MANEJO DE BASES DE DATOS EN PYTHON

En Python, el acceso a bases de datos se encuentra definido a modo de estándar en las especificaciones de DB-API. Esto, significa que independientemente de la base de datos que utilicemos, los métodos y procesos de conexión, lectura y escritura de datos, desde Python, siempre serán los mismos, más allá del conector.

En nuestro caso, utilizaremos MySQL, para lo cual, vamos a trabajar con el módulo **MySQLdb**.

A diferencia de los módulos de la librería estándar de Python, MySQLdb debe ser instalado manualmente. Para ello:

```
    pip install mysql-connector-python
```

o

```
    py -m pip install mysql-connector-python
```

CONECTARSE A LA BASE DE DATOS Y EJECUTAR CONSULTAS

Para conectarnos a la base de datos y ejecutar cualquier consulta, el procedimiento consta de los siguientes pasos:

1. Abrir la conexión y crear un cursor
2. Ejecutar la consulta
3. Cerrar el cursor y la conexión

Abrir o establecer la conexión

Lo primero que tendremos que hacer es realizar una conexión con el servidor de la base de datos. Esto se hace mediante la función **connect**, cuyos parámetros no están estandarizados y dependen de la base de datos a la que estemos conectándonos.

en el caso de MySQLdb, **connect** toma como parámetros la máquina en la que corre el servidor (host), el puerto (port), nombre de usuario con el que autenticarse (user), contraseña (password) y base de datos a la que conectarnos de entre las que se encuentran en nuestro SGBD (db).

La función connect devuelve un objeto de tipo Connection que representa la conexión con el servidor.

```
import MySQLdb
try:
    miConexion = MySQLdb.connect(host='localhost', user=
'root', passwd='', db='BBDD')
except:
    print("Error de conexión a la Base de Datos")
```

Crear un cursor

Las distintas operaciones que podemos realizar con la base de datos se realizan a través de un objeto **Cursor**. Para crear este objeto se utiliza el método **cursor()** del objeto Connection:

```
cur = miConexion.cursor()
```

Ejecutar consultas

Las operaciones se ejecutan a través del método **execute** de Cursor, pasando como parámetro una cadena con el código SQL a ejecutar.aa

```
cur.execute( "INSERT INTO empleados VALUES('12345678- A',
'Manuel Gil', 'Contabilidad')")
```

Si nuestra base de datos soporta transacciones, si estas están activadas, y si la característica de auto-commit está desactivada, será necesario llamar al método **commit** de la conexión para que se lleven a cabo las operaciones definidas en la transacción.

Si en estas circunstancias utilizáramos una herramienta externa para comprobar el contenido de nuestra base de datos sin hacer primero el commit nos encontraríamos entonces con una base de datos vacía.

Si comprobáramos el contenido de la base de datos desde Python, sin cerrar el cursor ni la conexión, recibiríamos el resultado del contexto de la transacción, por lo que parecería que se han llevado a cabo los cambios, aunque no es así, y los cambios sólo se aplican, como comentamos, al llamar a **commit**.

```
miConexion.commit()
```

Para realizar consultas a la base de datos también se utiliza **execute**

```
cur.execute( "SELECT nombre, apellido FROM usuarios" )
```

Para consultar las tuplas resultantes de la sentencia SQL se puede llamar a los métodos de Cursor ***fetchone***, ***fetchmany*** o ***fetchall*** o usar el objeto Cursor como un iterador.

```
cur.execute( "SELECT nombre, apellido FROM usuarios" )
for tupla in cur.fetchall():
      print tupla
```

o

```
cur.execute( "SELECT nombre, apellido FROM usuarios" )
for tupla in cur:
      print tupla
```

El método ***fetchone()*** devuelve la siguiente tupla del conjunto resultado o None cuando no existen más tuplas, ***fetchmany()*** devuelve el número de tuplas indicado por el entero pasado como parámetro o bien el número indicado por el atributo Cursor.arraysize si no se pasa ningún parámetro (Cursor.arraysize vale 1 por defecto) y ***fetchall()*** devuelve un objeto iterable con todas las tuplas.

Cerrar el cursor y la conexión

Por último, al final del programa se debe cerrar el cursor y la conexión con el método **close().**

```
cur.close()
miConexion.close()
```