CONTINOUS DISTRIBUTION

In [2]:

```
# for inline plots in jupyter
%matplotlib inline
# import matplotlib
import matplotlib.pyplot as plt
# for latex equations
from IPython.display import Math, Latex
# for displaying images
from IPython.core.display import Image
import numpy as np
```

In [3]:

```
# import seaborn
import seaborn as sns
# settings for seaborn plotting style
sns.set(color_codes=True)
# settings for seaborn plot sizes
sns.set(rc={'figure.figsize':(5,5)})
```

UNIFORM DISTRIBUTION

You can visualize uniform distribution in python with the help of a random number generator acting over an interval of numbers (a,b). You need to import the uniform function from scipy.stats module.

In [4]:

```
# import uniform distribution
from scipy.stats import uniform
```

The uniform function generates a uniform continuous variable between the specified interval via its loc and scale arguments. This distribution is constant between loc and loc + scale. The size arguments describe the number of random variates. If you want to maintain reproducibility, include a random_state argument assigned to a number.

In [5]:

```
# random numbers from uniform distribution
n = 10000
start = 10
width = 20
data_uniform = uniform.rvs(size=n, loc = start, scale=width)
```

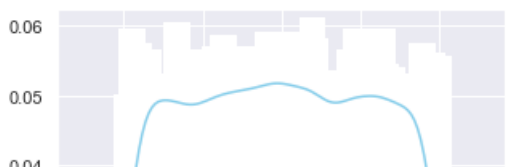
You can use Seaborn's distplot to plot the histogram of the distribution you just created. Seaborn's distplot takes in multiple arguments to customize the plot. You first create a plot object ax. Here, you can specify the number of bins in the histogram, specify the color of the histogram and specify density plot option with kde and linewidth option with hist_kws. You can also set labels for x and y axis using the xlabel and ylabel arguments.

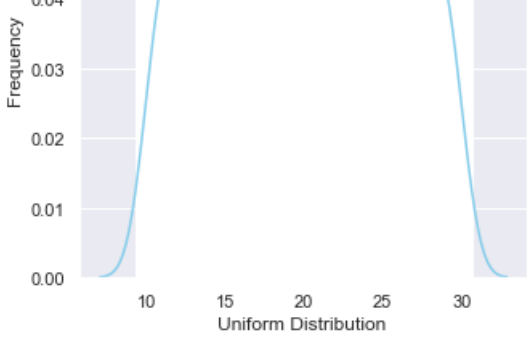
In [6]:

```
ax = sns.distplot(data_uniform,
                  bins=100,
                  kde=True,
                  color='skyblue',
                  hist_kws={"linewidth": 15,'alpha':1})
ax.set(xlabel='Uniform Distribution ', ylabel='Frequency')
```

Out[6]:

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Uniform Distribution ')]





In [ ]:

```
```

NORMAL DISTRIBUTION

Normal Distribution

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

In [7]:

```
from scipy.stats import norm
# generate random numbers from N(0,1)
data_normal = norm.rvs(size=10000,loc=0,scale=1)
```

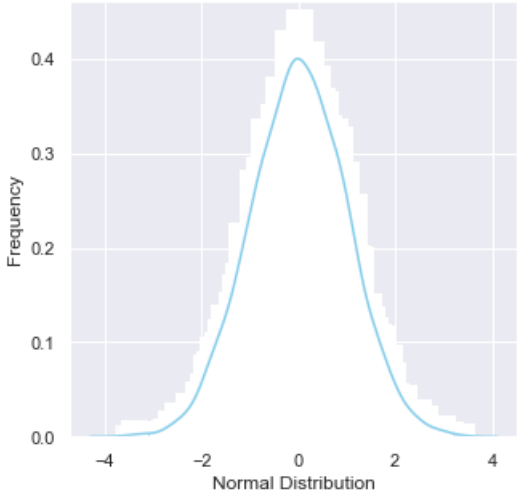
You can generate a normally distributed random variable using scipy.stats module's norm.rvs() method. The loc argument corresponds to the mean of the distribution. scale corresponds to standard deviation and size to the number of random variates. If you want to maintain reproducibility, include a random_state argument assigned to a number.

In [8]:

```
ax = sns.distplot(data_normal,
                  bins=100,
                  kde=True,
                  color='skyblue',
                  hist_kws={"linewidth": 15,'alpha':1})
ax.set(xlabel='Normal Distribution', ylabel='Frequency')
```

Out[8]:

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Normal Distribution')]



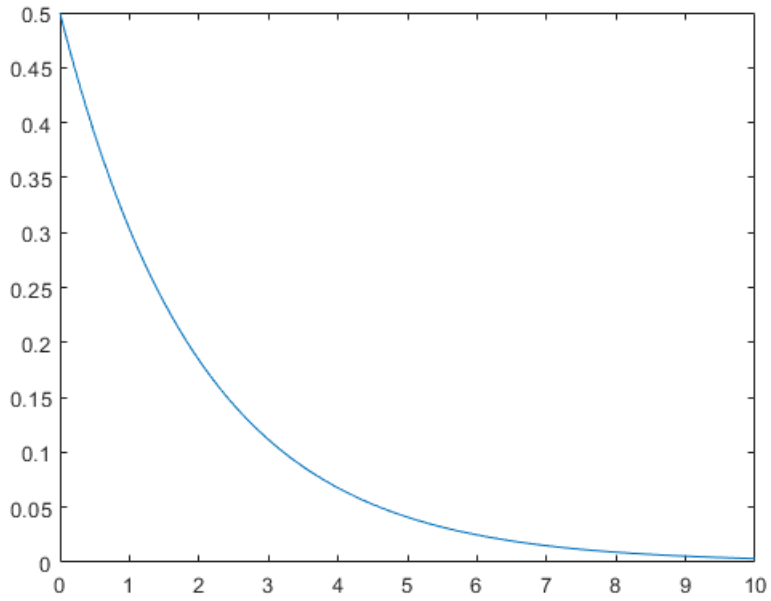
Exponential Distribution

The exponential distribution describes the time between events in a Poisson point process, i.e., a process in which events occur continuously and independently at a constant average rate. It has a parameter λ

called rate parameter, and its equation is described as :

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

A decreasing exponential distribution looks like :



In [9]:

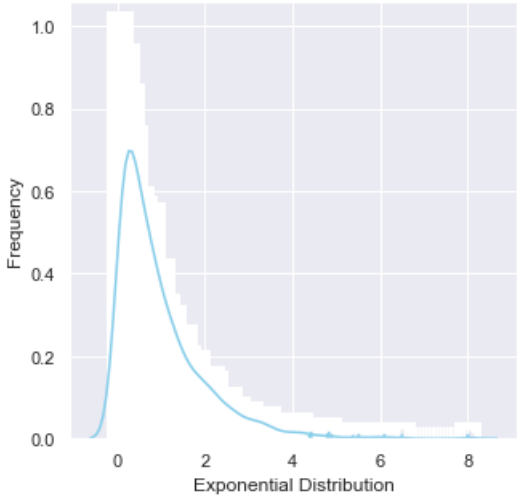
```
from scipy.stats import expon
data_expon = expon.rvs(scale=1,loc=0,size=1000)
```

In [10]:

```
ax = sns.distplot(data_expon,
                  kde=True,
                  bins=100,
                  color='skyblue',
                  hist_kws={"linewidth": 15,'alpha':1})
ax.set(xlabel='Exponential Distribution', ylabel='Frequency')
```

Out[10]:

[Text(0, 0.5, 'Frequency'), Text(0.5, 0, 'Exponential Distribution')]



Chi Square Distribution

Chi Square distribution is used as a basis to verify the hypothesis.

It has two parameters:

df - (degree of freedom).

size - The shape of the returned array.

Draw out a sample for chi squared distribution with degree of freedom 2 with size 2x3:

In [11]:

```
from numpy import random

x = random.chisquare(df=2, size=(2, 3))

print(x)
```

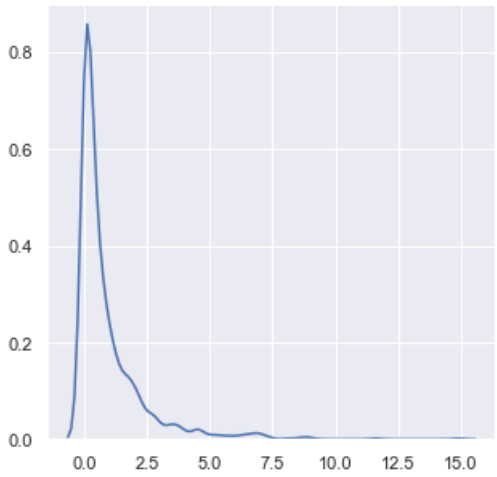
```
[[2.53587108 0.10908112 1.76364085]
 [4.62246558 1.91363218 0.92677457]]
```

In [12]:

```
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot(random.chisquare(df=1, size=1000), hist=False)

plt.show()
```



Weibull Distribution

In [13]:

```
a = 5. # shape

s = np.random.weibull(a, 1000)
```

In [14]:

```
#Display the histogram of the samples, along with the probability density function:
import matplotlib.pyplot as plt

x = np.arange(1,100.)/50.

def weib(x,n,a):

    return (a / n) * (x / n)**(a - 1) * np.exp(-(x / n)**a)
```

In [15]:

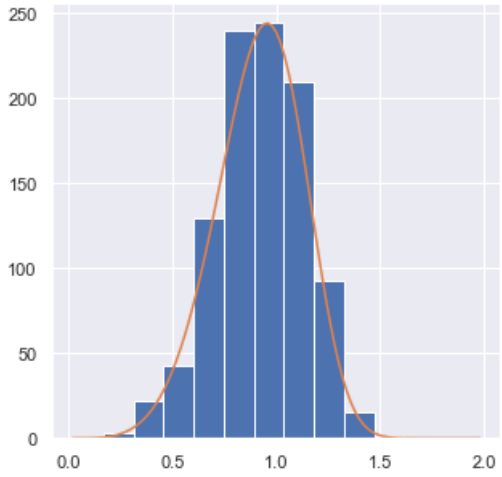
```
count, bins, ignored = plt.hist(np.random.weibull(5.,1000))

x = np.arange(1,100.)/50.

scale = count.max()/weib(x, 1., 5.).max()

plt.plot(x, weib(x, 1., 5.)*scale)

plt.show()
```



In [ ]:

```
```