TP0

On supposera ici que vous avez lu (et compris) les bases de python et que vous pouvez lancer et utiliser un ide comme pyCharm.

Pycharm (l'ide) est installé par défaut dans les salles linux de l'ÉCM. Pour le lancer, ouvrez un terminal (cliquez sur la télé) et tapez : `pycharm.sh` puis entrée. L'éditeur doit se lancer. Lors de la première utilisation il vous demandera plein de trucs, essayer de répondre intelligemment (vous ne pourrez de toute façon pas faire de raccourcis sur le bureau ou dans `/usr/local/bin`, lorsqu'il vous le demandera cliquez annuler).

Le but de ce TP est de vous dérouiller les méninges en algorithmie en essayant de résoudre le problème du sudoku. Nous nous contenterons de résoudre les sudokus très simples, libre à vous d'améliorer le procédé.

Lire un sudoku

Pour nous, un sudoku sera une liste de 9*9 nombres allant de 1 à 9 constituée de la concaténation de chaque ligne :

```
grille_1_complete = [5, 2, 9, 6, 7, 1, 8, 4, 3,
                     3, 1, 8, 4, 9, 2, 6, 5, 7,
                     6, 7, 4, 8, 5, 3, 2, 1, 9,
                     7, 4, 6, 3, 2, 5, 1, 9, 8,
                     2, 9, 3, 1, 8, 6, 4, 7, 5,
                     1, 8, 5, 7, 4, 9, 3, 6, 2,
                     4, 3, 2, 5, 6, 7, 9, 8, 1,
                     9, 6, 7, 2, 1, 8, 5, 3, 4,
                     8, 5, 1, 9, 3, 4, 7, 2, 6]
```

Un élément particulier

On vous demande de commencer par écrire une méthode (`element_sudoku`) permettant d'accéder un élément particulier du sudoku. On supposera en bon informaticien que les lignes et colonnes commencent à 0. La commande ci-après doit ainsi afficher : **`4, 5: 6`**

```
print("4, 5:", element_sudoku(4, 5, grille_1_complete))
```

Accéder aux lignes, colonnes et sous-matrices

Ecrivez 3 méthodes permettant de rendre une ligne, une colonne ou une sous-matrice particulière sous forme de liste. Les commandes ci-après :

```
print("ligne 4", ligne_sudoku(4, grille_1_complete))
print("colonne 5", colonne_sudoku(5, grille_1_complete))
print("sous-matrice contenant 4, 5", matrice_sudoku(4, 5,
grille_1_complete))
```

Afficheront :

```
ligne 4 [2, 9, 3, 1, 8, 6, 4, 7, 5]
colonne 5 [1, 2, 3, 5, 6, 9, 7, 8, 4]
sous-matrice contenant 4, 5 [3, 2, 5, 1, 8, 6, 7, 4, 9]
```

Sudoku correct ?

Utilisez les méthodes d'accès aux lignes, colonnes et sous-matrices que vous avez crées dans la partie précédente pour écrire une méthode permettant de savoir si un sudoku est correct ou pas.

Faites des tests avec des sudokus correct et des sudoku incorrect. Pour ma part :

```
print("sudoku correct ?", sudoku_correct(grille_1_complete))
```

répond :

```
sudoku correct ? True
```

Afficher un sudoku

Finalement, affichons le sudoku complet. Laissez libre court à votre imagination. Pour ma part, la commande :

```
affiche_sudoku(grille_1_complete)
```

affiche :

```
529 671 843
318 492 657
674 853 219

746 325 198
293 186 475
185 749 362

432 567 981
967 218 534
851 934 726
```