

Module 03

LINE Bot 資料庫整合

- 3-1: 整合資料庫
- 3-2: 桌遊店預約系統
- 3-3: 實作桌遊店
LINE Bot

3-1: 整合資料庫

- 創建資料庫
- 於 LINE Bot 中串接資料庫

本介紹資訊取自於 2020/03/03

- 在 Module1-2 中，學習了如何使用 SQLite3 資料庫
- 複習基本使用方法

建立資料表

插入資料

搜尋資料
並印出

```
2 import sqlite3
3
4 # 連結資料表
5 connect = sqlite3.connect('Lab_4-2.db')
6 cursor = connect.cursor()
7
8 # 建立資料表
9 cursor.execute("CREATE TABLE grades \
10                (ID integer primary key, \
11                 Name varchar(10), \
12                 Chinese integer, \
13                 English integer, \
14                 Pass bit)")
15 connect.commit()
16
17 # 插入資料至資料表中
18 cursor.execute("INSERT INTO grades (Name, Chinese, English, Pass) \
19                VALUES ('Amy', 90, 85, 1)")
20 connect.commit()
21
22 # 印出目前資料表中資料
23 grades_list = cursor.execute("SELECT * FROM grades")
24 for grade in grades_list:
25     print(grade)
```

- 假設情境為：建立資料表紀錄用戶與官方帳號的對話
- 創建資料表，儲存...
 - 用戶姓名 (name)
 - 用戶 ID (user_id)
 - 訊息內容 (msg)

id	name	user_id	msg
1	小明	123abc	您好
2	小明	123abc	我想要預約



- 假設情境為：建立資料表紀錄用戶與官方帳號的對話
- 創建資料表，儲存...
 - 用戶姓名 (name)
 - 用戶 ID (user_id)
 - 訊息內容 (msg)

id	name	user_id	msg

```
1 import sqlite3
2
3 # 連結至資料庫 history.db
4 connect = sqlite3.connect('history.db')
5
6 # 建立cursor，以利後續操作
7 cursor = connect.cursor()
8
9 # 建立資料表
10 cursor.execute("CREATE TABLE log \
11                (id integer primary key, \
12                 name varchar(20), \
13                 user_id text, \
14                 msg text)")
15 connect.commit()
```

建立資料表

- 假設情境為：建立資料表紀錄用戶與官方帳號的對話
- 於 Line Bot 接收訊息後儲存

```
47 # handle msg
48 @handler.add(MessageEvent, message=TextMessage)
49 def handle_message(event):
50     # get user info & message
51     user_id = event.source.user_id
52     msg = event.message.text
53     user_name = line_bot_api.get_profile(user_id).display_name
54
55     # save data to db
56     cursor.execute("INSERT INTO log (name, user_id, msg) \
57                     VALUES (?, ?, ?)", (user_name, user_id, msg))
58     connect.commit()
59     line_bot_api.reply_message(event.reply_token,
60                               TextSendMessage(text = '已收到您的訊息！'))
```

解析 event 資訊

插入資訊
進資料表

- 假設情境為：建立資料表紀錄用戶與官方帳號的對話
- 當用戶輸入關鍵字：「想看歷史訊息」
 - 回覆目前資料表中所有內容

```
if msg == '想看歷史訊息':
```

```
    logs = cursor.execute("SELECT * FROM log")
```

```
    log_text = ''
```

```
    for log in logs:
```

```
        log_text = log_text + '\n' + str(log)
```

```
    line_bot_api.reply_message(event.reply_token,  
                               TextSendMessage(text = log_text))
```

取出目前資料表中內容

將取出資料串接為字串

- Run Demo_3-1-1.py 範例程式碼
 - 建立資料表
- Run Demo_3-1-2.py 範例程式碼
 - 啟用 Line Bot
- 於 Line 中傳送多則訊息至此官方帳號
- 測試傳送訊息：「想看歷史訊息」

3-2: 桌遊店預約系統

- 模擬情境
- 實作預約系統

- 桌遊店提供多種桌遊遊戲，供顧客於當日預約
- 當用戶傳送訊息：「想看遊戲種類」
 - 回覆用戶尚未被預約的遊戲名稱
- 當用戶傳送訊息：「想預約 xxx」
 - xxx 為遊戲名稱
 - 若此遊戲無人預約，則回覆「已為您預約完成：xxx」
 - 若此遊戲已被預約，則回覆「此遊戲已被預約！抱歉」
 - 若無此款遊戲：則回覆「並沒有此款遊戲」
- 當用戶傳送訊息：「想取消預約」
 - 自動搜尋該名用戶預約之遊戲，取消預約且回覆「已為您取消預約」

- 整體程式架構

```
7 # run for 100 times
8 for i in range(1, 100):
9
10     msg = input('輸入訊息：')
11     current_user_id = '123abc'
12
13     # implement reservation
14     if msg == '想看遊戲種類':
15         # ...
16     elif msg[0:3] == '想預約':
17         # ...
18     elif msg == '想取消預約':
19         # ...
20     elif msg == '結束':
21         break
22     else:
23         print('您好！請問需要什麼樣的服務？')
```

模擬用戶資訊
及訊息

結束條件

- 建立遊戲列表

```
2 # create games dictionary
3 games = {'小瓦隆' : None,
4          '雨聲' : None,
5          '龍捲風' : None}
```



桌遊名稱	預約人 ID
小瓦隆	None
雨聲	None
龍捲風	None

- 列出遊戲種類

- 找出尚未被預約的遊戲：user_id 尚未被填寫
- 將未被預約的遊戲印出

語法補充

```
A = {'key': value}
for key, value in A.items():
    print(key, value)
```

```
if msg == '想看遊戲種類':
    game_list = []
    for game, user_id in games.items():
        if user_id is None:
            game_list.append(game)
    game_text = '尚可預約遊戲：\n'
    for game in game_list:
        game_text = game_text + game + '\n'
    print(game_text)
```

遍歷遊戲列表
檢查尚未被預約的遊戲

輸入訊息：想看遊戲種類
尚可預約遊戲：
小瓦隆
雨聲
龍捲風

將取出資料串接為字串

測試（無遊戲被預約時）

- 預約遊戲

- 擷取用戶傳送之訊息，取得遊戲名稱
- 搜尋此遊戲是否於遊戲列表中
- 確認其預約狀態，若無人預約則將其預約 / 若有人預約則回傳已被預約訊息

```
elif msg[0:3] == '想預約':  
    game_name = msg[3:]  
    found = False  
    for game, user_id in games.items():  
        if game == game_name:  
            found = True  
            if user_id is None:  
                games[game_name] = current_user_id  
                print('已為您預約完成:', game)  
            else:  
                print('此遊戲已被預約! 抱歉')  
    if found == False:  
        print('並沒有此款遊戲')
```

擷取遊戲名稱

遍歷遊戲列表
搜尋欲預約之遊戲
檢查是否已被預約

輸入訊息：想預約雨聲
已為您預約完成： 雨聲

輸入訊息：想看遊戲種類
尚可預約遊戲：
小瓦隆
龍捲風

測試（雨聲被預約時）

- 取消預約
 - 搜尋是否用戶 ID 存在於遊戲列表中
 - 若存在則將該遊戲之 ID 欄位重設為 None

```
elif msg == '想取消預約':  
    for game, user_id in games.items():  
        if user_id == current_user_id:  
            games[game] = None  
            print('已為您取消預約')
```

遍歷遊戲列表
檢查 ID 是否與當前用戶 ID 相同
若相同則設為 None

輸入訊息：想預約雨聲
已為您預約完成： 雨聲

輸入訊息：想看遊戲種類
尚可預約遊戲：

小瓦隆
龍捲風

輸入訊息：想取消預約
已為您取消預約

輸入訊息：想看遊戲種類
尚可預約遊戲：

小瓦隆
雨聲
龍捲風

- Run Demo_3-2.py 範例程式碼
- 測試所有流程
- 了解所有需要判斷與檢查的部分

3-3: 實作桌遊店 Line Bot

- 建立資料庫
- 連結資料庫於 Line Bot
- 實作預約系統於 Line Bot

- 依照模擬情境，建立所需資料庫與資料表並將遊戲填入

```
1 import sqlite3
2
3 # 連結至資料庫 game.db
4 connect = sqlite3.connect('game.db')
5
6 # 建立cursor，以利後續操作
7 cursor = connect.cursor()
8
9 # 建立資料表
```

```
10 cursor.execute("CREATE TABLE reservation \
11                 (id integer primary key, \
12                  game text, \
13                  user_id text)")
14 connect.commit()
```

建立資料表

```
16 # 填入遊戲
```

```
17 cursor.execute("INSERT INTO reservation (game) \
18                 VALUES ('小瓦隆')")
19 cursor.execute("INSERT INTO reservation (game) \
20                 VALUES ('雨聲')")
21 cursor.execute("INSERT INTO reservation (game) \
22                 VALUES ('龍捲風')")
23 connect.commit()
```

插入資料

桌遊名稱	預約人 ID
小瓦隆	NULL
雨聲	NULL
龍捲風	NULL

- SQL中的「無」是NULL
- Python中的「無」是None

- 系統所有行為都是使用者先傳訊，Bot被動式回覆
 - （觸發後續行為）列出遊戲種類、預約遊戲、取消預約
- 列出遊戲種類
 - 使用 SQL 語法 SELECT 進行篩選
 - 找出NULL的地方，代表無資料（沒被預約）

```
if msg == '想看遊戲種類':
```

找出尚未被預約的遊戲

```
    games = cursor.execute("SELECT * FROM reservation WHERE user id IS NULL")
```

```
    game_text = '尚可預約遊戲：\n'
```

```
    for game in games:
```

```
        game_text = game_text + game[1] + '\n'
```

將取出資料串接為字串

```
    line_bot_api.reply_message(event.reply_token,  
                               TextSendMessage(text = game_text))
```

字串操作 A = '想預約小瓦隆'

A[3] 為 '小'

A[3:] 為 '小瓦隆'

A[:5] 為 '想預約小瓦'

A[3:5] 為 '小瓦'

- 預約遊戲

```
elif msg[0:3] == '想預約':
```

擷取遊戲名稱

```
    game_name = msg[3:]
```

```
    games = cursor.execute("SELECT * FROM reservation")
```

取出所有資料

```
    found = False
```

```
    for game in games:
```

```
        if game[1] == game_name:
```

```
            found = True
```

```
            if game[2] is None:
```

```
                cursor.execute("UPDATE reservation \
                                SET user_id = ? \
                                WHERE game = ? ", (user_id, game_name))
```

```
                connect.commit()
```

```
                line_bot_api.reply_message(event.reply_token,
                                              TextSendMessage(text = '已為您預約完成：' + game_name))
```

```
            else:
```

```
                line_bot_api.reply_message(event.reply_token,
                                              TextSendMessage(text = '此遊戲已被預約！抱歉'))
```

```
    if found == False:
```

```
        line_bot_api.reply_message(event.reply_token,
                                    TextSendMessage(text = '並沒有此款遊戲'))
```

遍歷遊戲列表

檢查欲預約之遊戲是否存在

若存在則檢查是否被預約

- 取消預約

搜尋是否用戶 ID 存在於遊戲列表中

```
elif msg == '想取消預約':
```

```
    games = cursor.execute("SELECT * FROM reservation where user_id = ?", (user_id,))
```

```
    canceled_game_name = []
```

```
    found = False
```

```
    for game in games:
```

```
        canceled_game_name.append(game[1])
```

儲存要取消預約的遊戲名稱

```
    for game in canceled_game_name:
```

```
        cursor.execute("UPDATE reservation \
```

```
                        SET user_id = ? \
```

```
                        WHERE game = ? ", (None, game))
```

```
    connect.commit()
```

```
    found = True
```

將要取約遊戲的user_id設為 None
(但在SELECT的時候要用NULL)

```
    if found == True:
```

```
        line_bot_api.reply_message(event.reply_token,
```

```
                                   TextSendMessage(text = '已為您取消預約'))
```

- Run Demo_3-3-1.py 範例程式碼
 - 建立資料表
- Run Demo_3-3-2.py 範例程式碼
 - 啟用 Line Bot
- 於 Line 中傳送訊息至此官方帳號
- 測試所有流程

- 題目1：整合資料庫
 - 將用戶資訊與訊息存至資料庫
 - 設定關鍵字「秀xxx」， xxx 為用戶名稱
 - 收到後回覆資料庫中所有來自 xxx 的訊息
- 題目2：實作拍賣系統
 - 模擬情境在 Assignment_6-2.py 中
 - 利用變數儲存資料，達成模擬情境
- 題目 3：於LINE Bot 實作拍賣系統
 - 模擬情境在 Assignment_6-2.py 中
 - 利用資料庫儲存資料，實作於 Line Bot 之中

- 在 Line Bot 接收訊息後儲存資訊至資料庫
- 在 SQL 中塞入單個、多個變數的方法
- 設計預約系統流程
- 將預約系統套用到 Line Bot 中



- Lab01: 整合資料庫
- Lab02: 桌遊店預約系統
- Lab03: 實作桌遊店 Line Bot

Estimated time:
20 minutes

