07 Table - 外部程式的連結

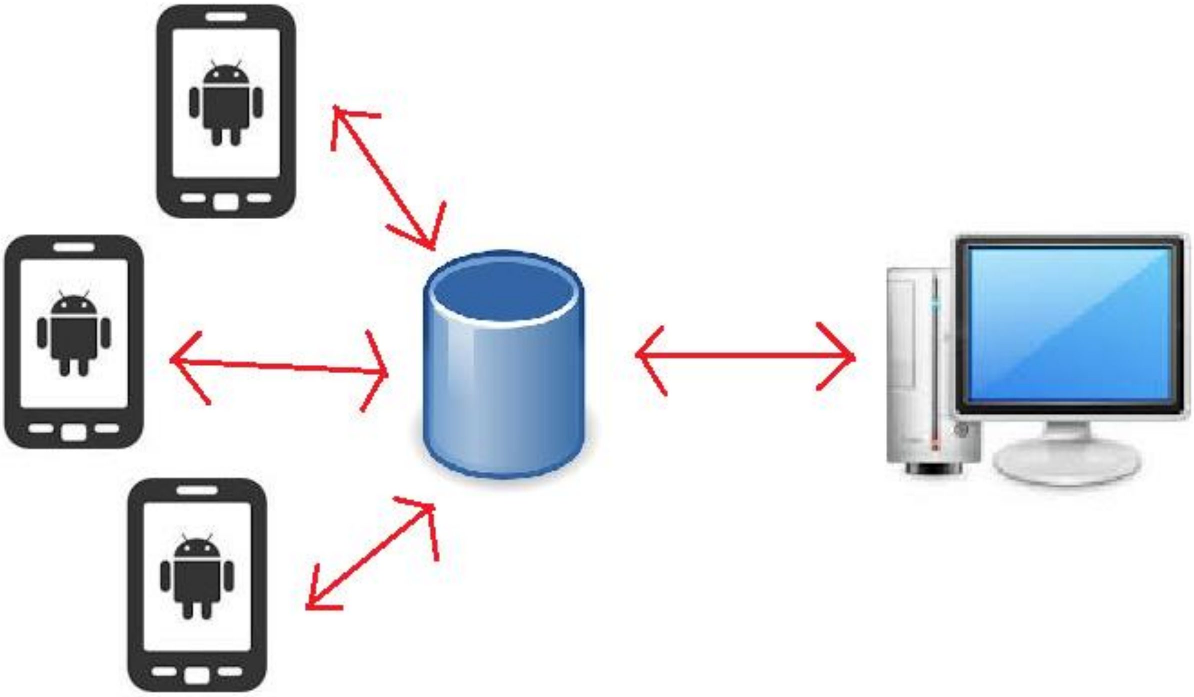
Telung Pan

telung@mac.com

期中測驗方式:

- 採網路線上方式
- 時間:28 日(二) 下午 14:00 - 23:59
- 範圍: 以作業題目為主

外部程式連結資料庫的需求



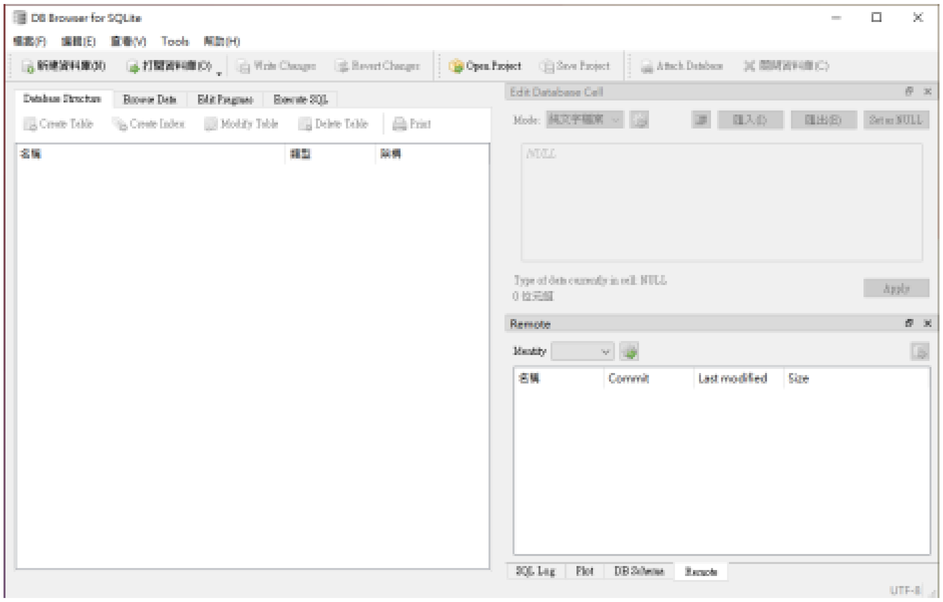
- Application (Windows, Mac)
- Mobile device
- Web

外部程式連結資料庫的方法

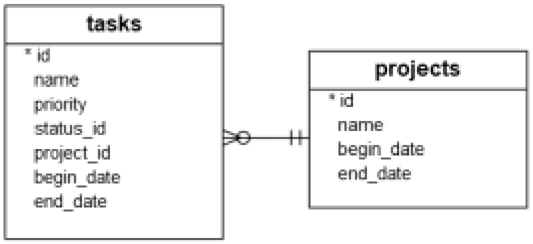
- Middle tier of database or program tools
  - C++ Builder, Delphi IDE
- ODBC
- Import Package

DB Browser for SQLite

DB Browser for SQLLite 網址



SQL Lite 語法範例



The following CREATE TABLE statements create these two tables:

```
-- projects table
CREATE TABLE IF NOT EXISTS projects (
  id integer PRIMARY KEY,
  name text NOT NULL,
  begin_date text,
  end_date text );

 -- tasks table
CREATE TABLE IF NOT EXISTS tasks (
  id integer PRIMARY KEY,
  name text NOT NULL,
  priority integer,
  project_id integer NOT NULL,
  status_id integer NOT NULL,
  begin_date text NOT NULL,
  end_date text NOT NULL,
  FOREIGN KEY (project_id) REFERENCES projects (id) );
```

Let’s see how to create new tables in Python.

First, develop a function called create_connection() that returns a Connection object which represents an SQLite database specified by the database file parameter db_file.

```
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)
    return conn
```

Second, develop a function named create_table() that accepts a Connection object and an SQL statement. Inside the function, we call the execute() method of the Cursor object to execute the CREATE TABLE statement.

```
def create_table(conn, create_table_sql):
    """ create a table from the create_table_sql statement
    :param conn: Connection object
    :param create_table_sql: a CREATE TABLE statement
    :return:
   """     try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)
```

Third, create a main() function to create the projects and tasks tables.

```
def main():     database = r"C:\sqlite\db\pythonsqlite.db"
    sql_create_projects_table = """ CREATE TABLE IF NOT EXISTS projects (
                                        id integer PRIMARY KEY,
                                        name text NOT NULL,
                                        begin_date text,
                                    end_date text
                                ); """
    sql_create_tasks_table = """CREATE TABLE IF NOT EXISTS tasks (
                                id integer PRIMARY KEY,
                                name text NOT NULL,
                                priority integer,
                                status_id integer NOT NULL,
                                project_id integer NOT NULL,
                                begin_date text NOT NULL,
                                end_date text NOT NULL,
                                FOREIGN KEY (project_id) REFERENCES projects (id)
                            );"""

    # create a database connection
    conn = create_connection(database)
    # create tables
    if conn is not None:
        # create projects table
        create_table(conn, sql_create_projects_table)
        # create tasks table
        create_table(conn, sql_create_tasks_table)
    else:
        print("Error! cannot create the database connection.")
```

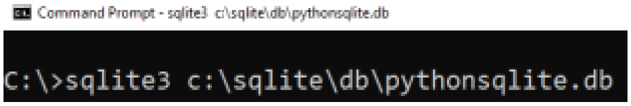
Fourth, execute the main() function.

```
if __name__ == '__main__':     main()
```

完整程式碼範例

```
import sqlite3
from sqlite3 import Error


def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)
     return conn


  def create_table(conn, create_table_sql):
    """ create a table from the create_table_sql statement
    :param conn: Connection object
    :param create_table_sql: a CREATE TABLE statement
    :return:
    """
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)


def main():
    database = r"C:\sqlite\db\pythonsqlite.db"
    sql_create_projects_table = """ CREATE TABLE IF NOT EXISTS projects (
                                        id integer PRIMARY KEY,
                                        name text NOT NULL,
                                        begin_date text,
                                        end_date text
                                    ); """

    sql_create_tasks_table = """CREATE TABLE IF NOT EXISTS tasks (
                                    id integer PRIMARY KEY,
                                    name text NOT NULL,
                                    priority integer,
                                    status_id integer NOT NULL,
                                    project_id integer NOT NULL,
                                    begin_date text NOT NULL,
                                    end_date text NOT NULL,
                                    FOREIGN KEY (project_id) REFERENCES projects (id)
                                );"""

    # create a database connection
    conn = create_connection(database)
    # create tables
    if conn is not None:
        # create projects table
        create_table(conn, sql_create_projects_table)
         # create tasks table
        create_table(conn, sql_create_tasks_table)
    else:
        print("Error! cannot create the database connection.")
  if __name__ == '__main__':     main()
```

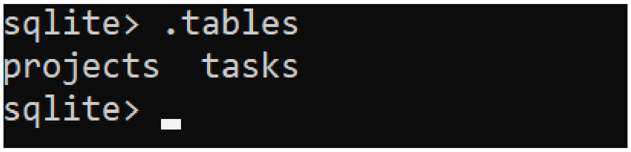
Verify if the program has created those tables successfully in the pythonsqlite.db database.



```
C:\>sqlite3 c:\sqlite\db\pythonsqlite.db
```

Then, use the .tables command to display the tables in the database.

sqlite>.tables projects tasks



```
sqlite> .tables
projects  tasks
sqlite> _
```

As you can see clearly from the output, we are having the projects and tasks tables in the pythonsqlite.db database. And the program works as expected.

In this tutorial, you have learned how to create new tables in the SQLite database using the execute() method of the Cursor object.

案例實作

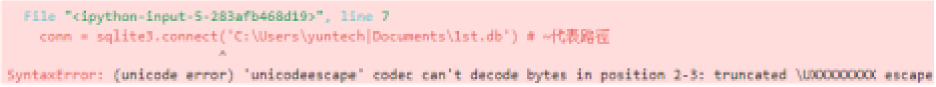
建立資料表及欄位設定

立一個資料表，以 YAHOO_PRICE 替 table 命名，並建立 3 個欄位，分別為：ID (INTEGER)、TITLE(TEXT)、PRICE(INTEGER)

其中 id 請將 PK、AI、U 勾選起來

記得點選Write Changes 儲存剛剛所做的所有變更

使用外部程式連結資料庫的 TABLE



```
  File "<ipython-input-5-283afb468d19>", line 7
    conn = sqlite3.connect('C:\Users\yuntech\Documents\1st.db') # ~代表路徑
                          ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape
```

1 - OCT1

Python 錯誤 - SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape

Python 讀取檔案時出現錯誤訊息:

SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape

原因: 編譯器將路徑中的 C:**\U**sers\ 視為 Unicode-Escape 編碼的跳脫字元，因此\U 被當成 Unicode Code 字串的起點，依照定義後面必需接 8 位數字 (ex. \U01000001) 來 Decode，在此後面接了一串字母，因此產生 Decode 失敗的錯誤訊息。

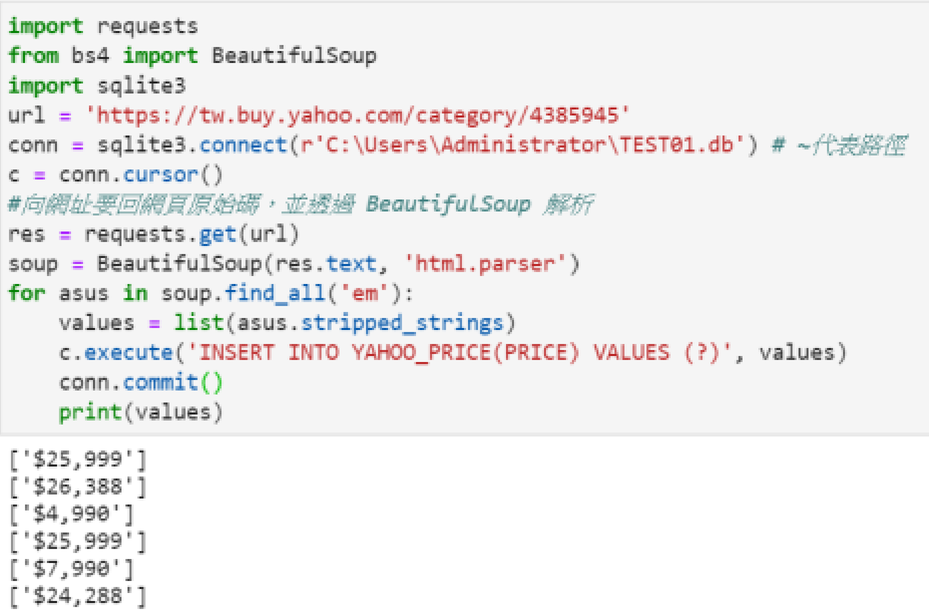
1. 解決方法一:在路徑前加 **r**，使編譯器將整組路徑視為 Raw String，字串中的跳脫字元都被當成一般字元處理。
2. 解決方法二:將路徑中的 \ 取代為 \\ (\\ 也是跳脫字元其意義為反斜線 \ )。
3. 解決方法三:直接改用別的路徑或相對路徑，避開 Windows 預設的\Users。

https://jasonyychiu.blogspot.com/2019/10/python-syntaxerror-unicode-error.html

撰寫互動程式 - 以 Python 為例

- 使用 requests 和 BeautifulSoup 這兩個套件，因此在程式的開頭請先 import 到程式。
- 將網址另外定義為一個 url 變數，然後透過 requests 套件向 url 要回網頁原始碼。
- 透過 BeautifulSoup 解析網頁原始碼。
- 觀察的網頁原始碼，我們可以發現到所需要的資訊都被 class = "item yui3-u srp-multi-image" 所包覆，因此可以透過 soup.find_all() 一次取得所有 class = "item yui3-u srp-multi-image" 的資訊，接著使用 for 迴圈將取得的資訊一個個處理，底下範例透

過 .stripped_strings 將取得的資訊拆解成多個，因此使用 list() 將其包覆，最後使用 if 判斷如果 補貨中 不在 List 中才印出結果



```
import requests
from bs4 import BeautifulSoup
import sqlite3
url = 'https://tw.buy.yahoo.com/category/4385945'
conn = sqlite3.connect(r'C:\Users\Administrator\TEST01.db') # ~代表路徑
c = conn.cursor()
#向網址要回網頁原始碼，並透過 BeautifulSoup 解析
res = requests.get(url)
soup = BeautifulSoup(res.text, 'html.parser')
for asus in soup.find_all('em'):
    values = list(asus.stripped_strings)
    c.execute('INSERT INTO YAHOO_PRICE(PRICE) VALUES (?)', values)
    conn.commit()
    print(values)
```

```
['$25,999']
['$26,388']
['$4,990']
['$25,999']
['$7,990']
['$24,288']
```

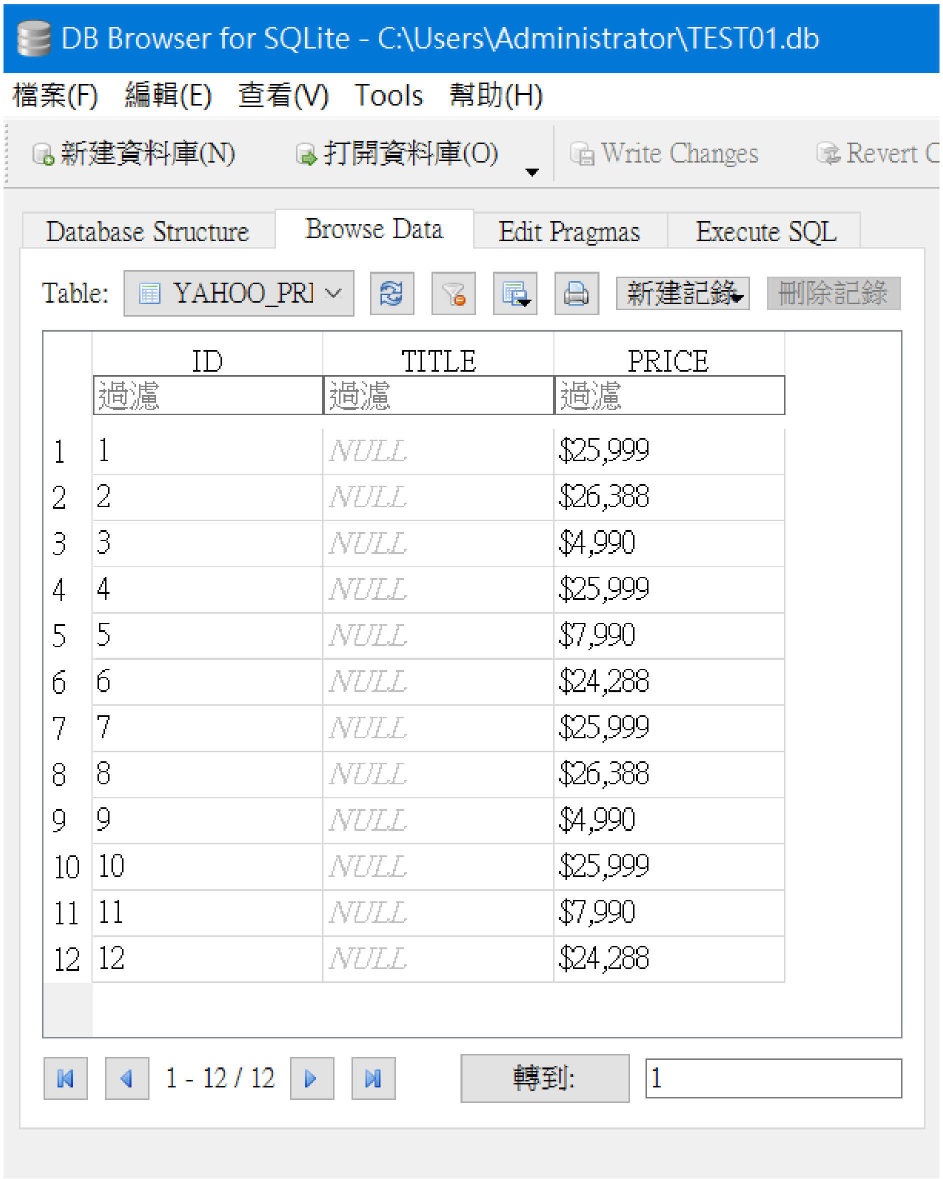
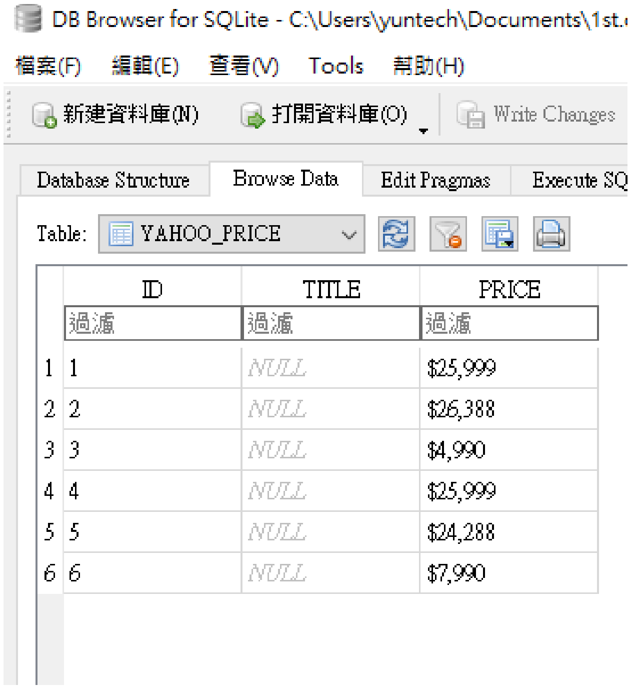


Table: YAHOO_PRI

	ID	TITLE	PRICE
	過濾	過濾	過濾
1	1	NULL	$25,999
2	2	NULL	$26,388
3	3	NULL	$4,990
4	4	NULL	$25,999
5	5	NULL	$7,990
6	6	NULL	$24,288
7	7	NULL	$25,999
8	8	NULL	$26,388
9	9	NULL	$4,990
10	10	NULL	$25,999
11	11	NULL	$7,990
12	12	NULL	$24,288

轉到: 1


DB Browser for SQLite - C:\Users\yuntech\Documents\1st.
檔案(F) 編輯(E) 查看(V) Tools 幫助(H)
新建資料庫(N) 打開資料庫(O) Write Changes
Database Structure Browse Data Edit Pragmas Execute SQ
Table: YAHOO_PRICE

	ID	TITLE	PRICE
	過濾	過濾	過濾
1	1	NULL	$25,999
2	2	NULL	$26,388
3	3	NULL	$4,990
4	4	NULL	$25,999
5	5	NULL	$24,288
6	6	NULL	$7,990

2 - 從網路擷取資料後存入資料庫