Exercice 1

Ouvrir le fichier `prepa_oral.csv` et observer le format de chaque ligne.

1. A partir de ce fichier, créer 1 liste qui contient des dictionnaires dont les clés seront les noms de colonne. On veut un dictionnaire par ligne de données.
2. Combien de séances sont prévues au mois de juin ?
3. Combien de minutes de séance de SI sont prévues ? On ne suppose pas que les séances débutent ou terminent à une heure ronde (prendre en compte les éventuelles minutes dans les heures données).
4. Quel type de séance revient le plus souvent ? Et en nombre de minutes ?

Exercice 2

1. Après avoir lu la documentation des tirages aléatoires du memento, donner un test python qui renvoie vrai avec une probabilité $p \in [0, 1]$.
2. On souhaite maintenant simuler informatiquement des variables aléatoires.
   (a) On considère $X_1 \hookrightarrow B(p)$. Rappeler les valeurs que peut prendre X_1 et créer une fonction python d'argument p et qui renvoie une valeur de X_1 avec la bonne probabilité.
   (b) Même question avec $X_2 \hookrightarrow B(n, p)$ où n est un entier naturel.

Exercice 3

On considère un graphe non orienté G dont l'ensemble des sommets est $[\![1, n]\!]$ et qui sera décrit par sa liste d'adjacence (de longueur n).

Une bonne illustration pour notre exercice basé sur l'image suivante

Les sommets sont les pays, et deux sommets sont liés s'ils ont une frontière commune (même pour les pays non connexes).

On se pose la question d'un choix judicieux de couleurs pour notre carte. Par soucis de lisibilité, on ne veut pas que deux pays limitrophes (reliés par une arête) soient coloriés de la même couleur.

On représente les couleurs disponibles par des entiers naturels (à partir de 0).

Un **coloriage** est alors une liste `c` de longueur n telle que le pays i soit colorié de la couleur `c[i]` et en respectant notre contrainte.

Initialement, on crée la liste `c` de longueur n qui contient uniquement `None` (aucune couleur affectée).

1. Notons `L` la liste d'adjacence. Quelle propriété vérifiée par `L` traduit le fait que le graphe n'est pas orienté ?
2. On considère dans cette question qu'un partie du coloriage est déjà effectué : on a colorié les sommets $0, 1 \ldots, s-1$ avec $s < n - 1$.

   On note `vois` les voisins du sommets s et on cherche la couleur la plus petite possible possible (notre algorithme est glouton) pour colorier ce sommet.

   (a) Montrer qu'on peut trouver une couleur inférieure où égale à `len(vois)` qui convienne.

   (b) Pour déterminer cette plus petite couleur convenable, on crée une liste de longueur `len(vois) + 1` dont tous les éléments sont initialisés à `True` (la couleur est disponible) puis on parcourt la liste `vois`.

   Implémenter cette idée dans une fonction `plus_petite_couleur(L, s, c)` qui prend comme argument une liste d'adjacence `L`, un numéro de sommet `s` et un coloriage partiel `c` et retourne la plus petite couleur possible pour colorier `s`.

3. Créer la fonction `colorie(L)` qui prend comme argument une liste d'adjacence et retourne le coloriage construit par la méthode précédente.
   Tester avec la liste `L = [[2], [3], [0,3], [1,2]]` . L'algorithme glouton est-il optimal ?

Exercice 4 (Math 2, 2019)

1. Créer une fonction `nombre_depuis_liste` qui prend en argument une liste de chiffres en base 10 et retourne l'entier correspondant. Les chiffres sont donnés dans l'ordre de lecture. Par exemple `nombre_depuis_liste([3,2,1,0])` retourne l'entier 3210
2. Créer une fonction `liste_depuis_nombre` qui effectue l'opération réciproque.
3. Créer une fonction `distincts(L1, L2)` qui prend deux listes en argument et retourne True les éléments de `L1` sont distincts deux à deux et ne sont pas présents dans `L2`

Exercice 5 (Math 2, 2021)

1. Créer une fonction `est_palindrome` qui prend une liste `L` comme argument et retourne le booléen indiquant si `L` est palindromique.
2. Créer une fonction `decompose` qui prend deux entiers `n, b` (dans cet ordre) avec $b \geqslant 2$, et qui retourne la liste des chiffres de n en base b, le chiffre des unités étant en première position.
3. Créer une fonction `bases` qui prend un entier `n` comme argument et retourne la liste de tous les entiers $b < n$ tels que l'écriture de n en base b est palindromique.
4. Trouver les entiers entre 3 et 1000 qui ont le plus de décompositions palindromiques (parmi les bases de la question précédente) et ceux qui en ont le moins.
5. Que remarquez-vous sur le nombre minimal de base(s) ainsi trouver ? justifier que $n > 2$ est toujours palindromique en base $n - 1$

Exercice 6 (Math 2, 2023)

On considère un graphe G donné par sa liste d'adjacence `L`.
Par convention, les sommets de G sont les entiers naturels entre 0 et `len(L) - 1`.

1. Tracer sur feuille le graphe dont la liste d'adjacence est `[[1,3], [0,2,3], [1,3], [0,1]]`. Est-ce un graphe orienté ?
2. Créer une fonction `est_oriente(L)` qui prend comme argument une liste d'adjacence `L` et retourne le booléen indiquant si le graphe G correspondant est un graphe orienté ou non.
3. Créer une fonction `non_oriente(L)` qui prend comme argument une liste d'adjacence et retourne la liste d'adjacence du graphe non orienté correspondant.
4. (Bonus) Créer une fonction qui transforme une liste d'adjacence en la matrice d'adjacence correspondante.

Exercice 7 (Math 2, 2023)

1. On considère le code suivant.

```
def fonction_mystere(n):
    L = [k for k in range(2, n + 1)]
    for p in range(2, int(math.sqrt(n)) + 1):
        L2 = []
        for k in L:
            if k == p or k % p != 0:
                L2.append(k)
        L = L2
    return L
```

   Quel est le type de la valeur renvoyée par `fonction_mystere` ?
   Tester sur plusieurs exemples. Que renvoie `fonction_mystere(n)` ?
2. Vérifier que $k(k-1)+41$ est premier pour tout $k \in [\![2, 40]\!]$
3. Écrire une fonction `test` prenant un entier p comme argument et renvoyant un booléen indiquant si $k(k-1)+p$ est premier pour tout $k \in [\![2, p-1]\!]$.
4. Quels sont les entiers inférieurs ou égaux à 100 qui passent le test précédent ?