Chapitre 10 - le SGBD Sqlite3 de python

SQLite est un système de gestion de base de données relationnelle (SGBD). Ce système et son code source sont entièrement dans le domaine public ce qui permet à tous d'utiliser et de participer à l'évolution de ce projet.

SQLite mémorise directement les données dans un fichier portant l'extension *.db* . Ce SGBD ne fonctionne pas sur une technologie de serveur comme la plupart de ses concurrents. Le fait que le code source ne soit régit par aucune licence, permet à SQLite d'être utilisé dans de nombreux logiciels et systèmes bien connus tels que Firefox, Skype, Android, l'iPhone et divers produits et projets.

1- IMPORTATION DE LA BIBLIOTHEQUE :

Il s'agit tout d'abord d'importer la bibliothèque *sqlite3* :

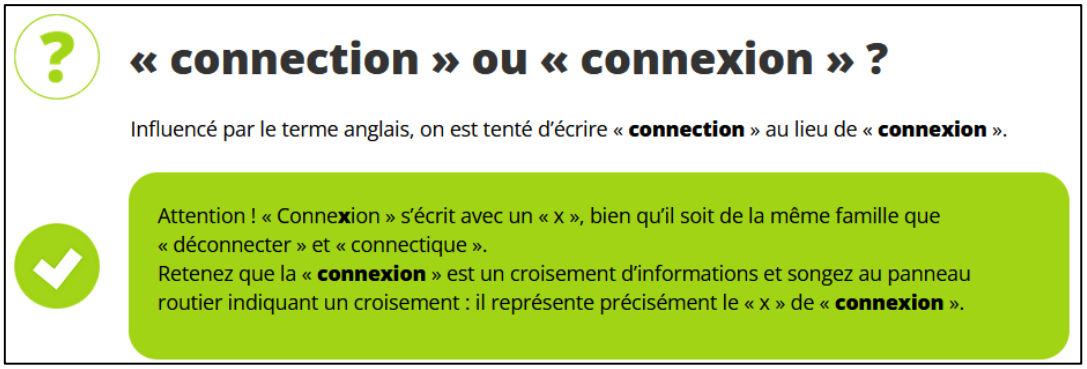
```
import sqlite3
```

```
connexion = sqlite3.connect("location.db")  #bdd dans le fichier "location.db"
```

« *connexion* » est un objet qui contient le résultat de la méthode « *connect()* » pour des objets de type *sqlite3*

```
# Il est aussi possible de stocker la bdd directement dans la RAM en utilisant le string clef ":memory:".
# Dans ce cas, il n'y aura  pas de persistance des données après la déconnexion.
connexion = sqlite3.connect(":memory:")  #bdd dans la RAM
```

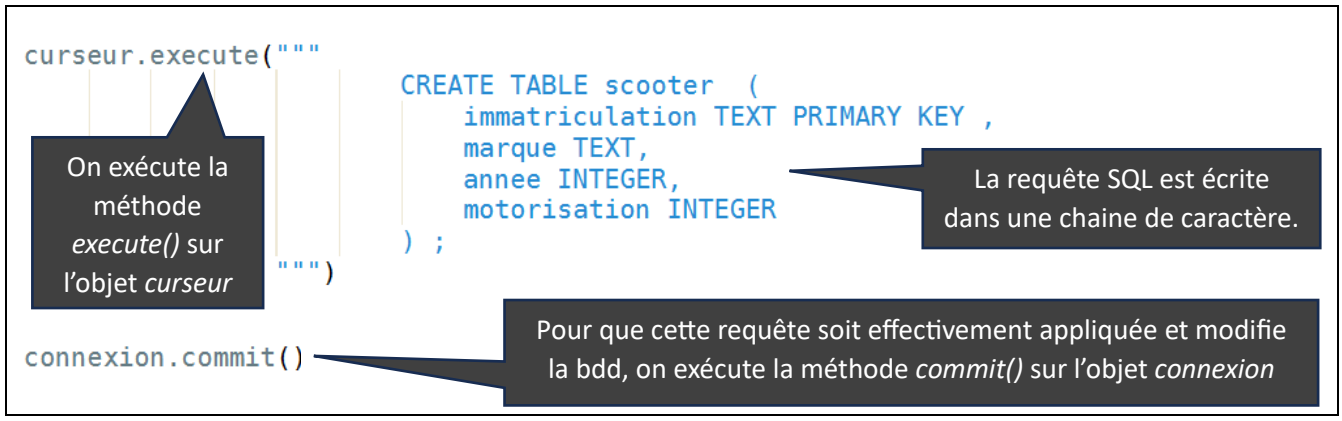
Attention, au risque de confusion évoqué ci-contre :



2- EXECUTER 1 REQUETE D'ECRITURE OU DE MODIFICATION D'UNE TABLE:

Pour exécuter une requête unique, nous allons nous servir d'un objet que l'on nomme ici *Curseur*, récupéré en faisant appel à la méthode `cursor()` de notre objet de type *connexion*.

```
curseur = connexion.cursor()  #Récupération d'un curseur (ne pas confondre cursor et curseur)
```



Si la méthode *commit()* n'est pas exécutée, la modification sur la bdd ne sera pas appliquée.

3- EXECUTER PLUSIEURS REQUETE D'ECRITURE OU DE MODIFICATION D'UNE TABLE:

Il existe plusieurs possibilités pour enchainer les requêtes SQL qui permettent de créer, remplir ou modifier une bdd. Par exemple pour remplir la table *scooter* de la bdd évoquée dans le poly de cours :

a. POSSIBILITE 1 : APPELER PLUSIEURS FOIS LA METHODE *EXECUTE()*

```
curseur.execute('''INSERT INTO scooter VALUES ('GV823AV','Yamaha',2024,125);''')
curseur.execute('''INSERT INTO scooter VALUES ('FZ154BB','Peugeot',2021,50);''')
curseur.execute('''INSERT INTO scooter VALUES ('GA101EB','Honda',2023,50);''')
curseur.execute('''INSERT INTO scooter VALUES ('GC098CC','Honda',2024,125);''')
connexion.commit()
```

Une instruction peut également s'écrire sous la forme :

```
curseur.execute('''INSERT INTO scooter VALUES (?,?,?,?);''',('GC098CC','Honda',2024,125))
connexion.commit()
```

Cela permet d'affecter plus facilement les valeurs des variables dans un code python.

b. POSSIBILITE 2 : APPELER *EXECUTE() PLUSIEURS FOIS* AVEC UNE LISTE

```
donnees = [
          ('GV823AV','Yamaha',2024,125),
          ('FZ154BB','Peugeot',2021,50),
          ('GA101EB','Honda',2023,50),
          ('GC098CC','Honda',2024,125)
          ]
for elt in donnees:
    curseur.execute('''INSERT INTO scooter VALUES (?,?,?,?)''', elt)

connexion.commit()
```

c. POSSIBILITE 3 : APPELER LA METHODE *EXECUTEMANY()* AVEC UNE LISTE DE TUPLES

```
donnees = [
            ('GV823AV','Yamaha',2024,125),
            ('FZ154BB','Peugeot',2021,50),
            ('GA101EB','Honda',2023,50),
            ('GC098CC','Honda',2024,125)
            ]
curseur.executemany('''INSERT INTO scooter VALUES (?,?,?,?)''', donnees)

connexion.commit()
```

d. POSSIBILITE 4 : APPELER LA METHODE *EXECUTEMANY()* AVEC UNE LISTE DE DICTIONNAIRES

```
donnees =   [
                {'immatriculation':'GV823AV','marque':'Yamaha','annee':2024,'motorisation':125},
                {'immatriculation':'FZ154BB','marque':'Peugeot','annee':2021,'motorisation':50},
                {'immatriculation':'GA101EB','marque':'Honda','annee':2023,'motorisation':50},
                {'immatriculation':'GC098CC','marque':'Honda','annee':2024,'motorisation':125},
            ]

curseur.executemany('''
                    INSERT INTO scooter VALUES (:immatriculation, :marque, :annee, :motorisation)
                    '''
                , donnees)

connexion.commit()
```

e. POSSIBILITE 5 : APPELER LA METHODE *EXECUTESCRIPT()*

```
curseur.executescript("""
                        INSERT INTO scooter VALUES
                        ('GV823AV','Yamaha',2024,125),
                        ('FZ154BB','Peugeot',2021,50),
                        ('GA101EB','Honda',2023,50),
                        ('GC098CC','Honda',2024,125);


                        UPDATE scooter SET motorisation = 49
                        WHERE motorisation = 50 AND marque = 'Honda';

                    """)

connexion.commit()
```

4- EXECUTER 1 REQUETE DE LECTURE ET RECUPERATION DES VALEURS :

L'exécution une requête *SELECT* avec la méthode *execute()* , nécessite d'exécuter ensuite la méthode *fetchall()* sur l'objet *curseur*, afin de pouvoir récupérer les données demandées.

Ces données sont alors retournées dans une liste de tuples, nommée *resultats* dans l'exemple donné ci-contre :

```
curseur.execute("""
                    SELECT * FROM scooter
                    WHERE annee > 2022
                """)

resultats = curseur.fetchall()
```

Le contenu de cette liste est ici :

```
>>> resultats
[('GV823AV', 'Yamaha', 2024, 125), ('GA101EB', 'Honda', 2023, 49),
('GC098CC', 'Honda', 2024, 125)]
```

Pour afficher chaque tuple individuellement, on peut effectuer un parcours de liste :

```
for tuples in resultats :
    print(tuples)
```

5- CLOTURER LA CONNEXION :

Lorsque toutes les requêtes ont été réalisées, il s'agit de clôturer la connexion qui a été créée avec le fichier de la bdd. On exécute ainsi la commande : `connexion.close()`