On Windows 11, VSCode, terminal

.\venv\Scripts\Activate.ps1

Accessing PyQT5 Designer, VSCode, terminal

qt5-tools designer

SQLite and SQLAlchemy are two different technologies that can be used in Python for working with databases, but they serve different purposes. Here is a comparison of the two:

SQLite:

SQLite is a C library that provides a lightweight, serverless, self-contained SQL database engine.

It is a file-based database system, which means that the entire database is stored in a single file on disk.

SQLite is a great choice for small to medium-sized applications, such as mobile apps, embedded systems, and lightweight web applications.

It is easy to set up and requires minimal configuration.

SQLite is not well-suited for high-concurrency or high-transaction-rate applications, as it does not support concurrent writes and has limited support for concurrent reads.

In Python, you can interact with SQLite using the built-in sqlite3 module, which provides a simple API for executing SQL queries and managing transactions.

SQLAlchemy:

SQLAlchemy is an Object Relational Mapper (ORM) library for Python, which means it provides a high-level, Pythonic API for working with relational databases.

It supports multiple database systems, including SQLite, PostgreSQL, MySQL, and others.

SQLAlchemy abstracts the underlying database system, allowing you to write more maintainable and portable code.

It provides a powerful and flexible query API, making it easier to express complex queries and join operations.

SQLAlchemy can automatically manage schema creation and migration, making it convenient to work with evolving data models.

It has a steeper learning curve compared to using raw SQL, but it can help to reduce the amount of boilerplate code and potential bugs when working with databases.

In summary, SQLite is a lightweight, file-based SQL database engine that is well-suited for small to medium-sized applications, while SQLAlchemy is a powerful ORM library that provides a high-level, Pythonic API for working with various relational databases. If you need to work with SQLite specifically, you can use SQLAlchemy as an abstraction layer on top of SQLite, or you can use the built-in sqlite3 module for more direct control. However, if you need to work with other databases or want to abstract away the underlying database system, SQLAlchemy is a better choice.