Numpy

-np.array.dot

-np.array.ndim

-np.arange

-np.array.dtype

-np.array.astype(np.float32)

-np.array.astype(np.int32)

-np.array.shape

return tuple : if one-dimensional : return one single value within the bracket

-range(len(np.array))

For Loop

- np.ones((row, column), dtype = np.int)

-np.zeros((row,column))

-np.zeros((row,column)) + the specified number

-np.eye(number)

Identity matrix

-np.diag((val\_r0c0, val\_r1c1, val\_r2c2))

The rest of the values are zeros

-np.arange(start\_val, stop\_val)

stop\_val = stop – 1

default start\_val = 0

-np.arange(start\_val, stop\_val, step)

-np.linspace(start\_val, real\_stop, num = val)

Used to create a particular number of evenly spaced values in a specified half-open interval

np**.**linspace(6., 15., num**=**10)

>>array([ 6., 7., 8., 9., 10., 11., 12., 13., 14., 15.])

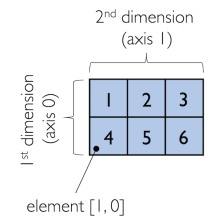
-np.array[start\_slice: end\_slice] # within the same row or the same column

-np.array[row, loc\_at\_that\_row]

-1 means the last element

0 means the first row

Comma “,” is used for slicing the arrays with more than one dimension



ary[:, 0] *# entire first column*

>>array([1, 4])

ary[:, :2] *# first two columns*

*>>*array([[1, 2],

[4, 5]])

-np.add(np.array, value)

-np.add.reduce(np.array, axis = 0) [or] np.array.sum(axis = 0)

-np.add.reduce(np.array, axis = 1) [or] np.array.sum(axis = 1)

ary **=** np**.**array([[1, 2, 3],

[4, 5, 6]]) *# rolling over the 1st axis, axis 0*

np**.**add**.**reduce(ary, axis**=**0)

>>array([5, 7, 9])

np**.**add**.**reduce(ary, axis**=**1) *# row sums*

>>array([ 6, 15])

-np.array.sum()

ary**.**sum()

>>21

-np.power(np.array, value)

-np.sqrt(np.array)

* np.mean (computes arithmetic mean or average)
* np.std (computes the standard deviation)
* np.var (computes variance)
* np.sort (sorts an array)
* np.argsort (returns indices that would sort an array)
* np.min (returns the minimum value of an array)
* np.max (returns the maximum value of an array)
* np.argmin (returns the index of the minimum value)
* np.argmax (returns the index of the maximum value)
* np.array\_equal (checks if two arrays have the same shape and elements)

-np.array.copy() #changes in a copied array does not affect the original array

-np.array > value #return an array including True and False