Un *Sudoku* classique est une grille de 9 lignes et 9 colonnes, donc composée de 9 blocs distincts de 3 lignes et 3 colonnes.

Remplir une grille de Sudoku consiste à utiliser tous les chiffres de 1 à 9 pour chacun des 9 blocs de sorte que chaque ligne et chaque colonne de la grille totale ne comporte aucun chiffre en double.

Dans cet exercice, on décidera de représenter une telle grille par une liste `S` de neuf listes à neuf éléments, chacune des listes représentant une ligne.

L'objectif de cet exercice est d'obtenir le remplissage de la grille page suivante (figure 2.10).

1 Compléter la variable :`S = [ [0,1,0,0,7,8,0,0,0] , ... ]`

2 Écrire une fonction `ligne(S,i)` qui retourne la liste des chiffres de 1 à 9 qui apparaissent sur la ligne i. On devra avoir pour notre grille :

```
>>> ligne(S,0)
[1,7,8]
```

	1			7	8			
	8			4		9		
		5	6				1	
1				6				5
	4		9	1	5		7	2
	6	7		8		4		
			3			1		
	7		8	9			2	3
					4			

Figure 2.10 – Grille de Sudoku à implémenter

3 Écrire une fonction `colonne(S,j)` qui retourne la liste des chiffres de 1 à 9 qui apparaissent dans la colonne j. On devra avoir pour notre grille :

```
>>> colonnes(S,0)
[1]
```

4 Écrire une fonction `bloc(S,i,j)` qui retourne la liste des chiffres de 1 à 9 qui apparaissent sur le bloc 3×3 auquel appartient la case de la ligne i et de la colonne j. On devra avoir pour notre grille :

```
>>> bloc(S,3,4)
[6,9,1,5,8]
```

5 Écrire alors une fonction `possibles(S,i,j)` qui retourne la liste des chiffres de 1 à 9 que l'on peut écrire dans la case de la ligne `i` et de la colonne `j` (en tenant compte des règles du jeu). On devra avoir pour notre grille :

```
>>> possibles(S,0,0)
[2,3,4,6,9]
```

6 On complète la grille de la ligne 0 à la ligne 8, de la colonne 0 à la colonne 8.
Écrire une fonction `suivante(i,j)` qui reçoit en paramètre la ligne `i` et la colonne `j` d'une case, et qui renvoie le tuple (ligne,colonne) de la case suivante. On devra avoir :

```
>>> suivante(0,0) , suivante(0,8) , suivante(8,8)
(0,1) , (1,0) , (9,0)
```

7 Compléter la fonction principale de résolution suivante :

```
def solve(S,i,j):
    if i == 9:
        return ...
    elif S[i][j] > 0:
        ...
        return ...
    for k in possibles(S,i,j):
        S[i][j] = k
        ...
        if solve(S,a,b):
            return ...
    S[i][j] = 0
    return False
```