```
import numpy as np
import sys
n = int(input('Enter number of unknowns: '))
a = np.zeros((n, n+1))
x = np.zeros(n)
print('Enter Augmented Matrix Coefficients:')
for i in range(n):
    for j in range(n+1):
        a[i][j] = float(input('a['+str(i)+']['
+ str(j)+']='))
for i in range(n):
    if a[i][i] == 0.0:
        sys.exit('Divide by zero detected!')
    for j in range(i+1, n):
        ratio = a[j][i]/a[i][i]
        for k in range(n+1):
            a[j][k] = a[j][k] - ratio * a[i][k]
x[n-1] = a[n-1][n]/a[n-1][n-1]
for i in range(n-2, -1, -1):
    x[i] = a[i][n]
    for j in range(i+1, n):
        x[i] = x[i] - a[i][j]*x[j]
    x[i] = x[i]/a[i][i]
print('\nRequired solution is: ')
for i in range(n):
    print('X%d = %0.2f' % (i, x[i]), end='\t')
```

```
def f1(x, y, z): return (7.85-0.1*y+0.2*z)/3
def f2(x, y, z): return (-19.3-0.1*x+0.3*z)/7
def f3(x, y, z): return (71.4-0.3*x+0.2*y)/10
x0 = 0
y0 = 0
z0 = 0
count = 1
e = float(input('Enter tolerable error: '))
print('\nCount\tx\ty\tz\n')
condition = True
while condition:
    x1 = f1(x0, y0, z0)
    y1 = f2(x0, y0, z0)
    z1 = f3(x0, y0, z0)
    print('%d\t%0.4f\t%0.4f\t%0.4f\n' % (count,
x1, y1, z1))
    e1 = abs(x0-x1)
    e2 = abs(y0-y1)
    e3 = abs(z0-z1)
    count += 1
    x0 = x1
    y0 = y1
    z0 = z1
    condition = e1 > e and e2 > e and e3 > e
print('\nSolution: x=%0.3f, y=%0.3f and z =
%0.3f\n' % (x1, y1, z1))
```

```
def f1(x, y, z): return (7.85-0.1*y+0.2*z)/3
def f2(x, y, z): return (-19.3-0.1*x+0.3*z)/7
def f3(x, y, z): return (71.4-0.3*x+0.2*y)/10
x0 = 0
y0 = 0
z0 = 0
count = 1
e = float(input('Enter tolerable error: '))
print('\nCount\tx\ty\tz\n')
condition = True
while condition:
    x1 = f1(x0, y0, z0)
    y1 = f2(x1, y0, z0)
    z1 = f3(x1, y1, z0)
    print('%d\t%0.4f\t%0.4f\t%0.4f\n' % (count,
x1, y1, z1))
    e1 = abs(x0-x1)
    e2 = abs(y0-y1)
    e3 = abs(z0-z1)
    count += 1
    x0 = x1
    y0 = y1
    z0 = z1
    condition = e1 > e and e2 > e and e3 > e
print('\nSolution: x=%0.3f, y=%0.3f and z =
%0.3f\n' % (x1, y1, z1))
```

```
MAX = 100
def luDecomposition(mat, n):
    lower = [[0 for x in range(n)]
             for y in range(n)]
    upper = [[0 for x in range(n)]
             for y in range(n)]
    for i in range(n):
        for k in range(i, n):
            sum = 0.0
            for j in range(i):
                sum += (lower[i][j] * upper[j][k])
            upper[i][k] = mat[i][k] - sum
        for k in range(i, n):
            if (i == k):
                lower[i][i] = 1  # Diagonal as 1
            else:
                sum = 0
                for j in range(i):
                    sum += (lower[k][j] * upper[j][i])
                lower[k][i] = int((mat[k][i] - sum) /
                                  upper[i][i])
    print("Lower Triangular\t\tUpper Triangular")
    for i in range(n):
        for j in range(n):
            print(lower[i][j], end="\t")
        print("", end="\t")
        for j in range(n):
            print(upper[i][j], end="\t")
        print("")
mat = [[4, -2, -3, 6],
       [-6, 7, 6.5, -6],
       [1, 7.5, 6.25, 5.5],
       [-12, 22, 15.5, -1]]
luDecomposition(mat, 4)
```