Ejercicio 1: Sistema de Gestión de Tienda

Crea un programa en Python que permita gestionar los productos de una tienda. El programa debe:

1. Usar **clases** para representar productos y la tienda.
2. Usar un **diccionario** para almacenar temporalmente los productos en memoria antes de guardarlos en la base de datos. (Opcional, la única ventaja, si el propósito es practicar estructuras de datos en memoria o para realizar operaciones rápidas antes de sincronizar con la base de datos)
3. Conectar con una base de datos SQLite para almacenar los productos de manera persistente.

Requisitos del programa:

1. **Clases:**
    - `Producto`: Representa un producto con atributos como `id`, `nombre`, `precio`, y `stock`.
    - `Tienda`: Administra una lista de productos usando un diccionario y realiza operaciones sobre ellos (agregar, eliminar, listar, etc.).
2. **Diccionarios:**
    - Usar un diccionario para almacenar productos con el formato `{id: producto}` antes de guardarlos en la base de datos.
3. **Base de datos:**
    - Crear una base de datos SQLite con una tabla `productos` que almacene los productos de la tienda.
    - Operaciones básicas: agregar, eliminar, actualizar, y listar productos.
4. **Menú interactivo:**
    - Permitir al usuario interactuar con el sistema a través de un menú en consola.

Ejercicio2: Sistema de Gestión de Tareas con Usuarios

Descripción

Crea un programa en Python que permita gestionar tareas de múltiples usuarios. El sistema debe permitir:

1. Registrar usuarios.
2. Asignar tareas a usuarios.
3. Consultar las tareas pendientes y completadas de un usuario.
4. Marcar tareas como completadas.
5. Guardar y recuperar la información desde una base de datos SQLite.

Este ejercicios es un poco más libre. Debes elegir la estructura de datos que más interesa de tal manera que el código sea óptimo.

Ejercicio3. Biblioteca

Se desea construir un sistema para gestionar una biblioteca digital. Este sistema debe permitir almacenar información sobre libros, autores y préstamos realizados a los lectores. Utiliza una **base de datos SQLite** para guardar los datos y deja libre la decisión de utilizar estructuras adicionales como listas o diccionarios para realizar operaciones temporales o manejar datos en memoria.

Tareas a realizar:

1. **Base de datos:**
   Diseña una base de datos SQLite que contenga las siguientes tablas:
   - **Libros**:
     - `id_libro` (clave primaria)
     - `titulo`
     - `autor`
     - `anio_publicacion`
     - `cantidad_disponible` (cantidad de copias en la biblioteca)
   - **Lectores**:

        - `id_lector` (clave primaria)
        - `nombre`
        - `correo`
    - **Prestamos**:
        - `id_prestamo` (clave primaria)
        - `id_lector` (clave foránea de la tabla Lectores)
        - `id_libro` (clave foránea de la tabla Libros)
        - `fecha_prestamo`
        - `fecha_devolucion`

2. **Funciones obligatorias del sistema:**
    - **Registrar libros y lectores:** Permite agregar nuevos libros y lectores a la base de datos. Los datos iniciales pueden cargarse manualmente desde el código o ingresarse dinámicamente durante la ejecución del programa.
    - **Registrar un préstamo:** Solicita al usuario:
        - ID del lector que solicita el préstamo.
        - ID del libro que desea prestar.
    - Debes verificar que:
        - El lector exista en la base de datos.
        - El libro exista y tenga copias disponibles.
        - Una vez registrado el préstamo, el número de copias disponibles del libro debe disminuir en uno.
    - **Listar préstamos activos:** Muestra todos los préstamos activos con detalles como:
        - Nombre del lector.
        - Título del libro.
        - Fechas de préstamo y devolución.
    - **Devolver un libro:** Solicita el ID del préstamo, marca la devolución y aumenta la cantidad disponible del libro correspondiente.
3. **Estructura libre adicional:**
    - Elige cómo manejar temporalmente los datos antes de insertarlos o mostrarlos, usando listas, diccionarios u otras estructuras de datos que prefieras.

- Por ejemplo: puedes usar un diccionario para cargar libros de manera inicial o una lista para mostrar préstamos activos en memoria antes de procesar.

4. **Menú principal:**
   Implementa un menú interactivo con las siguientes opciones:
   - Registrar un libro.
   - Registrar un lector.
   - Registrar un préstamo.
   - Listar préstamos activos.
   - Devolver un libro.
   - Salir del sistema.

Nota:

- Valida que los datos ingresados por el usuario sean correctos.
- Maneja errores relacionados con el acceso a la base de datos o datos inválidos.
- Puedes incluir ejemplos de datos iniciales para facilitar las pruebas del sistema.

Ejercicio 4: Gestión de una Tienda de Videojuegos

Diseña un sistema en Python que permita gestionar el inventario, los clientes y las compras en una tienda de videojuegos. El sistema debe usar SQLite como base de datos y proporcionar las siguientes funcionalidades:

Requisitos del Sistema

1. **Base de datos SQLite**
   La base de datos debe tener las siguientes tablas:
   - **Videojuegos**:
     - `id_videojuego` (clave primaria)
     - `titulo`
     - `genero`

        - `precio`
        - `cantidad_disponible`
    - **Clientes**:
        - `id_cliente` (clave primaria)
        - `nombre`
        - `correo`
    - **Compras**:
        - `id_compra` (clave primaria)
        - `id_cliente` (clave foránea de la tabla `Clientes`)
        - `id_videojuego` (clave foránea de la tabla `Videojuegos`)
        - `cantidad`
        - `fecha_compra`

2. **Funcionalidades del sistema**:
    - Registrar nuevos videojuegos en el inventario.
    - Registrar nuevos clientes.
    - Registrar una compra:
        - Solicitar el ID del cliente, el ID del videojuego y la cantidad deseada.
        - Verificar que el cliente exista.
        - Verificar que el videojuego exista y que haya suficiente stock.
        - Registrar la compra, disminuir el inventario del videojuego y mostrar el total de la compra.
    - Consultar el inventario de videojuegos: Mostrar todos los videojuegos disponibles con su información.
    - Consultar el historial de compras de un cliente: Mostrar todas las compras realizadas por un cliente específico, con detalles como título del videojuego, cantidad y fecha.