Código de Python

```
import mysql.connector

def conectar():
    conexion = mysql.connector.connect(
        host='localhost',
        user='root',
        password='',
        database='biblioteca'
    )
    return conexion

def consultar_libros_mas_populares():
    SQL = "SELECT titulo, calificacion FROM libros WHERE calificacion
>= 4 ORDER BY calificacion DESC"
    conexion = conectar()
    cursor = conexion.cursor()
    cursor.execute(SQL)
    resultado = cursor.fetchall()
    for libro in resultado:
        print(f"Título: {libro[0]}, Calificación: {libro[1]}")
    cursor.close()
    conexion.close()

def quitar_libro_menu():
    SQL = "SELECT id_libro, titulo FROM libros ORDER BY titulo"
    conexion = conectar()
    cursor = conexion.cursor()
    cursor.execute(SQL)
    libros = cursor.fetchall()
    if not libros:
        print("No hay libros para eliminar.")
        cursor.close()
        conexion.close()
        return

    print("Libros disponibles para eliminar:")
    for idx, libro in enumerate(libros, 1):
        print(f"{idx}: {libro[1]}")

    while True:
        try:
```

```
            opcion = int(input("Elige el número del libro a eliminar:
"))
            if 1 <= opcion <= len(libros):
                id_libro = libros[opcion-1][0]
                break
            else:
                print("Opción inválida, intenta de nuevo.")
        except ValueError:
            print("Ingresa un número válido.")

    SQL_delete = "DELETE FROM libros WHERE id_libro = %s"
    cursor.execute(SQL_delete, (id_libro,))
    conexion.commit()
    print(f'Libro "{libros[opcion-1][1]}" se elimino del sistema.')
    cursor.close()
    conexion.close()

def agregar_libro(titulo, autor, genero, publicacion,
cantidad_disponible, calificacion):
    SQL = "INSERT INTO libros (titulo, autor, genero, publicacion,
cantidad_disponible, calificacion) VALUES (%s, %s, %s, %s, %s, %s)"
    valores = (titulo, autor, genero, publicacion, cantidad_disponible,
calificacion)
    conexion = conectar()
    cursor = conexion.cursor()
    cursor.execute(SQL, valores)
    print(cursor.statement)
    conexion.commit()
    cursor.close()
    conexion.close()

def seleccionar_todos_los_libros():
    SQL = "SELECT * FROM libros ORDER BY titulo"
    conexion = conectar()
    cursor = conexion.cursor()
    cursor.execute(SQL)
    resultado = cursor.fetchall()
    cursor.close()
    conexion.close()
    return resultado

def busqueda_binaria_libros(libros, titulo):
    inicio = 0
```

```
    fin = len(libros) - 1
    while inicio <= fin:
        medio = (inicio + fin) // 2
        titulo_medio = libros[medio][1]
        if titulo_medio == titulo:
            return libros[medio]
        elif titulo_medio < titulo:
            inicio = medio + 1
        else:
            fin = medio - 1
    return None

def buscar_libro_por_titulo(titulo):
    libros = seleccionar_todos_los_libros()
    resultado = busqueda_binaria_libros(libros, titulo)
    if resultado:
        print("Libro encontrado:", resultado)
    else:
        print("Libro no encontrado")

if __name__ == "__main__":
    def opcion1():
        consultar_libros_mas_populares()

    def opcion2():
        quitar_libro_menu()

    def opcion3():
        agregar_libro("El gran Gatsby", "F. Scott Fitzgerald",
"Novela", "1948-05-21", 5, 4)

    def opcion4():
        buscar_libro_por_titulo("El gran Gatsby")

    switch = {
        1: opcion1,
        2: opcion2,
        3: opcion3,
        4: opcion4,
    }

    eleccion = int(input("Elige una opción (1-4): "))
  resultado = switch.get(eleccion, lambda: print("Opción no válida"))()
```

EXPLICACION DEL CODIGO DE PYTHON

1. `conectar()`

```
def conectar():
    conexion = mysql.connector.connect(
        host='localhost',
        user='root',
        password='',
        database='biblioteca'
    )
    return conexion
```

¿Qué hace?
Crea y devuelve una conexión a la base de datos MySQL llamada biblioteca usando el usuario root y sin contraseña.
¿Para qué sirve?
Permite que las demás funciones interactúen con la base de datos (leer, insertar, borrar, etc.) usando esta “puerta de entrada”.

2. `consultar_libros_mas_populares()`

```
def consultar_libros_mas_populares():
    SQL = "SELECT titulo, calificacion FROM libros WHERE calificacion
>= 4 ORDER BY calificacion DESC"
    conexion = conectar()
    cursor = conexion.cursor()
    cursor.execute(SQL)
    resultado = cursor.fetchall()
    for libro in resultado:
        print(f"Título: {libro[0]}, Calificación: {libro[1]}")
    cursor.close()
    conexion.close()
```

¿Qué hace?
Consulta y muestra en pantalla todos los libros con calificación igual o superior a 4, ordenados de mayor a menor calificación.
¿Para qué sirve?
Para ver rápidamente los libros mejor calificados en la biblioteca.

3. quitar_libro_menu()

```
def quitar_libro_menu():
    SQL = "SELECT id_libro, titulo FROM libros ORDER BY titulo"
    conexion = conectar()
    cursor = conexion.cursor()
    cursor.execute(SQL)
    libros = cursor.fetchall()
    if not libros:
        print("No hay libros para eliminar.")
        cursor.close()
        conexion.close()
        return

    print("Libros disponibles para eliminar:")
    for idx, libro in enumerate(libros, 1):
        print(f"{idx}: {libro[1]}")

    while True:
        try:
            opcion = int(input("Elige el número del libro a eliminar:
"))
            if 1 <= opcion <= len(libros):
                id_libro = libros[opcion-1][0]
                break
            else:
                print("Opción inválida, intenta de nuevo.")
        except ValueError:
            print("Ingresa un número válido.")

    SQL_delete = "DELETE FROM libros WHERE id_libro = %s"
    cursor.execute(SQL_delete, (id_libro,))
    conexion.commit()
    print(f'Libro "{libros[opcion-1][1]}" eliminado correctamente.')
    cursor.close()
    conexion.close()
```

¿Qué hace?
Muestra un menú numerado de todos los libros disponibles. El usuario escoge el número correspondiente al libro que quiere eliminar, y ese libro se borra de la base de datos.
¿Para qué sirve?
Para borrar libros fácilmente, eligiéndolos por nombre y no por ID.

4. agregar_libro(..

```
def agregar_libro(titulo, autor, genero, publicacion,
cantidad_disponible, calificacion):
    SQL = "INSERT INTO libros (titulo, autor, genero, publicacion,
cantidad_disponible, calificacion) VALUES (%s, %s, %s, %s, %s, %s)"
    valores = (titulo, autor, genero, publicacion, cantidad_disponible,
calificacion)
    conexion = conectar()
    cursor = conexion.cursor()
    cursor.execute(SQL, valores)
    print(cursor.statement)
    conexion.commit()
    cursor.close()
    conexion.close()
```

¿Qué hace?
Agrega un nuevo libro a la base de datos con los datos proporcionados: título, autor, género, año de publicación, cantidad disponible y calificación.
¿Para qué sirve?
Permite ingresar nuevos libros a la biblioteca.

5. seleccionar_todos_los_libros()

```
def seleccionar_todos_los_libros():
    SQL = "SELECT * FROM libros ORDER BY titulo"
    conexion = conectar()
    cursor = conexion.cursor()
    cursor.execute(SQL)
    resultado = cursor.fetchall()
    cursor.close()
    conexion.close()
    return resultado
```

¿Qué hace?
Devuelve una lista con todos los libros de la base de datos, ordenados alfabéticamente por título.
¿Para qué sirve?

Para obtener todos los datos de los libros, útil en búsquedas o para mostrar el catálogo completo.

6. busqueda_binaria_libros(libros, titulo)

```
def busqueda_binaria_libros(libros, titulo):

    inicio = 0
    fin = len(libros) - 1
    while inicio <= fin:
        medio = (inicio + fin) // 2
        titulo_medio = libros[medio][1]  # columna 1 = titulo
        if titulo_medio == titulo:
            return libros[medio]
        elif titulo_medio < titulo:
            inicio = medio + 1
        else:
            fin = medio - 1
    return None
```

¿Qué hace?
Realiza una búsqueda binaria en una lista de libros (ordenados por título) para encontrar un libro específico por su título exacto.
¿Para qué sirve?
Para buscar rápidamente un libro por título, sin recorrer toda la lista secuencialmente.

7. buscar_libro_por_titulo(titulo)

```
def buscar_libro_por_titulo(titulo):
    libros = seleccionar_todos_los_libros()
    resultado = busqueda_binaria_libros(libros, titulo)
    if resultado:
        print("Libro encontrado:", resultado)
    else:
        print("Libro no encontrado")
```

¿Qué hace?
Utiliza las dos funciones anteriores para buscar un libro por su título y muestra si fue encontrado o no.
¿Para qué sirve?
Para que el usuario pueda buscar cualquier libro por su nombre y ver sus datos.

8. Main y menú de opciones

```
if __name__ == "__main__":

    def opcion1():

        consultar_libros_mas_populares()



    def opcion2():

        quitar_libro_menu()



    def opcion3():

        agregar_libro("El gran Gatsby", "F. Scott Fitzgerald",
"Novela", "1948-05-21", 5, 4)



    def opcion4():

        buscar_libro_por_titulo("El gran Gatsby")



    switch = {

        1: opcion1,

        2: opcion2,

        3: opcion3,

        4: opcion4,

    }



    eleccion = int(input("Elige una opción (1-4): "))
```

```
    resultado = switch.get(eleccion, lambda: print("Opción no
válida"))()
```

¿Qué hace?
Define las funciones que ejecutan cada opción del menú principal del programa. Usa un diccionario (switch) para asociar el número elegido con la función respectiva. Luego, pide al usuario que elija una opción, y ejecuta la función correspondiente (por ejemplo, mostrar libros populares, eliminar un libro, etc.).

¿Para qué sirve?
Es la “puerta de entrada” del programa. Permite elegir fácilmente qué acción realizar en la biblioteca digital.