

Contenido

Introducción	2
Preparando la herramienta	2
Archivo de conexión en Python	3
Preparación de SGBD	4
Sentencias SQL para un CRUD	4
Referencias	5

Introducción

En el presente documento, trabajaremos para poder conectar un sistema realizado en Python a una base de datos relacional, específicamente MySQL.

Comenzaremos por preparar la herramienta Visual Studio Code y rápidamente en unas pocas líneas de código ya estaremos en condiciones de hacer consultas a la base de datos, traer registros de sus tablas y poder también persistir nuevos datos desde la consola de Python.

Preparando la herramienta

Debemos saber que para poder conectar un desarrollo realizado en Python con una base de datos, debemos tener en nuestra herramienta de trabajo, Visual Studio Code, puede ser otra claro, pero en este módulo siempre trabajaremos los ejemplos con este editor de código.

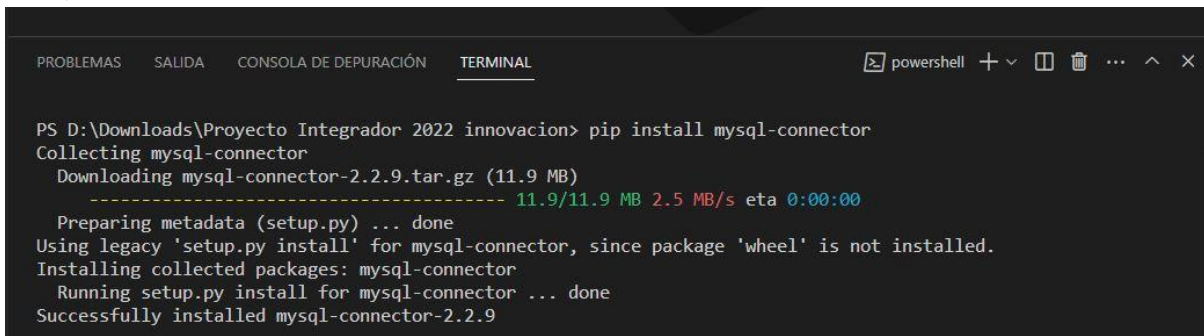
En VS Code, debemos instalar si no tenemos, el módulo PIP, que es la forma por la que instalaremos todos los paquetes necesarios para trabajar en Python, incluido el que utilizaremos para conectarnos con la base de datos. En referencias, dejo link para instalarlo.

Una vez instalado el comando PIP, debemos ir a Terminal->New terminal y escribir el siguiente comando:

```
pip install mysql-connector
```

Si ya lo tienen instalado y no lo sabían, les va a dejar un mensaje, y con la posibilidad de actualizarlo si hubiere una nueva versión.

Sino, lo va a instalar:



```
PS D:\Downloads\Proyecto Integrador 2022 innovacion> pip install mysql-connector
Collecting mysql-connector
  Downloading mysql-connector-2.2.9.tar.gz (11.9 MB)
    ----- 11.9/11.9 MB 2.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Using legacy 'setup.py install' for mysql-connector, since package 'wheel' is not installed.
Installing collected packages: mysql-connector
  Running setup.py install for mysql-connector ... done
Successfully installed mysql-connector-2.2.9
```

Con esto realizado, ya estamos en condiciones de conectar una base de datos relacional como MySQL con nuestro Python vía Visual Studio Code.

Archivo de conexión en Python

Para conectar una base de datos desde Python, lo primero que debemos hacer es abrir/crear un nuevo archivo de Python:

Archivo-> Nuevo archivo y colocamos el nombre y la extensión .py, que hace referencia a archivos de Python.

Supongamos que hemos creado el siguiente archivo: Prueba_Conexion_BaseDeDatos.py

En el mismo, lo primero que vamos a escribir es:

```
import mysql.connector
```

Y luego podemos crear un bloque con lo siguiente:

```
try:
    conexion = mysql.connector.connect(
        host = 'localhost',
        port = 3306,
        user = 'root',
        password = '**Aquí va una contraseña**',
        db='DB_PRUEBA_KEVIN'
    )
```

Los cinco atributos que se ven en la imagen son OBLIGATORIOS, y representan:

Host: es la “url” por la que vamos a conectarnos al SGBD, y digo entre comillas porque vamos a conectarnos, generalmente en este módulo, de manera local, a una base de datos que tengamos en nuestra computadora.

Port: es el puerto por el que autorizamos a VS Code a ingresar a nuestro SGBD.

User y password son las credenciales por las que el SGBD nos va a autorizar el ingreso, o no.

Db: es el nombre de la base de datos a la que nos queremos conectar.

Todos los valores de estos atributos deben coincidir exactamente con los que hayamos definido en el SGBD.

Si todos los datos son correctos y se puede conectar, dejaremos un mensaje para que se confirme:

```
if conexion.is_connected():
    print("LA CONEXION FUE EXITOSA")
```

Como vemos en la primera imagen, el bloque de código comienza con un try, ya que debemos asegurarnos que todo funcione correctamente. Si no lo hace, saldrá por el camino del except:

```
except:
    print("NO SE PUDO CONETAR A LA BASE DE DATOS")
```

Y por último, como una buena práctica programática definimos una última sentencia a nuestro bloque de conexión:

```
finally:
    if conexion.is_connected():
        conexion.close()
        print("LA CONEXION FUE CERRADA")
```

Es muy importante cerrar la conexión que abrimos al inicio, para habilitarla a otros usuarios que la requieran, o incluso nuestro mismo proyecto, en otra parte del mismo.

Con todo esto, podremos establecer una conexión con un SGBD y realizar consultas y persistir datos desde nuestro proyecto en Python.

Desde la parte que fuera del proyecto, podemos importar el bloque que acabamos de crear y utilizarlo donde y cuando necesitemos.

Preparación del SGBD

Tal como definimos en el apartado anterior, los datos de los atributos del módulo de conexión, deben coincidir con los del SGBD.

En ese sentido debemos tener en cuenta:

1. Tener instalado MySQL y el SGBD Workbench, en referencias dejo link de instalación e instructivo.
2. Conocer los datos de puerto, user y password que definimos en la instalación del SGBD.
3. Tener creada la base de datos con cual vamos a trabajar.

Sentencias SQL para un CRUD

Para realizar una acción que se relacione con una base de datos, debemos:

1. Crear una función con el nombre de la acción que vayamos a realizar, ejemplo: Consultar datos, listado de datos, actualizar datos, etc.

Ejemplo:

```
def ListarDatos(self):
```

Definimos una función y comenzamos a darle formato.

2. Consultamos si la conexión está abierta, mediante nuestro bloque de conexión ya definido:

```
if self.conexion.is_connected():
    try:
        cursor = self.conexion.cursor()
        sentenciaSQL = "SELECT * from interprete"
        cursor.execute(sentenciaSQL)
        resultados = cursor.fetchall()
        self.conexion.close()
        return resultados

    except mysql.connector.Error as descripcionError:
```

```
print("¡No se conectó!",descripcionError)
```

Vemos diferentes atributos, como cursor (función nativa del paquete mysql connector)

sentenciaSQL: en ella vamos a guardar como string la sentencia SQL que se va a ejecutar en MySQL Workbench.

Resultados: en ella vamos a guardar lo que regrese de la base de datos.

Y el return.

Además, agregamos un bloque que controle la posibilidad de que la conexión no esté abierta o haya algún error, mediante el bloque except.

Con esta estructura pueden crear funciones para las cuatro operaciones del CRUD

C: create (crear)/INSERT(insertar).

R: read (leer/seleccionar).

U: update (actualizar).

D: delete (eliminar).

Dejo ejemplos de cada uno:

CREATE:

```
def crearEntidad(self):
    if self.conexion.is_connected():
        try:
            cursor = self.conexion.cursor()
            sentenciaSQL = "CREATE TABLE Personas (DNI int not null,
Nombre varchar(30) not null, Apellido varchar(30) not null,
Telefono int )"
            cursor.execute(sentenciaSQL)
            self.conexion.close()
            return

        except mysql.connector.Error as descripcionError:
            print("¡No se conectó!",descripcionError)
```

INSERT:

```
def InsertarGenero(self,nombre):
    if self.conexion.is_connected():
        try:
            cursor = self.conexion.cursor()
            sentenciaSQL = "INSERT into genero values(null,%s)"

            data = (nombre,)

            cursor.execute(sentenciaSQL,data)
```

```

        self.conexion.commit()
        self.conexion.close()
        print("Género insertado correctamente")

    except mysql.connector.Error as descripcionError:
        print("¡No se conectó!",descripcionError)

```

READ:

```

def ListarDiscografica(self):
    if self.conexion.is_connected():
        try:
            cursor = self.conexion.cursor()
            sentenciaSQL = "SELECT * from discografica"
            cursor.execute(sentenciaSQL)
            resultados = cursor.fetchall()
            self.conexion.close()
            return resultados
        except mysql.connector.Error as descripcionError:
            print("¡No se conectó!",descripcionError)

```

UPDATE

```

def ActualizarPersona(self):
    if self.conexion.is_connected():
        try:
            cursor = self.conexion.cursor()
            sentenciaSQL = "UPDATE Persona set Nombre values('Kevin')
where DNI = 34555888"
            cursor.execute(sentenciaSQL)
            self.conexion.close()
            return resultados
        except mysql.connector.Error as descripcionError:
            print("¡No se conectó!",descripcionError)

```

DELETE

```

def EliminarPersona(self):
    if self.conexion.is_connected():
        try:
            cursor = self.conexion.cursor()
            sentenciaSQL = "DELETE from Personas where DNI = 34555888"
            cursor.execute(sentenciaSQL)
            self.conexion.close()

```

```
        return

    except mysql.connector.Error as descripcionError:
        print("¡No se conectó!",descripcionError)
```

Referencias

Instalador de PIP

[https://docs.python.org/es/3.9/library/ensurepip.html#:~:text=De%20forma%20predeterminada%2C%20pip%20se,hay%20ning%C3%BAn%20entorno%20virtual%20activo\).](https://docs.python.org/es/3.9/library/ensurepip.html#:~:text=De%20forma%20predeterminada%2C%20pip%20se,hay%20ning%C3%BAn%20entorno%20virtual%20activo).)

Instalador de MySQL y Workbench

<https://dev.mysql.com/downloads/workbench/>

Instructivo de instalación MySQL y Workbench

<https://docs.google.com/presentation/d/1iAmBc6G8ql0bBvi3wcjOxLRK6a4a1Nt/edit?usp=sharing&ouid=106638896470969516678&rtpof=true&sd=true>