Python

11.Write a python program to create 2d array using NumPy

Title:
- Creating a 2D Array Using NumPy in Python

Objective:
- To create a 2D array using NumPy, which allows for efficient manipulation and mathematical operations on arrays.

Task:
- Import NumPy library.
- Create a 2D array with specific values or random values.
- Print the 2D array to verify its structure.

Code:
```
# Importing the numpy library
import numpy as np

# Creating a 2D array with specific values
array_2d = np.array([[1, 2, 3],
                     [4, 5, 6],
                     [7, 8, 9]])

# Alternatively, create a 2D array with random values (3x3 array)
# array_2d = np.random.randint(1, 10, (3, 3))

# Printing the 2D array
print("The 2D array is:")
print(array_2d)
```

Sample Output:
```
The 2D array is:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Commented [MOU1]: "Sample Output" is only for the reference don't include them in the practical record file.

12.Write a program to convert a python list to a NumPy array.

Title:
- Converting a Python List to a NumPy Array

Objective:
- To convert a Python list into a NumPy array, enabling more efficient mathematical operations and array manipulations.

Python

Task:

- Import the NumPy library.
- Create a Python list.
- Convert the list to a NumPy array using the np.array() method.
- Print the resulting NumPy array.

Code:

```
# Importing the numpy library
import numpy as np

# Creating a Python list
python_list = [1, 2, 3, 4, 5]

# Converting the Python list to a NumPy array
numpy_array = np.array(python_list)

# Printing the NumPy array
print("The NumPy array is:")
print(numpy_array)
```

Sample Output:

```
The NumPy array is:
[1 2 3 4 5]
```

Commented [MOU2]: "Sample Output" is only for the reference don't include them in the practical record file.

13.Write a program to create matrix of 4x5 from 11 to 30.

Title:

- Creating a 4x5 Matrix from 11 to 30 Using NumPy

Objective:

- To create a 4x5 matrix filled with numbers ranging from 11 to 30 using NumPy. This will allow you to explore matrix creation and indexing in NumPy.

Task:

- Import the NumPy library.
- Generate a sequence of numbers from 11 to 30.
- Reshape the sequence into a 4x5 matrix using np.reshape().
- Print the resulting 4x5 matrix.

Code:

```
# Importing the numpy library
import numpy as np
```

Python

```
# Creating a sequence of numbers from 11 to 30
numbers = np.arange(11, 30 + 1)

# Reshaping the sequence into a 3x3 matrix
matrix_4x5 = numbers.reshape(4, 5)

# Printing the resulting 3x3 matrix
print("The 4x5 matrix is:")
print(matrix_4x5)
```

Sample Output:

```
The 4x5 matrix is:
[[11 12 13 14 15]
 [16 17 18 19 20]
 [21 22 23 24 25]
 [26 27 28 29 30]]
```

Commented [MOU3]: "Sample Output" is only for the reference don't include them in the practical record file.

14.Write a program to create a data frame to store data of candidates who appeared in interviews. The dataframe columns are name, score, attempts, and qualify (Yes/No). Assign rowindex as C001,C002, and so on.

Title:

- Creating a DataFrame to Store Interview Data of Candidates

Objective:

- To create a pandas DataFrame to organize and store data of candidates who appeared in interviews. The DataFrame will contain candidate details such as name, score, number of attempts, and whether they qualify for the next round.

Task:

- Import the pandas library.
- Create a DataFrame to store candidate details.
- Define the columns as 'name', 'score', 'attempts', and 'qualify'.
- Assign custom row indices such as C001, C002, etc.
- Display the DataFrame to verify the results.

Code:

```
# Importing the pandas library
import pandas as pd

# Creating data for candidates
data = {
   'name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
   'score': [85, 72, 90, 65, 78],
```

Python

```
    'attempts': [1, 2, 1, 3, 2],
    'qualify': ['Yes', 'No', 'Yes', 'No', 'Yes']
}

# Assigning row indices as 'C001', 'C002', etc.
index_labels = ['C001', 'C002', 'C003', 'C004', 'C005']

# Creating the DataFrame
df = pd.DataFrame(data, index=index_labels)

# Printing the DataFrame
print("Data of candidates who appeared in interviews:")
print(df)
```

Sample Output:

```
Data of candidates who appeared in interviews:
          name  score  attempts qualify
C001    Alice     85         1     Yes
C002      Bob     72         2      No
C003  Charlie     90         1     Yes
C004    David     65         3      No
C005      Eve     78         2     Yes
```

Commented [MOU4]: "Sample Output" is only for the reference don't include them in the practical record file.

15.Write a program to create a dataframe named player and store their data in the columns like team, no. of matches runs, and average. Assign player name as row index and Display only those player details whose score is more than 1000.

Title:

- Creating a Player DataFrame and Displaying Players with Scores Above 1000

Objective:

- To create a pandas DataFrame to store information about players, including their team, number of matches, runs, and average score. The program filters and displays only the players whose total runs exceed 1000.

Task:

- Import the pandas library.
- Create a DataFrame named player with columns: 'team', 'no. of matches', 'runs', and 'average'.
- Set the player names as the row index.
- Display only the players whose total runs are greater than 1000

Code:

```
# Importing the pandas library
```

Python

```
import pandas as pd

# Creating data for players
data = {
   'team': ['Team A', 'Team B', 'Team A', 'Team C', 'Team B'],
   'no. of matches': [50, 120, 75, 100, 85],
   'runs': [1200, 950, 1800, 800, 1500],
   'average': [24.0, 20.8, 32.0, 18.5, 28.0]
}

# Player names as row index
index_labels = ['Player1', 'Player2', 'Player3', 'Player4', 'Player5']

# Creating the DataFrame
player = pd.DataFrame(data, index=index_labels)

# Display only players whose runs are greater than 1000
filtered_players = player[player['runs'] > 1000]

# Printing the filtered player details
print("Players with runs greater than 1000:")
print(filtered_players)
```

Sample Output:

```
Players with runs greater than 1000:
          team  no. of matches  runs  average
Player1  Team A              50  1200     24.0
Player3  Team A              75  1800     32.0
Player5  Team B              85  1500     28.0
```

Commented [MOU5]: "Sample Output" is only for the reference don't include them in the practical record file.

16.Write a program to represent the data on the ratings of mobile games on bar chart. The sample data is given as: Games: Pubg, FreeFire, MineCraft, GTA-V, Callofduty, FIFA 22. Rating: 4.5,4.8,4.7,4.6,4.1,4.3.

Title:

- Representing Mobile Game Ratings Using a Bar Chart

Objective:

- To visualize the ratings of different mobile games on a bar chart. This helps in easily comparing the ratings of the games.

Task:

- Import the necessary libraries (matplotlib and numpy).
- Define the list of games and their respective ratings.

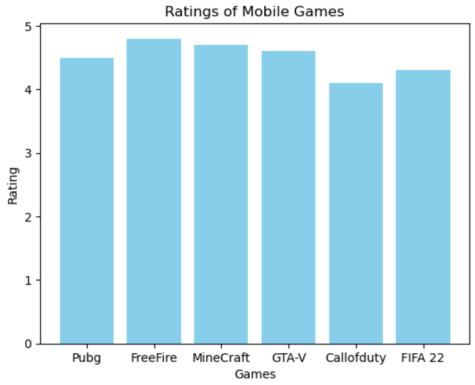
Python

- Use a bar chart to plot the ratings of each game.
- Display the chart with appropriate labels and title.

Code:

```
# Importing necessary libraries
import matplotlib.pyplot as plt

# Data for mobile games and their ratings
games = ['Pubg', 'FreeFire', 'MineCraft', 'GTA-V', 'Callofduty', 'FIFA 22']
ratings = [4.5, 4.8, 4.7, 4.6, 4.1, 4.3]

# Creating a bar chart
plt.bar(games, ratings, color='skyblue')

# Adding title and labels to the chart
plt.title('Ratings of Mobile Games')
plt.xlabel('Games')
plt.ylabel('Rating')

# Display the bar chart
plt.show()
```

Sample Output:



Commented [MOU6]: "Sample Output" is only for the reference don't include them in the practical record file.

17.Write a program to calculate variance and standard deviation for the given data: [33,44,55,67,54,22,33,44,56,78,21,31,43,90,21,33,44,55,87]

Title:

Python

- Calculating Variance and Standard Deviation

Objective:

- To calculate the variance and standard deviation for a given dataset. Variance measures the spread of data, and standard deviation is the square root of the variance, providing a measure of the average distance from the mean.

Task:

- Import the necessary library (numpy).
- Define the given dataset.
- Calculate the variance and standard deviation using the appropriate functions from numpy.
- Display the results.

Code:

```
# Importing the numpy library
import numpy as np

# Given data
data = [33, 44, 55, 67, 54, 22, 33, 44, 56, 78, 21, 31, 43, 90, 21, 33, 44, 55, 87]

# Calculating the variance
variance = np.var(data)

# Calculating the standard deviation
std_deviation = np.std(data)

# Printing the results
print("Variance:", variance)
print("Standard Deviation:", std_deviation)
```

Sample Output:

```
Variance: 416.5761772853186
Standard Deviation: 20.410197874722297
```

Commented [MOU7]: "Sample Output" is only for the reference don't include them in the practical record file.

18.Write a menu-driven program to calculate the mean, mode and median for the given data: [5,6,1,3,4,5,6,2,7,8,6,5,4,6,5,1,2,3,4]

Title:

- Menu-Driven Program to Calculate Mean, Mode, and Median

Objective:

Python

- To create a menu-driven program that calculates and displays the mean, mode, and median for a given dataset. These are fundamental statistical measures used to describe the central tendency of data.

Task:

- Create a menu that allows the user to select between calculating mean, mode, and median.
- Implement the functionality for each option using appropriate Python functions.
- Display the results based on the user's selection.

Code:

```
# Importing necessary libraries
import statistics

# Given dataset
data = [5, 6, 1, 3, 4, 5, 6, 2, 7, 8, 6, 5, 4, 6, 5, 1, 2, 3, 4]

# Function to calculate mean
def calculate_mean(data):
    return sum(data) / len(data)

# Function to calculate mode
def calculate_mode(data):
    return statistics.mode(data)

# Function to calculate median
def calculate_median(data):
    return statistics.median(data)

# Menu-driven program
def menu():
    while True:
        print("\nMenu:")
        print("1. Calculate Mean")
        print("2. Calculate Mode")
        print("3. Calculate Median")
        print("4. Exit")

        # Taking user input
        choice = int(input("Enter your choice: "))

        if choice == 1:
            mean = calculate_mean(data)
            print(f"Mean: {mean}")
```

Python

```
    elif choice == 2:
        mode = calculate_mode(data)
        print(f"Mode: {mode}")
    elif choice == 3:
        median = calculate_median(data)
        print(f"Median: {median}")
    elif choice == 4:
        print("Exiting the program...")
        break
    else:
        print("Invalid choice. Please try again.")

# Calling the menu function to run the program
menu()
```

Sample Output:

```
Menu:
1. Calculate Mean
2. Calculate Mode
3. Calculate Median
4. Exit
Enter your choice:  4
Exiting the program...
```

Commented [MOU8]: "Sample Output" is only for the reference don't include them in the practical record file.

19.Consider the following data of a cloths store and plot the data on the line chart and customize the chart as you wish:

Month	Jeans	T-Shirts	Shirts
March	1500	4400	6500
April	3500	4500	5000
May	6500	5500	5800
June	6700	6000	6300
July	6000	5600	6200
August	6800	6300	4500

Title:

- Plotting Sales Data of Clothes Store on a Line Chart

Objective:

- To represent the sales data of various clothing items (Jeans, T-Shirts, and Shirts) over several months on a line chart, and customize the chart with a title, axis labels, and legends to make it visually appealing.

Task:

- Define the data for each clothing item (Jeans, T-Shirts, and Shirts) for the months from March to August.

Python

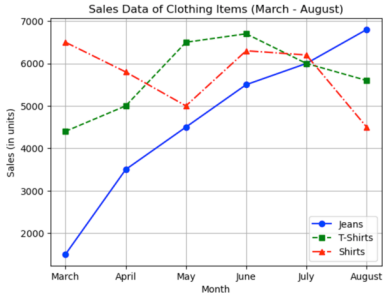
- Plot the sales data on a line chart.
- Customize the chart with appropriate labels for the axes, a title, and a legend.

Code:

```
# Importing necessary libraries
import matplotlib.pyplot as plt

# Data for each item in different months
months = ['March', 'April', 'May', 'June', 'July', 'August']
jeans_sales = [1500, 3500, 4500, 5500, 6000, 6800]
t_shirts_sales = [4400, 5000, 6500, 6700, 6000, 5600]
shirts_sales = [6500, 5800, 5000, 6300, 6200, 4500]

# Plotting the data
plt.plot(months, jeans_sales, label='Jeans', marker='o', linestyle='-', color='blue')
plt.plot(months, t_shirts_sales, label='T-Shirts', marker='s', linestyle='--', color='green')
plt.plot(months, shirts_sales, label='Shirts', marker='^', linestyle='-.', color='red')

# Customizing the chart
plt.title('Sales Data of Clothing Items (March - August)')
plt.xlabel('Month')
plt.ylabel('Sales (in units)')
plt.grid(True)
plt.legend()

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

Sample Output:

Commented [MOU9]: "Sample Output" is only for the reference don't include them in the practical record file.



Python

20.Observe the given data for monthly sales of one of the salesmen for 6 months. Plot them on the line chart.

Month	January	February	March	April	May	June
Sales	2500	2100	1700	3500	3000	3800

Apply the following customization to the chart:

- Give the title for the chart – “Sales Stats”
- Use the “Month” label for X-Axis and “Sales” for Y-Axis
- Display legends
- Use dashed lines with the width 5 point
- Use red color for the line
- Use dot marker with blue edge color and black fill color

Title:

- Sales Stats - Monthly Sales of a Salesman

Objective:

- To visualize the monthly sales data of a salesman on a line chart with specific customizations, including dashed lines, custom markers, legends, and labeled axes.

Task:

- Plot the sales data of a salesman for six months (January to June).
- Apply customizations such as:
- Title of the chart as "Sales Stats".
- X-Axis labeled as "Month".
- Y-Axis labeled as "Sales".
- Display legends for the plot.
- Use dashed lines with a width of 5 points.
- Set the line color to red.
- Use dot markers with blue edge color and black fill color.

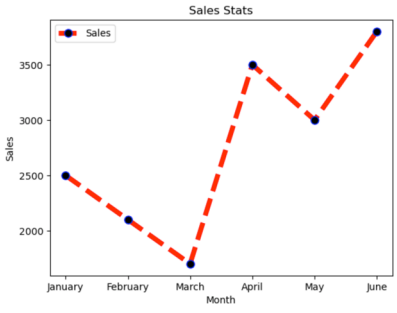
Code:

```
# Importing necessary libraries
import matplotlib.pyplot as plt

# Data for months and sales
months = ['January', 'February', 'March', 'April', 'May', 'June']
sales = [2500, 2100, 1700, 3500, 3000, 3800]

# Plotting the data
```

Python

```
plt.plot(months, sales, label='Sales', linestyle='--', color='red', linewidth=5, marker='o',
markersize=8, markeredgecolor='blue', markerfacecolor='black')

# Customizing the chart
plt.title('Sales Stats')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.legend()

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

Sample Code:



Commented [MOU10]: "Sample Output" is only for the reference don't include them in the practical record file.