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 with 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, arrays in python can only contain values corresponding to same data type

In [2]:
```
## initially Lets import numpy

import numpy as np
```

In [3]:
```
my_list=[1,2,3,4,5]

#Converting list into numpy array
a=np.array(my_list)
```

In [4]:
```
print(a)
```
```
[1 2 3 4 5]
```

In [5]:
```
type(a)
```
Out[5]:
```
numpy.ndarray
```

In [6]:
```
# Multinested array or lets say we are creating multidimensional array
# Trick: if our array contain one bracket then it is one dimensional array, if two then it is 2 -D and if 3 the
my_list1=[1,2,3,4,5]
my_list2=[2,3,4,5,6]
my_list3=[9,7,6,8,9]

b=np.array([my_list1,my_list2,my_list3])
```

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

In [8]:
```
type(b)
```
Out[8]:
```
numpy.ndarray
```

In [12]:
```
b.ndim   # here we are seeing dimension of b
```
Out[12]:
```
2
```

In [13]:
```
## check the shape of the array
b.shape
```
Out[13]:
```
(3, 5)
```

In [15]:
```
#we can reshape the b,
b.reshape(5,3)
```
Out[15]:
```
array([[1, 2, 3],
       [4, 5, 2],
       [3, 4, 5],
       [6, 9, 7],
       [6, 8, 9]])
```

Indexing or finding position through slicing

In [16]:
```
a
```
Out[16]:
```
array([1, 2, 3, 4, 5])
```

In [17]:
```
a[3]
```
Out[17]:
```
4
```

In [18]:
```
b
```

```
Out[18]: array([[1, 2, 3, 4, 5],
                [2, 3, 4, 5, 6],
                [9, 7, 6, 8, 9]])

In [ ]:  #suppose we need 5,6 and 8,9

In [19]: b[1:3,3:5]            #row 1 to 3(means upto 2);Column 3 to 5(means till 4)

Out[19]: array([[5, 6],
                [8, 9]])

In [21]: #Similarly, if we need 4,5,8 and 5,6,9
         b[0:3,3:5]

Out[21]: array([[4, 5],
                [5, 6],
                [8, 9]])

In [ ]:  # Replacing elements in array

In [22]: a

Out[22]: array([1, 2, 3, 4, 5])

In [23]: a[2]=100       #means needed 100 in 3rd place or 2nd indexing, instead of 3.

In [24]: a

Out[24]: array([  1,   2, 100,   4,   5])

In [25]: #we can change multiple elements with same number too
         a[2:]=100

In [26]: a

Out[26]: array([  1,   2, 100, 100, 100])

In [27]: ### Some conditions very useful in Exploratory Data Analysis (like in need number less than 3 from array)
         a[a<3]

Out[27]: array([1, 2])
```

Next method to create NumPy array: arange

```
In [29]: ## Create arrays and reshape
         np.arange(0,10).reshape(5,2)  #0 to 10 elements i.e. upto 9 and needed 5 rows and 2 columns.

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

In [30]: c=np.arange(0,15).reshape(3,5)     #another example of arange and reshape

In [31]: c

Out[31]: array([[ 0,  1,  2,  3,  4],
                [ 5,  6,  7,  8,  9],
                [10, 11, 12, 13, 14]])

In [39]: import random
         list2=[random.randint(1,30)for _ in range(15)]
         d=np.array(list2).reshape(3,5)

In [40]: d

Out[40]: array([[13,  3,  3, 30, 21],
                [26, 13, 27,  9, 16],
                [14,  7, 15,  5, 11]])

In [41]: c*d   # two array can be multiply so that NumPy is numerical python which is very useful in data science

Out[41]: array([[  0,   3,   6,  90,  84],
                [130,  78, 189,  72, 144],
                [140,  77, 180,  65, 154]])
```

Third method to create array - ones, all value will be one. We have already seen two methods np.array and np.arange. Simialrly as np.ones, there is zeros, linspace

```
In [42]: np.ones((2,5),dtype=int)          # two row and 5 column with data type integer, if we do not call datatype, def

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

In [43]:
```
## random distribution
np.random.rand(3,3)
```

Out[43]:
```
array([[0.76379753, 0.42651909, 0.58995456],
       [0.29077335, 0.03960679, 0.46631723],
       [0.52379082, 0.92313919, 0.23219766]])
```

In [47]:
```
df=np.linspace(2,5)         #default is 50 numbers
print(df)
```

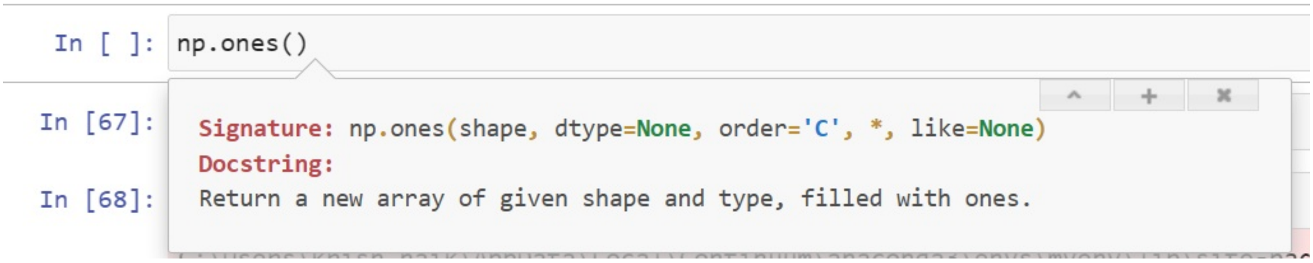
```
[2.         2.06122449 2.12244898 2.18367347 2.24489796 2.30612245
 2.36734694 2.42857143 2.48979592 2.55102041 2.6122449  2.67346939
 2.73469388 2.79591837 2.85714286 2.91836735 2.97959184 3.04081633
 3.10204082 3.16326531 3.2244898  3.28571429 3.34693878 3.40816327
 3.46938776 3.53061224 3.59183673 3.65306122 3.71428571 3.7755102
 3.83673469 3.89795918 3.95918367 4.02040816 4.08163265 4.14285714
 4.20408163 4.26530612 4.32653061 4.3877551  4.44897959 4.51020408
 4.57142857 4.63265306 4.69387755 4.75510204 4.81632653 4.87755102
 4.93877551 5.        ]
```

In [45]:
```
np.linspace(2,5,10)   #here by inserting 10, we keep default as 50
```

Out[45]:
```
array([2.        , 2.33333333, 2.66666667, 3.        , 3.33333333,
       3.66666667, 4.        , 4.33333333, 4.66666667, 5.        ])
```

NOTE: One most important: if we do not know what specefic function do, we can type and click shift+tab like np.linspace (CLICK SHIFT+TAB, it will show its function)

example:



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

In [67]:

In [68]:
```

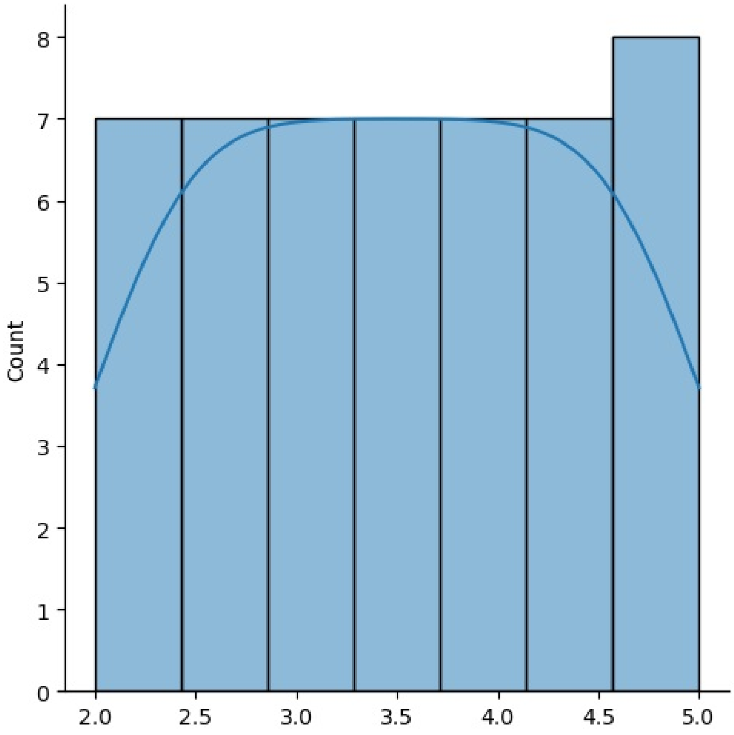
```
Signature: np.ones(shape, dtype=None, order='C', *, like=None)
Docstring:
Return a new array of given shape and type, filled with ones.
```

In [52]:
```
import seaborn as sns
import pandas as pd
```

In [54]:
```
sns.displot(df,kde=True)
```

```
C:\Users\USER\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed t
o tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[54]:
```
<seaborn.axisgrid.FacetGrid at 0x25418b954d0>
```



In [55]:
```
#Just wanted to show the picture, because we got output df from linspace and element of linspace is equally div
#it got graphical output as normal or gussian curve (Bell Shaped).
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js