Assignment No. 4

Problem Statement: To implement and visualize a normal distribution using Python for a dataset with mean = 100, standard deviation = 4, and dataset size = 100,000.

Objective:

To explore and understand the characteristics of the normal (Gaussian) distribution through simulation using Python. This includes generating normally distributed data, visualizing it, standardizing it with z-scores, and solving practical problems such as calculating probabilities. Additionally, tests like QQ plots, Shapiro-Wilk, and Kolmogorov-Smirnov are used to assess normality, with a focus on real-world applications such as height distributions.

Prerequisite :

1. Basic understanding of Python programming.
2. Understanding of libraries like Pandas, NumPy, Matplotlib, and Seaborn.
3. Familiarity with statistics, particularly normal distribution.
4. Knowledge of libraries such as NumPy and Matplotlib for data generation and visualization

Theory :

Normal Distribution:

The Normal or Gaussian Distribution is a continuous probability distribution , commonly referred to as a Probability Density Function (pdf). It is a probability distribution that is symmetric around its mean.

- **Mean (μ)** is the center of the distribution.
- **Standard deviation (σ)** measures the spread of the distribution.

The normal distribution represents the behaviour of many common random variables like **height/weight of people, marks scored by a students on an exam, performance rating of employees**, etc. The probability distribution of these random variables would resemble a bell curve because majority of the values would be clustered along the average or mean value. The number of values towards the upper or lower extreme would be very few in any given random sample.

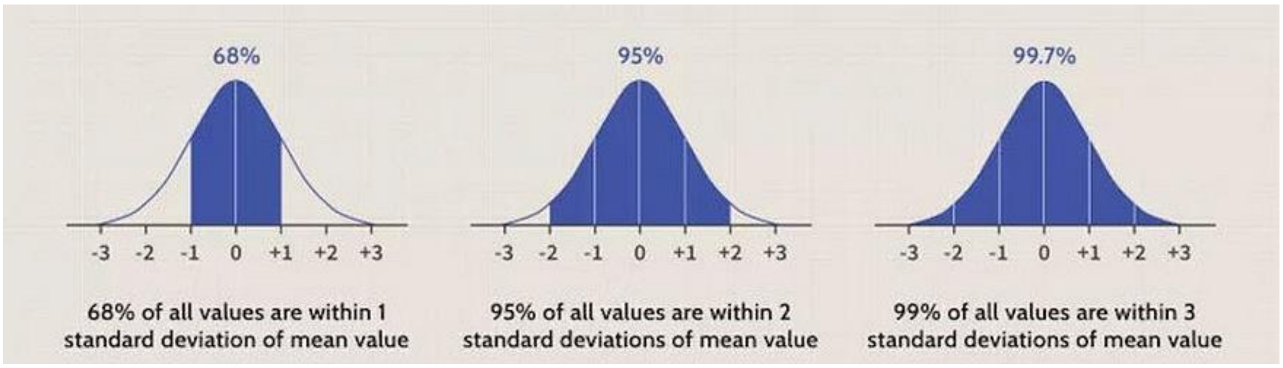
$$f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Where:

- **x** is the variable
- **μ** is the mean
- **σ** is the standard deviation
- **π** is a constant (approximately 3.14)
- **e** is Euler's number (approximately 2.718)

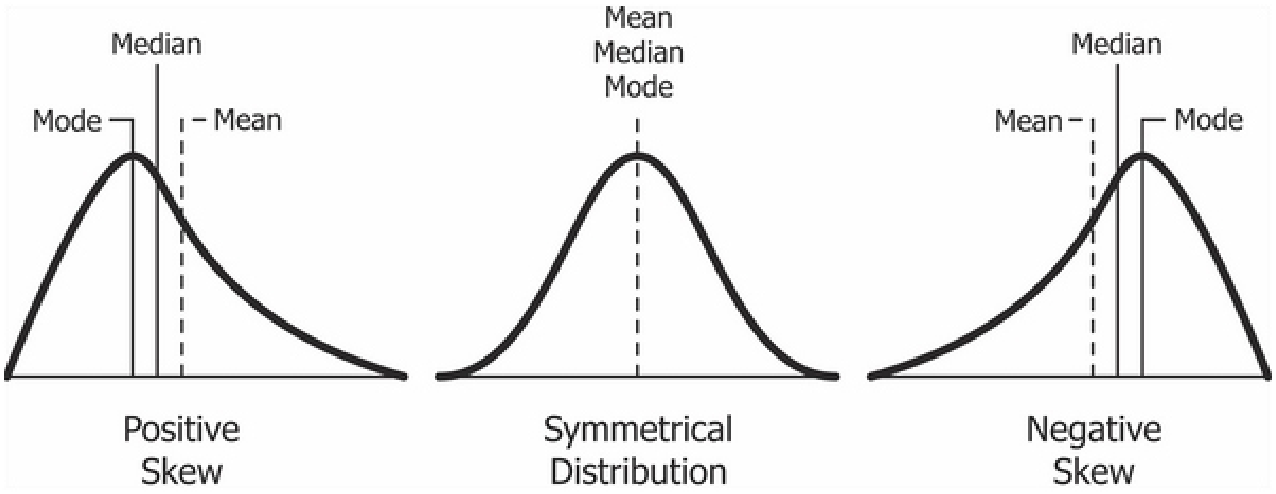
Properties of a Normal Distribution:

1. It is shaped like a bell curve and symmetric about its mean.
2. Since it is symmetric about the mean, which is the peak of the curve (50% values less than mean and 50% values greater than mean), its mode and median are equal to its mean.
3. The **empirical rule** or the 68–95–99 rule:
    - Around 68% of values lie within 1 standard deviation from the mean
    - Around 95% values lie between 2 standard deviations from the mean
    - Around 99.7% values lie between 3 standard deviations from the mean

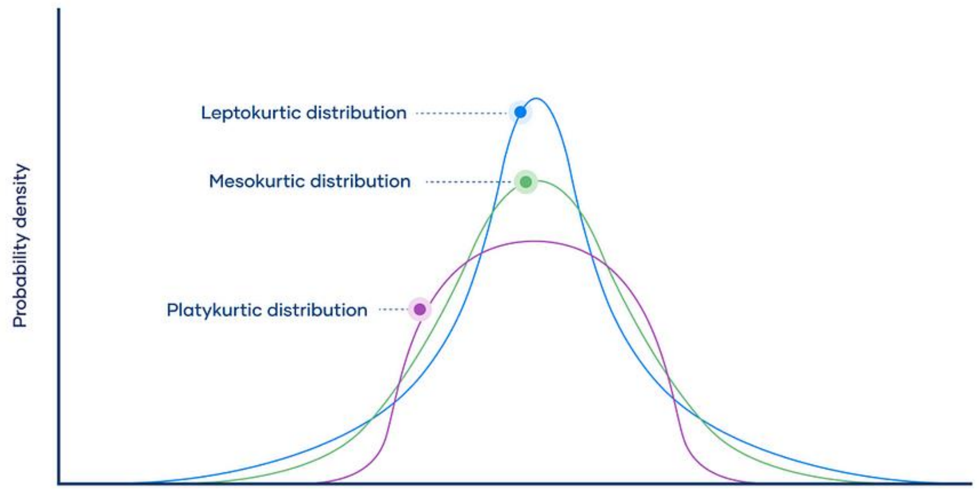


4. **Mean and standard deviation:** The mean shifts the normal distribution curve left or right. An increase in standard deviation makes the curve wider and flatter, while a decrease makes it narrower and steeper.

5. **Skewness:** Skewness measures asymmetry. A skewness of 0 indicates symmetry (normal distribution). Negative skew shifts the mean left, while positive skew shifts it right.



6. **Kurtosis:** Kurtosis measures tail heaviness. A value near 3 indicates normal distribution. Positive kurtosis (leptokurtic) means fatter tails, while negative kurtosis (platykurtic) means thinner tails.
   A distribution with kurtosis > 3 is called Leptokurtic and a distribution with kurtosis < 3 is called Platykurtic.



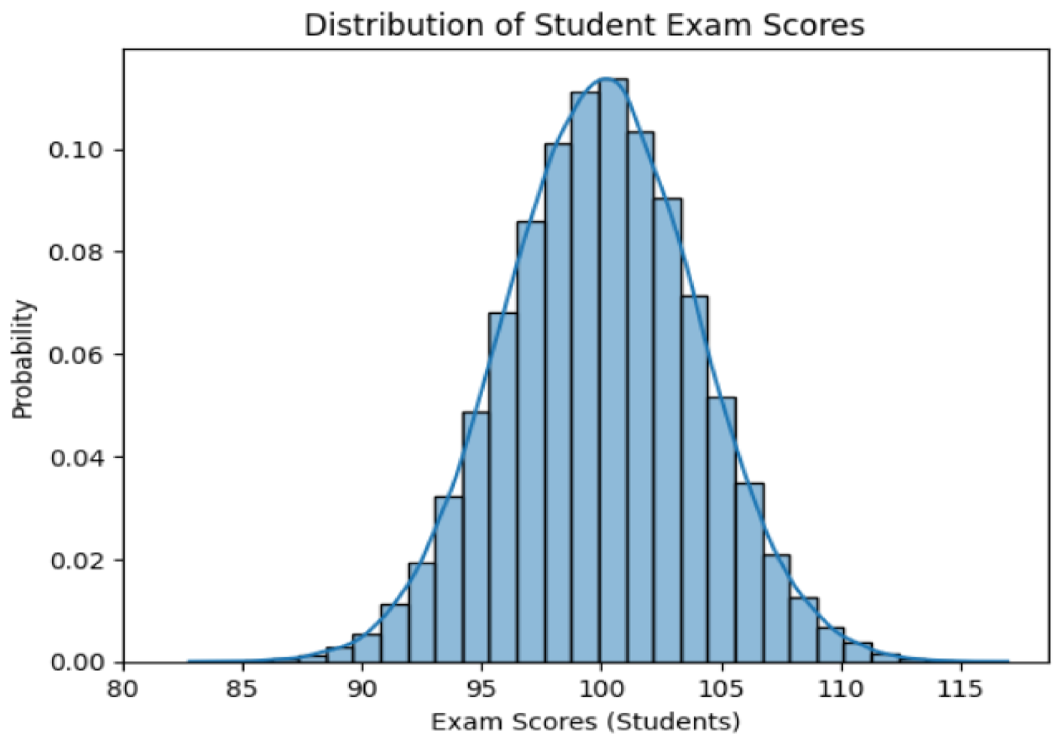
Standard Normal Distribution:

A standard normal distribution has a mean of 0 and a standard deviation of 1. Any normal distribution can be converted to this form by calculating the z-score for each value.
The z-score indicates how many standard deviations a value is from the mean.
Calculated as: (value - mean) / standard deviation.

$$z = \frac{x - \mu}{\sigma}$$

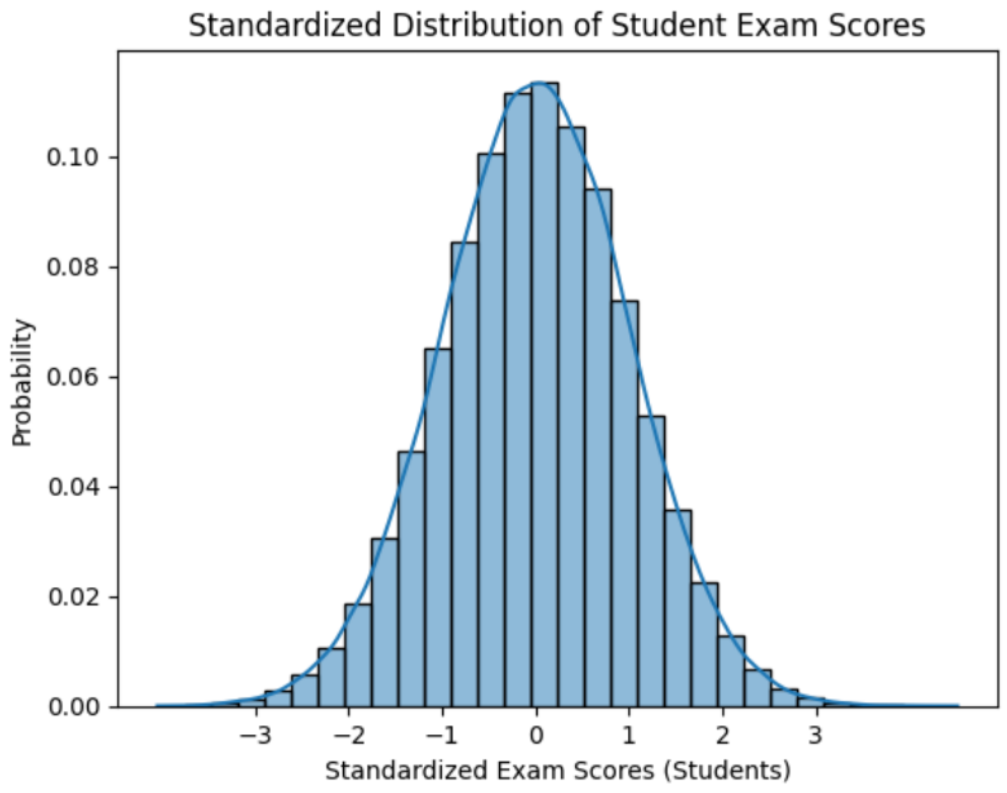
Code & Output:

```python
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Parameters for student exam scores
mu, sigma = 100, 4
data = np.random.normal(mu, sigma, 100000)
# Plotting the distribution
sns.histplot(data, bins=30, kde=True, stat='probability')
plt.xlabel('Exam Scores (Students)')
plt.ylabel('Probability')
plt.title("Distribution of Student Exam Scores")
plt.xticks(range(80, 120, 5))
plt.show()
```



```python
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
# Generating normally distributed data for Student Exam Scores
mu, sigma = 100, 4
data = np.random.normal(mu, sigma, 100000)

# Standardizing the data (finding z-scores)
data = (data - mu) / sigma

# Empirical Rule percentages
one_sd = len(data[(data > -1) & (data < 1)]) / len(data) * 100
two_sd = len(data[(data > -2) & (data < 2)]) / len(data) * 100
three_sd = len(data[(data > -3) & (data < 3)]) / len(data) * 100

print('Percentage of data within one standard deviation:', round(one_sd, 2))
print('Percentage of data within two standard deviations:', round(two_sd, 2))
print('Percentage of data within three standard deviations:', round(three_sd, 2))

# Plotting the standardized data
sns.histplot(data, bins=30, kde=True, stat='probability')
plt.xlabel('Standardized Exam Scores (Students)')
plt.ylabel('Probability')
plt.xticks(range(-3, 4, 1))
plt.title("Standardized Distribution of Student Exam Scores")
plt.show()
```

```
Percentage of data within one standard deviation: 68.56
Percentage of data within two standard deviations: 95.48
Percentage of data within three standard deviations: 99.72
```



```python
from scipy.stats import norm

# Parameters for student exam scores
mu, sigma = 100, 4

# Question 1: What % of students scored less than 102?
z_102 = (102 - mu) / sigma
p_102 = norm.cdf(z_102)
print(f'Percentage of students scoring less than 102: {round(p_102 * 100, 2)}%')
```

```
Percentage of students scoring less than 102: 69.15%
```

```python
# Question 2: What % of students scored less than 105?
z_105 = (105 - mu) / sigma
p_105 = norm.cdf(z_105)
print(f'Percentage of students scoring less than 105: {round(p_105 * 100, 2)}%')
```

```
Percentage of students scoring less than 105: 89.44%
```

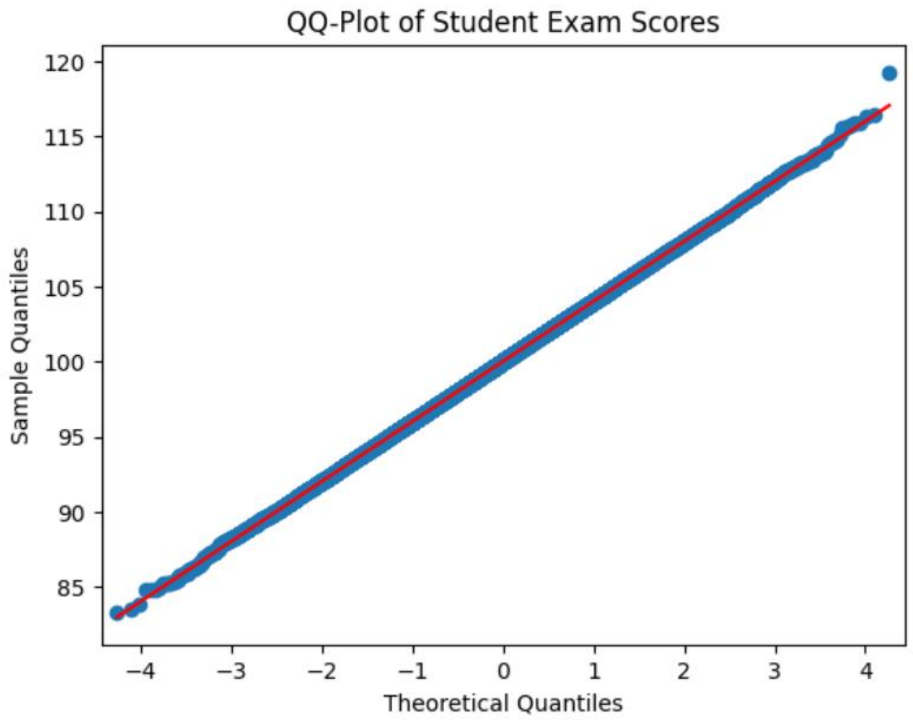
```python
# Question 3: What is the probability of scoring more than 98?
z_98 = (98 - mu) / sigma
p_98 = 1 - norm.cdf(z_98)
print(f'Percentage of students scoring more than 98: {round(p_98 * 100, 2)}%')
```

```
Percentage of students scoring more than 98: 69.15%
```

```python
# Question 4: What score corresponds to the 80th percentile?
z_80th = norm.ppf(0.8)
score_80th = z_80th * sigma + mu
print(f'Score corresponding to the 80th percentile: {round(score_80th, 2)}')
```

```
Score corresponding to the 80th percentile: 103.37
```

```python
import statsmodels.api as sm
import matplotlib.pyplot as plt
import numpy as np

# Generating normally distributed data for Student Exam Scores
mu, sigma = 100, 4
data = np.random.normal(mu, sigma, 100000)

# QQPlot
fig = sm.qqplot(data, line='s')
plt.title('QQ-Plot of Student Exam Scores')
plt.show()
```



```python
import numpy as np
from scipy.stats import norm, shapiro, kstest
# Generating normally distributed data for Student Exam Scores
mu, sigma = 100, 4
data = np.random.normal(mu, sigma, 1000)

# Shapiro-Wilk Test for Normality
test_stat, p_value = shapiro(data)
print('Result of Shapiro Test:')
if p_value < 0.10:
    print('Reject H0: Data is not Gaussian')
else:
    print('Fail to reject H0: Data is Gaussian')

# Kolmogorov-Smirnov Test for Normality
test_stat, p_value = kstest(data, norm.cdf, args=(data.mean(), data.std()))
print('-' * 20, '\nResult of KS Test:')
if p_value < 0.10:
    print('Reject H0: Data is not Gaussian')
else:
    print('Fail to reject H0: Data is Gaussian')
```

```
Result of Shapiro Test:
Fail to reject H0: Data is Gaussian
--------------------
Result of KS Test:
Fail to reject H0: Data is Gaussian
```

References :

https://medium.com/@snehabajaj108/the-normal-distribution-with-python-2cb3bf57ee47
https://www.geeksforgeeks.org/python-normal-distribution-in-statistics/
https://www.analyticsvidhya.com/blog/2020/04/statistics-data-science-normal-distribution/
https://365datascience.com/tutorials/statistics-tutorials/normal-distribution/

Github- https://github.com/dnyaneshwardhere/FODS

Conclusion :

Through Python simulation, we explored the key properties of the normal distribution, applied z-scores for standardization, and solved probability problems. Using tools like the Central Limit Theorem and normality tests, we confirmed its widespread relevance in statistics and real-world applications.