Le but de ce TP est de finir le DM en écrivant l'ensemble des algorithmes sur les matrices vus dans le cours de mathématiques.

On importera les fonctions du DM (la correction du DM est sur la page du cours) dans un nouveau fichier.

On écrira ensuite les fonctions permettant de résoudre un système triangulaire normalisé en utilisant la méthode de la remontée, d'effectuer la méthode du pivot de Gauss puis d'inverser une matrice.

Dans tout le TP, on travaillera sur des matrices inversibles, si ce n'est pas le cas les fonctions renverront une erreur.

Dans un fichier à part, on écrira des fonctions de test.

Exercice 1 :

(1) Écrire la fonction **remontee** qui prend en argument une matrice M et un vecteur y (un vecteur est une liste à une dimension) et qui renvoie le vecteur x tel que :

$$M \cdot x = y.$$

Si la matrice n'est pas triangulaire supérieure, on affichera une erreur et on renverra **None**

(2) Proposer un test pour vérifier que la fonction retourne bien la solution.

(3) Déterminer la complexité de cet algorithme

Exercice 2 :

On souhaite écrire la fonction **pivot_Gauss** qui prend en entrée une matrice M et un vecteur y et renvoie une matrice triangulaire supérieure M' et un vecteur y' tel que :

$$M \cdot x = y \Leftrightarrow M' \cdot x = y'.$$

On devra travailler sur une copie des matrices M et y, pour effectuer ces copies, on peut utiliser le code :

```
M2 = [list(M[i]) for i in range(len(M))]
y2 = list(y)
```

(1) Écrire la fonction **echange_lignes** qui prend en entrée une matrice M et échange les lignes i et j de celle-ci.

(2) En utilisant l'algorithme du cours, écrire la fonction **pivot_Gauss** en faisant bien attention de choisir à chaque fois un pivot non nul. On choisira le premier pivot non nul de la matrice.

(3) Proposer un test pour vérifier la fonction.

(4) Déterminer la complexité de cet algorithme.

Exercice 3 :

Pour inverser la matrice carrée M, il suffit de résoudre l'équation suivante :

$$M \cdot X = Id,$$

où Id est la matrice identité.

On remarquera que l'on peut résoudre les systèmes suivants indépendamment les uns des autres.

$$M \cdot X_i = Id_i \text{ pour tout i plus petit que le nombres de colonnes de M,}$$

où X_i est i-ème colonnes de X.

(1) En utilisant les fonctions précédentes, proposer un algorithme qui permet de calculer l'inverse d'un matrice carrée.

(2) Tester cet algorithme.

(3) Déterminer la complexité de celui-ci.

(4) Il est possible d'obtenir un algorithme de complexité de l'ordre de n^3 (où n est le nombre de lignes). En observant les calculs inutiles, proposer un algorithme de complexité de l'ordre de n^3

Exercice 4 :

Toutes ces opérations sont bien sûr possible en Python. Pour cela, on peut utiliser la bibliothèrque **numpy** que l'on peut charger de la manière suivante :

```
import numpy as np
```

Pour accéder à une fonction de **numpy**, on utilisera **np.nom_fonction**

Dans un nouveau fichier, tester toutes les fonctions du DM et de ce TP avec **numpy**. On pourra comparer l'efficacité de celles-ci.

En programmation, sauf si le professeur vous le demande explicitement, on ne réinvente jamais la roue. On utilise les bibliothèques et les fonctions existantes répondant à notre problème