Traitement de données en tables.

1 Exemple : base des établissements scolaires français.

Télécharger la base des établissements scolaires français au format csv à l'adresse :
`https://www.data.gouv.fr/fr/datasets/adresse-et-geolocalisation-des-etablissements-denseigner`
Ouvrir le fichier `etablissements.py`, qui ouvre ce fichier et l'importe dans une table.

Enregistrement des données : dans le fichier CSV, la première ligne est l'entête, donnant le nom de chaque champ.

On a choisi d'enregistrer les données sous forme d'une liste d'établissements ; chaque établissement est codé par un dictionnaire, dont les clés sont les noms des champs donnés par l'entête :

```
>>> Etablissements[0]
{'Latitude': '48.97544957717941', 'Lieu dit': '', 'Code région': '11',
'Région': 'Ile-de-France', 'Code nature': '151' ...
```

Par facilité, les valeurs des champs sont les chaînes de caractère lues dans le fichier CSV ; il pourrait être plus efficace de les convertir en entier, flotant, ... ce qui nécessiterait de faire un enregistrement spécifique pour chaque champ.

Filtrage des données : il est alors possible de faire une sous-liste ne contenant que les établissements vérifiant certains critères.

Tri de la table : on peut imaginer de nombreux critères pour trier la liste : selon le nom de l'établissement, par département, par type d'établissement... L'option `key` des fonctions `sort` et `sorted` permettent de choisir le critère de tri.

Enregistrement dans un fichier : après manipulation, on peut réenregistrer les données modifiées dans un fichier. On peut garder la syntaxe du fichier d'origine, en écrivant une ligne d'entête puis une ligne par établissement, en ne donnant que la valeur des champs séparés par un ';'.

Remarquer que dans l'exemple du fichier `etablissements.py`, l'ordre des champs du fichier d'origine n'est pas conservé : le parcours d'un dictionnaire ne donne pas nécessairement les champs dans leur ordre d'introduction.

Exemple d'utilisation des données : il est alors possible, par exemple, d'afficher les points des coordonnées des différents établissements. On obtient alors la carte de France des établissements.

Fusion de tables : il est possible de croiser les données entre deux tables à condition d'avoir un identifiant commun dans les deux tables.

Activité : trouver d'autres données sur les lycées de l'académie de Versailles, telles que ceux offrant l'option NSI ; fusionner les deux tables pour obtenir une liste des lycées de l'académie avec ces nouvelles données en plus. On pourra par exemple utiliser des données en provenance de :
`https://www.education.gouv.fr/bcp/mainFrame.jsp?p=1`

2 TP : base des prénoms de l'INSEE

L'objectif de ce TP est de réaliser un outil d'aide au choix d'un prénom. Cet outil s'appuiera sur un fichier de statistiques nationales pour afficher la fréquence d'un prénom donné, donner les 100 premiers prénoms donnés pour une année fixée, dessiner la courbe de fréquence d'un prénom dans le temps...

Le projet se fera **en binôme**.

3 Importer les données

L'INSEE tient à jour une base de données des prénoms donnés en France année par année, depuis 1900. On pourra télécharger le fichier *nat2017.txt* au format *txt* sur le site

`https://www.insee.fr/fr/statistiques/2540004#consulter`.

Ce fichier contient des lignes de la forme

```
sexe preusuel annais nombre
```

Où *sexe* vaut 1 pour les garçons, 2 pour les filles, *preusuel* désigne le prénom, *annais* l'année de naissance, *nombre* le nombre de naissances. Les quatre champs sont séparés par des tabulations (`\t`).

Pour certaines lignes, l'année n'est pas renseignée et est remplacée par `XXXX` ; **on ignorera ces lignes**.

Question 1 : écrire une fonction prenant en entrée le nom du fichier, et retournant cette base de données sous forme de deux dictionnaires de dictionnaires, un pour les prénoms féminins, l'autre pour les prénoms masculins :

```
Prénoms masculins :
{'ADAM' : {1900: 20, 1901: 11, ...}, ... }
Prénoms féminins :
{'ADÈLE' : {1900: 661, 1901: 625, ...}, ... }
```

4 Différentes fonctions

Question 2 : écrire une fonction `populaire(dic,debut,fin)` prenant en entrée un dictionnaire (celui des prénoms masculins ou féminins), une date de début et une date de fin et retournant une liste de couples `(prenom,nombre)` donnant le nombre de naissances pour chaque prénom sur la période, triée du plus courant au moins courant.

Pour le tri, on peut consulter : `https://wiki.python.org/moin/HowTo/Sorting`.

Question 3 : écrire une fonction `courbe(dic,prenom,debut,fin)` prenant en entrée un dictionnaire des prénoms (masculin ou féminin), un prénom, une date de début et une date de fin et affichant la courbe des naissances en fonction de l'année.

On pourra utiliser la fonction `plot` du module `matplotlib`, qui en prenant en entrée deux listes de valeurs x et y, affiche les y en fonction des x :

```
import matplotlib.pyplot as pl

pl.plot([1,2,3,4],[2,6,12,19])
pl.show()
```

Question 4 : écrire une fonction `stats(dic,prenom,debut,fin)` retournant l'année où il y a eu le plus de naissances, et le nombre moyen de naissances pour ce prénom entre les années début et fin.

5 Quelques requêtes

A l'aide des fonctions précédentes, ou en écrivant d'autres questions, trouver :

1. Le nombre total de personnes portant votre prénom.
2. Le top 10 des prénoms masculins les plus donnés de 1900 à 1910, de votre sexe lors de votre année de naissance.
3. Le nombre de prénoms masculins différents en 1900, en 2017.

4. Le nombre de prénoms mixtes, à la fois masculins et féminins.
   Cette dernière question revient à fusionner les tables des prénoms masculins et féminins. On peut le faire naïvement, en parcourant les prénoms masculins et en les cherchant dans le dictionnaire des prénoms féminins ; ou plus efficacement par la méthode de fusion de deux tableaux triés, en parcourant les deux tables dans l'ordre alphabétique des prénoms.

6 Quelques éléments de syntaxe

6.1 Lire dans un fichier

```
f= open("monFichier.txt",'r')
for ligne in f :
    bla bla bla
f.close()
```

Pour ne pas risquer d'oublier de fermer f, on peut se placer sans un bloc dans lequel f est ouvert :

```
with open("monFichier.txt",'r') as f :
    for ligne in f :
        bla bla bla
```

Dans la syntaxe précédente, chaque ligne finit par le caractère de fin de ligne. On peut l'éviter en écrivant plutôt :

```
with open("monFichier.txt",'r') as f :
    lignes=f.read().splitlines()
    for ligne in lignes :
        bla bla bla
```

6.2 Listes, Dictionnaires, ...

`https://docs.python.org/fr/3/tutorial/datastructures.html`