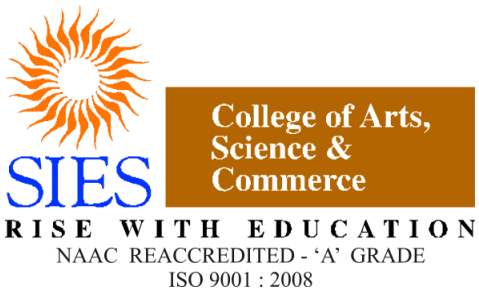
Name - Sumanth Ganeshan Udaiyar
Roll no - SCS2122076



S.I.E.S College of Arts, Science and Commerce
Sion(W), Mumbai – 400 022.

CERTIFICATE

This is to certify that Mr. **Sumanth Ganeshan Udaiyar** Roll No. **SCS2122076** Has successfully completed the necessary course of experiments in the subject **Linear Algebra Using Python** during the academic year **2021 – 2022** complying with the requirements of **University of Mumbai**,

for the course of **S.Y. BSc. Computer Science [Semester-4]**

Prof. In-Charge
Prof. Manoj Singh
(LAP)

Examination Date:
Examiner's Signature & Date:

Head of the Department
Prof. Manoj Singh

College Seal
And

Date

Practical 1

Aim: Write a program which demonstrates the following:

- Addition of two complex numbers.
- Displaying the conjugate of a complex number.
- Plotting a set of complex numbers.
- Creating a new plot by rotating the given number by a degree 90, 180, 270 degrees and also by scaling by a number a=1/2, a=1/3, a=2 etc.

Code:

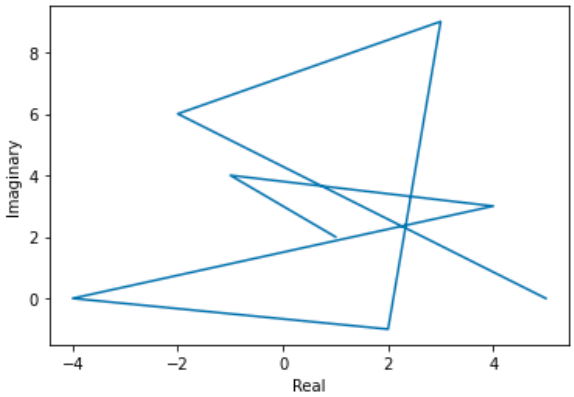
```
import matplotlib.pyplot as plt
import numpy as np

print("SELECT OPERATION: ")
print("1. ADDITION OF TWO COMPLEX NUMBERS")
print("2. DISPLAY CONJUGATE OF TWO COMPLEX NUMBERS")
print("3. PLOTING SET OF TWO COMPLEX NUMBERS")
print("4. ROTATION")
print("5. SCALING")
print("6. EXIT")

while True:
   ch= int(input("ENTER CHOICE OF OPERATION"))
   if ch==1:
      c1=complex(input("ENTER COMPLEX FIRST NUMBER: "))
      c2=complex(input("ENTER COMPLEX SECOND NUMBER: "))
      print("ADDITION OF TWO COMPLEX NUMBERS IS: ", c1+c2)
   elif ch==2:
      c1=complex(input("ENTER COMPLEX NUMBER"))
      res=np.conj(c1)
      print("CONJUGATE OF ",c1,"IS",res)
elif ch==3:
      data=[1+2j,-1+4j,4+3j,-4,2-1j,3+9j,-2+6j,5]
```

```
        x=[i.real for i in data]
        y=[i.imag for i in data]
        plt.plot(x,y)
        plt.ylabel('Imaginary')
        plt.xlabel('Real')
        plt.show()
    elif ch==4:
        S=[1+2j,-1+4j,4+3j,-4,2-1j,3+9j,-2+6j,5]
        angle=int(input("ENTER ANGLE OF ROTATION: "))
        if angle==90:
            S1={x*1j for x in S}
        elif angle==180:
            S1={x*(-1) for x in S}
        elif angle==270:
            S1={x*1j*(-1) for x in S}
        else:
            print("INVALID ANGLE ENTERED")
        X=[x.real for x in S1]
        Y=[x.imag for x in S1]
        plt.plot(X,Y,'ro')
        plt.show()
    elif ch==5:
        S=[1+2j,-1+4j,4+3j,-4,2-1j,3+9j,-2+6j,5]
        scale=float(input("ENTER SCALE POINT LIKE (.5) for 1/2"))
        S1={x*scale for x in S}
        X=[x.real for x in S1]
        Y=[x.imag for x in S1]
        plt.plot(X,Y,'ro')
        plt.show()
    elif ch==6:
        break
```

Output:

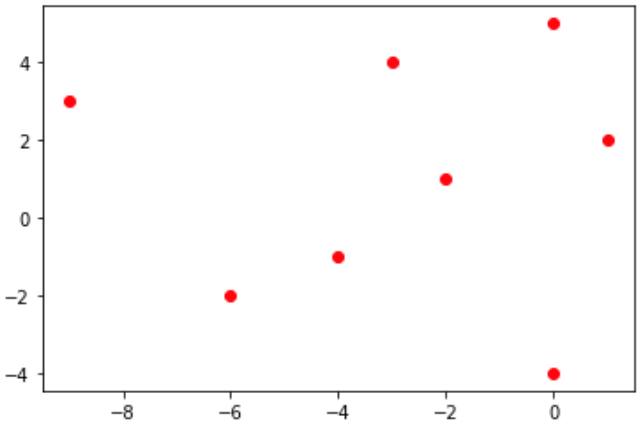
```
SELECT OPERATION:
1. ADDITION OF TWO COMPLEX NUMBERS
2. DISPLAY CONJUGATE OF TWO COMPLEX NUMBERS
3. PLOTING SET OF TWO COMPLEX NUMBERS
4. ROTATION
5. SCALING
6. EXIT
ENTER CHOICE OF OPERATION3
```

```
ENTER CHOICE OF OPERATION3
```

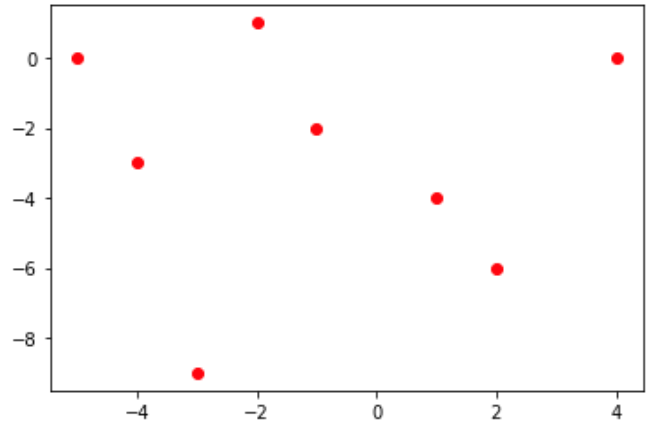


Real

```
ENTER CHOICE OF OPERATION4
ENTER ANGLE OF ROTATION: 90
```

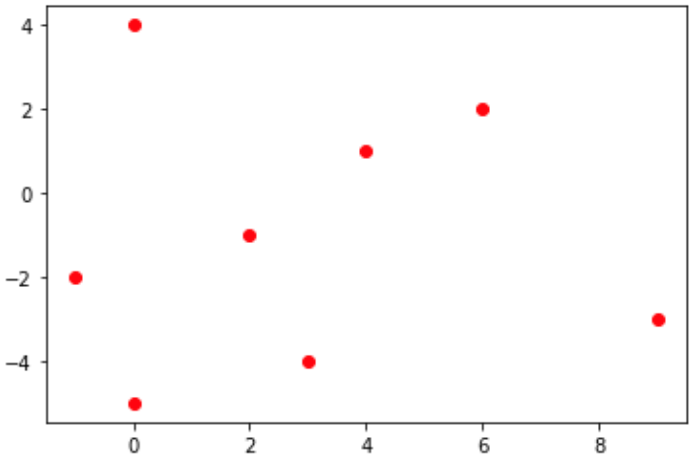


```
ENTER CHOICE OF OPERATION4
ENTER ANGLE OF ROTATION: 180
```



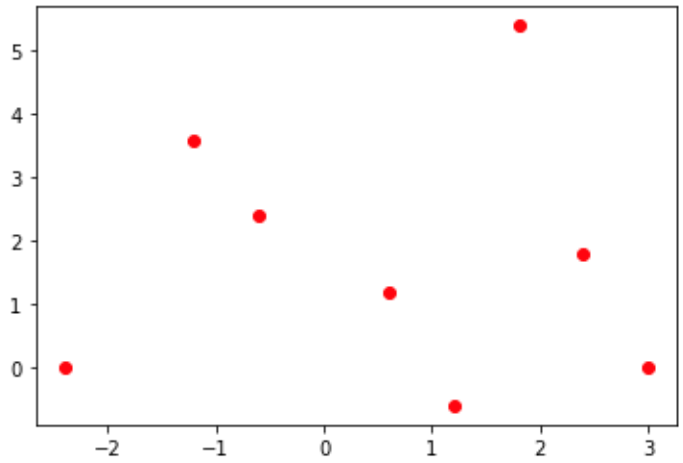

```
ENTER CHOICE OF OPERATION4
```

```
ENTER CHOICE OF OPERATION4
ENTER ANGLE OF ROTATION: 270
```

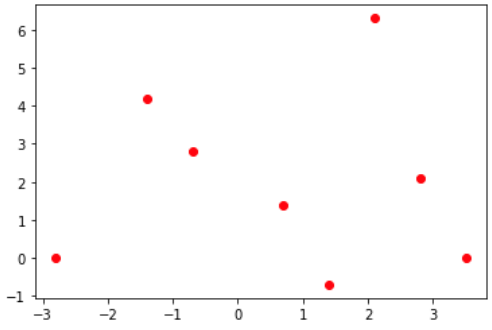



```
ENTER CHOICE OF OPERATION5
```

```
ENTER CHOICE OF OPERATION5
ENTER SCALE POINT LIKE (.5) for 1/2.6
```




```
SELECT OPERATION:
1. ADDITION OF TWO COMPLEX NUMBERS
2. DISPLAY CONJUGATE OF TWO COMPLEX NUMBERS
3. PLOTING SET OF TWO COMPLEX NUMBERS
4. ROTATION
5. SCALING
6. EXIT
ENTER CHOICE OF OPERATION2
ENTER COMPLEX NUMBER32
CONJUGATE OF  (32+0j) IS (32-0j)
ENTER CHOICE OF OPERATION1
ENTER COMPLEX FIRST NUMBER: 2
ENTER COMPLEX SECOND NUMBER: 4
ADDITION OF TWO COMPLEX NUMBERS IS:  (6+0j)
ENTER CHOICE OF OPERATION5
ENTER SCALE POINT LIKE (.5) for 1/2.7
```



Practical 2

Aim: Write a program to do the following:

- Enter a vector u as a n-list.
- Enter another vector v as a n-list.
- Find the vector au+bv for different values of a and b.
- Find the dot product of u and v

Code:

```
def addVec(x,y):
    return [x[i]+y[i] for i in range (len(x))]
def subVec(x,y):
    return [x[i]-y[i] for i in range (len(x))]
def scalarMul(x,p):
    return sum([x[i]*p for i in range (len(x))])
def dotMul(x,y):
    return sum([x[i]*y[i] for i in range (len(x))])
v=[]
u=[]
n=int(input("Enter the no. of Elements you want to enter in vector: "))
print("Enter Elements of vector u: ")
for i in range(n):
    e=int(input("Enter elements: "))
    u.append(e)
print("Vector u = ",u)
print("Enter Elements of vector v: ")
for i in range(n):
    e=int(input("Enter elements: "))
    v.append(e)
print("Vector v = ",v)
while True:
    print("Select Vector Operation: ")
    print("1: Addition")
    print("2: Subtraction")
    print("3: Scalar Multiplication")
```

```
print("4: Dot Product")
print("5: Exit")
ch=int(input("Enter your Choice: "))
if ch==1:
    print("Addition of two matrix are",addVec(u,v))
elif ch==2:
    print("Subtraction of two matrix are",subVec(u,v))
elif ch==3:
    print("To perform Scalar Multiplication")
    a=int(input("Enter the value of a: "))
    b=int(input("Enter the value of b: "))
    ua=scalarMul(u,a)
    vb=scalarMul(v,b)
    print("Addition of au+bv = ",ua+vb)
elif ch==4:
    print("Dot Product of two matrix is: ",dotMul(u,v))
elif ch==5:
    print("Program Terminated Successfully")
    break
```

Output:

```
Enter the no. of Elements you want to enter in vector: 3
Enter Elements of vector u:
Enter elements: 1
Enter elements: 2
Enter elements: 3
Vector u =  [1, 2, 3]
Enter Elements of vector v:
Enter elements: 2
Enter elements: 3
Enter elements: 5
Vector v =  [2, 3, 5]
Select Vector Operation:
1: Addition
2: Subtraction
3: Scalar Multiplication
4: Dot Product
5: Exit
Enter your Choice: 1
Addition of two matrix are [3, 5, 8]
```

```
Vector v =  [2, 3, 5]
Select Vector Operation:
1: Addition
2: Subtraction
3: Scalar Multiplication
4: Dot Product
5: Exit
Enter your Choice: 1
Addition of two matrix are [3, 5, 8]
Select Vector Operation:
1: Addition
2: Subtraction
3: Scalar Multiplication
4: Dot Product
5: Exit
Enter your Choice: 2
Subtraction of two matrix are [-1, -1, -2]
```

```
4: Dot Product
5: Exit
Enter your Choice: 2
Subtraction of two matrix are [-1, -1, -2]
Select Vector Operation:
1: Addition
2: Subtraction
3: Scalar Multiplication
4: Dot Product
5: Exit
Enter your Choice: 3
To perform Scalar Multiplication
Enter the value of a: 4
Enter the value of b: 5
Addition of au+bv =  74
Select Vector Operation:
1: Addition
2: Subtraction
3: Scalar Multiplication
4: Dot Product
5: Exit
Enter your Choice: 5
Program Terminated Successfully
```

Practical 3

Aim: Write a program to do the following:

- Enter two distinct faces as vectors u and v.
- Find a new face as a linear combination of u and v i.e., au+bv for a and b in R.
- Find the average face of the original faces

Code:

```
def lin_comb(vlist,clist):
    s=[scalarmul(vlist[i],clist[i]) for i in range(len(vlist))]
    l=[]
    for j in range(len(s[0])):
        su=0
        for i in range(len(s)):
            su=su+s[i][j]
        l.appned(su)
    return l
l=int(input("ENTER THE LENGTH OF VECTOR: "))
u=[]
v=[]
c=[]
print("ENTER ELEMENTS OF VECTOR U: ")
for i in range(l):
    n=int(input("ENTER NO:"))
    u.append(n)
print("ENTER THE ELEMENTS OF VECTOR V: ")
for i in range(l):
    m=int(input("ENTER NO: "))
    v.append(m)
print("ENTER VALUES FOR COEFFICIENT: ")
c1=int(input("ENTER FIRST COEFFICIENT :"))
c2=int(input("ENTER SECOND COEFFICIENT :"))
```

```
newface=[c1*u[i]+c2*v[i] for i in range(len(u))]
print("NEW FACE OF U and V: ",newface)
avgface=[((u[i]+v[i])/2) for i in range(len(u))]
print("AVERAGE FACE OF U AND V :",avgface)
```

Output:

```
ENTER THE LENGTH OF VECTOR: 3
ENTER ELEMENTS OF VECTOR U:
ENTER NO:1
ENTER NO:2
ENTER NO:3
ENTER THE ELEMENTS OF VECTOR V:
ENTER NO: 2
ENTER NO: 3
ENTER NO: 4
ENTER VALUES FOR COEFFICIENT:
ENTER FIRST COEFFICIENT :4
ENTER SECOND COEFFICIENT :3
NEW FACE OF U and V:  [10, 17, 24]
AVERAGE FACE OF U AND V : [1.5, 2.5, 3.5]
```

Practical 4

Aim: Write a program to do the following:

- Enter an r by c matrix M (r and c being positive integers).
- Display M in matrix format.
- Display the rows and columns of the matrix M Find the scalar multiplication of M for a given scalar.
- Find the transpose of the matrix M.

Code:

```
r= int(input("Enter the number of rows:"))
c = int(input("Enter the number of columns:"))
matrix = []
print("Enter the entries rowwise:")
for i in range(r):
    a =[]
    for j in range(c):
        a.append(int(input()))
        matrix.append(a)
print("Matrix M:")
for i in range(r):
    for j in range(c):
        print(matrix[i][j],end=" ")
    print()
print("Rows of matrix: ")
for i in range(r):
    print("row",i+1," : ")
    for j in range(c):
        rowm=matrix[i][j]
        print(matrix[i][j],end=" ")
print("Columns of matrix: ")
for i in range(c):
    print("Column",i+1," : ")
```

```
    for j in range(r):
        colm=matrix[j][i]
        print(matrix[j][i],end=" ")
    print()
b=[[0 for x in range(r)] for y in range(c)]
for i in range(c):
    for j in range(r):
        b[i][j]=matrix[j][i]
a=int(input("Enter the number to multiply with matrix M:"))
for i in range(r):
    for j in range(c):
        matrix[i][j]=matrix[i][j]*a
print("Scalar Matrix multiplication:")
for i in range(r):
    for j in range(c):
        print(matrix[i][j],end=" ")
print("Transpose of Matrix M: ")
for i in range(c):
    for j in range(r):
        print(b[i][j],end=" ")
    print()
```

Output:

```
Enter the number of rows:3
Enter the number of columns:3
Enter the entries rowwise:
1
1
2
3
4
5
6
7
6
Matrix M:
1 1 2
1 1 2
1 1 2
Rows of matrix:
row 1  :
1 1 2 row 2  :
1 1 2 row 3  :
1 1 2 Columns of matrix:
```

```
7
6
Matrix M:
1 1 2
1 1 2
1 1 2
Rows of matrix:
row 1  :
1 1 2 row 2  :
1 1 2 row 3  :
1 1 2 Columns of matrix:
Column 1  :
1 1 1
Column 2  :
1 1 1
Column 3  :
2 2 2
Enter the number to multiply with matrix M:2
Scalar Matrix multiplication:
8 8 16 8 8 16 8 8 16 Transpose of Matrix M:
1 1 1
1 1 1
2 2 2
```

Practical 5

Aim: Write a program to do the following:

- Find the vector –matrix multiplication of a r by c matrix M with an c-vector u.
- Find the matrix-matrix product of M with a c by p matrix N.

Code:

```
import numpy as np
r =int(input("Enter length of vector and number of rows in matrix:"))
c =int(input("Enter number of columns in matrix:"))
v=[]
M=[]
print("Enter Vectors")
for i in range(r):
    n=int(input("Enter number:"))
    v.append(n)
print("Enter Matrix")
for i in range(r):
    print("Enter Elements of Rows: ",i+1)
    M.append([])
    for j in range(c):
        n=int(input("Enter Number: "))
        M[i].append(n)
print("\nResult of Vector-Matrix multiplication is",np.dot(v,M))
M=[]
N=[]
r1 =int(input("Enter number of rows in matrix1:"))
c1 =int(input("Enter number of columns in matrix1:"))
print("Enter Matrix1")
for i in range(r1):
    print("Enter Elements of Rows: ",i+1)
    M.append([])
```

```
    for j in range(c1):
        n=int(input("Enter Number: "))
        M[i].append(n)
r2 =int(input("Enter number of rows in matrix2:"))
c2 =int(input("Enter number of columns in matrix2:"))
print("Enter Matrix2")
for i in range(r2):
    print("Enter Elements of Rows: ",i+1)
    N.append([])
    for j in range(c2):
        n=int(input("Enter Number: "))
        N[i].append(n)
print(M)
print(N)
if(c1 == r2):
    Mult=np.dot(M,N)
    print("\nResult of Matrix multiplication is")
    for i in Mult:
            print(i)
else:
    print("\n Matrix Multiplication not possible as c1 and r2 are not
equal")
```

Output:

```
Enter length of vector and number of rows in matrix:2
Enter number of columns in matrix:3
Enter Vectors
Enter number:123
Enter number:455
Enter Matrix
Enter Elements of Rows:  1
Enter Number: 67
Enter Number: 123
Enter Number: 45
Enter Elements of Rows:  2
Enter Number: 333
Enter Number: 345
Enter Number: 56
```

```
Result of Vector-Matrix multiplication is [159756 172104  31015]
Enter number of rows in matrix1:2
Enter number of columns in matrix1:4
Enter Matrix1
Enter Elements of Rows:  1
Enter Number: 123
Enter Number: 45
Enter Number: 656
Enter Number: 434
Enter Elements of Rows:  2
Enter Number: 233
Enter Number: 232
Enter Number: 123
Enter Number: 456
Enter number of rows in matrix2:2
Enter number of columns in matrix2:3
```

```
Enter number of columns in matrix1:4
Enter Matrix1
Enter Elements of Rows:  1
Enter Number: 123
Enter Number: 45
Enter Number: 656
Enter Number: 434
Enter Elements of Rows:  2
Enter Number: 233
Enter Number: 232
Enter Number: 123
Enter Number: 456
Enter number of rows in matrix2:2
Enter number of columns in matrix2:3
Enter Matrix2
Enter Elements of Rows:  1
Enter Number: 123
Enter Number: 456
Enter Number: 543
Enter Elements of Rows:  2
Enter Number: 213
Enter Number: 435
Enter Number: 344
[[123, 45, 656, 434], [233, 232, 123, 456]]
[[123, 456, 543], [213, 435, 344]]

 Matrix Multiplication not possible as c1 and r2 are not equal
```

Practical 6

Aim: Write a program to enter a matrix and check if it is invertible. If the inverse exists, find the inverse.

Code:

```
import numpy as np
def printMatrix(A):
    print("\nMatrix")
    for i in range(len(A)):
        print(A[i])
c=int(input("Enter the number of rows and coloumns of square matrix:
"))
r=c
M=[]
for i in range(r):
    print("Enter Elements of Rows: ")
    M.append([])
    for j in range(c):
        n=int(input("Enter Number: "))
        M[i].append(n)
printMatrix(M)

a=np.linalg.det(M)
print("Determinant of Matrix M is",a)
if a<=0:
    Minv=np.linalg.inv(M)
    print("The Inverse of Matrix m is\n",Minv)
else:
    print("Matrix is not invertible")
```

Output:

```
Enter the number of rows and coloumns of square matrix: 3
Enter Elements of Rows:
Enter Number: 12
Enter Number: 23
Enter Number: 34
Enter Elements of Rows:
Enter Number: 12
Enter Number: 34
Enter Number: 12
Enter Elements of Rows:
Enter Number: 12
Enter Number: 21
Enter Number: 23

Matrix
[12, 23, 34]
[12, 34, 12]
[12, 21, 23]
Determinant of Matrix M is -1980.0000000000002
The Inverse of Matrix m is
 [[-0.26767677 -0.09343434  0.44444444]
 [ 0.06666667  0.06666667 -0.13333333]
 [ 0.07878788 -0.01212121 -0.06666667]]
```

Practical 7

Aim: Write a program to convert a matrix into its row echelon form

Code:

```
def row_echelon(m):
    if not m:
        return
    lead= 0
    rowCount = len(m)
    colCount =len(m[0])
    for r in range(rowCount):
        if lead>= colCount:
            return
        i = r
        while m[i][lead] == 0:
            i+=1
            if i == rowCount:
                i = r
                lead +=1
                if colCount == lead:
                    return
                m[i],m[r] = m[r],m[i]
                v = m[r][lead]
                m[r] = [mrx/float(lv) for mrx in m[r]]
                for i in range(rowCount):
                    if i != r:
                        lv=m[i][lead]
                        m[i] = [iv - lv*rv for rv,iv in zip(m[r],m[i])]
                        lead+=1
print("Enter dimensions of matrix: ")
r=int(input("Enter the no of rows: "))
c=int(input("Enter the no of cols: "))
mtr=[]
```

```
for i in range (r):
    mtr.append([])
    print("Enter elements of rows ",i)
    for j in range(c):
        n=float(input("Enter element: "))
        mtr[i].append(n)
print("Enter the matrix: ")
for row in mtr:
    print(' '.join(("{0:.2f}".format(x) for x in row)))
    row_echelon(mtr)
    print("Row Echelon for matrix: ")
    for row in mtr:
        print(' '.join(("{0:.2f}".format(x) for x in row)))
```

Output:

```
Enter dimensions of matrix:
Enter the no of rows: 3
Enter the no of cols: 4
Enter elements of rows  0
Enter element: 12
Enter element: 32
Enter element: 34
Enter element: 43
Enter elements of rows  1
Enter element: 90
Enter element: 87
Enter element: 56
Enter element: 78
Enter elements of rows  2
Enter element: 34
Enter element: 53
Enter element: 21
Enter element: 34
Enter the matrix:
```

```
Enter elements of rows  2
Enter element: 34
Enter element: 53
Enter element: 21
Enter element: 34
Enter the matrix:
12.00 32.00 34.00 43.00
Row Echelon for matrix:
12.00 32.00 34.00 43.00
90.00 87.00 56.00 78.00
34.00 53.00 21.00 34.00
90.00 87.00 56.00 78.00
Row Echelon for matrix:
12.00 32.00 34.00 43.00
90.00 87.00 56.00 78.00
34.00 53.00 21.00 34.00
34.00 53.00 21.00 34.00
Row Echelon for matrix:
12.00 32.00 34.00 43.00
90.00 87.00 56.00 78.00
34.00 53.00 21.00 34.00
```

Practical 8

Aim: Write a program to do the Following:

- Find the GCD.
- Find Prime Factor of a number.

Code:

```
from math import *
def gcd(a,b):
    if a==0:
        return b
    return gcd(b%a,a)
pf=[]
n=int(input("Enter number:"))
x=n
while n%2==0:
    pf.append(2)
    n=n/2
i=3
while i<=sqrt(n):
    while n%1==0:
        pf.append(i)
        n=n/i
        i=i+2
if n>2:
    pf.append(n)
print("prime factor of",x,"are",pf)
pf1=set(pf)
nf=1
for f in pf1:
    cnt=0
    for f1 in pf:
        if f==f1:
```

```
            cnt+=1
            nf*=cnt+1
print("No of factors of ",x," = ",nf)
print("No of positive integral solutions = ",nf/2)
print("Greatest Common Divisor of 45 and 15 = ",gcd(15,45))
```

Output:

```
Enter number:42
prime factor of 42 are [2, 3, 5]
No of factors of  42  =  8
No of positive integral solutions =  4.0
Greatest Common Divisor of 45 and 15 =  15
```