**CYCLE 2**

1. **Create a three dimensional array specifying float data type and print it.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

array\_3d = np.array([

[

[1.0, 2.0, 3.0],

[4.0, 5.0, 6.0]

],

[

[7.0, 8.0, 9.0],

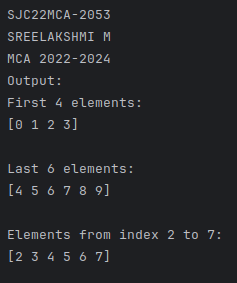
[10.0, 11.0, 12.0]

]

], dtype=float)

print(array\_3d)

**OUTPUT**



1. **Create a 2 dimensional array (2X3) with elements belonging to complex data type and print it. Also display**

**a. the no: of rows and columns**

**b. dimension of an array**

**c. reshape the same array to 3X2.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

array\_2d = np.array([[1 + 2j, 3 + 4j, 5 + 6j],

[7 + 8j, 9 + 10j, 11 + 12j]])

print("2D Array with Complex Data Type:")

print(array\_2d)

num\_rows, num\_cols = array\_2d.shape

print(f"Number of Rows: {num\_rows}")

print(f"Number of Columns: {num\_cols}")

dimensions = array\_2d.shape

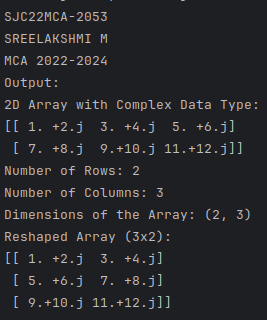
print("Dimensions of the Array:", dimensions)

array\_reshaped = array\_2d.reshape(3, 2)

print("Reshaped Array (3x2):")

print(array\_reshaped)

**OUTPUT**



1. **Familiarize with the functions to create**

**a) an uninitialized array**

**b) array with all elements as 1**

**c) all elements as 0.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

uninitialized\_array = np.empty((2, 3))

print("Uninitialized Array:")

print(uninitialized\_array)

ones\_array = np.ones((2, 3))

print("Array with All Elements as 1:")

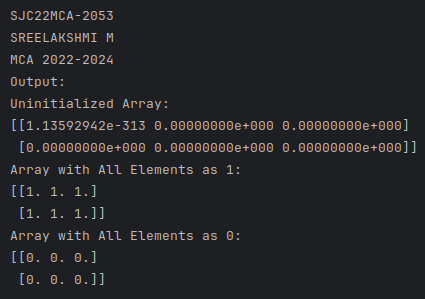
print(ones\_array)

zeros\_array = np.zeros((2, 3))

print("Array with All Elements as 0:")

print(zeros\_array)

**OUTPUT**

****

1. **Create an one dimensional array using arange function containing 10 elements. Display**

**a. First 4 elements**

**b. Last 6 elements**

**c. Elements from index 2 to 7**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

arr = np.arange(10)

print("First 4 elements:")

print(arr[:4])

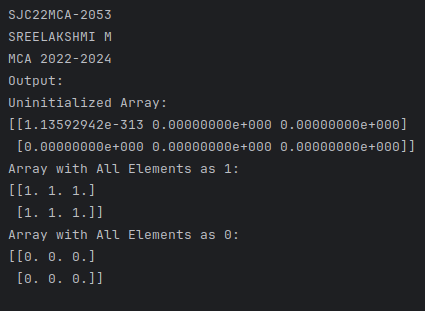
print("\nLast 6 elements:")

print(arr[4:])

print("\nElements from index 2 to 7:")

print(arr[2:8])

**OUTPUT**



1. **Create an 1D array with arange containing first 15 even numbers as elements**

**a. Elements from index 2 to 8 with step 2(also demonstrate the same using**

**slice function)**

**b. Last 3 elements of the array using negative index**

**c. Alternate elements of the array**

**d. Display the last 3 alternate elements.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

even\_numbers = np.arange(2, 31, 2)

slice\_a = even\_numbers[2:9:2]

print("a. Elements from index 2 to 8 with step 2:", slice\_a)

last\_3\_elements = even\_numbers[-3:]

print("b. Last 3 elements of the array using negative index:", last\_3\_elements)

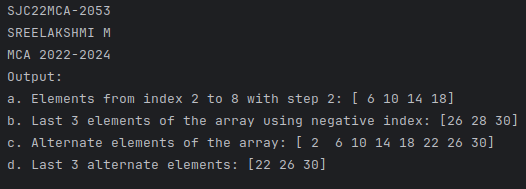
alternate\_elements = even\_numbers[::2]

print("c. Alternate elements of the array:", alternate\_elements)

last\_3\_alternate = alternate\_elements[-3:]

print("d. Last 3 alternate elements:", last\_3\_alternate)

**OUTPUT**

****

1. **Create a 2 Dimensional array with 4 rows and 4 columns.**

**a. Display all elements excluding the first row**

**b. Display all elements excluding the last column**

**c. Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row**

**d. Display the elements of 2 nd and 3 rd column**

**e. Display 2 nd and 3 rd element of 1 st row**

**f. Display the elements from indices 4 to 10 in descending order(use**

**–values).**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

array\_2d = np.array([[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

[13, 14, 15, 16]])

elements\_excluding\_first\_row = array\_2d[1:]

print("a. Elements excluding the first row:\n", elements\_excluding\_first\_row)

elements\_excluding\_last\_column = array\_2d[:, :-1]

print("b. Elements excluding the last column:\n", elements\_excluding\_last\_column)

elements\_1st\_2nd\_col\_2nd\_3rd\_row = array\_2d[1:3, 0:2]

print("c. Elements of the 1st and 2nd column in the 2nd and 3rd row:\n", elements\_1st\_2nd\_col\_2nd\_3rd\_row)

elements\_2nd\_3rd\_col = array\_2d[:, 1:3]

print("d. Elements of the 2nd and 3rd column:\n", elements\_2nd\_3rd\_col)

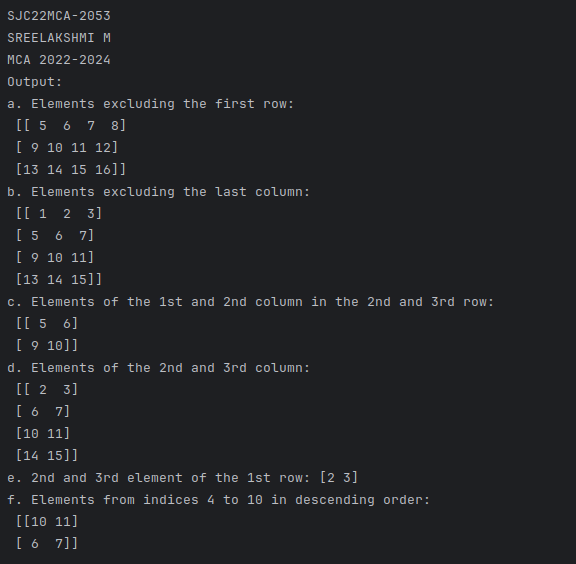
elements\_2nd\_3rd\_1st\_row = array\_2d[0, 1:3]

print("e. 2nd and 3rd element of the 1st row:", elements\_2nd\_3rd\_1st\_row)

elements\_descending\_order = array\_2d[1:3, 1:3][::-1]

print("f. Elements from indices 4 to 10 in descending order:\n", elements\_descending\_order)

**OUTPUT**

****

1. **Create two 2D arrays using array object and**

**a. Add the 2 matrices and print it**

**b. Subtract 2 matrices**

**c. Multiply the individual elements of matrix**

**d. Divide the elements of the matrices**

**e. Perform matrix multiplication**

**f. Display transpose of the matrix**

**g. Sum of diagonal elements of a matrix.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

matrix1 = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

matrix2 = np.array([[9, 8, 7],

[6, 5, 4],

[3, 2, 1]])

matrix\_sum = matrix1 + matrix2

print("a. Sum of two matrices:\n", matrix\_sum)

matrix\_diff = matrix1 - matrix2

print("b. Difference of two matrices:\n", matrix\_diff)

elementwise\_product = matrix1 \* matrix2

print("c. Elementwise product of two matrices:\n", elementwise\_product)

elementwise\_division = matrix1 / matrix2

print("d. Elementwise division of two matrices:\n", elementwise\_division)

matrix\_product = np.dot(matrix1, matrix2)

print("e. Matrix multiplication of two matrices:\n", matrix\_product)

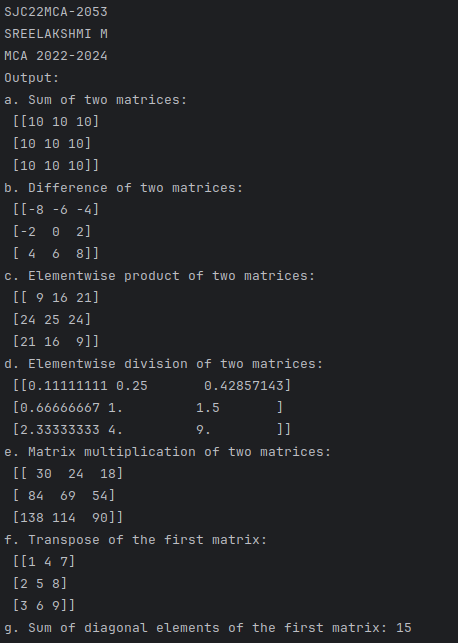
matrix1\_transpose = np.transpose(matrix1)

print("f. Transpose of the first matrix:\n", matrix1\_transpose)

diagonal\_sum = np.trace(matrix1)

print("g. Sum of diagonal elements of the first matrix:", diagonal\_sum)

**OUTPUT**



1. **Demonstrate the use of insert() function in 1D and 2D array.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

# Create a 1D array

arr\_1d = np.array([1, 2, 3, 4, 5])

new\_arr\_1d = np.insert(arr\_1d, 2, 6)

print("Original 1D Array:", arr\_1d)

print("New 1D Array after insertion:", new\_arr\_1d)

# Create a 2D array

arr\_2d = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

new\_row = np.array([10, 11, 12])

new\_arr\_2d = np.insert(arr\_2d, 1, new\_row, axis=0)

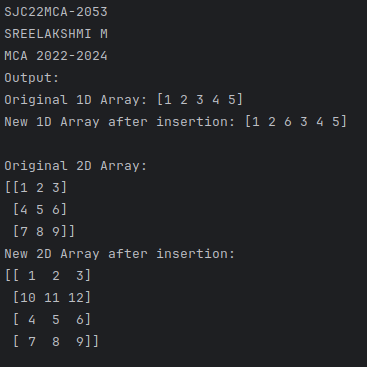
print("\nOriginal 2D Array:")

print(arr\_2d)

print("New 2D Array after insertion:")

print(new\_arr\_2d)

**OUTPUT**



1. **Demonstrate the use of diag() function in 1D and 2D array.(use both square matrix and matrix with different dimensions).**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

# Create a 1D array

arr\_1d = np.array([1, 2, 3, 4, 5])

diagonal\_1d = np.diag(arr\_1d)

print("Original 1D Array:")

print(arr\_1d)

print("\nDiagonal Array from 1D Array:")

print(diagonal\_1d)

# Create a square matrix

square\_matrix = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

diagonal\_square = np.diag(square\_matrix)

print("Original Square Matrix:")

print(square\_matrix)

print("\nDiagonal Elements of Square Matrix:")

print(diagonal\_square)

# Create a matrix with different dimensions

matrix = np.array([[1, 2, 3],

[4, 5, 6]])

diagonal\_matrix = np.diag(matrix)

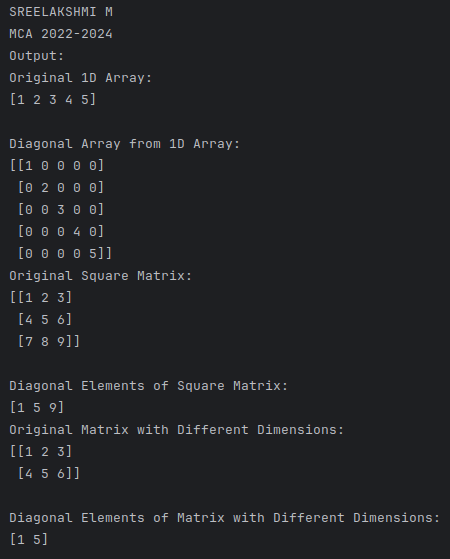
print("Original Matrix with Different Dimensions:")

print(matrix)

print("\nDiagonal Elements of Matrix with Different Dimensions:")

print(diagonal\_matrix)

**OUTPUT**



1. **Create a square matrix with random integer values(use randint()) and use**

**appropriate functions to find:**

**i) inverse**

**ii) rank of matrix**

**iii) Determinant**

**iv) transform matrix into 1D array**

**v) eigen values and vectors.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

n = 4

matrix = np.random.randint(1, 10, size=(n, n))

print("Matrix:")

print(matrix)

# i) Inverse of the matrix (if it's invertible)

try:

matrix\_inverse = np.linalg.inv(matrix)

print("\nInverse Matrix:")

print(matrix\_inverse)

except np.linalg.LinAlgError:

print("\nThe matrix is not invertible.")

# ii) Rank of the matrix

rank = np.linalg.matrix\_rank(matrix)

print("\nRank of the Matrix:", rank)

# iii) Determinant of the matrix

determinant = np.linalg.det(matrix)

print("\nDeterminant of the Matrix:", determinant)

# iv) Transform matrix into a 1D array

matrix\_1d = matrix.flatten()

print("\nMatrix as a 1D array:")

print(matrix\_1d)

# v) Eigenvalues and Eigenvectors

eigenvalues, eigenvectors = np.linalg.eig(matrix)

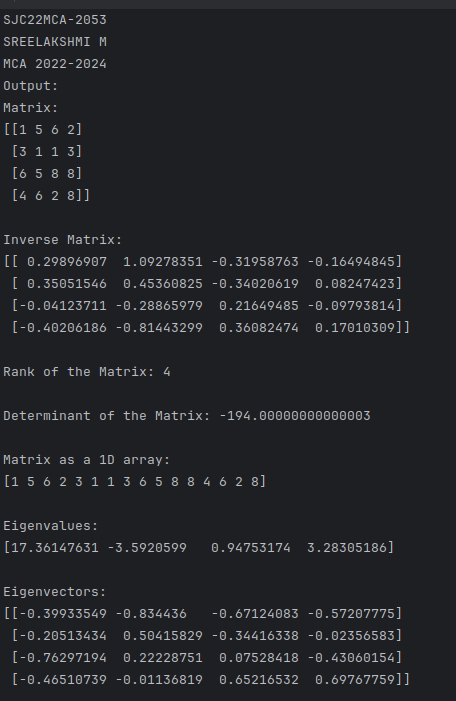
print("\nEigenvalues:")

print(eigenvalues)

print("\nEigenvectors:")

print(eigenvectors)

**OUTPUT**

****

1. **Create a matrix X with suitable rows and columns**

**i) Display the cube of each element of the matrix using different**

**methods(use multiply(), \*, power(),\*\*)**

**ii) Display identity matrix of the given square matrix.**

**iii) Display each element of the matrix to different powers.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

# Different methods

X = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

cubed\_matrix\_multiply = np.multiply(X, np.multiply(X, X))

cubed\_matrix\_multiply\_operator = X \* X \* X

cubed\_matrix\_power = np.power(X, 3)

cubed\_matrix\_double\_star = X \*\* 3

print("Cube of each element using multiply():\n", cubed\_matrix\_multiply)

print("\nCube of each element using \* operator:\n", cubed\_matrix\_multiply\_operator)

print("\nCube of each element using power():\n", cubed\_matrix\_power)

print("\nCube of each element using \*\* operator:\n", cubed\_matrix\_double\_star)

# Identity matrix for a square matrix

identity\_matrix = np.identity(X.shape[0])

print("\nIdentity matrix of X:\n", identity\_matrix)

# Display each element of the matrix to different powers

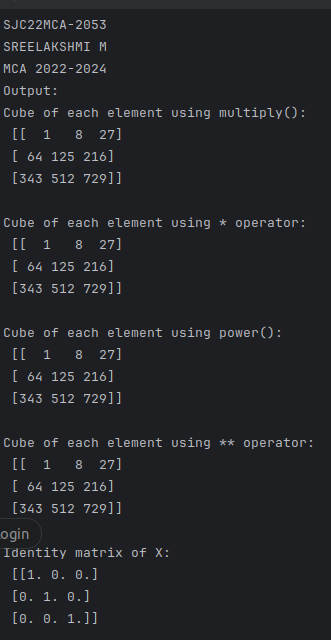
squared\_matrix = np.power(X, 2)

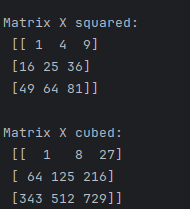
cubed\_matrix = np.power(X, 3)

print("\nMatrix X squared:\n", squared\_matrix)

print("\nMatrix X cubed:\n", cubed\_matrix)

**OUTPUT**

****

****

1. **Create a matrix Y with same dimension as X and perform the operation X^2+2Y.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

X = np.array([[2, 3, 4],

[5, 6, 7],

[8, 9, 10]])

Y = np.array([[1, 2, 3],

[4, 5, 6],

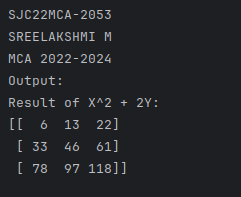
[7, 8, 9]])

result = X\*\*2 + 2\*Y

print("Result of X^2 + 2Y:")

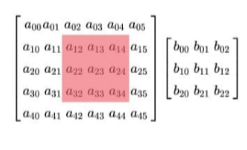
print(result)

**OUTPUT**

****

1. **Define matrices A with dimension 5x6 and B with dimension 3x3.**

**Extract a sub matrix of dimension 3x3 from A and multiply it with B. Replace the extracted sub matrix in A with the matrix obtained after multiplication.**

****

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

A = np.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]])

B = np.array([[2, 3, 4],

[5, 6, 7],

[8, 9, 10]])

sub\_matrix = A[:3, :3]

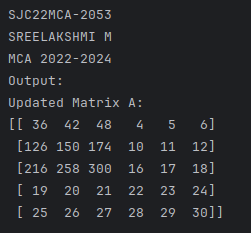
result = np.dot(sub\_matrix, B)

A[:3, :3] = result

print("Updated Matrix A:")

print(A)

**OUTPUT**

****

1. **Given 3 Matrices A, B and C. Write a program to perform matrix multiplication of the 3 matrices.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

# Define matrices A, B, and C

A = np.array([[1, 2],

[3, 4]])

B = np.array([[5, 6],

[7, 8]])

C = np.array([[9, 10],

[11, 12]])

# Perform matrix multiplication: A \* B \* C

result\_matrix = np.dot(np.dot(A, B), C)

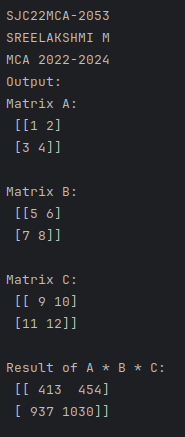
print("Matrix A:\n", A)

print("\nMatrix B:\n", B)

print("\nMatrix C:\n", C)

print("\nResult of A \* B \* C:\n", result\_matrix)

**OUTPUT**

****

1. **Write a program to check whether given matrix is symmetric or Skew Symmetric.**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

def is\_symmetric(matrix):

return np.array\_equal(matrix, matrix.T)

def is\_skew\_symmetric(matrix):

return np.array\_equal(matrix, -matrix.T)

matrix = np.array([[0, 1, -2],

[-1, 0, 3],

[2, -3, 0]])

if is\_symmetric(matrix):

print("The matrix is symmetric.")

elif is\_skew\_symmetric(matrix):

print("The matrix is skew-symmetric.")

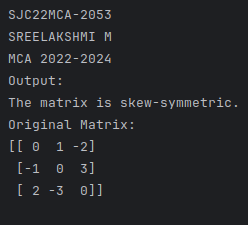
else:

print("The matrix is neither symmetric nor skew-symmetric.")

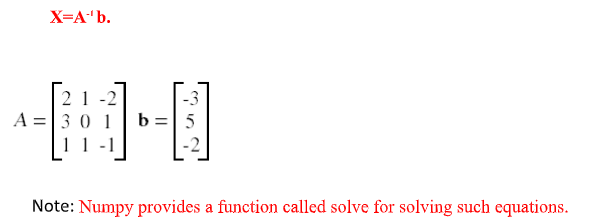
print("Original Matrix:")

print(matrix)

**OUTPUT**

****

1. **Given a matrix-vector equation AX=b. Write a program to find out the value of X using solve(), given A and b as below**

****

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

A = np.array([[2, 1],

[3, 4]])

b = np.array([5, 7])

X = np.linalg.solve(A, b)

print("Matrix A:")

print(A)

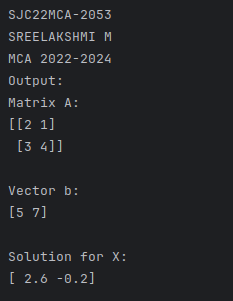
print("\nVector b:")

print(b)

print("\nSolution for X:")

print(X)

**OUTPUT**

****

1. **Write a program to perform the SVD of a given matrix A. Also reconstruct**

**the given matrix from the 3 matrices obtained after performing SVD.**

**Use the function: numpy.linalg.svd()**

import numpy as np

print("SJC22MCA-2053 \nSREELAKSHMI M \nMCA 2022-2024 \nOutput: ")

A = np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

U, S, VT = np.linalg.svd(A)

reconstructed\_A = np.dot(U, np.dot(np.diag(S), VT))

print("Original Matrix A:")

print(A)

print("\nU Matrix:")

print(U)

print("\nS Matrix (Diagonal Matrix):")

print(np.diag(S))

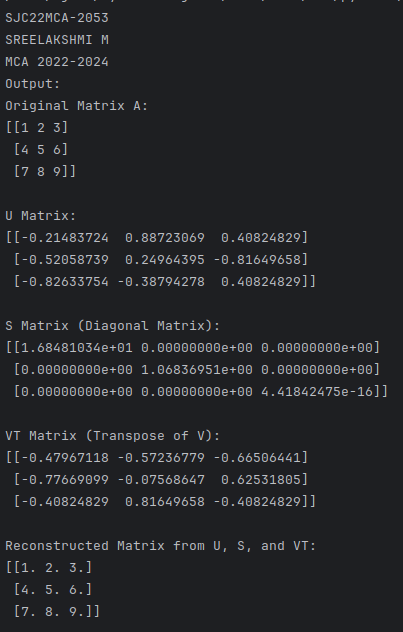
print("\nVT Matrix (Transpose of V):")

print(VT)

print("\nReconstructed Matrix from U, S, and VT:")

print(reconstructed\_A)

**OUTPUT**

****