

Introducción

Tabla de contenido

- [1 Introducción](#)
- [2 Descargar el código fuente](#)
- [3 Requisitos](#)
- [4 Instalar módulo PyMySQL](#)
- [5 Crear base de datos en MySQL](#)
- [6 Primera conexión](#)
- [7 Insertar datos en MySQL desde Python 3](#)
- [8 Consultar datos de MySQL con Python 3](#)
- [9 Consultar datos de MySQL con Python 3 usando WHERE](#)
- [10 Editar filas de MySQL con Python 3](#)
- [11 Eliminar filas o registros de MySQL con Python 3](#)
- [12 Conclusión](#)

Para conectar MySQL y Python 3 vamos a utilizar el paquete llamado **PyMySQL**..

Descargar el código fuente

Si lo deseas, puedes descargar todos los archivos que contienen el código fuente para conectar Python con MySQL.

Requisitos

- [Python versión 3 y pip](#)
- MySQL (tutorial con XAMPP [aquí](#))

Instalar módulo PyMySQL

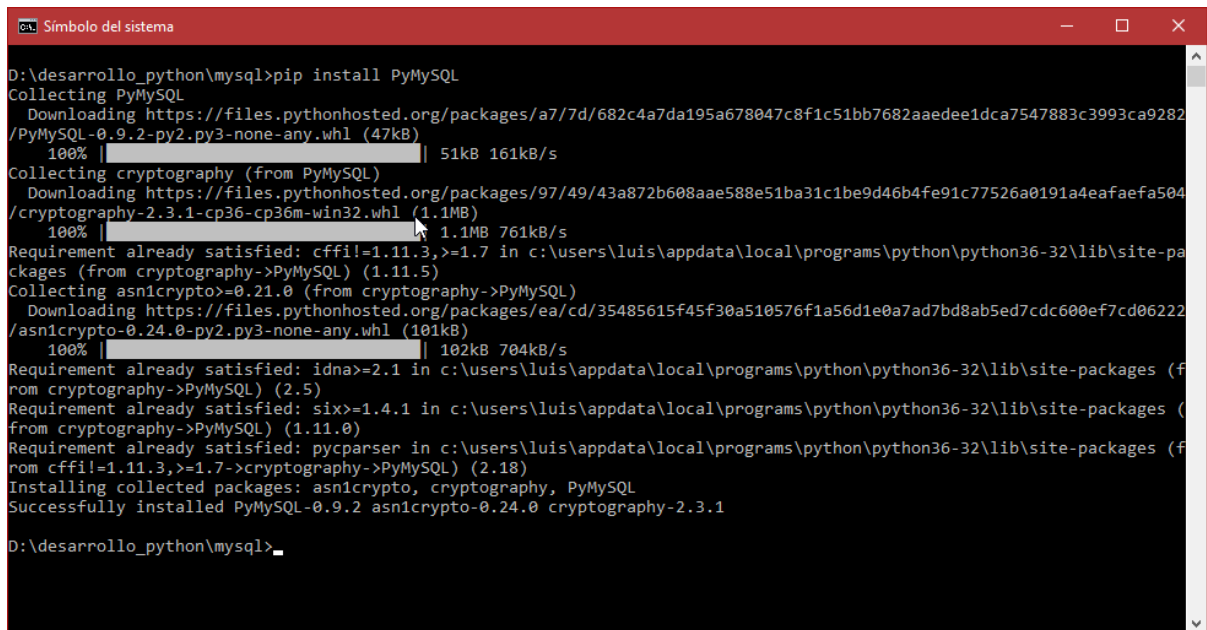
Para instalar lo que nos permitirá conectarnos con MySQL desde Python 3 ejecutamos el siguiente comando:

```
pip install PyMySQL
```

En caso de que ese comando no llegara a funcionar, podemos usar:

```
python -m pip install PyMySQL
```

Si incluso así sigue sin funcionar, recuerda [configurar e instalar Python](#). Cuando instales correctamente la librería, se mostrará algo así:



```
D:\desarrollo_python\mysql>pip install PyMySQL
Collecting PyMySQL
  Downloading https://files.pythonhosted.org/packages/a7/7d/682c4a7da195a678047c8f1c51bb7682aaedee1dca7547883c3993ca9282/PyMySQL-0.9.2-py2.py3-none-any.whl (47kB)
    100% |#####| 51kB 161kB/s
Collecting cryptography (from PyMySQL)
  Downloading https://files.pythonhosted.org/packages/97/49/43a872b608aae588e51ba31c1be9d46b4fe91c77526a0191a4eafaefa504/cryptography-2.3.1-cp36-cp36m-win32.whl (1.1MB)
    100% |#####| 1.1MB 761kB/s
Requirement already satisfied: cffi!=1.11.3,>=1.7 in c:\users\luis\appdata\local\programs\python\python36-32\lib\site-packages (from cryptography->PyMySQL) (1.11.5)
Collecting asn1crypto>=0.21.0 (from cryptography->PyMySQL)
  Downloading https://files.pythonhosted.org/packages/ea/cd/35485615f45f30a510576f1a56d1e0a7ad7bd8ab5ed7cdc600ef7cd06222/asn1crypto-0.24.0-py2.py3-none-any.whl (101kB)
    100% |#####| 102kB 704kB/s
Requirement already satisfied: idna>=2.1 in c:\users\luis\appdata\local\programs\python\python36-32\lib\site-packages (from cryptography->PyMySQL) (2.5)
Requirement already satisfied: six>=1.4.1 in c:\users\luis\appdata\local\programs\python\python36-32\lib\site-packages (from cryptography->PyMySQL) (1.11.0)
Requirement already satisfied: pyparsing in c:\users\luis\appdata\local\programs\python\python36-32\lib\site-packages (from cffi!=1.11.3,>=1.7->cryptography->PyMySQL) (2.18)
Installing collected packages: asn1crypto, cryptography, PyMySQL
Successfully installed PyMySQL-0.9.2 asn1crypto-0.24.0 cryptography-2.3.1

D:\desarrollo_python\mysql>
```

Instalación de PyMySQL con pip

Crear base de datos en MySQL

Para poder trabajar con Python 3 y MySQL debemos tener una base de datos en éste último. Para ello nos conectamos a nuestro servidor de MySQL y creamos una base de datos. Durante el tutorial trabajaremos con una base de datos de películas que tendrá una tabla llamada también películas, y sólo eso.

En mi caso voy a utilizar la [CLI de MySQL](#). La definición de la tabla y la creación de la base de datos queda así:


```
CREATE DATABASE peliculas;

USE peliculas;

CREATE TABLE IF NOT EXISTS peliculas(
    id BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
    titulo VARCHAR(255) NOT NULL,
```

```
        anio SMALLINT NOT NULL,  
        PRIMARY KEY(id)  
    );
```

Cuando lo ejecuto en la CLI de MySQL se ve así:



```
D:\desarrollo_php\mysql\bin\mysql.exe
MariaDB [(none)]> CREATE DATABASE peliculas;
Query OK, 1 row affected (0.01 sec)

MariaDB [(none)]> USE peliculas;
Database changed
MariaDB [peliculas]> CREATE TABLE IF NOT EXISTS peliculas(
    -> id BIGINT UNSIGNED AUTO_INCREMENT NOT NULL,
    -> titulo VARCHAR(255) NOT NULL,
    -> anio SMALLINT NOT NULL,
    -> PRIMARY KEY(id)
    -> );
Query OK, 0 rows affected (0.27 sec)

MariaDB [peliculas]> show tables;
+-----+
| Tables_in_peliculas |
+-----+
| peliculas            |
+-----+
1 row in set (0.02 sec)

MariaDB [peliculas]> describe peliculas;
+-----+-----+-----+-----+-----+-----+
| Field | Type                | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id     | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| titulo | varchar(255)        | NO   |     | NULL    |                |
| anio   | smallint(6)         | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.06 sec)
```

Puedes usar phpmyadmin o la interfaz de MySQL workbench, tú decide. Sólo ten en cuenta que **la base de datos y la tabla deben estar creadas y funcionando.**

Primera conexión

Antes de hacer el **CRUD** vamos a ver si nos podemos conectar a la base de datos. Para ello ponemos este fragmento de código en un **try/except**:

```
import pymysql
try:
    conexion = pymysql.connect(host='localhost',
                               user='root',
                               password='',
                               db='películas')
    print("Conexión correcta")
except (pymysql.err.OperationalError, pymysql.err.InternalError) as e:
    print("Ocurrió un error al conectar: ", e)
```

Aquí hay unos parámetros que debemos tener en cuenta.

- **host:** el host en donde nuestro servidor MySQL escucha. Normalmente es localhost o 127.0.0.1 pero igualmente podría ser otro, basta con poner la ip
- **user:** el usuario que puede administrar la base de datos
- **password:** la contraseña del usuario. Si está vacía dejamos las comillas vacías
- **db:** el nombre de la base de datos a la que intentamos conectarnos

Si todo va bien, al ejecutar el script debe mostrar el mensaje:

Conexión correcta

En caso de que no, se mostrarán errores. Aquí abajo dejo los más comunes.

```
D:\desarrollo_python\mysql>python conexion.py
Ocurrió un error al conectar: (2003, "Can't connect to MySQL server on 'localhosts' ([Errno 11001] getaddrinfo failed)")

D:\desarrollo_python\mysql>python conexion.py
Ocurrió un error al conectar: (1044, "Access denied for user '@localhost' to database 'películas'")

D:\desarrollo_python\mysql>python conexion.py
Ocurrió un error al conectar: (1049, "Unknown database 'peliculass'")

D:\desarrollo_python\mysql>
```

Errores que podemos encontrar al conectar Python 3 con MySQL

En el primer caso es porque puse localhosts en lugar de localhost. Esto también pasará si ponemos una IP en donde el servidor no esté escuchando.

En el segundo caso puse un usuario que no está registrado o que no tiene permiso para esa base de datos.

En el tercer caso puse el nombre de una base de datos que no existe.

Continuemos entonces.

Insertar datos en MySQL desde Python 3

Antes de continuar cabe mencionar que los ejemplos aquí son 100 % seguros. Es decir, nuestras consultas prevendrán las inyecciones SQL.

Para insertar una película hacemos esto:

```
import pymysql
try:
    conexion = pymysql.connect(host='localhost',
                               user='root',
                               password='',
                               db='peliculas')

    try:
        with conexion.cursor() as cursor:
            consulta = "INSERT INTO peliculas(titulo, anio) VALUES (%s, %s);"
            #Podemos llamar muchas veces a .execute con datos distintos
            cursor.execute(consulta, ("Volver al futuro 1", 1985))
            cursor.execute(consulta, ("Ready Player One", 2018))
            cursor.execute(consulta, ("It", 2017))
            cursor.execute(consulta, ("Pulp Fiction", 1994))
        conexion.commit()
    finally:
        conexion.close()
except (pymysql.err.OperationalError, pymysql.err.InternalError) as e:
    print("Ocurrió un error al conectar: ", e)
```

Como vemos, necesitamos tener la conexión. Y luego usamos un cursor para ejecutar nuestra consulta sobre la misma.

Es importante notar que usamos **%s** en lugar del valor real, y luego ponemos los verdaderos datos en la llamada a **.execute** del cursor. Esto es para que sea una consulta segura y las variables sean escapadas en caso de que un usuario malicioso quisiera aprovechar esa vulnerabilidad.

Más abajo hacemos un **commit**. Esto es para guardar los cambios que hicimos a la base de datos, **si no llamamos a este método ninguno de nuestros cambios se reflejará**.

Todo eso lo encerramos en un bloque try/finally (aparte del que pertenece a la conexión) y siempre cerramos la conexión. Así, si hay un problema con la inserción, la conexión no quedará abierta.

Consultar datos de MySQL con Python 3

Ya vimos cómo insertar, ahora vamos a ver cómo listar o consultar. Para ello usamos el siguiente código:

```
import pymysql
try:
    conexion = pymysql.connect(host='localhost',
                               user='root',
                               password='',
                               db='películas')

    try:
        with conexion.cursor() as cursor:
            # En este caso no necesitamos limpiar ningún dato
            cursor.execute("SELECT id, titulo, anio FROM películas;")

            # Con fetchall traemos todas las filas
            peliculas = cursor.fetchall()

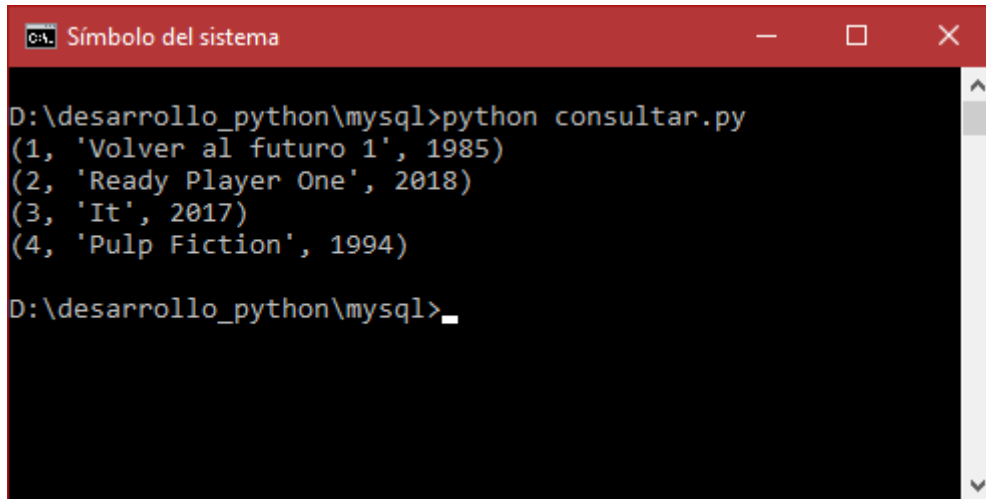
            # Recorrer e imprimir
            for pelicula in peliculas:
                print(pelicula)

    finally:
        conexion.close()

except (pymysql.err.OperationalError, pymysql.err.InternalError) as e:
    print("Ocurrió un error al conectar: ", e)
```

Siempre usamos el cursor. Ahora llamamos al método **fetchall** que traer todas las filas. No deberíamos usarlo para grandes datos, pues si hay 1000 filas traerá todas.

Si quisiéramos traer una por una podríamos llamar al método **fetchone**. En fin, al ejecutar el script sale esto:



```
Símbolo del sistema
D:\desarrollo_python\mysql>python consultar.py
(1, 'Volver al futuro 1', 1985)
(2, 'Ready Player One', 2018)
(3, 'It', 2017)
(4, 'Pulp Fiction', 1994)
D:\desarrollo_python\mysql>_
```

Consultar registros de MySQL con Python 3

Consultar datos de MySQL con Python 3 usando WHERE

Para hacer una consulta con WHERE tenemos que limpiar los datos para prevenir inyecciones. Aquí un ejemplo en donde consultamos las películas cuyo año sea mayor que el 2000.

```
import pymysql
try:
    conexion = pymysql.connect(host='localhost',
                               user='root',
                               password='',
                               db='peliculas')

    try:
        with conexion.cursor() as cursor:

            consulta = "SELECT id, titulo, anio FROM peliculas WHERE anio > %s;"
            cursor.execute(consulta, (2000))

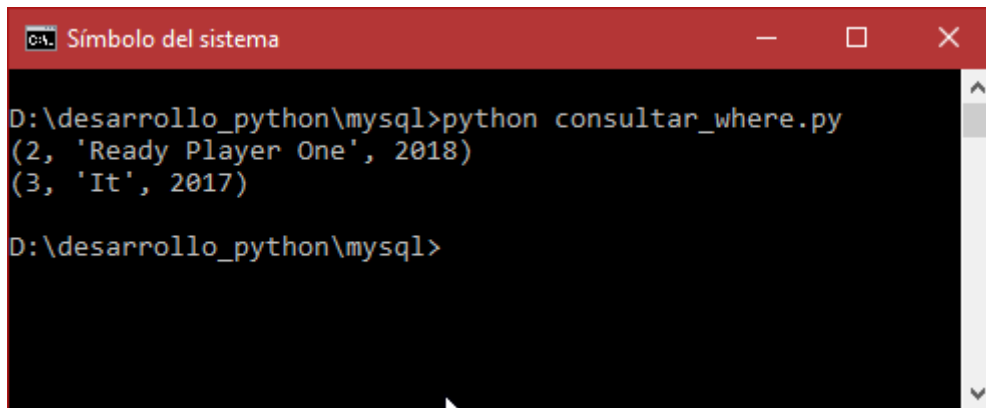
            # Con fetchall traemos todas las filas
            peliculas = cursor.fetchall()

            # Recorrer e imprimir
            for pelicula in peliculas:
                print(pelicula)

    finally:
        conexion.close()

except (pymysql.err.OperationalError, pymysql.err.InternalError) as e:
    print("Ocurrió un error al conectar: ", e)
```

Lo que hay que notar es que siempre usamos los placeholders %s en lugar de concatenar valores. Al ejecutar eso, este es el resultado:



```
Símbolo del sistema
D:\desarrollo_python\mysql>python consultar_where.py
(2, 'Ready Player One', 2018)
(3, 'It', 2017)
D:\desarrollo_python\mysql>
```

Consultar registros de MySQL con Python 3 usando WHERE

Editar filas de MySQL con Python 3

Veamos ahora cómo hacer un update o una edición de datos. Igualmente usaremos el cursor, y prepararemos la sentencia para que la misma sea segura.

Lo que hace el siguiente código es cambiar el nombre a “Ready Player One: comienza el juego” en donde la película tenga el id 2.

```
import pymysql
try:
    conexion = pymysql.connect(host='localhost',
                               user='root',
                               password='',
                               db='peliculas')

    try:
        with conexion.cursor() as cursor:

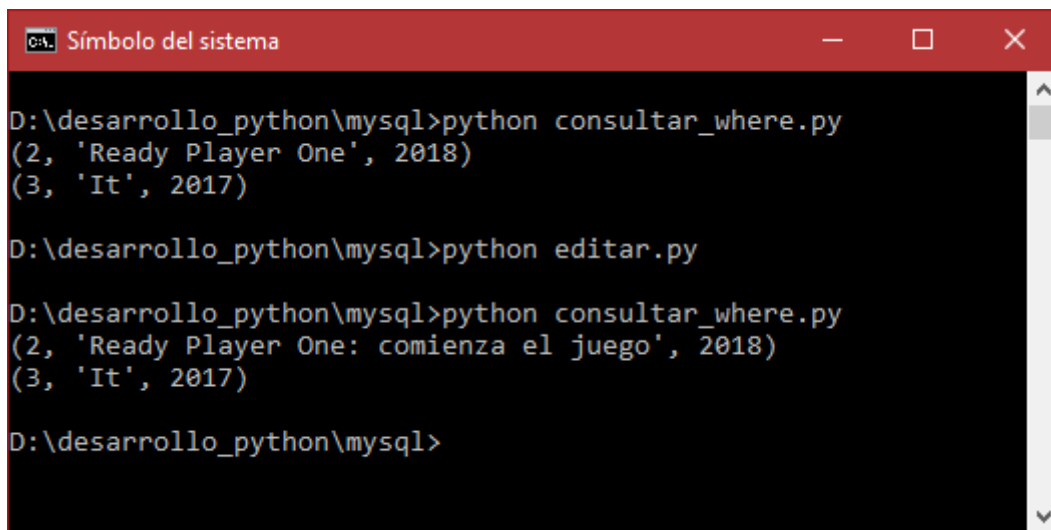
            consulta = "UPDATE peliculas SET titulo = %s WHERE id = %s;"
            nuevo_titulo = "Ready Player One: comienza el juego"
            id_editar = 2
            cursor.execute(consulta, (nuevo_titulo, id_editar))

            # No olvidemos hacer commit cuando hacemos un cambio a la BD
            conexion.commit()
    finally:
        conexion.close()
```

```
except (pymysql.err.OperationalError, pymysql.err.InternalError) as e:  
    print("Ocurrió un error al conectar: ", e)
```

Siempre tenemos que usar placeholders, nunca concatenar. Y cuando hacemos una operación que modifique la base de datos (insert, update, delete) tenemos que hacer commit para reflejar los cambios.

En la imagen podemos ver que primero listamos los datos y el título de la película es el antiguo. Luego ejecutamos el script para actualizar, y al volver a listar observamos que la película tiene un nuevo nombre:



```
Símbolo del sistema  
D:\desarrollo_python\mysql>python consultar_where.py  
(2, 'Ready Player One', 2018)  
(3, 'It', 2017)  
  
D:\desarrollo_python\mysql>python editar.py  
  
D:\desarrollo_python\mysql>python consultar_where.py  
(2, 'Ready Player One: comienza el juego', 2018)  
(3, 'It', 2017)  
  
D:\desarrollo_python\mysql>
```

Edición o actualización de datos en MySQL con Python 3

Eliminar filas o registros de MySQL con Python 3

Para terminar con este tutorial vamos a ver cómo eliminar datos. Es exactamente igual que cuando hacemos un update o insert.

El siguiente fragmento de código elimina las películas que se hayan estrenado antes del año 2000:

```
import pymysql
try:
    conexion = pymysql.connect(host='localhost',
                               user='root',
                               password='',
                               db='peliculas')

    try:
        with conexion.cursor() as cursor:

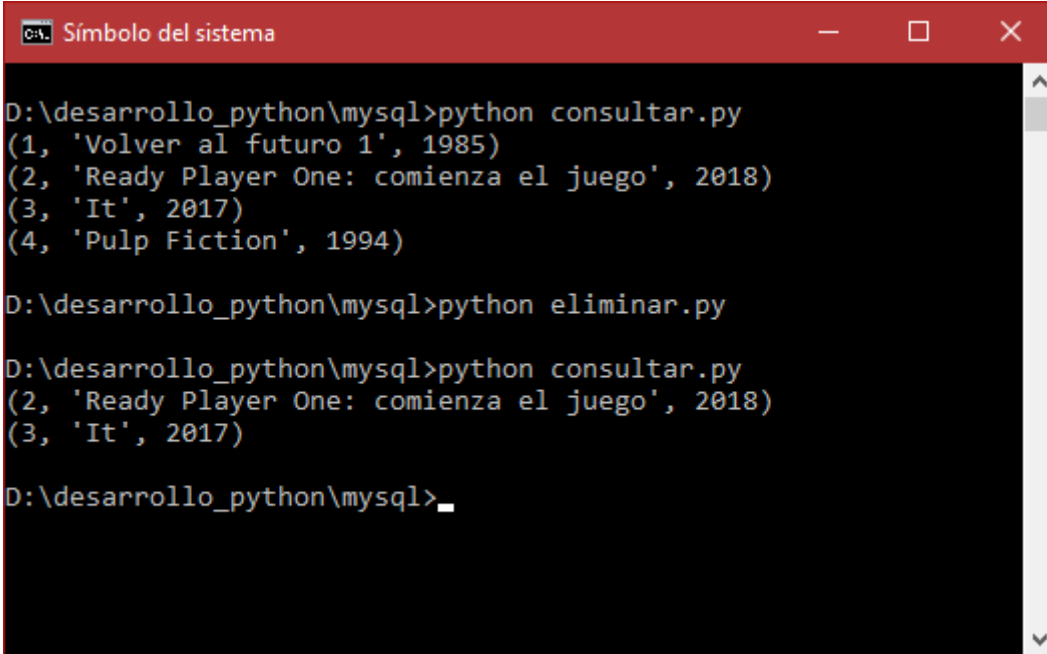
            consulta = "DELETE FROM peliculas WHERE anio < %s;"
            anio = 2000
            cursor.execute(consulta, (anio))

            # No olvidemos hacer commit cuando hacemos un cambio a la BD
            conexion.commit()
    finally:
        conexion.close()

except (pymysql.err.OperationalError, pymysql.err.InternalError) as e:
    print("Ocurrió un error al conectar: ", e)
```

Para comprobar que realmente está eliminando primero listamos las películas, luego ejecutamos la eliminación y finalmente volvemos a listar.

Al final sólo quedan 2 películas.



```
D:\desarrollo_python\mysql>python consultar.py
(1, 'Volver al futuro 1', 1985)
(2, 'Ready Player One: comienza el juego', 2018)
(3, 'It', 2017)
(4, 'Pulp Fiction', 1994)

D:\desarrollo_python\mysql>python eliminar.py

D:\desarrollo_python\mysql>python consultar.py
(2, 'Ready Player One: comienza el juego', 2018)
(3, 'It', 2017)

D:\desarrollo_python\mysql>_
```

Eliminación de filas, datos o registros de MySQL con Python 3