NUMPY PART - 3 BY VIRAT TIWARI

October 17, 2023

SORT , SEARCH & COUNTING FUNCTIONS IN NUMPY -

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

[13]:
```
arr=np.array([4,5,8,9,3,7,8,99,1,3,145,100,176,15,35,46,20,53,40])
```

[14]:
```
arr
```

[14]:
```
array([  4,   5,   8,   9,   3,   7,   8,  99,   1,   3, 145, 100, 176,
        15,  35,  46,  20,  53,  40])
```

[15]:
```
# sort ( ) function arrange the array in ascending order

np.sort(arr)
```

[15]:
```
array([  1,   3,   3,   4,   5,   7,   8,   8,   9,  15,  20,  35,  40,
        46,  53,  99, 100, 145, 176])
```

[17]:
```
# .searchsorted ( ) function gives the index value whwre our interger is going␣
 ↪to be placed

# Here we take the value 38 that placed on 17 index and that index value is␣
 ↪given by searchsorted function

np.searchsorted(arr,38)
```

[17]:
```
17
```

[18]:
```
arr1=np.array([0,56,485,975,315,2031,97,0,25,0])
```

[19]:
```
# count_nonzero ( ) function gives the total numbers that are non zero

np.count_nonzero(arr1)
```

[19]:
```
7
```

[20]:
```
arr
```

```
[20]: array([  4,   5,   8,   9,   3,   7,   8,  99,   1,   3, 145, 100, 176,
              15,  35,  46,  20,  53,  40])
```

```
[22]: # where( ) function give the index on which data is available greater than␣
      ↪define value

      np.where(arr>50)
```

```
[22]: (array([ 7, 10, 11, 12, 17]),)
```

```
[23]: # This is we exctract the data on behalf the passing data

      np.extract(arr>5,arr)
```

```
[23]: array([  8,   9,   7,   8,  99, 145, 100, 176,  15,  35,  46,  20,  53,
              40])
```

BYTE SWAPPING IN NUMPY - IT REPRESENT THE DATA INTO THE BYTES

```
[24]: arr
```

```
[24]: array([  4,   5,   8,   9,   3,   7,   8,  99,   1,   3, 145, 100, 176,
              15,  35,  46,  20,  53,  40])
```

```
[25]: # byteswap ( ) function every value of array into the bytes and present it in␣
      ↪bytes

      arr.byteswap()
```

```
[25]: array([  288230376151711744,   360287970189639680,   576460752303423488,
               648518346341351424,   216172782113783808,   504403158265495552,
               576460752303423488,  7133701809754865664,    72057594037927936,
               216172782113783808, -7998392938210000896,  7205759403792793600,
             -5764607523034234880,  1080863910568919040,  2522015791327477760,
              3314649325744685056,  1441151880758558720,  3819052484010180608,
              2882303761517117440])
```

COPIES AND VIEWS IN NUMPY -

```
[26]: arr
```

```
[26]: array([  4,   5,   8,   9,   3,   7,   8,  99,   1,   3, 145, 100, 176,
              15,  35,  46,  20,  53,  40])
```

```
[27]: # In case of copies it will create another copy with the change

      a=np.copy(arr)
```

```
[28]: # In case of view , it will change the original array without creating the new copy

      b=arr.view()
```

```
[29]: b
```

```
[29]: array([  4,   5,   8,   9,   3,   7,   8,  99,   1,   3, 145, 100, 176,
              15,  35,  46,  20,  53,  40])
```

```
[30]: arr
```

```
[30]: array([  4,   5,   8,   9,   3,   7,   8,  99,   1,   3, 145, 100, 176,
              15,  35,  46,  20,  53,  40])
```

```
[32]: b[0]=67
```

```
[33]: b
```

```
[33]: array([ 67,   5,   8,   9,   3,   7,   8,  99,   1,   3, 145, 100, 176,
              15,  35,  46,  20,  53,  40])
```

MATRIX LIBRARY IN NUMPY - MATRIX IS NOTHING BUT A SUB CLASS OF ARRAY

```
[36]: import numpy.matlib as nm
```

```
[37]: nm.zeros(5)
```

```
[37]: matrix([[0., 0., 0., 0., 0.]])
```

```
[38]: nm.ones((3,4))
```

```
[38]: matrix([[1., 1., 1., 1.],
              [1., 1., 1., 1.],
              [1., 1., 1., 1.]])
```

```
[39]: nm.eye(5)
```

```
[39]: matrix([[1., 0., 0., 0., 0.],
              [0., 1., 0., 0., 0.],
              [0., 0., 1., 0., 0.],
              [0., 0., 0., 1., 0.],
              [0., 0., 0., 0., 1.]])
```

NOTE - MATRIX FUNCTIONS IS ALMOST SIMILAR TO THE ARRAY

MOST IMPORTANT TOPIC - LINEAR ALGEBRA IN NUMPY

[40]:
```
arr1=np.random.randint([[2,3],[4,5]])
```

[41]:
```
arr2=np.random.randint([[2,3],[4,5]])
```

[42]:
```
arr1
```

[42]:
```
array([[1, 2],
       [2, 2]])
```

[43]:
```
arr2
```

[43]:
```
array([[0, 1],
       [1, 2]])
```

[44]:
```
# Matrix Multiplication by using dot ( ) function

np.dot(arr1,arr2)
```

[44]:
```
array([[2, 5],
       [2, 6]])
```

[45]:
```
# Matrix Multiplication by using @ ( ) function


arr1@arr2
```

[45]:
```
array([[2, 5],
       [2, 6]])
```

THANK YOU SO MUCH !!

YOURS VIRAT TIWARI :)