Autour du sudoku

Une grille de Sudoku est une grille de taille 9×9 découpée en 9 carrés de taille 3×3. Le but est de la remplir avec les chiffres $1, 2, \ldots, 9$ de sorte que chaque ligne chaque colonne, et chacun des 9 petits carrés de taille 3×3 ne contienne qu'une seul fois chaque chiffre. On dit que la grille est alors complète.

Le but du TP est d'écrire un programme permettant de résoudre une grille de Sudoku, puis de tester ce programme sur un ensemble de grilles. On admet que la solution est unique.

On utilise dans les exemples qui suivent la grille ci-après :

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

Les 9 carrés de taille 3×3 sont numérotés de 0 à 8 de gauche à droite puis de haut en bas (sens de la lecture).

On représente la grille par la liste `Ltest` suivante :

```
Ltest = [ [0,6,0,0,0,0,2,0,5], [4,0,0,9,2,1,0,0,0], [0,7,0,0,0,8,0,0,1], ...]
```

Copier les fichiers `TP_sudoku.py` et `TP_sudoku_tests.py` dans votre dossier de travail.

- Le premier fichier contient un ensemble de fonctions à compléter (ou pas).
- Pour plus de lisibilité, les tests seront réalisés dans le second fichier où plusieurs grilles de test sont déjà définies.

  Ce fichier test commence notamment par l'importation du fichier `TP_sudoku.py`.

  Ce fonctionnement permet de séparer la partie "programmation" de la partie "utilisation"

A- Vérification de la grille

A.1 La fonction `ligne_complete(L,i)` prend en argument une liste Sudoku `L` et un entier i entre 0 et 8 et renvoie `True` si la ligne i du Sudoku `L` vérifie les conditions de remplissage d'un Sudoku, `False` sinon. Analyser son écriture et tester la fonction sur la grille test.

A.2 Compléter la fonction `colonne_complete(L,j)` pour vérifier les colonnes,

A.3 Idem avec la fonction `carre_complet(L,k)` pour vérifier les petits carrés (rappel : les carrés sont numérotés de 0 à 8).

A.4 Écrire une fonction `grille_complete(L)` qui prend en argument une liste Sudoku `L` et renvoie `True` si la grille est complète et vérifie les règles de remplissage d'un Sudoku, `False` sinon. Tester votre fonction (sur la liste `Ltest` et sur une liste fausse de votre choix).

B- Résolution de la grille

B.1 La fonction `ligne(L,i)` définie ci-dessous renvoie la liste des nombres compris entre 1 et 9 présent sur la ligne d'indice i. Par exemple avec la grille initiale on obtient :

```
>>> ligne(L, 0)
[6, 2, 5]
```

Compléter sur le même principe la fonction `colonne(L,j)` pour la colonne d'indice j.

B.2 Pour obtenir la liste des éléments présents dans un petit carré on procède différemment. On écrit une fonction `carre(L,i,j)` qui prend pour paramètre la liste Sudoku L et les indices d'une case (i, j). Cette fonction retourne la liste des nombres qui apparaissent dans le carré 3×3 auquel appartient la case (i, j).

Expliquer le rôle des variables `icoin` et `jcoin` puis compléter le code proposé.

Avec la grille de l'énoncé, on doit obtenir par exemple

```
>>> carre(L,4,6)
[9,7,3]
```

B.3 La fonction `conflit(L,i,j)` renvoie la liste des chiffres qu'on ne peut pas écrire dans la case (i, j) sans contredire les règles du jeu (en autorisant les éventuels doublons)

Compléter la fonction `bons_chiffres(L,i,j)` qui renvoie la liste des chiffres que l'on peut écrire dans la case (i, j) :

Par exemple avec la grille initiale de l'énoncé, on doit obtenir

```
>>> bons_chiffres(L, 4, 2)
[2, 5, 8, 9]
```

Pour résoudre la grille de Sudoku de façon naïve, on commence par compléter les cases n'ayant qu'une seule possibilité. On réitère le processus jusqu'à ce que toutes les cases soient complétées. Cela suppose qu'il existe des cases pour lesquelles on n'a qu'un seul choix possible.

B.4 La fonction `untour(L)` prend comme argument une liste Sudoku L et retourne `True` s'il existe dans L une case avec une seule possibilité. La fonction doit également compléter toutes les cases avec un seul choix possible. La compléter.

B.5 Compléter enfin la fonction `resolution(L)` qui tente de remplir la grille de façon naïve comme décrit ci-dessus, et qui renvoie `True` si la grille a été complétée, `False` sinon.

Vous pourrez tester votre code en utilisant la fonction `affiche_grille(grille)`.

En particulier la grille `Ltest` n'est pas résoluble mais la grille `sudoku_resoluble` l'est (et sa solution est la grille `sudoku_complet`)

C - Bonus : test de l'algorithme

Pour cette partie c'est à vous d'écrire en totalité les fonctions ! Vous aurez besoin des méthodes de lecture/écriture vues au TP « Utilisation de modules sur l'exemple du traitement de données »

L'objectif de cette partie est de tester l'efficacité de cet algorithme de résolution, en l'utilisant pour des grilles de Sudoku contenant plus ou moins de chiffres déjà révélés, et stockées dans des fichiers où chaque ligne représente le contenu d'une grille de Sudoku.

C.1 Copier les fichiers `grilles1.txt` à `grilles5.txt` (Grilles récupérées sur le site `http://www.printable-sudoku-puzzles.com`) dans votre répertoire de travail. Les fichiers `grillesX.txt` contiennent un sudoku par ligne, chaque sudoku étant décrit par une chaîne de 81 caractères.

C.2 Écrire une fonction `chaineVersGrille` qui prend en entrée une chaîne de caractères contenant 81 chiffres et qui renvoie la grille correspondante. La tester en récupérant *(à la main)* la première ligne du fichier `grilles1.txt`.

C.3 Écrire une fonction `grilleVersChaine` qui réalise l'opération inverse.

C.4 Écrire une fonction `testeFichier` qui prend en argument une chaîne de caractères correspondant à un nom de fichier contenant des sudokus, qui tente de résoudre chaque sudoku, qui enregistre les grilles obtenues sous forme de chaînes de caractères dans un fichier, et qui renvoie le pourcentage de sudokus du fichier qu'on arrive à résoudre grâce à la fonction `resolution`. Si le fichier initial s'appelle `mon_fichier.txt`, le fichier des grilles obtenues avec la méthode de résolution sera nommé `resultats_mon_fichier.txt`.

C.5 Écrire une fonction `testeCinqFichiers` qui ne prend aucun paramêtre et qui renvoie la liste des pourcentages de sudokus résolus pour les fichiers `grilles1.txt` à `grilles5.txt`.