Spring 2026

Programmation 2

TP 7 — *Lisez attentivement l'ensemble du document.* — *Nabil Mustafa*

Consignes

(a) *L'objectif est d'apprendre la programmation en Python.* ***Inutile de se précipiter !*** *Prenez votre temps et terminez correctement tous les TP précédents avant d'aborder celui-ci.*

(b) *Pour réussir,* ***lisez le texte attentivement et lentement****. Ne parcourez pas les consignes en diagonale. Lisez chaque phrase une fois, puis une deuxième. Si un passage reste flou après plusieurs lectures, c'est le moment idéal pour poser une question au professeur. C'est ainsi que l'on apprend efficacement.*

(c) *Avant de commencer,* ***mettez votre téléphone en mode « ne pas déranger » et rangez-le dans votre sac****. Chaque notification consultée brise votre concentration et vous oblige à reprendre le fil de votre raisonnement, ce qui allonge considérablement le temps nécessaire pour terminer l'exercice et favorise les erreurs.*

(d) *Adoptez une approche progressive : lisez une instruction, puis exécutez-la immédiatement. Évitez de lire tout le TP d'un coup.* ***Avancez étape par étape pour rester concentré sur l'action immédiate et ne pas vous sentir submergé par un long texte.***

(e) *Souvenez-vous : les étudiants qui réussissent le mieux ne sont pas ceux qui « finissent en premier », mais ceux qui* ***lisent avec soin, posent des questions lorsqu'ils bloquent et restent concentrés*** *jusqu'au bout.*

Votre objectif aujourd'hui n'est pas d'aller vite, mais de comprendre profondément.

Tâche 0-a: Exercises

(a) Connectez-vous à `LINUX` avec votre compte utilisateur.

(b) Téléchargez le Jupyter Notebook fichier `Lec5-boucles2.ipynb`, disponible sur le site web du module.

(c) Pour le lire, ouvrez un `terminal` dans le *même dossier* que le fichier `Lec5-boucles2.ipynb`, et exécuter le command :

```
jupyter-notebook Lec5-boucles2.ipynb
```

(d) Pour exécuter un code Python dans le Jupyter Notebook, cliquez sur le code et appuyez sur "SHIFT + ENTER".

(e) **Lisez toutes les informations** dans cette Jupyter Notebook.

(f) **Faites tous les exercices** dans cette Jupyter Notebook.

Rappel: Sudoku

Le but du jeu est de remplir la grille de taille 9×9, avec une série de chiffres allant de 1 à 9 tous différents, qui ne se trouvent jamais plus d'une fois

- dans une même **ligne** (dont il y a 9), ou
- dans une même **colonne** (dont il y a 9), ou
- dans une même **bloc** (dont il y a 9)—les blocs ètant des carrés de 3×3.

Voici une grille remplie :

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

bloc

ligne

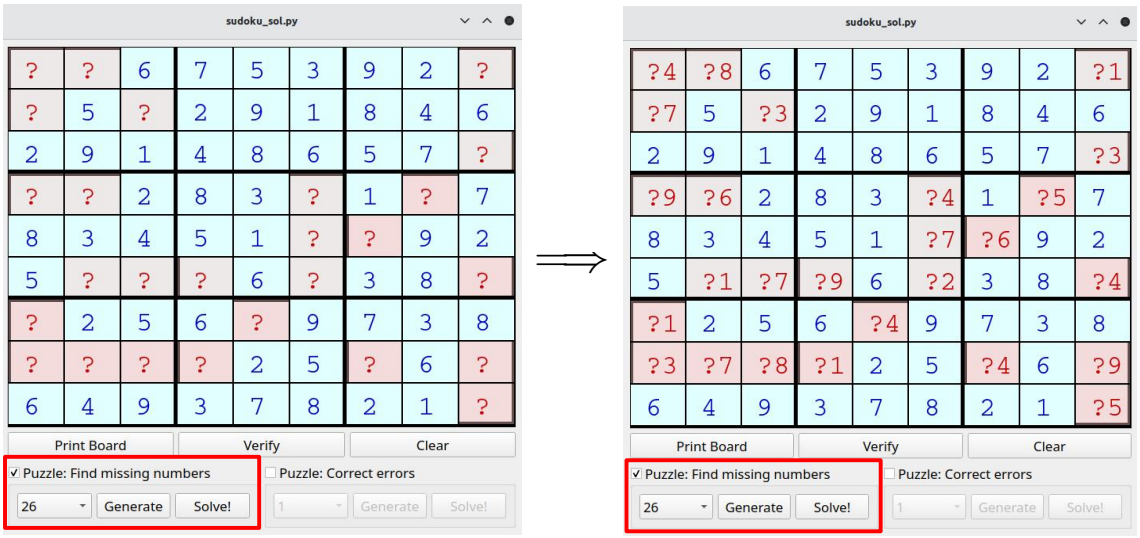
colonne

La règle du jeu :

chaque ligne, colonne et bloc ne doit contenir qu'une seule fois tous les chiffres de 1 à 9.

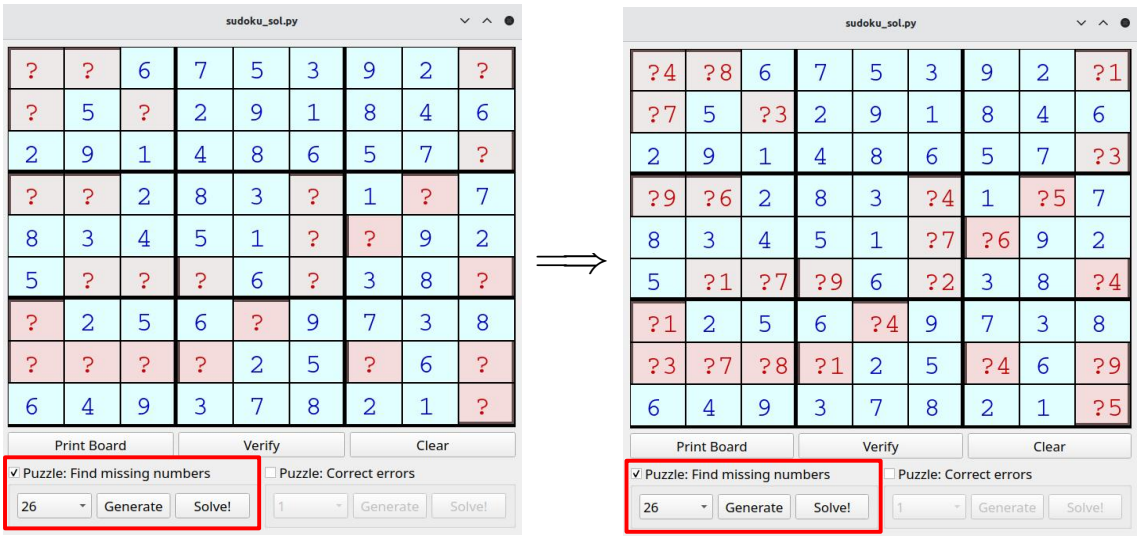
Le code que je vous ai donné peut générer de nombreuses nouvelles puzzles, où certains chiffres sont manquants et doivent être trouvés.

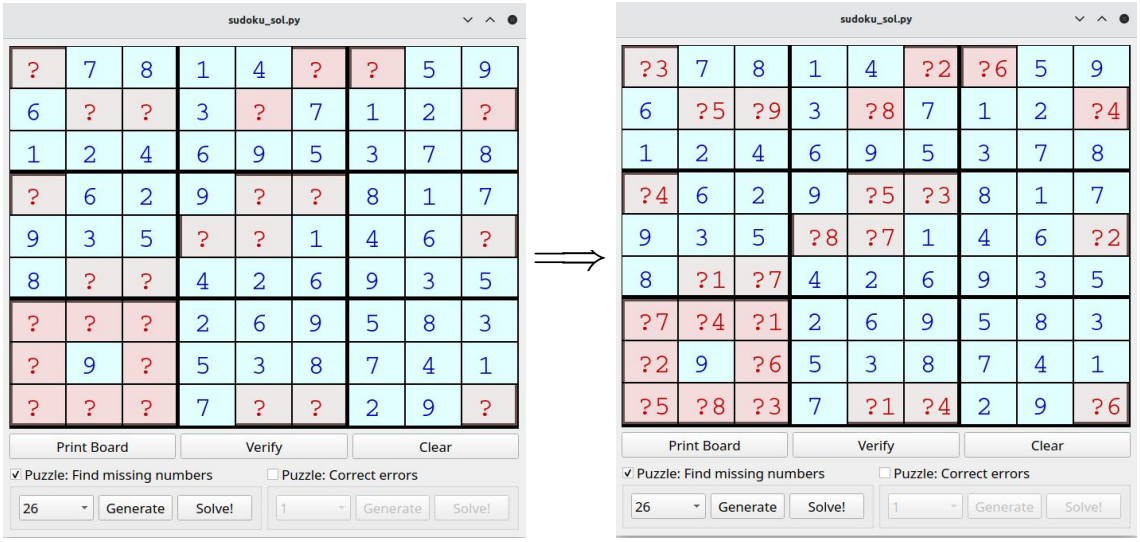
Le but de ce TP est d'écrire un algorithme pour résoudre chaque puzzle. Voici une exemple:

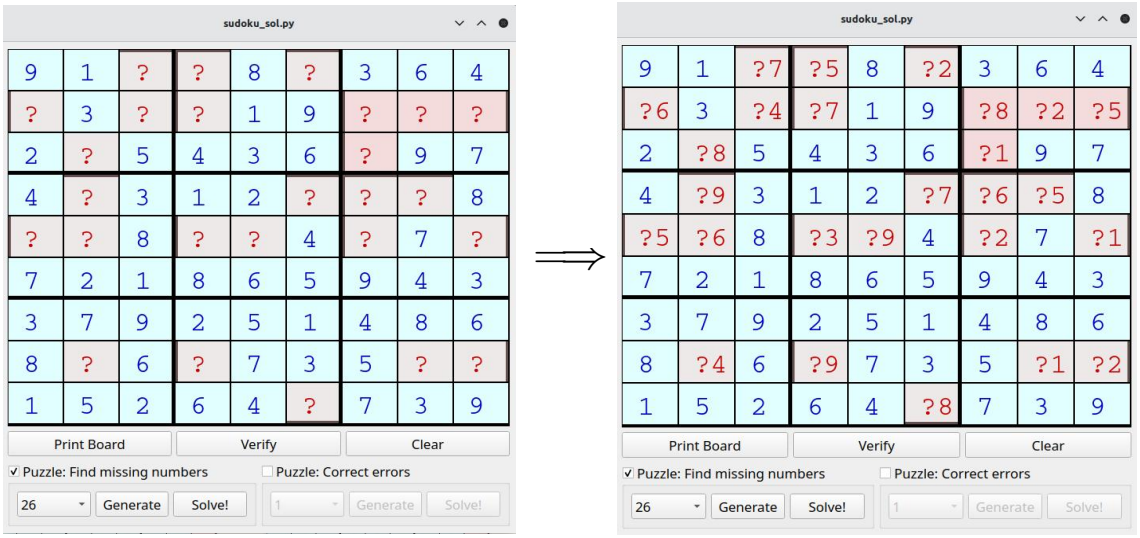


Rappel: Sudoku

Voici quelques puzzles, et leurs solutions trouvé par l'algorithme:







Sudoku en Python

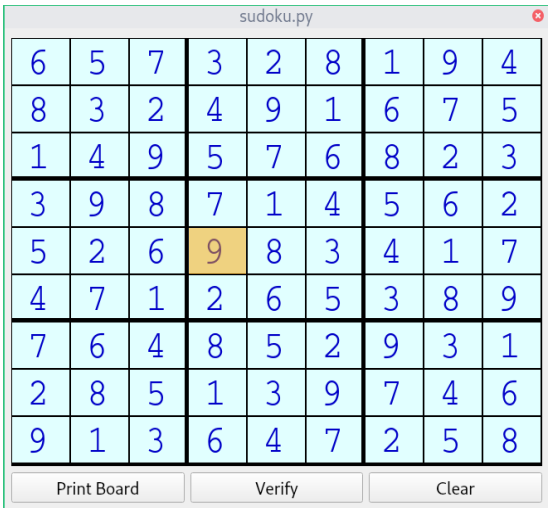
Le tableau de Sudoku est stocké dans une liste de listes nommé `data`, oú chaque liste contenant une ligne de tableau. Voici une exemple:

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

```
data = [ [1, 7, 5, 8, 9, 3, 4, 6, 2],
         [2, 6, 4, 1, 5, 7, 9, 3, 8],
         [8, 3, 9, 2, 4, 6, 5, 7, 1],
         [4, 1, 6, 5, 7, 8, 3, 2, 9],
         [9, 2, 3, 4, 6, 1, 7, 8, 5],
         [5, 8, 7, 9, 3, 2, 6, 1, 4],
         [3, 4, 2, 6, 1, 5, 8, 9, 7],
         [6, 5, 1, 7, 8, 9, 2, 4, 3],
         [7, 9, 8, 3, 2, 4, 1, 5, 6] ]
```

Cliquez sur une cellule pour imprimer sa position dans la liste `data`.

Par exemple:



$\Longrightarrow$ `data[4][3] = 9`

Tâche 1: Exécuter Python

(a) Connectez-vous à `LINUX` avec votre compte utilisateur.

(b) **Téléchargez** le fichier `sudoku.py`.

(c) **Exécuter** le code avec `Python`: ouvrez un `terminal` dans le même dossier que le fichier `sudoku.py`, et exécuter le command :

```
python3 sudoku.py
```

(d) **Éditer** ce fichier avec l'éditeur de votre choix. Par exemple, ouvrez un `terminal` dans le même dossier que le fichier `sudoku.py`, et exécuter le command:

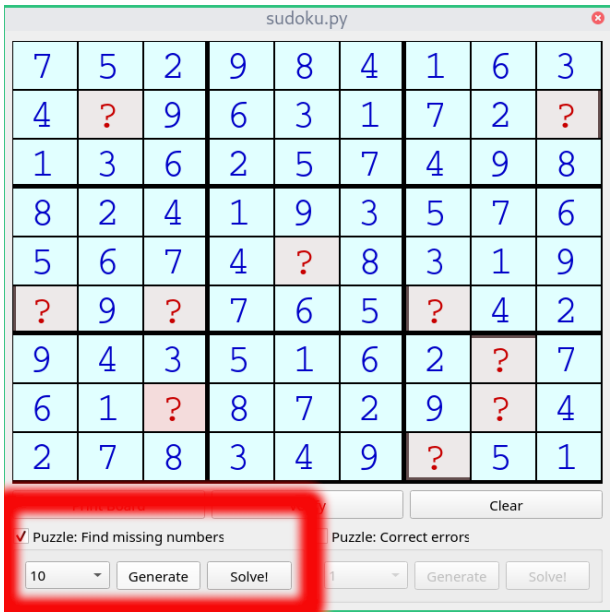
```
kate sudoku.py
```

Appuyez sur `F4` (ou sur menu `Outils` → `Afficher le terminal`) à voir le terminal où vous pouvez exécuter le code avec le command :

```
python3 sudoku.py
```

Vous verrez cette interface que j'ai créée pour vous.

Dans ce TP, nous ferons uniquement le puzzle du côté gauche → Puzzle : Find missing numbers.



Tâche 2: Ecrivez l'algorithme pour remplir valeurs manquantes

Votre objectif est d'écrire la fonction

`solveMissingPuzzle(data)`

qui doit remplir le tableau `data` avec les correct valeurs pour chaque cellule.

Notez que:

- la fonction modifie la liste d'entrée `data` elle-même, elle ne renvoie rien ;
- un 0 dans une cellule indique une valeur manquante, et la fonction doit trouver la valeur correcte correspondante et la remplacer par celle-ci ;
- cette fonction est appelée lorsque vous appuyez sur la bouton Solve! dans la partie Puzzle: Find missing numbers.

```
def solveMissingPuzzle(data):
    '''un 0 dans une cellule de data indique une valeur manquante. La fonction
       doit trouver la valeur correcte correspondante et la remplacer par celle-ci.'''

    #ECRIVEZ VOTRE CODE ICI
```

Pour ce problème, vous devez imaginer votre propre algorithme.

Réfléchissez aux stratégies à utiliser pour trouver les chiffres manquants!

Voici un exemple de input `data`, et la même liste après solveMissingPuzzle a rempli les valeurs:

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

⟹

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

```
data[0]:   [7, 0, 3, 8, 6, 4, 5, 1, 9]
data[1]:   [6, 8, 4, 1, 5, 9, 7, 0, 3]
data[2]:   [0, 1, 9, 2, 7, 3, 6, 8, 4]
data[3]:   [3, 5, 2, 7, 4, 0, 9, 6, 0]
data[4]:   [0, 7, 8, 6, 9, 1, 0, 5, 2]
data[5]:   [9, 6, 1, 5, 3, 2, 4, 7, 8]
data[6]:   [0, 0, 7, 0, 0, 6, 2, 9, 5]
data[7]:   [1, 4, 6, 9, 2, 5, 8, 3, 7]
data[8]:   [0, 0, 5, 3, 8, 7, 0, 4, 6]
```

```
data[0]:   [7, 2, 3, 8, 6, 4, 5, 1, 9]
data[1]:   [6, 8, 4, 1, 5, 9, 7, 2, 3]
data[2]:   [5, 1, 9, 2, 7, 3, 6, 8, 4]
data[3]:   [3, 5, 2, 7, 4, 8, 9, 6, 1]
data[4]:   [4, 7, 8, 6, 9, 1, 3, 5, 2]
data[5]:   [9, 6, 1, 5, 3, 2, 4, 7, 8]
data[6]:   [8, 3, 7, 4, 1, 6, 2, 9, 5]
data[7]:   [1, 4, 6, 9, 2, 5, 8, 3, 7]
data[8]:   [2, 9, 5, 3, 8, 7, 1, 4, 6]
```