Base de données en Python avec sqlite3

En Python, le module **sqlite3** permet de travailler avec le SGBD (Système de Gestion de Base de Données) SQLite.

1. Connexion et déconnexion

Pour réaliser des requêtes, il est nécessaire d'importer le module sqlite3.

```
import sqlite3
```

La méthode **connect** va permettre de se connecter avec la base de données (le nom de la base est passée en paramètre). Si la base n'existe pas, elle sera créée. Un objet de type **connection** est retourné par la fonction.

```
connexion = sqlite3.connect('base_jeu.db')
```

La méthode close permet de déconnecter l'objet de type **connection**.

```
connexion.close()
```

✍ Créer un dossier **BDR**.
✍ Saisir le script ci-dessous dans un fichier nommé bre_jeu_python.py enregistré dans le dossier **BDR**.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
connexion.close()
```

✍ Vérifier que la base **base_jeu.db** est bien présente dans le répertoire **BDR**.

2. Exécuter des requêtes

2.1. Exécuter une requête simple

Pour exécuter des requêtes, il faut créer un objet de type **cursor** à partir de la méthode **cursor** qu'on applique sur l'objet de type **connection**. Cet objet va permettre de manipuler la base.

```
curseur = connexion.cursor()
```

La méthode **execute** de l'objet **cursor** permet d'effectuer une requête SQL.

```
curseur.execute("""CREATE TABLE IF NOT EXISTS JOUEUR(
                id INT PRIMARY KEY, nom TEXT, prenom TEXT)""")
```

Pour valider une requête, il faut utiliser la méthode commit sur l'objet de type **connection**.

```
connexion.commit()
```

À la fin, il faut aussi déconnecter l'objet de type **cursor**.

```
curseur.close()
```

✍ Exécuter le script ci-dessous et vérifier, avec le logiciel **DB Browser for SQLite,** que la table **JOUEUR,** avec son premier joueur, a été créé dans la base **base_jeu**.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()
curseur.execute("""CREATE TABLE IF NOT EXISTS JOUEUR(
                id INT PRIMARY KEY, nom TEXT, prenom TEXT)""")
curseur.execute("""INSERT INTO JOUEUR(id, nom, prenom) VALUES(1, "Porée", "Eva")""")
connexion.commit()
curseur.close()
connexion.close()
```

2.2. Insérer une donnée à l'aide d'un tuple

La méthode **execute** peut prendre comme deuxième paramètre un tuple contenant les données à insérer. Les points d'interrogation présents dans la requête indiquent l'emplacement des données à insérer, données qui seront insérer dans l'ordre défini par le tuple.

```
donnees = (2, "Ochon", "Paul")
curseur.execute("""INSERT INTO JOUEUR(id, nom, prenom) VALUES(?,?,?)""", donnees)
```

✍ Tester le script suivant et vérifier le contenu de la base.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()

donnees = (2, "Ochon", "Paul")
curseur.execute("""INSERT INTO JOUEUR(id, nom, prenom) VALUES(?,?,?)""", donnees)

connexion.commit()
curseur.close()
connexion.close()
```

2.3. Insérer plusieurs données

Si l'on désire insérer plusieurs données contenues dans une liste, il faut alors utiliser la méthode **executemany** à la place de **execute**.

```
liste_donnees = [(3, "Ginna", "Laurent"), (4, "Calva", "Justin"), (5, "Menvuça", "Gérard")]
curseur.executemany("""INSERT INTO JOUEUR(id, nom, prenom) VALUES(?,?,?)""", liste_donnees)
```

✍ Remplacer le script ci-dessus dans le programme précédent et vérifier le contenu de la base.

2.4. Insérer une auto incrémentation

Pour ne plus avoir à gérer l'attribut **id** (clé primaire), il suffit d'ajouter la propriété d'auto incrémentation d'unicité lors de la création de la table (attention il est nécessaire d'utiliser INTEGER à la place du INT habituel).

```
curseur.execute("""CREATE TABLE IF NOT EXISTS JOUEUR(
                id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, nom TEXT, prenom TEXT)""")
```

✍ Effacer la base **base_jeu.db** et exécuter le script ci-dessous. Vérifier le résultat obtenu.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()

liste_donnees = [("Ginna", "Laurent"), ("Calva", "Justin"), ("Menvuça", "Gérard")]

curseur.execute("""CREATE TABLE IF NOT EXISTS JOUEUR(
                id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, nom TEXT, prenom TEXT)""")
curseur.executemany("""INSERT INTO JOUEUR(nom, prenom) VALUES(?,?)""", liste_donnees)

connexion.commit()
curseur.close()
connexion.close()
```

2.5. Exécuter plusieurs requêtes

Il est aussi possible plusieurs requêtes avec la méthode **executescript**.
✍ Effacer la base **base_jeu.db** et exécuter le script ci-dessous. Vérifier le résultat obtenu.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()

curseur.executescript("""CREATE TABLE IF NOT EXISTS JOUEUR(
                      id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, nom TEXT, prenom TEXT);

    INSERT INTO JOUEUR(nom, prenom) VALUES("Cape", "Andy");
    INSERT INTO JOUEUR(nom, prenom) VALUES("Provist", "Alain");
    INSERT INTO JOUEUR(nom, prenom) VALUES("Affrit", "Barack");
    """)

connexion.commit()
curseur.close()
connexion.close()
```

2.6. Saisie de données par l'utilisateur

Il est possible de faire saisir les données par l'utilisateur.

```
donnees = (str(input("nom ? ")), str(input("prenom ? ")))
```

✍ Exécuter le script ci-dessous. Vérifier le résultat obtenu.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()

donnees = (str(input("Nom ? ")), str(input("Prénom ? ")))

curseur.execute("""INSERT INTO JOUEUR(nom, prenom) VALUES(?,?)""", donnees)

connexion.commit()
curseur.close()
connexion.close()
```

2.7. Modifier une saisie

✍ Exécuter le script ci-dessous. Vérifier le résultat obtenu.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()

new_donnees = ("Obama", "Barack")

curseur.execute("""UPDATE JOUEUR SET nom = ? WHERE prenom = ?""", new_donnees)

connexion.commit()
curseur.close()
connexion.close()
```

2.8. Supprimer une saisie

✍ Exécuter le script ci-dessous. Vérifier le résultat obtenu.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()

suppr_donnees = ("Obama",)

curseur.execute("""DELETE FROM JOUEUR WHERE nom =?""", suppr_donnees)

connexion.commit()
curseur.close()
connexion.close()
```

Attention, comme le deuxième paramètre de la méthode **execute** doit être un tuple, il est nécessaire de placer une virgule après le premier élément du tuple : **("Obama",)**.

3. Interrogation de la base

3.1. Récupérer un résultat

La méthode **fetchone** retourne un résultat sous la forme d'un tuple, ou **None**, s'il n'y en a pas.
✍ Exécuter le script ci-dessous. Vérifier le résultat obtenu.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()

donnees = ("Cape",)
curseur.execute("""SELECT prenom FROM JOUEUR WHERE nom =?""", donnees)
prenom = curseur.fetchone()
print(prenom)

connexion.commit()
curseur.close()
connexion.close()
```

3.2. Récupérer une liste de résultats

La méthode **fetchall** retourne un résultat sous la forme d'une liste qui contient des tuples.
✍ Exécuter le script ci-dessous. Vérifier le résultat obtenu.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()

curseur.execute("""SELECT nom, prenom FROM JOUEUR""")
liste = curseur.fetchall()
print(liste)

connexion.commit()
curseur.close()
connexion.close()
```

4. Manipuler plusieurs tables

Une partie du script ci-dessous permet de créer une autre table (**JEU_1**) avec comme clé étrangère **id** de la table **JOUEUR**.
La table **JEU_1** contient les scores des participants au premier jeu.
✍ Analyser le script ci-dessous, l'exécuter, et vérifier le résultat.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()

score1 = (1, 15)
score2 = (2, 13)

curseur.execute("""CREATE TABLE IF NOT EXISTS JOUEUR(
    id INT PRIMARY KEY, nom TEXT, prenom TEXT)""")
curseur.execute("""INSERT INTO JOUEUR(nom, prenom) VALUES("Bon", "Jean")""")
curseur.execute("""CREATE TABLE IF NOT EXISTS JEU_1(
                id_jeu_1 INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, id_joueur INT, score INT,
                FOREIGN KEY (id_joueur) REFERENCES JOUEUR(id))""")

curseur.execute("""INSERT INTO JEU_1 (id_joueur, score) VALUES(?, ?)""", score1)
curseur.execute("""INSERT INTO JEU_1 (id_joueur, score) VALUES(?, ?)""", score2)

connexion.commit()
curseur.close()
connexion.close()
```

Pour récupérer le meilleur score du premier jeu.
✍ Exécuter le script ci-dessous. Vérifier le résultat obtenu.

```
import sqlite3

connexion = sqlite3.connect('base_jeu.db')
curseur = connexion.cursor()

curseur.execute("""SELECT nom, prenom, score FROM JEU_1
                JOIN JOUEUR ON JOUEUR.id = JEU_1.id_joueur
                ORDER BY score DESC LIMIT 1""")

liste = curseur.fetchall()
print(liste)

connexion.commit()
curseur.close()
connexion.close()
```

Source : https://pixees.fr/informatiquelycee/n_site/nsi_term_projet_1.html
https://zestedesavoir.com/tutoriels/1294/des-bases-de-donnees-en-python-avec-sqlite3/fonctionnalites-de-base/