**Numpy – My Notes**

* In Numpy an array is call as ‘array’ whereas in Python it is called as ‘List’
* In Numpy entire array elements can be accessible through fixed off-set from the beginning without performing looping operation.
* In Numpy array can be created in different ways, as following:
  + From Python List:
    - A1 = np.array( [1, 2, 3, 4, 5] )
  + A single item Python list to initialize an array of 10 times:
    - A1 = np.array( [0] \* 10 )
  + From Python ‘range’ function:
    - A1 = np.array( range( 10)
  + By using Numpy zeros():
    - A1 = np.zeros(10)
    - A1 = np.zeros(10, dtype = int)
  + By using Numpy ‘arange()’ function:
    - A1 = np.arange(0, 10) – endpoint – 1
    - A1 = np.arange(0, 10, 2) – where ‘2’ is the step
    - A1 = np.arange(10, 0, -1) – count down
  + By using Numpy ‘linspace()’ function
    - A1 = np.linspace( firstnumber, secondnumber, no. of elements in array)
    - A1 = np.linspace( 0, 5, 6)
  + Mathematical operations on Numpy Arrays called **‘vectorization’** i.e., scope is entire array:
    - * A1 = np.arange(0, 5)
      * A1 \* 2 - array( [0, 2, 4, 6, 8] )
    - Adding two arrays of same length:
      * A1 = np.arange(0, 5)
      * A2 = np.arange(5, 10)
      * A1 + A2
      * Array( [ 5, 7, 9, 11, 13] )
    - Subract two arrays of same length:
      * A1 = np.arange(0, 5)
      * A2 = np.arange(5, 10)
      * A1 - A2
    - Multipy two arrays of same length:
      * A1 = np.arange(0, 5)
      * A2 = np.arange(5, 10)
      * A1 \* A2
    - Divide two arrays of same length:
      * A1 = np.arange(0, 5)
      * A2 = np.arange(5, 10)
      * A1 / A2
    - Remainder two arrays of same length:
      * A1 = np.arange(0, 5)
      * A2 = np.arange(5, 10)
      * A1 % A2
  + **Type of Array**
    - A1 = np.arange(0, 5)
    - Type(a1) - numpy.ndarray
  + **How many elements**
    - A1 = np.arange(0, 5)
    - np.size(a1) - 5
  + **Data type of elements**
    - A1 = np.arange(0, 5)
    - A1.dtype - int32
    - A1 = np.arange(0, 5.0)
    - A1.dtype - float64
  + **Two-Dimensional Numpy Array**
    - np.array ( [ [1, 2 ], [3, 4 ] )
    - output => array( [ [1, 2], [3, 4] ] )
  + **Creating Two-Dimensional Numpy Array with ‘reshape()’ function**
    - M = np.arange(0, 20).reshape(5, 4) - reshape(no of rows, no of columns)
    - Np.size(m) – 20
    - Np.size(m, 0) – 5 => finding no. of rows
    - Np.size(m, 1) – 4 => finding no. of columns
  + **Accessig One-Dimensional Numpy Array Element**
    - A1 = np.arange(0, 5)
    - A1[0] => 0
    - A1[2] => 2
  + **Accessig Two-Dimensional Numpy Array Element**
    - A1 = np.arange(0, 9).reshape(3, 3)
    - A1[0] => throw entire first row i.e., array[0, 1, 2]
    - A1[0, 0] => accessing element at 0, 0
    - A1[0, 2] => accessing element at row 0 and element 3
    - A1[1, 1] => accessing element at row 2 and element 2
    - A1[1, ] => entire row
    - A1[ : , 0] => gives entire first column
  + **Logical Operations on Two-Dimensional Array**
    - a1 = np.arange(5)
    - a1 <= 2 => array( [true, true, true, false, false])
    - a1 = np.arange(5) (a1 < 2) | (a1==2) **=>** array([ True, True, True, False, False])
  + **Counting Elements Based on Logical Operation**
    - Np.sum(a < 3) where a = array
  + **Comparing One-Dimensional Arrays**
    - a1 = np.arange(0, 5)
    - a2 = np.arange(5, 0, -1)
    - print(a1)
    - print(a2)
    - a1 < a2 => [] – array ( [true, true, true, false, false] )
  + **Comparing Two-Dimensional Arrays**
    - a1 = np.arange(9).reshape(3, 3)
    - a2 = np.arange(9, 0, -1).reshape(3, 3)
    - print(a1)
    - print(a2)
    - print(a1 < a2)
      * [[0 1 2]
      * [3 4 5]
      * [6 7 8]]
      * [[9 8 7]
      * [6 5 4]
      * [3 2 1]]
      * [[ True True True]
      * [ True True False]
      * [False False False]]

**SLICING ARRAY**

* a1 = np.arange(1, 10)
* a1[3:8] => array([4, 5, 6, 7, 8])

**Slicing with step value:**

* a1 = np.arange(1, 20)
* a1[ : : 2]

**Revesring the Array:**

* a1 = np.arange(1, 20)
* a1[ : : -1]

**Slicing Two-Dimensional Array:**

* A0 = np.arange(9).reshape(3, 3)
* print(A0)
* A0[1:2, 1:2 ] - in order to pick **‘4’** in the matrix of 3x3

[[0 1 2]

[3 4 5]

[6 7 8]]

array([[4]])

* A0 = np.arange(9).reshape(3, 3)
* print(A0)
* A0[2:3, 2:3] - in order to pick ‘8’ in the above matrix
* A0 = np.arange(9).reshape(3, 3)
* print(A0)
* A0[0:2, 1:3] - in order to pick [ [1,2],

[4, 5] ]

* A0 = np.arange(9).reshape(3, 3)
* print(A0)
* A0[2:2+1, 1:1+1] => in order to pick ‘7’ from the above matrix. Note the ‘+’ sign.