Theoretical Questions

##Assignment Data Toolkit

Q1. What is NumPy, and why is it widely used in Python?

- NumPy is a fundamental package for scientific computing in Python. It provides support for arrays, matrices, and many mathematical functions to operate on these data structures. NumPy is widely used because it offers efficient storage and operations, making it ideal for numerical computations. It also serves as the foundation for many other scientific libraries in Python, such as SciPy and Pandas. Example:

```
import numpy as np
a = np.array([1, 2, 3])
print(a)
```

Q2. How does broadcasting work in NumPy?

- Broadcasting in NumPy allows arithmetic operations on arrays of different shapes. It automatically expands the smaller array to match the shape of the larger array. This feature simplifies code and improves performance by avoiding the need for explicit loops. Broadcasting is particularly useful in mathematical operations and data manipulation tasks. Example:

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([[1], [2], [3]])
print(a + b)
```

Q3. What is a pandas DataFrame?

- A pandas DataFrame is a two-dimensional, size-mutable, and heterogeneous tabular data structure with labeled axes (rows and columns). It is similar to a spreadsheet or SQL table and is widely used for data manipulation and analysis. DataFrames provide a range of functionalities for data cleaning, transformation, and aggregation, making them a powerful tool for data scientists and analysts. Example:

```
import pandas as pd
data = {'A': [1, 2], 'B': [3, 4]}
df = pd.DataFrame(data)
print(df)
```

Q4. Explain the use of the `groupby()` method in Pandas.

- The `groupby()` method in Pandas is used to split data into groups based on some criteria. It is often used for aggregation, transformation, and filtering operations on grouped data. This method is particularly useful for summarizing data and performing complex analyses on subsets of data. Example:

```
import pandas as pd
data = {'A': ['foo', 'bar', 'foo'], 'B': [1, 2, 3]}
df = pd.DataFrame(data)
grouped = df.groupby('A').sum()
print(grouped)
```

Q5. Why is Seaborn preferred for statistical visualizations?

- Seaborn is preferred for statistical visualizations because it provides a high-level interface for drawing attractive and informative statistical graphics. It is built on top of Matplotlib and integrates well with Pandas data structures. Seaborn simplifies the process of creating complex visualizations and offers a variety of plot types that are useful for statistical analysis. Example:

```
import seaborn as sns
import matplotlib.pyplot as plt
tips = sns.load_dataset('tips')
sns.boxplot(x='day', y='total_bill', data=tips)
plt.show()
```

Q6. What are the differences between NumPy arrays and Python lists?

- NumPy arrays are more efficient for numerical computations and provide more functionality for mathematical operations compared to Python lists. Arrays are fixed in size and support element-wise operations, while lists are more flexible and can store different data types. NumPy arrays also offer better performance due to their contiguous memory allocation and optimized C-based implementation. Example:

```
import numpy as np
a = np.array([1, 2, 3])
b = [1, 2, 3]
print(a * 2)
print([x * 2 for x in b])
```

Q7. What is a heatmap, and when should it be used?

- A heatmap is a graphical representation of data where individual values are represented as colors. It is used to visualize the magnitude of values in a matrix and is useful for identifying patterns, correlations, and outliers. Heatmaps are commonly used in fields

such as bioinformatics, finance, and machine learning to display complex data relationships. Example:

```
import seaborn as sns
import numpy as np
data = np.random.rand(10, 12)
sns.heatmap(data)
plt.show()
```

Q8. What does the term "vectorized operation" mean in NumPy?

- Vectorized operations in NumPy refer to performing element-wise operations on arrays without explicit loops. This leads to more concise and faster code. Vectorized operations leverage NumPy's optimized C-based implementation, resulting in significant performance improvements for large datasets. Example:

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = a + b
print(c)
```

Q9. How does Matplotlib differ from Plotly?

- Matplotlib is a static plotting library, while Plotly is an interactive plotting library. Matplotlib is suitable for creating static, publication-quality plots, whereas Plotly is ideal for creating interactive visualizations that can be embedded in web applications. Plotly offers features like zooming, panning, and tooltips, making it more suitable for exploratory data analysis. Example:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [4, 5, 6])
plt.show()

import plotly.express as px
fig = px.line(x=[1, 2, 3], y=[4, 5, 6])
fig.show()
```

Q10. What is the significance of hierarchical indexing in Pandas?

- Hierarchical indexing in Pandas allows for multiple levels of indexing on an axis, enabling more complex data structures and easier data manipulation. It is useful for working with high-dimensional data in a lower-dimensional form. Hierarchical indexing simplifies data analysis tasks such as slicing, dicing, and aggregating data across multiple dimensions. Example:

```
import pandas as pd
arrays = [['bar', 'bar', 'baz', 'baz'], ['one', 'two', 'one', 'two']]
index = pd.MultiIndex.from_arrays(arrays, names=['first', 'second'])
s = pd.Series([1, 2, 3, 4], index=index)
print(s)
```

Q11. What is the role of Seaborn's `pairplot()` function?

- Seaborn's `pairplot()` function creates a matrix of scatter plots for each pair of variables in a dataset. It is useful for visualizing relationships between multiple variables and identifying patterns or correlations. The `pairplot()` function also supports different plot types for the diagonal and off-diagonal elements, providing a comprehensive view of the data. Example:

```
import seaborn as sns
iris = sns.load_dataset('iris')
sns.pairplot(iris)
plt.show()
```

Q12. What is the purpose of the `describe()` function in pandas?

- The `describe()` function in pandas provides summary statistics of a DataFrame's numerical columns, including count, mean, standard deviation, min, and max values. It is useful for getting a quick overview of the data and identifying potential issues such as outliers or missing values. The `describe()` function can also be used with categorical data by specifying the `include` parameter. Example:

```
import pandas as pd
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
print(df.describe())
```

Q13. Why is handling missing data important in pandas?

- Handling missing data is important in pandas because missing values can lead to inaccurate analysis and results. Proper handling ensures data integrity and improves the quality of insights derived from the data. Pandas provides various methods for handling missing data, such as filling with default values, forward/backward filling, and interpolation. Example:

```
import pandas as pd
df = pd.DataFrame({'A': [1, 2, None], 'B': [4, None, 6]})
df = df.fillna(0)
print(df)
```

Q14. What are the benefits of using Plotly for data visualization?

- Plotly offers interactive and visually appealing plots that can be easily embedded in web applications. It supports a wide range of chart types and provides tools for exploring data interactively. Plotly's interactivity features, such as zooming, panning, and tooltips, make it ideal for exploratory data analysis and presentations. Additionally, Plotly integrates well with other libraries like Dash for building interactive web applications. Example:

```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x='sepal_width', y='sepal_length')
fig.show()
```

Q15. How does NumPy handle multidimensional arrays?

- NumPy handles multidimensional arrays using the `ndarray` object, which allows for efficient storage and manipulation of data in multiple dimensions. It supports various operations like slicing, reshaping, and broadcasting. NumPy's multidimensional arrays are essential for scientific computing tasks, such as linear algebra, Fourier transforms, and random number generation. Example:

```
import numpy as np
a = np.array([[1, 2], [3, 4]])
print(a.shape)
```

Q16. What is the role of Bokeh in data visualization?

- Bokeh is a Python library for creating interactive and scalable visualizations for web applications. It provides tools for building complex plots and dashboards with high performance. Bokeh's interactivity features, such as zooming, panning, and tooltips, make it suitable for exploratory data analysis and presentations. Bokeh also supports integration with other libraries like Flask and Django for building interactive web applications. Example:

```
from bokeh.plotting import figure, show
p = figure(title='example')
p.line([1, 2, 3], [4, 5, 6])
show(p)
```

Q17. Explain the difference between `apply()` and `map()` in Pandas.

- The `apply()` function in Pandas is used to apply a function along an axis of the DataFrame, while `map()` is used to apply a function element-wise on a Series. `apply()` is more flexible and can be used with DataFrames, whereas `map()` is limited to Series. `apply()` can also be used for more complex operations, such as applying functions to

rows or columns, while `map()` is typically used for simple element-wise transformations. Example:

```
import pandas as pd
df = pd.DataFrame({'A': [1, 2, 3]})
df['B'] = df['A'].apply(lambda x: x * 2)
print(df)
```

Q18. What are some advanced features of NumPy?

- Some advanced features of NumPy include broadcasting, vectorized operations, linear algebra functions, random number generation, and support for multidimensional arrays. These features make NumPy a powerful tool for scientific computing. NumPy also provides tools for integrating with other libraries, such as SciPy and Pandas, and supports various file formats for data storage and retrieval. Example:

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
print(np.dot(a, b))
```

Q19. How does Pandas simplify time series analysis?

- Pandas simplifies time series analysis by providing tools for date and time manipulation, resampling, and rolling window calculations. It supports time-based indexing and offers various methods for handling time series data. Pandas also provides functions for generating date ranges, shifting data, and calculating moving averages, making it a powerful tool for time series analysis. Example:

```
import pandas as pd
dates = pd.date_range('20230101', periods=6)
df = pd.DataFrame({'A': [1, 2, 3, 4, 5, 6]}, index=dates)
print(df)
```

Q20. What is the role of a pivot table in Pandas?

- A pivot table in Pandas is used to summarize data by aggregating values based on one or more keys. It allows for flexible data analysis and is useful for generating reports and insights. Pivot tables support various aggregation functions, such as sum, mean, and count, and can be used to reshape data for easier analysis. Example:

```
import pandas as pd
data = {'A': ['foo', 'bar', 'foo'], 'B': [1, 2, 3]}
df = pd.DataFrame(data)
pivot = df.pivot_table(values='B', index='A', aggfunc='sum')
print(pivot)
```

Q21. Why is NumPy's array slicing faster than Python's list slicing?

- NumPy's array slicing is faster than Python's list slicing because NumPy arrays are stored in contiguous memory blocks, allowing for efficient access and manipulation. Additionally, NumPy operations are implemented in C, providing further performance improvements. The optimized memory layout and low-level implementation of NumPy arrays result in faster slicing and indexing operations compared to Python lists. Example:

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])
print(a[1:4])
```

Q22. What are some common use cases for Seaborn?

- Common use cases for Seaborn include visualizing distributions, relationships, and categorical data. It is often used for creating histograms, scatter plots, box plots, and heatmaps. Seaborn's high-level interface and integration with Pandas make it a popular choice for data visualization tasks in data science and machine learning projects. Example:

```
import seaborn as sns
tips = sns.load_dataset('tips')
sns.histplot(tips['total_bill'])
plt.show()
```

Practical Questions

```
'''
Q1. How do you create a 2D NumPy array and calculate the sum of each
row?

'''
'''
Answer:-1

'''
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

sums = arr.sum(axis=1)

print("2D NumPy array:")
print(arr)
```

```
print("\nSum of each row:")
print(sums)
```

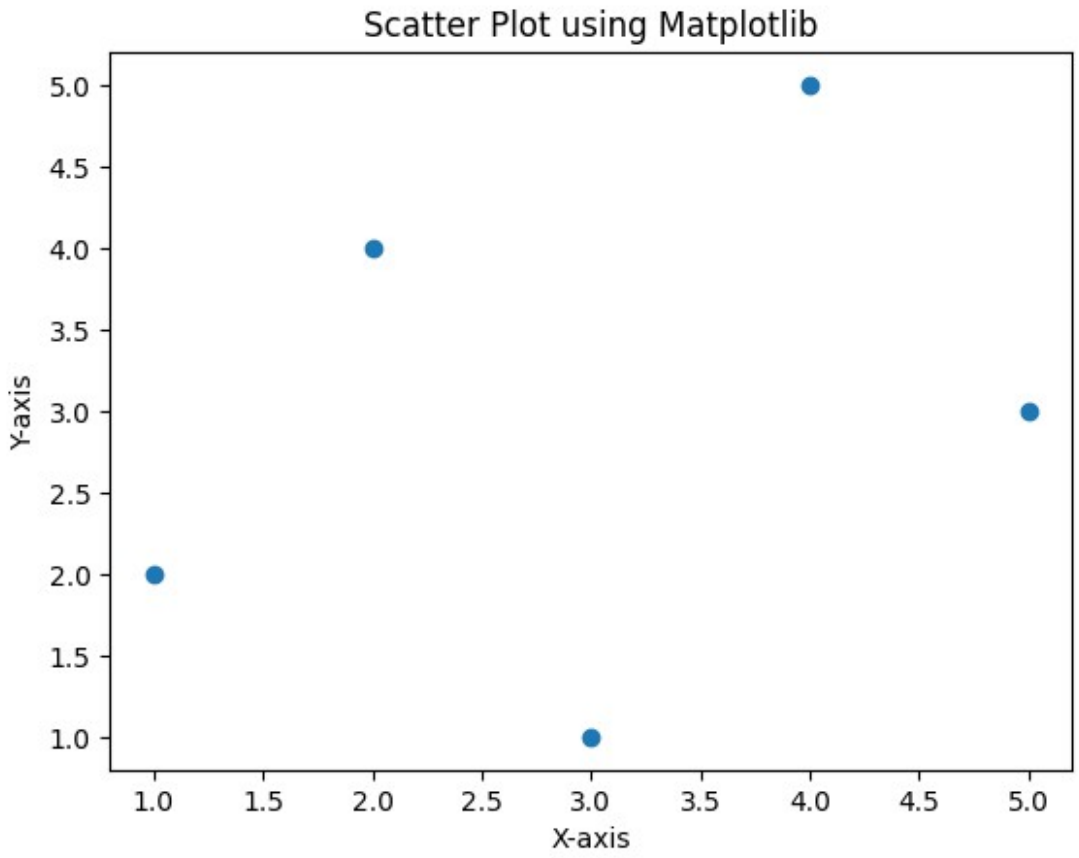
```
2D NumPy array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Sum of each row:
[ 6 15 24]
```

```
'''
Q2. Write a pandas script to find the mean of a specific column in a
DataFrame.

'''
'''
Answer:-2

'''
import pandas as pd

data = {'col1': [1, 2, 3, 4, 5],
        'col2': [6, 7, 8, 9, 10],
        'col3': [11, 12, 13, 14, 15]}
df = pd.DataFrame(data)

mean_col2 = df['col2'].mean()

# Print the mean
print("DataFrame:")
print(df)
print("\nMean of the 'col2' column:")
print(mean_col2)
```

```
DataFrame:
   col1  col2  col3
0     1     6    11
1     2     7    12
2     3     8    13
3     4     9    14
4     5    10    15

Mean of the 'col2' column:
8.0
```

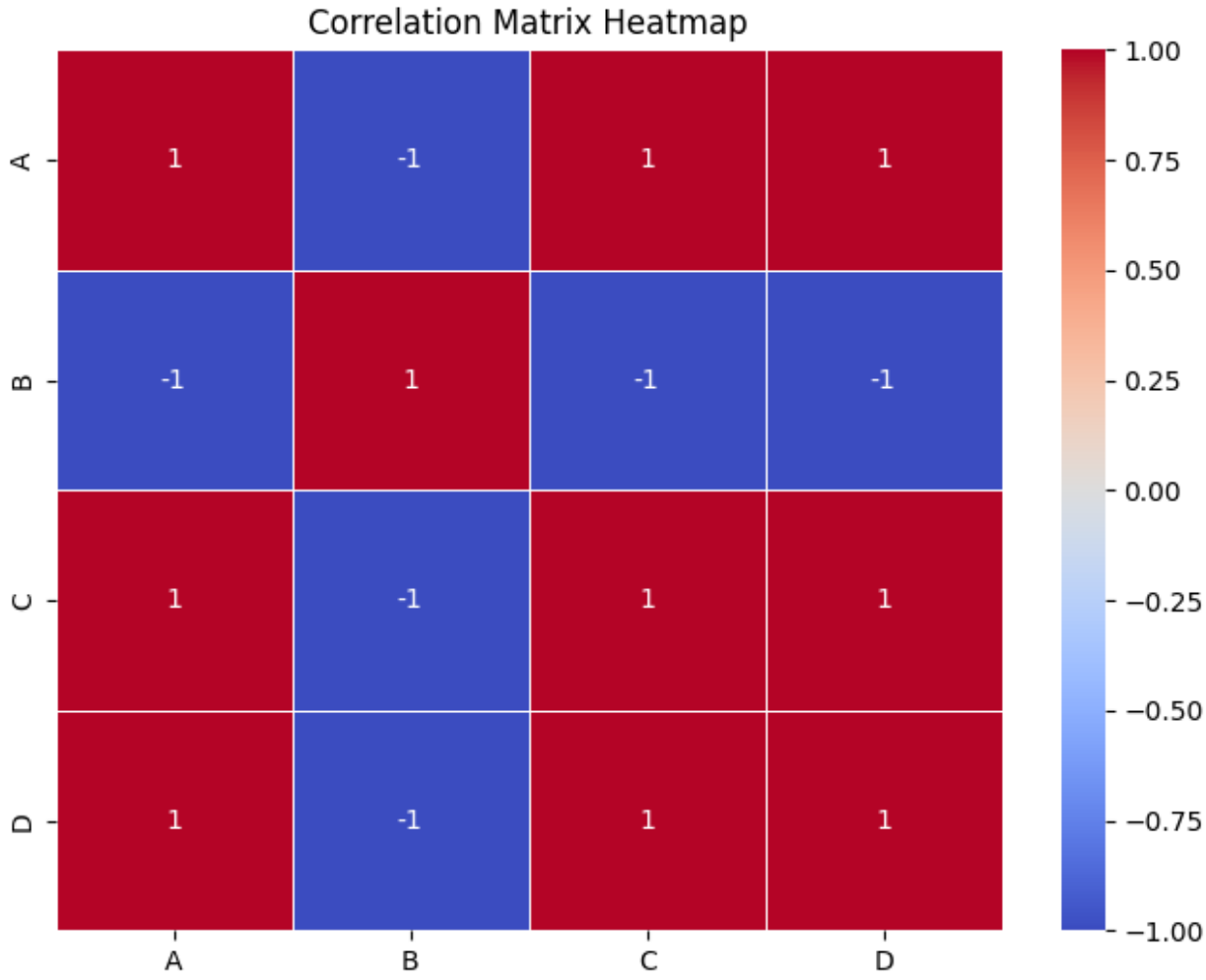
```
'''
Q3. Create a scatter plot using Matplotlib.

'''
'''
```

```
Answer:-3

'''
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 4, 1, 5, 3]

plt.scatter(x, y)

plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Scatter Plot using Matplotlib")

plt.show()
```




```
'''
Q4. How do you calculate the correlation matrix using Seaborn and
visualize it with a heatmap?

'''
'''
Answer:-4
```

```
'''
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = {'A': [1, 2, 3, 4, 5],'B': [5, 4, 3, 2, 1], 'C': [2, 3, 4, 5,
6],'D': [5, 6, 7, 8, 9]}
df = pd.DataFrame(data)

corr_matrix = df.corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

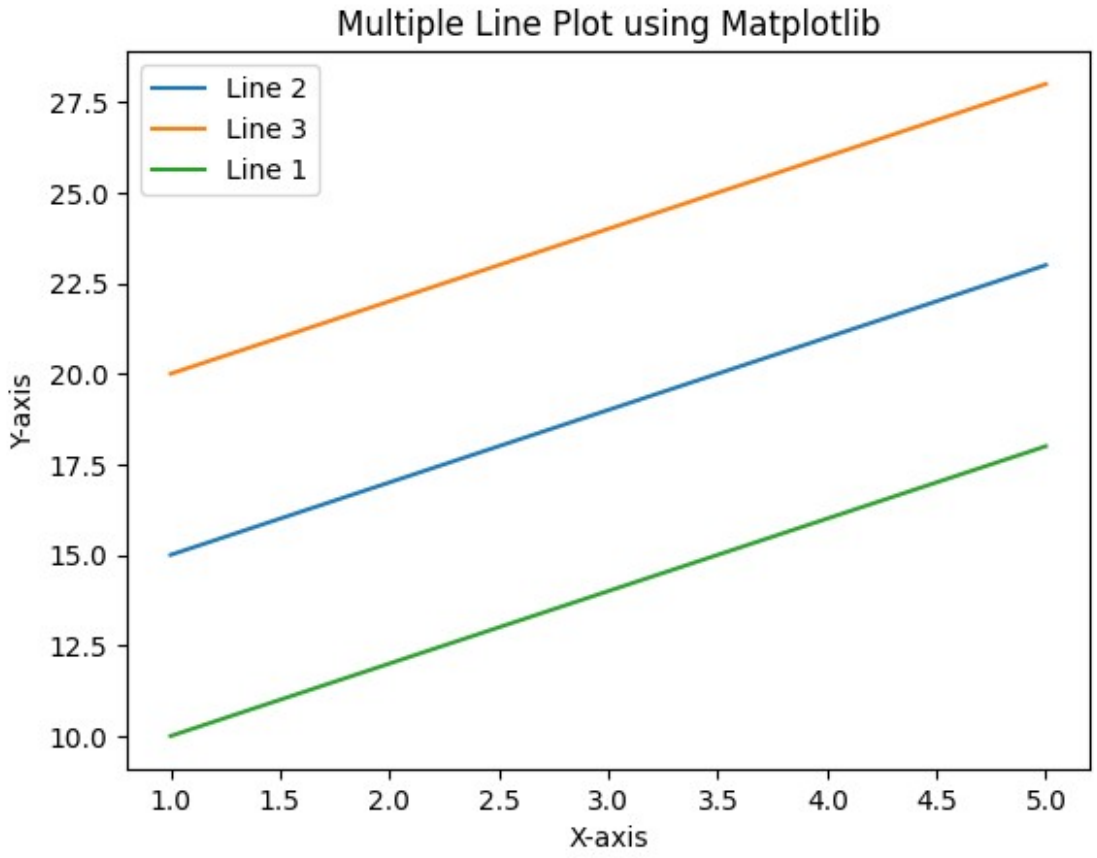


```
'''
Q5. Generate a bar plot using Plotly.

'''
'''
Answer:-5

'''
import plotly.graph_objects as go
import pandas as pd

data = {'Category': ['A', 'B', 'C', 'D', 'E'],
        'Values': [15, 25, 10, 30, 20]}
df = pd.DataFrame(data)

fig = go.Figure(data=[go.Bar(x=df['Category'], y=df['Values'])])

fig.update_layout(title='Bar Plot using Plotly',
                  xaxis_title='Category',
                  yaxis_title='Values')

fig.show()
'''
Q6. Create a DataFrame and add a new column based on an existing
column.

'''
'''
Answer:-6

'''
import pandas as pd

data = {'col1': [1, 2, 3, 4, 5], 'col2': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)

df['New column'] = df['col2'] * 2

df
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 5,\n  \"fields\": [\n
{\n      \"column\": \"col1\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 5,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          2,\n          5,\n          3\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"col2\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 15,\n        \"min\": 10,\n
\"max\": 50,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          20,\n          50,\n          30\n        ],\n
```

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"New column\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
31,\n        \"min\": 20,\n        \"max\": 100,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          40,\n
100,\n          60\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}

'''
Q7. Write a program to perform element-wise multiplication of two
NumPy arrays.

'''
'''
Answer:-7

'''
import numpy as np

array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([6, 7, 8, 9, 10])

result_array = array1 * array2

result_array

array([ 6, 14, 24, 36, 50])

'''
Q8. Create a line plot with multiple lines using Matplotlib.

'''
'''
Answer:-8

'''
import matplotlib.pyplot as plt

x1 = [1, 2, 3, 4, 5]
y1 = [10, 12, 14, 16, 18]

x2 = [1, 2, 3, 4, 5]
y2 = [15, 17, 19, 21, 23]

x3 = [1, 2, 3, 4, 5]
y3 = [20, 22, 24, 26, 28]

#just rearranged to make a tricolor
plt.plot(x2, y2, label='Line 2')
plt.plot(x3, y3, label='Line 3')
```

```
plt.plot(x1, y1, label='Line 1')

plt.title('Multiple Line Plot using Matplotlib')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

plt.legend()

plt.show()
```

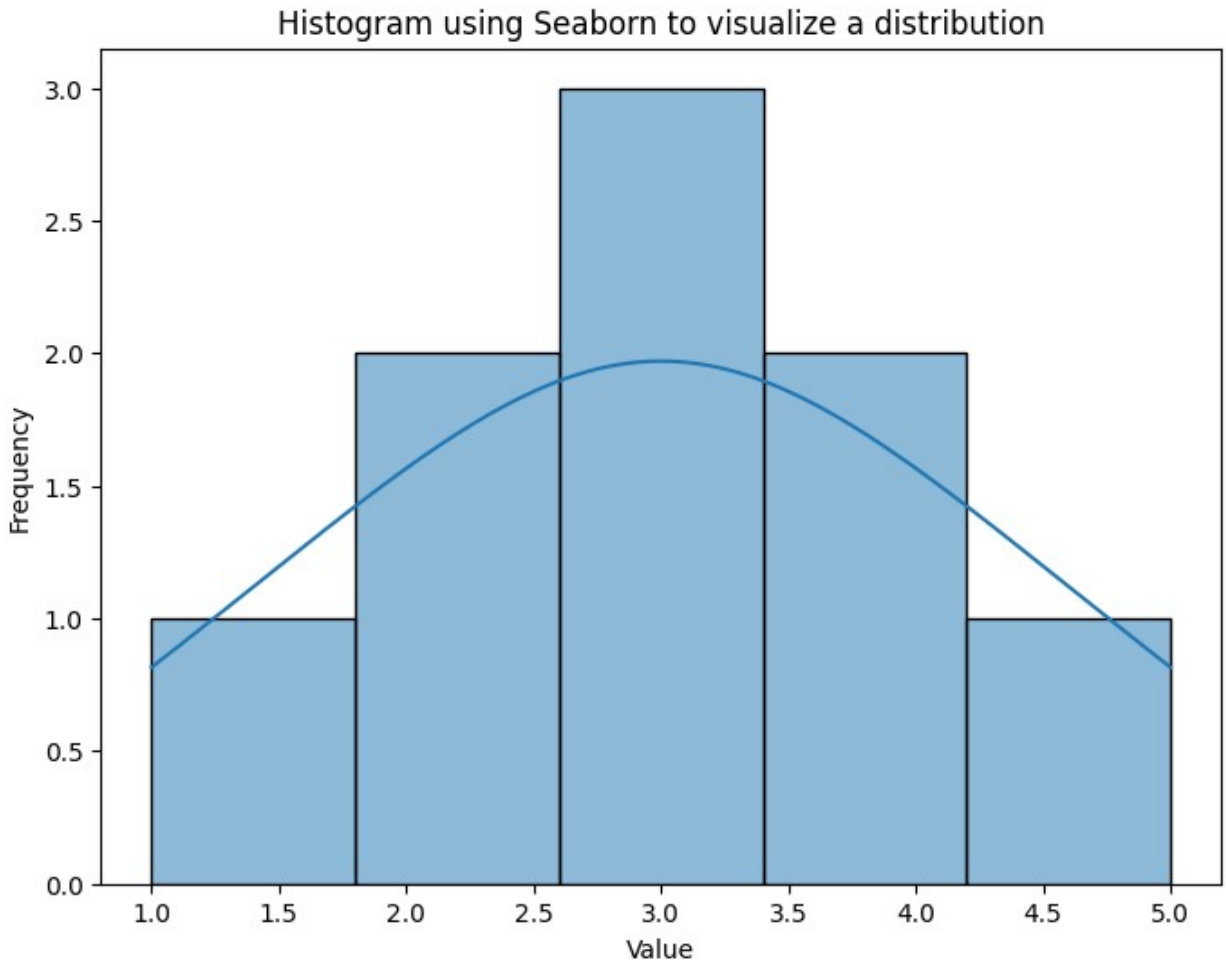



```
'''
Q9. Generate a pandas DataFrame and filter rows where a column value
is greater than a threshold.

'''
'''
Answer:-9

'''
import pandas as pd

data = {'col1': [1, 2, 3, 4, 5],
        'col2': [6, 7, 8, 9, 10],
```

```
        'col3': [11, 12, 13, 14, 15]}
df = pd.DataFrame(data)

threshold = 7

filtered_df = df[df['col2'] > threshold]

filtered_df
```


```
{"summary":"{\n  \"name\": \"filtered_df\",\n  \"rows\": 3,\n
\"fields\": [\n    {\n      \"column\": \"col1\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 3,\n        \"max\": 5,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          3,\n
4,\n          5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"col2\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 8,\n        \"max\": 10,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          8,\n
9,\n          10\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"col3\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 13,\n        \"max\": 15,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          13,\n
14,\n          15\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"filtered_df"}
```


```
'''
Q10. Create a histogram using Seaborn to visualize a distribution.

'''
'''
Answer:-10

'''
import matplotlib.pyplot as plt
import seaborn as sns

data = [1, 2, 2, 3, 3, 3, 4, 4, 5]

plt.figure(figsize=(8, 6))
sns.histplot(data, kde=True)

plt.title("Histogram using Seaborn to visualize a distribution")
plt.xlabel("Value")
plt.ylabel("Frequency")

plt.show()
```




```
'''
Q11. Perform matrix multiplication using NumPy.

'''
'''
Answer:-11

'''
import numpy as np

matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])

result_matrix = np.dot(matrix1, matrix2)

result_matrix
```

```
array([[19, 22],
       [43, 50]])
```

```
'''
Q12. Use Pandas to load a CSV file and display its first 5 rows.

'''
'''
Answer:-12

'''
import pandas as pd

file_name = 'my_file.csv'

try:
    df = pd.read_csv(file_name)
    print(df.head())
except FileNotFoundError:
    print(f"Error: '{file_name}' not found. Please provide the correct
file path.")
except pd.errors.EmptyDataError:
    print(f"Error: '{file_name}' is empty.")
except pd.errors.ParserError:
    print(f"Error: Unable to parse '{file_name}'. Check file format.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

```
      Name  Age  Score
0  AVINESH   21     95
1   ADITYA   22     90
2     AJAY   23     85
3  AVINESH   24     80
4   ADITYA   25     75
```

```
'''
Q13. Create a 3D scatter plot using Plotly.

'''
'''
Answer:-13

'''
import plotly.graph_objects as go
import numpy as np

x = np.random.rand(100)
y = np.random.rand(100)
z = np.random.rand(100)

fig = go.Figure(data=[go.Scatter3d(x=x, y=y, z=z, mode='markers')])

fig.update_layout(title='3D Scatter Plot using Plotly',
                  scene=dict(xaxis_title='X-axis',
```

```
                                yaxis_title='Y-axis',
                                zaxis_title='Z-axis'))

fig.show()
```