



*Base de données*

Samuel SORIN  
contact@samuelsorin.fr

Base de données

# SQLite - présentation

- SQLite est une base de données légère
- Permet d'intégrer une bdd dans le programme même.
- La librairie pour utiliser SQLite est intégrée par défaut dans python
- SQLite est essentiellement utilisé dans les environnements de développement ou en tant que base de données pour les applications mobiles.
- <https://www.sqlite.org/>

# SQLite - utilisation

- Vous pouvez administrer directement une base de données SQLite en utilisant un browser tel que <https://sqlitebrowser.org/>
- Se connecter à une base de données :  
**note** : Si la base de données n'existe pas, elle sera créée.

```
import sqlite3  
  
conn = sqlite3.connect("my_db_name.db")
```

- **Attention**, toutes connexion à la base de données nécessite une déconnexion en fin d'opération

```
conn.close()
```

# SQLite - utilisation

- Exécuter une requête SQL (create, insert, update, delete...):

Pour une requête de type create, insert, update, delete, (...) il est possible de l'exécuter directement puis de la valider (commit)

```
import sqlite3

conn = sqlite3.connect("my_db_name.db")

cursor = conn.cursor()
cursor.execute(""" ma requete SQL""")
conn.commit()

conn.close()
```

# SQLite - utilisation

- Exécuter une requête SQL

Pour une requête de type select, il faut l'exécuter puis récupérer une seule ligne de résultat

```
import sqlite3

conn = sqlite3.connect("my_db_name.db")

cursor = conn.cursor()
cursor.execute(""" ma requete SQL""")

result = cursor.fetchone()      # récupération d'une seule ligne de résultat

conn.close()
```

# SQLite - utilisation

- Exécuter une requête SQL:

Pour une requête de type select, il faut l'exécuter puis récupérer **plusieurs lignes de résultat**

```
import sqlite3

conn = sqlite3.connect("my_db_name.db")

cursor = conn.cursor()
cursor.execute(""" ma requete SQL""")

results = cursor.fetchall()           # récupération de x lignes de resultat
for row in results:
    print(row[0], row[1])

conn.close()
```

# SQLite - récap

Utiliser le module SQLite	<pre>import sqlite3</pre>
Créer / se connecter à une bdd	<pre>conn = sqlite3.connect('ma_base.db')</pre>
Déconnecter la bdd	<pre>conn.close()</pre>
Exécuter des requêtes	<pre>cursor = conn.cursor() cursor.execute(""" ma requete SQL """) conn.commit()</pre>
Récupérer des données	<pre>result = cursor.fetchone()</pre>
	<pre>results = cursor.fetchall() for row in results:     print(row[0], row[1])</pre>
Revenir au dernier commit	<pre>conn.rollback()</pre>



# SQLite - exercice 1

a - Dans un module Python, connectez-vous à une base de données SQLite et créez une table “article” avec les colonnes suivantes :

- id : *integer, auto-incrémentation, unique, obligatoire*
- title : string, obligatoire
- resume : string, obligatoire
- content : string, obligatoire

b - Insérer des données dans la table (4 ou 5 articles)

c - Consulter la nouvelle base de données directement avec le browser  
sqlitebrowser

## SQLite - exercice 2

- a - Dans un module Python, interroger la base de données et afficher l'ensemble des articles présents dans la table article (l'affichage doit se faire directement dans le terminal)
- b - Afficher un article en particulier (*par exemple l'article avec l'id 3*)
- c - Supprimer un article à l'aide d'une requête SQL (par exemple l'article avec l'id 3)

# SQLite - exercice 3

Reprenez l'exercice de todo liste et refaite le avec une base de données SQLite

Créer une todo liste en version html.

- Il doit être possible de d'ajouter, supprimer et vider des éléments de cette liste
- les données sont sauvegardé dans une base de données SQLite

Vous devez utiliser un serveur web python et jinja et SQLite

# SQLite - exercice 4

Création d'un blog avec une base de données SQLite. Le blog doit comprendre :

- une page listant tous les articles
- une page détail d'un article
- une page ajout d'un article

**Note :** Vous devez utiliser un serveur web python et jinja et SQLite

Si vous avez de l'avance, vous pouvez ajouter les fonctionnalités suivantes :

- modification / suppression d'un article
- protection de l'ajout d'article par un mot de passe

# MySQL - présentation

- MySQL est l'une des bases de données les plus utilisées pour les projets web
- Plus robuste et puissant que SQLite
- Il est nécessaire de l'installer et de le configurer avant de l'utiliser
- La base de données que l'on utilisera dans le projet devra être créée en amont

Installer la librairie Python pour dialoguer avec MySQL :

```
>>> pip install pymysql
```

# MySQL - récap

Utiliser le module SQLite	<pre>import pymysql</pre>
Créer / se connecter à une bdd	<pre>conn = pymysql.connect("SERVEUR_NAME", "USER", "PASSWORD", "DB_NAME" )</pre>
Déconnecter la bdd	<pre>conn.close()</pre>
Exécuter des requêtes	<pre>cursor = conn.cursor() cursor.execute(""" ma requete SQL """) conn.commit()</pre>
Récupérer des données	<pre>result = cursor.fetchone()</pre>
	<pre>results = cursor.fetchall() for row in results:     print(row[0], row[1])</pre>
Revenir au dernier commit	<pre>conn.rollback()</pre>

# MySQL - exercice

a - Reprenez les exercices 1 et 2 de SQLite et faite la même chose avec une base de données MySQL

b - Créez une fonction permettant de choisir la base de données à utiliser. Puis pour chaque fonction (affichage de tous les articles, affichage d'un article ...), demander à l'utilisateur la base de données qu'il souhaite utiliser