

# Chapter 3

## 用python讀寫資料檔/資料庫

### Reading and Writing Data with Python

~~※本教材不是以簡報為唯一目的，視為講義會更好！  
為了方便初學者把握住重點，快速學習。

所以會補充較多字體較小的相關說明，請細細品嘗~~

Instructor: Jiun-Ting Jiang

Email: [rjrj0510@gmail.com](mailto:rjrj0510@gmail.com)

上課，儘量往前座哦！

*Dept. of Computer Science and Information Engineering,  
Tamkang University*

# Outline

- 3-0-1: 政府資料開發平台(open data)
- 3-0: File，關於普通檔案的開檔、寫檔或讀檔、關檔。
- 3-1: 建立一個基本的資料檔案(dataset資料檔案)
  - 3-1-1: 快速製作數據檔來測試
  - 3-1-2: 讓簡單資料可在數據之間移動 (Passing Data Around)
  - 3-1-3: 建立一個簡單的資料檔案CSV-以諾貝爾獎得主為例(nobel)
- 3-2: 建立基本數據資料檔案\_CSV系列
- 3-3: JSON
- 3-4: SQL--SQLite:
  - 用SQLAlchemy將數據寫入SQLite (Creat/Read/Update/ Delete)
- 3-5: 「懶人數據工具」Dataset工具
- 3-6: NoSQL—MongoDB
- 3-7: 處理特殊情況，在python與JavaScript之間的問題
- 3-8: 補充：爬蟲相關---(1)爬JSON數據(2)爬CSV數據

# 3-0-1:政府資料開發平台(open data)

以csv格式為主。

也有非即時的格式轉換。

查詢資料集: 不動產實價登錄資訊 x

data.gov.tw/datasets/search?q=不動產實價登錄資訊-買賣案件-中和區

政府資料開放平台

全部資料集 網站導覽 互動專區 資料故事館 最新消息 諮詢小組 關於平台 EN 登入/註冊

資料集列表 | Datasets

篩選條件: x 不動產實價登錄資訊-買賣案件-中和區

排序方式: 上架日期 新至舊

共1筆, 本頁顯示1-1筆

不動產實價登錄資訊-買賣案件-中和區

1. 不動產買賣案件實價登錄資訊, 包含標的位置(去識別化)、面積、總價等資訊。  
2. 本資料集為每10日更新一次。-中和區

主要欄位說明:

district(鄉鎮市區)、rps01(交易標的)、rps02(土地區段位置建物區段門牌)、rps03(土地移轉總面積平方公尺)、rps04(都市土地使用分區)、rps05(非都市土地使用分區)、rps06(非都市土地使用編定)、rps07(交易年月日)、rps08(交易筆棟數)、rps09(移轉層次)、rps10(總樓層數)、rps11(建物型態)、rps12(主要用途)、rps13(主要建材)、rps14(建築完成年月)、rps15(建物移轉總面積平方公尺)、rps16(建物現況格局-房)、rps17(建物現況格局-廳)、rps18(建物現況格局-衛)、rps19(建物現況格局-隔間)、rps20(有無管理組織)、rps21(總價元)、rps22(單價元平方公尺)、rps23(車位類別)、rps24(車位移轉總面積平方公尺)、rps25(車位總價元)、rps26(備註)、rps27(編號)、rps28(主建物面積)、rps29(附屬建物面積)、rps30(陽台面積)、rps31(電梯)

新北市政府地政局 / 詮釋資料更新時間: 2020/09/22 08:52

CSV

共1筆, 本頁顯示1-1筆

匯出搜尋結果清單: csv xml json

資料集 互動專區 資料故事館 統計資料 諮詢小組 規範&授權

# csv轉其他格式

- 不動產實價登錄資訊-買賣案件-中和區
- 本資料,每10日會更新csv最新資料。目前2024年
- 而且，轉為其他格式：如：CSV, JSON, XML,
- 少部份有XLSX,ODS

data.gov.tw/dataset/146474

政府資料開放平台 DATA.GOV.TW

網站導覽

Language

資料集 高應用價值主題專區 資料故事館 互動專區 消息專區 諮詢小組 授權條款

## 不動產實價登錄資訊-買賣案件-中和區

1. 不動產買賣案件實價登錄資訊，包含標的位置、面積、總價等資訊。2. 本資料集為每10日更新一次。(中和區)

評分此資料集：  
☆☆☆☆☆  
平均 0.00 (0 人次投票)

瀏覽次數：529 下載次數：25 意見數：0

主要欄位說明 *粗體欄位為資料標準欄位	district(鄉鎮市區)、rps01(交易標的)、rps02(土地區段位置建物區段門牌)、rps03(土地移轉總面積平方公尺)、rps04(都市土地使用分區)、rps06(非都市土地使用編定)、 <b>rps07(交易年月日)</b> 、rps08(交易筆棟數)、rps09(移轉層次)、rps10(總樓層數)、rps11(態)、rps12(主要用途)、rps13(主要建材)、rps14(建築完成年月)、rps15(建物移轉總面積平方公尺)、 <b>rps16(建物現況格局-房)</b> 、rps17(廳)、 <b>rps18(建物現況格局-衛)</b> 、rps19(建物現況格局-隔間)、rps20(有無管理組織)、rps21(總價元)、 <b>rps22(單價元平方公尺)</b> 、rps23(別)、 <b>rps24(車位移轉總面積平方公尺)</b> 、 <b>rps25(車位總價元)</b> 、 <b>rps26(備註)</b> 、 <b>rps27(編號)</b> 、rps28(主建物面積)、rps29(附屬建物面積)、rps31(電梯)
資料資源下載網址	<a href="#">↓ CSV</a> <a href="#">檢視資料</a> 1. 不動產買賣案件實價登錄資訊，包含標的位置、面積、總價等資訊。2. 本資料集為每10日更新一次。...
提供機關	新北市政府地政局
提供機關聯絡人姓名	平小姐 (3368)
更新頻率	每10日
授權方式	政府資料開放授權條款-第1版
計費方式	免費
上架日期	2020-04-24
資料集上架方式	系統介接程式
詮釋資料更新時間	2024-10-04 10:35
服務分類	購屋及遷徙

進入檢視，可見到JSON格式的網址

# 3-0: python的file的開檔、寫檔或讀檔、關檔 (檔案文件的基本操作)

- 使用內建函數open打開文件檔案，該函數返回檔案的物件。
- 例如，命令`fp = open('sample.txt')`嘗試打開名為sample.txt的文件檔案。
- 文件檔案使用方法：

read()是讀完全部內容

read( k )是讀k個byte

產生文字檔的簡單三步驟  
(1)開檔:open「可寫入」檔  
(2)使用檔案: Print轉向，  
(3)關檔:close

以下是最簡單，逐行寫資料的方法：

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> import sys
>>> fp = open('c:\\in.txt', 'w')
>>> print("aaa,aaa2,aaa3", file=fp)
>>> print("bbb   bbb2   bbb3", file=fp)
>>> print("ccc ccc2 ccc3", file=fp)
>>> print("ddd ddd2,ddd3", file=fp)
>>> fp.close()
>>>
```

Calling Syntax	Description
<code>fp.read()</code>	Return the (remaining) contents of a readable file as a string.
<code>fp.read(k)</code>	Return the next $k$ bytes of a readable file as a string.
<code>fp.readline()</code>	Return (remainder of) the current line of a readable file as a string.
<code>fp.readlines()</code>	Return all (remaining) lines of a readable file as a list of strings.
<code>for line in fp:</code>	Iterate all (remaining) lines of a readable file.
<code>fp.seek(k)</code>	Change the current position to be at the $k^{th}$ byte of the file.
<code>fp.tell()</code>	Return the current position, measured as byte-offset from the start.
<code>fp.write(string)</code>	Write given string at current position of the writable file.
<code>fp.writelines(seq)</code>	Write each of the strings of the given sequence at the current position of the writable file. This command does <i>not</i> insert any newlines, beyond those that are embedded in the strings.
<code>print(..., file=fp)</code>	Redirect output of print function to the file.

# 【check1】 ch03實作練習1\_Lab#3-0a-FileOpen逐行讀資料的方法

- 有很多方法，建議是有with的open，就不必try，也不必管close()

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

>>> fp = open('c:\\in.txt')
>>> line=fp.readline()
>>> while line:
>>>     print(line, end='')
>>>     line=fp.readline()

aaa,aaa2,aaa3
bbb bbb2 bbb3
ccc ccc2 ccc3
ddd ddd2,ddd3

>>> fp = open('c:\\in.txt')
>>> for line in fp.readlines():
>>>     print(line, end='')

aaa,aaa2,aaa3
bbb bbb2 bbb3
ccc ccc2 ccc3
ddd ddd2,ddd3
>>> |
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

>>> fp = open('c:\\in.txt')
>>> print(fp.read())
aaa,aaa2,aaa3
bbb bbb2 bbb3
ccc ccc2 ccc3
ddd ddd2,ddd3
```

read()是讀完全部內容

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help

>>> fp=open('c:\\in.txt')
>>> for line in fp:
>>>     print(line, end='')

aaa,aaa2,aaa3
bbb bbb2 bbb3
ccc ccc2 ccc3
ddd ddd2,ddd3

>>> with open('c:\\in.txt') as fp:
>>>     for line in fp:
>>>         print(line, end='')

aaa,aaa2,aaa3
bbb bbb2 bbb3
ccc ccc2 ccc3
ddd ddd2,ddd3
>>>
```

這個寫法是高效、快速又簡單的程式寫法

# Lab#3-0b-FileOpen\_逐行寫資料的方法

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> import sys
>>> fp = open('c:\\in.txt', 'w')
>>> print("aaa,aaa2,aaa3", file=fp)
>>> print("bbb bbb2 bbb3", file=fp)
>>> print("ccc ccc2 ccc3", file=fp)
>>> print("ddd ddd2,ddd3", file=fp)
>>> fp.close()
>>>
```

## Lab#3-0b-FileOpen\_逐行寫資料的方法

In [101]:

```
1 cols=['xx1','yy2','zz3']
2 ss=', '.join(cols)
3 with open('c:\\in.txt', 'w') as f:
4     f.write(ss + '\n')
5     f.write('=====\n')
6     f.write(cols[1] + '\n')
7     f.write('=====')
8
9 ##用方法三印出內容、●● 這是配合 with 的高效率寫法！
10 with open('c:\\in.txt') as fp:
11     for line in fp:
12         print(line, end='')

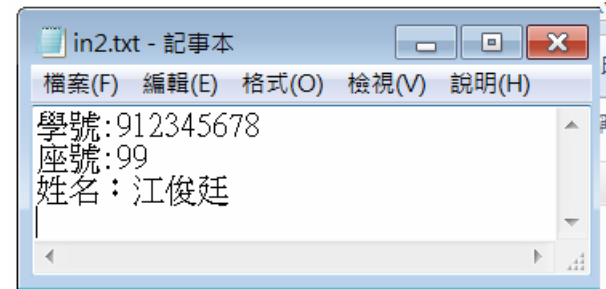
```

```
xx1,yy2,zz3
=====
yy2
=====
```



# Exercises: Ex3-0

- Ex3.0 Add1:請問要產生簡單文字檔的  
(a)簡單三步驟是什麼？(b)如果要在  
文字檔中，寫入三行資料，是自己的  
學號、座號、姓名(如右圖)。  
可執行的完整程式要如何寫？



- Ex3.0 Add2:如果有一個文字檔，檔名是c:\\in.txt. 請問可以一行一行逐步讀入資料的程式要如何寫？
- Ex3.0 Add3: python建議開檔是用with取代open主要好處有那些？
- Ex3.0Add4:請問在政府資料開發平台(open data)所提供的資料格式以什麼格式為主？請問該網站的資料更新週期是多久？該網站有提供其他格式的轉換，請舉出至少三種。
- =====以下為補充=====
- Ex3.0 Add5:檔案，(a)可分兩類，第一類是文字文件(例如utf-8格式)，第二類是二進位檔式的binary. 請問在開檔上，如何分辨？(b)二進位檔要如何設定位置？如何查知目前位置？(c)請舉實例簡單示範binary的操作。
- 作業題： Ex3.0Add1 ~~
- 補充：



- **3-1: 建立一個基本的資料檔案(dataset資料檔案)**
  - 3-1-1: 快速製作數據檔來測試
  - 3-1-2: 讓簡單資料可在數據之間移動 (Passing Data Around)
  - 3-1-3: 建立一個簡單的資料檔案CSV-以諾貝爾獎得主為例 (nobel)

# 3-1-1: 快速製作數據檔來測試

- 任何數據可視化工具基本技能之一就是「能夠移動數據」
- 主要放數據的地方：
  - 數據庫：資料庫(databases) 包括SQL與 NoSQL
  - 數據檔：檔案格式(file formats ) 包括JavaScript物件表示法格式(JSON)與逗號分隔值格式(CSV, Comma-separated values) 或其他更複雜的格式。
- 對數據處理人員而言，都應該都可以輕鬆讀取數據進行轉換並將其寫成更方便的形式。
- Python的一項強項就是可以快速處理各類數據，這是快速掌握Dataviz的基本工具鏈。
- 以前用像C這樣的低級語言進行編程時，要處理大量數據會很辛苦。光是讀取和寫入文件，就有很多令人討厭的代碼。再說，要從數據庫中讀取，也很困難，並且對於序列化數據(serializing data)的處理，仍然要很辛苦的放在記憶體中。
- 相對於Python，不論打開文件或處理數據，都簡單很多。最簡單的讀檔指令：`f = open('data.txt')`

# CSV/JSON, 與 SQL/NoSQL

- Python內建很多模塊來處理數據：
  - 關於檔案：CSV格式和JSON格式的文件已內建有很豐富的支援。
  - 關於數據庫：
    - (第1類)SQL資料庫--關聯式資料庫(Relational database)：
      - 可用標準的資料結構化查詢語言(SQL, Structured Query Language) 結構化查詢語言),對關聯式資料庫中資料的檢索和操作。python有一些很棒的庫可以與SQL數據庫進行交互，例如SQLAlchemy，是不錯的入門指南。
    - (第2類)NoSQL非關聯式資料庫(Not Only SQL)：
      - 數據儲存可以不需要固定的表格模式，也可認為就是「無綱要SQL (No Schema SQL)」(或「無表格架構資料庫」)
      - MongoDB是一種較新的NoSQL資料庫。配合Python的pymongo庫，適合新手。

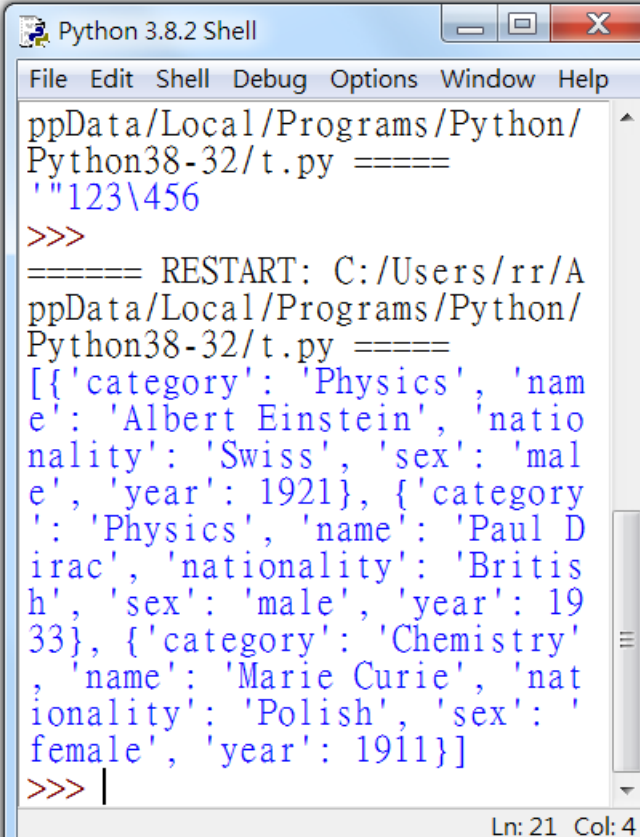
## 3-1-2: 讓簡單資料可在數據之間移動 (Passing Data Around)

- 後續要展示的是：
  - 「傳遞一個數據資料包，並在進行過程中對其進行讀和寫。」
- 在過程中要實際了解數據可視化人員使用的
  - 關鍵數據格式(key data formats)
  - 和數據庫(databases)。
- 要傳遞的數據是網路可視化中最常用的數據
  - 它是類似字典的物件(dictionary-like objects)的列表(請參見示例3-1)
  - 該數據集將以JSON格式傳輸到瀏覽器，
  - 這個格式是很容易由Python字典進行轉換。

### 3-1-3: 建立一個簡單的資料檔案CSV-以諾貝爾獎得主為例(nobel)

#Example 3-1. Our target list of data objects

```
nobel_winners = [  
    {'category': 'Physics',  
     'name': 'Albert Einstein',  
     'nationality': 'Swiss',  
     'sex': 'male',  
     'year': 1921},  
    {'category': 'Physics',  
     'name': 'Paul Dirac',  
     'nationality': 'British',  
     'sex': 'male',  
     'year': 1933},  
    {'category': 'Chemistry',  
     'name': 'Marie Curie',  
     'nationality': 'Polish',  
     'sex': 'female',  
     'year': 1911}  
]  
print(nobel_winners)
```



```
Python 3.8.2 Shell  
File Edit Shell Debug Options Window Help  
ppData/Local/Programs/Python/  
Python38-32/t.py =====  
"123\\456  
>>>  
===== RESTART: C:/Users/rr/A  
ppData/Local/Programs/Python/  
Python38-32/t.py =====  
[{'category': 'Physics', 'nam  
e': 'Albert Einstein', 'natio  
nality': 'Swiss', 'sex': 'mal  
e', 'year': 1921}, {'category  
': 'Physics', 'name': 'Paul D  
irac', 'nationality': 'Britis  
h', 'sex': 'male', 'year': 19  
33}, {'category': 'Chemistry'  
, 'name': 'Marie Curie', 'nat  
ionality': 'Polish', 'sex': '  
female', 'year': 1911}]  
>>> |  
Ln: 21 Col: 4
```

# Lab# 3-1-3 實作nobel的CSV檔

■ (行號1)從csv檔的第一列資料，取出當作物件的鍵（即“category”，“name” ...）

■ (行號3)使用Python的with語句來確保文件在離開該代碼塊或發生任何異常時均已關閉。

■ (行號4) join目的是把列表（此處為cols）轉為字串，並用,串接。即“category,name, ....”。當作key。

■ (行號6)使用nobel\_winners中物件的列(column)，建立list，然後，就可在行號7再用join轉為字串，即可寫到csv檔。

```
1 cols = nobel_winners[0].keys()
2 ### sorted(cols) sorted by keys 此時應不需排序。所以省略
3 with open('./nobel_winners.csv', 'w') as f:
4     f.write(','.join(cols) + '\n')
5     for o in nobel_winners:
6         row = [str(o[col]) for col in cols]
7         f.write(','.join(row) + '\n')
```

```
1 ##用方法三印出內容、●● 這是配合 with 的高效率寫法！
2 with open('./nobel_winners.csv', "r") as fp:
3     for line in fp:
4         print(line, end='')
5 print('-'*60)
6 print('上方是文字檔的內容')
7 print('-'*60)
```

```
category,name,nationality,sex,year
Physics,Albert Einstein,Swiss,male,1921
Physics,Paul Dirac,British,male,1933
Chemistry,Marie Curie,Polish,female,1911
```

上方是文字檔的內容

## ■ CSV 檔：

- 用逗點分隔值（**csv, Comma-Separated Values**）是一種簡單的文字檔格式，以逗號分隔不同欄位的資料，很多軟體在儲存與交換表格資料時都支援這樣的格式。
- 第一列會是欄位的名稱。第二列開始就全部都是資料。

- 在 **Python** 中若要讀取或產生 **csv** 的檔案，可直接用內建 **csv** 模組，更方便，以下是使用方式以及範例程式碼。

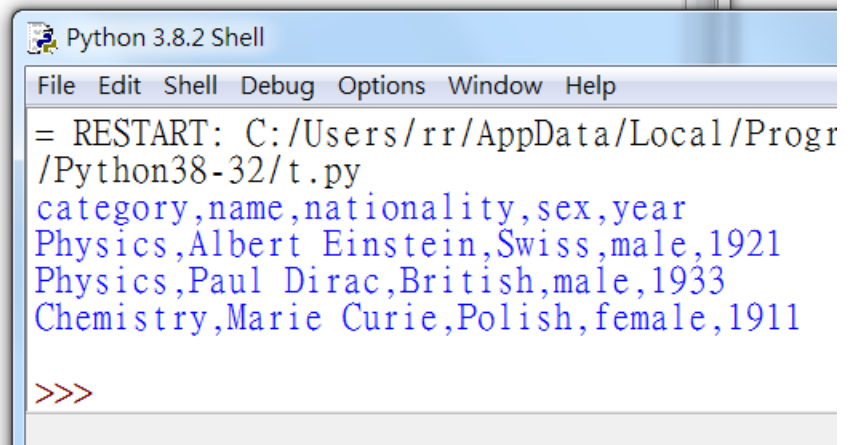


# 已經建立出正確csv檔案，可以使用了！

```
]
#print(nobel_winners)
cols = nobel_winners[0].keys() #print(cols)

##sort(cols) #因為dict_keys不可以排序！
#z=list(cols) #print(z)
#z.sort()
with open('c:/nobel_winners.csv', 'w') as f:
    f.write(','.join(cols) + '\n')
    for o in nobel_winners:
        row = [str(o[col]) for col in cols]
        f.write(','.join(row) + '\n')

with open('c:/nobel_winners.csv') as f:
    for line in f.readlines():
        print(line, end="")
print()
```



Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

= RESTART: C:/Users/rr/AppData/Local/Programs/Python/Python38-32/t.py

category,name,nationality,sex,year  
Physics,Albert Einstein,Swiss,male,1921  
Physics,Paul Dirac,British,male,1933  
Chemistry,Marie Curie,Polish,female,1911

>>>

- 建立好csv，下節就要使用Python的內置csv模塊來讀資料

# Exercise3-1

- **Ex3-1Add1:**「能夠移動數據」是數據可視化基本技能之一，請問可放數據的地方，主要有那兩類？分別又可概分為什麼？簡單說明即可。
- **Ex3-1Add2:**關於數據庫的**SQL**與**NoSQL**請簡單說明與比較，而且指出**python**在處理這些數據時常用的工具庫是什麼？
- **Ex3-1Add3:**如果在**python**已經有一個符合**JSON**格式的資料物件稱為**nobel\_winners**,請寫出可以將資料寫入**CSV**檔的程式碼。

## 3-2: 建立基本數據資料檔案\_CSV系列： 包括CSV, TSV and Row-column Data-formats

- 逗號分隔值（**CSV, Comma-Separated Values**）或製表符分隔值（**TSV, Tab-Separated Values**）可能是最普遍的基於文件的數據格式，並且作為數據可視化工具，這些格式常是用來處理數據的格式。
- 基本的技能是能夠讀寫**CSV**文件及其各種古怪的變體，
  - 如用豎線「|」或分號「;」來分隔。
  - 或在字串定義時使用單引號「'」代替標準雙引號。
- **Python**的**csv**模塊幾乎可以在這裡完成所有繁重的工作。
- **python**可以很方便的讀和寫**nobel\_winners**數據：

## 3-2a\_轉Dict為符合1列csv資料的DictWriter類

- 將我們的nobel\_winners數據（請參見示例3-1）寫入CSV文件是一件很簡單的事情。
- csv有一個專用的DictWriter類，它將我們的字典資料協助轉換為符合CSV的資料列，所以可以更方便的，直接用writerow(w)即可，其中的w是一個Dict.
- 我們要做的明確工作，就是
  - 將字典的鍵(key)指定清礎  
("category,name,nationality,sex,year)
  - 即(得獎類別，姓名，國籍，性別，年份)，並且要將這行標題，寫到CSV文件的第一行！

## Lab#3-2a\_轉Dict為符合1列csv資料的DictWriter類，所以，就可直接writerow(w)把 dict的w，寫到csv檔

3-2a\_轉Dict為符合csv資料的DictWriter類，所以，就可直接writerow(w)把 dict的w，寫到csv檔

```
1 import csv
2 ###with open('./nobel_winners.csv', 'wb') as f: ##python3則要把 'wb' 改為 'w'
3 #with open('./nobel_winners.csv', 'w') as f:
4 with open('./nobel_winners.csv', 'w', newline='') as f: #加入newline='', 才不會發生多輸出1個空白行的問題！
5     fieldnames = nobel_winners[0].keys() #要使用的欄位名稱，（在此例是： 'category', 'name', ..）
6     #fieldnames.sort()
7     writer = csv.DictWriter(f, fieldnames=fieldnames) #因為x fieldname，所以可檢查要印的data是否有完整的field名稱。
8     writer.writeheader()
9     for w in nobel_winners:
10         writer.writerow(w)
11
```

```
1 ##用方法三印出內容、●● 這是配合 with 的高效率寫法！
2 with open('./nobel_winners.csv', 'r') as fp:
3     for line in fp:
4         print(line, end='')
5 print('-'*60)
6 print('上方是文字檔的內容')
7 print('-'*60)
```

```
category,name,nationality,sex,year
Physics,Albert Einstein,Swiss,male,1921
Physics,Paul Dirac,British,male,1933
Chemistry,Marie Curie,Polish,female,1911
```

-----  
上方是文字檔的內容

## 3-2b\_轉換讀到的1列csv資料為Dict的DictReader類

- 使用**CSV**數據的一種更方便的方法是將每列資料自動轉換為**Python**字典。
- 轉換後的記錄形式，就是列表裝字典，也就是列表內每個元素都是字典。(a list of dicts).
- 為達此目的，**csv**有一個方便的**DictReader**：

# Lab#3-2b\_轉換讀到的1列csv資料為Dict的DictReader類

## 3-2b\_轉換讀到的csv資料為Dict的DictReader類

- 1 使用CSV數據的一種更方便的方法是將行轉換為Python字典。
- 2 轉換後的記錄形式，就是列表裝字典，也就是列表內每個元素都是字典。(a list of dicts).
- 3 為達此目的，csv有一個方便的DictReader：

```
1 import csv
2 with open('./nobel_winners.csv') as f:
3     reader = csv.DictReader(f)
4     nobel_winners = list(reader)    #把所有讀到的項目資料放進一個list容器物件。
5 nobel_winners
```

```
[{'category': 'Physics',
  'name': 'Albert Einstein',
  'nationality': 'Swiss',
  'sex': 'male',
  'year': '1921'},
 {'category': 'Physics',
  'name': 'Paul Dirac',
  'nationality': 'British',
  'sex': 'male',
  'year': '1933'},
 {'category': 'Chemistry',
  'name': 'Marie Curie',
  'nationality': 'Polish',
  'sex': 'female',
  'year': '1911'}]
```



# 3-2c-1以自訂的縮小版鳶尾花資料庫(只4筆資料的iris.csv)為例，測試DictReader

## 3-2c-1. 讀取csv而且轉為Dict DictReader

讀取 CSV 檔案 假設我們有一個 csv 檔案 iris.csv，其內容如下：

sepal\_length,sepal\_width,petal\_length,petal\_width,species 5.1,3.5,1.4,0.2,setosa 4.9,3,1.4,0.2,setosa 4.7,3,4.6,3.1,1.5,0.2,setosa

以使用下面這段程式碼將這個 csv 檔的內容讀取出來：

```
: 1 import csv ##※注意 csv裡「,」逗號前後都不可有多餘的空白哦！
2 str_irir_txt="""sepal_length,sepal_width,petal_length,petal_width,species
3 5.1, 3.5, 1.4, 0.2, setosa
4 4.9, 3, 1.4, 0.2, setosa
5 4.7, 3.2, 1.3, 0.2, setosa
6 4.6, 3.1, 1.5, 0.2, setosa"""
7 with open('./iris.txt', 'w', newline='') as csvfile:
8     print(str_irir_txt,file=csvfile)
9
10
11 # 開啟 CSV 檔案
12 with open('./iris.txt', newline='') as csvfile:
13     # 讀取 CSV 檔案內容
14     rows = csv.reader(csvfile)
15     #rows = csv.reader(csvfile, delimiter=':') #用:來分隔資料，例如linux的passwd
16
17     # 以迴圈輸出每一列
18     for row in rows:
19         print(row)
```

```
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
['5.1', ' 3.5', ' 1.4', ' 0.2', ' setosa']
['4.9', ' 3', ' 1.4', ' 0.2', ' setosa']
['4.7', ' 3.2', ' 1.3', ' 0.2', ' setosa']
['4.6', ' 3.1', ' 1.5', ' 0.2', ' setosa']
```

## 3-2c-2 讀取csv\_而且用DictReader轉為Dict

### 3-2c-2 讀取csv\_而且轉為Dict DictReader

```
1 #DictReader
2 #讀取成 Dictionary
3 #我們也可以將 csv 檔案的內容讀取進來之後，轉為 Python 的 dictionary 格式，這樣在存取各定資料時會方便一些：
4
5 import csv
6 with open('./iris.txt', newline='') as csvfile:
7     # 讀取 CSV 檔內容，將每一列轉成一個 dictionary
8     rows = csv.DictReader(csvfile)
9     # 以迴圈輸出指定欄位
10    for row in rows:
11        print(row['sepal_length'], row['species'])
12 # #這個例子中我們使用以 csv.DictReader 來讀取 CSV 檔案的內容，它會自動把第一列 (row) 當作欄位的名稱，將第二
13
14 # 5.1 setosa
15 # 4.9 setosa
16 # 4.7 setosa
17 # 4.6 setosa
18 # 5 setosa
19
```

```
5.1 setosa
4.9 setosa
4.7 setosa
4.6 setosa
```

## 3-2d: 請注意數據處理的潛在問題！

- 目前的DictReader, 所讀到的數字是以字串 (String)形式讀取的！
- 因為「數字」是以「字串 (String)」形式讀取的！
  - 如果您想對它們進行數字化處理，則需要將所有數字列轉換為其各自的類型
  - 例如：此時的年紀的類型應該是整數型態！

### 3-2d: 請注意數據處理的潛在問題！

```
1 目前的DictReader, 所讀到的數字是以字串(String)形式讀
2
3 因為「數字」是以「字串(String)」形式讀取的！
4 如果您想對它們進行數字化處理，則需要將所有數字列轉換為
5 例如：此時的年紀的類型應該是整數型態！
6
```

```
1  #把year的形態，由字串，轉為 整數(integer)
2  for w in nobel_winners:
3      w['year'] = int(w['year'])
4  nobel_winners

[{'name': 'nnn',
  'category': 'math',
  'year': 1999,
  'nationality': None,
  'sex': None,
  '_id': ObjectId('5f79654dd766d26c5c99dcb4')},
 {'name': 'nnn2',
  'category': 'math2',
  'year': 19992,
  'nationality': None,
  'sex': None,
  '_id': ObjectId('5f79654dd766d26c5c99dcb5')},
 {'name': 'Albert Einstein',
  'category': 'Physics',
  'year': 1921,
```

## Exercise3-2

- **Ex3-2Add1:**什麼是**CSV**格式的檔案？是否有那些變型？
- **Ex3-2Add2:** (a)為何在處理數據資料時，即使資料格式是python可輕易處理的**csv**,仍建議使用**panda**而不是用內建的程式庫**csv**? (b)數字與字串之間的轉換，會有意想不到的情況嗎？請舉一實例說明。
- **Ex3-2Add3:** python程式設計，請參考**iris**，自行設計類似**iris**的資料3筆以上即可。需可以將這些資料寫入**csv**的檔案，並且讀入**csv**檔並印出來。

## 3-3: JSON

- **json**模組的使用，算是非常簡單。只要 **json.dump()** 與 **json.load**即可寫出與讀入 **json**檔。
- 對於諸如字串，整數和浮點數之類的原始數據型態.用**json**模塊可以輕鬆地將**Python**字典 用**dump**的方法存到**JSON**文件中。
  - **dump**方法需要「一個**Python**容器」和「一個檔案文件指標」，並且會將容器的內容保存到後面的檔案中。
- 如果，想直接把現在時間**dumps**，卻會出現錯誤！？  
那要如何解決呢？

```
##注意：以下執行會發生錯誤！  
  
##直接把現在時間，dumps 就會出現錯誤！ 那要如何解決呢？  
from datetime import datetime  
json.dumps(datetime.now())  
  
...  
  
##直接把現在時間，dumps 就會出現錯誤！ 那要如何解決呢？  
from datetime import datetime  
json.dumps(datetime.now())  
# Out:  
# ...  
# TypeError: datetime.datetime(2015, 9, 13, 10, 25, 52,  
# is not JSON serializable  
...  
  
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-149-626ffea25c72> in <module>  
      3 ##直接把現在時間，dumps 就會出現錯誤！ 那要如何解決呢？  
      4 from datetime import datetime  
----> 5 json.dumps(datetime.now())  
      6
```

Note:比較 dumps轉換前 與 json.dumps()轉換後，幾乎相同.

- ##注意1：python在處理 json時，特別是對於 dict裡的pair\_data:(key:value)，裡的 key，會轉成雙引號！
- ##注意2: 其中的value，數值就沒有雙引號，字串會有雙引號！

●關於「字串」與key的命名區：  
要注意的是本來Python的字串是可以使用『雙引號/或單引號』來表示，轉到 Json 格式之後一律得使用『雙引號』來表示(如果要使用雙引號，就要用逸出字元)。所以在用 json.load()時，必須是正確的"雙引號"的格式。

●關於其他的 True/False 與 None 就會轉成JavaScript的 true/false與 null

※注意：原來的dict的Key是單引號或雙引號，但轉成Json會變為雙引號！

```
3 #接著我們來看段簡單的範例：
4 import json
5 data = {"today": 9, "is": 7, "nice": 8, 'day': [6,5,.4,True,None] }
6 output = json.dumps(data)
7 output
```

```
'{"today": 9, "is": 7, "nice": 8, "day": [6, 5, 0.4, true, null]}'
```

```
1 #接著我們來看段簡單的範例：
2 import json
3 data = {'today': "9", 'is': "7", "nice": 8, 'day': 6}
4 output = json.dumps(data)
5 output
```

```
'{"today": "9", "is": "7", "nice": 8, "day": 6}'
```

```
1 # python的json.dumps會將python合法資料型態 編碼為 json字串
2 t={"today": "9", "is": "7", "nice": 8, "day": true}
3 t = json.loads(t)
4 t
```

```
{'today': '9', 'is': '7', 'nice': 8, 'day': True}
```

```
1 # python的json.dumps會將python合法資料型態 編碼為 json字串
2 t2={"today": '9', 'is': "7", "nice": 8, "day": [6,5,.4,True,None] }
3 t2 = json.dumps(t2)
4 t2
5 #print(t2)
```

```
'{"today": "9", "is": "7", "nice": 8, "day": [6, 5, 0.4, true, null]}'
```

## Exercise3-3

- **Ex3-3Add1:** 用python直接處理json檔案很容易，請問要寫資料到檔案的指令是？要讀資料的指令是？
- **Ex3-3Add2:** 用python要寫資料到json檔案時，無法把現在時間用json.dump寫到json檔案的原因是？是否有比較好的解決方法？
- **Ex3-3Add3:** python在處理 json時，對於 dict裡的pair\_data : (key:value)，裡的 key，與value是否有特別要注意的地方？
- **Ex3-3Add4:** json執行dumps後的結果：參考下列程式，寫出會印出來的結果。
  - ❑ `import json`
  - ❑ `t2={"today": '9', 'is': "7", "nice": 8, "day": [6,5,.4,True,None] }`
  - ❑ `t2 = json.dumps(t2)`
  - ❑ `t2 #print(t2)`



## 3-4:SQL用SQLAlchemy程式庫連接與操作SQLite3建的SQL資料庫

- 要使用SqlAlchemy程式操作Sqlite3之前要先認識ORM.
- 想法：希望只寫python物件，就可定義sql中的表格檔！
  - 為了要有可繼承的基礎物件，所以要：使用 `declarative_base()` 這個函式來建立一個基礎類別。
  - 以後要使用ORM,就先描述要處理的資料庫表格, 然後把程式寫在對應到這些表格的類別裡面,
- 什麼是 ORM?
  - ORM，英文叫 (Object Relational Mapping) 翻譯成中文為「物件關聯對映」
  - ORM在網站開發結構，是位於『資料庫』和『Model資料容器』兩者之間的一項技術。
  - 目的是幫助使用者簡便、安全完成資料庫讀取。
  - ORM 的主要特性是，透過程式語言(例如python)，去操作資料庫語言( SQL )。
  - 是把資料庫的表格用物件導向的概念來產生的一種工具模式。

## 3-4: SQL--SQLite:

- SQLAlchemy: <https://zh.wikipedia.org/wiki/SQLAlchemy>
- SQLAlchemy: 使用SQLAlchemy將數據寫入SQLite文件檔案型資料庫，並且完成以下四種操作：
  - Creating engine/Defining Tables/ Adding Instances with a Session/ Querying
- 對於與SQL數據庫進行交互，SQLAlchemy是很容易上手的Python程式庫。
- 如果在乎速度和效率，可以使用原始SQL指令，也供了強大的「物件關係映射(ORM, object-relational mapping)」.所以可用高級Pythonic API在SQL的table上進行操作，卻又可以在本質上視為Python的類別！
- 在允許用戶將數據視為Python容器的同時，使用SQL讀取和寫入數據是一個複雜的過程，儘管SQLAlchemy比低級SQL引擎更易於使用，但它仍然是一個相當複雜的庫。
- 此處，將以之前數據為目標，介紹基礎知識

# 3-4-1: Creating the Database Engine

- 啟動SQLAlchemy會話(session)時，要做的第一件事是創建數據庫引擎(Engine)。
- 該引擎將與所討論的數據庫建立連接(connection)，執行SQLAlchemy生成的通用SQL指令所需的任何轉換，並傳回數據。
- 幾乎所有流行的數據庫都有引擎，還有一個內存選件，該選件將數據庫保存在RAM中，從而可以快速訪問進行測試。
- 建立引擎的優點：
- 由於所採用的資料庫是可互換的，意味著您可以使用基於文件的簡單SQLite數據庫來開發代碼，然後在生產期間通過更改單個配置字串(config string)，就可切換到更工業化的產品（例如Postgresql）。
- 
- 查看SQLAlchemy的文件，可以獲取更完整的功能說明。
- 指定數據庫URL的形式為：
- dialect+driver://username:password@host:port/database

# 因此，要連接到'nobel\_prize.db'需要執行以下操作。

## ■ (1)建立資料庫的engine

資料庫的engine:

dialect+driver://username:password@host:port/database

重點是要指明

- 所用的資料庫是？例如mysql, sqlite,...
- 所連的網址路徑是？例如 localhost, ...
- 其他參數。例如 `echo=True` 在執行時，才會有echo的回應

#sqlite3的圖形界面檢查 window DBrowser for SQLite download: <https://sqlitebrowser.org/dl/>

※註：如果對資料庫的運作很熟悉，也可以直接在SQLite的圖形界面中，建立所要的資料表格哦！

- 1 #因此，要連接到在localhost上運行的'nobel\_prize' 的MySQL數據庫，需要執行以下操作。
- 2 #注意：此時的create並未執行真正的SQL 哦！ 這是一種延遲初始化(lazy initialization)，只是先作好資料庫的準備，等到後續真正的資料庫操作，才會執行。

```
] : 1 ##engine = create_engine( 'mysql://kyran:myppswd@localhost/nobel_prize')
```

```
] : 1 from sqlalchemy import create_engine  
2 engine = create_engine('sqlite:///./nobel_prize.db', echo=True)
```

## 3-4-2: SQLAlchemy建表格綱要的方法

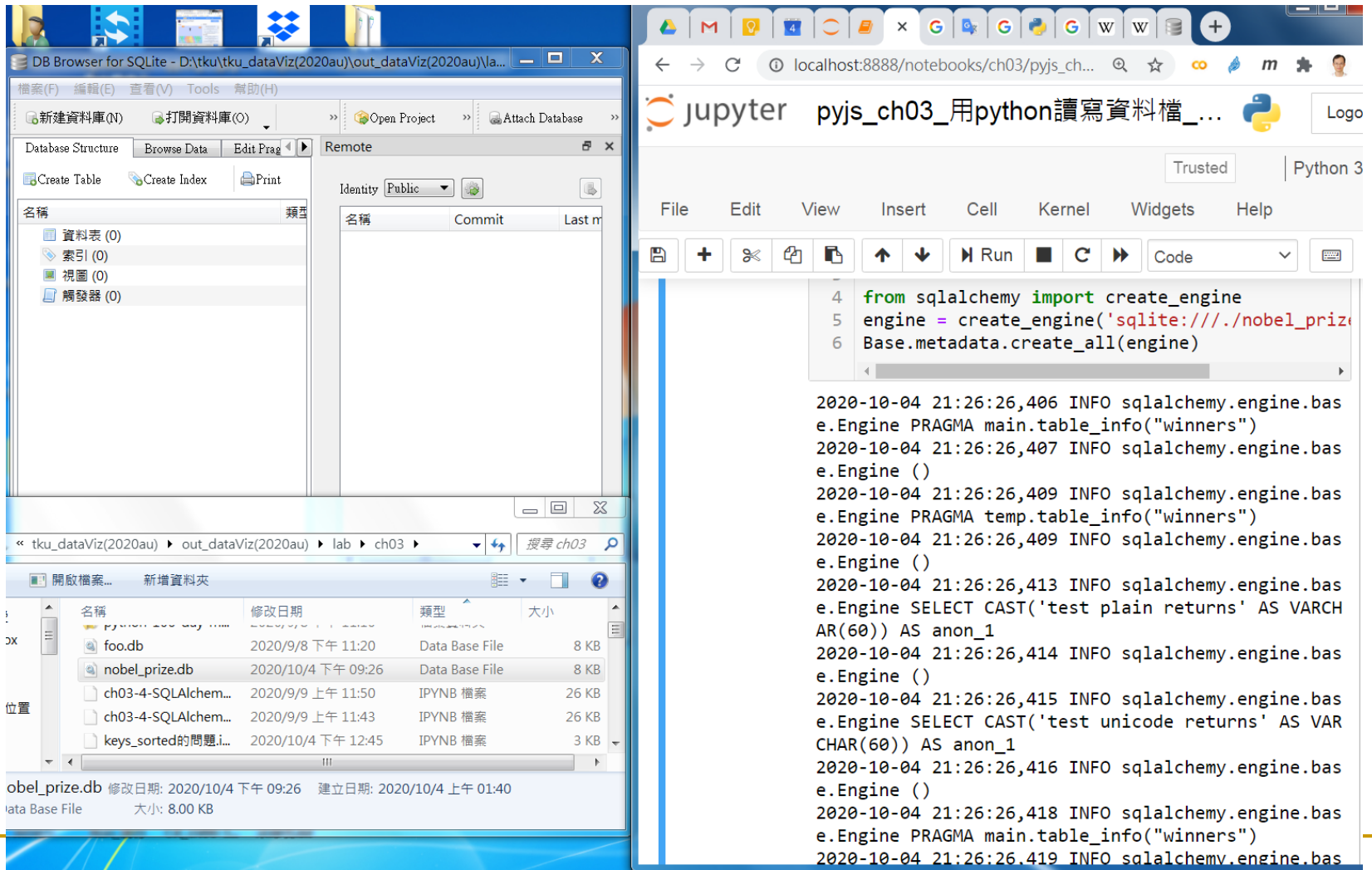
- (2)建好資料表格綱要\_\_利用ORM的基類(base class)\_在SQLAlchemy的基類是Base=declarative\_base()
- (2-2)用Base的metadata裡產生table.
- (2-2-a):有create\_all(engine) 方法，產生我們要的database.
- (2-2-b):有drop\_all(engine) 方法，刪除全部已建的tables.
- 您可以使用以下表類，用相當像Python的寫法與數據庫進行交互

```
1 from sqlalchemy import Column, Integer, String, Enum
2 #// ...
3
4 class Winner(Base): ##### 用python的物件寫法，寫出sql 建立的表格綱要(table schema)的敘述
5     __tablename__ = 'winners'      ### table 的名稱
6     id = Column(Integer, primary_key=True)    ### 第 0 項 (key) 項資料行(為了當作索引key的序號!)
7     name = Column(String)                ### 第1項資料行 name
8     category = Column(String)
9     year = Column(Integer)
10    nationality = Column(String)
11    sex = Column(Enum('male', 'female'))
12
13    ## 下面的函數是可有可無，不是必要，目的在session.query(Winner)時，可以印出較有意義的訊息，才加入這個
14    def __repr__(self): #這個自訂的 物件 Winner(xx)，在列印Winner時，只印出三項資料項。(當初的預設是會
15        return "訊息query顯示用: <Winner(name='%s', category='%s', year='%s')>" \
16            %(self.name, self.category, self.year)
```

- 請注意\_\_tablename\_\_成員（用於命名SQL表和用作檢索它的關鍵字），以及可選的自定義\_\_repr\_\_方法（將在打印表列的時候，使用）。

# 安裝sqlite3的資料browser瀏覽器

- DB browser for SQLite 下載<https://sqlitebrowser.org/dl/>



The image shows two side-by-side screenshots. The left screenshot is of the DB Browser for SQLite application. The right screenshot is of a Jupyter Notebook running in a web browser.

**DB Browser for SQLite Screenshot:**

- Title bar: DB Browser for SQLite - D:\tku\_dataViz(2020au)\out\_dataViz(2020au)\la...
- Menu bar: 檔案(F), 編輯(E), 查看(V), Tools, 幫助(H)
- Toolbar: 新建資料庫(N), 打開資料庫(O), Open Project, Attach Database
- Database Structure: Create Table, Create Index, Print
- Table List: 名稱, 類型, 資料表 (0), 索引 (0), 視圖 (0), 觸發器 (0)
- Table Details: Identity: Public, Commit, Last m
- Bottom Panel: 開啟檔案..., 新增資料夾, 名稱, 修改日期, 類型, 大小, 位置
- File List:

名稱	修改日期	類型	大小
foo.db	2020/9/8 下午 11:20	Data Base File	8 KB
nobel_prize.db	2020/10/4 下午 09:26	Data Base File	8 KB
ch03-4-SQLAlchem...	2020/9/9 上午 11:50	IPYNB 檔案	26 KB
ch03-4-SQLAlchem...	2020/9/9 上午 11:43	IPYNB 檔案	26 KB
keys_sorted的問題.i...	2020/10/4 下午 12:45	IPYNB 檔案	3 KB

nobel\_prize.db 修改日期: 2020/10/4 下午 09:26 建立日期: 2020/10/4 上午 01:40  
Data Base File 大小: 8.00 KB

**Jupyter Notebook Screenshot:**

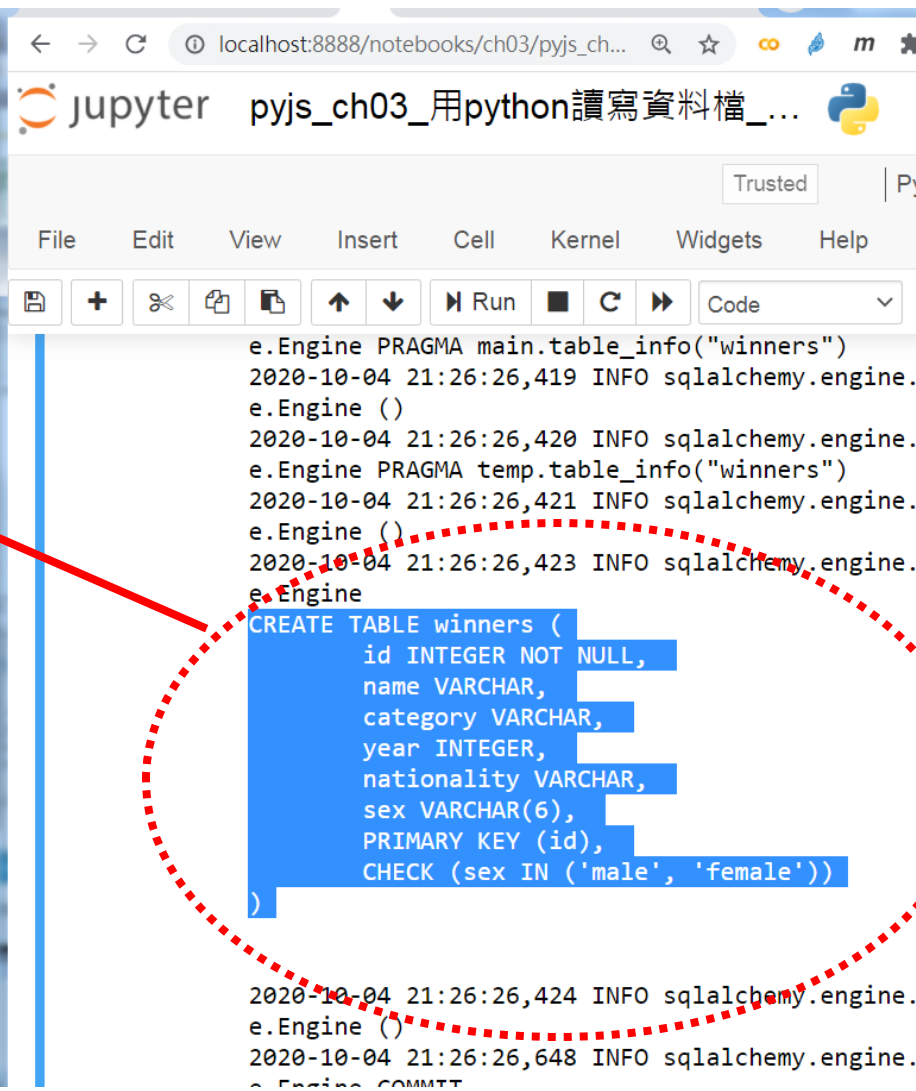
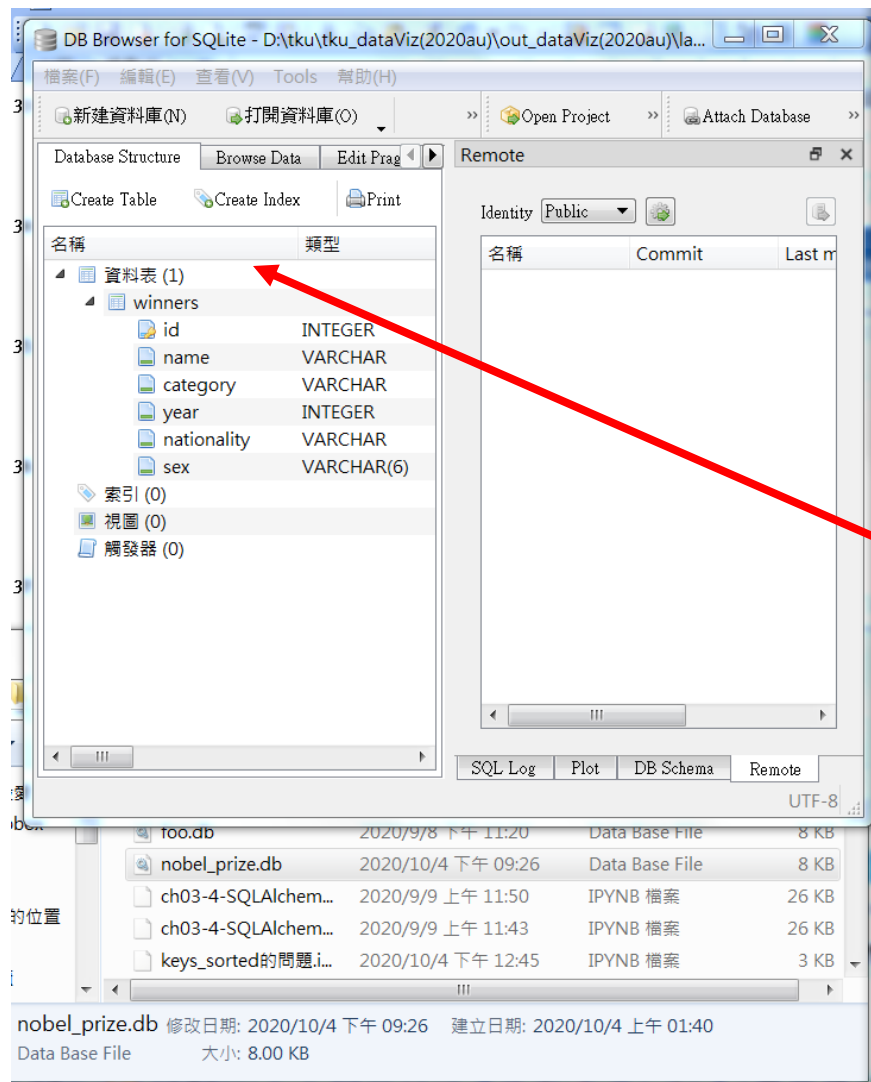
- Address bar: localhost:8888/notebooks/ch03/pyjs\_ch...
- Page Title: jupyter pyjs\_ch03\_用python讀寫資料檔...
- Language: Python 3
- Menu bar: File, Edit, View, Insert, Cell, Kernel, Widgets, Help
- Toolbar: Run, Code
- Code Cell:

```
4 from sqlalchemy import create_engine
5 engine = create_engine('sqlite:///./nobel_prize.db')
6 Base.metadata.create_all(engine)
```

2020-10-04 21:26:26,406 INFO sqlalchemy.engine.base.Engine PRAGMA main.table\_info("winners")  
2020-10-04 21:26:26,407 INFO sqlalchemy.engine.base.Engine ()  
2020-10-04 21:26:26,409 INFO sqlalchemy.engine.base.Engine PRAGMA temp.table\_info("winners")  
2020-10-04 21:26:26,409 INFO sqlalchemy.engine.base.Engine ()  
2020-10-04 21:26:26,413 INFO sqlalchemy.engine.base.Engine SELECT CAST('test plain returns' AS VARCHAR(60)) AS anon\_1  
2020-10-04 21:26:26,414 INFO sqlalchemy.engine.base.Engine ()  
2020-10-04 21:26:26,415 INFO sqlalchemy.engine.base.Engine SELECT CAST('test unicode returns' AS VARCHAR(60)) AS anon\_1  
2020-10-04 21:26:26,416 INFO sqlalchemy.engine.base.Engine ()  
2020-10-04 21:26:26,418 INFO sqlalchemy.engine.base.Engine PRAGMA main.table\_info("winners")  
2020-10-04 21:26:26,419 INFO sqlalchemy.engine.base.Engine ()



執行Base.metadata.create\_all(engine)就可新建table在資料庫Nobel中。





## 3-4-3:透過會話(Session)加入映射類別的實例

- 在建好 winners的Table之後，就可以開始向其中添加獲獎者實例了。所以需要一個會話(Session)來與以下對象進行交互：
- 接著，我們可以使用Winner類創建實例和表行，並將其添加到會話(session)中

### 3-4-3: # 透過會話(Session)加入映射類別的實例Adding Instances with a Session

```
1  在建好 winners的Table之後，就可以開始向其中添加獲獎者實例了。  
2  所以需要一個會話(Session)來與以下對象進行交互：
```

```
: 1  from sqlalchemy.orm import sessionmaker  
2  
3  Session = sessionmaker(bind=engine)  
4  session = Session()
```

```
: 1  import sqlalchemy  
2  sqlalchemy.__version__
```

```
: '1.3.18'
```

```
: 1  #ed_user_test = Winner('name', 'category', 'year')  
2  ed_user_test = Winner(name='nnn', category='math', year='1999')  
3  session.add(ed_user_test)
```

```
: 1  ed_user_test2 = Winner(name='nnn2', category='math2', year='19992')  
2  session.add(ed_user_test2)
```

```
: 1  session.dirty
```

```
: IdentitySet([])
```

```
: 1  session.new|
```

— IdentitySet([訊息query顯示用：<Winner(name='nnn', category='math', year='1999')>], 訊息query顯示用：<Winner(name='nnn2', category='math2', year='19992')>])

注意：

(1) \*\*的運算符，讓我們可以很方便的將第一個nobel\_winners成員解壓縮為鍵/值對：(name='Albert Einstein', category='Physics'...).

(2)其中的session.new 是已添加到此會話中的所有項目的集合。

另外，也要注意，所有數據庫插入和刪除操作均在Python中進行。只有當我們使用commit方法時，數據庫才會被更改。

```
In: 1 albert = Winner(**nobel_winners[0])
    2 session.add(albert)
    3 session.new
    4 # Out:
    5 #IdentitySet([<Winner(name='Albert Einstein', category='Physics', year='1921')>])

In: IdentitySet([訊息query顯示用: <Winner(name='Albert Einstein', category='Physics', year='1921')>])

In: 1 session

In: <sqlalchemy.orm.session.Session at 0x4d53400>
```

注意：

- (1) \*\*的運算符，讓我們可以很方便的將第一個nobel\_winners成員解壓縮為鍵/值對：(name='Albert Einstein', category='Physics'...).
- (2)其中的session.new 是已添加到此會話中的所有項目的集合。

另外，也要注意，所有數據庫插入和刪除操作均在Python中進行。只有當我們使用commit方法時，數據庫才會被更改。

提示：

- 關於commit, 應該盡可能少提交(commits), 使SQLAlchemy在幕後發揮作用。
- 提交(commits)時, SQLAlchemy應該匯總各種數據庫操作, 並以有效的方式進行通信。
- 提交(commits), 會涉及建立數據庫握手和協商事務, 這通常是一個緩慢的過程, 並且您想盡可能地限制它, 以充分利用SQLAlchemy的簿記功能(bookkeeping abilities)。

```
In: 1 #如 new 方法所示, 我們在會話中添加了Winner。
    2 # 我們也可以使用刪除, 來刪除對象, 留下一個空的IdentitySet:

In: 1 session.expunge(albert) #移除這個albert物件, 也可全清除, 用session.expunge_all
    2 session.new
    3 # Out:
    4 # IdentitySet([])

In: IdentitySet([])

In: 1 #此時, 尚未發生數據庫插入或刪除。
    2 #讓我們將nobel_winners列表中的所有成員添加到會話中並將它們提交到數據庫:
```

## 3-4-4: Querying the Database

- 要訪問數據，請使用會話的查詢(session's query)方法，該方法的結果可以進行過濾，分組和相交(filtered, grouped, and intersected)，從而可以進行所有標準SQL數據檢索。
- 您可以在SQLAlchemy文檔中籤出可用的查詢方法。
- 現在，我將快速瀏覽Nobel數據集上的一些最常見查詢。
- 首先，我們計算 winner's table 的資料列數：
- `session.query(Winner).count()`

# 計算 winner's table 的資料數，與設定查詢條件 WHERE winners.nationality = ?

8 首先，我們計算 winner's table 的資料列數：

```
1 session.query(Winner).count()|
2 # Out:
3 # 4
```

```
2020-10-04 23:23:19,184 INFO sqlalchemy.engine.base.Engine BEGIN (implicit)
2020-10-04 23:23:19,186 INFO sqlalchemy.engine.base.Engine SELECT count(*) AS count_1
FROM (SELECT winners.id AS winners_id, winners.name AS winners_name, winners.category AS winners_category, winners.year AS winners_year, winners.nationality AS winners_nationality, winners.sex AS winners_sex
FROM winners) AS anon_1
2020-10-04 23:23:19,187 INFO sqlalchemy.engine.base.Engine ()
```

4

```
1 #Next, let's retrieve all Swiss winners:
2 result = session.query(Winner).filter_by(nationality='Swiss')
3 list(result)
4 # Out:
5 # # [<Winner(name='Albert Einstein', category='Physics', year='1921')>]
6 ##### {'category': 'Physics', 'name': 'Albert Einstein', 'nationality': 'Swiss', 'sex': 'male', 'year': 1921}
```

```
2020-10-04 23:23:19,273 INFO sqlalchemy.engine.base.Engine SELECT winners.id AS winners_id, winners.name AS winners_name, winners.category AS winners_category, winners.year AS winners_year, winners.nationality AS winners_nationality, winners.sex AS winners_sex
FROM winners
```

```
WHERE winners.nationality = ?
```

```
2020-10-04 23:23:19,273 INFO sqlalchemy.engine.base.Engine ('Swiss',)
```

[訊息query顯示用: <Winner(name='Albert Einstein', category='Physics', year='1921')>]

# 將目前查到的數據物件轉為dict的技術

- 我們一開始有準備目標資料的字典，然後建立了資料庫。
- 現在如果想重建我們的目標資料的字典，其實有機會透過查詢返回的Winner對象，然後轉換為Python字典(dict)
- 為了達成這個找到原來資料字典，編寫小函數來根據SQLAlchemy類創建字典。
- 將使用一些表格自檢來獲取列標籤（請參見Example3-4）。

# 由Sqlalchemy目前查到的物件轉為dict\_(字典)

```
1 我們一開始有準備目標資料的字典，然後建立了資料庫。
2 現在如果想重建我們的目標資料的字典，其實有機會透過查詢返回的Winner對象，然後轉換為Python字典(dict)
3
4 為了達成這個找到原來資料字典，編寫小函數來根據SQLAlchemy類創建字典。
5 將使用一些表格自檢來獲取列標籤（請參見 Example3-4）。
6
```

```
1 #Example 3-4. Converts an SQLAlchemy instance to a dict
2 def inst_to_dict(inst, delete_id=True):
3     dat = {}
4     for column in inst.__table__.columns:
5         dat[column.name] = getattr(inst, column.name)
6     if delete_id:
7         dat.pop('id')    #If delete_id is true, remove the SQL primary ID field.
8     return dat
```

```
1 #先準備Example3-4，由SqlAlchemy查到的Winner物件，再透過Example3-4，就可還原出原來的資料物件了！
2 winner_rows = session.query(Winner)
3 nobel_winners = [inst_to_dict(w) for w in winner_rows]
4 nobel_winners
```

```
2020-10-04 23:42:48,434 INFO sqlalchemy.engine.base.Engine BEGIN (implicit)
2020-10-04 23:42:48,436 INFO sqlalchemy.engine.base.Engine SELECT winners.id AS winners_id, winners.category AS winners_category, winners.year AS winners_year, winners.nationality AS winners_nationality, winners.sex AS winners_sex
FROM winners
2020-10-04 23:42:48,436 INFO sqlalchemy.engine.base.Engine ()
```

```
[{'name': 'nnn2',
  'category': 'math2',
  'year': 19222,
  'nationality': None,
  'sex': None},
 {'name': 'Albert Einstein',
  'category': 'Physics',
  'year': 1921,
  'nationality': 'Swiss'.
```

## 3-4-4b: 查詢到所要處理的資料項之後，更新update

```
'year': 1933,  
'nationality': 'British',  
'sex': 'male'},  
{'id': 3,  
'name': 'Marie Curie',  
'category': 'Chemistry',  
'year': 1911,  
'nationality': 'Polish',  
'sex': 'female'}]
```

]: 1

### 3-4-4b: 查詢到所要處理的資料項之後，更新update

query到所要的資料項，也可過更對象的屬性更變，來輕鬆更新(update)數據庫行：

```
3]: 1 marie = session.query(Winner).get(3) #Fetches Marie Curie, nationality Polish.  
2 marie.nationality = 'French'  
3 session.dirty  
4 # Out:  
5 # IdentitySet([<Winner(name='Marie Curie', category='Chemistry', year='1911')>])
```

3]: IdentitySet([訊息query顯示用：<Winner(name='Marie Curie', category='Chemistry', year='1911')>])

上面程式：.dirty 顯示所有尚未提交到數據庫的已更改實例。讓我們進行瑪麗的更改，並檢查她的國籍已從波蘭(Polish)更改為法國(French)：

## Lab#執行:UPDATE winners SET nationality=? WHERE winners.id =?

```
: 1 session.commit()
2 # Out:
3 # 2020-10-05 00:16:52,687 INFO sqlalchemy.engine.base.Engine UPDATE winners SET nationality=? WHERE winners.id = ?
4 # 2020-10-05 00:16:52,688 INFO sqlalchemy.engine.base.Engine ('French', 3)
5 # 2020-10-05 00:16:52,689 INFO sqlalchemy.engine.base.Engine COMMIT
```

```
2020-10-05 00:16:52,687 INFO sqlalchemy.engine.base.Engine UPDATE winners SET nationality=? WHERE winners.id = ?
2020-10-05 00:16:52,688 INFO sqlalchemy.engine.base.Engine ('French', 3)
2020-10-05 00:16:52,689 INFO sqlalchemy.engine.base.Engine COMMIT
```

```
: 1 #先準備Example3-4，由SqaAlchemy查到的winner物件，再透過Example3-4，就可還原出原來的資料物件了！
2 winner_rows = session.query(Winner)
3 nobel_winners_3 = [inst_to_dict(w, delete_id=False) for w in winner_rows]
4 nobel_winners_3
```

```
2020-10-05 00:16:53,016 INFO sqlalchemy.engine.base.Engine BEGIN (implicit)
2020-10-05 00:16:53,019 INFO sqlalchemy.engine.base.Engine SELECT winners.id AS winners_id, winners.name AS winners_name, winners.category AS winners_category, winners.year AS winners_year, winners.nationality AS winners_nationality, winners.sex AS winners_sex
FROM winners
2020-10-05 00:16:53,021 INFO sqlalchemy.engine.base.Engine ()
```

```
: [{ 'id': 1,
    'name': 'Albert Einstein',
    'category': 'Physics',
    'year': 1921,
    'nationality': 'Swiss',
    'sex': 'male'},
  { 'id': 2,
    'name': 'Paul Dirac',
    'category': 'Physics',
    'year': 1933,
    'nationality': 'British',
    'sex': 'male'},
  { 'id': 3,
    'name': 'Marie Curie',
    'category': 'Chemistry',
    'year': 1911,
    'nationality': 'French',
    'sex': 'female'}]
```



## 3-4-4c: query除了可查更新數據庫行，您還可以刪除(delete)查詢結果

DELETE FROM winners WHERE winners.name = ?

3-4-4c: query除了可查更新數據庫行，您還可以刪除(delete)查詢結果：

```
1 session.query(Winner).filter_by(name='Albert Einstein').delete()
2 # Out:
3 # INFO:sqlalchemy.engine.base.Engine:DELETE FROM winners WHERE
4 # winners.name = ?
5 # INFO:sqlalchemy.engine.base.Engine:('Albert Einstein',)
6 # 1
7
```

```
2020-10-05 00:21:16,963 INFO sqlalchemy.engine.base.Engine:DELETE FROM winners WHERE winners.name = ?
2020-10-05 00:21:16,965 INFO sqlalchemy.engine.base.Engine:('Albert Einstein',)
```

1

```
1 list(session.query(Winner))
2 # Out:
3 # [<Winner(name='Paul Dirac', category='Physics', year='1933')>,
4 #  <Winner(name='Marie Curie', category='Chemistry', year='1911')>]
```

```
2020-10-05 00:21:23,637 INFO sqlalchemy.engine.base.Engine:SELECT winners.id AS winners_id, winners.name AS winners_name, winners.category AS winners_category, winners.year AS winners_year, winners.nationality AS winners_nationality, winners.sex AS winners_sex
FROM winners
2020-10-05 00:21:23,639 INFO sqlalchemy.engine.base.Engine:()
```

[訊息query顯示用: <Winner(name='Paul Dirac', category='Physics', year='1933')>,  
訊息query顯示用: <Winner(name='Marie Curie', category='Chemistry', year='1911')>]

```
1 session.commit()
```

```
2020-10-05 00:21:37,617 INFO sqlalchemy.engine.base.Engine:COMMIT
```

```
1 #先準備Example3-4，由SqaAlchemy查到的Winner物件，再透過Example3-4，就可還原出原來的資料物件了！
2 winner_rows = session.query(Winner)
3 nobel_winners_4 = [inst_to_dict(w, delete_id=False) for w in winner_rows]
4 nobel_winners_4
```

```
2020-10-05 00:22:15,166 INFO sqlalchemy.engine.base.Engine:BEGIN (implicit)
2020-10-05 00:22:15,168 INFO sqlalchemy.engine.base.Engine:SELECT winners.id AS winners_id, winners.name AS winners_name, winners.category AS winners_category, winners.year AS winners_year, winners.nationality AS winners_nationality, winners.sex AS winners_sex
FROM winners
2020-10-05 00:22:15,169 INFO sqlalchemy.engine.base.Engine:()
```

```
{'id': 2,
 'name': 'Paul Dirac',
 'category': 'Physics',
 'year': 1933,
 'nationality': 'British',
 'sex': 'male'},
 {'id': 3,
 'name': 'Marie Curie',
 'category': 'Chemistry',
 'year': 1911,
 'nationality': 'French',
 'sex': 'female']}
```

## Exercise 3-4

- Ex3-4Add1:請問什麼是「物件關聯對映」(ORM, Object Relational Mapping)，主要的優點與缺點是？
- Ex3-4Add2:ORM可幫使用者用python寫法來執行sql，是否也有不需要ORM來建立資料庫的方法？
- Ex3-4Add3:資料庫有四種常用函式：新增(Create)，讀取(Read)，更新(Update)，刪除>Delete)；通常會被簡寫為CRUD，用請由這四個方向，配合SQL的指令，來簡單說明SQLAlchemy會是如何？
- Ex3-4Add4:透過SqlAlchemy操作資料庫時，所有數據庫插入和刪除操作均在Python中進行。只有當我們使用什麼方法完成之後，數據庫才會被更改？這個操作一般是儘量越少越好，理由是？
- Ex3-4Add5:為了方便查閱，請設計在透過SqlAlchemy操作資料庫時可以將「目前所查到的資料物件」「轉為dict」的副程式。並簡單示範使用方法。

### 3-5:再簡化SQL存取程式的「懶人數據工具」Dataset工具

- **Dataset**程式庫(懶人數據庫)--可以再簡化**SQL**存取程式的程式庫！
- <https://dataset.readthedocs.io/en/latest/>
- 要另外安裝**Dataset**工具
  - `pip install dataset`
- 我們已知，
  - 儘管在「**關係數據庫(Relational database)**」中管理數據有很多好處，為什麼很少在中小型數據集的日常工作中使用它呢？
  - 儘管它們很難以增量方式查詢和更新，為什麼我們看到大量數據以**CSV**或**JSON**格式存儲在靜態文件中呢？
- 答案是程序員很懶！因此他們傾向於或更喜歡找到最簡單的解決方案。

# 主要優點：

- 在Python，數據庫不是儲存一堆結構化數據的最簡單解決方案。這就是dataset工具，打算要改變的東西！
- ●「Dataset工具」提供了一個簡單的抽象層，無需完整的ORM模型(可以刪除多數的直接SQL命令語句)，而且可以像JSON文件或NoSQL來使用數據庫。
- 主要優點：
  - 自動建立綱要(Automatic schema)：可忽略綱要就直接使用資料庫。如果寫入的表格或某行欄位的key, 在數據庫中不存在，則將自動創建它。
  - 有查詢助手(Query helper):方便查詢，例如all表中的行或distinct一組列中的所有值。
  - 具相容性(Compatibility)：建立在SQLAlchemy基礎上，所以dataset工具，可以配合其他主要數據庫一起使用，例如SQLite，PostgreSQL和MySQL。

# ●特別適合Dataset工具的場景與時機

- **Dataset** 庫有一個簡單但功能強大的API，因此我們可以很容易的，先把資料保存下來，之後再進行數據的整理！
- 因為**Dataset**工具 建立在 **SQLAlchemy**基礎上，所以很容易進行擴展。例如使用**Django**內建的**inspectdb**的管理命令，就可以把底層資料庫模型導入 **Django**中，所以，要與現有資料庫一同工作不會出現任何障礙。
- ●特別適合**Dataset**工具的場景與時機：
  - 當我們想要在不確定最終資料庫表(**Table**)會是什麼樣的情況下，可以快速收集資料，並保存到資料庫的時候，就是選用**Dataset**工具的最佳選擇。

## 3-5-1: 資料庫，要解決的CRUD的問題

### ●【check2】SQL如何處理資料庫CRUD的問題:

- (Create)新增資料
  - insert into 表名 values('資料1', '資料2', '資料3');
  - insert into 表名 (資料項1, 資料項2, 資料項3) values ('資料1', '資料2', '資料3');
- (Read)讀取資料
  - select 資料項 from 表名 where 條件;
- (Update)更新資料
  - update 表名 set 屬性項 where 表名;
- (Delete)刪除資料
  - delete from 表名 where 條件;
  - delete from 表名; //表中資料全部刪除

# 用「Dataset工具」來面對資料庫 CRUD的四種常用處理

- 配合**SQL**的指令，簡單說明如下：
  - 新增(Create):效果相當於SQL的Insert Into., 可直接用 `table.insert(dict(..))` 加入即可。(※若有新的key，就全體都增加新key的欄位！)
  - 讀取(Read):效果相當於SQL的Select \* From table Where ... 可直接用 `table.find(..)` 或 `.fine_one(..)` 執行
  - 更新(Update):UPDATE sometable SET gender="male男生" WHERE ...。可用進階的`db.query(statement)`來執行
  - 刪除>Delete):DELETE FROM customers WHERE ...。可用進階的`db.query(statement)`來執行

## 3-5-2: Dataset工具\_\_基本實驗\_解決CRUD的問題

- 如果不使用 **Dataset** 工具的程式碼：除了會更多行，也更複雜

```
1 ##### !!!!!!!!!!!!!!! #####
2 #Dataset工具，是另外安裝哦！ $pip install dataset
3 ##### !!!!!!!!!!!!!!! #####
4 import dataset
5 db2 = dataset.connect('sqlite:///dataset_simple_test.db')
6
7 table = db2['sometable']
8 table.insert(dict(name='Peop111第一人', age=50))
9 table.insert(dict(name='Peop222第二人', age=100, gender='female女生'))
10
11 john = table.find_one(name='Peop111第一人')
12 print("第一人find_one的內容", end=""); print(john)
13 list(table.find())
```

第一人find\_one的內容OrderedDict([('id', 1), ('name', 'Peop111第一人'), ('age', 50), ('gender', None)])

```
[OrderedDict([('id', 1),
              ('name', 'Peop111第一人'),
              ('age', 50),
              ('gender', None)]),
 OrderedDict([('id', 2),
              ('name', 'Peop222第二人'),
              ('age', 100),
              ('gender', 'female女生')])]
```

```
1 results = table.find(id=1)
2 list(results)
```

```
[OrderedDict([('id', 1),
              ('name', 'Peop111第一人'),
              ('age', 50),
              ('gender', None)])]
```



# 用query執行進階的Update, Delete

```
1 #更新 id=4的人, 把原來None的性別, 設為male男生
2 #更新Update資料項
3 statement = 'UPDATE sometable SET gender="male男生" WHERE id=4;'
4 db2.query(statement)
5
6 statement = 'SELECT * FROM sometable'
7 #statement = 'SELECT id, gender, COUNT(*) gender FROM sometable'
8 for row in db2.query(statement):
9     print( row['id'], row['gender'])
```

```
1 None
2 female女生
3 None
4 male男生
```

```
1 #增加Create新資料項
2 #INSERT INTO sometable VALUES ( 1, ... );
3 data = dict(title='Add_Title抬頭', name='ABC姓名', gender="female女生")
4 table.insert(data)
5 #印出全部資料的
6 statement = 'SELECT * FROM sometable'
7 for row in db2.query(statement):
8     print( row['id'],row['name'],row['gender'],row['title'])
```

```
1 Peop111第一人 None None
2 Peop222第二人 female女生 None
3 None None I am a banana第三人!!!!!!!!!!!!!!!!!!!!
4 None male男生 None
5 ABC姓名 female女生 Add_Title抬頭
```

```
1 #刪除(Delete)已有的資料
2 #DELETE FROM customers WHERE name='ABC姓名';
3 statement="DELETE FROM sometable WHERE name='ABC姓名'"
4 db2.query(statement)
5 #印出全部資料的
6 statement = 'SELECT * FROM sometable'
7 for row in db2.query(statement):
8     print( row['id'],row['name'],row['gender'],row['title'])
```

```
1 Peop111第一人 None None
2 Peop222第二人 female女生 None
3 None None I am a banana第三人!!!!!!!!!!!!!!!!!!!!
4 None male男生 None
```

### 3-5-3: Dataset工具—解決CRUD的問題。\_建立專案需要的「諾貝爾得獎資料庫Nobel」

- 經過前面 Dataset工具的基本實驗。可看出：
- Dataset工具：
  - 是python的模塊，特別設計的SQL數據庫
  - 希望可消除一大堆形式上的無謂手續。(例如傳統程式庫要求必須制定的綱要Schema定義。)

```
: 1 ##### !!!!!!!!!!!!!!! #####  
2 #Dataset工具，是另外安裝哦！ $pip install dataset  
3 ##### !!!!!!!!!!!!!!! #####  
4 import dataset  
5 db = dataset.connect('sqlite:///./nobel_prize.db')
```

現在，使用最早的nobel\_winners數據集（Example3-1）插入一些獲獎者字典。

我們使用數據庫交易(Transaction)和with語句來有效地插入我們的對象，然後提交它們。

```
|: 1 ## 在insert之後，在下一格的 find()就可發現已正確放入資料庫。  
2 with db as tx:  
3     for w in nobel_winners:  
4         tx['winners'].insert(w) ## 就是Insert into到Table
```

```
|: 1 #上面程式，使用with語句可確保將事務tx提交到數據庫db裡面。
```

檢查一下，看一切是否順利：

```
|: 1 list(db['winners'].find())  
  
|: [OrderedDict([('id', 1),  
                ('category', 'Physics'),  
                ('name', 'Albert Einstein'),  
                ('nationality', 'Swiss'),  
                ('sex', 'male'),  
                ('year', 1921)]),  
    OrderedDict([('id', 2),  
                ('category', 'Physics'),  
                ('name', 'Paul Dirac'),  
                ('nationality', 'British'),  
                ('sex', 'male'),  
                ('year', 1933)]),  
    OrderedDict([('id', 3),  
                ('category', 'Chemistry'),  
                ('name', 'Marie Curie'),  
                ('nationality', 'Polish'),  
                ('sex', 'female'),  
                ('year', 1911)])]
```

可發現，獲獎者已正確插入，其插入順序由OrderedDict保留。數據集非常適合基於SQL的基本工作高級的操作，它使您可以使用查詢方法進入SQLAlchemy的核心API。

## 3-5-4: Dataset工具——若要有更進階的數據用法

- **Dataset**工具，基本上就很好用，若要進階用法
- 也可寫出正確**SQL**指令，呼叫**SQLAlchemy**的**query**來執行哦！

```
1 ## 加入 SQL的查詢方法。 ## 成功！  
2 statement = 'SELECT * FROM winners'  
3 for row in db.query(statement):  
4     print(row['id'], row['name'], row['year'])  
5 # 1 Albert Einstein 1921  
6 # 2 Paul Dirac 1933  
7 # 3 Marie Curie 1911
```

```
1 Albert Einstein 1921  
2 Paul Dirac 1933  
3 Marie Curie 1911
```

## Exercise3-5

- **Ex3-5Add1:** 為何**Dataset**工具會被戲稱為「懶人程式庫**Dataset**」？
- **Ex3-5Add2:** **SQL**資料庫要解決**CRUD**的問題, 請問**CRUD**是那四個英文字分別代表那四種工作？請配合**Sql**說明。
- **Ex3-5Add3:** 用「**Dataset**工具」來處理數據，會有那些優點？
- **Ex3-5Add4:** 用「**Dataset**工具」來面對資料庫**CRUD**的四種常用處理，請配合**SQL**的指令，簡單說明。
- **Ex3-5Add5:** **Dataset**工具在執行**find(條件)**之後，得到的是什麼樣的容器物件型態？請問此型態與 **Dict**有何差異？有何好處？

## 3-6: NoSQL--MongoDB

- 也要先安裝好mongoDB。
  - \$pip install pymongo
- MongoDB是基於文檔(document-based)的通用NoSQL數據庫。
- 非關聯式資料庫（**NoSQL**）的特色：
  - NoSQL是非關聯式資料庫，當中有許多的設計方式，會與關聯式DBMS有所不同。主要差別：
    - 不需要事先定義好資料綱要(schema) 以及資料之間的關聯
    - 可以自由新增欄位，不需要回頭修改過去的資料文件(document)
    - 可以自由定義資料文件 (document) 的結構

# 安裝Robo3T\_MongoDB的GUI管理工具

## 建立connection在Local:27017

The image is a collage of screenshots from Robo 3T and Jupyter Notebook, illustrating the steps to connect to MongoDB and query data.

**Robo 3T - 1.4** (Top Left): Shows the "New Connection" dialog box. The "Name" field is "New Connection" and the "Address" field is "localhost:27017".

**Jupyter Notebook** (Top Middle): Shows a code cell with the following code:

```
18 ]
19 #print(nobel_winners)
```

Below the code cell, there is a text box with the following text:

如果，資料庫，已存在獲獎者資料集合(winners collection)，它將對其現在的情況)會創建它。

**Robo 3T - 1.4** (Top Right): Shows the "New Connection (3)" dialog box. The "Name" field is "New Connection" and the "Address" field is "localhost:27017".

**Jupyter Notebook** (Middle): Shows a code cell with the following code:

```
In [221]: 1 #####!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
2          2 要先安裝pymongo的工具庫 $pip install pymongo,
3          3 #####!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
4          4 from pymongo import MongoClient
5          5 client = MongoClient() #建立 Mongo的客戶端 client
6          6 db = client.nobel_prize #資料庫 database
7          7 coll = db.winners #群集 collection
```

**Robo 3T - 1.4** (Bottom Left): Shows the "New Connection (4)" dialog box. The "Name" field is "New Connection" and the "Address" field is "localhost:27017".

**Jupyter Notebook** (Bottom Middle): Shows a code cell with the following code:

```
from pymongo import MongoClient
def get_mongo_database(db_name, host='localhost', \
                        port=27017, username=None, password=None):
    conn = MongoClient(host, port)
    else:
        conn = MongoClient(host, port)
    return conn[db_name]
```

Below the code cell, there is a text box with the following text:

這樣就可以輕鬆插入我們的諾貝爾獎數據集：

**Robo 3T - 1.4** (Bottom Right): Shows the "New Connection (4)" dialog box. The "Name" field is "New Connection" and the "Address" field is "localhost:27017".

**Jupyter Notebook** (Bottom): Shows a code cell with the following code:

```
231: 1 db = get_mongo_database(DB_NOBEL_PRIZE)
2     2 coll = db[COLL_WINNERS] #另一種可行的寫法 coll = db.COLL_WINNERS
```

Below the code cell, there is a text box with the following text:

232: 1 ## 刪除 collection
2 2 coll.drop()

233: 1 ##### 到此才有 insert 新的資料表格產生哦! #####
2 2 ## coll.insert(nobel\_winners) ##舊版寫法
2 3 coll.insert\_many(nobel\_winners)
2 4 ##### insert是舊版用法，新版應該改為: insert\_one or insert\_many

233: <pymongo.results.InsertManyResult at 0x93cf440>

**Robo 3T - 1.4** (Bottom Right): Shows the "New Connection (4)" dialog box. The "Name" field is "New Connection" and the "Address" field is "localhost:27017".

**Jupyter Notebook** (Bottom): Shows a code cell with the following code:

```
1 db.getCollection('winners').find({})
```

Below the code cell, there is a text box with the following text:

Insert後可見到Nobel與collection產生 (要Fresh更新才見的到哦！)

# 用MongoDB建立資料庫與查詢

- 可發現\_id的內容不是一般會自動增加的數字id而已，而是ObjectID

```
1 #####!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
2 ##要先安裝pymongo的工具庫 $pip install pymongo,
3 #####!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
4 from pymongo import MongoClient
5 client = MongoClient() #建立 Mongo的客方端 client
6 db = client.nobel_prize #資料庫 database
7 coll = db.winners #群集 collection
```

1 建議，最好用「常數變數來取用資料庫」，以免發生，不同英文

```
1 #db = client.nobel_prize
2 #db = client['nobel_prize']
```

```
1 DB_NOBEL_PRIZE = 'nobel_prize'
2 COLL_WINNERS = 'winners'
3
4 db = client[DB_NOBEL_PRIZE]
5 coll = db[COLL_WINNERS]
```

```
1 coll.drop()
```

```
1 ## coll.insert(nobel_winners) ##舊版寫法
2 coll.insert_many(nobel_winners)
3 ##### insert是舊版用法，新版應該改為： insert_c
```

<pymongo.results.InsertManyResult at 0x8830d80>

```
1 #印出目前資料庫內容
2 res = coll.find()
3 list(res)
```

```
[{'_id': ObjectId('5f7b42973356fd4c3108bc55'),
  'category': 'Physics',
  'name': 'Albert Einstein',
  'nationality': 'Swiss',
  'sex': 'male',
  'year': 1921},
 {'_id': ObjectId('5f7b42973356fd4c3108bc56'),
  'category': 'Physics',
  'name': 'Paul Dirac',
  'nationality': 'British',
  'sex': 'male',
  'year': 1933},
 {'_id': ObjectId('5f7b42973356fd4c3108bc57'),
  'category': 'Chemistry',
  'name': 'Marie Curie',
  'nationality': 'Polish',
  'sex': 'female',
  'year': 1911}]
```



# 在MongoDB進行find

## 在MongoDB進行find

- 1 在得獎者收藏夾(winner collections)中有一些項目(item)，MongoDB的查找方法可以通過字典(dict)來設定查詢條件，使查找變得非常容易：

```
1 res = coll.find({'category': 'Chemistry'})
2 list(res)
```

```
[{'_id': ObjectId('5f7b42973356fd4c3108bc57'),
  'category': 'Chemistry',
  'name': 'Marie Curie',
  'nationality': 'Polish',
  'sex': 'female',
  'year': 1911}]
```

- 1 ※有許多特殊的「\$」前綴運算符，可以進行複雜的查詢。
- 2 可使用 \$gt (大於) 運算子找出1930年以後的所有贏家：

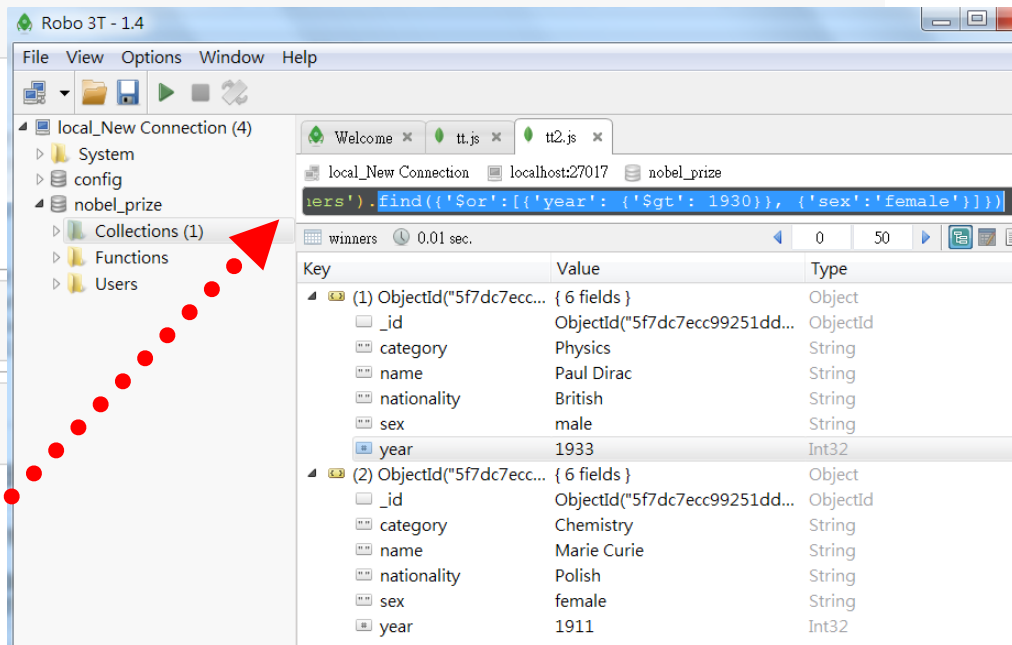
```
1 res = coll.find({'year': {'$gt': 1930}})
2 list(res)
```

```
[{'_id': ObjectId('5f7b42973356fd4c3108bc56'),
  'category': 'Physics',
  'name': 'Paul Dirac',
  'nationality': 'British',
  'sex': 'male',
  'year': 1933}]
```

- 1 例如，您還可以使用布林表達式來查找1930年之後的所有得獎者或所有女性得獎者：

```
1 res = coll.find({'$or': [{'year': {'$gt': 1930}}, \
2                      {'sex': 'female'}]})
3 list(res)
```

```
[{'_id': ObjectId('5f7b42973356fd4c3108bc56'),
  'category': 'Physics',
  'name': 'Paul Dirac',
  'nationality': 'British',
  'sex': 'male',
  'year': 1933},
 {'_id': ObjectId('5f7b42973356fd4c3108bc57'),
  'category': 'Chemistry',
  'name': 'Marie Curie',
  'nationality': 'Polish',
  'sex': 'female',
  'year': 1911}]
```





# MongoDB的ObjectId具有很多「隱藏功能」

- MongoDB的ObjectId具有很多「隱藏功能」，比簡單的隨機id或自動增量的id都要好得多！
- 例如，您可以獲取ObjectId的生成時間，從而可以訪問方便的時間戳記。而且在ObjectId中隱藏有電腦機器、行程的資訊，方便在分散式環境的數據處理時。詳情，請參考MongoDB的說明文件。

```
1 import bson
2 oid = bson.ObjectId()
3 oid.generation_time
4 |
5 #out: datetime.datetime(2020, 9, 28, 17, 3, tzinfo=<bson.tz_util.FixedOffset object at 0x0000000008378430>)
6
```

```
datetime.datetime(2020, 10, 5, 16, 43, 42, tzinfo=<bson.tz_util.FixedOffset object at 0x00000000055232E0>)
```

```
1 res = coll.find()
2 oid=res[0]['_id'] # '_id': ObjectId('5f7b42973356fd4c3108bc55')},
3 print(oid)
4 oid.generation_time
```

```
5f7b4d363356fd4c3108bc66
```

```
datetime.datetime(2020, 10, 5, 16, 43, 34, tzinfo=<bson.tz_util.FixedOffset object at 0x00000000055232E0>)
```

# 好用的工具副程式\_\_將目前的資料轉為dict

- 可將新的獲獎者收藏轉換成Python字典的列表(list of Dictionary)。
- 為該任務創建一個實用程序函數：

## 好用的工具副程式\_\_將目前的資料轉為dict

```
1 您可以在MongoDB文檔中找到可用查詢表達式的完整列表。
2
3 可將新的獲獎者收藏轉換成Python字典的列表(list of Dictionary)。
4 為該任務創建一個實用程序函數：
```

```
1 def mongo_coll_to_dicts(dbname='test', collname='test', \
2                           query={}, del_id=True, **kw):
3     db = get_mongo_database(dbname, **kw)
4     res = list(db[collname].find(query))
5     if del_id:
6         for r in res:
7             r.pop('_id')
8     return res
```

- MongoDB是無綱要結構的數據庫(No Schema SQL)，非常適合單人工作或小型團隊的快速原型製作。
- 另外，在正式綱要(schema)成為有用的參考和健全性檢查時，特別是對於大型代碼庫。那麼，在選擇數據模型時，可以輕鬆調整文件表單也是一個優點。

## Exercise3-6

- **Ex3-6Add1:** MonogoDB是非關聯式資料庫（NoSQL），相對於RDBMS有那些特點？
- **Ex3-6Add2:**用MongoDB建立資料時，與一般資料庫加入自動增量的id不同，而是採用ObjectID的設計。請問這樣有何好處？
- **Ex3-6Add3:**MongoDB是NoSQL中很適合入門數據師採用，可能的適用場合有那些？

## 3-7: 處理日期時間的特殊情況，在python與JavaScript之間的問題

- ※註：在MongoDB數據庫裡，有ObjectID的資訊，可以查詢建立時間。
- 但是，在很多情況，時間都沒有這麼方便，而需要數據處理時，進行各平台的轉換。所以就很容易發生意想不到的問題。
- 建議使用國際標準組織（ISO）8601時間格式作為日期和時間的字符串表示形式，並使用協調世界時（UTC）格式

- 以下是一些ISO 8601日期和日期時間字符串的示例：

- ❑ <date>T<time>      ※格式：%Y-%m-%d T %H:%M:%S
- ❑ <time>±hh:mm
- ❑ <time>Z      (註：Z是零UTC偏移量的區域標記)
- ❑ 2020-10-07T20:54:50.568948+08:00(+8時區微秒單位)
- ❑ 2020-10-07T20:40:34+08:00      (台北是 +8 時區)
- ❑ 特別的，若零時區可在後加一大寫字母Z表示。
- ❑ 2020-10-07T12:21:42Z

- 換算：台北(Taipei)時間=UTC+ 8:00

```
1 ##台北 utc+8 時區
2 from datetime import datetime
3 d = datetime.now()
4 d.isoformat()
```

```
'2020-10-07T21:22:24.703560'
```

```
1 d.astimezone().isoformat()
```

```
'2020-10-07T21:22:24.703560+08:00'
```

```
1 d.replace(microsecond=0).isoformat()
```

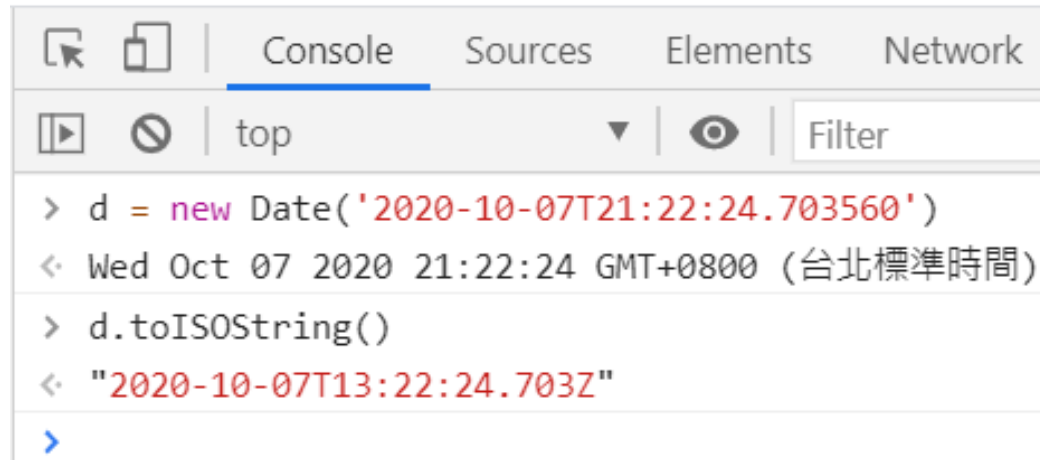
```
'2020-10-07T21:22:24'
```

```
1 ## utc +0時區
2 d2= datetime.utcnow()
3 d2.isoformat()
```

```
'2020-10-07T13:22:25.447602'
```

# Lab#3-7 把時間字串存存JSON再由JavaScript讀到，而且建一個Data物件

○



```
> d = new Date('2020-10-07T21:22:24.703560')
< Wed Oct 07 2020 21:22:24 GMT+0800 (台北標準時間)

> d.toISOString()
< "2020-10-07T13:22:24.703Z"

>
```

- 請注意，在從Python到JavaScript的旅程中，我們損失了一些分辨率，然後又返回，後者(Javascript)的時間是毫秒單位，而不是微秒！所以會產生誤差！
- 在一般的dataviz工作中，這不大可能成為問題，但是最好記住，以防萬一發生一些奇怪的時間錯誤。
- 請看下頁的示範。

# Lab#3-7\_Pyjs\_轉換後的時間\_最末秒數產生誤差

!

因為py是毫秒單位  
而JavaScript是微秒單位

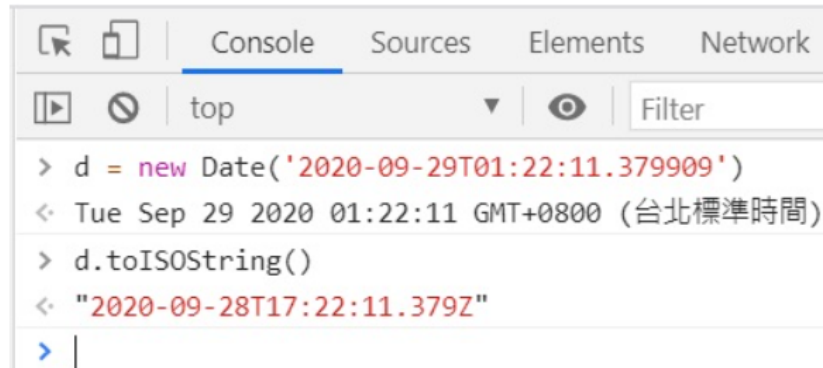
```
In [136]: 1 from datetime import datetime
          2
          3 d = datetime.now()
          4 d.isoformat()
```

```
Out[136]: '2020-09-29T01:22:11.379909'
```

1 然後可以將該字符串保存為JSON或CSV，由JavaScript讀取，並用於創建Date對象：

```
In [141]: 1 Image('jpg_Lab#3-7把時間字符串存JSON再由JavaScript讀到，而且建一個Data物件.jpg')
```

```
Out[141]:
```



```
> d = new Date('2020-09-29T01:22:11.379909')
< Tue Sep 29 2020 01:22:11 GMT+0800 (台北標準時間)
> d.toISOString()
< "2020-09-28T17:22:11.379Z"
> |
```

```
1 d = new Date('2020-09-29T01:22:11.379909')
2
3 d.toISOString()
```

```
In [142]: 1 from dateutil import parser
          2
          3 d = parser.parse("2020-09-28T17:22:11.379Z")
          4 d
```

```
Out[142]: datetime.datetime(2020, 9, 28, 17, 22, 11, 379000, tzinfo=tzutc())
```

## Exercise 3-7

- **Ex3-7Add1:**在數據處理中，清洗資料是很基本的工作，有時在數據移動中也可能發生問題！請問python與JavaScript在移動「日期時間」資料時，可能會出現的問題是什麼？
- **Ex3-7Add2:**如果收集跨國包括時間的資料，則會有時區的問題，為了表示與UTC相差0的時間，可用那個英文字母來簡單註明？請問台北的時區相對於UTC時間要如何計算？

# ●【check3】ch03實作練習2\_3-8 補充：爬蟲相關 --- (1)爬JSON數據 (2)爬CSV數據

- **Json**的方法：直接爬下**JSON**數據，並且解碼成字典物件**dict**。

網路上opendata的 **JSON** 用爬蟲方法，爬政府資料開放平台的 opendata <https://data.gov.tw/>

不動產實價登錄資訊-買賣案件-中和區 <https://data.gov.tw/dataset/146474>

```
1 ##網路上opendata的 JSON      用爬蟲方法，爬政府資料開放平台的 opendata      https://data.gov.tw/
2 # 不動產實價登錄資訊-買賣案件-中和區      https://data.gov.tw/dataset/146474
3 import requests
4 http='https://quality.data.gov.tw/dq_download_json.php?nid=146474&md5_url=cca3539e4734542313e6400
5 response = requests.get(http)  ##直接get會自動生成一個session()
6 #client = requests.session()
7 #response = client.get(url=http) # 執行 get 的http， ##另一個常見的是post
8 dic=response.json()
9 for i in range(3):
10     #print(dic[i])
11     print(dic[i]['district'], " 地段：" , dic[i]['rps02'][:16], " 平方公尺=",dic[i]['rps03'], " 總價=")
```

中和區 地段： 錦和段316地號 平方公尺= 14.4 總價= 1380960 每平方公尺價格= 1380960

中和區 地段： 新北市中和區興南路一段105巷2 平方公尺= 12.87 總價= 6900000 每平方公尺價格= 6900000

中和區 地段： 新北市中和區國光街102巷27號 平方公尺= 16.98 總價= 8100000 每平方公尺價格= 8100000



# csv方法一：直接爬下csv數據，就當作文字檔處理

csv方法一：直接爬下csv數據，就當作文字檔處理

```
: 1 ##方法一：直接爬下csv檔，就當作文字檔處理
2 ##網路上opendata的 JSON 用爬蟲方法，爬政府資料開放平台的 opendata https://data.gov.tw/
3 import requests
4 #http='https://quality.data.gov.tw/dq_download_csv.php?nid=125729&md5_url=0d9d46c4c71e7
5 # csv 2024年版
6 http='https://quality.data.gov.tw/dq_download_csv.php?nid=146474&md5_url=cca3539e473454
7 response = requests.get(http) ##直接get會自動生成一個session()
8 dicts=response.text|
9 dict=dicts.split('\n')
10 data_first_lint=dict[0].split(',')
11 for i in range(1,4):
12     datas=dict[i].split(',')
13     for j in range(len(datas)):
14         if data_first_lint[j]=='district':
15             print(datas[j] + ', ', end='')
16         if data_first_lint[j]=='rps02':
17             print(" 地段" + datas[j] + ', ', end='')
18         if data_first_lint[j]=='rps03':
19             print(" 平方公尺" + datas[j] + ', ', end='')
20     print()
```

中和區， 地段錦和段316地號， 平方公尺14.4，  
中和區， 地段新北市中和區興南路一段105巷20弄17號三樓， 平方公尺12.87，  
中和區， 地段新北市中和區國光街102巷27號四樓， 平方公尺16.98，

csv方法二：直接爬下csv的數據，然後先存為csv檔，再用 csv.DictReader(csvfile)轉為 dict 比較方便處理。

csv方法二：直接爬下csv的數據，然後先存為csv檔，再用 csv.DictReader(csvfile)轉為dict 比較方便處理。

```
: 1 ##網路上opendata的 JSON      用爬蟲方法，爬政府資料開放平台的 opendata      https://data.gov.tw/
2 import requests
3 #http='https://quality.data.gov.tw/dq_download_csv.php?nid=125729&md5_url=0d9d46c4c71e732c3449bf4
4 # csv 2024年版
5 http='https://quality.data.gov.tw/dq_download_csv.php?nid=146474&md5_url=cca3539e4734542313e64005
6 response = requests.get(http) ##直接get會自動生成一個session()
7 dicts=response.text
8 with open('in.csv', 'w', encoding='utf-8') as f:
9     f.write(dicts)
```

```
: 1 print('\ndat準備好了---用文字檔讀檔檢查檔案的內容')
2 print('-'*60)
3 ##用方法三印出內容、●● 這是配合 with 的高效率寫法！
4 with open('in.csv',"r", encoding='utf-8') as fp:
5     #for line in fp:
6     for i, row in enumerate(fp):
7         if i<5:
8             print(row, end='')

```

dat準備好了---用文字檔讀檔檢查檔案的內容

-----  
district,rps01,rps02,rps03,rps04,rps05,rps06,rps07,rps08,rps09,rps10,rps11,rps12,rps13,rps14,rps15,rp  
s16,rps17,rps18,rps19,rps20,rps21,rps22,rps23,rps24,rps25,rps26,rps27,rps28,rps29,rps30,rps31  
中和區,土地,錦和段316地號,14.4,都市：其他:人行步道,,,1120105,土地1建物0車位0,,,其他,,,,,0,0,0,0,有,無,13809  
0,95900,,,0,0,包含公共設施保留地用地；,RPRNMLLKJHIGFHF97EA,0,0,0,無

## Exercise 3-8

- **Ex3-8Add1:**如果有Json的網路資料，打算直接爬下JSON數據，並且解碼成字典物件dict，請完成這段簡單的python爬蟲程式。
- **Ex3-8Add2:**如果有csv的網路資料，打算先存成csv檔，再解碼成字典物件dict，請完成這段簡單的python爬蟲程式。

---

- Thank for your attention.