Practical 1 Of Linear Algebra

Roll\_no:6806

1. WAP to do the following (Menu driven program ) :

a. Addition of complex numbers.

b. Display conjugate of complex numbers.

c. Plotting a set of complex numbers.

d. Translating a complex number.

e. Creating a new plot by rotating a given number by 90,180,270 degrees.

f. Creating a new plot by scaling a given number a=1/2,a=2 etc.

Code:

import cmath

import matplotlib.pyplot as plt

import numpy as np

import matplotlib

from matplotlib import pyplot,transforms

print("Enter your complex number:")

x=int(input("Enter real part of the complex number:"))

y=int(input("Enter imaginary part of the complex number:"))

comp1=complex(x,y)

while True:

print("----Practical 1 Linear Algebra----")

print("----Choose from the following Options----")

print("----1.Addition of Complex numbers.----")

print("----2.Display conjugate of complex number.----")

print("----3.Plotting a set of complex numbers.----")

print("----4.Translating a complex number.----")

print("----5.Creating a new plot by rotating a given number by 90,180,270 degrees.----")

print("----6.Creating a new plot by scaling a given number a=1/2,a=2 etc.----")

print("----7. Exit.----")

choice = int(input("Enter your choice : "))

if choice == 1:

print("Enter the second complex number:")

a= int (input("Enter the real part of the complex number: "))

b= int (input("Enter the imaginary part of the complex number: "))

comp2=complex(a,b)

print("Addition of ",comp1,"and",comp2,"is",comp1+comp2)

if choice == 2:

print("Conjugate of the given complex number is: ",comp1.conjugate())

if choice == 3:

print("Enter the second complex number:")

e = int (input("Enter the real part of the complex number: "))

f = int (input("Enter the imaginary part of the complex number: "))

comp3 = complex(e,f)

plt.scatter(comp1.real,comp1.imag,color="red")

plt.scatter(comp3.real,comp3.imag,color="red")

plt.show()

if choice == 4:

print("Enter Transalting factor:")

g = int (input("Enter the real part of the complex number: "))

h = int (input("Enter the imaginary part of the complex number: "))

comp4 = complex(g,h)

cadd = comp1 + comp4

plt.scatter(comp1.real,comp1.imag,color="red")

plt.scatter(comp4.real,comp4.imag,color="green")

plt.show()

if choice == 5:

rot= int(input("Enter degrees to which the number is to be rotated:"))

nin = comp1 \* complex(0,1)

oneeig = comp1 \* -1

twoseven = comp1 \* complex(0,-1)

if rot == 90:

plt.scatter(comp1.real,comp1.imag,color="red")

plt.scatter(nin.real,nin.imag,color="green")

plt.show()

if rot == 180:

plt.scatter(comp1.real,comp1.imag,color="red")

plt.scatter(oneeig.real,oneeig.imag,color="green")

plt.show()

if rot == 270:

plt.scatter(comp1.real,comp1.imag,color="red")

plt.scatter(twoseven.real,twoseven.imag,color="green")

plt.show()

if choice == 6:

c = comp1 \* 0.5

d = comp1 \* 2

plt.scatter(c.real,c.imag,color="green")

plt.scatter(d.real,d.imag,color="red")

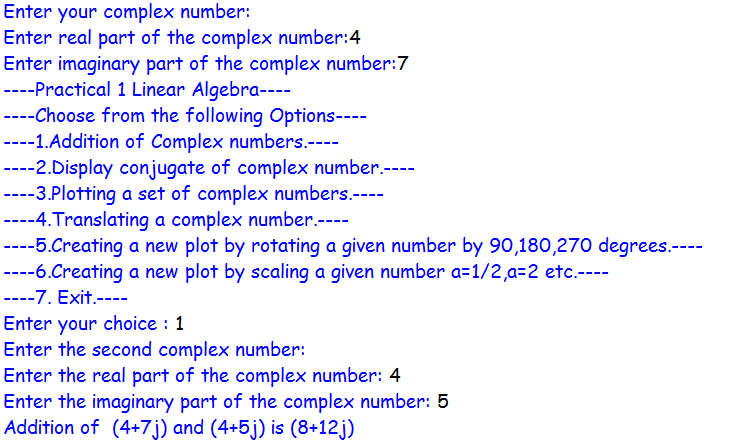
plt.show()

if choice == 7:

exit()

Output:

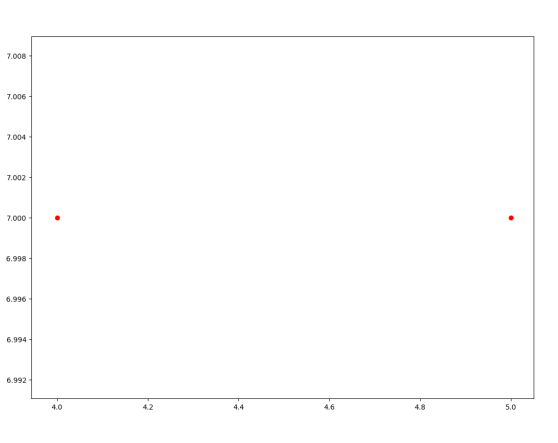
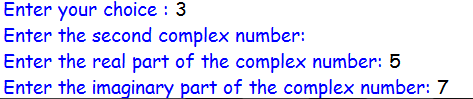
a.



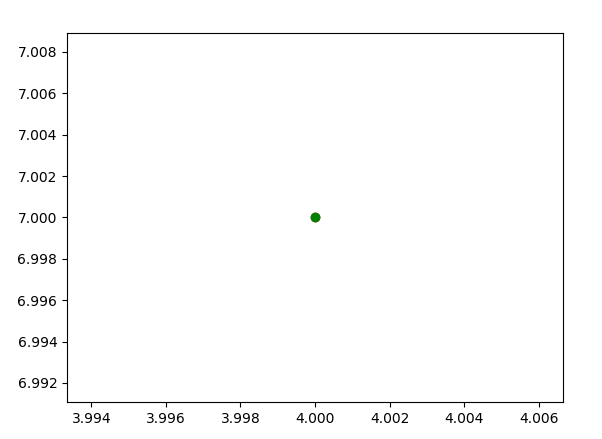
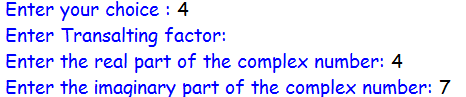
b.



c.

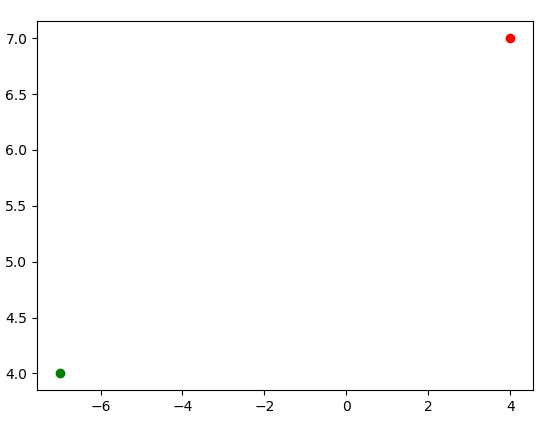
 

d.

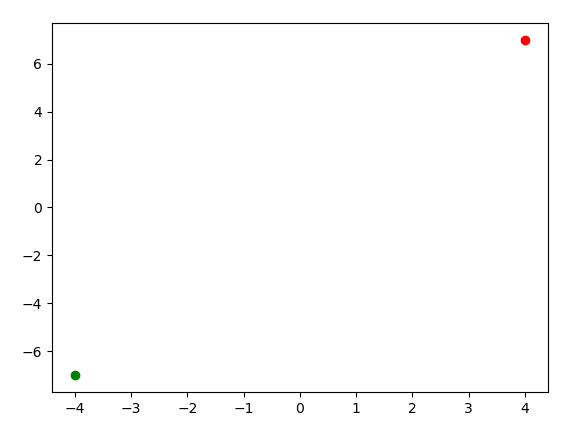
 

e.

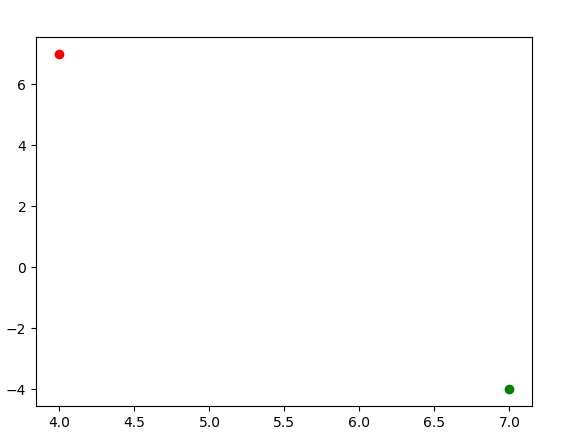
90 degrees



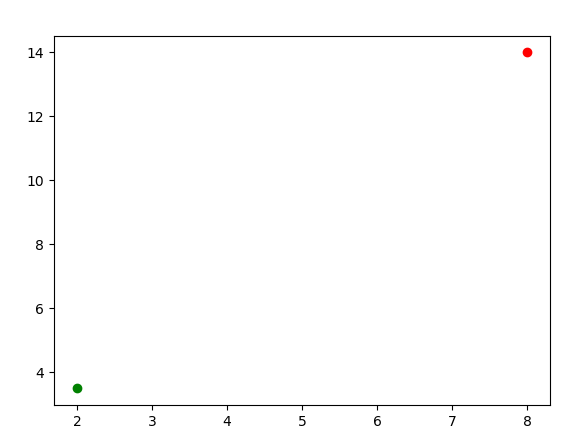
180 degrees



270 degrees



f.



LA Pract 2

1. write a program to do following

Enter a vector u as n list

Enter another vector v as n list

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

Find the dot product of u and v

Code:

def scalar(x,p,y,z):

return[p\*x[i]+z\*y[i]for i in range(len(x))]

u=[]

v=[]

n=int(input('enter no of elements for vector:'))

print('enter elements of vector u :')

for i in range(n):

values = int(input('enter element:'))

u.append(values)

print('enter elements of vector v:')

for i in range(n):

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

v.append(elem)

print("Vector u",u)

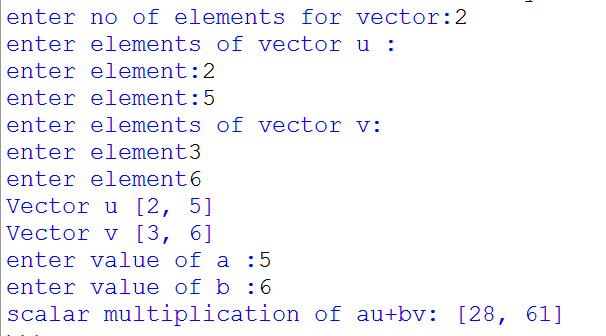
print("Vector v",v)

a=int(input('enter value of a :'))

b=int(input('enter value of b :'))

print('scalar multiplication of au+bv:',scalar(u,a,v,b))

Output:



DotProduct:

Code:

a=[]

b=[]

c=[]

def dot(x,y):

return ([x[i]\*y[i] for i in range(len(x))])

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

print("Values for a")

for i in range(n):

values = int(input('enter element:'))

a.append(values)

print(a)

print("Values for b")

for i in range(n):

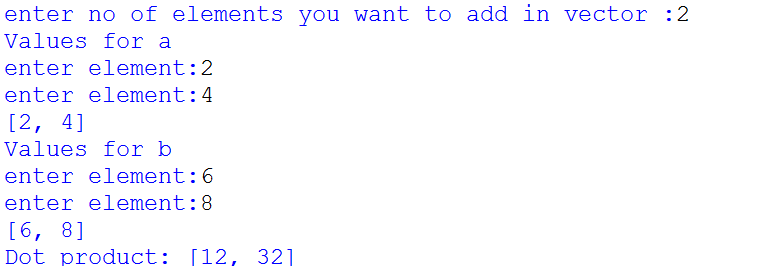
elem=int(input('enter element:'))

b.append(elem)

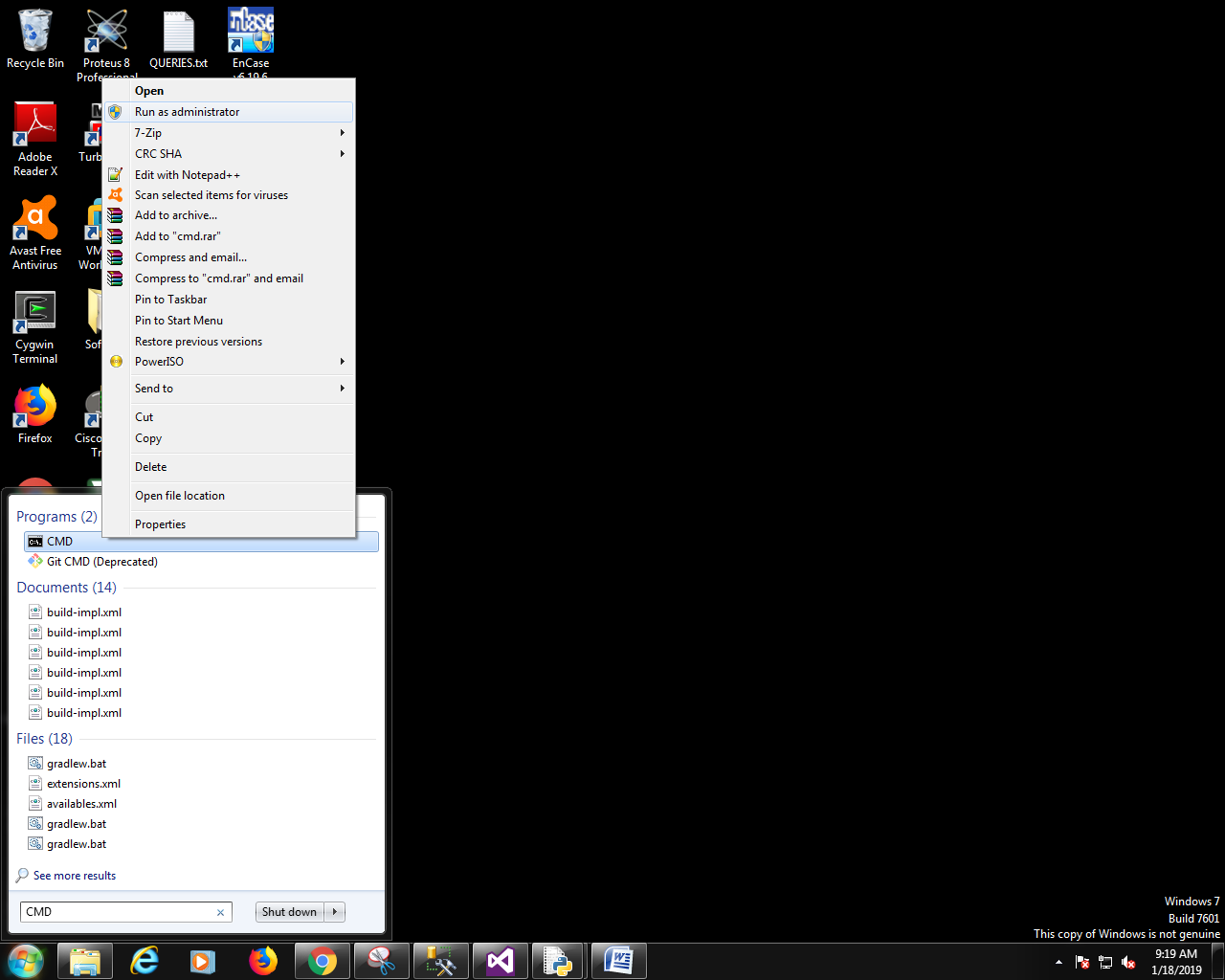
print(b)

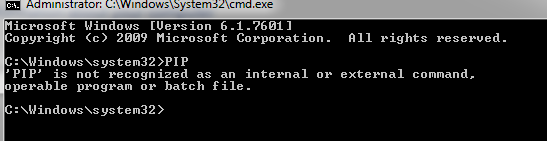
print('Dot product:',dot(a,b))

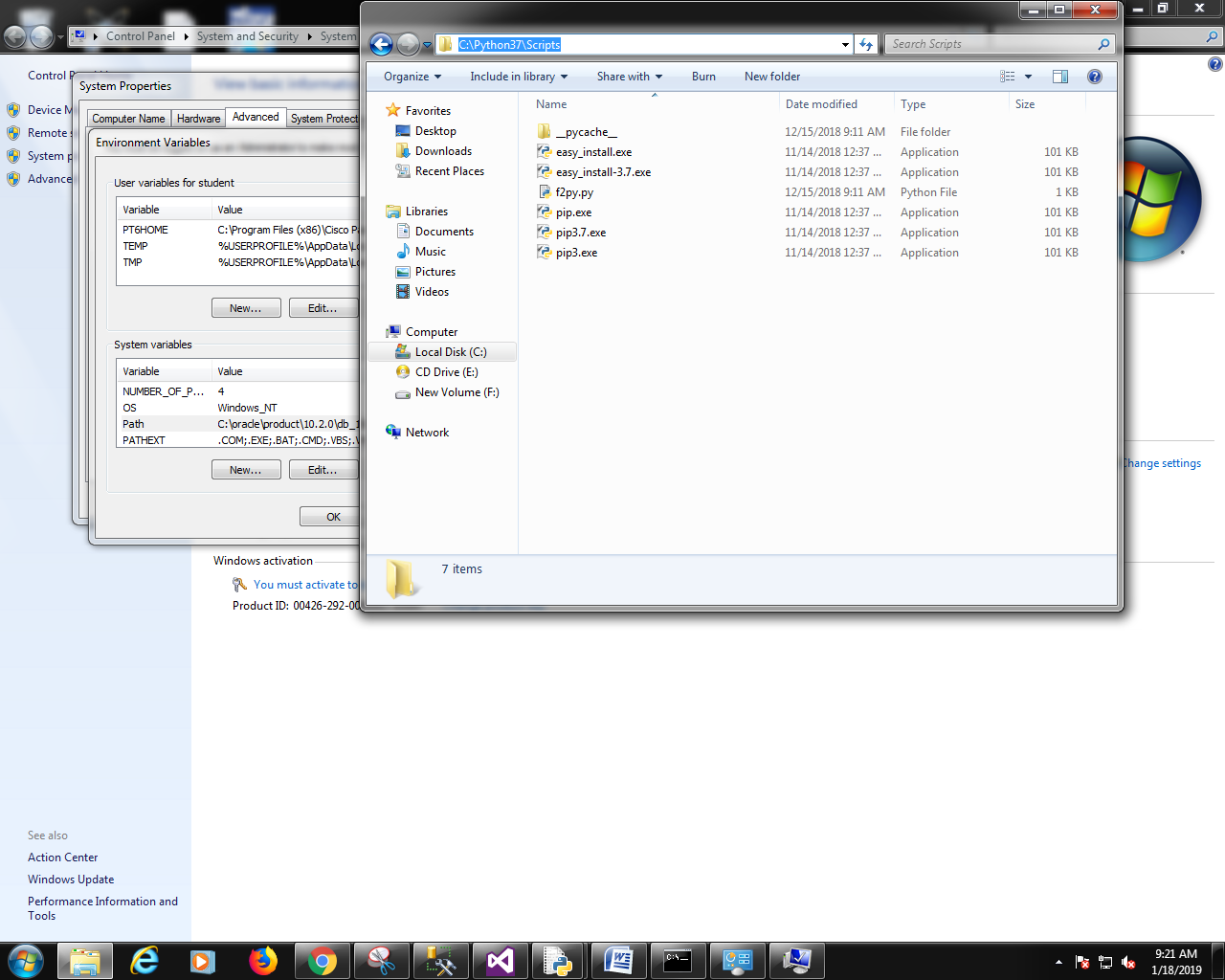
Output:

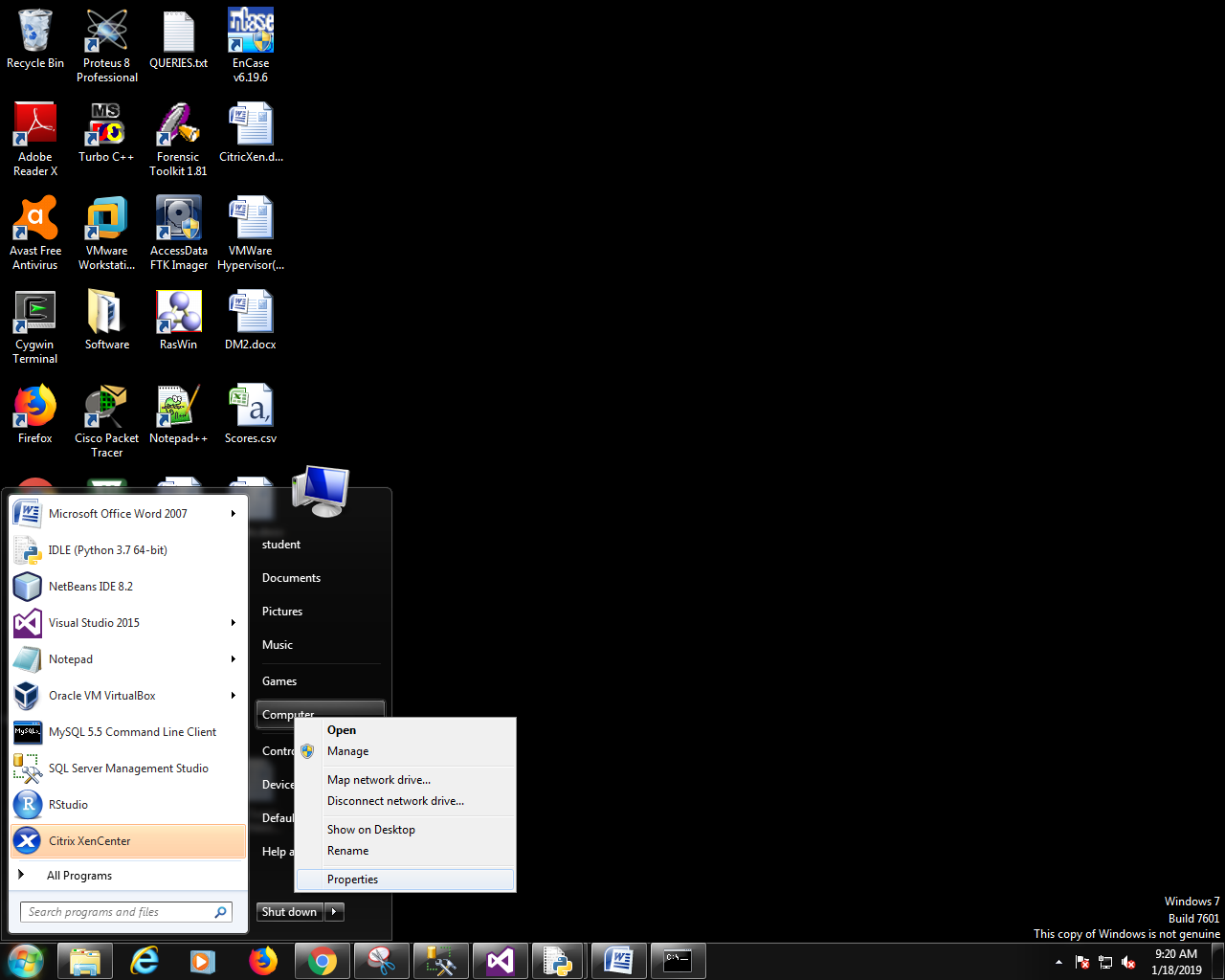


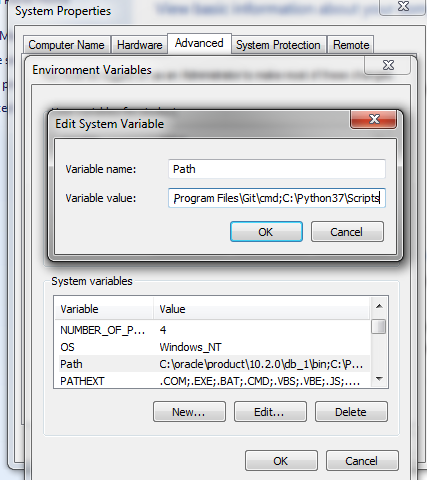
**PRACTICAL NO 03**

****

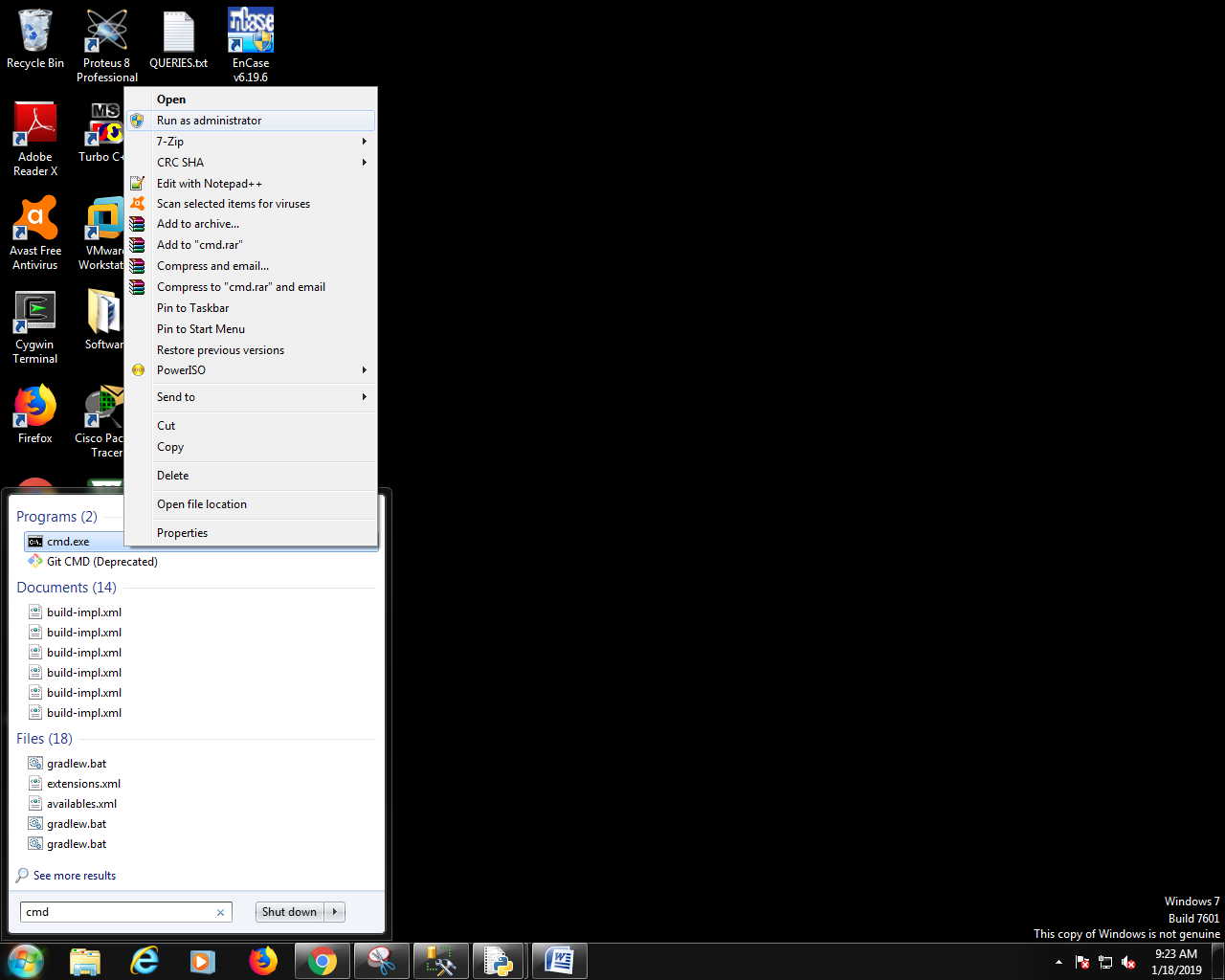
****

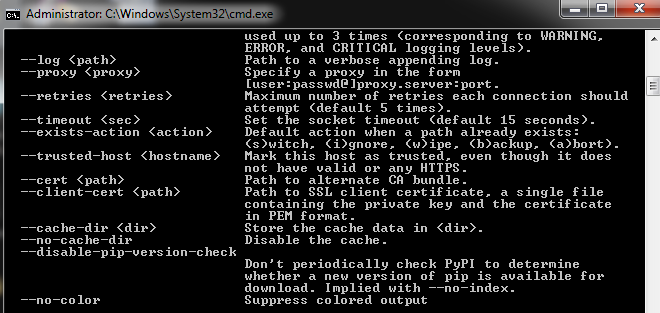
****

****

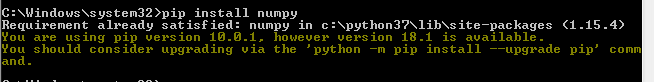
****

**open new cmd and run as administrator.**

****

****

****

****

**Write a program to do the following :**

**1. Enter 2 distinct faces as vector u and v:**

**SOURCE-CODE:**

import numpy as np

from PIL import Image

a = Image.open("img1.jpg")

b = Image.open("img2.jpg")

#resize an array using PIL

#def imgresize(im, sz):

#pil\_im = Image.fromarray(np.unit8(im))

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

#a1 = Image.from array(i1)

#a2 = Image.from array(i2)

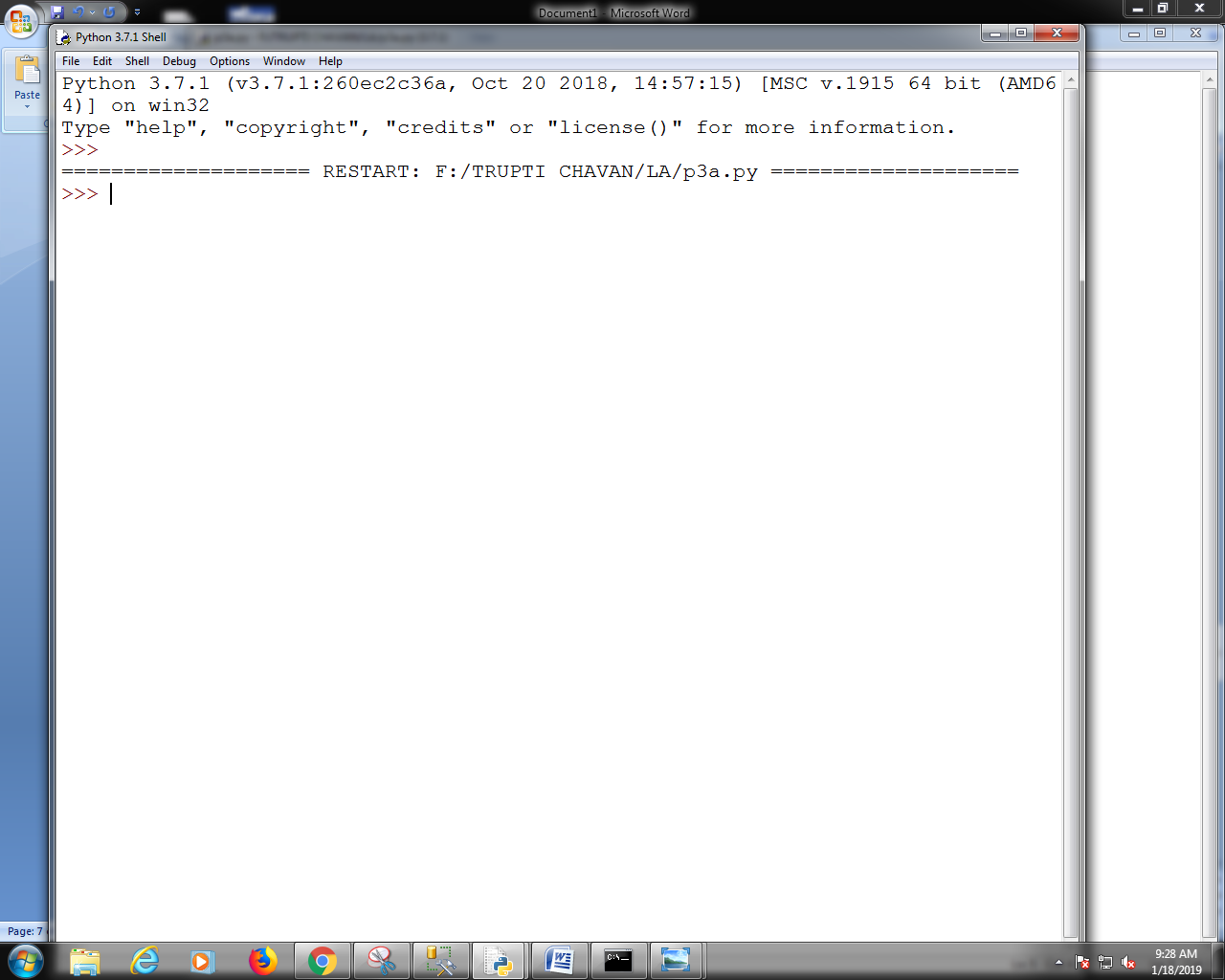
a.show()

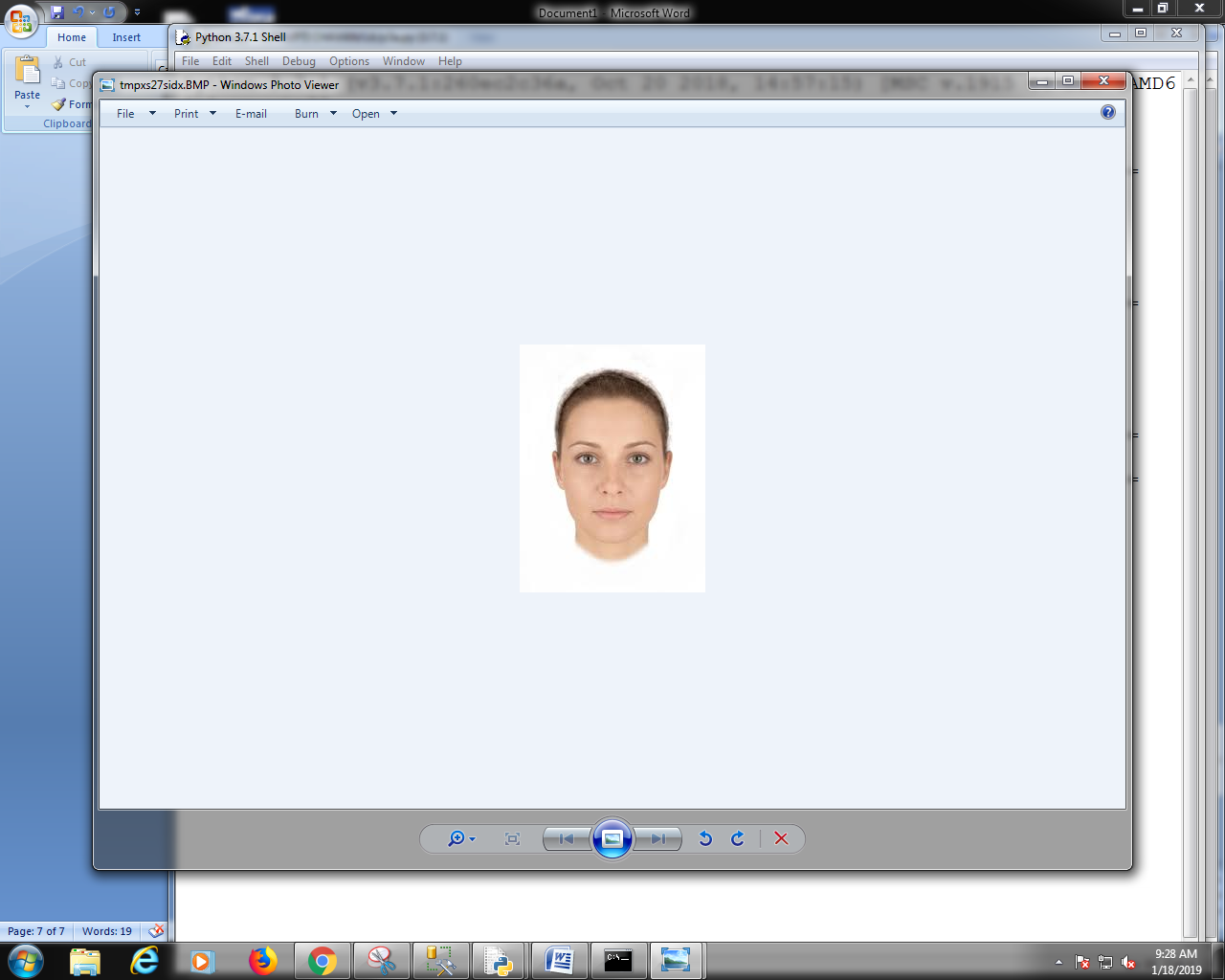
b.show()

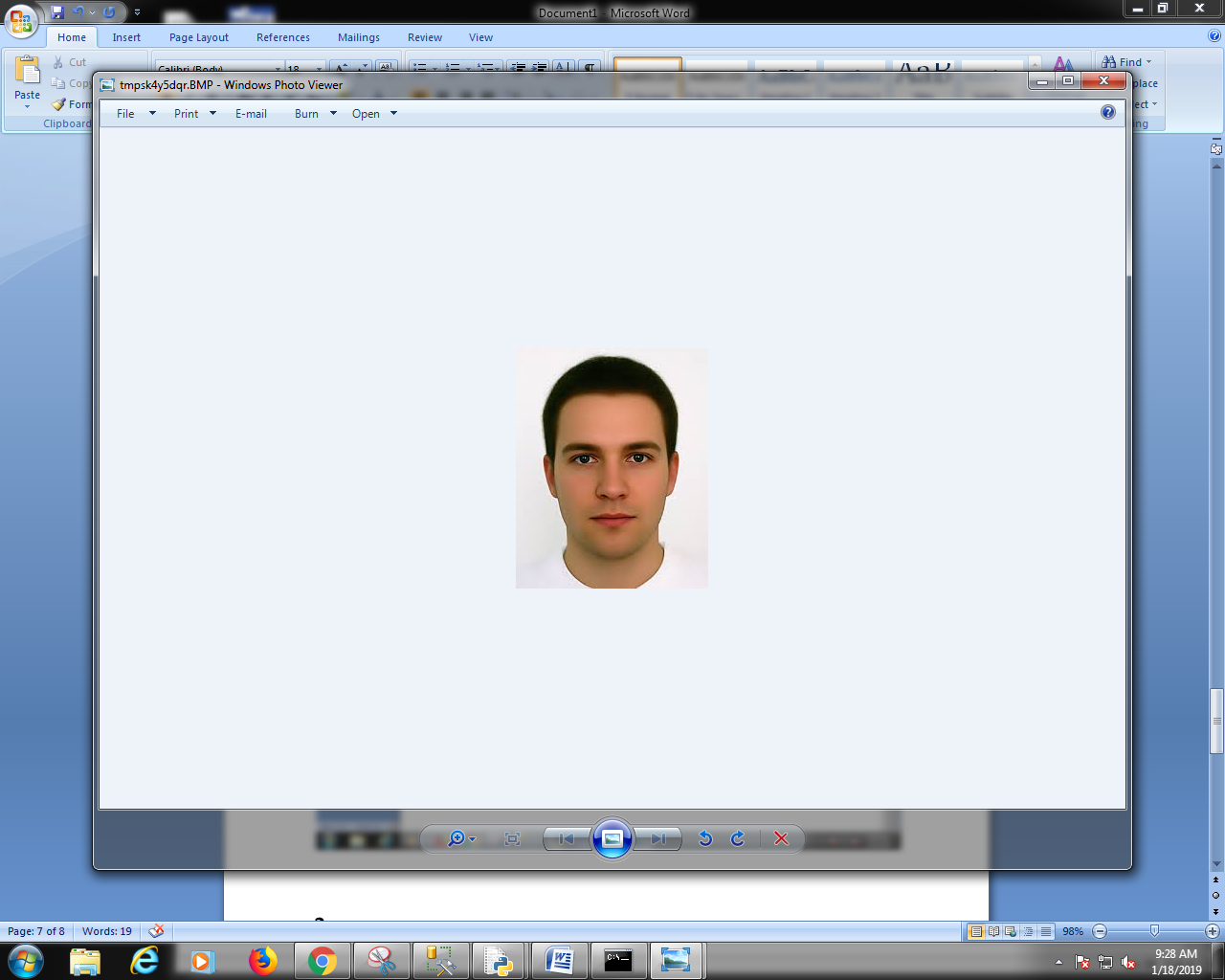
#a1.show()

#a2.show()

**OUTPUT:**

****

****

****

**2. Find a new face as a linear combination of u and v i.e. au + bv for a and b in r:**

**SOURCE-CODE:**

import numpy as np

from PIL import Image

a = Image.open("img1.jpg")

b = Image.open("img2.jpg")

i1 = np.asarray(a)

i2 = np.asarray(b)

#resize an array using PIL

def imgresize(im, sz):

pil\_im = Image.fromarray(np.uint8(im))

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

r1 = imgresize(i1,(600, 600))

r2 = imgresize(i2,(600, 600))

x = 1

y = 2

lc = x\*r1 + y\*r2

lc = Image.fromarray(lc)

lc.show()

#a1 = Image.from array(i1)

#a2 = Image.from array(i2)

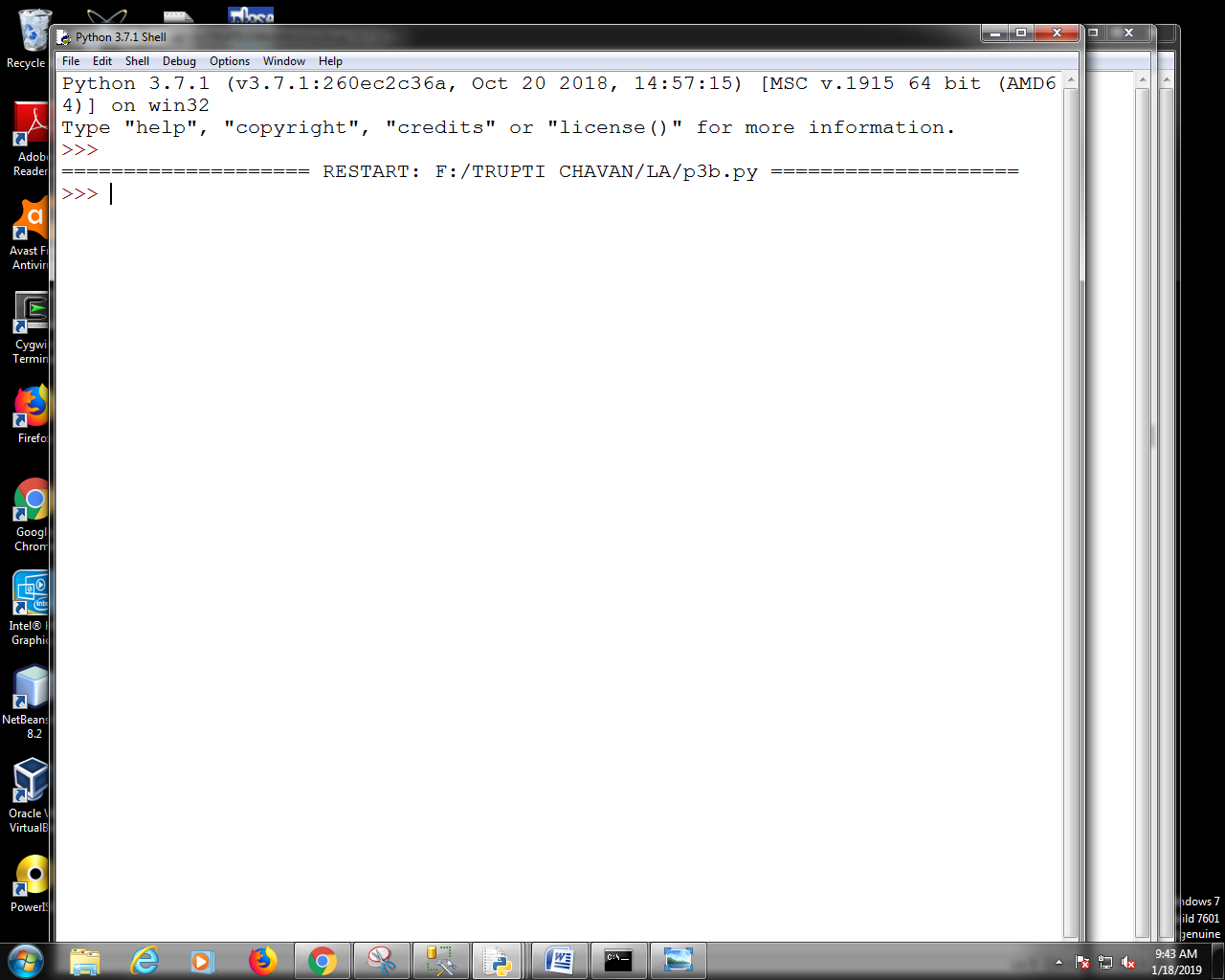
#a.show()

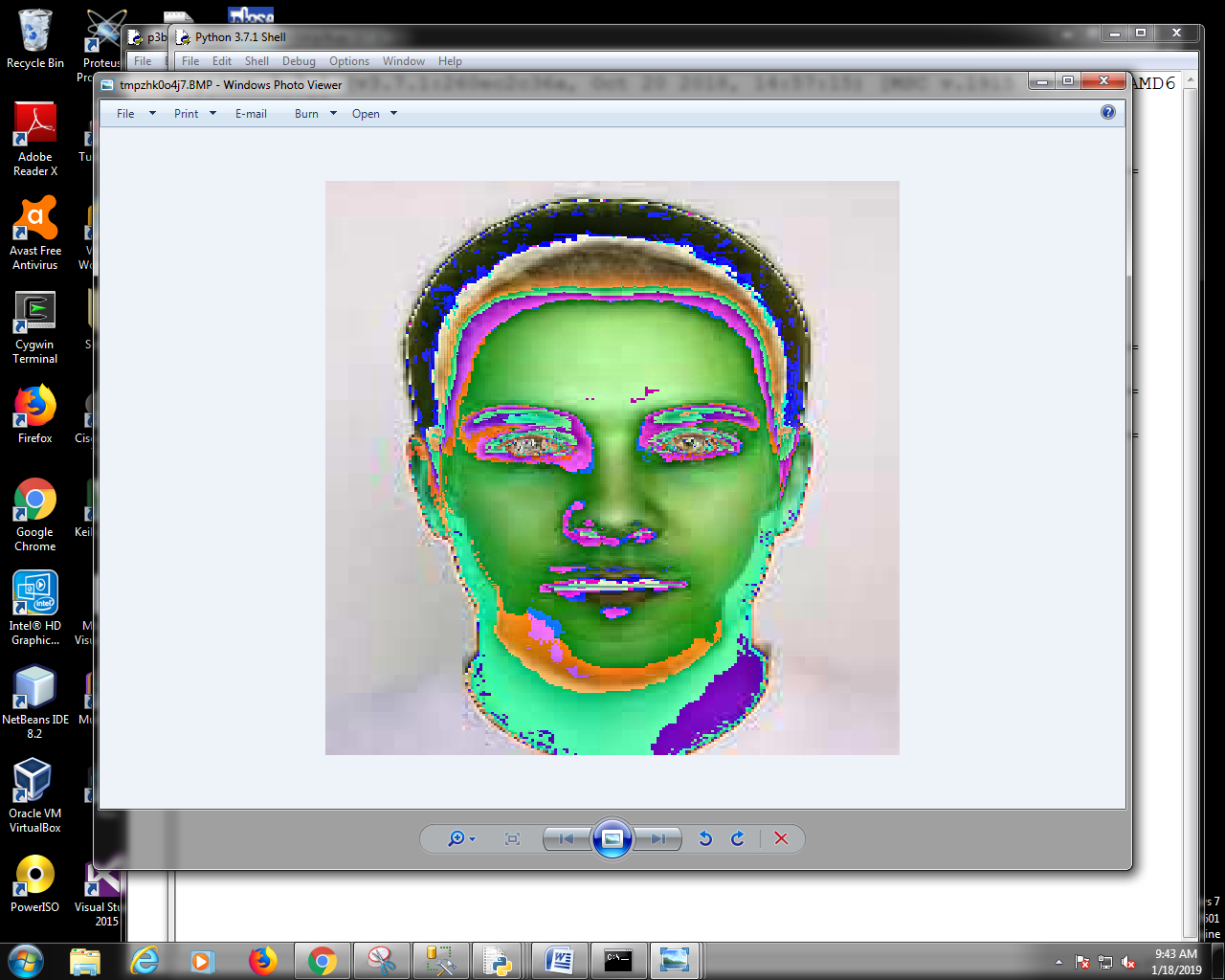
#b.show()

#a1.show()

#a2.show()

**OUTPUT:**

****

****

**3. Find the average face of the original faces:**

**SOURCE-CODE:**

import numpy as np

from PIL import Image

a = Image.open("img1.jpg")

b = Image.open("img2.jpg")

i1 = np.asarray(a)

i2 = np.asarray(b)

#resize an array using PIL

def imgresize(im, sz):

pil\_im = Image.fromarray(np.uint8(im))

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

r1 = imgresize(i1,(600, 600))

r2 = imgresize(i2,(600, 600))

r = (r1+r2)/2

avgimg = np.asarray(r)

finalimg = Image.fromarray(np.uint8(avgimg))

finalimg.show()

#a1 = Image.fromarray(i1)

#a2 = Image.fromarray(i1)

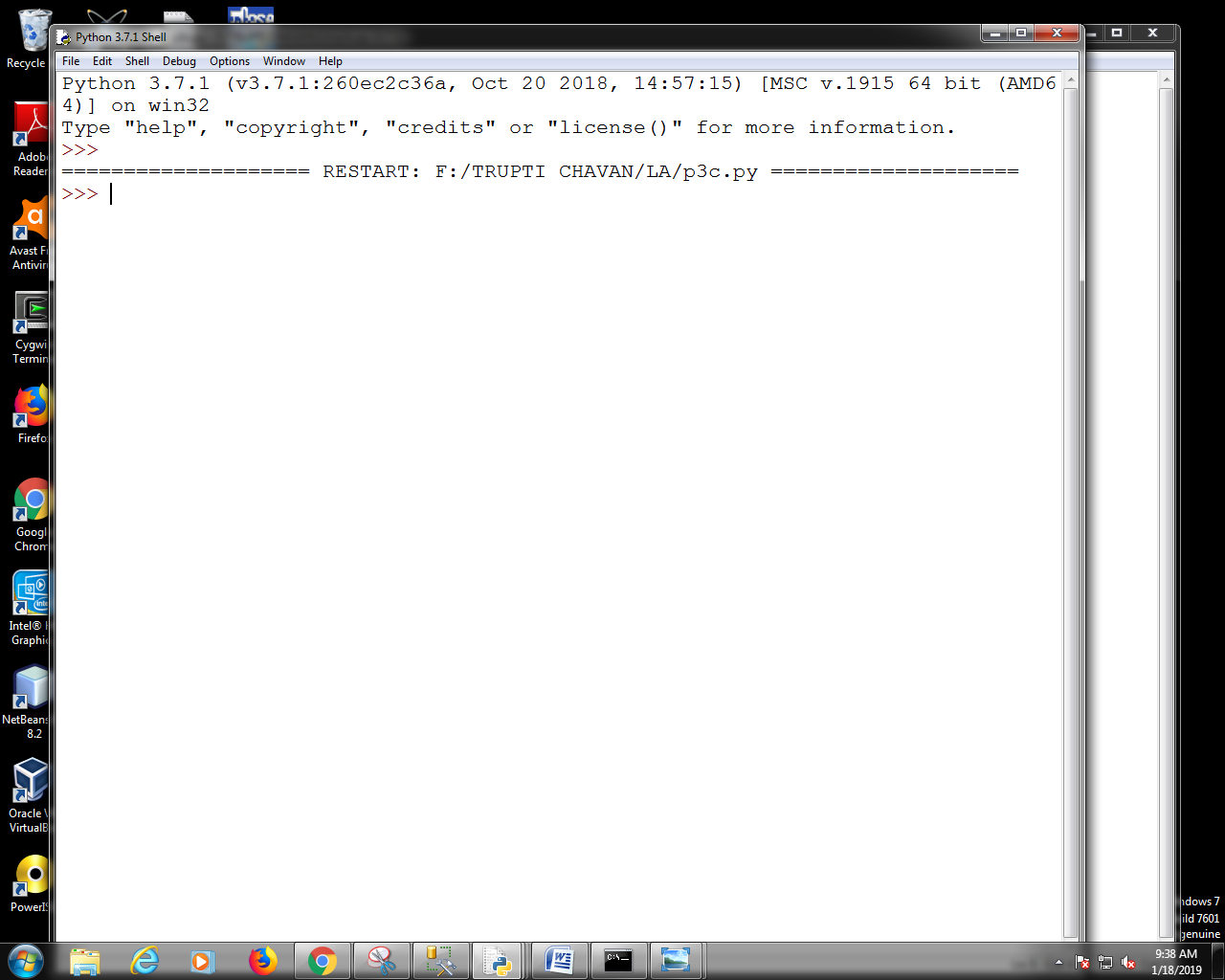
#a.show()

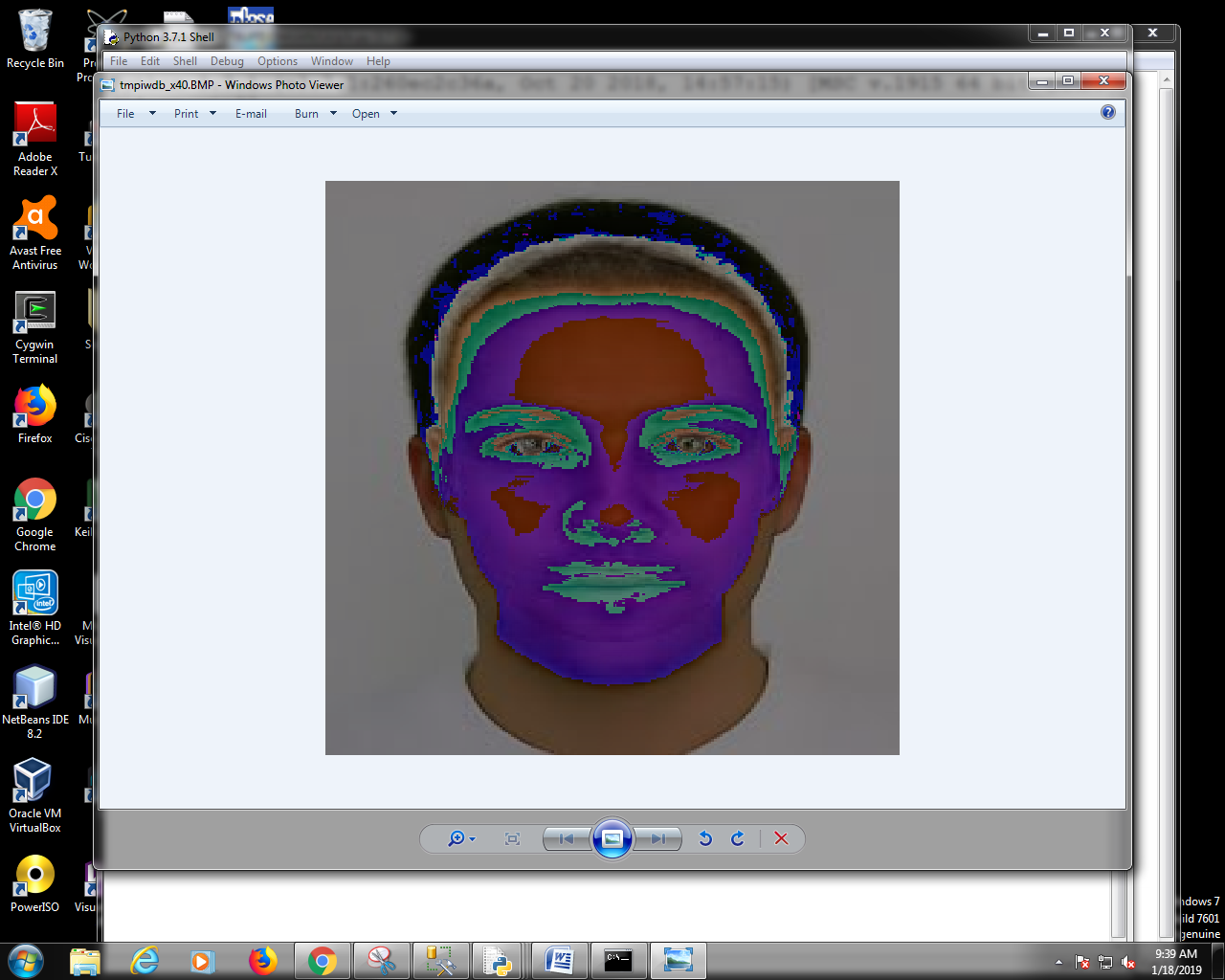
#b.show()

#a1.show()

#b1.show()

**OUTPUT:**

****

****

LA Pract4

Code:

global r,c

def matrix(A):

print('the entered matrix M is')

for i in range(r):

print(A[i])

def rows(A):

print('Rows of matrix')

for i in range (r):

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

def columns(A):

print('columns of matrix')

for j in range(c):

print('column%d='%j,end="")

for i in range(r):

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

print('\n')

def transpose(A):

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

print('Transpose of M.T=')

for j in range(c):

print(T[j])

def scalarmul(A,s):

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

print('The scalar multiplication =')

matrix(N)

print ('enter the dimensions of matrix ')

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

c=int(input('enter no of columns'))

a=[]

for i in range (r):

print('enter elements of row',i)

a.append([])

for j in range(c):

val=int(input('enter no'))

a[i].append(val)

print('Select operation')

print('1:Display Matrix')

print('2:Display rows of matrix')

print('3:Display columns of matrix')

print('4:Scalar Multiplication of matrix')

print('5:Transpose of matrix')

print('6:Exit')

while True:

ch=int(input('Enter choice for Operation'))

if ch==1:

matrix(a)

elif ch==2:

rows(a)

elif ch==3:

columns(a)

elif ch==4:

val=int(input('Enter scalar value'))

scalarmul(a,val)

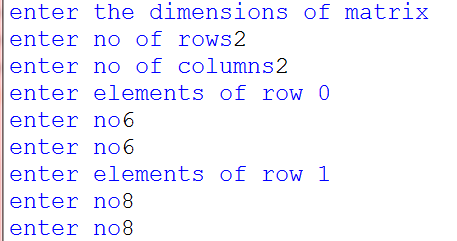
elif ch==5:

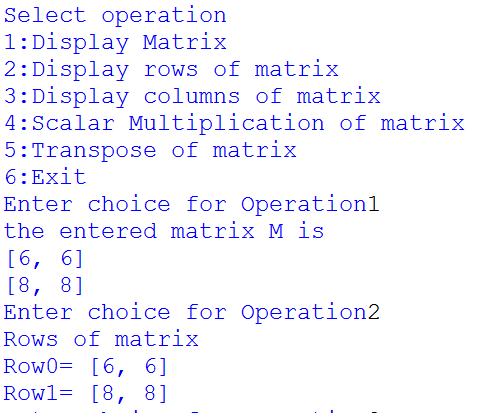
transpose(a)

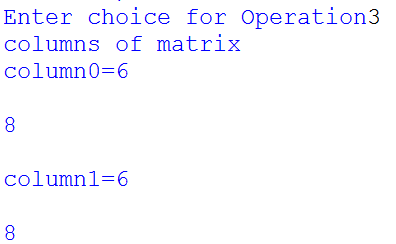
elif ch==6:

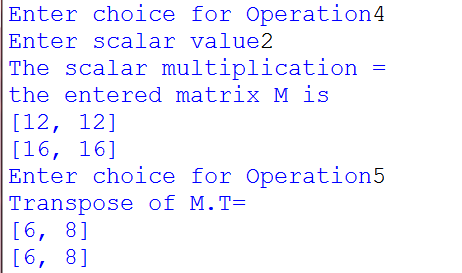
break;

Output:









Practical 5

Q1) Write a python program to find the

a)matrix vector multiplication of r by c matrix m with c vector

b)matrix matrix product of m with a matrix M

**SOURCE CODE:**

global r1,c1,r2,c2

def displaymatrix(M):

for i in range(len(M)):

print(M[i])

def matrixandvector(M,v):

Q=[sum(M[i][j]\*v[j]for j in range(len(v)))for i in range(len(M))]

print('Matrix vector multiplication=',Q)

def vectorandmatrix(v,M):

Q=[sum(v[j]\*M[j][i]for j in range(len(v)))for i in range(len(M))]

print('vector of matrix multiplication=',Q)

def matrixandmatrix(M1,M2):

Q=[[sum(M1[i][k]\*M2[k][j] for k in range(len(M2)))for j in range(len(M2[0]))]for i in range(len(M1))]

print('Multiplication of 2 matrix =')

displaymatrix(Q)

def dotmv(l1,v):

Q=[sum(l1[i]\*v[j] for j in range(len(v))) for i in range(len(l1))]

displaymatrix(Q)

def dotvm(v,M1):

Q=[sum(M[i]\*v[j] for j in range(len(v))) for i in range(r1)]

print('enter the dimensionof Matrix1')

r1=int(input('enter no of rows'))

c1=int(input('enter no of columns'))

M1=[]

for i in range(r1):

print('enter elements of row',i)

M1.append([])

for j in range(c1):

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

M1[i].append(n)

print('The entered Matrix M1 is')

displaymatrix(M1)

print('enter the dimensions of Matrix2')

r2=int(input('enter no of rows'))

c2=int(input('enter no of columns'))

M2=[]

for i in range(r2):

print('enter elements of row',i)

M2.append([])

for j in range(c2):

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

M2[i].append(n)

print('The entered Matrix M2 is')

displaymatrix(M2)

print("Select CHOICE")

print("1: Multiply matrix and vector")

print("2: Multiply vector and matrix")

print("3: Multiply matrix and matix")

print('6:Exit')

while True:

choice=int(input('Enter choice for operation'))

if choice==1:

a=input("enter the vectors")

vec=[int(x)for x in a.split()]

print(len(vec))

if len(vec)!=c1:

print("invalid vector")

else:

matrixandvector(M1,vec)

elif choice==2:

b=input("enter the vector")

vec2=[int(x)for x in b.split()]

if len(vec2)!=r1:

print("invalid vector")

else:

vectorandmatrix(vec2,M1)

elif choice==3:

if c1==r2:

matrixandmatrix(M1,M2)

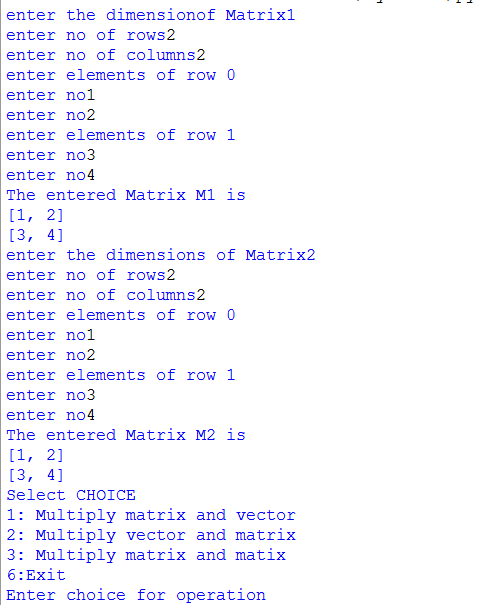
else:

print("cant do the operation")

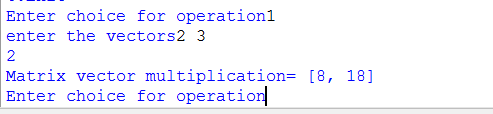
else:

break

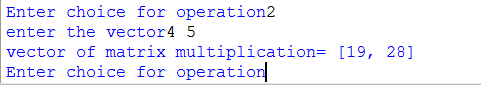
**OUTPUT:**



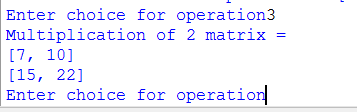
**1) matrix and vector**



**2) vector and matrix**



**3)matrix matrix**



LA - Practical 6

Inverse of a Matrix

Without numpy :

Code:

#inverse

#1/ad-bc \*[[d -b],[-c a]]

m1 = [[2,3],[1,4]]

d = 1/(m1[0][0]\*m1[1][1] - m1[0][1]\*m1[1][0])

im= [[d\*m1[1][1],-(d\*m1[0][1])],[-(d\*m1[1][0]),d\*m1[0][0]]]

print("Inverse of the given Matrix :",im)

Output:



With numpy :

Code:

#By using numpy

import numpy as np

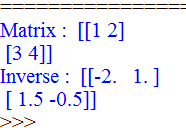
a = np.matrix([[1,2],[3,4]])

print("Matrix : ",a)

ainv = np.linalg.inv(a)

print("Inverse : ",ainv)

Output:



LA Practical 7

6806

Q1) Finding GCD

**sourcecode:**

def GCD(a,b):

if b>a:

return GCD(b,a)

if a%b==0:

return b

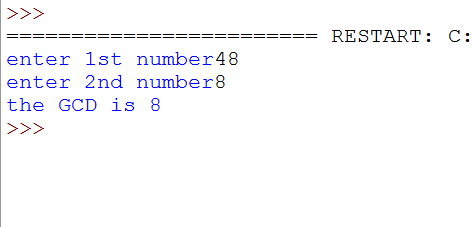
return GCD(b,a%b)

a=(int(input("enter 1st number")))

b=(int(input("enter 2nd number")))

print("the GCD is",GCD(a,b))

output :



Q2)

**sourcecode**:

n=21

for i in range(1,n+1):

for j in range(1,n+1):

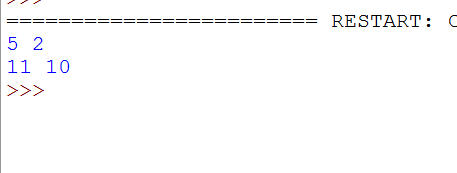
c=i\*\*2-j\*\*2

if(c==n):

print(i,j)

break

**output**:



practical 8

code:

import itertools as it

v=[1,1,1,1]

b=[1,2,4,5]

dot=0

dot=0

#for i in range(len(p)):

# dot=dot+p[i]

#p=[u[i]\*v[i] for i in range(len(v))]

def dotproduct(u,v):

dot=0

p=[u[i]\*v[i] for i in range(len(v))]

print("p",p)

for i in range(len(u)):

dot+=p[i]

return dot

a3=(dotproduct(b,v)/dotproduct(v,v))

print("a3",a3)

scalar=[]

for i in range(0,len(v)):

x=(a3)\*v[

i]

scalar.append(x)

p=[u[i]\*v[i] for i in range(len(v))]

f=[]

for i in range(0,len(v)):

ans=b[i]-scalar[i]

f.append(ans)

print("projection=",f)

Output=>

