TP 17
Traitement de données en tables

C'est bientôt la Saint-Valentin ! On en vient à parler couple, amour, naissance, choix du prénom pour le bébé… Quoi de plus romantique que d'écrire un programme Python pour manipuler la table de données de tous les prénoms des nouveaux-nés donnés en France et déclarés à l'état civil entre 2000 et 2009 ?

I Introduction

Le fichier `materiel.zip` contient un fichier `prenoms.csv` avec toute l'information dont nous avons besoin. Dans sa forme actuelle il contient 110 605 lignes, qu'on peut ouvrir avec un éditeur de texte même si cela est difficile pour travailler. Les premières lignes du fichier sont les suivantes :

```
"sexe","prenom","annee","nombre"
1,AARON,2000,118
1,ABBAS,2000,7
1,ABD,2000,6
1,ABD-ALLAH,2000,6
1,ABDALLAH,2000,68
1,ABDEL,2000,65
...
2,EMMA,2004,6634
2,EMMA-JANE,2004,3
2,EMMA-LISA,2004,4
2,EMMA-LOU,2004,10
2,EMMA-LOUISE,2004,4
2,EMMA-ROSE,2004,5
2,EMMANUELA,2004,5
2,EMMANUELLA,2004,21
2,EMMANUELLE,2004,219
...
```

Cela représente un tableau à quatre colonnes, comme leur nom l'indique. La colonne `"sexe"` vaut en fait 1 pour les garçons et 2 pour les filles (le fichier ne connait pas d'autre genre et ce n'est pas la question), la colonne `"prenom"` est en majuscule, la colonne `"annee"` est l'année de naissance et la colonne `"nombre"` le nombre de naissances avec ce prénom cette année.
Le fichier est ordonné avec d'abord les naissances de garçons de l'année 2000, puis les filles de 2000, puis les garçons de 2001, etc. Dans chacune de ces catégories, les prénoms sont classés par ordre alphabétique. Mais tout cela aura peu d'importance en pratique car nous traiterons le fichier à travers des boucles en Python ; il faut seulement se préoccuper du fait qu'un même prénom va revenir plusieurs fois, chaque année, éventuellement pour plusieurs sexes.
Il s'agit d'un **fichier CSV**, qui permet de représenter un tableau ou une base de données à la structure très simple, facile à traiter par l'ordinateur et dans une certaine mesure lisible par un humain : des colonnes décrites dans une en-tête, et dans chaque ligne les données correspondantes sont séparées par des virgules. Le mot CSV lui-même signifie « Comma-Separated Values » soit littéralement… « valeurs séparées par des virgules ». On ne s'est pas pris la tête pour trouver un nom ! On appellera chaque ligne une **entrée** de la **table** — ne pas confondre un prénom avec l'entrée toute entière.
Le code Python de démarrage ne fait qu'ouvrir ce fichier en utilisant la bibliothèque `csv` et le charge dans une grosse liste nommée `liste_prenoms` :

```
>>> len(liste_prenoms)
110604
```

La longueur est exactement un de moins que le nombre de lignes du fichier, à cause de l'en-tête.
Affichons un extrait en vrac de cette liste : par exemple, tout entre les indices entre 30 000 et 40 000 mais en sautant avec des pas de 1000 ce qui devrait afficher 10 prénoms :

```
>>> liste_prenoms[30000:40000:1000]
[{'sexe': '1', 'prenom': 'DAVIDSON', 'annee': '2003', 'nombre': '4'},
 {'sexe': '1', 'prenom': 'IRWIN', 'annee': '2003', 'nombre': '8'},
 {'sexe': '1', 'prenom': 'MARVIN', 'annee': '2003', 'nombre': '334'},
 {'sexe': '1', 'prenom': 'SAMET', 'annee': '2003', 'nombre': '37'},
 {'sexe': '2', 'prenom': 'AIMY', 'annee': '2003', 'nombre': '33'},
 {'sexe': '2', 'prenom': 'CYNTHIA', 'annee': '2003', 'nombre': '245'},
 {'sexe': '2', 'prenom': 'IRENA', 'annee': '2003', 'nombre': '13'},
 {'sexe': '2', 'prenom': 'LYANE', 'annee': '2003', 'nombre': '9'},
 {'sexe': '2', 'prenom': 'NOALIE', 'annee': '2003', 'nombre': '3'},
 {'sexe': '2', 'prenom': 'SWANA', 'annee': '2003', 'nombre': '3'}]
```

Comme chaque prénom a droit à sa propre entrée, une grande partie de cette liste est composée de prénoms rares, et toutes les variantes orthographiques ont aussi leur propre entrée ; les prénoms courant apparaissent une seule fois, mais avec une grande valeur pour la colonne `nombre`.
Chaque entrée de cette liste est nommée en Python un **dictionnaire** — nous y reviendrons dans un TP à part entière. Écrivons par exemple

```
>>> x = liste_prenoms[46090]
>>> print(x)
{'sexe': '2', 'prenom': 'EMMA', 'annee': '2004', 'nombre': '6634'}
```

alors on accède au prénom correspondant par `x["prenom"]`, à l'année par `x["annee"]` et de même pour les deux autres colonnes :

```
>>> x["sexe"]
'2'
>>> x["prenom"]
'EMMA'
>>> x["annee"]
'2004'
>>> x["nombre"]
'6634'
```

Cela se passe comme si chaque entrée était une petite liste, dont les indices ne sont pas des nombres mais sont les noms des colonnes avec lesquelles nous travaillons.
En fait on traitera tout le sujet en itérant directement sur la liste `for x in liste_prenoms` car il ne sera pas nécessaire de connaitre l'indice du prénom dans la liste totale. Cela simplifie aussi les notations.
Quelques dernières remarques :

1. Respectez les majuscules et les accents, dans les prénoms comme dans les intitulés de nos colonnes, cela a son importance. De plus, chaque orthographe correspond à une entrée différente. Les noms des colonnes sont en minuscule et sans accents. Les prénoms sont en majuscule et avec accent.
2. Toutes les données présentes sont des chaînes de caractères, de type `str`, y compris quand cela représente des nombres entiers. Pour accéder au nombre correspondant au prénom `x` il faut donc écrire `int(x["nombre"])`. Quant aux années, tant qu'on ne fait pas de calculs dessus, on peut les garder de type `str`, mais alors quand on donne une année il faut bien la mettre entre guillemets. Ainsi il faudra écrire des choses telles que `if x["annee"] == "2005"`. Le sexe lui est soit `"1"` soit `"2"`.

II Explorer et compter

Au tout début nous explorons le fichier et ce qu'il contient.

Exercice 1. Écrire une fonction `cherche(prenom)` qui prend en argument un prénom, parcourt toute la liste, et si le prénom est trouvé, affiche toute l'entrée correspondante.

Testez-la sur votre prénom, bien entendu, et observez un peu la liste.

Exercice 2. Écrire une fonction `nombre(prenom, annee, sexe)` qui prend en argument un prénom, une année et un sexe, et qui si elle trouve le prénom dans la liste, renvoie le nombre de fois où il a été donné.

Exercice 3. Écrire une fonction `nombre_prenoms(annee)` qui compte le nombre total de prénoms donnés pour l'année.

Maintenant il faut compter en utilisant la colonne `nombre`. Attention à bien convertir en `int` les valeurs lues !

Exercice 4. Écrire une fonction `compte(prenom)` qui prend en argument un prénom et renvoie le nombre de fois où il a été donné, sur toutes les années et éventuellement sur les deux sexes.

Exercice 5. Écrire une fonction `total_naissances(annee)` qui prend en argument une année, et qui renvoie le nombre total d'enfants nés cette année.

III Représenter graphiquement

La fonction `plt.bar(X, Y)` de la bibliothèque `matplotlib.pyplot` permet de tracer un diagramme en barres qui sera adapté à afficher le nombre de fois qu'un prénom a été donné chaque année. La liste `X` sera celle des années et la liste `Y` sera celle des nombres. La documentation ou les aide-mémoires de `matplotlib` permettent d'améliorer un peu l'aspect du graphique : titre avec `plt.title()`, noms des axes avec `plt.xlabel()` et `plt.ylabel()`, couleurs des barres etc.
Il faut donc créer une liste indiquant combien de fois le prénom a été donné, chaque année.

Exercice 6. Écrire une fonction `barre(prenom, sexe)` qui affiche un diagramme en barres du nombre de fois qu'un prénom a été donné en fonction de l'année.

IV Filtrer

Exercice 7.
1. Écrire une fonction `maximum(annee, sexe)` qui renvoie le prénom le plus donné, sur l'année et le sexe passés en argument.
2. Bonus : écrire une fonction `maximum2(annee, sexe)` qui renvoie le couple formé des deux prénoms les plus donnés.

On aimerait avoir le TOP 10 des prénoms les plus donnés, par année et par sexe. Mais ce n'est pas très simple à programmer, cela ressemble aux algorithmes de tri. On propose l'approche suivante, basée sur l'idée du tri par sélection (celui où on *sélectionne* à chaque fois le maximum suivant) :

1. On écrit une fonction `prochain_maximum(L, annee, sexe)` qui prend en argument une liste `L` d'entrées de la table (des entrées complètes, avec le prénom et le nombre) et qui renvoie l'entrée du prénom le plus donné, dans l'année et le sexe, *parmi ceux qui ne sont pas déjà dans L*.
2. On en déduit une fonction `top(n, annee, sexe)` qui renvoie la liste des n prénoms les plus donnés, en itérant la fonction précédente et en lui passant à chaque fois la liste des prénoms construite. Éventuellement c'est naturellement une fonction récursive !
3. Cela pourrait nécessiter d'écrire à part une fonction `est_deja_dedans(L, x)` qui prend en argument une liste d'entrées `L` et une entrée `x` et renvoie `True` si `x` est dans `L` et `False` sinon. On peut assi tricher en utilisant directement `if x in L` et la négation `if x not in L`.

Exercice 8. Écrire les fonctions précédentes, et donner le TOP 10.

V Choisir le prénom

On s'intéresse enfin au choix du prénom au hasard. Pour cela il ne s'agit pas simplement de choisir une entrée de la liste au hasard : on voudrait choisir un prénom de façon proportionnelle à sa fréquence d'apparition. Dans un premier temps on cherche parmi toutes les années et sexes confondus. Cela nécessite donc d'abord de compter le nombre de naissances totales, appelons le N. Ensuite on tire au hasard un nombre entre 1 et N. L'idée à traduire dans un algorithme, qui parcourt toute la liste est la suivante...

Imaginons qu'un premier prénom soit donné 3 fois, un deuxième est donné 8 fois, et le dernier est donné 2 fois. Cela fait 15 naissances, on prend un nombre au hasard entre 1 et 15. Alors on veut choisir le premier prénom si le nombre tiré est $1, 2, 3$, le deuxième si le nombre tiré est entre 4 et 12, et le troisième si le nombre tiré est $13, 14, 15$. Autrement dit dans une boucle on a besoin de compter les cumuls de naissances, et comparer le cumul avec le nombre tiré au hasard : dès que le cumul dépasse notre nombre choisi, on s'arrête et on considère qu'on choisit ce prénom !

Exercice 9.
1. Écrire une fonction `choix_prenom()` qui choisit un prénom au hasard par cette méthode, et renvoie le prénom.
2. Écrire une fonction `choix_groupe(n)` qui renvoie une liste de n prénoms choisis au hasard selon cette méthode, et observez par exemple en générant toute une classe de 30 élèves !

Il n'est pas difficile de modifier la méthode précédente pour générer uniquement un prénom rare, où rare signifie par exemple « donné moins de 10 fois » (le seuil est au choix).

Exercice 10. Écrire la fonction `choix_prenom_rare()`, et créer une liste de 30 prénoms rares.