Données en tables

On considère les deux fichiers `Prenoms1.csv` et `Prenoms2.csv`.

- Le premier fichier contient les prénoms d'enfants nés à Strasbourg en 2008.
- Le second fichier contient ceux d'enfants nés à Rennes en 2007.

À chaque ligne est indiqué le sexe du prénom, le prénom et le nombre de fois qu'il a été donné.
Pour informations, ces données proviennent d'un fichier obtenu sur https://www.data.gouv.fr/.
Un troisième fichier, `Tables.py`, contient des lignes permettant d'importer le premier fichier à l'aide du module `csv`. Son contenu initial est le suivant :

```
import csv
with open('Prenoms1.csv') as csvfile1:
    data1 = list(csv.reader(csvfile1))
```

Avec ces lignes, on ouvre le fichier `Prenoms1.csv` à l'aide d'un module permettant de manipuler les fichiers csv et on enregistre son contenu dans une liste `data1`.
Le fichier `Tables.py` est celui à modifier et à compléter dans cette activité.

1. On va tout d'abord reprendre ces données pour obtenir une liste de p-uplets (`tuple`), en transformant en type `int` les données numériques.

   On récupère tout d'abord la première ligne de données, qui contient les champs de données. Ajouter la ligne suivante dans le code :

   ```
   champs = tuple(data1[0])
   ```

   On a :

   ```
   >>> champs
   ('Sexe', 'Prénom', 'Nombre')
   ```

   On va s'aider de variables pour pour récupérer, pour chaque ligne de donnée, chacun des champs. Ajouter les lignes suivantes dans le code :

   ```
   indice={champs[i]:i for i in range(len(champs))}
   s =indice['Sexe']
   # s est l'indice du champ 'Sexe' de la table dans les données
   p =indice['Prénom']
   # p est l'indice du champ 'Prénom' de la table dans les données
   nb=indice['Nombre']
   # nb l'indice du champ 'Nombre' de la table dans les données
   ```

   Enfin, on transforme les données numériques en de véritables entiers (de type `int`) et on crée la liste `table1` qui contiendra toutes les données, qui sera celle que nous manipulerons par la suite.

   Ajouter les lignes suivantes dans le code :

```
table1 = []
for ligne in data1[1:]:
    ligne[nb]=int(ligne[nb])
    table1.append(tuple(ligne))
```

Ainsi, chaque ligne de la liste `table1` est un `tuple` qui contient 3 éléments : le sexe du prénom (l'élément d'indice `s`), le prénom (indice `p`) et le nombre de fois (indice `nb`) qu'il a été donné dans l'année. On utilisera maintenant la variable `table1` pour parcourir les données.

Ainsi par exemple :

```
>>> ligne = table1[0] # première ligne de données
>>> ligne[p] # prénom
'Emma'
>>> ligne[s] # sexe
'FEMME'
>>> ligne[nb] # nombre de fois qu'il a été donné
52
```

2. Ajouter et exécuter un code permettant d'afficher la liste des prénoms qui ont été donnés au moins 30 fois.
3. Les données dans la table sont triées d'abord par sexe, puis par nombre de naissances.

   Pour la trier seulement par prénom, on peut utiliser la fonction `sorted`, avec le paramètre `key` de la manière suivante (ajouter les lignes suivantes dans le code) :

```
table_triee=sorted(table1,key=lambda ligne:ligne[p])

# Tri par ordre décroissant :
table_triee_d=sorted(table1,key=lambda ligne:ligne[p],reverse=True)
```

   En utilisant le fait que la liste `table_triee` est triée par prénom, vérifier s'il y a ou non un prénom apparaissant deux fois (il existe des prénoms utilisés à la fois pour les hommes et les femmes).

   On pourra définir une fonction qui renvoie `True` s'il y a un double, `False` sinon, ou bien seulement se contenter d'afficher les éventuels prénoms mixtes.

4. (a) Définir une nouvelle variable `table_triee_nb` contenant le contenu de `table1` trié selon la colonne du champ `'Nombre'`, par ordre décroissant.

   (b) Retrouver alors la liste des prénoms donnés au moins 30 fois, en utilisant cette fois le fait que la liste `table_triee_nb` soit triée par ordre décroissant de nombres.

   On doit ici voir en principe les prénoms masculins intercalés avec les prénoms féminins.

5. Nous allons maintenant fusionner les données du fichier `Prenoms1.csv` avec celles du fichier `Prenoms2.csv`.

   (a) Ouvrir et transformer de même manière que pour le premier les données du deuxième fichier en les mettant dans une variable `table2` (suivre la même méthode en transformant en entiers les données numériques).

   On admet (mais on peut le vérifier) que les données ont les mêmes champs, dans le même ordre, avec les mêmes domaines de valeurs (c'est à dire les mêmes types de données).

   (b) Se contenter de concaténer les deux tables comme elles sont pour l'instant n'aurait pas de sens : on va préférer rajouter deux champs, `'Ville'` et `'Année'`. On a indiqué au

tout début que le premier fichier concerne Strasbourg en 2008 et que le second concerne Rennes en 2007.
Créer alors une variable `table` contenant l'ensemble des données des deux tables auxquelles sont ajoutées, pour chaque ligne, les deux données des champs supplémentaires.
Par exemple :

```
>>> table[0]
('FEMME', 'Emma', 52, 'Strasbourg', 2008)
>>> table[410]
('HOMME', 'Gaël', 6, 'Rennes', 2007)
```

(c) Modifier également la variable `champs` pour qu'elle contienne les champs initiaux suivis des deux nouveaux champs `'Ville'` et `'Année'`.

(d) On va maintenant exporter le tout dans un fichier csv nommé `Prenoms.csv`.
Pour ce faire, copier et exécuter le code suivant :

```
data=[champs]+table
with open('Prenoms.csv', 'w') as csvfile:
    ecriture = csv.writer(csvfile)
    for ligne in data:
        ecriture.writerow(ligne)
```

On pourra vérifier, à l'aide d'un tableur ou d'un simple éditeur de texte, le contenu du fichier.
Le contenu devrait ressembler à ceci, des 10 premières lignes :

```
Sexe,Prénom,Nombre,Ville,Année
FEMME,Emma,52,Strasbourg,2008
FEMME,Clara,42,Strasbourg,2008
FEMME,Chloé,41,Strasbourg,2008
FEMME,Sarah,40,Strasbourg,2008
FEMME,Manon,36,Strasbourg,2008
FEMME,Léa,35,Strasbourg,2008
FEMME,Louise,33,Strasbourg,2008
FEMME,Lucie,31,Strasbourg,2008
FEMME,Camille,30,Strasbourg,2008
```

... jusqu'aux 10 dernières lignes :

```
HOMME,Bastien,6,Rennes,2007
HOMME,Benjamin,6,Rennes,2007
HOMME,Camille,6,Rennes,2007
HOMME,Edgar,6,Rennes,2007
HOMME,Elouann,6,Rennes,2007
HOMME,Gaël,6,Rennes,2007
HOMME,Mathias,6,Rennes,2007
HOMME,Milan,6,Rennes,2007
HOMME,Oscar,6,Rennes,2007
HOMME,Timothée,6,Rennes,2007
HOMME,Youenn,6,Rennes,2007
```

6. Question subsidiaire : la variable `table` contenant toutes nos données, écrire du code Python permettant d'obtenir le nombre total de `"Léane"`.

Remarque La fusion de tables n'est en fait pas quelque chose de si simple. Il y a différentes manières possibles de fusionner des tables.

L'une d'elles est de **concaténer** simplement, comme on l'a fait, lorsque les champs sont les mêmes, dans le même ordre et ont les mêmes domaines de valeur.
L'autre est de faire une **jointure**, dans le cas où les tables se complètent l'une l'autre, avec des champs communs mais surtout des champs différents. Par exemple on considère une table dont les champs sont `'Pays'` et `'Capitale'`, et un autre table dont les champs sont `'Pays'` et `'Langue'`. Alors on peut fusionner les tables avec une jointure sur le `'Pays'`. Dans le cas où un Pays n'existe que dans une des tables, on peut choisir de ne pas le mettre dans la jointure, ou de mettre une valeur None pour le champ manquant. Ces choses-là seront vues plus en détail en terminale.