```
#PRACTICAL-1 WAP which demonstrate the following
#a)addition of two complex number
'''a=4+2j
b=3-5j
print("Addition of two complex number is:",a+b)

# o/p
 Addition of two complex number is: (7-3j)'''


#b)displaying the conjugate the complex number
'''a=4+2j
print(a.conjugate())

# o/p

(4-2j)'''


#c)plotting the a set of complex number
'''import matplotlib.pyplot as plt
x=2+2j
a=[-2+4j,-1+2j,0+2j,-1+4j,1+4j]
x=[x.real for x in a]
y=[x.imag for x in a]
plt.scatter(x,y,color="red")
plt.show()'''

#d)creating a new plot by rotating a even number by indegree 90,180,270
degreeand also by scaling by a number a=1/2,a=1/3,a=2 etc


'''import matplotlib.pyplot as plt
x=2+4j
z=1j
plt.scatter(x.real,x.imag,color='red')
c=x*2
plt.scatter(c.real,c.imag,color='blue')
plt.show()'''



#e) rotation by 180 degree
'''import matplotlib.pyplot as plt
x=2+4j
plt.scatter(x.real,x.imag,color='red')
plt.scatter(-1*x.real,-1*x.imag,color='green')
plt.show()'''



#f)rotation by 270 degree
'''import matplotlib.pyplot as plt
x=2+4j
z=-1j
plt.scatter(x.real,x.imag,color='red')
c=x*z
plt.scatter(c.real,c.imag,color='green')
plt.show()'''

#g)scaling by a=1/2,a=1/3,a=2
'''import matplotlib.pyplot as plt
x=2+4j
scale=0.5
```

```
scale1=0.33
scale2=2
plt.scatter(x.real,x.imag,color='red')
c=scale*x
d=scale1*x
e=scale2*x
plt.scatter(c.real,c.imag,color='green')
plt.scatter(d.real,d.imag,color='blue')
plt.scatter(e.real,e.imag,color='black')
plt.show()'''

#PRACTICAL-2 WAP which demonstrate the following
#a)enter a vector u as a list
#b)enter another vector v as list
#c)find the factor au+vb for different values of a&b
#d)find the dot product of u&v
'''import numpy as np
u=np.array([3,4,5])
v=np.array([1,2,7])
print("vector u",u)
print("vector v",v)
a=int(input("enter the value for a:"))
b=int(input("enter the value for b:"))
d=a*u+b*v
p=np.dot(u,v)
print("vector au+bv",d)
print("Dot product of u & v",p)

# o/p

vector u [3 4 5]
vector v [1 2 7]
enter the value for a:4
enter the value for b:5
vector au+bv [17 26 55]
Dot product of u & v 46
'''



#PRACTICAL-3
#a)basic matrix operation
'''import numpy as np
m1 = np.array([[1, 3, 4], [8, 5, 6]])
m2 = np.array([[8, 6, 9], [9, 0, 6]])
print("Add Matrix :")
a = np.add(m1, m2)
print(a)
print("Subtract Matrix")
b = np.subtract(m2, m1)
print(b)
x = np.array([[1, 7, 5], [4, 5, 3], [3, 2, 1]])
y = np.array([[6, 7, 7], [2, 3, 1], [2, 2, 3]])
t = np.array([[0, 0, 0], [0, 0, 0], [0, 0, 0]])
print("Multiplication Matrix")
for i in range(len(x)):
    for j in range(len(y[0])):
        for k in range(len(y)):
            t[i][j] += x[i][k] * y[k][j]
for r in t:
    print(r)


#  o/p
```

```
Add Matrix :
[[ 9  9 13]
 [17  5 12]]
Subtract Matrix
[[ 7  3  5]
 [ 1 -5  0]]
Multiplication Matrix
[30 38 29]
[40 49 42]
[24 29 26]'''

#b) check if matrix is invertiable & if yes then find inverse of the matrix
'''import numpy as np
m=np.array([[1,2,1],[2,1,0],[3,0,2]])
print("matrix is:",m)
c=np.linalg.det(m)
print("determinant is:",c)
if(c!=0):
 minv=np.linalg.inv(m)
 print("inverse of matrix is:",minv)
else:
 print("matrix is not invertible")

# o/p
matrix is: [[1 2 1]
 [2 1 0]
 [3 0 2]]
determinant is: -8.999999999999998
inverse of matrix is: [[-0.22222222  0.44444444  0.11111111]
 [ 0.44444444  0.11111111 -0.22222222]
 [ 0.33333333 -0.66666667  0.33333333]]'''

#prac 4
#a)WAP toconvert a matrix into its row-echelon form
'''from sympy import *
m=Matrix([[1,0,1,3],[2,3,4,7],[-1,-3,-3,-4]])
print("matrix:{}".format(m))
M_rref=m.rref()
print("the Row echelon form of matrix m and the pivot columns:
{}".format(M_rref))

# o/p
the Row echelon form of matrix m and the pivot columns:(Matrix([
[1, 0,   1,   3],
[0, 1, 2/3, 1/3],
[0, 0,   0,   0]]), (0, 1))'''

#b)WAP to find rank of a matrix
'''import numpy as np
my_matrix = np.array([[1,2,1],[3,4,7],[3,6,3]])
print("Matrix")
for row in my_matrix:
 print(row)
 rank= np.linalg.matrix_rank(my_matrix)
print("Rank of the given Matrix is :",rank)

# o/p
Matrix
[1 2 1]
[3 4 7]
[3 6 3]
Rank of the given Matrix is : 2'''
```

```
#prac5
'''import numpy as np
def opprojection(of_vec,on_vec):
 v1=np.array(of_vec)
 v2=np.array(on_vec)
 scal=np.dot(v1,v2)/np.dot(v2,v2)
 vec=scal*v2
 return round(scal,10),np.around(vec,decimals=10)
print(opprojection([4.0,4.0],[1.0,1.0]))
print(opprojection([4.0,4.0],[8.0,2.0]))


# o/p
(np.float64(4.0), array([4., 4.]))
(np.float64(0.5882352941), array([4.70588235, 1.17647059]))'''


#prac.6)
#WAP to calculate eigen value and order 2 and order 3
#A)eigen value or eigen vector of order 2*2 matrix
'''import numpy as np
A=np.matrix("3 -2; 1 0")
print("A\n",A)
print("Eigenvalues",np.linalg.eigvals(A))
eigenvalues,eigenvectors=np.linalg.eig(A)
print("first tuple of eig",eigenvalues)
print("second tuple of eig\n",eigenvectors)
for i in range(len(eigenvalues)):
 print("left",np.dot(A,eigenvectors[:,i]))
 print(...)

#  o/p
A
 [[ 3 -2]
 [ 1  0]]
Eigenvalues [2. 1.]
first tuple of eig [2. 1.]
second tuple of eig
 [[0.89442719 0.70710678]
 [0.4472136  0.70710678]]
left [[1.78885438]
 [0.89442719]]
Ellipsis
left [[0.70710678]
 [0.70710678]]
Ellipsis '''

# B)eigen value and eigen vector of order #*3 matrix
'''import numpy as np
A=np.matrix("2 0 0;0 3 4;0 4 9")
print("A\n",A)
print("Eigenvalues",np.linalg.eigvals(A))
eigenvalues,eigenvectors=np.linalg.eig(A)
print("first tuple of eig",eigenvalues)
print("second tuple of eig\n",eigenvectors)
for i in range(len(eigenvalues)):
 print("left",np.dot(A,eigenvectors[:,i]))
 print(...)

#  o/p
A
 [[2 0 0]
 [0 3 4]
 [0 4 9]]
```

```
Eigenvalues [11.  1.  2.]
first tuple of eig [11.  1.  2.]
second tuple of eig
 [[ 0.          0.          1.        ]
 [ 0.4472136   0.89442719  0.        ]
 [ 0.89442719 -0.4472136   0.        ]]
left [[0.        ]
 [4.91934955]
 [9.8386991 ]]
Ellipsis
left [[ 0.        ]
 [ 0.89442719]
 [-0.4472136 ]]
Ellipsis
left [[2.]
 [0.]
 [0.]]
Ellipsis'''


# prac.7)
#implement googles page rank algorithm
'''import networkx as nx
import pylab as plt
D=nx.DiGraph()
D.add_weighted_edges_from([('A','B',1),('C','A',1),('B','C',1)])
print(nx.pagerank(D))
nx.draw(D,with_labels=True)
plt.show()

#  o/p
{'A': 0.3333333333333333, 'B': 0.3333333333333333, 'C': 0.3333333333333333}'''


# prac-8

'''rows = int(input("enter number of rows: "))
columns = int(input("enter number of columns: "))
m = []

for i in range(rows):
    m.append([])
    for j in range(columns):
        e = int(input("enter element: "))
        m[i].append(e)

def scal(m):
    a = int(input('enter value of a: '))
    unew = []
    for i in range(rows):
        unew.append([])
        for j in range(columns):
            unew[i].append([m[i][j] * a])
    for i in range(rows):
        print(unew[i])

def tran(m):
    rmatrix = []
    rmatrix = [[0] * rows for i in range(columns)]
    for i in range(len(m)):
        for j in range(len(m[0])):
            rmatrix[j][i] = m[i][j]
    for r in rmatrix:
        print(r)
```

```
def dis(m):
    for i in range(rows):
        print("row", [i])
        print(m[i])
    for i in range(len(m[0])):
        print("column", [i])
        for j in range(len(m)):
            print("[", m[j][i], "]")

ch = True
while ch:
    print("\n1.Display matrix\n2.Display rows and columns\n3.Scalar
multiplication\n4.Matrix Transpose\n5.Exit")
    ch = int(input("enter choice: "))
    if ch == 1:
        for i in range(rows):
            print(m[i])
    elif ch == 2:
        dis(m)
    elif ch == 3:
        scal(m)
    elif ch == 4:
        tran(m)
    elif ch == 5:
        print("Exit")
        ch = False
    else:
        print("\nInvalid Choice")'''
'''
# o/p
enter number of rows: 2
enter number of columns: 2
enter element: 4
enter element: 4
enter element: 5
enter element: 6

1.Display matrix
2.Display rows and columns
3.Scalar multiplication
4.Matrix Transpose
5.Exit
enter choice: 1
[4, 4]
[5, 6]

1.Display matrix
2.Display rows and columns
3.Scalar multiplication
4.Matrix Transpose
5.Exit
enter choice: 2
row [0]
[4, 4]
row [1]
[5, 6]
column [0]
[ 4 ]
[ 5 ]
column [1]
[ 4 ]
[ 6 ]
```

```
1.Display matrix
2.Display rows and columns
3.Scalar multiplication
4.Matrix Transpose
5.Exit
enter choice: 3
enter value of a: 3
[[12], [12]]
[[15], [18]]

1.Display matrix
2.Display rows and columns
3.Scalar multiplication
4.Matrix Transpose
5.Exit
enter choice: 4
[4, 5]
[4, 6]

1.Display matrix
2.Display rows and columns
3.Scalar multiplication
4.Matrix Transpose
5.Exit
enter choice: 4
[4, 5]
[4, 6]

1.Display matrix
2.Display rows and columns
3.Scalar multiplication
4.Matrix Transpose
5.Exit
enter choice: 5
Exit'''
```