LINEAR ALGEBRA USING PYTHON PRACTICAL MANUAL

PROF.AJAY PASHANKAR

WWW.PROFAJAYPASHANKAR.COM

PRACTICAL-1:

Write a program which demonstrate the following-

i)Addition of two complex numbers.

ii) Displaying the conjugate of a complex number.

iii) Plotting a set of complex numbers.

iv) Creating a new plot by rotating the given number by a degree 90, 180, 270 degrees and also by scaling b a number a=1/2, a=1/3, a=2 etc.

```
import matplotlib.pyplot as plt
S={3+3j,4+3j,2+1j,2.5+1j,3+1j,3.25+1j}
print('Select operation')
print('1: Addition of two 2 complex number ')
print('2: Plot points from set of complex number')
print('3:Translation')
print('4:Scaling')
print('5:Rotation')
print('6:Exit')
```

```
while True:
    ch=int(input('Enter choice for operation'))
    if ch==1:
        c1=complex(input('Enter complex no c1'))
        c2=complex(input('Enter complex no c2'))
        print('Addition of two 2 complex number (c1+c2)is',c1+c2)
    elif ch==2:   #plotting
        S1={x for x in S}
    elif ch==3:   #Translation
        c1=complex(input('Enter Translation in complex no format'))
        S1={x+c1 for x in S}
    elif ch==4:   #Scaling
        scale=float(input('Enter scale pointlike(0.5) for 1/2'))
        S1={x*scale for x in S}
    elif ch==5:
```

```
        angle=int(input('Enter angle of rotation 90/180/270'))
        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 Enter only 90/180/270 degree')
    else:
        break


X=[x.real for x in S1]
Y=[x.imag for x in S1]
plt.plot(X,Y,'ro')
```

```
plt.axis([-6,6,-6,6])

plt.show()
```

Output:

PRACTICAL-2(a):

Write a program to do the following-

i) Enter a vector u as a n-list.
ii) Enter another vector v as a n-list.
iii) Find the vector au +bv for different values of a and b.
iv) Find the dot produvt of u and v.

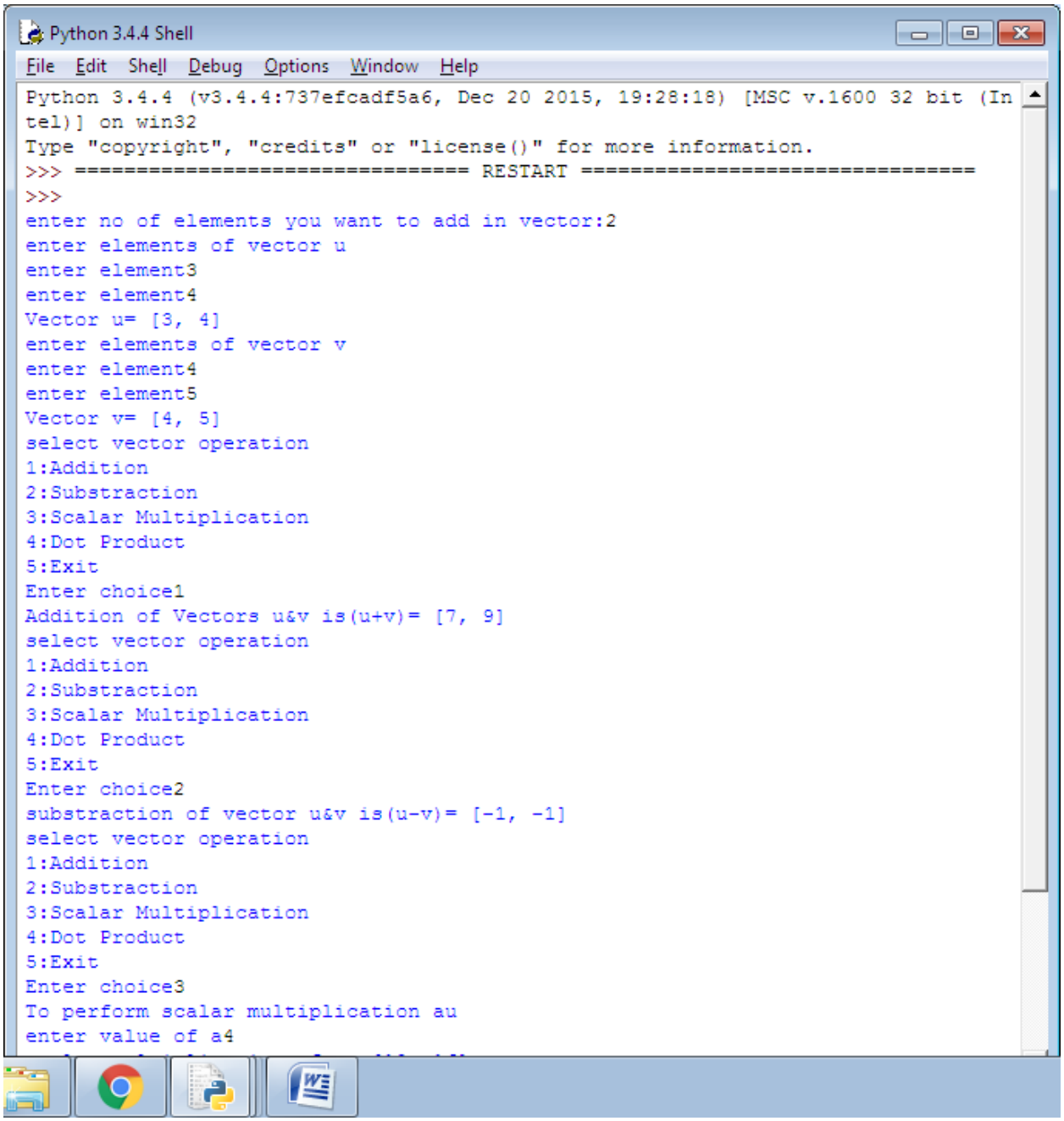
```
def addvec(x,y):
    return[x[i]+y[i]for  i in range(len(x))]
def subvec(x,y):
    return[x[i]-y[i]for  i in range(len(x))]
def scalarmul(x,p):
```

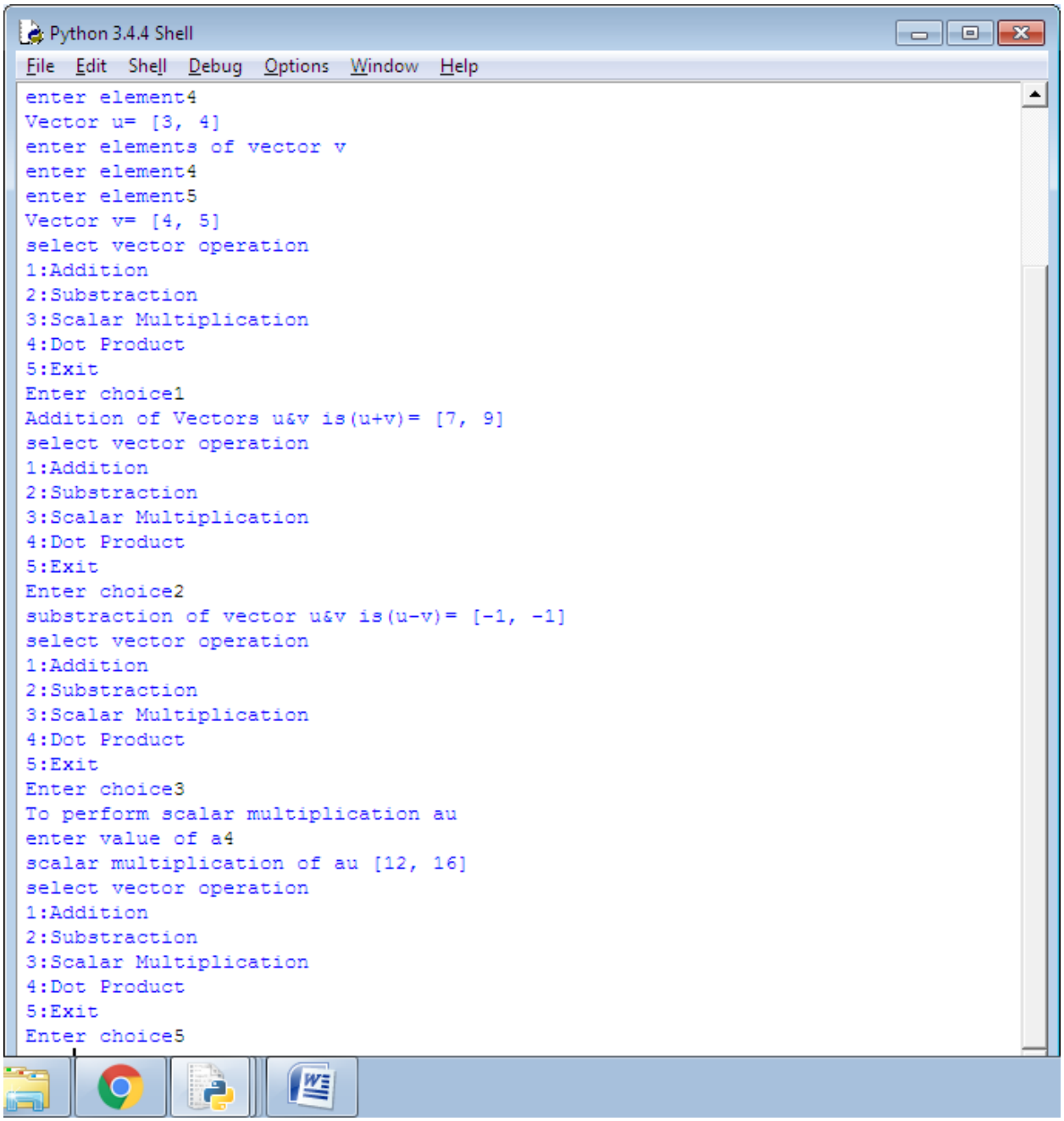
```
    return[p*x[i]for i in range(len(x))]
def dotprod(x,y):
    return sum([x[i]*y[i] for i in range(len(x))])
v=[]
u=[]
n=int(input('enter no of elements you want to add in vector:'))
print('enter elements of vector u')
for i in range(n):
    elem=int(input('enter element'))
    u.append(elem)
print('Vector 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)
while True:
```

```
    print('select  vector operation')
    print('1:Addition')
    print('2:Substraction')
    print('3:Scalar  Multiplication')
    print('4:Dot  Product')
    print('5:Exit')
    ch=int(input('Enter  choice'))
    if ch==1:
        print('Addition  of Vectors u&v
is(u+v)=',addvec(u,v))
    elif ch==2:
        print('substraction  of vector u&v is(u-
v)=',subvec(u,v))
    elif ch==3:
        print('To perform  scalar multiplication  au')
        a=int(input('enter  value of a'))
        print('scalar  multiplication  of au',scalarmul(u,a))
    elif ch==4:
```

```
        print('Dot product of u & v(u,v)',dotprod(u,v))
    else:
        break
```

Output:


Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
enter no of elements you want to add in vector:2
enter elements of vector u
enter element3
enter element4
Vector u= [3, 4]
enter elements of vector v
enter element4
enter element5
Vector v= [4, 5]
select vector operation
1:Addition
2:Substraction
3:Scalar Multiplication
4:Dot Product
5:Exit
Enter choice1
Addition of Vectors u&v is(u+v)= [7, 9]
select vector operation
1:Addition
2:Substraction
3:Scalar Multiplication
4:Dot Product
5:Exit
Enter choice2
substraction of vector u&v is(u-v)= [-1, -1]
select vector operation
1:Addition
2:Substraction
3:Scalar Multiplication
4:Dot Product
5:Exit
Enter choice3
To perform scalar multiplication au
enter value of a4



PRACTICAL-2(b):

Vector Implementation using Class.

```
def setitem(v,d,val):v.f[d]=val

def getitem(v,d):
    if d in v.f:
        return v.f[d]
    else:
        return 0
def scalar_mul(v,a):
    return Vector(v.D,{d:a*getitem(v,d)for d in v.D})
def add(u,v):
    return Vector(u.D,{d:getitem(u,d)+getitem(v,d)for d
in u.D})
def neg(v):return scalar_mul(v,-1)
def zero_vec(D):return Vec(D,{})
def list_dot(u,v):return sum([u[i]*v[i]for i in
range(len(u))])
class Vector:
    def __init__(self,labels,function):
        self.D=labels
```

```
        self.f=function
```

Output:

PRACTICAL-3(a):

Write a program to do the following-

Enter two distinct faces as vectors u and v.

i) **Find a new face as a liner combination of u and v i.e. au + bv for a and b in R.**
ii) **Find the average face of the original faces.**

Linear combination.

```
def lin_comb(vlist,clist):
 S=[scalermul(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.append(su)
 return l
```

Output:

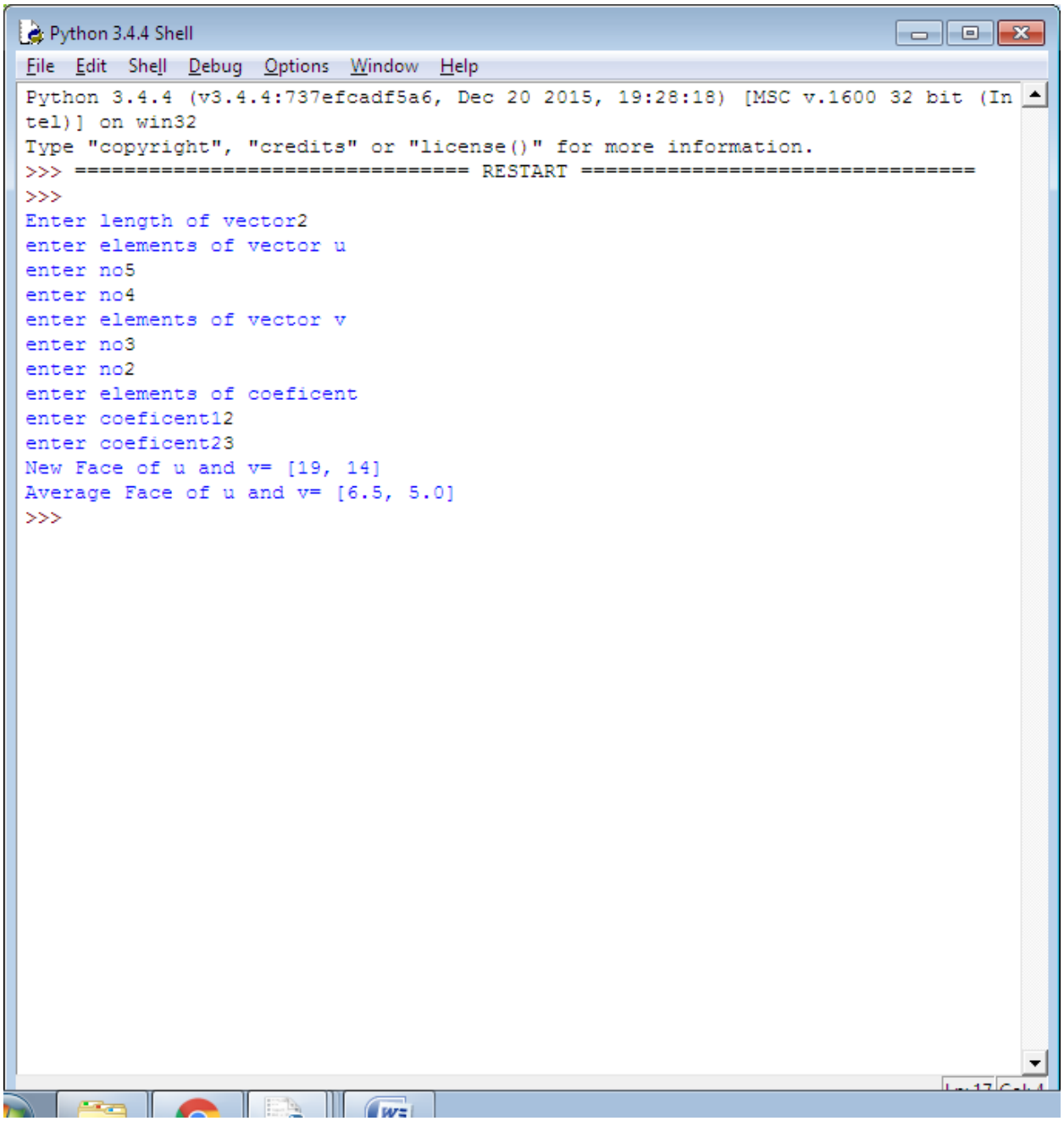
PRACTICAL-3(b):

Average face value.

```
l=int(input('Enter 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 elements of vector v')
```

```
for i in range(l):
    n=int(input('enter no'))
    v.append(n)
print('enter elements of coeficent')
c1=int(input('enter coeficent1'))
c2=int(input('enter coeficent2'))
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:



PRACTICAL-4:

Write a program to do the following-

i) Enter an r by c matrix M(r and c being positive integers).
ii) Display M in matrix format.
iii) Display the row and columns of the matrix M.
iv) Find the scalar multiplication of M for a given scalar.
v) Find the transpose of the matrix M.

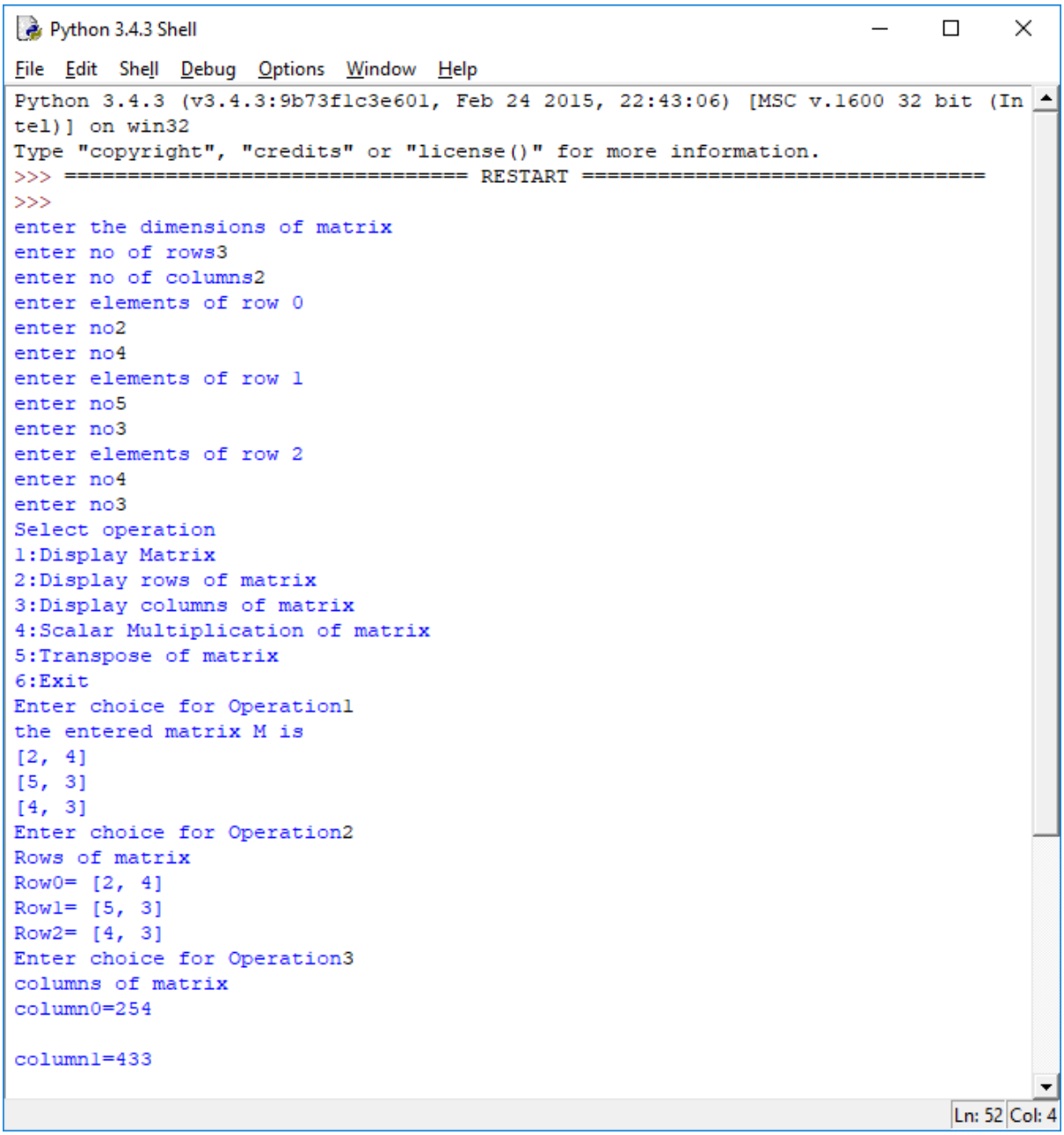
```
global r,c
#display M in matrix format
def printmatrix(A):
    print('the entered matrix M is')
    for i in range(r):
        print(A[i])
#display rows of matrix
def printrows(A):
    print('Rows of matrix')
    for i in range (r):
        print('Row%d='%i,A[i])
#display columns of matrix
```

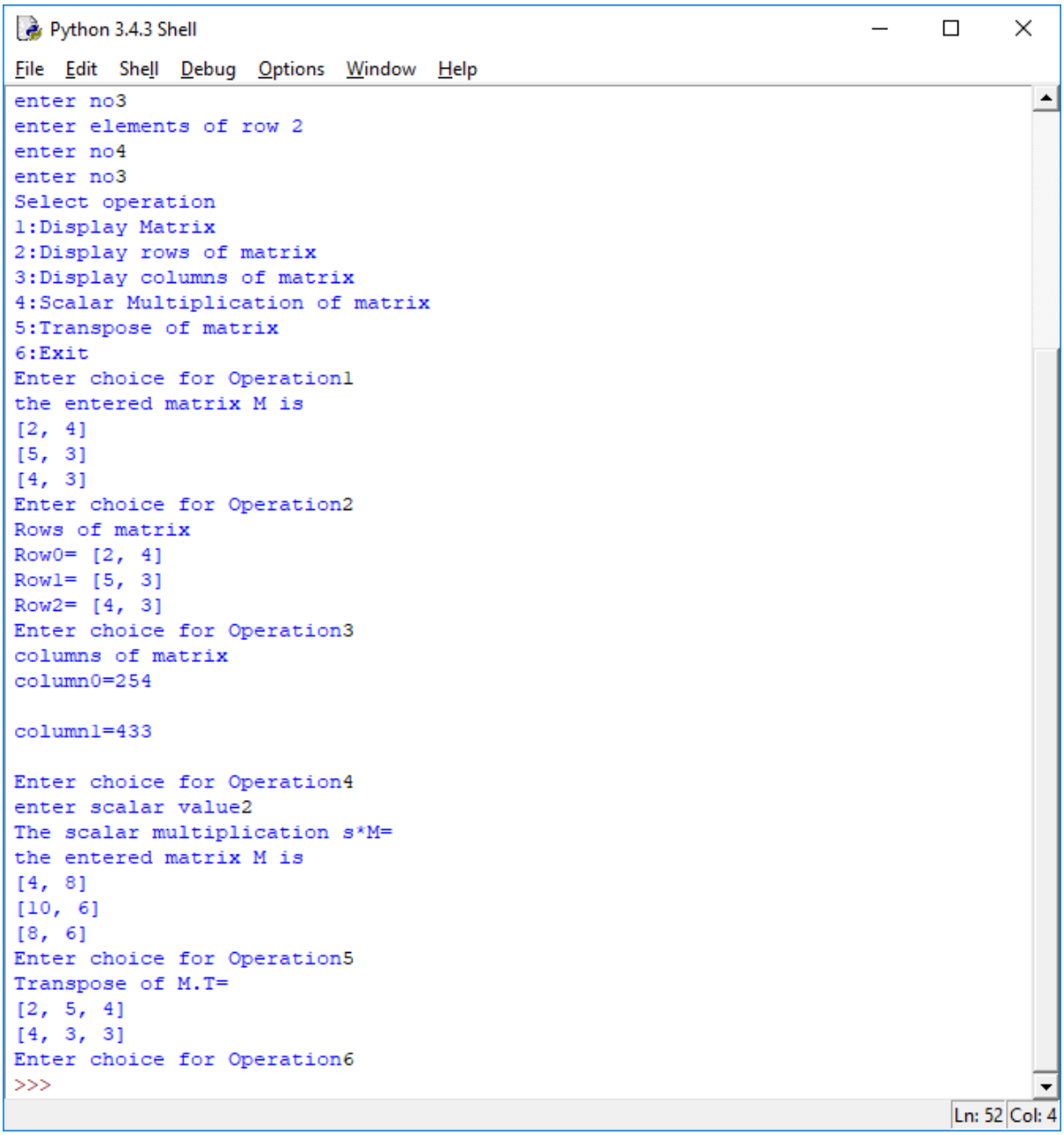
```
def printcolumns(A):
        print('columns of matrix')
        for j in range(c):
            print('column%d='%j,end="")
            for i in range(r):
                print(A[i][j],end="")
            print('\n')
#Scalar Multiplication
def scalarmul(A,s):
        N=[[s*A[i][j] for j in range(c)] for i in range(r)]
        print('The scalar multiplication s*M=')
        printmatrix(N)
#tranpose of matrix M
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])
```

```
#Enter rXc Matrix M
print ('enter the dimensions of matrix ')
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('enter no'))
        M[i].append(n)
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:
          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:
          break;
```

Output:


Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
enter the dimensions of matrix
enter no of rows3
enter no of columns2
enter elements of row 0
enter no2
enter no4
enter elements of row 1
enter no5
enter no3
enter elements of row 2
enter no4
enter no3
Select operation
1:Display Matrix
2:Display rows of matrix
3:Display columns of matrix
4:Scalar Multiplication of matrix
5:Transpose of matrix
6:Exit
Enter choice for Operation1
the entered matrix M is
[2, 4]
[5, 3]
[4, 3]
Enter choice for Operation2
Rows of matrix
Row0= [2, 4]
Row1= [5, 3]
Row2= [4, 3]
Enter choice for Operation3
columns of matrix
column0=254
column1=433
Ln: 52 Col: 4



PRACTICAL -5:

Write a program to do the following-

i) Find the vector-matrix multiplication of a r by c matrix M with an c- vector u.

ii) Find the matrix- matrix product of M with a c by p matrix N.

```
global r1,c1,r2,c2
#display M in matrix format
def printmatrix(A):
    for i in range(len(A)):
        print(A[i])
def matrixadd(A,B):
    C=[[A[i][j]+B[i][j] for j in range(len(B[0]))] for i in range(len(A))]
    print('Addtion of two matrix=')
    printmatrix(c)
def matrixmul(A,B):
    C=[[sum(A[i][k]*B[k][j] for k in range(len(B)))for j in range(len(B[0]))]for i in range(len(A))]
    print('Multiplication of 2 matrix =')
    printmatrix(C)
```

```
def matrixvecmul(A,v):
    C=[sum(A[i][j]*v[j]for j in range(len(v)))for i in range(len(A))]
    print('Matrix vector multiplication (M1*v)=',C)
def vecmatrixmul(v,A):
    C=[sum(v[j]*A[j][i]for j in range(len(v)))for i in range(len(A))]
    print('vector of matrix multiplication (v*M1)=',C)
'''T=[]
for j in range(c):
        T.append([])
        for i in range(r):
                T[j].append(M[i][j])'''
print('enter the dimensionof Matrix1')
r1=int(input('enter no of rows'))
c1=int(input('enter no of columns'))

M=[]
```

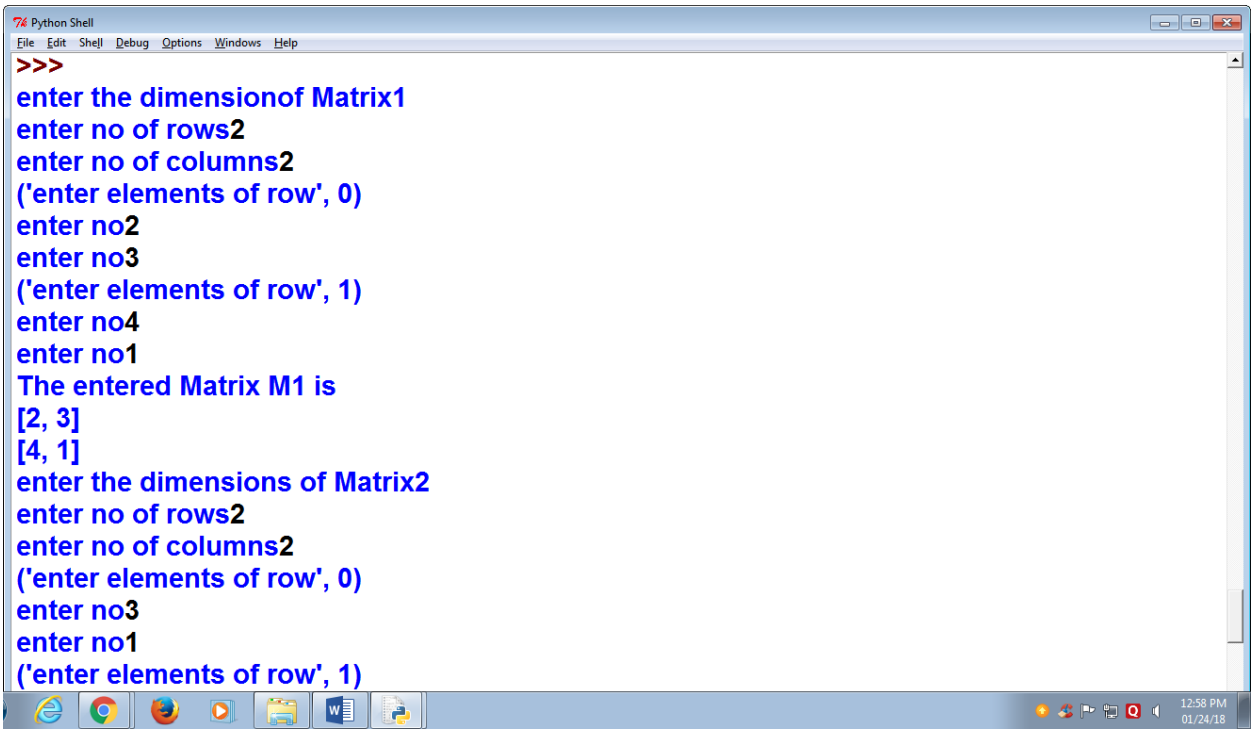
```
for i in range(r1):
    print('enter elements of row',i)
    M.append([])
    for j in range(c1):
        n=int(input('enter no'))
        M[i].append(n)
print('The entered Matrix M1 is')
printmatrix(M)

print('enter the dimensions of Matrix2')
r2=int(input('enter no of rows'))
c2=int(input('enter no of columns'))
N=[]
for i in range(r2):
    print('enter elements of row',i)
    N.append([])
    for j in range(c2):
        n=int(input('enter no'))
```

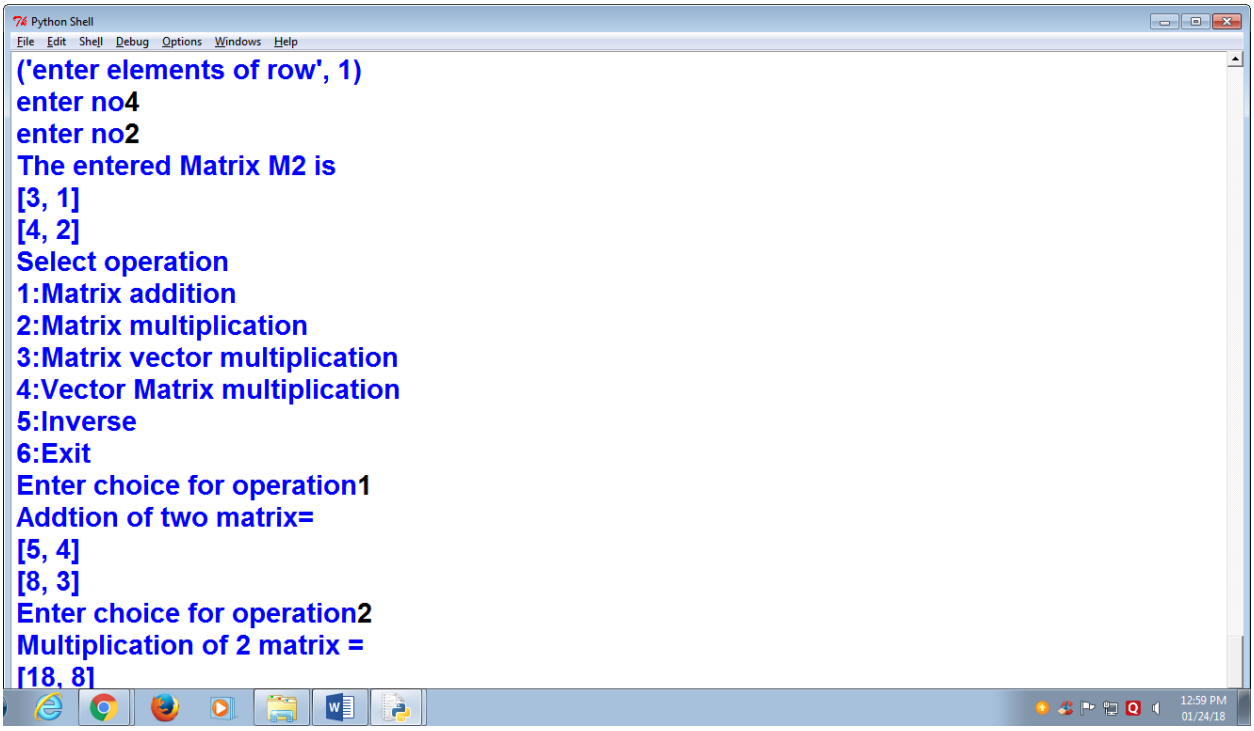
```
        N[i].append(n)
print('The entered Matrix M2 is')
printmatrix(N)


print('Select operation')
print('1:Matrix addition')
print('2:Matrix multiplication')
print('3:Matrix Vector Multiplication')
print('4:Vector Matrix multiplication')
print('5:Inverse')
print('6:Exit')


while True:
    ch=int(input('Enter choice for operation'))
    if ch==1:
        if(r1,c1)==(r2,c2):
            matrixadd(M,N)
        else:
```

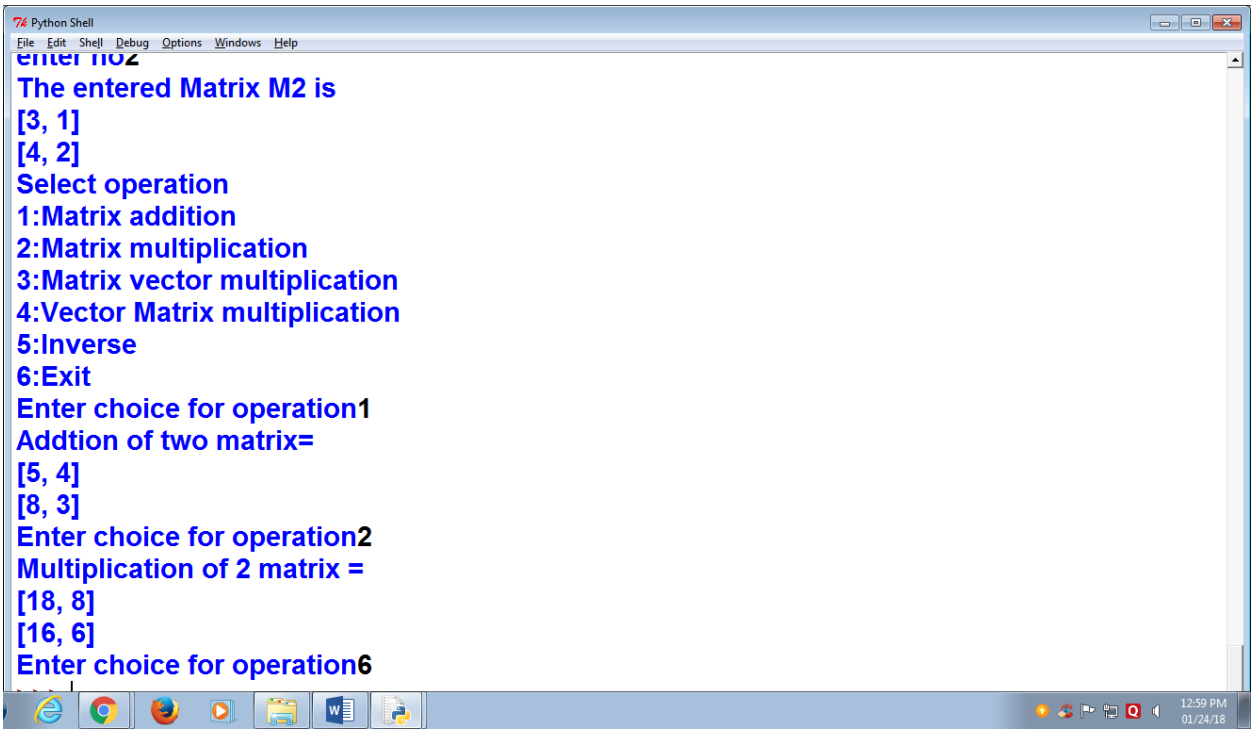
```
        print('Invalid Matrix: Addition is performed on same sizes of matrix')
    elif ch==2:
        if c1==r2:
            matrixmul(M,N)
        else:
            print('Invalid Matrix: To multiply twomatrices matrix1 column = matrix2 rows')
    elif ch==3:
        s=input('Enter elements of vector separated by spaces')
        v=[int(x)for x in s.split()]
        print(len(v))
        if len(v)!=c1:
            print('invalid vector add vector of%d elements (Columns of M1)%c1')
        else:
            matrixvecmul(M,v)
```

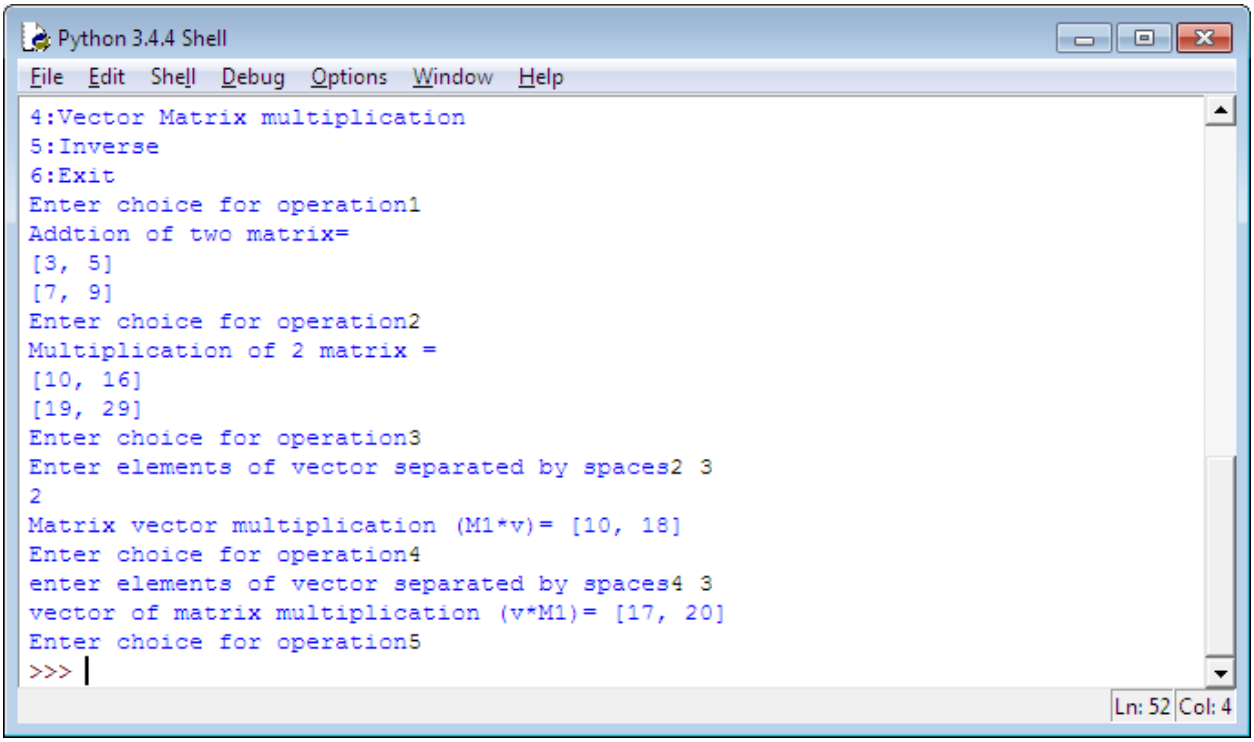
```
    elif ch==4:
        s=input('enter elements of vector separated by spaces')
        v=[int(x)for x in s.split()]
        if len(v)!=r1:
                print('invalid vector add vector of%d elements (Rows of M1)%r1')
        else:
            vecmatrixmul(v,M)
    elif ch==5:
        break
    else:
        break
```

Output:


Python Shell
File Edit Shell Debug Options Windows Help
>>>
enter the dimensionof Matrix1
enter no of rows2
enter no of columns2
('enter elements of row', 0)
enter no2
enter no3
('enter elements of row', 1)
enter no4
enter no1
The entered Matrix M1 is
[2, 3]
[4, 1]
enter the dimensions of Matrix2
enter no of rows2
enter no of columns2
('enter elements of row', 0)
enter no3
enter no1
('enter elements of row', 1)
12:58 PM
01/24/18


Python Shell
File Edit Shell Debug Options Windows Help
('enter elements of row', 1)
enter no4
enter no2
The entered Matrix M2 is
[3, 1]
[4, 2]
Select operation
1:Matrix addition
2:Matrix multiplication
3:Matrix vector multiplication
4:Vector Matrix multiplication
5:Inverse
6:Exit
Enter choice for operation1
Addtion of two matrix=
[5, 4]
[8, 3]
Enter choice for operation2
Multiplication of 2 matrix =
[18, 8]
12:59 PM
01/24/18


Python Shell
File Edit Shell Debug Options Windows Help
The entered Matrix M2 is
[3, 1]
[4, 2]
Select operation
1:Matrix addition
2:Matrix multiplication
3:Matrix vector multiplication
4:Vector Matrix multiplication
5:Inverse
6:Exit
Enter choice for operation1
Addtion of two matrix=
[5, 4]
[8, 3]
Enter choice for operation2
Multiplication of 2 matrix =
[18, 8]
[16, 6]
Enter choice for operation6


Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
4:Vector Matrix multiplication
5:Inverse
6:Exit
Enter choice for operation1
Addtion of two matrix=
[3, 5]
[7, 9]
Enter choice for operation2
Multiplication of 2 matrix =
[10, 16]
[19, 29]
Enter choice for operation3
Enter elements of vector separated by spaces2 3
2
Matrix vector multiplication (M1*v)= [10, 18]
Enter choice for operation4
enter elements of vector separated by spaces4 3
vector of matrix multiplication (v*M1)= [17, 20]
Enter choice for operation5
>>>
Ln: 52 Col: 4

PRACTICAL-6:

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

```
#diplay M in matrix format
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))]for j in range(len(A))]
    return T
```

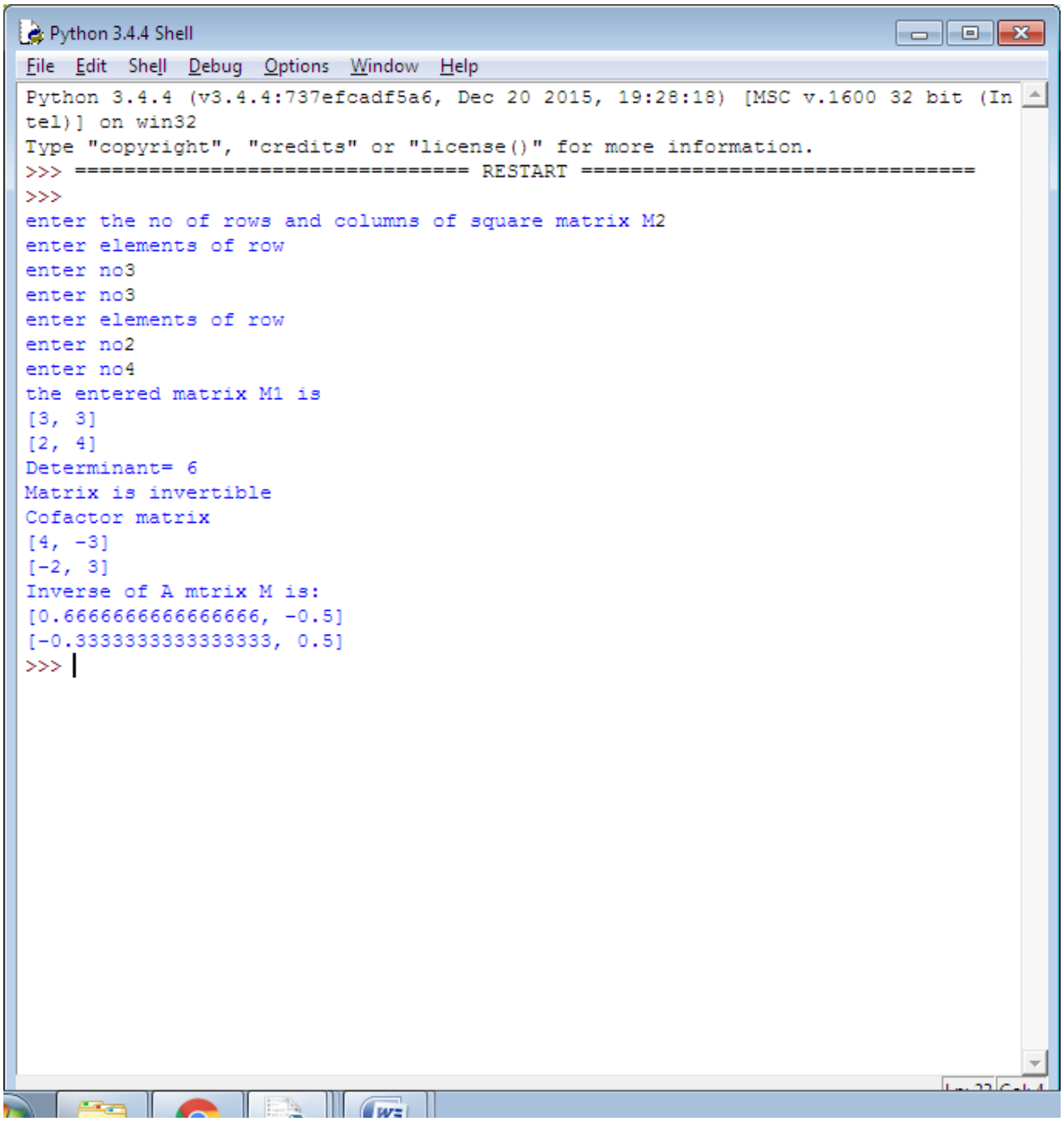
```
#Enter the number of rows and columns of matrix M
'enter the no of rows and columns of square matrix'
c=int(input('enter the no of rows and columns of square matrix M'))
r=c
M=[]
for i in range(r):
    print('enter elements of row')
    M.append([])
    for j in range(c):
        n=int(input('enter no')
        M[i].append(n)
print('the entered matrix M1 is')
printmatrix(M)
determinant=0
if r==2:
            determinant=M[0][0]*M[1][1]-M[0][1]*M[1][0]
            print('Determinant=',determinant)
```

```
if determinant ==0:
        print('Mtrix is not invertible')
else:
    print('Matrix is invertible')
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')
printmatrix(CFM)
MI=[]
for i in range(2):
          MI.append([])
          for j in range(2):
```

```
MI[i].append(CFM[i][j]/determinant)
            print('Inverse of A mtrix M is:')
            printmatrix(MI)
else:
            for i in range(3):

determinant=detrminant+(M[0][i]*M[1][(i+1)%3]*M[2][
(i+2)%3]-M[1][(i+2)%3]*M[2][(i+1)%3]);
            print('Determinant=',determinant)
            if determinant==0:
                  print('Matrix is not invertible')
            else:
                print('Matrix is invertible')



                CFM=[]
                for i in range(3):
                     CFM.append([])
```

```
            for j in range(3):

v=(M[(i+1)%3][(j+1)%3]*M[(i+2)%3][(j+2)%3])-
(M[i+1%3][(j+2)%3]*M[(i+2)%3][(j+1)%3])

                    CFM[i].append(v)

        print('Cofactor  matrix')

        printmatrix(CFM)

        AdjM=transpose(CFM)

        MI=[]

        for i in range(3):

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

        print('Inverse  of A ,Matrix  M is:')

        printmatrix(MI)
```

Output:


Python 3.4.4 Shell
File Edit Shell Debug Options Window Help
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
enter the no of rows and columns of square matrix M2
enter elements of row
enter no3
enter no3
enter elements of row
enter no2
enter no4
the entered matrix M1 is
[3, 3]
[2, 4]
Determinant= 6
Matrix is invertible
Cofactor matrix
[4, -3]
[-2, 3]
Inverse of A mtrix M is:
[0.6666666666666666, -0.5]
[-0.3333333333333333, 0.5]
>>>

PRACTICAL-7:

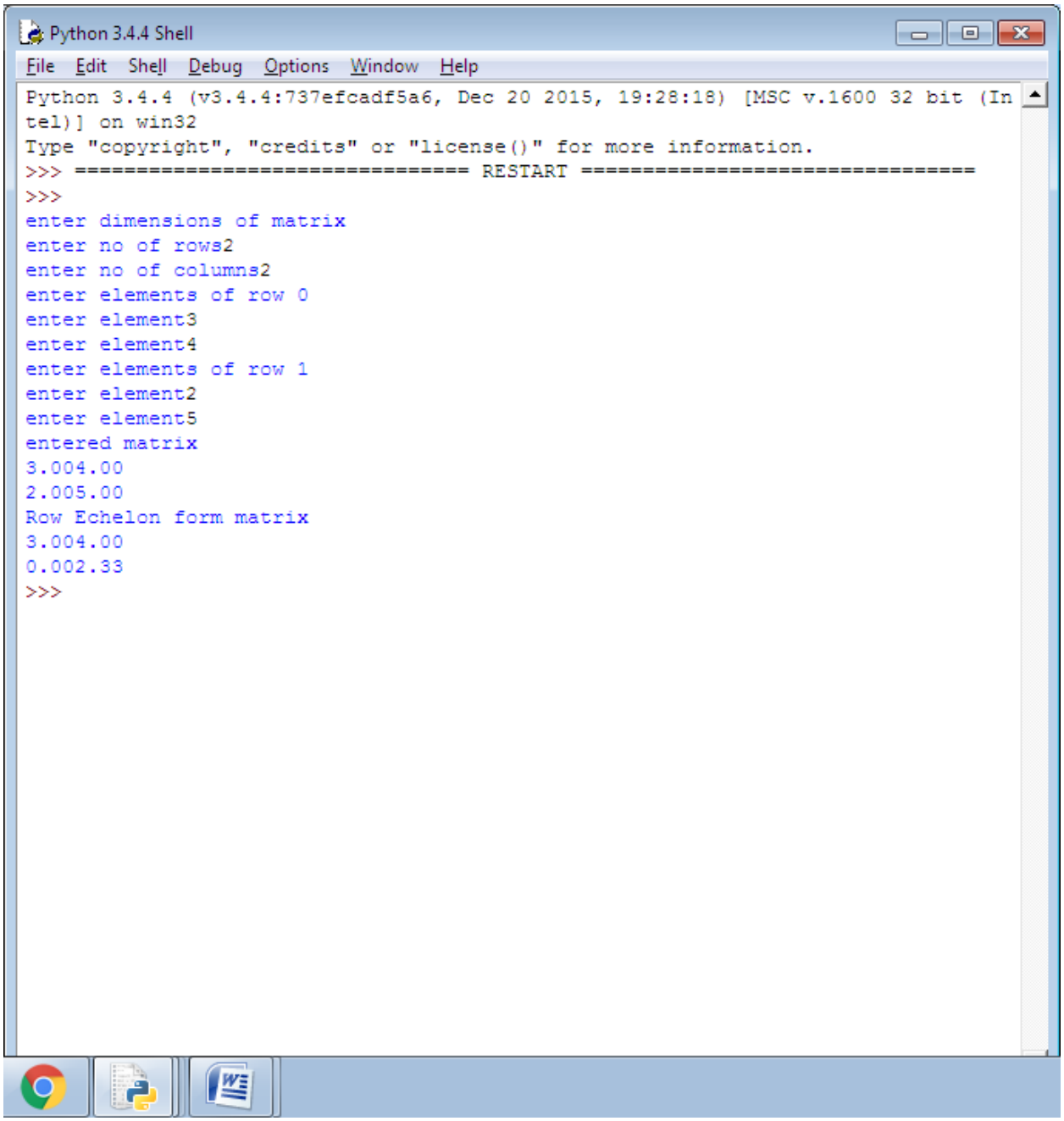
Write a program to convert a matrix into its row echelon form.

```
#multiply a list by number
def row_mult(row,num):
    return[x*num  for x in row]
#subtract one row from another
def row_sub(row_left,row_right):
    return[a-b  for (a,b) in zip(row_left,row_right)]
#row_echelon ::Matrix->Matrix
def row_echelon(mtx):
    temp_mtx=list(mtx)
    def echelonify(rw,col):
        for i,row in enumerate(temp_mtx[(col+1):]):
            i+=1
            if rw[col]==0:continue

temp_mtx[i+col]=row_sub(row,row_mult(rw,row[col]/rw[col]))
    for i in range(len(mtx)):
```

```
        active_row=temp_mtx[i]
        echelonify(active_row,i)
        #flatten out values very close to zero
    temp_mtx=[
        [(0 if(0.0000000001>x>-0.0000000001)else  x)
        for x in row]
        for row in temp_mtx
        ]
    return  temp_mtx
#accept input of square matrix
print('enter  dimensions of matrix')
r=int(input('enter  no of rows'))
c=int(input('enter  no of columns'))
mtx=[]
mtx_result=[]
for i in range(r):
    mtx.append([])
    print('enter  elements of row',i)
```

```
    for j in range(c):
        n=float(input('enter  element'))
        mtx[i].append(n)
print('entered  matrix')
for row in mtx:
    print(".join(("{0:.2f}".format(x)for  x in row)))
mtx_result=row_echelon(mtx)
print('Row  Echelon form matrix')
for row in mtx_result:
    print(".join(("{0:.2f}".format(x)for  x in row)))
```

Output:



PRACTICAL-8(a):

Write a program to do the following-

i) **Enter a positive number N and find numbers a and b such that a^2-b^2=N.**

ii) **Find the gcd of two numbers using Euclid's algorithm.**

GCD

```
def gcd(a,b):
    if b>a:
        return gcd(b,a)
    if a%b==0:
        return gcd(b,a%b)
```

Output:

PRACTICAL-8(b):

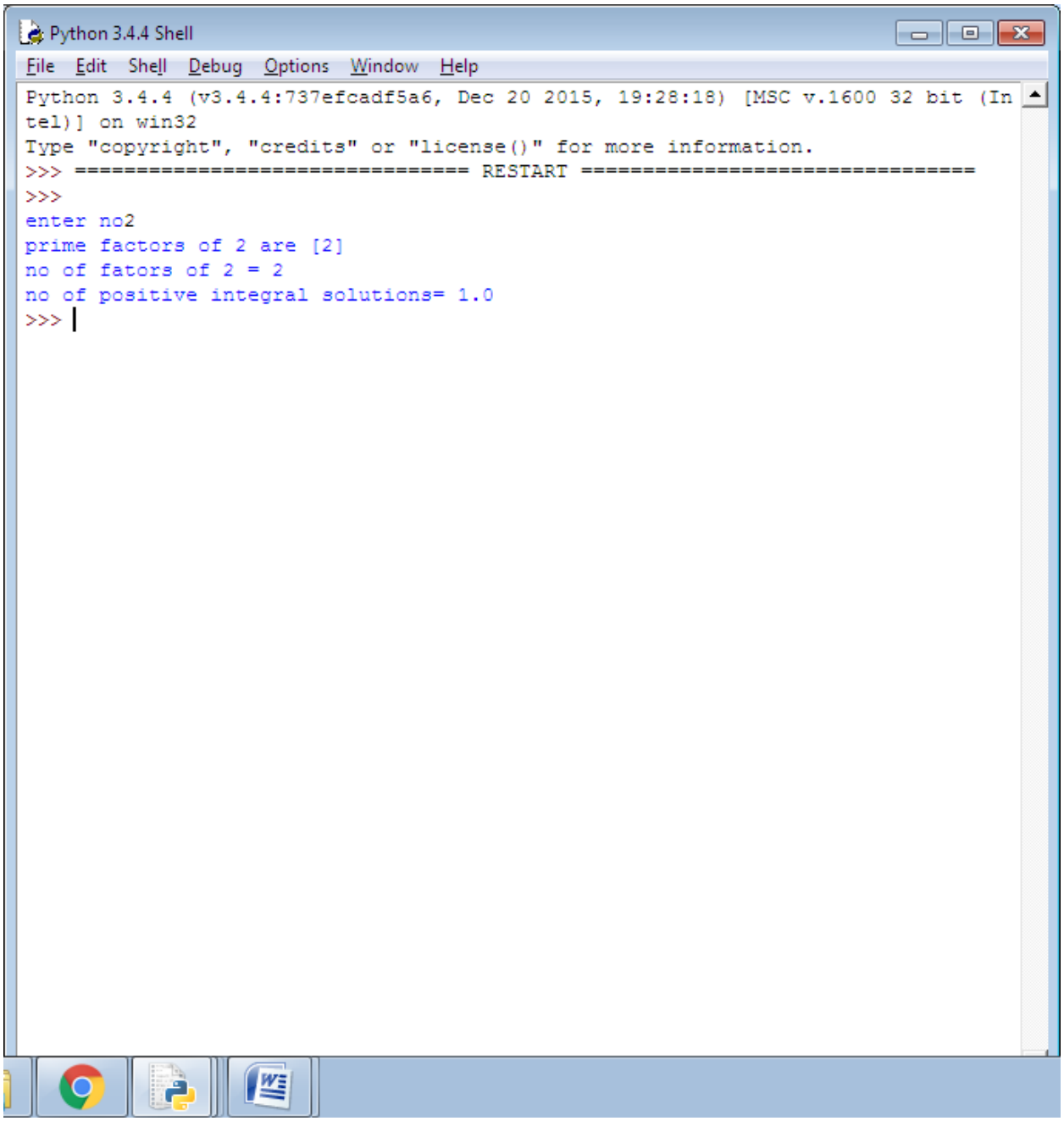
Integral solution.

#Find the no of positive integral solutions for a^2-b^2=N

```
from math import *
pf=[]
n=int(input('enter no'))
x=n
while n%2==0:
    pf.append(2)
    n=n/2
i=3
while i<=sqrt(n):
    while n%i==0:
        pf.append(i)
        n=n/i
    i=i+2
if n>2:
    pf.append(n)
```

```
print('prime factors 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 fators of',x,'=',nf)
print('no of positive integral solutions=',nf/2)
```

Output:



PRATICAL-9:

Write a program to do the following-

i) **Enter a vector b and find the projection of b orthogonal to a given vector u.**

ii) **Find the projection of b orthogonal to a set of given vectors.**

```
def dot(x,y):
  return sum([x[i]*y[i] for i in range(len(x))])
def scalar(a,v):
  return[a*v[i] for i in range(len(v))]
def sub(u,v):
  return[u[i]-v[i] for i in range(len(v))]
def project_along(b,v):
  sigma=(dot(b,v)/dot(v,v)) if dot(v,v) !=0 else 0
  return scalar(sigma,v)
def project_orthogonal(b,v):return
sub(b,project_along(b,v))

def project_orthogonalvectorset(b,s):
  for i in range(len(s)):
```

```
        v=s[i]
        b=project_orthogonal(b,v)
    return(b)
```

Output: