Python for Beginners

Database Programming with SQLite3

Programming a Database with Python

You can use several databases with python:

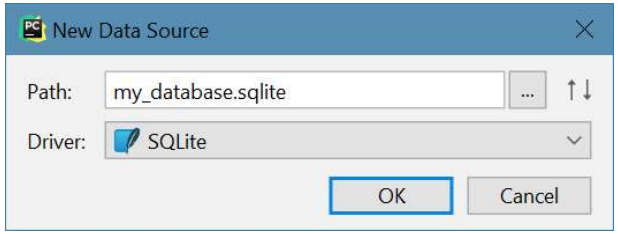
- MySQL
  - very popular (free alternative: MariaDB)
  - install the MySQL Driver Library with PIP:

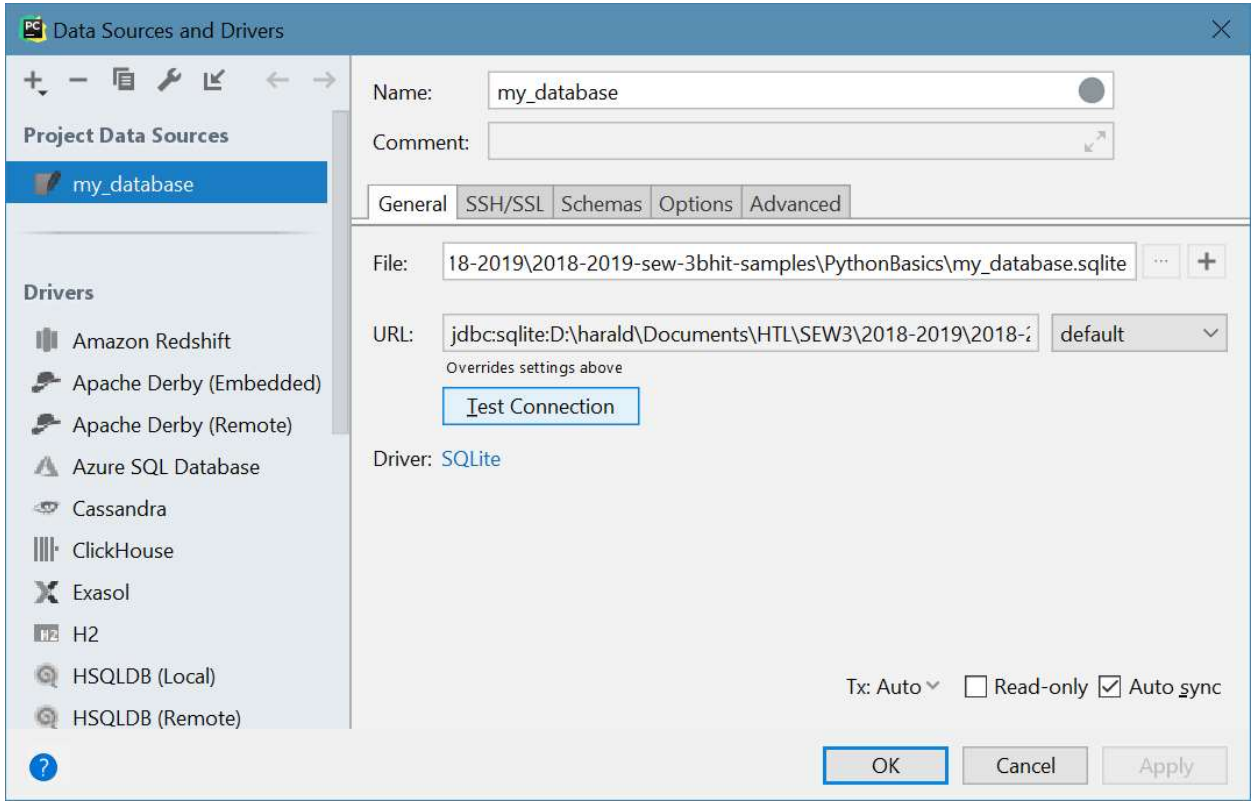
    ```
    python -m pip install mysql-connector
    ```

  - For more details see:
    https://www.w3schools.com/python/python_mysql_getstarted.asp
    http://www.mysqltutorial.org/python-mysql
    https://dev.mysql.com/doc/connector-python/en/connector-python-examples.html
- SQLite
  - built in standard python library SQLite3
  - very easy and small DB
    - no installation of a DBMS needed (the database is just a file)
  - Ideal for small projects and prototyping
  - For more details see:
    https://docs.python.org/3/library/sqlite3.html
    http://www.sqlitetutorial.net/sqlite-python

Create the database

- Create an empty file at your preferred place
- Or with PyCharm: Create a new DataSource
  - Select the SQLite driver
  - Download the SQLite driver
  - Test the connection





Steps of Database Programming

1. Connect to the database
2. Create the database schema (=tables) unless it is already created (of course this can be done separately with other tools, too)
3. Get a cursor from the connection
4. Execute a SQL statement with this cursor
5. Commit the changes if it was an insert, update or delete statement
6. Get the result(s) if it was a select statement
7. Close the connection

Connect to the database

- Import the sqlite3 module
- Get a connection to the database file

```
import sqlite3

path = "d:\data\my_database.sqlite" # absolute path
path = "my_database.sqlite" # or in the project directory

connection = sqlite3.connect(path)
```

Create the database schema

- Define the SQL create table statement

```
sql_create_table = "CREATE TABLE IF NOT EXISTS products (id integer PRIMARY KEY, name text NOT NULL, price decimal NOT NULL)"
```

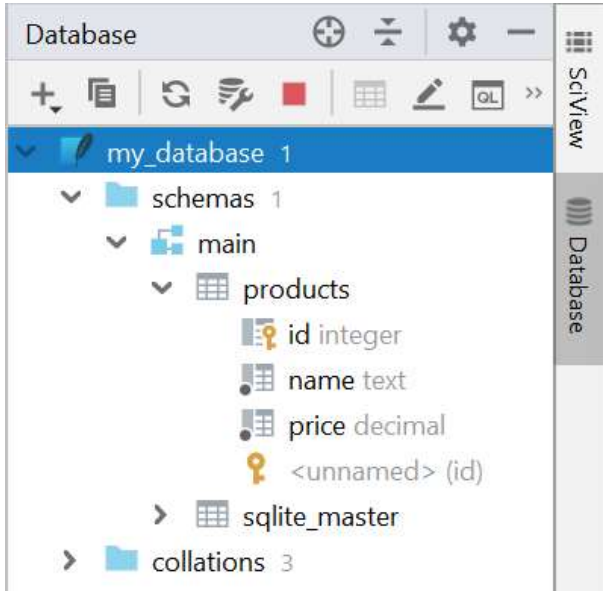
- Get the cursor

```
cursor = connection.cursor()
```

- Execute the statement

```
cursor.execute(sql_create_table)
```

- Check the result



Insert a row

- Define the SQL insert statement

```
sql_insert = "INSERT INTO products(name,price) VALUES(?,?)"
```

? … placeholder for parameter

- Create a list of values

```
values = ['Laptop', 1290]
```

- Execute the insert statement with the values

```
cursor.execute(sql_insert, values)
```

- Get the id of the inserted row created by the database

```
id = cursor.lastrowid
```

- Commit the changes (= save the inserted row)

```
connection.commit()
```

Update a row

- Define the SQL update statement

```
sql_update = "UPDATE products SET name=?, price=? WHERE id=?"
```

- Create a list of values

```
values = ['Laptop for sale', 849, 3]
```

- Execute the insert statement with the values

```
cursor.execute(sql_update, values)
```

- Commit the changes (= save the updated row)

```
connection.commit()
```

Select all rows

- Define the SQL select statement

```
sql_select_all = "SELECT * FROM products"
```

- Execute the select statement

```
cursor.execute(sql_select_all)
```

- Fetch the resulting rows

```
rows = cursor.fetchall()

for row in rows:
    print(row)
```

Select a specific row

- Define the SQL select statement

```
sql_select = "SELECT * FROM products WHERE id = ?"
```

- Create a list of values

```
values = [3]
```

- Execute the select statement with the values

```
cursor.execute(sql_select, values)
```

- Fetch the resulting row

```
row = cursor.fetchone()
print(row)
```

→ (3, 'Laptop for sale', 849)

Delete a row

- Define the SQL delete statement

```
sql_delete = "DELETE FROM products WHERE id=?"
```

- Create a list of values

```
values = [3]
```

- Execute the delete statement with the values

```
cursor.execute(sql_delete, values)
```

- Commit the changes (= remove the deleted row)

```
connection.commit()
```