SQLite 資料庫讀取

Data Processing - SQLite

Python Data Analysis

關聯式資料庫

- 關聯式資料庫系統（RDBS, Relational Database System）是目前資料庫系統的主流。
- 市面上大部分資料庫管理系統都屬於關聯式資料庫管理系統（RDBMS, Relational Database Management System），例如：SQLite、MySQL、SQL Server、Access 和Oracle等。
- 關聯式資料庫（Relational Database）是由一個或多個資料表所組成，在多個資料表間使用欄位的資料值來建立連接，以便實作資料表之間的關聯性。

學號	姓名	電話	生日
S0201	周傑倫	02-11111111	1973/10/3
S0202	林俊傑	02-22222222	1978/2/2
S0203	張振嶽	03-33333333	1982/3/3
S0204	許慧幸	03-44444444	1981/4/4

學號	課程編號	課程名稱	學分
S0201	CS302	專題製作	2
S0202	CS102	資料庫系統	3
S0202	CS104	程式語言(1)	3
S0203	CS201	區域網路實務	3
S0203	CS102	資料庫系統	3
S0203	CS301	專案研究	2
S0204	CS301	專案研究	2

關聯式資料庫

- 在關聯式資料庫是使用二維表格的資料表來儲存記錄資料，在各資料表間使用欄位值建立關聯性，透過關聯性來存取其他資料表的資料。

- 例如：使用【學號】欄位值建立兩個資料表之間的關聯性，如右圖所示：

學號	姓名	電話	生日
S0201	周傑倫	02-11111111	1973/10/3
S0202	林俊傑	02-22222222	1978/2/2
S0203	張振嶽	03-33333333	1982/3/3
S0204	許慧幸	03-44444444	1981/4/4

學號	課程編號	課程名稱	學分
S0201	CS302	專題製作	2
S0202	CS102	資料庫系統	3
S0202	CS104	程式語言(1)	3
S0203	CS201	區域網路實務	3
S0203	CS102	資料庫系統	3
S0203	CS301	專案研究	2
S0204	CS301	專案研究	2

關聯式資料庫

- 關聯式資料庫的資料是儲存在資料庫的「資料表」（Tables），每一個資料表使用「欄位」（Fields）分類成多個群組，每一個群組是一筆「記錄」（Records），例如：通訊錄資料表的記錄，如下表所示：

編號	姓名	地址	電話	生日	電子郵件地址
1	陳會安	台北縣五股成泰路一段1000號	02-11111111	1967/9/3	hueyan@ms2.hinet.net
2	江小魚	中和景平路1000號	02-22222222	1978/2/2	jane@ms1.hinet.net
3	劉得華	桃園市三民路1000號	02-33333333	1982/3/3	lu@tpts2.seed.net.te
4	郭富成	台中市中港路三段500號	03-44444444	1981/4/4	ko@gcn.net.tw
5	離明	台南市中正路1000號	04-55555555	1978/5/5	light@ms11.hinet.net
6	張學有	高雄市四維路1000號	05-66666666	1979/6/6	geo@ms10.hinet.net
7	陳大安	台北市羅斯福路1000號	02-99999999	1979/9/9	an@gcn.net.tw

SQLite 資料庫 簡介

- 資料庫在商業應用中無所不在。與試算表不同的地方在於資料庫的表格是相連的，所以一個資料表可連結到另一個資料表的列或欄位。
- Python 內建一個非常小巧的嵌入式資料庫SQLite，它使用一個文件儲存整個資料庫，操作方便，重要的是它可以使用SQL語法管理資料庫，執行新增、修改、刪除和查詢。
- 底下我們先來瞭解資料庫的相關概念及SQL 語法，再來使用Python進行SQLite的操作。

SQLite 資料庫 簡介

- SQLite 是一個小型的關聯式資料庫管理系統，它不是一個用戶端/伺服器結構的資料庫引擎，而是被整合在用戶程式中。
- SQLite 是目前世界上最廣泛使用的免費資料庫引擎，支援標準資料庫查詢語言 ANSI-SQL 92 大部分的語法。
- SQLite 它不需要管理、不需要伺服器、也不需要安裝設定，不但體積輕巧，而且還是一套支援交易（Transaction）的SQL資料庫引擎。
- SQLite 資料庫系統只是一個檔案，可以使用檔案權限來管理，而不用自行處理資料庫的使用者權限管理。
- SQLite 容量支援到2TB，記憶體需求小，使用Unicode，免費使用。
- SQLite 是手機Android、IOS 作業系統內建的資料庫系統，因不需要啟動，因此不會浪費行動裝置的記憶體資源。

SQL語言的基礎 - DDL/DML

- 「SQL」（Structured Query Language）為「ANSI」（American National Standards Institute）標準的資料庫語言，可以存取和更新資料庫的記錄資料。
- SQL 語法主要有兩大類，如下所示：
  - 資料定義語言（Data Definition Language，DDL）：建立資料表和定義資料表欄位。
  - 資料操作語言（Data Manipulation Language，DML）：資料表記錄的查詢、插入、刪除和更新。

SQL語言的基礎 - 建立資料表 & 新增資料

- CREATE TABLE 語法可以在資料庫建立資料表，如下所示：

```
CREATE TABLE test(
  _id integer, title text, price real)
```

- SQL 新增記錄 INSERT 語法可以新增一筆記錄。例如在 test 資料表新增一筆記錄的 SQL 語法：

```
INSERT INTO test(title, price)
VALUES ('Access', 800)
```

- SQL 語法的括號中是欄位元清單（不用全部，但需包含所有not null欄位），字串欄位值使用單引號括起，數字沒有。

SQL語言的基礎 - 更新/刪除記錄

- SQL 更新記錄 UPDATE 語法是將資料表內符合條件的記錄，更新其欄位內容，例如：在titles資料表更改書價的SQL語法，如下所示，SQL語法的WHERE條件為書名title欄位，然後使用SET子句更新price欄位資料。

```
UPDATE test SET price =650
WHERE title='Access'
```

- SQL刪除記錄DELETE語法是將資料表內符合條件的記錄刪除掉。例如：在titles資料表刪除記錄的SQL語法，如下所示：

```
DELETE FROM test WHERE title='Access'
```

- SQL語法的WHERE條件為書名title欄位，也就是將符合書名條件的圖書記錄刪除掉。

SQL語言的基礎 - 查詢記錄

- 在SQL語言的資料查詢語法主要為SELECT語法，基本語法如下：

```
SELECT 欄位1, 欄位2,...
FROM 資料表
WHERE conditions
ORDER BY 欄位清單
```

- SELECT 語法的欄位元元1~2為記錄的欄位，conditions 為查詢條件，使用口語來說就是「從資料表取回符合WHERE子句條件的記錄，顯示欄位1和2，並且以ORDER BY子句的欄位來排序」。

- SELECT 語法可以使用"*"符號代表資料表的所有欄位，表示取回資料表記錄的所有欄位：

```
SELECT * FROM titles
```

SQL語言的基礎 - 查詢記錄

- SQL查詢如果是單一條件，在WHERE子句條件的基本規則和範例，如下所示：

  - 文字欄位元元需要使用單引號括起，例如：書名為Python時的SQL語法字串，如下所示：

    ```
    SELECT * FROM titles WHERE title='Python'
    ```

  - 數字欄位元元不需要使用單引號括起，例如：價格為600元的SQL語法，如下所示：

    ```
    SELECT * FROM titles WHERE price=600
    ```

- 文字和備註欄位可以使用【LIKE】包含運算元，只需包含此字串即符合條件，再配合"%"或"_"萬用字元，可以代表任何字串或單一字元，只需包含的子字串就符合條件。例如：查詢擁有 Python 子字串圖書資料的SQL語法，如下所示：

  ```
  SELECT * FROM titles WHERE title LIKE '%Python%'
  ```

- 數字或日期/時間欄位可以使用<>、>、<、>=和<= 等運算元建立多樣化的查詢條件。

SQLite3 模組

Python 2.5 以上的版本預設已有 SQLite 模組，因此毋須安裝。底下為主要常使用的函式：

- sqlite3.connect(database [, timeout, other optional arguments])
- connnection.cursor([cursorClass])
- connection.execute(sql [, optional parameters])
- connection.commit()
- connnection.rollback()
- connection.close()
- cursor.execute(sql [, optional parameters])
- cursor.fetchone()
- cursor.fetchmany([size=cursor.arraysize])
- cursor.fetchall()

SQLite3 模組

- **sqlite3.connect(database [, timeout, other optional arguments])**

  該API打開一個SQLite資料庫檔案database 的連結。如果資料庫成功打開，則返回一個連線物件。Timeout參數表示連接等待鎖定的持續時間，直到發生異常斷開連接。Timeout參數預設是5.0(5秒)。如果給定的資料庫名稱filename不存在，則該呼叫將創建一個資料庫。

- **connnection.cursor([cursorClass])**

  該函式建立一個cursor，將在Python資料庫程式設計中用到。

- **connnection.execute(sql [, optional parameters]**

  該函式執行一個SQL語法。該SQL語法可以被參數化(使用預留位置/格式化方式)。Sqlite3 模組支援預留位置(命名樣式)。例如 cursor.execute("insert into people values (?,?)", (who, age))

SQLite3 模組

- **connection.commit()**

  該函式提交目前的交易。如果您未呼叫該方法，那麼自你上一次呼叫commit()以及所做的任何動作對其它資料庫連接來說是不可見的。

- **connnection.rollback()**

  該函式回復自上一次呼叫commit()以來對資料庫所做的更改。

- connection.close()

  該函式關閉資料閉連接。但要注意的是，這不會自動呼叫commit()。如果您之前未呼叫commit()函式，就直接關閉資料庫連接，您所做的所有更改將全部不會執行。

SQLite3 模組

- **cursor.fetchone()**

  該函式獲取查詢結果集中的下一行，返回一個單一的序列，當沒有更多可用的資料時，則返回None。

- **cursor.fetchmany([size=cursor.arraysize])**

  該函式獲取查詢結果集中的下一行組，返回一個列表。當沒有更多的可用的行時，則返回一個空的列表。讓方法嘗試獲取由size參數指定的盡可能多的行。

- **cursor.fetchall()**

  該函式獲取查詢結果集中所有(剩餘)的行，返回一個列表。當沒有可用的行時，則返回一個空的列表。

SQLite3 模組

- Python內建了 sqlite3 模組，藉由sqlite3模組的函式，我們就可以操作SQLite資料庫，首先我們先建立資料庫的連線。

- 只要匯入 sqlite3 模組, 再以 connect() 函式連接資料庫後，即可建立一個資料庫的連線，如果該資料庫不存在，就會建立一個新的資料庫，如果資料庫已存在，就直接開啟連線，並傳回一個連線 (Connection) 物件。語法如下：

```
import sqlite3
con=sqlite3.connect(資料庫)
con.close()
```

- Connection 物件可提供的函式如下：

函式	說明
execute(SQL語法)	執行SQL語法，可以完成資料表的建立、新增、修改、刪除或查詢動作
cursor()	建立一個cursor物件，利用這個cursor物件的execute() 方法可以完成資料表的建立、新增、修改、刪除或查詢動作
commit()	執行資料庫的更新
close()	關閉資料庫的連線

SQLite3 模組 - cursor()

- connect()函數會建立一個連線物件(本例子名稱為 con)，利用這個連線物件的execute()方法執行SQL命令，就可完成資料表的建立、新增、修改、刪除或查詢動作。

- 由於預設並不會主動更新，必須執行commit()方法資料庫才會變更，程式結束則需以close()方法關閉資料庫。

```
import sqlite3
con=sqlite3.connect('test.sqlite')

# 建立一個資料表
sqlstr = 'CREATE TABLE IF NOT EXISTS table01 \
("num" INTEGER PRIMARY KEY NOT NULL, "tel" TEXT)'
con.execute(sqlstr)

# 新增一筆記錄
sqlstr = 'INSERT INTO table01 VALUES(1, "02-1234567")'
con.execute(sqlstr)

# 新增多筆記錄
sqlstr = '''INSERT INTO table01(num, tel)
            VALUES(2, "03-2345678"),(3, "04-3456789")'''
con.execute(sqlstr)

con.commit() # 主動更新
con.close() # 關閉資料庫連線
```

SQLite3 模組

- 以連線物件的 execute() 函式執行 SQL DQL 語法後，會傳回一個 cursor 物件，它是<sqlite3.cursor object>型別的物件，利用 cursor 物件提供的函式可以進行資料的查詢。

- cursor物件提供下列三個函式：

方法	說明
fetchone()	以數組方式取得資料表符合查詢條件的第一筆記錄資料(row)，如果沒有資料，則傳回None。 cursor.fetchone()：將只取最上面的第一條結果，返回單個數組如('id','name')，然後多次循環使用cursor.fetchone()，依次取得下一條結果，直到為空。
fetchall()	以二維方式(一維：list；二維：tuple) 取得符合查詢條件的資料，若無資料傳回None 返回多個數組，即返回多條記錄(rows),如果沒有結果,則返回()
fetchmany (number_of_records)	此方法接受要提取的記錄數，並返回數組，其中每個記錄本身就是一個數組。 如果沒有更多記錄，則返回一個空數組。

SQLite3 模組 - fetchall()

- 使用 fetchall() 顯示 table01 資料表所有的資料，每一列的資料都是一筆數組資料，可用row[0]、row[1]取得資料表前面兩個欄位。

```
import sqlite3
conn=sqlite3.connect('test.sqlite')
# 建立 cursor 物件並顯示
cursor=conn.execute('select * from table01')
print(cursor)
print()
# 從cursor 物件取出所有資料，並輸出資料
rows= cursor.fetchall()
print(rows)
print()
# 依序輸出資料
for row in rows:
    print("{:d}\t{:s}".format(row[0], row[1]))
conn.close() # 關閉資料庫連線
```

```
<sqlite3.Cursor object at 0x0000026467A51730>

[(1, '02-1234567'), (2, '03-2345678'), (3, '04-3456789')]

1   02-1234567
2   03-2345678
3   04-3456789
```

```
import sqlite3
conn=sqlite3.connect('test.sqlite')

# 建立 cursor 物件並顯示
cur=conn.execute('select * from table01')

# 依序輸出資料
for row in cur:
    print(row)
```

```
(1, '02-1234567')
(2, '03-2345678')
(3, '04-3456789')
```

SQLite3 模組 - fetchone()

- 使用cursor查詢資料-fetchone()
- 使用 fetchone() 顯示 table01 資料表中「num=1」的第一筆資料，傳回的也是一筆數組資料，可用row[0]、row[1]取得資料表前面兩個欄位。

```
import sqlite3
conn=sqlite3.connect('test.sqlite')

# 建立 cursor 物件
cur=conn.execute("SELECT * FROM table01 where num=1")

# 從cur物件中擷取一筆資料 (tuple型態)
row= cur.fetchone()

# 顯示資料
if row!=None:
    print("{}\t{}".format(row[0],row[1]))

# 關閉資料連結
conn.close()
```

```
1    02-1234567
```

SQLite3 範例 - 1

```python
# 匯入所需套件
import sqlite3
# 建立與test.db的連結，如果資料庫成功打開，會返回一個連線物件conn
conn = sqlite3.connect('test.db')
print("Opened database successfully")
# 建立一個cursor物件c
c = conn.cursor()
# 執行SQL語法建立表格，主鍵為整數型態的ID。
# 欄位有文字型態的NAME，整數型態的AGE，字元型態的ADDRESS，實數型態的SALARY
c.execute('''CREATE TABLE COMPANY
   (ID INT PRIMARY KEY NOT NULL,
   NAME TEXT    NOT NULL,
   AGE  INT NOT NULL,
   ADDRESS   CHAR(50),
   SALARY    REAL);''')
print("Table created successfully")
# 確認執行SQL語法與關閉連結資料庫
conn.commit()
conn.close()
```

```
Opened database successfully
Table created successfully
```

Table: COMPANY

ID	NAME	AGE	ADDRESS	SALARY
過濾	過濾	過濾	過濾	過濾

SQLite3 範例 - 2

```python
# 匯入所需套件
import sqlite3
# 建立與test.db的連結，如果資料庫成功打開，會返回一個連線物件conn
conn = sqlite3.connect('test.db')
# 建立一個cursor物件c
c = conn.cursor()
print("Opened database successfully")
# 執行SQL語法建立表格，主鍵為整數型態的ID。
# 欄位有文字型態的NAME，整數型態的AGE，字元型態的ADDRESS，實數型態的SALARY
c.execute('''CREATE TABLE COMPANY
   (ID INT PRIMARY KEY NOT NULL,
   NAME TEXT    NOT NULL,
   AGE  INT NOT NULL,
   ADDRESS    CHAR(50),
   SALARY     REAL);''')
print("Table created successfully")
# 執行SQL語法，從COMPANY表新增紀錄
c.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
 VALUES (1, 'Paul', 32, 'California', 20000.00 )");
c.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
 VALUES (2, 'Allen', 25, 'Texas', 15000.00 )");
c.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
 VALUES (3, 'Teddy', 23, 'Norway', 20000.00 )");
c.execute("INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) \
 VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 )");
# 確認執行SQL語法
conn.commit()
print("Records created successfully")
# 關閉連結資料庫
conn.close()
```

```
Opened database successfully
Table created successfully
Records created successfully
```

Table: COMPANY

	ID	NAME	AGE	ADDRESS	SALARY
	過濾	過濾	過濾	過濾	過濾
1	1	Paul	32	California	20000.0
2	2	Allen	25	Texas	15000.0
3	3	Teddy	23	Norway	20000.0
4	4	Mark	25	Rich-Mond	65000.0

SQLite3 範例 - 3

```python
# 匯入所需套件
import sqlite3
# 建立與test.db的連結，如果資料庫成功打開，會返回一個連線物件conn
conn = sqlite3.connect('test.db')
# 建立一個cursor物件c
c = conn.cursor()
print("Opened database successfully")
# 執行SQL語法，從COMPANY表中獲取並顯示紀錄
cursor = c.execute("SELECT id, name, address, salary  from COMPANY")
for row in cursor:
   print("ID = ", row[0])
   print("NAME = ", row[1])
   print("ADDRESS = ", row[2])
   print("SALARY = ", row[3], "\n")
print("Operation done successfully")
conn.close()
```

```
Opened database successfully
ID =  1
NAME =  Paul
ADDRESS =  California
SALARY =  20000.0

ID =  2
NAME =  Allen
ADDRESS =  Texas
SALARY =  15000.0

ID =  3
NAME =  Teddy
ADDRESS =  Norway
SALARY =  20000.0

ID =  4
NAME =  Mark
ADDRESS =  Rich-Mond
SALARY =  65000.0

Operation done successfully
```

CSV資料存入SQLite資料庫

- 當我們從網頁資料建立成 CSV 字串後，即可將CSV資料存入SQLite資料庫，首先將 CSV字串轉換成串列：

```python
import sqlite3

book = "P0002,Python程式設計,500"
f = book.split(",")

# 建立資料庫連接
conn = sqlite3.connect("Books.sqlite")
# 建立SQL指令INSERT字串
sql = "INSERT INTO Books (id, title, price) VALUES ('{0}','{1}',{2})"
sql = sql.format(f[0], f[1], f[2])
print(sql)
cursor = conn.execute(sql)    # 執行SQL指令
print(cursor.rowcount)
conn.commit() # 確認交易
conn.close()  # 關閉資料庫連接
```

	id	title	price
	過濾	過濾	過濾
1	D0001	Access入門與…	450
2	P0001	C語言程式設計	510
3	P0002	Python程式設計	500

```
INSERT INTO Books (id, title, price) VALUES ('P0002','Python程式設計',500)
1
```

- 程式碼建立資料庫連接後，使用format()函數建立SQL插入記錄的SQL語法字串，在字串中的3個參數值 '{0}', '{1}', '{2}'是對應串列的3個項目。

- 在建立SQL語法字串後，呼叫execute()函數執行新增記錄，rowcount 屬性是影響的記錄數，接著執行commit()函數來真正變更資料庫，執行結果可以新增記錄。

JSON資料存入SQLite資料庫

- 同樣的，我們可以將JSON資料存入SQLite資料庫，首先將JOSN資料轉換成字典：

```python
import sqlite3

d = {
    "id": "P0003",
    "title": "Node.js程式設計",
    "price": 650
}

# 建立資料庫連接
conn = sqlite3.connect("Books.sqlite")
# 建立SQL指令INSERT字串
sql = "INSERT INTO Books (id, title, price) VALUES ('{0}','{1}',{2})"
sql = sql.format(d['id'], d['title'], d['price'])
print(sql)
cursor = conn.execute(sql)    # 執行SQL指令
print(cursor.rowcount)
conn.commit() # 確認交易
conn.close()  # 關閉資料庫連接
```

	id	title	price
	過濾	過濾	過濾
1	D0001	Access入門與…	450
2	P0001	C語言程式設計	510
3	P0002	Python程式設計	500
4	P0003	Node.js程式設計	650

```
INSERT INTO Books (id, title, price) VALUES ('P0003','Node.js程式設計',650)
1
```

- 程式碼建立資料庫連接後，使用 format() 函數建立SQL插入記錄的SQL語法字串，在字串中的3個參數值 '{0}', '{1}', '{2}' 是對應串列的3個項目。

- 在建立SQL語法字串後，呼叫 execute() 函數執行新增記錄，rowcount 屬性是影響的記錄數，接著執行 commit() 函數來真正變更資料庫，執行結果可以新增記錄。

SQLite3 實例 - 帳號密碼管理系統

- 以下範例我們使用SQLite來操作管理帳號和密碼。

```
#自訂函式input_data建立帳號和密碼
def input_data():
    # 建立 cursor 物件
    cursor=conn.cursor()
    # 建立一個資料表
    sqlstr = 'CREATE TABLE IF NOT EXISTS password("name" Text PRIMARY KEY NOT NULL, "pass" TEXT)'
    cursor.execute(sqlstr)

    # 新增一筆記錄
    name=input("請輸入你欲增新的帳號：")
    passwd=input("請輸入你欲增新的密碼：")
    sqlstr = "insert into password values('{}', '{}')".format(name, passwd)
    cursor.execute(sqlstr)
    conn.commit() # 主動更新
    input("新增帳號完成，請按任意鍵返回主選單")
```

SQLite3 實例 - 帳號密碼管理系統

- 以下範例我們使用SQLite來操作管理帳號和密碼。

```
#自定函式menu定義選項功能表
def menu():
    os.system("cls")
    print("帳號、密碼管理系統")
    print("-------------------------")
    print("1. 輸入帳號、密碼")
    print("2. 顯示帳號、密碼")
    print("3. 修　改　密　碼")
    print("4. 刪除帳號、密碼")
    print("0. 結　束　程　式")
    print("-------------------------")

    #自訂函式disp_data顯示所有帳號和密碼
def disp_data():
    cursor = conn.execute('select * from password') #讀取password資料表所有資料，並以cursor傳回
    print("帳號\t密碼")   # 顯示所有cursor資料
    print("================")
    for row in cursor:
        print("{}\t{}".format(row[0],row[1]))
    input("按任意鍵返回主選單")
```

SQLite3 實例 - 帳號密碼管理系統

- 以下範例我們使用SQLite來操作管理帳號和密碼。

```
#輸入新密碼取代舊密碼，並將資料寫回資料庫中
        password=input("請輸入新密碼：")
        sqlstr = "update password set pass='{}' where name='{}'".format(password, name)
        conn.execute(sqlstr)
        conn.commit()
        input("密碼更改完畢，請按任意鍵返回主選單")
        break
#自訂函式delete_data 刪除帳號
def delete_data():
    while True:
        name =input("請輸入要刪除的帳號(Enter==>停止輸入)")
#如果帳號不存在，不允許刪除
        if name=="": break
        sqlstr="select * from password where name='{}'" .format(name)
        cursor=conn.execute(sqlstr)
        row = cursor.fetchone()
        if row==None:
            print("{} 帳號不存在!".format(name))
            continue
        print("確定刪除{}的資料!：".format(name))
#確認刪除後，刪除指定的帳號，並將資料寫回資料庫中
        yn=input("(Y/N)?")
        if (yn=="Y" or yn=="y"):
            sqlstr = "delete from password where name='{}'".format(name)
            conn.execute(sqlstr)
            conn.commit()
            input("已刪除完畢，請按任意鍵返回主選單")
            break
```

SQLite3 實例 - 帳號密碼管理系統

- 以下範例我們使用SQLite來操作管理帳號和密碼。

```
### 主程式從這裡開始 ###

import os,sqlite3 #主程式中import相關的套件

conn = sqlite3.connect('Sqlite01.sqlite') #建立資料庫連線
#依choice的輸入值，執行各項操作
while True:
    menu()
    choice = int(input("請輸入您的選擇："))
    print()
    if choice==1:
        input_data()
    elif choice==2:
        disp_data()
    elif choice==3:
        edit_data()
    elif choice==4:
        delete_data()
    else:
        break

conn.close() #關閉資料庫連線
print("程式執行完畢！")
```

Q & A