Utiliser une base de données en Python

Dans certains cas, il est utile d'utiliser une base de données dans une application programmée en Python (ou autre). Par exemple si la quantité de données à manipuler est trop grande pour les structures de données en mémoire comme les tableaux ou les dictionnaires. Ou encore si les données doivent être stockées de façon permanente et ne pas disparaître quand le programme se termine.

Voyons comment exploiter notre base de données SQLite LivresAuteurs.db et exécuter des requêtes SQL depuis un programme Python.

Ouvrir la base de données

Commençons par importer le module `sqlite3` dans un programme.

```
import sqlite3
```

La première chose à faire ensuite est d'ouvrir le fichier de la base de données (noter que si le fichier n'existe pas, une nouvelle base de données sera créée) :

```
bdd = sqlite3.connect("LivresAuteurs.db")
```

puis de créer un curseur :

```
curseur = bdd.cursor()
```

Ce curseur est un objet Python qui va nous permettre d'exécuter des requêtes et de récupérer les résultats de ces requêtes.

Exécuter des requêtes de sélection

Exécutons maintenant notre première requête avec la méthode `execute()` du curseur. La requête SQL est une chaîne de caractères passée en paramètre à `execute()`.

```
requete = "SELECT * FROM Pays;"
curseur.execute(requete)
```

La curseur permet ensuite de visualiser le résultat de cette requête par l'une ou l'autre de ces méthodes :

- `fetchone()` pour récupérer une ligne du résultat puis avancer le curseur d'un cran
- `fetchall()` pour récupérer d'un coup tous les lignes du résultat.

Pour mieux comprendre comment fonctionne le curseur, exécutons plusieurs fois l'instruction suivante dans la console :

```
>>> curseur.fetchone()
```

On constate que le curseur permet de se déplacer une ligne après l'autre dans le résultat de la requête.

Essayons maintenant `fetchall()` pour récupérer toutes les lignes du résultats dans un seul p-uplet.

```
>>> curseur.fetchall()
```

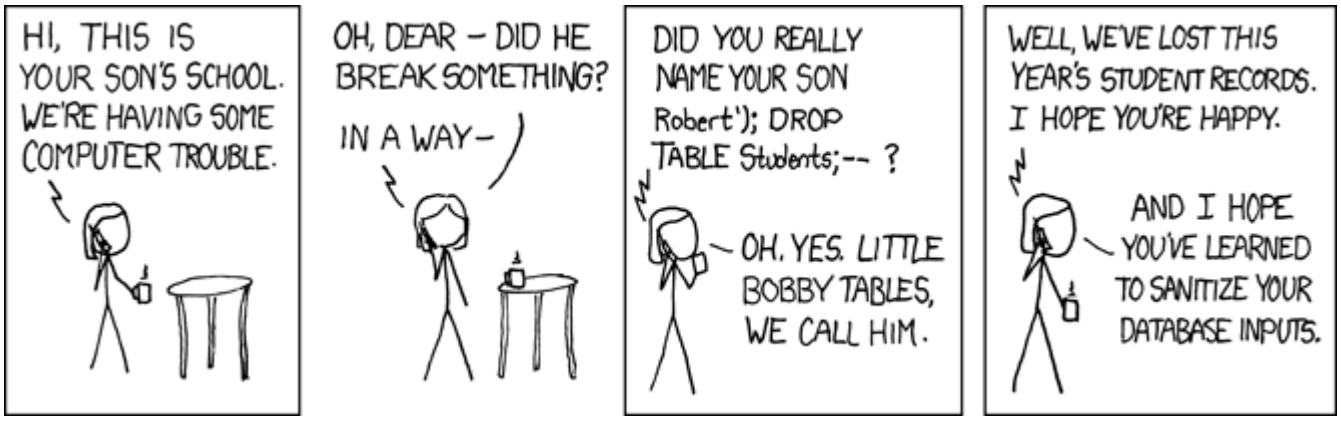
On constate qu'il manque les premières lignes. En effet, elles ont déjà été lues par les premiers `fetchone()` exécutées, le curseur s'est déplacé après. Et si on réexécute une nouvelle fois la méthode `fetchall()` du curseur, celle-ci ne renvoie plus rien, le premier `fetchall()` a positionné le curseur après la dernière ligne du résultat. Pour retrouver à nouveau toutes les lignes du résultat, il faut réexécuter la requête.

Construire des requêtes à partir de variables python

Construisons un programme qui affiche les informations d'un auteur dont on demande le nom. La requête est une chaîne de caractère, on est tenté de la construire par concaténation de chaînes :

```
nom = imput("entrez un nom d'auteur")
requete = "SELECT Prenom FROM Auteur WHERE Nom =" + nom
curseur.execute(requete)
print(curseur.fetchone())
```

⚠️ Attention, c'est une **très mauvaise approche** qui conduira à de nombreux problèmes, et en particulier des injections SQL ! Que se passerait-il si l'utilisateur entrait une requête SQL pour nom d'auteur ?



La bonne approche consiste à constuire la requête avec des `?` pour désigner des variables que l'on veut insérer dans la requête et à passer en second paramètre la liste des valeurs à substituer dans la requête :

```
nom = imput("entrez un nom d'auteur")
requete = "SELECT Prenom FROM Auteur WHERE Nom = ?"
curseur.execute(requete, [nom])
print(curseur.fetchone())
```