NumPy (Numerical Python)

-By Athresh Kumar Labde

-> Numpy is written in C++ so it is faster than python. -> We use numpy so that the execution time will alter, let's take an example where we will print million strings using normal python and numpy we can clearly see the execution time with the help of (%time) or (%timeit) magic function.

Basic functions

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

In [2]:
```
array1 = np.array([2,3,5,6,11,2])
print(array1)
print(type(array1))
```

```
[ 2  3  5  6 11  2]
<class 'numpy.ndarray'>
```

In [3]:
```
array1.shape
```

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

In [4]:
```
array1.dtype
```

Out[4]:
```
dtype('int32')
```

In [5]:
```
np.zeros(5)
```

Out[5]:
```
array([0., 0., 0., 0., 0.])
```

In [6]:
```
np.zeros((4,5))
```

Out[6]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

In [7]:
```
np.ones(5)
```

Out[7]:
```
array([1., 1., 1., 1., 1.])
```

In [8]:
```
array1
```

Out[8]:
```
array([ 2,  3,  5,  6, 11,  2])
```

In [9]:
```
array1*array1
```

Out[9]:
```
array([  4,   9,  25,  36, 121,   4])
```

In [10]:
```
array1+array1
```

Out[10]:
```
array([ 4,  6, 10, 12, 22,  4])
```

In [11]:
```
array1-array1
```

Out[11]:
```
array([0, 0, 0, 0, 0, 0])
```

In [12]:
```
array1/array1
```

Out[12]:
```
array([1., 1., 1., 1., 1., 1.])
```

In [13]:
```
1/array1
```

Out[13]:
```
array([0.5       , 0.33333333, 0.2       , 0.16666667, 0.09090909,
       0.5       ])
```

In [14]:
```
2/array1
```

Out[14]:
```
array([1.        , 0.66666667, 0.4       , 0.33333333, 0.18181818,
       1.        ])
```

Slicing in NumPy

In [15]:
```
array1
```

Out[15]:
```
array([ 2,  3,  5,  6, 11,  2])
```

In [16]:
```
arr = array1[2:4]
```

In [17]:
```
arr[0] = 9
```

In [18]:
```
array1
```

Out[18]:
```
array([ 2,  3,  9,  6, 11,  2])
```

NumPy Functions

In [19]:
```
ar1 = np.array([[1,21,32,5],
                [4,2,4,6]])
```

In [20]:
```
ar1.sum(axis=0)
```

Out[20]:
```
array([ 5, 23, 36, 11])
```

In [21]:
```
ar1.sum(axis=1)
```

Out[21]:
```
array([59, 16])
```

In [22]:
```
ar1
```

Out[22]:
```
array([[ 1, 21, 32,  5],
       [ 4,  2,  4,  6]])
```

In [23]:
```
np.sort(ar1)
```

Out[23]:
```
array([[ 1,  5, 21, 32],
       [ 2,  4,  4,  6]])
```

In [24]:
```
np.sort(ar1, axis=0)
```

Out[24]:
```
array([[ 1,  2,  4,  5],
       [ 4, 21, 32,  6]])
```

In [26]:
```
%%timeit
np.sort(ar1, axis=0)
```

```
4.46 µs ± 234 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

In [25]:
```
%%timeit
np.sort(ar1, axis=0, kind='mergesort')
```

```
6.15 µs ± 266 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

In [27]:
```
%%timeit
np.sort(ar1, axis=0, kind='quicksort')
```

```
4.47 µs ± 261 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

More on NumPy

In [29]:
```
a = np.arange(7)
```

In [30]:
```
a
```

Out[30]:
```
array([0, 1, 2, 3, 4, 5, 6])
```

In [34]:
```
b = np.array([1,2,3,1,12,3,1,2,3,1,2,3,1,2,2,2,21,23])
b = b.reshape(6,3)
```

In [35]:
```
b
```

Out[35]:
```
array([[ 1,  2,  3],
       [ 1, 12,  3],
       [ 1,  2,  3],
       [ 1,  2,  3],
       [ 1,  2,  2],
       [ 2, 21, 23]])
```

In [39]:
```
b = b.reshape(9,2)
b
```

Out[39]:
```
array([[ 1,  2],
       [ 3,  1],
       [12,  3],
       [ 1,  2],
       [ 3,  1],
       [ 2,  3],
       [ 1,  2],
       [ 2,  2],
       [21, 23]])
```

In [40]:
```
c = np.array([1,2,33,12,12,36,11,22,13,1,3,3,1,2,2,2,21,23])
```

In [41]:
```
c
```

Out[41]:
```
array([ 1,  2, 33, 12, 12, 36, 11, 22, 13,  1,  3,  3,  1,  2,  2,  2, 21,
       23])
```

In [42]:
```
np.argsort(c)
```

Out[42]:
```
array([ 0, 12,  9,  1, 15, 14, 13, 10, 11,  6,  3,  4,  8, 16,  7, 17,  2,
       5], dtype=int64)
```

In [43]:
```
np.argmin(c)
```

Out[43]:
```
0
```

In [44]:
```
np.argmax(c)
```

Out[44]:
```
5
```

In [49]:
```
b = np.array([1,2,3,1,12,3,1,2,3,1,2,3,1,2,2,2,21,23])
b = b.reshape(6,3)
b
```

Out[49]:
```
array([[ 1,  2,  3],
       [ 1, 12,  3],
       [ 1,  2,  3],
       [ 1,  2,  3],
       [ 1,  2,  2],
       [ 2, 21, 23]])
```

In [50]:
```
np.argsort(b,axis=0)
```

Out[50]:
```
array([[0, 0, 4],
       [1, 2, 0],
       [2, 3, 1],
       [3, 4, 2],
       [4, 1, 3],
       [5, 5, 5]], dtype=int64)
```

In [51]:
```
np.argsort(b,axis=1)
```

Out[51]:
```
array([[0, 1, 2],
       [0, 2, 1],
       [0, 1, 2],
       [0, 1, 2],
       [0, 1, 2],
       [0, 1, 2]], dtype=int64)
```

In [52]:
```
np.argmin(b)
```

Out[52]:
```
0
```

In [53]:
```
np.argmax(b)
```

Out[53]:
```
17
```

In [ ]:
```

```