Programmation 2 Spring, 2026

TD 9: Sets, Tuples

1. Une fois les lignes de code suivantes exécutées, quelles valeurs sont stockées dans `output_set` ?

```
input_list = [3,1,4,1,5,9,2,6,5,3,5,9]
output_set = set([])
for i in input_list:
    output_set.add(i)
```

2. Écrivez un code en une seule ligne pour supprimer les doublons d'une liste.

   `Example:` Pour `[1, 1, 2, 3, 4, 2]`, il faut calculer `[1,2,3,4]`.

3. Écrivez une fonction `myenumerate` qui prend une liste et renvoie une liste de *tuples* contenant (index, item) pour chaque élément de la liste. Faites-le de deux manières :

   i) avec une boucle, et

   ii) avec une compréhension de liste.

   `Example:` Pour `[1, 1, 2, 8]`, il faut retourner `[(0,1), (1,1), (2,2), (3,8)]`.

4. Écrivez une fonction `mytriplets` qui prend un nombre n en argument et renvoie une liste de *toutes* les triplets tels que la somme des deux premiers éléments du triplet soit égale au troisième élément. Chaque nombre utilisé dans un triplet doit être supérieur à 0, et inférieur ou égal à n.

   Veuillez noter que (a, b, c) et (b, a, c) représentent le même triplet.

   Donnez deux codes : le premier utilisant des boucles, le second utilisant des compréhensions de liste.

   `Example:` Pour `n = 5`, il va retourner `[(1,1,2), (1,2,3), (1,3,4), (1,4,5), (2,2,4), (2,3,5)]`.

5. La fonction `zip`, déjà présente en Python, prend deux itérables en entrée et renvoie un itérateur contenant une paire du premier et du deuxième itérables.

   Le i-ième tuple est : $\Big($**i-ième élément du premier itérable, i-ième élément du deuxième itérable**$\Big)$.

   `Example:`

```
A = [3,5,1, "hello"]
B = [2,9]

C = zip(A,B)

print(type(C))

D = set(C)
print(D)
```

⇓

```
<class 'zip'>
{(3, 2), (5, 9)}
```

Supposons que nous ayons deux listes, A et B, qui donnent les coordonnées x et y d'un ensemble de points. Créez **une liste** avec les coordonnées (x, y) sous forme de tuple à l'aide de la fonction `zip`.

6. (**Sudoku TP.**) La grille de Sudoku est stockée dans une liste de listes nommée `data`, où chaque liste contient une ligne de la grille. Voici un exemple :

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

Un 0 dans une cellule indique une valeur manquante.

Étant donné l'input `data` et une ligne i ne contenant qu'un seul zéro, comment trouver la valeur manquante ?

7. (**Sudoku TP.**) La grille de Sudoku est stockée dans une liste de listes nommée `data`, où chaque liste contient une ligne de la grille. Voici un exemple :

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

Un 0 dans une cellule indique une valeur manquante.

Étant donné l'input `data` et une colonne j qui ne contient qu'un seul 0, comment déterminer la valeur manquante dans cette colonne ?

8. (**Sudoku TP.**) La grille de Sudoku est stockée dans une liste de listes nommée `data`, où chaque liste contient une ligne de la grille. Voici un exemple :

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

Un 0 dans une cellule indique une valeur manquante.

Notez qu'il y a des blocs de taille 3×3 :

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

Étant donné l'entrée `data` et une cellule (i, j), déterminez si **le bloc contenant** (i, j) ne contient qu'un seul 0 à la position (i, j), et le cas échéant, trouvez la valeur manquante.

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

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

Un 0 dans une cellule indique une valeur manquante.

Étant donné l'input `data` et une cellule (i, j), comment déterminer si cette cellule n'a qu'un seul candidat possible après avoir examiné les chiffres présents dans sa ligne, sa colonne et son bloc ?

Il faut utiliser un `set`.