LAB CYCLE:-2

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

```
import numpy as np
depth = int(input("Enter the depth: "))
rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))
array_3d = np.empty((depth, rows, cols), dtype=float)

for d in range(depth):
   print("Enter values for 2D array :")
   for i in range(rows):
       row = input().split()
       for j in range(cols):
           array_3d[d, i, j] = float(row[j])

print("3D Array:")
print(array_3d)
```

```
Enter the depth: 2
Enter the number of rows: 2
Enter the number of columns: 3
Enter values for 2D array :
1.0 2.0 3.0
4.0 5.0 6.0
Enter values for 2D array :
7.0 8.0 9.0
1.0 11.0 12.0
3D Array:
[[[ 1.  2.  3.]
  [ 4.  5.  6.]]

 [[ 7.  8.  9.]
  [ 1. 11. 12.]]]
```

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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
num_row=int(input("Enter the number of rows: "))
num_col=int(input("Enter the number of rows: "))
complex_array=np.empty((num_row,num_col),dtype=complex)
print("Enter complex elemnts for the array")
for i in range(num_row):
   for j in range(num_col):
       real_part=float(input(f"Enter the real part for element at position
({i},{j}) : "))
       imag_part=float(input(f"Enter the imaginary part for element at position
({i},{j}) : "))
       complex_array[i,j]=complex(real_part,imag_part)
print("Complex Array")
print(complex_array)
print("number of rows: ",num_row)
print("number of columns: ",num_col)
diamensions=complex_array.ndim
print("Diamensions of the array: ",diamensions)
reshaped_array=complex_array.reshape(num_col,num_row)
print("Reshapes array:")
print(reshaped_array)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Enter the number of rows: 2
Enter the number of rows: 3
Enter complex elemnts for the array
Enter the real part for element at position (0,0) : 1
Enter the imaginary part for element at position (0,0) : 2
Enter the real part for element at position (0,1) : 3
Enter the imaginary part for element at position (0,1) : 4
Enter the real part for element at position (0,2) : 5
Enter the imaginary part for element at position (0,2) : 6
Enter the real part for element at position (1,0) : 7
Enter the imaginary part for element at position (1,0) : 8
Enter the real part for element at position (1,1) : 9
Enter the imaginary part for element at position (1,1) : 10
Enter the real part for element at position (1,2) : 11
Enter the imaginary part for element at position (1,2) : 12
Complex Array
[[ 1. +2.j  3. +4.j  5. +6.j]
 [ 7. +8.j  9.+10.j 11.+12.j]]
number of rows:  2
number of columns:  3
Diamensions of the array:  2
Reshapes array:
[[ 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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
rows=int(input("Enter the number of rows: "))
```

```
cols=int(input("Enter the number of columns: "))
uninitialized_array=np.empty((rows,cols))
ones_array=np.ones((rows,cols))
zeros_array=np.zeros((rows,cols))
print("Uninitialized array")
print(uninitialized_array)
print("\nArray with all elements are 1")
print(ones_array)
print("\nArray with all elements zero")
print(zeros_array)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Enter the number of rows: 2
Enter the number of columns: 3
Uninitialized array
[[1.92564338e-316 0.00000000e+000 6.93792485e-310]
 [6.93792487e-310 6.93792487e-310 6.93791575e-310]]

Array with all elements are 1
[[1. 1. 1.]
 [1. 1. 1.]]

Array with all elements zero
[[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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
start=int(input("Enter the starting value for the array: "))
```

```
end=int(input("Enter the ending value for the array: "))
array=np.arange(start,end)
print("First 4 elements")
print(array[:4])
print("\nLast 6 elements")
print(array[-6:])
print("\nElements from index 2 to 7")
print(array[2:8])
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Enter the starting value for the array: 1
Enter the ending value for the array: 10
First 4 elements
[1 2 3 4]

Last 6 elements
[4 5 6 7 8 9]

Elements from index 2 to 7
[3 4 5 6 7 8]
```

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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
start=int(input("Enter the starting value: "))
end=int(input("Enter the end value: "))
array=np.arange(start,end+1,step=2)
```

```
# a. Elements from index 2 to 8 with step 2
slice_a=array[2:9:2]
# b. Last 3 elements of the array using negative index
slice_b=array[-3:]
# c. Alternate elements of the array
slice_c=array[::2]
# d. Display the last 3 alternate elements
slice_d=slice_c[-3:]
print("Elements from index 2 to 8 with step 2:")
print(slice_a)
print("Last 3 elements of the array using negative index:")
print(slice_b)
print("Alternate elements of the array:")
print(slice_c)
print("Display the last 3 alternative elements:")
print(slice_d)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Enter the starting value: 2
Enter the end value: 20
Elements from index 2 to 8 with step 2:
[ 6 10 14 18]
Last 3 elements of the array using negative index:
[16 18 20]
Alternate elements of the array:
[ 2  6 10 14 18]
Display the last 3 alternative elements:
[10 14 18]
```

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)

```python
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
# Create a 2D array with 4 rows and 4 columns
array_2d=np.arange(1,17).reshape(4,4)
# a. Display all elements excluding the first row
a_result=array_2d[1:]
# b. Display all elements excluding the last column
b_result=array_2d[:,:-1]
# c. Display the elements of the 1st and 2nd column in the 2nd and 3rd row
c_result=array_2d[1:3,0:2]
# d. Display the elements of the 2nd and 3rd column
d_result=array_2d[:,1:3]
# e. Display the 2nd and 3rd element of the 1st row
e_result=array_2d[0,1:3]
# f. Display the elements from indices 4 to 10 in descending order
f_result=array_2d[3:0:-1,0:4]
print("1.Display all elements excluding the first row:")
print(a_result)
print("\n2.Display all elements excluding the last column:")
print(b_result)
print("\n3.Display the elements of the 1st and 2nd column in the 2nd and 3rd
row:")
print(c_result)
print("\n4.Display the elements of the 2nd snd 3rd column:")
print(d_result)
print("\n5.Display the 2nd snd 3rd elements of the 1st row:")
print(e_result)
print("\n6.Display the elements from indices 4 to 10 in descending order:")
print(f_result)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
1.Display all elements excluding the first row:
[[ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

2.Display all elements excluding the last column:
[[ 1  2  3]
 [ 5  6  7]
 [ 9 10 11]
 [13 14 15]]

3.Display the elements of the 1st and 2nd column in the 2nd and 3rd row:
[[ 5  6]
 [ 9 10]]

4.Display the elements of the 2nd snd 3rd column:
[[ 2  3]
 [ 6  7]
 [10 11]
 [14 15]]

5.Display the 2nd snd 3rd elements of the 1st row:
[2 3]


6.Display the elements from indices 4 to 10 in descending order:
[[13 14 15 16]
 [ 9 10 11 12]
 [ 5  6  7  8]]
```

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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
rows=int(input("Enter the number of rows for both matrices: "))
cols=int(input("Enter the number of columns for both matrices: "))
matrix1=np.empty((rows,cols))
matrix2=np.empty((rows,cols))
print("Enter the elements for the first matrix: ")
for i in range(rows):
   for j in range(cols):
       matrix1[i,j]=float(input(f"Enter elements at the position ({i},{j}): "))
print("\nEnter elements for the second row: ")
for i in range(rows):
   for j in range(cols):
       matrix2[i,j]=float(input(f"Enter elements at the position({i},{j}): "))
add_result=matrix1+matrix2
subtract_matrix=matrix1-matrix2
#Multiply the individual elements of matrix
multi_result=matrix1*matrix2
division_result=matrix1/matrix2
#matrix multiplication
matrix_multi=np.dot(matrix1,matrix2)
transpose_result=np.transpose(matrix1)
diagonal_sum=np.trace(matrix1)
print("\n1.Adding 2 matrices:")
print(add_result)
print("\n2.Subtracting 2 matrices:")
print(subtract_matrix)
print("\n3.Multiplication of individual elements of 2 matrices:")
print(matrix_multi)
print("\n4.DIvision of elememts of 2 matrices:")
print(division_result)
print("\n5.Matrix Multilication:")
print(matrix_multi)
print("\n6.Transpose of the matrix:")
print(transpose_result)
print("\n7.Sum of the diagonal elements of the matrix:")
print(diagonal_sum)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Enter the number of rows for both matrices: 2
Enter the number of columns for both matrices: 2
Enter the elements for the first matrix:
Enter elements at the position (0,0): 1
Enter elements at the position (0,1): 2
Enter elements at the position (1,0): 3
Enter elements at the position (1,1): 4

Enter elements for the second row:
Enter elements at the position(0,0): 5
Enter elements at the position(0,1): 6
Enter elements at the position(1,0): 7
Enter elements at the position(1,1): 8

1.Adding 2 matrices:
[[ 6.  8.]
 [10. 12.]]

2.Subtracting 2 matrices:
[[-4. -4.]
 [-4. -4.]]
```

```
3.Multiplication of individual elements of 2 matrices:
[[19. 22.]
 [43. 50.]]

4.DIvision of elememts of 2 matrices:
[[0.2        0.33333333]
 [0.42857143 0.5       ]]

5.Matrix Multilication:
[[19. 22.]
 [43. 50.]]

6.Transpose of the matrix:
[[1. 3.]
 [2. 4.]]

7.Sum of the diagonal elements of the matrix:
5.0

Process finished with exit code 0
```

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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
arr1d=np.array([1,2,3,4,5])
insert_arr=np.insert(arr1d,2,6)
print("\noriginal 1D array:")
print(arr1d)
print("\n1D Array after insertion:")
print(insert_arr)
arr2d=np.array([[1,2,3],[4,5,6],[7,8,9]])
insert_arr=np.insert(arr2d,1,[10,11,12],axis=0)
print("\nOriginal Array:")
print(arr2d)
print("\n2D Array after insertion:")
print(insert_arr)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA

original 1D array:
[1 2 3 4 5]

1D Array after insertion:
[1 2 6 3 4 5]

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

2D Array after insertion:
[[ 1  2  3]
 [10 11 12]
 [ 4  5  6]
 [ 7  8  9]]
```

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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
A=np.array([1,2,3,4,5])
D=np.diag(A)
print("\nOriginal 1D array:")
print(A)
print("\nDiagonal Matrix:")
print(D)
B=np.array([[1,2,3],
            [4,5,6],
            [7,8,9]])
D_square=np.diag(B)
print("\nOriginal square matrix:")
print(B)
print("\nDiagonal elements of the square matrix:")
```

```
print(D_square)
c=np.array([[1,2,3],
            [4,5,6]])
D_nonsquare=np.diag(c)
print("\nOriginal non-square matrix:")
print(c)
print("\nDiagonal matrix from non-square matrix:")
print(D_nonsquare)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA

Original 1D array:
[1 2 3 4 5]

Diagonal Matrix:
[[1 0 0 0 0]
 [0 2 0 0 0]
 [0 0 3 0 0]
 [0 0 0 4 0]
 [0 0 0 0 5]]

Original square matrix:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Diagonal elements of the square matrix:
[1 5 9]

Original non-square matrix:
[[1 2 3]
 [4 5 6]]

Diagonal matrix from non-square matrix:
[1 5]
```

10. 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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np

# Define the size of the square matrix (change as needed)
matrix_size = 4

# Create a random square matrix with integer values between 1 and 10
random_matrix = np.random.randint(1, 11, size=(matrix_size, matrix_size))

print("Random Square Matrix:")
print(random_matrix)

# Calculate the inverse matrix (if it exists)
try:
   inverse_matrix = np.linalg.inv(random_matrix)
   print("\nInverse Matrix:")
   print(inverse_matrix)
except np.linalg.LinAlgError:
   print("\nInverse does not exist for this matrix.")

# Calculate the rank of the matrix
rank = np.linalg.matrix_rank(random_matrix)
print("\nRank of the Matrix:", rank)

# Calculate the determinant of the matrix
determinant = np.linalg.det(random_matrix)
print("\nDeterminant of the Matrix:", determinant)

# Transform the matrix into a 1D array
matrix_1d = random_matrix.flatten()
print("\nMatrix as a 1D Array:")
print(matrix_1d)

# Calculate eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(random_matrix)
print("\nEigenvalues:")
print(eigenvalues)
```

```
print("\nEigenvectors:")
print(eigenvectors)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Random Square Matrix:
[[ 4  4 10  6]
 [ 1 10  1  9]
 [ 6  5  2  8]
 [ 5  9 10  8]]

Inverse Matrix:
[[-0.24024024 -0.19219219  0.17117117  0.22522523]
 [-0.28378378 -0.02702703 -0.05405405  0.2972973 ]
 [ 0.11186186 -0.01051051 -0.07657658  0.0045045 ]
 [ 0.32957958  0.16366366  0.04954955 -0.35585586]]

Rank of the Matrix: 4

Determinant of the Matrix: 1331.999999999998

Matrix as a 1D Array:
[ 4  4 10  6  1 10  1  9  6  5  2  8  5  9 10  8]

Eigenvalues:
[24.98309225  6.43428769 -1.37028446 -6.04709549]
```

```
Eigenvectors:
[[-0.47147304 -0.59117439 -0.53496254  0.49940883]
 [-0.43805262  0.71696829 -0.49704756 -0.25209017]
 [-0.43708699 -0.32107959  0.07893587 -0.68284125]
 [-0.62831365 -0.1826943   0.67862209  0.46986082]]
```

11.. 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.

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
X = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])
cubed_matrix1 = np.power(X, 3)
cubed_matrix2 = X ** 3
cubed_matrix3 = np.multiply(X, X, X)
cubed_matrix4 = X * X * X
print("Matrix X:")
print(X)
print("\nCube of each element (using np.power()):")
print(cubed_matrix1)
print("\nCube of each element (using ** operator):")
print(cubed_matrix2)
print("\nCube of each element (using np.multiply()):")
print(cubed_matrix3)
print("\nCube of each element (using * operator):")
print(cubed_matrix4)
identity_matrix = np.identity(X.shape[0])
print("\nIdentity Matrix of X:")
print(identity_matrix)
exponentials = [2, 3, 4]
powered_matrices = [np.power(X, exp) for exp in exponentials]
for i, exp in enumerate(exponentials):
   print(f"\nMatrix X to the power of {exp}:")
   print(powered_matrices[i])
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Matrix X:
[[ 1  4  9]
 [16 25 36]
 [49 64 81]]

Cube of each element (using np.power()):
[[  1   8  27]
 [ 64 125 216]
 [343 512 729]]

Cube of each element (using ** operator):
[[  1   8  27]
 [ 64 125 216]
 [343 512 729]]

Cube of each element (using np.multiply()):
[[ 1  4  9]
 [16 25 36]
 [49 64 81]]

Cube of each element (using * operator):
[[     1     64    729]
 [  4096  15625  46656]
 [117649 262144 531441]]
```

```
Identity Matrix of X:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

Matrix X to the power of 2:
[[   1   16   81]
 [ 256  625 1296]
 [2401 4096 6561]]

Matrix X to the power of 3:
[[     1     64    729]
 [  4096  15625  46656]
 [117649 262144 531441]]

Matrix X to the power of 4:
[[       1      256     6561]
 [   65536   390625  1679616]
 [ 5764801 16777216 43046721]]
```

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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
print("SJC22MCA-2027 : Georgekutty Biju")
print("S3MCA")
X = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

Y = np.array([[10, 20, 30],
              [40, 50, 60],
              [70, 80, 90]])
result = np.power(X, 2) + 2 * Y
print("Matrix X:")
print(X)
print("\nMatrix Y:")
print(Y)
print("\nResult of X^2 + 2Y:")
print(result)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Matrix X:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Matrix Y:
[[10 20 30]
 [40 50 60]
 [70 80 90]]

Result of X^2 + 2Y:
[[ 21  44  69]
 [ 96 125 156]
 [189 224 261]]
```

13. 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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
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, 3, 4],
              [5, 6, 7],
              [8, 9, 10]])
submatrix_A = A[:3, :3]
result = np.dot(submatrix_A, B)
A[:3, :3] = result
# Display the updated matrix A
print("Updated Matrix A:")
print(A)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Updated Matrix A:
[[ 36  42  48   4   5   6]
 [126 150 174  10  11  12]
 [216 258 300  16  17  18]
 [ 19  20  21  22  23  24]
 [ 25  26  27  28  29  30]]
```

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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
print("S3MCA")
A = np.array([[1, 2, 3],
              [4, 5, 6]])
B = np.array([[7, 8],
              [9, 10],
              [11, 12]])
C = np.array([[13, 14],
              [15, 16]])
result = np.dot(np.dot(A, B), C)
print("Result of Matrix Multiplication (A * B * C):")
print(result)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
S3MCA
Result of Matrix Multiplication (A * B * C):
[[1714 1836]
 [4117 4410]]
```

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

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
def is_symmetric(matrix):
    transpose = np.transpose(matrix)
    return np.array_equal(matrix, transpose)
def is_skew_symmetric(matrix):
    transpose = np.transpose(matrix)
    return np.array_equal(matrix, -transpose)
# Input the dimensions of the matrix
rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))
# Initialize an empty matrix
matrix = []
# Input the matrix elements
print("Enter the matrix elements row by row:")
for i in range(rows):
    row = []
    for j in range(cols):
        element = float(input(f"Enter element at position ({i+1}, {j+1}): "))
        row.append(element)
    matrix.append(row)

matrix = np.array(matrix)
if is_symmetric(matrix):
    print("The matrix is symmetric.")
elif is_skew_symmetric(matrix):
    print("The matrix is skew-symmetric (antisymmetric).")
else:
    print("The matrix is neither symmetric nor skew-symmetric.")
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Enter the number of rows: 3
Enter the number of columns: 3
Enter the matrix elements row by row:
Enter element at position (1, 1): 0
Enter element at position (1, 2): 1
Enter element at position (1, 3): -2
Enter element at position (2, 1): -1
Enter element at position (2, 2): 0
Enter element at position (2, 3): 3
Enter element at position (3, 1): 2
Enter element at position (3, 2): -3
Enter element at position (3, 3): 0
The matrix is skew-symmetric (antisymmetric).
```

16. 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.

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
A = np.array([[2, 3, -1],
              [1, 2, 1],
              [3, 1, -2]])
b = np.array([7, 3, 8])
try:
   X = np.linalg.solve(A, b)
   print("Solution X:")
   print(X)
except np.linalg.LinAlgError:
   print("Matrix A is singular. The system of equations may not have a unique
solution.")
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Solution X:
[ 2.   0.8 -0.6]
```

Note: Numpy provides a function called solve for solving such equations.

17. 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()

```
print("SJC22MCA-2025-FEBIN FATHIMA\nS3MCA")
import numpy as np
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
U, S, Vt = np.linalg.svd(A)
A_hat = U @ np.diag(S) @ Vt
print("Original Matrix A:")
print(A)
print("\nSingular Values:")
print(S)
print("\nReconstructed Matrix A_hat:")
print(A_hat)
```

```
SJC22MCA-2025-FEBIN FATHIMA
S3MCA
Original Matrix A:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

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

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