DATA SCIENCE LAB

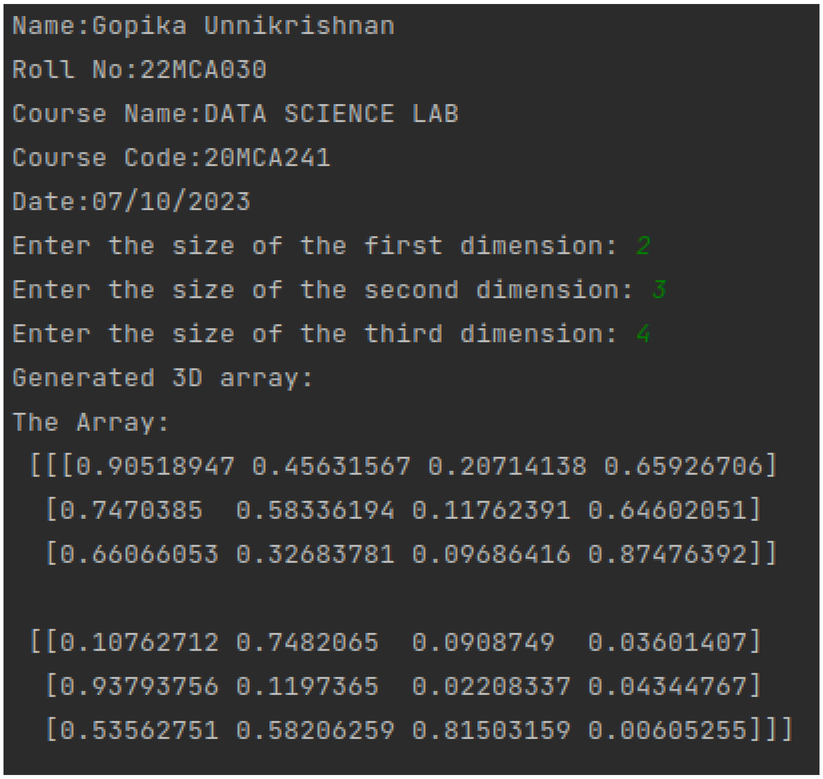
CYCLE-2

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

CODE:

```
import numpy as np
try:
    dim1 = int(input("Enter the size of the first dimension: "))
    dim2 = int(input("Enter the size of the second dimension: "))
    dim3 = int(input("Enter the size of the third dimension: "))
except ValueError:
    print("Please enter valid integer values for dimensions.")
    exit()
array_3d = np.random.rand(dim1, dim2, dim3).astype(float)
print("Generated 3D array:")
print("The Array:",array_3d)
```

OUTPUT:



```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:07/10/2023
Enter the size of the first dimension: 2
Enter the size of the second dimension: 3
Enter the size of the third dimension: 4
Generated 3D array:
The Array:
 [[[0.90518947 0.45631567 0.20714138 0.65926706]
  [0.7470385  0.58336194 0.11762391 0.64602051]
  [0.66066053 0.32683781 0.09686416 0.87476392]]

 [[0.10762712 0.7482065  0.0908749  0.03601407]
  [0.93793756 0.1197365  0.02208337 0.04344767]
  [0.53562751 0.58206259 0.81503159 0.00605255]]]
```

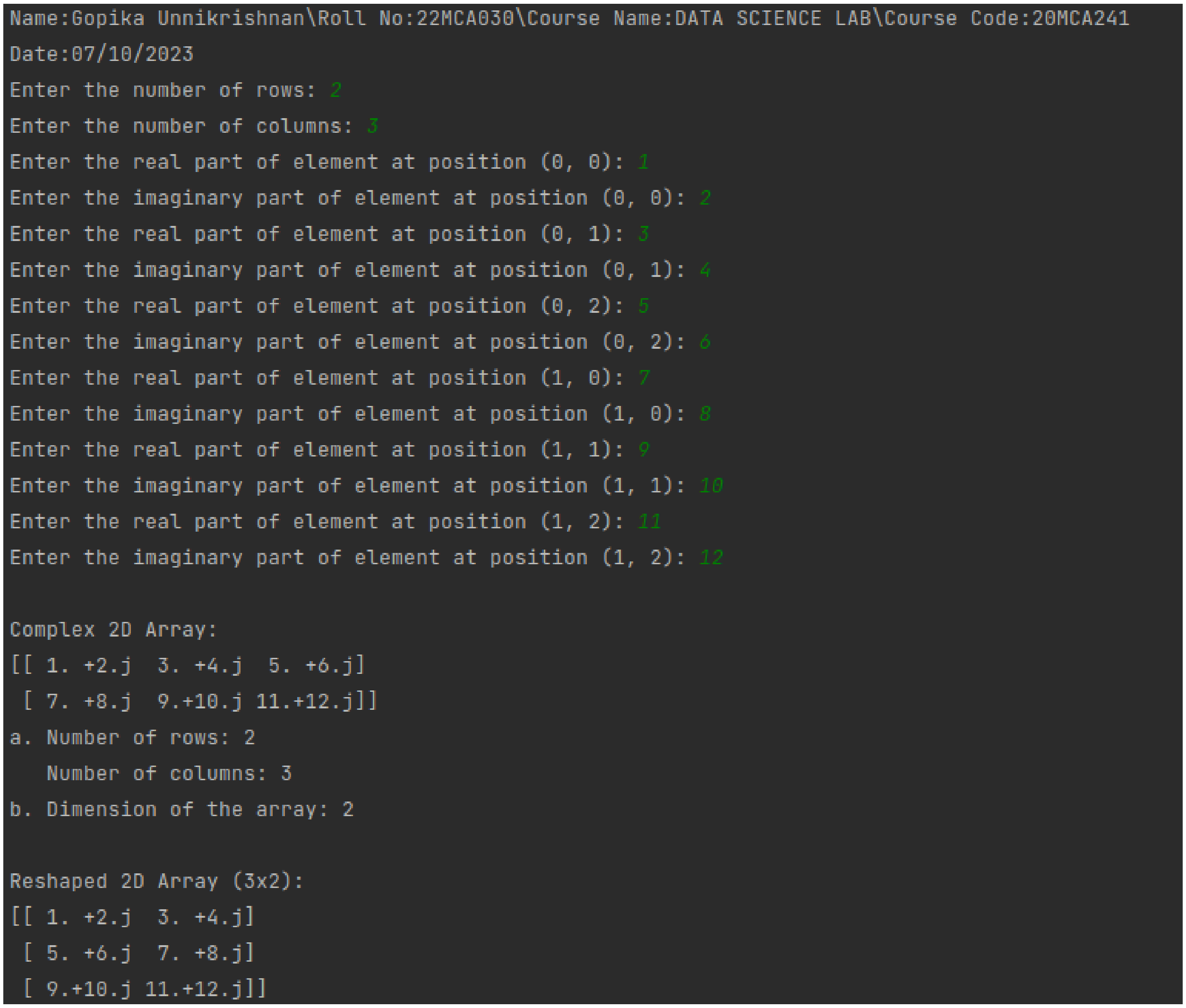
2. Create a 2 dimensional array (2X3) with elements belonging to complex data type and print it. Also Display

- **a) the no: of rows and columns**
- **b) dimension of an array**
- **c) reshape the same array to 3X2**

CODE:

```
import numpy as np
try:
   print(
      "Name:Gopika Unnikrishnan\Roll No:22MCA030\Course Name:DATA SCIENCE
LAB\Course Code:20MCA241\nDate:07/10/2023")
   rows = int(input("Enter the number of rows: "))
   columns = int(input("Enter the number of columns: "))
except ValueError:
   print("Please enter valid integer values for rows and columns.")
   exit()
complex_array = np.zeros((rows, columns), dtype=complex)
for i in range(rows):
   for j in range(columns):
      real_part = float(input(f"Enter the real part of element at position ({i}, {j}): "))
      imag_part = float(input(f"Enter the imaginary part of element at position ({i}, {j}): "))
      complex_array[i, j] = complex(real_part, imag_part)
print("\nComplex 2D Array:")
print(complex_array)
print(f"a. Number of rows: {rows}")
print(f"   Number of columns: {columns}")
dimensions = complex_array.ndim
print(f"b. Dimension of the array: {dimensions}")
reshaped_array = complex_array.reshape(3, 2)
print("\nReshaped 2D Array (3x2):")
print(reshaped_array)
```

OUTPUT:



```
Name:Gopika Unnikrishnan\Roll No:22MCA030\Course Name:DATA SCIENCE LAB\Course Code:20MCA241
Date:07/10/2023
Enter the number of rows: 2
Enter the number of columns: 3
Enter the real part of element at position (0, 0): 1
Enter the imaginary part of element at position (0, 0): 2
Enter the real part of element at position (0, 1): 3
Enter the imaginary part of element at position (0, 1): 4
Enter the real part of element at position (0, 2): 5
Enter the imaginary part of element at position (0, 2): 6
Enter the real part of element at position (1, 0): 7
Enter the imaginary part of element at position (1, 0): 8
Enter the real part of element at position (1, 1): 9
Enter the imaginary part of element at position (1, 1): 10
Enter the real part of element at position (1, 2): 11
Enter the imaginary part of element at position (1, 2): 12

Complex 2D Array:
[[ 1. +2.j  3. +4.j  5. +6.j]
 [ 7. +8.j  9.+10.j 11.+12.j]]
a. Number of rows: 2
   Number of columns: 3
b. Dimension of the array: 2

Reshaped 2D Array (3x2):
[[ 1. +2.j  3. +4.j]
 [ 5. +6.j  7. +8.j]
 [ 9.+10.j 11.+12.j]]
```

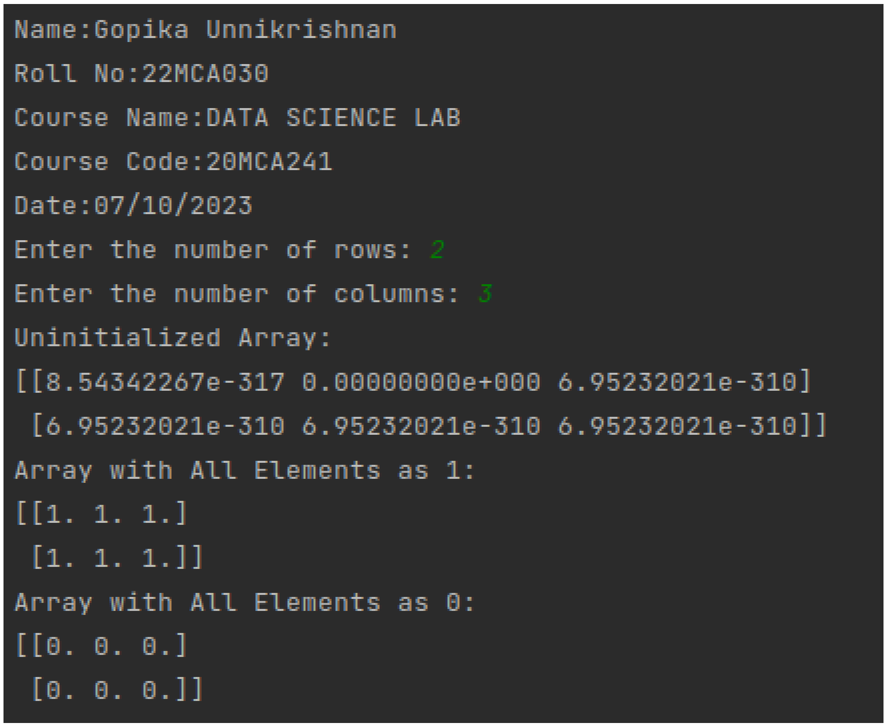
3. Familiarize with the functions to create

a) an uninitialized array
b) array with all elements as 1,
c) all elements as 0

CODE:

```
import numpy as np
try:
   print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA
SCIENCE LAB\nCourse Code:20MCA241\nDate:07/10/2023")
   rows = int(input("Enter the number of rows: "))
   columns = int(input("Enter the number of columns: "))
except ValueError:
   print("Please enter valid integer values for rows and columns.")
   exit()
uninitialized_array = np.empty((rows, columns))
print("Uninitialized Array:")
ones_array = np.ones((rows, columns))
print(uninitialized_array)
print("Array with All Elements as 1:")
print(ones_array)
zeros_array = np.zeros((rows, columns))
print("Array with All Elements as 0:")
print(zeros_array)
```

OUTPUT:



```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:07/10/2023
Enter the number of rows: 2
Enter the number of columns: 3
Uninitialized Array:
[[8.54342267e-317 0.00000000e+000 6.95232021e-310]
 [6.95232021e-310 6.95232021e-310 6.95232021e-310]]
Array with All Elements as 1:
[[1. 1. 1.]
 [1. 1. 1.]]
Array with All Elements as 0:
[[0. 0. 0.]
 [0. 0. 0.]]
```

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

a) First 4 elements

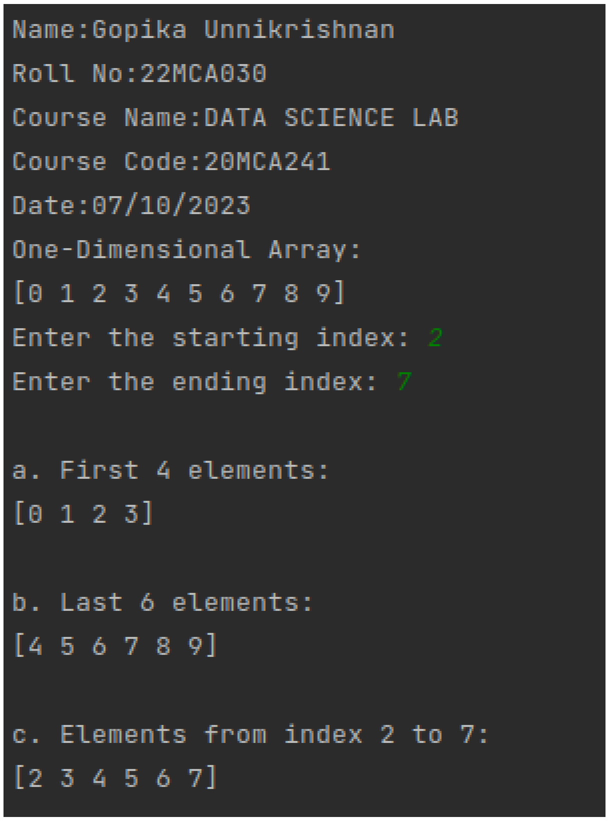
b) Last 6 elements

c) Elements from index 2 to 7

CODE:

```
print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA SCIENCE LAB\nCourse Code:20MCA241\nDate:07/10/2023")
import numpy as np
arr = np.arange(10)
print("One-Dimensional Array:")
print(arr)
try:
    start_index = int(input("Enter the starting index: "))
    end_index = int(input("Enter the ending index: "))
except ValueError:
    print("Please enter valid integer values for indices.")
    exit()
if start_index >= 0 and end_index <= 9:
    a_slice = arr[:4]  # First 4 elements
    b_slice = arr[4:]  # Last 6 elements
    c_slice = arr[start_index:end_index + 1]

    print("\na. First 4 elements:")
    print(a_slice)

    print("\nb. Last 6 elements:")
    print(b_slice)

    print("\nc. Elements from index {} to {}: ".format(start_index, end_index))
    print(c_slice)
else:
    print("Invalid indices. The indices should be between 0 and 9.")
```

OUTPUT:



```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:07/10/2023
One-Dimensional Array:
[0 1 2 3 4 5 6 7 8 9]
Enter the starting index: 2
Enter the ending index: 7

a. First 4 elements:
[0 1 2 3]

b. Last 6 elements:
[4 5 6 7 8 9]

c. Elements from index 2 to 7:
[2 3 4 5 6 7]
```

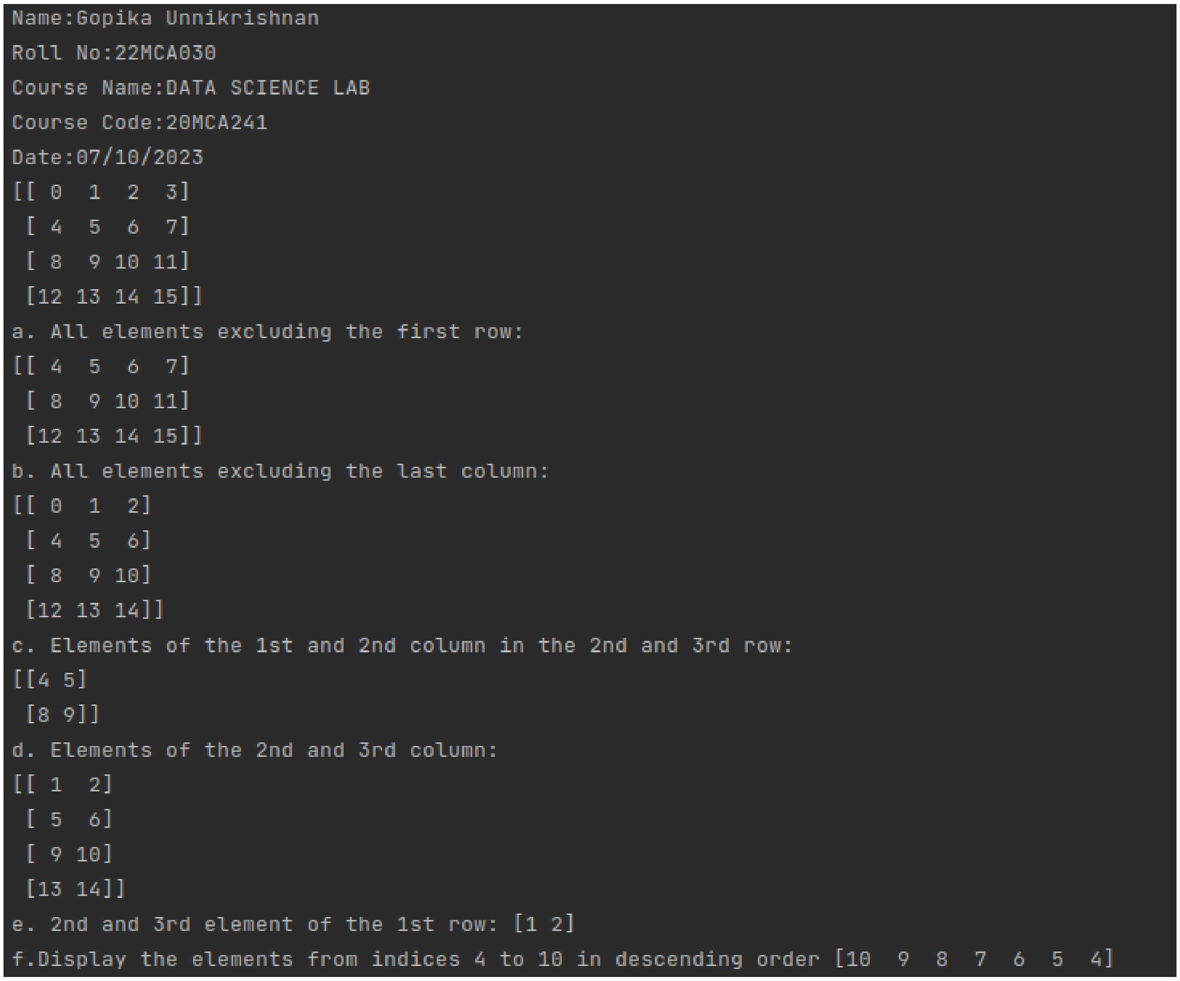
5. Create an 1D array with arange containing first 15 even numbers as elements

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**

CODE:

```
print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA SCIENCE LAB\nCourse Code:20MCA241\nDate:07/10/2023")
import numpy as np
even_numbers = np.arange(2, 32, 2)
print("First 15 even numbers as elements:")
print(even_numbers)
elements_from_2_to_8_step_2 = even_numbers[2:9:2]

print("a. Elements from index 2 to 8 with step 2:", elements_from_2_to_8_step_2)
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_elements = even_numbers[-1::-2][:3]
print("d. Last 3 alternate elements of the array:", last_3_alternate_elements)
```

OUTPUT:



```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:07/10/2023
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
a. All elements excluding the first row:
[[ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
b. All elements excluding the last column:
[[ 0  1  2]
 [ 4  5  6]
 [ 8  9 10]
 [12 13 14]]
c. Elements of the 1st and 2nd column in the 2nd and 3rd row:
[[4 5]
 [8 9]]
d. Elements of the 2nd and 3rd column:
[[ 1  2]
 [ 5  6]
 [ 9 10]
 [13 14]]
e. 2nd and 3rd element of the 1st row: [1 2]
f.Display the elements from indices 4 to 10 in descending order [10  9  8  7  6  5  4]
```

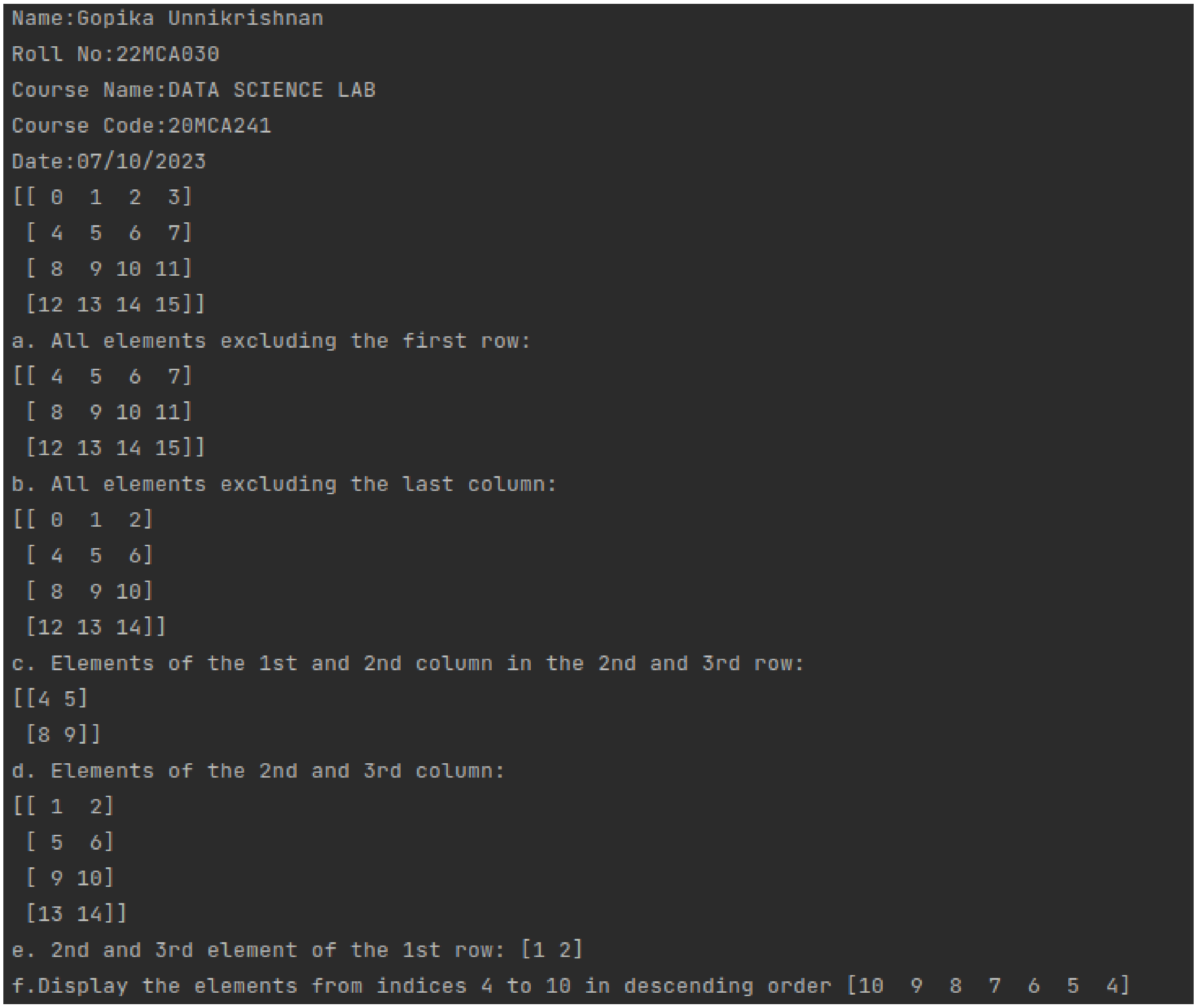
6. Create a 2 Dimensional array with 4 rows and 4 columns.

- **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)**

CODE:

```
print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA SCIENCE LAB\nCourse Code:20MCA241\nDate:07/10/2023")
import numpy as np
two_dimensional_array = np.arange(16).reshape(4,4)
print(two_dimensional_array)
elements_excluding_first_row = two_dimensional_array[1:, :]
print("a. All elements excluding the first row:")
print(elements_excluding_first_row)
elements_excluding_last_column = two_dimensional_array[:, :-1]
print("b. All elements excluding the last column:")
print(elements_excluding_last_column)
elements_1st_2nd_column_2nd_3rd_row = two_dimensional_array[1:3, 0:2]
print("c. Elements of the 1st and 2nd column in the 2nd and 3rd row:")
print(elements_1st_2nd_column_2nd_3rd_row)
elements_2nd_3rd_column = two_dimensional_array[:, 1:3]
print("d. Elements of the 2nd and 3rd column:")
print(elements_2nd_3rd_column)
elements_2nd_3rd_1st_row = two_dimensional_array[0, 1:3]
print("e. 2nd and 3rd element of the 1st row:", elements_2nd_3rd_1st_row)
x=two_dimensional_array.flatten()
ele=x[4:11]
ele_sorted_descending = np.sort(ele)[::-1]
print("f.Display the elements from indices 4 to 10 in descending order",ele_sorted_descending)
```

OUTPUT:



```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:07/10/2023
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
a. All elements excluding the first row:
[[ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
b. All elements excluding the last column:
[[ 0  1  2]
 [ 4  5  6]
 [ 8  9 10]
 [12 13 14]]
c. Elements of the 1st and 2nd column in the 2nd and 3rd row:
[[4 5]
 [8 9]]
d. Elements of the 2nd and 3rd column:
[[ 1  2]
 [ 5  6]
 [ 9 10]
 [13 14]]
e. 2nd and 3rd element of the 1st row: [1 2]
f.Display the elements from indices 4 to 10 in descending order [10  9  8  7  6  5  4]
```

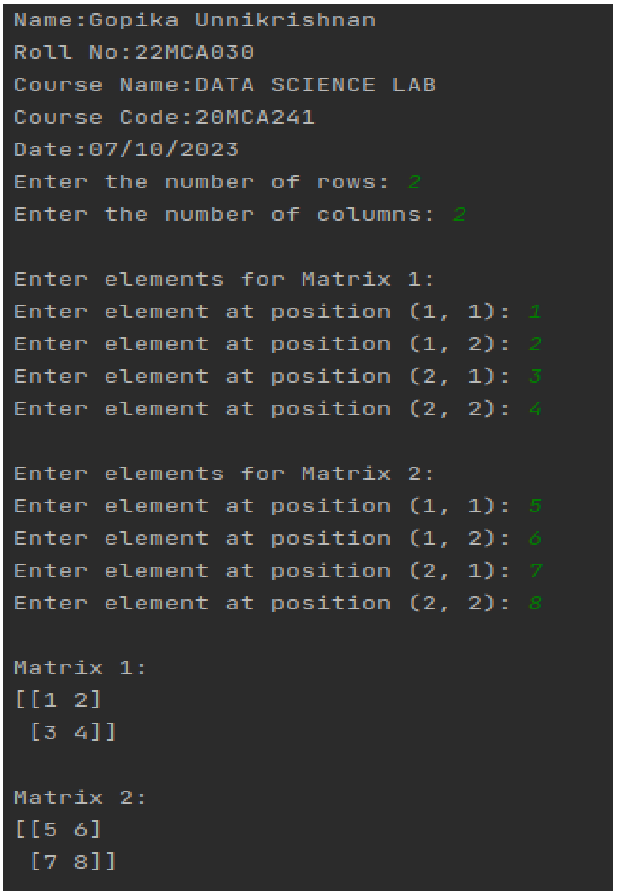
7. Create two 2D arrays using array object and

- **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**

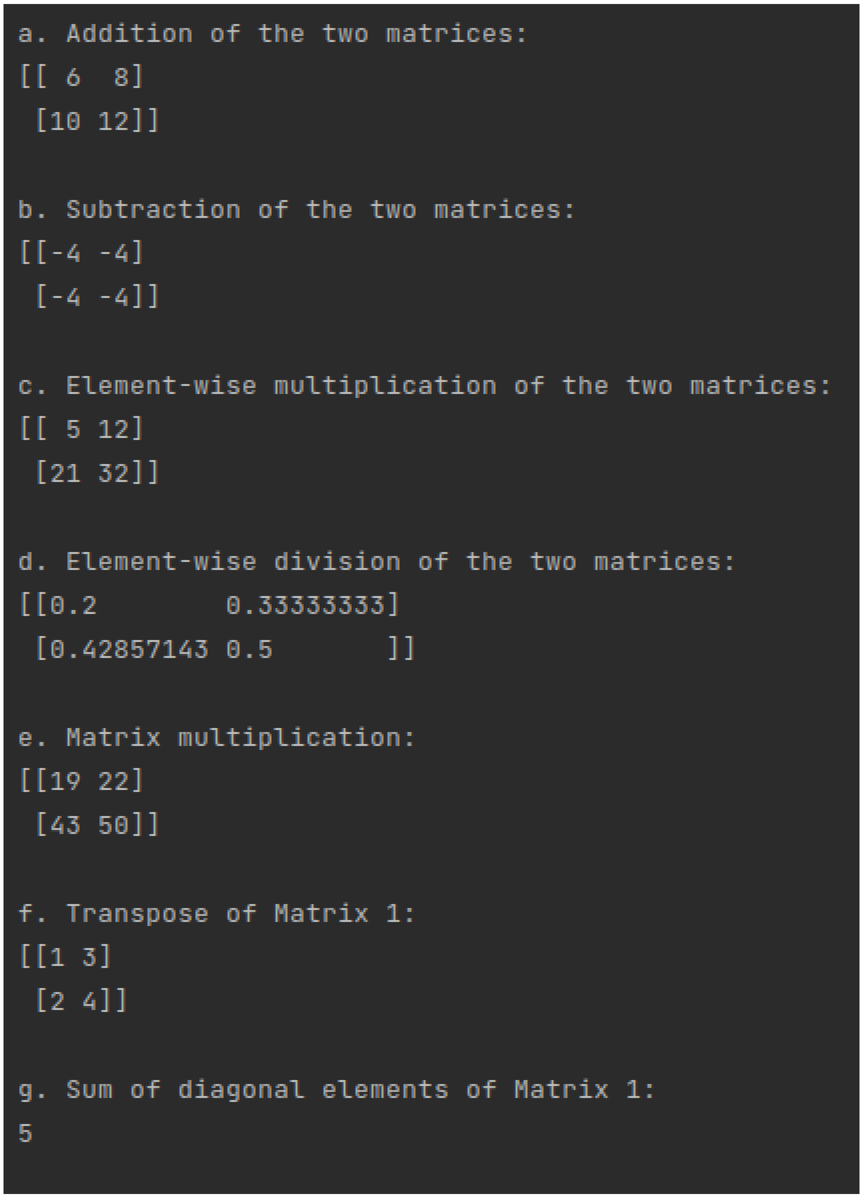
CODE:

```
print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA SCIENCE LAB\nCourse Code:20MCA241\nDate:07/10/2023")
import numpy as np
try:
    rows = int(input("Enter the number of rows: "))
    columns = int(input("Enter the number of columns: "))
except ValueError:
    print("Please enter valid integer values for rows and columns.")
    exit()
print("\nEnter elements for Matrix 1:")
matrix1 = np.zeros((rows, columns), dtype=int)
for i in range(rows):
    for j in range(columns):
        matrix1[i, j] = int(input(f"Enter element at position ({i+1}, {j+1}): "))
print("\nEnter elements for Matrix 2:")
matrix2 = np.zeros((rows, columns), dtype=int)
for i in range(rows):
    for j in range(columns):
        matrix2[i, j] = int(input(f"Enter element at position ({i+1}, {j+1}): "))
result_addition = matrix1 + matrix2
result_subtraction = matrix1 - matrix2
result_elementwise_multiplication = matrix1 * matrix2
result_elementwise_division = matrix1 / matrix2
result_matrix_multiplication = np.dot(matrix1, matrix2)
matrix1_transpose = np.transpose(matrix1)
diagonal_sum = np.trace(matrix1)
print("\nMatrix 1:")
print(matrix1)
print("\nMatrix 2:")
print(matrix2)
print("\na. Addition of the two matrices:")
print(result_addition)
print("\nb. Subtraction of the two matrices:")
print(result_subtraction)
print("\nc. Element-wise multiplication of the two matrices:")
print(result_elementwise_multiplication)
print("\nd. Element-wise division of the two matrices:")
print(result_elementwise_division)
print("\ne. Matrix multiplication:")
print(result_matrix_multiplication)
print("\nf. Transpose of Matrix 1:")
print(matrix1_transpose)
print("\ng. Sum of diagonal elements of Matrix 1:")
print(diagonal_sum)
```

OUTPUT:



```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:07/10/2023
Enter the number of rows: 2
Enter the number of columns: 2

Enter elements for Matrix 1:
Enter element at position (1, 1): 1
Enter element at position (1, 2): 2
Enter element at position (2, 1): 3
Enter element at position (2, 2): 4

Enter elements for Matrix 2:
Enter element at position (1, 1): 5
Enter element at position (1, 2): 6
Enter element at position (2, 1): 7
Enter element at position (2, 2): 8

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

Matrix 2:
[[5 6]
 [7 8]]
```



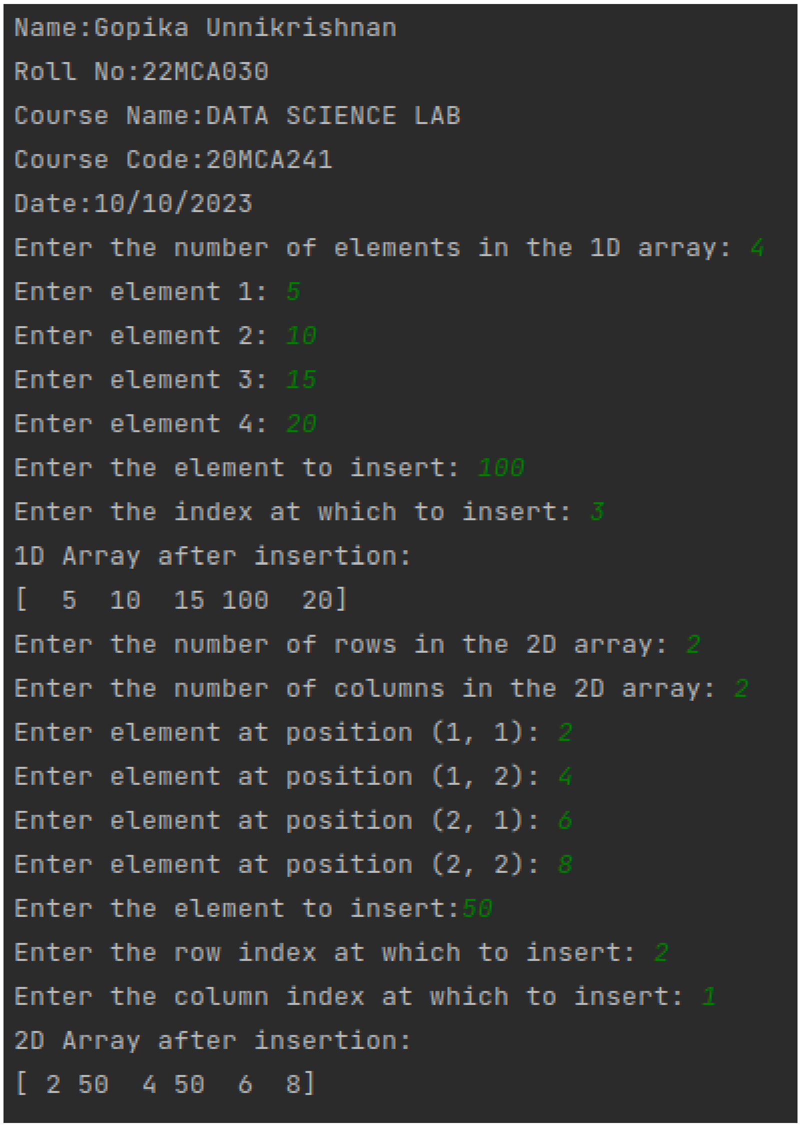
```
a. Addition of the two matrices:
[[ 6  8]
 [10 12]]

b. Subtraction of the two matrices:
[[-4 -4]
 [-4 -4]]

c. Element-wise multiplication of the two matrices:
[[ 5 12]
 [21 32]]

d. Element-wise division of the two matrices:
[[0.2        0.33333333]
 [0.42857143 0.5       ]]

e. Matrix multiplication:
[[19 22]
 [43 50]]

f. Transpose of Matrix 1:
[[1 3]
 [2 4]]

g. Sum of diagonal elements of Matrix 1:
5
```

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

CODE:

```
print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA SCIENCE
LAB\nCourse Code:20MCA241\nDate:10/10/2023")
import numpy as np
n = int(input("Enter the number of elements in the 1D array: "))
one_d_array = np.empty(n, dtype=int)
for i in range(n):
    one_d_array[i] = int(input(f"Enter element {i + 1}: "))
element_to_insert = int(input("Enter the element to insert: "))
index_to_insert = int(input("Enter the index at which to insert: "))
result_1d = np.insert(one_d_array, index_to_insert, element_to_insert)
print("1D Array after insertion:")
print(result_1d)
rows = int(input("Enter the number of rows in the 2D array: "))
columns = int(input("Enter the number of columns in the 2D array: "))
two_d_array = np.empty((rows, columns), dtype=int)
for i in range(rows):
    for j in range(columns):
        two_d_array[i, j] = int(input(f"Enter element at position ({i + 1}, {j + 1}): "))
element_to_insert = int(input("Enter the element to insert:"))
row_to_insert = int(input("Enter the row index at which to insert: "))
column_to_insert = int(input("Enter the column index at which to insert: "))
result_2d = np.insert(two_d_array, (row_to_insert, column_to_insert),
element_to_insert)
print("2D Array after insertion:")
print(result_2d)
```

OUTPUT:



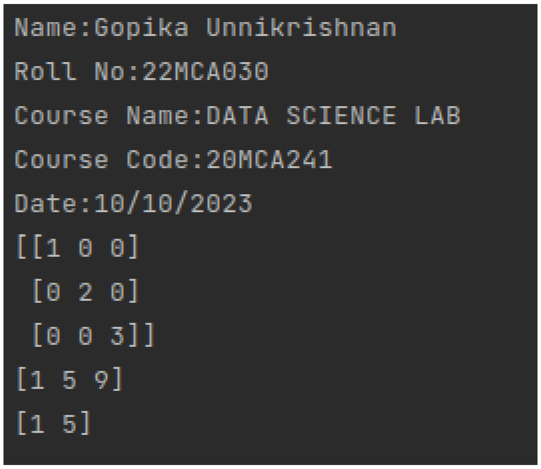
```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:10/10/2023
Enter the number of elements in the 1D array: 4
Enter element 1: 5
Enter element 2: 10
Enter element 3: 15
Enter element 4: 20
Enter the element to insert: 100
Enter the index at which to insert: 3
1D Array after insertion:
[  5  10  15 100  20]
Enter the number of rows in the 2D array: 2
Enter the number of columns in the 2D array: 2
Enter element at position (1, 1): 2
Enter element at position (1, 2): 4
Enter element at position (2, 1): 6
Enter element at position (2, 2): 8
Enter the element to insert:50
Enter the row index at which to insert: 2
Enter the column index at which to insert: 1
2D Array after insertion:
[ 2 50  4 50  6  8]
```

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

CODE:

```
print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA SCIENCE LAB\nCourse Code:20MCA241\nDate:10/10/2023")
import numpy as np
arr_1d = np.array([1, 2, 3])
diagonal_matrix = np.diag(arr_1d)
print(diagonal_matrix)
square_matrix = np.array([[1, 2, 3],
                          [4, 5, 6],
                          [7, 8, 9]])
diagonal_1d = np.diag(square_matrix)
print(diagonal_1d)
rectangular_matrix = np.array([[1, 2, 3],
                               [4, 5, 6]])
diagonal_1d = np.diag(rectangular_matrix)
print(diagonal_1d)
```

OUTPUT:



```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:10/10/2023
[[1 0 0]
 [0 2 0]
 [0 0 3]]
[1 5 9]
[1 5]
```

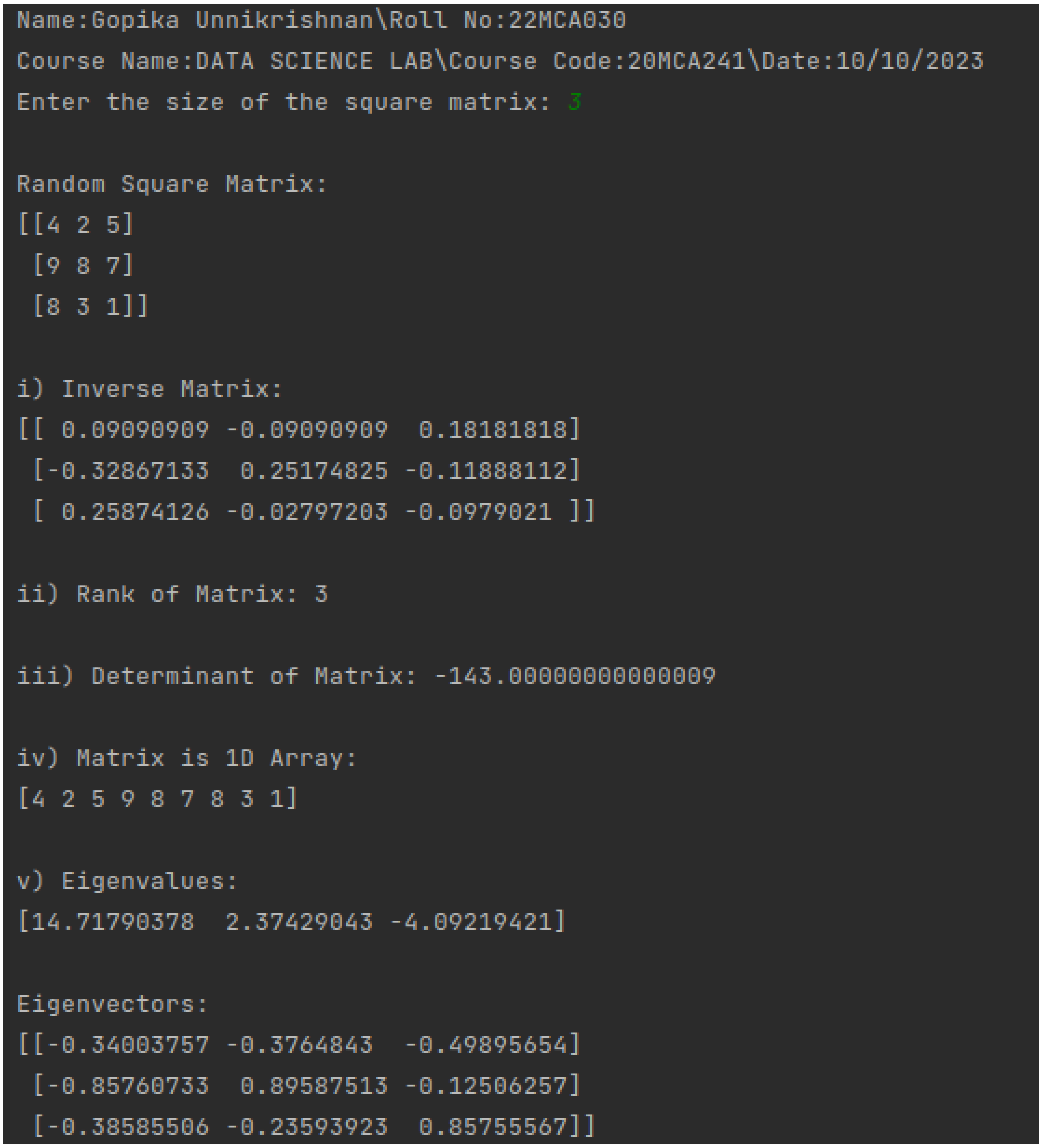
10. Create a square matrix with random integer values(use randint()) and use appropriate functions to find:

a) **Inverse**
b) **rank of matrix**
c) **Determinant**
d) **transform matrix into 1D array**
e) **eigen values and vectors**

CODE:

```
print("Name:Gopika Unnikrishnan\Roll No:22MCA030\nCourse Name:DATA SCIENCE LAB\Course Code:20MCA241\Date:10/10/2023")
import numpy as np
n = int(input("Enter the size of the square matrix: "))
random_matrix = np.random.randint(1, 10, (n, n))
try:
    inverse_matrix = np.linalg.inv(random_matrix)
except np.linalg.LinAlgError:
    inverse_matrix = None
    print("Matrix is not invertible.")
rank = np.linalg.matrix_rank(random_matrix)
determinant = np.linalg.det(random_matrix)
matrix_as_1d_array = random_matrix.flatten()
eigenvalues, eigenvectors = np.linalg.eig(random_matrix)
print("\nRandom Square Matrix:")
print(random_matrix)
if inverse_matrix is not None:
    print("\ni) Inverse Matrix:")
    print(inverse_matrix)
print("\nii) Rank of Matrix:", rank)
print("\niii) Determinant of Matrix:", determinant)
print("\niv) Matrix is 1D Array:")
print(matrix_as_1d_array)
print("\nv) Eigenvalues:")
print(eigenvalues)
print("\nEigenvectors:")
print(eigenvectors)
```

OUTPUT:



```
Name:Gopika Unnikrishnan\Roll No:22MCA030
Course Name:DATA SCIENCE LAB\Course Code:20MCA241\Date:10/10/2023
Enter the size of the square matrix: 3

Random Square Matrix:
[[4 2 5]
 [9 8 7]
 [8 3 1]]

i) Inverse Matrix:
[[ 0.09090909 -0.09090909  0.18181818]
 [-0.32867133  0.25174825 -0.11888112]
 [ 0.25874126 -0.02797203 -0.0979021 ]]

ii) Rank of Matrix: 3

iii) Determinant of Matrix: -143.00000000000009

iv) Matrix is 1D Array:
[4 2 5 9 8 7 8 3 1]

v) Eigenvalues:
[14.71790378  2.37429043 -4.09219421]

Eigenvectors:
[[-0.34003757 -0.3764843  -0.49895654]
 [-0.85760733  0.89587513 -0.12506257]
 [-0.38585506 -0.23593923  0.85755567]]
```

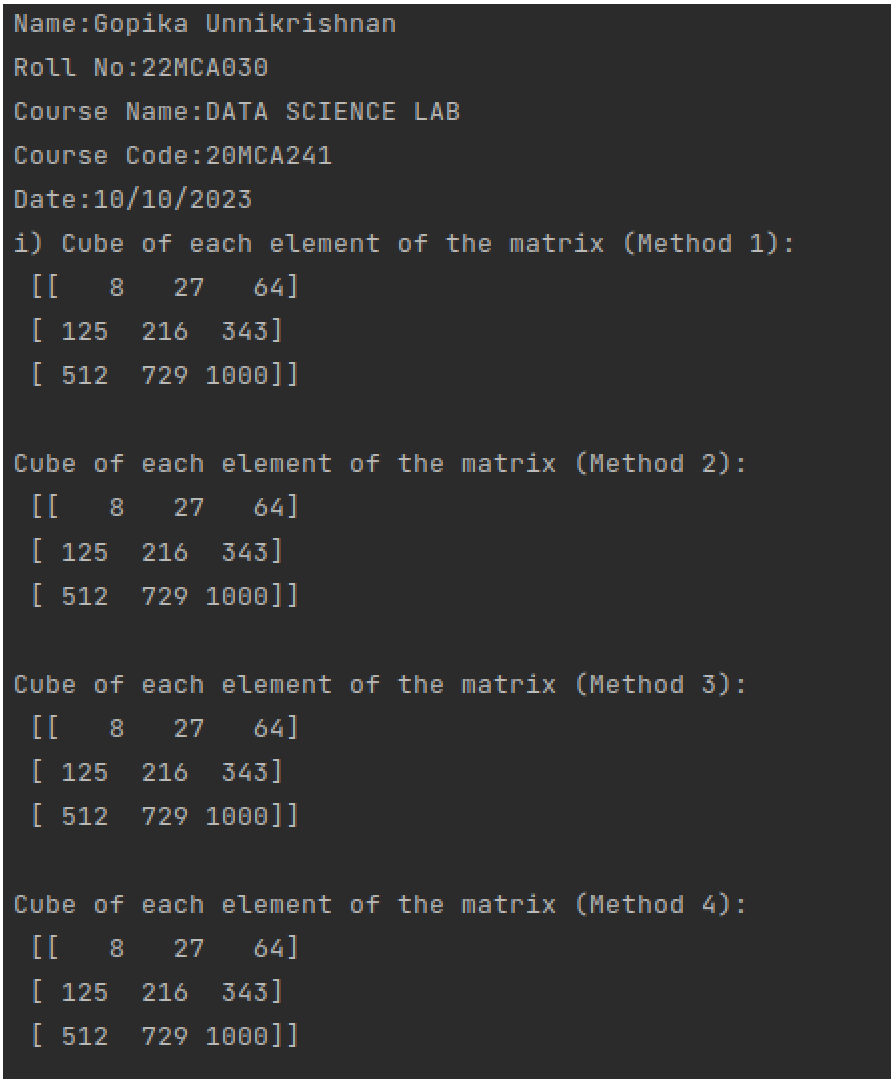
11. Create a matrix X with suitable rows and columns

a) **Display the cube of each element of the matrix using different methods(use multiply(), *, power(),**)**
b) **Display identity matrix of the given square matrix.**
c) **Display each element of the matrix to different powers.**
d) **Create a matrix Y with same dimension as X and perform the operation X^2 + 2Y.**

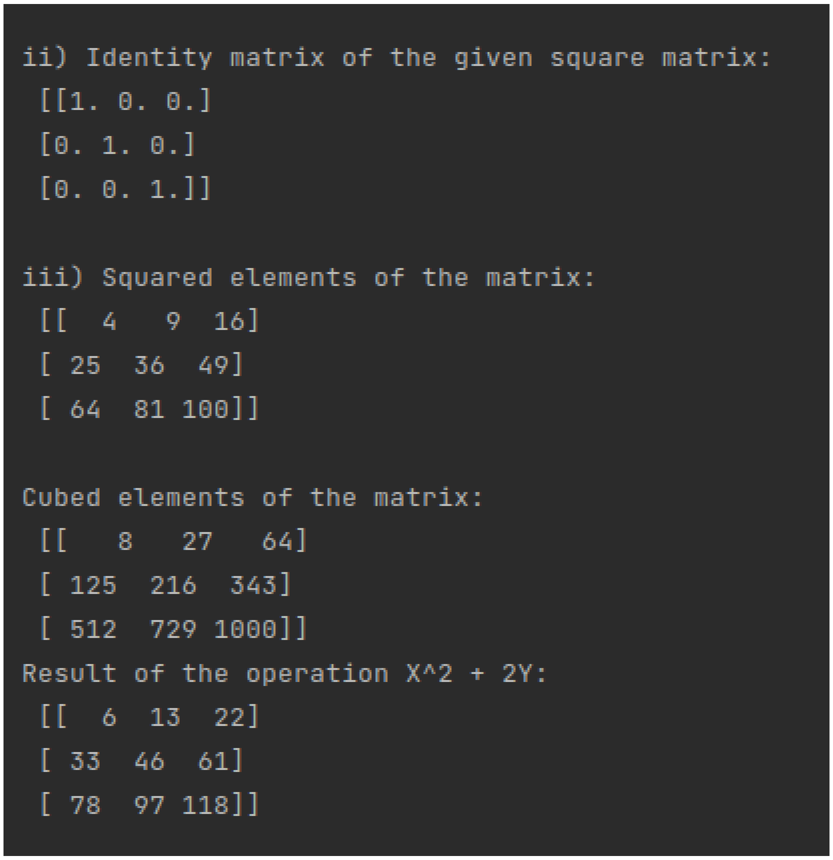
CODE:

```
print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA SCIENCE LAB\nCourse Code:20MCA241\nDate:10/10/2023")
import numpy as np
X = np.array([[2, 3, 4],
            [5, 6, 7],
            [8, 9, 10]])
cubed_elements_1 = X ** 3
cubed_elements_2 = np.power(X, 3)
cubed_elements_3 = np.multiply(X, X) * X
cubed_elements_4 = X * X * X
print("i) Cube of each element of the matrix (Method 1):\n", cubed_elements_1)
print("\nCube of each element of the matrix (Method 2):\n", cubed_elements_2)
print("\nCube of each element of the matrix (Method 3):\n", cubed_elements_3)
print("\nCube of each element of the matrix (Method 4):\n", cubed_elements_4)
identity_matrix = np.identity(X.shape[0])
print("\nii) Identity matrix of the given square matrix:\n", identity_matrix)
squared_elements = np.power(X, 2)
cubed_elements = np.power(X, 3)
print("\niii) Squared elements of the matrix:\n", squared_elements)
print("\nCubed elements of the matrix:\n", cubed_elements)
Y = np.array([[1, 2, 3],
            [4, 5, 6],
            [7, 8, 9]])
result = X**2 + 2*Y
print("Result of the operation X^2 + 2Y:\n", result)
```

OUTPUT:

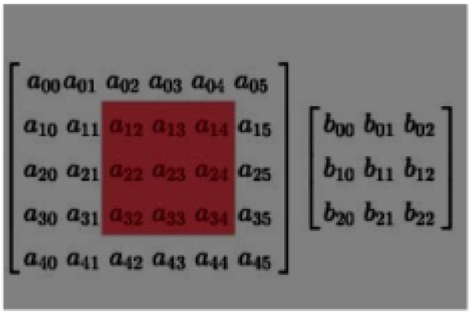


```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:10/10/2023
i) Cube of each element of the matrix (Method 1):
 [[   8   27   64]
 [ 125  216  343]
 [ 512  729 1000]]

Cube of each element of the matrix (Method 2):
 [[   8   27   64]
 [ 125  216  343]
 [ 512  729 1000]]

Cube of each element of the matrix (Method 3):
 [[   8   27   64]
 [ 125  216  343]
 [ 512  729 1000]]

Cube of each element of the matrix (Method 4):
 [[   8   27   64]
 [ 125  216  343]
 [ 512  729 1000]]
```



```
ii) Identity matrix of the given square matrix:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

iii) Squared elements of the matrix:
 [[  4   9  16]
 [ 25  36  49]
 [ 64  81 100]]

Cubed elements of the matrix:
 [[   8   27   64]
 [ 125  216  343]
 [ 512  729 1000]]
Result of the operation X^2 + 2Y:
 [[  6  13  22]
 [ 33  46  61]
 [ 78  97 118]]
```

12. 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.

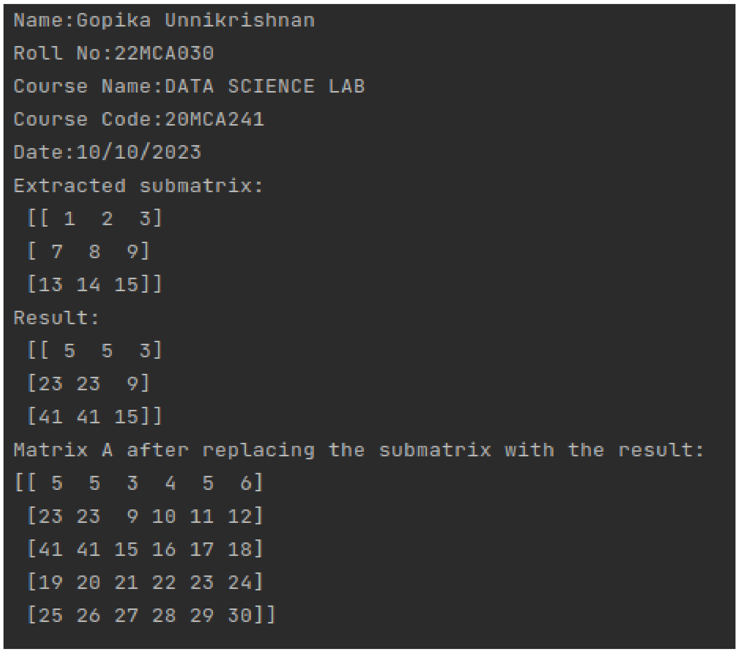


a_{00} a_{01} a_{02} a_{03} a_{04} a_{05}
a_{10} a_{11} a_{12} a_{13} a_{14} a_{15}
a_{20} a_{21} a_{22} a_{23} a_{24} a_{25}
a_{30} a_{31} a_{32} a_{33} a_{34} a_{35}
a_{40} a_{41} a_{42} a_{43} a_{44} a_{45}
b_{00} b_{01} b_{02}
b_{10} b_{11} b_{12}
b_{20} b_{21} b_{22}

CODE:

```
print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA SCIENCE LAB\nCourse Code:20MCA241\nDate:10/10/2023")
import numpy as np
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, 1, 0],
          [0, 2, 0],
          [1, 0, 1]])
submatrix = A[:3, :3]
print('Extracted submatrix:\n', submatrix)
result = np.dot(submatrix, B)
print('Result: \n', result)
A[:3, :3] = result
print("Matrix A after replacing the submatrix with the result:")
print(A)
```

OUTPUT:



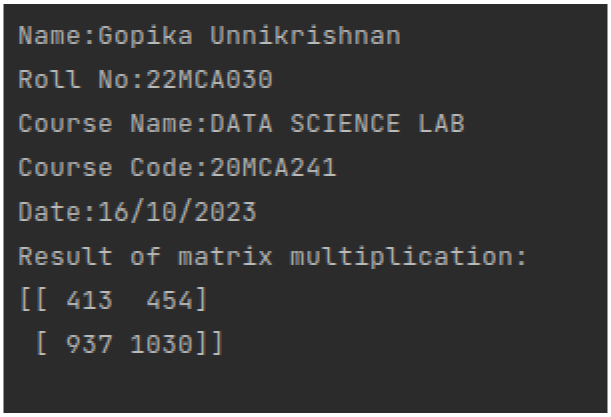
```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:10/10/2023
Extracted submatrix:
 [[ 1  2  3]
 [ 7  8  9]
 [13 14 15]]
Result:
 [[ 5  5  3]
 [23 23  9]
 [41 41 15]]
Matrix A after replacing the submatrix with the result:
[[ 5  5  3  4  5  6]
 [23 23  9 10 11 12]
 [41 41 15 16 17 18]
 [19 20 21 22 23 24]
 [25 26 27 28 29 30]]
```

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

CODE:

```
print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA SCIENCE LAB\nCourse Code:20MCA241\nDate:16/10/2023")
import numpy as np
A = np.array([[1, 2],
            [3, 4]])
B = np.array([[5, 6],
            [7, 8]])
C = np.array([[9, 10],
            [11, 12]])
result = np.dot(np.dot(A, B), C)
print("Result of matrix multiplication:")
print(result)
```

OUTPUT:



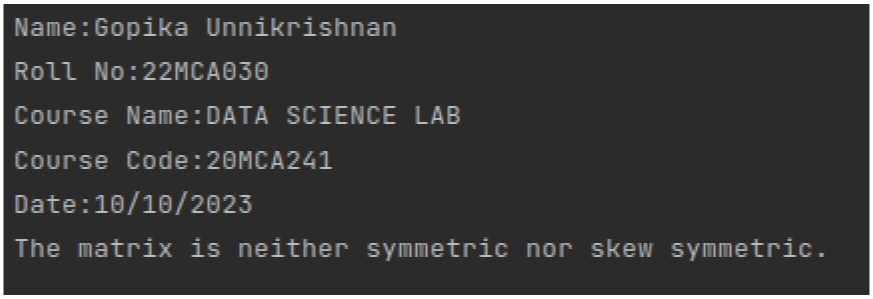
```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:16/10/2023
Result of matrix multiplication:
[[ 413  454]
 [ 937 1030]]
```

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

CODE:

```
print("Name:Gopika Unnikrishnan\nRoll No:22MCA030\nCourse Name:DATA SCIENCE
LAB\nCourse Code:20MCA241\nDate:10/10/2023")
import numpy as np
"""matrix = np.array([[1, 2, 3],
                  [2, 4, 5],
                  [3, 5, 6]])"""
"""matrix = np.array([[0, 2, -3],
                 [-2, 0, 4],
                 [3, -4, 0]])"""
matrix= np.array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])
transpose_matrix = np.transpose(matrix)
if np.array_equal(matrix, transpose_matrix):
    print("The matrix is symmetric.")
elif np.array_equal(matrix, -transpose_matrix):
    print("The matrix is skew symmetric.")
else:
    print("The matrix is neither symmetric nor skew symmetric.")
```

OUTPUT:



```
Name:Gopika Unnikrishnan
Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:10/10/2023
The matrix is neither symmetric nor skew symmetric.
```

15. 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.

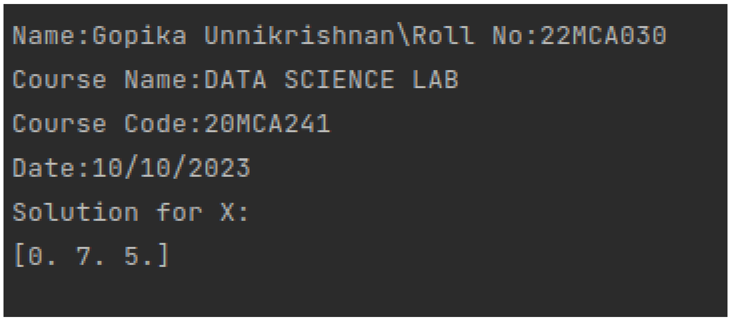
$$X=A^{-1}b.$$

$$A = \begin{bmatrix} 2 & 1 & -2 \\ 3 & 0 & 1 \\ 1 & 1 & -1 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -3 \\ 5 \\ -2 \end{bmatrix}$$

CODE:

```
print("Name:Gopika Unnikrishnan\Roll No:22MCA030\nCourse Name:DATA SCIENCE LAB\nCourse Code:20MCA241\nDate:10/10/2023")
import numpy as np
A = np.array([[2, 1,-2],
             [3,0,1],
             [1,1,-1]])
b = np.array([-3,5,2])
X = np.linalg.solve(A, b)
print("Solution for X:")
print(X)
```

OUTPUT:



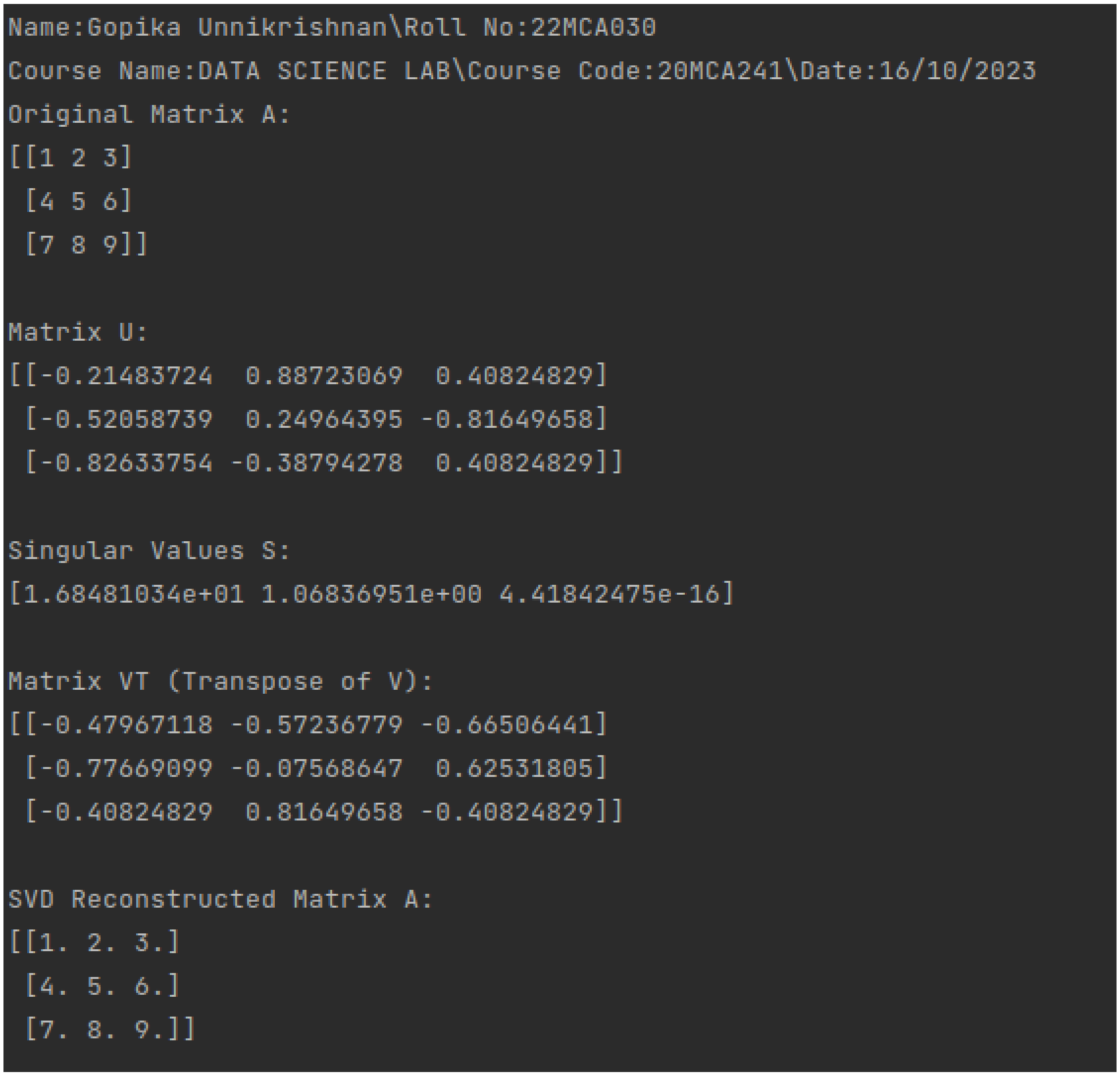
```
Name:Gopika Unnikrishnan\Roll No:22MCA030
Course Name:DATA SCIENCE LAB
Course Code:20MCA241
Date:10/10/2023
Solution for X:
[0. 7. 5.]
```

16.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().

CODE:

```
print("Name:Gopika Unnikrishnan\Roll No:22MCA030\nCourse Name:DATA SCIENCE LAB\Course Code:20MCA241\Date:16/10/2023")
import numpy as np

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("\nMatrix U:")
print(U)

print("\nSingular Values S:")
print(S)

print("\nMatrix VT (Transpose of V):")
print(VT)

print("\nSVD Reconstructed Matrix A:")
print(reconstructed_A)
```

OUTPUT:



```
Name:Gopika Unnikrishnan\Roll No:22MCA030
Course Name:DATA SCIENCE LAB\Course Code:20MCA241\Date:16/10/2023
Original Matrix A:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Matrix U:
[[-0.21483724  0.88723069  0.40824829]
 [-0.52058739  0.24964395 -0.81649658]
 [-0.82633754 -0.38794278  0.40824829]]

Singular Values S:
[1.68481034e+01 1.06836951e+00 4.41842475e-16]

Matrix VT (Transpose of V):
[[-0.47967118 -0.57236779 -0.66506441]
 [-0.77669099 -0.07568647  0.62531805]
 [-0.40824829  0.81649658 -0.40824829]]

SVD Reconstructed Matrix A:
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```