**What is Numpy**

NumPy is a module for Python. The name is an acronym for "Numeric Python" or "Numerical Python". It is an extension module for Python, mostly written in C. This makes sure that the precompiled mathematical and numerical functions and functionalities of Numpy guarantee great execution speed. Furthermore, NumPy enriches the programming language Python with powerful data structures, implementing multi-dimensional arrays and matrices. These data structures guarantee efficient calculations with matrices and arrays. The implementation is even aiming at huge matrices and arrays, better know under the heading of "big data". Besides that the module supplies a large library of high-level mathematical functions to operate on these matrices and arrays.

This module is basically based on Numpy array which is a powerful N-dimensional array object which is in the form of rows and columns. We can initialize numpy arrays from nested Python lists and access its element. NumPy is memory efficiency, meaning it can handle the vast amount of data more accessible than any other library. Besides, NumPy is very convenient to work with, especially for matrix multiplication and reshaping. On top of that, NumPy is fast. In fact, TensorFlow and Scikit learn to use NumPy array to compute the matrix multiplication in the back end.

**Numpy Functions:**

import numpy (as np)

This is used to include all functions or methods present in numpy into our program.

1. np.array(values) #one set of values will display as a row

This function is used to form an array by using some set of values which may be a list or tuple or set. For example

arr1 = np.array((1,2,3,4,5))

print(arr1)

Output will be [1,2,3,4,5]. We can store 'values' in some variable and then print it also.

2. np.arange(start, stop, step) #create range of elements in a row wise order

This function is also used to create an array. However this print values from start value to one less than of stop value. 'step' option is optional which denotes how much value to be left in a defined range. For example,

arr1 = np.arange(10,100,5)

print(arr1)

This means we want value between 10 to 99(one less than of 100) and want every 5th value i.e. 10, 15, 20 and so on upto 95.

3. np.zeros((row, column)) #create an array having all values equal to zero.

We can pass a single argument also, for example:

val1 = np.zeros(5)

print(val1)

This will print [0,0,0,0,0]. However if we mention 2 values then first will be considered as row value and second will be considered as column, i.e. np.zeros((4,5)) then this will print an array having 4 rows and 5 column in which all value will be equal to zero.

4. np.ones((row, column)) #create an array having all values equal to one

This is similar to zeros functions, however instead of zero it will print all values equal to 1. In this also we can pass a single argument also, for example:

val1 = np.ones(5)

print(val1)

This will print [1,1,1,1,1]. However if we mention 2 values then first will be considered as row value and second will be considered as column, i.e. np.ones((4,5)) then this will print an array having 4 rows and 5 column in which all value will be equal to 1.

5. np.linspace(start,stop,num=50)#it equally divides values between start to stop. total number of values will be equal to num.

Suppose you want to distributes values between 1 to 10 into 20 equal parts. Python provides a function named "linspace" for that. This will divides the values between start and stop into num parts. For example,

val1 = np.linspace(1,10, num=20)

print(val1)

This will print values between 1 to 10 in 20 equal parts.

6. np.eye(value)

This will print a square array of size "value" having diagonal value equal to 1 and rest will be zero.

7. np.random.function(V1,V2,....Vn) #print random values by using different "function".

e.g. print(np.random.rand(i,j)) #creates an 2D array of size iXj having random floating values between 0 to 1

print(np.random.randint(min\_value,max\_value,size=(i,j))) #creates an array of size iXj having random values between min\_value to max\_value

8. array\_name.reshape(row,column)

It change the occurrence of elements of given array\_name according to given arguements in the form of row and column. Suppose

arr1 = np.array([1,2,3,4,5,6])

arr2 = arr1.reshape(3,2)

print(arr2)

Output will be in the form of a 2D array having 3 rows and 2 columns. However one important point to be noted here is that reshape arguments should be given in such a way that the total number of elements of reshape array must be equal to the total number of elements in the original array. Like here (3,2) will contain 6 elements and arr1 is also containing 6 elements, so here without any error arr1 will reshaped.

NOTE: putting -1 in place of row or column means it automatically decides the number of rows and columns respectively.

9. i) array\_name.max()

It returns maximum value from the given array.

II) array\_name.min()

It return minimum value from the given array.

**Numpy Indexing and selection**

1. if arr1 is 1D array, then arr1[i] represents ith element of array arr1. For example, arr1[3] represents element at 3rd place of array arr1.

2. Similarly we can also apply slicing rule here, i.e. arr1[p:q] represents values from pth position to qth position of array arr1. For example, arr1[2:7] represents element from position 2 to 6.

3. We can even store slice of an array to a variable. For example,

variable\_name = arr1[2:7]

So wherever we print its value it always print elements from position from 2 to 6.

4. if arr1 is 2D array, then arr1[i][j] represents element at ith row and jth column. For example arr1[1][2] which means element at 1st row and 2nd column.

5. Range rule will be same in case of 2d array. only difference is that both row and column have separate ranges. for example arr1[1:3,0:2], this represents all elements from row 1 to 2 to column from zero to 1.

6. Selection of a portion of an array is used especially when we want to assign some another value to it. For this we can directly assign value to it. for example, let array be of size 4X5:

arr1[2,3]=Y #means value at 2nd row and 3rd column will be replaced by value Y

arr1[2:5,0:2]= Y # means each value present in range row from 2 to 4 and column from zero to 1 will be replaced by value Y.

**Numpy Operations**

1. arithmetic operations between two arrays. Let arr1 be an array, so

arr1 + arr1

arr1 \* arr1 and so on are all valid statements in python.

2. we can also apply \*\* operator between two arrays

arr2\*\*arr1, where arr1 and arr2 are two different arrays.

If you would like to cube the individual elements, or even higher up, use the 'power' function. Here, each element of the array is raised to the power 3.

np.power(array1,3)

3. np.sqrt(arr1), this will calculate square root of values of array arr1.

4. To find max and min value within an array, numpy provides two functions as following:

np.max(arr1) #to find max value present in array arr1

np.min(arr1) #to find min value present in array arr1

5. Numpy with conditional operators:

To match the values of a numpy array with a certain conditions we can use conditional operators.

array5 = array1 >= 30

array5

Output: array([False, False, True, True, True])

In order to print those values that are matched with our condition we can either send array5 to array function as an argument like

array1[array5]

this will print all those values which are true, or we can use following way:

array1[array1>=30]