Practical Part 1

Q.No.1 Write a Python program to generate a random variable and display its value.

Ans-

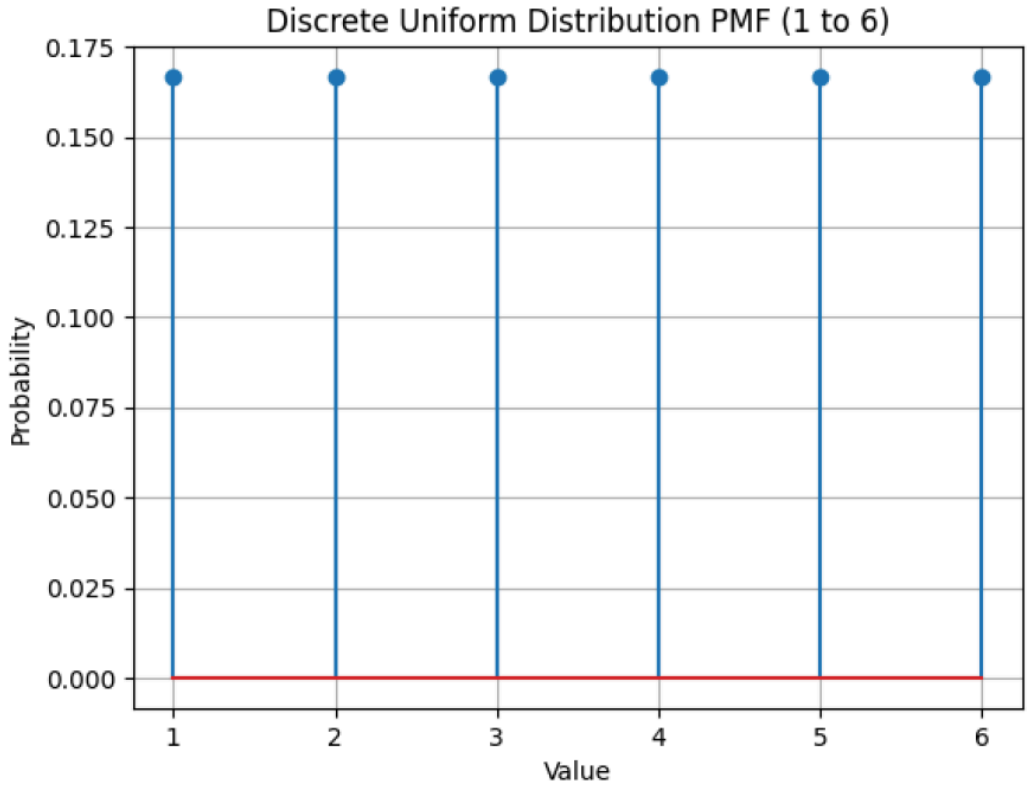
```
import numpy as np
random_variable = np.random.rand()
print("Random Variable:", random_variable)
```

```
Random Variable: 0.09608016810299902
```

Q.No.2 Generate a discrete uniform distribution using Python and plot the probability mass function (PMF).

Ans-

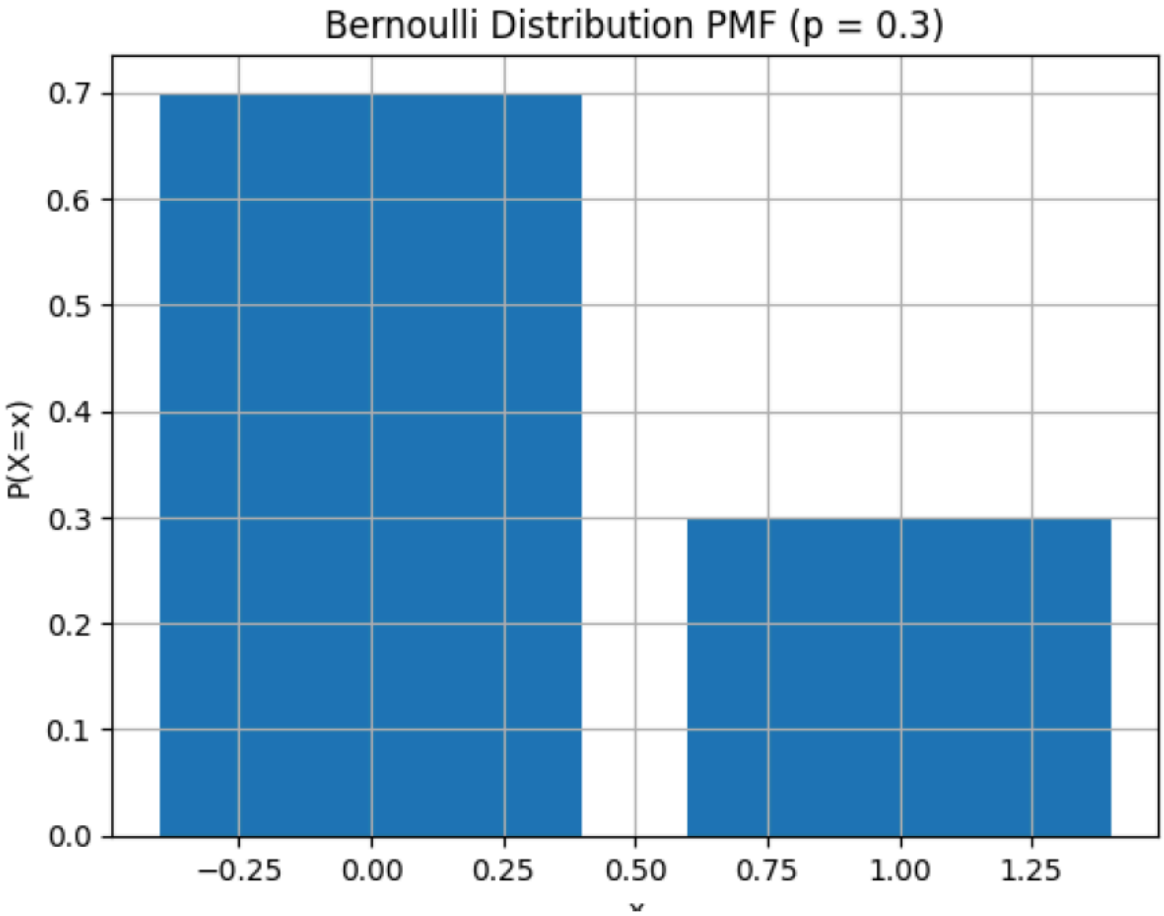
```
import matplotlib.pyplot as plt
values = np.arange(1, 7)
probs = [1/6] * len(values)
plt.stem(values, probs)
plt.title("Discrete Uniform Distribution PMF (1 to 6)")
plt.xlabel("Value")
plt.ylabel("Probability")
plt.grid(True)
plt.show()
```



Q.No.3 Write a Python function to calculate the probability distribution function (PDF) of a Bernoulli distribution .

Ans-

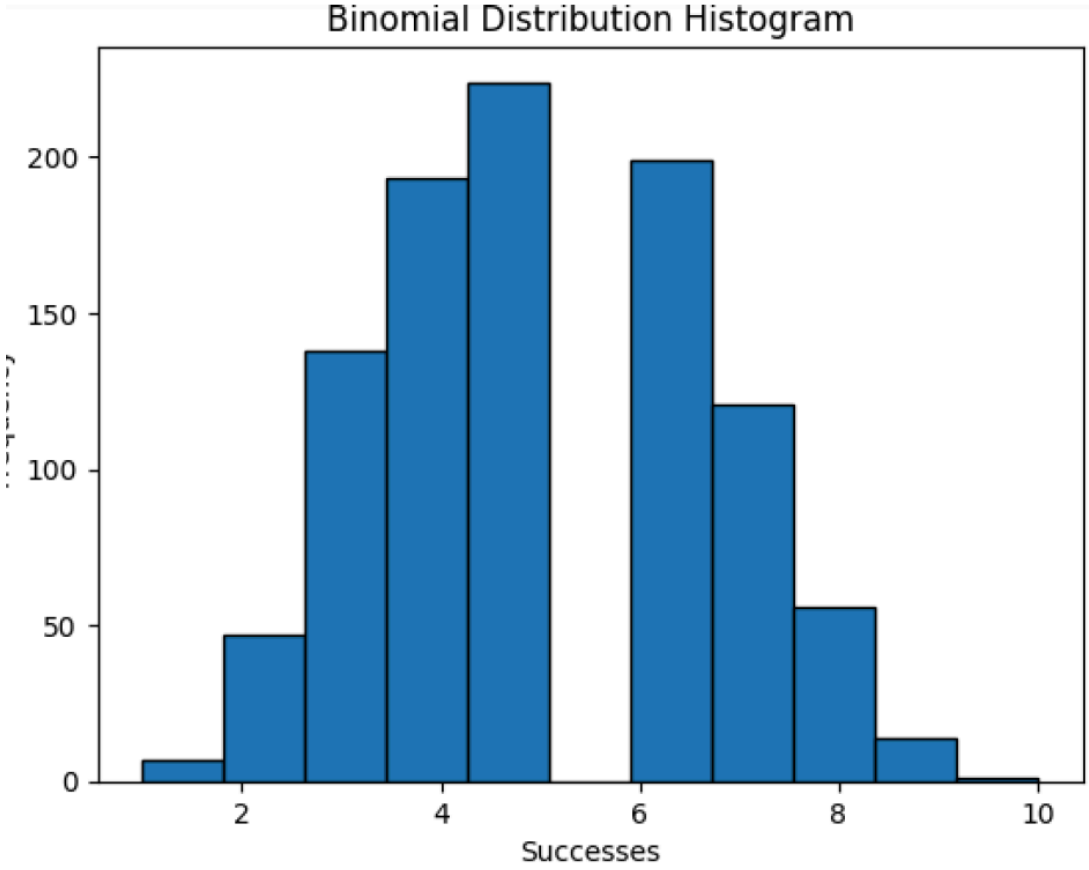
```
from scipy.stats import bernoulli
x = [0, 1]
probs = bernoulli.pmf(x, p=0.3)
plt.bar(x, probs)
plt.title("Bernoulli Distribution PMF (p = 0.3)")
plt.xlabel("x")
plt.ylabel("P(X=x)")
plt.grid(True)
plt.show()
```



Q.No.4 Write a Python script to simulate a binomial distribution with n=10 and p=0.5, then plot its histogram.

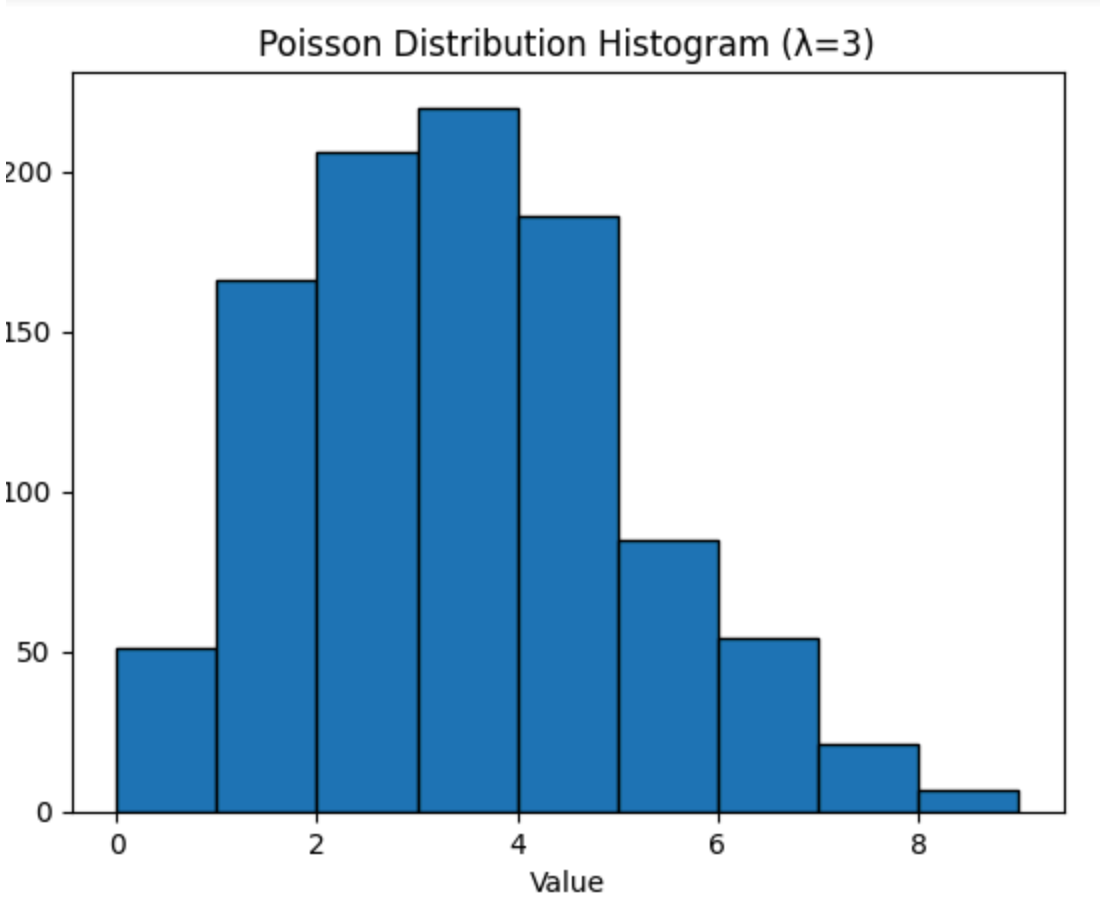
Ans-

```
from scipy.stats import binom
data = binom.rvs(n=10, p=0.5, size=1000)
plt.hist(data, bins=11, edgecolor='black')
plt.title("Binomial Distribution Histogram")
plt.xlabel("Successes")
plt.ylabel("Frequency")
plt.show()
```



Q.No.5 Create a Poisson distribution and visualize it using Python.
Ans-

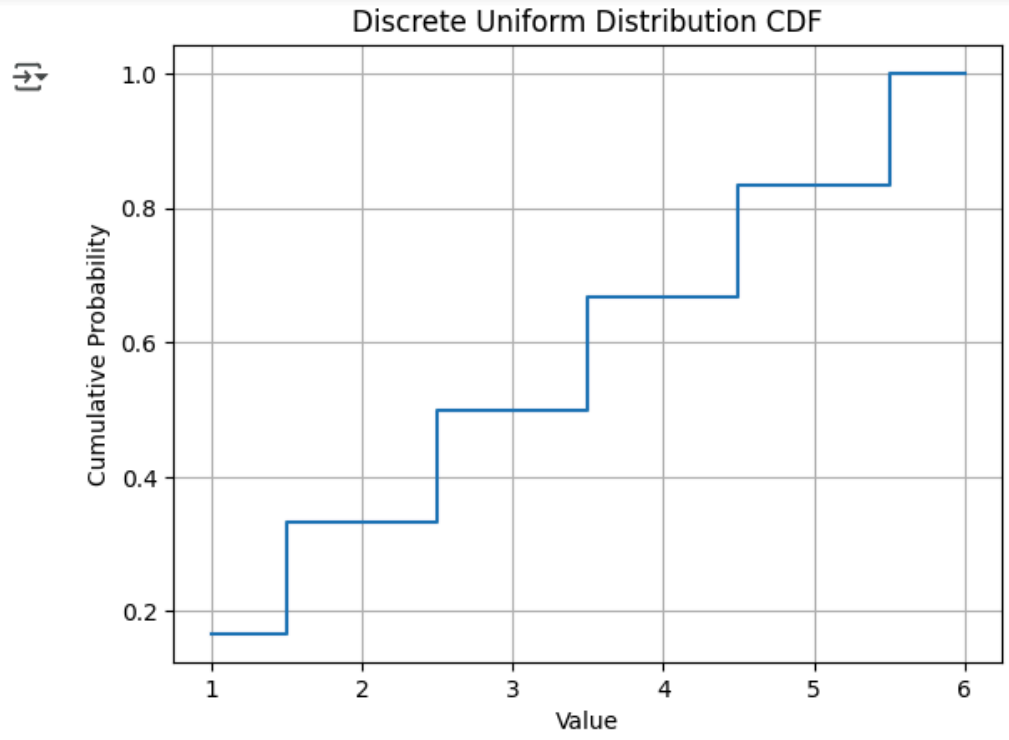
```
[ ]  from scipy.stats import poisson
     data = poisson.rvs(mu=3, size=1000)
     plt.hist(data, bins=range(10), edgecolor='black')
     plt.title("Poisson Distribution Histogram (λ=3)")
     plt.xlabel("Value")
     plt.ylabel("Frequency")
     plt.show()
```



Q.No.6 Write a Python program to calculate and plot the cumulative distribution function (CDF) of a discrete uniform distribution.

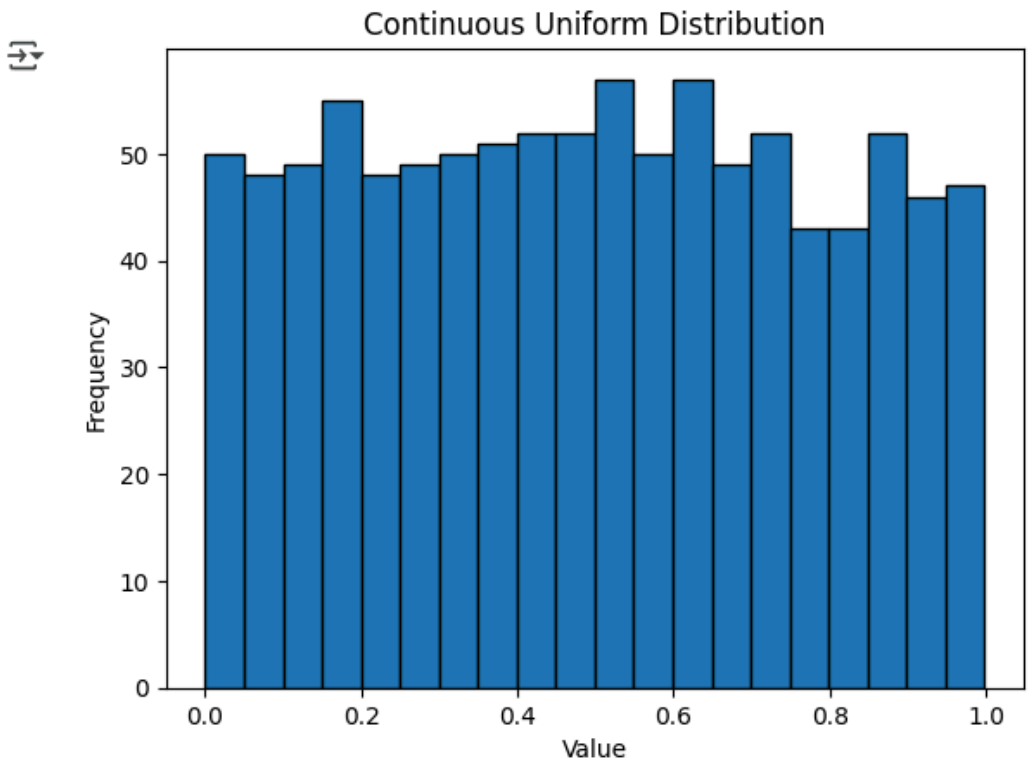
Ans-

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt

     values = np.arange(1, 7)
     probs = [1/6] * len(values)

     cdf = np.cumsum(probs)

     plt.step(values, cdf, where='mid')
     plt.title("Discrete Uniform Distribution CDF")
     plt.xlabel("Value")
     plt.ylabel("Cumulative Probability")
     plt.grid(True)
     plt.show()
```



Q.No.7 Generate a continuous uniform distribution using NumPy and visualize it.
Ans-

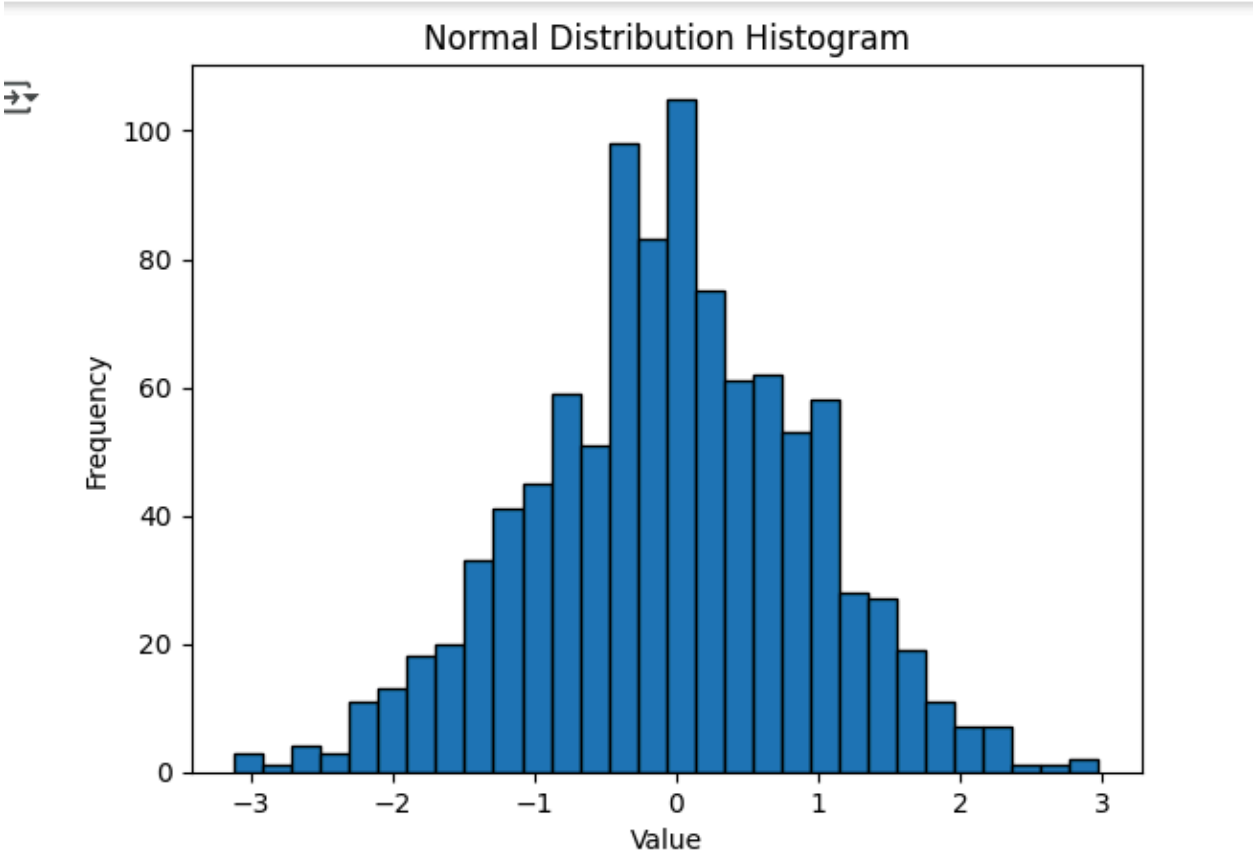
```
[ ]  from scipy.stats import uniform
     data = uniform.rvs(loc=0, scale=1, size=1000)
     plt.hist(data, bins=20, edgecolor='black')
     plt.title("Continuous Uniform Distribution")
     plt.xlabel("Value")
     plt.ylabel("Frequency")
     plt.show()
```



Q.No.8 Simulate data from a normal distribution and plot its histogram.

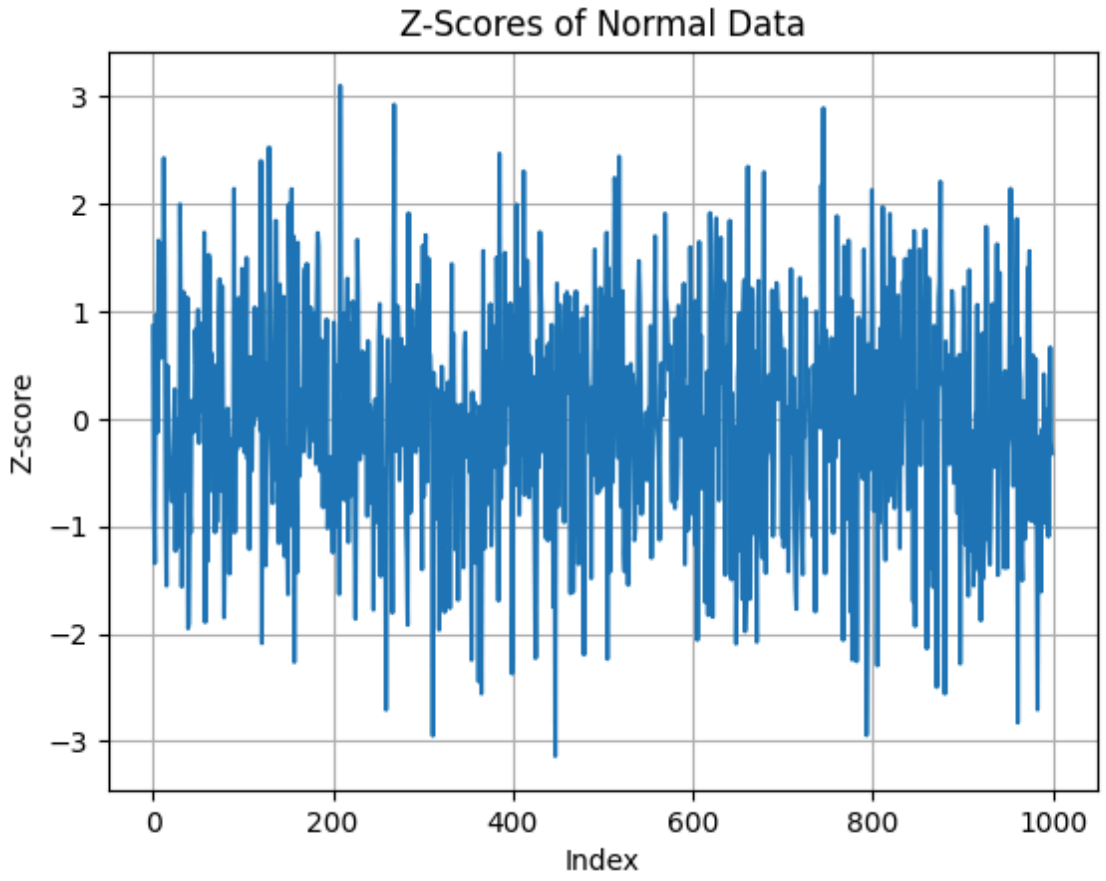
Ans-

```
[ ]  from scipy.stats import norm
     data = norm.rvs(loc=0, scale=1, size=1000)
     plt.hist(data, bins=30, edgecolor='black')
     plt.title("Normal Distribution Histogram")
     plt.xlabel("Value")
     plt.ylabel("Frequency")
     plt.show()
```



Q.No.9 Write a Python function to calculate Z-scores from a dataset and plot them.
Ans-

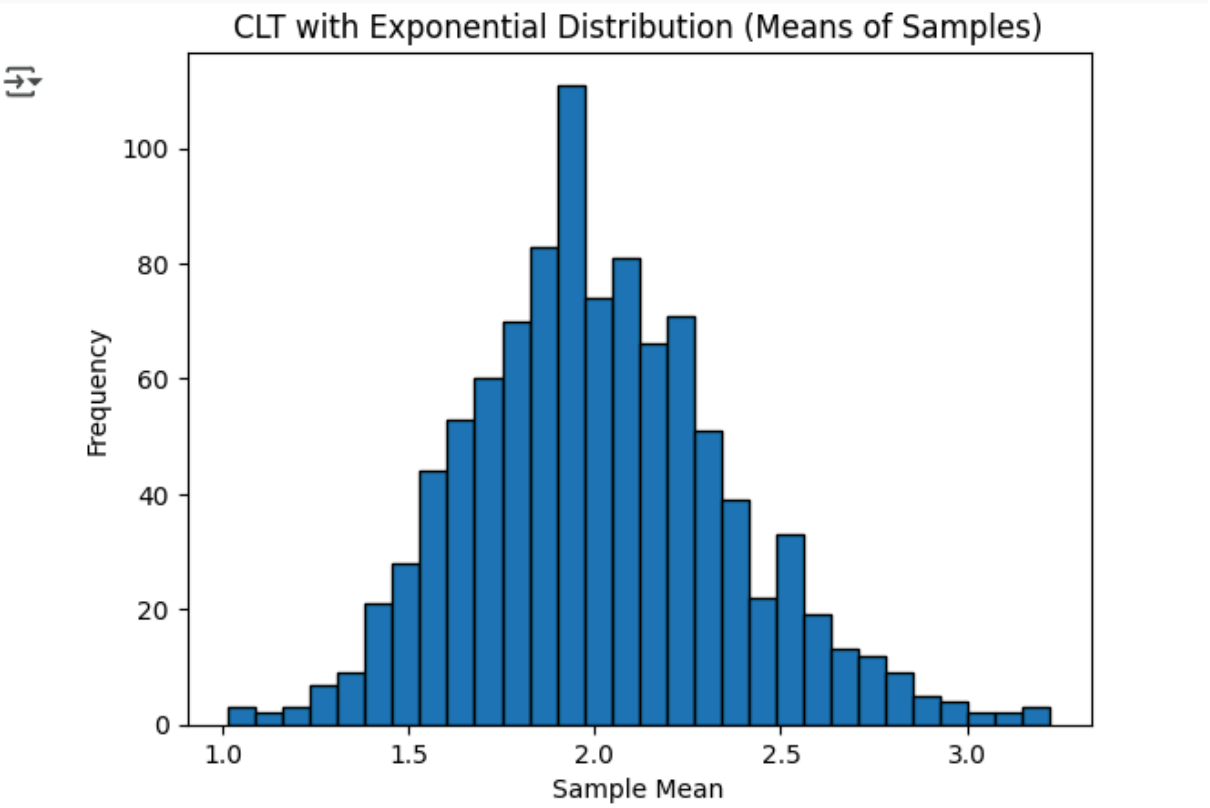
```
[ ]  from scipy.stats import zscore
     z_scores = zscore(data)
     plt.plot(z_scores)
     plt.title("Z-Scores of Normal Data")
     plt.xlabel("Index")
     plt.ylabel("Z-score")
     plt.grid(True)
     plt.show()
```



Q.No.10 Implement the Central Limit Theorem (CLT) using Python for a non-normal distribution. Java + DSA.

Ans-

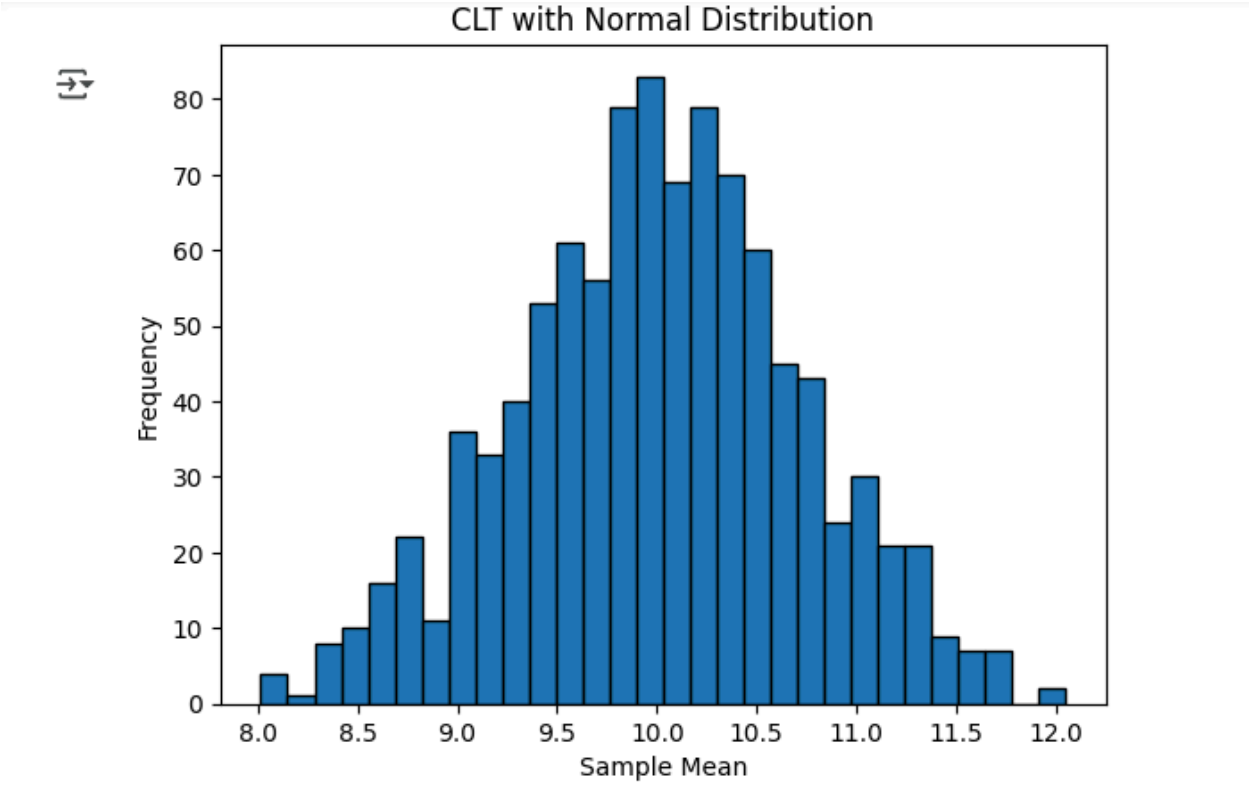
```
[ ]  sample_means = [np.mean(np.random.exponential(scale=2, size=30)) for _ in range(1000)]
     plt.hist(sample_means, bins=30, edgecolor='black')
     plt.title("CLT with Exponential Distribution (Means of Samples)")
     plt.xlabel("Sample Mean")
     plt.ylabel("Frequency")
     plt.show()
```



Q.No.11 Simulate multiple samples from a normal distribution and verify the Central Limit Theorem.

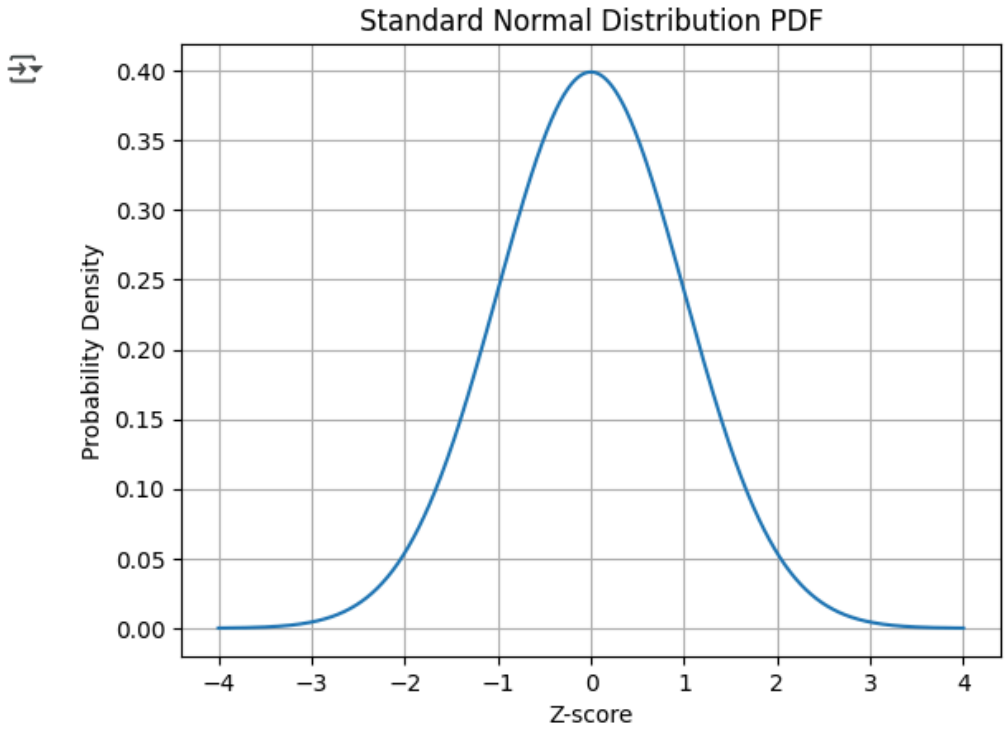
Ans-

```
[ ]  samples = [np.mean(norm.rvs(loc=10, scale=5, size=50)) for _ in range(1000)]
     plt.hist(samples, bins=30, edgecolor='black')
     plt.title("CLT with Normal Distribution")
     plt.xlabel("Sample Mean")
     plt.ylabel("Frequency")
     plt.show()
```



Q.No.12 Write a Python function to calculate and plot the standard normal distribution (mean = 0, std = 1).
Ans-

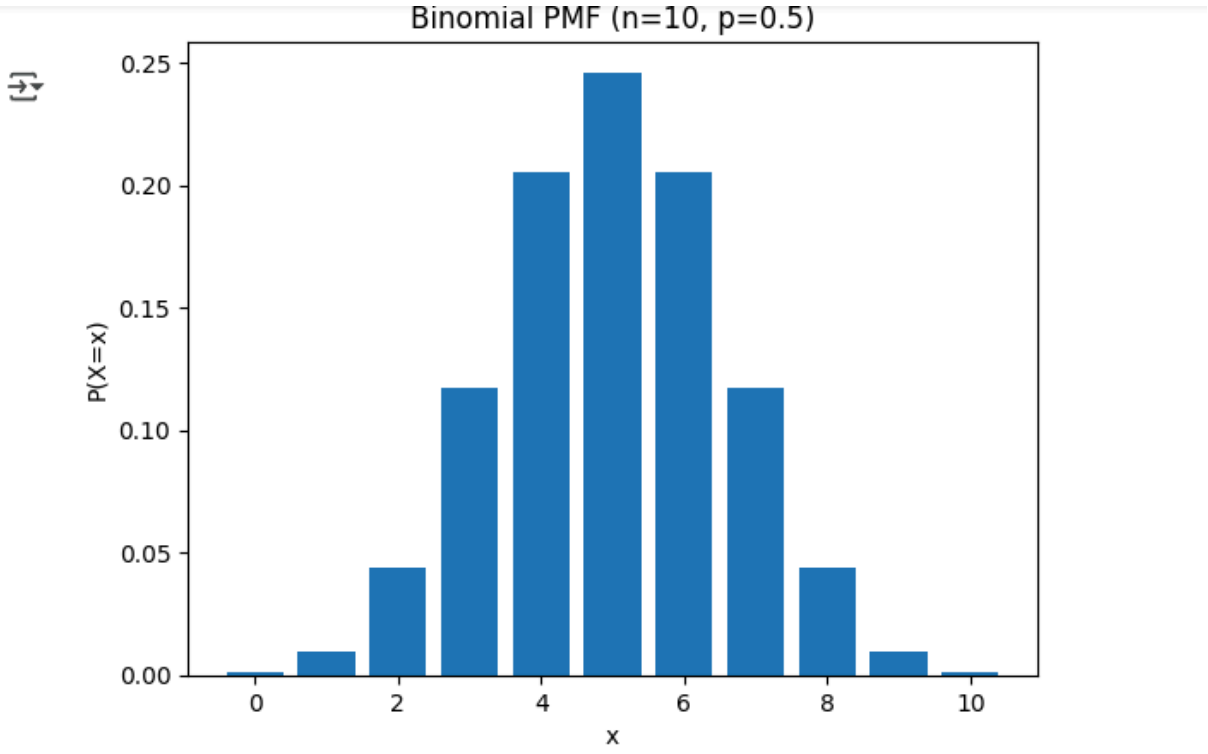
```
x = np.linspace(-4, 4, 1000)
pdf = norm.pdf(x)
plt.plot(x, pdf)
plt.title("Standard Normal Distribution PDF")
plt.xlabel("Z-score")
plt.ylabel("Probability Density")
plt.grid(True)
plt.show()
```



Q.No.13 Generate random variables and calculate their corresponding probabilities using the binomial distribution.

Ans-

```
x = np.arange(0, 11)
pmf = binom.pmf(x, n=10, p=0.5)
plt.bar(x, pmf)
plt.title("Binomial PMF (n=10, p=0.5)")
plt.xlabel("x")
plt.ylabel("P(X=x)")
plt.show()
```



Q.No.14 Write a Python program to calculate the Z-score for a given data point and compare it to a standard normal distribution.
Ans-

```
[ ]  x = 72
     mu = 70
     sigma = 5
     z = (x - mu) / sigma
     print("Z-score:", z)
```

```
Z-score: 0.4
```

Q.No.15 Implement hypothesis testing using Z-statistics for a sample dataset.
Ans-

```
[ ]  sample_mean = 102
     population_mean = 100
     std_dev = 10
     n = 50

     z_stat = (sample_mean - population_mean) / (std_dev / np.sqrt(n))
     p_value = 1 - norm.cdf(z_stat)

     print("Z-statistic:", z_stat)
     print("P-value:", p_value)
```

```
Z-statistic: 1.4142135623730951
P-value: 0.0786496035251425
```

Q.No.16 Create a confidence interval for a dataset using Python and interpret the result .
Ans-

```
[ ]  sample = norm.rvs(loc=100, scale=10, size=50)
     mean = np.mean(sample)
     std = np.std(sample, ddof=1)
     z_critical = norm.ppf(0.975)
     margin = z_critical * (std / np.sqrt(len(sample)))
     ci = (mean - margin, mean + margin)

     print("95% Confidence Interval:", ci)
```

```
95% Confidence Interval: (np.float64(97.51411360103526), np.float64(102.59099038451413))
```

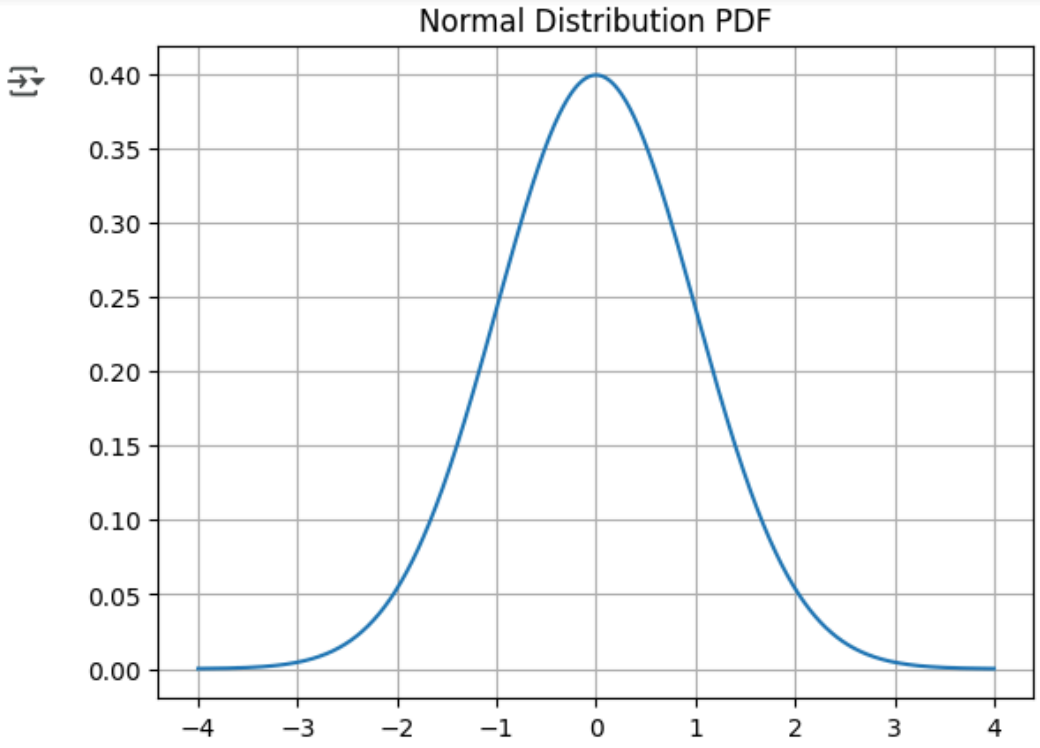
Q.No.17 Generate data from a normal distribution, then calculate and interpret the confidence interval for its mean .
Ans-

```
[ ]  data = norm.rvs(loc=50, scale=10, size=100)
     mean = np.mean(data)
     std = np.std(data, ddof=1)
     z_critical = norm.ppf(0.975)
     margin = z_critical * (std / np.sqrt(len(data)))
     print("Confidence Interval:", (mean - margin, mean + margin))
```

```
Confidence Interval: (np.float64(48.81381243949354), np.float64(53.12814813159061))
```

Q.No.18 Write a Python script to calculate and visualize the probability density function (PDF) of a normal distribution.
Ans-

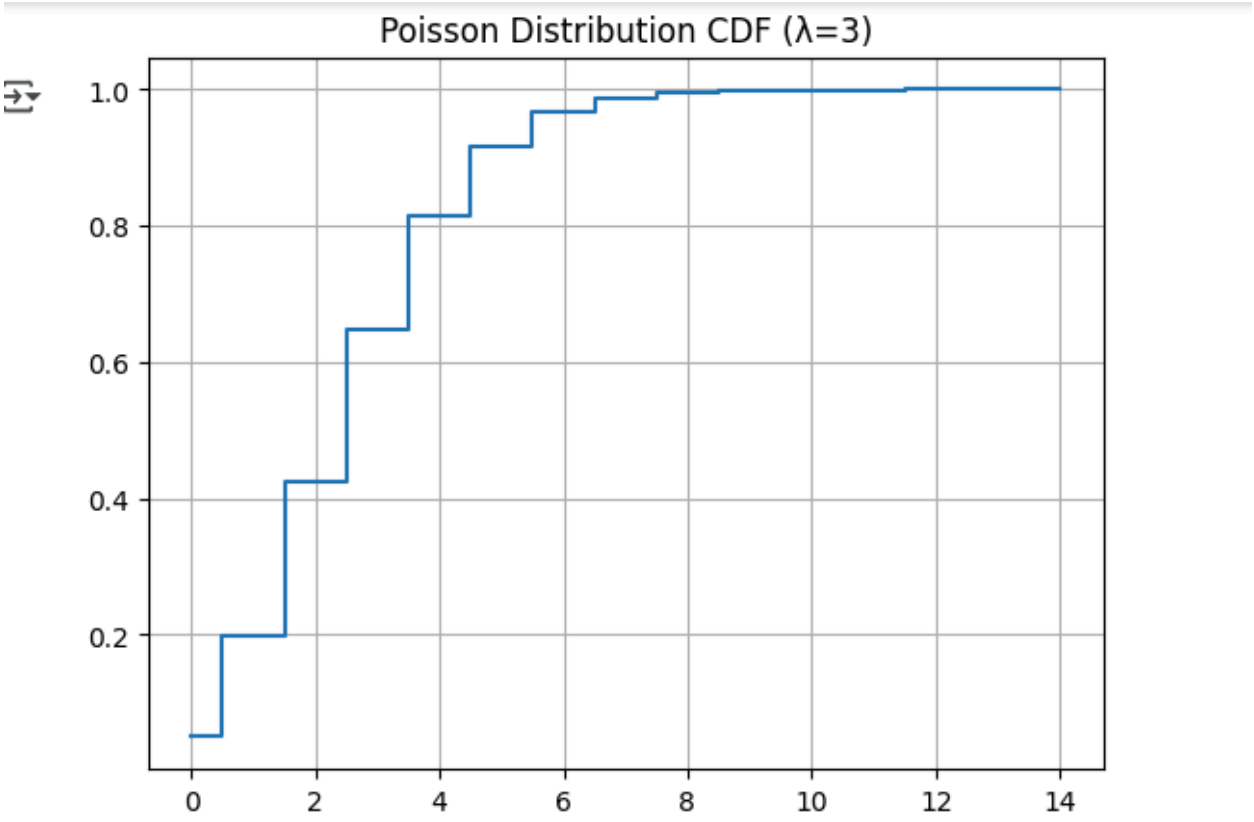
```
[ ]  x = np.linspace(-4, 4, 1000)
     pdf = norm.pdf(x, loc=0, scale=1)
     plt.plot(x, pdf)
     plt.title("Normal Distribution PDF")
     plt.grid(True)
     plt.show()
```



Q.No.19 Use Python to calculate and interpret the cumulative distribution function (CDF) of a Poisson distribution.

Ans-

```
[ ]  x = np.arange(0, 15)
     cdf = poisson.cdf(x, mu=3)
     plt.step(x, cdf, where='mid')
     plt.title("Poisson Distribution CDF (λ=3)")
     plt.grid(True)
     plt.show()
```



Q.No.20 Simulate a random variable using a continuous uniform distribution and calculate its expected value.
Ans-

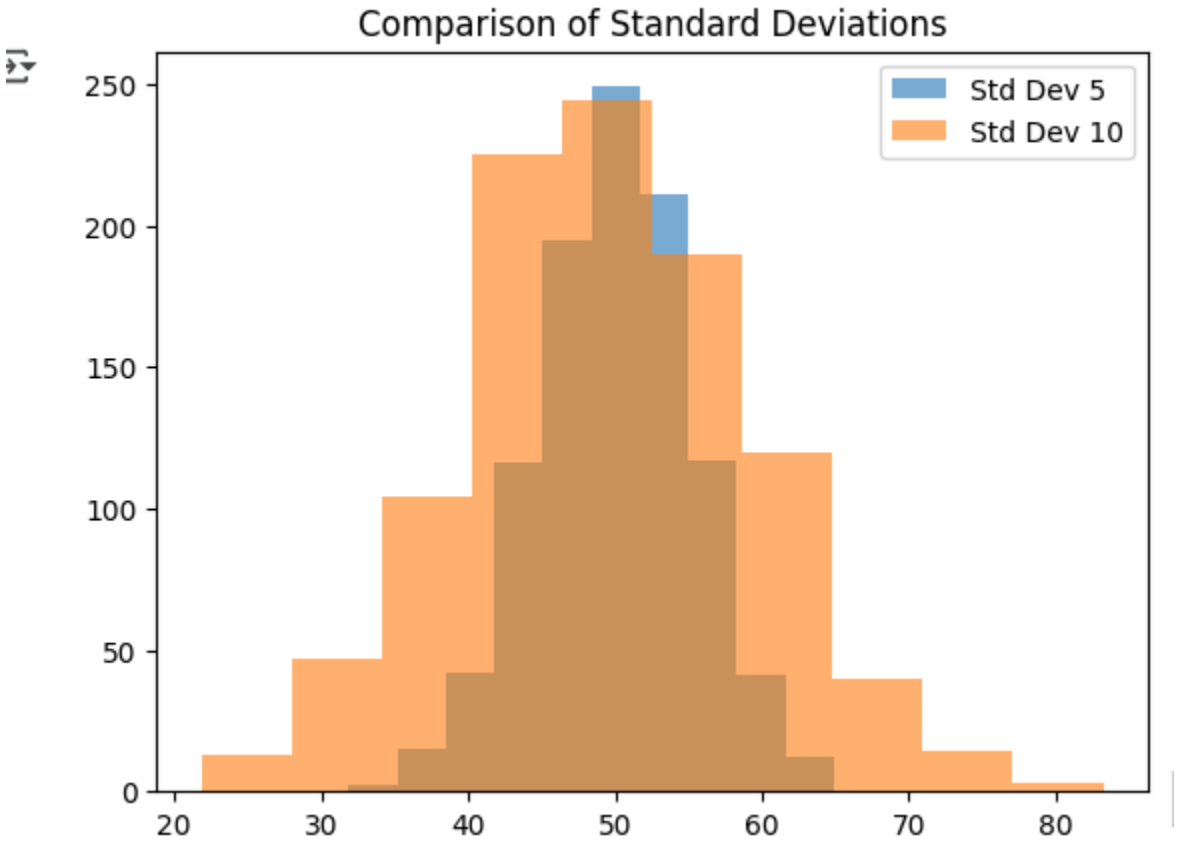
```
a, b = 5, 15
data = uniform.rvs(loc=a, scale=b-a, size=1000)
expected = (a + b) / 2
print("Expected Value:", expected)
```

```
Expected Value: 10.0
```

Q.No.21 Write a Python program to compare the standard deviations of two datasets and visualize the difference.
Ans-

```
[ ]  data1 = np.random.normal(50, 5, 1000)
     data2 = np.random.normal(50, 10, 1000)
     print("Std Dev of Data 1:", np.std(data1))
     print("Std Dev of Data 2:", np.std(data2))

     plt.hist(data1, alpha=0.6, label='Std Dev 5')
     plt.hist(data2, alpha=0.6, label='Std Dev 10')
     plt.legend()
     plt.title("Comparison of Standard Deviations")
     plt.show()
```

```
Std Dev of Data 1: 5.237641900916251
Std Dev of Data 2: 9.721674567236205
```



Q.No.22 Calculate the range and interquartile range (IQR) of a dataset generated from a normal distribution.

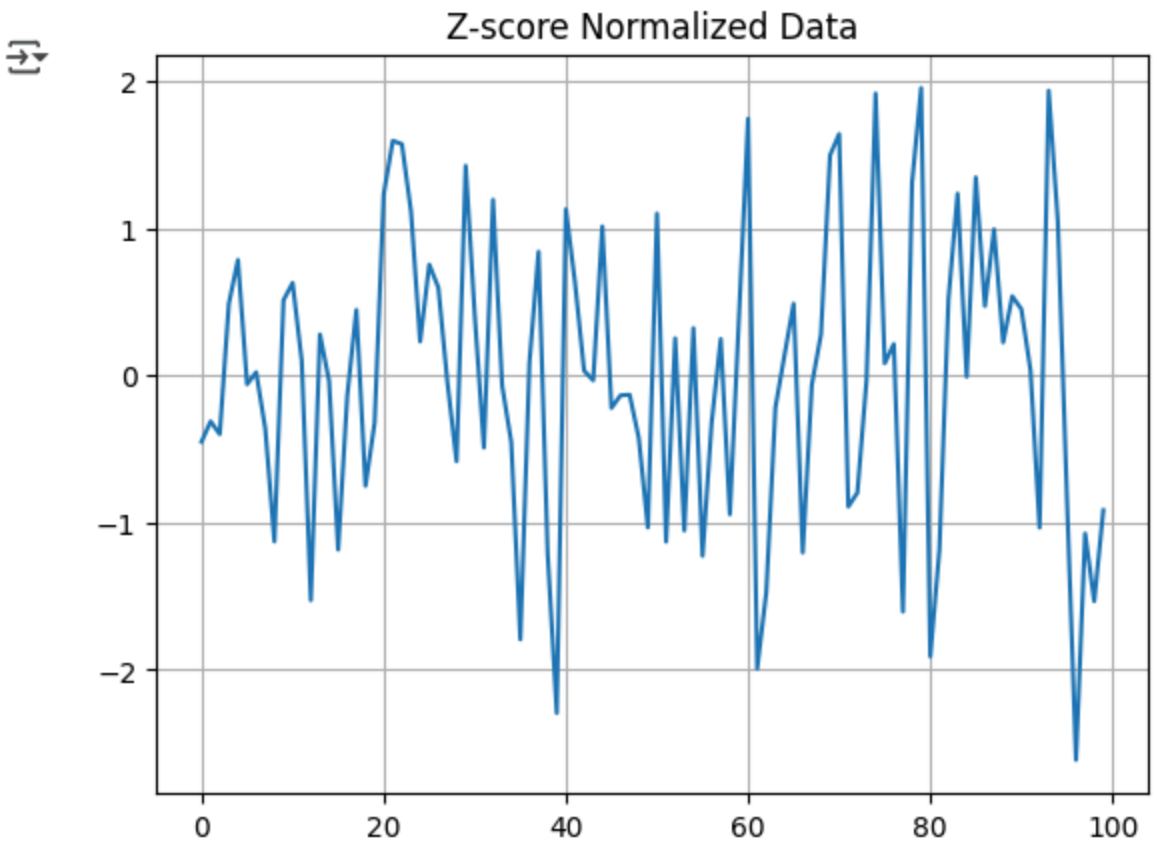
Ans-

```
[ ]  from scipy.stats import iqr
     data = np.random.normal(0, 1, 1000)
     range_val = np.ptp(data)
     iqr_val = iqr(data)
     print("Range:", range_val)
     print("Interquartile Range (IQR):", iqr_val)
```

```
Range: 6.616240275381009
Interquartile Range (IQR): 1.2836377492966502
```

Q.No.23 Implement Z-score normalization on a dataset and visualize its transformation.
Ans-

```
[ ]  data = np.random.normal(50, 10, 100)
     z_data = zscore(data)
     plt.plot(z_data)
     plt.title("Z-score Normalized Data")
     plt.grid(True)
     plt.show()
```



Q.No.24 Write a Python function to calculate the skewness and kurtosis of a dataset generated from a normal distribution.

Ans-

```
[ ]  from scipy.stats import skew, kurtosis
     data = np.random.normal(0, 1, 1000)
     print("Skewness:", skew(data))
     print("Kurtosis:", kurtosis(data))
```

```
Skewness: 0.021188557512549624
Kurtosis: 0.018899745318462724
```