Practical-7

Aim :

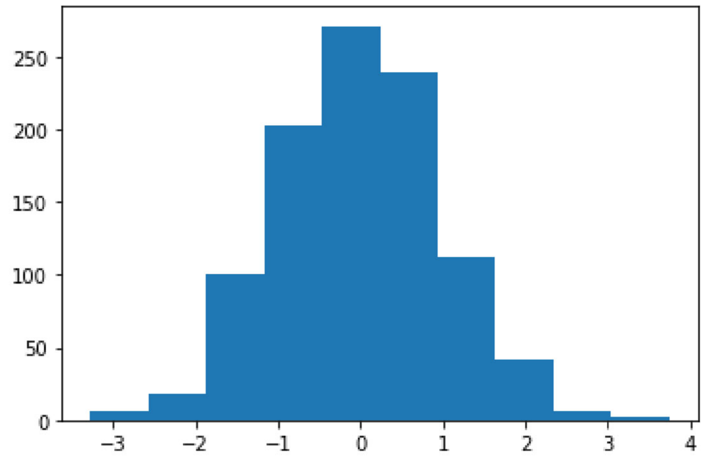
Write a python program to download appropriate dataset and explore random variable, Probability mass function, Probability density function, Cumulative distribution function, Discrete probability distribution and continuous probability distribution using scipy.stats , rv_discrete class and rv_continuous class .

```
import pandas as pd
import numpy as np
from numpy.random import normal
from scipy.stats import binom
from scipy.stats import norm
from scipy.stats import rv_continuous
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns

print("12002040701067")
#Random Variable
x = np.random.choice([3,5,7,9], p=[0.1,0.3,0.6,0.0], size=(100))
print(x)

#Probability Mass Function(PMF)
n = 10
p = 0.35
r = list(range(n + 1))
return_val = binom.pmf(r, n, p)
print("Return Value")
print("Loop to atleast our observation : ")
for i in range(0,n+1):
  print(return_val[i]+1-return_val[i])
```

```
12002040701067
[7 7 3 7 5 5 7 7 5 5 7 7 7 7 7 7 5 3 7 7 7 7 7 5 7 7 3 5 7 7 7 7 5 5 5 3 5
 5 7 7 5 7 7 7 7 5 3 7 5 7 7 7 7 7 7 7 7 5 5 5 7 7 7 3 7 3 7 5 5 7 5 7 7 7
 7 5 7 7 7 7 7 3 7 5 5 7 7 7 7 3 5 5 7 7 7 7 7 7 7 5]
Return Value
Loop to atleast our observation :
0.9999999999999999
1.0
1.0
1.0
1.0
0.9999999999999999
1.0
1.0
0.9999999999999999
```

```
     1.0
     1.0
```

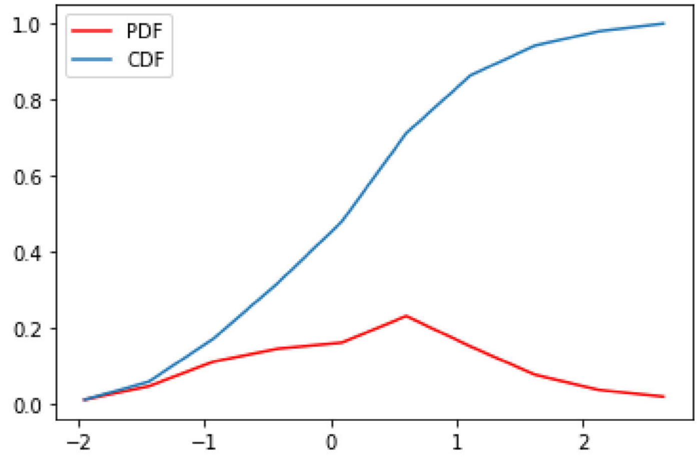
```
#Probability Density Function(PDF)
sample = normal(size=1000)
plt.hist(sample,bins=10)
plt.show()
# sample = normal(loc=50,scale=5,size=1000)
# sample
sample_mean = np.mean(sample)
sample_std = np.std(sample)
print("Mean=%.3f,Standard Dev=%.3f"%(sample_mean,sample_std))
dist = norm(sample_mean,sample_std)
dist
values = [value for value in range(30,70)]
probs = [dist.pdf(value) for value in values]
probs
```



```
Mean=-0.020,Standard Dev=1.004
[2.3571792178950818e-195,
 1.64771962505251e-208,
 4.268974363519013e-222
```

```
#Cummulative Distribution Function
n = 500
data = np.random.randn(n)
count,bins_count = np.histogram(data,bins=10)
pdf = count/sum(count)
cdf = np.cumsum(pdf)
plt.plot(bins_count[1:],pdf,color="red",label="PDF")
plt.plot(bins_count[1:],cdf,label="CDF")
plt.legend()
```

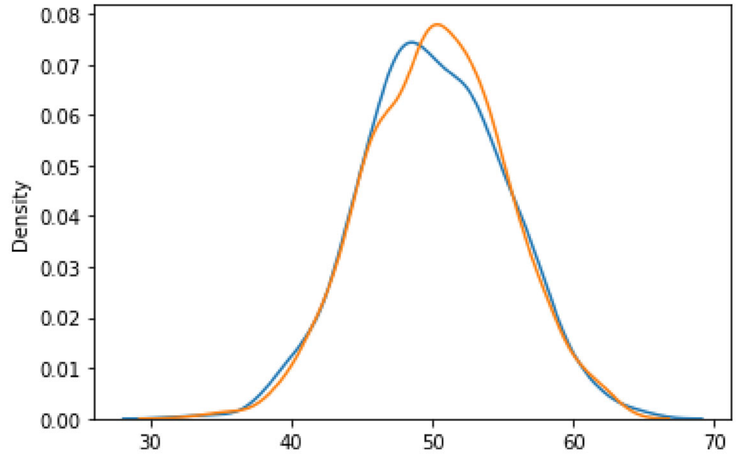
```
<matplotlib.legend.Legend at 0x7fc552607390>
```



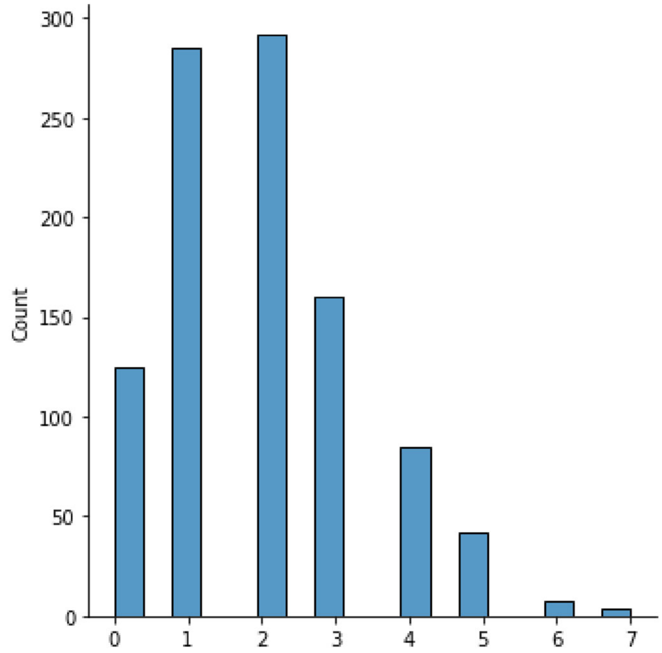
```
0 0
```

```
#Discrete Probability Distribution
# 1.Binomial
sns.distplot(np.random.normal(loc=50, scale=5, size=1000),hist=False, label='normal')
sns.distplot(np.random.binomial(n=100, p=0.5, size=1000),hist=False, label='binomial')
plt.show()

#2.Poisson
x = np.random.poisson(lam=2, size=10)
print(x)
sns.displot(np.random.poisson(lam=2, size=1000), kde=False)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distpl
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distpl
  warnings.warn(msg, FutureWarning)
```



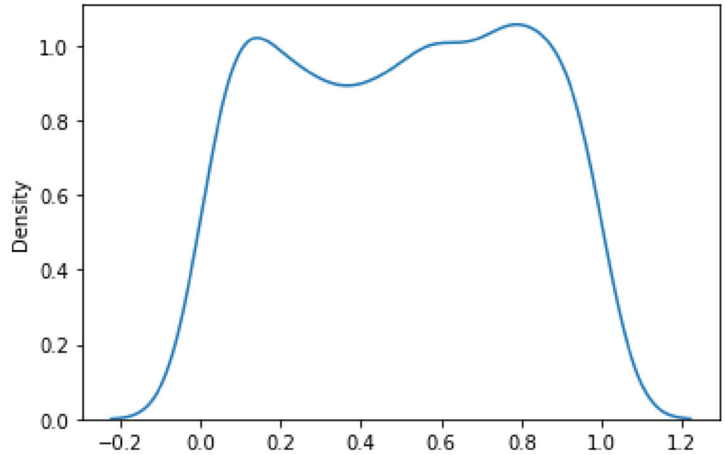
```
[1 3 3 1 0 3 3 3 2 2]
```



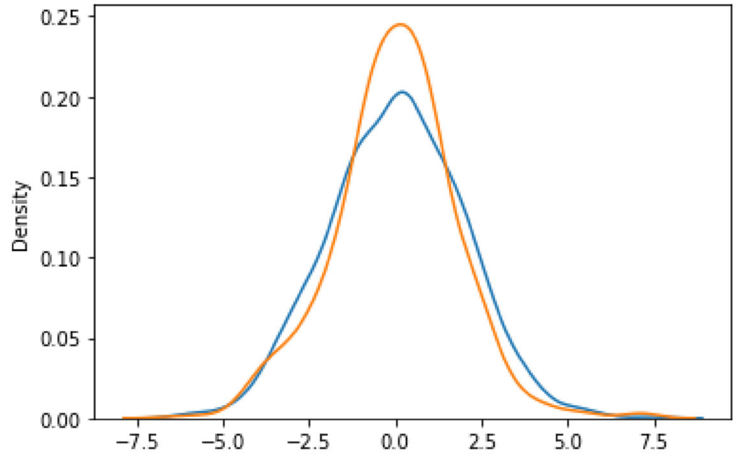
```
#Continuous Probability Distribution
#1.Uniform Distribution
sns.distplot(np.random.uniform(size=1000), hist=False)
plt.show()

# 2.Logistic Distribution
sns.distplot(np.random.normal(scale=2, size=1000),hist=False, label='normal')
sns.distplot(np.random.logistic(size=1000),hist=False, label='logistic')
plt.show()

#3.Multinormal Distribution
x =np.random.multinomial(n=6, pvals=[1/6,1/6,1/6,1/6,1/6,1/6])
print(x)

#4.Exponential Distribution
```

```
sns.distplot(np.random.exponential(size=1000), hist=False)
plt.show()
```

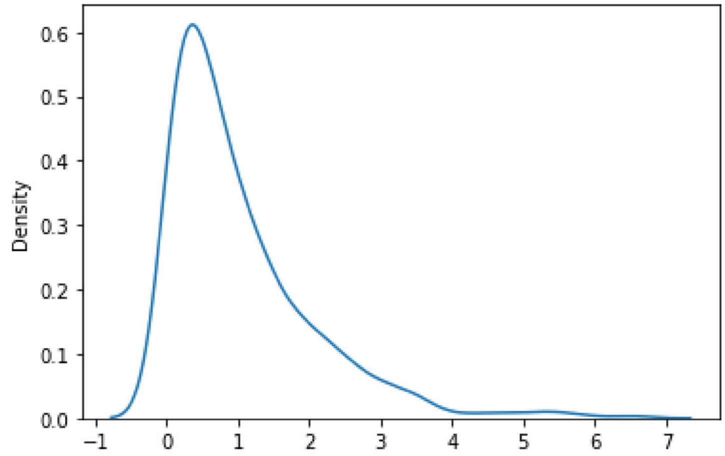
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplo
  warnings.warn(msg, FutureWarning)
```



```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplo
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplo
  warnings.warn(msg, FutureWarning)
```

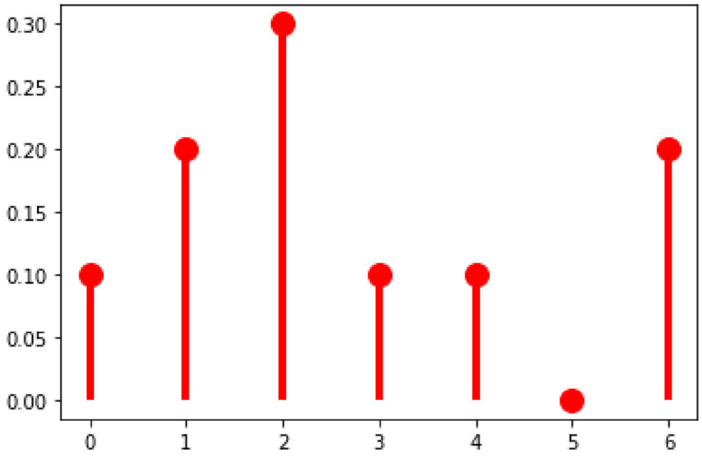


```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplo
  warnings.warn(msg, FutureWarning)
[1 2 1 0 0 2]
```



```
#rv_discrete class
```

```
xk = np.arange(7)
pk = (0.1,0.2,0.3,0.1,0.1,0.0,0.2)
custom = stats.rv_discrete(name="custom", values=(xk,pk))
fig, ax = plt.subplots(1,1)
ax.plot(xk,custom.pmf(xk),"ro",ms=12,mec="r")
ax.vlines(xk,0,custom.pmf(xk),colors='r',lw=4)
plt.show()
```



```
#rv_continuous class
class gaussian_gen(rv_continuous):
  '''Gaussian distribution'''
  def _pdf(self,x):
    return np.exp(-x**2 / 2.) / np.sqrt(2.0 * np.pi)

gaussian = gaussian_gen(name = 'gaussian')
x = 2.0
gaussian._pdf(x)
```

```
     0.05399096651318806
```