Numpy tutorials

Numpy is a general-purpose array-processing package.It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing Python

What is an array

An array is a data structure that stores values of same data type in python,this is the main difference between arrays and lists. while python lists can contain values corresponding to different data types, array in python can only contain values corresponding to same data type

In [ ]:

In [1]:
```
## initailly lets import numpy library
import numpy as np
```

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

In [3]:
```
type(arr)
```

Out[3]:
```
numpy.ndarray
```

In [4]:
```
arr
```

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

In [5]:
```
arr.shape
```

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

In [ ]:

In [ ]:

In [12]:
```
## multinested array
my_lst1=[1,2,3,4,5]
my_lst2=[2,3,4,5,6]
my_lst3=[9,7,6,8,9]

arr=np.array([my_lst1,my_lst2,my_lst3])
```

In [ ]:

In [10]:
```
arr
```

Out[10]:
```
array([[1, 2, 3, 4, 5],
       [2, 3, 4, 5, 6],
       [9, 7, 6, 8, 9]])
```

In [11]:
```
arr.shape  ## row and column find ##check the shape of the array
```

Out[11]:
```
(3, 5)
```

In [13]:
```
arr.reshape(5,3)
```

Out[13]:
```
array([[1, 2, 3],
       [4, 5, 2],
       [3, 4, 5],
       [6, 9, 7],
       [6, 8, 9]])
```

In [14]:
```
arr.reshape(1,15)
```

Out[14]:
```
array([[1, 2, 3, 4, 5, 2, 3, 4, 5, 6, 9, 7, 6, 8, 9]])
```

In [15]:
```
arr.shape
```

Out[15]:
```
(3, 5)
```

Indexing

In [19]:
```
## accessing the array elements
arr=np.array([1,2,3,4,5,6,7,8,9])
```

In [20]:
```
arr[3]   ## 0,1,2,3
```

Out[20]:
```
4
```

In [21]:
```
arr
```

Out[21]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [22]:
```
arr=np.array([my_lst1,my_lst2,my_lst3])
```

In [23]:
```
arr
```

Out[23]:
```
array([[1, 2, 3, 4, 5],
       [2, 3, 4, 5, 6],
       [9, 7, 6, 8, 9]])
```

In [24]:
```
arr[:,:]
```

Out[24]:
```
array([[1, 2, 3, 4, 5],
       [2, 3, 4, 5, 6],
       [9, 7, 6, 8, 9]])
```

In [25]:
```
arr[:,3]
```

Out[25]:
```
array([4, 5, 8])
```

In [26]:
```
arr[0:2,0:2]
```

Out[26]:
```
array([[1, 2],
       [2, 3]])
```

In [27]:
```
arr[1:3,2:4]
```

Out[27]:
```
array([[4, 5],
       [6, 8]])
```

In [28]:
```
arr=np.arange(0,10)
```

In [29]:
```
arr
```

Out[29]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [31]:
```
arr=np.arange(0,10,step=2)
```

In [32]:
```
arr
```

Out[32]:
```
array([0, 2, 4, 6, 8])
```

In [42]:
```
np.linspace(1,10,50) #equally interval number
```

Out[42]:
```
array([ 1.        ,  1.18367347,  1.36734694,  1.55102041,  1.73469388,
        1.91836735,  2.10204082,  2.28571429,  2.46938776,  2.65306122,
        2.83673469,  3.02040816,  3.20408163,  3.3877551 ,  3.57142857,
        3.75510204,  3.93877551,  4.12244898,  4.30612245,  4.48979592,
        4.67346939,  4.85714286,  5.04081633,  5.2244898 ,  5.40816327,
        5.59183673,  5.7755102 ,  5.95918367,  6.14285714,  6.32653061,
        6.51020408,  6.69387755,  6.87755102,  7.06122449,  7.24489796,
        7.42857143,  7.6122449 ,  7.79591837,  7.97959184,  8.16326531,
        8.34693878,  8.53061224,  8.71428571,  8.89795918,  9.08163265,
        9.26530612,  9.44897959,  9.63265306,  9.81632653, 10.        ])
```

In [50]:
```
arr=np.array(my_lst)
```

In [51]:
```
arr
```

Out[51]:
```
array([1, 2, 3, 4, 5])
```

In [56]:
```
## copy function and broadcasting
arr[3:]=100
```

In [57]:
```
arr
```

Out[57]:
```
array([  1,   2,   3, 100, 100])
```

In [58]:
```
arr1=arr
```

In [59]:
```
arr1[3:]=500
```

In [60]:
```
arr1
```

Out[60]:
```
array([  1,   2,   3, 500, 500])
```

In [61]:
```
arr
```

Out[61]:
```
array([  1,   2,   3, 500, 500])
```

In [62]:
```
arr1=arr.copy()
```

In [64]:
```
print(arr)
arr1[3:]=1000
print(arr1)
```

```
[  1   2   3 500 500]
[   1    2    3 1000 1000]
```

In [65]:
```
arr
```

Out[65]:
```
array([  1,   2,   3, 500, 500])
```

In [66]:
```
arr1
```

Out[66]:
```
array([   1,    2,    3, 1000, 1000])
```

In [67]:
```
### some conditions very useful in exloratory Data Analysis
val=2
arr<2
```

Out[67]:
```
array([ True, False, False, False, False])
```

In [68]:
```
arr*2
```

Out[68]:
```
array([   2,    4,    6, 1000, 1000])
```

In [70]:
```
arr[arr<3]
```

Out[70]:
```
array([1, 2])
```

In [72]:
```
## create arrays and reshape
np.arange(0,10).reshape(5,2)
```

Out[72]:
```
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

In [75]:
```
arr1=np.arange(0,10).reshape(2,5)
arr2=np.arange(0,10).reshape(2,5)
```

In [76]:
```
arr1*arr2
```

Out[76]:
```
array([[ 0,  1,  4,  9, 16],
       [25, 36, 49, 64, 81]])
```

In [77]:
```
np.ones(4)
```

Out[77]:
```
array([1., 1., 1., 1.])
```

In [84]:
```
np.ones((2,5),dtype=int)
```

Out[84]:
```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])
```

In [85]:
```
## random distribution
np.random.rand(3,3)
```

Out[85]:
```
array([[0.77221945, 0.3638006 , 0.92147395],
       [0.84130223, 0.33009227, 0.43531594],
       [0.06420557, 0.69372344, 0.17059882]])
```

In [86]:
```
arr_ex=np.random.randn(4,4)
```

In [87]:
```
arr_ex
```

Out[87]:
```
array([[ 0.56600224, -0.15289136,  0.50319489, -1.31846805],
       [ 0.17949899,  0.55372939,  0.54540858,  0.86074895],
       [ 0.2554157 ,  1.59367473, -0.7947132 , -1.0738529 ],
       [ 1.41062225, -1.02765942, -0.37963489, -0.08371565]])
```

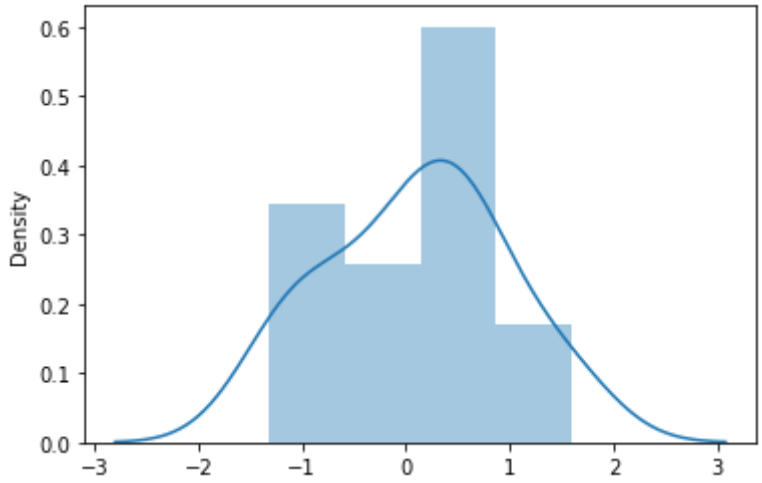
In [88]:
```
import seaborn as sns
import pandas as pd
```

```
Matplotlib is building the font cache; this may take a moment.
```

In [89]:
```
sns.distplot(pd.DataFrame(arr_ex.reshape(16,1)))
```

```
E:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[89]:
```
<AxesSubplot:ylabel='Density'>
```



In [90]:
```
np.random.randint(0,100,8).reshape(4,2)
```

Out[90]:
```
array([[60,  9],
       [52, 67],
       [25, 32],
       [40, 67]])
```

In [91]:
```
np.random.random_sample((1,5))
```

Out[91]:
```
array([[0.95482703, 0.33746571, 0.74155651, 0.53500702, 0.26426734]])
```

In [ ]: