COURS DE NSI (Numérique et Sciences Informatiques)

Classe : terminale	LYCÉE INTERNATIONAL COURS LUMIÈRE
séquence 1	**Projet 1 : REPERTOIRE TELEPHONIQUE**
Mr BANANKO K.	**Chap 1**

Création d'un répertoire téléphonique (version web + base de données)

Avant d'évoquer le projet en tant que tel, nous allons commencer par travailler sur les requêtes SQL effectuées depuis un programme Python. Il est donc nécessaire de bien maitriser toute la partie consacrée aux bases de données et particulièrement le cours sur les requêtes SQL (voir le cours si nécessaire)

Activité 1

Après avoir créé un répertoire "projet_bd". Créez, à l'aide de spyder, un fichier Python (à vous de choisir le nom) puis saisissez et exécutez le programme suivant :

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

cur.execute("CREATE TABLE LIVRES(id INT, titre TEXT, auteur TXT, ann_publi INT, note INT)")
conn.commit()

cur.close()
conn.close()
```

Analysons le programme ci-dessus :

Ce programme va vous permettre de vous "**connecter**" à une base de données (si cette dernière n'existe pas, elle sera créée). Ensuite nous créons une table (une relation) nommée LIVRES, cette table contient 4 attributs : id (de type entier), titre (de type texte), auteur (de type texte), ann_publi (de type entier) et note (de type entier).

Entrons un peu dans les détails en analysant le programme ligne par ligne :

```
import sqlite3
```

Nous commençons par importer la bibliothèque **sqlite3**. Cette bibliothèque va nous permettre d'effectuer des requêtes SQL sur une base de données. Comme dans le cours sur les bases de données, nous utiliserons le SGBD SQLite.

```
conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()
```

Nous créons un objet de type "**connection**" (conn) qui va nous permettre d'interagir avec la base de données "baseDonnees.db" (comme dit plus haut, si cette base de données n'existe pas, elle est créée). Vous devriez donc avoir un fichier "baseDonnees.db" dans le même répertoire que votre fichier Python.

Nous créons ensuite un objet de type "cursor" à partir de l'objet de type "connection". Cet objet de type "cursor" va nous permettre de manipuler la base de données et d'obtenir des résultats lorsque nous effectuerons des requêtes.

```
cur.execute("CREATE TABLE LIVRES(id INT, titre TEXT, auteur TXT, ann_publi INT, note INT)")
```

La méthode "**execute**" de notre objet de type "cursor" nous permet d'effectuer une requête SQL. Cette requête SQL est en tout point identique à ce que nous avons vu dans le cours sur les bases de données.

```
conn.commit()
```

Pour véritablement exécuter les requêtes, il est nécessaire d'appliquer la méthode "commit" à l'objet de type "**connection**".

```
cur.close()
conn.close()
```

Avant de terminer le programme, il nécessaire de "fermer" l'objet de type "cursor" et l'objet de type "connection".

Nous allons systématiquement retrouver cette structure dans nos futurs programmes :

- *création d'un objet de type "connection"*
- *création d'un objet de type "cursor"*
- *préparation d'une ou plusieurs requête(s) (méthode "execute" sur l'objet de type "cursor")*
- *exécution réelle des requêtes (méthode "commit" sur l'objet de type "connection")*
- *"fermeture" de l'objet de type "cursor" puis de l'objet de type "connection"*

Si vous exécutez une deuxième fois le programme proposé au " **Activité 1**", vous aurez droit à une erreur : "**OperationalError: table LIVRES already exists**". Afin d'éviter ce genre de problème, il est possible de modifier le programme afin que la requête de création de la table LIVRES ne se fasse pas si la table LIVRES existe déjà:

Activité 2

Après avoir effacé le fichier "baseDonnees.db", saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')

cur = conn.cursor()



cur.execute("CREATE TABLE IF NOT EXISTS LIVRES(id INT, titre TEXT, auteur TXT, ann_publi IN
T, note INT)")



conn.commit()



cur.close()

conn.close()
```

Comme vous pouvez le constater, si vous exécutez le programme plusieurs fois de suite, il n'y a plus d'erreur.

Activité 3

Ouvrez le fichier "baseDonnees.db" à l'aide du logiciel "DB Browser for SQLite" et vérifiez que la table LIVRES a bien été créée.

Maintenant que notre table LIVRES a été créée, nous allons pouvoir commencer à la "remplir" avec des données :

Activité 4

Saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS LIVRES(id INT, titre TEXT, auteur TXT, ann_publi IN
T, note INT)")

cur.execute("INSERT INTO LIVRES(id,titre,auteur,ann_publi,note) VALUES(1,'1984','Orwell',19
49,10)")



conn.commit()



cur.close()

conn.close()
```

Rien de particulier à signaler, la requête INSERT est identique à ce qui a été vu dans le cours sur les bases de données.

Activité 5

Ouvrez le fichier "baseDonnees.db" à l'aide du logiciel "DB Browser for SQLite" et vérifiez que les données ont bien été ajoutées à la table LIVRES.

Nous avons inclus les données à insérer directement dans la requête. Il est possible de procéder autrement en séparant les données à insérer et la requête (cela s'avérera particulièrement pratique dans le futur)

Activité 6

Après avoir effacé le fichier "baseDonnees.db", saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3



conn = sqlite3.connect('baseDonnees.db')

cur = conn.cursor()



data = (1,'1984','Orwell',1949,10)



cur.execute("CREATE TABLE IF NOT EXISTS LIVRES(id INT, titre TEXT, auteur TXT, ann_publi IN
T, note INT)")

cur.execute("INSERT INTO LIVRES(id,titre,auteur,ann_publi,note) VALUES(?, ?, ?, ?, ?)", dat
a)

conn.commit()



cur.close()

conn.close()
```

Première chose à remarquer, nous avons créé un tuple (**data**) contenant toutes les informations. En effet, la méthode "**execute**" peut prendre un deuxième paramètre un tuple contenant les données à insérer.

Les points d'interrogation présents dans la requête indiquent l'emplacement des données à insérer. Le premier ? sera remplacé par le premier élément du tuple (dans notre cas 1),

le deuxième ? sera remplacé par le deuxième élément du tuple (dans notre cas '1984') et ainsi de suite...

Si l'on désire insérer plusieurs données, il est possible de regrouper toutes les données à insérer dans un tableau et d'utiliser la méthode "**executemany**" à la place de la méthode "**execute**".

Activité 7

Après avoir effacé le fichier "baseDonnees.db", saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3


conn = sqlite3.connect('baseDonnees.db')

cur = conn.cursor()


datas = [

        (1,'1984','Orwell',1949,10),

        (2,'Dune','Herbert',1965,8),

        (3,'Fondation','Asimov',1951,9),

        (4,'Le meilleur des mondes','Huxley',1931,7),

        (5,'Fahrenheit 451','Bradbury',1953,7),

        (6,'Ubik','K.Dick',1969,9),

        (7,'Chroniques martiennes','Bradbury',1950,8),

        (8,'La nuit des temps','Barjavel',1968,7),

        (9,'Blade Runner','K.Dick',1968,8),

        (10,'Les Robots','Asimov',1950,9),

        (11,'La Planète des singes','Boulle',1963,8),

        (12,'Ravage','Barjavel',1943,8),

        (13,'Le Maître du Haut Château','K.Dick',1962,8),

        (14,'Le monde des Ā','Van Vogt',1945,7),

        (15,'La Fin de l’éternité','Asimov',1955,8),

        (16,'De la Terre à la Lune','Verne',1865,10)

 ]

cur.execute("CREATE TABLE IF NOT EXISTS LIVRES(id INT, titre TEXT, auteur TXT, ann_publi IN
T, note INT)")

cur.executemany("INSERT INTO LIVRES(id,titre,auteur,ann_publi,note) VALUES(?, ?, ?, ?, ?)",
datas)

conn.commit()

cur.close()

conn.close()
```

Activité 8

Ouvrez le fichier "baseDonnees.db" à l'aide du logiciel "DB Browser for SQLite" et vérifiez que les données ont bien été ajoutées à la table LIVRES.

Il n'est pas très pratique d'avoir à gérer l'id (clé primaire). En effet, si je veux ajouter un nouveau livre, il faudra que je connaisse l'id du précédent (incrémentation de l'id). Heureusement, il est possible d'automatiser cette incrémentation.

Activité 9

Après avoir effacé le fichier "baseDonnees.db", saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3


conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()


datas = [
        ('1984','Orwell',1949,10),
        ('Dune','Herbert',1965,8),
        ('Fondation','Asimov',1951,9),
        ('Le meilleur des mondes','Huxley',1931,7),
        ('Fahrenheit 451','Bradbury',1953,7),
        ('Ubik','K.Dick',1969,9),
        ('Chroniques martiennes','Bradbury',1950,8),
        ('La nuit des temps','Barjavel',1968,7),
        ('Blade Runner','K.Dick',1968,8),
        ('Les Robots','Asimov',1950,9),
        ('La Planète des singes','Boulle',1963,8),
```

```
        ('Ravage','Barjavel',1943,8),
        ('Le Maître du Haut Château','K.Dick',1962,8),
        ('Le monde des Ā','Van Vogt',1945,7),
        ('La Fin de l'éternité','Asimov',1955,8),
        ('De la Terre à la Lune','Verne',1865,10)
 ]


cur.execute("CREATE TABLE IF NOT EXISTS LIVRES(id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
titre TEXT, auteur TXT, ann_publi INT, note INT)")
cur.executemany("INSERT INTO LIVRES(titre,auteur,ann_publi,note) VALUES(?, ?, ?, ?)", datas
)
conn.commit()


cur.close()
conn.close()
```

Ouvrez le fichier "baseDonnees.db" à l'aide du logiciel "DB Browser for SQLite" et vérifiez que les données ont bien été ajoutées à la table LIVRES.

Vous pouvez constater que nous avons bien l'attribut "id", même si ce dernier n'a pas été renseigné dans les données (absence d'id dans le tableau datas).

Désormais l'id sera incrémenté automatiquement grâce au "**id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE**" (attention il est nécessaire d'utiliser INTEGER à la place du INT habituel) présent dans la requête de création de la table LIVRES. Attention, de bien penser à supprimer un ? dans la requête d'insertion (chaque tuple contient maintenant 4 éléments (nous en avions 5 quand l'id n'était pas géré automatiquement)).

Il est tout à fait possible de rajouter une nouvelle donnée :

Activité 10

Saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')

cur = conn.cursor()

nvx_data = ('Hypérion','Simmons',1989,8)


cur.execute("INSERT INTO LIVRES(titre,auteur,ann_publi,note) VALUES(?, ?, ?, ?)", nvx_data)

conn.commit()


cur.close()

conn.close()
```

Ouvrez le fichier "baseDonnees.db" à l'aide du logiciel "DB Browser for SQLite" et vérifiez que les données ont bien été ajoutées à la table LIVRES.

Vous pouvez remarquer que le nouvel enregistrement a bien l'id 17 et que nous n'avons pas eu à nous en occuper.

Il est possible de modifier des données déjà présentes dans la table.

Activité 11

Saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')

cur = conn.cursor()

modif = (7, 'Hypérion')

cur.execute('UPDATE LIVRES SET note = ? WHERE titre = ?', modif)

conn.commit()

cur.close()

conn.close()
```

Ouvrez le fichier "baseDonnees.db" à l'aide du logiciel "DB Browser for SQLite" et vérifiez que les données ont bien été modifiées dans la table LIVRES.

Comme vous pouvez le constater, il est possible d'utiliser la clause WHERE avec un ?

Il est aussi possible de supprimer une donnée :

Activité 12

Saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3


conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()


suppr = ('Hypérion',)
cur.execute('DELETE FROM LIVRES WHERE titre = ?', suppr)
conn.commit()


cur.close()
conn.close()
```

Ouvrez le fichier "baseDonnees.db" à l'aide du logiciel "DB Browser for SQLite" et vérifiez que l'entrée "Hypérion" a été supprimée de la table LIVRES.

Attention, le deuxième paramètre de la méthode "execute" doit être un tuple, si on écrit seulement suppr = ('Hypérion'), suppr est une chaine de caractère et pas un tuple. Pour avoir un tuple avec un seul élément il est nécessaire d'ajouter une virgule (d'où le suppr = ('Hypérion',))

Enfin, pour terminer cette introduction sur l'utilisation de sqlite en Python, nous devons nous intéresser aux requêtes de type "SELECT" :

Activité 13

Saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3

conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()

cur.execute('SELECT * FROM LIVRES')
conn.commit()

liste = cur.fetchall()

cur.close()
conn.close()
```

À l'aide de la console, déterminez la valeur référencée par la variable liste

Comme vous pouvez le constater, la variable liste est un tableau qui contient des tuples. Chaque tuple est un enregistrement de la table LIVRES. La méthode "fetchall" d'un objet de type "cursor" renvoie un tableau contenant des tuples

Il est possible d'avoir des requêtes plus sélectives :

Activité 14

Saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3


conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()


cur.execute('SELECT titre FROM LIVRES WHERE ann_publi < 1970')
conn.commit()


liste = cur.fetchall()


cur.close()
conn.close()
```

À l'aide de la console, déterminez la valeur référencée par la variable liste

Vous pouvez constater que l'on obtient bien un tableau contenant des tuples (nous avons bien des tuples même si seuls les titres ont été sélectionnés)

Il est possible d'utiliser les points d'interrogation dans une requête de type SELECT:

Activité 15

Saisissez le programme ci-dessous, puis exécutez-le

```
import sqlite3


conn = sqlite3.connect('baseDonnees.db')
cur = conn.cursor()


recherche = (1960, 8)
```

```
cur.execute('SELECT titre FROM LIVRES WHERE ann_publi < ? AND note > ?', recherche)
conn.commit()


liste = cur.fetchall()


cur.close()
conn.close()
```

À l'aide de la console, déterminez la valeur référencée par la variable liste

Projet "Création d'un répertoire téléphonique Simple"

La gestion de répertoire téléphonique en utilisant 3 fonctions : "menu", "lecture" et "ecriture".

Assurez-vous d'avoir un fichier texte nommé "repertoire.txt" pour stocker les noms et numéros de téléphone.

```
# Fonction pour afficher le menu
def menu():
    print("0-quitter")
    print("1-écrire dans le répertoire")
    print("2-rechercher dans le répertoire")
    choix = input("Votre choix ? ")
    return choix
# Fonction pour écrire dans le répertoire
def ecriture():
    with open("repertoire.txt", "a") as fichier:
        while True:
            nom = input("Nom (0 pour terminer) : ")
            if nom == "0":
                break
            numero = input("Numéro de téléphone : ")
            fichier.write(f"{nom} : {numero}\n")

# Fonction pour rechercher dans le répertoire
def lecture():
    nom_recherche = input("Entrer un nom : ")
    found = False
    with open("repertoire.txt", "r") as fichier:
        for ligne in fichier:
            if nom_recherche in ligne:
                found = True
                print("Le numéro recherché est :", ligne.split(":")[1].strip())
```

```
            break
    if not found:
        print("Inconnu")
# Programme principal
while True:
    choix = menu()
    if choix == "0":
        break
```

Projet "Création d'un répertoire téléphonique Simple (version base de données)"

```
import sqlite3
import tkinter as tk

from tkinter import ttk

# Création de la base de données ou connexion à une existante

conn = sqlite3.connect('repertoire_telephonique.db')

cursor = conn.cursor()

# Création de la table pour stocker les contacts

cursor.execute('''CREATE TABLE IF NOT EXISTS contacts (id INTEGER PRIMARY KEY AUTOINCREMENT, nom TEXT,numero TEXT)''')

conn.commit()

# Fonction pour ajouter un contact à la base de données

def ajouter_contact():

    nom = nom_entry.get()

    numero = numero_entry.get()

    cursor.execute("INSERT INTO contacts (nom, numero) VALUES (?, ?)", (nom, numero))

    conn.commit()

    afficher_contacts()


# Fonction pour afficher les contacts dans l'interface utilisateur

def afficher_contacts():

    for row in tree.get_children():

        tree.delete(row)

    cursor.execute("SELECT * FROM contacts")

    contacts = cursor.fetchall()

    for contact in contacts:

        tree.insert("", "end", values=contact)


# Interface utilisateur

root = tk.Tk()

root.title("Répertoire Téléphonique")


frame = ttk.Frame(root)

frame.pack(padx=10, pady=10)
```

```
nom_label = ttk.Label(frame, text="Nom:")
nom_label.grid(row=0, column=0)

numero_label = ttk.Label(frame, text="Numéro de téléphone:")
numero_label.grid(row=1, column=0)

nom_entry = ttk.Entry(frame)
nom_entry.grid(row=0, column=1)

numero_entry = ttk.Entry(frame)
numero_entry.grid(row=1, column=1)

ajouter_button = ttk.Button(frame, text="Ajouter", command=ajouter_contact)
ajouter_button.grid(row=2, column=0, columnspan=2)

tree = ttk.Treeview(frame, columns=("ID", "Nom", "Numéro de téléphone"))
tree.grid(row=3, column=0, columnspan=2)
tree.heading("ID", text="ID")
tree.heading("Nom", text="Nom")
tree.heading("Numéro de téléphone", text="Numéro de téléphone")
tree.column("ID", width=30)
tree.column("Nom", width=150)
tree.column("Numéro de téléphone", width=100)
afficher_contacts()
root.mainloop()
# Fermer la connexion à la base de données lorsqu'on quitte l'application
conn.close()
import sqlite3
import tkinter as tk
from tkinter import ttk
# Création de la base de données ou connexion à une existante
conn = sqlite3.connect('repertoire_telephonique.db')
cursor = conn.cursor()
# Création de la table pour stocker les contacts
```

```
cursor.execute("'CREATE TABLE IF NOT EXISTS contacts ( id INTEGER PRIMARY KEY AUTOINCREMENT, nom TEXT, numero TEXT )")

conn.commit()

# Fonction pour ajouter un contact à la base de données

def ajouter_contact():

    nom = nom_entry.get()

    numero = numero_entry.get()

    cursor.execute("INSERT INTO contacts (nom, numero) VALUES (?, ?)", (nom, numero))

    conn.commit()

    afficher_contacts()

# Fonction pour afficher les contacts dans l'interface utilisateur

def afficher_contacts():

    for row in tree.get_children():

        tree.delete(row)

    cursor.execute("SELECT * FROM contacts")

    contacts = cursor.fetchall()

    for contact in contacts:

        tree.insert("", "end", values=contact)


# Interface utilisateur

root = tk.Tk()

root.title("Répertoire Téléphonique")


frame = ttk.Frame(root)

frame.pack(padx=10, pady=10)


nom_label = ttk.Label(frame, text="Nom:")
nom_label.grid(row=0, column=0)

numero_label = ttk.Label(frame, text="Numéro de téléphone:")
numero_label.grid(row=1, column=0)

nom_entry = ttk.Entry(frame)
nom_entry.grid(row=0, column=1)

numero_entry = ttk.Entry(frame)
numero_entry.grid(row=1, column=1)

ajouter_button = ttk.Button(frame, text="Ajouter", command=ajouter_contact)
ajouter_button.grid(row=2, column=0, columnspan=2)

tree = ttk.Treeview(frame, columns=("ID", "Nom", "Numéro de téléphone"))
tree.grid(row=3, column=0, columnspan=2)
tree.heading("ID", text="ID")
```

```
tree.heading("Nom", text="Nom")
tree.heading("Numéro de téléphone", text="Numéro de téléphone")
tree.column("ID", width=30)
tree.column("Nom", width=150)
tree.column("Numéro de téléphone", width=100)

afficher_contacts()

root.mainloop()

# Fermer la connexion à la base de données lorsqu'on quitte l'application
conn.close()
```