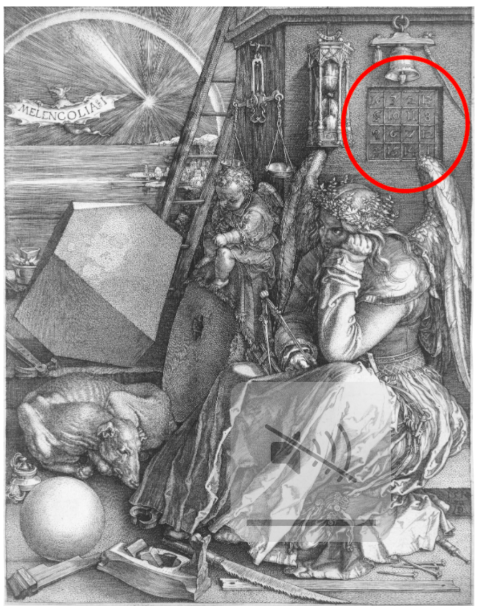
TP de synthèse : carrés magiques

AVERTISSEMENT

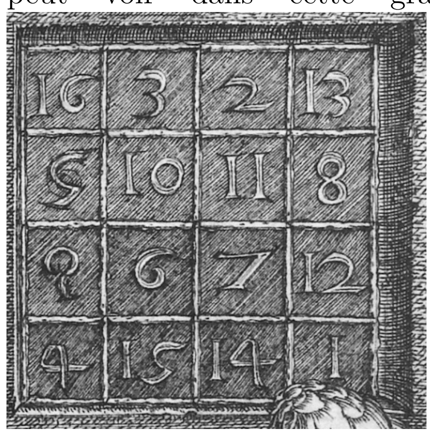
Numérotez les questions en utilisant un commentaire #Question 1, etc.

INTRODUCTION

Un carré magique est un arrangement de nombres tous différents dans un tableau bi-dimensionnel de sorte que les sommes des éléments de chaque ligne, de chaque colonne, et des deux diagonales soient toutes identiques.



A gauche, une gravure du peintre Albrecht DURER (en 1514).
Ci-dessous un agrandissement du carré magique que l'on peut voir dans cette gravure.



16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Ci-dessus, une représentation plus moderne de ce carré magique.

On codera ce carré par tableau à double entrée par une liste de listes :

$$T = \texttt{[[16,3,2,13],[5,10,11,8],[9,6,7,12],[4,15,14,1]]}$$

On a par exemple, `T[0][1]=3` et `T[3][2]=14`.

PARTIE I : une fonction indiquant si un carré est magique

On se propose dans un premier temps d'écrire une fonction Python `EstMagique(T)` qui prend en entrée un tableau bi-dimensionnel `T` carré de taille quelconque et qui renvoie un booléen indiquant si le tableau est magique ou pas. On procède par étapes.

1. Écrire une fonction `SommeLigne(T,i)` qui prend en argument un tableau bi-dimensionnel carré `T`, un entier i et renvoie la somme des entrée `T[i][k]` figurant sur la ligne `T[i]` de `T`.
2. Écrire une fonction `SommeColonne(T,j)` qui prend en argument un tableau bi-dimensionnel carré `T`, un entier j et renvoie la somme des entrée `T[j][k]` figurant sur la colonne `T[j]` de `T`.

3. Écrire une fonction `SommeDiag(T)` qui prend en argument un tableau bi-dimensionnel carré `T` et renvoie la somme des éléments de la diagonale principale de `T`. Dans l'exemple ci-dessus, cette fonction renvoie la somme de $16+10+7+1$.
4. Écrire une fonction `SommeAntiDiag(T)` qui prend en argument un tableau bi-dimensionnel carré `T` et renvoie la somme des éléments de l'autre diagonale de `T`. Dans l'exemple ci-dessus, cette fonction renvoie la somme de $4+6+11+13$.
5. A l'aide des fonctions précédentes, écrire la fonction `EstMagique` demandée.
6. Le carré magique de DURER donné en exemple ci-dessus possède une autre particularité, il contient $4 \times 4 = 16$ cases et il utilise tous les nombres de 1 à 16. Un tel carré est dit **complet**.

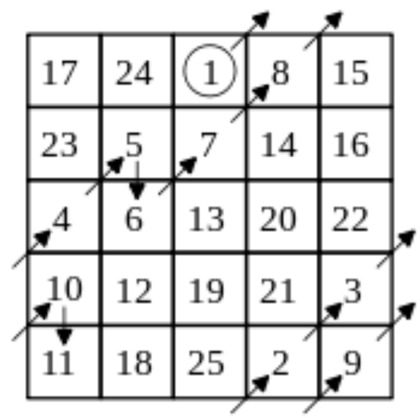
   Un tableau de taille $n \times n$ sera donc appelé **carré magique complet** s'il est **magique ET** s'il contient **tous les nombres entiers de** 1 **à** n^2.

   Écrire une fonction `EstComplet(T)` qui renvoie un booléen indiquant si `T` est est un carré magique complet ou non.

PARTIE II : une méthode pour remplir un carré semi-magique complet

En 1693, De la Loubère donna une méthode pour mettre les nombres $1, 2, \ldots, n^2$ dans un carré $n \times n$, avec n impair, de sorte que SommeLigne et SommeColonne renvoient toujours la même valeur.

Attention : dans cette partie, on ne donne **pas la contrainte des diagonales**, on dira que ce carré est semi-magique complet.



La méthode est la suivante : expliquée avec un carré 5×5 ci-contre.

On place 1 à n'importe quel endroit du carré puis on se déplace diagonalement vers le nord-est, en mettant les nombres 2, 3 etc. Quand on arrive au bord (ici pour 2), on ressort par le côté opposé.
On continue alors de se déplacer vers le nord-est, tant qu'on ne tombe pas sur une case déjà occupée. Quand on tombe sur une case déjà occupée, comme cela se produit après 5 ci-dessous, on descend d'une case puis on recommence à se déplacer vers le nord-est.

1. Expliquez la méthode de De la Loubère sur un carré de taille 3×3 en plaçant le 1 au centre du carré. Écrire la liste de listes correspondantes dans l'`IDLE`.
2. Pour implémenter la méthode de De la Loubère, on commence par créer un tableau bi-dimensionnel rempli de zéros. Écrire une fonction `TableauZeros(n)` renvoyant un tableau bi-dimensionnel de taille $n \times n$ (sous forme d'une liste de listes) rempli de zéros.
3. Écrire une fonction `Init(n)` qui renvoie un couple de nombres aléatoires (a, b) avec a et b dans $[\![0, n-1]\!]$. On pourra utiliser `import random as rd` et `rd.randint(a,b)`.
4. Écrire une fonction `NordEst(i,j,n)` qui prend en entrée des entiers i et j supposés dans $[\![0, n-1]\!]$ et qui renvoie le couple de coordonnées de la case située au nord-est de la case (i, j) dans un tableau (i, j). **Indication :** Si $(i-1, j+1) \in [\![0, n-1]\!]^2$, c'est le couple recherché, sinon, il faut adapter...
5. Écrire une fonction `CaseDuDessous(i,j,n)` avec les mêmes entrées que précédemment, et qui renvoie renvoie la case située au-dessous de la case (i, j).
6. A l'aide des fonctions précédentes, écrire une fonction `DLLB(n)` qui renvoie un tableau $n \times n$ obtenu par la méthode de De la Loubère.