```
"""
#================================================================================
ECS - Laborator#03
Biblioteca NUMPY
#================================================================================
"""


"""
OBS#1: Biblioteca numpy este esențială pentru rezolvarea și optimizarea
sistemelor de ecuații liniare și trebuie importată sub forma:
        import numpy as np

Pentru a apela în mod optim funcțiile acestei biblioteci, matricele și
vectorii vor fi definiți folosind funcția np.array astfel:
    1. matricele se definesc ca np.array-uri aplicate unei liste de liste, e.g.
       A = np.array([[1, 0],
                     [0, 1]]);
    2. vectorii sunt definiți ca vectori coloană, i.e. matrice cu mai multe
       linii și o singura coloană:
       b = np.array([[1],
                     [0]]);

Este foarte important tipul elementelor np.array-urilor! Dacă matricea A și
vectorul b ar fi definite ca mai sus, Python le-ar interpreta ca matrice și
vector de elemente întregi.
Intrucât lucrăm cu elemente de tip float (virgulă mobilă), două definiții
echivalente și corecte ar fi următoarele:
    1. A = np.array([[1, 0],
                     [0, 1]]).astype(float);
    2. A = np.array([[1., 0.],
                     [0., 1.]]);
"""


"""
OBS#2: np.shape(A) -> returnează un tuple cu dimensiunile array-ului.
    Pentru un vector coloană:
        np.shape(b)[0] -> număr de linii ale lui b i.e. lungimea vectorului,
                          echivalent cu operația len(b);
    Pentru o matrice:
        np.shape(A)[0] -> număr de linii ale lui A;
        np.shape(A)[1] -> număr de coloane ale lui A;
"""


"""
OBS#3: Apelarea în np.array
    (i) Pentru un vector b:
        b[i] -> elementul de pe poziția i, i = 0, 1, ..., np.shape(b)[0] - 1;
```

```
        b[-i] -> elementul de pe poziția i numărând de la ultimul la primul,
                 e.g. b[-1] ultimul element, b[-2] penultimul etc.;
        b[start:stop] -> toate elementele de la poziția start (inclusiv) până
                         la poziția stop (exclusiv), slicing-ul functioneaza
                         la fel ca range-ul, e.g.
                         b[2:4] -> elementele 2 și 3, dar fără elementul 4;
        b[:stop] -> dacă nu specificam parametrul de start, este luat automat
                    primul element, i.e. b[0];
        b[start:] -> dacă nu specificăm parametrul de stop, este luat automat
                     ultimul element, i.e. b[np.shape(b)[0] - 1];

    (ii) Pentru o matrice A:
        A[i,j] -> elementul de pe linia i (0, 1, ..., np.shape(A)[0] - 1) și
                  coloana j (0, 1, ..., np.shape(A)[1] - 1);
        A[i,:] -> linia i (0, 1, ..., np.shape(A)[0] - 1)
        A[:,j] -> coloana j (0, 1, ..., np.shape(A)[1] - 1)
        A[i,:j] -> elemenetele de pe linia i, de la coloana 0 (inclusiv) până
                    la coloana j (exclusiv) e.g.
                    A[1, :3] -> elementele A[1,0], A[1,1] și A[1,2];
"""


"""
OBS#4: Înmulțirea np.array-urilor se face folosind operația "@":
    (i) Pentru A de dimensiune (n,m) și B de dimensiune (m,p),
        C <- A @ B este produsul matricelor A și B de dimensiune (n,p);

    (ii) Produsul scalar dintre un vector de tip linie, l, și un vector de tip
    coloană, c, de aceeași dimensiune poate fi calculat folosind operația "@":
        x <- l @ c este produsul scalar al vectorului linie l și a vectorului
             coloană c;
"""


"""
OBS#5: Calculul determinantului unei matrice presupune un număr mare de
operații și, prin urmare, șansa de a obține erori cauzate de operații cu numere
a căror reprezentare în virgulă mobilă produce erori de rotunjire, este mare.
Acest lucru devine problematic la verificarea inversabilității unei matrice.

Matricea A = np.array([[3., 5., 3.], [2., 2., 3.], [-1., -3., 0.]]) nu
este inversabilă, dar np.linalg.det(A) = 1.3322676295501906e-15 > 0.
Prin urmare, spunem că o matrice este inversabilă dacă și numai dacă modulul
determinantului său este mai mare decât un număr un pic mai mare decât
precizia mașinii, de exemplu, 1e-14.
"""
```