```python
class TriMat(list):

    """
    Méthode TriMat_test.
    Détermine si une matrice triangulaire est inférieure ou supérieure.
    """
    def TriMat_test(self):
        if len(self[0]) == 1 :
            return("Tinf")
        else:
            return("Tsup")

    """
    Méthode TriMat_det.
    Calcule le déterminant d'une matrice triangulaire
    (produit des éléments diagonaux).
    """
    def TriMat_det(self):
        n = len(self)
        # Cas particulier
        if n == 1:
            return(self[0][0])
        else:
            d = 1
            # Cas d'une triangulaire supérieure
            # On multiplie entre eux les éléments diagonaux, c'est à dire les
            # premiers éléments des listes dans self.
            if self.TriMat_test() == "Tsup":
                for i in range(len(self)):
                    d *= self[i][0]
            # Cas d'une triangulaire inférieure
            # On multiplie entre eux les éléments diagonaux, c'est à dire les
            # derniers éléments des listes dans self.
            else:
                for i in range(len(self)):
                    d *= self[i][-1]
            return(d)

    """
    Méthode TriMat_transpose.
    Calcule la transposée d'une matrice triangulaire.
    """
    def TriMat_transpose(self):
        n = len(self)
        # Cas particulier
        if n == 1:
            return(self)
        else:
            # Initialisation de TR (une liste de listes vides)
            TR = []
            for i in range(n):
                TR.append([])
            # Cas d'une triangulaire supérieure
            if self.TriMat_test() == "Tsup":
                for i in range(n):
                    for j in range(len(self[i])):
                        TR[i+j].append(self[i][j])
            # Cas d'une triangulaire inférieure
            else:
                for i in range(n):
                    for j in range(n-i):
                        TR[i].append(self[i+j][i])
            # Fin
            return(TriMat(TR))

    """
    Méthode TriMat_inverse.
```

```
    Calcule l'inverse d'une matrice triangulaire.
    """
    def TriMat_inverse(self):
        if self.TriMat_det() == 0:
            return(False)
        else:
            n = len(self)
            # Cas d'une triangulaire supérieure
            if self.TriMat_test() == "Tsup":
                # Initialisation de l'inverse TI
                TI = []
                # Eléments diagonaux
                for i in range(n):
                    TI.append([1/self[i][0]])
                # Autres éléments (ligne par ligne)
                for i in range(n):
                    # Pour une ligne (i) donnée, on calcule les éléments
                    # d'indices (informatiques) de colonne de i+1 à n-1
                    for k in range(i+1,n):
                        S = 0
                        for j in range(k-i):
                            S += TI[i][j]*self[i+j][k-j-i]
                        TI[i].append(-S/self[k][0])
                # Fin
            # Cas d'une triangulaire inférieure
            else:
                # On transpose self pour se ramener au cas précédent...
                T = self.TriMat_transpose()
                X = T.TriMat_inverse()
                TI = X[1].TriMat_transpose()
            return(True,TriMat(TI))

# Pour tester...
# Deux matrices triangulaires supérieures :
# A = TriMat([[4,1],[2]])
# A = TriMat([[2,1,2],[2,2],[4]])
# Une matrice triangulaire inférieure :
A = TriMat([[1],[2,2],[1,5,4]])
# Et parce que les problèmes d'arrondis ne sont jamais très loin...
# A = TriMat([[2,0,-2,3],[5,-2,1],[5,-1],[-2]])
X = A.TriMat_inverse()[1]
print(X)
```