In [1]:
```
##  Built-in Functions
```

In [11]:
```
import numpy as np

# To check the syntax and other parameters you can take help with the command

?np.min
```

In [13]:
```
# arange
#  Returns the evenly spaced values for the specified interval
# np.arange(start_value, upperbound, stepsize)
```

In [15]:
```
np.arange(0,20)
```

Out[15]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19])
```

In [16]:
```
np.arange(0,20,2)
```

Out[16]:
```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

In [17]:
```
## linspace   Returns evenly spaced numbers for a given interval.

np.linspace(0, 20, 4 )
```

Out[17]:
```
array([ 0.        ,  6.66666667, 13.33333333, 20.        ])
```

In [18]:
```
np.linspace(0, 20, 50)
```

Out[18]:
```
array([ 0.        ,  0.40816327,  0.81632653,  1.2244898 ,  1.63265306,
        2.04081633,  2.44897959,  2.85714286,  3.26530612,  3.67346939,
        4.08163265,  4.48979592,  4.89795918,  5.30612245,  5.71428571,
        6.12244898,  6.53061224,  6.93877551,  7.34693878,  7.75510204,
        8.16326531,  8.57142857,  8.97959184,  9.3877551 ,  9.79591837,
       10.20408163, 10.6122449 , 11.02040816, 11.42857143, 11.83673469,
       12.24489796, 12.65306122, 13.06122449, 13.46938776, 13.87755102,
       14.28571429, 14.69387755, 15.10204082, 15.51020408, 15.91836735,
       16.32653061, 16.73469388, 17.14285714, 17.55102041, 17.95918367,
       18.36734694, 18.7755102 , 19.18367347, 19.59183673, 20.        ])
```

In [19]:
```
##  Zeros and Ones
```

In [20]:
```
np.zeros(10)
```

Out[20]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [21]:
```
np.zeros((4,4))
```

Out[21]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [22]:
```
np.ones(10)
```

```
Out[22]: array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```
In [23]: np.ones((4,4))
```

```
Out[23]: array([[1., 1., 1., 1.],
                [1., 1., 1., 1.],
                [1., 1., 1., 1.],
                [1., 1., 1., 1.]])
```

```
In [24]: ##  Identity Matrix and Array
```

```
In [25]: np.eye(4)
```

```
Out[25]: array([[1., 0., 0., 0.],
                [0., 1., 0., 0.],
                [0., 0., 1., 0.],
                [0., 0., 0., 1.]])
```

```
In [26]: # Identity Matrix

         np.matrix(np.eye(4))
```

```
Out[26]: matrix([[1., 0., 0., 0.],
                 [0., 1., 0., 0.],
                 [0., 0., 1., 0.],
                 [0., 0., 0., 1.]])
```

```
In [28]: ###  Random function :- It is used to generate random numbers with uniform distributic

         ## In uniform distribution, the values are between 0 to 1 .

         np.random.rand(12)
```

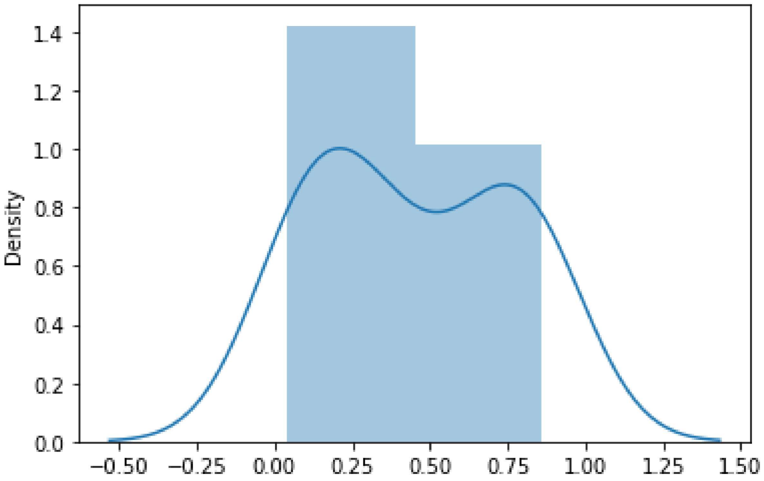
```
Out[28]: array([0.10445617, 0.27264585, 0.689318  , 0.50536033, 0.0255189 ,
                0.55105059, 0.43564862, 0.69809544, 0.76044088, 0.86578539,
                0.30771743, 0.87051317])
```

```
In [31]: # To visualize the uniform distribution

         import seaborn as sns
         %matplotlib inline
         sns.distplot(np.random.rand(12))
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7f57cfaf3450>
```



In [40]:

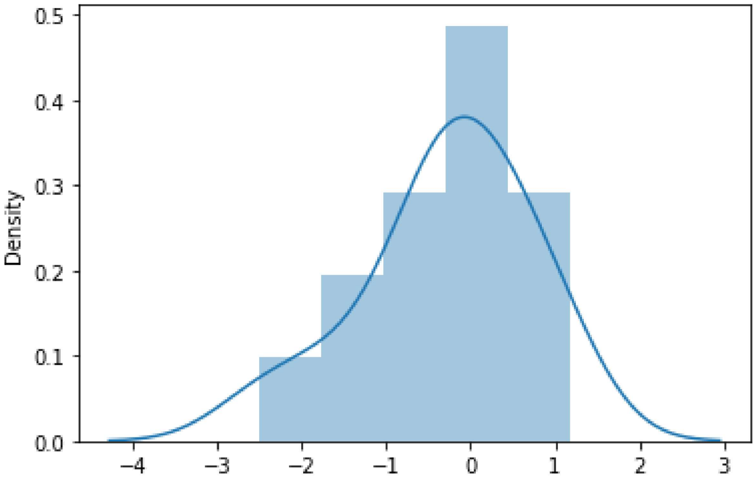
```
###  Normal  Distribution   Bell Curve

sns.distplot(np.random.randn(14))
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: 
`distplot` is a deprecated function and will be removed in a future version. Please a
dapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[40]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f57cf046f50>
```



In [42]:

```
###  How to generate random integers

np.random.randint(1, 100, 9)
```

Out[42]:

```
array([72, 21, 58, 91, 63, 60, 90, 79, 85])
```

In [43]:

```
#  Min Max Function
```

In [44]:

```
array = np.random.randint(1,100,9)
```

In [45]:

```
array
```

Out[45]:

```
array([31, 72,  8, 38, 94, 10, 82,  2, 93])
```

```
In [48]: # To find out the minimum number

         array.min()

Out[48]: 2
```

```
In [49]: # To find out the maximum number

         array.max()

Out[49]: 94
```

```
In [50]: # To find the index where minimum number is located

         array.argmin()

Out[50]: 7
```

```
In [51]: # To find the index where maximum number is located

         array.argmax()

Out[51]: 4
```

```
In [ ]:
```