numpy

May 13, 2019

1 PYTHON-BRUSH UP !!

2 Numpy Arrays

```
In [ ]: # numpy contains objects representation of integer, floating point
        # containers:list,dictioners - internally built..can perform mathematical computation
```

```
In [4]: #it provides
        #extension package to python for mul dim array
        #[    ] 1 dim vector or array 2 dim matrix[], 3d cube where cells are in a cube and da
        #efficiency, mul dim array
        # array oriented computing
```

```
In [1]: import numpy as np
        a=np.array([0,1,2,3])  # list of elements
        print(a)
```

```
[0 1 2 3]
[0 1 2 3 4 5 6 7 8 9]
```

```
In [2]: # calculating square for the list of 1000 nos   GENERAL LIST
        L= range (1000)
        %timeit [i **2 for  i in L]
```

```
810 ţs ś 3.24 ţs per loop (mean ś std. dev. of 7 runs, 1000 loops each)
```

```
In [3]: a = np.arange(1000)   # squaring the number using numpy with lesser time
        %timeit a**2
```

```
3.39 ţs ś 204 ns per loop (mean ś std. dev. of 7 runs, 100000 loops each)
```

```
# numpy array will be created by giving  alist of numbers (1 dim - vector)
```

```
In [4]: b = np.array([1,2,3,4,50])

        b.ndim  # dimension 1

        b.shape  # 5 elements

Out[4]: (5,)
```

3 2 dim array

```
In [5]: dim = np.array([ [1,2,3] , [3,4,5], [2,3,4] ])  # each list take it as a row [1,2,3] [
        dim

Out[5]: array([[1, 2, 3],
               [3, 4, 5],
               [2, 3, 4]])

In [6]: dim.ndim

Out[6]: 2

In [7]: dim.shape  # row x column format (m x n)

Out[7]: (3, 3)

In [8]: len(dim)  # number of rows  or size of 1st dim


        # 3 dim array  list---list---list

        c=np.array( [ [ [1,2],[2,3] ], [ [0,0], [1,2] ] ] )

        c

Out[8]: array([[[1, 2],
                [2, 3]],

               [[0, 0],
                [1, 2]]])
```

4 Numpy functions

```
In [10]: print(np.arange(10))  # creates a value between 0 & n-1


         print(np.arange(1,10,2))

[0 1 2 3 4 5 6 7 8 9]
[1 3 5 7 9]
```

5 notable functions

```
np.linespace(start, end, number of points)
np.ones()
np.random.rand(array size)
```

6 Numerical operations on Numpy

Elementwise operations

```
In [12]: a = np.array([1,2,3,4])

         a+5

Out[12]: array([6, 7, 8, 9])

In [13]: # Matrix Multiplication

         x = np.diag([1,2,3,4])

         print(x*x)

         print("--------------------")

         print(x.dot(x))
```

```
[[ 1  0  0  0]
 [ 0  4  0  0]
 [ 0  0  9  0]
 [ 0  0  0 16]]
--------------------
[[ 1  0  0  0]
 [ 0  4  0  0]
 [ 0  0  9  0]
 [ 0  0  0 16]]
```

7 statistics

```
In [15]: x = np.array([1,2,3,1])

         x.mean()

Out[15]: 1.75
```

Similarly we can find median , std values for teh array

```
In [ ]:
```