Python SQLite 操作

Python /SQLite

SQLite

- SQL stands for "Structured Query Languages"
- SQL is the main language that large database packages use.
- SQLite is free and can be downloaded from www.sqlite.org
- To download select the "Precomplied Binaries" for your operating system (Mac OS, Windows or Linux)
- To use SQLite you need to load the "DB Browser for SQLite" from https://sqlitebrowser.org

Data types for fields

- `Integer`: the value is an integer value
- `Real`: the value is a floating-pint value
- `Text`: the value is a string text
- `Blob`: the value is stored exactly as it was input.
- You can also specify if the field cannot be left blank by adding `NOT NULL` to the end of the field when you create it

Python SQLite3 - Example code

- `import sqlite3` Allows Python to use the SQLite3 library

- `Db = sqlite3.connect("university.db")`
  `cursor=db.cursor()` Connects to the `university` database. If no such database exists, it will create one. The file will be stored in the same folder as the programme.

```
cursor.execute("""CREATE TABLE IF NOT EXIST students(
        id integer PRIMARY KEY,
        name text NOT NULL,
        class text NOT NULL,
        grade integer);""")
```

creates a table called `students` which has four fields (id, name, class and grade). It specifies the data type for each field, defines which field is the primary key and which field cannot be left blank. The triple speech marks allow the code to be split over several lines to make it easier to read rather than having it all displayed in one line.

想一想，练一练

- 创建 university 数据库
- 创建students表

- 提示：
- db = sqlite3.connect(“university.db”)
- cursor=db.cursor()
- 在 university 数据库中创建 students 表，至少包括 ID，姓名，性别，年龄等必要字段，并自己为这些字段选择合适的字段类型。

Example code

```
cursor.execute("""INSERT INTO students(id,name, class,grade)
  VALUES(1,Mary,"Python",67)""")
```

`db.commit()` Inserts data into the `students` table. The `db.commit()` line saves the changes.

```
newID = input ("Enter ID number: ")
newName = input("Enter name: ")
newClass = input("Enter class: ")
newGrade = input("Enter grade: ")
cursor.execute("""INSERT INTO students(id, name, class, grade)
  VALUES(?, ?, ?, ?)""", (newID, newName, newClass, newGrade))
```

`db.commit()` allows a user to enter new data which is then inserted into the `students` table

```
cursor.execute("SELECT * FROM students")
Print(cursor.fetchall()
```

Displays all the data from the `students` table.

```
db.close()
```

This must be the last line in the programme to close the database.

想一想，练一练

- 增加和更新students 表的数据
- 至少有3名同学（周边）

- 在university 数据库中建立class表
- 增加和更新表的数据

- 至少有三门以上课程
- 提示，至少包括课号，课程名，教师。

Example code

```
cursor.execute("SELECT * FROM students")
for x in cursor.fetchall():
  print(x)
```
Displays all the data from the `students` table and displays each record on a separate line

```
cursor.execute("SELECT * FROM students ORDER By name")
for x in cursor.fetchall():
  print(x)
```
Selects all the data from the `students` table, sorted by name and displays each record on a separate line.

```
cursor.execute("SELECT * FROM students WHERE grade>50")
```
Selects all the data from the `students` table where the grade is over 50.

```
cursor.execute("SELECT * FROM students WHERE class = 'Python'")
```
selects all the data from the `students` table where the class is "Python".

```
cursor.execute("""SELECT students.id, students.name,
students.lecturer
 FROM students, class WHERE students.class=class.class
 AND students.grade > 70""")
```
Selects the ID and name fields from the `students` table and the lecturer field from the `class` table if the grade is over 70.

```
cursor.execute("SELECT id, name, grade FROM students")
```
Selects the ID, name and grade from the `students` table.

Example code

```
whichClass = input(Enter a class: ")
cursor.execute("SELECT * FROM employees WHERE class=?",
[whichClass])
for x in cursor.fetchall():
   print (x)
```

allows the user to enter a class and displays the records of all the students in that class.

```
cursor.execute("""SELECT students.id, students.name,
class.lecturer
  FROM students,class WHERE students.class=
class.class""")
```

selects the ID and name fields from the students table and the lecturer filed from the class table, using the class filed to link the data. If you do not specify how the tables are linked, Python will assume every students takes every class and you will not get the results you are expecting.

```
cursor.execute("UPDATE students SET name = 'Richard'
WHERE id =1") db.commit()
```

updates the data in the table(overwriting the original) to change the name to "Richard" for student ID 1

```
cursor.execute("DELETE students WHERE id=1")
```

deletes any data in the `students` table where the id is 1

Example code

Create an SQL database called PhoneBook that contains a table called Names with the following data as seen in the code

```
import sqlite3
# Connect to the database called PhoneBook or create one if there is none
with sqlite3.connect("PhoneBook.db") as db:
    cursor = db.cursor()

#    Create a table called Names with foru fields
cursor.execute(""" CREATE TABLE IF NOT EXISTS Names(
id integer PRIMARY KEY,
firstname text,
surname text,
phonenumber text); """)

# Insert data into the table
cursor.execute(""" INSERT INTO Names(id,firstname,surname,phonenumber)
VALUES ("1", "Simon","Pierre","0141647 1367")""")
db.commit() # Saves the changes

# Insert data into the table Names
cursor.execute(""" INSERT INTO Names(id, firstname,surname,phonenumber)
VALUES ("2", "Rita","McVey","0141887 2354")""")
db.commit() # saves the chnages

# Insert data into a table called Names
cursor.execute(""" INSERT INTO Names(id,firstname,surname,phonenumber)
VALUES ("3", "Marc", "Blondel", "0123456 7987")""")
db.commit() # saves the changes

# Select everything from the table called Names and prints one row per line.
cursor.execute(" SELECT * FROM Names")
for x in cursor.fetchall():
    print(x)

db.close() # close the database
```

```
In [146]: runfile('C:/Users/mireilla/.s
(1, 'Simon', 'Pierre', '0141647 1367')
(2, 'Rita', 'McVey', '0141887 2354')
(3, 'Marc', 'Blondel', '0123456 7987')
```

Spilt Tables for easy use

You will notice that more than one student takes the same class. In most databases you will find repetitive data such as this. To make database work more efficiently, the repeated data is often stored in a separate table. In this case there is a **class** table which would store all the information about each class to save having to repeat all the class details for each student.

Table: students

ID	name	class	grade
1899877D	Mary	Python	67
2223998M	John	Maths	34
2348990M	Anne	Python	70

Table: class

class	lecturer
Python	Jack
Maths	Laurie
Java	Joe

By splitting the data into two tables, if we need to update the lecturer, it will only need to be updated once rather than updating it several times, which would have happened if it was all stored in one table.

This is known as **one-to-many** relationship as on class can have many students taking it.

想一想，练一练

- 在university 数据库中创建选课表enrolled表，

- 增加和更新表的数据
- 提示：
- 至少包括：学生ID，选课的课号，考试分数。

想一想，练一练

- 使用 INNER JOIN 查询学生的个人信息和每个课程的分数。

- 尝试outer join操作， left join, right join 和Full JOIN.

Tables

- Students 信息表
- Class 课程表
- Enrolled选课表

想一想，练一练（课外选做）

- 结合Tkinter，做一个成绩查询的SQLite系统

- 提示：
- 创建一个查询按钮，查询所有students表