Week 1: Basic Data Analytics Using NumPy (8 Programs)

In [1]:
```
##Aim
To learn basic data analytics using NumPy for numerical computations and array manipulations.
```

In [3]:
```
##Programs
#Program 1: Basic Array Operations
Procedure: Create a NumPy array and perform basic arithmetic operations.
```

In [4]:
```
#Code:

import numpy as np
# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])
print("Array:", arr)
# Perform arithmetic operations
print("Add 5:", arr + 5)
print("Multiply by 2:", arr * 2)
```

```
Array: [1 2 3 4 5]
Add 5: [ 6  7  8  9 10]
Multiply by 2: [ 2  4  6  8 10]
```

In [5]:
```
#Program 2: Array Statistics
Procedure: Calculate mean, median, and standard deviation
```

In [6]:
```
#Code:

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
print("Mean:", np.mean(arr))
print("Median:", np.median(arr))
print("Standard Deviation:", np.std(arr))
```

```
Mean: 5.5
Median: 5.5
Standard Deviation: 2.8722813232690143
```

In [7]:
```
#Program 3: Reshaping Arrays
Procedure: Reshape a 1D array into a 2D array
```

In [8]:
```
#Code:

arr = np.arange(1, 13)
reshaped_arr = arr.reshape(3, 4)
print("Reshaped Array:\n", reshaped_arr)
```

```
Reshaped Array:
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

In [9]:
```
#Program 4: Array Indexing
Procedure: Demonstrate indexing and slicing in NumPy arrays
```

In [10]:
```
#Code:

arr = np.array([10, 20, 30, 40, 50])
print("First Element:", arr[0])
print("Last Element:", arr[-1])
print("Slice (1 to 3):", arr[1:4])
```

```
First Element: 10
Last Element: 50
Slice (1 to 3): [20 30 40]
```

In [11]:
```
#Program 5: Array Concatenation
Procedure: Concatenate two arrays
```

In [12]:
```
#Code:

arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
concatenated = np.concatenate((arr1, arr2))
print("Concatenated Array:", concatenated)
```

```
Concatenated Array: [1 2 3 4 5 6]
```

In [13]:
```
#Program 6: Boolean Indexing
Procedure: Use boolean conditions to filter arrays
```

In [14]:
```
#Code:

arr = np.array([1, 2, 3, 4, 5])
```

```
filtered_arr = arr[arr > 2]
print("Filtered Array (greater than 2):", filtered_arr)
```

```
Filtered Array (greater than 2): [3 4 5]
```

In [15]:
```
#Program 7: Dot Product
Procedure: Calculate the dot product of two arrays.
```

In [16]:
```
#Code:

arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
dot_product = np.dot(arr1, arr2)
print("Dot Product:", dot_product)
```

```
Dot Product: 32
```

In [17]:
```
#Program 8: Linear Algebra Operations
Procedure: Perform matrix operations like inverse and determinant
```

In [18]:
```
#Code:

matrix = np.array([[1, 2], [3, 4]])
determinant = np.linalg.det(matrix)
inverse = np.linalg.inv(matrix)
print("Determinant:", determinant)
print("Inverse:\n", inverse)
```

```
Determinant: -2.0000000000000004
Inverse:
 [[-2.   1. ]
 [ 1.5 -0.5]]
```

Week 2: Basics of Data Wrangling Features (10 Programs)

In [19]:
```
#Aim
To explore basic data wrangling features using Pandas for data manipulation.
```

In [20]:
```
#Programs
Program 1: Creating a DataFrame
Procedure: Create a DataFrame from a dictionary.
```

In [24]:
```
#Code:

import pandas as pd
data = {'Name': ['Alice', 'Bob', 'Charlie'],
 'Age': [24, 27, 22],
 'City': ['New York', 'Los Angeles', 'Chicago']}
df = pd.DataFrame(data)
print(df)
```

```
      Name  Age         City
0    Alice   24     New York
1      Bob   27  Los Angeles
2  Charlie   22      Chicago
```

In [25]:
```
#Program 2: Reading CSV Files
Procedure: Read a CSV file into a DataFrame.
```

In [26]:
```
#Code:

df = pd.read_csv('sample.csv') # Assuming 'sample.csv' exists
print(df.head())
```

```
   Unnamed: 0 Unnamed: 1 Unnamed: 2 Unnamed: 3  Unnamed: 4  Unnamed: 5  \
0         NaN        NaN        NaN        NaN         NaN         NaN
1         NaN        NaN        NaN        NaN         NaN         NaN
2         NaN        NaN        NaN        NaN         NaN         NaN
3         NaN       Name        Age       City         NaN         NaN
4         0.0      Alice         24   New York         NaN         NaN

   Unnamed: 6
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN
```

In [27]:
```
#Program 3: Data Cleaning
Procedure: Clean data by removing missing values.
```

In [28]:
```
#Code:

df = pd.DataFrame({'A': [1, 2, None, 4], 'B': [None, 2, 3, 4]})
df_cleaned = df.dropna()
```

```
print(df_cleaned)
```

```
     A    B
1  2.0  2.0
3  4.0  4.0
```

In [29]:
```
#Program 4: Filtering Data
Procedure: Filter rows based on a condition
```

In [30]:
```
#Code:

df = pd.DataFrame({'A': [1, 2, 3, 4], 'B': [5, 6, 7, 8]})
filtered_df = df[df['A'] > 2]
print(filtered_df)
```

```
   A  B
2  3  7
3  4  8
```

In [31]:
```
#Program 5: Grouping Data
Procedure: Group data and calculate aggregates.
```

In [32]:
```
#Code:

df = pd.DataFrame({'A': ['foo', 'bar', 'foo', 'bar'],
 'B': [1, 2, 3, 4]})
grouped = df.groupby('A').sum()
print(grouped)
```

```
     B
A
bar  6
foo  4
```

In [33]:
```
#Program 6: Merging DataFrames
Procedure: Merge two DataFrames.
```

In [34]:
```
#Code:

df1 = pd.DataFrame({'A': ['foo', 'bar'], 'B': [1, 2]})
df2 = pd.DataFrame({'A': ['foo', 'bar'], 'C': [3, 4]})
merged_df = pd.merge(df1, df2, on='A')
print(merged_df)
```

```
     A  B  C
0  foo  1  3
1  bar  2  4
```

In [35]:
```
#Program 7: Pivot Tables
Procedure: Create a pivot table from data.
```

In [36]:
```
#Code:

df = pd.DataFrame({'A': ['foo', 'bar', 'foo', 'bar'],
 'B': [1, 2, 3, 4],
 'C': [5, 6, 7, 8]})
pivot_table = df.pivot_table(values='C', index='A', columns='B', aggfunc='sum')
print(pivot_table)
```

```
B      1    2    3    4
A
bar  NaN  6.0  NaN  8.0
foo  5.0  NaN  7.0  NaN
```

In [37]:
```
#Program 8: DataFrame Transformation
Procedure: Apply transformations to a DataFrame.
```

In [38]:
```
#Code:

df = pd.DataFrame({'A': [1, 2, 3, 4], 'B': [5, 6, 7, 8]})
transformed_df = df.transform(lambda x: x ** 2)
print(transformed_df)
```

```
    A   B
0   1  25
1   4  36
2   9  49
3  16  64
```

In [39]:
```
#Program 9: Handling Duplicates
Procedure: Identify and remove duplicates from a DataFrame.
```

In [40]:
```
#Code:

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

```
   A  B
0  1  4
2  2  5
3  3  6
```

In [41]:
```
#Program 10: Saving DataFrames to CSV
Procedure: Save a DataFrame to a CSV file
```

In [43]:
```
#Code:

df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
df.to_csv('output.csv', index=False)
print("DataFrame saved to output.csv")
```

```
DataFrame saved to output.csv
```

Week 3: Pandas Time Series Analysis (5 Programs)

In [44]:
```
#Aim
To perform time series analysis using Pandas for analyzing time-based data.
```

In [45]:
```
#Programs
Program 1: Creating Time Series Data
Procedure: Create a time series DataFrame.
```

In [46]:
```
#Code:

dates = pd.date_range(start='2023-01-01', periods=5, freq='D')
data = [1, 2, 3, 4, 5]
ts = pd.Series(data, index=dates)
print(ts)
```

```
2023-01-01    1
2023-01-02    2
2023-01-03    3
2023-01-04    4
2023-01-05    5
Freq: D, dtype: int64
```

In [47]:
```
#Program 2: Time Series Indexing
Procedure: Index a time series using specific dates.
```

In [48]:
```
#Code:

ts = pd.Series([1, 2, 3, 4, 5], index=pd.date_range('2023-01-01', periods=5))
print("Data on 2023-01-03:", ts['2023-01-03'])
```

```
Data on 2023-01-03: 3
```

In [49]:
```
#Program 3: Resampling Time Series Data
Procedure: Resample time series data to a different frequency.
```

In [50]:
```
#Code:

ts = pd.Series([1, 2, 3, 4, 5], index=pd.date_range('2023-01-01', periods=5, freq='D'))
resampled_ts = ts.resample('2D').sum()
print(resampled_ts)
```

```
2023-01-01    3
2023-01-03    7
2023-01-05    5
Freq: 2D, dtype: int64
```

In [51]:
```
#Program 4: Shifting Time Series Data
Procedure: Shift time series data forward or backward.
```

In [52]:
```
#Code:

ts = pd.Series([1, 2, 3, 4, 5], index=pd.date_range('2023-01-01', periods=5))
shifted_ts = ts.shift(1)
print(shifted_ts)
```

```
2023-01-01    NaN
2023-01-02    1.0
2023-01-03    2.0
2023-01-04    3.0
2023-01-05    4.0
Freq: D, dtype: float64
```

In [53]:
```
#Program 5: Rolling Window Calculation
Procedure: Calculate rolling mean on time series data.
```

In [54]:
```
#Code:

ts = pd.Series([1, 2, 3, 4, 5], index=pd.date_range('2023-01-01', periods=5))
```

```
rolling_mean = ts.rolling(window=3).mean()
print(rolling_mean)
```

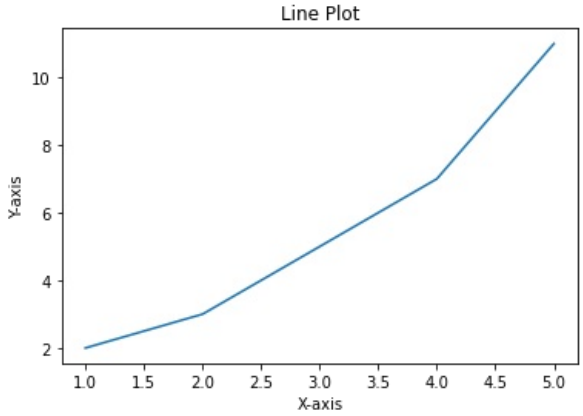
```
2023-01-01    NaN
2023-01-02    NaN
2023-01-03    2.0
2023-01-04    3.0
2023-01-05    4.0
Freq: D, dtype: float64
```

Week 4: Data Visualization (Different Types of Plotting)

In [55]:
```
#Aim
To explore various plotting techniques using Matplotlib and Seaborn for data visualization.
```

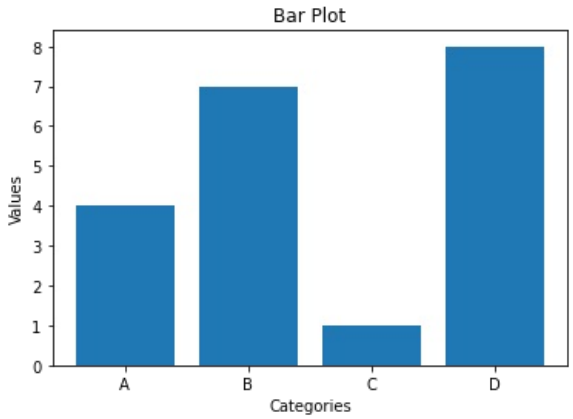
In [56]:
```
#Programs
#Program 1: Line Plot
Procedure: Create a simple line plot
```

In [57]:
```
#Code:

import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]
plt.plot(x, y)
plt.title("Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



In [58]:
```
#Program 2: Bar Plot
Procedure: Create a bar plot.
```

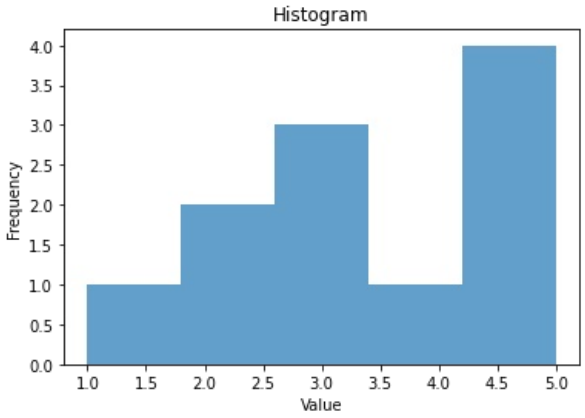
In [62]:
```
#Code:

categories = ['A', 'B', 'C', 'D']
values = [4, 7, 1, 8]
plt.bar(categories, values)
plt.title("Bar Plot")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.show()
```



In [63]:
```
#Program 3: Histogram
Procedure: Create a histogram to display frequency distribution.
```

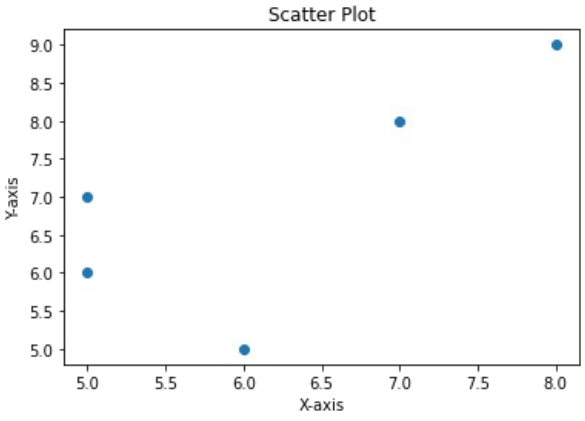
```
  Input In [63]
    Procedure: Create a histogram to display frequency distribution.
                                                                    ^
SyntaxError: invalid syntax
```

In [61]:
```
#Code:

data = [1, 2, 2, 3, 3, 3, 4, 5, 5, 5, 5]
plt.hist(data, bins=5, alpha=0.7)
plt.title("Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



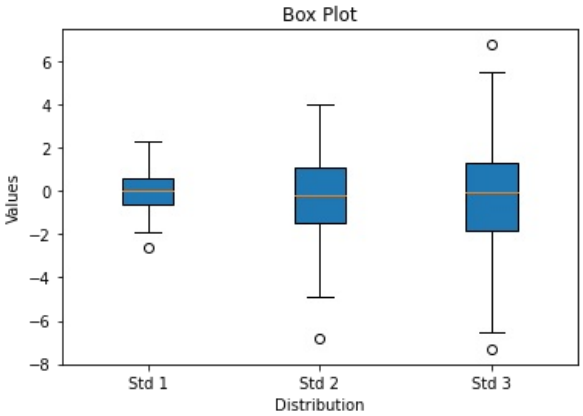
In [64]:
```
#Program 4: Scatter Plot
Procedure: Create a scatter plot.
```

In [65]:
```
#Code:

x = [5, 7, 8, 5, 6]
y = [7, 8, 9, 6, 5]
plt.scatter(x, y)
plt.title("Scatter Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



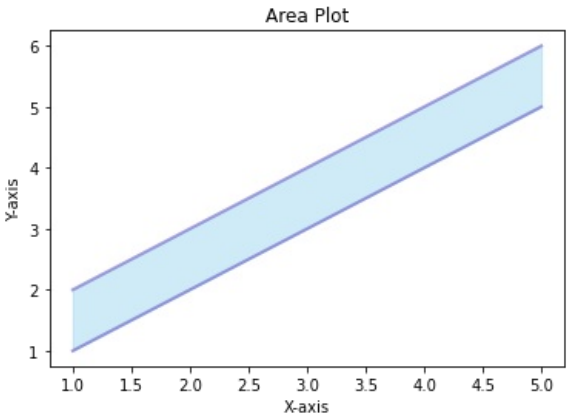
In [66]:
```
#Program 5: Box Plot
Procedure: Create a box plot to show data distribution.
```

In [67]:
```
#Code:

import numpy as np
import matplotlib.pyplot as plt
data = [np.random.normal(0, std, 100) for std in range(1, 4)]
plt.boxplot(data, vert=True, patch_artist=True)
plt.title("Box Plot")
plt.xlabel("Distribution")
plt.ylabel("Values")
plt.xticks([1, 2, 3], ['Std 1', 'Std 2', 'Std 3'])
plt.show()
```



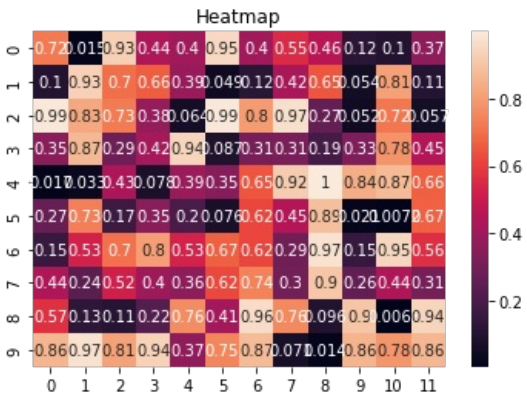
In [68]:
```
#Program 6: Area Plot
Procedure: Create an area plot.
```

In [69]:
```
#Code:

x = np.arange(1, 6)
y1 = [1, 2, 3, 4, 5]
y2 = [2, 3, 4, 5, 6]
plt.fill_between(x, y1, y2, color='skyblue', alpha=0.4)
plt.plot(x, y1, color='Slateblue', alpha=0.6, linewidth=2)
plt.plot(x, y2, color='Slateblue', alpha=0.6, linewidth=2)
plt.title("Area Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



In [70]:
```
#Program 7: Heatmap
Procedure: Create a heatmap to represent data values.
```

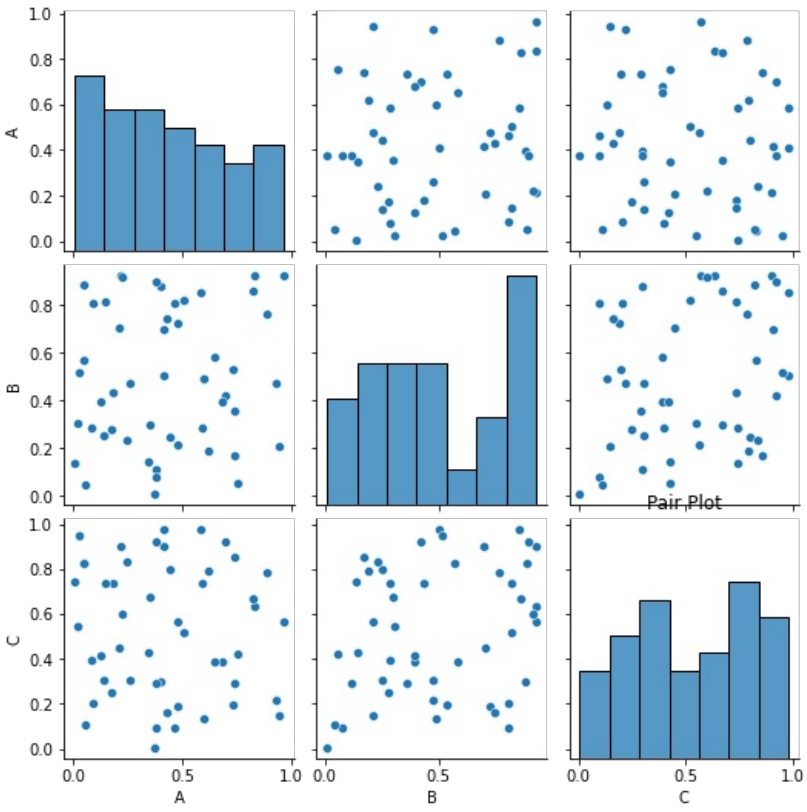
In [71]:
```
#Code:

import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
data = np.random.rand(10, 12)
sns.heatmap(data, annot=True)
plt.title("Heatmap")
plt.show()
```



In [72]:
```
#Program 8: Pair Plot
Procedure: Create a pair plot for pairwise relationships in a dataset.
```

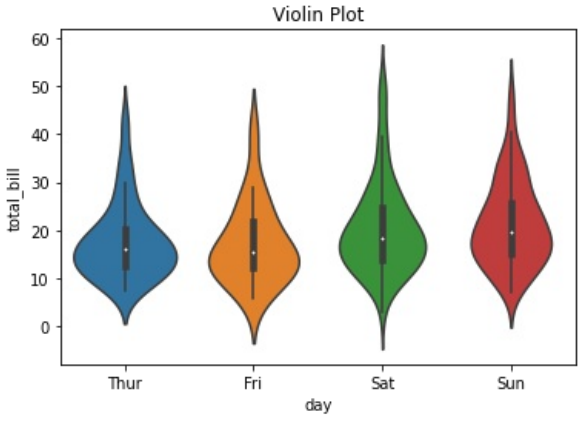
In [73]:
```
#Code:
```

```
import seaborn as sns
import pandas as pd
df = pd.DataFrame({
 'A': np.random.rand(50),
 'B': np.random.rand(50),
 'C': np.random.rand(50)
})
sns.pairplot(df)
plt.title("Pair Plot")
plt.show()
```



In [74]:
```
#Program 9: Violin Plot
Procedure: Create a violin plot to visualize the distribution of data across different categories.
```

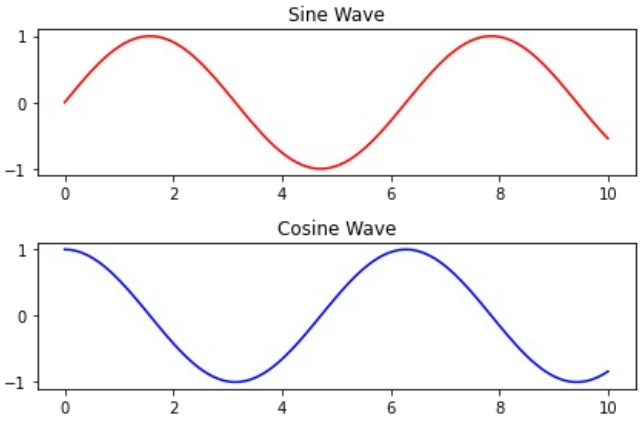
In [75]:
```
#Code:

import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset("tips")
sns.violinplot(x='day', y='total_bill', data=data)
plt.title("Violin Plot")
plt.show()
```



In [76]:
```
#Program 10: Subplots
"Procedure: Create multiple plots in a single figure.
```

In [77]:
```
#Code:

import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
fig, axs = plt.subplots(2, 1)
```

```
axs[0].plot(x, y1, 'r')
axs[0].set_title('Sine Wave')
axs[1].plot(x, y2, 'b')
axs[1].set_title('Cosine Wave')
plt.tight_layout()
plt.show()
```

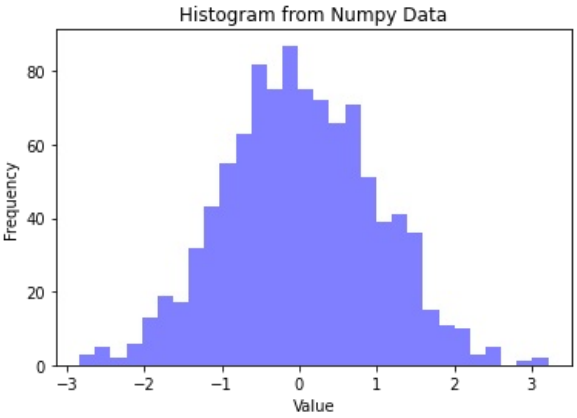


Week 5: Data Visualization Using Numpy (3 Programs)

In [78]:
```
#Aim
To visualize data using Numpy arrays with Matplotlib.
```

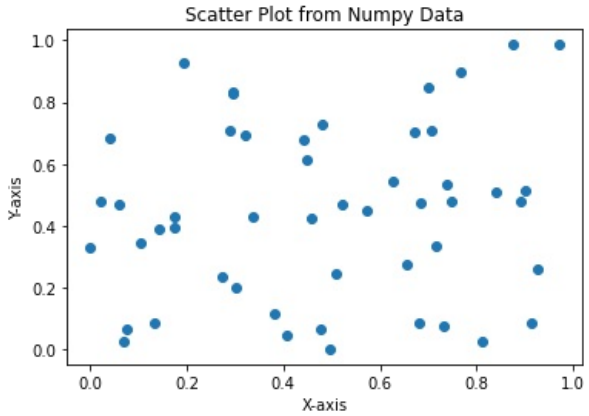
In [79]:
```
#Programs
Program 1: Numpy Histogram
Procedure: Create a histogram from Numpy data.
```

In [80]:
```
#Code:

import numpy as np
import matplotlib.pyplot as plt
data = np.random.randn(1000)
plt.hist(data, bins=30, alpha=0.5, color='blue')
plt.title("Histogram from Numpy Data")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



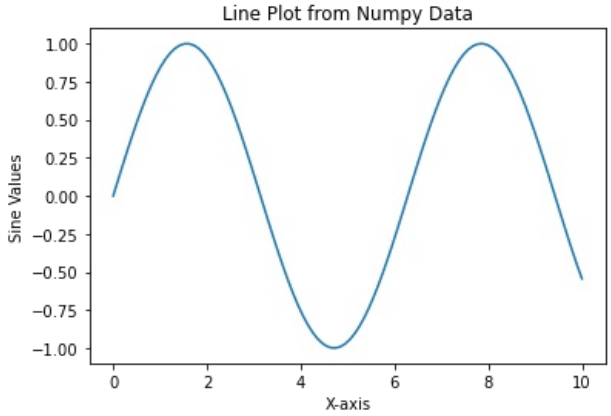
In [81]:
```
#Program 2: Numpy Scatter Plot
Procedure: Create a scatter plot using Numpy arrays.
```

In [82]:
```
#Code:

import numpy as np
import matplotlib.pyplot as plt
x = np.random.rand(50)
y = np.random.rand(50)
plt.scatter(x, y)
plt.title("Scatter Plot from Numpy Data")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



In [83]:
```
#Program 3: Numpy Line Plot
Procedure: Create a line plot using Numpy data.
```

In [84]:
```
#Code:

import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y)
plt.title("Line Plot from Numpy Data")
plt.xlabel("X-axis")
plt.ylabel("Sine Values")
plt.show()
```

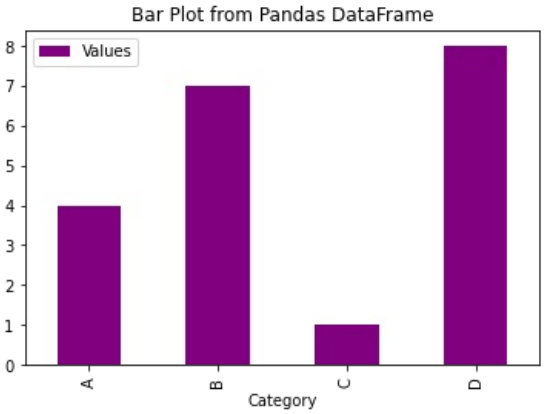


Week 6: Data Visualization Using Pandas (3 Programs)

In [85]:
```
#Aim
To perform data visualization using Pandas DataFrames.
```

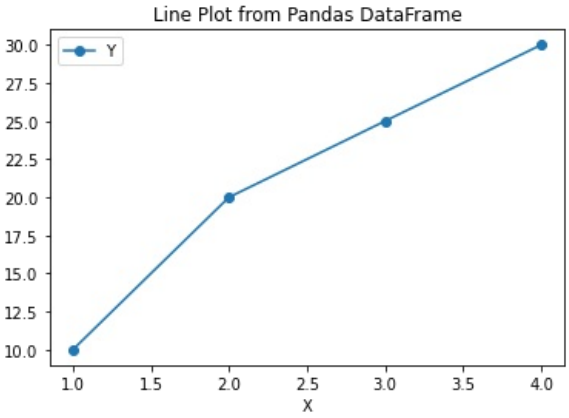
In [86]:
```
#Programs
Program 1: Bar Plot with Pandas
Procedure: Create a bar plot directly from a DataFrame.
```

In [87]:
```
#Code:

import pandas as pd
import matplotlib.pyplot as plt
df = pd.DataFrame({
 'Category': ['A', 'B', 'C', 'D'],
 'Values': [4, 7, 1, 8]
})
df.plot(kind='bar', x='Category', y='Values', color='purple')
plt.title("Bar Plot from Pandas DataFrame")
plt.show()
```



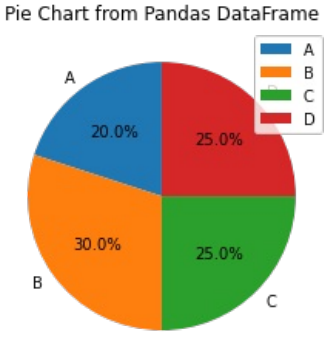
In [88]:
```
#Program 2: Line Plot with Pandas
Procedure: Create a line plot directly from a DataFrame.
```

In [89]:
```
#Code:

df = pd.DataFrame({
 'X': [1, 2, 3, 4],
 'Y': [10, 20, 25, 30]
})
df.plot(kind='line', x='X', y='Y', marker='o')
plt.title("Line Plot from Pandas DataFrame")
plt.show()
```



In [90]:
```
#Program 3: Pie Chart with Pandas
Procedure: Create a pie chart from a DataFrame.
```

In [91]:
```
#Code:

df = pd.DataFrame({
 'Categories': ['A', 'B', 'C', 'D'],
 'Values': [20, 30, 25, 25]
})
df.plot.pie(y='Values', labels=df['Categories'], autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart from Pandas DataFrame")
plt.ylabel('')
plt.show()
```



Week 7: How to Create CSV File and Save and Load

In [92]:
```
#Aim
To learn how to create, save, and load CSV files using Pandas.
```

In [93]:
```
#Programs
```

In [93]:
```
#Programs
Program 1: Creating and Saving a CSV File
Procedure: Create a DataFrame and save it as a CSV file.
```

In [94]:
```
#Code:

df = pd.DataFrame({
 'Name': ['Alice', 'Bob', 'Charlie'],
 'Age': [25, 30, 35],
 'City': ['New York', 'Los Angeles', 'Chicago']
})
df.to_csv('people.csv', index=False)
print("CSV file 'people.csv' created successfully.")
```

```
CSV file 'people.csv' created successfully.
```

In [95]:
```
#Program 2: Loading a CSV File
Procedure: Load the CSV file created in the previous step.
```

In [96]:
```
#Code:

loaded_df = pd.read_csv('people.csv')
print(loaded_df)
```

```
      Name  Age         City
0    Alice   25     New York
1      Bob   30  Los Angeles
2  Charlie   35      Chicago
```

In [97]:
```
#Program 3: Appending to a CSV File
Procedure: Append new data to an existing CSV file.
```

In [98]:
```
#Code:

new_data = pd.DataFrame({
 'Name': ['David', 'Eva'],
 'Age': [40, 22],
 'City': ['San Francisco', 'Seattle']
})
new_data.to_csv('people.csv', mode='a', header=False, index=False)
print("New data appended to 'people.csv'.")
```

```
New data appended to 'people.csv'.
```

In [99]:
```
#Program 4: Reading a CSV File with Different Delimiter
Procedure: Load a CSV file with a different delimiter (semicolon).
```

In [100…
```
#Code:

df = pd.read_csv('people.csv', delimiter=';')
print(df)
```

```
            Name,Age,City
0       Alice,25,New York
1      Bob,30,Los Angeles
2      Charlie,35,Chicago
3  David,40,San Francisco
4          Eva,22,Seattle
```

In [101…
```
#Program 5: Loading CSV File with Custom Column Names
Procedure: Load a CSV file and specify custom column names.
```

In [102…
```
#Code:

df = pd.read_csv('people.csv', names=['Full Name', 'Age', 'Location'], header=0)
print(df)
```

```
  Full Name  Age       Location
0     Alice   25       New York
1       Bob   30    Los Angeles
2   Charlie   35        Chicago
3     David   40  San Francisco
4       Eva   22        Seattle
```

Week 8: Heart Disease Program

In [119…
```
#Aim:To analyze a heart disease dataset and visualize relevant insights.
Programs:
#Program 1: Loading Heart Disease Dataset
#Procedure: Load the heart disease dataset and display its structure.
```

In [120…
```
import pandas as pd

# Load the dataset
df = pd.read_csv('heart_disease.csv')
```

```
# Print the first few rows of the dataset to get an overview
print(df.head())

# Check the dataset structure (columns and datatypes)
print(df.info())
```

```
   Unnamed: 0  Age  Sex  CP  Trestbps  Chol  FBS  Restecg  Thalach  Exang  \
0         NaN   63    1   1       145   233    1        2      150      0
1         NaN   37    1   2       130   250    0        2      187      0
2         NaN   41    0   1       130   204    0        2      172      0
3         NaN   56    1   1       120   236    0        2      178      0
4         NaN   57    0   2       120   354    0        0      163      0

   Oldpeak  Slope  CA  Thal  Target
0      2.3      3   0     6       1
1      3.5      3   0     3       1
2      1.4      2   0     3       1
3      0.8      2   0     3       1
4      0.6      2   0     3       1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  0 non-null      float64
 1   Age         5 non-null      int64
 2   Sex         5 non-null      int64
 3   CP          5 non-null      int64
 4   Trestbps    5 non-null      int64
 5   Chol        5 non-null      int64
 6   FBS         5 non-null      int64
 7   Restecg     5 non-null      int64
 8   Thalach     5 non-null      int64
 9   Exang       5 non-null      int64
 10  Oldpeak     5 non-null      float64
 11  Slope       5 non-null      int64
 12  CA          5 non-null      int64
 13  Thal        5 non-null      int64
 14  Target      5 non-null      int64
dtypes: float64(2), int64(13)
memory usage: 728.0 bytes
None
```

In [121…

```
# Generate summary statistics for the dataset
print(df.describe())
```

```
       Unnamed: 0        Age       Sex        CP    Trestbps        Chol  \
count         0.0   5.000000  5.000000  5.000000    5.000000    5.000000
mean          NaN  50.800000  0.600000  1.400000  129.000000  255.400000
std           NaN  11.189281  0.547723  0.547723   10.246951   57.600347
min           NaN  37.000000  0.000000  1.000000  120.000000  204.000000
25%           NaN  41.000000  0.000000  1.000000  120.000000  233.000000
50%           NaN  56.000000  1.000000  1.000000  130.000000  236.000000
75%           NaN  57.000000  1.000000  2.000000  130.000000  250.000000
max           NaN  63.000000  1.000000  2.000000  145.000000  354.000000

            FBS   Restecg    Thalach  Exang   Oldpeak     Slope   CA  \
count  5.000000  5.000000    5.00000    5.0  5.000000  5.000000  5.0
mean   0.200000  1.600000  170.00000    0.0  1.720000  2.400000  0.0
std    0.447214  0.894427   14.19507    0.0  1.194571  0.547723  0.0
min    0.000000  0.000000  150.00000    0.0  0.600000  2.000000  0.0
25%    0.000000  2.000000  163.00000    0.0  0.800000  2.000000  0.0
50%    0.000000  2.000000  172.00000    0.0  1.400000  2.000000  0.0
75%    0.000000  2.000000  178.00000    0.0  2.300000  3.000000  0.0
max    1.000000  2.000000  187.00000    0.0  3.500000  3.000000  0.0

           Thal  Target
count  5.000000     5.0
mean   3.600000     1.0
std    1.341641     0.0
min    3.000000     1.0
25%    3.000000     1.0
50%    3.000000     1.0
75%    3.000000     1.0
max    6.000000     1.0
```

In [122…

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

# Calculate the correlation matrix
correlation = df.corr()

# Visualize the correlation matrix using a heatmap
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix for Heart Disease Dataset")
plt.show()
```