

Conexión Python MySQL



Ayúdame a crear un programa de Python para conectarse a una MySQL



Para conectar Python con una base de datos MySQL, puedes utilizar el paquete `mysql-connector-python`. Aquí te dejo un ejemplo de cómo crear un programa simple que se conecta a una base de datos MySQL.

Paso 1: Instalar el conector MySQL para Python

Primero, necesitas instalar el paquete:

```
bash
```

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

Paso 2: Crear el programa en Python

Aquí tienes un ejemplo básico de un script en Python para conectarse a una base de datos MySQL:

```
python
```

```
import mysql.connector
from mysql.connector import Error

def connect_to_mysql():
    try:
        # Establecer la conexión con la base de datos
        connection = mysql.connector.connect(
            host='tu_host', # por ejemplo, 'localhost'
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )

        if connection.is_connected():
            db_Info = connection.get_server_info()
            print("Conectado a MySQL Server versión ", db_Info)
            cursor = connection.cursor()
            cursor.execute("SELECT DATABASE();")
            record = cursor.fetchone()
            print("Conectado a la base de datos ", record)

    except Error as e:
        print("Error al conectar con MySQL", e)

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")
```

```
# Llamar a la función para conectarse a la base de datos
connect_to_mysql()
```

Paso 3: Configurar el script

- **host:** Especifica la dirección del servidor de la base de datos. Si estás trabajando en tu máquina local, usa `'localhost'`.
- **database:** Nombre de la base de datos a la que te quieres conectar.
- **user:** Tu nombre de usuario de MySQL.
- **password:** La contraseña asociada con ese usuario.

Ejecución

Guarda el script en un archivo `.py` y ejecútalo. Si los detalles de la conexión son correctos, deberías ver un mensaje confirmando que te has conectado exitosamente a MySQL, seguido de información sobre la versión del servidor y el nombre de la base de datos conectada.

Este script básico se puede expandir para realizar diversas operaciones con la base de datos como consultas, inserciones, actualizaciones, etc.



Ahora quiero consultar los registros de la tabla `curso` y que los convierta en formato DataFrame y los exporte a un Excel



Para consultar registros de la tabla `curso`, convertirlos a un DataFrame y exportarlos a un archivo Excel, puedes seguir los pasos a continuación.

Paso 1: Instalar los paquetes necesarios

Asegúrate de tener instalados los paquetes `pandas` y `openpyxl`, ya que los usaremos para manejar DataFrames y exportar a Excel.

```
bash
```

```
pip install pandas openpyxl
```

Paso 2: Modificar el script

Vamos a modificar el script para realizar la consulta a la tabla `curso`, convertir los resultados en un DataFrame y exportarlos a un archivo Excel.

```
python
```

```
import mysql.connector
from mysql.connector import Error
import pandas as pd
```

```
def export_curso_to_excel():
    try:
        # Establecer la conexión con la base de datos
        connection = mysql.connector.connect(
            host='tu_host', # por ejemplo, 'localhost'
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )

        if connection.is_connected():
            print("Conectado a la base de datos")
            cursor = connection.cursor()

            # Ejecutar la consulta para obtener los registros de la tabla curso
            query = "SELECT * FROM curso"
            cursor.execute(query)

            # Obtener los nombres de las columnas
            columns = [col[0] for col in cursor.description]

            # Convertir los registros en un DataFrame
            records = cursor.fetchall()
            df = pd.DataFrame(records, columns=columns)

            # Exportar el DataFrame a un archivo Excel
            df.to_excel("curso_exportado.xlsx", index=False)
            print("Los datos han sido exportados exitosamente a 'curso_exportado.xlsx'")

    except Error as e:
        print("Error al conectar con MySQL", e)

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")

# Llamar a la función para exportar los datos
export_curso_to_excel()
```

Paso 3: Configurar y ejecutar el script

1. **Conexión a la base de datos:** Asegúrate de configurar los parámetros de conexión (`host`, `database`, `user`, `password`) según tus credenciales y detalles de la base de datos.
2. **Consulta:** El script ejecuta la consulta `SELECT * FROM curso`, que recupera todos los registros de la tabla `curso`.
3. **Conversión a DataFrame:** Los resultados se convierten en un DataFrame utilizando `pandas`.
4. **Exportación a Excel:** Finalmente, el DataFrame se exporta a un archivo Excel llamado `curso_exportado.xlsx` en el mismo directorio donde se ejecuta el script.

Ejecución

Guarda el script en un archivo `.py` y ejecútalo. Si todo está configurado correctamente, deberías ver un archivo `curso_exportado.xlsx` con los datos de la tabla `curso`.



Ahora quiero consultar los registros de la tabla profesor



Para consultar los registros de la tabla `profesor` y exportarlos a un archivo Excel, solo necesitas modificar el script anterior para hacer la consulta a la tabla `profesor` en lugar de la tabla `curso`. Aquí te dejo el script actualizado:

Paso 1: Asegúrate de tener los paquetes necesarios

Instala los paquetes `pandas` y `openpyxl` si no lo has hecho ya:

```
bash
```

```
pip install pandas openpyxl
```

Paso 2: Modificar el script para consultar la tabla `profesor`

```
python
```

```
import mysql.connector
from mysql.connector import Error
import pandas as pd

def export_profesor_to_excel():
    try:
        # Establecer la conexión con la base de datos
        connection = mysql.connector.connect(
            host='tu_host', # por ejemplo, 'localhost'
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )

        if connection.is_connected():
            print("Conectado a la base de datos")
            cursor = connection.cursor()

            # Ejecutar la consulta para obtener los registros de la tabla profesor
            query = "SELECT * FROM profesor"
            cursor.execute(query)

            # Obtener los nombres de las columnas
            columns = [col[0] for col in cursor.description]

            # Convertir los registros en un DataFrame
            records = cursor.fetchall()
            df = pd.DataFrame(records, columns=columns)

            # Exportar el DataFrame a un archivo Excel
            df.to_excel("profesor_exportado.xlsx", index=False)
            print("Los datos han sido exportados exitosamente a 'profesor_exportado.xlsx'")

        except Error as e:
            print("Error al conectar con MySQL", e)

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")
```

```
# Llamar a la función para exportar los datos
export_profesor_to_excel()
```

Paso 3: Configurar y ejecutar el script

1. **Conexión a la base de datos:** Asegúrate de configurar los parámetros de conexión (`host`, `database`, `user`, `password`) según tus credenciales y detalles de la base de datos.
2. **Consulta:** El script ejecuta la consulta `SELECT * FROM profesor`, que recupera todos los registros de la tabla `profesor`.
3. **Conversión a DataFrame:** Los resultados se convierten en un DataFrame utilizando `pandas`.
4. **Exportación a Excel:** El DataFrame se exporta a un archivo Excel llamado `profesor_exportado.xlsx` en el mismo directorio donde se ejecuta el script.

Ejecución

Guarda el script en un archivo `.py` y ejecútalo. Si todo está configurado correctamente, deberías ver un archivo `profesor_exportado.xlsx` con los datos de la tabla `profesor`.



el código solo para consultar la tabla profesor sin almacenar en excel



Aquí tienes el código en Python para consultar los registros de la tabla `profesor` y mostrarlos en la consola, sin exportarlos a un archivo Excel:

python

```
import mysql.connector
from mysql.connector import Error
import pandas as pd

def consultar_profesor():
    try:
        # Establecer la conexión con la base de datos
        connection = mysql.connector.connect(
            host='tu_host', # por ejemplo, 'localhost'
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )

        if connection.is_connected():
            print("Conectado a la base de datos")
            cursor = connection.cursor()

            # Ejecutar la consulta para obtener los registros de la tabla profesor
            query = "SELECT * FROM profesor"
            cursor.execute(query)

            # Obtener los nombres de las columnas
            columns = [col[0] for col in cursor.description]

            # Convertir los registros en un DataFrame
            records = cursor.fetchall()
            df = pd.DataFrame(records, columns=columns)
```

```
# Mostrar los datos en la consola
print(df)

except Error as e:
    print("Error al conectar con MySQL", e)

finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("Conexión a MySQL cerrada")

# Llamar a la función para consultar los datos
consultar_profesor()
```

Explicación:

1. **Conexión a la base de datos:** El script establece la conexión con tu base de datos MySQL utilizando los parámetros que proporcionas.
2. **Consulta:** Se ejecuta la consulta ``SELECT * FROM profesor`` para recuperar todos los registros de la tabla ``profesor``.
3. **Conversión a DataFrame:** Los resultados se almacenan en un DataFrame de ``pandas`` para una mejor visualización.
4. **Mostrar datos:** El DataFrame se imprime en la consola para que puedas ver los datos directamente.
5. **Cierre de conexión:** La conexión a la base de datos se cierra al final para liberar recursos.

Ejecución

Guarda el script en un archivo `.py`, ejecuta el archivo, y verás los datos de la tabla ``profesor`` impresos en la consola.



genera una consulta para sumar el campo nAsignaturas



Para sumar el campo ``nAsignaturas`` de la tabla ``profesor``, puedes usar una consulta SQL con la función ``SUM()``. A continuación, te muestro cómo podrías integrar esta consulta en un script de Python:

```
python

import mysql.connector
from mysql.connector import Error

def sumar_nasignaturas():
    try:
        # Establecer la conexión con la base de datos
        connection = mysql.connector.connect(
            host='tu_host', # por ejemplo, 'localhost'
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )
```

```

if connection.is_connected():
    print("Conectado a la base de datos")
    cursor = connection.cursor()

    # Ejecutar la consulta para sumar el campo nAsignaturas
    query = "SELECT SUM(nAsignaturas) FROM profesor"
    cursor.execute(query)

    # Obtener el resultado de la suma
    total_nasignaturas = cursor.fetchone()[0]

    # Mostrar el total en la consola
    print(f"Total de nAsignaturas: {total_nasignaturas}")

except Error as e:
    print("Error al conectar con MySQL", e)

finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("Conexión a MySQL cerrada")

# Llamar a la función para sumar el campo nAsignaturas
sumar_nasignaturas()

```

Explicación:

1. **Conexión a la base de datos:** El script se conecta a la base de datos utilizando las credenciales que proporcionas.
2. **Consulta:** La consulta ``SELECT SUM(nAsignaturas) FROM profesor`` suma todos los valores del campo ``nAsignaturas`` en la tabla ``profesor``.
3. **Obtener el resultado:** Se recupera el resultado de la suma y se almacena en la variable ``total_nasignaturas``.
4. **Mostrar el resultado:** El total se imprime en la consola.
5. **Cierre de conexión:** La conexión a la base de datos se cierra para liberar recursos.

Ejecución

Guarda este código en un archivo `.py` y ejecútalo. Verás el total de ``nAsignaturas`` en la consola.



por favor genera e código para insertar el curso con `idCurso=50`, `nombreDescriptivo="Estructura de Datos"` y `nAsignaturas=104`



Para insertar un nuevo curso en la tabla ``curso`` con los valores proporcionados (``idCurso=50``, ``nombreDescriptivo="Estructura de Datos"``, y ``nAsignaturas=104``), puedes usar el siguiente código en Python:

python

```

import mysql.connector
from mysql.connector import Error

```

```
def insertar_curso():
    try:
        # Establecer la conexión con la base de datos
        connection = mysql.connector.connect(
            host='tu_host', # por ejemplo, 'localhost'
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )

        if connection.is_connected():
            print("Conectado a la base de datos")
            cursor = connection.cursor()

            # Consulta para insertar el nuevo curso
            query = """
                INSERT INTO curso (idCurso, nombreDescriptivo, nAsignaturas)
                VALUES (%s, %s, %s)
            """
            values = (50, "Estructura de Datos", 104)

            # Ejecutar la inserción
            cursor.execute(query, values)
            connection.commit() # Asegurar que los cambios se guarden en la base de datos

            print("Curso insertado exitosamente")

        except Error as e:
            print("Error al conectar con MySQL o al insertar datos", e)

        finally:
            if connection.is_connected():
                cursor.close()
                connection.close()
                print("Conexión a MySQL cerrada")

    # Llamar a la función para insertar el curso
    insertar_curso()
```

Explicación:

1. **Conexión a la base de datos:** El script establece la conexión a tu base de datos MySQL utilizando las credenciales proporcionadas.
2. **Consulta de inserción:** La consulta `INSERT INTO curso (idCurso, nombreDescriptivo, nAsignaturas) VALUES (%s, %s, %s)` inserta un nuevo registro en la tabla `curso`. Los valores (`%s`) se reemplazan con los datos que has proporcionado: `50`, `"Estructura de Datos"`, y `104`.
3. **Ejecución y confirmación:** El comando `cursor.execute(query, values)` ejecuta la consulta con los valores proporcionados, y `connection.commit()` asegura que los cambios se guarden en la base de datos.
4. **Mensajes de estado:** Si la inserción es exitosa, se imprime un mensaje confirmando que el curso ha sido insertado. Si ocurre algún error, se captura y se muestra en la consola.
5. **Cierre de conexión:** Finalmente, la conexión a la base de datos se cierra para liberar recursos.

Ejecución

Guarda este código en un archivo `.py` y ejecútalo. Si todo está configurado correctamente, el curso con `idCurso=50`, `nombreDescriptivo="Estructura de Datos"`, y `nAsignaturas=104` será insertado en la tabla `curso`.



ahora genera el código para actualizar el registro idCurso=50 con nombreDescriptivo="Gestion de Proyectos de Software"



Aquí tienes el código en Python para actualizar el registro con `idCurso=50`, cambiando el valor de `nombreDescriptivo` a `"Gestión de Proyectos de Software"`:

python

```
import mysql.connector
from mysql.connector import Error

def actualizar_curso():
    try:
        # Establecer la conexión con la base de datos
        connection = mysql.connector.connect(
            host='tu_host', # por ejemplo, 'localhost'
            database='tu_base_de_datos',
            user='tu_usuario',
            password='tu_contraseña'
        )

        if connection.is_connected():
            print("Conectado a la base de datos")
            cursor = connection.cursor()

            # Consulta para actualizar el nombre descriptivo del curso
            query = """
                UPDATE curso
                SET nombreDescriptivo = %s
                WHERE idCurso = %s
            """
            values = ("Gestión de Proyectos de Software", 50)

            # Ejecutar la actualización
            cursor.execute(query, values)
            connection.commit() # Asegurar que los cambios se guarden en la base de datos

            print("Curso actualizado exitosamente")

    except Error as e:
        print("Error al conectar con MySQL o al actualizar datos", e)

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Conexión a MySQL cerrada")

# Llamar a la función para actualizar el curso
actualizar_curso()
```

Explicación:

1. **Conexión a la base de datos:** El script se conecta a tu base de datos MySQL utilizando las credenciales que proporcionas.
2. **Consulta de actualización:** La consulta `UPDATE curso SET nombreDescriptivo = %s WHERE idCurso = %s` actualiza el campo `nombreDescriptivo` del registro donde `idCurso = 50`. Los valores proporcionados son `"Gestión de Proyectos de Software"` para `nombreDescriptivo` y `50` para `idCurso`.

3. **Ejecución y confirmación:** El comando `cursor.execute(query, values)` ejecuta la actualización, y `connection.commit()` asegura que los cambios se guarden en la base de datos.
4. **Mensajes de estado:** Si la actualización es exitosa, se imprime un mensaje confirmando que el curso ha sido actualizado. Si ocurre algún error, se captura y se muestra en la consola.
5. **Cierre de conexión:** La conexión a la base de datos se cierra al final para liberar recursos.

Ejecución

Guarda este código en un archivo `.py` y ejecútalo. Si todo está configurado correctamente, el curso con `idCurso=50` tendrá su `nombreDescriptivo` actualizado a `"Gestión de Proyectos de Software"` en la tabla `curso`.

ChatGPT puede cometer errores. Considera verificar la información importante.