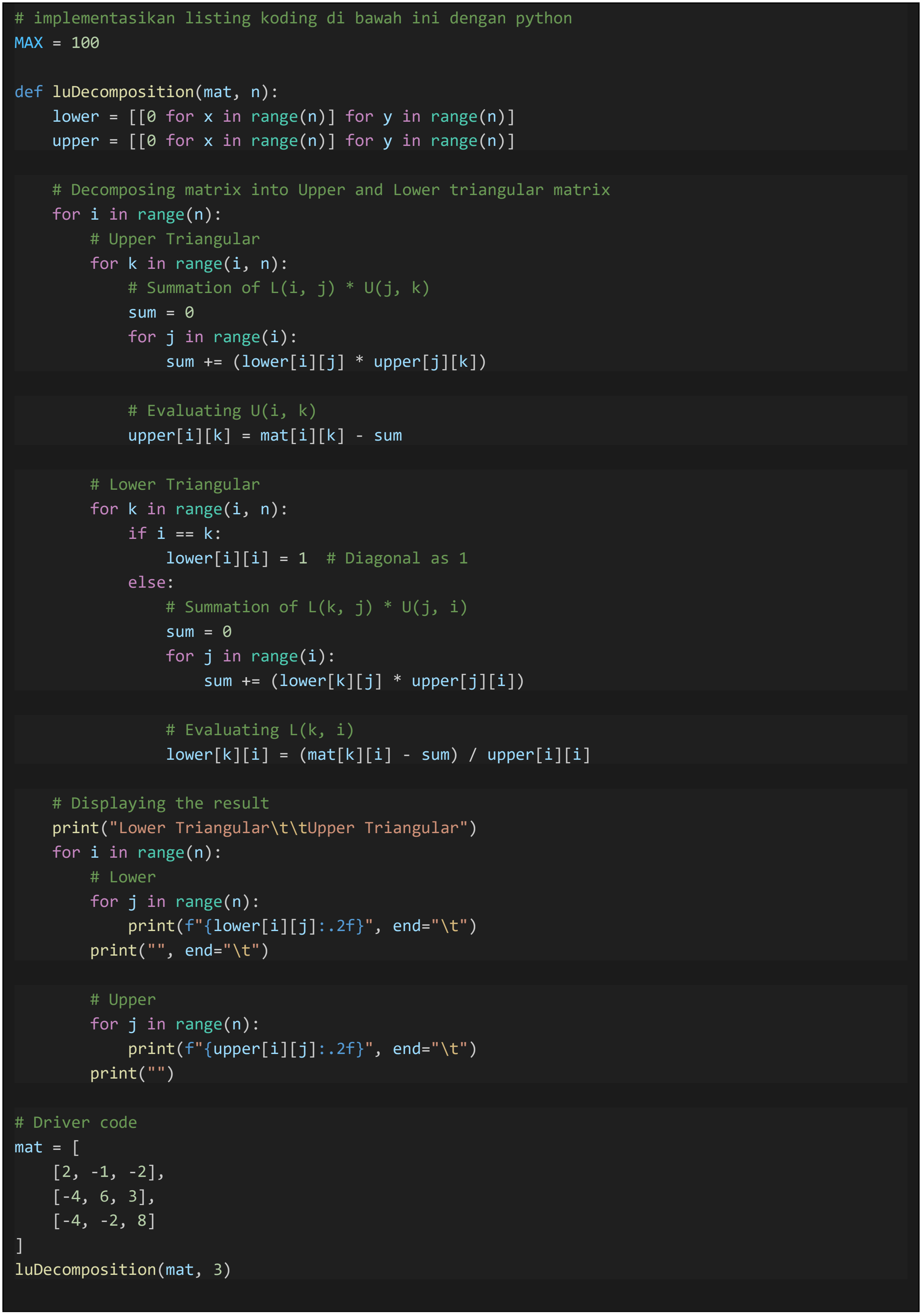
Nama   : Iyan Zuli Armanda
NIM   : 23051204165
Kelas   : TI E 23

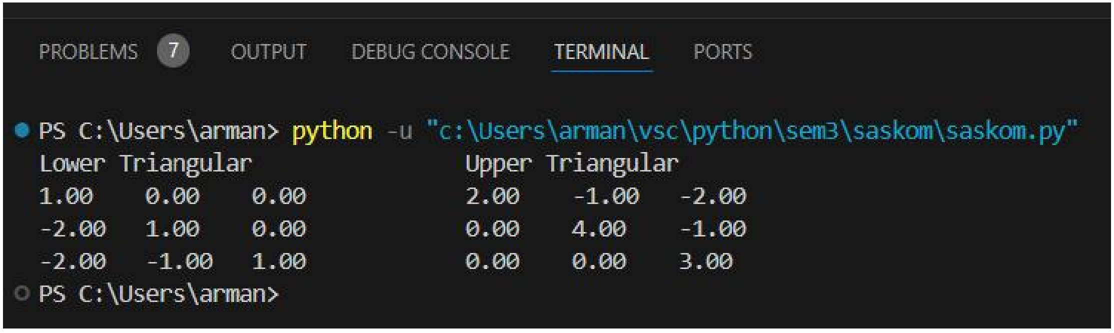
Praktik LU Decomposition

Implemenentasikan listing koding di bawah ini dg python!



```python
# implementasikan listing koding di bawah ini dengan python
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)]

    # Decomposing matrix into Upper and Lower triangular matrix
    for i in range(n):
        # Upper Triangular
        for k in range(i, n):
            # Summation of L(i, j) * U(j, k)
            sum = 0
            for j in range(i):
                sum += (lower[i][j] * upper[j][k])

            # Evaluating U(i, k)
            upper[i][k] = mat[i][k] - sum

        # Lower Triangular
        for k in range(i, n):
            if i == k:
                lower[i][i] = 1  # Diagonal as 1
            else:
                # Summation of L(k, j) * U(j, i)
                sum = 0
                for j in range(i):
                    sum += (lower[k][j] * upper[j][i])

                # Evaluating L(k, i)
                lower[k][i] = (mat[k][i] - sum) / upper[i][i]

    # Displaying the result
    print("Lower Triangular\t\tUpper Triangular")
    for i in range(n):
        # Lower
        for j in range(n):
            print(f"{lower[i][j]:.2f}", end="\t")
        print("", end="\t")

        # Upper
        for j in range(n):
            print(f"{upper[i][j]:.2f}", end="\t")
        print("")

# Driver code
mat = [
    [2, -1, -2],
    [-4, 6, 3],
    [-4, -2, 8]
]
luDecomposition(mat, 3)
```

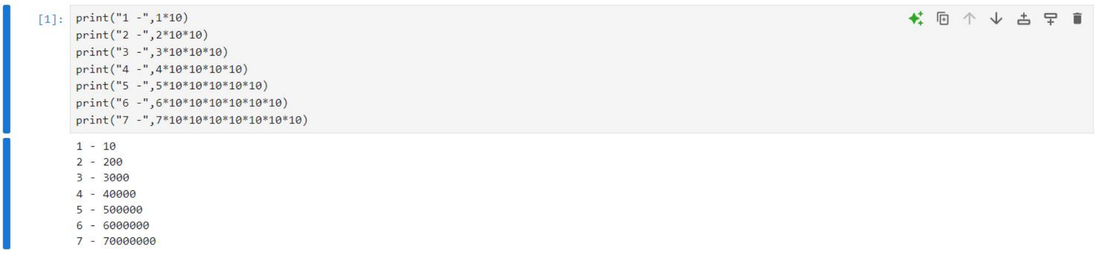
Output nya :



```
PS C:\Users\arman> python -u "c:\Users\arman\vsc\python\sem3\saskom\saskom.py"
Lower Triangular                  Upper Triangular
1.00     0.00     0.00            2.00     -1.00    -2.00
-2.00    1.00     0.00            0.00     4.00     -1.00
-2.00    -1.00    1.00            0.00     0.00     3.00
PS C:\Users\arman>
```

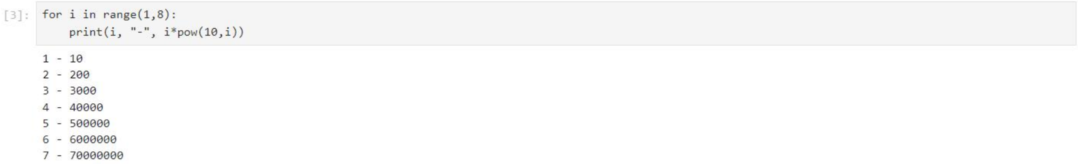
Kode itu sudah selesai, hasilnya ialah program mencetak matriks segitiga bawah dan atas setelah melakukan dekomposisi.

Praktik Eliminasi Gauss

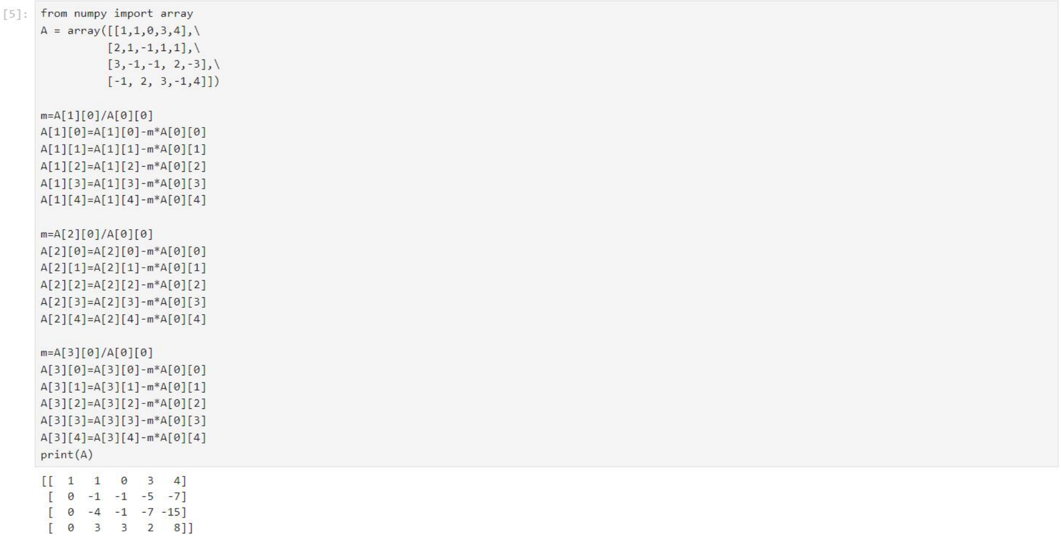


```
[1]: print("1 -",1*10)
     print("2 -",2*10*10)
     print("3 -",3*10*10*10)
     print("4 -",4*10*10*10*10)
     print("5 -",5*10*10*10*10*10)
     print("6 -",6*10*10*10*10*10*10)
     print("7 -",7*10*10*10*10*10*10*10)

     1 - 10
     2 - 200
     3 - 3000
     4 - 40000
     5 - 500000
     6 - 6000000
     7 - 70000000
```

Hasil Optimalisasi nya:

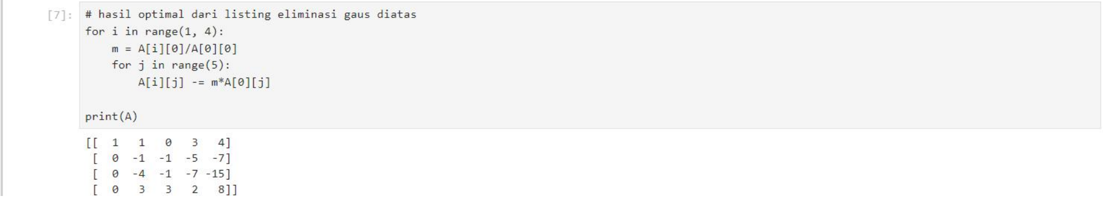


```
[3]: for i in range(1,8):
         print(i, "-", i*pow(10,i))

     1 - 10
     2 - 200
     3 - 3000
     4 - 40000
     5 - 500000
     6 - 6000000
     7 - 70000000
```

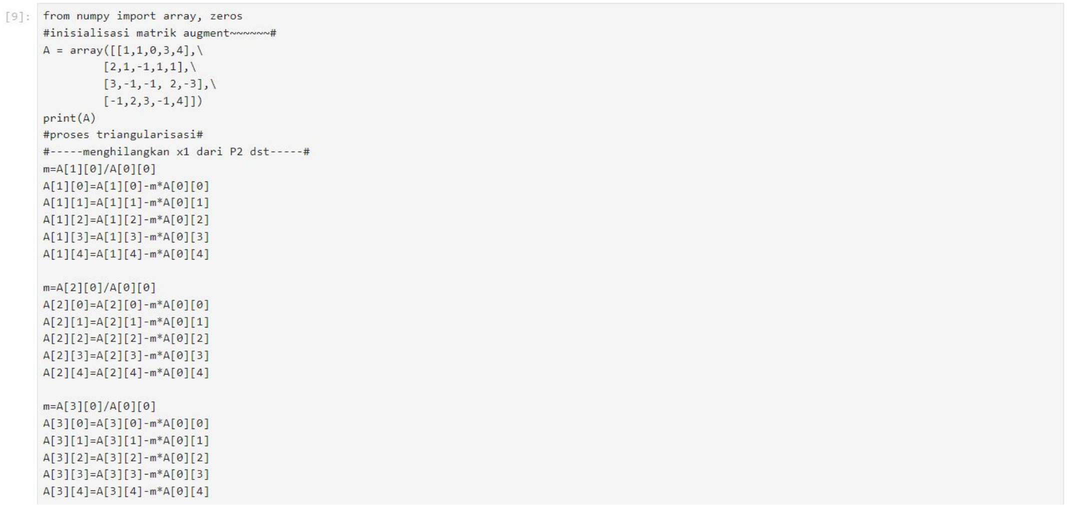


```
[5]: from numpy import array
     A = array([[1,1,0,3,4],\
               [2,1,-1,1,1],\
               [3,-1,-1, 2,-3],\
               [-1, 2, 3,-1,4]])

     m=A[1][0]/A[0][0]
     A[1][0]=A[1][0]-m*A[0][0]
     A[1][1]=A[1][1]-m*A[0][1]
     A[1][2]=A[1][2]-m*A[0][2]
     A[1][3]=A[1][3]-m*A[0][3]
     A[1][4]=A[1][4]-m*A[0][4]

     m=A[2][0]/A[0][0]
     A[2][0]=A[2][0]-m*A[0][0]
     A[2][1]=A[2][1]-m*A[0][1]
     A[2][2]=A[2][2]-m*A[0][2]
     A[2][3]=A[2][3]-m*A[0][3]
     A[2][4]=A[2][4]-m*A[0][4]

     m=A[3][0]/A[0][0]
     A[3][0]=A[3][0]-m*A[0][0]
     A[3][1]=A[3][1]-m*A[0][1]
     A[3][2]=A[3][2]-m*A[0][2]
     A[3][3]=A[3][3]-m*A[0][3]
     A[3][4]=A[3][4]-m*A[0][4]
     print(A)

     [[  1   1   0   3   4]
      [  0  -1  -1  -5  -7]
      [  0  -4  -1  -7 -15]
      [  0   3   3   2   8]]
```

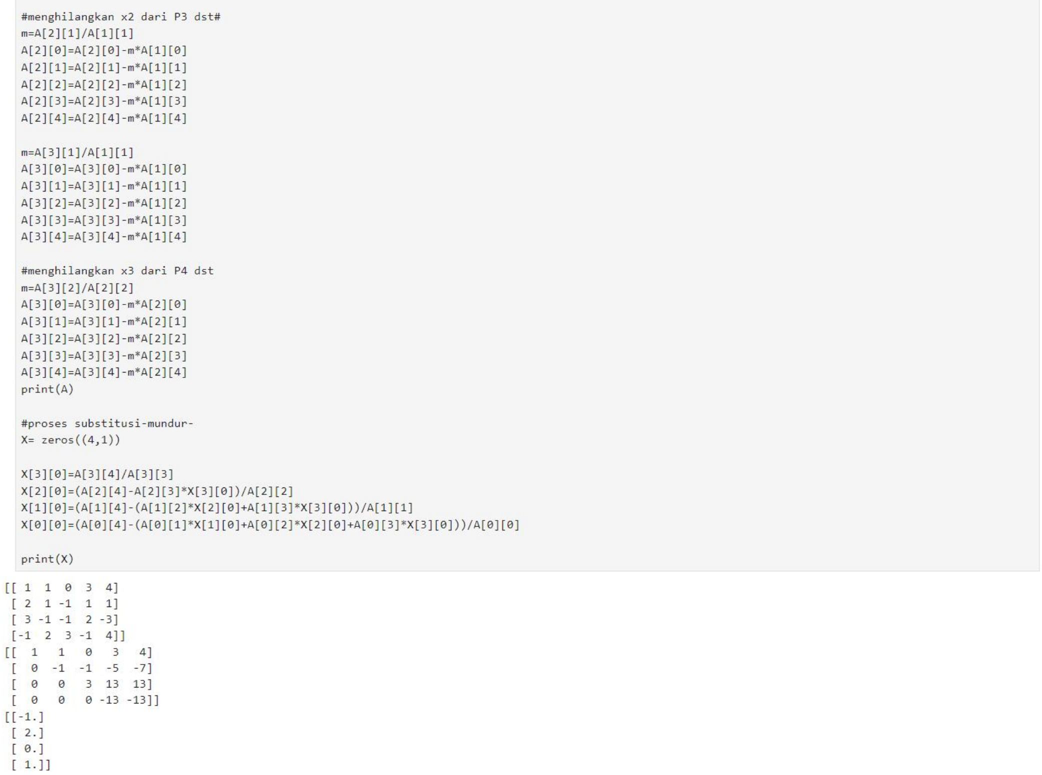
Hasil optimalisasinya :



```
[7]: # hasil optimal dari listing eliminasi gaus diatas
     for i in range(1, 4):
         m = A[i][0]/A[0][0]
         for j in range(5):
             A[i][j] -= m*A[0][j]

     print(A)

     [[  1   1   0   3   4]
      [  0  -1  -1  -5  -7]
      [  0  -4  -1  -7 -15]
      [  0   3   3   2   8]]
```



```
[9]: from numpy import array, zeros
     #inisialisasi matrik augment~~~~~~#
     A = array([[1,1,0,3,4],\
              [2,1,-1,1,1],\
              [3,-1,-1, 2,-3],\
              [-1,2,3,-1,4]])
     print(A)
     #proses triangularisasi#
     #-----menghilangkan x1 dari P2 dst-----#
     m=A[1][0]/A[0][0]
     A[1][0]=A[1][0]-m*A[0][0]
     A[1][1]=A[1][1]-m*A[0][1]
     A[1][2]=A[1][2]-m*A[0][2]
     A[1][3]=A[1][3]-m*A[0][3]
     A[1][4]=A[1][4]-m*A[0][4]

     m=A[2][0]/A[0][0]
     A[2][0]=A[2][0]-m*A[0][0]
     A[2][1]=A[2][1]-m*A[0][1]
     A[2][2]=A[2][2]-m*A[0][2]
     A[2][3]=A[2][3]-m*A[0][3]
     A[2][4]=A[2][4]-m*A[0][4]

     m=A[3][0]/A[0][0]
     A[3][0]=A[3][0]-m*A[0][0]
     A[3][1]=A[3][1]-m*A[0][1]
     A[3][2]=A[3][2]-m*A[0][2]
     A[3][3]=A[3][3]-m*A[0][3]
     A[3][4]=A[3][4]-m*A[0][4]
```



```
#menghilangkan x2 dari P3 dst#
m=A[2][1]/A[1][1]
A[2][0]=A[2][0]-m*A[1][0]
A[2][1]=A[2][1]-m*A[1][1]
A[2][2]=A[2][2]-m*A[1][2]
A[2][3]=A[2][3]-m*A[1][3]
A[2][4]=A[2][4]-m*A[1][4]

m=A[3][1]/A[1][1]
A[3][0]=A[3][0]-m*A[1][0]
A[3][1]=A[3][1]-m*A[1][1]
A[3][2]=A[3][2]-m*A[1][2]
A[3][3]=A[3][3]-m*A[1][3]
A[3][4]=A[3][4]-m*A[1][4]

#menghilangkan x3 dari P4 dst
m=A[3][2]/A[2][2]
A[3][0]=A[3][0]-m*A[2][0]
A[3][1]=A[3][1]-m*A[2][1]
A[3][2]=A[3][2]-m*A[2][2]
A[3][3]=A[3][3]-m*A[2][3]
A[3][4]=A[3][4]-m*A[2][4]
print(A)

#proses substitusi-mundur-
X= zeros((4,1))

X[3][0]=A[3][4]/A[3][3]
X[2][0]=(A[2][4]-A[2][3]*X[3][0])/A[2][2]
X[1][0]=(A[1][4]-(A[1][2]*X[2][0]+A[1][3]*X[3][0]))/A[1][1]
X[0][0]=(A[0][4]-(A[0][1]*X[1][0]+A[0][2]*X[2][0]+A[0][3]*X[3][0]))/A[0][0]

print(X)
```

```
[[ 1  1  0  3  4]
 [ 2  1 -1  1  1]
 [ 3 -1 -1  2 -3]
 [-1  2  3 -1  4]]
[[  1   1   0   3   4]
 [  0  -1  -1  -5  -7]
 [  0   0   3  13  13]
 [  0   0   0 -13 -13]]
[[-1.]
 [ 2.]
 [ 0.]
 [ 1.]]
```

Hasil optimalisasinya :

[15]:

```
# hasil optimalisasi dari coding eliminasi gauss diatas
# inisialisasi matrik augment
print(A)

# proses triangularisasi
for i in range(len(A)):
    for j in range(i+1, len(A)):
        m=A[j][i] / A[i][i]
        A[j]=A[j] - m*A[i]
print(A)

# proses substitusi mundur
n = len(A)
X = zeros((n, 1))

for i in range(n-1, -1, -1):
    X[i] = (A[i][-1] - sum(A[i][j] * X[j] for j in range(i+1, n))) / A[i][i]

print(X)
```

```
[[  1   1   0   3   4]
 [  0  -1  -1  -5  -7]
 [  0   0   3  13  13]
 [  0   0   0 -13 -13]]
[[  1   1   0   3   4]
 [  0  -1  -1  -5  -7]
 [  0   0   3  13  13]
 [  0   0   0 -13 -13]]
[[-1.]
 [ 2.]
 [ 0.]
 [ 1.]]
```