**PRACTICAL NO: 01**

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

print('Select operation')

print('1: Addition of two 2 complex number ')

print('2: Displaying the conjugate of a complex number. ')

print('3: pltting a set of complex numbers. ')

print('4: Creating a new plt by rotating the given number ')

print('5: Creating a new plt by Scalling ')

print('6: Exit')

while True:

ch=int(input("Enter choice for operation "))

if ch==1:

c1=complex(input("Enter the First Complex number "))

c2=complex(input("Enter the Second Complex number "))

print("Addition of Complex number is ",(c1+c2))

elif ch==2:

c=complex(input("Enter the Complex number "))

print("The Conjugate of Complex number is ",c.conjugate())

elif ch==3:

p1=complex(input("Enter the Starting point(Complex number) " ))

mylist=[]

p2=int(input("Enter how many points you want to draw "))

for x in range(p2):

mylist.append(complex(input("Enter the graph points ")))

p1=[p1.real for p1 in range(p2)]

y=[p1.imag for p1 in range(p2)]

plt.scatter(p1,y,color='red')

plt.show()

elif ch==4:

c=complex(input("Enter Complex number "))

angle=int(input("Enter angle of rotation 90/180/270 " ))

if angle==90:

z=1j

plt.scatter(c.real,c.imag,color='red')

y=c\*z

plt.scatter(y.real,y.imag)

plt.show()

elif angle==180:

plt.scatter(c.real,c.imag,color='red')

plt.scatter(-1\*c.real,-1\*c.imag)

plt.show()

elif angle==270:

z=-1j

plt.scatter(c.real,c.imag,color='red')

y=c\*z

plt.scatter(y.real,y.imag)

plt.show()

else:

print("Invalid Input")

elif ch==5:

c=complex(input("Enter Complex number "))

scale=0.5

scale1=0.33

scale2=2

plt.scatter(c.real,c.imag,color='red')

c=scale\*c

d=scale1\*c

e=scale2\*c

plt.scatter(c.real,c.imag,color='blue')

plt.scatter(d.real,d.imag,color='green')

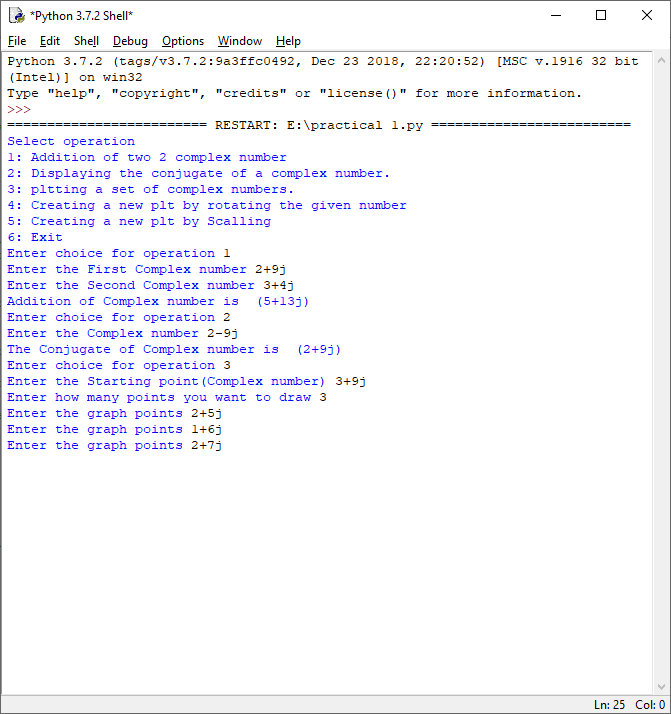
plt.scatter(e.real,e.imag,color='black')

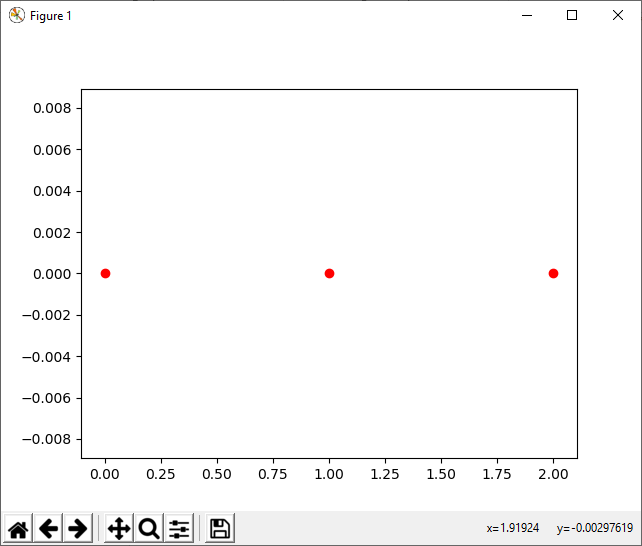
plt.show()

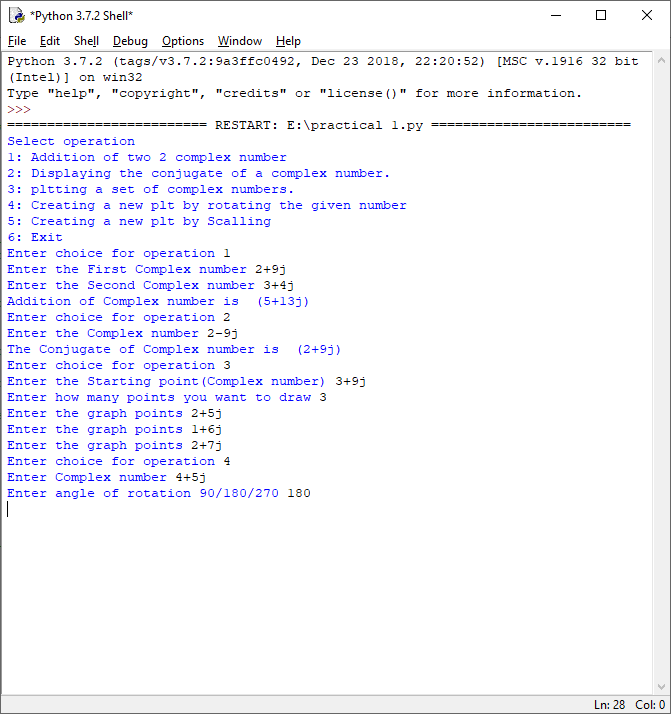
elif ch==6:

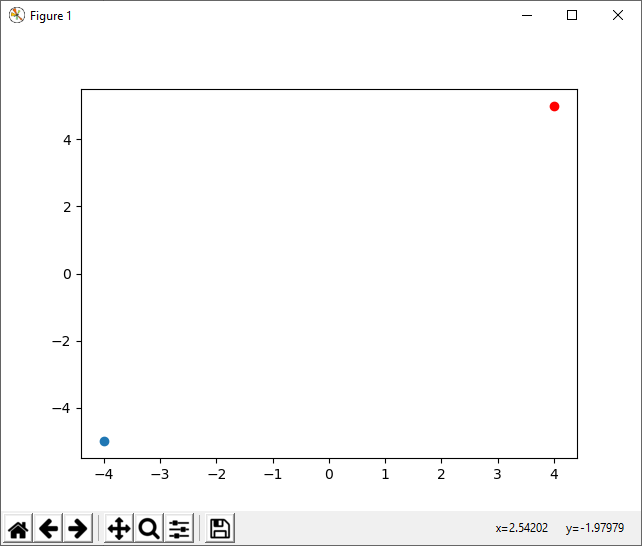
break

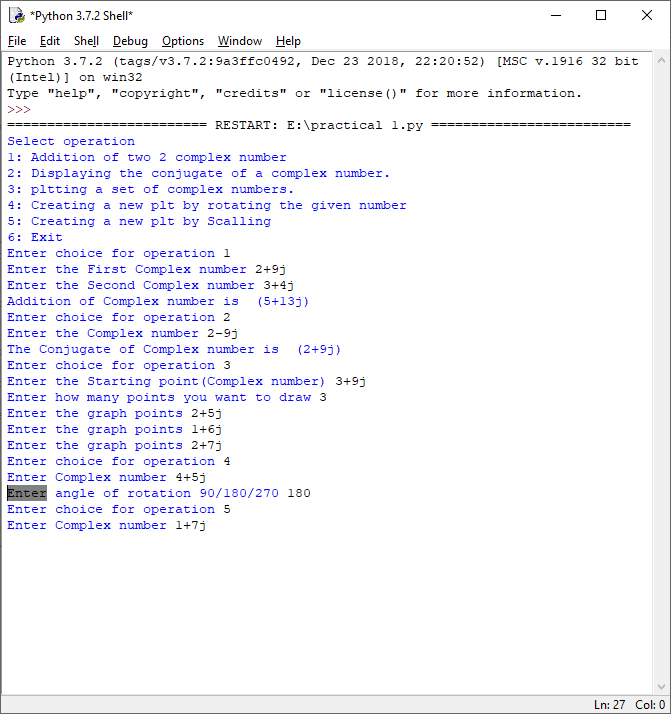
OUTPUT:

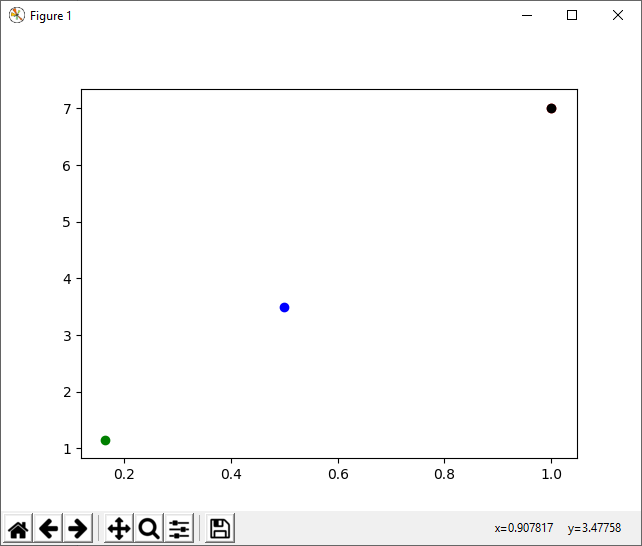












**PRACTICAL NO : 02**

Write a program to do the following:

1) Enter a vector u as a n-list

2) Enter another vector v as a n-list

3) Find the vector au+bv for different values of a and b

4) Find the dot product of u and v

CODE:

import numpy as np

print('select vector operation')

print('1:Enter a vector u as a n-list')

print('2:Enter a vector v as a n-list. ')

print('3:Find the vector au +bv for different values of a and b. ')

print('4:Dot Product')

print('5:Exit')

while True:

ch=int(input('Enter Your choice'))

if ch==1:

u=[]

n=int(input('enter no of elements you want to add in vector:'))

print('enter elements of vector u')

for i in range(n):

elem=int(input('enter element'))

u.append(elem)

print('Vector u=',u)

elif ch==2:

v=[]

n=int(input('enter no of elements you want to add in vector:'))

print('enter elements of vector v')

for i in range(n):

elem=int(input('enter element'))

v.append(elem)

print('Vector v=',v)

elif ch==3:

v=[]

u=[]

n=int(input('enter no of elements you want to add in vector:'))

print('enter elements of vector u')

for i in range(n):

elem=int(input('enter element'))

u.append(elem)

x=np.array(u)

print('Vector u=',x)

print('enter elements of vector v')

for i in range(n):

elem=int(input('enter element'))

v.append(elem)

y=np.array(v)

print('Vector v=',y)

a=int(input("Enter Value of a "))

b=int(input("Enter Value of b "))

print("The Value of au+bv is ",(a\*x+b\*y))

elif ch==4:

v=[]

u=[]

n=int(input('enter no of elements you want to add in vector:'))

print('enter elements of vector u')

for i in range(n):

elem=int(input('enter element'))

u.append(elem)

print('Vector u=',u)

print('enter elements of vector v')

for i in range(n):

elem=int(input('enter element'))

v.append(elem)

print('Vector v=',v)

print("The Dot product of u and v are ",np.dot(u,v))

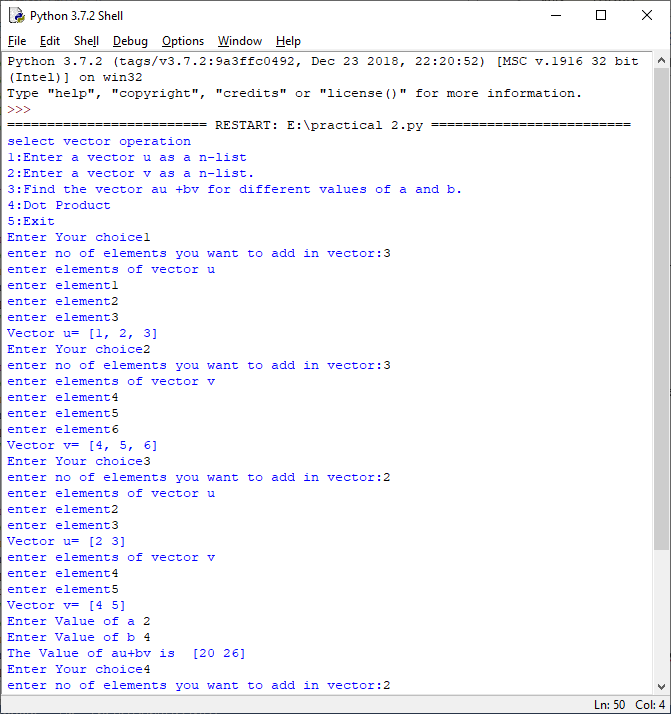
elif ch==5:

break

else:

print("Invalid Input")

OUTPUT:





**PRACTICAL NO: 03**

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:

from PIL import Image

import numpy as np

a=Image.open('C:/Users/jamil/Downloads/image1.jpg')

b=Image.open('C:/Users/jamil/Downloads/image2.jpg')

i1=np.array(a)

i2=np.array(b)

def imresize(im,sz):

"""Raise an image using pil"""

pil\_im=image.fromarray(uint8(im))

return array(pil\_im.resize(sz))

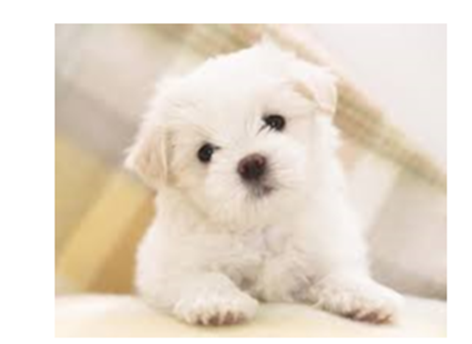
a1=Image.fromarray(i1)

b1=Image.fromarray(i2)

a1.show()

b1.show()

Output:



Q2.find a new face as a linear combination of u and v i.e . au+bv for a and b in R

from PIL import Image

import numpy as np

def imgresize(img,sz):

pil\_img=Image.fromarray((img))

return np.array(pil\_img.resize(sz))

img1=Image.open('C:/Users/jamil/Downloads/image1.jpg')

img2=Image.open('C:/Users/jamil/Downloads/image2.jpg')

i1=np.array(img1)

i2=np.array(img2)

u=imgresize(i1,(200,200))

v=imgresize(i2,(200,200))

a=1

b=2

lincomb=a\*u+b\*v

linimg=Image.fromarray(lincomb)

linimg.show()

output:



3)Find the average face of the original faces.

from PIL import Image

import numpy as np

a=Image.open('C:/Users/jamil/Downloads/image1.jpg')

b=Image.open('C:/Users/jamil/Downloads/image2.jpg')

i1=np.array(a)

i2=np.array(b)

def imresize(im,sz):

"""Resize an image using pil"""

pil\_im=Image.fromarray(im)

return np.array(pil\_im.resize(sz))

a1=imresize(i1,(200,200))

b1=imresize(i2,(200,200))

r=a1+b1

avgImage=r/2

avg=Image.fromarray(avgImage.astype('uint8'))

avg.show()

output:



**PRACTICAL NO: 04**

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:

import numpy as np

def printmatrix(A):

print("the entered matrix M is:")

for i in range(r):

print(A[i])

#Display rows of matrix

def printrows(A):

print("Rows of matrix:")

for i in range(r):

print("Row %d="%i,A[i])

#display columns of matrix

def printcolumns(A):

for j in range(c):

print("columns %d =\n"%j,end="")

for i in range(r):

print(a[i][j],end="")

print("\n")

#scalar multiplication

def scalarmul(A,s):

N=[[s\*A[i][j] for j in range(c)] for i in range(r)]

print("the scalar multiplication of matrix s\*M")

printmatrix(N)

#transpose of matrix

def transpose(A):

T=[[A[i][j] for i in range(r)]for j in range(c)]

print("Transpose of M= ")

for j in range(c):

print(T[j])

def printvector(A,M):

l=np.dot(A,M)

print(l)

def printvectormat(A,v):

vm=np.dot(v,A)

print(vm)

#Enter the r\*c

print("enter the dimension of ")

r= int(input("enter no of rows "))

c=int(input("enter no of columns:"))

M=[]

for i in range(r):

print("enter the element of row",i)

M.append([])

for j in range(c):

n=int(input("enter no "))

M[i].append(n)

print("select operation ")

print("1:Display matrix ")

print("2:rows of matrix ")

print("3:columns of matrix ")

print("4:scalar mul")

print("5:transpose of matrix ")

print("6:Matrix-Vector Multiplication")

print("7:vector-Matrix multiplication :")

print("8:exit")

while True:

ch=int(input("enter choice of operation"))

if ch==1:

printmatrix(M)

elif ch==2:

printrows(M)

elif ch==3:

printcolumns(M)

elif ch==4:

sc=int(input("enter scalar value: "))

scalarmul(M,sc)

elif ch==5:

transpose(M)

elif ch==6:

v=[]

for j in range(c):

n=int(input("enter number for :"))

v.append(n)

printvector(M,v)

elif ch==7:

v=[]

for j in range(r):

n=int(input("enter number for :"))

v.append(n)

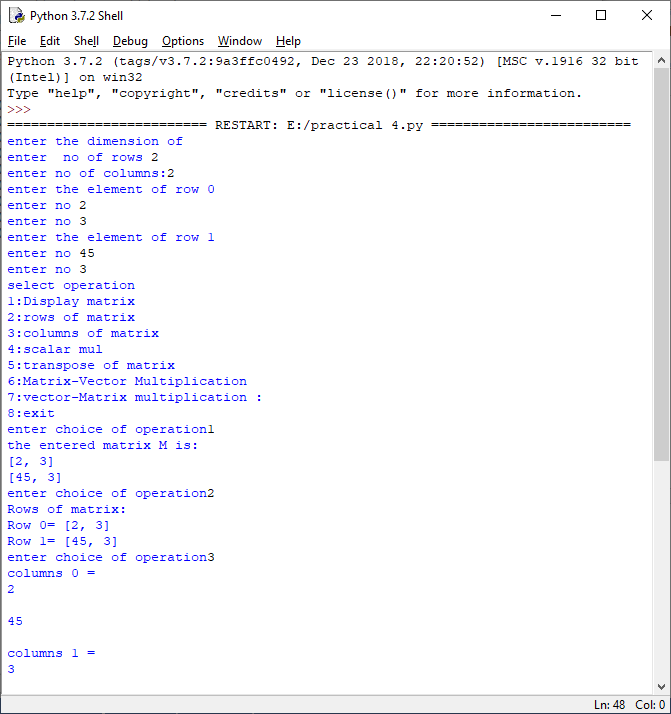
printvectormat(M,v)

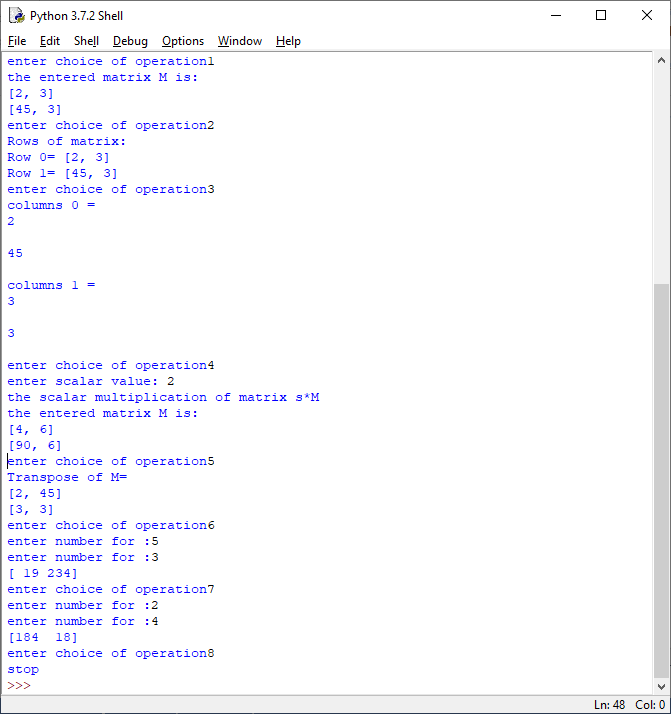
else:

print("stop")

break

OUTPUT:





**PRACTICAL NO : 05**

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

m=np.array([[2,0,1],[5,3,4]])

v=[1,4,3]

print(np.dot(m,v))

print('--------------------------------------')

print('enter the demision')

r= int(input('enter number of rows:'))

c=int(input('enter number of columns:'))

M=[]

for i in range(r):

p=[]

for j in range(c):

n=int(input('enter number:'))

p.append(n)

M.append(p)

print(np.array(M))

v=[]

for i in range(c):

a=int(input('enter the number for vector:'))

v.append(a)

print("Select option\n1.Column vector\n2.Row vector")

while True:

ch=int(input("enter choice"))

if ch==1:

print(np.dot(M,v))

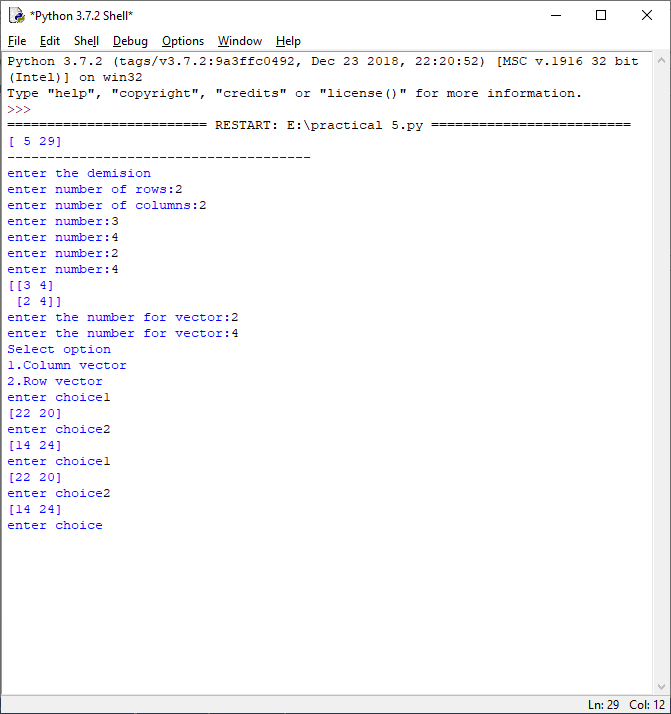
elif ch==2:

print(np.dot(v,M))

elif ch==3:

break;

OUTPUT:



**PRACTICAL NO: 06**

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

1ST method:

CODE:

from numpy.linalg import inv

from numpy.linalg import det

import numpy as np

try:

print("Enter the Dimension ")

r=int(input('Enter no of rows: '))

c=int(input('Enter no of columns: '))

M=[]

for i in range(r):

print('Enter elements of row ',i)

M.append([])

for j in range(c):

n=int(input())

M[i].append(n)

print(M)

print("Determinant is: ",det(M))

if(det(M)!=0):

print("the Determinant of Matrix Exist ")

print("The inverse is: ",inv(M))

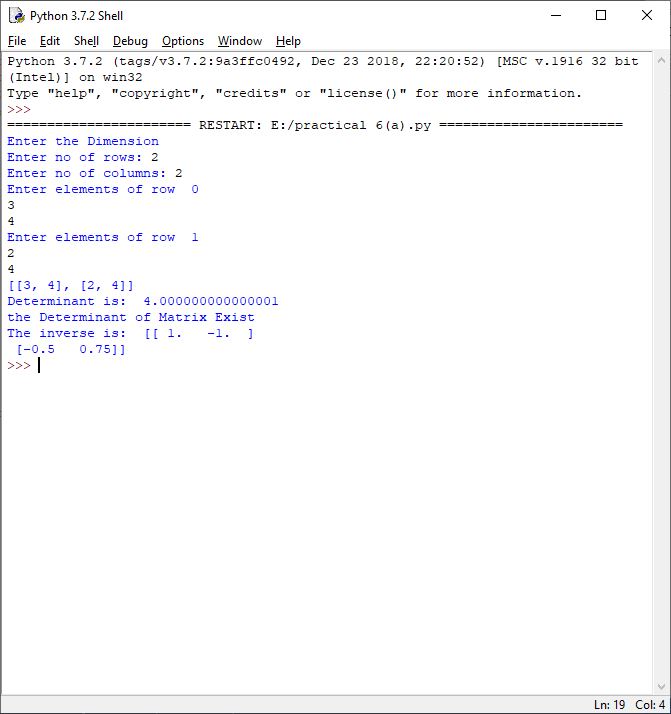
else:

print("not invertible ")

except(Exception):

print("Matrix should be a Square Matrix ")

OUTPUT:



Matrix inverse by using Co-Factors

Code:

def printmatrix(A):

for i in range(len(A)):

print(A[i])

def transpose(A):

T=[[A[i][j] for i in range(len(A))]]

return T

c=int(input("Enter the no of rows and columns of square matrix: "))

r=c

M=[]

for i in range(r):

print("Enter elements of row: ",i)

M.append([])

for j in range(c):

n=int(input("Enter no: "))

M[i].append(n)

print("The entered matrix is: ")

printmatrix(M)

determinant=0

if r==2:

determinant=M[0][0]\*M[1][1]-M[0][1]\*M[1][0]

print("Det= ",determinant)

if determinant==0:

print("Matrix is not invertible ")

else:

print("Matrix is invertible ")

#calculating matrix of cofactors by finding minormatrix

CFM=[]

for i in range(2):

CFM.append([])

CFM[0].append(M[1][1])

CFM[0].append(-M[0][1])

CFM[1].append(-M[1][0])

CFM[1].append(M[0][0])

print("Cofactor Matrix: ")

print(CFM)

MI=[]

for i in range(2):

MI.append([])

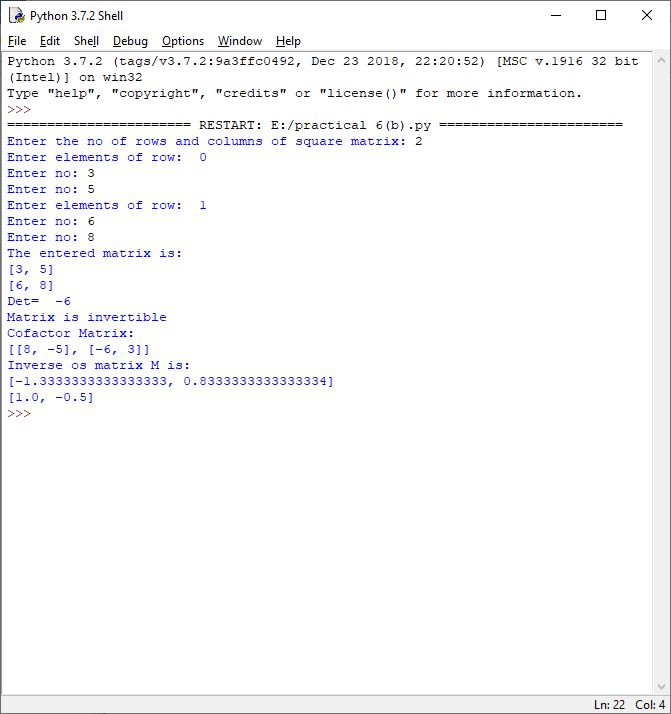
for j in range(2):

MI[i].append(CFM[i][j]/determinant)

print("Inverse os matrix M is: ")

printmatrix(MI)

OUTPUT:



**PRACTICAL NO: 07**

Write a program to convert a matrix into its row echelon form.

CODE:

def RE(M):

if not M:return

lead=0

rowcount=len(M)

columncount=len(M[0])

for r in range(rowcount):

if lead>=columncount:

return

i=r

while M[i][lead]==0:

i+=1

if i==rowcount:

i=r

lead+=1

if columncount==lead:

return

M[i],M[r]=M[r],M[i]

lv=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

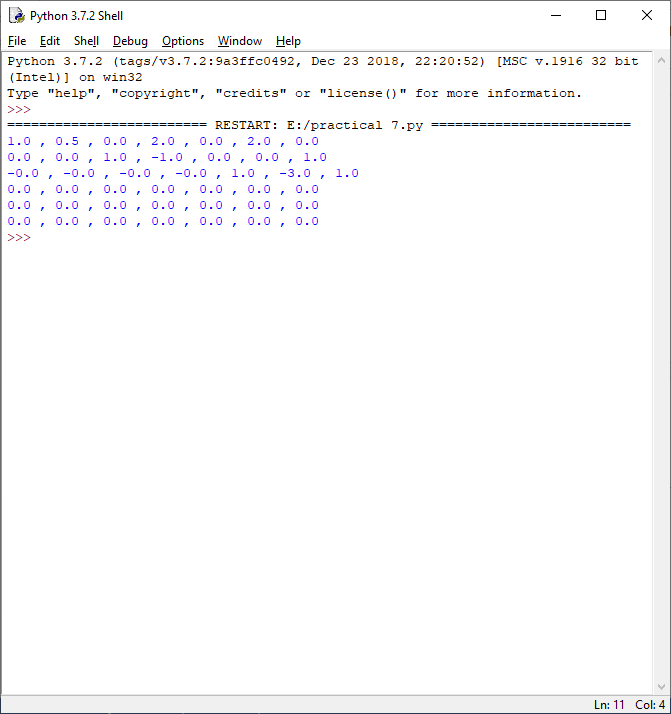
mtx=[[2,1,1,3,0,4,1],[4,2,4,4,1,5,5],[2,1,3,1,0,4,3],[6,3,4,8,1,9,5],[0,0,3,-3,0,0,3],[8,4,2,14,1,13,3]]

RE(mtx)

for rw in mtx:

print(' , '.join(str(rv) for rv in rw))

OUTPUT:



**PRACTICAL NO : 08**

CODE:

""" Write a program find the gcd of two numbers using euclids algorithm

"""

def gcd(a,b):

if(b>a):

if(b%a==0):

return a

else:

return gcd(b%a,a)

else:

if(b%a==0):

return b

else:

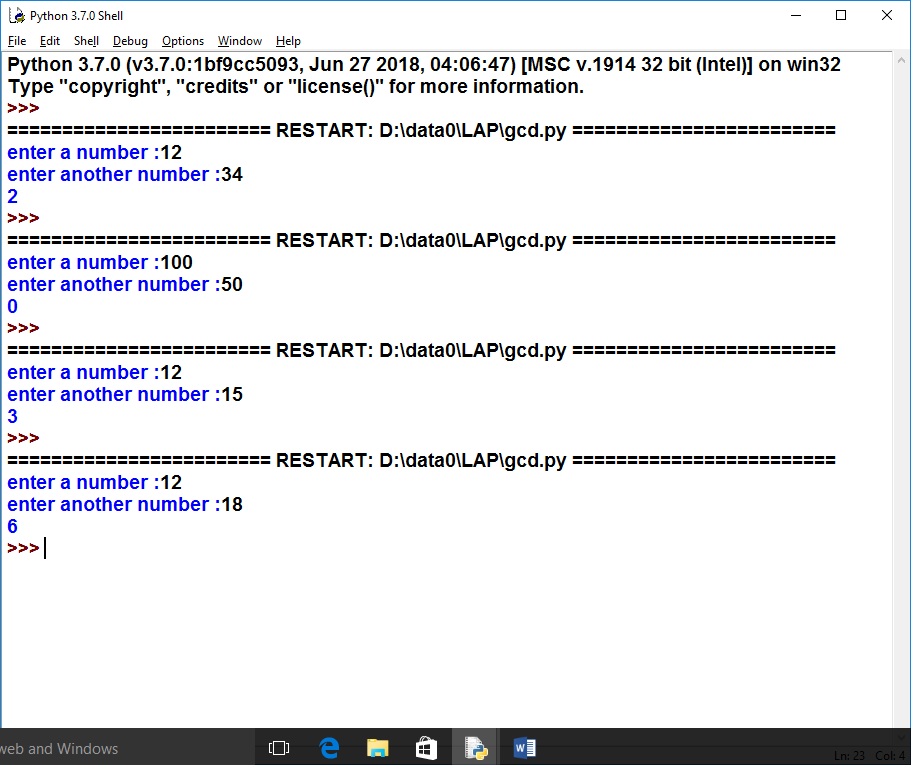
return gcd(b,a%b)

a=int(input("enter a number :"))

b=int(input("enter another number :"))

print (gcd(a,b))

OUTPUT:



b)

#enter a positive number N, and find numbers a &b such that a\*\*2 -b\*\*2=N

import random

n=int(input("Enter no "))

for i in range(0,n):

for j in range(0,n):

if (i\*i-j\*j==n):

print("The value of i and j are",i,j)

OUTPUT:



**PRACTICAL NO : 10**

CODE:

# Write a program to enter a given matrix and an eigen value of the same.Find its eigen vector

import numpy as np

A=np.mat("3 -2 ; 1 0")

print("A",A)

print("eign values",np.linalg.eigvals(A))

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

print ("first tuple of eig ",eigenvalues)

print ("second tuple of eig ",eigenvectors)

for i in range(len(eigenvalues)):

print ("left",np.dot(A,eigenvectors[:,i]))

print ("---------------")

OUTPUT:



**PRACTICAL NO: 09**

Enter a vector b and find the projection of b orthongonal to a given vector u

CODE:

import numpy as np

b=[]

v=[]

i=int(input("enter number of element in vector:"))

print("enter elements for vector b:")

for j in range(i):

c=int(input())

b.append(c)

print("enter elements for vector v:")

for j in range(i):

c=int(input())

v.append(c)

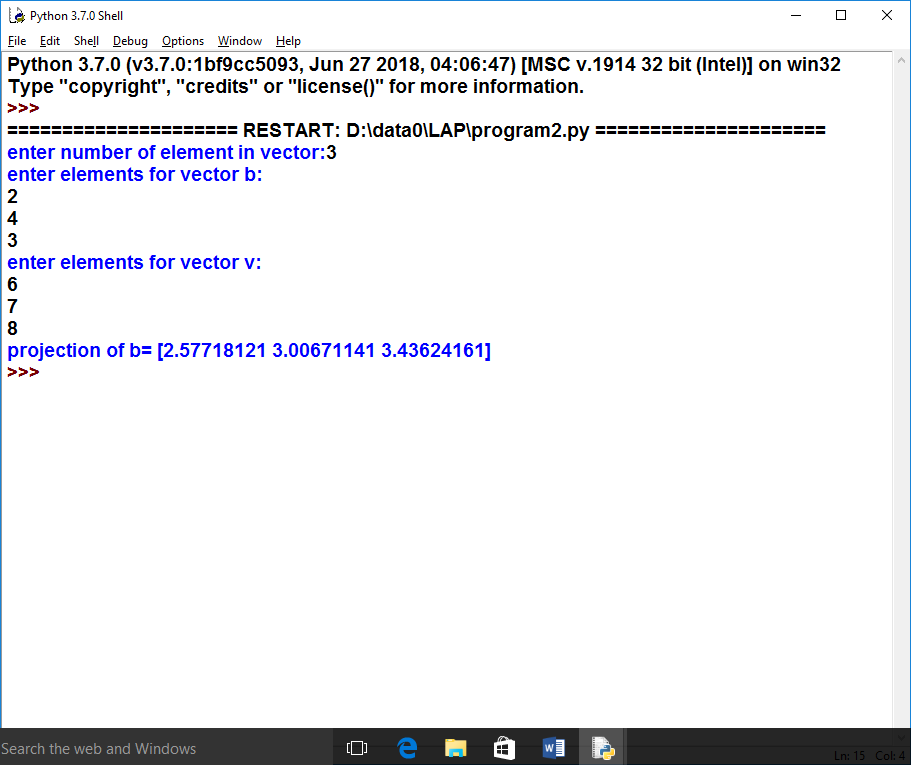
c=np.dot(b,v)

d=np.dot(v,v)

B=np.dot((c/d),v)

print("projection of b=",B)

OUTPUT:



**FIND PRIME FACTOR OF THE GIVEN INTEGER**

**CODE:**

n=int(input("Enter an integer:"))

print("Factors are:")

i=1

while(i<=n):

k=0

if(n%i==0):

j=1

while(j<=i):

if(i%j==0):

k=k+1

j=j+1

if(k==2):

print(i)

i=i+1

OUTPUT:

