RND. Tables

I/ Introduction

Les données informatiques sont de plus en plus nombreuses, elles doivent être stockées, gérées, partagées et modifiées à distance depuis l'avènement des réseaux. La recherche sur des systèmes offrant ces possibilités datent des années 1960 en particulier avec les missions Apollo. Des sociétés comme IBM et Rockwell travaillaient sur ce projet. Charles Bachman, avec la société General Electric, produit alors un nouveau système de gestion des informations qui ne sont plus stockées de manière hiérarchique comme avec un explorateur de fichiers.

En 1970, une première théorie sur les modèles relationnels permettant le partage des données est publiée par Edgar Codd.

A noter : les bases de données seront étudiées en Terminale.

En savoir plus sur les bases de données relationnelles : https://pixees.fr/lhistoire-des-base-de-donnees-ou-presque/

II/ Importation d'une table

Une table peut être représentée sous la forme d'un tableau à double entrée :

- La première ligne représente les **champs** (ou clés ou attributs) et agit comme un « titre » pour chaque colonne.
- Les lignes suivantes sont des **enregistrements** qui sont des structures de données auxquelles on peut accéder grâce à un nom.

Important : **tous les champs d'un enregistrement doivent être complétés par une valeur, même pas défaut.**

Exemple :

Table des 8 premiers pays du monde par ordre alphabétique

Nom	Continent	Superficie	Population	Capitale
Afghanistan	Asie	652864.0	34124800	Kaboul
Angola	Afrique	1246700.0	30355900	Luanda
Albanie	Europe	28748.0	3048000	Tirana
Andorre	Europe	468.0	85600	dorre-la-Vieille
Argentine	nérique du s	2791800.0	44293300	Buenos Aires
Arménie	Asie	29800.0	3045200	Erevan
Australie	Océanie	7692060.0	23470100	Canberra
Autriche	Europe	83871.0	8754400	Vienne

Champs : « Nom », « Continent », « Superficie », « Population », « Capitale »
Exemple d'enregistrement : « Angola », « Afrique », « 1246700 », « 30355900 », « Luanda ».

Un tableur (comme Excel) permet d'enregistrer ces données au format **CSV**. C'est celui qui sera étudié ici.
Dans ce format, chaque donnée est **séparée** par un **point-virgule** (***norme française***) ou **virgule** (***norme anglo-saxonne***) et chaque enregistrement correspond à une **ligne**.

A noter : il faudra faire attention à l'origine de certains fichiers .csv à cause du séparateur.

Le langage Python permet aisément de charger ce type de fichiers et de gérer les données.

Voici le code minimal pour importer une table au format .csv. (à comprendre)

```
f = open("Pays.csv","r")   # Ouverture du fichier "monde.csv" en mode lecture ("r" pour read)
champs = f.readline()       # Lecture de la première ligne (les champs)
lignes = f.readlines()      # Lecture des autres lignes jusqu'à la fin du fichier
table = []

for ligne in lignes :
    liste = ligne.rstrip().split(";")  # Le point virgule permet d'indiquer la position
                                       # de chaque donnée. liste une de type "list"
    table.append(liste)     # Ajout de la liste à la table

f.close()                   # Fermeture du fichier

print("Fichier brut :",lignes)   # Affichage du fichier "brut", "\n" représente un saut à la ligne
print("\n")
print("Table :",table)    # Affichage de la liste de listes
```

```
Fichier brut : ['Afghanistan;Asie;652864.0;34124800;Kaboul\n', 'Angola;Afrique;1246700.0;30355900;Lua
748.0;3048000;Tirana\n', 'Andorre;Europe;468.0;85600;Andorre-la-Vieille\n', 'Argentine;Amérique du su
s Aires\n', 'Arménie;Asie;29800.0;3045200;Erevan\n', 'Australie;Océanie;7692060.0;23470100;Canberra\n
0;8754400;Vienne\n']


Table : [['Afghanistan', 'Asie', '652864.0', '34124800', 'Kaboul'], ['Angola', 'Afrique', '1246700.0'
['Albanie', 'Europe', '28748.0', '3048000', 'Tirana'], ['Andorre', 'Europe', '468.0', '85600', 'Andor
ne', 'Amérique du sud', '2791800.0', '44293300', 'Buenos Aires'], ['Arménie', 'Asie', '29800.0', '304
lie', 'Océanie', '7692060.0', '23470100', 'Canberra'], ['Autriche', 'Europe', '83871.0', '8754400', '
```

Remarques :

- **Toutes les données** sont affichées sous forme de **chaînes de caractères**.
- Il **faudra convertir** certaines données en nombre (par exemple la population) si on souhaite faire des calculs ou des comparaisons avec.

Voici un autre code utilisant la bibliothèque csv du langage (le fichier top14.csv contient ici la liste des joueurs du top 14 de rugby).

```
import csv
%matplotlib inline

# Chargement des données du fichier top14.csv dans la variable joueurs (liste de dictionnaires)
f = open('top14.csv', "r", encoding = 'utf-8')
donnees = csv.DictReader(f)
joueurs = []
for ligne in donnees:
    joueurs.append(dict(ligne))

f.close()
```

Créer **une liste de dictionnaires** est souvent avantageux pour l'exploitation des données. Plusieurs activités y sont consacrées.

III/ Recherche dans une table

Le but d'une recherche est bien sûr de trouver les enregistrements (ou parties d'enregistrements) qui répondent à une **requête**. Elle s'effectue à partir de critères de comparaisons numériques et/ou de relations booléennes.

Exemple : on souhaite trouver les pays européens ayant plus de 5 millions d'habitants dans la table précédente.

```
f = open("Pays.csv","r")   # Ouverture du fichier "monde.csv" en mode lecture ("r" pour read).
champs = f.readline().rstrip().split(";")       # Lecture de la première ligne (les champs)
                                                # et conversion en liste.
# On récupère les indices des champs qui nous intéressent
indices = [champs.index('Nom'),champs.index('Continent'),champs.index('Population')]

lignes = f.readlines()      # Lecture des autres lignes jusqu'à la fin du fichier.
table = []

for ligne in lignes :
    liste = ligne.rstrip().split(";")  # Le point virgule permet d'indiquer la position.
                                       # de chaque donnée. liste une de type "list".
    liste[indices[2]] = float(liste[indices[2]])   # Conversion en flottant de la population.
    table.append(liste)     # Ajout de la liste à la table.

f.close()                   # Fermeture du fichier.
```

```
requete = []   # Liste vide qui contiendra les pays européens de plus de 5 millions d'habitants.
for ligne in table :   # On cherche les pays européens de plus de 5 millions d'habitants.
    if ligne[indices[1]] == 'Europe' and ligne[indices[2]] > 5000000 :
        requete.append(ligne[indices[0]])

print(requete)
```

```
['Autriche']
```

Remarque : la liste *indices* permet de connaître l'indice du champ cherché. On peut toutefois indiquer directement l'indice si on le connaît. C'est pour cela qu'une liste de dictionnaires est avantageuse puisqu'il suffit de faire une recherche par mot clé.

IV/ Tri d'une table

Il peut être intéressant de trier une liste en fonction des valeurs d'un champ si l'on cherche par exemple les 3 pays les plus peuplés dans l'exemple précédent. On utilise pour cela la fonction ***sorted()*** pour y parvenir.

Vidéo à voir sur l'utilisation de ***lambda*** et ***map*** en langage Python ici :
https://www.youtube.com/watch?v=LNGVowVIJ1g
Auteur : getCodingKnowledge, durée 6 min 20 sec

Voici un exemple de programme en langage Python triant les pays selon leur continent :

```
# Tri dans l'ordre croissant selon les continents
tableTriée = sorted(table, key = lambda table : table[indices[1]])

print(tableTriée)
```

```
[['Angola', 'Afrique', '1246700.0', 30355900.0, 'Luanda'], ['Argentine', 'Amérique du sud'
res'], ['Afghanistan', 'Asie', '652864.0', 34124800.0, 'Kaboul'], ['Arménie', 'Asie', '298
e', 'Europe', '28748.0', 3048000.0, 'Tirana'], ['Andorre', 'Europe', '468.0', 85600.0, 'Ar
ope', '83871.0', 8754400.0, 'Vienne'], ['Australie', 'Océanie', '7692060.0', 23470100.0, '
```

```
# Tri dans l'ordre décroissant selon les continents
tableTriée = sorted(table, key = lambda table : table[indices[1]], reverse = True)

print(tableTriée)
```

```
[['Australie', 'Océanie', '7692060.0', 23470100.0, 'Canberra'], ['Albanie', 'Europe', '287
e', 'Europe', '468.0', 85600.0, 'Andorre-la-Vieille'], ['Autriche', 'Europe', '83871.0', 8
'Asie', '652864.0', 34124800.0, 'Kaboul'], ['Arménie', 'Asie', '29800.0', 3045200.0, 'Erev
d', '2791800.0', 44293300.0, 'Buenos Aires'], ['Angola', 'Afrique', '1246700.0', 30355900.
```

A savoir : **le tri sur des chaînes de caractères se fait caractère par caractère de gauche à droite en suivant l'alphabet. Cela s'appelle le tri (par ordre) lexicographique.**

Exemples : bouleau > aulne (« b » est après « a ») ; travail > télétravail (« t » = « t » mais « r » est après « é »).
On fera aussi attention aux lettres majuscules / minuscules lors des comparaisons.

Attention : **De même « 8 » est plus grand que « 112 « puisque 8 > 1**. Il faut donc faire attention lorsque l'on souhaite comparer des valeurs numériques : il faut les convertir en nombres entiers (fonction int(*valeur*)) ou flottants (float(*valeur*)).

V/ Recherche de doublons dans une table

Certaines données ne doivent pas être dupliquées dans ce type de fichier. Par exemple, dans le fichier précédent, le champ 'pays' doit être composé de valeurs toutes différentes. A l'inverse, il n'est pas problématique de trouver plusieurs fois le même continent ni la même population /superficie.
Cette notion est importante à comprendre et est le fondement des bases de données relationnelles : un tel champ s'appelle **clé primaire** et permet d'identifier à coup sûr un enregistrement. Il faut donc que la table ne contienne qu'un enregistrement par clé primaire.

On supposera que la table est triée. Voici un exemple qui permet de supprimer les doublons d'une table (triée), on a choisi le « nom du pays » comme champ.

```
f = open("Pays.csv","r")   # Ouverture du fichier "monde.csv" en mode lecture ("r" pour read).
champs = f.readline().rstrip().split(";")       # Lecture de la première ligne (les champs)
                                                # et conversion en liste.
# On récupère les indices des champs qui nous intéressent
indices = [champs.index('Nom'),champs.index('Continent'),champs.index('Population')]

lignes = f.readlines()      # Lecture des autres lignes jusqu'à la fin du fichier.
table = []

for ligne in lignes :
    liste = ligne.rstrip().split(";")  # Le point virgule permet d'indiquer la position.
                                       # de chaque donnée. liste une de type "list".
    liste[indices[2]] = float(liste[indices[2]])   # Conversion en flottant de la surface.
    liste[3] = int(liste[3])     # Conversion en entier de la population.
                                 # Remarque : on sait ici que la population est en
                                 # 4ème position : pas besoin de passer par "indice".
    table.append(liste)     # Ajout de la liste à la table.

f.close()                   # Fermeture du fichier. A ne pas oublier.

# Ajout de deux doublons
table.append(['Arménie','Asie',29800,3045200,'Erevan'])
table.append(['Andorre','Europe',468,85600,'Andorre-la-Vieille'])
```

```
# Tri dans l'ordre croissant selon les pays
tableTriée = sorted(table, key = lambda table : table[indices[0]])

print(tableTriée)   # On retrouve bien les doublons
```

```
[['Afghanistan', 'Asie', '652864.0', 34124800, 'Kaboul'], ['Albanie', 'Europe', '28748.0', 3048000, 'Tirana'],
ope', '468.0', 85600, 'Andorre-la-Vieille'], ['Andorre', 'Europe', 468, 85600, 'Andorre-la-Vieille'], ['Angola'
46700.0', 30355900, 'Luanda'], ['Argentine', 'Amérique du sud', '2791800.0', 44293300, 'Buenos Aires'], ['Armér
800.0', 3045200, 'Erevan'], ['Arménie', 'Asie', 29800, 3045200, 'Erevan'], ['Australie', 'Océanie', '7692060.0'
berra'], ['Autriche', 'Europe', '83871.0', 8754400, 'Vienne']]
```

```
def cleUnique(table,index) :
    tableSansDoublon = [table[0]]   # Le premier enregistrement est bien sûr unique.
    for ligne in table :            # Parcours de la table
        if ligne[index] != tableSansDoublon[-1][index] : # Comparaison ligne lue et la dernière
                                                         # ligne enregistrée
            tableSansDoublon.append(ligne)
    return tableSansDoublon

print(cleUnique(tableTriée,0))   # Le nom du pays est en première position
```

```
[['Afghanistan', 'Asie', '652864.0', 34124800, 'Kaboul'], ['Albanie', 'Europe', '28748.0', 3048000, 'Tirana'],
ope', '468.0', 85600, 'Andorre-la-Vieille'], ['Angola', 'Afrique', '1246700.0', 30355900, 'Luanda'], ['Argentir
sud', '2791800.0', 44293300, 'Buenos Aires'], ['Arménie', 'Asie', '29800.0', 3045200, 'Erevan'], ['Australie',
060.0', 23470100, 'Canberra'], ['Autriche', 'Europe', '83871.0', 8754400, 'Vienne']]
```