Numpy Tutorial

Learn Numpy

Learn from following :

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

Conversion of Arrays from existing python data type

```
In [2]: myarr = np.array([1,2,3,4],np.int32)
```

```
In [3]: myarr[0]
```

```
Out[3]: 1
```

```
In [4]: two_dim_arr = np.array([[1,2,3],[4,8,9]],np.int32)
```

```
In [5]: two_dim_arr[1][1]
```

```
Out[5]: 8
```

```
In [6]: two_dim_arr.shape
```

```
Out[6]: (2, 3)
```

```
In [7]: two_dim_arr.dtype
```

```
Out[7]: dtype('int32')
```

```
In [8]: two_dim_arr.size
```

```
Out[8]: 6
```

```
In [9]: arr1 = np.array({1,2,3})
```

```
In [10]: arr1.dtype
```

```
Out[10]: dtype('O')
```

Intrinsic Numpy Array Creation Objects

```
In [11]: zeros = np.zeros((2,5))
```

```
In [12]: zeros
```

```
Out[12]: array([[0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0.]])
```

```
In [13]: zeros.dtype
```

```
Out[13]: dtype('float64')
```

```
In [14]: zeros.shape
```

```
Out[14]: (2, 5)
```

```
In [15]: rng = np.arange(15)
```

```
In [16]: rng
```

```
Out[16]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

```
In [17]: ## Equally spaced numbers between two numbers
         lspace = np.linspace(1,4,5)
```

```
In [18]: lspace
```

```
Out[18]: array([1.  , 1.75, 2.5 , 3.25, 4.  ])
```

```
In [19]: emp = np.empty((4,6))
```

```
In [20]: emp
```

```
Out[20]: array([[6.23042070e-307, 4.67296746e-307, 1.69121096e-306,
                 2.22521510e-306, 8.34448957e-308, 1.02360867e-306],
                [7.56602524e-307, 1.42419938e-306, 7.56603881e-307,
                 8.45603440e-307, 3.56043054e-307, 1.60219306e-306],
                [6.23059726e-307, 1.06811422e-306, 3.56043054e-307,
                 1.37961641e-306, 9.45697982e-308, 8.01097889e-307],
                [1.78020169e-306, 7.56601165e-307, 1.02359984e-306,
                 1.33510679e-306, 2.22522597e-306, 2.05837121e-312]])
```

```
In [21]: # To generate a numpy array from an existing numpy array
         emp_like = np.empty_like(lspace)
```

```
In [22]: emp_like
```

```
Out[22]: array([1.  , 1.75, 2.5 , 3.25, 4.  ])
```

```
In [23]: ide = np.identity(45)
```

```
In [24]: ide
```

```
Out[24]: array([[1., 0., 0., ..., 0., 0., 0.],
                [0., 1., 0., ..., 0., 0., 0.],
                [0., 0., 1., ..., 0., 0., 0.],
                ...,
                [0., 0., 0., ..., 1., 0., 0.],
                [0., 0., 0., ..., 0., 1., 0.],
                [0., 0., 0., ..., 0., 0., 1.]])
```

In [25]:
```
ide.shape
#ide.dtype
```

Out[25]:
```
(45, 45)
```

In [26]:
```
np_arr = np.arange(99)
```

In [27]:
```
np_arr
```

Out[27]:
```
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, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98])
```

In [28]:
```
np_arr.reshape(3,33)
```

Out[28]:
```
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, 24, 25, 26, 27, 28, 29, 30, 31,
        32],
       [33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48,
        49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64,
        65],
       [66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
        82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
        98]])
```

In [29]:
```
# To convert back from 2d to 1d numpy array
np_arr.ravel()
```

Out[29]:
```
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, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98])
```

In [30]:
```
np_arr.shape
```

Out[30]:
```
(99,)
```

In [31]:
```
x=[[1,2,3],[4,5,6],[7,8,9]]
```

In [32]:
```
arr = np.array(x)
```

In [33]:
```
arr.shape
```

Out[33]:
```
(3, 3)
```

In [34]:
```
arr.sum(axis = 0)
```

Out[34]:
```
array([12, 15, 18])
```

In [35]:
```
arr.sum(axis = 1)
```

Out[35]:
```
array([ 6, 15, 24])
```

In [36]:
```
arr
```

Out[36]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [37]:
```
# TO transpose a numpy array
arr.T
```

Out[37]:
```
array([[1, 4, 7],
       [2, 5, 8],
       [3, 6, 9]])
```

In [38]:
```
arr
```

Out[38]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [39]:
```
# TO iterate over a numpy array
for item in arr.flat:
    print(item)
```

```
1
2
3
4
5
6
7
8
9
```

In [40]:
```
#Number of Dimensions
arr.ndim
```

Out[40]:
```
2
```

In [41]:
```
#Number of elements in array
arr.size
```

Out[41]:
```
9
```

In [42]:
```
# Total bytes consumed
arr.nbytes
```

Out[42]:
```
36
```

In [43]:
```
one = np.array([1,3,4,634,2])
```

In [44]:
```
# To find index where max element exists
one.argmax()
```

Out[44]:
```
3
```

In [45]:
```
# To find index where min element exists
one.argmin()
```

```
Out[45]: 0

In [46]: # To find indices from any array such tha it gets sorted
         one.argsort()

Out[46]: array([0, 4, 1, 2, 3], dtype=int64)

In [47]: arr.argmin()

Out[47]: 0

In [48]: arr.argmax()

Out[48]: 8

In [49]: arr.argsort()

Out[49]: array([[0, 1, 2],
                [0, 1, 2],
                [0, 1, 2]], dtype=int64)

In [50]: arr.argmin(axis=0)

Out[50]: array([0, 0, 0], dtype=int64)

In [51]: arr.argmin(axis=1)

Out[51]: array([0, 0, 0], dtype=int64)

In [52]: arr.argsort(axis=0)

Out[52]: array([[0, 0, 0],
                [1, 1, 1],
                [2, 2, 2]], dtype=int64)

In [53]: arr2 = np.array([[0, 1, 2],
                [0, 1, 2],
                [0, 1, 2]])

In [54]: arr2

Out[54]: array([[0, 1, 2],
                [0, 1, 2],
                [0, 1, 2]])

In [55]: arr + arr2

Out[55]: array([[ 1,  3,  5],
                [ 4,  6,  8],
                [ 7,  9, 11]])

In [56]: arr

Out[56]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])
```

```
In [57]: arr * arr2

Out[57]: array([[ 0,  2,  6],
                [ 0,  5, 12],
                [ 0,  8, 18]])

In [58]: arr.min()

Out[58]: 1

In [59]: arr.max()

Out[59]: 9

In [60]: arr.sort()

In [61]: arr

Out[61]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])

In [62]: arr.sum()

Out[62]: 45

In [63]: # Gives a set of 2 tuples with indices for axis 0 and axis 1 respectively
         np.where(arr>5)

Out[63]: (array([1, 2, 2, 2], dtype=int64), array([2, 0, 1, 2], dtype=int64))
```

np.count_nonzero(arr)

```
In [64]: np.nonzero(arr)

Out[64]: (array([0, 0, 0, 1, 1, 1, 2, 2, 2], dtype=int64),
          array([0, 1, 2, 0, 1, 2, 0, 1, 2], dtype=int64))

In [65]: import sys

In [66]: py_ar = [0,4,55,2]

In [67]: np_ar = np.array(py_ar)

In [68]: sys.getsizeof(py_ar[0])*len(py_ar)

Out[68]: 112

In [69]: np_ar.nbytes

Out[69]: 16

In [70]: np_ar.itemsize*np_ar.size

Out[70]: 16

In [71]: np_ar.tolist()
```

```
Out[71]: [0, 4, 55, 2]

In [72]: np_ar.nbytes

Out[72]: 16

In [73]: np_ar.sort()

In [74]: np_ar

Out[74]: array([ 0,  2,  4, 55])

In [ ]:
```