Principales commandes SQLite3

Créer Base :

```
base = sqlite3.connect('nom_base.db')
```

(Commande qui crée la base, ou qui permet la connexion si elle existe déjà)

Créer Table dans la base :

```
with base:
    c = base.cursor()
    c.execute("CREATE TABLE nom_table (nom_champ TYPE, nom_champ TYPE,.....)")
```

Pour exécuter des commandes dans la base on crée un "curseur " par la commande nom_base.cursor().
Les commandes (appelées **requêtes**) sont des **chaînes** de caractères, donc ne pas oublier les guillemets.
Le TYPE d'un champ peut être : TEXT, INTEGER,REAL,
ce qui correspond en python à str, int, float.
Chaque commande doit ensuite être validée par c.commit(), sauf si on utilise "with" (employé dans la suite)

Insérer une donnée dans la table :

```
c.execute("INSERT INTO nom_table VALUES (valeur1,valeur2,...)")
```

Exemple :
On veut créer une base qui gère les membres d'un club.
Appelons la base "club" et la table "perso" qui va contenir les champs :
nom, prénom, année d'adhésion, ville.
Si on est sûr qu'il n'y aura pas 2 lignes identiques on peut se dispenser d'un identifiant supplémentaire. Mais le plus souvent on ajoute un champ numérique, qui sert de clé primaire et qui s'incrémente automatiquement.
Voici le code qui crée la base et une table, qui insère 3 personnes dans la base,
puis qui affiche le contenu de la table:

```
import sqlite3
base = sqlite3.connect('club.db')
with base:
    c = base.cursor()
    c.execute("CREATE TABLE perso (id INTEGER PRIMARY KEY, nom TEXT,prenom
TEXT,annee INTEGER,ville TEXT)")
    c.execute("INSERT INTO perso VALUES (NULL,'DURAND' , 'Pierre' , 2009 , 'Rougiers')")
    c.execute("INSERT INTO perso VALUES (NULL,'ARNAUD' , 'Alain' , 2009 , 'Nans')")
    c.execute("INSERT INTO perso VALUES (NULL,'PETIT' , 'Aude' , 2011 , 'Rougiers')")

with base:
    c = base.cursor()
    c.execute("SELECT * FROM perso")
    rows = c.fetchall()
    for row in rows:
        print (row)
base.close()
```

Remarque
Si les données à insérer sont dans des **variables** (très fréquent!), par exemple n, p, a et v on écrit

```
c.execute("INSERT INTO perso VALUES(?,?,?,?,?)",(NULL,n,p,a,v))
```

Affichage d'une table
Pour afficher le contenu de la table, la base étant ouverte on peut écrire :

```
with base:
    c = base.cursor()
    c.execute("SELECT * FROM perso")
    rows = c.fetchall()
    for row in rows:
        print (row)
```

fetchall() fournit une liste des lignes trouvées
fetchone() parcourt une ligne après l'autre.

Recherche dans la table
On veut sélectionner les membres du club qui habitent Rougiers :

```
c.execute("SELECT * FROM perso WHERE ville = 'Rougiers' ")
rows = c.fetchall()
```

Modification d'une donnée

```
c.execute("UPDATE perso SET ville='Tourves' WHERE nom='DURAND' ")
```

Suppression d'une donnée

```
c.execute("DELETE FROM perso WHERE nom='DURAND' ")
```

Remarques
Une base peut contenir plusieurs tables,
par exemple pour un club :
- la table des membres
- la table des activités du club
- la table des tarifs....

On peut faire des requêtes avec les opérateurs logiques habituels :

```
c.execute("SELECT * FROM perso WHERE ville = 'Rougiers' AND annee = 2009 ")
```

On peut mettre des variables dans toutes les requêtes :
On remplace les valeurs par des points d'interrogation, puis on précise les variables entre parenthèses à l'extérieur de la chaîne de requête.

```
c.execute("UPDATE perso SET ville = ? WHERE nom= ?",(v,n))
```

(les variables v et n contenant des chaînes)