37 Working with SQLite

37.1 About

In this activity you will use the sqlite3 library to create a database, and use SQL statements to insert and query data.

37.2 Before you start

Close any open projects in Pycharm.

37.3 Create a database

To use an SQLite database, we have to create a connection to the database. If the database doesn't exist, it is created.

Create new project in Pycharm in your Python workspace folder. Call the project **sqlite3Project** or similar.

Start by adding the necessary import statement at the top of the script.

```
import sqlite3
```

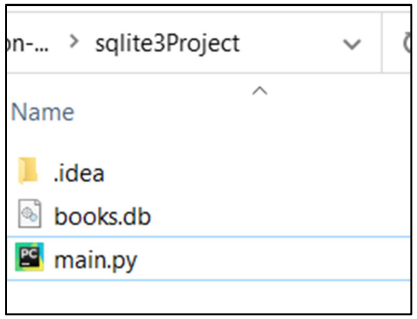
Add a line to create a database connection, a line to get a cursor, and a line to close the connection again.

```
con = sqlite3.connect("books.db")
cur = con.cursor()

con.close()
```

Note that all the following lines should be between the **connect** and **close** statements.

Save and run at this point. There will be no visible output from the program, but it should finish without any errors, and if you check in the folder where the program is stored, you will find a new file.



Note that the file extension is not compulsory, but **.db** is traditional.

Now that we have a database connection, we can use it to work with the database. Add a line that creates a table:

```
cur.execute("create table books (title text, author text)")
con.commit()
```

Save and run. Again the program should finish with no errors. But if you run it a second time an error appears:

```
Traceback (most recent call last):
  File "C:\python-workspace\sqlite3Project\main.py", line 5, in <module>
    conn.execute("create table books (title text, author text)")
sqlite3.OperationalError: table books already exists

Process finished with exit code 1
```

To avoid this happening, change the create table line to read as follows:

```
cur.execute("create table if not exists books (title text, author text)")
```

Run again. This time the program should complete with no errors.

37.4 Insert data

Data is inserted into an SQLite table using the SQL **insert into** statement. Insert statements can use data originating from many places- it can be hard coded, or from tuples, or read from a file. We will start with some hard coded data. Add the following lines to the script- substitute your own favourite books if you want:

```
cur.execute("insert into books values('War and Peace', 'Tolstoy, Leo')")
cur.execute("insert into books values('Pride and Prejudice', 'Austen, Jane')")
cur.execute("insert into books values('Great Expectations', 'Dickens, Charles')")
con.commit()
```

Save and run. The program will finish with no errors but there will be no visible output yet.

37.5 Read data

Let's check that the data has been successfully added to the database. To do that, we'll use a **select** statement. When the statement is executed, it generates a set of results that can either be read one at a time, or all at once- the **fetchone** command can be used to get one row at a time, but in this case we'll use **fetchall**. The result set is a list, so it can be printed to see the output.

```
cur.execute("select * from books")
res = cur.fetchall()
```

This time when the program runs the result set is displayed.

```
[('War and Peace', 'Tolstoy, Leo'), ('Pride and Prejudice', 'Austen, Jane'), ('Great Expectations', 'Dickens, Charles')]

Process finished with exit code 0
```

37.6 Insert data from a tuple

If we have several records to insert at the same time, instead of writing several **insert into** statements it's a bit easier to put the data into a list and write a single insert statement instead, using the **executemany** method in place of **execute**.
Add the following lines to the script, before the **select** statement:

```
bookdata = [("Jude the Obscure", "Hardy, Thomas"),
            ("Middlemarch", "Eliot, George"),
            ("Animal Farm", "Orwell, George")]
cur.executemany("insert into books values(?, ?)", bookdata)
con.commit()
```

Note- the **?** in the SQL statement indicates it is a **parameterised query**. In this case, the values of the parameters are specified in the tuple. It is generally considered safer to do this than to use Python's standard string formatting (for example using string concatenation or the **{}** style). When using the **?** notation, the SQLite library understands it is a parameter and can do any necessary sanitisation or validation on the parameter value. This means that security issues such as SQL injection can be avoided.

Save and run and check the output to make sure the data has been inserted:

```
[('War and Peace', 'Tolstoy, Leo'), ('Pride and Prejudice', 'Austen, Jane'), ('Great Expectations', 'Dickens, Charles'), ('Jude the Obscure',
  'Hardy, Thomas'), ('Middlemarch', 'Eliot, George'), ('Animal Farm', 'Orwell, George')]

Process finished with exit code 0
```

Note- if your output runs off the side of the screen, switch on soft wrapping using the button by the side of the output panel.

37.7 Insert data from a file

If we have a lot of data to insert at a time, it would be convenient to store the data in a file such as a CSV file and read it from the file in the script and use the data in insert statements.

Start by creating a new file in the same folder as the Python script. Call the file **books.csv**. Add some lines to the file. Note, since the author names contain commas, it is necessary to surround the names with quotes.

```
"Brighton Rock","Greene, Graham"
"The Moon and Sixpence", "Maugham, W Somerset"
"One Hundred Years of Solitude", "Marquez, Gabriel Garcia"
"Frankenstein", "Shelley, Mary"
```

In the script, add lines to read the file. We can use Python's **csv** library to do this. First add the **import** at the top of the script:

```
import csv
```

Start by adding some lines that read the file and print the contents to check the file is being processed correctly:

```
infile = open("books.csv")
for line in csv.reader(infile, quotechar='"', delimiter=',', quoting=csv.QUOTE_ALL, skipinitialspace=True):
    print(line)

infile.close()
```

Run the script and check the output:

```
['Brighton Rock', 'Greene, Graham']
['The Moon and Sixpence', 'Maugham, W Somerset']
['One Hundred Years of Solitude', 'Marquez, Gabriel Garcia']
['Frankenstein', 'Shelley, Mary']

Process finished with exit code 0
```

Now modify the loop so that as each line is read from the file, the data is inserted into the table.

```
for line in csv.reader(infile, quotechar='"', delimiter=',', quoting=csv.QUOTE_ALL, skipinitialspace=True):
    cur.execute("insert into books values(?, ?)", line)
```

37.8 Checkpoint

At this point the script should look like this:

```
import sqlite3
import csv

con = sqlite3.connect("books.db")
cur = con.cursor()

cur.execute("create table if not exists books (title text, author text)")
commit()

cur.execute("insert into books values('War and Peace', 'Tolstoy, Leo')")
cur.execute("insert into books values('Pride and Prejudice', 'Austen, Jane')")
cur.execute("insert into books values('Great Expectations', 'Dickens, Charles')")
commit()

bookdata = [("Jude the Obscure", "Hardy, Thomas"),
```

```
            ("Middlemarch", "Eliot, George"),
            ("Animal Farm", "Orwell, George")]
cur.executemany("insert into books values(?, ?)", bookdata)
commit()

infile = open("books.csv")
for line in csv.reader(infile, quotechar='"', delimiter=',', quoting=csv.QUOTE_ALL, skipinitialspace=True):
    cur.execute("insert into books values(?, ?)", line)
infile.close()
commit()

cur.execute("select * from books")
res = cur.fetchall()

print(res)

con.close()
```

37.9 Run the script

Run the script and check the output.

```
[('War and Peace', 'Tolstoy, Leo'), ('Pride and Prejudice', 'Austen, Jane'), ('Great Expectations', 'Dickens, Charles'), ('Jude the Obscure',
  'Hardy, Thomas'), ('Middlemarch', 'Eliot, George'), ('Animal Farm', 'Orwell, George'), ('Brighton Rock', 'Greene, Graham'), ('The Moon and
 Sixpence', 'Maugham, W Somerset'), ('One Hundred Years of Solitude', 'Marquez, Gabriel Garcia'), ('Frankenstein', 'Shelley, Mary')]

Process finished with exit code 0
```

37.10 Modify the query

37.10.1 Alphabetical order

At the moment when the book list is returned, the result set is unsorted. Change the query so that the list is shown in alphabetical order of the author's name.

```
cur.execute("select * from books order by author")
```

Check that the output is correct:

```
[('Pride and Prejudice', 'Austen, Jane'), ('Great Expectations', 'Dickens, Charles'), ('Middlemarch', 'Eliot, George'), ('Brighton Rock',
 'Greene, Graham'), ('Jude the Obscure', 'Hardy, Thomas'), ('One Hundred Years of Solitude', 'Marquez, Gabriel Garcia'), ('The Moon and
 Sixpence', 'Maugham, W Somerset'), ('Animal Farm', 'Orwell, George'), ('Frankenstein', 'Shelley, Mary'), ('War and Peace', 'Tolstoy, Leo')]

Process finished with exit code 0
```

We probably wouldn't want to list all the books in the table every time. To get results just for one author, add a **where** clause to the query.

```
cur.execute("select * from books where author = 'Austen, Jane' order by author")
```

Check the output:

```
[('Pride and Prejudice', 'Austen, Jane')]

Process finished with exit code 0
```

37.11 Challenge

In the previous query we used the exact author's name to get a result. If this database was part of a book searching system, sometimes we would probably want to enter just part of a name or title and get all the matching results. SQL has a **like** operator that can be used in place of **=** to do this. For example, saying **select * from books where author like '%george%'** would return results by **Orwell, George** and **Eliot, George**. Note that the search term must be surrounded by **%** symbols.

Modify the script so that after loading the data into the table, the script asks the user for a search term and returns all the results where either the title or the author's name matches the search term. Repeat this until the input is blank.

The output should be similar to this:

```
Enter search term : geo
[('Middlemarch', 'Eliot, George'), ('Animal Farm', 'Orwell, George')]
Enter search term : mar
[('Middlemarch', 'Eliot, George'), ('One Hundred Years of Solitude', 'Marquez, Gabriel Garcia'), ('Frankenstein', 'Shelley, Mary')]
Enter search term : fra
[('Frankenstein', 'Shelley, Mary')]
Enter search term :

Process finished with exit code 0
```

37.12 Updating and deleting data

Data in tables can be updated and deleted using SQL statements.

37.12.1 Updating

Our book database has decided that authors with initialisations in their names should be stored using their full names. So for example **W Somerset Maugham** should be **William Somerset Maugham**.

Update this record in the database, and check that the replacement has been successful, using the following commands:

```
cur.execute("update books set author = 'Maugham, William Somerset' where author = 'Maugham, W Somerset'")
cur.execute("select * from books where author like '%maugham%'")
res = cur.fetchall()
print(res)
```

Run the script and check the output:

```
[('The Moon and Sixpence', 'Maugham, William Somerset')]

Process finished with exit code 0
```

37.12.2 Deleting

The book database owner has decided to remove books by George Orwell from the database.
Add these commands to the script (after the data has been loaded into the table):

```
cur.execute("delete from books where author = 'Orwell, George'")
cur.execute("select * from books")
res = cur.fetchall()
print(res)
```

Run the script and make sure Orwell is no longer present:

```
[('War and Peace', 'Tolstoy, Leo'), ('Pride and Prejudice', 'Austen, Jane'), ('Great Expectations', 'Dickens, Charles'), ('Jude the Obscure',
  'Hardy, Thomas'), ('Middlemarch', 'Eliot, George'), ('Brighton Rock', 'Greene, Graham'), ('The Moon and Sixpence', 'Maugham, William
 Somerset'), ('One Hundred Years of Solitude', 'Marquez, Gabriel Garcia'), ('Frankenstein', 'Shelley, Mary')]

Process finished with exit code 0
```

Note- a few years ago Amazon removed some books from customer's Kindle ebook readers without their knowledge during a copyright dispute. The books included George Orwell's dystopian classic Nineteen Eighty-Four (usually just known as 1984). See https://www.theguardian.com/technology/2009/jul/17/amazon-kindle-1984