BTS SN-IR	Manipulation de bases de données MySQL avec Python	2ème année

I. Installer le connecteur MySQL

La manipulation des bases de données MySQL avec Python suit la même logique que dans les autres langages de programmation.

Pour commencer il vous faudra installer la bibliothèque mysql.connector.

Pour cela on utilise :

```
Apt-get install python3-mysql.connector
```

Si vous êtes sur Windows, vous pouvez utiliser la commande conda dans la fenêtre de commande Anaconda avec :

```
conda install -c anaconda mysql-connector-python
```

II. Se connecter à la base

Une fois ces étapes effectuées, vous êtes parés pour utiliser MySQL directement depuis Python.

Voici un code d'exemple pour se connecter à une base de données : « exemple »

```
import mysql.connector
from mysql.connector import Error
try:
   print("Try to connected to MySQL Server")
   connection = mysql.connector.connect(host='localhost',
                                        database='exemple',
                                        user="admin",
                                        password="12345")

   db_Info = connection.get_server_info()
   print("Connected to MySQL Server version", db_Info)


except Error as e:
   print("Error while connecting to MySQL", e)
```

Dans cet exemple, la base de données se nomme « exemple », l'utilisateur « admin », le mot de passe de connexion : « 12345 » et enfin on se connecte depuis la même machine que celle où est stockée la bdd donc l'hôte est « localhost ».

La connexion s'effectue toujours avec l'instruction Try et renverra donc une erreur si jamais la connexion échoue.

III. Exécuter des requêtes

Les requêtes SQL sont des chaînes de caractères et s'exécute grâce à la commande « execute » sur un object de type cursor.

Exemple : `cursor.execute("show databases;")`

Le résultat de la requête se récupère ensuite grâce à la méthode fetchone ou fetchall (en fonction de si le résultat est unique ou non) :

Voici un code permettant de savoir à quel base l'utilisateur est connecté, puis de montrer toutes les bases de données existantes :

```
cursor = connection.cursor()
cursor.execute("select database();")
record = cursor.fetchone()
print("You are connected to database: ", record)

cursor = connection.cursor()
cursor.execute("show databases;")
records = cursor.fetchall()
print("Databases: ", records)
```

En général, les requêtes SQL s'effectuent à l'intérieur d'une instruction « try » pour récupérer les messages d'erreurs s'il y en a.
Voici un exemple de fonction permettant d'insérer deux valeurs étant stockées dans une liste.

```
def insertion(capt):
    try:
        mySql_insert_query = "INSERT INTO mesures(id_capteur,valeur) VALUES('"
+
        str(capt[0]) + "','" + str(capt[1]) + "')"
        cursor = connection.cursor()
        result = cursor.execute(mySql_insert_query)
        connection.commit()
        cursor.close()


    except mysql.connector.Error as error:
        print("Failed to insert record into table {}".format(error))
        return False
    return True
```

Dans cet exemple, on stock la chaîne de caractère dans la variable mySql_insert_query avant de l'exécuter.

IV. Fermer la connexion

Lorsque les requêtes sont terminées il faut TOUJOURS penser à fermer la connexion avec MySQL pour des raisons de sécurités et pour éviter les erreurs d'insertion malheureuses.

Voici le code nécessaire :

```
if connection.is_connected():
   cursor.close()
   connection.close()
   print("MySQL connection is closed")
```

V. Exemple de code complet

Voici un exemple complet avec insertion de 3 nombres aléatoires dans la table.

```
import mysql.connector
from mysql.connector import Error
from random import randint, uniform


def insertion(capt):
    try:
        mySql_insert_query = "INSERT INTO mesures(id_capteur,valeur)
VALUES('" +
       str(capt[0]) + "','" + str(capt[1]) + "')"
        cursor = connection.cursor()
        result = cursor.execute(mySql_insert_query)
        connection.commit()
        cursor.close()


    except mysql.connector.Error as error:
        print("Failed to insert record into table {}".format(error))
        return False
    return True


try:
   print("Try to connected to MySQL Server")
   connection = mysql.connector.connect(host='localhost',
                                        database='exempleProjetSN2',
                                        user="prof",
                                        password="prof")

   db_Info = connection.get_server_info()
   print("Connected to MySQL Server version", db_Info)
   cursor = connection.cursor()
   cursor.execute("select database();")
```

```
   record = cursor.fetchone()
   print("You are connected to database: ", record)

   cursor = connection.cursor()
   cursor.execute("show databases;")
   records = cursor.fetchall()
   print("Databases: ", records)

   cursor = connection.cursor()
   cursor.execute("show tables;")
   records = cursor.fetchall()
   print("All tables: ", records)


except Error as e:
   print("Error while connecting to MySQL", e)


insertion([randint(1,3),round(uniform(5,35),1)])
insertion([randint(4,6),randint(0,1000)])
insertion([randint(7,9),round(uniform(0,5),1)])


if connection.is_connected():
   cursor.close()
   connection.close()
   print("MySQL connection is closed")
```