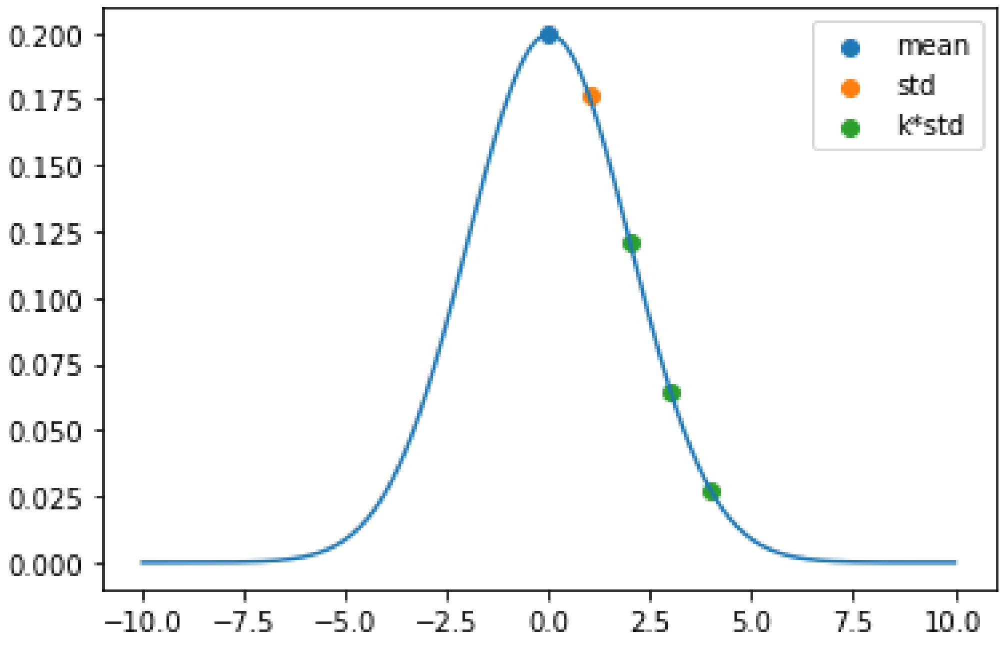
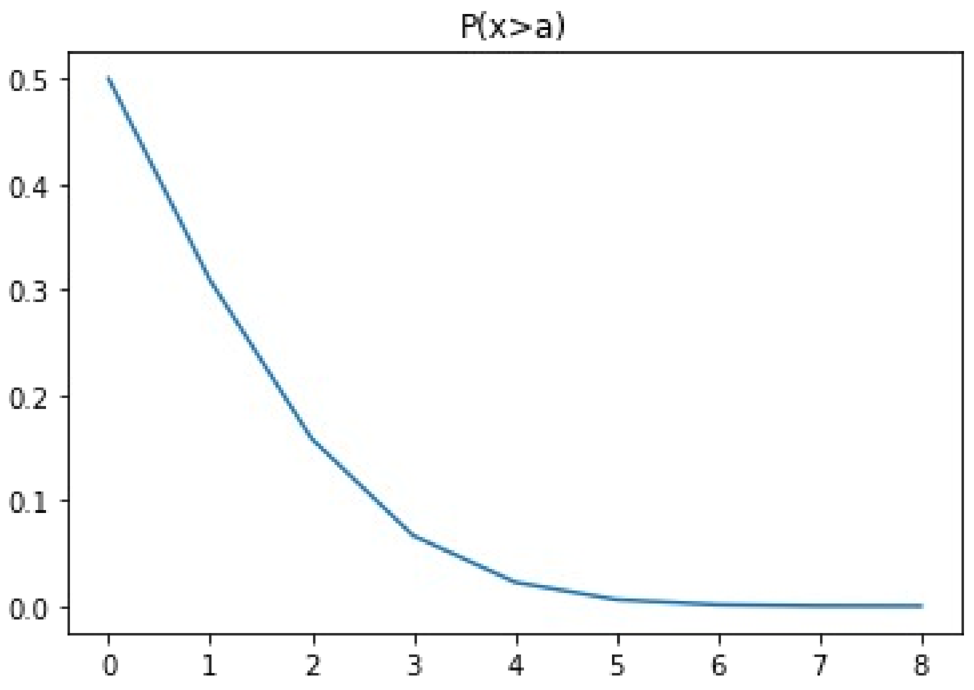
```
from scipy.stats import poisson
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import expon
```

Gaussian distribution

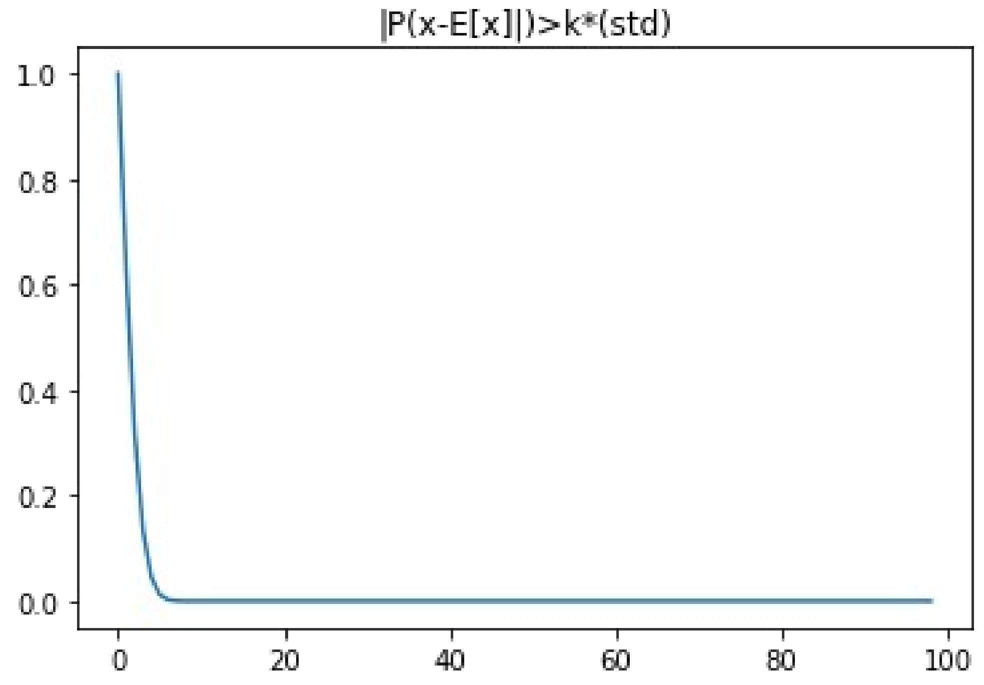
```
x=np.linspace(-10,10,1000)
y=norm.pdf(x,loc=0,scale=2)
y1=norm.mean(loc=0,scale=2)
plt.plot(x,y)
x=0
x1=1
x2=[2*x1,3*x1,4*x1]
plt.scatter(x,norm.pdf(x,loc=0,scale=2),label='mean')
plt.scatter(x1,norm.pdf(x1,loc=0,scale=2),label='std')
plt.scatter(x2,
[norm.pdf(2*x1,loc=0,scale=2),norm.pdf(3*x1,loc=0,scale=2),norm.pdf(4*
x1,loc=0,scale=2)],label='k*std')
#plt.plot(x,y)
plt.legend()
plt.show()
```



```
for i in range(10):
    x=np.arange(0,i,1)
    y=norm.sf(x, loc=0, scale=2)
plt.plot(x,y)
plt.title('P(x>a)')
plt.show()
```

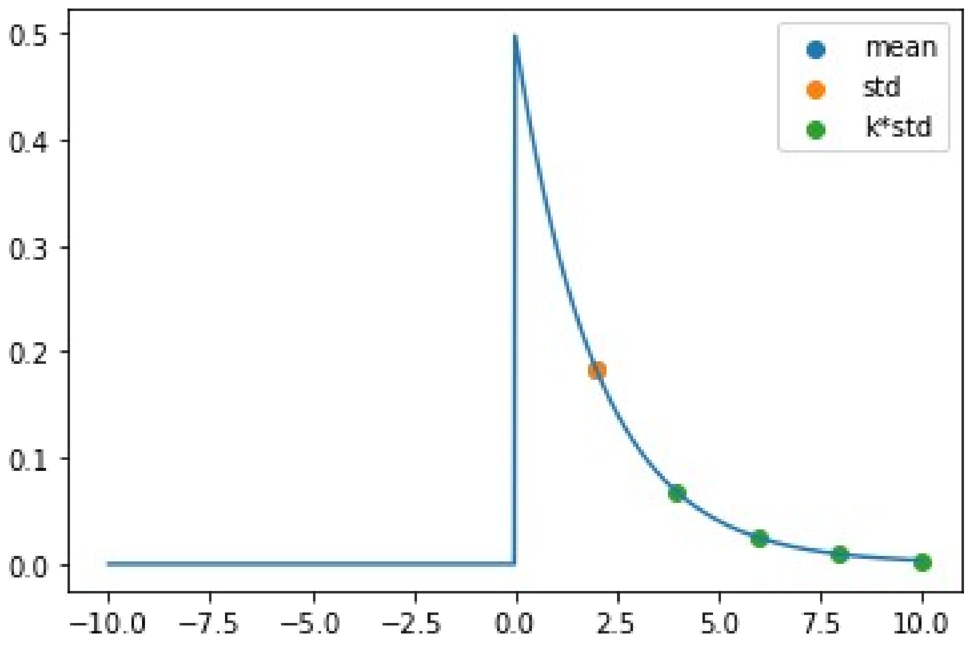


```
for k in range(100):
    x=np.arange(0,k,1)
    x1=x
    x2=-1*x
    y=norm.sf(x1,loc=0,scale=2)
    y1=norm.cdf(x2,loc=0,scale=2)
    y2=y+y1
plt.plot(x,y2)
plt.title('|P(x-E[x]|)>k*(std)')
plt.show()
```

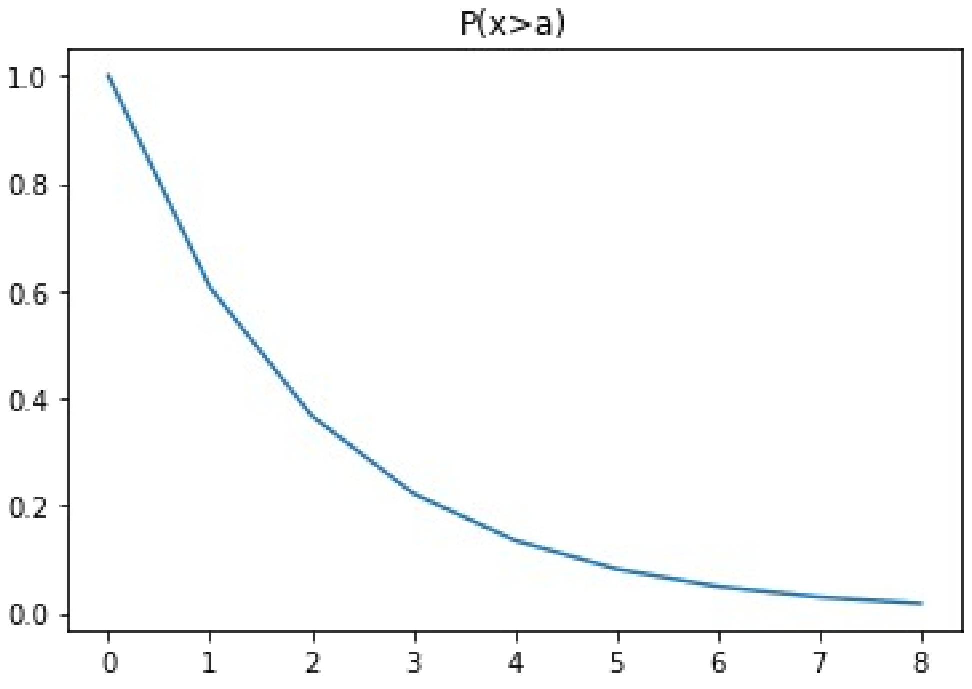


Exponential distribution

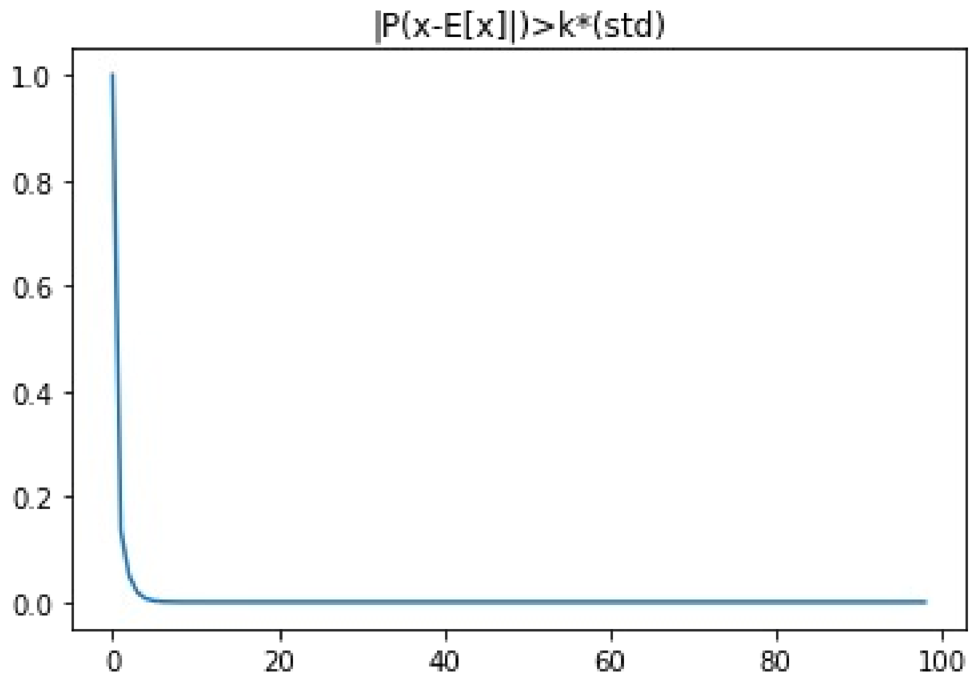
```
x=np.linspace(-10,10,1000)
y=expon.pdf(x,loc=0, scale=2)
plt.plot(x,y)
x1=expon.mean(loc=0,scale=2)
x2=expon.std(loc=0,scale=2)
x3=[2*x1,3*x1,4*x1,5*x1]
y=[expon.pdf(2*x1,loc=0, scale=2),expon.pdf(3*x1,loc=0,
scale=2),expon.pdf(4*x1,loc=0, scale=2),expon.pdf(5*x1,loc=0,
scale=2)]
plt.scatter(x1,expon.pdf(x1,loc=0, scale=2),label='mean')
plt.scatter(x2,expon.pdf(x2,loc=0, scale=2),label='std')
plt.scatter(x3,[expon.pdf(2*x1,loc=0, scale=2),expon.pdf(3*x1,loc=0,
scale=2),expon.pdf(4*x1,loc=0, scale=2),expon.pdf(5*x1,loc=0,
scale=2)],label='k*std')
plt.legend()
plt.show()
```



```
for i in range(10):
    x=np.arange(0,i,1)
    y=expon.cdf(x, loc=0, scale=2)
    y1=1-y
plt.plot(x,y1)
plt.title('P(x>a)')
plt.show()
```

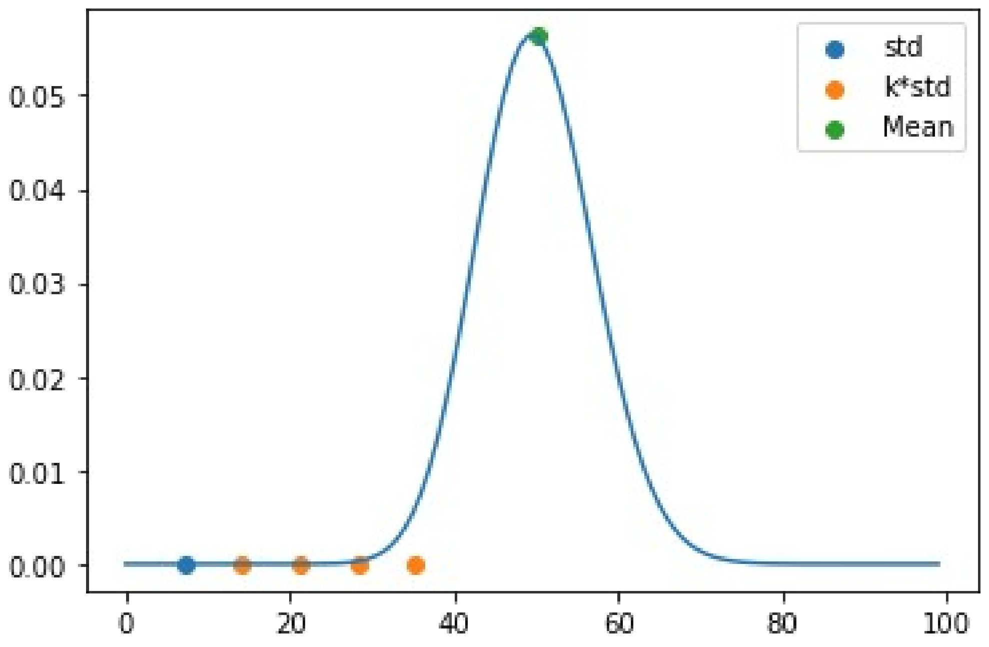


```
y_=expon.std(loc=0,scale=2)
for k in range(100):
    x=np.arange(0,k,1)
    x1=x*y_+y_
    x2=y_-x*y_
    y=expon.sf(x1,loc=0,scale=2)
    y1=expon.cdf(x2,loc=0,scale=2)
    y2=y+y1
plt.plot(x,y2)
plt.title('|P(x-E[x]|)>k*(std)')
plt.show()
```

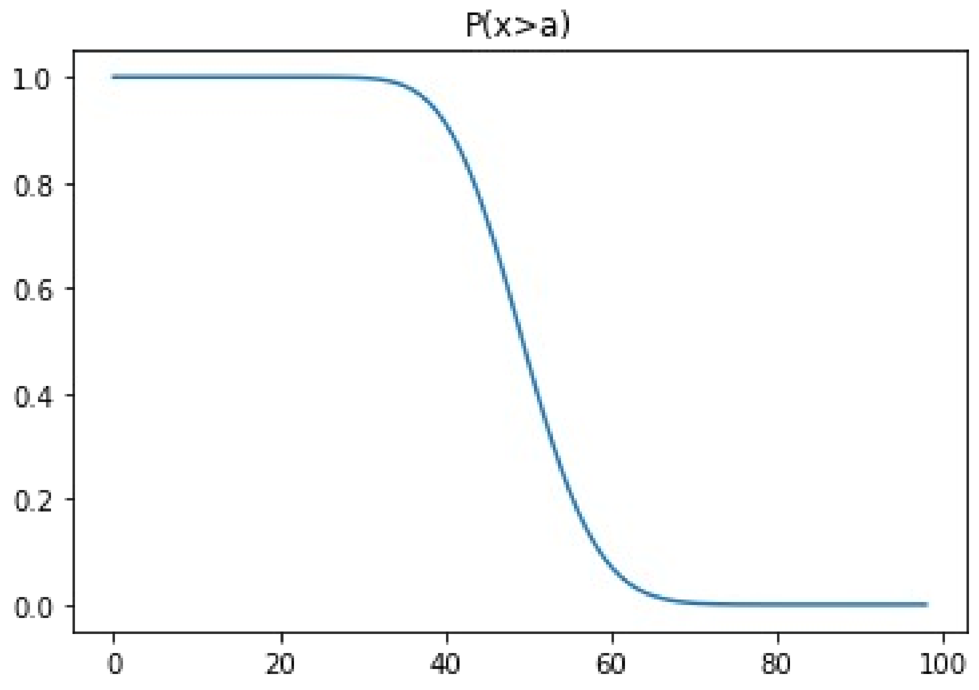


Poisson distribution

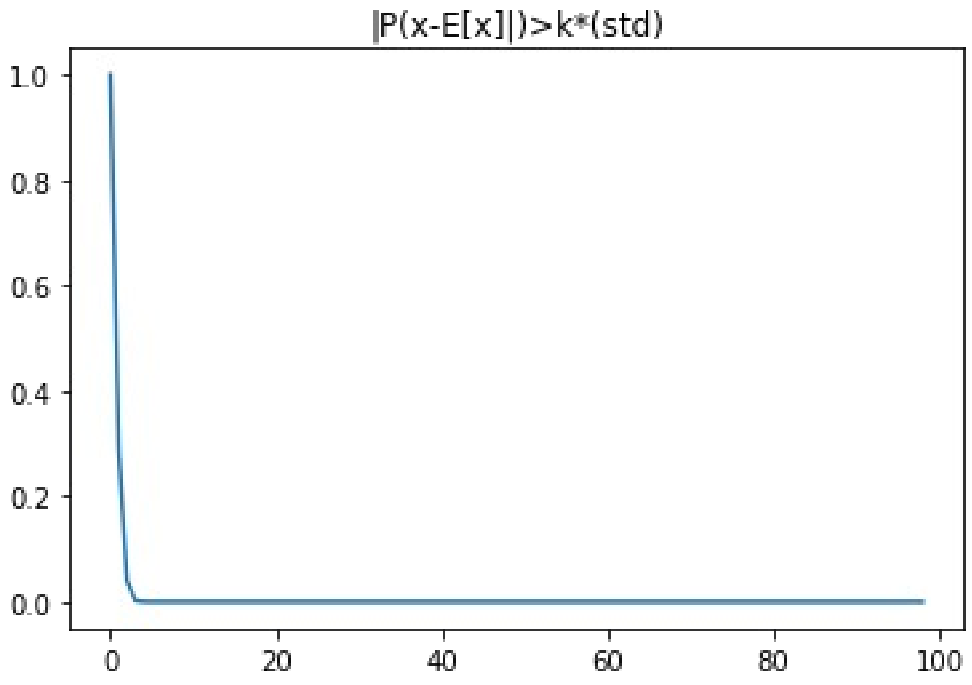
```
x=np.arange(0,100,1)
y=poisson.pmf(x,mu=50,loc=0)
plt.plot(x,y)
x=50
x1=np.sqrt(x)
y=0
x2=[2*x1,3*x1,4*x1,5*x1]
plt.scatter(x1,poisson.pmf(x1,mu=50,loc=0),label='std')
plt.scatter(x2,
[poisson.pmf(x1*2,mu=50,loc=0),poisson.pmf(x1*3,mu=50,loc=0),poisson.p
mf(x1*4,mu=50,loc=0),poisson.pmf(5*x1,mu=50,loc=0)],label='k*std')
plt.scatter(x,poisson.pmf(x,mu=50,loc=0),label='Mean')
plt.legend()
plt.show()
```



```
for i in range(100):
    x=np.arange(0,i,1)
    y=poisson.cdf(x,mu=50,loc=0)
    y1=1-y
plt.plot(x,y1)
plt.title('P(x>a)')
plt.show()
```

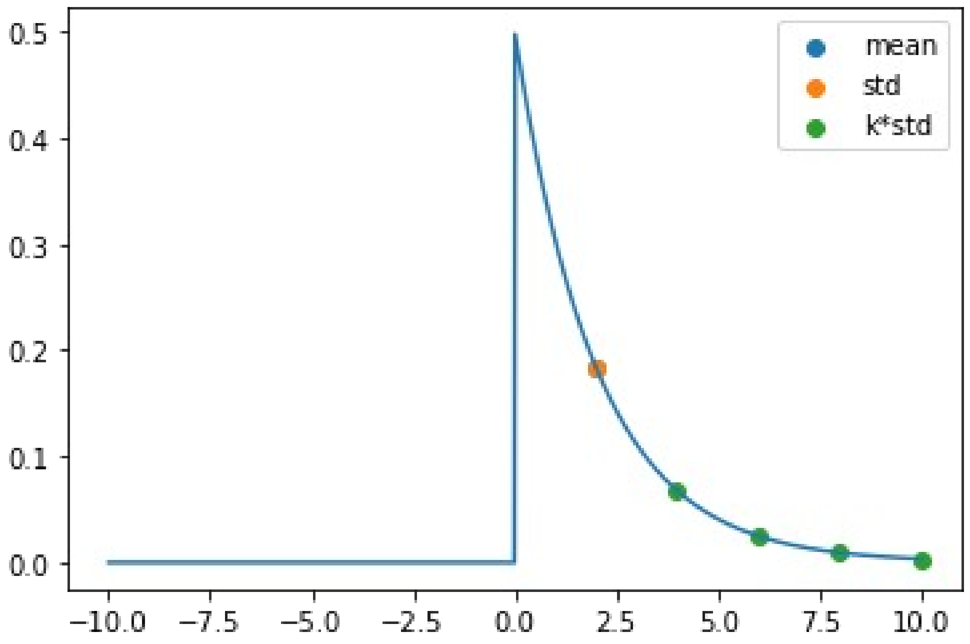


```
mu=50
for k in range(100):
    x=np.arange(0,k,1)
    x1=x*np.sqrt(mu)+mu
    x2=mu-x*np.sqrt(mu)
    y=poisson.sf(x1,mu,loc=0)
    y1=poisson.cdf(x2,mu,loc=0)
    y2=y+y1
plt.plot(x,y2)
plt.title('|P(x-E[x]|)>k*(std)')
plt.show()
```

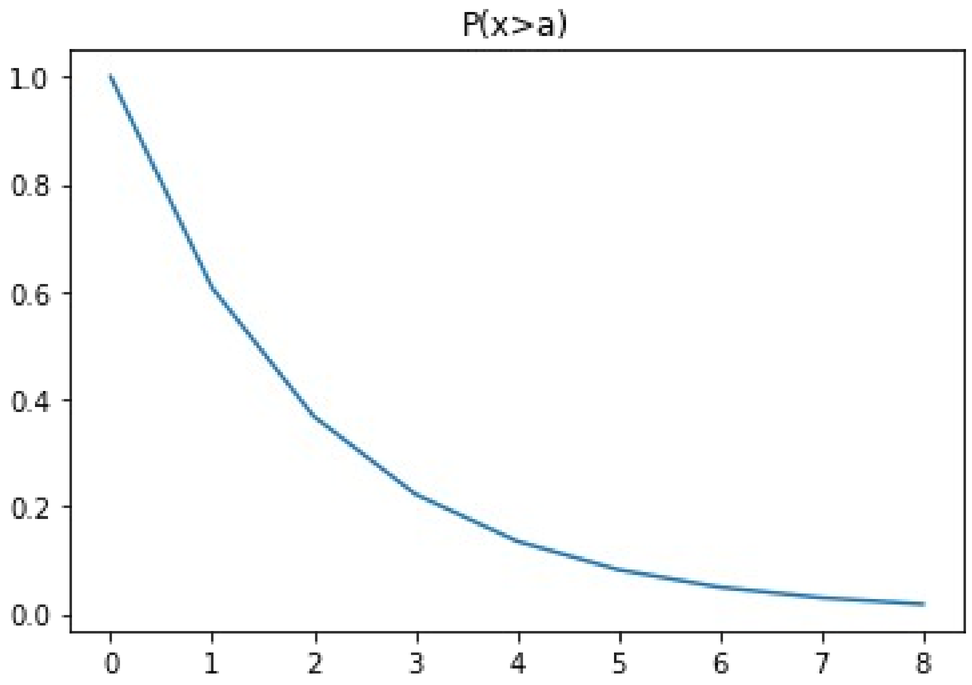


Exponential Distribution

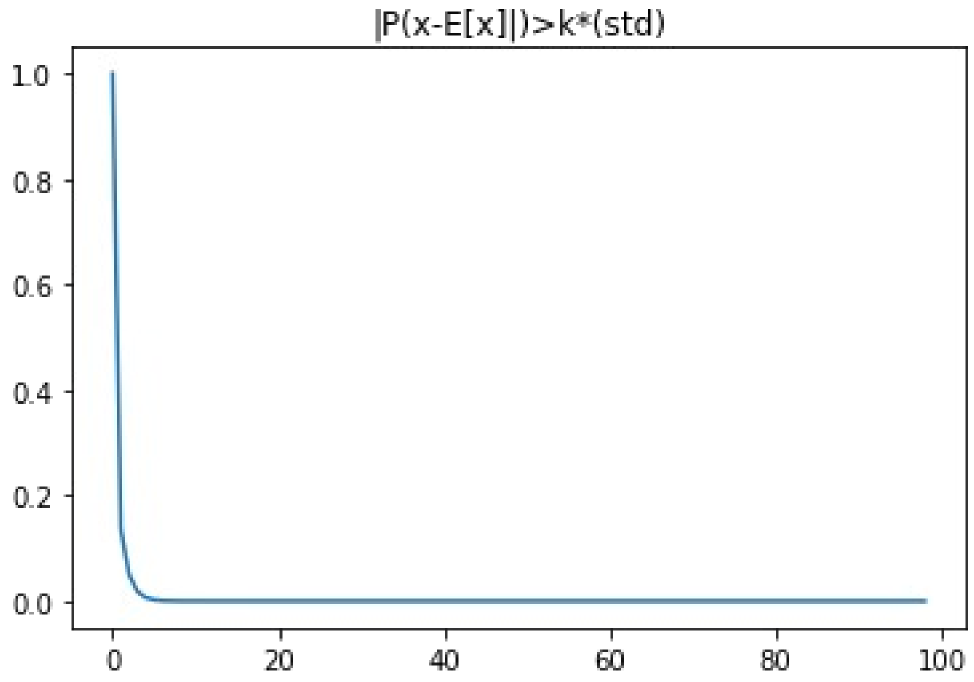
```
from scipy.stats import expon
x=np.linspace(-10,10,1000)
y=expon.pdf(x,loc=0, scale=2)
plt.plot(x,y)
x1=expon.mean(loc=0,scale=2)
x2=expon.std(loc=0,scale=2)
x3=[2*x1,3*x1,4*x1,5*x1]
y=[expon.pdf(2*x1,loc=0, scale=2),expon.pdf(3*x1,loc=0,
scale=2),expon.pdf(4*x1,loc=0, scale=2),expon.pdf(5*x1,loc=0,
scale=2)]
plt.scatter(x1,expon.pdf(x1,loc=0, scale=2),label='mean')
plt.scatter(x2,expon.pdf(x2,loc=0, scale=2),label='std')
plt.scatter(x3,[expon.pdf(2*x1,loc=0, scale=2),expon.pdf(3*x1,loc=0,
scale=2),expon.pdf(4*x1,loc=0, scale=2),expon.pdf(5*x1,loc=0,
scale=2)],label='k*std')
plt.legend()
plt.show()
```



```
for i in range(10):
    x=np.arange(0,i,1)
    y=expon.cdf(x, loc=0, scale=2)
    y1=1-y
plt.plot(x,y1)
plt.title('P(x>a)')
plt.show()
```

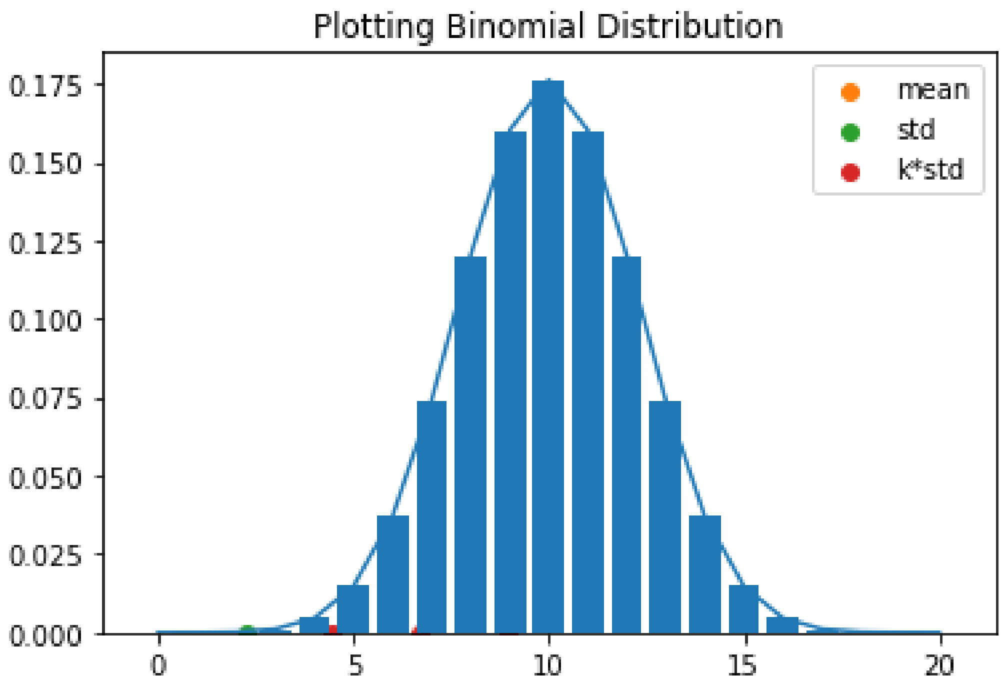


```
y_=expon.std(loc=0,scale=2)
for k in range(100):
    x=np.arange(0,k,1)
    x1=x*y_+y_
    x2=y_-x*y_
    y=expon.sf(x1,loc=0,scale=2)
    y1=expon.cdf(x2,loc=0,scale=2)
    y2=y+y1
plt.plot(x,y2)
plt.title('|P(x-E[x]|)>k*(std)')
plt.show()
```

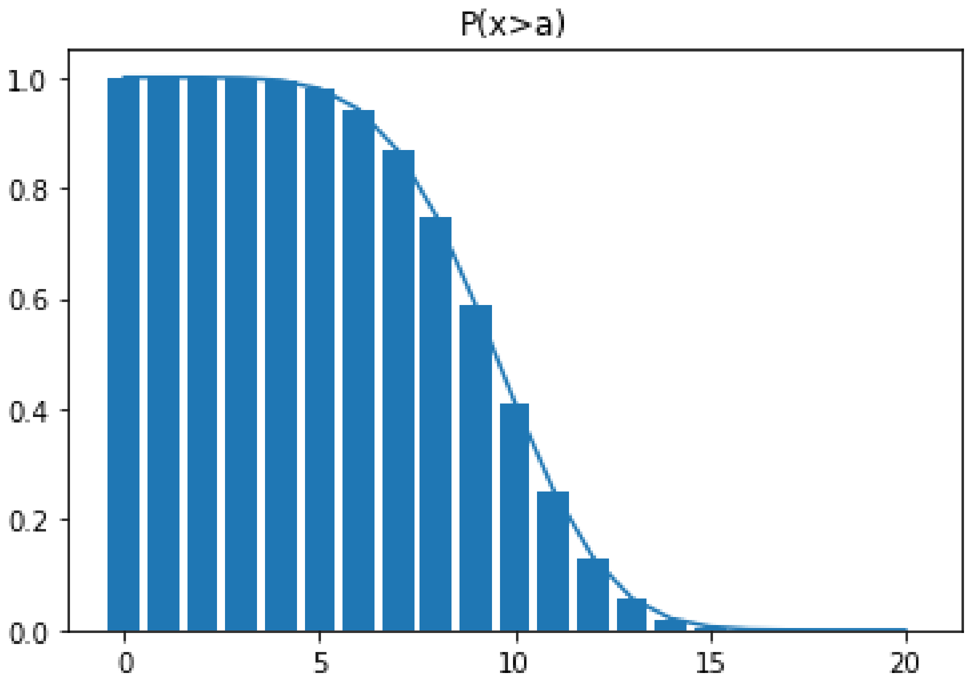


Binomial distribution

```
from scipy.stats import binom
n=20
p=0.5
rval=list(range(n+1))
bdis=[]
for r in rval:
    bdis.append(binom.pmf(r,n,p))
plt.plot(rval,bdis)
plt.bar(rval,bdis)
plt.title('Plotting Binomial Distribution')
x=n*p
y=0
x1=n*p*(1-p)
x2=np.sqrt(x1)
x3=[2*x2,3*x2,4*x2]
plt.scatter(x,binom.pmf(x1,n,p),label='mean')
plt.scatter(x2,binom.pmf(x2,n,p),label='std')
plt.scatter(x3,
[binom.pmf(2*x2,n,p),binom.pmf(3*x2,n,p),binom.pmf(4*x2,n,p)],label='k
*std')
plt.legend()
plt.show()
```

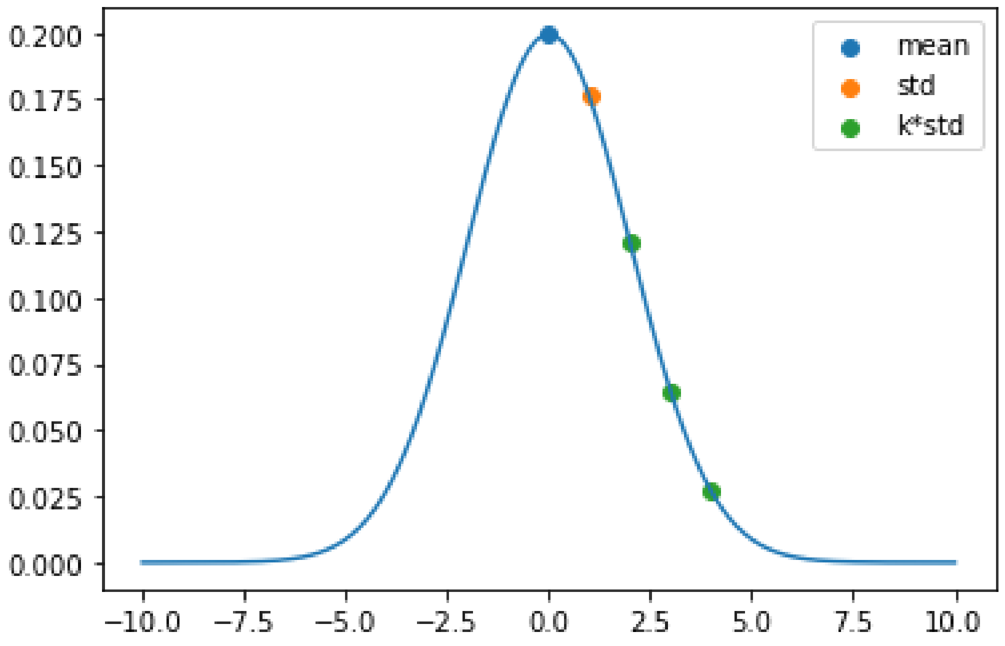


```
bsf=[]
for r in rval:
    bsf.append(binom.sf(r,n,p,loc=0))
plt.plot(rval,bsf)
plt.title('P(x>a)')
plt.bar(rval,bsf)
plt.show()
```

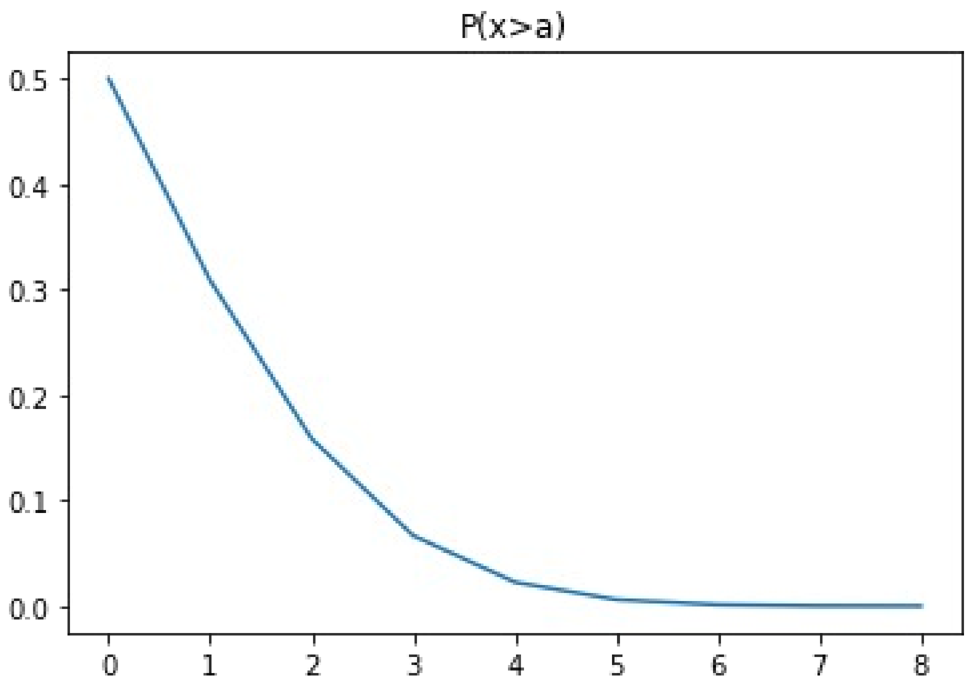


Gaussian distribution

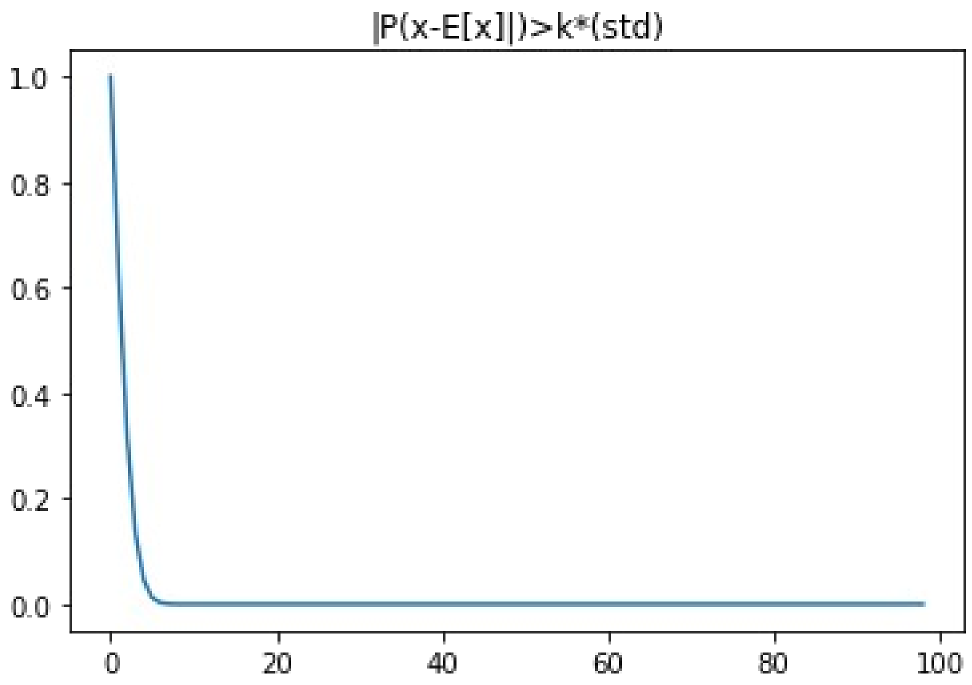
```
from scipy.stats import norm
import matplotlib.pyplot as plt
x=np.linspace(-10,10,1000)
y=norm.pdf(x,loc=0,scale=2)
y1=norm.mean(loc=0,scale=2)
plt.plot(x,y)
x=0
x1=1
x2=[2*x1,3*x1,4*x1]
plt.scatter(x,norm.pdf(x,loc=0,scale=2),label='mean')
plt.scatter(x1,norm.pdf(x1,loc=0,scale=2),label='std')
plt.scatter(x2,
[norm.pdf(2*x1,loc=0,scale=2),norm.pdf(3*x1,loc=0,scale=2),norm.pdf(4*
x1,loc=0,scale=2)],label='k*std')
#plt.plot(x,y)
plt.legend()
plt.show()
```



```
for i in range(10):
    x=np.arange(0,i,1)
    y=norm.sf(x, loc=0, scale=2)
plt.plot(x,y)
plt.title('P(x>a)')
plt.show()
```



```
for k in range(100):
    x=np.arange(0,k,1)
    x1=x
    x2=-1*x
    y=norm.sf(x1,loc=0,scale=2)
    y1=norm.cdf(x2,loc=0,scale=2)
    y2=y+y1
plt.plot(x,y2)
plt.title('|P(x-E[x]|)>k*(std)')
plt.show()
```

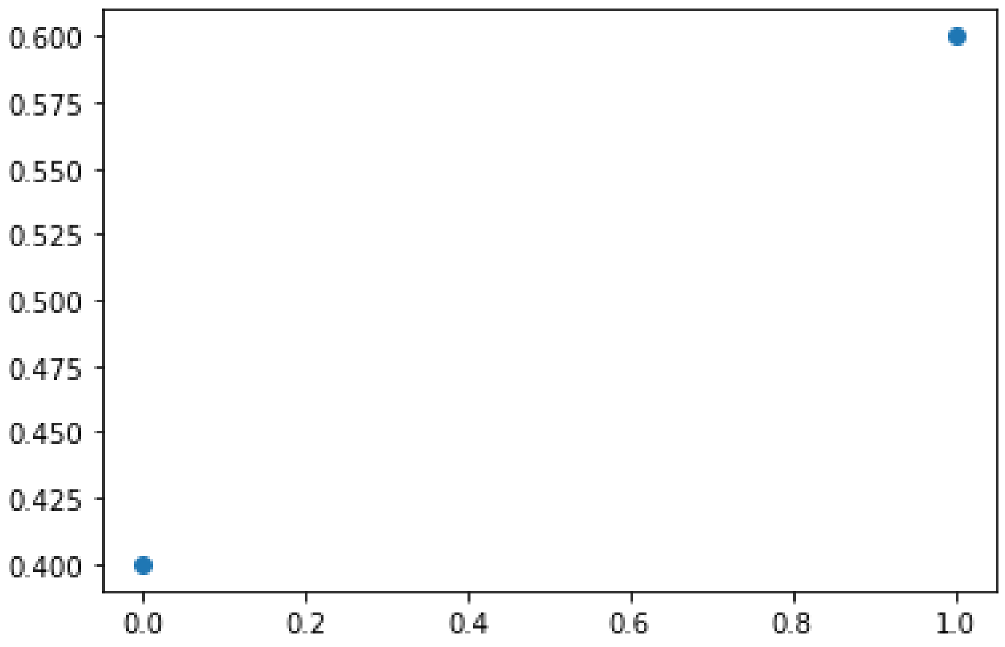


Bernauli Distribution

```python
def bernouliPMF(a,p):
    if a==0:
        pmf=1-p
    elif a==1:
        pmf=p
    return pmf

def bernouliCMF(a,p):
    if a<0:
        return 0
    elif a>=0 and a<1:
        return bernouliPMF(0,p)
    elif X>=1:
        return bernouliPMF(1,p)

x=[0,1]
p=0.6
y=[bernouliPMF(i,p) for i in x]
plt.scatter(x,y)
plt.show()
```


0.600
0.575
0.550
0.525
0.500
0.475
0.450
0.425
0.400
0.0
0.2
0.4
0.6
0.8
1.0