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 Distribution**, also known as the **Gaussian Distribution**, is a continuous **probability** distribution often referred to as a Probability Density Function (pdf). It is characterized by its symmetry around the mean.

- The mean (μ) serves as the midpoint of the distribution.
- The standard deviation (σ) indicates how spread out the values are from the mean.

This distribution is frequently applied to model real-world phenomena like the lifespans of electronic devices, the time taken for a machine to complete a task, or the daily temperatures in a region. It typically forms a bell-shaped curve, where most values cluster near the mean. Fewer data points occur at the higher and lower ends, creating thin tails at both extremes of the curve

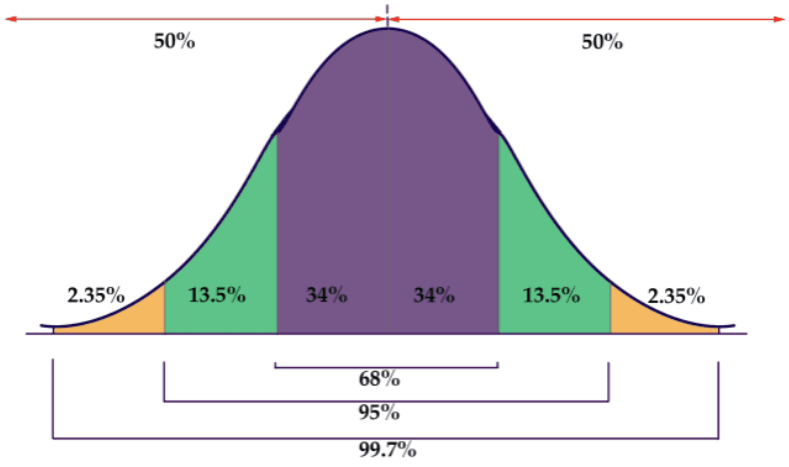
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Where:

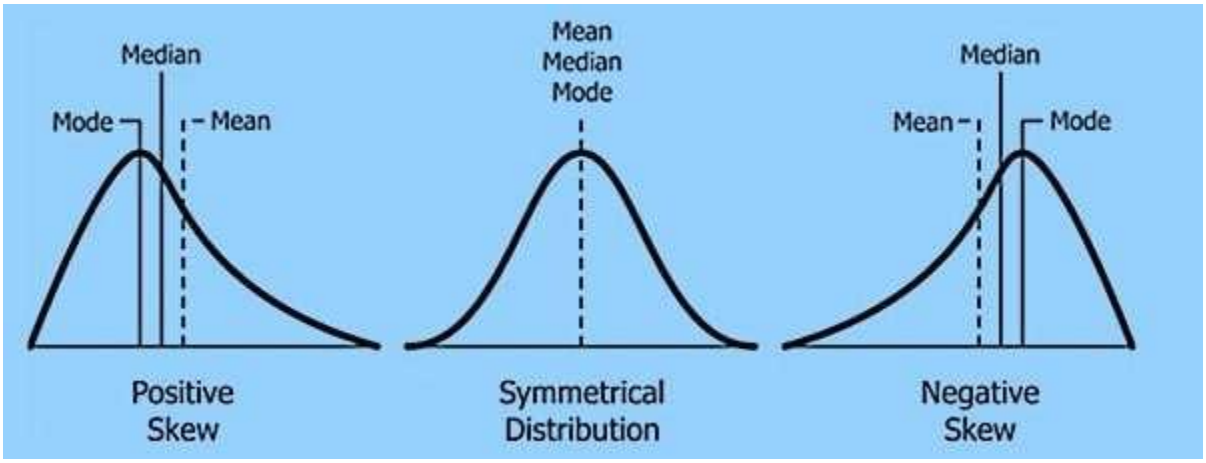
- **x** is the variable
- **μ** is the mean
- **σ** is the standard deviation
- **π** is a constant (approximately 3.14)
- **e** is Euler's number (approximately 2.718)

Properties of the Normal Distribution:

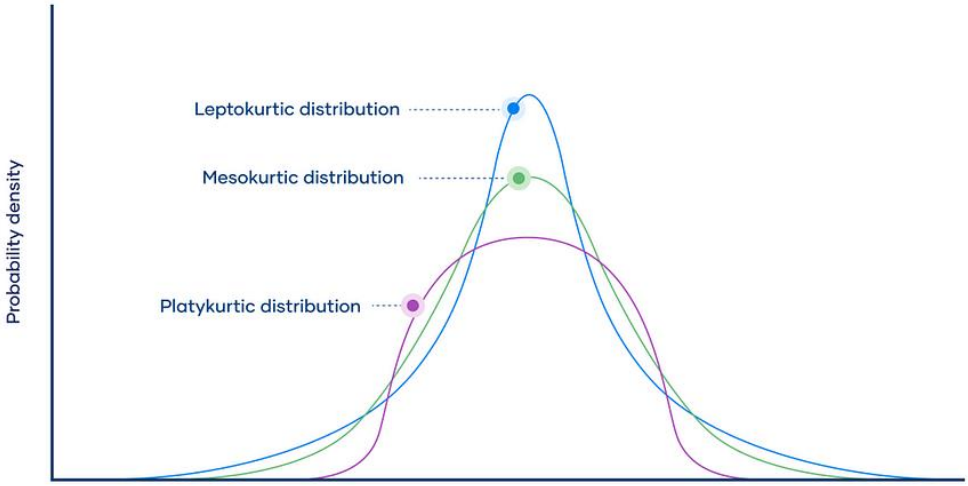
1. It has a bell-shaped curve and is symmetric around its mean.
2. Due to this symmetry, the mean is the highest point on the curve, with 50% of the data points falling below the mean and 50% above it. This makes the mean, median, and mode all equal.
3. The empirical rule, also known as the 68–95–99.7 rule:
   - About 68% of data lies within one standard deviation of the mean.
   - Around 95% of values fall within two standard deviations.
   - Roughly 99.7% of values are found within three standard deviations from the mean.



4. **Mean and Standard Deviation:** The mean determines the position of the normal distribution curve, shifting it left or right along the x-axis. An increase in standard deviation causes the curve to widen and flatten, while a decrease makes the curve narrower and taller.
5. **Skewness:** Skewness measures the asymmetry of the distribution. A skewness of 0 indicates a perfectly symmetric, normal distribution. Negative skew pulls the mean to the left, whereas positive skew pushes it to the right.



6. **Kurtosis:** Kurtosis quantifies the thickness of a distribution's tails. A value close to 3 signifies a normal distribution. Positive kurtosis (leptokurtic) indicates heavier tails, meaning more extreme values, while negative kurtosis (platykurtic) reflects lighter tails with fewer extreme values. A kurtosis greater than 3 is termed leptokurtic, and a value less than 3 is referred to as platykurtic.

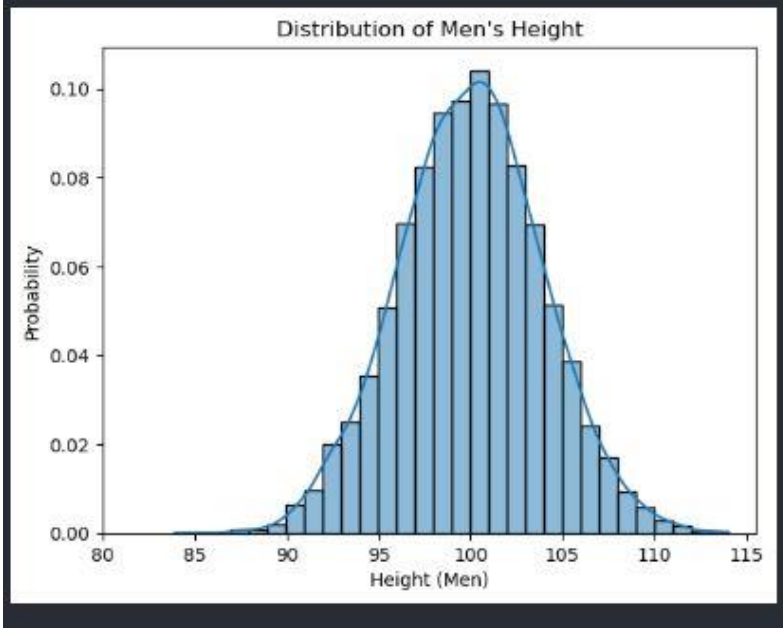


Standard Normal Distribution: A standard normal distribution has a mean of 0 and a standard deviation of 1. Any normal distribution can be transformed into this format by determining the z-score for each data point. The z-score represents how far a particular value is from the mean, measured in terms of standard deviations. It is calculated using the formula:

$$\text{z-score} = \frac{\text{value} - \text{mean}}{\text{standard deviation}}$$

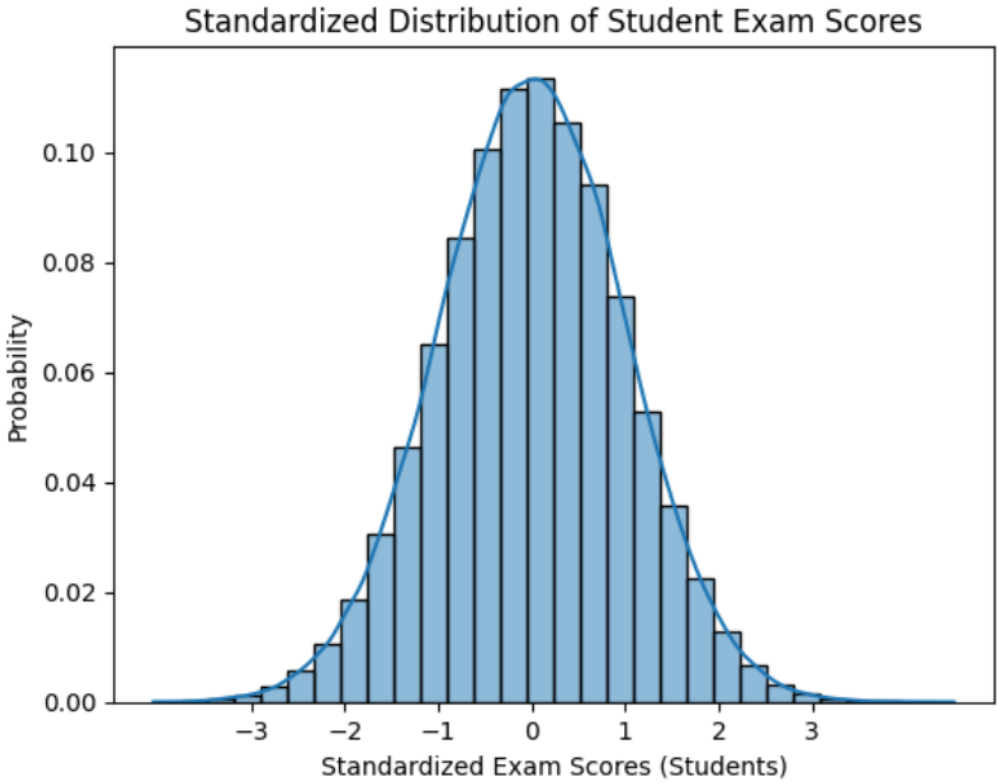
Code & Output:

```python
import matplotlib.pyplot as plt  # Importing the Matplotlib library for plotting
import seaborn as sns            # Importing the Seaborn library for enhanced data visualization
import numpy as np               # Importing NumPy for numerical operations

mu, sigma = 100, 4               # Defining the mean (mu) and standard deviation (sigma) for the normal distribution
data = np.random.normal(mu, sigma, 10000)  # Generating 10,000 random samples from a normal distribution

# Creating a histogram with a kernel density estimate (KDE) overlay
sns.histplot(data, bins=30, kde=True, stat='probability')
plt.xlabel('Height (Men)')       # Labeling the x-axis as 'Height (Men)'
plt.ylabel('Probability')        # Labeling the y-axis as 'Probability'
plt.title("Distribution of Men's Height")  # Setting the title of the plot
plt.xticks(range(80, 120, 5))    # Setting the x-ticks to range from 80 to 120 with a step of 5
plt.show()                       # Displaying the plot
```



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

#Generating normally distributed data for Height of Men
mu, sigma = 100,4
data = np.random.normal(mu, sigma, 10000)

#Finding the z-score for every value in the data
data = (data - mu)/sigma

#Proving the empirical rule
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 deviation:', round(two_sd,2))
print('Percentage of data within three standard deviation:', round(three_sd,2))

#Plotting the z-scores
sns.histplot(data, bins=30, kde=True, stat='probability')
plt.xlabel('Height (Men)')
plt.ylabel('Probability')
plt.xticks(range(-3,4,1))
plt.show()
```

```
Percentage of data within one standard deviation: 68.36
Percentage of data within two standard deviation: 95.54
Percentage of data within three standard deviation: 99.75
```



```
from scipy.stats import norm

mu = 100
sigma = 4

# Change the value to something closer to the mean
x = 110  # New value

# Find the z-score
z = (x - mu) / sigma

# Find the probability (area under the curve until x)
p = norm.cdf(z)

# Print the result
print('Percentage of men shorter than', x, 'cm:', round(p * 100, 2))
```

```
Percentage of men shorter than 110 cm: 99.38
```

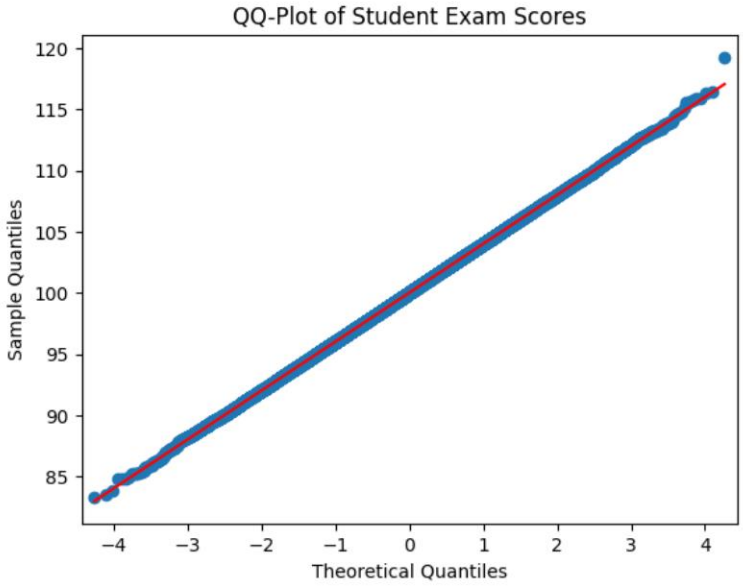
```
#Find the z-score
z = (100-mu)/sigma

#Find the probability (1 - area under the curve until 177 cm)
p = 1 - norm.cdf(z)

print('Percentage of men taller than 100 cm:', round(p*100,2))
```

```
Percentage of men taller than 100 cm: 50.0
```

```
#Find the z-score for 80th percentile
z = norm.ppf(0.85) #percent point function, gives percentile
x = (z*sigma) + mu #find the value using z-score formula

print("The height for which 85% men are shorter:", round(x,2))
```

```
The height for which 85% men are shorter: 104.15
```

```
#QQPlot

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

#Generating normally distributed data for height of men
mu, sigma = 100, 4
data = np.random.normal(mu, sigma, 10000)

#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
from scipy.stats import shapiro, kstest

# Generating normally distributed data for height of men
mu, sigma = 100, 4
data = np.random.normal(mu, sigma, 10000)

# H0: Data is Gaussian
# Ha: Data is not Gaussian

# Shapiro Test for Normality
test_stat, p_value = shapiro(data)

print('Result of Shapiro Test:')
if p_value < 0.05:
    print('Reject H0')
    print('Data is not Gaussian')
else:
    print('Fail to reject H0')
    print('Data is Gaussian')

# Kolmogorov-Smirnov Test for Normality
# Specify the mean and std for the normal distribution
test_stat, p_value = kstest(data, 'norm', args=(mu, sigma))

print('-' * 20, '\nResult of KS Test:')
if p_value < 0.05:
    print('Reject H0')
    print('Data is not Gaussian')
else:
    print('Fail to reject H0')
    print('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/Vishalgodalkar/Data-Science

Conclusion :

Through Python simulation, we examined the fundamental characteristics of the normal distribution, utilized z-scores for standardization, and addressed probability problems. By employing concepts such as the Central Limit Theorem and conducting normality tests, we validated the normal distribution's extensive significance in statistics and its practical applications in the real world.