Manipulation d'une base de données en Python

SQLite est un moteur de base de données relationnelle accessible par le langage SQL. Le module python **`sqlite3`** permet d'utiliser une base de données relationnelle sous Python.

Dans ce chapitre, on utilisera le fichier « **`repertoire.db`** » (que vous trouverez sur le réseau) contenant un répertoire téléphonique. A l'aide de SQL OnLine IDE (https://sqliteonline.com/), ouvrir ce fichier et examiner sa structure (le schéma relationnel) ainsi que son contenu. Afin de pouvoir travailler correctement avec cette base de données, copier ce fichier dans votre répertoire « **`perso`** ». Si jamais ce fichier se retrouvait « abimé » (et ça risque d'arriver !), vous pourrez le récupérer à nouveau sur le réseau.

I./ Exemple d'utilisation du module sqlite3 :

Dans le même répertoire que celui où vous avez placé le fichier « **`repertoire.db`** », taper et enregistrer le script Python suivant :

```
# importation de la librairie sqlite3
import sqlite3

# crée une connexion à la base de données (si le fichier n'existe pas, il sera créé)
connexion = sqlite3.connect('repertoire.db')

# création d'un curseur
curseur = connexion.cursor()

# exécution d'une requête SQL
curseur.execute('SELECT * FROM Fiches;')

# on récupère toutes les lignes (rows) dans une liste
rows = curseur.fetchall()

# on affiche la liste ainsi récupérée
print(rows)

# on ferme le curseur
curseur.close()

# on se déconnecte de la base de données
connexion.close()
```

Lancer le script : quel résultat obtient-on ?

La première étape crée une connexion à la base de données. La deuxième étape crée un curseur, un objet qui permettra d'interagir avec la base de données. Par exemple la méthode « **`execute`** » permet d'exécuter une commande SQL unique (ici **`SELECT * FROM Fiches;`**). Le résultat de la commande est stocké dans l'objet curseur, et on peut le consulter grâce à la méthode « **`fetchall`** », qui va mettre les enregistrements sélectionnés dans une liste de tuples (chaque tuple correspondant à un enregistrement).

Vous trouverez plus d'informations sur le module **`sqlite3`** à l'adresse : https://docs.python.org/3/library/sqlite3.html#

II./ Création de nouveaux enregistrements :

Taper maintenant le script suivant, et l'enregistrer sous le nom « **`repertoire.py`** » :

```
# Importation de la librairie sqlite 3
import sqlite3

# Crée une connexion au fichier "repertoire.db"
# (Le fichier sera créé s'il n'existe pas.)
connexion = sqlite3.connect("repertoire.db")

# Création d'un curseur
curseur = connexion.cursor()

# Création de la table "Fiches" si celle-ci n'existe pas déjà
curseur.execute("CREATE TABLE IF NOT EXISTS Fiches(Nom TEXT, Prenom TEXT, Numero TEXT);")
connexion.commit()

fini = False

# Boucle principale du programme
while (not fini):
```

```
    # On affiche le contenu du répertoire
    curseur.execute("SELECT * FROM Fiches ORDER BY Nom, Prenom;")
    lignes = curseur.fetchall()
    if lignes == []:
        print("Le répertoire est vide")
    else:
        for ligne in lignes:
            print(ligne[0], ligne[1], ligne[2])
        print()

    # On demande les informations pour une nouvelle fiche
    nom = input("Entrer le nom de famille : (Q pour quitter) ")
    nom = nom.upper()
    if nom != "Q":
        prenom = input("Entrer le prénom : ")
        prenom = prenom.upper()
        numero = input("Entrer le numéro de téléphone : ")

        # On insère la nouvelle fiche dans la base de données
        curseur.executescript("INSERT INTO Fiches VALUES ('" + nom + "', '" + prenom + "', '" + numero + "');")
    else:
        fini = True

# On enregistre les modifications
connexion.commit()

# On ferme le curseur
curseur.close()

# On se déconnecte de la base de données
connexion.close()
```

Parmi les nouveautés, on peut remarquer l'utilisation de la méthode « **`commit`** » de l'objet « **`connexion`** », qui va enregistrer les modifications apportées à la base de données dans le fichier. On peut aussi remarquer l'utilisation de la méthode « **`executescript`** » de l'objet « **`curseur`** » qui permet d'exécuter plusieurs requêtes SQL, séparées par des points-virgules.

Exécuter le script, et entrez quelques fiches supplémentaires avant de quitter (avec « **`Q`** »). Vérifiez ensuite, à l'aide de SQL OnLine IDE que le fichier « **`repertoire.db`** » a bien été modifié.

III./ Injection SQL :

Le langage SQL permet d'effectuer des requêtes permettant de manipuler une base de données, pour la consulter ou pour la modifier. Dans certains cas, on peut essayer d'injecter dans une requête SQL prévue par un programme, d'autres requêtes qui elles ne seront pas prévues, afin de réaliser des actions normalement interdites par le système. Ce type de technique est appelé *injection SQL*.

Relancer le script précédent et, lorsque le programme demande le nom, entrer la chaîne de caractères suivante :

`t', 't', '0'); DROP TABLE Fiches; --`

Taper ensuite n'importe quoi pour le prénom et le numéro de téléphone (par exemple « **`t`** » et « **`0`** »).

Que remarque-t-on ? Ouvrir le fichier « **`repertoire.db`** » à l'aide de SQL OnLine IDE pour vérifier sa structure et son contenu.

Que s'est-il passé ? Si on regarde la chaîne de caractères transmise à l'instruction « **`curseur.executescript`** », on obtient :

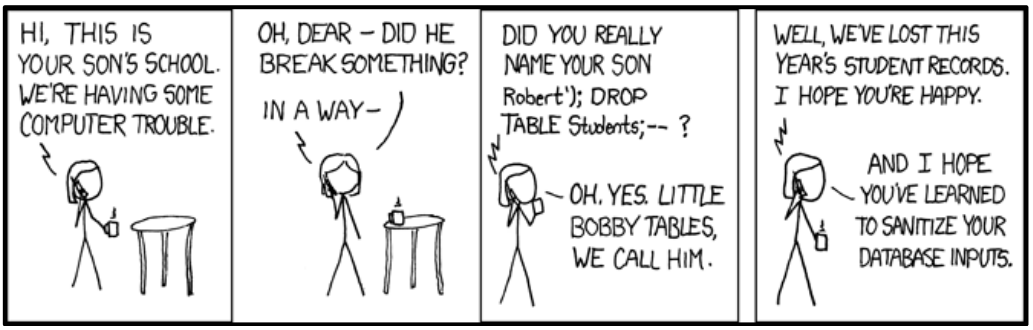
`INSERT INTO Fiches VALUES (' t', 't', '0'); DROP TABLE Fiches; --', 't', '0');`

On insère donc tout d'abord l'enregistrement suivant : **`INSERT INTO Fiches VALUES (' t', 't', '0');`**

Puis on supprime la table « **`Fiches`** » : **`DROP TABLE Fiches;`**

Le reste de la requête SQL commence par « **`--`** », ce qui correspond à un commentaire en langage SQL. Tout ce qui vient après sera donc ignoré.

Cet exemple est artificiel car le script précédent a été construit spécialement pour être vulnérable à ce type d'attaque, mais il montre le principe de base de la méthode. C'est au développeur de l'application de sécuriser les entrées de l'utilisateur afin d'interdire la possibilité d'injecter du code non prévu. Vous trouverez plus d'informations sur les injections SQL sur le site suivant : https://igm.univ-mlv.fr/~dr/XPOSE2011/injections_SQL/index.php



(*Source :* https://xkcd.com/327/)

Comment pourriez-vous modifier le script pour éviter ce genre de chose ?

IV./ Mini-projet : réalisation d'un logiciel de répertoire téléphonique :

En gardant la même structure pour la base de données, créer un logiciel de répertoire téléphonique qui permettra de consulter les enregistrements, de créer de nouvelles fiches, d'effacer des fiches, de rechercher le numéro de téléphone correspondant à une personne donnée, d'effacer tous les enregistrements, etc…

On réalisera un menu dans la console du type :

```
REPERTOIRE TELEPHONIQUE
-----------------------

1 : Liste des fiches
2 : Création d'une fiche
3 : Suppression d'une fiche
4 : Recherche d'une fiche
5 : Effacer totalement le répertoire
6 : Quitter le programme

Quel est votre choix :
```

Pour voir un exemple de programme, copier le fichier « **`repertoire_SQL.exe`** » dans le même répertoire que celui où vous avez copié le fichier « **`repertoire.db`** ». Lancer cet exécutable et tester les différentes fonctions.

Améliorations possibles :

- On peut rajouter des informations supplémentaires pour chaque fiche, comme l'adresse postale ou l'adresse mail. Il faudra modifier la structure de la base de données en conséquence, en rajoutant les attributs nécessaires.
- On peut réaliser une interface utilisateur graphique, à l'aide de la librairie « **`tkinter`** » par exemple :