[2, 0],
		[0, 2]])
			Element-wise product of the array X and Y

43	C.T		Transposed of C

43	A = np.array([[0, 1, 1], [1, 0, 1]])
	B = np.array([[1, 1], [1, 1], [-1, 1]])
	Z = np.dot(A,B)
	Z
		array([[0, 2],
		[0, 2]])
			Matrix multiplication with the numpy arrays A and B