NumPy In Python

- NumPy stand for Numerical Python.
- It is a library which consist of multidimensional array objects.
- We usually create numpy with an alias called np
- NumPy is used to work with arrays and that array object in NumPy are called ndarray.
- We can create a NumPy ndarray object by using the array() function.

In [1]:
```
import numpy as np
```

In [2]:
```
arr = np.array([1, 2, 3, 4, 5, 6, 7 , 8])

print(arr)
```
```
[1 2 3 4 5 6 7 8]
```

In [3]:
```
type(arr)
```
Out[3]:
```
numpy.ndarray
```

In [4]:
```
arr.shape
```
Out[4]:
```
(8,)
```

In [5]:
```
#Multinested array

list1=[2, 3, 4, 5, 6]
list2=[4, 9, 16, 25, 36]
list3=[8, 127, 64, 125, 216]
```

In [6]:
```
arr=np.array([list1,list2,list3])
```

In [7]:
```
arr
```
Out[7]:
```
array([[  2,   3,   4,   5,   6],
       [  4,   9,  16,  25,  36],
       [  8, 127,  64, 125, 216]])
```

In [8]:
```
arr.shape
```
Out[8]:
```
(3, 5)
```

In [9]:
```
arr.reshape(5,3)  #3 rows and 5 columns
```
Out[9]:
```
array([[  2,   3,   4],
       [  5,   6,   4],
       [  9,  16,  25],
       [ 36,   8, 127],
       [ 64, 125, 216]])
```

In [10]:
```
print(np.__version__)
```
```
1.20.1
```

Dimensions in Arrays

1D array has one opening and one closing brackets

In [11]:
```
arr = np.array ([2,3,4,5])
print(arr)
```
```
[2 3 4 5]
```

In [12]:
```
arr[2]
```
Out[12]:
```
4
```

In [13]:
```
a=np.arange(6)
print(a)
```
```
[0 1 2 3 4 5]
```

2D array has two opening and two closing brackets

In [14]:
```
arr = np.array([[2 , 3 , 4 ,5],[4, 9, 16, 25]])
print(arr)
```
```
[[ 2  3  4  5]
 [ 4  9 16 25]]
```

In [15]:
```
arr.shape
```
Out[15]:
```
(2, 4)
```

In [16]:
```
arr.reshape(4,2)  #2 rows and 4 columns
```
Out[16]:
```
array([[ 2,  3],
       [ 4,  5],
       [ 4,  9],
       [16, 25]])
```

In [17]:
```
arr[0:,2:]
```
Out[17]:
```
array([[ 4,  5],
       [16, 25]])
```

In [18]:
```
a=np.arange(12)

print(a)
```
```
[ 0  1  2  3  4  5  6  7  8  9 10 11]
```

In [19]:
```
a.reshape(3,4)
```
Out[19]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [20]:
```
a.reshape(4,3)
```
Out[20]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

3D array has three opening and closing brackets

In [21]:
```
arr = np.array([[[2, 3, 4, 5], [4, 9, 16, 25],[8, 27, 64, 125]]])
print(arr)
```
```
[[[  2   3   4   5]
  [  4   9  16  25]
  [  8  27  64 125]]]
```

In [22]:
```
arr.shape
```
Out[22]:
```
(1, 3, 4)
```

In [23]:
```
arr.reshape(4,3)
```
Out[23]:
```
array([[  2,   3,   4],
       [  5,   4,   9],
       [ 16,  25,   8],
       [ 27,  64, 125]])
```

In [24]:
```
x=np.arange(24)

print(x)
```
```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
```

In [25]:
```
x.shape
```
Out[25]:
```
(24,)
```

In [26]:
```
x.reshape(2,3,4)
```
Out[26]:
```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

NumPy Arrays provides the ndim attribute that returns an integer value that tells us how many dimensions does our array have.

In [27]:
```
a = np.array(5)
b = np.array ([2,3,4,5])
c = np.array([[2 , 3 , 4 ,5],[4, 9, 16, 25]])
d = np.array([[[2, 3, 4, 5], [4, 9, 16, 25],[8, 27, 64, 125]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```
```
0
1
2
3
```

Slicing/Adding/Delete

In [28]:
```
a = [0]
np.append(a, [1,2])
```
Out[28]:
```
array([0, 1, 2])
```

In [29]:
```
x=[5,1,2]
np.insert(x,2, [4,6])
```
Out[29]:
```
array([5, 1, 4, 6, 2])
```

In [30]:
```
x=[3,4,6,7,8,9,1]
np.delete(x,-1)
```
Out[30]:
```
array([3, 4, 6, 7, 8, 9])
```

In [31]:
```
np.sort(x)
```
Out[31]:
```
array([1, 3, 4, 6, 7, 8, 9])
```

In [32]:
```
a = np.array([[1,2,3],[3,4,5],[4,5,6]])

print(a)
```
```
[[1 2 3]
 [3 4 5]
 [4 5 6]]
```

In [33]:
```
a[1,2]
```
Out[33]:
```
5
```

In [34]:
```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr[0])
```
```
[[1 2 3]
 [4 5 6]]
```

In [35]:
```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr)
```
```
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]
```

In [36]:
```
arr[0:]
```
Out[36]:
```
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[ 7,  8,  9],
        [10, 11, 12]]])
```

In [37]:
```
arr[0:,:1]
```
Out[37]:
```
array([[[1, 2, 3]],

       [[7, 8, 9]]])
```

In [38]:
```
arr[0:,:1,1]
```
Out[38]:
```
array([[2],
       [8]])
```

In [39]:
```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr[0, 1, 2])
```
```
6
```

In [40]:
```
a = np.array([[1,2,3],[3,4,5],[4,5,6]])

print(a)
```
```
[[1 2 3]
 [3 4 5]
 [4 5 6]]
```

In [41]:
```
a[1,2]
```
Out[41]:
```
5
```

In [42]:
```
val=5
```

In [43]:
```
arr
```
Out[43]:
```
array([[[ 1,  2,  3],
        [ 4,  5,  6]],

       [[ 7,  8,  9],
        [10, 11, 12]]])
```

In [44]:
```
arr/2
```
Out[44]:
```
array([[[0.5, 1. , 1.5],
        [2. , 2.5, 3. ]],

       [[3.5, 4. , 4.5],
        [5. , 5.5, 6. ]]])
```

In [45]:
```
arr<5
```
Out[45]:
```
array([[[ True,  True,  True],
        [ True, False, False]],

       [[False, False, False],
        [False, False, False]]])
```

In [46]:
```
arr[arr<5]
```
Out[46]:
```
array([1, 2, 3, 4])
```

In [47]:
```
arr1 = np.arange(15).reshape(3,5)
```

In [48]:
```
arr2 = np.arange(15).reshape(3,5)
```

In [49]:
```
arr1*arr2
```
Out[49]:
```
array([[  0,   1,   4,   9,  16],
       [ 25,  36,  49,  64,  81],
       [100, 121, 144, 169, 196]])
```

zeros and ones return a new array of a given shape and type, filled with ones.

In [50]:
```
np.zeros((3,3))
```
Out[50]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [51]:
```
#If you want to create an array with 1s:

np.ones((3,3))
```
Out[51]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

In [52]:
```
np.ones(16)  #by default data type is float)
```
Out[52]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [53]:
```
# here i am convert float data type to integer
np.ones(16,dtype=int)
```
Out[53]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

Random Distribution

In [54]:
```
np.random.rand(4,2) #By default numpy takes value between 0 to 1
                    # If i execute each and every time the elements get change because it randomly selected
```
Out[54]:
```
array([[0.90819763, 0.6268552 ],
       [0.52046806, 0.98305897],
       [0.89572335, 0.18885446],
       [0.83418194, 0.2605902 ]])
```

- Randint returns a random number given between the range

In [55]:
```
#Return random integers from low (inclusive) to high (exclusive).

#randint(low, high=None, size=None, dtype=int)
np.random.randint(2,50,10).reshape(2,5)
```
Out[55]:
```
array([[39, 15, 13, 35,  8],
       [31, 27, 26, 47, 38]])
```

Some Basic Calculation

In [56]:
```
x=np.array([[1,2,3],[4,2,6]])
x
```
Out[56]:
```
array([[1, 2, 3],
       [4, 2, 6]])
```

In [57]:
```
x.sum()
```
Out[57]:
```
18
```

In [58]:
```
#axis 0 -> column
#axis 1 -> row
x.sum(axis=0)
```
Out[58]:
```
array([5, 4, 9])
```

In [59]:
```
x.sum(axis=1)
```
Out[59]:
```
array([ 6, 12])
```

In [60]:
```
x.mean()
```
Out[60]:
```
3.0
```

In [61]:
```
x.std()
```
Out[61]:
```
1.632993161855452
```

In [62]:
```
x.max()
```
Out[62]:
```
6
```

In [63]:
```
arr.min()
```
Out[63]:
```
1
```

Argmax returns the index of maximum number

Argmin returns the index of minimum number

* Notes: if axis value is not given then numpy start index with 0

In [64]:
```
a= np.array([[21,100,3,4],[11,12,20,13],[56,21,34,45]])
a
```
Out[64]:
```
array([[ 21, 100,   3,   4],
       [ 11,  12,  20,  13],
       [ 56,  21,  34,  45]])
```

In [65]:
```
a.argmax()#argmax is 1 becuase maximum value of index 1 is 100 so our output is 1
```
Out[65]:
```
1
```

In [66]:
```
a.argmin() #argmin is 2 becuase minimum value of index 2 is 3 so our ind is 2
```
Out[66]:
```
2
```

* axis = 0 means that the operation is performed down the columns of a 2D array (Index start with 0)

In [67]:
```
np.argmax(a, axis= 0) #Here max no of index 2 in column 1 is 56,index 0 in column 2 is 100,index 2 in column 3 is 34 and so
```
Out[67]:
```
array([2, 0, 2, 2], dtype=int64)
```

In [68]:
```
np.argmin(a, axis= 0) #min val of index 2 in column 1 is 11 and so on
```
Out[68]:
```
array([1, 1, 0, 0], dtype=int64)
```

* axis = 1 means that the operation is performed down the rows of a 2D array (Index start with 0)

In [69]:
```
ar= np.array([[7,14,21,28],[9,18,27,62],[10,50,12,45]])
ar
```
Out[69]:
```
array([[ 7, 14, 21, 28],
       [ 9, 18, 27, 62],
       [10, 50, 12, 45]])
```

In [70]:
```
np.argmax(ar, axis= 1)   # 3-> 28, 3->62, 1->50
```
Out[70]:
```
array([3, 3, 1], dtype=int64)
```

In [71]:
```
np.argmin(ar, axis= 1)
```
Out[71]:
```
array([0, 0, 0], dtype=int64)
```

In [72]:
```
#Logical operation
np.all([False,True,True,True])
```
Out[72]:
```
False
```

In [76]:
```
a = np.zeros((1000,1000))
np.any(a!=0)
```
Out[76]:
```
False
```

In [80]:
```
np.all(a==0)
```
Out[80]:
```
True
```

In [74]:
```
np.any([True,False,False,True])
```
Out[74]:
```
True
```

In [ ]:

In [ ]: