Question Bank : Unit 1

1. Line Plot

Question: Write a Python script to plot the population growth of a country over the years using a line plot. The years are from 2010 to 2020, and the population data is given as:

Years: [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Population (in millions): [2.5, 2.6, 2.7, 2.8, 2.85, 2.9, 3.0, 3.1, 3.2, 3.25, 3.3]
Include appropriate labels for the X and Y axes, title, and grid lines for better readability.

```
import matplotlib.pyplot as plt
years = [2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
population = [2.5, 2.6, 2.7, 2.8, 2.85, 2.9, 3.0, 3.1, 3.2, 3.25, 3.3]

plt.plot(years, population, marker='o', linestyle='-', color='b')
plt.xlabel('Years')
plt.ylabel('Population (in millions)')
plt.title('Population Growth Over the Years (2010-2020)')
plt.grid(True)
plt.show()
```

2. Bar Chart

Question: Write a Python script to create a bar chart showing the sales of different products in a store. The products and their corresponding sales are as follows:

Products: ['A', 'B', 'C', 'D', 'E']
Sales: [100, 150, 90, 200, 130]
Ensure that each bar has a unique color, label the X-axis as 'Products', the Y-axis as 'Sales', and give the chart a title.

```
products = ['A', 'B', 'C', 'D', 'E']
sales = [100, 150, 90, 200, 130]

plt.bar(products, sales, color=['red', 'blue', 'green', 'purple', 'orange'])
plt.xlabel('Products')
plt.ylabel('Sales')
plt.title('Product Sales in Store')
plt.show()
```

3. Histogram

Question: Write a Python script to plot a histogram of student scores in a test. The scores are as follows:

Scores: [88, 92, 100, 67, 85, 79, 95, 80, 78, 90, 85, 99, 76, 65, 70, 91, 87, 82]
Set the number of bins to 5 and make sure to label the X-axis as 'Scores' and the Y-axis as 'Number of Students'. Give the plot an appropriate title.

```
scores = [88, 92, 100, 67, 85, 79, 95, 80, 78, 90, 85, 99, 76, 65, 70, 91, 87, 82]

plt.hist(scores, bins=5, edgecolor='black', color='skyblue')
plt.xlabel('Scores')
plt.ylabel('Number of Students')
plt.title('Distribution of Student Scores')
plt.show()
```

Histogram with Bins Concept

4.Question: Write a Python script to create a histogram that shows the distribution of ages in a group of 40 people. The age data is as follows:

Ages: [22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 30, 40, 45, 55, 60]
Create a histogram with 5 bins. Label the X-axis as 'Ages' and the Y-axis as 'Number of People'. Add a grid to the plot and give it an appropriate title.

```
ages = [22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22,
25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 22, 30, 40, 45, 55, 60]

plt.hist(ages, bins=5, edgecolor='black', color='lightblue')
plt.xlabel('Ages')
plt.ylabel('Number of People')
plt.title('Age Distribution in a Group of People')
plt.grid(True)
plt.show()
```

5. Scatter Plot

Question: Write a Python script to create a scatter plot for the relationship between the age of employees and their salaries in a company. The data is given as:

Ages: [22, 25, 26, 28, 30, 32, 34, 36, 40, 45, 50]
Salaries (in $1000s): [35, 38, 40, 50, 60, 70, 65, 75, 90, 100, 110]
Include appropriate labels for the X and Y axes, and a title. Use a different color for each point.

```
import matplotlib.pyplot as plt

ages = [22, 25, 26, 28, 30, 32, 34, 36, 40, 45, 50]
salaries = [35, 38, 40, 50, 60, 70, 65, 75, 90, 100, 110]

plt.scatter(ages, salaries, color='green')
plt.xlabel('Age')
plt.ylabel('Salary (in $1000s)')
plt.title('Age vs Salary of Employees')
plt.show()
```

6. Pie Chart
Question: Write a Python script to create a pie chart representing the market share of different companies in a specific industry. The companies and their market shares are given as:

Companies: ['Company A', 'Company B', 'Company C', 'Company D']
Market Shares (in %): [25, 30, 20, 25]
Make sure to label each slice with the percentage, and give the chart a title.

```
import matplotlib.pyplot as plt
companies = ['Company A', 'Company B', 'Company C', 'Company D']
market_shares = [25, 30, 20, 25]

plt.pie(market_shares, labels=companies, autopct='%1.1f%%', startangle=90, colors=['gold',
'lightblue', 'lightgreen', 'coral'])
plt.title('Market Share of Companies')
plt.show()
```

7. Box Plot

Question: Write a Python script to create a box plot for the test scores of students in three different subjects: Math, Science, and English. The scores are as follows:

Math: [88, 92, 75, 85, 90, 95, 89, 91, 76, 85]
Science: [80, 85, 88, 92, 91, 87, 90, 95, 89, 88]
English: [78, 82, 80, 85, 88, 90, 91, 86, 89, 92]
Plot a box plot for each subject, and label the X-axis as 'Subjects' and the Y-axis as 'Scores'. Give the plot a title and display the outliers, if any.

```
import matplotlib.pyplot as plt

# Data


math_scores = [88, 92, 75, 85, 90, 95, 89, 91, 76, 85]
science_scores = [80, 85, 88, 92, 91, 87, 90, 95, 89, 88]
english_scores = [78, 82, 80, 85, 88, 90, 91, 86, 89, 92]

# Combine data
data = [math_scores, science_scores, english_scores]

# Box plot
plt.boxplot(data, patch_artist=True, labels=['Math', 'Science', 'English'])
plt.xlabel('Subjects')
plt.ylabel('Scores')
plt.title('Box Plot of Student Scores in Different Subjects')
plt.grid(True)
plt.show()
```

8. Heatmap

Question: Write a Python script to create a heatmap representing the performance of different salespersons over five months. The sales data is given as:

Salespersons: ['John', 'Sara', 'Mike', 'Linda', 'Paul']
Sales over months (in thousands of dollars) for each salesperson:
John: [35, 40, 50, 45, 48]
Sara: [40, 42, 48, 44, 47]
Mike: [38, 41, 45, 47, 46]
Linda: [30, 35, 42, 38, 40]
Paul: [50, 52, 53, 55, 58]
Create a heatmap of this sales data using matplotlib. Label the axes, and include a color bar to indicate the sales intensity.

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

# Data
salespersons = ['John', 'Sara', 'Mike', 'Linda', 'Paul']
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
sales_data = [
   [35, 40, 50, 45, 48],  # John
   [40, 42, 48, 44, 47],  # Sara
   [38, 41, 45, 47, 46],  # Mike
   [30, 35, 42, 38, 40],  # Linda
```

```
    [50, 52, 53, 55, 58],  # Paul
]

# Create heatmap
plt.imshow(sales_data,       cmap='coolwarm',       interpolation='nearest')
plt.colorbar(label='Sales                        (in                        $1000s)')
plt.xticks(ticks=np.arange(len(months)),                    labels=months)
plt.yticks(ticks=np.arange(len(salespersons)),          labels=salespersons)
plt.xlabel('Months')
plt.ylabel('Salespersons')
plt.title('Sales Performance Heatmap')
plt.show()
```

9. Contour Plot

Question: Write a Python script to create a contour plot of a function $Z=f(X,Y)$, where $Z=X2+Y2$. The X and Y values should range from -5 to 5. Generate a contour plot with 10 contour levels, and label both the axes and the contour lines.

Solution:

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

# Generate grid data
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = X**2 + Y**2  # Z = X^2 + Y^2

# Create contour plot
plt.contour(X, Y, Z, levels=10, cmap='viridis')
plt.colorbar(label='Z     =     X^2     +     Y^2')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Contour Plot of Z = X^2 + Y^2')
plt.clabel(plt.contour(X, Y, Z, levels=10, cmap='viridis'), inline=True, fontsize=8)
plt.show()
```

Question Bank : Unit 2

1.Calculates the mean, median, mode, and standard deviation for the following array using NumPy and SciPy:

arr = np.array([15, 18, 20, 18, 21, 17, 20, 19, 15, 16])

Explore the importance of each statistic and provide the output from your code.

```
import numpy as np
from scipy import stats

# Array provided
arr = np.array([15, 18, 20, 18, 21, 17, 20, 19, 15, 16])

# Calculating statistics
mean = np.mean(arr)
median = np.median(arr)
mode = stats.mode(arr).mode[0]
std_dev = np.std(arr)
```

2.Create a 2D NumPy array as follows:

arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
Calculate the mean, median, mode, and standard deviation of the array along both axes (rows and columns) using NumPy and SciPy. Provide explanations of the results obtained.

3.A dataset represents the daily temperatures of a city in degrees Celsius over the last 10 days:
temperatures = np.array([29.2, 31.5, 30.1, 32.0, 29.8, 33.1, 30.5, 32.5, 31.2, 29.9])
Calculate the mean, median, mode, and standard deviation of the temperatures. Elaborate how each of these statistics could help in understanding the climate pattern of the city.

4.Given the following 1D NumPy arrays representing the monthly sales of two different products in a store:
product_A = np.array([120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230])
product_B = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130, 135])

Compute the mean, median, mode, and standard deviation for both product_A and product_B. Compare and contrast the results for both products, explaining any insights that can be gained from the statistical analysis.

5.Suppose you have the following 2D array where some data points are missing (np.nan):
data = np.array([[1, 2, np.nan], [4, np.nan, 6], [7, 8, 9]])
Calculates the mean, median, mode, and standard deviation for this array, while handling missing values appropriately. Explain how missing data can affect statistical analysis and how it should be addressed.

6. Create a 1D NumPy array with 50 random integers between 10 and 100 using Python code
arr = np.random.randint(10, 100, size=50)
Using Python code calculate the mean, median, mode, and standard deviation of the array. Additionally, plot a histogram of the array to visualize its distribution. How do the statistical values relate to the shape of the distribution?

7. A NumPy array contains the ages of people in a group:
ages = np.array([18, 22, 25, 26, 35, 40, 45, 46, 50, 55, 60, 85])
Using Python program to calculate the mean, median, mode, and standard deviation of the ages. Identify and discuss any outliers in the dataset and explain how these outliers affect the statistics. What methods can be used to deal with outliers?

8. Two groups of students took the same test, and their scores are given by the following 1D arrays:
group_A = np.array([85, 87, 90, 92, 88, 86, 84, 91, 89, 93])
group_B = np.array([78, 85, 92, 90, 83, 87, 95, 70, 91, 88])

Using Python program to calculate the mean, median, mode, and standard deviation for both groups. Which group shows more consistency in test scores, and why? Explain the significance of the standard deviation in this context.

```
import numpy as np
```

```
from scipy import stats

# Data for the two groups
group_A = np.array([85, 87, 90, 92, 88, 86, 84, 91, 89, 93])
group_B = np.array([78, 85, 92, 90, 83, 87, 95, 70, 91, 88])

# Calculations for Group A
mean_A = np.mean(group_A)
median_A = np.median(group_A)
mode_A = stats.mode(group_A, keepdims=True)[0][0]  # Mode using scipy.stats
std_A = np.std(group_A)

# Calculations for Group B
mean_B = np.mean(group_B)
median_B = np.median(group_B)
mode_B = stats.mode(group_B, keepdims=True)[0][0]  # Mode using scipy.stats
std_B = np.std(group_B)

# Displaying the results
print("Group A Statistics:")
print(f"Mean: {mean_A}")
print(f"Median: {median_A}")
print(f"Mode: {mode_A}")
print(f"Standard Deviation: {std_A}\n")

print("Group B Statistics:")
print(f"Mean: {mean_B}")
print(f"Median: {median_B}")
print(f"Mode: {mode_B}")
print(f"Standard Deviation: {std_B}\n")

# Determining consistency
if std_A < std_B:
    print("Group A shows more consistency in test scores.")
elif std_A > std_B:
    print("Group B shows more consistency in test scores.")
else:
    print("Both groups have the same level of consistency in test scores.")
```

9.Using Python code take any 1D or 2D NumPy array as input and returns the mean, median, mode, and standard deviation for the array. Demonstrate the function with an example array of your choice, and explain the code.

```
import numpy as np
array_1d= np.array([10, 20, 20, 30, 40, 50])
array_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
flatten_array_2d = array_2d.flatten()
np.mean(array_1d), p.median(array_1d), p.mode(array_1d)
np.mean(array_flatten_array_2d),p.median(array_flatten_array_2d),
p.mode(array_flatten_array_2d)
```

10. Consider the following 1D array:
arr = np.array([1, 3, 5, 7, 9, 11, 13, 15])
Using Python program to calculate the mean, median, mode, and standard

deviation of the array. Now, apply a linear transformation to the array (arr_transformed = 2 * arr + 5). Recalculate the statistical values for the transformed array. Explain how transformations affect the statistical measures and why.

Pandas Questions:
Create a pandas Series as follows:

```
import pandas as pd
data = pd.Series([10, 12, 14, 13, 15, 16, 14, 18, 19, 20])
```

Write a Python program that calculates the mean, median, mode, standard deviation, and variance for this Series. Explain the importance of each statistical measure and provide the output.
Ans

```
mean = data.mean(), mode = data.mode() , median = data.median()
std_dev = data.std()
```

2. Basic Statistical Measures on a pandas DataFrame:
Given the following pandas DataFrame representing the ages and heights of 5 individuals:

```
import pandas as pd
data = {'Age': [23, 25, 31, 22, 35],  'Height': [167, 175, 180, 165, 170]}
```

Write a Python program to calculate the mean, median, mode, and standard deviation for each column in the DataFrame. Provide a brief explanation of the results obtained.

```
df = pd.DataFrame(data)
mean = df.mean(), mode = df.mode(), median = df.median(), std_dev = df.std()
print(mean)
print(mode)
print(median)
print(std_dev)
```

3. Given the following DataFrame representing the sales data for different products:

```
import pandas as pd
data = {'Product': ['A', 'B', 'A', 'B', 'A', 'B'], 'Sales': [100, 200, 150, 250, 130, 300]}
df = pd.DataFrame(data)
```

Write a Python program to group the data by the Product column and calculate the mean, median, and standard deviation of the Sales for each product. Explain how grouping the data can provide better insights into the sales performance of each product.

```
import pandas as pd
df = pd.DataFrame(data)

# Grouping the data by 'Product'
grouped = df.groupby('Product')['Sales']

# Calculating mean, median, and standard deviation for each group
stats = grouped.agg(['mean', 'median', 'std'])
# Displaying the results
print(stats)
```

Insights from Grouping:
Sales Performance Comparison:
Grouping helps break down the overall sales data into specific categories (in this case, by Product), allowing you to directly compare the performance of different products.
Understanding Distribution:
Calculating statistics like the mean and standard deviation helps to understand the central tendency and variability of sales for each product.
Strategic Decision-Making:
Products with higher variability (e.g., Product B with std=50.0) might need a more consistent sales strategy.
Products with lower mean sales (e.g., Product A with mean=126.67) might require additional promotions or marketing efforts.

4. Statistical Summary Using describe() in pandas:
Create a pandas DataFrame as follows:

```
import pandas as pd
data = {'Age': [23, 25, 31, 22, 35],
       'Height': [167, 175, 180, 165, 170],
       'Weight': [65, 70, 75, 68, 72]}
df = pd.DataFrame(data)
```

Write a Python program to use the describe() function in pandas to generate a statistical summary of the DataFrame. Explain the meaning of the statistics provided by describe() (e.g., mean, standard deviation, min, max, etc.).

```
df = pd.DataFrame(data) # Generating the statistical summary
summary = df.describe() # Displaying the summary print(summary)
print(summary)
```

5. Handling Missing Data in pandas DataFrame:
Consider the following pandas DataFrame with some missing data (NaN values):

```
import pandas as pd
data = {'Age': [23, 25, None, 22, 35], 'Height': [167, 175, 180, None, 170],
       'Weight': [65, 70, None, 68, 72]}
df = pd.DataFrame(data)
```

Write a Python program to calculate the mean, median, and standard deviation of each column, while appropriately handling the missing data (NaN). Explain how missing data can affect statistical analysis and what strategies can be used to handle it.

6. Comparing Statistics of Different DataFrames:
Create two DataFrames representing the monthly sales of two different regions:

```
df1 = pd.DataFrame({'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'], 'Sales': [200, 250,
300, 400,  500]})
df2 = pd.DataFrame({'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'],’Sales': [180, 220,
320, 410, 480]})
```

Write a Python program to calculate the mean, median, and standard deviation of the Sales in both regions. Compare the sales performance across the two regions using these statistics.

7. Column-wise and Row-wise Statistical Calculations:

Consider the following DataFrame representing the marks scored by 3 students in 4 subjects:

import pandas as pd
data = {'Math': [85, 92, 78], 'English': [88, 79, 85],'Science': [90, 87, 92],'History': [80, 70, 88]}
df = pd.DataFrame(data, index=['Student1', 'Student2', 'Student3'])
Write a Python program to calculate the mean, median, and standard deviation for each student (row-wise) and for each subject (column-wise). Explain the significance of calculating statistics both row-wise and column-wise in this context.
Ans

```
df = pd.DataFrame(data, index=['Student1', 'Student2', 'Student3'])
row_mean = df.mean(axis=1) # Mean for each student
row_median = df.median(axis=1) # Median for each student
row_std = df.std(axis=1) # Standard deviation for each student
# Calculating column-wise statistics (for each subject)
col_mean = df.mean(axis=0) # Mean for each subject
col_median = df.median(axis=0) # Median for each subject
col_std = df.std(axis=0) # Standard deviation for each subject
```

9. Custom Function for Statistical Measures in pandas:
Write a Python function that takes a pandas DataFrame as input and returns the mean, median, mode, and standard deviation for each column. Demonstrate the function using the following DataFrame and explain the results:
data = {'Age': [23, 25, 31, 22, 35], 'Height': [167, 175, 180, 165, 170],'Weight': [65, 70, 75, 68, 72]}
df = pd.DataFrame(data)

Understanding Bernoulli Trials:
11.Define a Bernoulli trial and give an example of a real-world scenario where Bernoulli trials can be applied. Write a Python program that simulates 10 independent Bernoulli trials with a success probability of 0.6 and computes the mean and variance of the outcomes. Explain how the mean and variance relate to the probability of success.

```
import numpy as np
p_success = 0.6
n_trials = 10
trials = np.random.binomial(1, p_success, n_trials)
mean_outcome = np.mean(trials)
variance_outcome = np.var(trials)
print(trials)
print(mean_outcome)
print(variance_outcome)
```

12. Simulating Bernoulli Trials with NumPy:

Write a Python program that simulates 1000 Bernoulli trials with a success probability of 0.4 using NumPy. Calculate the proportion of successes and compare it with the theoretical probability. Plot a bar chart to visualize the number of successes and failures. Explain the relationship between the simulated results and the theoretical probability of success.

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

p_success = 0.4   # Probability of success
```

```
n_trials = 1000   # Number of trials
trials = np.random.binomial(1, p_success, n_trials)
proportion_success = np.mean(trials)
# Theoretical probability of success
theoretical_prob = p_success
# Count the number of successes and failures
n_success = np.sum(trials)
n_failure = n_trials - n_success
# Display the results
print(f"Number of successes: {n_success}")
print(f"Number of failures: {n_failure}")
print(f"Proportion of successes (simulated): {proportion_success}")
print(f"Theoretical probability of success: {theoretical_prob}")
# Plot the bar chart
labels = ['Success', 'Failure']
counts = [n_success, n_failure]
plt.bar(labels, counts, color=['green', 'red'])
plt.ylabel('Count')
plt.title('Bernoulli Trials: Successes vs Failures')
plt.show()
```

13. Bernoulli Distribution: Mean and Variance:
Write a Python program to generate 500 Bernoulli trials with p=0.5 and calculate the sample mean and sample variance of the trials. Compare the results with the theoretical mean and variance, explaining any discrepancies.

```
import numpy as np

# Parameters
p = 0.5  # Probability of success
n = 500  # Number of trials
# Generate 500 Bernoulli trials
trials = np.random.binomial(1, p, n)
# Calculate sample mean and variance
sample_mean = np.mean(trials)
sample_variance = np.var(trials)
# Theoretical mean and variance
theoretical_mean = p
theoretical_variance = p * (1 - p)
# Display results
print(f"Sample Mean: {sample_mean}")
print(f"Theoretical Mean: {theoretical_mean}")
print(f"Sample Variance: {sample_variance}")
print(f"Theoretical Variance: {theoretical_variance}")

# Explain discrepancies
if abs(sample_mean - theoretical_mean) < 0.05 and abs(sample_variance - theoretical_variance) < 0.05:
   print("The sample mean and variance are close to the theoretical values as expected.")
else:
   print("The sample mean and variance differ slightly due to randomness in the sample.")
```

5. Binomial Distribution as a Series of Bernoulli Trials:

Explain how the binomial distribution is related to a series of Bernoulli trials. Write a Python program to generate 1000 random samples from a binomial distribution with n=10, and p=0.3. Calculate the mean, variance, and standard deviation of the samples, and compare these with the theoretical values of the binomial distribution.

```
import numpy as np,  import matplotlib.pyplot as plt
# Parameters of the binomial distribution
n = 10      # Number of trials
p = 0.3     # Probability of success
samples = 1000  # Number of samples

# Generate 1000 random samples from the binomial distribution
binom_samples = np.random.binomial(n, p, samples)
# Calculate sample mean, variance, and standard deviation
sample_mean = np.mean(binom_samples)
sample_variance = np.var(binom_samples)
sample_std_dev = np.std(binom_samples)
# Theoretical mean, variance, and standard deviation
theoretical_mean = n * p
theoretical_variance = n * p * (1 - p)
theoretical_std_dev = np.sqrt(theoretical_variance)

# Display the results
print(f"Sample Mean: {sample_mean}")
print(f"Sample Variance: {sample_variance}")
print(f"Sample Standard Deviation: {sample_std_dev}")

print(f"Theoretical Mean: {theoretical_mean}")
print(f"Theoretical Variance: {theoretical_variance}")
print(f"Theoretical Standard Deviation: {theoretical_std_dev}")

# Plot a histogram of the binomial samples
plt.hist(binom_samples, bins=np.arange(0, n+2)-0.5, density=True, color='blue',
alpha=0.7, edgecolor='black')
plt.title('Histogram of Binomial Distribution Samples (n=10, p=0.3)')
plt.xlabel('Number of successes')
plt.ylabel('Probability')
plt.xticks(range(0, n+1))
plt.grid(True)
plt.show()
```

14. Understanding the Normal Distribution:
Define the normal distribution and explain the significance of its parameters μ (mean) and σ(standard deviation). Write a Python program to generate 1000 random samples from a normal distribution with μ=50 and σ=10. Plot a histogram of the samples and superimpose the probability density function (PDF). Explain the shape of the histogram and its relation to the normal PDF.

```
import numpy as np, import matplotlib.pyplot as plt, import scipy.stats as stats
# Parameters of the normal distribution
mu = 50    # Mean
sigma = 10  # Standard deviation
samples = 1000  # Number of random samples
# Generate 1000 random samples from the normal distribution
normal_samples = np.random.normal(mu, sigma, samples)
```

```
# Calculate the range of x values for the PDF plot
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 1000)
# Compute the PDF for the normal distribution with mu=50 and sigma=10
pdf = stats.norm.pdf(x, mu, sigma)
# Plot the histogram of the random samples
plt.hist(normal_samples, bins=30, density=True, alpha=0.6, color='b', edgecolor='black')
# Superimpose the PDF
plt.plot(x, pdf, 'r', linewidth=2)
# Add labels and title
plt.title('Histogram of Samples and Normal Distribution PDF ($\mu=50$, $\sigma=10$)')
plt.xlabel('Value')
plt.ylabel('Density')
# Show the plot
plt.show()
```

7. Properties of the Normal Distribution:
The 68-95-99.7 rule (Empirical rule) states that for a normal distribution:
68% of the data falls within 1 standard deviation from the mean,
95% within 2 standard deviations, and
99.7% within 3 standard deviations.

Write a Python program to generate 10,000 samples from a normal distribution with mean μ=100, μ=100 and standard deviation σ=15. Verify the empirical rule by calculating the proportion of data points within 1, 2, and 3 standard deviations of the mean.

Questions and Answers: Unit 3

1.Provide an example to illustrate the basic strurctuire of an HTML document. In your response, ensure you include details about the essential tags used to structure an HTML document.

2.Describe the purpose and usage of the HTML tags: <head><body><title><meta><link>
Include examples to demonstrate how these tags are used in an HTML document.

3.How are hyperlinks created in HTML using the <a> tag? Explain the attributes associated with the <a> tag and provide examples to show how links can be opened in new tabs, link to an email, or link to external websites.

4.Enumerate the difference between ordered and unordered lists in HTML. Describe the purpose of the <ol> and <ul>, <li> tags and provide examples showing how to create both types in a web document.

5.Consider the following simple html structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>
       <title>Simple Page</title>
  </head>
  <body>
    <h1>Welcome to My Webpage</h1>
    <p>This is a simple paragraph of text.</p>
  </body>
</html>
```

Modify the HTML to achieve the following:
Center the heading <h1> both horizontally on the page.
Change the font size of the heading to 36px and the paragraph text to 18px.
Set the heading's text color to blue and the paragraph's text color to green.

6.Consider the following HTML structure for an ordered list:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Ordered List Example</title>
  </head>
  <body>
    <ol>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ol>
  </body>
</html>
```

Modify the HTML to achieve the following:

Change the list item numbers to be displayed as Roman numerals (I, II, III).
Set the font size of the list items to 20px.
Set the text color of the second item to red.

7.Consider the following HTML structure for an unordered list:

```
<!DOCTYPE html>
<html lang="en">
  <head>
     <title>Unordered List Example</title>
  </head>
  <body>
    <ul>
      <li>Apples</li>
      <li>Oranges</li>
      <li>Bananas</li>
    </ul>
  </body>
</html>
```

Modify the HTML to achieve the following: Change the bullet points to square shapes.
Set the font size of the list items to 18px and Set the text color of the first item (Apples) to blue and the third item (Bananas) to green.

Questions and Answers: Unit 4

1.Consider the following example of an HTML structure:

```
<!DOCTYPE html>
```

```
<html lang="en">
  <head>
     <title>Interactive Button</title>
  </head>
  <body>
    <h1>Click the button to display a message:</h1>
    <button id="myButton">Click Me</button>
    <p id="message"></p>

    <script>
      // JavaScript will go here
    </script>
  </body>
</html>
```

Modify the HTML and JavaScript to achieve the following:

When the button is clicked, a JavaScript function should display the message "Hello, welcome to the site!" inside the <p> element.
Create a second button that, when clicked, clears the message from the <p> element.
Ensure that both buttons respond correctly to user interaction.
Write the complete solution by adding JavaScript inside the <script> tag.

2.Consider the following HTML structure:

html
Copy code

```
<!DOCTYPE html>
<html lang="en">
  <head>
     <title>Alert Box Example</title>
  </head>
  <body>
    <h1>Welcome to My Webpage</h1>
    <button id="alertButton">Show Alert</button>

    <script>
      // JavaScript will go here
    </script>
  </body>
</html>
```

Modify the HTML and JavaScript to do the following:

When the button with id alertButton is clicked, display an alert dialog box with the message "This is an alert message!".
Write a JavaScript function named showAlert() that will handle the click event and trigger the alert.
Ensure that the function is properly connected to the button's click event using JavaScript.
Write the complete solution by adding JavaScript inside the <script> tag.

3.Consider the following HTML structure:

```
<!DOCTYPE html>
<html lang="en">
```

```
  <head>
    <title>Confirmation Box Example</title>
  </head>
  <body>
    <h1>Delete Confirmation</h1>
    <button id="deleteButton">Delete Item</button>
    <script>
      // JavaScript will go here
    </script>
  </body>
</html>
```

Modify the HTML and JavaScript to achieve the following:

When the deleteButton is clicked, display a confirm dialog box asking the user, "Are you sure you want to delete this item?".
If the user clicks "OK", display an alert box that says "Item deleted successfully".
If the user clicks "Cancel", display an alert box that says "Action cancelled".
Write the complete solution by adding JavaScript inside the <script> tag.

4.Consider the following HTML structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Input Alert Example</title>
  </head>
  <body>
    <h1>Enter Your Name:</h1>
    <input type="text" id="nameInput" placeholder="Your name">
    <button id="greetButton">Greet Me</button>

    <script>
      // JavaScript will go here
    </script>
  </body>
</html>
```

Modify the HTML and JavaScript to achieve the following:

When the greetButton is clicked, retrieve the value entered in the text input field with id nameInput.
Display an alert box with a personalized greeting, such as "Hello, [name]!".
If the input field is empty, display an alert saying "Please enter your name".
Write the complete solution by adding JavaScript inside the <script> tag.

5.Consider the following simple HTML structure:

html
Copy code

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Function Example</title>
  </head>
  <body>
    <h1>Sum of Two Numbers</h1>
    <button id="calculateButton">Calculate Sum</button>

    <script>
      // JavaScript will go here
    </script>
  </body>
</html>
```

Modify the HTML and JavaScript to do the following:

Create a JavaScript function named calculateSum that takes two numbers as parameters and returns their sum. Inside the calculateSum function, use the numbers 5 and 10 as arguments and display the result in an alert box when the button is clicked. Ensure the button click triggers the calculateSum function using an event listener in JavaScript.
Write the complete solution by adding the JavaScript code inside the <script> tag.

6.Consider the following HTML structure for a grading system:

html
Copy code

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Grading System</title>
  </head>
  <body>
    <h1>Check Grade</h1>
    <button id="gradeButton">Get Grade</button>

    <script>
      // JavaScript will go here
    </script>
  </body>
</html>
```

Modify the HTML and JavaScript to achieve the following:

Write a JavaScript function named getGrade that takes a student's score (e.g., 85) as an argument and returns the corresponding grade based on this scale:90 and above: "A", 80 to 89: "B", 70 to 79: "C"
Below 70: "Fail". Inside the getGrade function, use a conditional (if-else) statement to evaluate the score and return the appropriate grade.
When the gradeButton is clicked, display an alert box showing the grade for a score of 85.
Write the complete solution by adding the JavaScript code inside the <script> tag.