```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 24 11:14:32 2023

@author: devna
"""
#%%
""" Loading input data """
import numpy as np

# backsllash is used to continue the same code statement in next line

Num_Nodes,Num_elemt,Num_Mats,Prob_type,Thickness,Num_Load_BC,\
    Num_Disp_BC= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM assignment/Input.txt')

# converting data type from float 64 to int 32

Num_Nodes=Num_Nodes.astype(int)
Num_elemt= Num_elemt.astype(int)
Num_Mats= Num_Mats.astype(int)
Num_Load_BC= Num_Load_BC.astype(int)
Num_Disp_BC= Num_Disp_BC.astype(int)


COORD= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM assignment/COORD.txt')
MAT= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM assignment/MAT.txt')
NCA= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM assignment/NCA.txt').astype(np.int32)
LOAD_BC= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM assignment/LOAD_BC.txt')
DISP_BC= np.loadtxt('C:/Users/devna/Documents/Python Scripts/Final FEM assignment/DISP_BC.txt')

""" Initialization """
Dof_pn= 2
Total_Dof= Num_Nodes*Dof_pn
# initializing global stiffness matrix and load array
GSTIFF= np.zeros((Total_Dof,Total_Dof))
F= np.zeros(Total_Dof)

#%%
""" Geometry """
import matplotlib.pyplot as plt
for elemt in range(1,Num_elemt+1,1):
    N1= NCA[elemt,1]
    N2= NCA[elemt,2]
    N3= NCA[elemt,3]

    X1N1= COORD[N1,1]
    X2N1= COORD[N1,2]
    X1N2= COORD[N2,1]
    X2N2= COORD[N2,2]
    X1N3= COORD[N3,1]
    X2N3= COORD[N3,2]

    X= [X1N1, X1N2, X1N3, X1N1]
    Y= [X2N1, X2N2, X2N3, X2N1]

    MAT_NUM= NCA[elemt,4]

    CGX= (X1N1+X1N2+X1N3)/3.0
    CGY= (X2N1+X2N2+X2N3)/3.0

    if MAT_NUM == 1:
        plt.fill(X,Y, color = 'red')
```

```
            E= MAT[1,1]
            PR= MAT[1,2]
        elif MAT_NUM == 2:
            plt.fill(X,Y, color = 'yellow')
            E= MAT[2,1]
            PR= MAT[2,2]
        plt.plot(X, Y)
        plt.scatter(X, Y)
        plt.text(CGX, CGY, str(elemt),bbox=dict(boxstyle="round"))

    # B matrix
        two_delta_matrix= np.array([[1,X1N1,X2N1],[1,X1N2,X2N2],[1,X1N3,X2N3]])
        two_delta= np.linalg.det(two_delta_matrix)

        B1= X2N2-X2N3
        B2= X2N3-X2N1
        B3= X2N1-X2N2
        G1= X1N3-X1N2
        G2= X1N1-X1N3
        G3= X1N2-X1N1

    # Initializing B matrix
        B= np.zeros((3,6))
        B[0,0]= B1
        B[0,2]= B2
        B[0,4]= B3
        B[1,1]= G1
        B[1,3]= G2
        B[1,5]= G3
        B[2,0]= G1
        B[2,1]= B1
        B[2,2]= G2
        B[2,3]= B2
        B[2,4]= G3
        B[2,5]= B3

        B= B/ two_delta

    # D matrix
        D= np.zeros((3,3))
        if Prob_type== 21.0:
            CONST= E/(1-PR**2)
            D[0,0]= 1
            D[0,1]= PR
            D[1,0]= PR
            D[1,1]= 1
            D[2,2]= (1-PR)/2.0
            D= D*CONST
        elif Prob_type== 22.0:
            CONST= E/((1+PR)*(1-2*PR))
            D[0,0]= 1-PR
            D[0,1]= PR
            D[1,0]= PR
            D[1,1]= 1-PR
            D[2,2]= (1-2*PR)/2.0
            D= D*CONST
            Thickness= 1.0

            # problem type can be defined with number of elif condition
            # problem type: plane stress(Prob_type=21), plane strain(Prob_type=22)
            # anisotropy(Prob_type=23)

    # Element stiffness matrix
        ESTIFF= (B.T @ D @ B)*Thickness*two_delta/2.0

        # updating global stiffness matrix
        CN= [2*N1-2, 2*N1-1, 2*N2-2, 2*N2-1, 2*N3-2, 2*N3-1]
```

```
        CN_Index= np.array(6*CN).reshape((6,6))
        RN_Index= CN_Index.T

        GSTIFF[RN_Index,CN_Index]= GSTIFF[RN_Index,CN_Index] + ESTIFF

    #%%
    # load boundary condition
    for i in range(1,Num_Load_BC+1):
        Load_type = LOAD_BC[i,2]

        if Load_type== 1.0:
            N= LOAD_BC[i,1].astype(int)
            F[2*N-2]= F[2*N-2]+ LOAD_BC[i,3]

        elif Load_type== 2.0:
            N= LOAD_BC[i,1].astype(int)
            F[2*N-1]= F[2*N-1]+ LOAD_BC[i,4]

        elif Load_type== 12.0:
            N= LOAD_BC[i,1].astype(int)
            F[2*N-2]= F[2*N-2]+ LOAD_BC[i,3]
            F[2*N-1]= F[2*N-1]+ LOAD_BC[i,4]

    #%%
    # Displacement boundary condition
    GSTIFF_copy= GSTIFF.copy()
    for i in range(1,Num_Disp_BC+1):
        Disp_type= DISP_BC[i,2]

        if Disp_type== 1.0:
            N= DISP_BC[i,1].astype(int)
            F[2*N-2]= F[2*N-2]+ DISP_BC[i,3]*10000000000000000
            GSTIFF_copy[2*N-2, 2*N-2]= GSTIFF_copy[2*N-2, 2*N-2]+ 10000000000000000

        elif Disp_type== 2.0:
            N= DISP_BC[i,1].astype(int)
            F[2*N-1]= F[2*N-1]+ DISP_BC[i,4]*10000000000000000
            GSTIFF_copy[2*N-1, 2*N-1]= GSTIFF_copy[2*N-1, 2*N-1]+ 10000000000000000

        elif Disp_type== 12.0:
            N= DISP_BC[i,1].astype(int)
            F[2*N-2]= F[2*N-2]+ DISP_BC[i,3]*10000000000000000
            GSTIFF_copy[2*N-2, 2*N-2]= GSTIFF_copy[2*N-2, 2*N-2]+ 10000000000000000
            F[2*N-1]= F[2*N-1]+ DISP_BC[i,4]*10000000000000000
            GSTIFF_copy[2*N-1, 2*N-1]= GSTIFF_copy[2*N-1, 2*N-1]+ 10000000000000000

    #%%
    # Det_GSTIFF= np.linalg.det(GSTIFF)
    # Det_GSTIFF_copy = np.linalg.det(GSTIFF_copy)
    Disp= np.linalg.solve(GSTIFF_copy,F)
    Disp = Disp.reshape(-1,2)
    print(Disp)
```