Un Sudoku classique est une grille de 9 lignes et 9 colonnes et composé de 9 blocs distincts de 3 lignes et 3 colonnes.

Pour ce mini projet nous prendrons une grille 4 x 4 (4 lignes et 4 colonnes) et composée de 4 blocs (bleu, vert, jaune, violet) de 4 cases présentée figure 3.

Remplir une grille de Sudoku et donc la résoudre, consiste à utiliser tous les chiffres de 1 à 4 pour chacun des 4 blocs de sorte que chaque ligne et chaque colonne (et aussi chaque bloc) de la grille ne comportent aucun chiffre en double. La solution du Sudoku précédent (figure 1) et donnée figure 2.

1		3	
3		2	
		4	

Figure 1

1	2	3	4
3	4	2	1
2	1	4	3
4	3	1	2

Figure 2

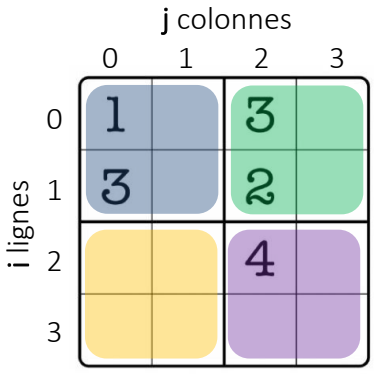

Figure 3

On représente une telle grille par une liste S de 4 listes à 4 éléments, chacune des listes représentant une ligne.

L'objectif de ce mini projet et d'obtenir par programmation la résolution de la grille présentée figure 1 pour obtenir celle résolue présentée figure 2. **Le fichier Sudoku_4x4_elev.py est à compléter.**
A noter que votre programme devra à l'issu des questions résoudre un Sudoku de 9x9. Donc i et j ne sont pas des constantes mais des paramètres.

a. ***Donner*** *la variable S qui code la grille de sudoku 4x4 de la figure1. Les cases vides sont codées par des 0.*

b. ***Écrire*** *une fonction ligne (S, i) qui retourne la liste des chiffres qui apparaissent sur la ligne i. Les commandes suivantes doivent renvoyer :*

```
>>> ligne (S,0)
[1,3]
>>> ligne (S,1)
[3, 2]
>>> ligne (S,2)
[4]
```

c. ***Ajouter*** *une ou des assertions à la fonction ligne(S, i) qui vérifient l'ensemble des valeurs possibles pour le paramètre i.*

d. ***Écrire*** *une fonction colonne (S, j) qui retourne la liste des chiffres qui apparaissent sur la colonne j. Les commandes suivantes doivent renvoyer :*

```
>>> colonne (S,0)
[1,3]
>>> colonne (S,2)
[3,2,4]
```

e. ***Ajouter*** *une ou des assertions à la fonction colonne(S, j) qui vérifient l'ensemble des valeurs possibles pour le paramètre j.*

Les blocs sont identifiés par les coordonnées (h, g) avec h son indice de ligne et g son indice de colonne, de leur case 'en haut à gauche'. Soit (0, 0) pour le bloc bleu, (0,2) pour le bloc vert…

Pour trouver le bloc auquel appartient la case (i, j) on calcule la position (h, g) de la case 'en haut à gauche' de ce bloc par cette méthode :

- On calcule le quotient de i par deux et on multiplie par deux. Soit : h= (i//2) *2
- Idem pour calculer g avec j.

Pour s'en convaincre vous pouvez vérifier que pour le bloc jaune les 4 couples (i, j) qui sont (2, 0) (2, 1) (3, 0) (3, 1) donnent tous le couple (h, g) = (2, 0) qui est la case 'en haut à gauche' qui référence le bloc jaune.

Pour se déplacer dans le bloc, de la gauche vers la droite, on demarre à gauche soit la position g et on va à la position g+t-1, avec $t = \sqrt{len(S)}$. On retrouve la même logique pour se déplacer vers le bas à partir de h pour aller à h+t-1. La figure 4 illustre le principe de ce calcul.

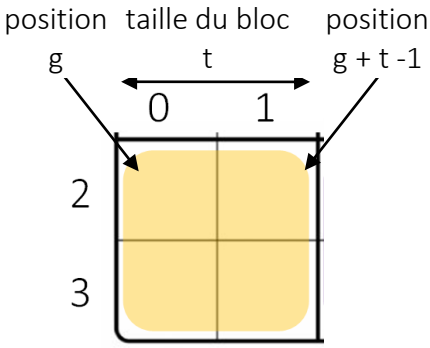

Figure 4

f. ***Écrire*** *une fonction bloc (S, i, j) qui retourne la liste des chiffres qui apparaissent dans le bloc auquel appartient la case de la ligne i et de la colonne j. Les commandes suivantes doivent renvoyer :*

```
>>>bloc (S, 1, 1)
[1, 3]
>>>bloc (S, 0, 3)
[3,2]
>>>bloc (S, 3, 2)
[4]
>>>bloc (S, 2, 1)
[]
```

La fonction possibles (S, i, j) retourne la liste des chiffres de 1 à 4 que l'on peut écrire dans la case de la ligne i et de la colonne j (en tenant compte des règles du jeu). Cette fonction utilise les trois fonctions précédentes.

g. ***Écrire*** *la doc string de la fonction possible (S, i, j)*

h. ***Compléter*** *la fonction possibles (S, i, j) qui retourne la liste des chiffres de 1 à 4 que l'on peut écrire dans la case de la ligne i et de la colonne j. Les commandes suivantes doivent renvoyer :*

```
>>> possibles (S, 3, 1)
[1,2,3,4]
>>> possibles (S, 0, 3)
[4]
>>> possibles (S, 0, 1)
[2,4]
>>> possibles (S, 3, 2)
[1]
```

La grille est complétée de la ligne 0 à la ligne 3 comme le montre la figure 5.

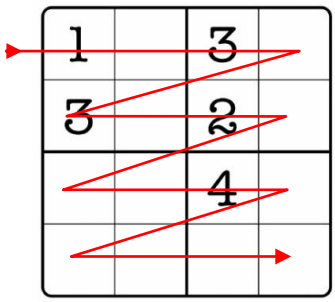

figure 5

i. *Compléter la fonction suivante (i, j) qui reçoit en paramètres la ligne i est la colonne j de la case actuelle et renvoie le tuple (ligne, colonne) de la case suivante. Les commandes suivantes doivent renvoyer :*

```
>>> suivante (0,0) renvoie
(0,1)
>>> suivante (0,3) renvoie
( 1,0)
>>> suivante (3,3) renvoie
(4,0)
```

La résolution de la grille de Sudoku se fait récursivement, la méthode consiste à tester tous les nombres et à revenir en arrière si la solution n'est pas satisfaisante on appelle ça le backtracking. Quatre cas s'envisagent :

- le cas de base :
  Si i est égale à 4 :
  c'est que l'on est arrivé à remplir la grille donc on renvoie True
- un cas récursif :
  Sisinon la case (i, j) est déjà remplie :
  on passe à la case suivante (k, l)
  on 'lance' la récurence avec les coordonnées (k, l) de la case suivante
- un cas récursif :
  Sisinon la case (i, j) est vide :
  on parcourt tous les chiffres possibles :
  on affecte le chiffre possible à la case (i, j)
  on passe à la case suivante (k, l)
  Si la récurtion de la case suivante (k, l) retourne True :
  renvoie True
- pour les autres cas : affecte 0 à la case(i, j) et renvoie False

j. *Compléter la fonction resolution (S, i, j) qui retourne True si la grille est résolvable ou False sinon. Cette fonction utilise les deux fonctions précédentes.*