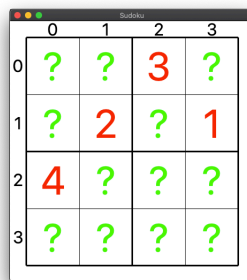


1 Sudoku

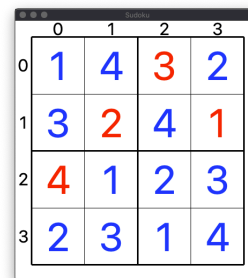
Un sudoku d'ordre n ¹ est une grille carrée de $n^2 \times n^2$, décomposée en $n \times n$ blocs de taille $n \times n$, qu'il faut remplir avec des entiers de 1 à n^2 tel que :

1. les lignes et les colonnes contiennent chaque nombre exactement une fois (carré latin);
2. les blocs contiennent chaque nombre exactement une fois.

Certaines cases sont remplies dans la grille initiale et ne peuvent pas être modifiées. Voici quelques exemples :

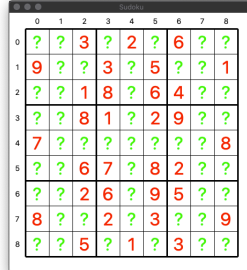


| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | ? | ? | 3 | ? |
| 1 | ? | 2 | ? | 1 |
| 2 | 4 | ? | ? | ? |
| 3 | ? | ? | ? | ? |

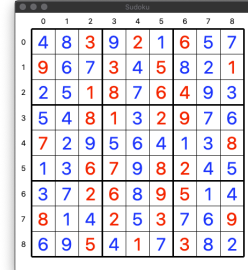


| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 4 | 3 | 2 |
| 1 | 3 | 2 | 4 | 1 |
| 2 | 4 | 1 | 2 | 3 |
| 3 | 2 | 3 | 1 | 4 |

Une grille initiale et sa solution pour un sudoku d'ordre 2



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | ? | ? | 3 | ? | 2 | ? | 6 | ? |
| 1 | 9 | ? | ? | 3 | ? | 5 | ? | 1 |
| 2 | ? | ? | 1 | 8 | ? | 6 | 4 | ? |
| 3 | ? | ? | 8 | 1 | ? | 2 | 9 | ? |
| 4 | 7 | ? | ? | ? | ? | ? | ? | 8 |
| 5 | ? | ? | 6 | 7 | ? | 8 | 2 | ? |
| 6 | ? | ? | 2 | 6 | ? | 9 | 5 | ? |
| 7 | 8 | ? | ? | 2 | ? | 3 | ? | 9 |
| 8 | ? | ? | 5 | ? | 1 | ? | 3 | ? |




| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 8 | 3 | 9 | 2 | 1 | 6 | 5 |
| 1 | 9 | 6 | 7 | 3 | 4 | 5 | 8 | 2 |
| 2 | 2 | 5 | 1 | 8 | 7 | 6 | 4 | 9 |
| 3 | 5 | 4 | 8 | 1 | 3 | 2 | 9 | 7 |
| 4 | 7 | 2 | 9 | 5 | 6 | 4 | 1 | 3 |
| 5 | 1 | 3 | 6 | 7 | 9 | 8 | 2 | 4 |
| 6 | 3 | 7 | 2 | 6 | 8 | 9 | 5 | 1 |
| 7 | 8 | 1 | 4 | 2 | 5 | 3 | 7 | 6 |
| 8 | 6 | 9 | 5 | 4 | 1 | 7 | 3 | 8 |

Une grille initiale et sa solution pour un sudoku d'ordre 3



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|----|----|----|---|----|----|----|---|----|---|----|----|----|----|----|----|----|
| 0 | 4 | 7 | 2 | 1 | 11 | 16 | 2 | 7 | ? | ? | ? | ? | 2 | 16 | 7 | ? |
| 1 | ? | 8 | 3 | 12 | ? | ? | 7 | 10 | 5 | 13 | ? | ? | ? | 11 | 14 | 1 |
| 2 | ? | 15 | 7 | ? | ? | ? | 6 | 14 | 3 | 11 | ? | ? | ? | 9 | 16 | ? |
| 3 | 10 | 11 | ? | ? | ? | ? | 3 | ? | ? | ? | ? | 14 | ? | ? | ? | 6 |
| 4 | 14 | ? | ? | ? | ? | ? | 5 | ? | ? | ? | ? | 13 | ? | ? | ? | 11 |
| 5 | 15 | ? | ? | 11 | 9 | 2 | ? | ? | ? | ? | ? | 3 | 1 | 4 | ? | ? |
| 6 | ? | 9 | 1 | ? | ? | ? | 4 | ? | ? | ? | ? | 5 | ? | ? | ? | 13 |
| 7 | ? | 2 | 6 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | 8 | 3 |
| 8 | ? | 14 | 2 | ? | ? | ? | ? | 13 | ? | ? | ? | ? | ? | ? | ? | 4 |
| 9 | ? | ? | 1 | 5 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | 10 |
| 10 | 16 | ? | ? | ? | ? | 14 | 8 | ? | ? | ? | ? | ? | 9 | 11 | 3 | ? |
| 11 | 6 | ? | ? | ? | ? | 12 | ? | ? | ? | ? | ? | ? | ? | ? | ? | 14 |
| 12 | 13 | 5 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | 8 |
| 13 | ? | ? | 3 | 12 | ? | ? | ? | 5 | 4 | 11 | 2 | ? | ? | ? | ? | 16 |
| 14 | ? | ? | 7 | 9 | 14 | ? | ? | ? | 8 | 6 | 16 | 10 | ? | ? | ? | 5 |
| 15 | ? | ? | ? | ? | 4 | 3 | 9 | 7 | ? | ? | 15 | 5 | 10 | ? | ? | 12 |



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 4 | 1 | ? | ? | 11 | 16 | ? | ? | ? | ? | ? | ? | 2 | 16 | ? | ? |
| 1 | ? | 8 | 3 | 12 | ? | ? | 7 | 10 | 5 | 13 | ? | ? | ? | 11 | 14 | 1 |
| 2 | ? | 15 | 7 | ? | ? | ? | 6 | 14 | 3 | 11 | ? | ? | ? | 9 | 16 | ? |
| 3 | 10 | 11 | ? | ? | ? | ? | 3 | ? | ? | ? | ? | 14 | ? | ? | ? | 6 |
| 4 | 14 | ? | ? | ? | ? | ? | 5 | ? | ? | ? | ? | 13 | ? | ? | ? | 11 |
| 5 | 15 | ? | ? | 11 | 9 | 2 | ? | ? | ? | ? | ? | 3 | 1 | 4 | ? | ? |
| 6 | ? | 9 | 1 | ? | ? | ? | 4 | ? | ? | ? | ? | ? | ? | ? | ? | 13 |
| 7 | ? | 2 | 6 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | 8 |
| 8 | ? | 14 | 2 | ? | ? | ? | ? | 13 | ? | ? | ? | ? | ? | ? | ? | 4 |
| 9 | ? | ? | 1 | 5 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | 10 |
| 10 | 16 | 10 | 4 | 7 | 14 | 8 | 15 | 5 | 2 | 6 | 9 | 11 | 3 | 12 | 13 | 1 |
| 11 | 6 | 13 | 11 | 3 | 7 | 12 | 16 | 9 | 1 | 4 | 5 | 10 | 8 | 15 | 2 | 14 |
| 12 | 13 | 5 | 16 | 10 | 16 | 7 | 12 | 1 | 14 | 9 | 6 | 4 | 2 | 11 | 8 | 3 |
| 13 | 1 | 3 | 12 | 6 | 15 | 10 | 5 | 4 | 11 | 2 | 8 | 7 | 13 | 16 | 14 | 9 |
| 14 | 11 | 7 | 9 | 14 | 2 | 13 | 8 | 6 | 16 | 10 | 12 | 3 | 5 | 1 | 4 | 16 |
| 15 | 2 | 16 | 8 | 4 | 3 | 9 | 14 | 11 | 13 | 1 | 15 | 5 | 10 | 7 | 6 | 12 |

Une grille initiale et une grille partiellement remplie pour un sudoku d'ordre 4

1. Le format le plus usuel est celui d'ordre 3. Cependant, votre programme doit savoir gérer des sudokus d'ordre quelconque.

2 Représentation en mémoire d'une grille de sudoku

Il faut d'abord commencer par définir la représentation mémoire d'une grille d'ordre n . La structure de données que vous allez utiliser doit permettre de :

1. représenter des grilles à deux dimensions ;
2. stocker pour chaque case : i) est-elle modifiable ? ii) quelle est la valeur stockée ? iii) savoir différencier pour une case si la valeur stockée est une réponse (c'est-à-dire un entier entre 1 et n^2 ou s'il s'agit d'une valeur par défaut pour les cases qui n'ont pas encore été remplies) ;
3. modifier le contenu des cases.

Question 1. Ecrivez une fonction dont la spécification est la suivante :

Entrée l'ordre de la grille.

Sortie une grille vide : c'est-à-dire qu'aucune valeur n'est stockée dans la grille et toutes les cases sont modifiables.

Le module `parser` permet de lire dans des fichiers textes des grilles de sudoku. Le format des fichiers est le suivant :

1. La première ligne contient l'ordre n du sudoku ;
2. les n^2 lignes suivantes représentent les n^2 lignes de la grille ;
3. chaque ligne est composée de n^2 entiers entre 0 et n^2 séparés par exactement un espace et dont le dernier nombre est immédiatement suivi d'un retour à la ligne ;
4. chaque nombre représente la valeur stockée dans la colonne correspondante ;
5. un 0 signifie que la valeur de la case n'est pas fixée dans la grille initiale.

Le module `parser` contient une unique fonction `read` dont la spécification est la suivante :

Entrée une chaîne de caractères contenant le nom du fichier à lire ; si le fichier se trouve dans le même répertoire que le script python, il suffit d'indiquer le nom du fichier sinon il faut fournir un chemin (relatif ou absolu) ; 5 fichiers vous sont fournis sur ARCHE, vous pouvez en créer d'autres mais ils doivent respecter le format décrit ci-dessus.

Sorties La fonction retourne deux choses :

1. un entier n représentant l'ordre du sudoku
2. une liste (`list`) de tuples contenant les cases pour lesquelles une valeur est fournie ; chaque tuple contient 3 entiers : i) le premier entier est la ligne de la case (entre 0 et $n^2 - 1$) ; ii) le second entier est la colonne de la case (entre 0 et $n^2 - 1$) ; iii) le troisième entier est la valeur stockée dans la case (entre 1 et n^2).

Le code ci-dessous donne un exemple d'utilisation du module :

```
import parser

n, valeurs = parser.read('sudoku_4_4_1.txt')
```

Question 2. Ecrire une fonction dont la spécification est la suivante :

Entrée une chaîne de caractères représentant le nom du fichier contenant l'instance de sudoku.

Sortie une grille (respectant la structure de données que vous avez définies) contenant la grille décrite dans le fichier.

3 Représentation graphique d'une grille de sudoku

Le but de cette partie est d'afficher de manière graphique une grille. Pour cela, le module `render` vous est fourni. Ce module contient les fonctions suivantes :

- `draw_sudoku_grid(n)` affiche une grille vide d'ordre n ;
- `write(line, column, value, color)` affiche la valeur `value` dans la case `(line, column)` (les lignes et les colonnes sont numérotées de 0 à $n^2 - 1$; la case `(0, 0)` étant dans le coin supérieur gauche) dans la couleur `color` (une chaîne de caractères représentant la couleur).
- `wait_quit()` attend la fermeture de la fenêtre ; il doit s'agir de la dernière instruction de votre programme pour que la fenêtre graphique ne se ferme pas automatiquement.

Un exemple d'utilisation du module est :

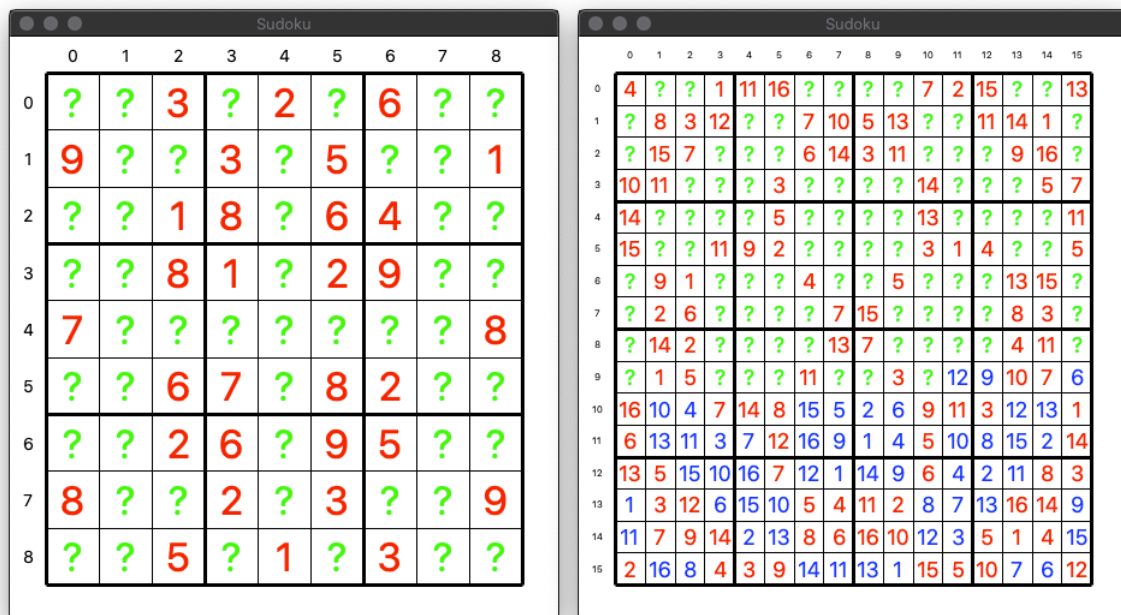
```
import render

render.draw_sudoku_grid(2)

render.write(0, 0, 1, "red")
render.write(3, 2, 12, "blue")
render.write(1, 1, "?", "green")

render.wait_quit()
```

Question 3. Ecrire une fonction qui prend en paramètre une grille et qui l'affiche graphiquement tel que :
i) les nombres fixés soient en rouge ; ii) les nombres "non fixés" soient en vert ; iii) les cases pour lesquels aucune valeur n'est fournie soit un "?" vert.



Remarque : Cette fonction vous servira principalement à afficher l'ensemble de la grille au départ. Durant l'exécution du programme, il sera plus efficace de ne répercuter que les changements à l'aide de la fonction `write` du module `render`.

4 Jouer au sudoku

Le but est de mettre en place une partie de sudoku. Votre programme devra avoir les caractéristiques suivantes :

1. il faut demander à l'utilisateur le nom du fichier contenant la grille, construire la grille initiale et l'afficher ;
2. le programme doit s'arrêter lorsque la grille est complète et afficher un message de félicitation ² ;
3. à chaque tour, il faut demander à l'utilisateur : i) le numéro de la ligne ; ii) le numéro de la colonne ;
4. si les coordonnées ne sont pas correctes (en dehors de l'intervalle, coordonnées d'une case que l'on ne peut pas modifier, valeur qui n'est pas un nombre...), le programme doit indiquer qu'il y a une erreur, quelle est l'erreur et recommencer la saisie des coordonnées ;
5. lorsqu'une case est choisie, le programme doit afficher les valeurs possibles qui peuvent être attribuées à la case (il faudra prévoir une valeur qui représente le fait que l'on "retire" la valeur de la case) ;
6. comme précédemment, il faut vérifier que la saisie est correcte ;
7. il faut que l'affichage graphique reflète les modifications ³ ;
8. si à un moment, l'utilisateur n'entre aucune valeur et appuie juste sur retour, il faut annuler toutes les saisies et revenir au début du tour.

Voici un exemple d'affichage des interactions :

```
Entrez le nom du fichier (entrée pour prendre le fichier par défaut): sudoku_9_9_1.txt
Il reste 49 cases à remplir
Entrez un numéro de ligne ou entrée pour annuler : 0
Entrez un numéro de colonnes ou entrée pour annuler :
Il reste 49 cases à remplir
Entrez un numéro de ligne ou entrée pour annuler : 0
Entrez un numéro de colonnes ou entrée pour annuler : 1
Les valeurs possibles sont : 0 4 5 7 8
Entrez une valeur possible ou appuyez sur entrée pour annuler le coup : 4
Il reste 48 cases à remplir
Entrez un numéro de ligne ou entrée pour annuler : 1
Entrez un numéro de colonnes ou entrée pour annuler : 1
Les valeurs possibles sont : 0 2 6 7 8
Entrez une valeur possible ou appuyez sur entrée pour annuler le coup :
Il reste 48 cases à remplir
Entrez un numéro de ligne ou entrée pour annuler : adz
Il faut entrer un nombre entier
Entrez un numéro de ligne ou entrée pour annuler : 1
Entrez un numéro de colonnes ou entrée pour annuler : 0
Il n'est pas possible de modifier cette case
Entrez un numéro de ligne ou entrée pour annuler : 31
Le numéro de ligne n'est pas correct. Il doit être entre 0 et 8
Entrez un numéro de ligne ou entrée pour annuler : 3
Entrez un numéro de colonnes ou entrée pour annuler : 1
Les valeurs possibles sont : 0 3 5
Entrez une valeur possible ou appuyez sur entrée pour annuler le coup : 3
```

...

-
2. On supposera que les grilles que l'on traite ont une solution.
 3. Vous n'afficherez que les modifications, évitez de réafficher toute la grille.

5 Mis en place d'un historique

Le but est maintenant de mettre en place un historique des coups joués et de permettre à l'utilisateur d'annuler le dernier coup joué. Votre programme doit :

- à chaque coup joué, sauvegarder le coup (case, ancienne valeur, nouvelle valeur) dans l'historique ;
- si l'historique n'est pas vide, affichez à l'écran le dernier coup joué et demander à l'utilisateur s'il veut l'annuler avant de demander la case où il veut jouer ;
- si l'utilisateur veut annuler le coup, il faut pour le tour courant, mettre à jour la grille, retirer le coup de l'historique, l'avant-dernier coup joué s'il existe devient le prochain coup à annuler.

Voici un exemple d'affichage des interactions :

```
Entrez le nom du fichier (entrée pour prendre le fichier par défaut): sudoku_9_9_1.txt
Il reste 49 cases à remplir
Entrez un numéro de ligne ou entrée pour annuler : 0
Entrez un numéro de colonnes ou entrée pour annuler : 0
Les valeurs possibles sont : 0 4 5
Entrez une valeur possible ou appuyez sur entrée pour annuler le coup : 4
Il reste 48 cases à remplir
Dernier coup joué = ( 0 , 0 ) : 0 -> 4
Voulez-vous l'annuler ? [O]ui / [N]on : N
Entrez un numéro de ligne ou entrée pour annuler : 3
Entrez un numéro de colonnes ou entrée pour annuler : 4
Les valeurs possibles sont : 0 3 4 5 6
Entrez une valeur possible ou appuyez sur entrée pour annuler le coup : 6
Il reste 47 cases à remplir
Dernier coup joué = ( 3 , 4 ) : 0 -> 6
Voulez-vous l'annuler ? [O]ui / [N]on : 0
Il reste 48 cases à remplir
Dernier coup joué = ( 0 , 0 ) : 0 -> 4
Voulez-vous l'annuler ? [O]ui / [N]on : N
Entrez un numéro de ligne ou entrée pour annuler : 8
Entrez un numéro de colonnes ou entrée pour annuler : 0
Les valeurs possibles sont : 0 6
Entrez une valeur possible ou appuyez sur entrée pour annuler le coup : 6
Il reste 47 cases à remplir
Dernier coup joué = ( 8 , 0 ) : 0 -> 6
Voulez-vous l'annuler ? [O]ui / [N]on : 0
Il reste 48 cases à remplir
Dernier coup joué = ( 0 , 0 ) : 0 -> 4
Voulez-vous l'annuler ? [O]ui / [N]on : 0
Il reste 49 cases à remplir
Entrez un numéro de ligne ou entrée pour annuler : 0
Entrez un numéro de colonnes ou entrée pour annuler : 0
Les valeurs possibles sont : 0 4 5
Entrez une valeur possible ou appuyez sur entrée pour annuler le coup : 4
Il reste 48 cases à remplir
Dernier coup joué = ( 0 , 0 ) : 0 -> 4
Voulez-vous l'annuler ? [O]ui / [N]on : ddze
Veuillez répondre 'O' ou 'N'
Voulez-vous l'annuler ? [O]ui / [N]on :
...
```

6 Résolution automatique

Dans cette partie, après avoir demandé le fichier à l'utilisateur et afficher la grille initiale, il faut demander à l'utilisateur s'il veut jouer ou résoudre automatiquement la grille :

```
Entrez le nom du fichier (entrée pour prendre le fichier par défaut): sudoku_9_9_1.txt
Voulez-vous "Jouer" ou "Résoudre" ? Résoudre
Grille résolue.
```

Pour résoudre le problème, nous allons utiliser une stratégie simple dite "brute force"⁴. C'est-à-dire que l'on va essayer toutes les possibilités jusqu'à ce que l'on trouve la solution. Une telle stratégie peut s'implémenter assez facilement avec une fonction récursive de type "backtracking". C'est-à-dire que l'on prend des décisions les unes après les autres. Si on arrive dans un cas où l'on ne peut pas continuer, on annule la dernière décision et on essaie une nouveau choix. On s'arrête quand on a atteint le statut que l'on recherche. Il vous faut donc implémenter la fonction récursive suivante :

```
def solve_recursive(grid, freecells):
    ...
```

Les entrées sont :

- grid : la grille que l'on remplit
- freecells : une liste des cases auxquelles on peut attribuer une valeur

La sortie de la fonction est un booléen : True si on a réussi à remplir la grille en affectant des valeurs aux cases de freecells, False sinon.

Une exécution générique de la fonction est alors ⁵ :

1. choisir une case à remplir ;
2. choisir une valeur à attribuer à la case ;
3. mettre à jour la grille ;
4. faire l'appel récursif ;
5. si le retour de l'appel récursif retourne False, on essaie une autre valeur à affecter à la case ;
6. si on a testé toutes les valeurs possibles que pouvait prendre la case, c'est qu'un choix fait auparavant n'est pas bon, on fait donc un retour en arrière⁶ en arrêtant l'exécution de la fonction en revoyant la bonne valeur⁷.

Il faudra répercuter sur l'affichage graphique les tentatives de l'algorithme.

4. "force brute" en bon français.

5. Il faudra aussi définir les cas d'arrêt de votre fonction ainsi que l'appel initial à la fonction.

6. backtrack en anglais d'où le nom de l'algorithme : backtracking.

7. Rappel : il n'y a que deux possibilités True ou False.

7 Consignes

- Le projet se fait en monôme.
- Le langage de programmation est Python. Un projet rendu en tout autre langage aura la note 0.
- Il est interdit d'utiliser des notions de Python non vues en cours. Le non respect de cette consigne entraînera une note inférieure à 10.
- Le programme ne doit pas lever d'exception. Il vaut mieux ne pas mettre une fonctionnalité qui causerait un bug. Le non respect de cette consigne entraînera une note inférieure à 10.
- Tout code recopié d'internet ou d'une source quelconque et non identifié comme tel sera considéré comme du plagiat. La note du projet sera 0.
- Un projet qui serait trop constitué de code recopié d'internet ou d'une autre source ne sera pas considéré comme un travail personnel et entraînera la note de 0.
- Si des codes rendus sont avérés être des copies les uns des autres, un seul projet sera corrigé et les étudiants répartiront les points entre eux.
- Si les noms des variables et des fonctions ne sont pas explicites, la note finale sera inférieure à 10.
- S'il n'y a pas un découpage à minima des fonctions, la note sera inférieure à 10.
- L'utilisation de variables globales (autrement dit une fonction ne doit pas accéder à une variable qui n'est ni un de ses paramètres, ni déclarée localement) entraînera une note inférieure à 10.
- Le projet sera à déposer sur ARCHE dans un dépôt prévu à cet effet. L'ensemble du code sera à placer dans un fichier `sudoku.py` (seul ce fichier sera à rendre).
- La date limite de rendu du projet est le **dimanche 17 janvier 2021 à 23h59m59s**. Tout retard entraînera un malus de 10 points par heure de retard.

Un site donnant quelques conseils sur ce qu'est un code "propre"⁸ :

<https://damien.pobel.fr/post/clean-code/>

8. Les remarques vont bien au-delà de ce que l'on fait dans ce cours.