BHARATIYA
VIDYA BHAVAN’SSARDAR PATEL INSTITUTE OF TECHNOLOGY
SARDAR PATEL INSTITUTE OF TECHNOLOGY
Bhavan’s Campus, Munshi Nagar, Andheri (West), Mumbai – 400058-India
Department of Computer Engineering

Name	**Debjit Ghosal**
UID no.	**2023300065**
Experiment No.	**3**

AIM: **NumPy**

Program 1

PROBLEM STATEMENT :

1) Problem Statement: You are given an array representing the scores of students in a class for a recent exam. Your tasks are: Given: scores of 10 students: scores = [88, 92, 79, 94, 85, 76, 90, 89, 77, 83] a. Create an Array: Create a numpy array with the given scores. b. Compute Statistics: a. Calculate the average (mean) score. b. Find the highest and lowest scores. c. Calculate the standard deviation of the scores to understand the variability. c. Modify the Array: a. Increase each score by 5 points to simulate a curve adjustment. b. Find the number of scores above the original average score

PROGRAM:

```
# Question 1

import numpy as np

scores = np.array([88, 92, 79, 94, 85, 76, 90, 89, 77, 83])

mean_score = np.mean(scores)
print(f"The average (Mean) Score is: {mean_score}")

highest_score = np.max(scores)
lowest_score = np.min(scores)
print(f"Highest Score: {highest_score}")
print(f"Lowest Score: {lowest_score}")

std_deviation = np.std(scores)
print(f"Standard Deviation: {std_deviation}")
added_scores = scores + 5
print(f"Added Scores: {added_scores}")
```

```
scores_above_mean = np.sum(scores > mean_score)
print(f"Number of Scores Above Original Average:
{scores_above_mean}")
```

RESULT:

```
The average (Mean) Score is: 85.3
Highest Score: 94
Lowest Score: 76
Standard Deviation: 6.034069936618236
Added Scores: [93 97 84 99 90 81 95 94 82 88]
Number of Scores Above Original Average: 5
```

Program 2

PROBLEM STATEMENT :

2) Problem Statement: You have two numpy arrays representing the scores of two different exams for the same group of students. Your tasks are: a. Create the Arrays: Define two arrays exam1 and exam2 with the given scores. b. Compute the Average Scores: Calculate the average score for each exam. c. Combine the Scores: Create a new array where each student's combined score is the sum of their scores in both exams. d. Find the Number of Students with a Combined Score Greater Than 160: Count how many students have a combined score exceeding 160. Given Data: exam1 = [78, 85, 92, 88, 76, 95, 89, 84] exam2 = [82, 90, 85, 91, 80, 92, 87, 86]

PROGRAM:

```
# Question 2
import numpy as np

exam1 = np.array([78, 85, 92, 88, 76, 95, 89, 84])
exam2 = np.array([82, 90, 85, 91, 80, 92, 87, 86])
avg1 = np.mean(exam1)
avg2 = np.mean(exam2)
```

```
print(f"Average Score for Exam 1: {avg1}")
print(f"Average Score for Exam 2: {avg2}")

comb = np.add(exam1 + exam2)
print(f"Combined Scores: {comb}")

students = np.sum(comb > 160)
print(f"Number of Students with a Combined Score Greater Than 160: {students}")
```

RESULT:

```
Average Score for Exam 1: 85.875
Average Score for Exam 2: 86.625
Combined Scores: [160 175 177 179 156 187 176 170]
Number of Students with a Combined Score Greater Than 160: 6
```

Program 3

PROBLEM STATEMENT:

3) Problem Statement: You are working with data from multiple sensors recorded over time. The data is in a 1D array, but you need to reshape it to analyze it as a matrix of sensor readings over multiple time steps. a. Create a 1D Sensor Data Array: Define a 1D array of length 60, where each value represents a sensor reading. b. Reshape to 3x4x5: Reshape this 1D array into a 3D array with shape 3x4x5, where 3 represents time steps, 4 represents different sensors, and 5 represents readings at each sensor. c. Display the Reshaped Array: Show the resulting 3D array.

PROGRAM:

```
# Question 3

import numpy as np

sensor = np.arange(1, 61)
print("1D Sensor Data Array:")
print(sensor)

reshaped = sensor.reshape(3, 4, 5)
print("\nReshaped 3D Array (3x4x5):")
print(reshaped)
```

RESULT:

```
1D Sensor Data Array:
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 49 50 51 52 53 54 55 56 57 58 59 60]

Reshaped 3D Array (3x4x5):
[[[ 1  2  3  4  5]
  [ 6  7  8  9 10]
  [11 12 13 14 15]
  [16 17 18 19 20]]

 [[21 22 23 24 25]
  [26 27 28 29 30]
  [31 32 33 34 35]
  [36 37 38 39 40]]

 [[41 42 43 44 45]
  [46 47 48 49 50]
  [51 52 53 54 55]
  [56 57 58 59 60]]]
```

Program 4

PROBLEM STATEMENT: 4) Write a NumPy program to create a 3x4 matrix filled with values from 10 to 21.

PROGRAM:

```
# Question 4

import numpy as np

values = np.arange(10, 22)

matrix = values.reshape(3, 4)

print("3x4 Matrix filled with values from 10 to 21:")
print(matrix)
```

RESULT:

```
3x4 Matrix filled with values from 10 to 21:
[[10 11 12 13]
 [14 15 16 17]
 [18 19 20 21]]
```

Program 5

PROBLEM STATEMENT:

5) Problem Statement: You have an array of temperatures recorded over a week. Your tasks are: a. Create the Array: Define a numpy array with the given temperatures. b. Find the Average Temperature: Calculate the average temperature over the week. c. Filter Temperatures Above Average: Create an array with temperatures that are above the average temperature. d. Find the Number of Days with Temperatures Below Freezing: Count how many days had temperatures below 0°C. Given temperatures = [12, 15, 11, 10, 9, 14, 13]

PROGRAM:

```
# Question 5

import numpy as np

temperatures = np.array([12, 15, 11, 10, 9, 14, 13])
print("Temperatures over the week were: ", temperatures)

mean = np.mean(temperatures)
print(f"Average (Mean) Temperatures: {mean}")

temp_above_mean = temperatures[temperatures > mean]
print(f"Number of Temperatures Above Average: {temp_above_mean}")

freezing = np.sum(temperatures < 0)
print("Number of days with temperatures below freezing:", freezing)
```

RESULT:

```
Temperatures over the week were:  [12 15 11 10  9 14 13]
Average (Mean) Temperatures: 12.0
Number of Temperatures Above Average: [15 14 13]
Number of days with temperatures below freezing: 0
```

Program 6

PROBLEM STATEMENT: 6) Problem Statement: You are working with grayscale images in a 2D matrix format. You need to prepare these images for machine learning models, which often require the data in a flattened 1D format. a. Create a Grayscale Image Matrix: Define a 4x4 grayscale image matrix with random values between 0 and 255. b. Flatten the Image Matrix: Convert the 2D image matrix into a 1D array. c. Display the Results: Show the original 2D image matrix and the flattened 1D array.

PROGRAM:

```
 # Question 6

import numpy as np
matrix = (4, 4)

image = np.random.randint(0, 256, size=matrix)
print("Original 4x4 Grayscale Image Matrix:")
print(image)

flat = image.flatten()
print("\nFlattened 1D Array:")
print(flat)
```

RESULT:

```
Original 4x4 Grayscale Image Matrix:
[[194  71  55   9]
 [244  39 206  27]
 [153  71 224 114]
 [253  99 135 249]]

Flattened 1D Array:
[194  71  55   9 244  39 206  27 153  71 224 114 253  99 135 249]
```

Program 7

PROBLEM STATEMENT: 7) Write a NumPy program to append values to the end of an array.

PROGRAM:

```
# Question 7

import numpy as np
array_1d = np.array([1, 2, 3, 4])
print("Original 1D Array:")
print(array_1d)
```

```
values = [5, 6, 7, 8, 9, 10]
arraynew = np.append(array_1d, values)
print("\n1D Array after appending values:")
print(arraynew)
```

RESULT:

```
Original 1D Array:
[1 2 3 4]

1D Array after appending values:
[ 1  2  3  4  5  6  7  8  9 10]
```

Program 8

PROBLEM STATEMENT: 8) Given a 2D array of shape (3, 5) and a 1D array of shape (3,). Write a Numpy program that transposes the 2D array and add the 1D array to each row of the transposed array

PROGRAM:

```
# Question 8

import numpy as np

array_2d = np.array([
    [1, 2, 3, 4, 5],
    [6, 7, 8, 9, 10],
    [11, 12, 13, 14, 15]
])
print("Original 2D Array (shape 3x5):")
print(array_2d)

array_1d = np.array([10, 20, 30])
print("\n1D Array (shape 3):")
print(array_1d)

transposed_array_2d = array_2d.T
print("\nTransposed 2D Array (shape 5x3):")
print(transposed_array_2d)

result = transposed_array_2d + array_1d
print("\nResult after adding the 1D array to each row of the transposed array:")
print(result)
```

RESULT:

```
Original 2D Array (shape 3x5):
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]]

1D Array (shape 3):
[10 20 30]

Transposed 2D Array (shape 5x3):
[[ 1  6 11]
 [ 2  7 12]
 [ 3  8 13]
 [ 4  9 14]
 [ 5 10 15]]

Result after adding the 1D array to each row of the transposed array:
[[11 26 41]
 [12 27 42]
 [13 28 43]
 [14 29 44]
 [15 30 45]]
```

Program 9

PROBLEM STATEMENT: 9) Write a NumPy program to create a 3D array of shape (3, 3, 3), then reshape it into a 1D array using ravel () and finally print the result.

PROGRAM:

```
# Question 9

import numpy as np
matrix = (3,3,3)

array1 = np.random.randint(0, 256, size=matrix)
print("3D Array ", array1)

flat = array1.ravel()
print("\nFlattened 1D Array:")
print(flat)
```

RESULT:

```
3D Array  [[[247   0 236]
  [147 179  59]
  [ 56 215 249]]

 [[123   5  54]
  [ 10  28 224]
  [242 124 234]]

 [[201 254 126]
  [112 115  92]
  [175 114 146]]]

Flattened 1D Array:
[247   0 236 147 179  59  56 215 249 123   5  54  10  28 224 242 124 234
 201 254 126 112 115  92 175 114 146]
```

Program 10

PROBLEM STATEMENT: 10) Write a NumPy program that creates a 1D array of 16 elements, reshape it to (4, 4), then change its shape to (2, 8) without changing the underlying data.

PROGRAM:

```
# Question 10

import numpy as np

array_1d = np.arange(1, 17)
print("Original 1D Array:")
print(array_1d)

array1 = array_1d.reshape(4, 4)
print("\nReshaped 2D Array (4x4):")
print(array1)

array2 = array1.reshape(2, 8)
print("\nReshaped 2D Array (2x8):")
print(array2)
```

RESULT:

```
Original 1D Array:
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]

Reshaped 2D Array (4x4):
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

Reshaped 2D Array (2x8):
[[ 1  2  3  4  5  6  7  8]
 [ 9 10 11 12 13 14 15 16]]
```

CONCLUSION:	Over the 10 NumPy experiments, I have gained expertise in several key areas. You created and manipulated arrays, calculated statistics like mean and standard deviation, and modified array values. You combined and filtered data, reshaped arrays into multidimensional formats for analysis, and worked with matrices, including flattening and transposing. You handled grayscale image data, converting it from 2D to 1D arrays. Lastly, you practiced advanced reshaping techniques using functions like `ravel()`, deepening your understanding of array transformations. These skills are essential for effective data manipulation and analysis in data science.