Un élève de CPGE s'intéresse aux matrices triangulaires réelles (franchement, il a bien raison ! ☺).

Programmant en langage Python mais ne disposant pas de la bibliothèque `numpy` sur son ordinateur (que n'utilise-t-il Pyzo ???), il décide de représenter une telle matrice sous forme d'une liste (celles des lignes de la matrice considérée) mais en ne stockant pas les « 0 » situés au-dessus de la diagonale (pour les triangulaires inférieures) ou sous la diagonale (pour les triangulaires supérieures).
Ainsi, il représente les matrices :

$$A = \begin{pmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & -5 & 13 \end{pmatrix} \text{ et } B = \begin{pmatrix} 4 & -7 & 20 & 0 \\ 0 & 0 & -3 & 5 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 10 \end{pmatrix}$$

par les listes :

```
LA=[[2],[-1,1],[3,-5,13]]
```

```
LB=[[4,-7,20,0],[0,-3,5],[2,1],[10]]
```

Dans la suite, toutes les matrices triangulaires seront représentées de la sorte.

1. Ecrire une fonction Python `TriMat_test` qui reçoit une matrice triangulaire T et renvoie la chaîne de caractère `Tinf` ou `Tsup` selon que la matrice T est triangulaire inférieure ou supérieure.

2. Ecrire une fonction Python `TriMat_det` qui reçoit une matrice triangulaire T et renvoie son déterminant.

3. Ecrire une fonction Python `TriMat_transpose` qui reçoit une matrice triangulaire T et renvoie sa transposée.

4. Ecrire une fonction Python `TriMat_inverse` qui reçoit une matrice triangulaire T et :
   - teste l'inversibilité de T.
   - renvoie le booléen `True` et la matrice T^{-1} ou seulement le booléen `False` suivant que T est ou non inversible.

   Evaluer la complexité (nombre d'opérations) du calcul de T^{-1}.

5. **Bonus**. L'élève a finalement décidé de créer une sous-classe `TriMat` de la classe `list` pour manipuler ses matrices triangulaires.

   Reprendre la question 2 en récrivant `TriMat_det` en tant que méthode de la sous-classe `TriMat`.