Projet : Sudoku

Pour ce projet, vous vous mettrez par groupes de trois étudiants, si possible de niveaux hétérogènes en informatique. Le code commenté devra être renvoyé par courriel à l'adresse math@quirin.science.

Le but du projet est d'écrire un programe Python qui résout des grilles de Sudoku. On renvoie à la page Wikipédia pour la définition d'un sudoku. On pourra récupérer sur ma page un fichier contenant une liste de cinq sudokus ainsi qu'une liste de leurs résolutions.

Les sudokus seront représentés sous forme de la liste de leur lignes, chaque ligne étant une liste d'entiers. Une case vide contiendra l'entier 0.

Le projet se fait en trois parties, de difficulté croissante.

Partie 1. Écrire un programme affichant joliment une grille de sudoku.

Partie 2. Écrire un programme vérifiant qu'une grille de sudoku complète est bien valide.

Pour cela, on pourra écrire des fonctions calculant la ligne, la colonne et le carré associés à une case. Une autre fonction pourra tester qu'une liste contient bien une et une seule fois tous les entiers de 1 à 9. Il suffira alors de tester chacune des cases de la grille.

Partie 3. Écrire une fonction résolvant un sudoku. On pourra coder les fonctions suivantes :

- On transforme la grille en grille d'hypothèses : chaque case vide au départ doit contenir la liste des nombres possibles qu'elle peut contenir (*i.e.* qui ne sont pas déjà sur la même ligne, la même colonne ou le même carré) ; chaque case non vide est donc remplacée par une liste ne contenant que l'élément en question.
- On va ensuite tester toutes les hypothèses possibles : on écrit une fonction choisissant une case ayant le minimum d'éléments possibles, sauf 1, et renvoyant $(-1,-1)$ si toutes les cases n'ont qu'un élément.
- On actualise la grille : si on fixe une certaine case (i,j) à une valeur, on la supprime de toutes les listes d'entiers possibles pour les cases des ligne, colonne et carré.

La fonction principale prendra en argument une liste d'hypothèses, et renverra une liste d'hypothèses et un booléen. Elle fera les opérations suivantes :

- On commence par actualiser la grille en fixant les valeurs des cases ne contenant qu'une possibilité.
- On choisit la case (i,j) sur laquelle on veut travailler : si c'est la case $(-1,-1)$, on a en fait déjà gagné (on renvoie le couple (grille,True)) ; si elle est de longueur 0, c'est perdu (on renvoie le couple (grille,False))
- On copie la grille (on pourra utiliser la fonction `deepcopy` du module `copy`, ou écrire une fonction de copie), dans laquelle on fixe une valeur possible pour la case (i,j) et on actualise la grille
- Récursivement, on essaye de résoudre cette nouvelle grille : si le booléen est vrai, on a terminé. Sinon, l'hypothèse faite était fausse : on l'enlève de la liste des possibilités, puis on réessaye récursivement de la résoudre.