EJ 1, TEMA 3 : Conexión con peewee, modelo de base de datos, verificamos que existe, creamos tabla, insertamos 5 registros, mensajes informativos, cerramos las conexiones

EJ 2, TEMA 3 : filtrar por lo que ponga en un atributo, eliminar un registro en especifico, elimina todos los registros que cumplan una condicion

EJ 3, TEMA 3 : Transacciones, db.atomic()

EJ 4, TEMA 4: Transacciones tambien

Pasar de peewee a ZODB (de chatgpt no lo he probado)

```
from peewee import *
from ZODB import DB
from ZODB.FileStorage import FileStorage
import transaction

# Configuración de la base de datos MySQL para Peewee
mysql_db = MySQLDatabase(
        'mi_base_datos',  # Cambia esto por el nombre de tu base de datos
        user='root',      # Usuario de MySQL
        password='root',  # Contraseña de MySQL
        host='localhost', # Host de la base de datos
        port=3306       # Puerto (3306 es el predeterminado)
)

# Definición del modelo Peewee para la tabla `coches`
class Coche(Model):
        id = AutoField()
        marca = CharField()
        modelo = CharField()
        anio = IntegerField()
        precio = FloatField()

        class Meta:
        database = mysql_db

# Configuración de la base de datos ZODB
storage = FileStorage('coches_data.fs')
db = DB(storage)
connection = db.open()
root = connection.root

# Función principal para transferir los datos
def transferir_coches():
        # Conectar a la base de datos MySQL
        mysql_db.connect()

        # Crear una estructura en ZODB si no existe
```

```
        if 'coches' not in root:
        root['coches'] = {}

        # Leer los registros de la tabla `coches`
        coches = Coche.select()

        # Transferir registros a ZODB
        with transaction.manager:
        for coche in coches:
        root['coches'][coche.id] = {
                'marca': coche.marca,
                'modelo': coche.modelo,
                'anio': coche.anio,
                'precio': coche.precio
        }

        print(f"{len(coches)} registros transferidos a ZODB.")

        # Cerrar la conexión a MySQL
        mysql_db.close()

# Llamar a la función principal
if __name__ == "__main__":
        transferir_coches()
        # Cerrar la conexión a ZODB
        connection.close()
        db.close()
```

Pasar de ZODB a peewee (de chatgpt no lo he probado)

```
from peewee import *
from ZODB import DB
from ZODB.FileStorage import FileStorage
import transaction

# Configuración de la base de datos MySQL para Peewee
mysql_db = MySQLDatabase(
        'mi_base_datos',  # Cambia esto por el nombre de tu base de datos
        user='root',      # Usuario de MySQL
        password='root',  # Contraseña de MySQL
        host='localhost', # Host de la base de datos
        port=3306       # Puerto (3306 es el predeterminado)
)

# Definición del modelo Peewee para la tabla `coches`
class Coche(Model):
        id = AutoField()
        marca = CharField()
```

```
        modelo = CharField()
        anio = IntegerField()
        precio = FloatField()

        class Meta:
        database = mysql_db

# Configuración de la base de datos ZODB
storage = FileStorage('coches_data.fs')
db = DB(storage)
connection = db.open()
root = connection.root

# Función principal para importar coches desde ZODB a MySQL
def importar_coches():
        # Conectar a la base de datos MySQL
        mysql_db.connect()

        # Comprobar si la estructura de ZODB existe
        if 'coches' not in root:
        print("No se encontraron datos de coches en ZODB.")
        return

        # Leer los datos de coches desde ZODB
        coches_zodb = root['coches']
        print(f"Se encontraron {len(coches_zodb)} registros en ZODB.")

        # Insertar registros en MySQL
        with mysql_db.atomic():  # Transacción para optimizar la inserción
        for coche_id, coche_data in coches_zodb.items():
        Coche.create(
                id=coche_id,
                marca=coche_data['marca'],
                modelo=coche_data['modelo'],
                anio=coche_data['anio'],
                precio=coche_data['precio']
        )
        print(f"{len(coches_zodb)} registros importados a MySQL.")

        # Cerrar la conexión a MySQL
        mysql_db.close()

# Llamar a la función principal
if __name__ == "__main__":
        importar_coches()
        # Cerrar la conexión a ZODB
        connection.close()
        db.close()
```

EJ 1, TEMA 4 : almacenar y recuperar una lista de 3 objetos, transacciones
EJ 2, TEMA 4 : crear clase del objeto asignado, almacenar objetos de esa clase, transacciones, recorrer y comprobar si el objeto tiene un atributo en concreto, comprobar si un atributo cumple una condición y mostrarlo
EJ 3, TEMA 4 : actualizar un registro, commit
EJ 4, TEMA 4 : transacciones con segundo enfoque
EJ 5, TEMA 5 : segunda tabla
EJ 6, TEMA 4: deepcopy