```
# Data Toolkit Assignment
```

```
# Theory Questions
```

```
Start coding or generate with AI.
```

```
#Ans.1
#NumPy, short for Numerical Python, is a fundamental open-source library in Python for numerical computing.
#It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to c
```

```
#Ans.2
#Broadcasting in NumPy allows us to perform arithmetic operations on arrays of different shapes without reshaping them.
#It automatically adjusts the smaller array to match the larger array's shape by replicating its values along the necessary dimensions.
```

```
#Ans.3
#A Pandas DataFrame is a two-dimensional, size-mutable, and potentially heterogeneous tabular data structure in the Pandas library for F
#It is a fundamental data structure for data manipulation and analysis, widely used in data science and machine learning.
```

```
#Ans.4
#The pandas groupby() function is particularly useful for analyzing and summarizing large datasets, helping to identify patterns or anom
```

```
#Ans.5
#Its ability to simplify statistical plotting, integrate with pandas, and support a wide range of customization options makes it an esse
```

```
#Ans.6
#NumPy arrays are homogeneous, fixed-size, and stored in contiguous memory, enabling efficient math operations and vectorization impleme
#Python lists are heterogeneous, dynamic-size, storing pointers to Python objects, offering flexibility but slower performance and great
```

```
#Ans.7
#Heatmaps are a popular data visualization technique that uses color to represent different levels of data magnitude, allowing you to qu
```

```
#Ans.8
#Vectorization refers to improving the performance of operations on data, particularly with large data sets, by executing a single opera
```

```
#Ans.9
#Matplotlib: Is often preferred for academic or highly customized plots because you can fine-tune just about any aspect of the figure—fo
#Plotly: While still highly customizable, Plotly's real strength lies in interactivity and web-based visuals.
```

```
#Ans.10
#Hierarchical Indexing, also known as MultiIndexing, is a powerful feature in Pandas that allows you to have multiple levels of indexing
#This capability is particularly useful when dealing with high-dimensional data.
```

```
#Ans.11
#Seaborn's pairplot function is designed to create a grid of Axes such that each variable in the data will by shared across the y-axes a
#The primary use of pairplot is to visualize the distribution of single variables and the relationships between two variables.
```

```
#Ans.12
#The describe() method returns description of the data in the DataFrame.
#If the DataFrame contains numerical data, the description contains these information for each column: count -
#The number of not-empty values. mean - The average (mean) value. std - The standard deviation.
```

```
#Ans.13
#Handling missing values in Python Pandas is a crucial step in data cleaning and preparation.
#Pandas represents missing values primarily as NaN (Not a Number) for numerical data and None for object-type data,
#both of which are treated as missing by Pandas' isna() and notna() functions.
```

```
#Ans.14
#These visualizations allow us to easily understand any patterns, trends, or outliers in a data set.
#Data visualization also makes data accessible to the general public or specific audiences without technical knowledge.
```

```
#Ans.15
#Numpy arrays can have more than one dimension. One way to create such array is to start with a 1-dimensional array and use the numpy re
```

```
#Ans.16
#BBokeh is a Python library that is used to make highly interactive graphs and visualizations. This is done in bokeh using HTML and Java
#This makes it a powerful tool for creating projects, custom charts, and web design-based applications.
```

```
#Ans.17
#apply() : Works on both Series and DataFrames. It's your go-to when dealing with multiple columns or complex row-wise operations.
#map() : Specifically for Series. Perfect for simple, element-wise transformations
```

```
#Ans.18
#Pandas is primarily used for data analysis. It supports working with tabular data like CSV, Excel sheets, etc.
#NumPy, by default, supports data in the form of matrices and arrays since it is focused on numerical computations.
```

```
#Ans.19
#Pandas makes time series analysis intuitive and efficient by combining specialized date indexing with powerful tools for resampling, ro
#It's an essential toolbox for any temporal data workflow.
```

```
#Ans.20
#The main purpose of a pivot table is to summarize, analyze, and explore large datasets by rearranging and aggregating data into a more
#They allow users to quickly calculate, summarize, and analyze data, revealing patterns, trends, and relationships within the data.
```

```
#Ans.21
#NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently
```

```
#Ans.22
#Seaborn is a library mostly used for statistical plotting in Python.
#It is built on top of Matplotlib and provides beautiful default styles and color palettes to make statistical plots more attractive.
```

Start coding or generate with AI.

```
# Practical Questions
```

```
#Ans.1
import numpy as np

# Create a 2D NumPy array
array = np.array([[1, 2, 3],
                  [4, 5, 6],
                  [7, 8, 9]])

# Calculate the sum of each row
row_sums = np.sum(array, axis=1)

print("Original Array:")
print(array)
print("\nSum of Each Row:")
print(row_sums)
```

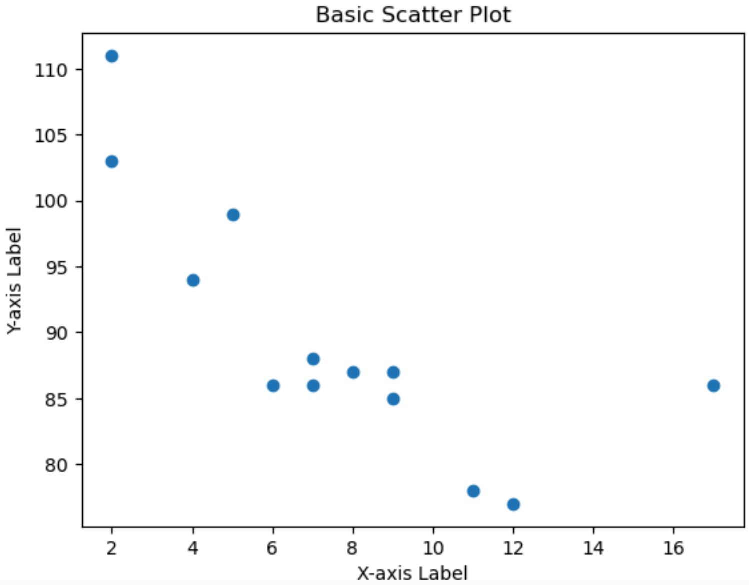
```
Original Array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Sum of Each Row:
[ 6 15 24]
```

```
#Ans.2
import pandas as pd

# Sample DataFrame
data = {'A': [10, 20, 30, 40],
        'B': [5, 15, 25, 35],
        'C': [1, 2, 3, 4]}

df = pd.DataFrame(data)

# Calculate the mean of column 'A'
mean_A = df['A'].mean()

print(f"Mean of column 'A': {mean_A}")
```

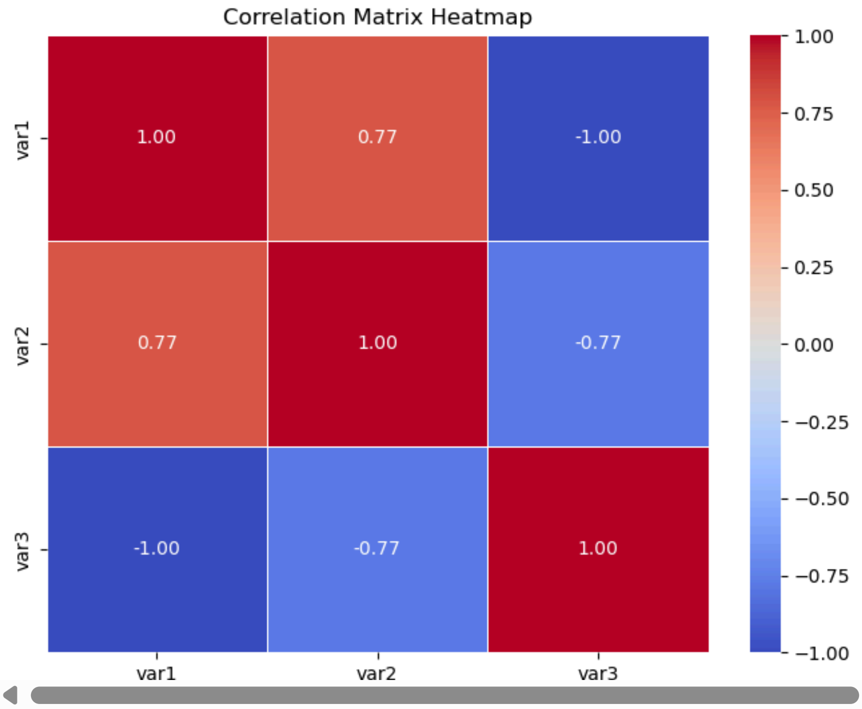
```
Mean of column 'A': 25.0
```

```
#Ans.3
import matplotlib.pyplot as plt
import numpy as np

# Sample data
x = np.array([5, 7, 8, 7, 2, 17, 2, 9, 4, 11, 12, 9, 6])
y = np.array([99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86])
```

```
# Create scatter plot
plt.scatter(x, y)

# Add labels and title
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.title('Basic Scatter Plot')

# Display the plot
plt.show()
```



```
#Ans.4
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have a Pandas DataFrame named 'data'
# Replace this with your actual data loading method
data = pd.DataFrame({'var1': [1, 2, 3, 4, 5],
                     'var2': [2, 4, 5, 4, 5],
                     'var3': [5, 4, 3, 2, 1]})

# Calculate the correlation matrix
correlation_matrix = data.corr()

# Create the heatmap
plt.figure(figsize=(8, 6))  # Adjust figure size as needed
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



```
#Ans.5
import plotly.express as px

# Sample data
data = {
    'Category': ['A', 'B', 'C', 'D'],
    'Value': [10, 15, 7, 12]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Create a bar plot
fig = px.bar(df, x='Category', y='Value', title='Category vs Value')

# Show the plot
fig.show()
```

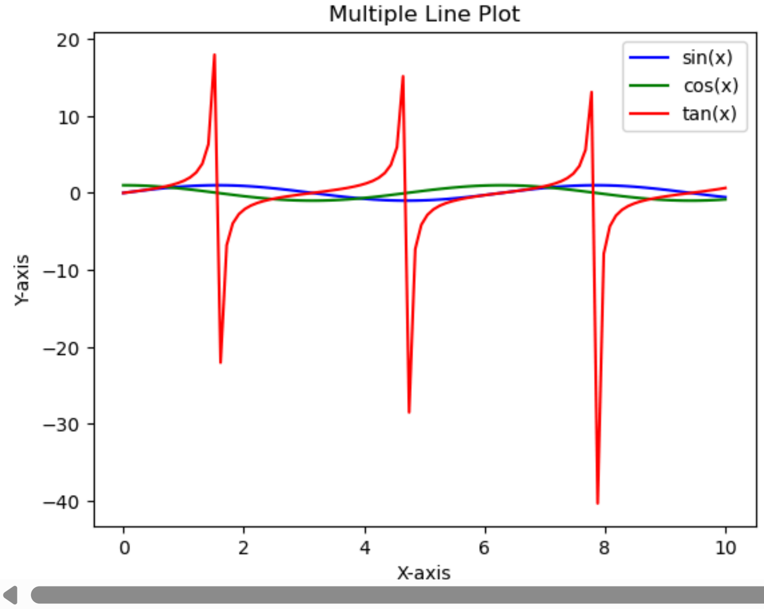
```
#Ans.6
import pandas as pd

# Sample data
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': [25, 30, 35]}

# Create DataFrame
df = pd.DataFrame(data)

# Add a new column 'AgeInMonths' based on 'Age'
df['AgeInMonths'] = df['Age'] * 12

print(df)
```

```
      Name  Age  AgeInMonths
0    Alice   25          300
1      Bob   30          360
2  Charlie   35          420
```

```
#Ans.7
import numpy as np

# Define two 1D arrays
array1 = np.array([1, 2, 3, 4])
array2 = np.array([5, 6, 7, 8])

# Method 1: Using the * operator
result_operator = array1 * array2

# Method 2: Using np.multiply()
result_multiply = np.multiply(array1, array2)

# Print the results
print("Element-wise multiplication using * operator:", result_operator)
print("Element-wise multiplication using np.multiply():", result_multiply)
```

```
Element-wise multiplication using * operator: [ 5 12 21 32]
Element-wise multiplication using np.multiply(): [ 5 12 21 32]
```

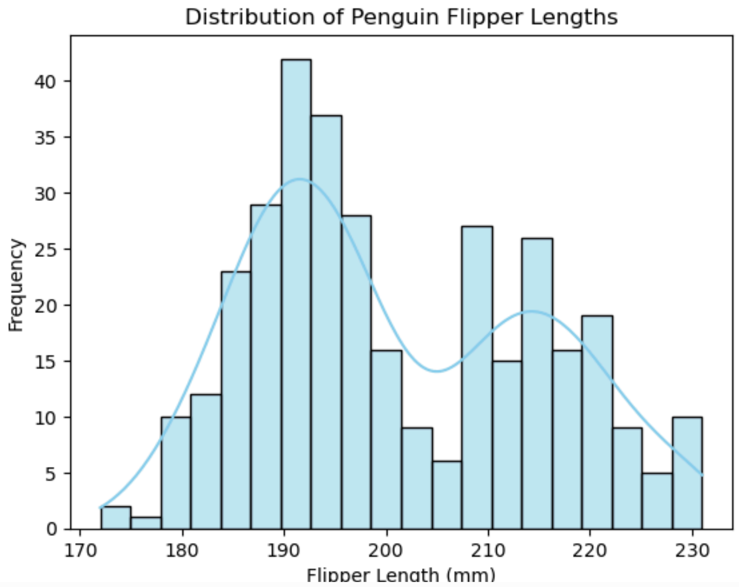
```
#Ans.8
import matplotlib.pyplot as plt
import numpy as np

# Sample data
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)

# Create line plots
plt.plot(x, y1, label='sin(x)', color='blue')
plt.plot(x, y2, label='cos(x)', color='green')
plt.plot(x, y3, label='tan(x)', color='red')

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Multiple Line Plot')

# Add a legend
plt.legend()

# Display the plot
plt.show()
```



```
#Ans.9
import pandas as pd

# Sample data
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25, 32, 45, 28],
    'Score': [85, 90, 78, 88]
}

# Create DataFrame
df = pd.DataFrame(data)

# Define threshold
threshold = 80

# Filter rows where 'Score' is greater than threshold
filtered_df = df[df['Score'] > threshold]

# Display the filtered DataFrame
print(filtered_df)
```

```
    Name  Age  Score
0  Alice   25     85
1    Bob   32     90
3  David   28     88
```

```
#Ans.10
import seaborn as sns
import matplotlib.pyplot as plt

# Load the 'penguins' dataset
penguins = sns.load_dataset("penguins")

# Create a histogram of flipper lengths
sns.histplot(penguins, x="flipper_length_mm", kde=True, color="skyblue", bins=20)

# Customize the plot
plt.title("Distribution of Penguin Flipper Lengths")
plt.xlabel("Flipper Length (mm)")
plt.ylabel("Frequency")

# Display the plot
plt.show()
```



```
#Ans.11
import numpy as np

# Define two 2D arrays (matrices)
A = np.array([[1, 2],
              [3, 4]])

B = np.array([[5, 6],
              [7, 8]])

# Method 1: Using np.matmul()
result = np.matmul(A, B)

# Method 2: Using the @ operator
result_operator = A @ B

# Method 3: Using np.dot()
result_dot = np.dot(A, B)

# Display the results
print("Matrix A:")
print(A)
print("\nMatrix B:")
print(B)
print("\nMatrix Product (np.matmul):")
print(result)
print("\nMatrix Product (@ operator):")
print(result_operator)
print("\nMatrix Product (np.dot):")
print(result_dot)
```

```
Matrix A:
[[1 2]
 [3 4]]

Matrix B:
[[5 6]
 [7 8]]

Matrix Product (np.matmul):
[[19 22]
 [43 50]]

Matrix Product (@ operator):
[[19 22]
 [43 50]]

Matrix Product (np.dot):
[[19 22]
 [43 50]]
```

```
#Ans.12
import pandas as pd

# Load the CSV file into a DataFrame
df = pd.read_csv('your_file.csv')
```

```
# Display the first 5 rows
print(df.head())


#Ans.13
import plotly.express as px

# Load sample data
df = px.data.iris()

# Create a 3D scatter plot
```