Introduction aux bases de données

I Manipulation du tableur

On peut trier une plage de données (*Données/Trier…*) :

- en décidant que la première ligne comporte ou non des étiquettes de colonne (onglet *options*) ;
- suivant une ou plusieurs clés de tri :
  - les lignes seront triées suivant la première clé (en comparant les valeurs de la colonne correspondante),
  - puis dans les groupes de lignes ayant les mêmes valeurs dans cette colonne, en comparant les valeurs de la prochaine colonne.

II Tri de dictionnaires

1) Tri des listes avec Python

Nous n'allons pas coder d'algorithme de tri dans ce chapitre (même si c'est au programme de première), mais utiliser la fonction `sorted` ou la méthode `.sort()` (qui utilisent bien sûr des algorithmes de tri !).

```
>>> L = [4, 3, 5, 2]
>>> L
[4, 3, 5, 2]
>>> sorted(L)
[2, 3, 4, 5]
```

```
>>> L
[4, 3, 5, 2] # L intacte
>>> L.sort()
>>> L
[2, 3, 4, 5] # modifiée
```

2) Tri des listes de dictionnaires avec Python

Python ne « sait » pas trier une liste de dictionnaires.

```
>>> affiche(sorted(liste_de_dicos))
[...]
TypeError: '<' not supported between instances of 'dict'
```

Comme pour le tableur, il faut indiquer quelle clé de tri utiliser.
Remarque : on utilise encore le mot « clé » mais cela n'a a priori rien à voir avec les clés des `dict`s. C'est comme un « score » calculé par dico.

```
>>> def somme(d):
      return d['A'] + d['B'] + d['C'] + d['D']

>>> affiche(sorted(liste_de_dicos, key=somme))
{'A': 3, 'B': 1, 'C': 2, 'D': 1}
{'A': 1, 'B': 2, 'C': 3, 'D': 2}
{'A': 2, 'B': 1, 'C': 4, 'D': 3}
{'A': 4, 'B': 2, 'C': 1, 'D': 4}
>>>
>>> # On peut prendre pour clé de tri
>>> # la valeur des dictionnaires stockée à la clé A
>>> # (attention au vocabulaire !
>>> def valeur_en_A(dictionnaire):
      return dictionnaire["A"]

>>> affiche(sorted(liste_de_dicos, key=valeur_en_A))
{'A': 1, 'B': 2, 'C': 3, 'D': 2}
{'A': 2, 'B': 1, 'C': 4, 'D': 3}
{'A': 3, 'B': 1, 'C': 2, 'D': 1}
{'A': 4, 'B': 2, 'C': 1, 'D': 4}
```

La fonction `sorted` n'accepte qu'un paramètre nommé `key`, comment faire pour trier suivant la valeur à la clé B, puis à la clé A ? (voir fichier)

III Les systèmes de gestion de bases de données

Cette année nous utiliserons SQLITE.

1) Utilisation avec un client graphique

DB Browser est une interface graphique pour SQLITE.
Il manipule des fichiers `.db` (comme Calc manipule les `.ods`).
On peut créer ou ouvrir un tel fichier et manipuler les données :

- en utilisant principalement les menus et les boutons,
- et/ou en tapant des requêtes SQL.

Récupérer le logiciel dans DosSup, à copier sur votre clef et testez-le.

a) Fichier pour stocker les données

Démarer DB Browser
Bouton [Nouvelle base de données], fichier à enregistrer dans :
`mon_dossier_NSI\BDD\contacts.db`

b) Structure

Bouton [Nouvelle table] : `personne`.
ATTENTION aux ESPACES et aux ACCENTS
Bouton [Ajouter un champs] pour les « colonnes » :

`Nom` (type TEXT), cocher NON NULL
`Prénom` (type TEXT), cocher NON NULL
`jour_naiss` (type INT)
...

On remarque que du code est généré au fur et à mesure.

Ne pas oublier le bouton [Enregistrer les modifications] de temps en temps.

c) Données : écriture et lecture

Dans l'onglet [Parcourir les données], appuyer sur le petit bouton [+].
On peut filtrer comme avec les tableurs, mais on peut faire bien mieux !

d) Premières requêtes

Ouvrir l'onglet [Exécuter le SQL], taper les requêtes une par une, puis [>].

```
SELECT * FROM personne
SELECT nom, annee_naiss FROM personne
SELECT nom FROM personne WHERE mois_naiss = 1
```

2) Utilisation depuis un programme Python

Récupérer le fichier Python sur DosSup.

```
import sqlite3  # si bug avec Idle, utiliser Spyder
```

Corriger les erreurs laissées dans le fichier en lisant les messages d'erreur.

IV Structurer une base de données

1) Problème de redondance

Si on veut indiquer le lycée dans lequel étudie chaque contact, une façon simple de faire serait d'ajouter une colonne `lycee` dans la table `personne`. Mais dans ce cas :

- Il peut y avoir une faute d'orthographe sur quelques lycées : les données ne sont pas cohérentes.
- Il peut y avoir la même faute d'orthographe sur tous les lycées et, même s'il y a des méthodes pour changer plusieurs enregistrements à la fois (requête de type UPDATE dans une boucle depuis Python ou avec un WHERE),

<u>On veut donc centraliser le nom des lycées dans une table séparée.</u>

2) Vue « données » et vue « structure »

Nous sommes habitués à la vue « données » (ressemble aux tableurs), mais les professionnels utilisent la vue « structure ».
clé soulignée -> clé primaire
clé avec un croisillon (#) -> clé étrangère

Comparons les deux versions :

Exemples de requête avec cette structure

SELECT nom FROM eleves, lycees ;

- Se lit « **DEPUIS** les tables **eleves, lycees**, j'affiche le **nom »**.
- Problème : le **nom** est un champ ambigu (il apparaît dans les deux tables).
- Solution : préfixer par le nom de la table.

SELECT eleves.nom, lycees.nom FROM eleves, lycees ;

- « **DEPUIS** les tables **eleves, lycees**, j'affiche le **nom** dans chaque table **»**.
- Problème : toutes les associations eleve-lycée sont produites dans la liste des résultats ! C'est ce qu'on appelle un produit cartésien.
- Solution : ajouter une contrainte pour ne faire apparaître que les résultats où **l'identifiant lycée** est le même.

SELECT eleves.nom, lycees.nom FROM eleves, lycees
WHERE eleves.lycee = lycees.id;

- Se lit « **DEPUIS** les tables **eleves, lycees**, j'affiche le **nom** dans chaque table des résultats qui ont le même **id lycée »**.

Exercice
Modélisez une base de données pour la situation suivante :
Lors d'une soirée, les élèves décident de laisser leurs portables dans une boîte pour ne pas les utiliser avec leurs camarades.
Des élèves de NSI proposent de stocker les noms, prénoms des élèves et la marque et le modèle des téléphones pour pouvoir les restituer à la fin.
Vous donnerez une vue de type « structure ».

Méthode
Commencez par souligner les concepts importants qui finiront par être représentés par des **tables**. Ensuite cherchez les données qui peuvent être identiques pour plusieurs individus ou objets, c'est ce que vous devrez « factoriser ».

Écrire les requêtes pour:
- Afficher Nom Prénom Modèle Marque (pour chaque élève)
- Afficher les modèles de la marque Samsung
- Afficher le nombre de modèles de chaque marque
- Afficher le nom des élèves ayant un Apple

Approfondissement : Comment structurer la base de données de façon à pouvoir attribuer plusieurs téléphones aux élèves ?