Mathematics For Artificial Intelligence, Lab Program Codes

By Ajay Prasad P K (MSC CS with AI , 2022-2024)

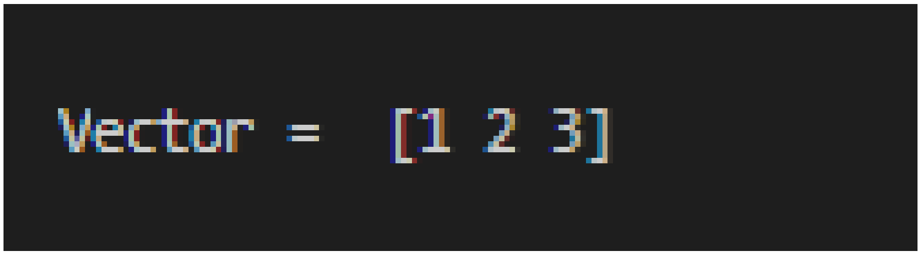
SET-1

1)Define a vector

CODE:

```
import numpy as np
vectorlist = [1,2,3]
vector = np.array(vectorlist)
print ("\nVector = " ,vector)
```

OUTPUT :



```
Vector =  [1 2 3]
```

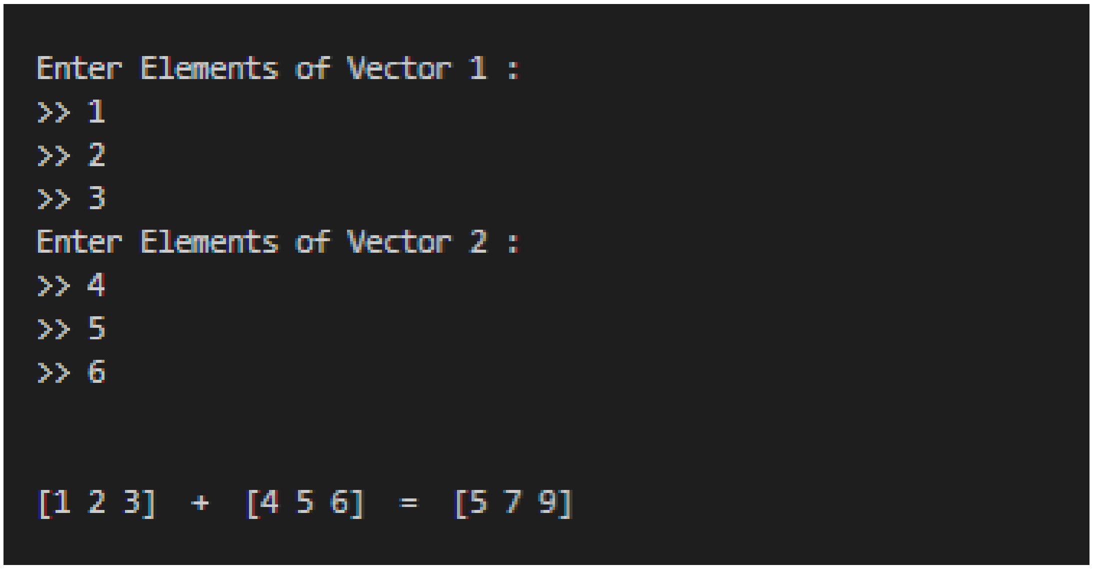
2)Add two vectors using NumPy Arrays

CODE:

```
import numpy as np
vectorlist1 = []
vectorlist2 = []
print("\nEnter Elements of Vector 1 : ")
```

```
for i in range(3):

    j = int(input(">> "))

    vectorlist1.append(j)

print("Enter Elements of Vector 2 : ")

for i in range(3):

    j = int(input(">> "))

    vectorlist2.append(j)

vector1 = np.array(vectorlist1)

vector2 = np.array(vectorlist2)

added = vector1 + vector2

print("\n")

print(vector1 , " + " , vector2 , " = " , added)
```

OUTPUT :



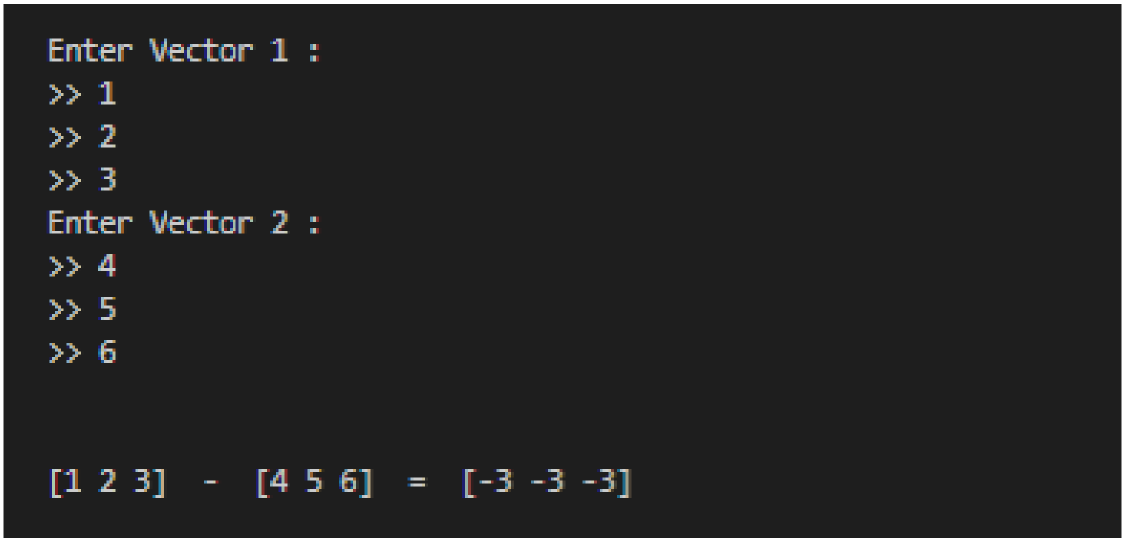
```
Enter Elements of Vector 1 :
>> 1
>> 2
>> 3
Enter Elements of Vector 2 :
>> 4
>> 5
>> 6


[1 2 3]  +  [4 5 6]  =  [5 7 9]
```

3)Subtract two vectors using NumPy Arrays

CODE:

```
import numpy as np

vectorlist1 = []

vectorlist2 = []

print("\nEnter Vector 1 : ")

for j in range(3):

    j = int(input(">> "))

    vectorlist1.append(j)

print("Enter Vector 2 : ")

for j in range(3):

    j = int(input(">> "))

    vectorlist2.append(j)

vector1 = np.array(vectorlist1)

vector2 = np.array(vectorlist2)

subbed = vector1 - vector2

print("\n")

print(vector1, " - ", vector2, " = ", subbed)
```

OUTPUT :



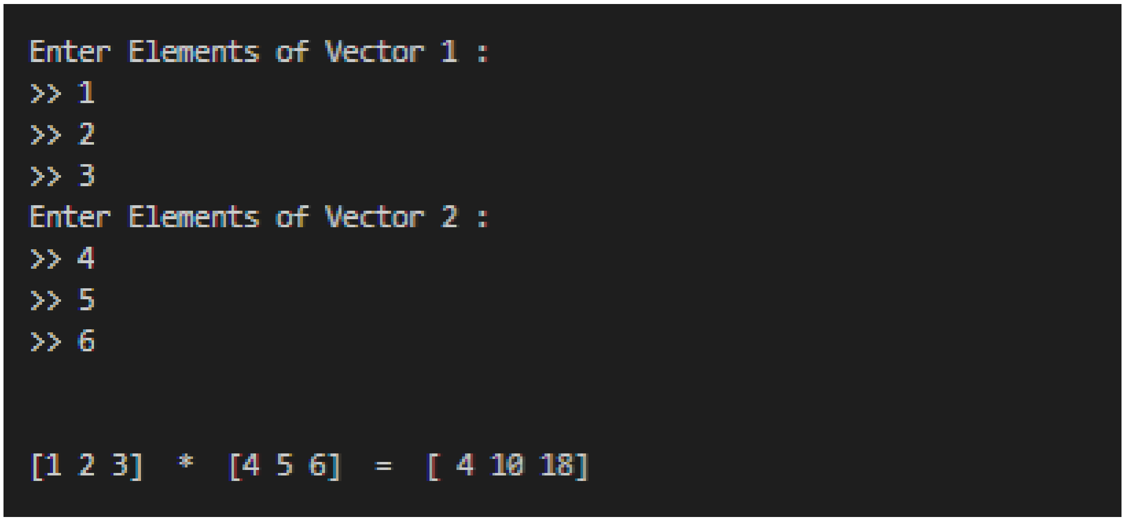
```
Enter Vector 1 :
>> 1
>> 2
>> 3
Enter Vector 2 :
>> 4
>> 5
>> 6


[1 2 3]  -  [4 5 6]  =  [-3 -3 -3]
```

4)Multiply two vectors using NumPy Arrays

CODE:

```
import numpy as np

vectorlist1 = []

vectorlist2 = []

print("\nEnter Elements of Vector 1 : ")

for i in range(3):

    j = int(input(">> "))

    vectorlist1.append(j)

print("Enter Elements of Vector 2 : ")

for i in range(3):

    j = int(input(">> "))
    vectorlist2.append(j)

vector1 = np.array(vectorlist1)

vector2 = np.array(vectorlist2)

multi = vector1 * vector2

print("\n")

print(vector1 , " * " , vector2 , " = " , multi)
```

OUTPUT :



```
Enter Elements of Vector 1 :
>> 1
>> 2
>> 3
Enter Elements of Vector 2 :
>> 4
>> 5
>> 6


[1 2 3]  *  [4 5 6]  =  [ 4 10 18]
```

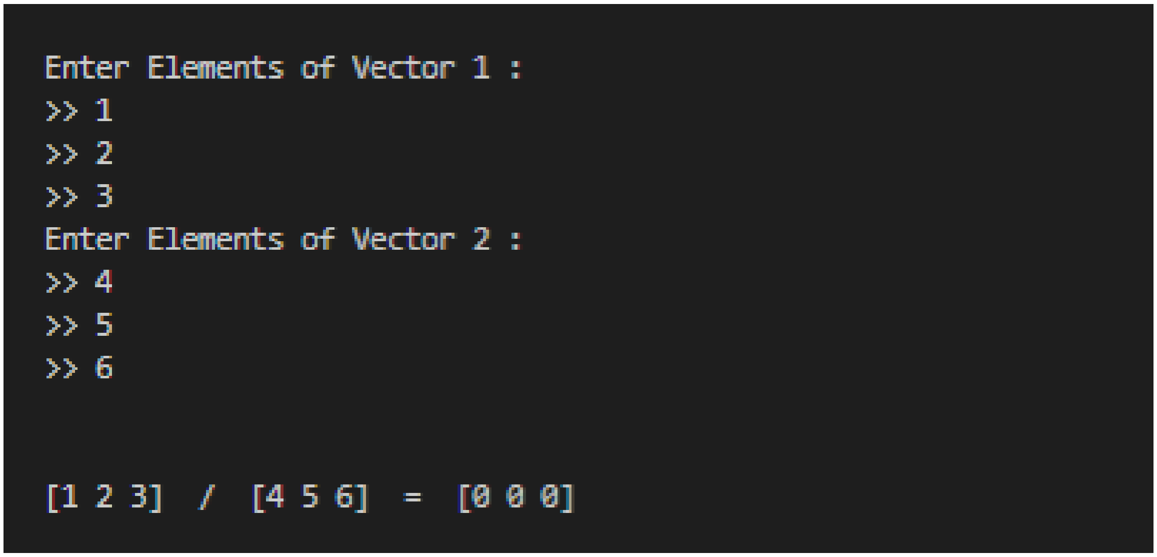
5)Divide two vectors using NumPy Arrays

CODE:

```
import numpy as np

vectorlist1 = []

vectorlist2 = []

print("\nEnter Elements of Vector 1 : ")
```

```
for i in range(3):
    j = int(input(">> "))
    vectorlist1.append(j)
print("Enter Elements of Vector 2 : ")
for i in range(3):
    j = int(input(">> "))
    vectorlist2.append(j)
vector1 = np.array(vectorlist1)
vector2 = np.array(vectorlist2)
div = vector1 // vector2
print("\n")
print(vector1 , " / " , vector2 , " = " , div)
```

OUTPUT :



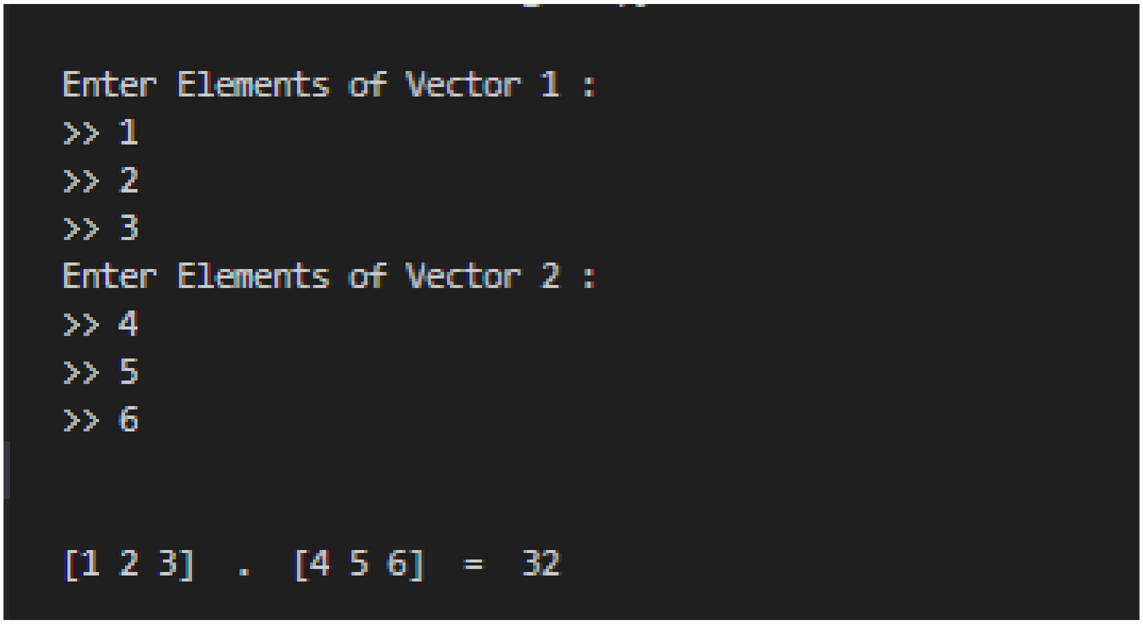
```
Enter Elements of Vector 1 :
>> 1
>> 2
>> 3
Enter Elements of Vector 2 :
>> 4
>> 5
>> 6


[1 2 3]  /  [4 5 6]  =  [0 0 0]
```

6)Find dot product of two vectors

CODE:

```
import numpy as np

vectorlist1 = []

vectorlist2 = []

print("\nEnter Elements of Vector 1 : ")

for i in range(3):

    j = int(input(">> "))

    vectorlist1.append(j)

print("Enter Elements of Vector 2 : ")

for i in range(3):

    j = int(input(">> "))

    vectorlist2.append(j)

vector1 = np.array(vectorlist1)

vector2 = np.array(vectorlist2)

dot = np.dot(vector1,vector2)

print("\n")

print(vector1 , " . " , vector2 , " = " , dot)
```

OUTPUT :



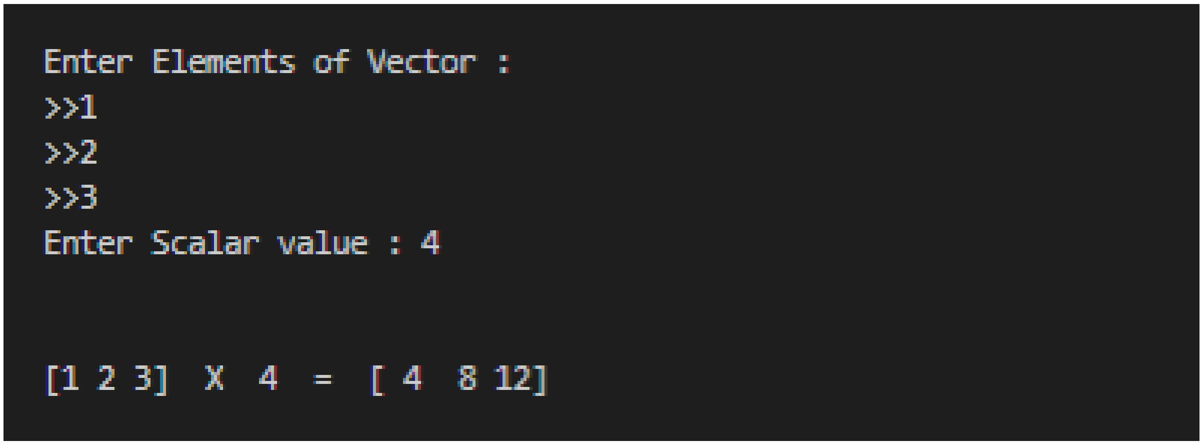
```
Enter Elements of Vector 1 :
>> 1
>> 2
>> 3
Enter Elements of Vector 2 :
>> 4
>> 5
>> 6


[1 2 3]  .  [4 5 6]  =  32
```

7)Perform vector Scalar Multiplication

CODE:

```
import numpy as np

vectorlist = []

print("\nEnter Elements of Vector : ")

for i in range(3):

    j = int(input(">>"))

    vectorlist.append(j)

vector = np.array(vectorlist)

k = int(input("Enter Scalar value : "))

sproduct = k * vector

print("\n")

print(vector , " X ", k , " = ", sproduct)
```

OUTPUT :



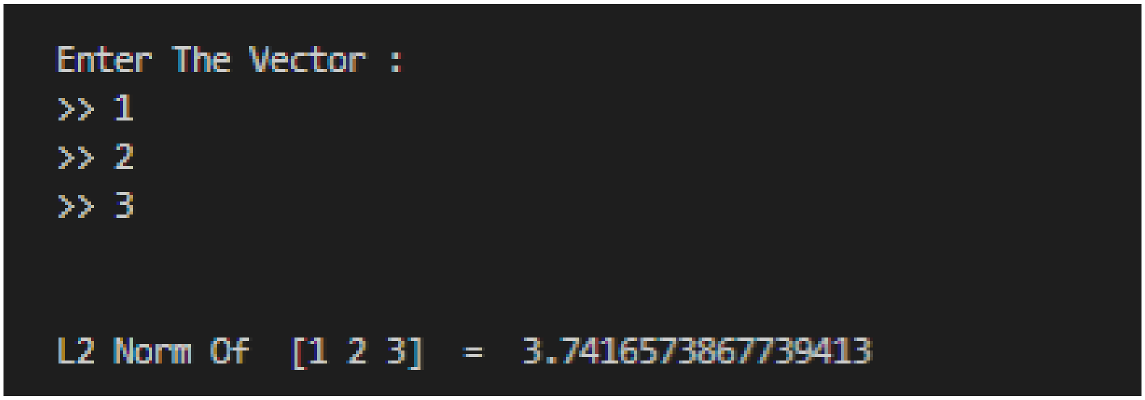
```
Enter Elements of Vector :
>>1
>>2
>>3
Enter Scalar value : 4


[1 2 3]  X  4  =  [ 4  8 12]
```

8)Calculate L2 Norm of a vector

CODE:

```
import numpy as np

vectorlist = []

print("\nEnter The Vector : ")

for i in range(3):

    j = int (input(">> "))

    vectorlist.append(j)

vector = np.array(vectorlist)

l2norm = np.linalg.norm(vector)

print("\n")

print("L2 Norm Of ", vector , " = ", l2norm)
```

OUTPUT :



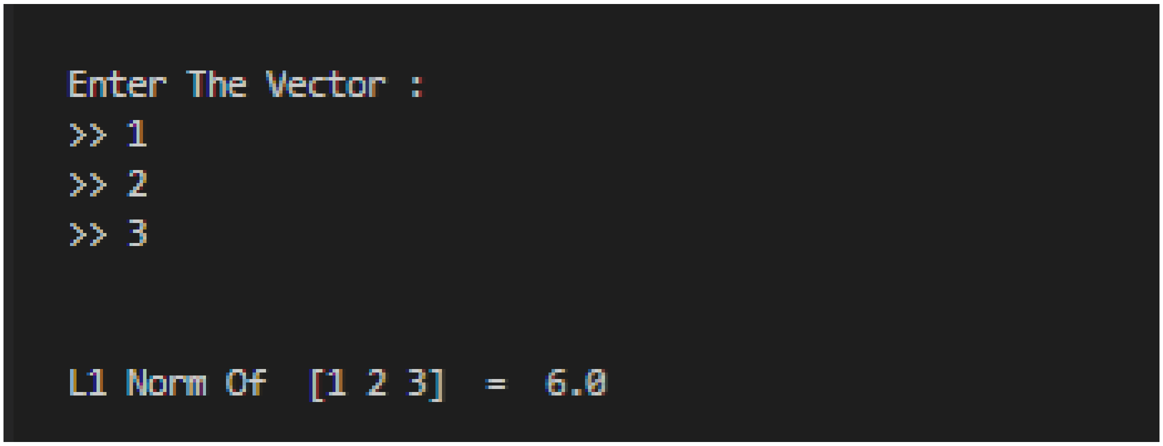
```
Enter The Vector :
>> 1
>> 2
>> 3


L2 Norm Of  [1 2 3]  =  3.7416573867739413
```

9)Calculate L1 Norm of a vector

CODE:

```
import numpy as np

vectorlist = []

print("\nEnter The Vector : ")

for i in range(3):

    j = int(input(">> "))

    vectorlist.append(j)

vector = np.array(vectorlist)

l1norm = np.linalg.norm(vector, 1)

print("\n")

print("L1 Norm Of ", vector , " = ", l1norm)
```

OUTPUT :



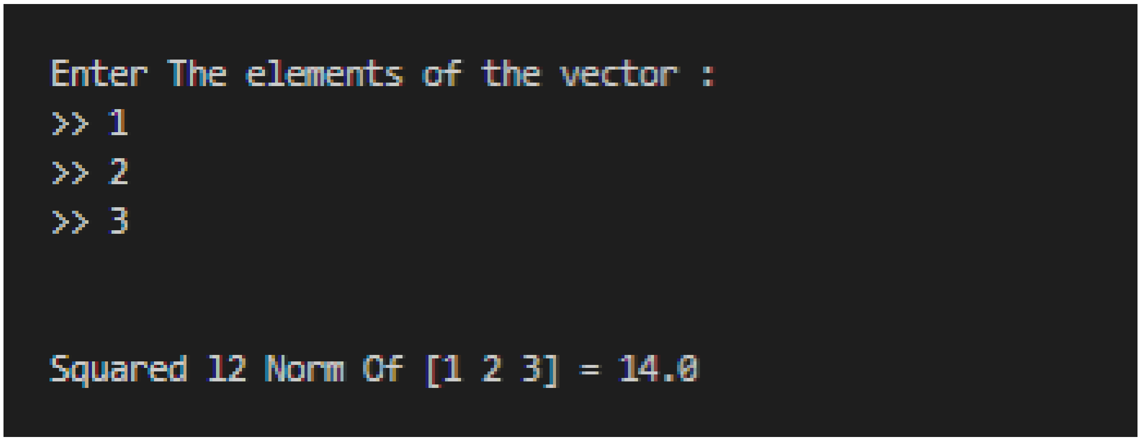
```
Enter The Vector :
>> 1
>> 2
>> 3


L1 Norm Of  [1 2 3]  =  6.0
```

10)Calculate Squared L2 Norm of a vector

<u>CODE:</u>

```
import numpy as np

vector_list = []


print ("\nEnter The elements of the vector : " )

for i in range (3):

    j = int(input(">> "))

    vector_list.append(j)


vector = np.array(vector_list)

l2norm = np.linalg.norm(vector)

squaredL2Norm = l2norm * l2norm

print("\n")

print(f"Squared l2 Norm Of {vector} = {squaredL2Norm}"  )
```

OUTPUT :



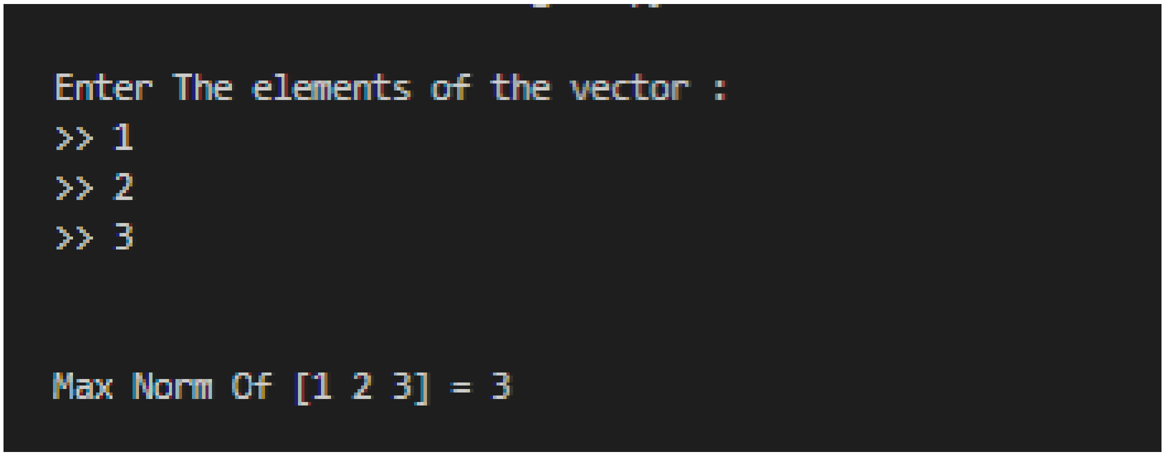
```
Enter The elements of the vector :
>> 1
>> 2
>> 3


Squared l2 Norm Of [1 2 3] = 14.0
```

11)Calculate Max Norm of a vector

CODE:

```
import numpy as np

vector_list = []

print ("\nEnter The elements of the vector : " )

for i in range (3):

   j = int(input(">> "))

   vector_list.append(j)

vector = np.array(vector_list)

maxnorm  = vector.flat[abs(vector).argmax()]

print("\n")

print(f"Max Norm Of {vector} = {abs(maxnorm)}"  )
```

OUTPUT :



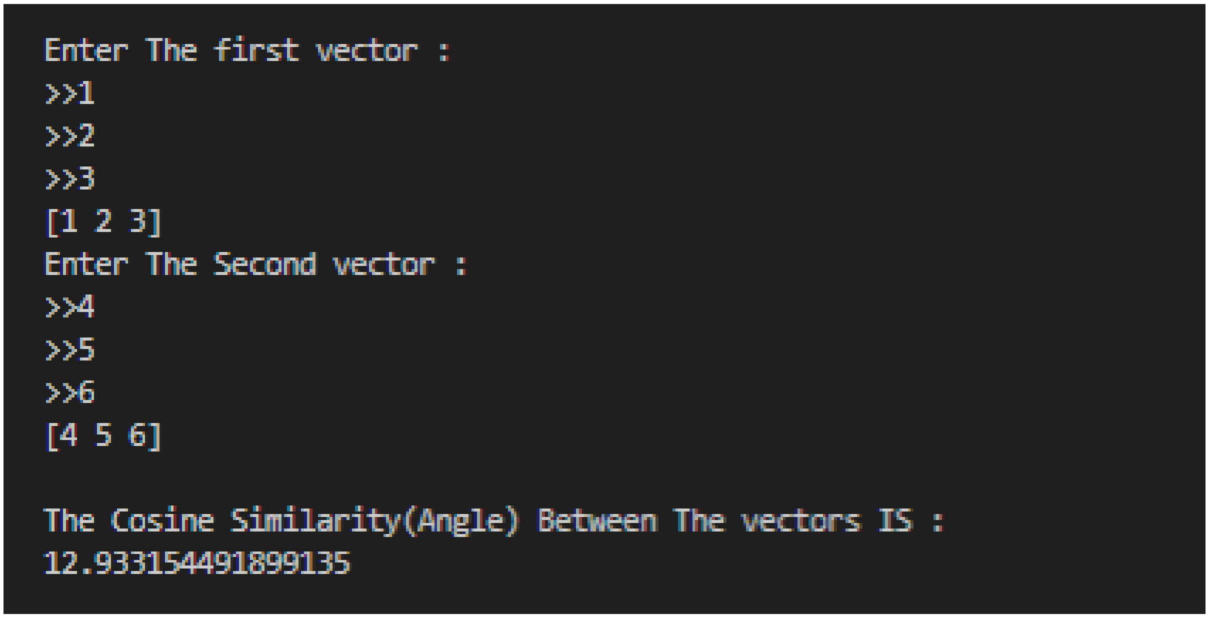
```
Enter The elements of the vector :
>> 1
>> 2
>> 3


Max Norm Of [1 2 3] = 3
```

12)Cosine Similarity of Two Vectors

CODE:

```
import numpy as np

import math

def det(elements):

    sum = 0

    for element in elements:

        elesq = element * element

        sum = sum+elesq

    dete = math.sqrt(sum)

    return dete

vector_list1 = []

vector_list2 = []

print ("\nEnter The first vector :")

for i in range(3):

    j = int(input(">>"))
    vector_list1.append(j)
vector1 = np.array(vector_list1)
print(vector1)
print ("Enter The Second vector :")
for i in range(3):
    j = int(input(">>"))
    vector_list2.append(j)
vector2 = np.array(vector_list2)
print(vector2)
dot = np.dot(vector1,vector2)
detvector1 = det(vector1)
detvector2 = det(vector2)
cos = (dot/(detvector1*detvector2))
angle = math.degrees(math.acos(cos))
print("\nThe Cosine Similarity(Angle) Between The vectors IS :")
print(angle)
```

OUTPUT :



```
Enter The first vector :
>>1
>>2
>>3
[1 2 3]
Enter The Second vector :
>>4
>>5
>>6
[4 5 6]

The Cosine Similarity(Angle) Between The vectors IS :
12.933154491899135
```

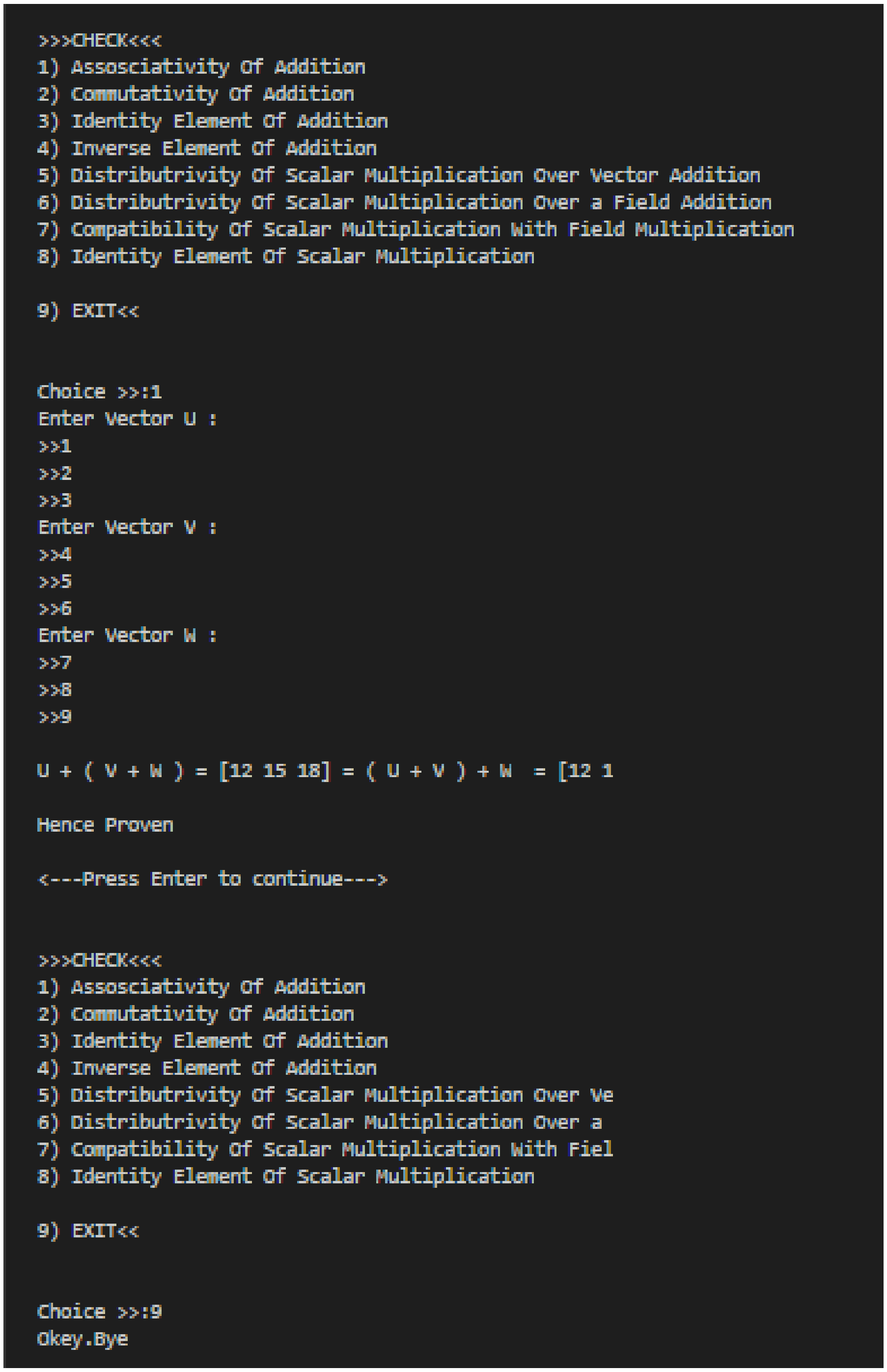
13)Check The Axioms

CODE:

```
import numpy as np

def acceptvector():

    vector_list = []

    for i in range(3):

        j = int(input(">>"))

        vector_list.append(j)

    vector = np.array(vector_list)

    return vector


while True :

    print("\n\n>>>CHECK<<<\n1) Assosciativity Of Addition\n2) Commutativity Of Addition\n3) Identity Element Of Addition\n4) Inverse Element Of Addition\n5) Distributrivity Of Scalar Multiplication Over Vector Addition \n6) Distributrivity Of Scalar Multiplication Over a Field
```

```
Addition \n7) Compatibility Of Scalar Multiplication With Field Multiplication\n8) Identity Element
Of Scalar Multiplication \n\n9) EXIT<<\n\n")
    choice = int(input("Choice >>:"))
    if choice == 1 :
        print("Enter Vector U :")
        vector1 = acceptvector()
        print("Enter Vector V :")
        vector2 = acceptvector()
        print("Enter Vector W :")
        vector3 = acceptvector()
        lhs = vector1 + (vector2 + vector3)
        rhs = (vector1 + vector2) + vector3
        print(f'\nU + ( V + W ) = {lhs} = ( U + V ) + W  = {rhs} ,\n\nHence Proven\n')
        b = input("<---Press Enter to continue--->")
    elif choice == 2 :
        print("Enter Vector U :")
        vector1 = acceptvector()
        print("Enter Vector V :")
        vector2 = acceptvector()
        lhs = vector1 + vector2
        rhs = vector2 + vector1
        print(f"\nU + V = {lhs} = V + U = {rhs} ,\n\nHence Proven\n")
        b = input("<---Press Enter to continue--->")
    elif choice == 3 :
        print("Enter Vector V :")
        vector1 =acceptvector()
        ans = vector1 + 0
        print(f"\nV + I = {ans} , \n\nHence Proven\n")
        input("<---Press Enter to continue--->")
    elif choice == 4 :
        print("Enter Vector V :")
        vector1 =acceptvector()
        ans = vector1 + (-vector1)
        print(f"\nV + (-V) = {ans} ,\n\nHence Proven\n")
        input("<---Press Enter to continue--->")
    elif choice == 5 :
        print("Enter Vector U :")
        vector1 = acceptvector()
        print("Enter Vector V :")
        vector2 = acceptvector()
        const = int(input("Enter Value For Constant (C)\n>>"))
        lhs = const * (vector1 + vector2)
        rhs = (const * vector1) + (const * vector2)
        print(f"\nC * ( U + V ) = {lhs} = C * U + C * V = {rhs} ,\n\nHence Proven\n")
        input("<---Press Enter to continue--->")
    elif choice == 6 :
        print("Enter Vector V :")
        vector1 = acceptvector()
        const1 = int(input("Enter Value For Constant (C)\n>>"))
        const2 = int(input("Enter Value For Constant (B)\n>>"))
        lhs = (const1 + const2) * vector1
        rhs = (const1 * vector1) + (const2 * vector1)
        print(f"\n( C + B ) * V = {lhs} = ( C * V ) + ( B * V ) = {rhs} ,\n\nHence Proven\n")
        input("<---Press Enter to continue--->")
    elif choice == 7 :
        print("Enter Vector V :")
        vector1 = acceptvector()
        const1 = int(input("Enter Value For Constant (C)\n>>"))
        const2 = int(input("Enter Value For Constant (B)\n>>"))
        lhs = const1 * (const2 * vector1)
        rhs = (const1 * const2) * vector1
        print(f"\nC * ( B * V  ) = {lhs} = ( C * B ) * V = {rhs} ,\n\nHence Proven\n")
        input("<---Press Enter to continue--->")
    elif choice == 8 :
        print("Enter Vector V :")
        vector1 = acceptvector()
        ans = vector1 * 1
        print(f"\nV * I = {ans} = V ,\n\nHence Proven\n")
        input("<---Press Enter to continue--->")
    elif choice == 9 :
        print("Okey.Bye")
        exit()
    else :
        print("\nPlease Enter Any Valid Choice !!!!\n")

        input("<---Press Enter to continue--->")
```

OUTPUT :



```
>>>CHECK<<<
1) Assosciativity Of Addition
2) Commutativity Of Addition
3) Identity Element Of Addition
4) Inverse Element Of Addition
5) Distributrivity Of Scalar Multiplication Over Vector Addition
6) Distributrivity Of Scalar Multiplication Over a Field Addition
7) Compatibility Of Scalar Multiplication With Field Multiplication
8) Identity Element Of Scalar Multiplication

9) EXIT<<


Choice >>:1
Enter Vector U :
>>1
>>2
>>3
Enter Vector V :
>>4
>>5
>>6
Enter Vector W :
>>7
>>8
>>9

U + ( V + W ) = [12 15 18] = ( U + V ) + W  = [12 1

Hence Proven

<---Press Enter to continue--->


>>>CHECK<<<
1) Assosciativity Of Addition
2) Commutativity Of Addition
3) Identity Element Of Addition
4) Inverse Element Of Addition
5) Distributrivity Of Scalar Multiplication Over Ve
6) Distributrivity Of Scalar Multiplication Over a
7) Compatibility Of Scalar Multiplication With Fiel
8) Identity Element Of Scalar Multiplication

9) EXIT<<


Choice >>:9
Okey.Bye
```

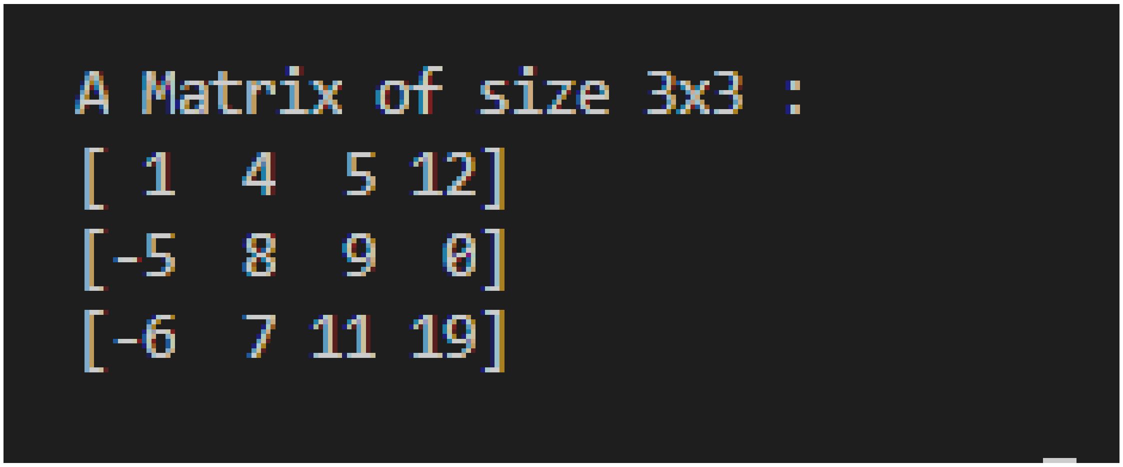
SET-2

1)Define a matrix

CODE:

```
import numpy as np

A = np.array([[1, 4, 5, 12],

   [-5, 8, 9, 0],

   [-6, 7, 11, 19]])

print("\nA Matrix of size 3x3 :")

for row in A:

   print(row)
```

OUTPUT :



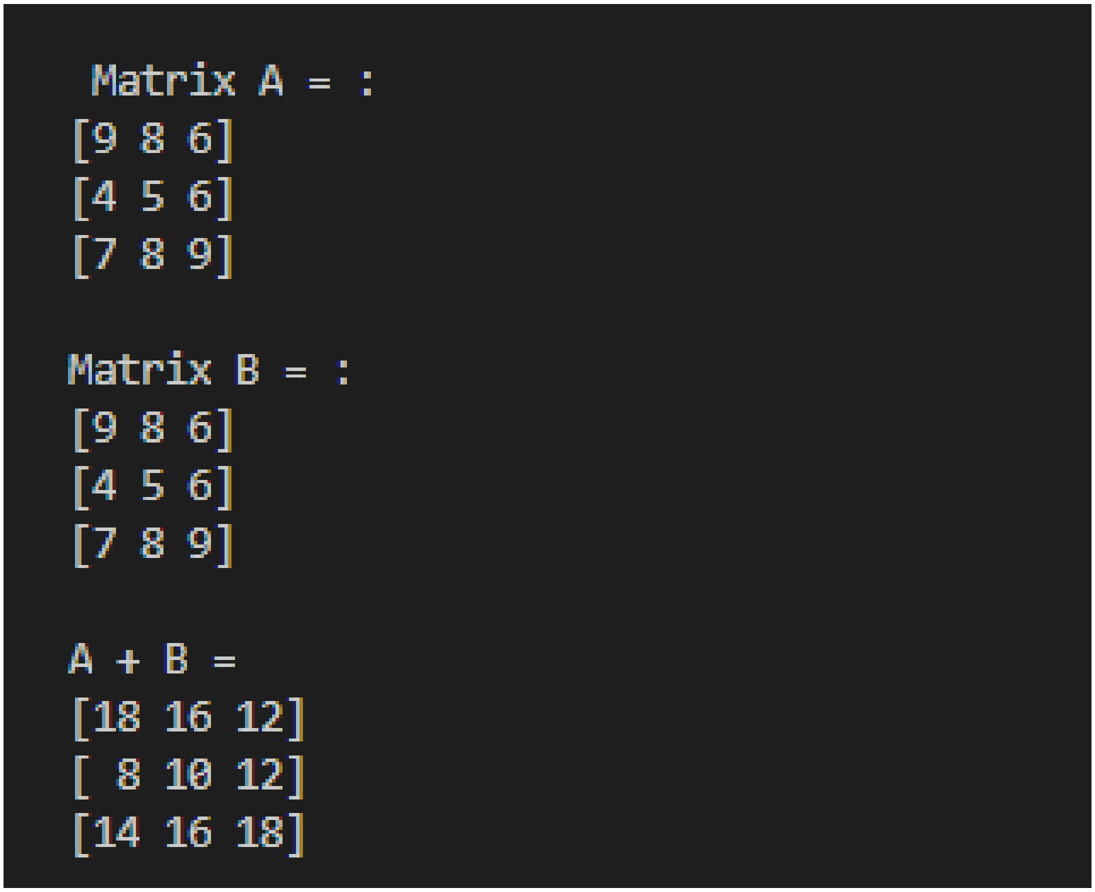
```
A Matrix of size 3x3 :
[ 1  4  5 12]
[-5  8  9  0]
[-6  7 11 19]
```

2)Add two matrices

CODE:

```
import numpy as np

A = np.array([[9,8,6],
  [4,5,6],
   [7,8,9]
   ])
B = np.array([[9,8,6],
   [4,5,6],
   [7,8,9]
   ])
print("\n Matrix A = :")
for row in A:
   print(row)
print("\nMatrix B = :")
for row in B:
   print(row)
print ("\nA + B = ")
c = A + B
for row in c:
   print(row)
```

OUTPUT :



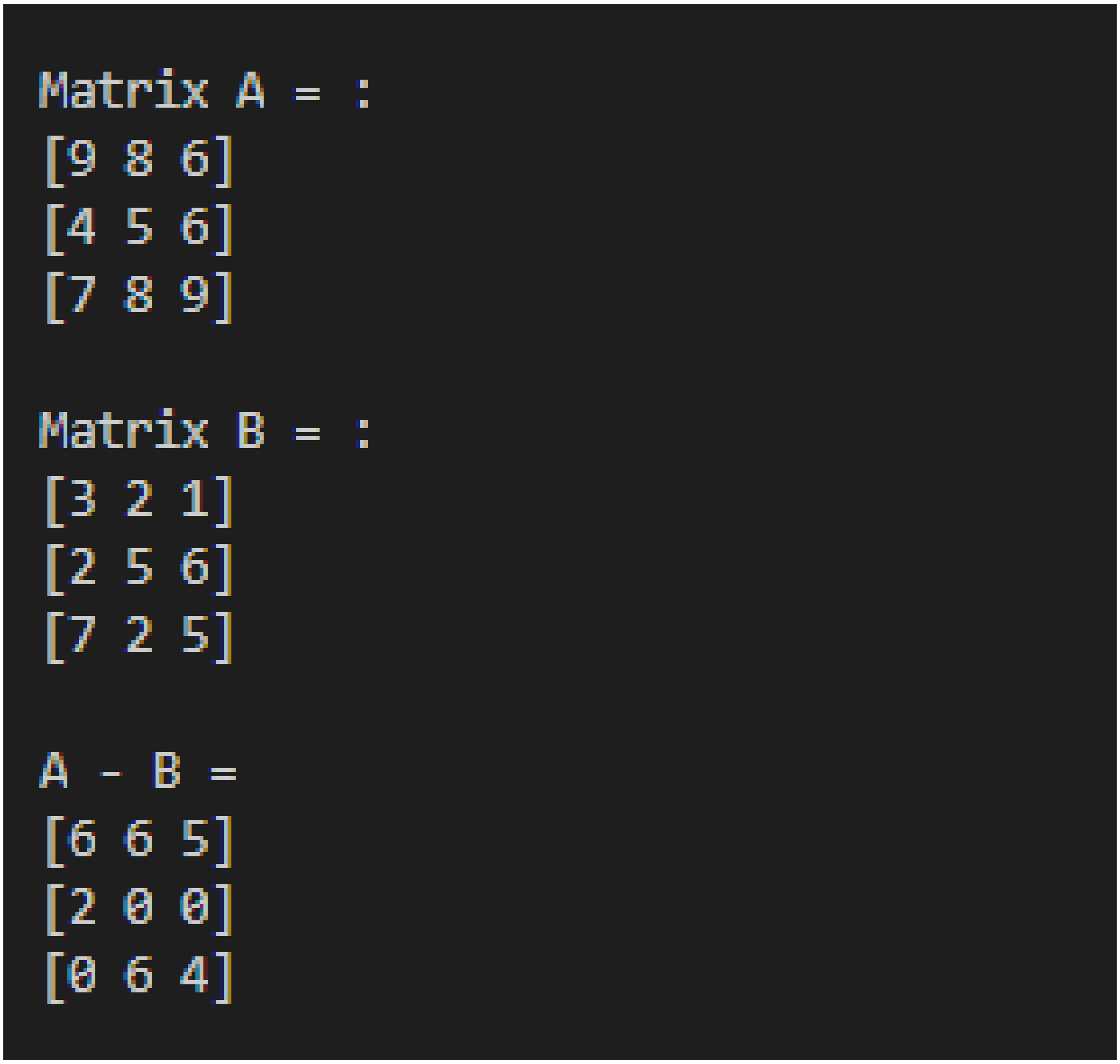
```
 Matrix A = :
[9 8 6]
[4 5 6]
[7 8 9]

Matrix B = :
[9 8 6]
[4 5 6]
[7 8 9]

A + B =
[18 16 12]
[ 8 10 12]
[14 16 18]
```

3)Subtract two matrices

CODE:

```
import numpy as np

A = np.array([[9,8,6],

    [4,5,6],

    [7,8,9]

    ])

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

    [2,5,6],
    [7,2,5]

    ])

print("\nMatrix A = :")

for row in A:

    print(row)

print("\nMatrix B = :")

for row in B:

    print(row)

print ("\nA - B = ")

c = A - B

for row in c:

    print(row)
```

OUTPUT :



```
Matrix A = :
[9 8 6]
[4 5 6]
[7 8 9]

Matrix B = :
[3 2 1]
[2 5 6]
[7 2 5]

A - B =
[6 6 5]
[2 0 0]
[0 6 4]
```

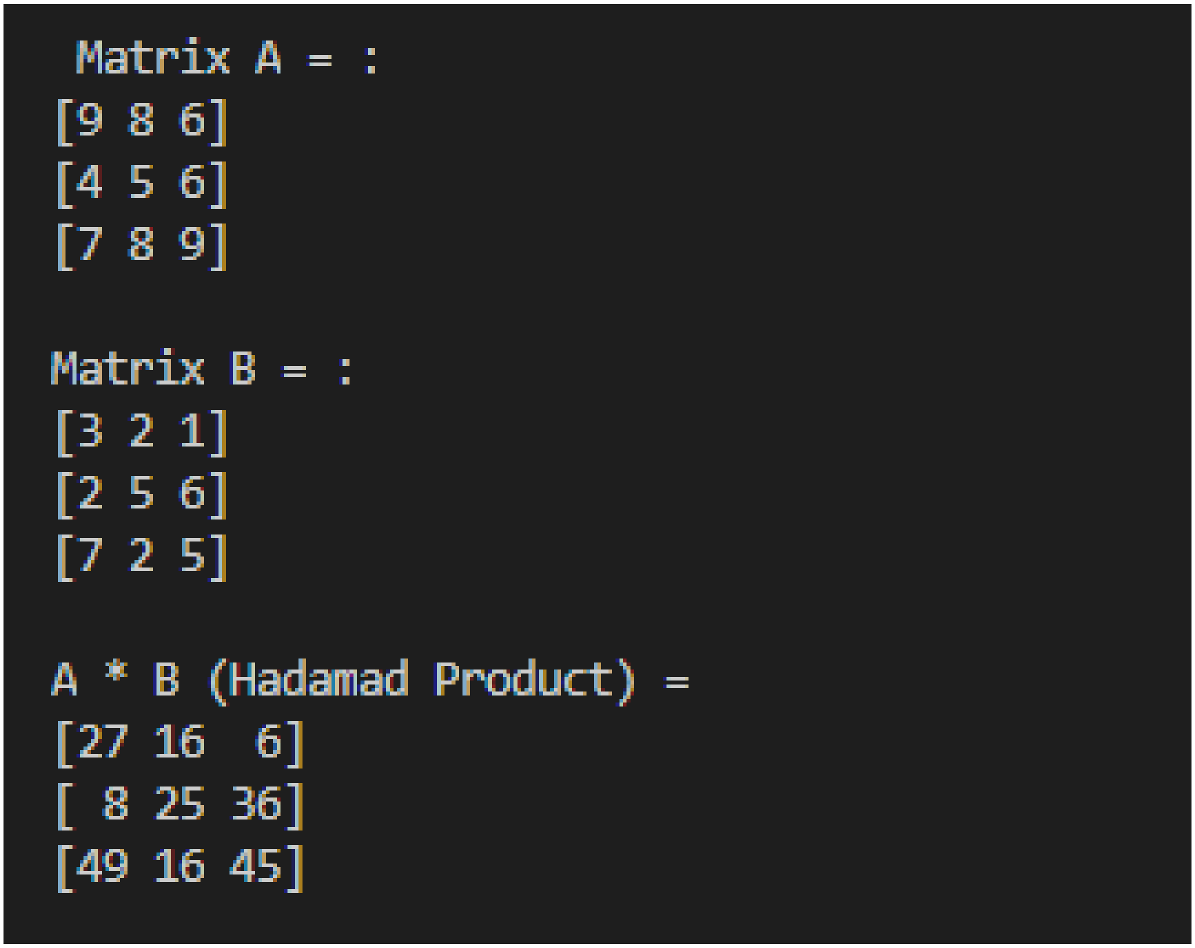
4)Find Hadamard product of two matrices

<u>CODE:</u>

```
import numpy as np

A = np.array([[9,8,6],

   [4,5,6],

   [7,8,9]

   ])
```

```
B = np.array([[3,2,1],
    [2,5,6],
    [7,2,5]
    ])
print("\n Matrix A = :")
for row in A:
    print(row)
print("\nMatrix B = :")
for row in B:
    print(row)
print ("\nA * B (Hadamad Product) = ")
c = A * B
for row in c:
    print(row)
```

OUTPUT :



```
 Matrix A = :
[9 8 6]
[4 5 6]
[7 8 9]

Matrix B = :
[3 2 1]
[2 5 6]
[7 2 5]

A * B (Hadamad Product) =
[27 16  6]
[ 8 25 36]
[49 16 45]
```

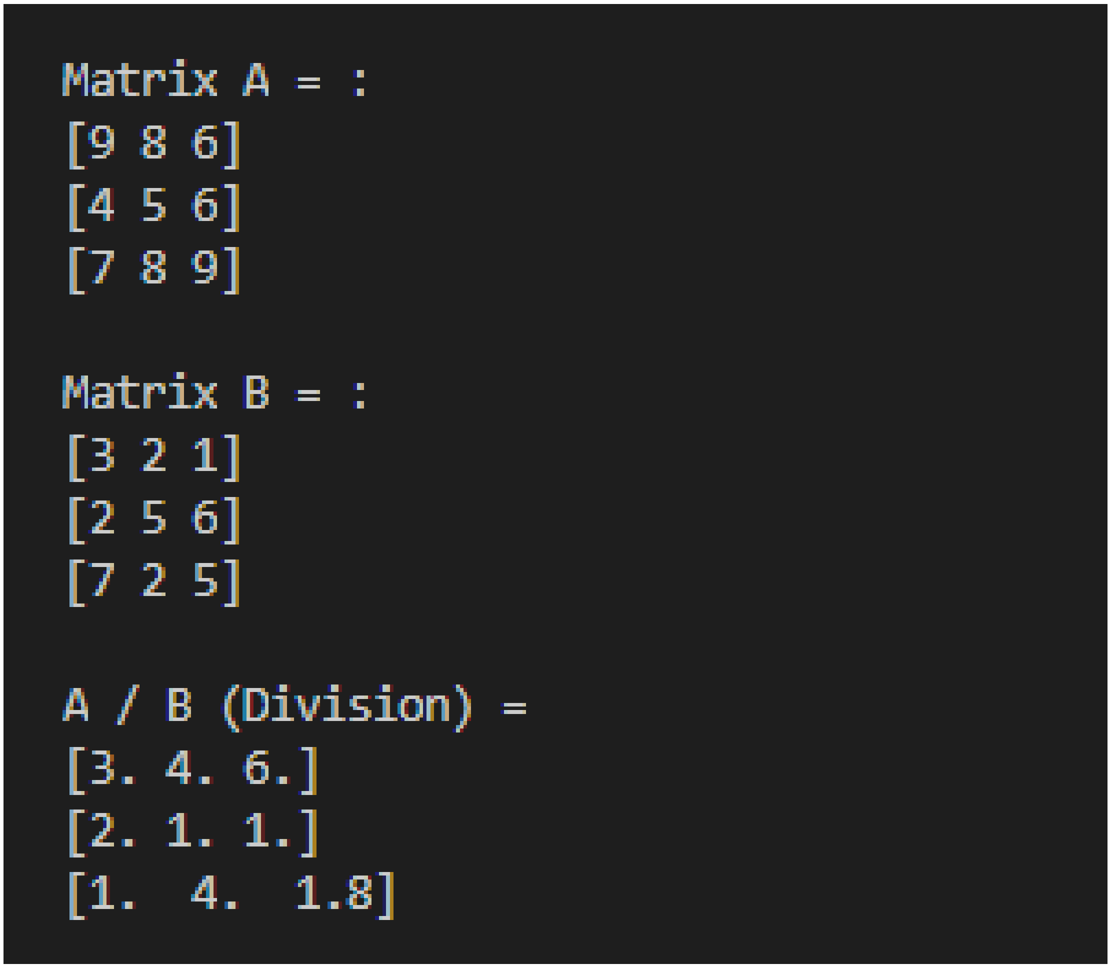
5)Divide two matrices

CODE:

```
import numpy as np

A = np.array([[9,8,6],

   [4,5,6],

   [7,8,9]

   ])
```

```
B = np.array([[3,2,1],

    [2,5,6],

    [7,2,5]

    ])

print("\nMatrix A = :")

for row in A:

    print(row)

print("\nMatrix B = :")

for row in B:

    print(row)

print ("\nA / B (Division) = ")

c = A / B

for row in c:

    print(row)
```

OUTPUT :



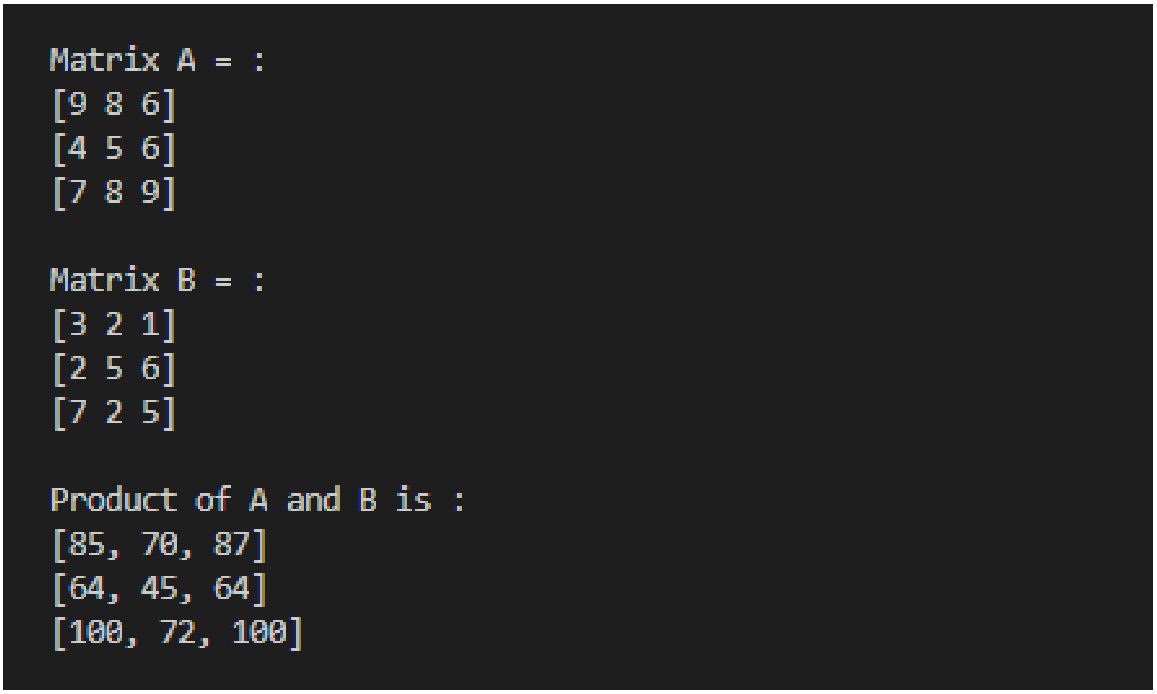
```
Matrix A = :
[9 8 6]
[4 5 6]
[7 8 9]

Matrix B = :
[3 2 1]
[2 5 6]
[7 2 5]

A / B (Division) =
[3. 4. 6.]
[2. 1. 1.]
[1.  4.  1.8]
```

6)Find product of two matrices

CODE:

```
import numpy as np

A = np.array([[9,8,6],

   [4,5,6],

   [7,8,9]

   ])

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

   [2,5,6],
    [7,2,5]

    ])

print("\nMatrix A = :")

for row in A:

    print(row)

print("\nMatrix B = :")

for row in B:

    print(row)

Result = [[0, 0, 0],

            [0, 0, 0],

            [0, 0, 0]]

for m in range(len(A)):

   for n in range(len(B[0])):

          for o in range(len(B)):

               Result[m][n] = Result[m][n] + A[m][o] * B[o][n]

print("\nProduct of A and B is :")

for row in Result:

   print(row)
```

OUTPUT :



```
Matrix A = :
[9 8 6]
[4 5 6]
[7 8 9]

Matrix B = :
[3 2 1]
[2 5 6]
[7 2 5]

Product of A and B is :
[85, 70, 87]
[64, 45, 64]
[100, 72, 100]
```

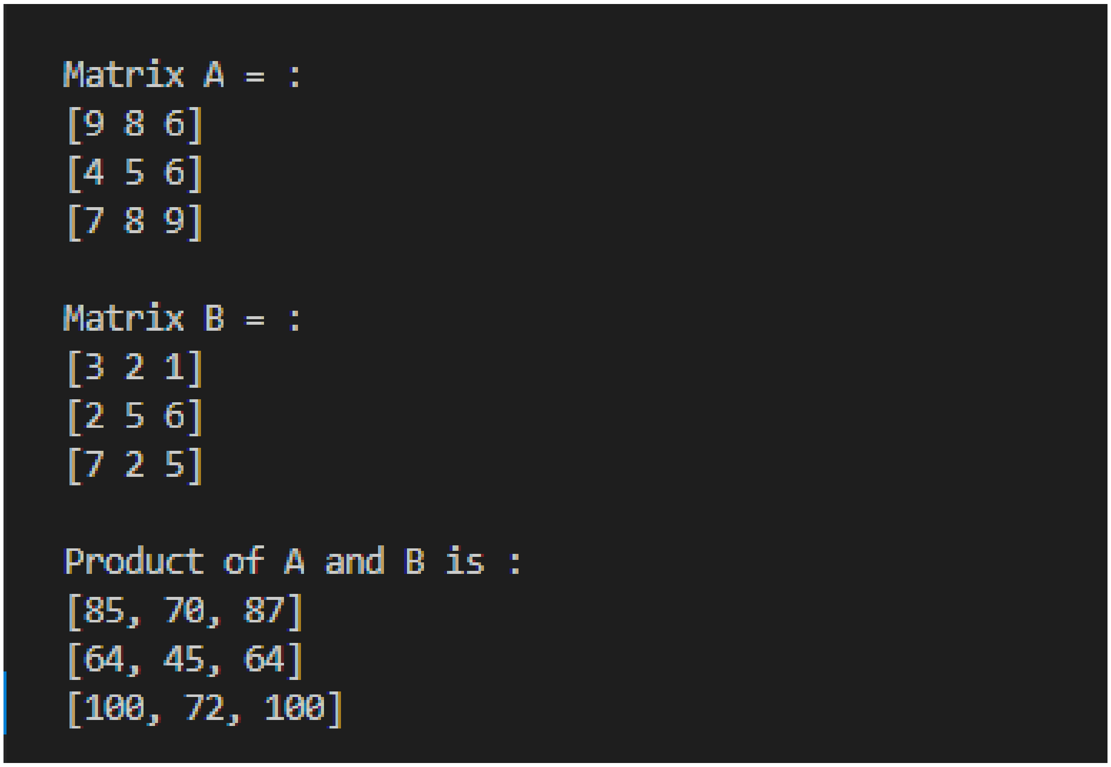
7)Perform vector matrix multiplication

CODE:

```
import numpy as np

A = np.array([[9,8,6],

    [4,5,6],

    [7,8,9]

    ])

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

    )

print("\nMatrix A = :")

for row in A:

    print(row)

    print("\nVector B = :")
```

```
for row in B:
    print(row ,end= " ")
print ("\n\nMatrice A and Vector B Vector Product = ")
c = np.dot(A,B)
for row in c:
    print(row)
```

OUTPUT :



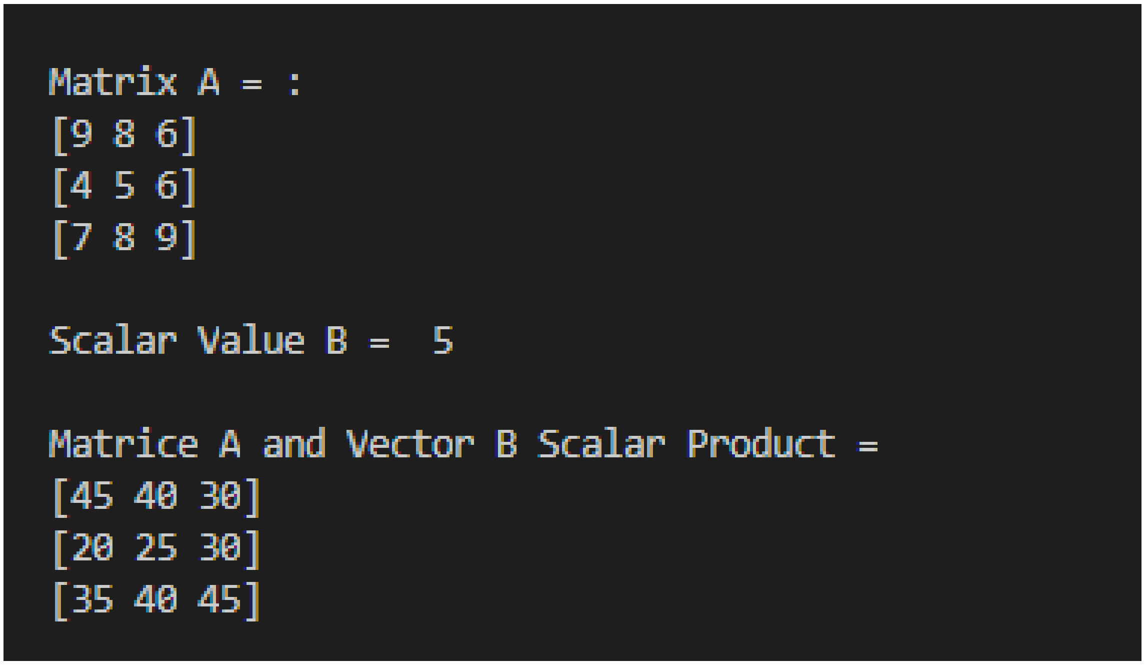
```
Matrix A = :
[9 8 6]
[4 5 6]
[7 8 9]

Matrix B = :
[3 2 1]
[2 5 6]
[7 2 5]

Product of A and B is :
[85, 70, 87]
[64, 45, 64]
[100, 72, 100]
```

8)Perform scalar matrix multiplication

CODE:

```
import numpy as np
A = np.array([[9,8,6],
    [4,5,6],
    [7,8,9]
    ])
print("\nMatrix A = :")
for row in A:
    print(row)
B = 5
print("\nScalar Value B = ",B)
print ("\nMatrice A and Vector B Scalar Product = ")
c = np.dot(A,B)
for row in c:
    print(row)
```

OUTPUT :



```
Matrix A = :
[9 8 6]
[4 5 6]
[7 8 9]

Scalar Value B =  5

Matrice A and Vector B Scalar Product =
[45 40 30]
[20 25 30]
[35 40 45]
```

9)Define a 3X3 square matrix . Extract the main diagonal as vector . Create the diagonal matrix from the extracted vector

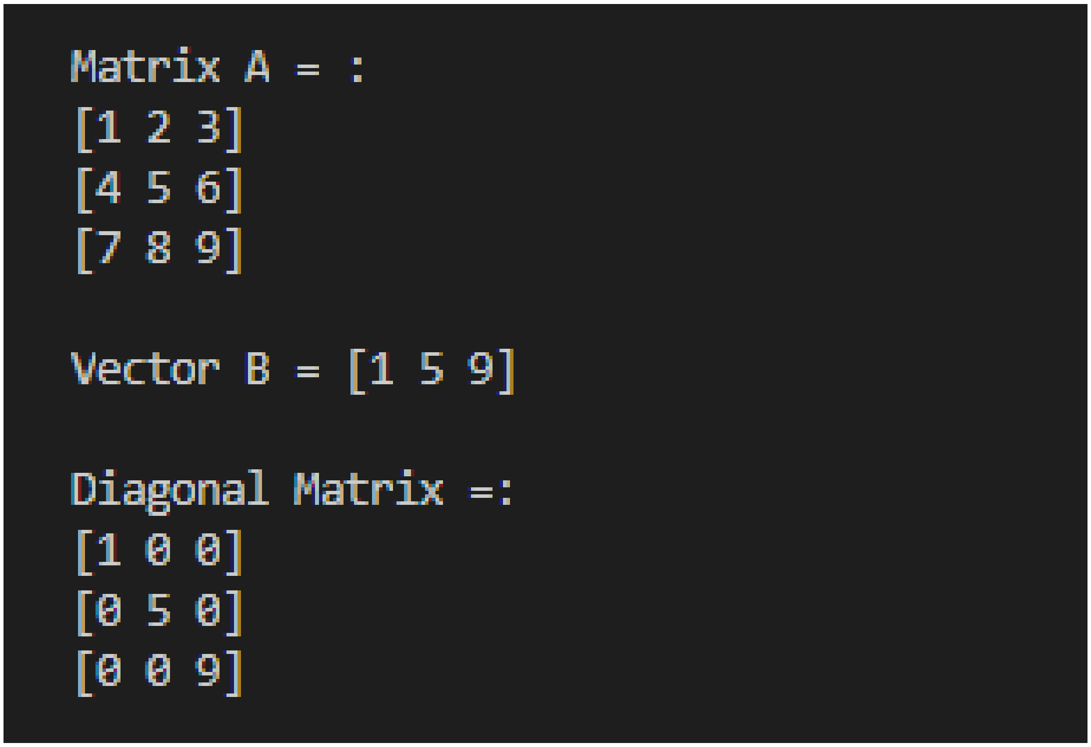
CODE:

```
import numpy as np

def diagonal(vector):

    result = []

    for i in range(3):

        row = []

        for j in range(3):

            if i == j :

                element = vector[i]

                row.append(element)

            else :
```

```
       element = 0
                row.append(element)
        result.append(row)
    return np.array(result)
def asVector(matrix):
    varray = []
    for i in range(3):
        for j in range(3):
            if i == j :
                element = matrix[i][j]
                varray.append(element)
    result = np.array(varray)
    return result
matrix = np.array([
    [1,2,3],
    [4,5,6],
    [7,8,9]
])
print("\nMatrix A = :")
for row in matrix:
    print(row)
vector = asVector(matrix)
print(f"\nVector B = {vector}")
```

```
diagonalMatrix = diagonal(vector)

print("\nDiagonal Matrix =:")

for row in diagonalMatrix :

    print(row)
```

OUTPUT :



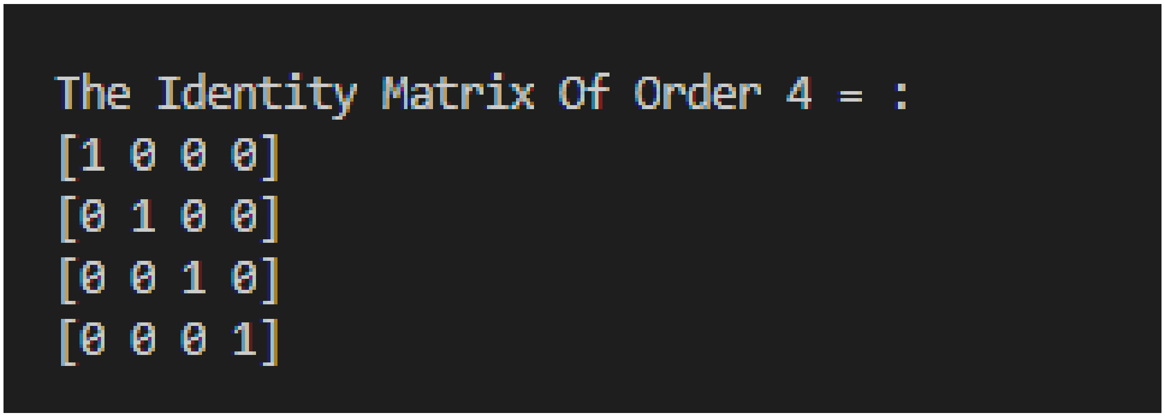
```
Matrix A = :
[1 2 3]
[4 5 6]
[7 8 9]

Vector B = [1 5 9]

Diagonal Matrix =:
[1 0 0]
[0 5 0]
[0 0 9]
```

10)Create an identity matrix of order 4

CODE:

```
import numpy as np

def createMatrix():

    result = []

    for i in range(4):
        row = []
        for j in range(4):
            if i == j :
                element = 1
                row.append(element)
            else :
                element = 0
                row.append(element)
        result.append(row)
    return np.array(result)
matrix = createMatrix()
print("\nThe Identity Matrix Of Order 4 = : ")
for row in matrix :
    print(row)
```

<u>OUTPUT :</u>



```
The Identity Matrix Of Order 4 = :
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

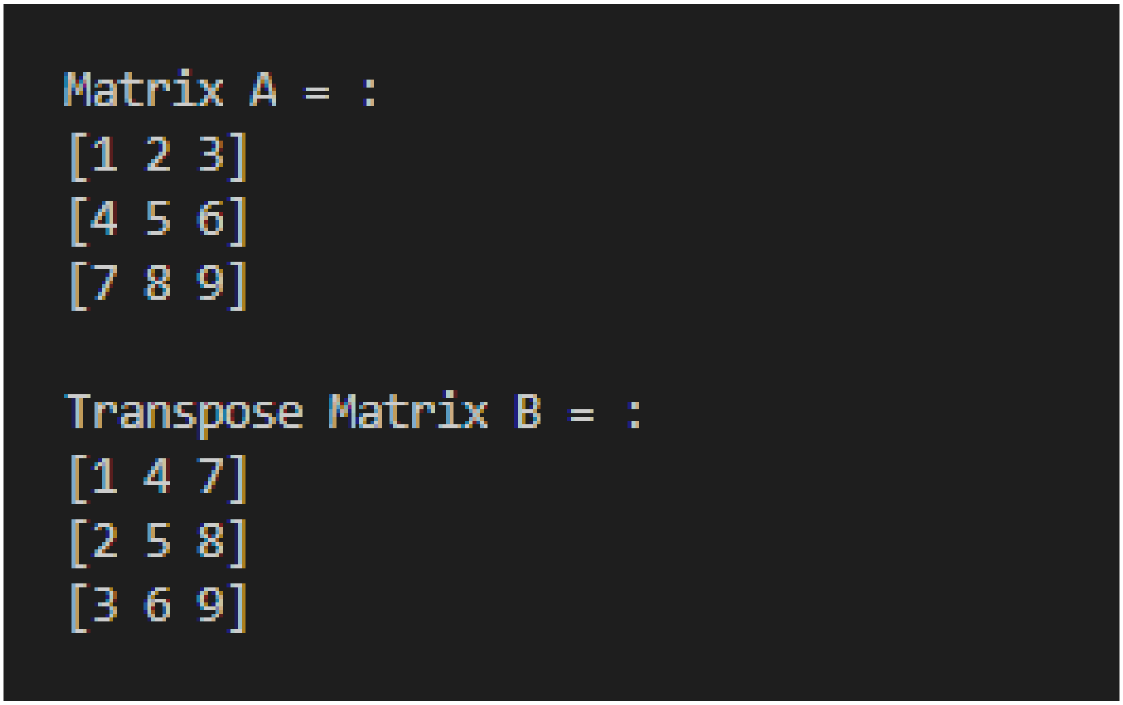
11)Find transpose of a matrix

CODE:

```
import numpy as np

def Transpose(matrix):

    result = []

    for i in range(3):

        row = []

        for j in range(3):

            element = matrix[j][i]

            row.append(element)

        result.append(row)

    return np.array(result)

matrix = np.array([

    [1,2,3],

    [4,5,6],

    [7,8,9]

])

print("\nMatrix A = :")

for row in matrix:

    print(row)

transpose = Transpose(matrix)
```

```
print("\nTranspose Matrix B = :")

for row in transpose:

    print(row)
```

OUTPUT :



```
Matrix A = :
[1 2 3]
[4 5 6]
[7 8 9]

Transpose Matrix B = :
[1 4 7]
[2 5 8]
[3 6 9]
```

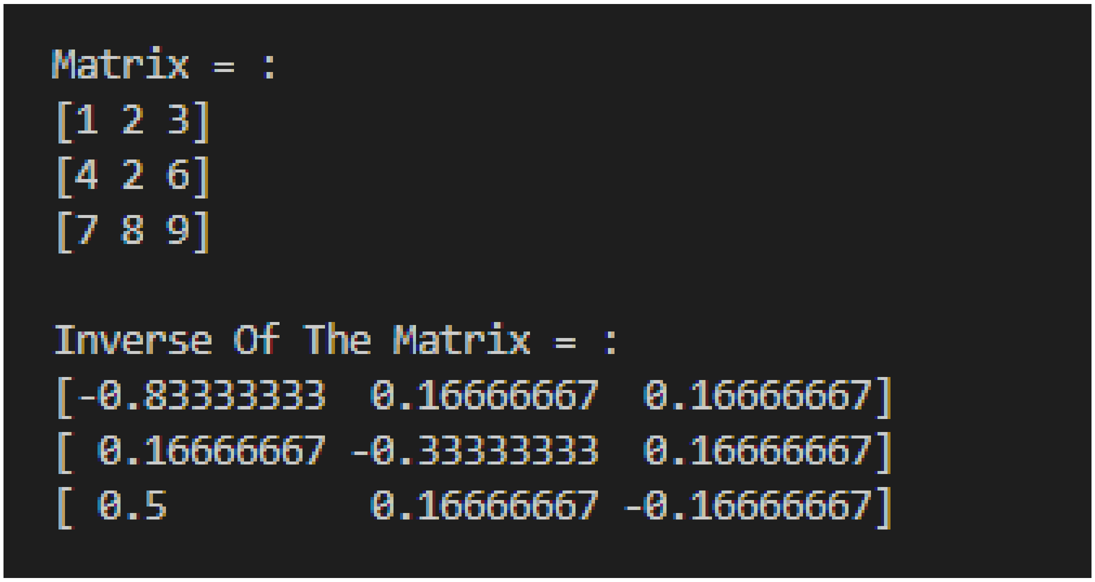
12)Print inverse of a matrix

CODE:

```
import numpy as np

from numpy.linalg import inv, det

def Inverse(matrix):

    determinent = det(matrix)

    if determinent == 0 :

        print("\nAn Inverse For The Matrix Does not Exist.")
```

```
     exit()
else :
        result = inv(matrix)
        return result
matrix = np.array([
    [1,2,3],
    [4,2,6],
    [7,8,9]
])
print("\nMatrix = :")
for row in matrix:
    print(row)
inverse = Inverse(matrix)
print("\nInverse Of The Matrix = :")
for row in inverse:
    print(row)
```

OUTPUT :



```
Matrix = :
[1 2 3]
[4 2 6]
[7 8 9]

Inverse Of The Matrix = :
[-0.83333333  0.16666667  0.16666667]
[ 0.16666667 -0.33333333  0.16666667]
[ 0.5         0.16666667 -0.16666667]
```

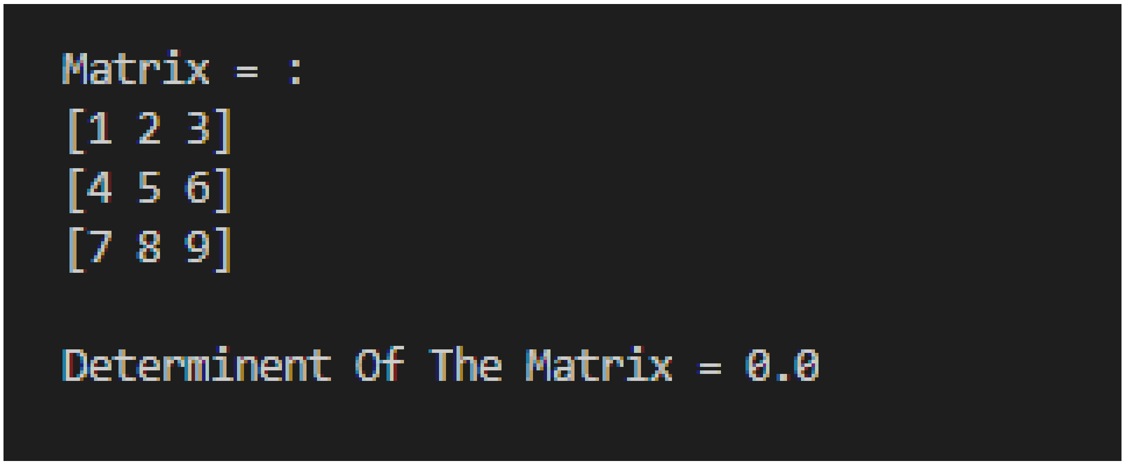
13)Print the determinant of a matrix

CODE:

```
import numpy as np
from numpy.linalg import det
def Determinent(matrix):
    determinent = det(matrix)
    return determinent
matrix = np.array([
    [1,2,3],
    [4,5,6],
    [7,8,9]
])
print("\nMatrix = :")
for row in matrix:
    print(row)
```

```
determinent = Determinent(matrix)

print(f"\nDeterminent Of The Matrix = {determinent}\n")
```

OUTPUT :



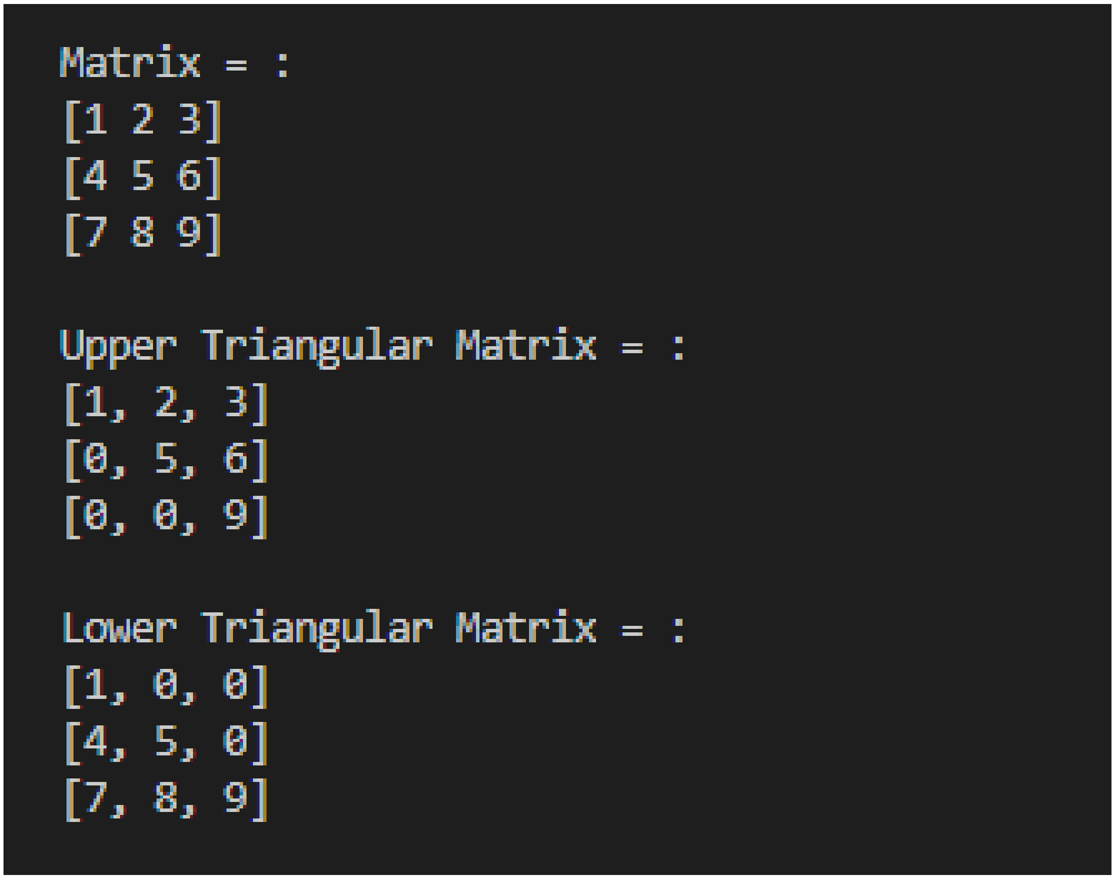
```
Matrix = :
[1 2 3]
[4 5 6]
[7 8 9]

Determinent Of The Matrix = 0.0
```

a)Lower and Upper Triangular Matrix

CODE:

```
import numpy as np

def Upper(matrix):

    result = []

    for i in range(3):

        row = []

        for j in range(3):

            if i > j :

                element = 0

                row.append(element)

            else :
                element = matrix[i][j]
                row.append(element)
        result.append(row)
    return result
def Lower(matrix):
    result = []
    for i in range(3):
        row = []
        for j in range(3):
            if i < j :
                element = 0
                row.append(element)
            else :
                element = matrix[i][j]
                row.append(element)
        result.append(row)
    return result
matrix = np.array([
    [1,2,3],
    [4,5,6],
    [7,8,9]
])
print("\nMatrix = :")
for row in matrix:
    print(row)
upperTriangular = Upper(matrix)
lowerTriangular = Lower(matrix)
print(f"\nUpper Triangular Matrix = :")
for row in upperTriangular:
    print(row)
print(f"\nLower Triangular Matrix = :")
for row in lowerTriangular:
    print(row)
```

OUTPUT :



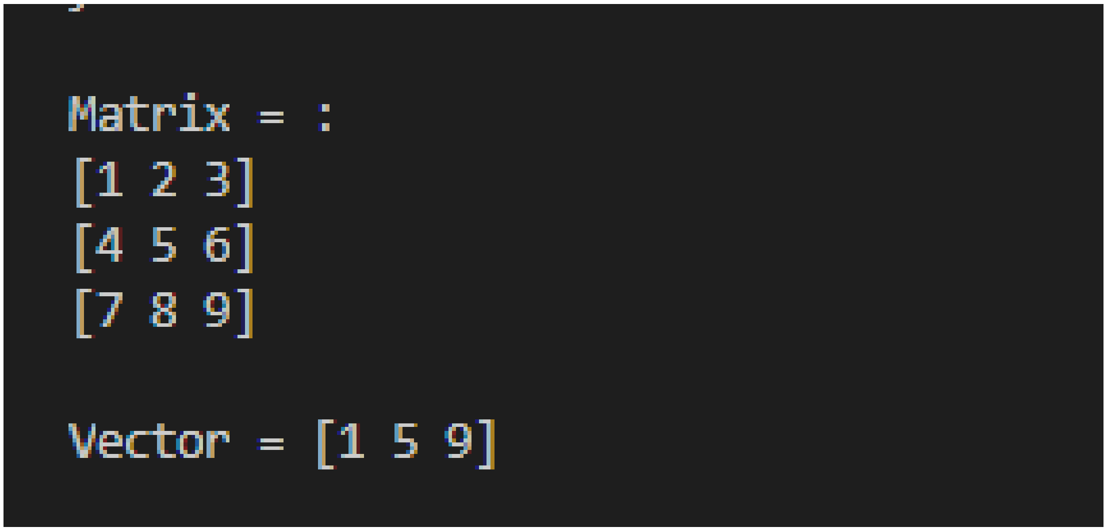
```
Matrix = :
[1 2 3]
[4 5 6]
[7 8 9]

Upper Triangular Matrix = :
[1, 2, 3]
[0, 5, 6]
[0, 0, 9]

Lower Triangular Matrix = :
[1, 0, 0]
[4, 5, 0]
[7, 8, 9]
```

b)Main Diagonal Of a Matrix As Vector

CODE:

```
import numpy as np
def asVector(matrix):
    varray = []
    for i in range(3):
        for j in range(3):
            if i == j :
                element = matrix[i][j]
                varray.append(element)
    result = np.array(varray)
    return result
matrix = np.array([
    [1,2,3],
    [4,5,6],
    [7,8,9]
])
print("\nMatrix = :")
for row in matrix:
    print(row)
vector = asVector(matrix)
print(f"\nVector = {vector}")
```

OUTPUT :



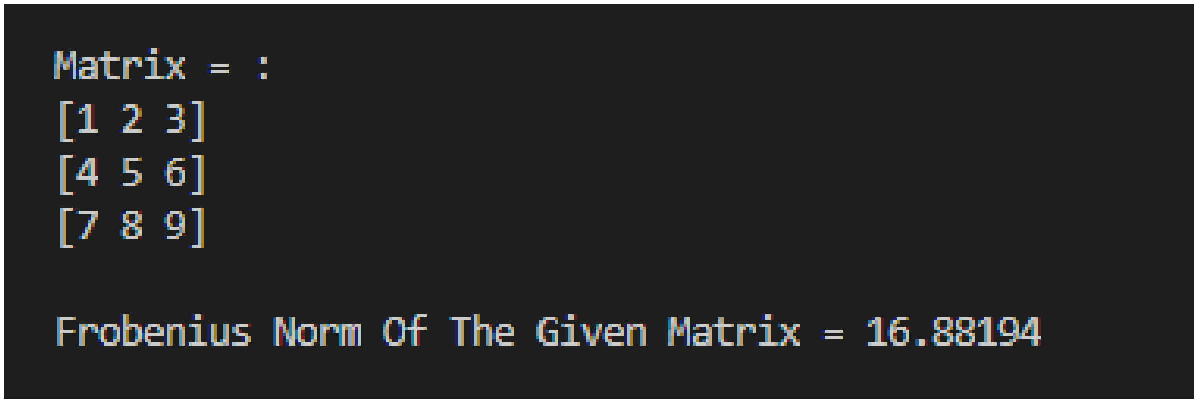
```
Matrix = :
[1 2 3]
[4 5 6]
[7 8 9]

Vector = [1 5 9]
```

c)Frobenius Norms

CODE:

```
from math import sqrt

import numpy as np

def frobenius(matrix):

    sum = 0

    for row in matrix:

        for element in row:

            elementsquare = element * element

            sum = sum + elementsquare

    result = sqrt(sum)

    return round(result,5)

matrix = np.array([

    [1,2,3],
    [4,5,6],

    [7,8,9]

])

print("\nMatrix = :")

for row in matrix:

    print(row)

frobenius = frobenius(matrix)

print(f"\nFrobenius Norm Of The Given Matrix = {frobenius}")
```

OUTPUT :



```
Matrix = :
[1 2 3]
[4 5 6]
[7 8 9]

Frobenius Norm Of The Given Matrix = 16.88194
```

SET-3

1)Create an orthogonal matrix and check Q'Q=Q Q'=I

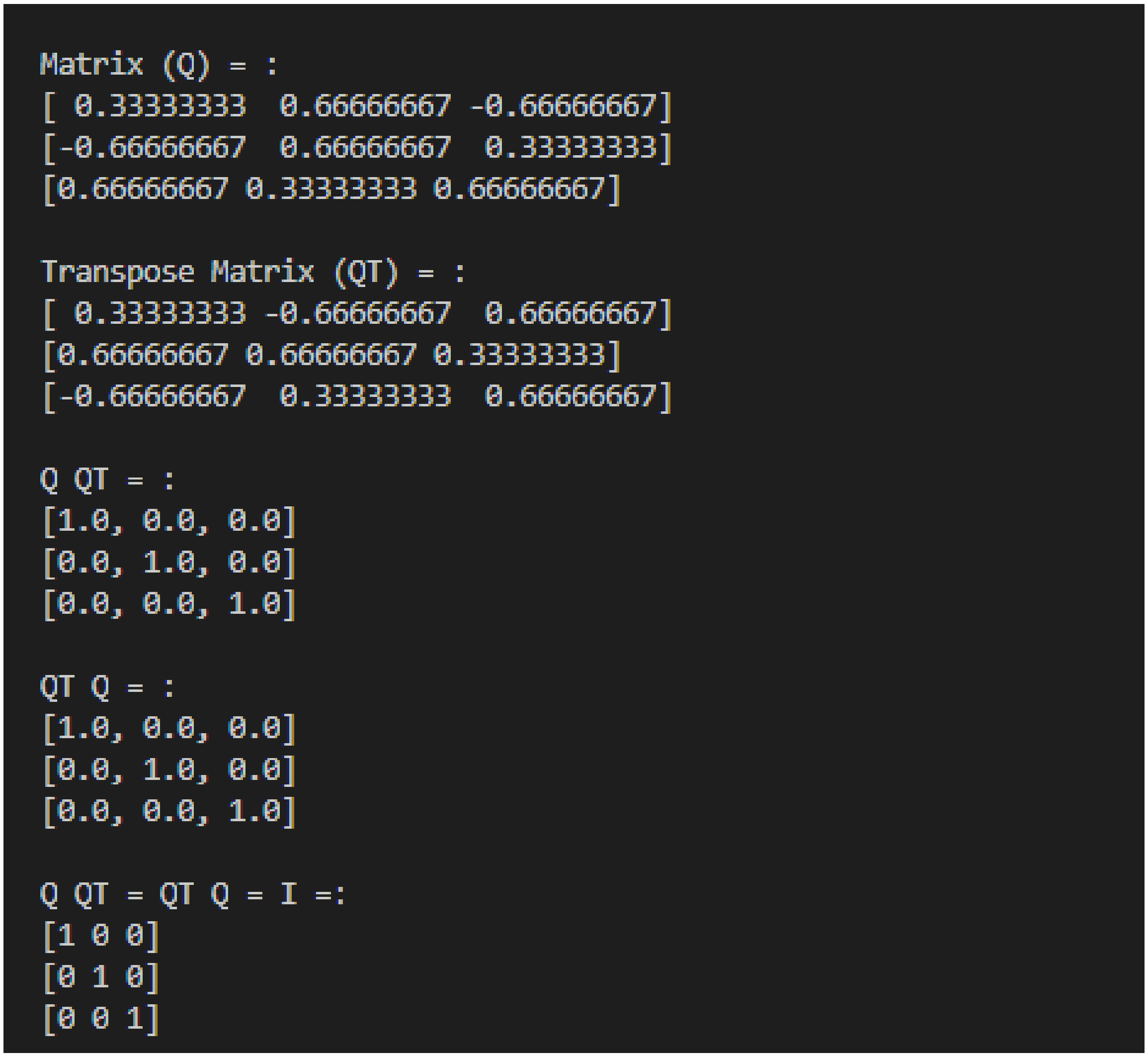
CODE:

```
import numpy as np

def multiplication(A , B):

    Result = [[0, 0, 0],

            [0, 0, 0],
            [0, 0, 0]]
     for m in range(len(A)):
        for n in range(len(B[0])):
            for o in range(len(B)):
                Result[m][n] = Result[m][n] + A[m][o] * B[o][n]
    return Result
def createidentity(rows,col):
    result = []
    for i in range(rows):
        row = []
        for j in range(col):
            if i == j :
                element = 1
            else :
                element = 0
            row.append(element)
        result.append(row)
    return np.array(result)
def transpose(matrix):
    result=[]
    row = len(matrix)
    col = len(matrix[0])
    for i in range(row):
        rowoft = []
```

```
      for j in range(col):

            element = matrix[j][i]

            rowoft.append(element)

        result.append(rowoft)

        return result

matrix = np.array([

    [1/3,2/3, -2/3],

    [-2/3,2/3,1/3],

    [2/3,1/3,2/3]])

print("\nMatrix (Q) = :")

for row in matrix:

    print(row)

transpose = np.array(transpose(matrix))

print("\nTranspose Matrix (QT) = :")

for row in transpose:

    print(row)

rows = len(matrix)

cols = len(matrix[0])

identitymatrix = createidentity(rows,cols)

QQT = multiplication(matrix , transpose)

print(f"\nQ QT = :")

for row in QQT:

    print(row)

QTQ = multiplication(transpose ,matrix )
```

```
print(f"\nQT Q = :")

for row in QTQ:

    print(row)

print("\nQ QT = QT Q = I =: ")

for row in identitymatrix:

    print(row)
```

OUTPUT :



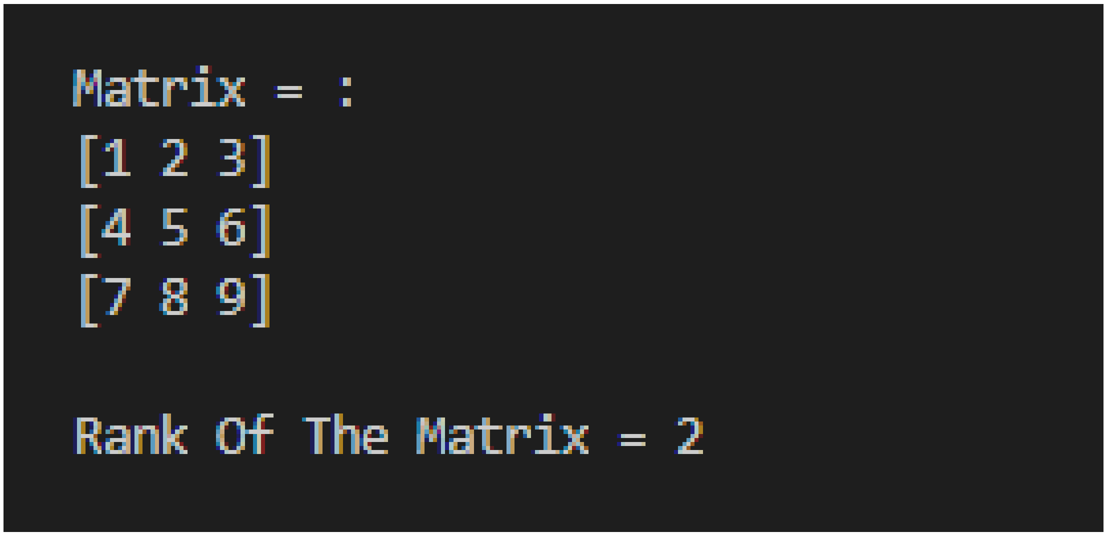
```
Matrix (Q) = :
[ 0.33333333  0.66666667 -0.66666667]
[-0.66666667  0.66666667  0.33333333]
[0.66666667 0.33333333 0.66666667]

Transpose Matrix (QT) = :
[ 0.33333333 -0.66666667  0.66666667]
[0.66666667 0.66666667 0.33333333]
[-0.66666667  0.33333333  0.66666667]

Q QT = :
[1.0, 0.0, 0.0]
[0.0, 1.0, 0.0]
[0.0, 0.0, 1.0]

QT Q = :
[1.0, 0.0, 0.0]
[0.0, 1.0, 0.0]
[0.0, 0.0, 1.0]

Q QT = QT Q = I =:
[1 0 0]
[0 1 0]
[0 0 1]
```

2)Print rank of a matrix

CODE:

```
import numpy as np
def rankof(matrix):
    result = np.linalg.matrix_rank(matrix)
    return result
matrix = np.array([
    [1,2,3],
    [4,5,6],
    [7,8,9]
])
print("\nMatrix = :")
for row in matrix:
    print(row)
rank =rankof(matrix)
print(f"\nRank Of The Matrix = {rank}")
```

OUTPUT :



```
Matrix = :
[1 2 3]
[4 5 6]
[7 8 9]

Rank Of The Matrix = 2
```

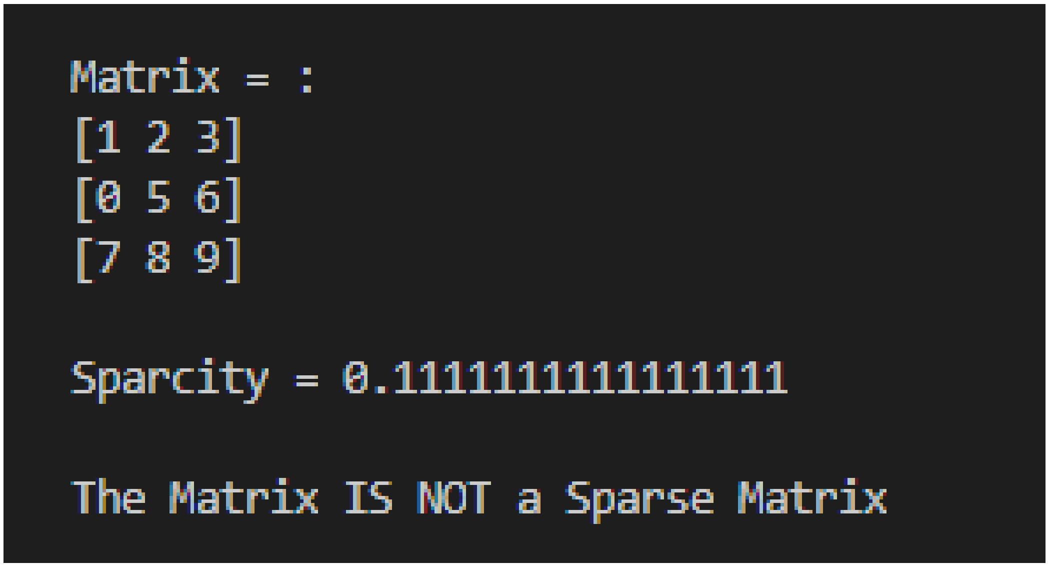
3)Calculate Sparsity of a matrix

CODE:

```
import numpy as np
def sparcity(matrix):
    count = 0
    for row in matrix:
        for element in row :
            if element == 0:
                count = count+1
    rows = len(matrix)
    cols = len(matrix[0])
    check = (rows*cols)/2
    sparse =  count/(rows*cols)
    print (f"\nSparcity = {sparse}")
    if count > check :
```

```
print("\nThe Matrix Is An Sparse Matrix")

    else :

        print("\nThe Matrix IS NOT a Sparse Matrix")

matrix = np.array([

    [1,2,3],

    [0,5,6],

    [7,8,9]

])

print("\nMatrix = :")

for row in matrix:

    print(row)

sparcity(matrix)
```

OUTPUT :



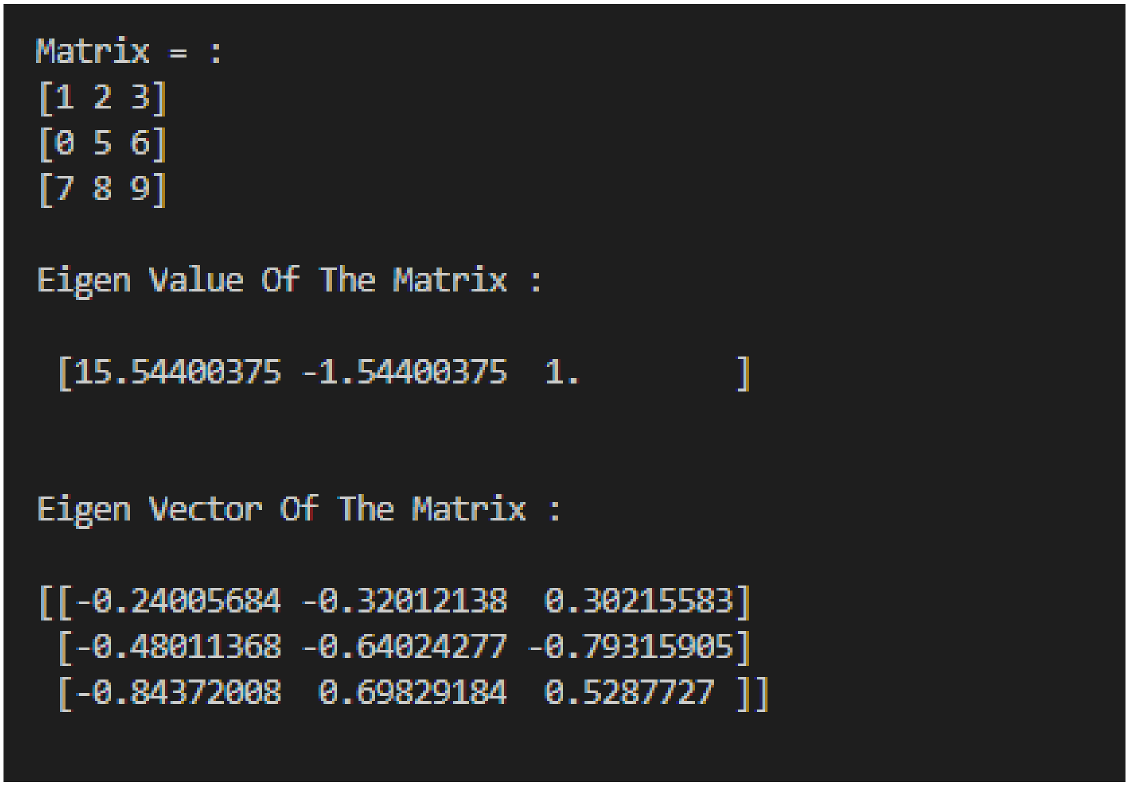
```
Matrix = :
[1 2 3]
[0 5 6]
[7 8 9]

Sparcity = 0.1111111111111111

The Matrix IS NOT a Sparse Matrix
```

4)Print Eigen values and Eigen vectors of a matrix

CODE:

```
import numpy as np
from numpy.linalg import eig
def Eigen(matrix):
    Evalue,Evector = eig(matrix)
    value = np.array(Evalue)
    vector = np.array(Evector)
    print(f"\nEigen Value Of The Matrix :\n\n {value}\n\n\nEigen Vector Of The Matrix :\n\n{vector}\n\n")
matrix = np.array([
    [1,2,3],
    [0,5,6],
    [7,8,9]
])
print("\nMatrix = :")
for row in matrix:
    print(row)
Eigen(matrix)
```

OUTPUT :



```
Matrix = :
[1 2 3]
[0 5 6]
[7 8 9]

Eigen Value Of The Matrix :

 [15.54400375 -1.54400375  1.        ]


Eigen Vector Of The Matrix :

[[-0.24005684 -0.32012138  0.30215583]
 [-0.48011368 -0.64024277 -0.79315905]
 [-0.84372008  0.69829184  0.5287727 ]]
```

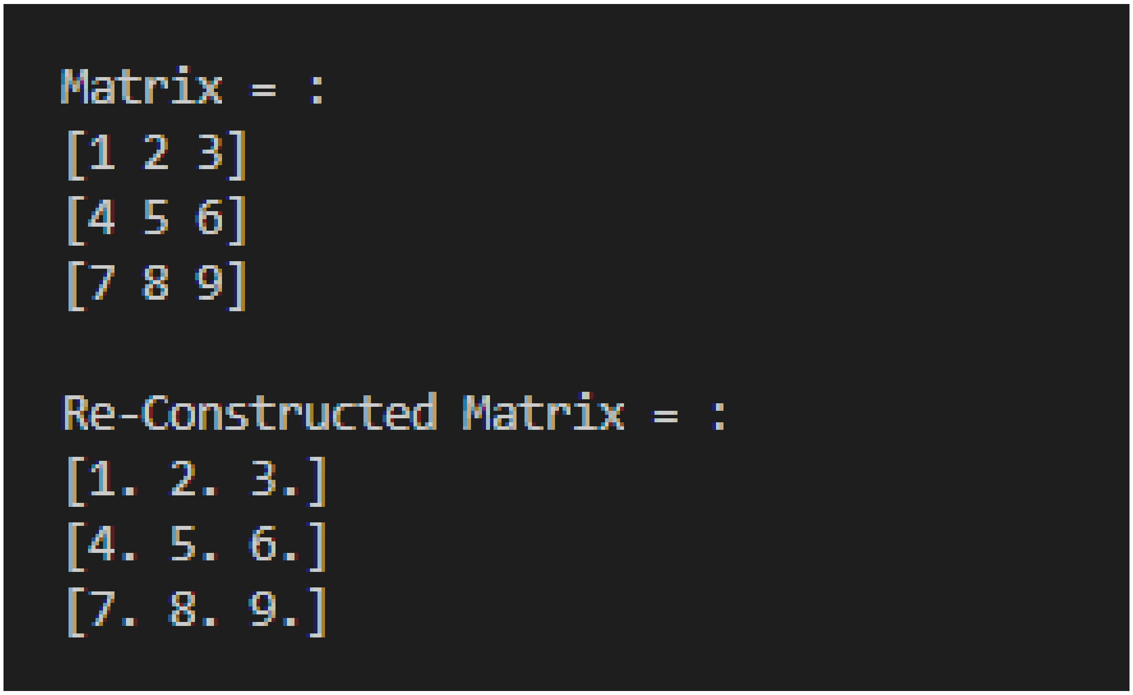
5)Calculate Eigen values and Eigen Vectors of a matrix and reconstruct The matrix

CODE:

```
from numpy import diag

import numpy as np

from numpy import dot

from numpy.linalg import inv

from numpy.linalg import eig


matrix = np.array([

   [1,2,3],

   [4,5,6],

   [7,8,9]])
```

```
print("\nMatrix = :")
for row in matrix:
    print(row)
Evalues, Evectors = eig(matrix)
invEig = inv(Evectors)
Diagfrmvect = diag(Evalues)
rematrix = Evectors.dot(Diagfrmvect).dot(invEig)
print("\nRe-Constructed Matrix = :")
for row in rematrix:
    print(row)
```

OUTPUT :



```
Matrix = :
[1 2 3]
[4 5 6]
[7 8 9]

Re-Constructed Matrix = :
[1. 2. 3.]
[4. 5. 6.]
[7. 8. 9.]
```

6)Define a 5X2 matrix data set , split it into x and y components and plot the dataset as scatterplot

CODE:

```
import matplotlib.pyplot as plt

import numpy as np

matrix = np.array([

   [8,99],[6,85],[4,88],[12,100],[8,85]]

)

x,y=np.split(matrix,2,axis=1)

plt.scatter(x, y)

plt.show()
```

OUTPUT :

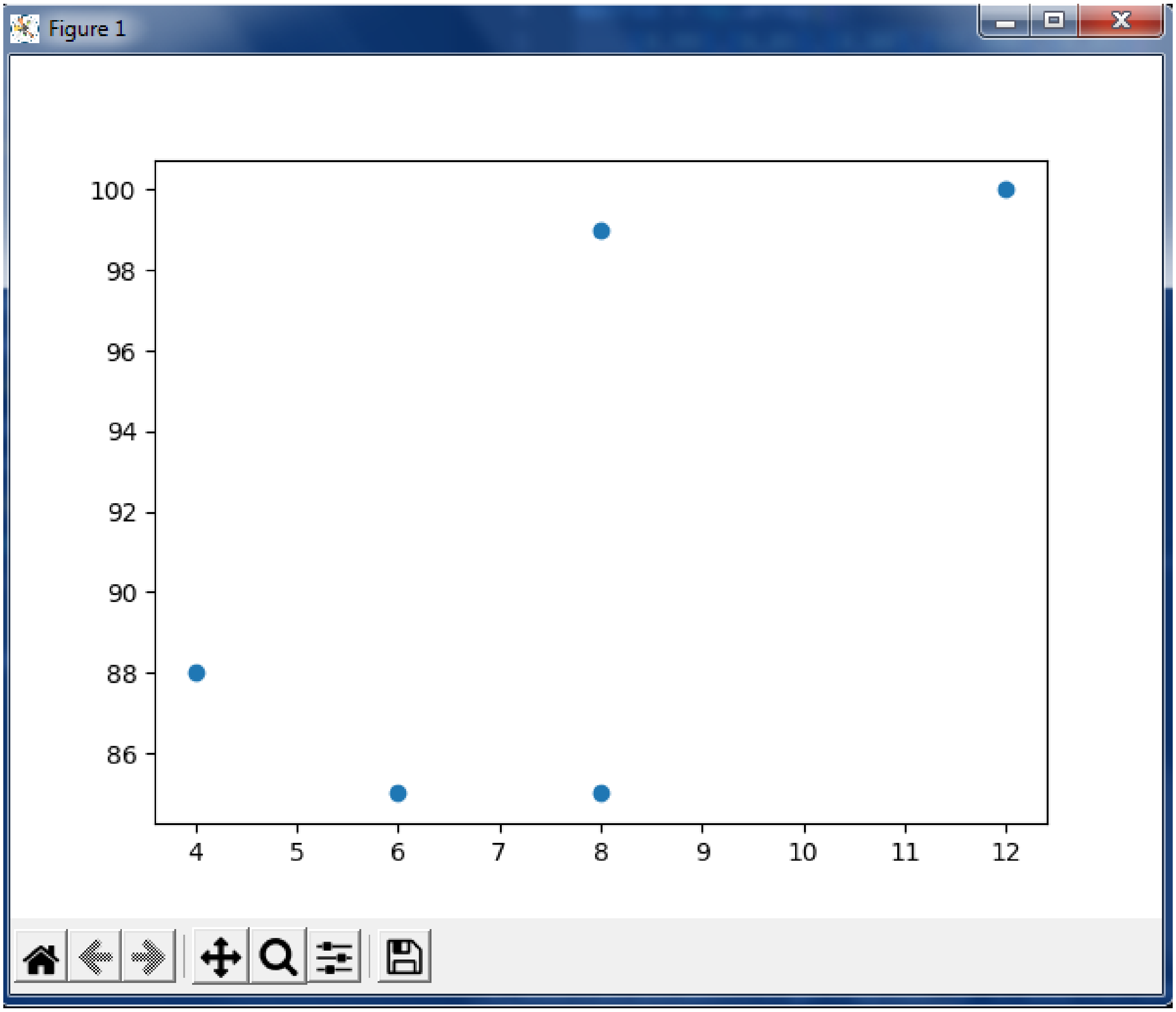


Figure 1
100
98
96
94
92
90
88
86
4
5
6
7
8
9
10
11
12