第一章：Python 程設語言 5
1.1 基本概念 5
交互式編程.....5
腳本.....6
Python 中文编码 7
Python 保留字符 7
行和縮排.....8
多行語句.....9
Python 引號 10
Python 註釋 10
Python 空行 11
多個語句構成代碼組.....12
Python 變數類型 12
Python 數據類型轉換 15
print():列印輸出內容 18
String（字串） 18
List (串列).....19
Tuple（元組） 23
Dictionary（字典） 24
1.2 運算符 28
比較運算符.....28
賦值運算符.....28
位運算符.....29
運算符優先級.....29
1.3 條件語句&循環語句.....31
if…in.....33
循環語句 for 33
break 與 continue 命令 36
循環使用 for else.....36
While 循環語句 36
無限循環 while.....37
循環使用 while else 37
1.4 函數 39
1.5 模組(module).....40
1.6 補充 42
區域變數 vs 全域變數 42
type()函式來顯示資料型態.....42

*args 的用法 (一個星號) 42
**kwargs 的用法(二個星號) 43
dir()的用法 43
python 研究-with as 用法 44
help()的用法 45
Decorator(裝飾器) 45
1.6 物件導向(Object Oriented Programming)程式開發 50
定義類別(Create Object) 50
The __init__() Function 50
封裝(Encapsulation) 51
繼承(Inheritance) 52
覆寫函式(Override) 53
classmethod() 55
1.7 套件(Package) 56
1.8 例外處理(try except) 58
1.9 多執行緒(Multi-Threads) 60
建立子執行緒 61
多個子執行緒與參數 62
Multi threads by overriding the run() method in a subclass 64
1.10 Python 處理 JSON 66
如何建立 JSON 字串 66
JSON 在 python 中的使用 67
第三章：Python Web Djangle 71
Django 網站框架 (Python)教學 73
3-1 Django 平台建置(windows) 74
3-1 Django 平台建置(windows+ Visual Studio Code) 83
3-1 複製專案 88
3-1 Django 平台建置(Linux) 91
3-1 Django 平台建置(Linux+ Visual Studio Code+Remote Development) 98
3-2 建立 Django 專案 106
3-3 建立 Application 應用程式 106
3-4 視圖(view)與 url 112
Diango 的 Framework 架構 112
設定 urls.py 114
加入 static 靜態檔案 117
從網址中載取資料 119
3-5 視圖、模版與 Template 語言 122

3-6 模板(Template)語言-變量.....124
變量.....124
3-7 模板(Template)語言-標籤.....126
3-8 以 GET 及 POST 傳送資料.....132
3-9 Django 資料庫連結與應用(使用 MariaDB).....146
安裝 MariaDB.....146
解決 MariaDB(10.5.12)本地無密碼直接登錄的問題.....146
Setting MySQL/MariaDB root password.....146
MySQL 開啟遠端連線權限允許遠端裝置連線資料庫.....148
使用 Wordbench 開啟 MariaDB 資料庫.....152
How to Reset MySQL/MariaDB Database Root Password?.....153
安裝 mysqlclient 與設定 Django 參數.....154
Field Type 欄位型別.....158
Field Option 欄位選項.....160
3-11 資料庫新增、刪除、修改與查詢(使用 SQL 語法).....163
資料庫查詢(SQL-SELECT 語法).....165
資料新增(SQL-INSERT 語法).....172
資料刪除(SQL-DELETE 語法).....175
資料更新(使用 GET) (SQL-UPDATE 語法).....177
資料更新(使用 GET) (SQL-UPDATE 語法).....181
3-12 資料庫查詢 (使用 ORM).....184
SQLite vs MySQL vs PostgreSQL.....185
資料庫查詢(SQL-SELECT 語法).....189
資料新增(SQL-INSERT 語法).....196
資料刪除(SQL-DELETE 語法).....198
資料更新(使用 GET) (SQL-UPDATE 語法).....199
資料更新(使用 GET) (SQL-UPDATE 語法).....203
3-13 Django 使用 ORM vs SQL 語法.....206
建立環境.....206
ORM 欄位與引數說明.....206
建立資料表.....207
新增、更新、刪除.....231
多資料表關聯查詢.....234
使用 JOIN 結合資料表 by SQL.....234
Django ORM 一對一、一對多、多對多.....238
UPDATE using INNER JOIN.....252
Delete using INNER JOIN.....252
3-14 Cookies 與 Sessions.....253

建立環境.....253
關於 Cookie 253
關於 Session 254
Cookie 的使用 254
Session 的使用 262
3-15 使用者管理 271
建立環境.....271
讀取 Django auth 使用者.....271
HttpRequest.user 物件 272
登入和登出.....275
Registration 小專案(會員註冊與登入).....279

第一章：Python 程設語言

1989 年，人在阿姆斯特丹的 Guido van Rossum 於耶誕假期著手開發 Python ，其目的是設計出一種優美而強大，提供給非專業程式設計師使用的語言，同時採取開放策略，使 Python 能夠完美結合如 C 、 C++ 和 Java 等其他語言。時至今日， Python 已經是相當受歡迎的入門教學語言。

Python 程式語言有著程式碼易學、易讀、清晰等特性，因而被廣泛作為入門程式語言教授，具有跨平台的特性加上強悍完整的模組支援，許多網頁程式或是系統管理都是可以透過 Python 來完成。而在 Raspberry Pi Linux 開放系統的支援之下，顛覆了以往 Python 難以控制硬體的印象。

傳統程式言語的學習過程，由於語法複雜且缺乏互動而使得學習程式枯燥乏味且易產生挫折感，導致多樣性功能應用目的開發產生高門檻。透過 Raspberry Pi 開放硬體架構，Python 高階語言的學習得以變得更為全面，Python 豐富的函式庫，讓開發者足以應付許多中大型專案的需求。

教學資源:

https://www.python.org/
http://www.w3big.com/zh-TW/python3/default.html
http://www.w3big.com/
https://www.w3schools.com/python/
https://www.tutorialspoint.com/index.htm
http://www.w3big.com/python/default.html
http://www.runoob.com/

指令下達範例：

```
[root@ localhost ~]# apt-get  install  python  python3
```

1.1 基本概念

➢ 交互式編程

交互式編程不需要創建腳本文件，是通過 Python 解釋器的交互模式進來編寫代碼。linux 上你只需要在命令行中輸入 Python 命令即可啟動交互式編程,提示窗口如下：

指令下達範例：

```
[root@ localhost ~]# python
➔print(“Hello Python”)
[root@ localhost ~]# exit()
(離開直譯器的互動介面)
[root@ localhost ~]# python3
➔print(“Hello Python”)
[root@ localhost ~]# exit()
(離開直譯器的互動介面)
```

- **腳本**

直譯器的互動介面便於直接測試程式碼，程式碼也可以寫到副檔名為 .py 的檔案之中：

通過腳本參數調用解釋器開始執行腳本，直到腳本執行完畢。當腳本執行完成後，解釋器不再有效。讓我們寫一個簡單的 Python 腳本程序。所有 Python 文件將以.py 為擴展名。將以下的源代碼拷貝至 test.py 文件中。

1、假設你已經設置了 Python 解釋器 PATH 變量
指令下達範例：

```
[root@ localhost ~]# nano  hello.py
內容:
print("Hello,Python!")
[root@ localhost ~]# python  hello.py
[root@ localhost ~]# python3  hello.py
```

2、假定您的 Python 解釋器在/usr/bin 目錄中，使用以下命令執行腳本：
指令下達範例：

```
[root@ localhost ~]# which  python
[root@ localhost ~]# nano  hello.py
#!/usr/bin/python
print("Hello,Python!")
```

```
print("您好")

[root@ localhost ~]# chmod +x hello.py

[root@ localhost ~]# ./hello.py
```

➢ Python 中文编码

Python 文件中如果未指定編碼，在執行過程會出現錯誤：

```
root@raspberrypi:~# ./hello.py
  File "./hello.py", line 3
SyntaxError: Non-ASCII character '\xe6' in file ./hello.py on li
ne 3, but no encoding declared; see http://www.python.org/peps/p
ep-0263.html for details
```

解決方法為只要在文件開頭加入#coding=utf-8。
指令下達範例：

```
[root@ localhost ~]# nano  01.py

內容:

#!/usr/bin/python

#coding=utf-8

print("Hello,Python!")

print("您好")

[root@ localhost ~]# ./01.py
```

範例 basic-01.py

➢ Python 保留字符

下面的列表顯示了在 Python 中的保留字。這些保留字不能用作常數或變數，或任何其他標識符名稱。所有 Python 的關鍵字只包含小寫字母。

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

➢ 行和縮排

學習 Python 與其他語言最大的區別就是，Python 的代碼塊不使用大括號({}) 來控制類，函數以及其他邏輯判斷。python 最具特色的就是用縮進來寫模塊。

縮進的空白數量是可變的，但是所有代碼塊語句必須包含相同的縮進空白數量，這個必須嚴格執行。如下所示：

```
if True:
    print "True"
else:
  print "False"
```

以下代碼將會執行錯誤：

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
# 文件名：test.py

 if True:
    print "Answer"
    print "True"
else:
    print "Answer"
    # 没有严格缩进，在执行时保持
  print "False"
```

```
$ python test.py
  File "test.py", line 5
    if True:
    ^
IndentationError: unexpected indent
```

IndentationError: unexpected indent 錯誤是 python 編譯器是在告訴你"Hi，老兄，你的文件里格式不對了，可能是 tab 和空格沒對齊的問題"，所有 python 對格式要求非常嚴格。如果是 IndentationError: unindent does not match any outer indentation level 錯誤表明，你使用的縮進方式不一致，有的是 tab 鍵縮進，有的是空格縮進，改為一致即可。因此，在 Python 的代碼塊中必須使用相同數目的行首縮進空格數。建議在每個縮進層次使用「單個製表符」或「兩個空格」或「四個空格」，切記不能混用。

➢ 多行語句

Python 語句中一般以新行作為謂語句的結束符。但是我們可以使用斜杠(\)將一行的語句分為多行顯示，如下所示：

```
total = item_one + \
        item_two + \
        item_three
```

語句中包含[], {} 或() 括號就不需要使用多行連接符。如下實例：

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

➢ Python 引號

Python 接收單引號(')，雙引號(")，三引號(“”’”) 來表示字符串，引號的開始與結束必須的相同類型的。其中三引號可以由多行組成，編寫多行文本的快捷語法，常用語文檔字符串，在文件的特定地點，被當做註釋。

```
word = 'word'
sentence = "这是一个句子。"
paragraph = """这是一个段落。
包含了多个语句"""
```

➢ Python 註釋

python 中單行註釋採用 # 開頭

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
# 文件名：test.py

# 第一个注释
print "Hello, Python!";  # 第二个注释
```

输出结果：

```
Hello, Python!
```

注释可以在语句或表达式行末：

```
name = "Madisetti" # 这是一个注释
```

python 中多行注释使用三个单引号(''')或三个单引号(""")。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
# 文件名：test.py


'''
这是多行注释，使用单引号。
这是多行注释，使用单引号。
这是多行注释，使用单引号。
'''

"""
这是多行注释，使用双引号。
这是多行注释，使用双引号。
这是多行注释，使用双引号。
"""
```

➢ Python 空行

函數之間或類的方法之間用空行分隔，表示一段新的代碼的開始。類和函數入口之間也用一行空行分隔，以突出函數入口的開始。空行與代碼縮進不同，空行並不是 Python 語法的一部分。書寫時不插入空行，Python 解釋器運行也不會出錯。但是空行的作用在於分隔兩段不同功能或含義的代碼，便於日後代碼的維護或重構。記住：空行也是程序代碼的一部分。

➢ 多個語句構成代碼組

縮進相同的一組語句構成一個代碼塊，我們稱之代碼組。像 if、while、def 和 class 這樣的複合語句，首行以關鍵字開始，以冒號(:)結束，該行之後的一行或多行代碼構成代碼組。我們將首行及後面的代碼組稱為一個子句(clause)。如下實例：

```
if expression :
    suite
elif expression :
    suite
else :
    suite
```

➢ Python 變數類型

變數存儲在內存中的值。這就意味著在創建變數時會在內存中開闢一個空間。基於變數的數據類型，解釋器會分配指定內存，並決定什麼數據可以被存儲在內存中。因此，變數可以指定不同的數據類型，這些變數可以存儲整數，小數或字符。

Python 有五個標準的數據類型：

1、 Numbers（數字）
2、 String（字串）
3、 List（串列）
4、 Tuple（元組）
5、 Dictionary（字典）

Python 本身無陣列類型，可以使用 NumPy 來解決。

- 補充 NumPy：

NumPy 是 Python 在進行科學運算時，一個非常基礎的 Package，同時也是非常核心的 library，它具有下列幾個重要特色：

1、 提供非常高效能的多維陣列(multi-dimensional array)數學函式庫。
2、 可整合 C/C++及 Fortran 的程式碼。
3、 方便有用的線性代數(Linear Algebra)及傅立葉轉換(Fourier Transform)能力。
4、 利用 NumPy Array 替代 Python List。
5、 可定義任意的數據型態(Data Type)，使得能輕易及無縫的與多種資料庫

整合。

NumPy Array：

學習資料科學(Data Science)或機器學習(Machine Learning)時，利用 NumPy 在陣列的操作是非常重要，其主要功能都架構在多重維度(N-dimensional array)的 ndarray 上，ndarray 是一個可以裝載相同類型資料的多維容器，維度的大小及資料類型分別由 shape 及 dtype 來定義。通常我們會稱一維陣列為向量(vector)，二維陣列為矩陣(matrix)。

- 變數附值

Python 中的變數不需要聲明，變數的賦值操作既是變量聲明和定義的過程。每個變數在內存中創建，都包括變數的標識，名稱和數據這些信息。每個變數在使用前都必須賦值，變數賦值以後該變數才會被創建。等號（=）用來給變數賦值。等號（=）運算符左邊是一個變數名,等號（=）運算符右邊是存儲在變數中的值。例如：

```
#!/usr/bin/python

#coding=utf-8


counter = 100 # 赋值整型变量

miles = 1000.0 # 浮点型

name = "John" # 字符串


print counter

print miles

print name
```

範例 basic-02.py

- 多個變數賦值

指令下達範例：

```
[root@ localhost ~]# python

=>a=b=c=1
```

```
=>print a
=>print b
=>print c
=>a,b,c=1,2,”tony”
=>print a
=>print b
=>print c
```

- Numbers（數字）

數字數據類型用於存儲數值。他們是不可改變的數據類型，這意味著改變數字數據類型會分配一個新的對象。當你指定一個值時，Number 對象就會被創建：

当你指定一个值时，Number对象就会被创建：

```
var1 = 1
var2 = 10
```

您也可以使用del语句删除一些对象引用。

del语句的语法是：

```
del var1[,var2[,var3[....,varN]]]]
```

您可以通过使用del语句删除单个或多个对象。例如：

```
del var
del var_a, var_b
```

Python支持四种不同的数值类型：

- int（有符号整型）
- long（长整型[也可以代表八进制和十六进制]）
- float（浮点型）
- complex（复数）

实例

一些数值类型的实例：

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEI	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

- 长整型也可以使用小写"L"，但是还是建议您使用大写"L"，避免与数字"1"混淆。Python使用"L"来显示长整型。
- Python还支持复数，复数由实数部分和虚数部分构成，可以用a + bj,或者complex(a,b)表示， 复数的实部a和虚部b都是浮点型

➢ Python 數據類型轉換

有時候，我們需要對數據內置的類型進行轉換，數據類型的轉換，你只需要將數據類型作為函數名即可。以下幾個內置的函數可以執行數據類型之間的轉換。這些函數返回一個新的對象，表示轉換的值。

函数	描述
int(x [,base])	将x转换为一个整数
long(x [,base])	将x转换为一个长整数
float(x)	将x转换到一个浮点数
complex(real [,imag])	创建一个复数
str(x)	将对象 x 转换为字符串
repr(x)	将对象 x 转换为表达式字符串
eval(str)	用来计算在字符串中的有效Python表达式,并返回一个对象
tuple(s)	将序列 s 转换为一个元组

list(s)	将序列 s 转换为一个列表
set(s)	转换为可变集合
dict(d)	创建一个字典。d 必须是一个序列 (key,value)元组。
frozenset(s)	转换为不可变集合
chr(x)	将一个整数转换为一个字符
unichr(x)	将一个整数转换为Unicode字符
ord(x)	将一个字符转换为它的整数值
hex(x)	将一个整数转换为一个十六进制字符串
oct(x)	将一个整数转换为一个八进制字符串

```
#!/usr/bin/python

#coding=utf-8

# INT 函數能夠

# 把符合數學格式的數字型字符串轉換成整數

# 把浮點數轉換成整數，但是只是簡單的取整，而非四捨五入。

aa = int("124")    #Correct

print "aa = ", aa  #result=124


bb = int(123.45) #correct

print "bb = ", bb #result=123


cc = int("-123.45")  #Error,Can't Convert to int

print "cc = ",cc


# float 將整數和字符串轉換成浮點數


aa = float("124")     #Correct
```

```
print "aa = ", aa     #result = 124.0

bb = float("123.45")  #Correct

print "bb = ", bb     #result = 123.45

cc = float(-123.6)    #Correct

print "cc = ",cc      #result = -123.6


# str 函數將數字轉換成字符

aa = str(123.4)     #Correct

print aa            #result = '123.4'

bb = str(-124.a)    #SyntaxError: invalid syntax

print bb

cc = str("-123.45") #correct

print cc            #result = '-123.45'
```

➢ print():列印輸出內容

print(項目 1, 項目 2, sep=分隔字元, end=結束字元)

```
print(str1 + " " + str2)
print(str1 ,str2,sep="",end="\n")
print(str1 ,str2,sep="&",end="")
```

參數格式化:%

```
print("the length of (%s) is %d" %("Hello World",len("Hello World")))
print("PI = %f" % math.pi)
print("PI = %10.3f" % math.pi)
print("PI = %6d" % int(math.pi))
print("%4s %4d %4d" % ("王大明",10,100))
```

參數格式化:format() (Python3 後新增)

```
name="林小明"
score=80
print("{}的成績為{}".format(name,score))
```

字串插值（Formatted String Literal）: f (Python 3.6 後新增)

```
text = 'world'
print(f'Hello, {text}')
```

範例 basic-03.py

➢ String（字串）

String（字串）是 Python 中使用最頻繁的數據類型。字串可以完成大多數集合類的數據結構實現。它支持字符，數字，字符串甚至可以包含字串。字串用[ ]標識。是 python 最通用的複合數據類型。看這段代碼就明白。字串中的值得分割也可以用到變數[頭下標:尾下標]，就可以截取相應的字串，從左到右索引默認 0 開始的，從右到左索引默認-1 開始，下標可以為空表示取到頭或尾。加號（+）是字串連接運算符，星號（*）是重複操作。如下實例：

```
#!/usr/bin/python

var1 = 'Hello World!'
var2 = "Python w3big"

print "var1[0]: ", var1[0]
print "var2[1:5]: ", var2[1:5]
```

```
#!/usr/bin/python

#coding=utf-8

str = 'Hello World!'

print  str  #輸出完整字符串

print  str[0] #輸出字符串中的第一個字符

print  str[2:5]  #輸出字符串中第三個至第五個之間的字符串

print  str[2:]  #輸出從第三個字符開始的字符串

print  str * 2  #輸出字符串兩次

print  str  +  "TEST"  #輸出連接的字符串
```

範例 basic-04.py

➢ List (串列)

串列是 Python 中最基本的數據結構。序列中的每個元素都分配一個數字-它的位置或索引，第一個索引是 0，第二個索引是 1，依此類。Python 中有 6 個序列的內置類型，但最常見的是列表和元組。序列都可以進行的操作包括索引，切片，加，乘，檢查成員。此外，python 已經內置確定序列的長度以及確定最大和最小的元素的方法。列表是最常用的 Python 的數據類型，它可以作為一個方括號內的逗號分隔值出現。列表的數據項不需要具有相同的類型。創建一個列表，只要把逗號分隔的不同的數據項使用方括號括起來即可如下所示：

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"];
```

與字符串的索引一樣，列表索引從 0 開始。列表可以進行截取，組合等。

建立與顯示串列：

```
#!/usr/bin/python

#coding=utf-8


list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5, 6, 7 ];


print "list1[0]: ", list1[0]

print "list2[1:5]: ", list2[1:5]
```

更新串列：
你可以對列表的數據項進行修改或更新，你也可以使用追加 append（）方法來添加列表項，如下所示：

```
#!/usr/bin/python

#coding=utf-8


list2 = [1, 2, 3, 4, 5, 6, 7 ];

list2[2]=20;

print list[2];
```

注意：我們會在接下來的章節討論 append()方法的使用

刪除元素：
可以使用 del 語句來刪除清單的的元素，如下實例：

```
#!/usr/bin/python

#coding=utf-8
```

```
list2 = [1, 2, 3, 4, 5, 6, 7 ];
del list2[2];
print list2;
```

Python 列表腳本操作符：
清單對 + 和 * 的操作符與字串相似。+ 號用於組合清單，* 號用於重複列表。如下所示：

Python 表达式	结果	描述
len([1, 2, 3])	3	长度
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	组合
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	重复
3 in [1, 2, 3]	True	元素是否存在于列表中
for x in [1, 2, 3]: print x,	1 2 3	迭代

Python 列表截取：

```
>>> L = ['Google', 'Runoob', 'Taobao']
```

描述：

Python 表达式	结果	描述
L[2]	'Taobao'	读取列表中第三个元素
L[-2]	'Runoob'	读取列表中倒数第二个元素
L[1:]	['Runoob', 'Taobao']	从第二个元素开始截取列表

Python 列表函数&方法：

Python包含以下函数:

序号	函数
1	cmp(list1, list2) 比较两个列表的元素
2	len(list) 列表元素个数
3	max(list) 返回列表元素最大值
4	min(list) 返回列表元素最小值
5	list(seq) 将元组转换为列表

Python包含以下方法:

序号	方法
1	list.append(obj) 在列表末尾添加新的对象
2	list.count(obj) 统计某个元素在列表中出现的次数
3	list.extend(seq) 在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）
4	list.index(obj) 从列表中找出某个值第一个匹配项的索引位置
5	list.insert(index, obj) 将对象插入列表
6	list.pop(obj=list[-1]) 移除列表中的一个元素（默认最后一个元素），并且返回该元素的值
7	list.remove(obj) 移除列表中某个值的第一个匹配项
8	list.reverse() 反向列表中元素
9	list.sort([func]) 对原列表进行排序

範例 basic-05.py

➢ Tuple（元組）

元組是另一個數據類型，類似於 List（串列）。元組用"()"標識。內部元素用逗號隔開。但是元素不能二次賦值，也就是說不能刪除或修改元素，但可以刪除此變數。

Tuple create very simple, only need to add the elements in parentheses and separated by commas can be. The following examples:

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

以下是元组无效的，因为元组是不允许更新的。而列表是允许更新的：

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tuple[2] = 1000 # 元组中是非法应用
list[2] = 1000 # 列表中是合法应用
```

```
#!/usr/bin/python

#coding=utf-8


tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )

tinytuple = (123, 'john')


print tuple # 輸出完整元組

print tuple[0] # 輸出元組的第一個元素

print tuple[1:3] # 輸出第二個至第三個的元素

print tuple[2:] # 輸出從第三個開始至列表末尾的所有元素
```

```
print tinytuple * 2 # 輸出元組兩次

print tuple + tinytuple # 打印組合的元組


#delete tuple

del tup

print tup
```

範例 basic-06.py

➢ Dictionary（字典）

字典(Dictionary)是除列表以外 python 之中最靈活的內置數據結構類型。串列(List)是有序的對象結合，字典(Dictionary)是無序的對象集合。兩者之間的區別在於：字典當中的元素是通過鍵來存取的，而不是通過偏移存取。字典用"{ }"標識。字典由索引(key)和它對應的值 value 組成。
(簡單說，就是索引值可以是字串或數字)

Each dictionary key (key => value) of the colon**(:) divided between each pair with a comma (,)division, including the entire dictionary in curly braces({}),the format is as follows:**

```
d = {key1 : value1, key2 : value2 }
```

Key must be unique, but the value is not necessary.

```
#!/usr/bin/python

#coding=utf-8


dict = {}

dict['one'] = "This is one"

dict[2] = "This is two"


tinydict = {'name': 'john','code':6734, 'dept': 'sales'}
```

```
print dict['one'] # 輸出鍵為'one' 的值
print dict[2] # 輸出鍵為 2 的值
print tinydict # 輸出完整的字典
print tinydict.keys() # 輸出所有鍵
print tinydict.values() # 輸出所有值

#修改字典
tinydict[‘name’]=”marry”
print tinydict[‘name’] # 輸出所有值

#刪除字典元素
del tinydict[‘name’] # 刪除键是'Name'的条目
print tinydict[‘name’] # 輸出所有值
```

範例 basic-07.py

字典內置函數&方法：

Python字典包含了以下內置函数：

序号	函数及描述
1	cmp(dict1, dict2) 比较两个字典元素。
2	len(dict) 计算字典元素个数，即键的总数。
3	str(dict) 输出字典可打印的字符串表示。
4	type(variable) 返回输入的变量类型，如果变量是字典就返回字典类型。

Python字典包含了以下内置方法：

序号	函数及描述
1	dict.clear() 删除字典内所有元素
2	dict.copy() 返回一个字典的浅复制
3	dict.fromkeys(seq[, val])) 创建一个新字典，以序列 seq 中元素做字典的键，val 为字典所有键对应的初始值
4	dict.get(key, default=None) 返回指定键的值，如果值不在字典中返回default值
5	dict.has_key(key) 如果键在字典dict里返回true，否则返回false
6	dict.items() 以列表返回可遍历的(键, 值) 元组数组
7	dict.keys() 以列表返回一个字典所有的键
8	dict.setdefault(key, default=None) 和get()类似, 但如果键不存在于字典中，将会添加键并将值设为default
9	dict.update(dict2) 把字典dict2的键/值对更新到dict里
10	dict.values() 以列表返回字典中的所有值
11	pop(key[,default]) 删除字典给定键 key 所对应的值，返回值为被删除的值。key值必须给出。 否则，返回default值。
12	popitem() 随机返回并删除字典中的一对键和值。

常用 Methods

4	dict.get(key, default=None) For *key* key, returns value or default if key not in dictionary

EX:

list_test = {'a': 1, 'b': 2}

list_test.get('a') # 得到結果 1

```
list_test.get('c') #  得到結果 none
list_test.get('c', 3) #  得到設定的默認值 3
list_test['b'] #  得到結果  2
list_test['c'] #  返回一個鍵值錯誤類型
```

參考文獻:

https://www.tutorialspoint.com/python/python_dictionary.htm

1.2 運算符

Python算术运算符

以下假设变量a为10，变量b为20：

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 30
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除 - x除以y	b / a 输出结果 2
%	取模 - 返回除法的余数	b % a 输出结果 0
**	幂 - 返回x的y次幂	a**b 为10的20次方， 输出结果 100000000000000000000
//	取整除 - 返回商的整数部分	9//2 输出结果 4 , 9.0//2.0 输出结果 4.0

以下实例演示了Python所有算术运算符的操作：

範例 basic-08.py

➢ 比較運算符

以下假设变量a为10，变量b为20：

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 true.
<>	不等于 - 比较两个对象是否不相等	(a <> b) 返回 true。这个运算符类似 != 。
>	大于 - 返回x是否大于y	(a > b) 返回 False。
<	小于 - 返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的大写。	(a < b) 返回 true。
>=	大于等于 - 返回x是否大于等于y。	(a >= b) 返回 False。
<=	小于等于 - 返回x是否小于等于y。	(a <= b) 返回 true。

以下实例演示了Python所有比较运算符的操作：

➢ 賦值運算符

以下假设变量a为10，变量b为20：

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

以下实例演示了Python所有赋值运算符的操作：

➢ 位運算符

按位运算符是把数字看作二进制来进行计算的。Python中的按位运算法则如下：

运算符	描述	实例
&	按位与运算符	(a & b) 输出结果 12 ，二进制解释： 0000 1100
|	按位或运算符	(a | b) 输出结果 61 ，二进制解释： 0011 1101
^	按位异或运算符	(a ^ b) 输出结果 49 ，二进制解释： 0011 0001
~	按位取反运算符	(~a) 输出结果 -61 ，二进制解释： 1100 0011， 在一个有符号二进制数的补码形式。
<<	左移动运算符	a << 2 输出结果 240 ，二进制解释： 1111 0000
>>	右移动运算符	a >> 2 输出结果 15 ，二进制解释： 0000 1111

以下实例演示了Python所有位运算符的操作：

➢ 運算符優先級

以下表格列出了从最高到最低优先级的所有运算符：

运算符	描述
**	指数 (最高优先级)
~ + -	按位翻转, 一元加号和减号 (最后两个的方法名为 +@ 和 -@)
* / % //	乘，除，取模和取整除
+ -	加法减法
>> <<	右移，左移运算符
&	位 'AND'
^ |	位运算符
<= < > >=	比较运算符
<> == !=	等于运算符
= %= /= //= -= += *= **=	赋值运算符
is is not	身份运算符
in not in	成员运算符
not or and	逻辑运算符

以下实例演示了Python所有运算符优先级的操作：

1.3 條件語句&循環語句

Python 编程中 if 语句用于控制程序的执行，基本形式为：

```
if 判断条件：
    执行语句……
else：
    执行语句……
```

```
if name == 'bill':
    flag = True
    print 'Wlecome boss'
else:
    print name

if flag:
    print name, "is my family!"
else:
    print "Who are you?"
```

當判斷條件為多個值時,可以使用以下形式:

```
if 判断条件1:
    执行语句1……
elif 判断条件2:
    执行语句2……
elif 判断条件3:
    执行语句3……
else:
    执行语句4……
```

```
# elif 用法
if num == 3: # 判斷 num 的值
  print 'boss'
elif num == 2:
  print 'user'
elif num == 1:
  print 'worker'
elif num < 0: # 值小於零時輸出
  print 'error'
else:
  print 'roadman' # 條件均不成立時輸出
```

```
# 判斷值是否在 0~10 之間
if num >= 0 and num <= 10:
    print 'hello'
```

```
# 判斷值是否在 0~5 或者 10~15 之間
if (num >= 0 and num <= 5) or (num >= 10 and num <= 15):
    print 'hello'
else:
    print 'undefine'
```

你也可以在同一行的位置上使用 if 條件判斷語句，如下實例：

```
var = 100
if ( var  == 100 ) : print "變量 var 的值為 100"
print "Good bye!"
```

範例 basic-09.py
範例 basic-10.py
範例 basic-11.py

➢ if…in

```
# Initializing list
test_list = [ 1, 6, 3, 5, 3, 4 ]

print("Checking if 4 exists in list ( using loop ) : ")

# Checking if 4 exists in list
# using loop
for i in test_list:
    if(i == 4) :
        print ("Element Exists")

print("Checking if 4 exists in list ( using in ) : ")

# Checking if 4 exists in list
# using in
if (4 in test_list):
    print ("Element Exists")
```

範例 basic-11-2.py

與 Java、C\C++等語言不同，Python 中是不提供 switch/case 語法。可以使用字典實現 switch/case 這種方式易維護，同時也能夠減少代碼量。如下是使用字典模擬的 switch/case 實現：
範例 basic-12.py

參考文獻:
http://wyp8711.blogspot.com/2019/09/python-dict-caseswitch.html

➢ 循環語句 for

语法：

for循环的语法格式如下：

```
for iterating_var in sequence:
    statements(s)
```

```
#! /usr/bin/python

#coding=utf-8


for letter in 'python':

    print 'current letter:',letter




##############################################


fruits = ['banana','apple', 'mango']


for fruit in fruits:

    print 'current fruit:', fruit


#############################################




for num in range(1,5):

    print num


############################################


print "1~10,num=num+2"

for num in range(1,10,2):

    print num
```

```
##########################################

print "10~1,num=num-2"

for num in range(10,1,-2):

    print num
```

範例 basic-13.py

範例 basic-14.py

➢ break 與 continue 命令

```
print("........1.........")
for i in range(1,11):
    if(i==6):
        break
    print(i)

print("........2.........")
for i in range(1,11):
    if(i==6):
        continue
    print(i)
```

範例 basic-15.py

➢ 循環使用 for else

```
for 變數 in 串列:
    程式區塊一
    if(條件式):
        程式區塊二
        break
else:
    程式區塊三
```

範例 basic-16.py

若 for 迴圈正常執行完每一次程式區塊，就會執行 else 的程式區塊三；若迴圈中任何一次條件成立就以 break 命令離開迴圈，將不會執行 else 的程式區塊三。

➢ While 循環語句

Python 編程中 while 語句用於循環執行程序，即在某條件下，循環執行某段程序，以處理需要重複處理的相同任務。其基本形式為：

```
while 判断条件：
    执行语句……
```

```
count = 0
while (count < 9):
   print 'The count is:', count
   count = count + 1
```

範例 basic-17.py

➢ 無限循環 while

```
while(True):  # 該條件永遠為 true，迴圈將無限執行下去
   num = raw_input("Enter a number  :")
   print "You entered: ", num

print "Good bye!"
```

範例 basic-18.py

➢ 循環使用 while else

```
# 判斷質數
import math
num = int(input('請輸入一個整數？'))
j = 2
while j < math.sqrt(num):
```

```
    if (num%j == 0):
        print(num, '不是質數')
        break
    j += 1
else:
    print(num, '是質數')
```

1.4 函數

將一堆程式包起來，形成一支函式，便於重複使用。

```
# 定義
def multiply(x,y):
    return x*y


def operation(x,y):
    multiply=x*y
    add=x+y
    minus=x-y
    divide=float(x)/float(y)
    return [multiply,add,minus,divide]



num1=multiply(2,3)
print ("multiply:"+str(num1))


num=opertion(2,3)
print ("operation:"+str(num[0]))
print ("operation:"+str(num[1]))
print ("operation:"+str(num[2]))
print ("operation:"+str(num[3]))
```

範例 basic-19.py

1.5 模組(module)

一個較大型專案，程式是由許多類別或函式組成，為了程式的分工和維護，可以適度地將程式分割成許多模組，再匯入並呼叫這些模組。

建立 math_operation.py

範例: math_operation.py

使用方法 1(import 模組名稱)：

每次要加上 hello_module，不方便。

```
import math_operation

num=math_operation.operation(4,6)
```

範例 basic-20.py

使用方法 2(匯入模組內所有函式):

匯入 math_operation 模組內全部的東西後，便能直接呼叫函式 hello，但有可能現有程式碼同名之函式，會造成衝突。

```
from math_operation import *

num=operation(4,6)
```

範例 basic-21.py

使用方法 3(匯入模組內函式):

選擇性地只匯入需要的東西

```
from math_operation import operation

num=operation(10,33)
```

範例 basic-22.py

使用方法 4(使用 as 指定模組別名):

可用 import/as 語法，將匯入的命名空間取個簡短的名稱

```
import math_operation as m

num=m.operation(4,6)
```

範例 basic-23.py

參考文獻:
http://wiki.alarmchang.com/index.php?title=Python_import_%E6%8C%87%E5%AE%9A%E7%9B%AE%E9%8C%84%E8%A3%A1%E9%9D%A2%E7%9A%84_.py_%E6%AA%94%E6%A1%88

1.6 補充

➢ 區域變數 vs 全域變數

```
#!/usr/bin/python
#coding=utf-8

def scope():
    var1 = 1 #區域變數
    print(var1,var2)
var1 = 10 #全域變數
var2 = 20 #全域變數
scope()
print(var1,var2)
```

```
#!/usr/bin/python
#coding=utf-8

def scope():
    global var1 #在函式內使用全域變數，以global宣告
    var1 = 1
    var2 = 2 #區域變數
    print(var1,var2)
var1 = 10 #全域變數
var2 = 20 #全域變數
scope()
print(var1,var2)
```

範例 basic-24.py

範例 basic-25.py

➢ type()函式來顯示資料型態

type() 函數如果你只有第一個參數則返回對象的類型

```
pi@raspberrypi:~/python_code $ python
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> print(type(1))
<type 'int'>
>>> print(type('runbbo'))
<type 'str'>
>>> print(type([2,3]))
<type 'list'>
>>> print(type({0:"zero",1:"one"}))
<type 'dict'>
```

➢ *args 的用法 (一個星號)

打星號可以引入不定數量的參數：

一個星號會以 tuple 的方式引入

```
def buy(*price):

    print price

buy(1,2,3,4)


data=("bill",1,"tt")

buy(*data)
```

範例 basic-26.py

➢ **kwargs 的用法(二個星號)

打星號可以引入不定數量的參數：
兩個星號會以 dict 的方式引入

```
def sell(**price):

    print price


sell(apple=10, ball=15, cat=25)


data = {"arg3": 3, "arg2": "two"}

sell(**data)
```

範例 basic-27.py

➢ dir()的用法

dir([obj]) 用於查詢指定物件 obj 的全部屬性與方法，當不帶參數時，列出目前記憶體中的所有物件。

```
print(dir())
```

```
>>> dir()
['__builtins__', '__doc__', '__name__', '__package__']
```

➢ python 研究-with as 用法

有一些任務，可能事先需要設置，事後做清理工作。對於這種場景，Python 的 with 語句提供了一種非常方便的處理方式。一個很好的例子是文件處理，你需要獲取一個文件句柄，從文件中讀取資料，然後關閉文件句柄。

如果不用 with 語句，程式碼如下：

```
file = open("/tmp/foo.txt")
data = file.read()
file.close()
```

這裡有兩個問題：
一是可能忘記關閉文件句柄；
二是文件讀取資料發生異常，沒有進行任何處理。

下面是處理異常的加強版本：

```
try:
    f = open('xxx')
except:
    print 'fail to open'
    exit(-1)
try:
    do something
except:
    do something
finally:
    f.close()
```

雖然這段程式碼執行良好，但是太冗長了。

這時候就是 with 一展身手的時候了。除了有更優雅的語法，with 還可以很好的處理上下文環境產生的異常。

下面是 with 版本的程式碼：

```
with open("/tmp/foo.txt") as file:
```

```
    data = file.read()
```

with 如何工作?
緊跟 with 後面的語句被求值後，返回物件的 __enter__() 方法被呼叫，這個方法的返回值將被賦值給 as 後面的變數。
當 with 後面的程式碼塊全部被執行完之後，將呼叫前面返回物件的 __exit__() 方法。

文獻:
https://icodding.blogspot.com/2016/05/python-with-as.html

➢ help()的用法

help(obj) 函數：用於查詢物件 obj 的詳細操作說明

```
help(__builtins__)
```

```
Help on built-in module __builtin__:

NAME
    __builtin__ - Built-in functions, exceptions, and other objects.

FILE
    (built-in)

DESCRIPTION
    Noteworthy: None is the `nil' object; Ellipsis represents `...' in sl
ices.

CLASSES
    object
        basestring
            str
            unicode
        buffer
        bytearray
```

參考文獻:
https://note.pcwu.net/2017/03/03/python-arbitrary/

➢ Decorator(裝飾器)

Decorator(裝飾器)被大量廣泛的使用在各方 library，是非常實用和必須了解的基礎。Decorator(裝飾器)是一個函式，用來包著其他的函式。裝飾其他的函式，讓被裝飾或是說被包著的函式能力增強。使用時機，當一個函數中，不同邏輯混雜在一起的時候，程序的可讀性會大打折扣。這個時候，可以考慮用一種叫做“裝飾器”的東西來重新整理代碼。

裝飾詞的優點：

1、降低程式碼重複率
2、易讀性高
3、靈活度高

```python

def is_prime(num):
    if num < 2:
        return False
    elif num == 2:
        return True
    else:
        for i in range(2, num):
            if num % i == 0:
                return False
        return True

def prime_nums():
    for i in range(2,100):
        if is_prime(i):
            print(i)

prime_nums()
```

範例 basic-28_1.py

此程式的缺點，prime_nums()包含時間計算與判斷質數。

```python
import time

def is_prime(num):
    if num < 2:
        return False
    elif num == 2:
        return True
    else:
        for i in range(2, num):
            if num % i == 0:
                return False
        return True

def prime_nums():
    t1 = time.time()
    for i in range(2,100):
        if is_prime(i):
            print(i)
    t2 = time.time()
    print(t2-t1)

prime_nums()
```

範例 basic-28_2.py

```python
#!/usr/bin/python
#coding=utf-8

import time

def display_time(func):
    def wrapper():
        t1 = time.time()
        func()
        t2 = time.time()
        print(t2-t1)
    return wrapper

def is_prime(num):
    if num < 2:
        return False
    elif num == 2:
        return True
    else:
        for i in range(2, num):
            if num % i == 0:
                return False
        return True

def prime_nums():
    for i in range(2,100):
        if is_prime(i):
            print(i)

f = display_time(prime_nums) #return f which is a function
f()
```

範例 basic-28_3.py

```
C: > Users > tony > DOCUME~1 > MobaXterm > slash > RemoteFiles >
1  #!/usr/bin/python
2  #coding=utf-8
3
4  import time
5
6  def display_time(func):
7      def wrapper():
8          t1 = time.time()
9          func()
10         t2 = time.time()
11         print(t2-t1)
12     return wrapper
13
14 def is_prime(num):
15     if num < 2:
16         return False
17     elif num == 2:
18         return True
19     else:
20         for i in range(2, num):
21             if num % i == 0:
22                 return False
23         return True
24
25 @display_time
26 def prime_nums():
27     for i in range(2,100):
28         if is_prime(i):
29             print(i)
30
31 prime_nums()
```

將時間計算與判斷質數分開，可以使用裝飾詞(Decorator)。裝飾詞就是一個函數。首先執行 31 行 prime_nums()，接著執行 25 行裝飾詞，因此會執行第 6 行 display_time()。接著執行 wrapper()，先計算時間，在 call func()，執行第 26 行 prime_nums()，最後執行第 10~12 行。

範例 basic-28_4.py

```
C: > Users > tony > DOCUME~1 > MobaXterm > slash
import time

def display_time(func):
    def wrapper(*args):
        t1 = time.time()
        result = func(*args)
        t2 = time.time()
        print(t2-t1)
        return result
    return wrapper

def is_prime(num):
    if num < 2:
        return False
    elif num == 2:
        return True
    else:
        for i in range(2, num):
            if num % i == 0:
                return False
        return True
```

```
@display_time
def count_prime_nums(maxnum):
    count = 0
    for i in range(2,maxnum):
        if is_prime(i):
            count = count + 1
    return count

count = count_prime_nums(100)
print(count)
```

範例 basic-28_5.py

參考文獻:

https://www.maxlist.xyz/2019/12/07/python-decorator/
https://www.youtube.com/watch?v=QqRvteWBSWg

1.6 物件導向(Object Oriented Programming)程式開發

Python 的類別機制是 C++ 以及 Modula-3 的綜合體，當我們在使用 Python 的類別時並不用去宣告該類別的型態，也不用去宣告這個類別是否為 public 或 private，Python 所有的類別與其包含的成員都是 public 的，對於類別（Class）來說最重要的一些特性在 Python 程式語言裡面都有完全的保留，如最重要的單一繼承與多重繼承，一個衍生／子類別（Derived class）可以覆載（Override）其所有基礎類別（base class）的任何方法（method），方法（method）也可以呼叫一個基礎類別（base class）的同名方法。

➢ 定義類別(Create Object)

在 Python 程式語言裡要定義一個類別必須使用 class 敘述句，然後在敘述句後面接著類別的名稱，並不用去定義是否為 public 或 private 等，也不用去定義資料型態，如下定義：

```
#! /usr/bin/python
#coding=utf-8
class MyClass:
    text="ABC"

    def clear(self):
        self.text=""

```

範例: class-01.py

```
#! /usr/bin/python
#coding=utf-8
class ShadowingTest:
    x=10
    y=50
    def printInfo(self,x):
        print("區域變數"+str(x))
        print("屬性"+str(self.x))
        print("屬性"+str(self.y))
        return ("x="+str(x+50))
```

範例: class-02.py

➢ The __init__() Function

類別內的方法（method）宣告方式跟函數一樣，您可以想像類別方法就像是將函數放進類別裡面，差異性在於函數的引數部份必須要加入 self 敘述句，self 敘述句就像其他程式語言裡面的 this。

Python 程式語言裡的類別有一個特別方法（method）稱為__init__()，當建立一個實例（Instance）的同時類別就會去執行這個函數，其實這個就像其他程

式語言的類別所定義的建構函數（Constructors）。

```
#! /usr/bin/python
#coding=utf-8

class TaipeiBank:
    balance=0
    def __init__(self, balance):
        self.balance=balance
    def printBalance(self):
        print("存款餘額:" + str(self.balance))

```

範例: class-03.py

```
#! /usr/bin/python
#coding=utf-8
class SmallMath:
    x=0
    y=0
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def add(self):
        print("加法結果:"+str(self.x+self.y))
    def mul(self):
        print("乘法結果:"+str(self.x*self.y))

```

範例: class-04.py

➢ 封裝(Encapsulation)

類別內的成員變數，可以讓外部引用的稱公有(public)屬性，而可以讓外部引用的方法稱公有方法。但任何類別的屬性與方法可以供外部隨意存取，這個設計概念最大的風險是有資訊安全的疑慮。

1、private 有安全性

2、外部無法存取 private variable

3、private variable 僅供類別內部使用

在屬性和方法前面加上「__」兩個_字元，就成為 private 屬性和方法。

補充：python 變量初始化為空值分別是

- 數值: digital_value = 0
- 字串: str_value = ""

- 串列: list_value = []
- 字典: ditc_value = {}
- 元組: tuple_value = ()

```
#! /usr/bin/python
#coding=utf-8

class Student():
    Sno=""
    Sname=""
    Age = 18
    __Money = 100

    def Iam(self):
        print("I am " + self.Sno + ":" +self.Sname + " 3Q!")
        self.__Money = 120
        print("I hava " + str(self.__Money)+" dollors")
```

範例: class-05.py

```
#! /usr/bin/python
#coding=utf-8

class Student():
    Sno="" #public
    Sname="" #public
    Age = 18
    __Money = 100 #private

    def Iam(self):
        print("I am " + self.Sno + ":" +self.Sname + " 3Q!")
        print("I hava " + str(self.__Money)+" dollors")

    def setMoney(self,money):
        self.__Money = money

    def getMoney(self):
        return(self.__Money)
```

範例: class-06.py

➢ 繼承(Inheritance)

可以繼承原有的類別，修改延伸出新的類別，原有的類別被稱為基礎類別(base class)或雙親類別(parent class)，新的類別被稱為衍生類別(derived class)或子類別(child class)，這個衍生類別就自動擁有基礎類別的變數與函式。

使用「class 衍生類別(基礎類別)」來定義類別間的繼承關係，衍生類別就繼承了基礎類別；在衍生類別中使用「super().基礎類別的函式」可以呼叫基礎類別的函

式來幫忙，若衍生類別所需要的功能已經在基礎類別定義過了，就可以呼叫基礎類別幫忙，重複利用已經撰寫過的程式碼。

若繼承父類別，即擁有父類別的所屬性與方法，也可以修改原來屬性與方法的內容。

設定格式如下：

```python
#! /usr/bin/python
#coding=utf-8

class Animal():
    def __init__(self, name):
        self.name = name
    def fly(self):
        print(self.name + " fly!")

class Bird(Animal):
    def __init__(self, name):
        self.name = "red " + name
    def sing(self):
        print(self.name + " sing!")

```

範例: class-07py

其中，與 java 不同，父類別的建構方法會被子類別覆寫。

➢ 覆寫函式(Override)

衍生類別可以繼承基礎類別的資料與函式，在衍生類別內重新改寫基礎類別的函式，讓衍生類別與基礎類別的相同函式有不同功能，這樣的改寫函式稱作「覆寫(Override)」。

```python
#! /usr/bin/python
#coding=utf-8

class Emp:
    Salary=0
    def set_salary(self,Salary):
        if (Salary>40000):
            self.Salary=40000
        else:
            self.Salary=Salary
    def ShowSal(self):
        print(str(self.Salary))

class Manager(Emp):
    Bonus=0
    def set_salary(self,Salary):
        if (Salary>60000):
            self.Salary=60000
        else:
            self.Salary=Salary

    def ShowSal(self):
        print(str(self.Salary+self.Bonus))
```

範例: class-08py

```python
#! /usr/bin/python
#coding=utf-8

__metaclass__ = type
class Father:
    x = 50
    def printInfo(self):
        print(".....1.....")

class Child(Father):
    x = 100
    def printInfo(self):
        super(Child, self).printInfo()
        print("父x="+str(Father.x))
        print("子x="+str(self.x))
```

範例: class-09py

補充:

In Python 2, a declaration __metaclass__ = type makes declarations that would otherwise create old-style classes create new-style classes instead.

```python
#! /usr/bin/python
#coding=utf-8
__metaclass__ = type
class Animal:
    def __init__(self, name):
        self.name = name

    def eat(self):
        print(self.name + "正在吃食物")

    def sleep(self):
        print(self.name + "正在睡覺")

class Dog(Animal):
    def __init__(self, name):
        super(Dog, self).__init__(name)
```

範例: class-10.py

補充(pass 語法)：

Python pass 是空語句，是為了保持程序結構的完整性。

pass 不做任何事情，一般用做佔位語句。

```python
#! /usr/bin/python
#coding=utf-8

class Animal(object):
    def __init__(self, name):
        self.name = name
    def sound(self):
        pass

class Dog(Animal):
    def __init__(self, name):
        super(Dog,self).__init__('小狗'+name)

    def sound(self):
        return '汪汪叫'
```

範例: class-11.py

➢ classmethod()

classmethod()是 Python 中的內置函數，它為給定函數返回類方法。

用法： classmethod(function)

參數：該函數接受函數名稱作為參數。

參考文獻:

https://vimsky.com/zh-tw/examples/usage/classmethod-in-python.html

1.7 套件(Package)

把兩個檔案包在一個新的目錄 sample_package 底下：

```
sample_package/
├── __init__.py
├── sample_module.py
└── sample_module_import.py
```

很重要的是新增那個 __init__.py 檔。它是空的沒關係，但一定要有，有點宣稱自己是一個 package 的味道。

這時候如果是進到 sample_package 裡面跑一樣的指令，那沒差。但既然都打包成 package 了，通常是在整個專案的其他地方需要用到的時候 import 它，這時候裡面的 import 就要稍微做因應。

```
├──── package-01.py
├──── mypackage
        └──── __init__.py
        └──── Hello.py
        └──── myClass.py
```

class-12.py | myClass.py ● | Hello.py

C: > Users > tony > DOCUME~1 > MobaXterm > slash > RemoteFiles >

```python
#coding=utf-8
__metaclass__ = type
class Rectangle():        #定義父類別
    def __init__(self, width,height):
        self.width = width     #定義共用屬性
        self.height = height   #定義共用屬性
    def area(self):            #定義共用方法
        return self.width * self.height

class Triangle(Rectangle): #定義子類別
    def area2(self):        #定義子類別的共用方法
        return (self.width * self.height)/2
```

class-12.py | myClass.py ● | Hello.py ×

C: > Users > tony > DOCUME~1 > MobaXterm > slash > RemoteFiles

```python
def sayHello():
    print("Hello")
```

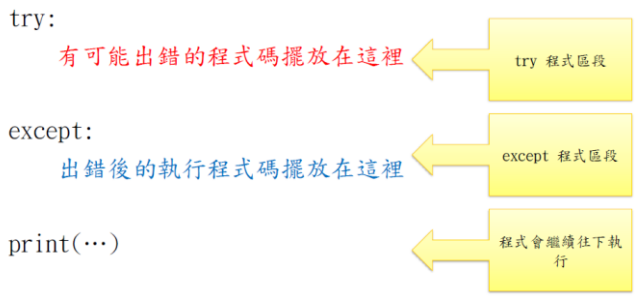
範例: package-01.py

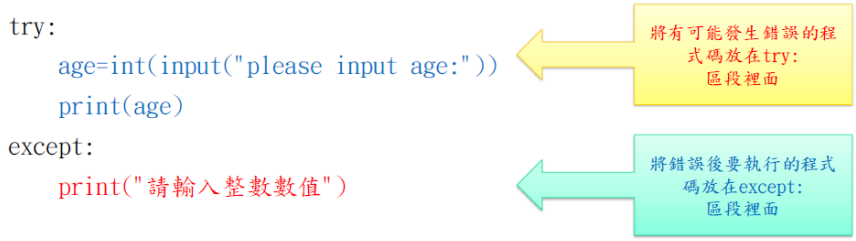
參考文獻：

https://medium.com/pyladies-taiwan/python-%E7%9A%84-import-%E9%99%B7%E9%98%B1-3538e74f57e3

1.8 例外處理(try except)

在執行程式的過程中產生錯誤，程式會中斷執行，發出例外訊息，以下介紹例外的程式區塊，與實作自訂的例外類別。





- try-except

使用程式區塊「try…except…」可以攔截例外，在 try 區塊中撰寫可能發生錯誤的程式，若發生錯誤，則會跳到 except 區塊執行進行後續的處理。

try: 　　pwd = raw_input('請輸入密碼') except: 　　print('發生錯誤')	輸入數字與字元則不會發生錯誤，密碼會儲存在變數 pwd，若輸入「ctrl+D」，則顯示「發生錯誤」。

範例: try-01.py

- try-except-else

使用程式區塊「try…except…else…finally」可以攔截例外，在 try 區塊中撰寫可能發生錯誤的程式，若發生錯誤，則會跳到 except 區塊進行後續的處理，若沒有發生錯誤，則會跳到 else 區塊執行。不論是否發生錯誤，都會執行 finally 的程式碼。

except 後可以接指定錯誤類型，常見錯誤類型，如下表。

錯誤類型	說明

KeyboardInterrupt	當使用者輸入中斷(Ctrl+C)時，發出此錯誤。
ZeroDivisionError	除以 0 時，發出此錯誤。
EOFError	接受到 EOF(end of file)訊息時，發出此錯誤。
NameError	區域變數或全域變數找不到時，發出此錯誤。
OSError	與作業系統有關的錯誤
FileNotFoundError	檔案或資料夾找不到時，發出此錯誤。
ValueError	輸入資料與程式預期輸入資料型別不同時，發出此錯誤。

```python
#! /usr/bin/python
#coding=utf-8

try:
    a=int(input("請輸入第一個整數："))
    b=int(input("請輸入第二個整數："))
    r = a % b
except ValueError:
    print("發生輸入非數值的錯誤!")
except Exception as e:
    print("發生" + str(e) +"的錯誤，包括分母為 0 的錯誤!")
else:
    print("ans=" + r)
finally:
    print("一定會執行的程式區塊")
```

範例: try-02.py

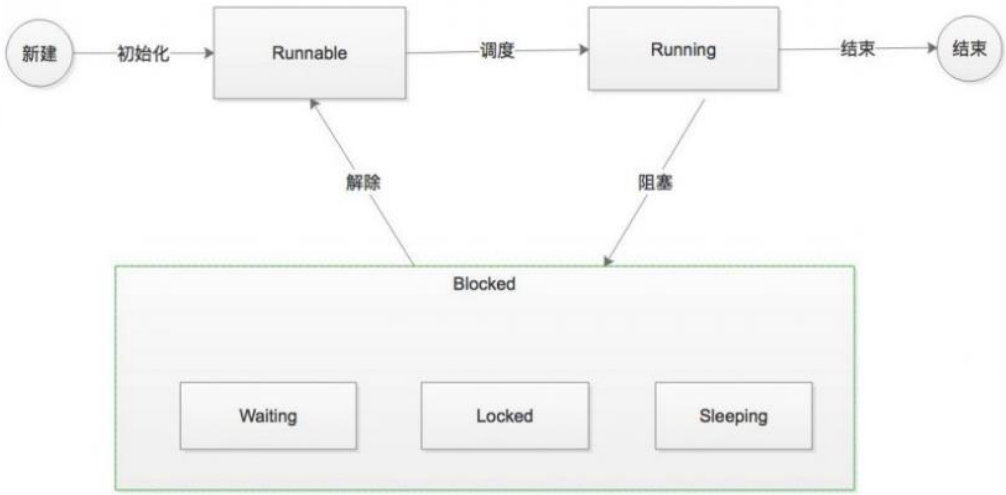
1.9 多執行緒(Multi-Threads)

作業系統原理相關的書，基本都會提到一句很經典的話：「程序是資源分配的最小單位，執行緒則是 CPU 排程的最小單位」。
執行緒是作業系統能夠進行運算排程的最小單位。它被包含在程序之中，是程序中的實際運作單位。一條執行緒指的是程序中一個單一順序的控制流，一個程序中可以併發多個執行緒，每條執行緒並行執行不同的任務。
好處 ：
1.易於排程。
2.提高併發性。通過執行緒可方便有效地實現併發性。程序可建立多個執行緒來執行同一程式的不同部分。
3.開銷少。建立執行緒比建立程序要快，所需開銷很少。
4.利於充分發揮多處理器的功能。通過建立多執行緒程序，每個執行緒在一個處理器上執行，從而實現應用程式的併發性，使每個處理器都得到充分執行。

多執行緒似於同時執行多個不同程序，多執行緒有如下優點：

- 使用線程可以把佔據長時間的程序中的任務放到後台去處理。
- 用戶界面可以更加吸引人，這樣比如用戶點擊了一個按鈕去觸發某些事件的處理，可以彈出一個進度條來顯示處理的進度
- 程序的運行速度可能加快
- 在一些等待的任務實現上如用戶輸入，文件讀寫和網絡收發數據等，線程就比較有用了。在這種情況下我們可以釋放一些珍貴的資源如內存佔用等等。

執行緒的狀態



在這裡我簡單來解釋以下這幾種狀態的含義:

1. New 新建:使用執行緒的第一步就是建立執行緒，建立後的執行緒只是進入可執行的狀態，也就是 Runnable。
2. Runnable:進入此狀態的執行緒還並未開始執行，一旦 CPU 分配時間片給這個執行緒後，該執行緒才正式的開始執行。
3. Running:執行緒正式開始執行，在執行過程中執行緒可能會進入阻塞的狀態，即 Blocked。
4. Blocked:在該狀態下，執行緒暫停執行，解除阻塞後，執行緒會進入 Runnable 狀態，等待 CPU 再次分配時間片給它。
5. Dead 結束:執行緒方法執行完畢或者因為異常終止返回。

這就和人的一生——出生、學習(工作之前的準備)、工作、休假——是相似的。其中最複雜的是執行緒從 Running 進入 Blocked 狀態,通常有三種情況:睡眠:執行緒主動呼叫 sleep 或 join 方法後。等待：執行緒中呼叫 wait 方法，此時需要有其他執行緒通過 notify 方法來喚醒。
同步:執行緒中獲取執行緒鎖，但是因為資源已經被其他執行緒佔用時。
到現在,我們對執行緒有個基本的概念,光說不練假把式,下面我們就通過是三個小的示例來聊聊執行緒的使用以及執行緒中最終的兩個概念:同步和通訊。

https://docs.python.org/2/library/threading.html?highlight=threading#module-threading

This class represents an activity that is run in a separate thread of control. There are two ways to specify the activity: by passing a callable object to the constructor, or by overriding the run() method in a subclass. No other methods (except for the constructor) should be overridden in a subclass. In other words, only override the __init__() and run() methods of this class.

➢ 建立子執行緒

```
#!/usr/bin/python

#coding=utf-8

import threading

import time
```

```
# 子執行緒的工作函數
def job():
  for i in range(5):
    print "Child thread:%d"% i
    time.sleep(1)

# 建立一個子執行緒
t = threading.Thread(target = job)

# 執行該子執行緒
t.start()

# 主執行緒繼續執行自己的工作
for i in range(3):
  print "Main thread:%d"% i
  time.sleep(1)

# 等待 t 這個子執行緒結束
t.join()

print("Done.")
```

範例: thread-01.py

➢ 多個子執行緒與參數

```
#!/usr/bin/python
#coding=utf-8

import threading
import time

# 子執行緒的工作函數
def job(num):
    for i in range(10):
        print("Thread", num)
        time.sleep(1)

# 建立 5 個子執行緒
threads = []
for i in range(5):
  threads.append(threading.Thread(target = job, args = (i,)))
  threads[i].start()

# 主執行緒繼續執行自己的工作
# ...

# 等待所有子執行緒結束
for i in range(5):
  threads[i].join()
```

```
print("Done.")
```

範例: thread-02.py

➢ Multi threads by overriding the run() method in a subclass

This class represents an activity that is run in a separate thread of control. There are two ways to specify the activity: by passing a callable object to the constructor, or by overriding the run() method in a subclass. No other methods (except for the constructor) should be overridden in a subclass. In other words, only override the __init__() and run() methods of this class.

```
#!/usr/bin/python
#coding=utf-8

import threading
import time

# 繼承 threading.Thread 的子執行緒類別
class MyThread(threading.Thread):
  #建構子
  def __init__(self, num):
    threading.Thread.__init__(self)
    self.num = num
  #重新改寫基礎類別的函式 run
  def run(self):
    print("Thread", self.num)
    time.sleep(1)
```

```
# 建立 5 個子執行緒
threads = []
for i in range(5):
  threads.append(MyThread(i))
  threads[i].start()

# 主執行緒繼續執行自己的工作
# ...

# 等待所有子執行緒結束
for i in range(5):
  threads[i].join()

print("Done.")
```

範例: thread-03.py

1.10 Python 處理 JSON

JSON(JavaScript Object Notation) 是一種輕量級的數據交換格式。它使得人們很容易的進行閱讀和編寫。同時也方便了機器進行解析和生成。它是基於 JavaScript Programming Language , Standard ECMA-262 3rd Edition - December 1999 的一個子集。 JSON 採用完全獨立於程序語言的格式，但是也使用了類似 C 語言的習慣（包括 C, C++, C#, Java, JavaScript, Perl, Python 等）。這些特性使 JSON 成為理想的數據交換語言。

JSON 是個以純文字為基底去儲存和傳送簡單結構資料，你可以透過特定的格式去儲存任何資料(字串,數字,陣列,物件)，也可以透過物件或陣列來傳送較複雜的資料。一旦建立了您的 JSON 資料，就可以非常簡單的跟其他程式溝通或交換資料，因為 JSON 就只是純文字個格式。
JSON 的優點如下:
1、相容性高
2、格式容易瞭解，閱讀及修改方便
3、支援許多資料格式 (number,string,booleans,nulls,array,associative array)
4、許多程式都支援函式庫讀取或修改 JSON 資料

➢ 如何建立 JSON 字串

可以透過底下規則來建立 JSON 字串：
1、JSON 字串可以包含陣列 Array 資料或者是物件 Object 資料
2、陣列可以用[ ]來寫入資料
3、物件可以用{ }來寫入資料
4、name / value 是成對的，中間透過 (:) 來區隔

物件或陣列的 value 值可以如下：
1、數字 (整數或浮點數)
2、字串 (請用””括號)
3、布林函數 (boolean) (true 或 false)
4、陣列 (請用 [ ])
5、物件 (請用 { })
6、NULL

➢ JSON 在 python 中的使用

- 介紹

JSON(JavaScript Object Notation) 是一種輕量級的數據交換格式。它使得人們很容易的進行閱讀和編寫。同時也方便了機器進行解析和生成。它是基於 JavaScript Programming Language , Standard ECMA-262 3rd Edition - December 1999 的一個子集。 JSON 採用完全獨立於程序語言的格式，但是也使用了類似 C 語言的習慣（包括 C, C++, C#, Java, JavaScript, Perl, Python 等）。這些特性使 JSON 成為理想的數據交換語言。

JSON 是個以純文字為基底去儲存和傳送簡單結構資料，你可以透過特定的格式去儲存任何資料(字串,數字,陣列,物件)，也可以透過物件或陣列來傳送較複雜的資料。一旦建立了您的 JSON 資料，就可以非常簡單的跟其他程式溝通或交換資料，因為 JSON 就只是純文字個格式。

JSON 的優點如下:

1、相容性高

2、格式容易瞭解，閱讀及修改方便

3、支援許多資料格式 (number,string,booleans,nulls,array,associative array)

4、許多程式都支援函式庫讀取或修改 JSON 資料

可以透過底下規則來建立 JSON 字串：

1、JSON 字串可以包含陣列 Array 資料或者是物件 Object 資料

2、陣列可以用[ ]來寫入資料

3、物件可以用{ }來寫入資料

4、name / value 是成對的，中間透過 (:) 來區隔

物件或陣列的 value 值可以如下:

1、數字 (整數或浮點數)

2、字串 (請用””括號)

3、布林函數 (boolean) (true 或 false)

4、陣列 (請用 [ ])

5、物件 (請用 { })

6、NULL

- JSON 在 python 中的使用

Python 2.6 開始加入了 JSON 模塊，無需另外下載，Python 的 JSON 模塊序列化與反序列化的過程分別是 encoding 和 decoding。

encoding：把一個 python 對象編碼轉換成 JSON 字符串
decoding：把 json 格式字符串解碼轉換成 Python 對象

JSON 函数

使用 JSON 函数需要导入 json 库：**import json**。

函数	描述
json.dumps	将 Python 对象编码成 JSON 字符串
json.loads	将已编码的 JSON 字符串解码为 Python 对象

通過輸出的結果可以看出，簡單類型通過 encode 之後跟其原始的 repr()輸出結果非常相似，但是有些數據類型進行了改變，例如上例中的元組則轉換為了列表。在 json 的編碼過程中，會存在從 python 原始類型向 json 類型的轉化過程，具體的轉化對照如下：

Python	JSON
dict	object
list, tuple	array
str, unicode	string
int, long, float	number
True	true
False	false
None	null

loads 方法返回了原始的對象，但是仍然發生了一些數據類型的轉化。比如，上例中'abc'轉化為了 unicode 類型。從 json 到 python 的類型轉化對照如下：

JSON	Python
object	dict
array	list
string	unicode
number (int)	int, long
number (real)	float
true	True
false	False
null	None

```
[
  {
    "name" : "Molecule Man",
    "age" : 29,
    "secretIdentity" : "Dan Jukes",
    "powers" : [
      "Radiation resistance",
      "Turning tiny",
      "Radiation blast"
    ]
  },
  {
    "name" : "Madame Uppercut",
    "age" : 39,
    "secretIdentity" : "Jane Wilson",
    "powers" : [
      "Million tonne punch",
      "Damage resistance",
      "Superhuman reflexes"
    ]
  }
]
```

encoding & decoding

```
# josn encoding

import json

#list 內放 dict

data = [{"ID":"1","STATUS":"0"},{"ID":"2","STATUS":"1"}]

data_string = json.dumps(data) #encoding


# json decoding

decoded = json.loads(data_string) # decoding


id1 = decoded[0]["ID"]

status1 = decoded[0]["STATUS"]


id2 = decoded[1]["ID"]

status2 = decoded[1]["STATUS"]
```

範例: json-01.py
範例: json-02.py
範例: json-03.py
範例: json-04.py

參考文獻:

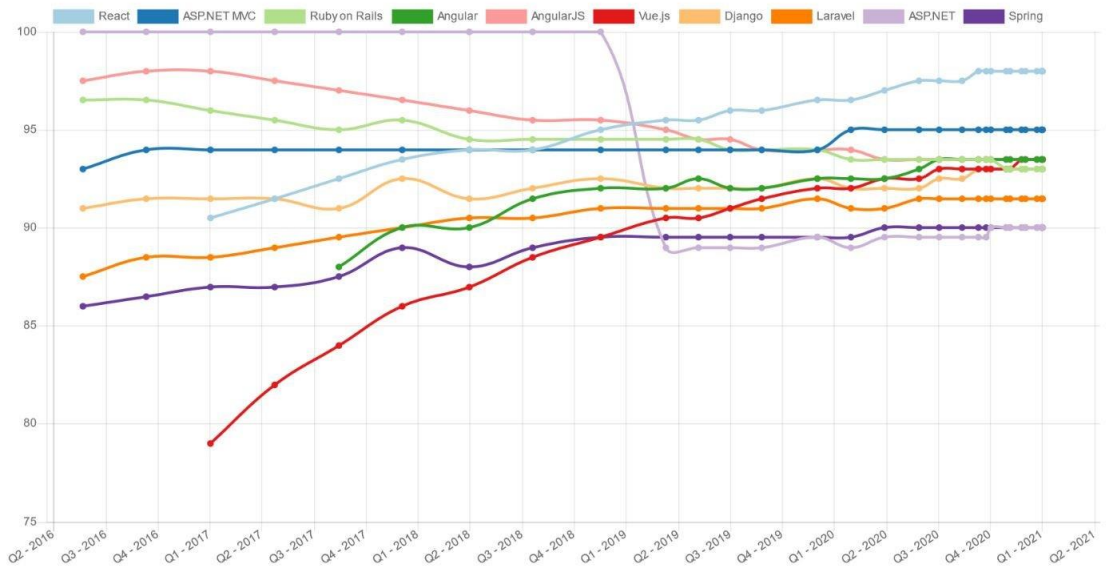
https://www.cnblogs.com/huchong/p/9037142.html
http://opendata2.epa.gov.tw/AQI.json

第三章：Python Web Djangle

Django 是一個高級的 Python 網路框架，可以快速開發安全和可維護的網站。由經驗豐富的開發者構建，Django 負責處理網站開發中麻煩的部分，因此你可以專注於編寫應用程序，而無需重新開發。它是免費和開源的，有活躍繁榮的社區、豐富的文檔、以及很多免費和付費的解決方案。Django 可以使你的應用具有以下優點:

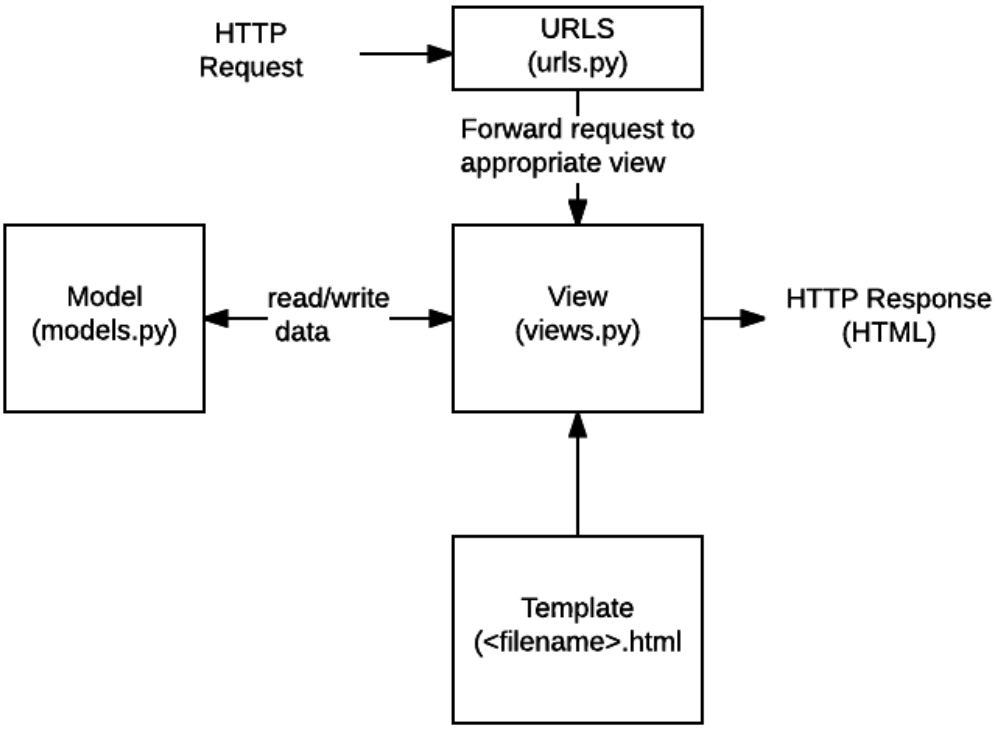
- **完備**：Django 遵循"功能完備"的理念，提供開發人員可能想要"開箱即用"的幾乎所有功能。
- **通用**：Django 可以（並已經）用於構建幾乎任何類型的網站—從內容管理系統和維基，到社交網絡和新聞網站。它可以與任何客戶端框架一起工作，並且可以提供幾乎任何格式（包括 HTML、RSS、JSON、XML 等）的內容。
- **安全**：Django 幫助開發人員，通過提供一個被設計為"做正確的事情"來自動保護網站的框架，來避免許多常見的安全錯誤。例如，Django 提供了一種安全的方式，來管理用戶帳號和密碼，避免了常見的錯誤，比如將 session 放在 cookie 中這種易受攻擊的做法（取而代之的是，cookies 只包含一個密鑰，實際數據存儲在數據庫中），或直接存儲密碼，而不是密碼的 hash 值。
- 密碼 hash ，是讓密碼通過加密 hash 函數，而創建的固定長度值。 Django 能通過運行 hash 函數，來檢查輸入的密碼 - 就是將輸出的 hash 值，與存儲的 hash 值進行比較是否正確。然而由於功能的"單向"性質，假使存儲的 hash 值受到威脅，攻擊者也難以解出原始密碼。
- **可擴展**：Django 使用基於組件的"無共享"架構 (架構的每一部分獨立於其他架構，因此可以根據需要進行替換或更改)。在不同部分之間，有明確的分隔，意味著它可以通過在任何級別添加硬件，來擴展服務：緩存服務器，數據庫服務器，或應用程序服務器。
- **可維護**：Django 代碼編寫，是遵照設計原則和模式，鼓勵創建可維護和可重複使用的代碼。特別是，它使用了不要重複自己（DRY）原則，所以沒有不必要的重複，減少了代碼的數量。 Django 還將相關功能，分組到可重用的"應用程序"中，並且在較低級別，將相關代碼分組或模塊（模型視圖控制器 Model View Controller (MVC)模式）。
- **可移植**：Django 是用 Python 編寫的，它在許多平台上運行。這意味著，你不受任務特定的服務器平台的限制，並且可以在許多種類的 Linux，Windows 和 Mac OS X 上運行應用程序。

Find your new favorite web framework：
https://hotframeworks.com/



在傳統的數據驅動網站中，Web 應用程序會等待來自 Web 瀏覽器（或其他客戶端）的 HTTP 請求。當接收到請求時，應用程序根據 URL 和可能的 `POST` 數據或 `GET` 數據中的信息確定需要的內容。根據需要，可以從數據庫讀取或寫入信息，或執行滿足請求所需的其他任務。然後，該應用程序將返回對 Web 瀏覽器的響應，通常通過將檢索到的數據插入 HTML 模板中的佔位符來動態創建用於瀏覽器顯示的 HTML 頁面。

Django 網絡應用程序通常將處理每個步驟的代碼分組到單獨的文件中：



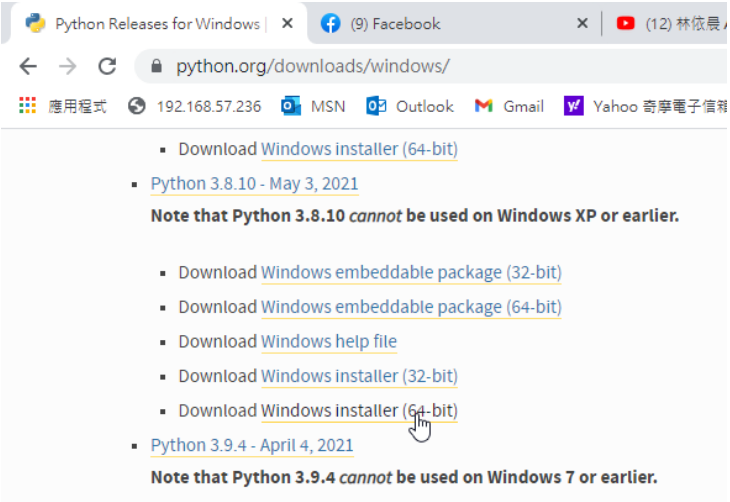
- **URLs:**雖然可以通過單個功能來處理來自每個 URL 的請求，但是編寫單獨的視圖函數來處理每個資源是更加可維護的。URL 映射器用於根據請求 URL 將 HTTP 請求重定向到相應的視圖。URL 映射器還可以匹配出現在 URL 中的字符串或數字的特定模式，並將其作為數據傳遞給視圖功能。
- **View:**視圖是一個請求處理函數，它接收 HTTP 請求並返回 HTTP 響應。視圖通過模型訪問滿足請求所需的數據，並將響應的格式委託給模板。
- **Models:**模型是定義應用程序數據結構的 Python 對象，並提供在數據庫中管理（添加，修改，刪除）和查詢記錄的機制。
- **Templates:**模板是定義文件（例如 HTML 頁面）的結構或佈局的文本文件，用於表示實際內容的佔位符。一個視圖可以使用 HTML 模板，從數據填充它動態地創建一個 HTML 頁面模型。可以使用模板來定義任何類型的文件的結構;它不一定是 HTML！

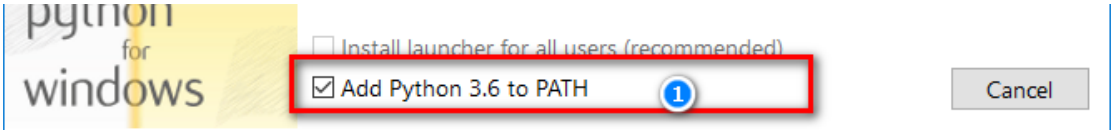
參考文獻:
https://developer.mozilla.org/zh-TW/docs/Learn/Server-side/Django/Introduction

➢ Django 網站框架 (Python)教學

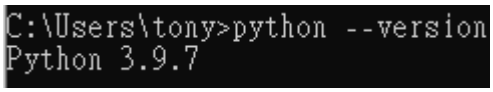
1. https://developer.mozilla.org/zh-TW/docs/Learn/Server-side/Django/Models
2. https://www.liujiangblog.com/course/django/2

3-1 Django 平台建置(windows)

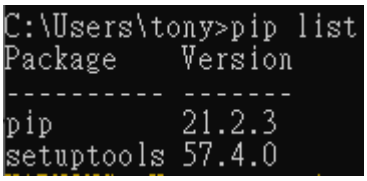





```
# python --version

# python -V
```




```
# pip list (查看已安裝的套件)
```



安裝虛擬環境

```
# pip install virtualenv
```

```
C:\Users\tony>pip list
Package      Version
------------ -------
distlib      0.3.6
filelock     3.9.0
pip          21.2.3
platformdirs 3.0.0
setuptools   57.4.0
virtualenv   20.19.0
```

virtualenv 這工具就是建立一個虛擬的環境，可以在這虛擬環境中安裝所需套件或是函式庫。因為這建立起來的環境是虛擬的，當不需要只需刪除即可，完全不會影響到原有作業系統。

pip uninstall virtualenv(刪除虛擬環境套件)

cd c:\

virtualenv dvds (建立一個虛擬環境 dvds)

```
c:\>virtualenv dvds
created virtual environment CPython3.8.10.final.0-64 in 1242ms
  creator CPython3Windows(dest=C:\dvds, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\tony\A
ppData\Local\pypa\virtualenv)
    added seed packages: pip==21.1.2, setuptools==57.0.0, wheel==0.36.2
  activators BashActivator,BatchActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
```

進入虛擬環境:

cd dvds

Scripts\activate

```
c:\>cd dvds

c:\dvds>Scripts\activate

(dvds) c:\dvds>_
```

安裝 Django 套件：

pip3 install Django

```
# pip3 list
```

```
(dvds) c:\dvds>pip3 list
Package    Version
---------- -------
asgiref    3.6.0
Django     4.1.6
pip        23.0
setuptools 67.1.0
sqlparse   0.4.3
tzdata     2022.7
wheel      0.38.4
```

退出虛擬環境:

```
# deactivate
```

建立專案：

```
# Django-admin startproject project1
```

建立應用程式：

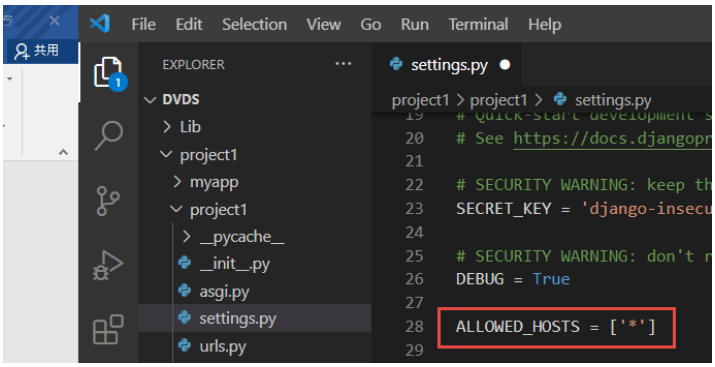
```
# cd project1

# python manage.py startapp myapp
```

```
(dvds) c:\dvds>cd project1

(dvds) c:\dvds\project1>python manage.py startapp myapp
```

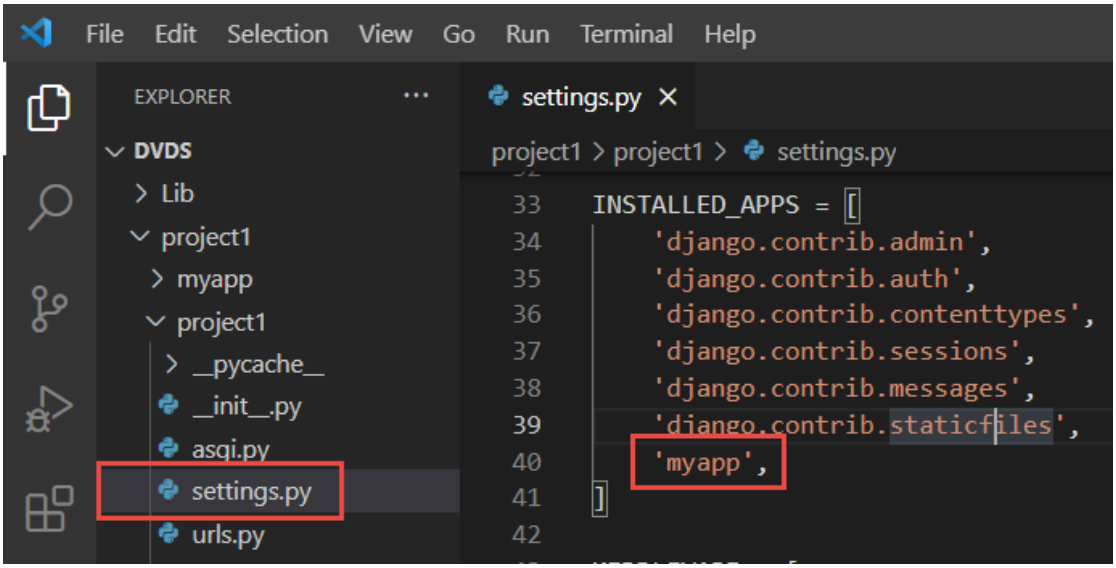
```
# mkdir templates

# mkdir static
```

修改 settings.py

```
ALLOWED_HOSTS = ['*']
```




```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myapp',
]
```




```
TEMPLATES = [
```

```
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

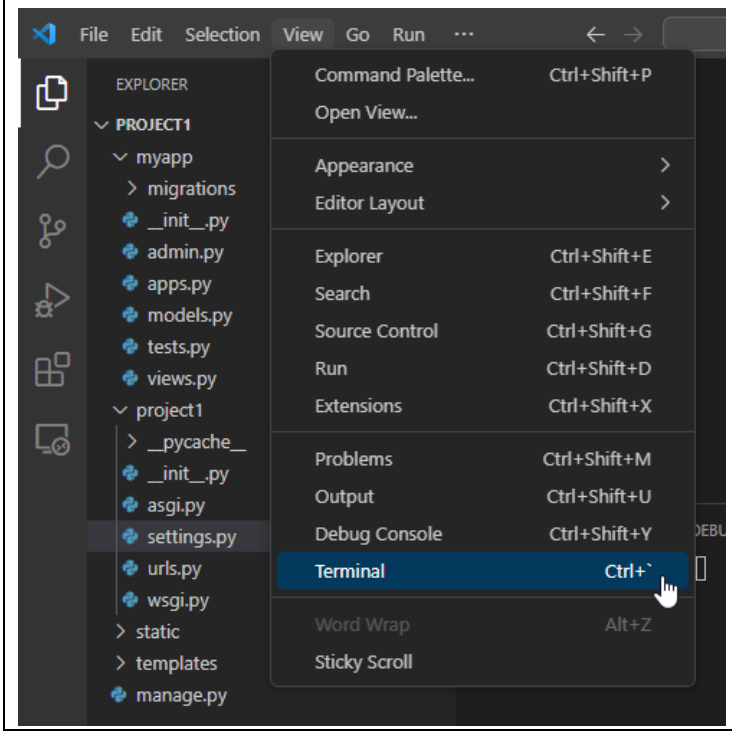
```
53  ROOT_URLCONF = 'project1.urls'
54
55  TEMPLATES = [
56      {
57          'BACKEND': 'django.template.backends.django.DjangoTemplates',
58          'DIRS': [BASE_DIR / 'templates'],
59          'APP_DIRS': True,
60          'OPTIONS': {
61              'context_processors': [
62                  'django.template.context_processors.debug',
63                  'django.template.context_processors.request',
64                  'django.contrib.auth.context_processors.auth',
65                  'django.contrib.messages.context_processors.messages',
66              ],
67          },
68      },
69  ]
```

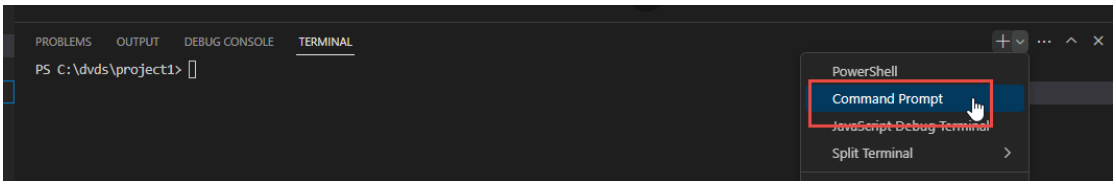
```
LANGUAGE_CODE = 'zh-Hant'
TIME_ZONE = 'Asia/Taipei'
```

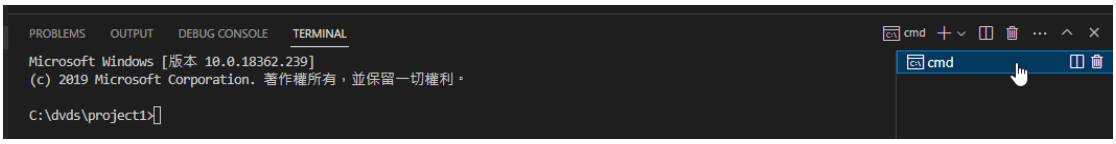
```
107    LANGUAGE_CODE = 'zh-Hant'
108
109    TIME_ZONE = 'Asia/Taipei'
110
111    USE_I18N = True
112
113    USE_TZ = True
```

```
STATIC_URL = 'static/'

STATICFILES_DIRS = [

    BASE_DIR / 'static',

]
```

```
116    # Static files (CSS, JavaScript, Images)
117    # https://docs.djangoproject.com/en/4.1/howto/static-files/
118
119    STATIC_URL = 'static/'
120    STATICFILES_DIRS = [
121        BASE_DIR / 'static',
122    ]
123
```







python manage.py makemigrations myapp (建立 migration 資料檔)

python manage.py migrate (模型與資料庫同步)

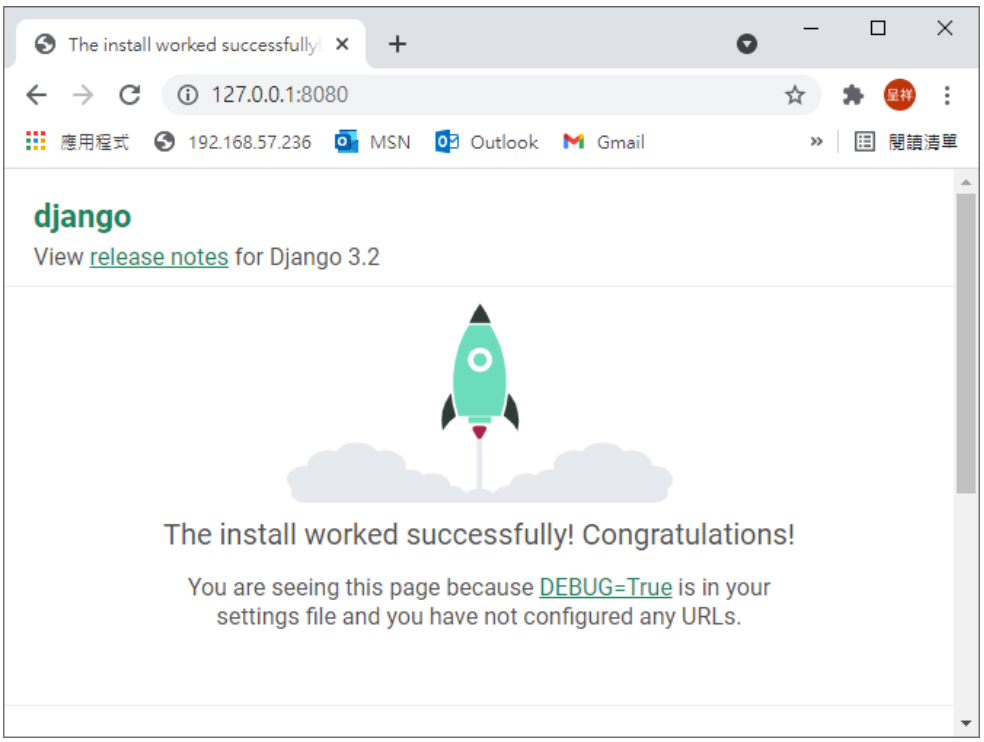
```
(dvds) c:\dvds\project1>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

dir

```
(dvds) c:\dvds\project1>dir
 磁碟區 C 中的磁碟沒有標籤。
 磁碟區序號:  8657-D998

 c:\dvds\project1 的目錄

2021/06/11  下午 03:56    <DIR>          .
2021/06/11  下午 03:56    <DIR>          ..
2021/06/11  下午 03:56           131,072 db.sqlite3
2021/06/11  下午 03:46               686 manage.py
2021/06/11  下午 03:54    <DIR>          myapp
2021/06/11  下午 03:47    <DIR>          project1
2021/06/11  下午 03:49    <DIR>          static
2021/06/11  下午 03:49    <DIR>          templates
               2 個檔案         131,758 位元組
               6 個目錄  133,715,230,720 位元組可用
```

```
#python manage.py runserver 0.0.0.0:8080
```

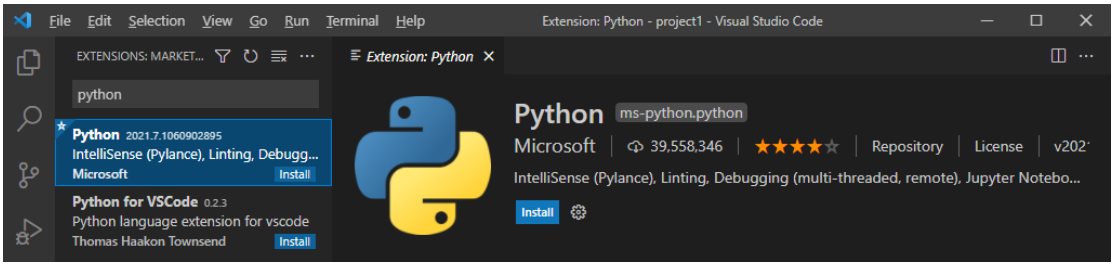
```
(dvds) c:\dvds\project1>python manage.py runserver 0.0.0.0:8080
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 11, 2021 - 16:01:42
Django version 3.2.4, using settings 'project1.settings'
Starting development server at http://0.0.0.0:8080/
Quit the server with CTRL-BREAK.
```

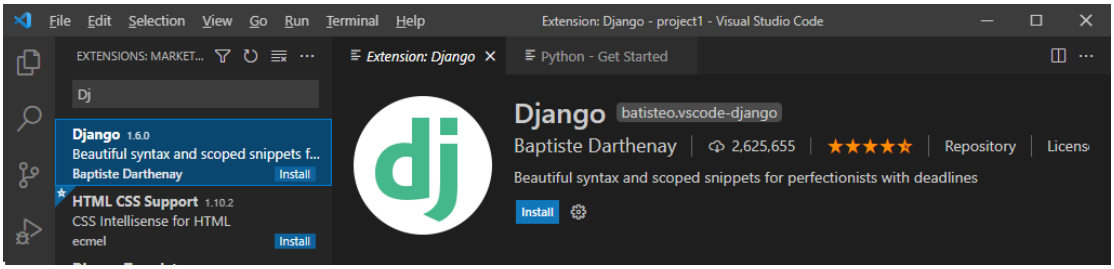

The install worked successfully!
127.0.0.1:8080
應用程式
192.168.57.236
MSN
Outlook
Gmail
閱讀清單
django
View release notes for Django 3.2
The install worked successfully! Congratulations!
You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

3-1 Django 平台建置(windows+ Visual Studio Code)

安裝 VS Code 套件

1. Python
2. Django
3. Django Template





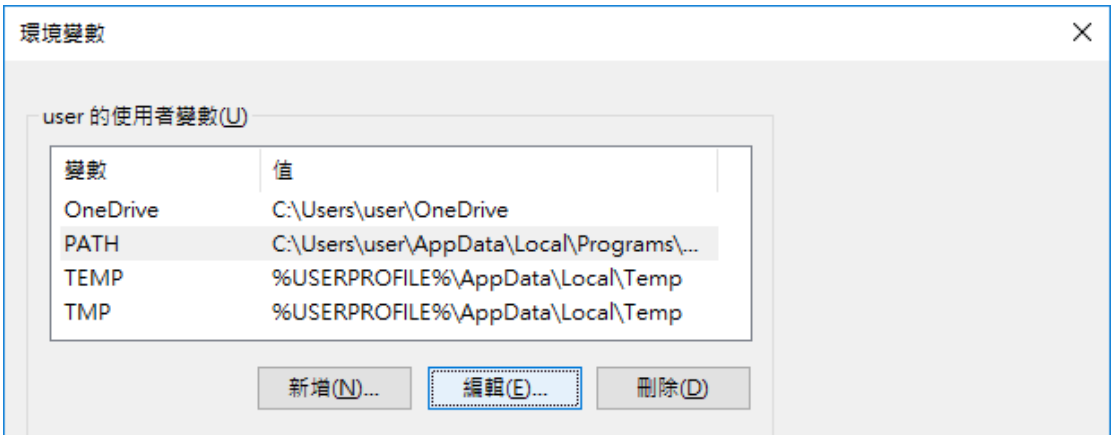
2021 Django Python 開發 VSCode 套件推薦 Top 10 [必裝]:
https://www.codecroc.com.tw/2021/09/30/2021-django-python-dev-vscode-extension-recommendation-top-10-must-have/

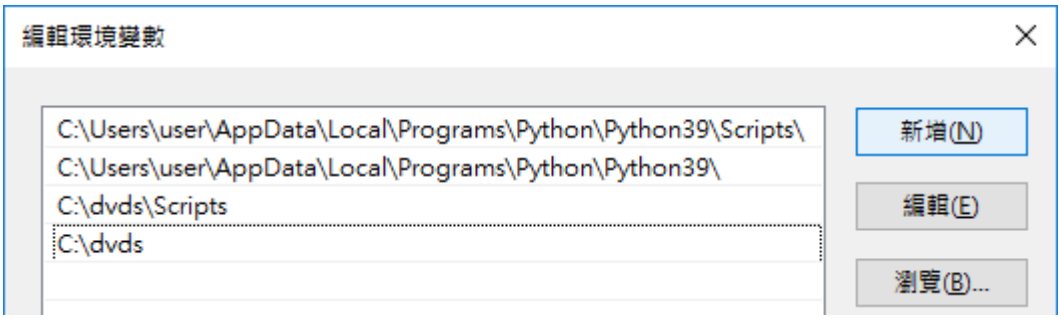
設定環境變數：

本機 > 內容 > 進階設定 > 環境變數，在 Path 變數內新增以下兩個路徑，請依據您 Python 安裝的路徑及名稱而定：

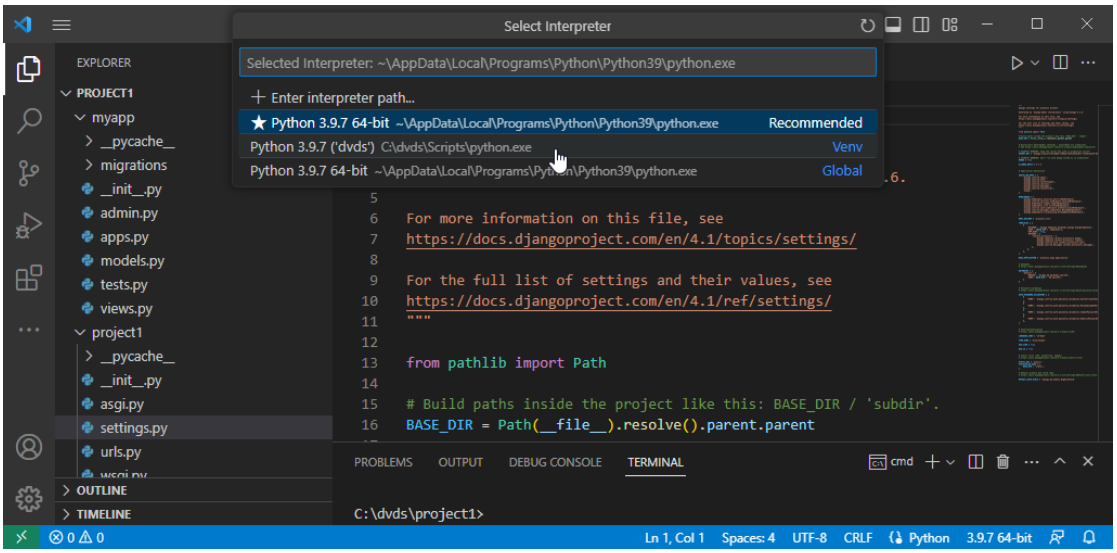
C:\dvds

C:\dvds\Scripts

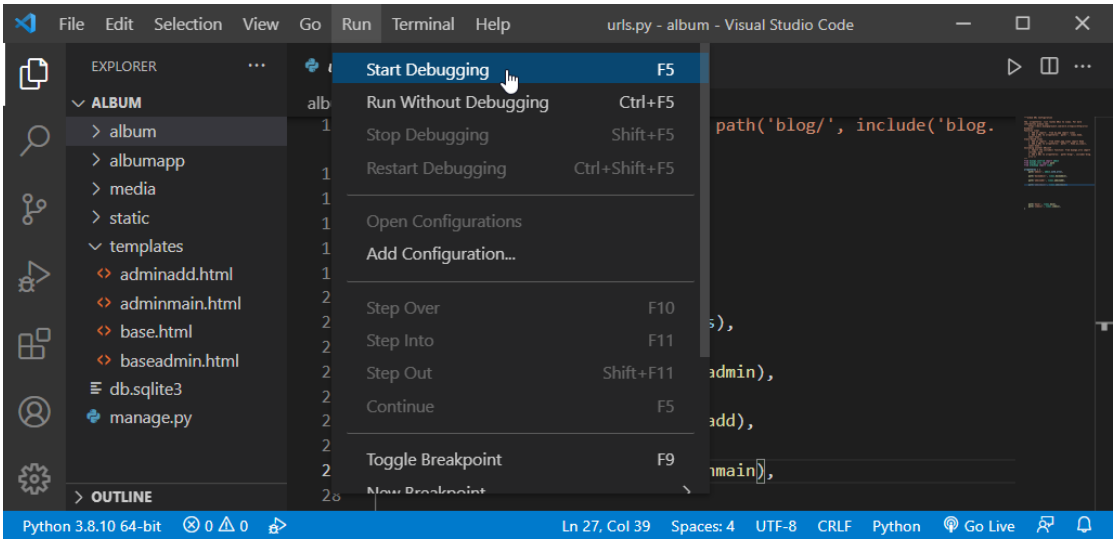




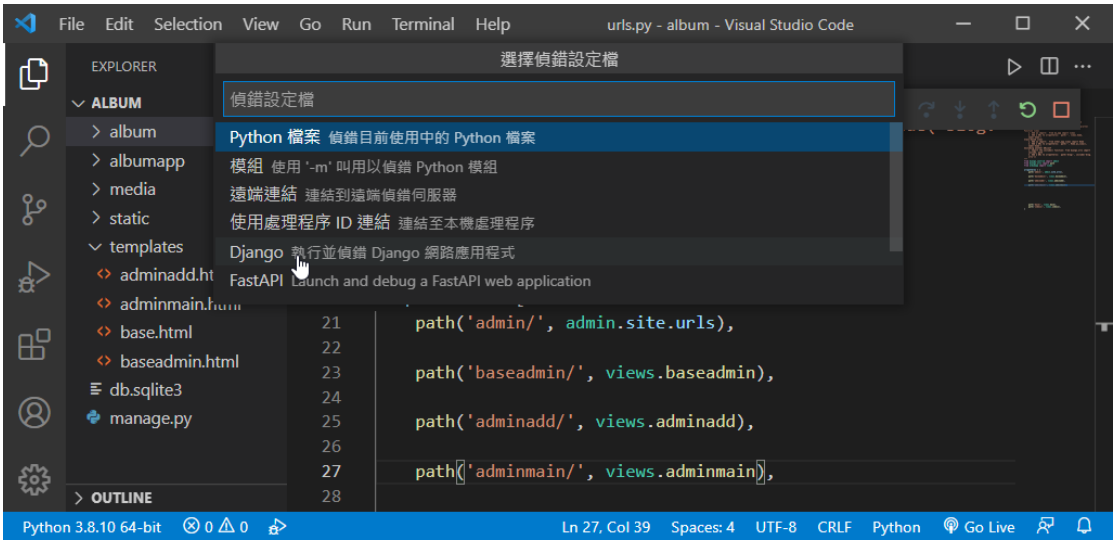
重啟 vscode 後，進入到專案，點選最下一排(左下角或右下角)python...，選擇當前專案使用的虛擬目錄，如下：

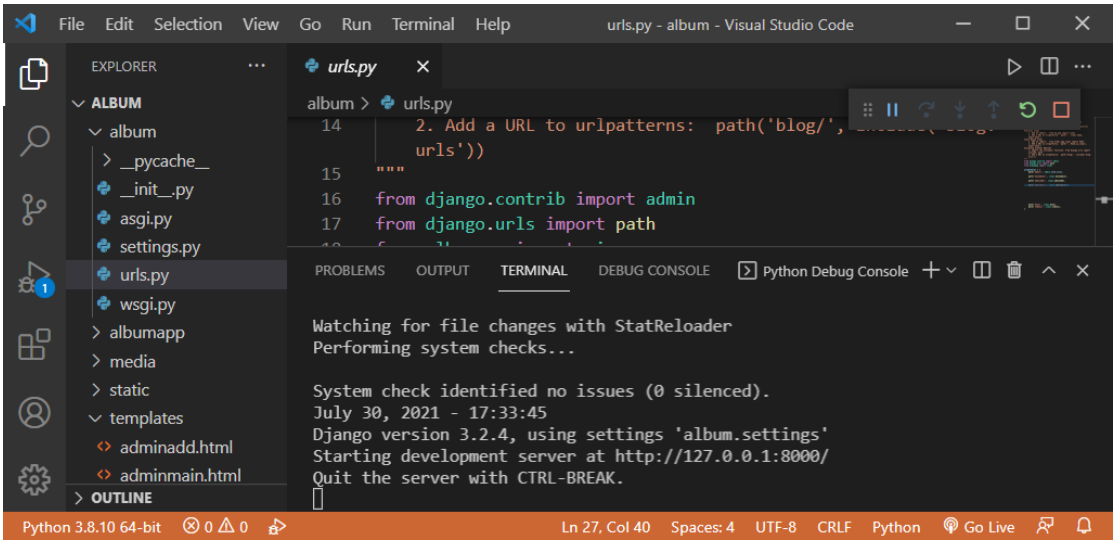


點選「start debugging」

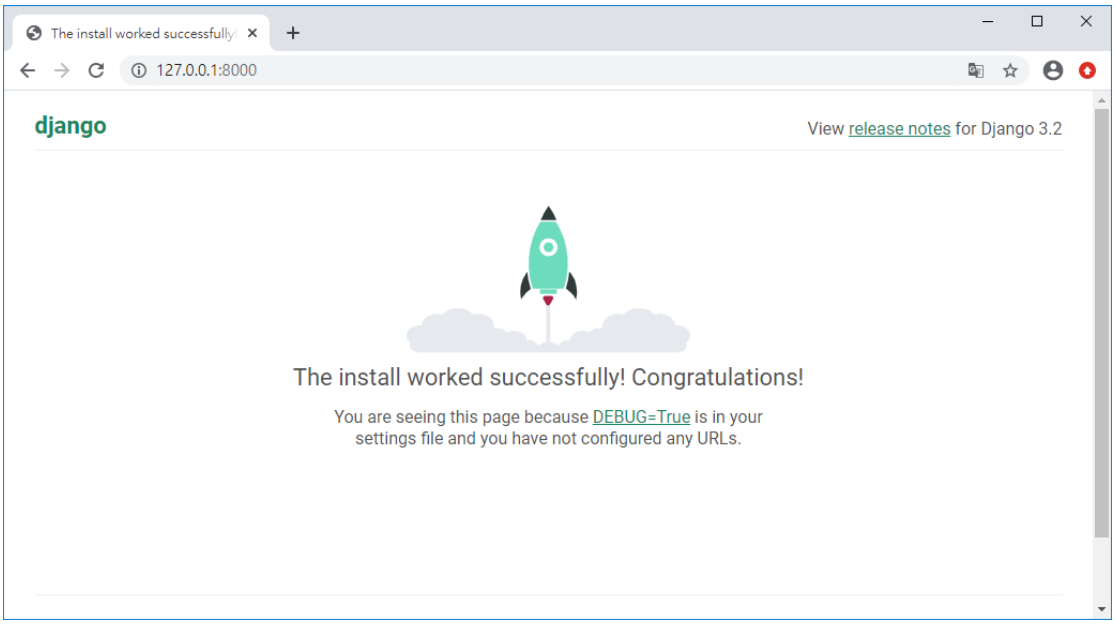

File Edit Selection View Go Run Terminal Help
urls.py - album - Visual Studio Code
EXPLORER
ALBUM
album
albumapp
media
static
templates
adminadd.html
adminmain.html
base.html
baseadmin.html
db.sqlite3
manage.py
OUTLINE
Start Debugging F5
Run Without Debugging Ctrl+F5
Stop Debugging Shift+F5
Restart Debugging Ctrl+Shift+F5
Open Configurations
Add Configuration...
Step Over F10
Step Into F11
Step Out Shift+F11
Continue F5
Toggle Breakpoint F9
path('blog/', include('blog.
Python 3.8.10 64-bit
Ln 27, Col 39 Spaces: 4 UTF-8 CRLF Python Go Live

點選 Django


File Edit Selection View Go Run Terminal Help
urls.py - album - Visual Studio Code
選擇偵錯設定檔
偵錯設定檔
Python 檔案 偵錯目前使用中的 Python 檔案
模組 使用 '-m' 叫用以偵錯 Python 模組
遠端連結 連結到遠端偵錯伺服器
使用處理程序 ID 連結 連結至本機處理程序
Django 執行並偵錯 Django 網路應用程式
FastAPI Launch and debug a FastAPI web application
path('admin/', admin.site.urls),
path('baseadmin/', views.baseadmin),
path('adminadd/', views.adminadd),
path('adminmain/', views.adminmain),
Python 3.8.10 64-bit
Ln 27, Col 39 Spaces: 4 UTF-8 CRLF Python Go Live


File Edit Selection View Go Run Terminal Help
urls.py - album - Visual Studio Code
urls.py
album > urls.py
2. Add a URL to urlpatterns: path('blog/',
urls'))
"""
from django.contrib import admin
from django.urls import path
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Python Debug Console
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
July 30, 2021 - 17:33:45
Django version 3.2.4, using settings 'album.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
Python 3.8.10 64-bit
Ln 27, Col 40 Spaces: 4 UTF-8 CRLF Python Go Live

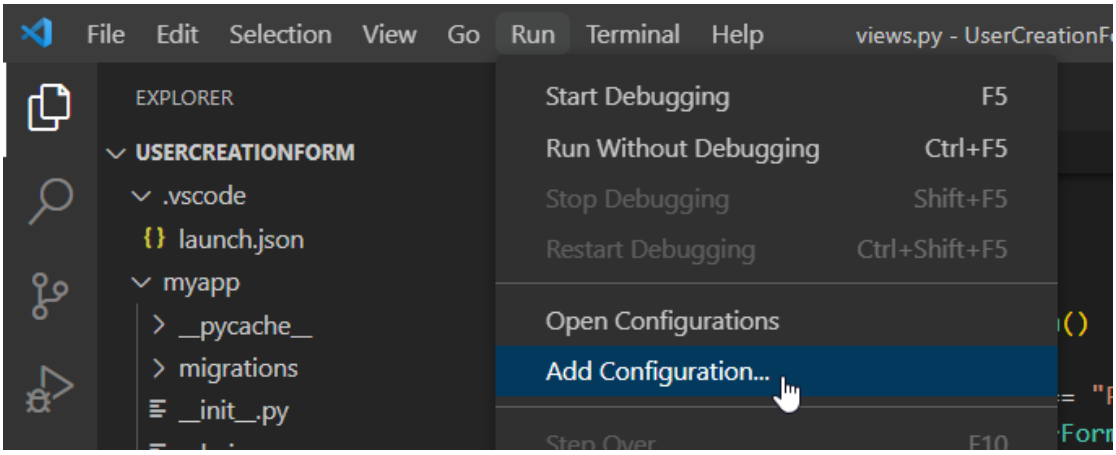
輸入網頁測試如下：



參考文獻:

https://yijay131724.blogspot.com/2021/02/python-visual-studio-code-python.html
https://dotblogs.com.tw/brian90191/2019/04/02/115926

允許非本機連線:



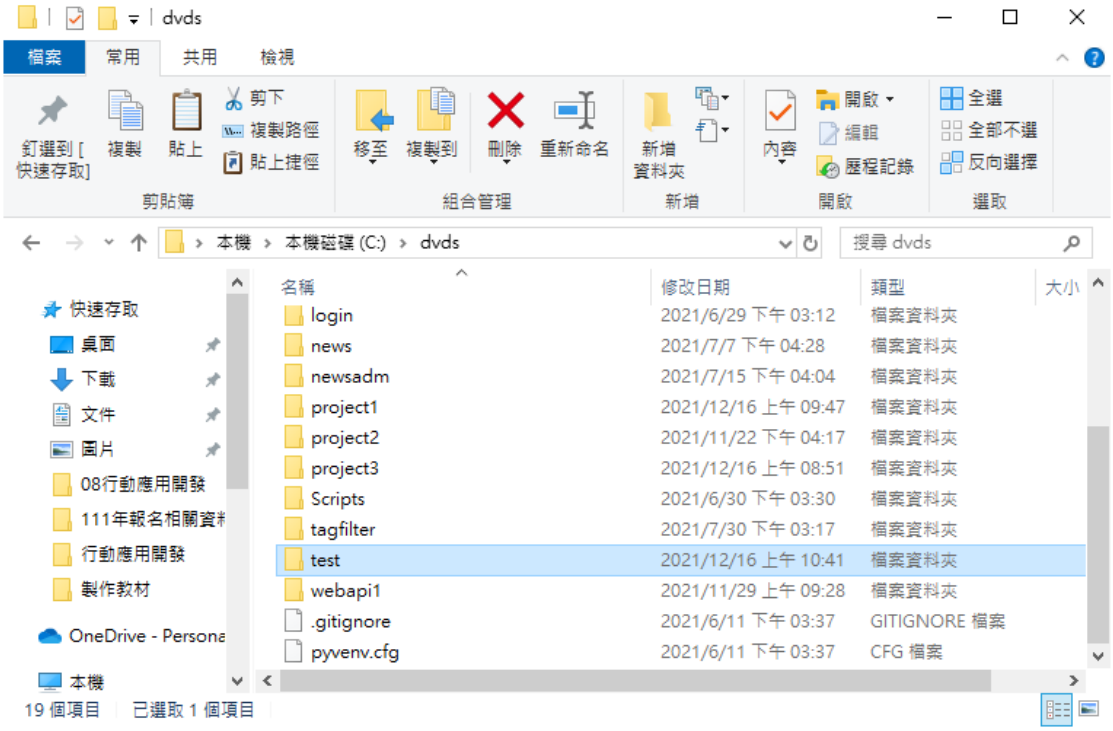
views.py launch.json

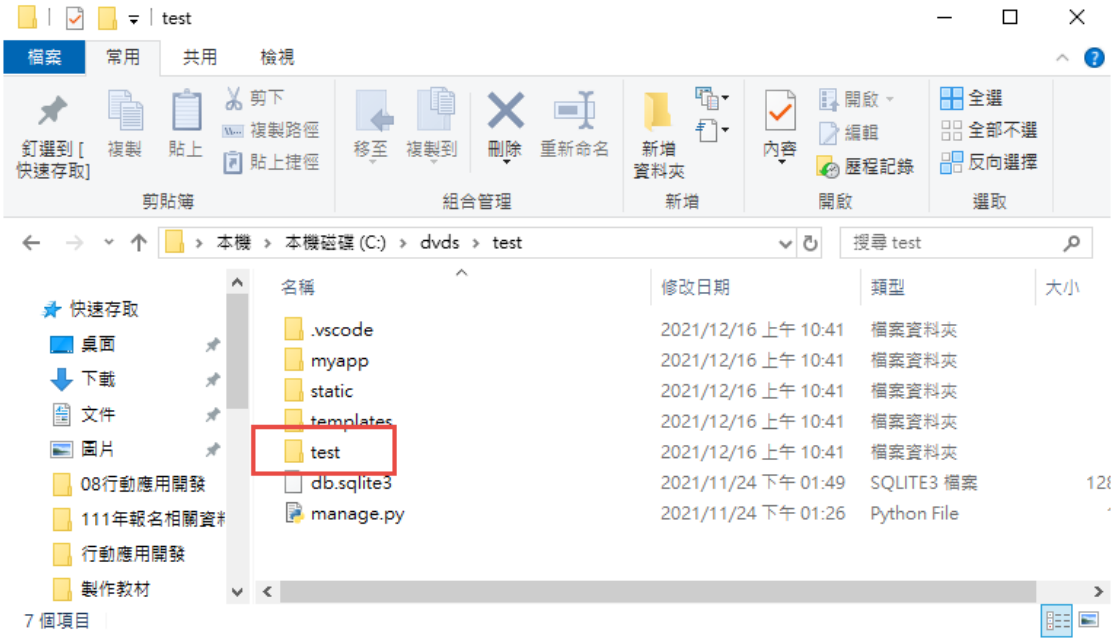
.vscode > launch.json > ...

```
1  {
2      // Use IntelliSense to learn about possible attributes.
3      // Hover to view descriptions of existing attributes.
4      // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5      "version": "0.2.0",
6      "configurations": [
7          {
8              "name": "Python: Django",
9              "type": "python",
10             "request": "launch",
11             "program": "${workspaceFolder}\\manage.py",
12             "args": [
13                 "runserver","0.0.0.0:8080"
14             ],
15             "django": true,
16             "justMyCode": true
17         }
18     ]
19 }
```

3-1 複製專案

1、複製專案並重命名 test，內層也要 test





2、使用 vscode 開啟此專案，更改檔案內的參數名稱
在 settings.py、wsgi.py、manage.py 與 asgi.py 中更改名稱
結果如下:

settings.py

test > settings.py > ...

```
46      'django.contrib.sessions.middleware.SessionMiddleware',
47      'django.middleware.common.CommonMiddleware',
48      'django.middleware.csrf.CsrfViewMiddleware',
49      'django.contrib.auth.middleware.AuthenticationMiddleware',
50      'django.contrib.messages.middleware.MessageMiddleware',
51      'django.middleware.clickjacking.XFrameOptionsMiddleware',
52  ]
53
54  ROOT_URLCONF = 'test.urls'
```

settings.py

test > settings.py > ...

```
70  ]
71
72  WSGI_APPLICATION = 'test.wsgi.application'
73
```

settings.py wsgi.py

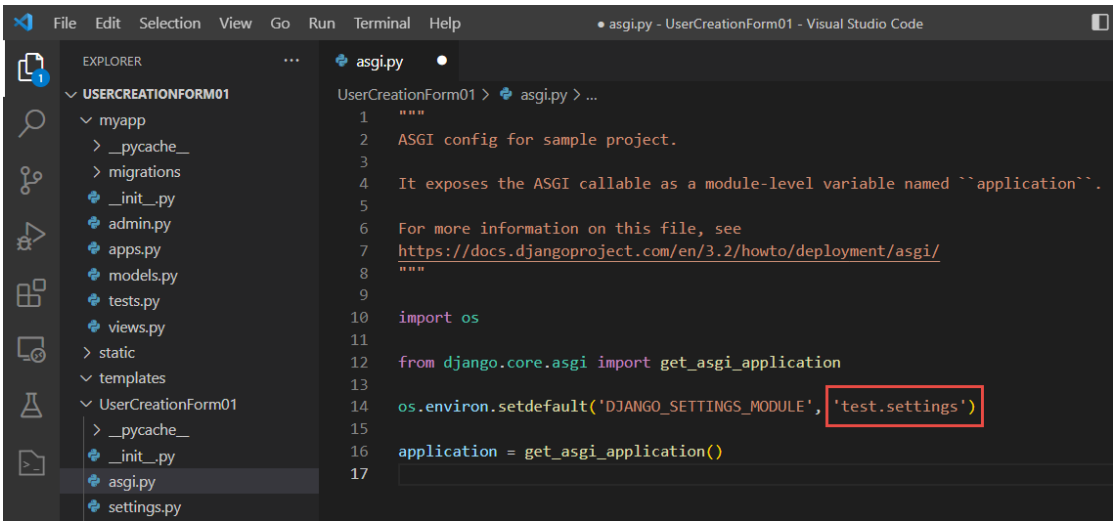
test > wsgi.py > ...

```
4   It exposes the WSGI callable as a module-level variable named ``a
5
6   For more information on this file, see
7   https://docs.djangoproject.com/en/3.2/howto/deployment/wsgi/
8   """
9
10  import os
11
12  from django.core.wsgi import get_wsgi_application
13
14  os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'test.settings')
15
16  application = get_wsgi_application()
```

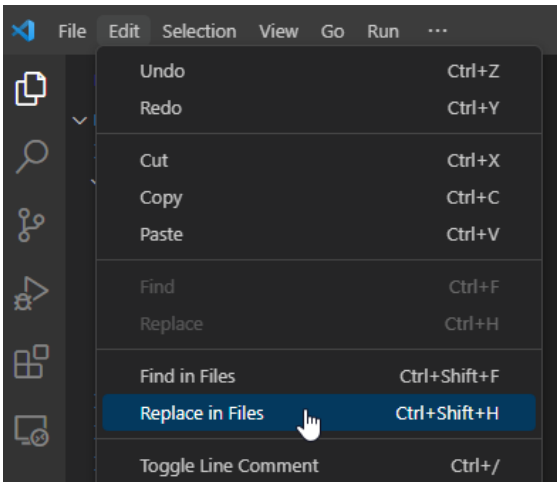
settings.py manage.py wsgi.py

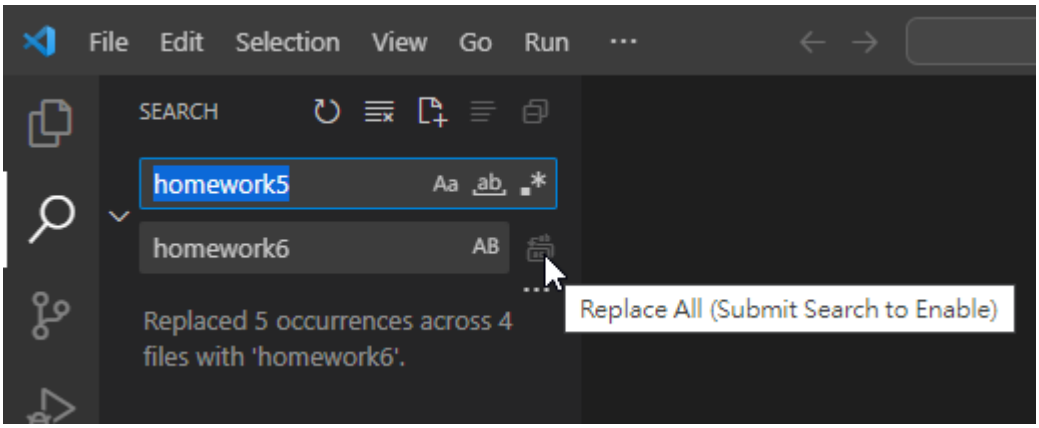
manage.py > main

```
7   def main():
8       """Run administrative tasks."""
9       os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'test.settings')
10      try:
11          from django.core.management import execute_from_command_line
```


File Edit Selection View Go Run Terminal Help
• asgi.py - UserCreationForm01 - Visual Studio Code
EXPLORER
USERCREATIONFORM01
myapp
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
tests.py
views.py
static
templates
UserCreationForm01
__pycache__
__init__.py
asgi.py
settings.py
asgi.py
UserCreationForm01 > asgi.py > ...
"""
ASGI config for sample project.
It exposes the ASGI callable as a module-level variable named ``application``.
For more information on this file, see
https://docs.djangoproject.com/en/3.2/howto/deployment/asgi/
"""
import os
from django.core.asgi import get_asgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'test.settings')
application = get_asgi_application()

2、使用取代更改名稱


File Edit Selection View Go Run
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Find Ctrl+F
Replace Ctrl+H
Find in Files Ctrl+Shift+F
Replace in Files Ctrl+Shift+H
Toggle Line Comment Ctrl+/


File Edit Selection View Go Run
SEARCH
homework5
homework6
Replaced 5 occurrences across 4 files with 'homework6'.
Replace All (Submit Search to Enable)

3-1 Django 平台建置(Linux)

sudo apt-get update -y

sudo apt-get upgrade -y (先不要用)

python3 --version (確認版本)

```
(dvdsenv) pi@raspberrypi:~/dvds2 $ python3 --version
Python 3.5.3
```

- 建立虛擬環境

sudo pip3 install virtualenv (安裝)

sudo pip3 uninstall virtualenv (刪除)

pip3 list

virtualenv dvds

(使用 virtualenv 建立虛擬環境名稱為 dvds)

```
pi@raspberrypi:~ $ sudo virtualenv dvds
created virtual environment CPython3.5.3.final.0-32 in 1553ms
  creator CPython3Posix(dest=/home/pi/dvds, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/root/.local/shar
e/virtualenv)
    added seed packages: pip==20.3.4, setuptools==50.3.2, wheel==0.37.0
  activators BashActivator,FishActivator,CShellActivator,NushellActivator,PowerShellActivator,PythonActivator
```

source dvds/bin/activate (啟動虛擬環境)

```
pi@raspberrypi:~ $ source dvds/bin/activate
(dvds) pi@raspberrypi:~ $
```

python --version (查看 python 版本)

```
(dvds) pi@raspberrypi:~ $ python --version
Python 3.9.2
```

deactivate(離開虛擬環境)

- Installing Django:

pip3 install django

```
(dvds) pi@raspberrypi:~ $ pip install django
DEPRECATION: Python 3.5 reached the end of its life on September 13th, 2020. Please upgrade your Python as Python 3.5 is n
o longer maintained. pip 21.0 will drop support for Python 3.5 in January 2021. pip 21.0 will remove support for this func
tionality.
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting django
  Using cached https://www.piwheels.org/simple/django/Django-2.2.25-py3-none-any.whl (7.5 MB)
Collecting sqlparse>=0.2.2
  Using cached https://www.piwheels.org/simple/sqlparse/sqlparse-0.4.2-py3-none-any.whl (40 kB)
Collecting pytz
  Using cached https://www.piwheels.org/simple/pytz/pytz-2021.3-py3-none-any.whl (511 kB)
Installing collected packages: sqlparse, pytz, django
Successfully installed django-2.2.25 pytz-2021.3 sqlparse-0.4.2
```

pip3 list(查詢有哪些安裝的套件)

```
(dvds) pi@raspberrypi:~ $ pip3 list
Package    Version
---------- -------
asgiref    3.6.0
Django     4.1.6
pip        23.0
setuptools 67.1.0
sqlparse   0.4.3
wheel      0.38.4
```

建立專案：

cd dvds

django-admin startproject project1

ls

```
(dvds) pi@raspberrypi:~/dvds $ django-admin startproject project1
(dvds) pi@raspberrypi:~/dvds $ ls
bin  lib  project1  pyvenv.cfg
```

建立應用程式：

cd project1

python manage.py startapp myapp

ls

```
(dvds) pi@raspberrypi:~/dvds $ cd project1/
(dvds) pi@raspberrypi:~/dvds/project1 $ python manage.py startapp myapp
(dvds) pi@raspberrypi:~/dvds/project1 $ ls
manage.py  myapp  project1
```

mkdir templates

```
# mkdir static
```

修改 settings.py

```
# vim project1/settings.py

ALLOWED_HOSTS = ['*']

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myapp',
]
```

```
pi@raspberrypi: ~/dvds/project1
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-qirwg9*#z10u#sjtzhf-%wlrsqy7ndnv5l0&3=%r#j
   )nk6r-rg'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = ['*']
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'myapp',
41 ]
42
-- INSERT --
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

```
pi@raspberrypi: ~/dvds/project1
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
51 ]
52
53 ROOT_URLCONF = 'project1.urls'
54
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates',
58         'DIRS': [BASE_DIR / 'templates'],
59         'APP_DIRS': True,
60         'OPTIONS': {
61             'context_processors': [
62                 'django.template.context_processors.debug',
63                 'django.template.context_processors.request',
64                 'django.contrib.auth.context_processors.auth',
65                 'django.contrib.messages.context_processors.messages',
66             ],
67         },
68     },
69 ]
70
71 WSGI_APPLICATION = 'project1.wsgi.application'
-- INSERT --
```

```
LANGUAGE_CODE = 'zh-Hant'

TIME_ZONE = 'Asia/Taipei'
```

pi@raspberrypi: ~/dvds/project1

```
 95     {
 96         'NAME': 'django.contrib.auth.password_validation.CommonPasswordV
    alidator',
 97     },
 98     {
 99         'NAME': 'django.contrib.auth.password_validation.NumericPassword
    Validator',
100     },
101 ]
102
103
104 # Internationalization
105 # https://docs.djangoproject.com/en/4.1/topics/i18n/
106
107 LANGUAGE_CODE = 'zh-Hant'
108
109 TIME_ZONE = 'Asia/Taipei'
110
111 USE_I18N = True
112
113 USE_TZ = True
114
115
-- INSERT --
```

```
STATIC_URL = 'static/'

STATICFILES_DIRS = [

    BASE_DIR / 'static',

]
```

pi@raspberrypi: ~/dvds/project1

```
106
107 LANGUAGE_CODE = 'zh-Hant'
108
109 TIME_ZONE = 'Asia/Taipei'
110
111 USE_I18N = True
112
113 USE_TZ = True
114
115
116 # Static files (CSS, JavaScript, Images)
117 # https://docs.djangoproject.com/en/4.1/howto/static-files/
118
119 STATIC_URL = 'static/'
120
121 STATICFILES_DIRS = [
122     BASE_DIR / 'static',
123 ]
124
125
126
127 # Default primary key field type
128 # https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field
-- INSERT --
```

python manage.py makemigrations myapp (建立 migration 資料檔)

python manage.py migrate (模型與資料庫同步)

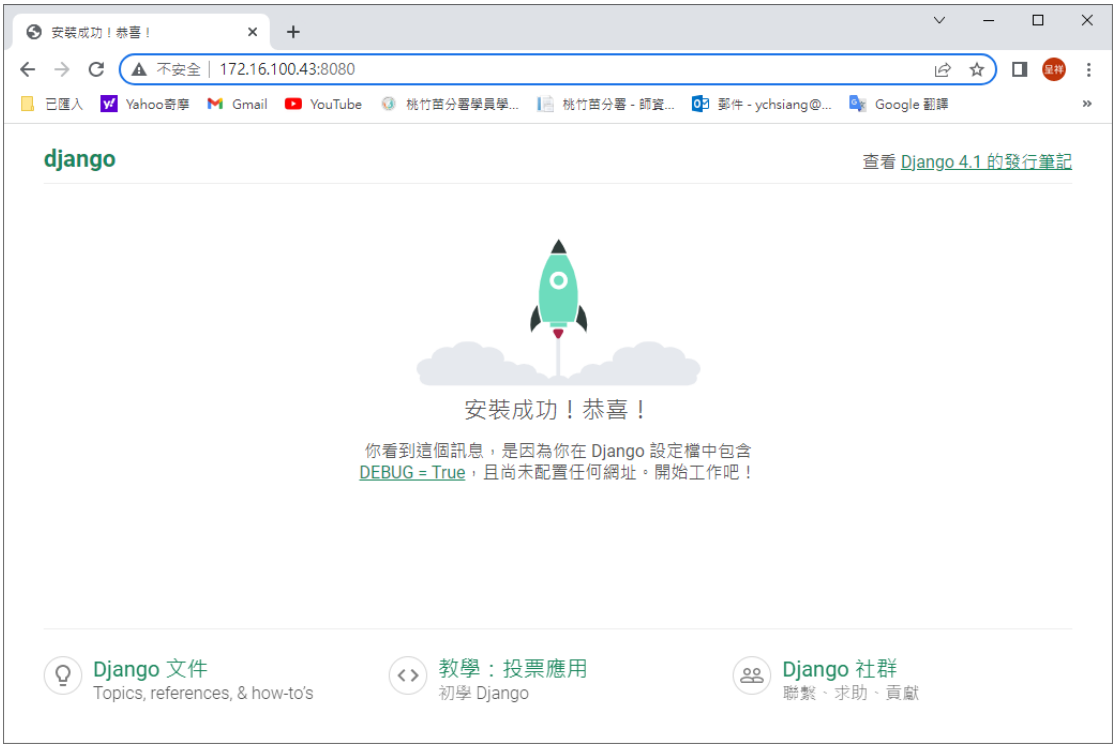
```
(dvds) pi@raspberrypi:~/dvds/project1 $ python manage.py makemigrations myapp
No changes detected in app 'myapp'
(dvds) pi@raspberrypi:~/dvds/project1 $ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
```

ls

```
(dvds) pi@raspberrypi:~/dvds/project1 $ ls
db.sqlite3  manage.py  myapp  project1  static  templates
```

python manage.py runserver 0.0.0.0:8080

```
pi@raspberrypi: ~/dvds/project1
(dvds) pi@raspberrypi:~/dvds/project1 $ python manage.py runserver 0.0.0.0:8080
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
February 13, 2023 - 11:45:18
Django version 4.1.6, using settings 'project1.settings'
Starting development server at http://0.0.0.0:8080/
Quit the server with CONTROL-C.
[13/Feb/2023 11:45:18] "GET / HTTP/1.1" 200 10626
[13/Feb/2023 11:45:19] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[13/Feb/2023 11:45:19] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.
1" 200 86184
[13/Feb/2023 11:45:19] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP
/1.1" 200 85876
[13/Feb/2023 11:45:19] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1
.1" 200 85692
Not Found: /favicon.ico
[13/Feb/2023 11:45:19] "GET /favicon.ico HTTP/1.1" 404 2116
```



參考文獻:

https://mikesmithers.wordpress.com/2017/02/21/configuring-django-with-apache-on-a-raspberry-pi/

https://www.cnblogs.com/baby123/p/12122703.html

https://developer.mozilla.org/zh-TW/docs/Learn/Server-side/Django/skeleton_website

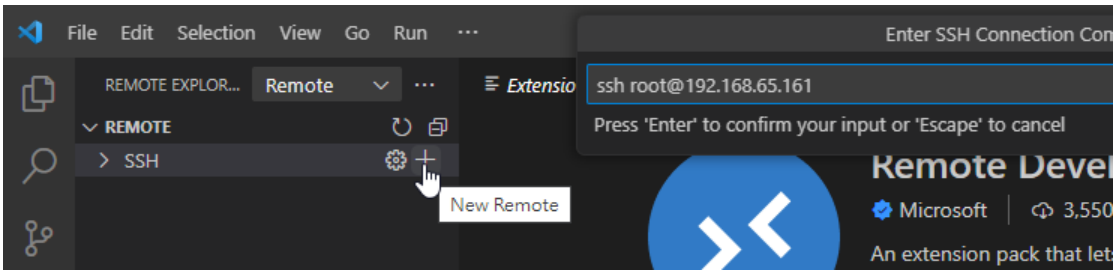
Django 中直接使用 sql 语句 操作数据库
https://blog.csdn.net/haeasringnar/article/details/82080476

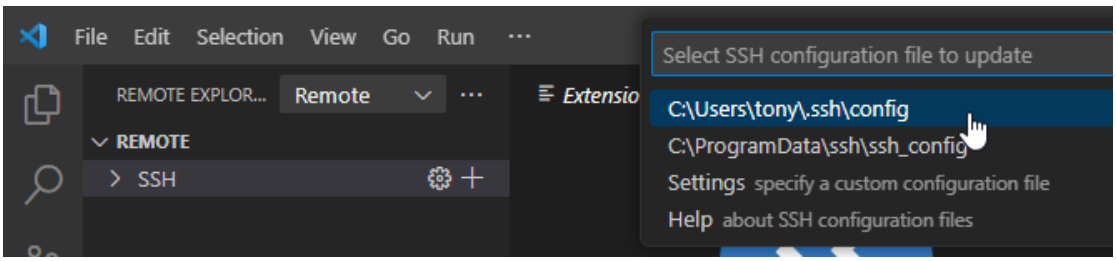
https://openhome.cc/Gossip/CodeData/PythonTutorial/AppModelPy3.html

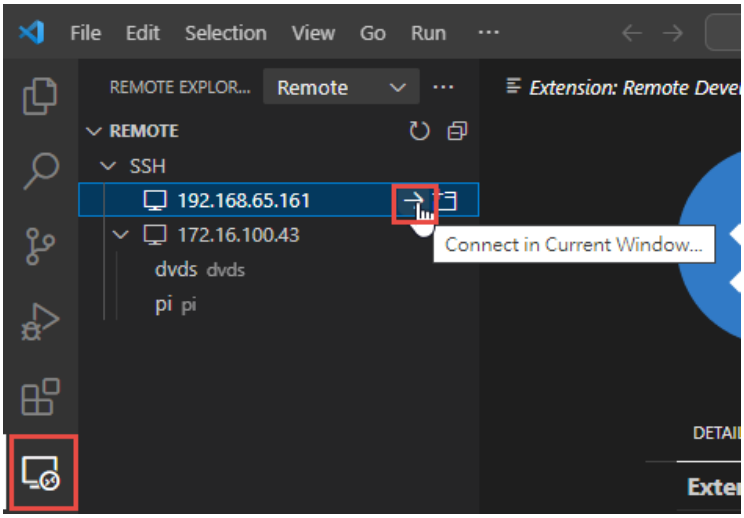
在 Django 使用 MySQL 資料庫
https://jerrynest.io/django-mysql-database/

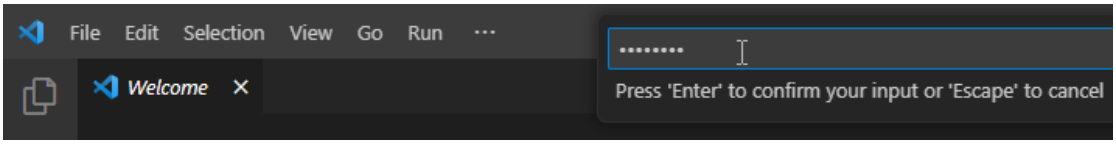
3-1 Django 平台建置(Linux+ Visual Studio Code+Remote Development)

1、安裝 Remote- Development
2、建立連結

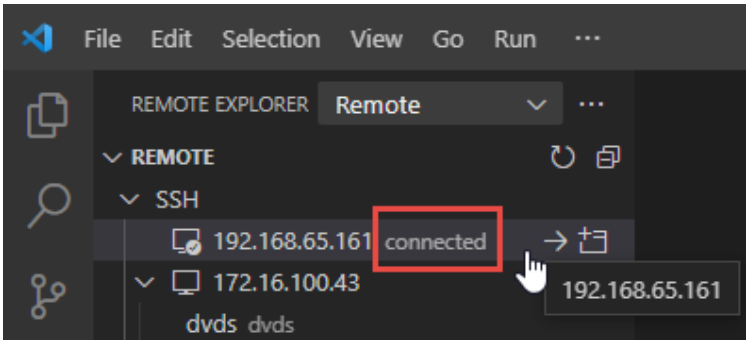




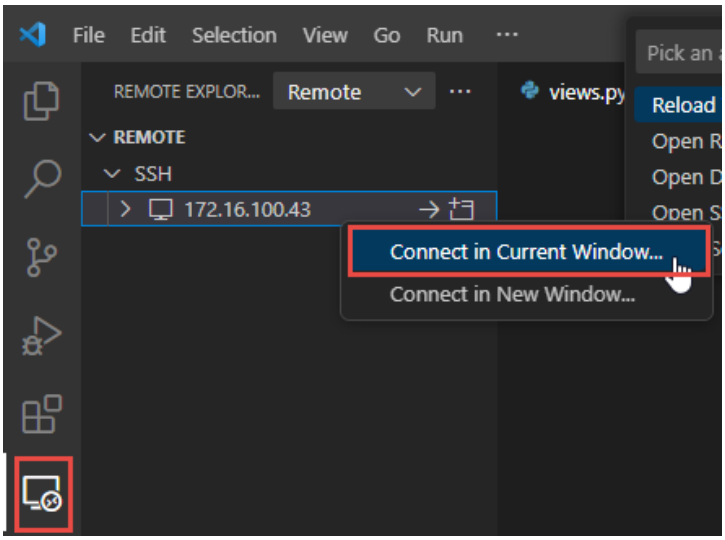


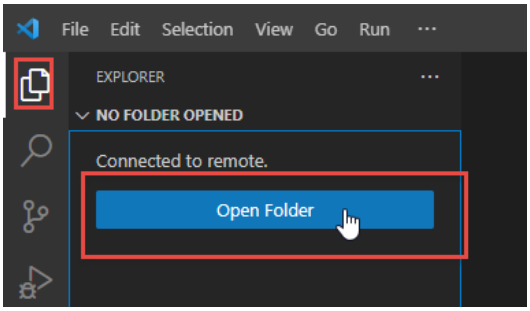


已連結如下圖

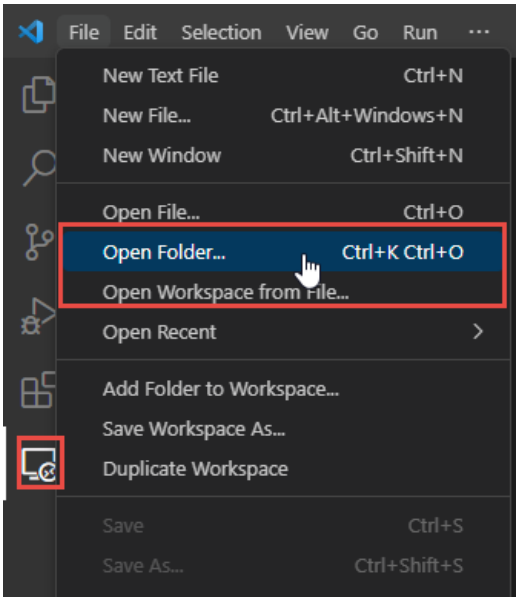


or 若未連結，則按「connect in current window」

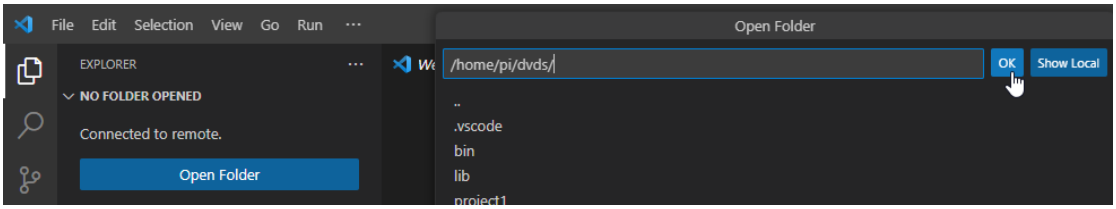


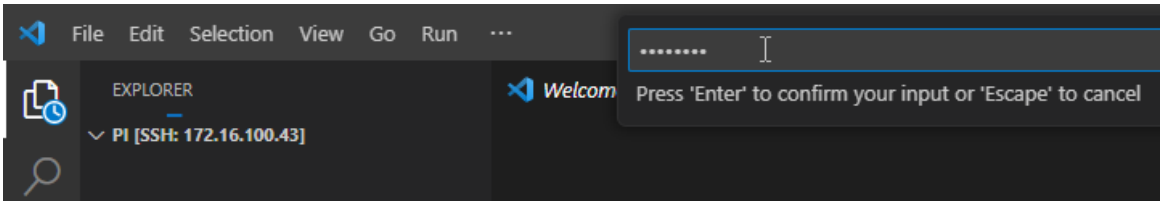


or

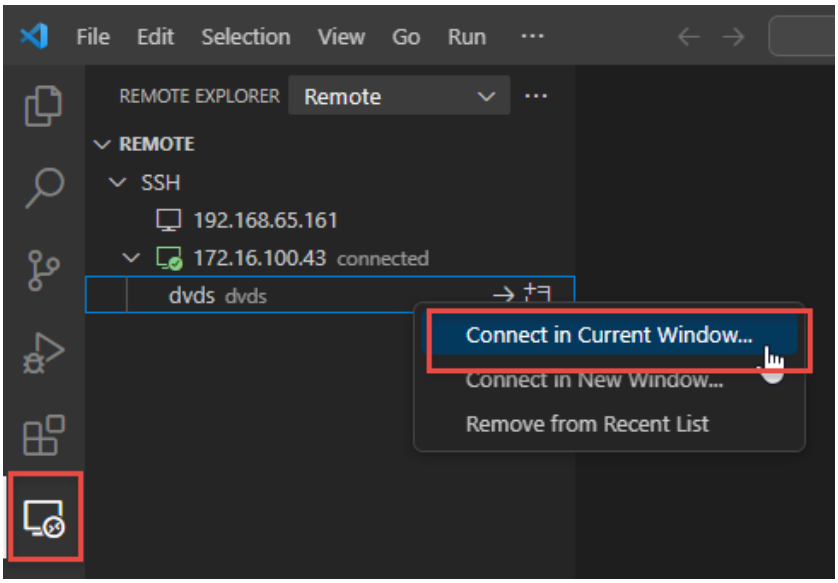


注意:指定到/home/pi/dvds/，不要指定到專案之下/home/pi/dvds/projectName/

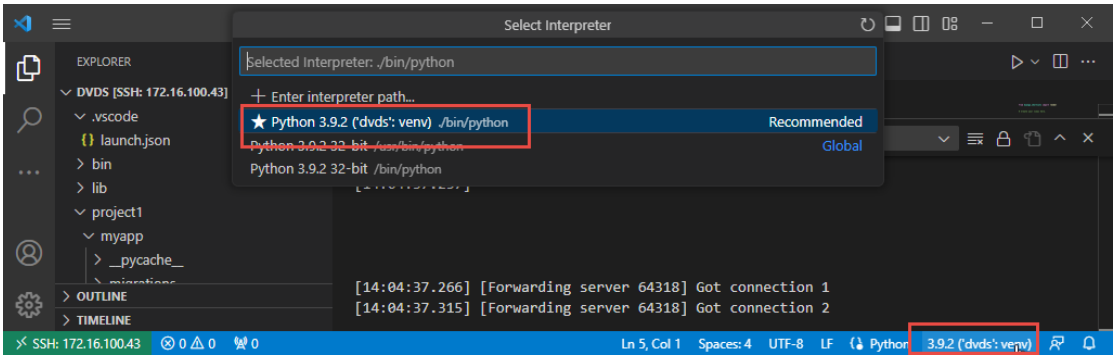


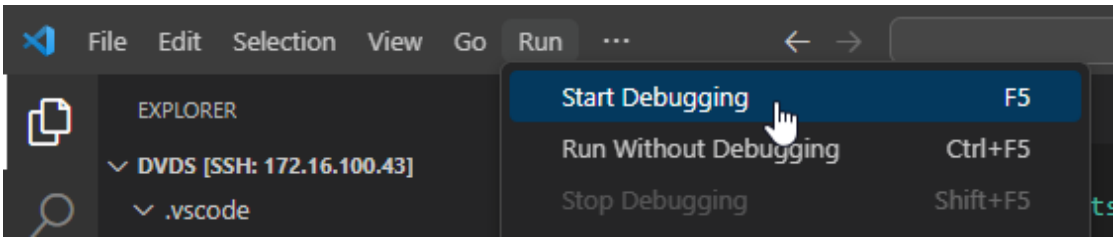


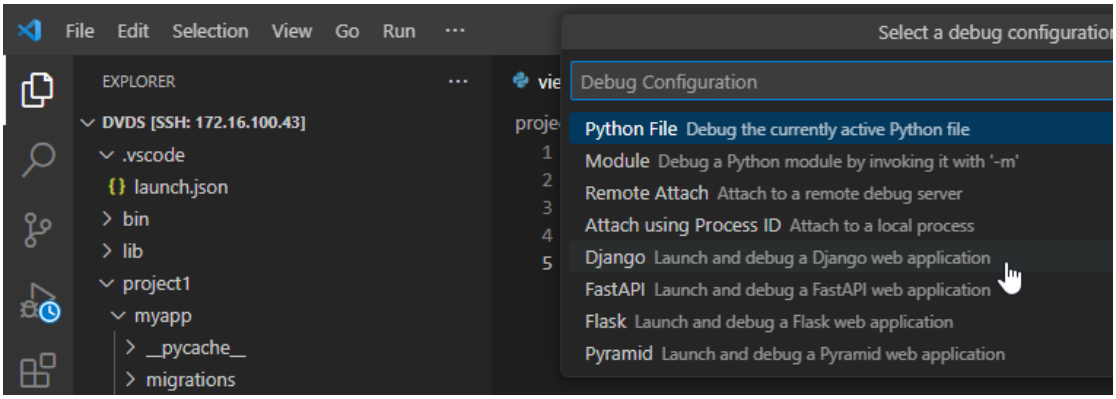
若已安裝資料夾連結，則由下圖操作

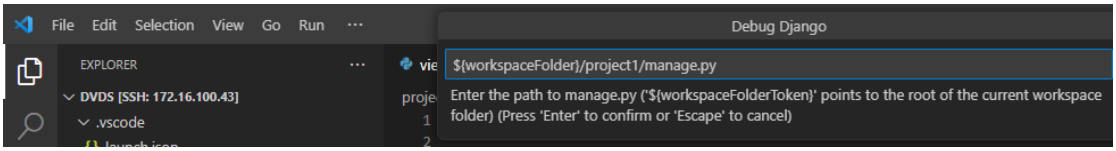


2、安裝 python 套件:

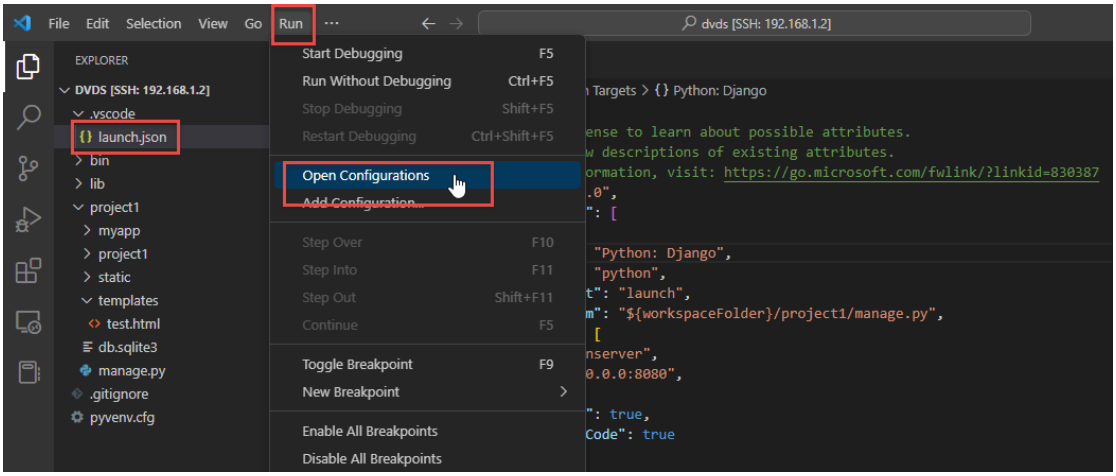





File Edit Selection View Go Run
Select a debug configuratio
EXPLORER
DVDS [SSH: 172.16.100.43]
.vscode
launch.json
bin
lib
project1
myapp
__pycache__
migrations
Debug Configuration
Python File Debug the currently active Python file
Module Debug a Python module by invoking it with '-m'
Remote Attach Attach to a remote debug server
Attach using Process ID Attach to a local process
Django Launch and debug a Django web application
FastAPI Launch and debug a FastAPI web application
Flask Launch and debug a Flask web application
Pyramid Launch and debug a Pyramid web application


File Edit Selection View Go Run
Debug Django
EXPLORER
DVDS [SSH: 172.16.100.43]
.vscode
${workspaceFolder}/project1/manage.py
Enter the path to manage.py ('${workspaceFolderToken}' points to the root of the current workspace folder) (Press 'Enter' to confirm or 'Escape' to cancel)

修正參數


File Edit Selection View Go Run
dvds [SSH: 192.168.1.2]
EXPLORER
DVDS [SSH: 192.168.1.2]
.vscode
launch.json
bin
lib
project1
myapp
project1
static
templates
test.html
db.sqlite3
manage.py
.gitignore
pyvenv.cfg
Start Debugging F5
Run Without Debugging Ctrl+F5
Stop Debugging Shift+F5
Restart Debugging Ctrl+Shift+F5
Open Configurations
Add Configuration...
Step Over F10
Step Into F11
Step Out Shift+F11
Continue F5
Toggle Breakpoint F9
New Breakpoint
Enable All Breakpoints
Disable All Breakpoints
Targets > { } Python: Django

launch.json

.vscode > launch.json > ...

```
1  {
2      // Use IntelliSense to learn about possible attributes.
3      // Hover to view descriptions of existing attributes.
4      // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5      "version": "0.2.0",
6      "configurations": [
7          {
8              "name": "Python: Django",
9              "type": "python",
10             "request": "launch",
11             "program": "${workspaceFolder}/project1/manage.py",
12             "args": [
13                 "runserver",
14                 "0.0.0.0:8080",
15             ],
16             "django": true,
17             "justMyCode": true
18         }
19     ]
20 }
```

若要重來，或改變專案，可以刪除或修正下圖參數

launch.json

.vscode > launch.json > ...

```
1  {
2      // Use IntelliSense to learn about possible attributes.
3      // Hover to view descriptions of existing attributes.
4      // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5      "version": "0.2.0",
6      "configurations": [
7          {
8              "name": "Python: Django",
9              "type": "python",
10             "request": "launch",
11             "program": "${workspaceFolder}/project1/manage.py",
12             "args": [
13                 "runserver","0.0.0.0:8080",
14             ],
15             "django": true,
16             "justMyCode": true
17         }
18     ]
19 }
```

launch.json × views.py

.vscode > launch.json > Launch Targets > {} Python: Django

```
1  {
2      // Use IntelliSense to learn about possible attributes.
3      // Hover to view descriptions of existing attributes.
4      // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5      "version": "0.2.0",
6      "configurations": [
7          {
8              "name": "Python: Django",
9              "type": "python",
10             "request": "launch",
11             "program": "${workspaceFolder}/project1/manage.py",
12             "args": [
13                 "runserver",
14                 "0.0.0.0:8080",
15             ],
16             "django": true,
17             "justMyCode": true
18         }
19     ]
20 }
```

File Edit Selection View Go Run ··· dvds [SSH: 172.16.100.43]

EXPLORER

DVDS [SSH: 172.16.100.43]
- .vscode
  - launch.json
- bin
- lib
- project1
  - myapp
    - __pycache__
    - migrations
    - __init__.py
    - admin.py
    - apps.py
    - models.py
    - tests.py
    - views.py
  - project1
  - static
  - templates
  - db.sqlite3
  - manage.py
- .gitignore
- pyvenv.cfg

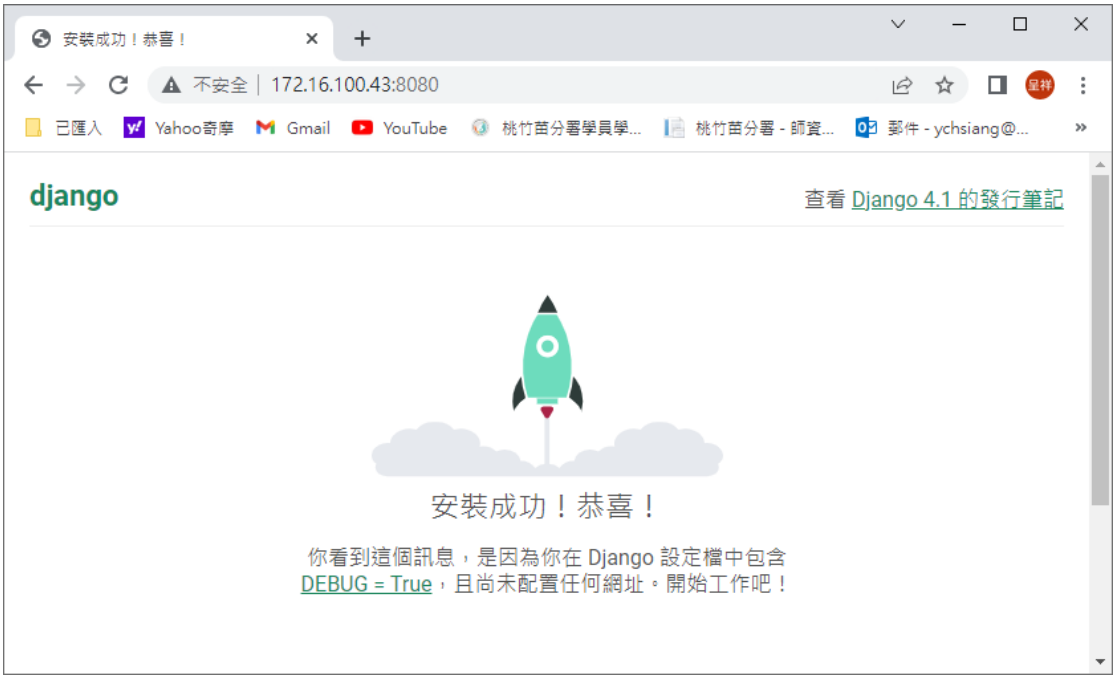
views.py ×

project1 > myapp > views.py

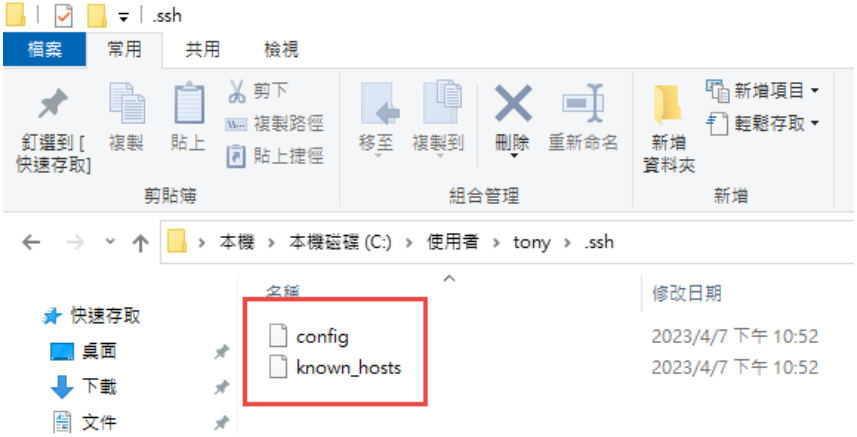
```
1  from django.shortcuts import render
2
3
4  # Create your views here.
5
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2 Python Debug Console

```
(dvds) pi@raspberrypi:~/dvds $ cd /home/pi/dvds ; /usr/bin/env /home/pi/dvds/bin/python /home/pi/.vscode-server/extensions/ms-python.python-
2023.2.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 37701 -- /home/pi/dvds/project1/manage.py runserver 0.0.0.0:8080
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
February 13, 2023 - 14:40:13
Django version 4.1.6, using settings 'project1.settings'
Starting development server at http://0.0.0.0:8080/
Quit the server with CONTROL-C.
```



注意:若連線有問題，請將 windows 系統中，c:\使用者\user\.ssh 內的資料刪除



3-2 建立 Django 專案

- 建立 Django 專案:

```
# cd ~/dvds

# source ./dvds/dvdsenv/bin/activate (啟動虛擬環境)

# django-admin startproject project1 (建立 project1 專案)

# cd project1
```

```
(dvdsenv) pi@raspberrypi:~/dvds/project1 $ ls
db.sqlite3  manage.py  myapp  project1  static  templates
```

```
# tree
```

```
(dvdsenv) pi@raspberrypi:~/project1 $ tree
.
├── manage.py
└── project1
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py

1 directory, 5 files
```

檔案或目錄	說明
manage.py	Python 命令檔，提供專案管理的功能，包含建立 app、啟動 Server 和 Shell 等
__init__.py	一個空檔，使得該目錄成為一個 Python package
settings.py	本專案的設定檔
urls.py	url 配置檔
wsgi.py	網頁伺服器和 Django 的介面設定檔 (托管)

3-3 建立 Application 應用程式

Application 應用程式相當於 Project 專案的元件，簡稱為 app, 而且是可以當作其他專案的模組，每個 Project 專案可以建立一個或多個 Application 應用程式。

- 建立應用程式

```
# sudo python3 manage.py startapp myapp (pyhton3)

# sudo python manage.py startapp myapp (pyhton2)
```

- 建立 templates 資料夾與 static 資料夾

Diango 是使用 MTV 的架構，將顯示的模版(.html)，放置 templates 資料夾中，其中資料夾須在專案的最上層目錄下建立。將使用的圖形、CSS、JavaScript 檔案，常以本機的方式儲存在 static 資料夾中，其中資料夾須在專案的最上層目錄下建立。

```
# mkdir templates static
```

```
(dvdsenv) pi@raspberrypi:~/dvds/project1 $ ls
manage.py  myapp  project1  static  templates
```

```
# tree
```

```
(dvdsenv) pi@raspberrypi:~/dvds/project1 $ tree
.
├── db.sqlite3
├── manage.py
├── myapp
?   ├── admin.py
?   ├── apps.py
?   ├── __init__.py
?   ├── migrations
?   ?   ├── __init__.py
?   ?   └── __pycache__
?   ?       └── __init__.cpython-35.pyc
?   ├── models.py
?   ├── __pycache__
?   ?   ├── admin.cpython-35.pyc
?   ?   ├── __init__.cpython-35.pyc
?   ?   └── models.cpython-35.pyc
?   ├── tests.py
?   └── views.py
├── static
└── templates
```

- 除錯模式設定與權限瀏覽

```
# vim project1/settings.py
```

其中，「*」是指全部 ip

```
23 SECRET_KEY = 'z7bj0fgrhh)3s5$n42j^-hl9_kt+(9+j2@u+tl*h7(^r=_4j)n'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = ['*']
29
30
```

add the original IP and/or hostname also:

```
ALLOWED_HOSTS = [
  'localhost',
  '127.0.0.1',
  '111.222.333.444',
  'mywebsite.com']
```

- 加入 Application 應用程式:

```
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'myapp',
41 ]
```

- 設定 Template 路徑:

Django 是使用 MTV 架構，將顯示的模版放置在 templates 資料夾中，因此要在 TEMPATES 的 DIRS 鍵中設定其路徑。

輸入前新增「import os」

```
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates'
   ,
58         'DIRS': [os.path.join(BASE_DIR,'templates')],
59         'APP_DIRS': True,
60         'OPTIONS': {
61             'context_processors': [
62                 'django.template.context_processors.debug',
63                 'django.template.context_processors.request',
64                 'django.contrib.auth.context_processors.auth',
65                 'django.contrib.messages.context_processors.messages
   ',
66             ],
67         },
68     },
69 ]
```

- 設定語系與時區：

sudo vim project1/settings.py

```
107  LANGUAGE_CODE = 'zh-Hant'
108
109  TIME_ZONE = 'Asia/Taipei'
110
111  USE_I18N = True
112
113  USE_L10N = True
114
115  USE_TZ = True
```

- 設定 static 靜態檔的路徑：

sudo vim project1/settings.py

STATIC_URL = 'static/'

STATICFILES_DIRS = [

BASE_DIR / 'static',

]

```
121  STATIC_URL = 'static/'
122  STATICFILES_DIRS = [
123      BASE_DIR / 'static',
124  ]
```

- 建立 migration 資料檔：

Diango 要使用資料庫，通常會將建立資料表的架構和版本記錄下來，以利以後的追蹤。

sudo python3 ./manage.py makemigrations (python3)

sudo python ./manage.py makemigrations (python2)

```
(dvdsenv) pi@raspberrypi:~/dvds2/project1 $ sudo ./manage.py makemigrations
No changes detected
```

- 模型與資料庫同步

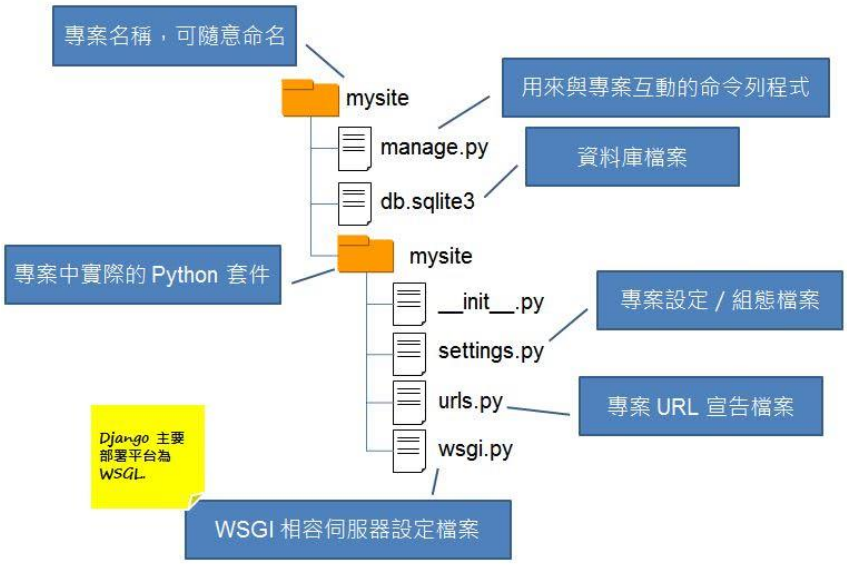
sudo python3 ./manage.py migrate (python3)

sudo python ./manage.py migrate (python2)

```
(dvdsenv1) pi@raspberrypi:~/project1 $ sudo ./manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying sessions.0001_initial... OK
```

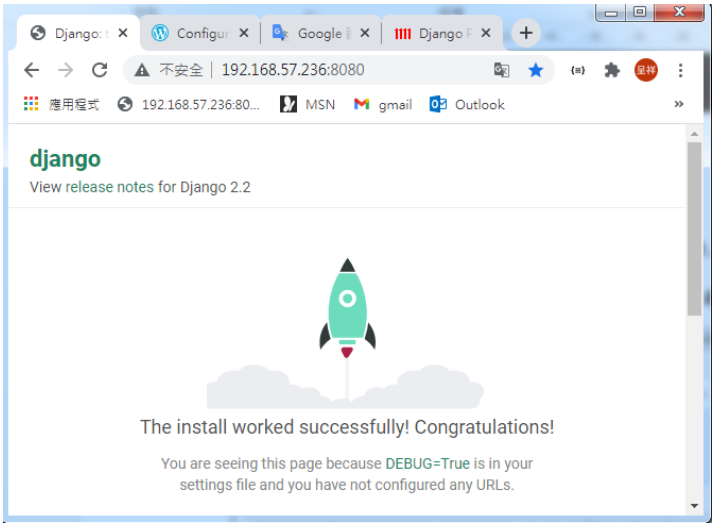
tree

```
(dvdsenv1) pi@raspberrypi:~/project1 $ tree
.
├── db.sqlite3
├── manage.py
└── project1
    ├── __init__.py
    ├── __init__.pyc
    ├── settings.py
    ├── settings.pyc
    ├── urls.py
    ├── urls.pyc
    └── wsgi.py

1 directory, 9 files
```



- 啟動 Server

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

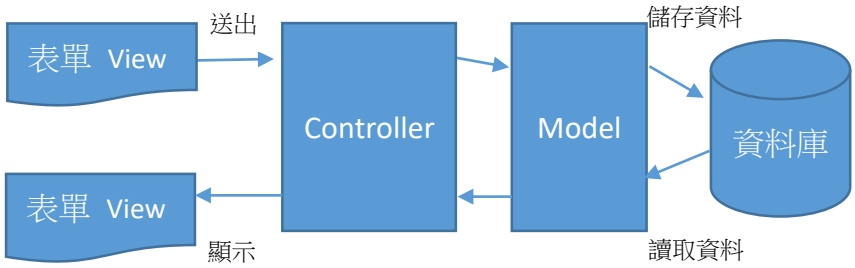


3-4 視圖(view)與 url

➢ Diango 的 Framework 架構

MVC(Model-view-controller):

基本部分：模型（Model）、視圖（View）和控制器（Controller）。



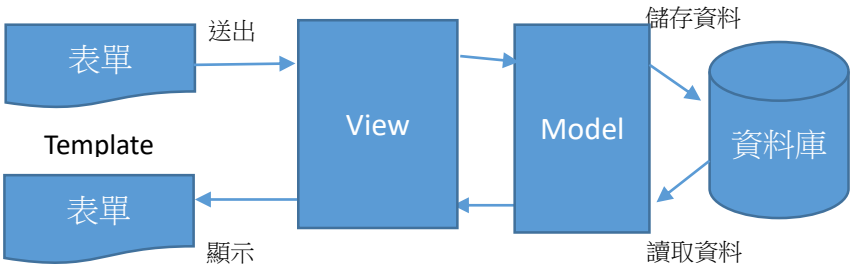
模型（Model）：代表商業邏輯與資料庫存取有關的程式。

視圖（View）：代表輸入和輸出畫面的呈現。

控制器（Controller）：介於 Model 及 View 之間，判斷該呼叫哪一個 Model 存取資料庫並透過。

MVT(Model-view-template):

基本部分：模型（Model）、視圖（View）和模版（Template）。



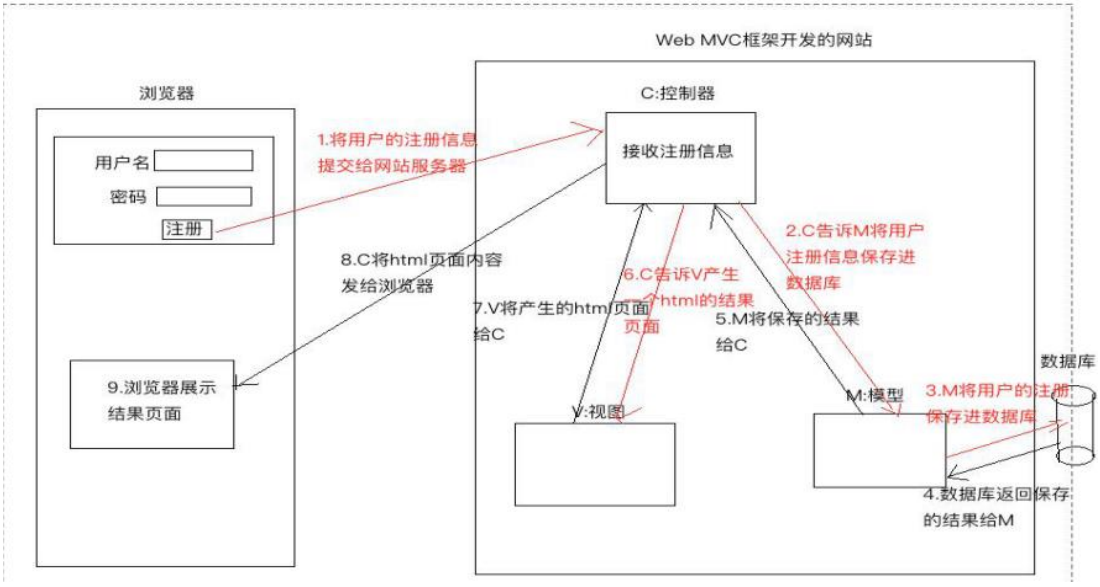
模型（Model）：代表商業邏輯與資料庫存取有關的程式。

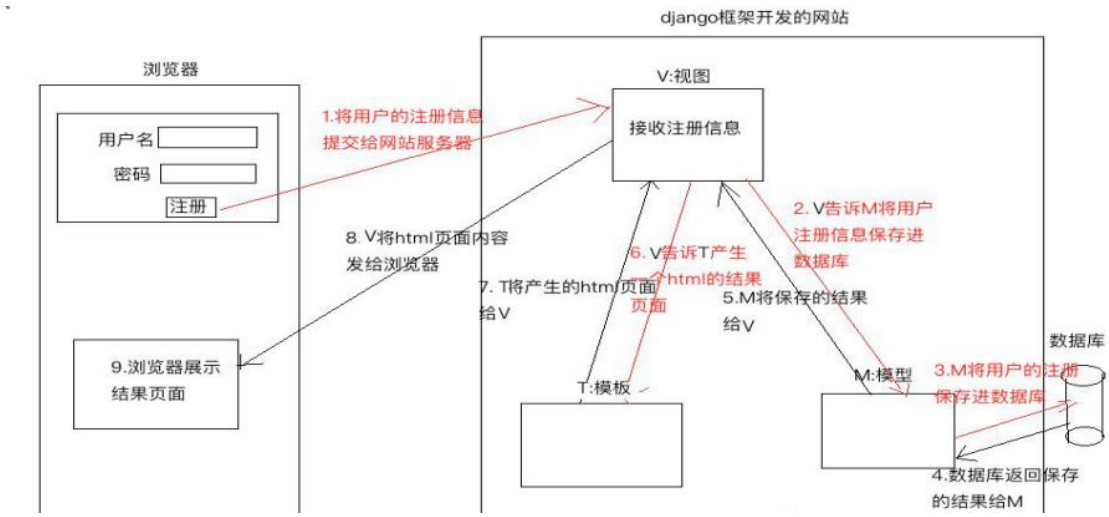
視圖（View）：處理 Model 存取資料庫和 Template 介面資料的輸入或顯示。

模版（Template）：代表介面輸入表單或顯示資料。

MVC VS MVT

任務	MVC 架構	MTV
資料庫存取	M=Model=模型	M=Model=模型
輸入表單或顯示資料	V=View=視圖	T=Template=模版
控制與整合	C=Controller=控制	V=View=視圖





参考文獻:

https://blog.csdn.net/u014745194/article/details/73718041

➢ 設定 urls.py

Django 的程式架構是採用 urlpattern 網址和函式對照方式，主要有兩個步驟:
1、設定<url.py>檔 urlpatterns 串列中 url 網址和函式的對照。
2、在<views.py>中撰寫函式。

url 中可以傳遞任意數量的參數給 view 中對應的函式，參數型別為字典，且前後必須加上 <> 字元，傳遞的參數會自動作型別轉換。
傳遞參數型別：

- str：字串，匹配任何非空字符串，但不含「/」字元，這是預設值。
- int：匹配 0 及正整數，傳回一個 int 型別。
- slug：匹配字母、數字以及 /、 組成的字串，是 url 在最後一部分的註釋文字。
- uuid：匹配格式化的 uuid，為了防止衝突，規定必須使用「-」字元，所有字母必須小寫。如 075194d3-6885-417e-a8a8-6c931e272f00。它會傳回一個 UUID 物件。
- path：匹配任何非空字符串，包含路徑分隔符號「/」

參考文獻:
http://blog.e-happy.com.tw/django2-0-%E4%BB%A5-path-%E5%87%BD%E5%BC%8F%E8%A8%AD%E5%AE%9A-urlpatterns/

```
# sudo vim project1/urls.py
```

```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('', views.sayhello),
23
```

其中表示瀏覽「127.0.0.1:8000/admin/」這個網址，將會執行 admin.site.urls 函式，因為 admin.site.urls 函式在 django.contrib 模組中的 admin 中，因此必須 import 進來。

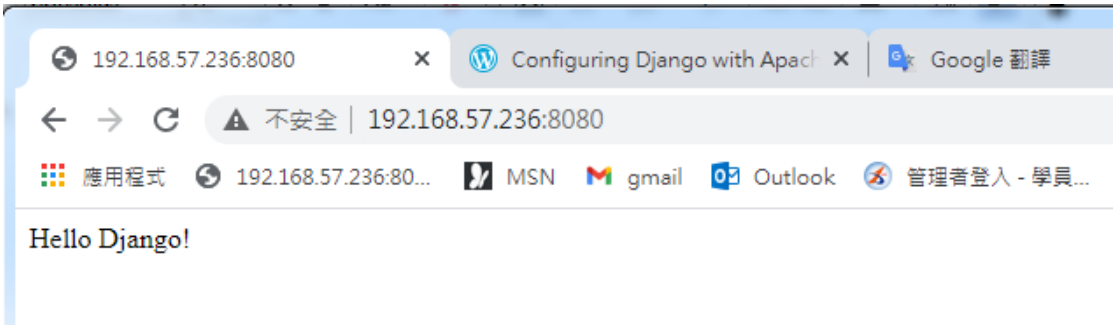
url 語法: path(網址,函式)

- 定義函式：

sudo vim myapp/views.py

```
from django.shortcuts import render
from django.http import HttpResponse
# Create your views here.
def sayhello(request):
    return HttpResponse("Hello Django!")
```

sudo python3 manage.py runserver 0.0.0.0:8080



ex:

path(‘hello/<str:name>/’,hello)

即瀏覽「127.0.0.1/hello/david/」這個網址，就會執行 hello 函式，並且傳送「david」參數值給 hello 函式。

sudo vim project1/urls.py

```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('', views.sayhello),
23      path('hello1/<str:username>', views.hello1),
```

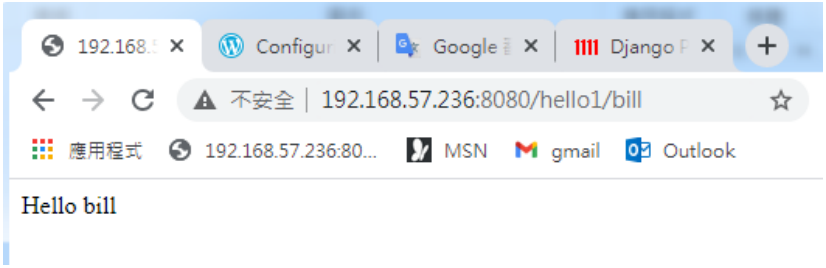
定義函式：

sudo vim myapp/views.py

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

from django.shortcuts import render
from django.http import HttpResponse

# Create your views here.
def sayhello(request):
    return HttpResponse("Hello Django!")

def hello1(request,username):
    return HttpResponse("Hello "+ username)
```

測試結果：

sudo python3 manage.py runserver 0.0.0.0:8080



- 模版的使用

vim templates/hello2.html

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9       <h1>歡迎光臨:{{ username }}</h1>
10      <h2>現在時刻:{{ now }}</h2>
11  </body>
12  </html>
```

sudo vim project1/urls.py

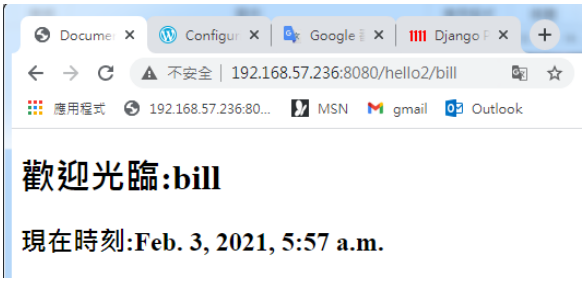
```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('', views.sayhello),
23      path('hello1/<str:username>', views.hello1),
24      path('hello2/<str:username>', views.hello2),
```

sudo vim myapp/views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from datetime import datetime
# Create your views here.
def sayhello(request):
    return HttpResponse("Hello Django!")

def hello1(request,username):
    return HttpResponse("Hello "+ username)

def hello2(request,username):
    now=datetime.now()
    return render(request,"hello2.html",locals())
```

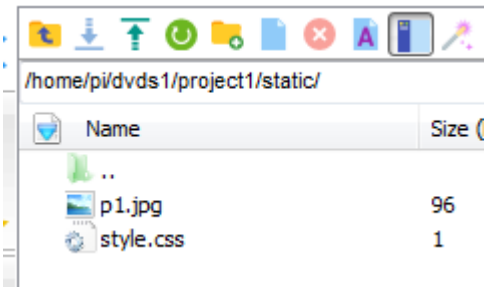
測試結果：

sudo python3 manage.py runserver 0.0.0.0:8080



➢ 加入 static 靜態檔案

在 static 加入檔案



vim project1/settings.py

加入

```
121 STATIC_URL = '/static/'
122 STATICFILES_DIRS = [
123     os.path.join(BASE_DIR, 'static'),
124 ]
125
```

sudo vim project1/urls.py

```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('', views.sayhello),
23      path('hello1/<str:username>', views.hello1),
24      path('hello2/<str:username>', views.hello2),
25      path('hello3/<str:username>', views.hello3),
26
```

sudo vim myapp/views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from datetime import datetime
# Create your views here.
def sayhello(request):
    return HttpResponse("Hello Django!")

def hello1(request,username):
    return HttpResponse("Hello "+ username)

def hello2(request,username):
    now=datetime.now()
    return render(request,"hello2.html",locals())

def hello3(request,username):
    now=datetime.now()
    return render(request,"hello3.html",locals())
```

vim templates/hello3.html

hello3.html ●

C: > Users > tony > DOCUME~1 > MobaXterm > slash > RemoteFiles > 591376_2_5 > hello3.html > ...

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <meta charset="utf-8">
5   <title>第一個模版</title>
6   {% load staticfiles %}
7   <link href="{% static './style.css' %}" rel="stylesheet" type="text/css" />
8
9   </head>
10  <body>
11
12  <h1>Flask網站
13      <img src="{% static './p1.jpg' %}" width="100px"></img>
14  </h1>
15  <h2>{{name}}，歡迎光臨!!</h2>
16  <h4>現在時間:{{now}}</h4>
17
18  </body>
19  </html>
```

注意:必須在.html 檔中以檔案中，以{% load staticfiles %}宣告使用靜態檔案，同時以{% static 靜態檔案 %}格式設定靜態檔案的路徑。

注意:若是 windows 請將{% load staticfiles %}改成{% load static %}

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

➢ 從網址中載取資料

ex:

path(‘hello/<str:name1>/<str:name2>/’, function)
path(‘hello/<str:name1>/hello/’, function)

```
# sudo vim project1/urls.py
```

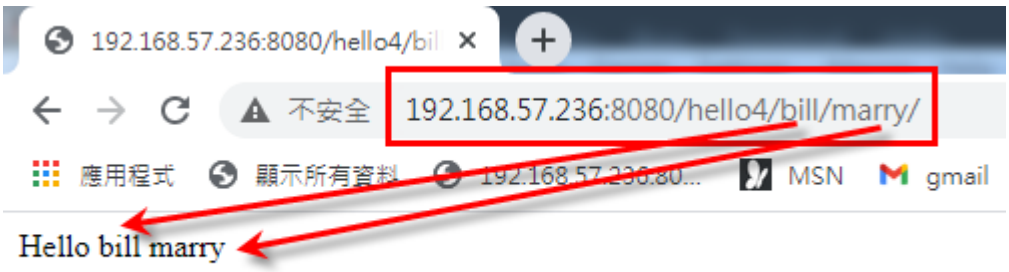
```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('', views.sayhello),
23      path('hello1/<str:username>', views.hello1),
24      path('hello2/<str:username>', views.hello2),
25      path('hello3/<str:username>', views.hello3),
26      path('hello4/<str:username1>/<str:username2>/',views.hello4),
27      path('hello5/<str:username>/hello/',views.hello5),
28
```

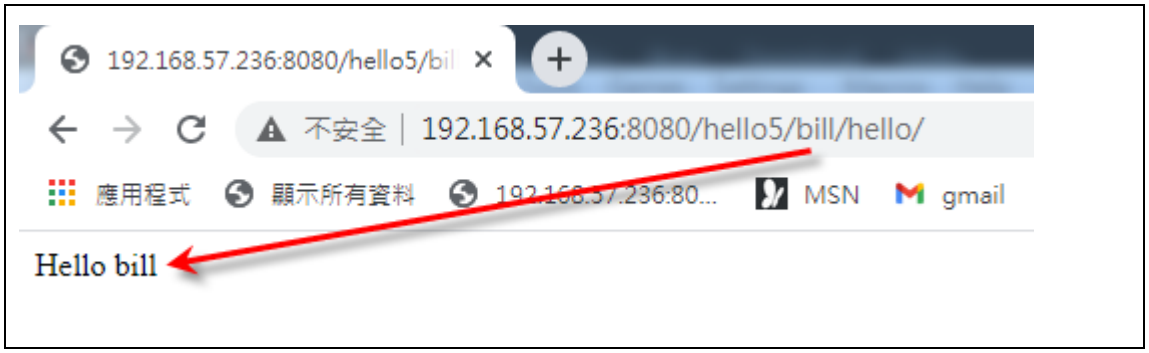
```
# sudo vim myapp/views.py
```

```
22  def hello4(request,username1,username2):
23          return HttpResponse("Hello " + username1 + " " + username2)
24
25  def hello5(request,username):
26          return HttpResponse("Hello " + username)
27
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```




192.168.57.236:8080/hello5/bil
不安全 | 192.168.57.236:8080/hello5/bill/hello/
應用程式
顯示所有資料
192.168.57.236:80...
MSN
gmail
Hello bill

3-5 視圖、模版與 Template 語言

- 傳遞變數到 Template 模板檔案

使用 render 函式，必須 import 該模組，系統預設已經以 from django.shortcuts import render 加入模組，因此可以直接使用 render 函式。

render 的語法如下：

render(request, template_name, locals())

第一個參數:HttpRequest 物件

第二個參數:模板名稱

第三個參數:傳遞所有區域變數給顯示的模版

no=1

dict1={“name”:”Amy”, ”age”:20}

return render(request, “dice.html”, locals())

or

return render(request, “dice.html”, {“no”:1, “dict1”: {“name”:”Amy”, ”age”:20} })

經過 locals()函式轉換後會建立{“no”:1, “dict1”: {“name”:”Amy”, ”age”:20} }

- 模版顯示變數

由於 locals()將所有區域變數轉換成字典，在模版中要顯示 locals()字典的內容，就可以讀取字典的語法來讀取它。

no=1 已由 locals()轉換為{“no”:1}

{{ no }}

若是字典的變數，語法如下:

{{ 字典變數.鍵 }}

{{ dict1.name }}

```
# sudo vim project1/urls.py
```

```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('', views.sayhello),
23      path('hello1/<str:username>', views.hello1),
24      path('hello2/<str:username>', views.hello2),
25      path('hello3/<str:username>', views.hello3),
26      path('hello4/<str:username1>/<str:username2>/',views.hello4),
27      path('hello5/<str:username>/hello/',views.hello5),
28  ##########################################################################
29      ath('dice1/', views.dice1),
```

sudo vim myapp/views.py

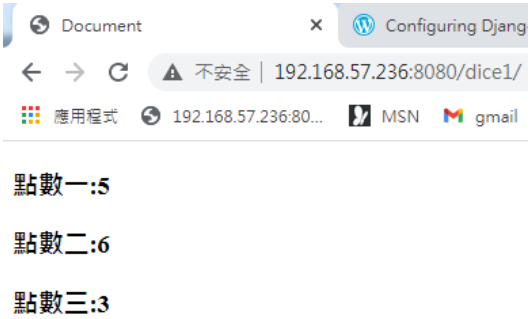
```
from django.http import HttpResponse
from datetime import datetime
import random

# Create your views here.
def sayhello(request):
    return HttpResponse("Hello Django!")

def hello1(request,username):
    return HttpResponse("Hello "+ username)

def hello2(request,username):
    now=datetime.now()
    return render(request,"hello2.html",locals())

def hello3(request,username):
    now=datetime.now()
    return render(request,"hello3.html",locals())

def dice1(request):
    no1=random.randint(1,6)
    no2=random.randint(1,6)
    no3=random.randint(1,6)
    return render(request,"dice1.html",locals())
```

vim templates/dice1.html

```
dice1.html ●
C: > Users > tony > DOCUME~1 > MobaXterm > slash > RemoteFiles > 591376_2_7 > dice1.html > ...
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9       <h3>點數一:{{ no1 }}</h3>
10      <h3>點數二:{{ no2 }}</h3>
11      <h3>點數三:{{ no3 }}</h3>
12  </body>
13  </html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```



3-6 模板(Template)語言-變量

Template 模板有自己的語言，可以顯示變數，同時也有 if 條件指令和 for 迴圈指令，也可以加上註解。

名稱	說明	範例
變量	將 views 傳送內容顯示在模版指定的位置上。	{{ username }}
標籤	If 條件指令和 for 迴圈指令。	{% if found %} {% for item in items %}
多行註解	語法: {% comment %} 註解文字一 註解文字二 {% endcomment %}	語法: {% comment %} 註解文字一 註解文字二 {% endcomment %}
單行註解	語法:{# 註解文字 #}	{# 這是註解文字#}
文字	HTML 標籤或文字	<title>顯示的模板</title>

➢ 變量

變量就是要顯示的變數，變數可是一般的變數，也可以使用字典或串列，分別以「{{ 變數 }}」、「{{ 變數變數.鍵 }}」、「{{ 串列.索引 }}」語法來表示。

變量類型	Template 語法	Python 語法	說明
字典	{{dict1.name}}	dict1[name]	name 是字典的鍵
方法	{{obj1.show}}	obj1.show	show()是 obj1 物件方法
串列	{{list1.0}}	list1[0]	list1 是串列，例如： list1=[“a”,”b”,”c”]。

sudo vim project1/urls.py

```
from django.contrib import admin
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.sayhello),
    path('hello1/<str:username>', views.hello1),
    path('hello2/<str:username>', views.hello2),
    path('hello3/<str:username>', views.hello3),
    path('hello4/<str:username1>/<str:username2>/',views.hello4),
    path('hello5/<str:username>/hello/',views.hello5),
##########################################################################
    path('dice1/', views.dice1),
    path('dice2/', views.dice2),
```

sudo vim myapp/views.py

```
def dice2(request):
    student = {'id':'12345','name':'張三','sex':'男'}
    fruit = ['apple','banana','Guava']
    return render(request, 'dice2.html',locals())
```

vim templates/dice2.html

dice2.html

C: > Users > tony > DOCUME~1 > MobaXterm > slash > RemoteFiles

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>第一個模版</title>
</head>
<body>
    <h4>姓名:{{ student.name }}</h4>
    <h4>性別:{{ student.sex }}</h4>
    <h4>最喜歡的水果:{{ fruit.2 }}</h4>
</body>
</html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

3-7 模板(Template)語言-標籤

- Operators

Operators

There are some built-in operators you can use when evaluating `if` statements:

Variable	Description	
==	is equal to	Example »
!=	is not equal to	Example »
<	is less than	Example »
<=	is less than, or equal to	Example »
>	is greater than	Example »
>=	is greater than, or equal to	Example »
and	condition1 *and* condition2 must be true	Example »
or	condition1 *or* condition2 must be true	Example »
in	an item must be present in an object	Example »
is	is the same value as	Example »
is not	is not the same value as	Example »
not in	is not in	Example »

Reference:
https://www.w3schools.com/django/ref_tags_if.php

- 條件指令

```
{%  if 條件  %}

   程式區塊
```

```
{%  endif  %}


EX:
{%  if score >=60  %}
   及格
{%  endif  %}
```

```
{%  if 條件  %}
   程式區塊一
{%  else  %}
   程式區塊二
{%  endif  %}


EX:
{%  if score>=60  %}
   及格
{%  else  %}
   不及格
{%  endif  %}
```

```
{%  if 條件一  %}
   程式區塊一
{%  elif 條件二  %}
   程式區塊二
```

```
{%  elif 條件三  %}
   程式區塊三
{%  else  %}
   程式區塊四
{%  endif  %}


EX:
{%  if score>= 90  %}
   優等
{%  elseif score>= 80  %}
   甲等
{%  elseif score>= 70  %}
   乙等
{%  else  %}
   丙等
{%  endif  %}
```

and

To check if more than one condition is true.

Example

```
{% if greeting == 1 and day == "Friday" %}
  <h1>Hello Weekend!</h1>
{% endif %}
```

or

To check if one of the conditions is true.

Example

```
{% if greeting == 1 or greeting == 5 %}
  <h1>Hello</h1>
{% endif %}
```

Rerence:
https://www.w3schools.com/django/django_tags_if.php

- for 迴圈指令

```
{% for  變數  in  串列  %}
程式區塊
{%  endfor  %}

EX:
list1 = range(1,6)
{% for  i  in  list1  %}
{{ i }}
{%  endfor  %}
```

forloop 屬性	說明
forloop.counter	由 1 開始遞增到疊代總數。
forloop.counter0	由 0 開始遞增到疊代總數。
forloop.revcounter	由串列元素總教開始遞減到 1。
forloop.revcounter0	由串列元素總教開始遞減到 0。
forloop.first	判斷是否第一次 for 迴圈，其值為 True 或 False

forloop.last	判斷是否最後一次 for 迴圈，其值為 True 或 False
forloop.parentloop	父迴圈(上一層迴圈)的 forloop

sudo vim project1/urls.py

```
from django.contrib import admin
from django.urls import path
from myapp.views import sayhello, hello1, hello2, hello3, dice1, dice2, dice3

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', sayhello),
    path('hello1/<str:username>', hello1),
    path('hello2/<str:username>', hello2),
    path('hello3/<str:username>', hello3),
    path('dice1/', dice1),
    path('dice2/', dice2),
    path('dice3/', dice3),
]
```

sudo vim myapp/views.py

```
def dice3(request):
    person1={"name":"Amy","phone":"049-1234567","age":20}
    person2={"name":"Jack","phone":"02-4455666","age":25}
    person3={"name":"Nacy","phone":"04-9876543","age":17}
    persons=[person1,person2,person3]
    return render(request,"dice3.html",locals())
```

vim templates/dice3.html

```
dice3.html ●
C: > Users > tony > DOCUME~1 > MobaXterm > slash > RemoteFiles > 591376_2_9 >
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <meta charset='utf-8'>
5       <title>基本資料</title>
6   </head>
7   <body>
8       <h3>
9       {% for person in persons%}
10          <h2>第 {{forloop.counter}} 位員工</h2>
11          <ul>
12              <li>姓名：{{person.name}}</li>
13              <li>手機：{{person.phone}}</li>
14              <li>年齡：{{person.age}}</li>
15          </ul>
16      {% empty %}
17          沒有任何資料
18      {% endfor %}
19      </h3>
20  </body>
21  </html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

Reference:

https://www.w3schools.com/django/django_tags_for.php

3-8 以 GET 及 POST 傳送資料

網頁最常用來傳送資料的方式就是 GET 及 POST。HTTP Method：表單中的 GET 與 POST 有什麼差別？先舉個例子，如果 HTTP 代表現在我們現實生活中寄信的機制，那麼信封的撰寫格式就是 HTTP。我們姑且將信封外的內容稱為 http-header，信封內的書信稱為 message-body，那麼 HTTP Method 就是你要告訴郵差的寄信規則。

假設 GET 表示信封內不得裝信件的寄送方式，如同是明信片一樣，你可以把要傳遞的資訊寫在信封(http-header)上，寫滿為止，價格比較便宜，但看的到不安全。

然而 POST 就是信封內有裝信件的寄送方式（信封有內容物），不但信封可以寫東西，信封內 (message-body) 還可以置入你想要寄送的資料或檔案，價格較貴，包在裡面安全看不到。

使用 GET 的時候我們直接將要傳送的資料以 Query String（一種 Key/Vaule 的編碼方式）加在我們要寄送的地址(URL)後面，然後交給郵差傳送。使用 POST 的時候則是將寄送地址(URL)寫在信封上，另外將要傳送的資料寫在另一張信紙後，將信紙放到信封裡面，交給郵差傳送(message-body)。

接著我來介紹一下實際的運作情況：

我們先來看看 GET 怎麼傳送資料的，當我們送出一個 GET 表單時，如下範例：

```
<form method="get" action="">
<input type="text" name="id" />
<input type="submit" />
</form>
```

當表單 Submit 之後瀏覽器的網址就變成 "http://xxx.toright.com/?id=010101"，瀏覽器會自動將表單內容轉為 Query String 加在 URL 進行連線。

當使用 GET 的方法時，會將表單資訊附加在 URL 上並作為 QueryString 的一部分，QueryString 是一種 key/value 的組合，從問號「?」開始，每一組值都是用「&」隔開，如下圖

```
GET /getCustomer.aspx?Id=123&name=marcus HTTP/1.1
Host: www.testwebsite.com
```

這時後來看一下 HTTP Request 封包的內容：

```
GET /?id=010101 HTTP/1.1
Host: xxx.toright.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-TW; rv:1.9.2.13) Gecko
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-tw,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: UTF-8,*
Keep-Alive: 115
Connection: keep-alive
```

在 HTTP GET Method 中是不允許在 message-body 中傳遞資料的，因為是 GET 就是要取資料的意思。從瀏覽器的網址列就可以看見我們表單要傳送的資料，若是要傳送密碼豈不是"一覽無遺".....這就是大家常提到安全性問題。

再來看看 POST 傳送資料

```
<form method="post" action="">
<input type="text" name="id" />
<input type="submit" />
</form>
```

網址列沒有變化，那我們來看一下 HTTP Request 封包的內容：

```
POST / HTTP/1.1
Host: xxx.toright.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-TW; rv:1.9.2.13) Gecko
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-tw,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: UTF-8,*
Keep-Alive: 115
Connection: keep-alive

Content-Type: application/x-www-form-urlencoded
</code><code>Content-Length: 9
id=020202
```

看出個所以然了嗎？原來 POST 是將表單資料放在 message-body 進行傳送，在不偷看封包的情況下似乎安全一些些。此外在傳送檔案的時候會使用到 multi-part 編碼，將檔案與其他的表單欄位一併放在 message-body 中進行傳送。這就是 GET 與 POST 發送表單的差異。

	GET	POST
網址差異	網址會帶有 HTML Form 表單的參數與資料。	資料傳遞時，網址並不會改變。
資料傳遞量	由於是透過 URL 帶資料，所以有長度限制。	由於不透過 URL 帶參數，所以不受限於 URL 長度限制。
安全性	表單參數與填寫內容可在 URL 看到。	透過 HTTP Request 方式，故參數與填寫內容不會顯示於 URL。

- 以 GET 傳送資料

以 GET 傳送參數值的語法:

網路?參數 1=值 1 & 參數 2=值 2&參數 3=值 3

Ex:

http://127.0.0.1:5000/test?name=bill&tel=092322423

Django 使用 request 模組的 GET 方法即可取得 GET 參數值，語法為：

request.GET[‘參數名稱’]

Django 使用 request 模組的 POST 方法即可取得 POST 參數值，語法為：

request.POST[‘參數名稱’]

- 使用 GET 傳遞參數(有參數)

```
# sudo vim project1/urls.py
```

urls.py × views.py

project1 > urls.py > ...

```
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.sayhello),
    path('hello1/<str:username>',views.hello1),
    path('hello2/<str:username>',views.hello2),
    path('hello3/<str:username>',views.hello3),
    path('hello4/<str:username1>/<str:username2>',views.hello4),
    path('hello5/<str:username1>/hello',views.hello5),
    path('dice1',views.dice1),
    path('dice2',views.dice2),
    path('dice3',views.dice3),
    path('get1',views.get1),
```

```
# sudo vim myapp/views.py
```

urls.py　views.py　get1.html

myapp > views.py > getpost2

```
def get1(request):
    name = request.GET['name']
    city = request.GET['city']
    # print(name)
    # print(city)
    # return HttpResponse("test")
    return render(request, "get1.html",locals())
```

```
# sudo vim templates/get1.html
```

get1.html　urls.py　views.py

templates > get1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Django網站</h1>
    <h2>歡迎來自{{ city }}的{{ name }}光臨本網站</h2>
</body>
</html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8000
```

Document

127.0.0.1:8000/get1?name=bill&city=台北

應用程式　桃竹苗分署學員學...　桃竹苗分署 - 師資...　show django

Django網站

歡迎來自台北的bill光臨本網站

- 使用 GET 傳遞參數(有參數與無參數)

```
# sudo vim project1/urls.py
```

urls.py × views.py

project1 > urls.py > ...

```python
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.sayhello),
    path('hello1/<str:username>',views.hello1),
    path('hello2/<str:username>',views.hello2),
    path('hello3/<str:username>',views.hello3),
    path('hello4/<str:username1>/<str:username2>',views.hello4),
    path('hello5/<str:username1>/hello',views.hello5),
    path('dice1',views.dice1),
    path('dice2',views.dice2),
    path('dice3',views.dice3),
    path('get1',views.get1),
    path('get2',views.get2),
```

sudo vim myapp/views.py

urls.py views.py × get2.html

myapp > views.py > ...

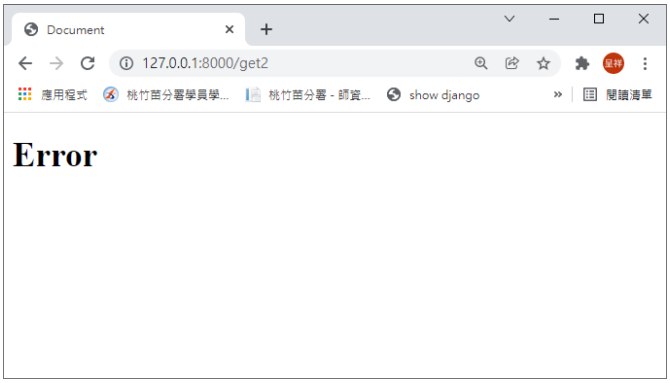
```python

def get2(request, name=None, city=None):
    status=True
    if 'name' in request.GET:
        name = request.GET['name']
    else:
        status=False

    if 'city' in request.GET:
        city = request.GET['city']
    else:
        status=False

    return render(request, "get2.html",locals())
```

sudo vim templates/get2.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    {% if status%}
        <h1>Django網站</h1>
        <h2>歡迎來自{{ city }}的{{ name }}光臨本網站</h2>
    {% else %}
        <h1>Error</h1>
    {% endif %}
</body>
</html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8000
```



- 使用 GET 傳遞參數(使用表單)

```
# sudo vim project1/urls.py
```

urls.py × views.py get3_response.html

project1 > urls.py > ...

```
30      path('dice3/',views.dice3),
31      path('get1/',views.get1),
32      path('get2/',views.get2),
33      path('get3/<str:mode>/',views.get3),
```

```
# sudo vim myapp/views.py
```

urls.py views.py × get3_response.html get3.html

myapp > views.py > ...

```
83  def get3(request, mode=None):
84      if mode == "save":
85          username = request.GET['username']
86          passwd = request.GET['passwd']
87          # return HttpResponse("test1")
88          return render(request, "get3_response.html",locals())
89      elif mode=="load":
90          # return HttpResponse("test2")
91          return render(request, "get3.html",locals())
```

```
# sudo vim templates/get3.html
```

urls.py views.py get3.html × get3_response.html

templates > get3.html > html

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Document</title>
8   </head>
9   <body>
10      <h1>Django網站</h1>
11      <form action="/get3/save/" method="get">
12          <p>帳號: <input type="text" name="username" id="username"></p>
13          <p>密碼: <input type="password" name="passwd" id="passwd"></p>
14          <p><button type="submit">確定</button> <button type="reset">清除</button></p>
15      </form>
16  </body>
17  </html>
```

```
# sudo vim templates/get3_response.html
```

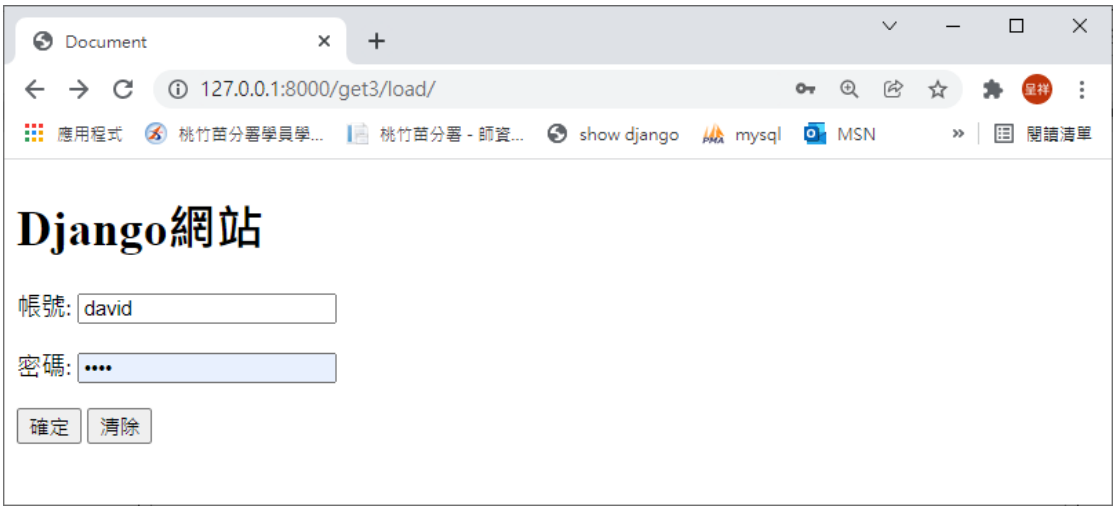
get3_response.html × urls.py views.py get3.html

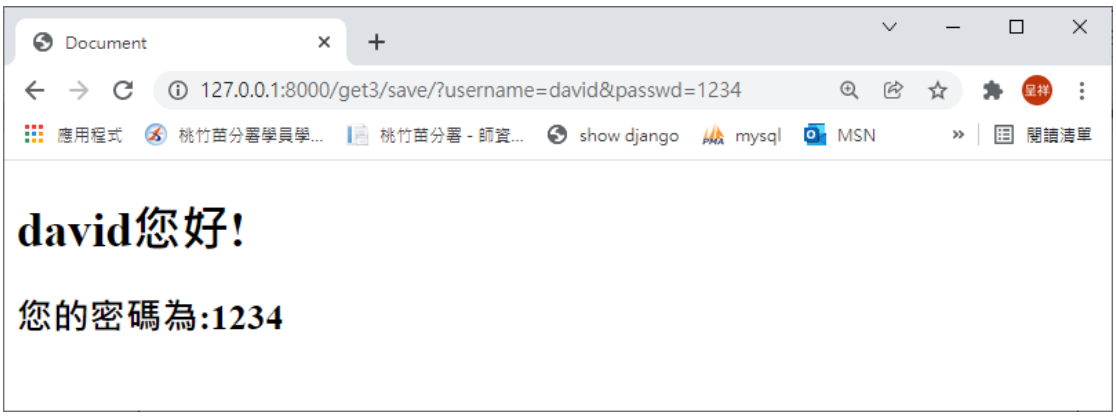
templates > get3_response.html > html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>{{username}}您好!</h1>
    <h2>您的密碼為:{{passwd}}</h2>
</body>
</html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8000
```





- 使用 POST 傳遞參數(使用表單，單選題)

```
# sudo vim project1/urls.py
```

urls.py × views.py

project1 > urls.py > ...

```
30      path('dice3/',views.dice3),
31      path('get1/',views.get1),
32      path('get2/',views.get2),
33      path('get3/<str:mode>/',views.get3),
34      path('post1',views.post1),
```

```
# sudo vim myapp/views.py
```

urls.py views.py × get3_response.html

myapp > views.py > getpost4

```
93  def post1(request):
94      if request.method == "POST":
95          username = request.POST['username']
96          passwd = request.POST['passwd']
97          # print(username)
98          # print(passwd)
99          # return HttpResponse("有資料")
100         if username=="david" and passwd=="1234":
101             status = True
102             # return HttpResponse("歡迎光臨本網站")
103         else:
104             status = False
105             # return HttpResponse("帳號或密碼錯誤!")
106         return render(request, "post1_response.html",locals())
107
108     else:
109         # return HttpResponse("無資料")
110         return render(request, "post1.html",locals())
```

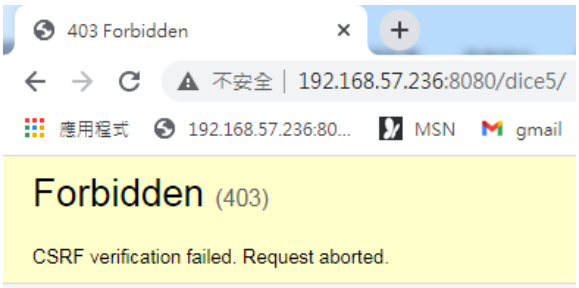
```
# sudo vim templates/post1.html
```

post1.html × urls.py views.py post1_response.html

templates > post1.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Django網站</h1>
    <form action="/post1/" method="post">
        {% csrf_token %}
        <p>帳號: <input type="text" name="username" id="username"></p>
        <p>密碼: <input type="password" name="passwd" id="passwd"></p>
        <p><button type="submit">確定</button> <button type="reset">清除</button></p>
    </form>
</body>
</html>
```

其中「{% csrf_token %}」是加入 CSRF 驗證，在 Django 中所有以 POST 方式送出資料的表單，都要加入「{% csrf_token %}」以增加安全性，如果沒有加入，在送出表單時會產生錯誤如下：




```
# sudo vim templates/post1_response.html
```

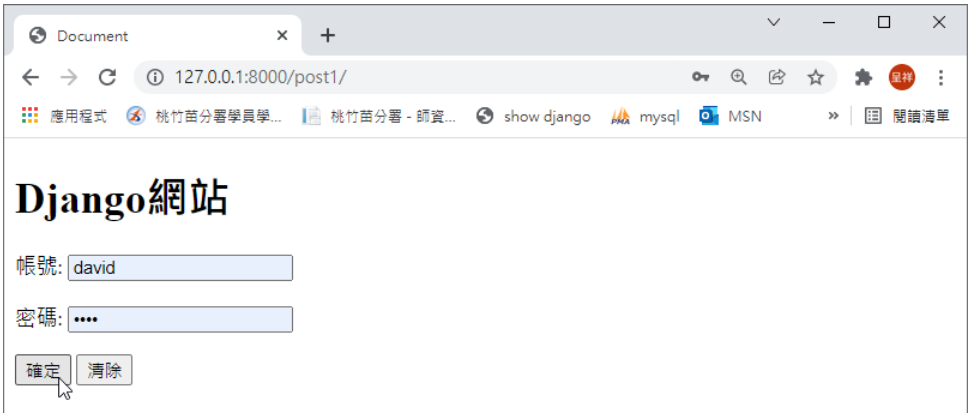
urls.py　views.py　post1_response.html

templates > post1_response.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    {% if status%}
    <h1>Django網站</h1>
    <h2>歡迎光臨{{ username }}</h2>
    {% else %}
    <h1>帳號或密碼錯誤!</h1>
    {% endif %}
</body>
</html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8000
```



- 使用 POST 傳遞參數(使用表單，複選題)

補充(表單使用 post 傳送，複選題):

```
sudo vim project1/urls.py
```

```
urls.py   views.py   post2.html
project1 > urls.py > ...
30      path('dice3/',views.dice3),
31      path('get1/',views.get1),
32      path('get2/',views.get2),
33      path('get3/<str:mode>/',views.get3),
34      path('post1/',views.post1),
35      path('post2/',views.post2),
36  ]
```

```
# sudo vim myapp/views.py
```

```
urls.py   views.py   post2.html   post2_response.html
myapp > views.py > ...
111
112 def post2(request):
113     if request.method == "POST":
114         items = request.POST.getlist('items')
115         # print(request.POST.getlist('items'))
116         # return HttpResponse("有資料")
117         return render(request, "post2_response.html",locals())
118     else:
119         # return HttpResponse("無資料")
120         return render(request, "post2.html",locals())
```

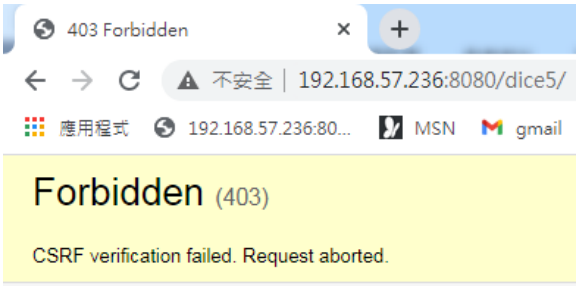
```
# sudo vim templates/post2.html
```

post2.html

templates > post2.html > html > body > form > div

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form method="POST" action="/post2/">
        {% csrf_token %}
        <div>您學會的項目為 :</div>
        <div>
            <input type="checkbox" id="Linux" name="items" value="Linux"/>
            <label for="Linux">Linux</label>
            <input type="checkbox" id="Apache" name="items" value="Apache"/>
            <label for="Apache">Apache</label>
            <input type="checkbox" id="PHP" name="items" value="PHP"/>
            <label for="Linux">PHP</label>
            <input type="checkbox" id="MySQL" name="items" value="MySQL"/>
            <label for="MySQL">MySQL</label>
        </div>
        <div>
            <input type="submit" value="送出" />
        </div>

</body>
</html>
```

其中「{% csrf_token %}」是加入 CSRF 驗證，在 Django 中所有以 POST 方式送出資料的表單，都要加入「{% csrf_token %}」以增加安全性，如果沒有加入，在送出表單時會產生錯誤如下：




```
# sudo vim templates/post2_response.html
```

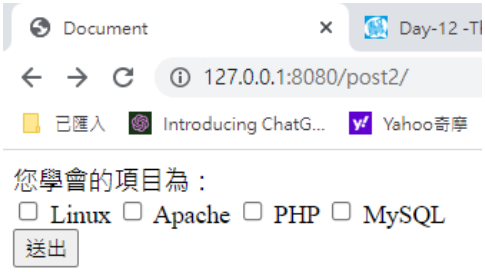
post2_response.html × urls.py views.py post2.html

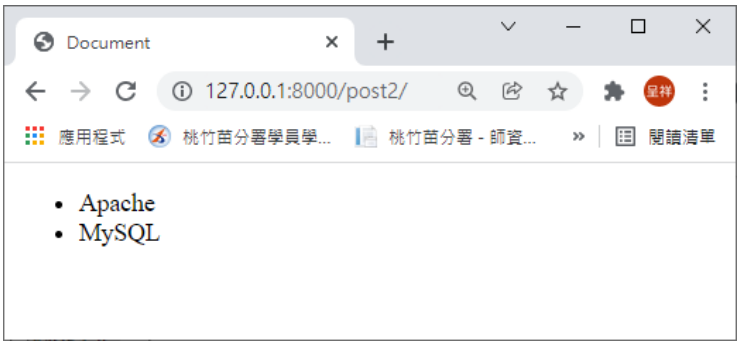
templates > post2_response.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <ul>
        {% for data in items %}
        <li>{{data}}</li>
        {% endfor %}
    </ul>
</body>
</html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8000
```





3-9 Django 資料庫連結與應用(使用 MariaDB)

➢ 安裝 MariaDB

```
# sudo  apt  update

# apt-cache  search  mysql-server
```

```
pi@raspberry:~ $ apt-cache  search  mysql-server
mariadb-server-10.3 - MariaDB database server binaries
mariadb-server-core-10.3 - MariaDB database core server files
default-mysql-server - MySQL database server binaries and system database setup (metapackage)
default-mysql-server-core - MySQL database server binaries (metapackage)
```

```
# sudo  apt-get  install  mariadb-server

# sudo  systemctl  status  mariadb.service
```

➢ 解決 MariaDB(10.5.12)本地無密碼直接登錄的問題

在 Debian11 中安裝了 MariaDB，版本號為 10.5.12，安裝後輸入 mysql 不需要密碼便可直接登錄。

```
# sudo  mysql  -u  root  -p

-> use  mysql;

-> SELECT  *  FROM  global_priv;

(root 用戶默認是 unix_socker 類型，)

-> ALTER USER root@localhost IDENTIFIED VIA mysql_native_password

USING PASSWORD("password");

->flush privileges;

(刷新權限)

->exit
```

Reference:
https://blog.csdn.net/csgd2000/article/details/82751606

➢ Setting MySQL/MariaDB root password

```
# sudo  mysql_secure_installation


You already have your root account protected, so you can safely
answer 'n'.
Switch to unix_socket authentication [Y/n]
n
```

When prompted, answer the questions below by following the guide.

- Enter current password for root (enter for none): Just press Enter
- Set root password? [Y/n]: Y
- New password: Enter password
- Re-enter new password: Repeat password
- Remove anonymous users? [Y/n]: Y
- Disallow root login remotely? [Y/n]: Y
- Remove test database and access to it? [Y/n]: Y
- Reload privilege tables now? [Y/n]: Y

其中：

依照提示輸入 root 的密碼 Change the root password? [Y/n]

移除匿名使用者 Remove anonymous users? [Y/n]

不允許 root 遠端登入 Disallow root login remotely? [Y/n]

移除 test 資料庫及存取權 Remove test database and access to it? [Y/n]

重新載入權限的資料表 Reload privilege tables now? [Y/n]

```
# sudo  mysql  -u  root  -p
or
# sudo  mysql  -uroot  -p1qaz@wsx
```

參考文獻:
https://websiteforstudents.com/mariadb-installed-without-password-prompts-for-root-on-ubuntu-17-10-18-04-beta/

➢ MySQL 開啟遠端連線權限允許遠端裝置連線資料庫

查看目前 MySQL 所監聽的連接埠：

```
# sudo netstat -tlnp | grep mariadb
```

(netstat 指令可以顯示出網路連線、路由表、介面統計、偽裝連線和多播成員的資訊。-l 參數可以只顯示正在監聽中的連線。若使用 netstat 指令時都沒給任何參數的話，會忽略掉監聽中的連線。-t 參數可以只顯示 TCP 連線。-n 參數可以讓 IP 可以直接被輸出，而不需透過 DNS 反查其對應的網域名稱。-p 參數可以顯示佔用連線的行程。)

讓 MySQL 監聽其它網路介面：

MySQL 的主設定檔路徑為/etc/mysql/my.cnf 或是/etc/my.cnf 內，在設定檔中可能會看到其又引入了其它設定檔。

```
# sudo  vim  /etc/mysql/my.cnf
```

```
pi@raspberry: ~ [122x32]
連線(C)  編輯(E)  檢視(V)  視窗(W)  選項(O)  說明(H)
  GNU nano 3.2                                    /etc/mysql/my.cnf

# The MariaDB configuration file
#
# The MariaDB/MySQL tools read configuration files in the following order:
# 1. "/etc/mysql/mariadb.cnf" (this file) to set global defaults,
# 2. "/etc/mysql/conf.d/*.cnf" to set global options.
# 3. "/etc/mysql/mariadb.conf.d/*.cnf" to set MariaDB-only options.
# 4. "~/.my.cnf" to set user-specific options.
#
# If the same option is defined multiple times, the last one will apply.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.

#
# This group is read both both by the client and the server
# use it for options that affect everything
#
[client-server]

# Import all .cnf files from configuration directory
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mariadb.conf.d/
```

```
# ls  /etc/mysql/mariadb.conf.d/
```

```
pi@raspberry:~ $ ls /etc/mysql/mariadb.conf.d/
50-client.cnf  50-mysql-clients.cnf  50-mysqld_safe.cnf  50-server.cnf
```

```
# sudo  vim  /etc/mysql/mariadb.conf.d/50-server.cnf
```

```
pi@raspberry: ~ [122x32]
連線(C)  編輯(E)  檢視(V)  視窗(W)  選項(O)  說明(H)
  GNU nano 3.2                                /etc/mysql/mariadb.con

#
user                    = mysql
pid-file                = /run/mysqld/mysqld.pid
socket                  = /run/mysqld/mysqld.sock
#port                   = 3306
basedir                 = /usr
datadir                 = /var/lib/mysql
tmpdir                  = /tmp
lc-messages-dir         = /usr/share/mysql
#skip-external-locking

# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0
```

如果要綁定所有網路介面，那就把 bind-address 欄位值設成 0.0.0.0 吧！如果要綁定多個網路介面，也是設成 0.0.0.0，再用防火牆去擋住其它不需要用到的網路介面。

```
# sudo  systemctl  restart  mysql
# sudo  netstat  -tlnp  |  grep  mariadbd
```

```
pi@raspberry:~ $ sudo netstat -tlnp | grep mysqld
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN      6075/mysqld
```

第一種方法：

MySQL 因為安全性 root 帳號，預設只允許 localhost 連線，想要遠端裝置連線本地端的資料庫，需要設置帳號權限修改為所有 host 均可訪問：

```
# update user set host = '%' where user = 'root';
```

(以上是直接修改 root 權限的做法)，

第二種方法：

此方法較第一種好，新增一個使用者，給予 hoot 均可訪問的權限'%'，或使指定只能訪問指定某資料庫權限：

MySQL 創建一個新用戶：

```
# mysql -u root -p
mysql> CREATE USER admin IDENTIFIED BY 'password';
```

(建立使用者名稱及密碼)

admin 表示使用者帳號為 admin，若後面有加入'admin'@'localhost'表示只能 localhost 本端連入，所以我們需要所有 hoot 均可訪問權限的設置'%'，這邊只需要填入帳號名稱即可'admin'

```
mysql> GRANT  ALL  PRIVILEGES  ON  *.*  TO  'admin';
```

(給定使用者存取權限，*.*為所有使用權限)

```
mysql> GRANT ALL PRIVILEGES ON database_name.* TO 'admin';
```

(可針對資料庫 database_name.*如下:只能訪問 database_name 這個資料庫)

```
mysql> FLUSH PRIVILEGES;
```

(重新載入使用者權限設定)

==========

補充指令:

```
mysql> SELECT User FROM mysql.user;
```

(查詢資料庫使用者)

```
mysql> SELECT User, Host FROM mysql.user;
```

(查詢資料庫使用者與來源主機)

```
mysql> DROP USER 'admin';
```

(刪除使用者)

修改一般使用者密碼：

```
# mysql -uroot –p1qaz@wsx
# SET PASSWORD FOR 'admin'@'localhost' = PASSWORD('password');
```

(本機使用者)

```
# SET PASSWORD FOR 'admin' = PASSWORD('password');
```

(使用者可連遠端)

```
# flush privileges;
```

匯入資料庫:

```
# mysql  -u  root  -p  class < students.sql

Enter password



# sudo  mysql  -u  root  -p

# CREATE DATABASE myproject CHARACTER SET UTF8;

# show databases;
```

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| myproject          |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.00 sec)

MariaDB [(none)]> exit
```


參考文獻:

https://www.ucamc.com/articles/430-mysql

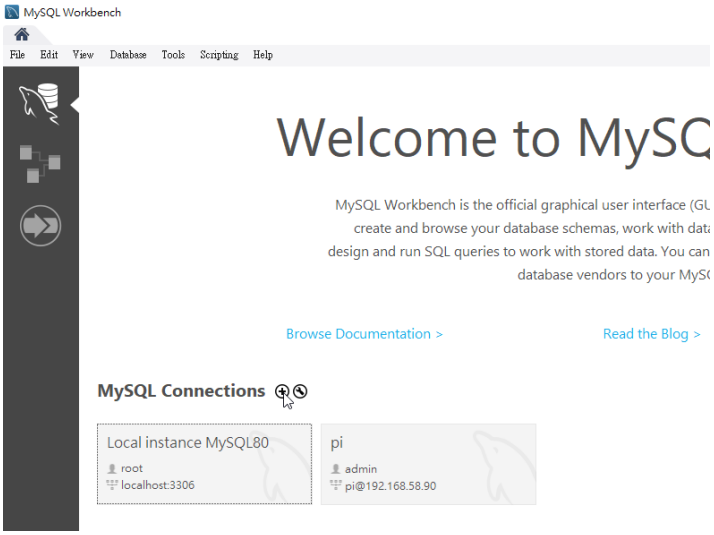
https://stackoverflow.com/questions/62564439/mysql-mariadb-server-raspberry-pi-remote-access

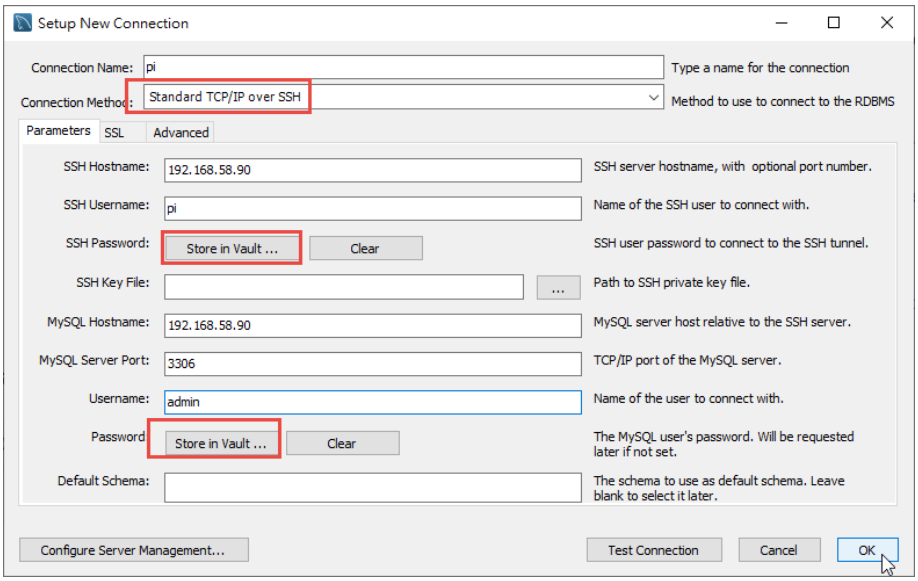
https://magiclen.org/mysql-remote/

https://emn178.pixnet.net/blog/post/87659567

https://www.dotblogs.com.tw/CodeBrewTea/2022/02/15/150432

➢ 使用 Wordbench 開啟 MariaDB 資料庫





➢ How to Reset MySQL/MariaDB Database Root Password?

```
# sudo  systemctl  stop  mariadb


# sudo  mysqld_safe  --skip-grant-tables  --skip-networking  &
```

```
pi@raspberry:~ $ 220702 23:16:24 mysqld_safe Logging to syslog.
220702 23:16:24 mysqld_safe Starting mysqld daemon with databases from /var/lib/mysql
```

按 enter

```
# sudo  mysql  -u  root
```

```
# FLUSH  PRIVILEGES;
(重新載入使用者權限設定)
# ALTER  USER  'root'@'localhost'  IDENTIFIED  BY '1234';
# exit
# sudo  mysql  -u  root  -p
```

➢ 安裝 mysqlclient 與設定 Django 參數

```
● 安裝 mysql 與設定 Django 參數
Windows:
# pip install mysqlclient (Windows)

Linux:
# sudo apt-get install python3-dev default-libmysqlclient-dev
build-essential # Debian / Ubuntu

# sudo yum install python3-devel mysql-devel # Red Hat / CentOS

Then you can install mysqlclient via pip now:
# pip install mysqlclient

Refenrce:
https://pypi.org/project/mysqlclient/
```

Linux

Note that this is a basic step. I can not support complete step for build for all environment. If you can see some error, you should fix it by yourself, or ask for support in some user forum. Don't file a issue on the issue tracker.

You may need to install the Python 3 and MySQL development headers and libraries like so:

- `$ sudo apt-get install python3-dev default-libmysqlclient-dev build-essential` # Debian / Ubuntu
- `% sudo yum install python3-devel mysql-devel` # Red Hat / CentOS

Then you can install mysqlclient via pip now:

```
$ pip install mysqlclient
```

- 資料庫設定類型

sqlite3 vs MySQL vs PostgreSQL

在專案中建立第一個 App 時會在第一層專案目錄下產生預設的空白 SQLite 資料庫檔案 db.sqlite3，此資料庫優點是備份方便（非伺服器型之單一檔案）且與 MySQL 相容，適合測試開發或小型專案使用，但在擴充性與效能上較不足，實際佈署營運時通常改用 MySQL 或 PostgreSQL 等資料庫（Django 社群較偏愛 PostgreSQL，例如免費主機 Heroku 支援 PostgreSQL）。

Django 採用 ORM 架構，因此只要修改資料庫設定，不需修改應用程式。

vim project1/settings.py

使用 SQLite 資料庫時，設定語法如下：

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
```

使用 MySQL 資料庫時，設定語法如下：

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'myproject',
        'USER': 'tony',
        'PASSWORD': '1qaz2wsx',
        'HOST': 'localhost',
        'PORT': '3306',
        'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        'charset': 'utf8mb4',
    }
}
```

```
73 # Database
74 # https://docs.djangoproject.com/en/2.2/ref/settings/#databases
75
76 DATABASES = {
77     'default': {
78         #'ENGINE': 'django.db.backends.sqlite3',
79         #'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
80         'ENGINE': 'django.db.backends.mysql',
81         'NAME': 'myproject',
82         'USER': 'root',
83         'PASSWORD': '1qaz2wsx',
84         'HOST': 'localhost',
85         'PORT': '3306',
86         'OPTIONS': {
87                 'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
88                 'charset': 'utf8mb4',
89                 }
90     }
91 }
```

- **ENGINE** -- 你要使用的資料庫引擎。例如：
  - MySQL: `django.db.backends.mysql`
  - SQLite 3: `django.db.backends.sqlite3`
  - PostgreSQL: `django.db.backends.postgresql_psycopg2`
- **NAME** -- 你的資料庫名稱

使用 Postgree 資料庫時，設定語法如下：

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'myproject',
        'USER': 'tony',
        'PASSWORD': '1qaz2wsx',
```

```
        'HOST': 'localhost',
        'PORT': '5432',
    }
```

sudo python3 manage.py makemigrations

(建立資料庫和 Django 間的中介檔)(生成表結構的快取)

sudo python3 manage.py migrate

(同步更新資料庫內容)(建立表結構)

source ./dvdsenv/bin/activate (啟動虛擬環境)

cd project1/

vim project1/settings.py (確認 myapp 加入)

```
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'myapp'
41 ]
```

- 建立 student 資料模型

sudo vim myapp/models.py

```
 1 from django.db import models
 2
 3 # Create your models here.
 4 class student(models.Model):
 5         cName = models.CharField(max_length=20, null=False)
 6         cSex = models.CharField(max_length=2, default='M', null=False)
 7         cBirthday = models.DateField(null=False)
 8         cEmail = models.EmailField(max_length=100, blank=True, default='')
 9         cPhone = models.CharField(max_length=50, blank=True, default='')
10         cAddr = models.CharField(max_length=255,blank=True, default='')
```

```
from django.db import models

class student(models.Model):

                cName = models.CharField(max_length=20, null=False)

                cSex = models.CharField(max_length=2, default='M', null=False)
```

```
cBirthday = models.DateField(null=False)
cEmail = models.EmailField(max_length=100, blank=True, default='')
cPhone = models.CharField(max_length=50, blank=True, default='')
cAddr = models.CharField(max_length=255,blank=True, default='')
```

預設已匯入 django.db.models 模組，定義資料表結構 (資料模型) 只要加入下列一個部份即可 ：
繼承 models.Model 類別定義一個資料模型子類別 (即資料表)

Django 所有的資料模型都繼承自 django.db.models.Model 類別，每一個繼承 Model 的子類別相當於是資料表 (table)；其實體 (物件) 對應到一筆紀錄 (row 或 record)；物件的屬性則對應到資料欄位 (fields)，而物件方法則對應到資料庫的 CRUD 操作，如下表所示 ：

ORM 模型	關聯式資料庫
類別 (Class)	資料表 (Table)
物件 (Object)	紀錄 (Row)
屬性 (Attribute)	欄位 (Field)
方法 (Method)	CRUD 操作

➢ Field Type 欄位型別

欄位格式	參數	說明
BooleanField		True 或 False 用於 checkbox 輸入資料
CharField	max_length:最大字串長度	用於單行輸入字串資料
SlugField		與 CharField()相同，但用來儲存 URL 的一部分
TextField		多行輸入字串資料，用於 HTML 表單的 textarea 欄位
IntegerField		整數: -2147483648~2147483647
BigInteger()		儲存長度 64 位元的大整數
PositiveIntegerField()		儲存正整數(0 ~ 2147483647)
DecimalField()	max_digits=最大位數 decimal_places=整數位數	儲存固定精度之十進位數 (Decimal 物件)
FloatField()		儲存浮點數
DateField()	auto_now=自動儲存今日日期 auto_now_add=只在建立時儲存	儲存日期，格式 datetime.date。

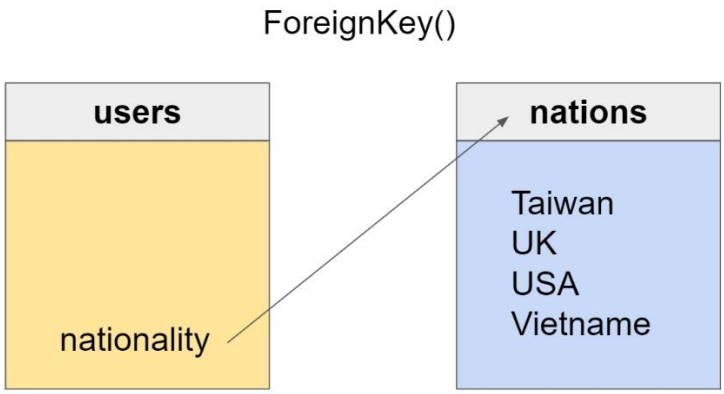
	今日日期	
DateTimeField()	auto_now=自動儲存今日日期時間 auto_now_add=只在建立時儲存今日日期時間	儲存日期時間，格式 datetime.datetime。
EmailField()	max_length=最大字元數(上限 254)	儲存有效之電子郵件
FileField()		檔案上傳欄位
ImageField()		圖檔欄位(繼承自 FileField，須配合使用 Pillow 套件)
URLField()	max_length=最大字元數(預設 200)	儲存完整的 URL (繼承自 CharField)
AutoField()	primary_key=True	自動增量主鍵欄位
ForeignKey()	第一參數= 所指之資料表類別名 第二參數: on_delete=models.CASCADE	關聯欄位，用來指向其他資料表的主鍵(預設=id)

參考文獻：
https://developer.mozilla.org/zh-TW/docs/Learn/Server-side/Django/Models

最後的兩個方法 AutoField() 與 ForeignKey() 所定義的欄位比較特殊，其中 AutoField()用來設定自動增量主鍵(primary_key)欄位，其實 Django 會自動為每一個資料表添加一個主鍵欄位 id，但若想自行指定主鍵欄位可以使用 AutoField() 方法。

ForeignKey()欄位則是用來關聯到其他資料表(稱為外部鍵)，此欄位的第一參數所關聯之資料表類別名稱，用來指向該資料表的主鍵，Django 預設會自動將其關聯到該資料表之 id 主鍵，例如上面的 users 資料表中可以添加一個 nationality (國籍) 欄位指向另一個資料表 nations：

nationality=models.ForeignKey(nations, on_delete=models.CASCADE)



ForeignKey()的第二參數用來設定當被參照的外部鍵被刪除時參照者應採取之動作，Django 的 models 套件定義了如下常用之屬性值:

- models.CASCADE：同步執行刪除動作
- models.PROTECT：不刪除且拋出一個 ProtectedError 例外
- models.SET_NULL：將此外部鍵設為 null (須先以 null=True 參數定義此欄位)
- models.SET_DEFAULT：將此外部鍵設為預設值 (須先以 default 參數定義此欄位預設值)
- models.DO_NOTHING ：不採取動作

參考文獻：
https://docs.djangoproject.com/en/3.1/topics/db/models/#automatic-primary-key-fields

➢ Field Option 欄位選項

呼叫欄位型態函數時除了必要參數外，還可以傳入選項參數，常用選項參數如下表：

欄位選項	說明
null	欄位值是否可為 null，值=True/False(預設)
blank	欄位值是否可為空白，值=True/False(預設)
default	欄位預設值(或可呼叫物件)
unique	欄位值是否為唯一，值=True/False(預設)
primary_key	欄位是否為主鍵，值=True/False(預設)
editable	欄位是否顯示於 admin 後台，值=True(預設)/False
choices	設定 select 欄位之選項(可用 list 或 tuple)
help_text	顯示於表單元件上的額外資訊
verbose_name	欄位之人類可讀名稱，未指定以欄位名稱代替(底線變空白)

補充:

使用 blank=True(允許空值)，會設定 null=True(欄位內容允許 null)

空值=>白紙; null=>沒有東西

參考文獻：

https://docs.djangoproject.com/en/3.1/ref/models/fields/

https://sites.google.com/site/djangonote/basic/models/field-type

補充(要使用 ORM 寫法才有效果):

一、 django设置字段动态默认时间的四种方式：

```
from django.db import models
from datetime import datetime


class User(models.Model):
    id = models.BigAutoField('主键', primary_key=True)

    name = models.CharField('名字', max_length=20, db_index=True, default='')

    create_time_one = models.DateTimeField('创建时间', default=datetime.now())
    update_time_one = models.DateTimeField('更新时间', default=datetime.now)

    create_time_tow = models.DateTimeField('创建时间', auto_now_add=True)
    update_time_tow = models.DateTimeField('更新时间', auto_now=True)
```

1. default=datetime.now()

model每次初始化，都会自动设置该字段的默认值为初始化时间。

2. default=datetime.now

model每次进行新增或修改操作，都会自动设置该字段的值为操作时间。设置后仍可以使用ORM手动修改该字段。

3. auto_now_add=True

默认值为False，若设置为True，model每次进行新增操作，都会自动设置该字段的值为操作时间。设置为True后无法使用ORM手动修改该字段，哪怕填充了字段的值也会被覆盖。

4. auto_now=True

默认值为False，若设置为True，model每次进行新增或修改操作，都会自动设置该字段的值为操作时间。设置为True后无法使用ORM手动修改该字段，哪怕填充了字段的值也会被覆盖。

5. 要注意的点

除非想设置动态默认时间为项目的启动时间，否则default= datetime.now()这种用法是错误的，会得到期望之外的结果。

Reference:

https://blog.csdn.net/kuanggudejimo/article/details/99291026

參考文獻：

https://www.techiediaries.com/django/django-3-tutorial-and-crud-example-with-mysql-and-bootstrap/
https://www.jianshu.com/p/02732229fe59
https://djangogirlstaipei.gitbooks.io/django-girls-taipei-tutorial/content/django/models.html
https://my.oschina.net/yimingkeji/blog/2873227
https://www.jianshu.com/p/bc41a8bf9d9b

3-11 資料庫新增、刪除、修改與查詢(使用 SQL 語法)

CRUD 指的是，Create (新增)、Read (讀取)、Update (修改)、Delete (刪除) 等常見的資料庫操作。

Executing custom SQL directly 有時候 Manager.raw() 是不夠的，像是你可能需要 queries 沒有完全 map 到 models 的資料，或是執行 UPDATE, INSERT, or DELETE。當我們使用這個方法時，是完全的繞過 model，直接 access database。你可以通過匯入 django.db.connection 對像來輕鬆實現，它代表當前資料庫連線。要使用它，需要通過 connection.cursor()得到一個遊標對像。

參考文獻:

https://docs.djangoproject.com/en/3.2/topics/db/sql/

https://docs.djangoproject.com/zh-hans/4.2/topics/db/sql/

https://github.com/twtrubiks/django-rest-framework-tutorial

- 設定 Django 連 mysql 的參數

```
# vim project2/settings.py
DATABASES = {
    'default': {
        #'ENGINE': 'django.db.backends.sqlite3',
        #'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'myproject',
        'USER': 'tony',
        'PASSWORD': '1qaz2wsx',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS':{
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
            'charset': 'utf8mb4',
        }
    }
}
```

```
# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        #'ENGINE': 'django.db.backends.sqlite3',
        #'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'myproject',
        'USER': 'root',
        'PASSWORD': '1qaz2wsx',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {
                'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
                'charset': 'utf8mb4',
                }
    }
}
```

- **ENGINE** -- 你要使用的資料庫引擎。例如：
  - MySQL: `django.db.backends.mysql`
  - SQLite 3: `django.db.backends.sqlite3`
  - PostgreSQL: `django.db.backends.postgresql_psycopg2`
- **NAME** -- 你的資料庫名稱

- 建立 student 資料模型

sudo vim myapp/models.py

models.py ×

myapp > models.py > ...

```
from django.db import models


class students(models.Model):
    cName = models.CharField(max_length=20, null=False)
    cSex = models.CharField(max_length=2, default="M", null=False)
    cBirthday = models.DateField(null=False)
    cEmail = models.EmailField(max_length=100, blank=True, default='')
    cPhone = models.CharField(max_length=20, blank=True, default='')
    cAddr = models.CharField(max_length=255, blank=True, default='')

```

```
from django.db import models


class students(models.Model):

                    cName = models.CharField(max_length=20, null=False)

                    cSex = models.CharField(max_length=2, default='M', null=False)

                    cBirthday = models.DateField(null=False)

                    cEmail = models.EmailField(max_length=100, blank=True, default='')

                    cPhone = models.CharField(max_length=50, blank=True, default='')

                    cAddr = models.CharField(max_length=255,blank=True, default='')
```

- 建立 migration 資料檔:

Diango 要使用資料庫，通常會將建立資料表的架構和版本記錄下來，以利以後的追蹤。

```
# sudo python3 ./manage.py makemigrations
```

```
(dvdsenv) pi@raspberrypi:~/dvds2/project1 $ sudo ./manage.py makemigrations
No changes detected
```

- 模型與資料庫同步

```
# sudo python3 ./manage.py migrate
```

```
(dvdsenv1) pi@raspberrypi:~/project1 $ sudo ./manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying sessions.0001_initial... OK
```

➢ 資料庫查詢(SQL-SELECT 語法)

新增瀏覽位置(urls.py):

```
# sudo vim project2/urls.py
```

urls.py

ch3_12 > urls.py > ...

```
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('list/', views.list),
```

新增 view functions

sudo vim myapp/views.py

```
8   def list(request):
9       if 'cName' in request.GET:
10          # print("ok")
11          cName = request.GET['cName']
12      else:
13          cName=None
14          # print("false")
15      print(cName)
16
17      #原始寫法
18      if cName:
19          sql = "SELECT * FROM myapp_students WHERE cName= %s"
20          val = [cName]
21          # print("ok")
22      else:
23          sql = "SELECT * FROM myapp_students"
24          val = []
25          # print("false")
26      cursor = connections['default'].cursor() #連接資料庫
27      cursor.execute(sql,val) #執行sql語法
28      result = cursor.fetchall() #取得資料
```

```
52      #轉換格式
53      field_name=cursor.description #取得資料表欄位名稱
54      cursor.close()
55      # print(field_name)
56
57      resultList=[]
58      # 兩層foreach
59      for data in result:
60          # print(data)
61          i=0
62          dict_data={}
63          for d in data:
64              # print(d)
65              dict_data[field_name[i][0]]=d
66              i=i+1
67          resultList.append(dict_data)
68
69      errormessage=""
70      if not resultList: #判斷有無資料
71          errormessage="無此資料"
72      # print(errormessage)
73      # print(resultList)
74      # return HttpResponse("test...")
75      return render(request,"list.html",locals())
```

新增 template list.html

sudo vim templates/listall.html

list.html

templates > list.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <h1>test</h1> -->
    <h1>{{ errormessage }}</h1>
    {% for data in resultList %}
        <h2>顯示student資料表的資料</h2>
        編號:{{ data.id }} <br>
        姓名:{{ data.cName }} <br>
        性別:{{ data.cSex }} <br>
        生日:{{ data.cBirthday }} <br>
        郵件:{{ data.cEmail }} <br>
        電話:{{ data.cPhone }} <br>
        地址:{{ data.cAddr }} <br>
    {% endfor %}
</body>
</html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

Document

127.0.0.1:8000/list/

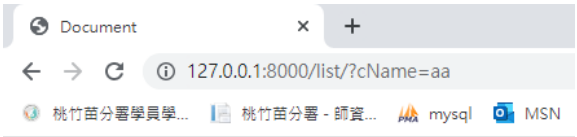
桃竹苗分署學員學... 桃竹苗分署 - 師資... mysql MSN Outlook Gmail

顯示student資料表的資料

編號:3
姓名:潘四敬
性別:F
生日:八月 11, 1987
郵件:sugie@superstar.com
電話:0914530768
地址:台北市中央路201號7樓

顯示student資料表的資料

編號:4
姓名:賴勝恩
性別:M
生日:六月 20, 1984
郵件:shane@superstar.com
電話:0946820035
地址:台北市建國路177號6樓



顯示student資料表的資料

編號:11
姓名:aa
性別:M
生日:九月 1, 2022
郵件:aa
電話:
地址:

無此資料

新增瀏覽位置(urls.py):

sudo vim project2/urls.py

urls.py

ch3_12 > urls.py > ...

```
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('list/', views.list),
    path('search_name/', views.search_name),
```

新增 view functions

```
def search_name(request):
    # return HttpResponse("test...")
    return render(request,"search_name.html",locals())

```

新增 search_name.html

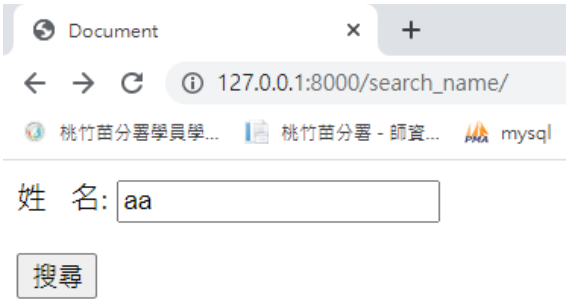
search_name.html

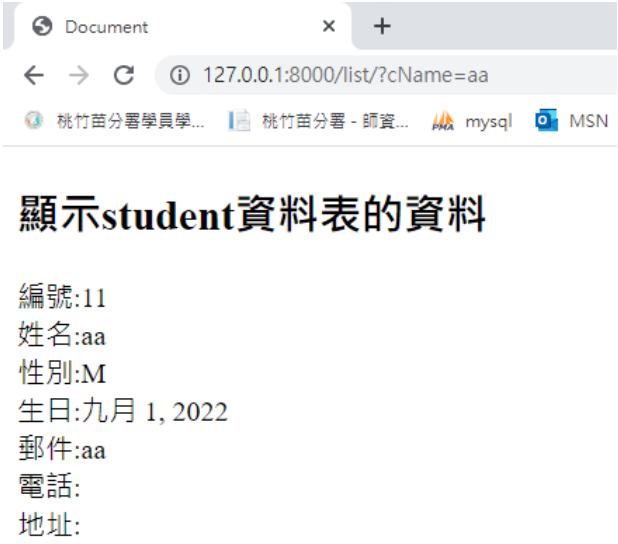
templates > search_name.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <h1>test</h1> -->
    <form action="/list/" method="GET">
        <label for="fname">姓   名:</label>
        <input type="text" name="cName" id="cName">
        <p><button type="submit">搜尋</button></p>

    </form>
</body>
</html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```





新增瀏覽位置(urls.py):

sudo vim project2/urls.py

```
from django.contrib import admin
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('listone/',views.listone),
    path('index/', views.index),
```

新增 view functions

sudo vim myapp/views.py

```
    sql = "SELECT * FROM myapp_students"
    cursor = connections['default'].cursor() #連接資料庫
    cursor.execute(sql,[]) #執行sql語法
    result = cursor.fetchall() #取得資料
    # print(result)

    #轉換格式
    field_name=cursor.description #取得資料表欄位名稱
    # print(field_name)
```

```
    ##############################################
    #for each
    resultList=[]
    for data in result:
        i=0
        dict_data={}
        for d in data:
            # print(d)
            dict_data[field_name[i][0]]=d
            i=i+1
        print(dict_data)
        resultList.append(dict_data)
    print(resultList)
    ##############################################
```

```
    errormessage=""
    status=True
    if not resultList: #判斷有無資料
        errormessage="無此資料"
        status=False
    # print(errormessage)
    data_count=len(resultList)
    # print(data_count)
    # return HttpResponse("test...")
    return render(request,"index.html",locals())
```

新增 template base.html index.html

vim templates/base.html

base.html

C: > Users > tony > DOCUME~1 > MobaXterm > slash > RemoteFiles >

```
<!-- base.html -->
<!DOCTYPE html>
<html>
    <head>
        {% block title %}{% endblock %}
    </head>
    <body>
        {% block content %} {% endblock %}
    </body>
</html>
```

vim templates/index.html

```
{% extends 'base.html' %}
{%block title%}
<title>學生資料管理系統</title>
```

```
<div>
    {% if status %}
    <table>
        <tr>
            <th>學號</th><th>姓名</th><th>性別</th><th>生日</th><th>信箱</th><th>電話</th><th>地址</th><th>編輯</th>
        </tr>
        {% for data in resultList %}
        <tr>
            <td>{{ data.id }}  </td>
            <td>{{ data.cName }}</td>
            <td>
                {% if data.cSex == "M"  %} 男 {%else%} 女 {%endif%}
            </td>
            <td>{{ data.cBirthday }}</td>
            <td>{{ data.cEmail }}</td>
            <td>{{ data.cPhone }} </td>
            <td>{{ data.cAddr }}</td>
            <td>
                <a href="/edit1/{{ data.id }}/load/">編輯1</a> <!-- get -->
                <a href="/edit2/{{ data.id }}/">編輯2</a> <!-- post -->
                <a href="/delete/{{ data.id }}/">刪除</a> <!-- get -->
            </td>
        </tr>
        {% endfor %}
    </table>
    {% else %}
        <h1>無資料</h1>
    {% endif %}
</div>
```

測試結果：

sudo python3 manage.py runserver 0.0.0.0:8080

學號	姓名	性別	生日	信箱	電話	地址	編輯
3	潘四敬	女	1987-08-11	sugie@superstar.com	0914530768	台北市中央路201號7樓	編輯1 編輯2 刪除
4	賴勝恩	男	1984-06-20	shane@superstar.com	0946820035	台北市建國路177號6樓	編輯1 編輯2 刪除
5	黎楚寧	女	1988-02-15	ivy@superstar.com	0920981230	台北市忠孝東路520號6樓	編輯1 編輯2 刪除
6	蔡中穎	男	1987-05-05	zhong@superstar.com	0951983366	台北市三民路1巷10號	編輯1 編輯2 刪除
7	徐佳螢	女	1985-08-30	lala@superstar.com	0918123456	台北市仁愛路100號	編輯1 編輯2 刪除
8	林雨煊	女	1986-12-10	crystal@superstar.com	0907408965	台北市民族路204號	編輯1 編輯2 刪除
9	林心儀	女	1988-12-01	peggy@superstar.com	0916456723	台北市建國北路10號	編輯1 編輯2 刪除
10	王燕博	男	1993-08-10	albert@superstar.com	0918976588	台北市北環路2巷80號	編輯1 編輯2 刪除
11	aa	男	2022-09-01	aa			編輯1 編輯2 刪除
12	bb	男	2022-09-08	bb33			編輯1 編輯2 刪除

➢ 資料新增(SQL-INSERT 語法)

新增瀏覽位置(urls.py):

sudo vim project1/urls.py

```
from django.contrib import admin
from django.urls import path
from myapp import views
```

```
    path('post1/', views.post1),
```

新增 view functions：

sudo vim myapp/views.py

```
from django.shortcuts import redirect
def post1(request):
    if request.method == "POST":
        cName = request.POST["cName"]
        cSex = request.POST["cSex"]
        cBirthday = request.POST["cBirthday"]
        cEmail = request.POST["cEmail"]
        cPhone = request.POST["cPhone"]
        cAddr = request.POST["cAddr"]
```

```
186         #可檢查版本
187         sql = "INSERT INTO myapp_students (cName,cSex,cBirthday,cEmail,cPhone,cAddr)"
188         sql += "VALUES('%s','%s','%s','%s','%s','%s')"
189         sql %= (cName,cSex,cBirthday,cEmail,cPhone,cAddr)
190         # print(sql)
191         cursor = connections["default"].cursor()
192         cursor.execute(sql,[])
193         cursor.close()
194
195         # return HttpResponse("已送出...")
196         return redirect('/index/') #自動轉址
197     else:
198         # return HttpResponse("test...")
199         return render(request,"post1.html",locals())
```

新增 template post1.html：

sudo vim templates/post1.html

post1.html ×

templates > post1.html

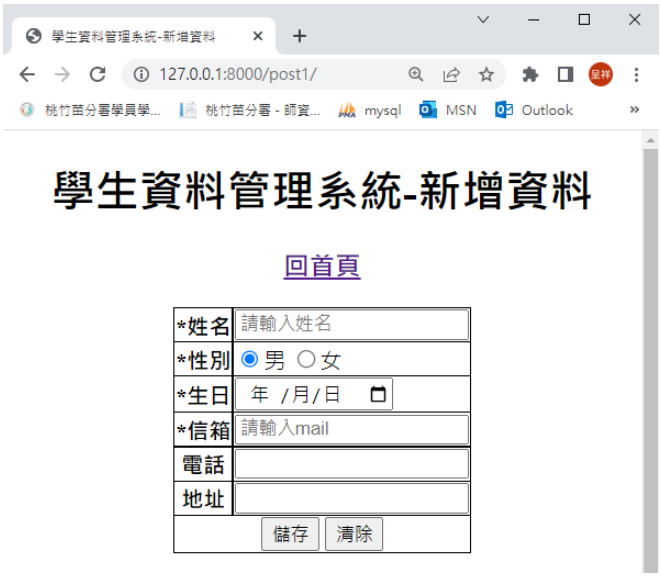
```
1   {% extends 'base.html' %}
2
3   {%block title%}
4   <title>學生資料管理系統-新增資料</title>
5   <style>
6       h1, h3{
7           text-align: center;
8       }
9       table{
10          margin-left: auto;
11          margin-right: auto;
12      }
13      table, th, td{
14          border: 1px solid black;
15          border-collapse: collapse;
16      }
17
18  </style>
19  {% endblock %}
20
21  {% block content %}
22  <div>
23  <h1 class="title">學生資料管理系統-新增資料</h1>
24  <a href="/index/"><h3>回首頁</h3></a>
25  </div>
```

```
27  <div>
28      <form action="/post1/" method="POST">
29          {% csrf_token %}
30          <table>
31              <tr>
32                  <th>*姓名</th><td><input type="text" id="cName" name="cName" required placeholder="請輸入姓名"></td>
33              </tr>
34              <tr>
35                  <th>*性別</th>
36                  <td>
37                      <input type="radio" id="cSex" name="cSex" value="M" checked>男
38                      <input type="radio" id="cSex" name="cSex" value="F">女
39                  </td>
40              </tr>
41              <tr>
42                  <th>*生日</th><td><input type="date" id="cBirthday" name="cBirthday" required></td>
43              </tr>
44              <tr>
45                  <th>*信箱</th><td><input type="mail" id="cEmail" name="cEmail" required placeholder="請輸入mail"></
                    td>
46              </tr>
47              <tr>
48                  <th>電話</th><td><input type="text" id="cPhone" name="cPhone"></td>
49              </tr>
50              <tr>
51                  <th>地址</th><td><input type="text" id="cAddr" name="cAddr"></td>
52              </tr>
```

```
53              <tr>
54                  <th colspan="2" style="text-align:center;">
55                      <input type="submit" name="button" id="button" value="儲存">
56                      <input type="reset" name="button2" id="button2" value="清除">
57                  </th>
58
59              </tr>
60          </table>
61
62      </form>
63  </div>
64
65  {% endblock %}
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```



➢ 資料刪除(SQL-DELETE 語法)

新增瀏覽位置(urls.py):

```
# sudo vim project2/urls.py
```

urls.py

ch3_12 > urls.py > ...

```python
from django.contrib import admin
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('list/', views.list),
    path('search_name/', views.search_name),
    path('index/',views.index),
    path('post1/',views.post1),
    path('edit1/<int:id>/<str:mode>/', views.edit1),
    path('edit2/<int:id>/', views.edit2),
    path('delete/<int:id>/', views.delete),


]
```

新增 view functions：

```
# sudo vim myapp/views.py
```

views.py × urls.py

myapp > views.py > index

```python
def delete(request, id=None):
    if request.method == "POST":
        sql = "DELETE FROM myapp_students WHERE id=%s"
        sql %= (id)
        cursor = connections["default"].cursor()
        cursor.execute(sql,[])
        cursor.close()
        print(sql)
        # return HttpResponse("按刪除")
        return redirect('/index/')
    else:
        sql = "SELECT * FROM myapp_students WHERE id = %s" %(id)
        # print(sql)
        cursor = connections["default"].cursor()
        cursor.execute(sql,[])
        result = cursor.fetchall()#資料
        # print(result)
        field_names = cursor.description#欄位名稱
        # print(field_names)
        cursor.close()

        dict_data={}
        for data in result:
            i=0
            for d in data:
                # print(d)
                dict_data[field_names[i][0]]=d
                i=i+1
            # print(dict_data)
        print(dict_data)
        # return HttpResponse("test.......")
        return render(request,"delete.html", locals())
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

➢ 資料更新(使用 GET) (SQL-UPDATE 語法)

新增瀏覽位置(urls.py):

```
# sudo vim project2/urls.py
```

urls.py

ch3_12 > urls.py > ...

```
from django.contrib import admin
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('list/', views.list),
    path('search_name/', views.search_name),
    path('index/',views.index),
    path('post1/',views.post1),
    path('edit1/<int:id>/<str:mode>/', views.edit1),
    path('edit2/<int:id>/', views.edit2),
    path('delete/<int:id>/', views.delete),
```

新增 view functions：

sudo vim myapp/views.py

```
201 def edit1(request, id=None, mode=None):
202     # print(id)
203     # print(mode)
204     if mode == "edit":
205         cName = request.GET["cName"]
206         cSex = request.GET["cSex"]
207         cBirthday = request.GET["cBirthday"]
208         cEmail = request.GET["cEmail"]
209         cPhone = request.GET["cPhone"]
210         cAddr = request.GET["cAddr"]
```

```
221         sql = "UPDATE myapp_students SET "
222         sql += "cName='%s', cSex='%s',cBirthday='%s', cEmail='%s', cPhone='%s',
            cAddr='%s' WHERE id=%s"
223         sql %= (cName, cSex, cBirthday, cEmail, cPhone, cAddr, id)
224         print(sql)
225
226         cursor = connections["default"].cursor() #資料庫連線
227         cursor.execute(sql,[]) #執行sql語法
228         cursor.close()
229
230         # return HttpResponse("修改.......")
231         return redirect('/index/')
```

```
224     elif mode == "load":
225         sql = "SELECT * FROM myapp_students WHERE id = %s" %(id)
226         # print(sql)
227         cursor = connections["default"].cursor() #資料庫連線
228         cursor.execute(sql,[]) #執行sql語法
229         result = cursor.fetchall() #取得資料
230         print(result)
231         field_names = cursor.description#欄位名稱
232         cursor.close()
233         print(field_names)
234
235         dict_data={}
236         for data in result:
237             i=0
238             for d in data:
239                 # print(d)
240                 dict_data[field_names[i][0]]=d
241                 i=i+1
242             # print(dict_data)
243         print(dict_data)
244         # return HttpResponse("test.......")
245         return render(request,"edit1.html", locals())
```

新增 template edit1.html：

sudo vim templates/edit1.html

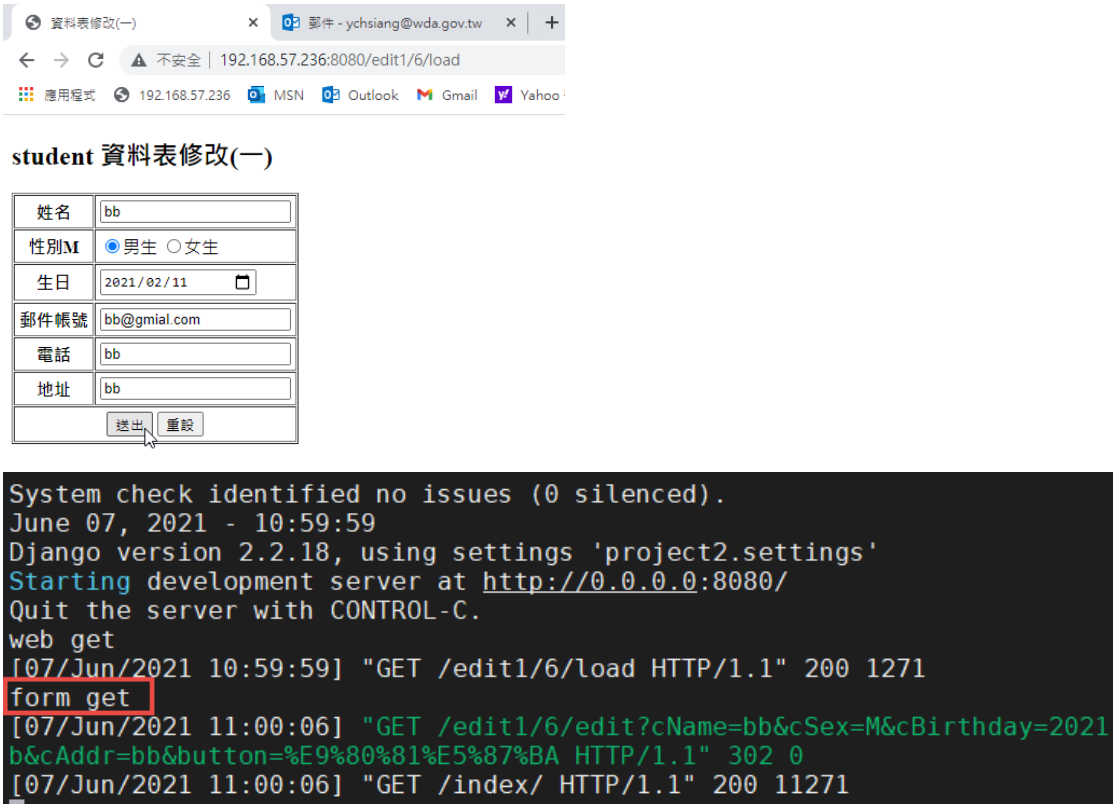
edit1.html

templates > edit1.html

```html
{% extends 'base.html' %}

{% block title %}
<title>學生資料管理系統-修改資料一(GET)</title>
{% endblock title %}

{% block content %}
<h1>學生資料管理系統-修改資料一(GET)</h1>
<a href="/index/"><h3>回首頁</h3></a>
<form action="/edit1/{{dict_data.id}}/edit/" method="get">
    <table>
        <tr>
            <th>姓名</th><td><input type="text" id="cName" name="cName" value="{{ dict_data.cName }}"></td>
        </tr>
        <tr>
            <th>性別</th>
            <td>
                <input type="radio" id="cSex" name="cSex" value="M" {% if dict_data.cSex == "M"%} checked {% endif %} >男
                <input type="radio" id="cSex" name="cSex" value="F" {% if dict_data.cSex == "F"%} checked {% endif %} >女

            </td>
        </tr>
        <tr>
            <th>生日</th><td><input type="date" id="cBirthday" name="cBirthday" value="{{ dict_data.cBirthday }}"></td>
        </tr>
        <tr>
            <th>信箱</th><td><input type="text" id="cEmail" name="cEmail" value="{{ dict_data.cEmail }}"></td>
        </tr>
        <tr>
            <th>電話</th><td><input type="text" id="cPhone" name="cPhone" value="{{ dict_data.cPhone }}"></td>
        </tr>
        <tr>
            <th>地址</th><td><input type="text" id="cAddr" name="cAddr" value="{{ dict_data.cAddr }}"></td>
        </tr>
        <tr>
            <th colspan="2" style="text-align:center;">
                <input type="submit" name="button" id="button" value="儲存">
                <input type="reset" name="button2" id="button2" value="重設">
            </th>
        </tr>
    </table>
</form>
{% endblock content %}
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```



測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

➢ 資料更新(使用 GET) (SQL-UPDATE 語法)

新增瀏覽位置(urls.py):

```
# sudo vim project2/urls.py
```

urls.py

ch3_12 > urls.py > ...

```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('list/', views.list),
23      path('search_name/', views.search_name),
24      path('index/',views.index),
25      path('post1/',views.post1),
26      path('edit1/<int:id>/<str:mode>/', views.edit1),
27      path('edit2/<int:id>/', views.edit2),
28      path('delete/<int:id>/', views.delete),
29
30
31  ]
```

新增 view functions：

```
# sudo vim myapp/views.py
```

```python
def edit2(request, id=None):
    # print(id)
    if request.method == "POST":
        cName = request.POST["cName"]
        cSex = request.POST["cSex"]
        cBirthday = request.POST["cBirthday"]
        cEmail = request.POST["cEmail"]
        cPhone = request.POST["cPhone"]
        cAddr = request.POST["cAddr"]
        sql = "UPDATE myapp_students SET "
        sql += "cName='%s', cSex='%s',cBirthday='%s', cEmail='%s', cPhone='%s', cAddr='%s' WHERE id=%s"
        sql %= (cName, cSex, cBirthday, cEmail, cPhone, cAddr, id)
        print(sql)

        cursor = connections["default"].cursor()
        cursor.execute(sql,[])
        cursor.close()

        # return HttpResponse("修改.......")
        return redirect('/index/')
```

```python
    else:
        sql = "SELECT * FROM myapp_students WHERE id = %s" %(id)
        # print(sql)
        cursor = connections["default"].cursor()
        cursor.execute(sql,[])
        result = cursor.fetchall() #資料
        print(result)
        field_names = cursor.description #欄位名稱
        print(field_names)
        cursor.close()

        dict_data={}
        for data in result:
            i=0
            for d in data:
                # print(d)
                dict_data[field_names[i][0]]=d
                i=i+1
            # print(dict_data)
        print(dict_data)
        # return HttpResponse("test.......")
        return render(request,"edit2.html", locals())
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

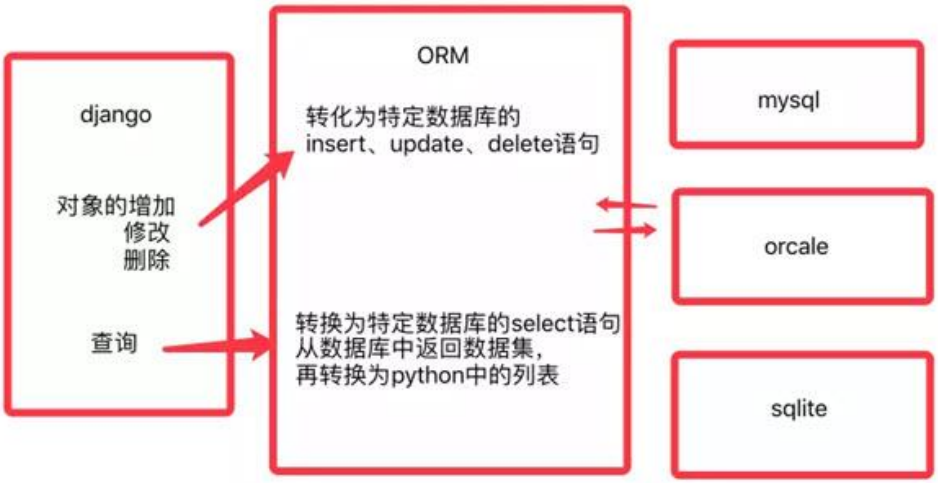
資料表修改(二)
192.168.57.2
不安全 | 192.168.57.236:8080/ec
應用程式
192.168.57.236
MSN
Outlo
student 資料表修改(二)
姓名
bb33
性別F
男生
女生
生日
2021/06/12
郵件帳號
bb33@gmial.com
電話
0999933
地址
pp
送出
重設

顯示所有資料
192.168.57.236 / localhost / my
不安全 | 192.168.57.236:8080/index/
應用程式
192.168.57.236
MSN
Outlook
Gmail
Yahoo 奇摩電子信箱
Google 翻譯
Yahoo奇摩
YouTu
顯示 student 資料表所有資料
新增資料
編號 姓名 性別 生日 郵件帳號 電話 地址 編輯
6 bb33 M 2021年6月12日 bb33@gmial.com 0999933 pp 編輯一 編輯二 刪除

3-12 資料庫查詢 (使用 ORM)

Django 資料庫操作並不需要直接面對 SQL，而是使用抽象化的物件關聯對應模型 ORM (Object Relational Model) 將底層 SQL 操作包裝成物件方法，具體來說就是提供了 django.db.models.Model 類別作為應用程式與資料庫之間的介面 (MTV 架構中的 M 指的就是這個 Model 類別)，只要使用由這個類別所建立之資料庫模型物件就能操作資料庫與資料表。由於各家資料庫系統所使用之 SQL 語法存在差異，ORM 將操作 SQL 的細節包裝起來最大的好處就是幾乎可以無痛轉換資料庫，從預設的 SQLite 轉換至 MySQL 或 Postgre 等資料庫時只要改換設定即可，不需要修改程式或與調整 SQL 語法。

CRUD 指的是，Create (新增)、Read (讀取)、Update (修改)、Delete (刪除) 等常見的資料庫操作。ORM，即 Object-Relational Mapping（Object-Relational Mapping，Object-Relational Mapping），它的作用是在關係型數據庫和業務實體對像作一個映射，ORM 為之間關係型數據庫提供了抽象的抽象，它增加了開發人員不用 SQL，只需要數據寫代碼在數據庫中創建、讀取更新和刪除。開發人員能夠使用他們熟悉的編程語言來處理數據庫，而無需寫入 SQL 語句或存儲過程。



ORM 主要任務是：

1. 可根據對象的類型生成表結構。
2. 將對象、列表的操作，轉換為 sql 語句。
3. 將 sql 查詢到的結果轉換為對象、列表。

ORM 的優點：

1. 隱藏數據訪問細節，ORM 提供了對數據庫的映射，不用 sql 直接編碼，才能像操作對像一樣從數據庫獲取數據。

2. 提高了開發效率，幾乎所有的 ORM 框架都通過對像模型構造關係數據庫結構的功能提供。
3. 實現了數據模型與數據庫的解耦，即數據模型的不需要設計依賴於特定的數據庫，通過簡單的配置就可以輕鬆更換數據庫。

ORM 的缺點：

1. 現在的各種 ORM 框架都在嘗試使用各種方法來實踐這塊（LazyLoad，Cache）。
2. 對於復雜的查詢力不從心，例如分組算列，case，group，order by，exists 等。
3. 需要消耗更多的記憶體，使用 ORM 對像這樣的全部數據都再提取到記憶體對像中，然後進行過濾和加工處理，就容易產生性能問題。
4. 將復雜性從數據庫轉移到應用程序代碼中，沒有在應用程序和存儲過程之間將代碼進行分解，從而提高了 Python 的總代碼量。

Django 模型（model）系統－常用查詢語法：
https://www.itread01.com/content/1543740968.html

Django 筆記-模型與資料庫:
http://dokelung-blog.logdown.com/posts/220606-django-notes-5-model-and-database

➢ SQLite vs MySQL vs PostgreSQL

SQLite：

優點：

1、占用空間小：顧名思義，SQLite 庫非常輕巧。儘管它使用的空間因安裝的系統而異，但可以占用不到 600KiB 的空間。此外，它是完全獨立的，這意味著您無需在系統上安裝任何外部依賴項即可運行 SQLite。

2、用戶友好：SQLite 有時被描述為「零配置」資料庫，可以直接使用。SQLite 不會作為伺服器進程運行，這意味著它永遠都不需要停止，啟動或重新啟動，也不需要任何需要管理的配置文件。這些功能有助於簡化從安裝 SQLite 到將其與應用程式集成的路徑。

3、可移植：與通常將數據存儲為一大批單獨文件的其他資料庫管理系統不同，整個 SQLite 資料庫存儲在單個文件中。該文件可以位於目錄層次結構中的任何位置，並且可以通過可移動媒體或文件傳輸協議共享。

缺點：

1、有限的並發性：儘管多個進程可以同時訪問和查詢 SQLite 資料庫，但是在任何給定時間只有一個進程可以對資料庫進行更改。這意味著與大多數其他嵌入式資料庫管理系統相比，SQLite 支持更大的並發性，但是不如 MySQL 或 PostgreSQL 這樣的客戶端/伺服器 RDBMS。

2、沒有用戶管理：資料庫系統通常帶有對用戶的支持，或具有對資料庫和表的預定義訪問權限的託管連接。由於 SQLite 直接讀取和寫入普通磁碟文件，因此唯一適用的訪問權限是基礎作業系統的典型訪問權限。對於需要多個具有特殊訪問權限的用戶的應用程式，這使 SQLite 成為糟糕的選擇。

3、安全性：在某些情況下，使用伺服器的資料庫引擎比無伺服器的資料庫（如 SQLite）可以更好地保護客戶端應用程式中的錯誤。例如，客戶端中的雜散指針不能破壞伺服器上的內存。而且，由於伺服器是單個持久性進程，因此與無伺服器的資料庫相比，客戶端-伺服器的資料庫可以更精確地控制數據訪問，從而實現更細粒度的鎖定和更好的並發性。

MySQL：

優點：

1、受歡迎程度和易用性：作為世界上最流行的資料庫系統之一，擁有使用 MySQL 經驗的資料庫管理員並不乏。同樣，關於如何安裝和管理 MySQL 資料庫，還有大量印刷和在線文檔，以及許多旨在簡化資料庫入門過程的第三方工具，例如 phpMyAdmin。

2、安全性：MySQL 隨附了一個腳本，該腳本可通過設置安裝的密碼安全級別，為 root 用戶定義密碼，刪除匿名帳戶以及刪除默認情況下可訪問的測試資料庫來幫助您提高資料庫的安全性所有用戶。另外，與 SQLite 不同，MySQL 確實支持用戶管理，並允許您逐個用戶授予訪問權限。

3、速度：通過選擇不實現 SQL 的某些功能，MySQL 開發人員可以優先考慮速度。儘管最近的基準測試表明，其他 PostgreSQL 之類的 RDBMS 在速度方面可以與 MySQL 匹敵，或者至少接近 MySQL，但 MySQL 仍然享有極快的資料庫解決方案聲譽。

4、複製：MySQL 支持許多不同類型的複製，這是在兩個或多個主機之間共享信息的實踐，以幫助提高可靠性，可用性和容錯能力。這對於設置資料庫備份解決方案或橫向擴展資料庫很有幫助。

缺點：

1、已知局限性：由於 MySQL 是為了提高速度和易用性而不是完全符合 SQL 的要求而設計的，因此它具有某些功能局限性。例如，它不支持 FULL JOIN 子句。

2、許可和專有功能：MySQL 是雙重許可的軟體，具有根據 GPLv2 許可的免費開源社區版本，以及根據專有許可發行的若干付費商業版本。因此，某些功能和插件僅適用於專有版本。

3、發展緩慢：自從 MySQL 項目於 2008 年被 Sun Microsystems 收購，然後在 2009 年被 Oracle Corporation 收購後，用戶抱怨 DBMS 的開發過程已經大大減慢了，因為社區不再擁有代理機構快速應對問題並實施更改。

PostgreSQL：

優點：

1、SQL 合規性：PostgreSQL 比 SQLite 或 MySQL 還要嚴格遵守 SQL 標準。根據 PostgreSQL 的官方文檔，除了一長串可選功能之外，PostgreSQL 還支持 179 種完全符合 SQL：2011 核心要求的功能。

2、開源和社區驅動：PostgreSQL 的原始碼是一個完全開放原始碼的項目，它是由一個龐大而專門的社區開發的。同樣，Postgres 社區維護並提供了許多在線資源，這些資源描述了如何使用 DBMS，包括官方文檔，PostgreSQL Wiki 和各種在線論壇。

3、可擴展：用戶可以通過其目錄驅動的操作及其對動態加載的使用，以編程方式即時擴展 PostgreSQL。可以指定一個目標代碼文件，例如共享庫，而 PostgreSQL 將根據需要加載它。

缺點:

1、內存性能：對於每個新的客戶端連接，PostgreSQL 都會派生一個新進程。每個新進程都分配了大約 10MB 的內存，對於具有大量連接的資料庫而言，這可以快速增加內存。因此，對於簡單的繁重的操作，PostgreSQL 通常比其他 RDBMS（如 MySQL）的性能要差。

2、流行度：儘管近年來使用更為廣泛，但從歷史上看，PostgreSQL 在流行度方面一直落後於 MySQL。其結果之一是，可以幫助管理 PostgreSQL 資料庫的第三方工具仍然較少。同樣，與擁有 MySQL 經驗的資料庫管理員相比，擁有經驗豐富的 Postgres 資料庫的資料庫管理員也要少得多。

- 設定 Django 連 mysql 的參數

```
# vim project2/settings.py
DATABASES = {
    'default': {
        #'ENGINE': 'django.db.backends.sqlite3',
        #'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'myproject',
        'USER': 'tony',
        'PASSWORD': '1qaz2wsx',
```

```
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS':{
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
            'charset': 'utf8mb4',
        }
    }
}
```

```
73 # Database
74 # https://docs.djangoproject.com/en/2.2/ref/settings/#databases
75
76 DATABASES = {
77     'default': {
78         #'ENGINE': 'django.db.backends.sqlite3',
79         #'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
80         'ENGINE': 'django.db.backends.mysql',
81         'NAME': 'myproject',
82         'USER': 'root',
83         'PASSWORD': '1qaz2wsx',
84         'HOST': 'localhost',
85         'PORT': '3306',
86         'OPTIONS': {
87                 'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
88                 'charset': 'utf8mb4',
89                 }
90     }
91 }
```

- **ENGINE** -- 你要使用的資料庫引擎。例如：
  - MySQL: `django.db.backends.mysql`
  - SQLite 3: `django.db.backends.sqlite3`
  - PostgreSQL: `django.db.backends.postgresql_psycopg2`
- **NAME** -- 你的資料庫名稱

- 建立 student 資料模型

sudo vim myapp/models.py

models.py ×

myapp > models.py > ...

```
1   from django.db import models
2
3
4   class students(models.Model):
5       cName = models.CharField(max_length=20, null=False)
6       cSex = models.CharField(max_length=2, default="M", null=False)
7       cBirthday = models.DateField(null=False)
8       cEmail = models.EmailField(max_length=100, blank=True, default='')
9       cPhone = models.CharField(max_length=20, blank=True, default='')
10      cAddr = models.CharField(max_length=255, blank=True, default='')
11
```

```
from django.db import models


class students(models.Model):
                    cName = models.CharField(max_length=20, null=False)
                    cSex = models.CharField(max_length=2, default='M', null=False)
                    cBirthday = models.DateField(null=False)
                    cEmail = models.EmailField(max_length=100, blank=True, default='')
                    cPhone = models.CharField(max_length=50, blank=True, default='')
                    cAddr = models.CharField(max_length=255,blank=True, default='')
```

- 建立 migration 資料檔:

Diango 要使用資料庫，通常會將建立資料表的架構和版本記錄下來，以利以後的追蹤。

sudo python3 ./manage.py makemigrations

```
(dvdsenv) pi@raspberrypi:~/dvds2/project1 $ sudo ./manage.py makemigrations
No changes detected
```

- 模型與資料庫同步

sudo python3 ./manage.py migrate

```
(dvdsenv1) pi@raspberrypi:~/project1 $ sudo ./manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying sessions.0001_initial... OK
```

➢ 資料庫查詢(SQL-SELECT 語法)

新增瀏覽位置(urls.py):

sudo vim project2/urls.py

urls.py

ch3_12 > urls.py > ...

```python
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('list/', views.list),
```

新增 view functions

sudo vim myapp/views.py

```python
def list(request):
    ########################################
    # 可檢查sql寫法
    if 'cName' in request.GET:
        # print("ok")
        cName = request.GET['cName']
        print(cName)
        # ORM:
        resultList = students.objects.filter(cName__contains=cName)
    else:
        # print("false")
        # resultList= students.objects.all().order_by("-cID")  #讀取資料表, 依 id 遞減排序
        # ORM:
        resultList= students.objects.all().order_by("id")  #讀取資料表, 依 id 遞增排序

    errormessage=""
    if not resultList: #判斷有無資料
        errormessage="無此資料"
    # print(errormessage)
    # print(resultList)
    # return HttpResponse("test...")
    return render(request,"list.html",locals())
```

新增 template list.html

sudo vim templates/listall.html

list.html

templates > list.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <h1>test</h1> -->
    <h1>{{ errormessage }}</h1>
    {% for data in resultList %}
        <h2>顯示student資料表的資料</h2>
        編號:{{ data.id }} <br>
        姓名:{{ data.cName }} <br>
        性別:{{ data.cSex }} <br>
        生日:{{ data.cBirthday }} <br>
        郵件:{{ data.cEmail }} <br>
        電話:{{ data.cPhone }} <br>
        地址:{{ data.cAddr }} <br>
    {% endfor %}
</body>
</html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

Document

127.0.0.1:8000/list/

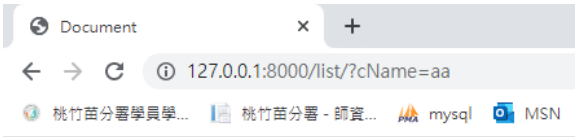
桃竹苗分署學員學... 桃竹苗分署 - 師資... mysql MSN Outlook Gmail

顯示student資料表的資料

編號:3
姓名:潘四敬
性別:F
生日:八月 11, 1987
郵件:sugie@superstar.com
電話:0914530768
地址:台北市中央路201號7樓

顯示student資料表的資料

編號:4
姓名:賴勝恩
性別:M
生日:六月 20, 1984
郵件:shane@superstar.com
電話:0946820035
地址:台北市建國路177號6樓



顯示student資料表的資料

編號:11
姓名:aa
性別:M
生日:九月 1, 2022
郵件:aa
電話:
地址:

無此資料

新增瀏覽位置(urls.py):

sudo vim project2/urls.py

urls.py

ch3_12 > urls.py > ...

```
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('list/', views.list),
    path('search_name/', views.search_name),
```

新增 view functions

```
def search_name(request):
    # return HttpResponse("test...")
    return render(request,"search_name.html",locals())

```

新增 search_name.html

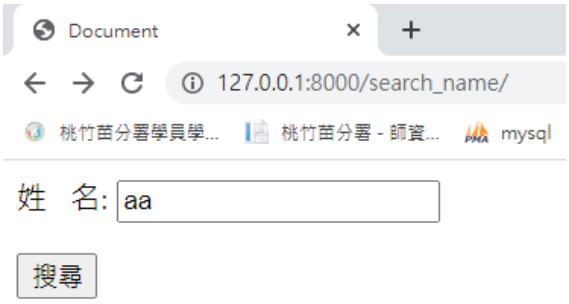
search_name.html

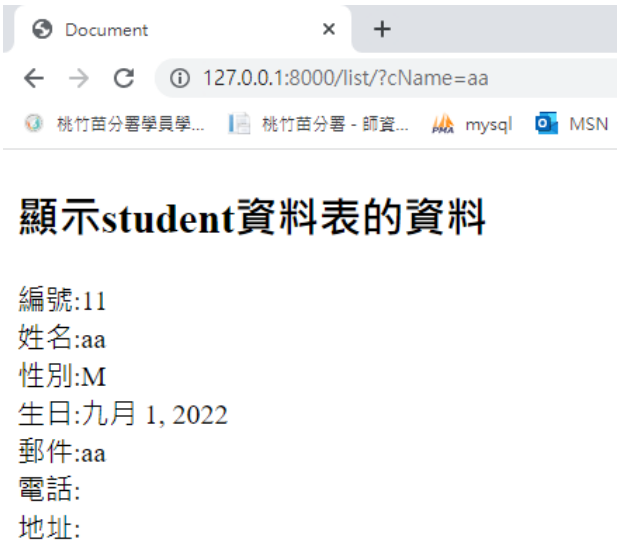
templates > search_name.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <h1>test</h1> -->
    <form action="/list/" method="GET">
        <label for="fname">姓   名:</label>
        <input type="text" name="cName" id="cName">
        <p><button type="submit">搜尋</button></p>

    </form>
</body>
</html>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```





新增瀏覽位置(urls.py):

sudo vim project2/urls.py

```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('listone/',views.listone),
23      path('index/', views.index),
```

新增 view functions

sudo vim myapp/views.py

```
37  def index(request):
38
39      resultList= students.objects.all().order_by("id")  #讀取資料表, 依 id 遞增排序
40
41      errormessage=""
42      status=True
43      if not resultList: #判斷有無資料
44          errormessage="無此資料"
45          status=False
46      # print(errormessage)
47      data_count=len(resultList)
48      # print(data_count)
49      # return HttpResponse("test...")
50      return render(request,"index.html",locals())
```

新增 template base.html index.html

vim templates/base.html

```
base.html ×
C: > Users > tony > DOCUME~1 > MobaXterm > slash > RemoteFiles >
1   <!-- base.html -->
2   <!DOCTYPE html>
3   <html>
4       <head>
5           {% block title %}{% endblock %}
6       </head>
7       <body>
8           {% block content %} {% endblock %}
9       </body>
10  </html>
```

```
# vim templates/index.html
```

```
1   {% extends 'base.html' %}
2   {%block title%}
3   <title>學生資料管理系統</title>
```

```
34  <div>
35      {% if status %}
36      <table>
37          <tr>
38              <th>學號</th><th>姓名</th><th>性別</th><th>生日</th><th>信箱</th><th>電話</th><th>地址</th><th>編輯</th>
39          </tr>
40          {% for data in resultList %}
41          <tr>
42              <td>{{ data.id }}  </td>
43              <td>{{ data.cName }}</td>
44              <td>
45                  {% if data.cSex == "M"  %} 男 {%else%} 女 {%endif%}
46              </td>
47              <td>{{ data.cBirthday }}</td>
48              <td>{{ data.cEmail }}</td>
49              <td>{{ data.cPhone }} </td>
50              <td>{{ data.cAddr }}</td>
51              <td>
52                  <a href="/edit1/{{ data.id }}/load/">編輯1</a> <!-- get -->
53                  <a href="/edit2/{{ data.id }}/">編輯2</a> <!-- post -->
54                  <a href="/delete/{{ data.id }}/">刪除</a> <!-- get -->
55              </td>
56          </tr>
57          {% endfor %}
58      </table>
59      {% else %}
60          <h1>無資料</h1>
61      {% endif %}
62  </div>
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

➢ 資料新增(SQL-INSERT 語法)

新增瀏覽位置(urls.py):

sudo vim project1/urls.py

```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
```

```
26      path('post1/', views.post1),
```

新增 view functions：

sudo vim myapp/views.py

```
77  from django.shortcuts import redirect
78  def post1(request):
79      if request.method == "POST":
80          cName = request.POST["cName"]
81          cSex = request.POST["cSex"]
82          cBirthday = request.POST["cBirthday"]
83          cEmail = request.POST["cEmail"]
84          cPhone = request.POST["cPhone"]
85          cAddr = request.POST["cAddr"]
86          # print(cName+" "+cSex+" "+cBirthday+" "+cEmail+" "+cPhone+" "+cAddr+" ")
87
88           # orm:
89          add = students(cName=cName, cSex=cSex, cBirthday=cBirthday, cEmail=cEmail, cPhone=cPhone, cAddr=cAddr)
90          add.save()
91
92          # return HttpResponse("已送出...")
93          return redirect('/index/') #自動轉址
94      else:
95          # return HttpResponse("test...")
96          return render(request,"post1.html",locals())
```

新增 template post1.html：

sudo vim templates/post1.html

post1.html

templates > post1.html

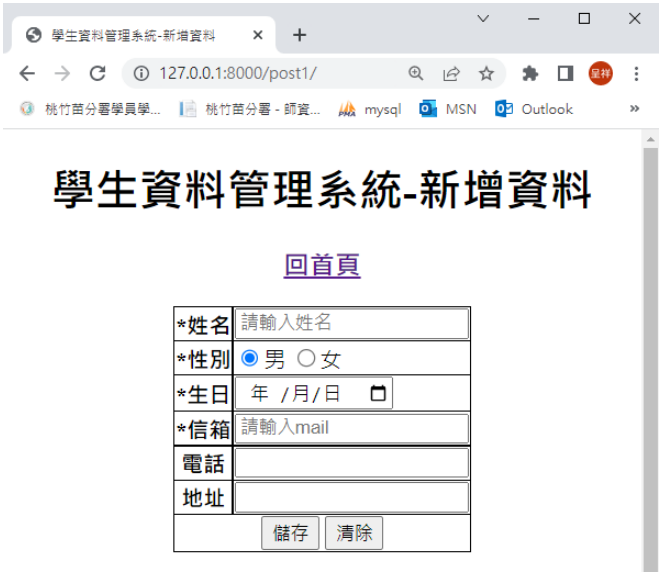
```
1   {% extends 'base.html' %}
2
3   {%block title%}
4   <title>學生資料管理系統-新增資料</title>
5   <style>
6       h1, h3{
7           text-align: center;
8       }
9       table{
10          margin-left: auto;
11          margin-right: auto;
12      }
13      table, th, td{
14          border: 1px solid black;
15          border-collapse: collapse;
16      }
17
18  </style>
19  {% endblock %}
20
21  {% block content %}
22  <div>
23  <h1 class="title">學生資料管理系統-新增資料</h1>
24  <a href="/index/"><h3>回首頁</h3></a>
25  </div>
```

```
27  <div>
28      <form action="/post1/" method="POST">
29          {% csrf_token %}
30          <table>
31              <tr>
32                  <th>*姓名</th><td><input type="text" id="cName" name="cName" required placeholder="請輸入姓名"></td>
33              </tr>
34              <tr>
35                  <th>*性別</th>
36                  <td>
37                      <input type="radio" id="cSex" name="cSex" value="M" checked>男
38                      <input type="radio" id="cSex" name="cSex" value="F">女
39                  </td>
40              </tr>
41              <tr>
42                  <th>*生日</th><td><input type="date" id="cBirthday" name="cBirthday" required></td>
43              </tr>
44              <tr>
45                  <th>*信箱</th><td><input type="mail" id="cEmail" name="cEmail" required placeholder="請輸入mail"></
                    td>
46              </tr>
47              <tr>
48                  <th>電話</th><td><input type="text" id="cPhone" name="cPhone"></td>
49              </tr>
50              <tr>
51                  <th>地址</th><td><input type="text" id="cAddr" name="cAddr"></td>
52              </tr>
```

```
53              <tr>
54                  <th colspan="2" style="text-align:center;">
55                      <input type="submit" name="button" id="button" value="儲存">
56                      <input type="reset" name="button2" id="button2" value="清除">
57                  </th>
58
59              </tr>
60          </table>
61
62      </form>
63  </div>
64
65  {% endblock %}
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```



➢ 資料刪除(SQL-DELETE 語法)

新增瀏覽位置(urls.py):

```
# sudo vim project2/urls.py
```

```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('list/', views.list),
23      path('search_name/', views.search_name),
24      path('index/',views.index),
25      path('post1/',views.post1),
26      path('edit1/<int:id>/<str:mode>/', views.edit1),
27      path('edit2/<int:id>/', views.edit2),
28      path('delete/<int:id>/', views.delete),
29
30
31  ]
```

新增 view functions：

```
# sudo vim myapp/views.py
```

```python
def delete(request, id=None):
    if request.method == "POST":
        # orm:
        delete_data = students.objects.get(id=id)
        delete_data.delete()
        # return HttpResponse("按刪除")
        return redirect('/index/')
    else:
        # orm:
        dict_data = students.objects.get(id=id)
        # return HttpResponse("test.......")
        return render(request,"delete.html", locals())
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

➢ 資料更新(使用 GET) (SQL-UPDATE 語法)

新增瀏覽位置(urls.py):

sudo vim project2/urls.py

urls.py

ch3_12 > urls.py > ...

```python
from django.contrib import admin
from django.urls import path
from myapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('list/', views.list),
    path('search_name/', views.search_name),
    path('index/',views.index),
    path('post1/',views.post1),
    path('edit1/<int:id>/<str:mode>/', views.edit1),
    path('edit2/<int:id>/', views.edit2),
    path('delete/<int:id>/', views.delete),
```

新增 view functions：

sudo vim myapp/views.py

```python
def edit1(request, id=None, mode=None):
    # print(id)
    # print(mode)
    if mode == "edit":
        cName = request.GET["cName"]
        cSex = request.GET["cSex"]
        cBirthday = request.GET["cBirthday"]
        cEmail = request.GET["cEmail"]
        cPhone = request.GET["cPhone"]
        cAddr = request.GET["cAddr"]
        # print(cName)
        # print(cSex)
        # print(cBirthday)
        # print(cEmail)
        # print(cPhone)
        # print(cAddr)

        # orm:
        update = students.objects.get(id=id)
        update.cName = cName
        update.cSex = cSex
        update.cBirthday = cBirthday
        update.cEmail = cEmail
        update.cPhone = cPhone
        update.cAddr = cAddr
        update.save()
```

```
        # return HttpResponse("修改.......")
        return redirect('/index/')
    elif mode == "load":
        # orm:
        dict_data = students.objects.get(id=id)
        # return HttpResponse("test.......")
        return render(request,"edit1.html", locals())
```

新增 template edit1.html：

sudo vim templates/edit1.html

edit1.html

templates > edit1.html

```
{% extends 'base.html' %}

{% block title %}
<title>學生資料管理系統-修改資料一(GET)</title>
{% endblock title %}

{% block content %}
<h1>學生資料管理系統-修改資料一(GET)</h1>
<a href="/index/"><h3>回首頁</h3></a>
<form action="/edit1/{{dict_data.id}}/edit/" method="get">
    <table>
        <tr>
            <th>姓名</th><td><input type="text" id="cName" name="cName" value="{{ dict_data.cName }}"></td>
        </tr>
        <tr>
            <th>性別</th>
            <td>
                <input type="radio" id="cSex" name="cSex" value="M" {% if dict_data.cSex == "M"%} checked {% endif %} >男
                <input type="radio" id="cSex" name="cSex" value="F" {% if dict_data.cSex == "F"%} checked {% endif %} >女

            </td>
        </tr>
        <tr>
            <th>生日</th><td><input type="date" id="cBirthday" name="cBirthday" value="{{ dict_data.cBirthday }}"></td>
        </tr>
```

```
        </tr>
        <tr>
            <th>信箱</th><td><input type="text" id="cEmail" name="cEmail" value="{{ dict_data.cEmail }}"></td>
        </tr>
        <tr>
            <th>電話</th><td><input type="text" id="cPhone" name="cPhone" value="{{ dict_data.cPhone }}"></td>
        </tr>
        <tr>
            <th>地址</th><td><input type="text" id="cAddr" name="cAddr" value="{{ dict_data.cAddr }}"></td>
        </tr>
        <tr>
            <th colspan="2" style="text-align:center;">
                <input type="submit" name="button" id="button" value="儲存">
                <input type="reset" name="button2" id="button2" value="重設">
            </th>
        </tr>
    </table>
</form>
{% endblock content %}
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

```
System check identified no issues (0 silenced).
June 07, 2021 - 10:59:59
Django version 2.2.18, using settings 'project2.settings'
Starting development server at http://0.0.0.0:8080/
Quit the server with CONTROL-C.
web get
[07/Jun/2021 10:59:59] "GET /edit1/6/load HTTP/1.1" 200 1271
form get
[07/Jun/2021 11:00:06] "GET /edit1/6/edit?cName=bb&cSex=M&cBirthday=2021
b&cAddr=bb&button=%E9%80%81%E5%87%BA HTTP/1.1" 302 0
[07/Jun/2021 11:00:06] "GET /index/ HTTP/1.1" 200 11271
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

➢ 資料更新(使用 GET) (SQL-UPDATE 語法)

新增瀏覽位置(urls.py):

```
# sudo vim project2/urls.py
```

urls.py

ch3_12 > urls.py > ...

```
16  from django.contrib import admin
17  from django.urls import path
18  from myapp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('list/', views.list),
23      path('search_name/', views.search_name),
24      path('index/',views.index),
25      path('post1/',views.post1),
26      path('edit1/<int:id>/<str:mode>/', views.edit1),
27      path('edit2/<int:id>/', views.edit2),
28      path('delete/<int:id>/', views.delete),
29
30
31  ]
```

新增 view functions：

sudo vim myapp/views.py

```
133 def edit2(request, id=None):
134     # print(id)
135     if request.method == "POST":
136         cName = request.POST["cName"]
137         cSex = request.POST["cSex"]
138         cBirthday = request.POST["cBirthday"]
139         cEmail = request.POST["cEmail"]
140         cPhone = request.POST["cPhone"]
141         cAddr = request.POST["cAddr"]
142         # print(cName)
143         # print(cSex)
144         # print(cBirthday)
145         # print(cEmail)
146         # print(cPhone)
147         # print(cAddr)
148
149         # orm:
150         update = students.objects.get(id=id)
151         update.cName = cName
152         update.cSex = cSex
153         update.cBirthday = cBirthday
154         update.cEmail = cEmail
155         update.cPhone = cPhone
156         update.cAddr = cAddr
157         update.save()
```

```
159         # return HttpResponse("修改.......")
160         return redirect('/index/')
161     else:
162         # orm:
163         dict_data = students.objects.get(id=id)
164         print(dict_data)
165         # return HttpResponse("test.......")
166         return render(request,"edit2.html", locals())
```

測試結果：

sudo python3 manage.py runserver 0.0.0.0:8080

資料表修改(二)
192.168.57.2
不安全 | 192.168.57.236:8080/ec
應用程式
192.168.57.236
MSN
Outlo
student 資料表修改(二)
姓名
bb33
性別F
男生
女生
生日
2021/06/12
郵件帳號
bb33@gmial.com
電話
0999933
地址
pp
送出
重設
顯示所有資料
192.168.57.236 / localhost / my
不安全 | 192.168.57.236:8080/index/
應用程式
192.168.57.236
MSN
Outlook
Gmail
Yahoo 奇摩電子信箱
Google 翻譯
Yahoo奇摩
YouTu
顯示 student 資料表所有資料
新增資料
編號
姓名
性別
生日
郵件帳號
電話
地址
編輯
6
bb33
M
2021年6月12日
bb33@gmial.com
0999933
pp
編輯一 編輯二 刪除

3-13 Django 使用 ORM vs SQL 語法

➢ 建立環境

```
● 建立 Django 專案:

# cd ~/dvds

# source ./dvds/dvdsenv/bin/activate (啟動虛擬環境)

# django-admin startproject DB (建立 DB 專案)

# cd DB

# tree
```

➢ ORM 欄位與引數說明

每個欄位有一些特有的引數，例如，CharField 需要 max_length 引數來指定 VARCHAR 資料庫欄位的大小。還有一些適用於所有欄位的通用引數。這些引數在文件中有詳細定義，這裡我們只簡單介紹一些最常用的：

欄位	說明
CharField	字串欄位，用於較短的字串 要求必須有一個引數 maxlength
IntegerField	用於儲存一個整數
FloatField	double ex: models. FloatField(null=True)
DecimalField	一個浮點數，必須提供兩個引數 max_digits：總位數(不包括小數點和符號) decimal_places：小數位數 ex: 要儲存最大值為 999 (小數點後儲存 2 位)，定義欄位: models.DecimalField(..., max_digits=5, decimal_places=2)
AutoField	一個 IntegerField，新增記錄時它會自動增長。 EX: my_id=models.AutoField(primary_key=True) 如果你不指定主鍵的話，系統會自動新增一個主鍵欄位到你的 model。
BooleanField	A true/false field。admin 用 checkbox 來表示此類欄位。

TextField	一個容量很大的文字欄位。
EmailField	一個帶有檢查 Email 合法性的 CharField，不接受 maxlength 引數。
DateField	一個日期欄位。共有下列額外的可選引數： Argument：描述 auto_now：當物件被儲存時(更新或者新增都行)，自動將該欄位的值設定為當前時間。通常用於表示 "last-modified" 時間戳。 auto_now_add：當物件首次被建立時，自動將該欄位的值設定為當前時間，通常用於表示物件建立時間。
DateTimeField	一個日期時間欄位。 類似 DateField 支援同樣的附加選項。
ImageField	類似 FileField，不過要校驗上傳物件是否是一個合法圖片。
FileField	一個檔案上傳欄位

引數	說明
null	如果為 True，Django 將用 NULL 來在資料庫中儲存空值。 預設值是 False。
blank	如果為 True，該欄位允許不填。預設為 False。
default	欄位的預設值。可以是一個值或者可呼叫物件。如果可呼叫，每有新物件被建立它都會被呼叫，如果你的欄位沒有設定可以為空，那麼將來如果我們後新增一個欄位，這個欄位就要給一個 default 值。
primary_key	如果為 True，那麼這個欄位就是模型的主鍵。如果你沒有指定任何一個欄位的 primary_key=True，Django 就會自動新增一個 IntegerField 欄位做為主鍵，所以除非你想覆蓋預設的主鍵行為，否則沒必要設定任何一個欄位的 primary_key=True。
unique	如果該值設定為 True，這個資料欄位的值在整張表中必須是唯一的。
auto_now_add	配置 auto_now_add=True，建立資料記錄的時候會把當前時間新增到資料庫。
auto_now	配置上 auto_now=True，每次更新資料記錄的時候會更新該欄位，標識這條記錄最後一次的修改時間。

參考文獻:

https://iter01.com/432964.html

➢ 建立資料表

名稱	欄位名稱	資料型態	屬性	NULL	其他
座號	cID	TINYINT(2)	USIGNED ZEROFILL	否	主鍵，auto_increment
姓名	cName	VARCHAR(20)		否	
性別	cSex	ENUM('F','M')		否	預設為 F (female, 女性)
生日	cBirthday	DATE		否	
電子郵件	cEmail	VARCHAR(100)		是	
電話	cPhone	VARCHAR(50)		是	
住址	cAddr	VARCHAR(255)		是	

#vim DBapp/models.py

```
from django.db import models

# Create your models here.
class students(models.Model):
    cID = models.AutoField(primary_key=True)
    cName = models.CharField(max_length=20, blank=False)
    cSex =  models.CharField(max_length=1, blank=False, default='F')
    cBirthday = models.DateTimeField(auto_now_add=False)
    cEmail = models.CharField(max_length=100, blank=False)
    cPhone = models.CharField(max_length=50, blank=False)
    cAddr = models.CharField(max_length=255, blank=False)
    cHeight = models.IntegerField(blank=True, null=True)
    cWeight = models.IntegerField(blank=True, null=True)

#一個學生有多個成績(one to many)(students to scorelist)
class scorelist(models.Model):
    id = models.AutoField(primary_key=True)
    cID = models.ForeignKey('students', on_delete=models.CASCADE, null=True)
    course = models.CharField(max_length=20, blank=False)
    score = models.IntegerField(blank=False)
```

sudo python3 manage.py makemigrations

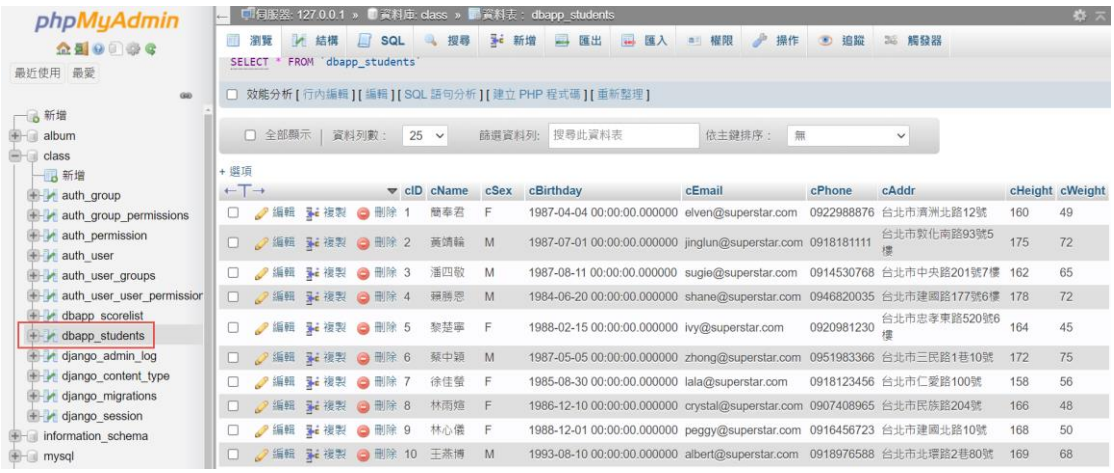
(建立資料庫和 Django 間的中介檔)

sudo python3 manage.py migrate

(同步更新資料庫內容)

```
(dvds) c:\dvds\DB>python manage.py makemigrations
Migrations for 'DBapp':
  DBapp\migrations\0001_initial.py
    - Create model students

(dvds) c:\dvds\DB>python manage.py migrate
System check identified some issues:

WARNINGS:
?: (mysql.W002) MariaDB Strict Mode is not set for database connection 'default'
        HINT: MariaDB's Strict Mode fixes many data integrity problems in MariaDB, such as data truncation upon insertio
n, by escalating warnings into errors. It is strongly recommended you activate it. See: https://docs.djangoproject.com/e
n/3.2/ref/databases/#mysql-sql-mode
Operations to perform:
  Apply all migrations: DBapp, admin, auth, contenttypes, sessions
Running migrations:
  Applying DBapp.0001_initial... OK
```

➢ 匯入資料表資料



➢ 設定 Django 平台

- url 配置檔

新增瀏覽位置(urls.py):

```
# sudo vim DB/urls.py
```

```
from django.urls import path
from DBapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('test/', views.test),

]
```

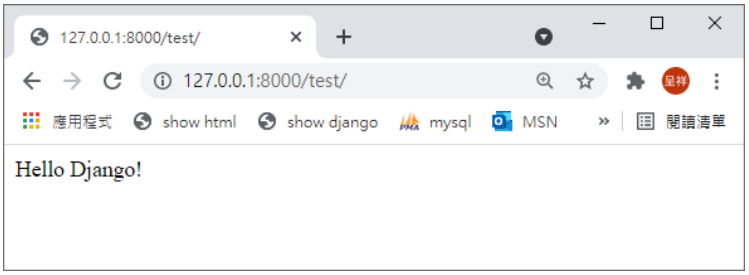
- 新增 view functions

```
# sudo vim DBapp/views.py
```

```
from django.shortcuts import render
from django.http import HttpResponse

def test(request):
    return HttpResponse("Hello Django!")
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

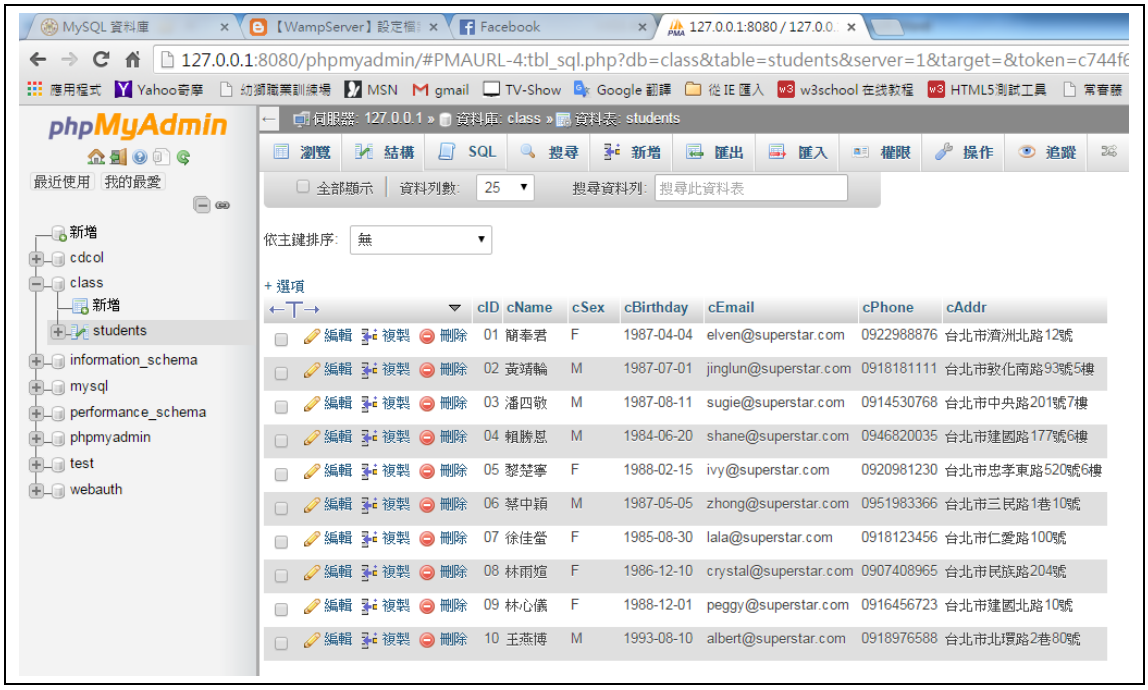


➢ 查詢資料

選取所有的欄位：

SQL：

```
SELECT * FROM `students`;
```



ORM：

```
datas = students.objects.all()
```

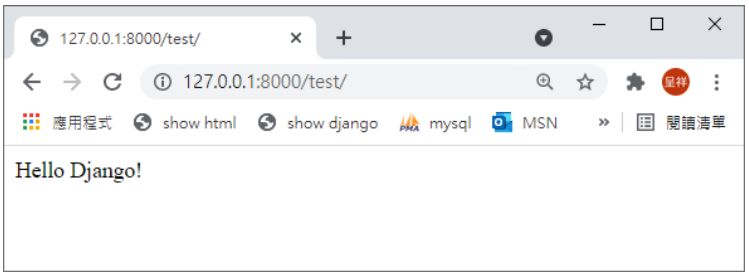
- 修改 view functions

```
# sudo vim DBapp/views.py
```

```
def test(request):
    datas = students.objects.all()
    for data in datas:
        print('%s %s %s %s %s %s %s' % (data.cID, data.cName, data.cSex ,data.cBirthday, data.
        cEmail, data.cPhone, data.cAddr))
    return HttpResponse("Hello Django!")
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```



```
System check identified no issues (0 silenced).
August 03, 2021 - 15:17:27
Django version 3.2.4, using settings 'DB.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
1 簡奉君 F 1987-04-04 00:00:00+00:00 elven@superstar.com 0922988876 台北市濟洲北路12號
2 黃靖輪 M 1987-07-01 00:00:00+00:00 jinglun@superstar.com 0918181111 台北市敦化南路93號5樓
3 潘四敬 M 1987-08-11 00:00:00+00:00 sugie@superstar.com 0914530768 台北市中央路201號7樓
4 賴勝恩 M 1984-06-20 00:00:00+00:00 shane@superstar.com 0946820035 台北市建國路177號6樓
5 黎楚寧 F 1988-02-15 00:00:00+00:00 ivy@superstar.com 0920981230 台北市忠孝東路520號6樓
6 蔡中穎 M 1987-05-05 00:00:00+00:00 zhong@superstar.com 0951983366 台北市三民路1巷10號
7 徐佳螢 F 1985-08-30 00:00:00+00:00 lala@superstar.com 0918123456 台北市仁愛路100號
8 林雨煊 F 1986-12-10 00:00:00+00:00 crystal@superstar.com 0907408965 台北市民族路204號
9 林心儀 F 1988-12-01 00:00:00+00:00 peggy@superstar.com 0916456723 台北市建國北路10號
10 王燕博 M 1993-08-10 00:00:00+00:00 albert@superstar.com 0918976588 台北市北環路2巷80號
[03/Aug/2021 15:17:27] "GET /test/ HTTP/1.1" 200 13
```

ORM：

```
datas = students.objects.values('cID','cName', 'cEmail')


透過 QuerySet 的.values()方法，將 QuerySet 轉換為 ValuesQuerySet


for data in datas:

  print('%s %s %s' % (data['cID'], data['cName'], data['cEmail']))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
1 簡奉君 elven@superstar.com
2 黃靖輪 jinglun@superstar.com
3 潘四敬 sugie@superstar.com
4 賴勝恩 shane@superstar.com
5 黎楚寧 ivy@superstar.com
6 蔡中穎 zhong@superstar.com
7 徐佳螢 lala@superstar.com
8 林雨煊 crystal@superstar.com
9 林心儀 peggy@superstar.com
10 王燕博 albert@superstar.com
[20/Aug/2021 11:06:13] "GET /test/ HTTP/1.1" 200 4
```

SELECT DISTINCT:去除重複資料顯示一筆：

Ex:想知道全班有幾種性別

SQL：

```
SELECT DISTINCT `cSex` FROM `students`;
```

ORM：

```
datas = students.objects.values('cSex').distinct()

print(datas)


datas = students.objects.values('cSex').distinct().count()

print(datas)
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
<QuerySet [{'cSex': 'F'}, {'cSex': 'M'}]>
2
```

- values vs values_list

values

Returns a QuerySet that returns dictionaries, rather than model instances, when used as an iterable.

Ex:

list(Article.objects.values_list('id', flat=True))

flat=True will remove the tuples and return the list

[1, 2, 3, 4, 5, 6]

value_list

This is similar to values() except that instead of returning dictionaries, it returns

tuples when iterated over.

Ex:

list(Article.objects.values('id'))

[{'id':1}, {'id':2}, {'id':3}, {'id':4}, {'id':5}, {'id':6}]

- WHERE：設定篩選條件

在查詢資料時，並不是每一次都要顯示所有的內容。我們可能會為顯示的資料設定一些條件，來篩選顯示的內容，這就是 WHERE 指令的功能。

WHERE 基本語法：

WHERE 的基本語法格式為：

SELECT 欄位名稱 FROM 資料表名稱 WHERE 條件敘述句;

Ex:想要由 students 資料表中挑出 cID 的學生資料

SQL:

```
SELECT * FROM `students` WHERE `cID `=1;
```

ORM：

```
獲取單條資料（有且僅有一條，id 唯一）(回傳物件)

datas = students.objects.get(cID=1)

or

datas = students.objects.filter(cID=1)(回傳 QuerySet)

for data in datas:

    print('%s %s %s %s %s %s %s' % (data.cID, data.cName,

data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
1 簡奉君 F 1987-04-04 00:00:00+00:00 elven@superstar.com 0922988876 台北市濟洲北路12號
[03/Aug/2021 16:44:39] "GET /test/ HTTP/1.1" 200 13
```

Ex:想要由 students 資料表中挑出所有男性的資料

SQL:

```
SELECT * FROM `students` WHERE `cSex`='M';
```

+ 選項

←T→	cID	cName	cSex	cBirthday	cEmail	cPhone	cAddr
編輯 複製 刪除	02	黃靖輪	M	1987-07-01	jinglun@superstar.com	0918181111	台北市敦化南路93號5樓
編輯 複製 刪除	03	潘四敬	M	1987-08-11	sugie@superstar.com	0914530768	台北市中央路201號7樓
編輯 複製 刪除	04	賴勝恩	M	1984-06-20	shane@superstar.com	0946820035	台北市建國路177號6樓
編輯 複製 刪除	06	蔡中穎	M	1987-05-05	zhong@superstar.com	0951983366	台北市三民路1巷10號
編輯 複製 刪除	10	王燕博	M	1993-08-10	albert@superstar.com	0918976588	台北市北環路2巷80號

ORM：

```
datas = students.objects.filter(cSex='M')

for data in datas:

        print('%s %s %s %s %s %s %s' % (data.cID, data.cName,

data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
2 黃靖輪 M 1987-07-01 00:00:00+00:00 jinglun@superstar.com 0918181111 台北市敦化南路93號5樓
3 潘四敬 M 1987-08-11 00:00:00+00:00 sugie@superstar.com 0914530768 台北市中央路201號7樓
4 賴勝恩 M 1984-06-20 00:00:00+00:00 shane@superstar.com 0946820035 台北市建國路177號6樓
6 蔡中穎 M 1987-05-05 00:00:00+00:00 zhong@superstar.com 0951983366 台北市三民路1巷10號
10 王燕博 M 1993-08-10 00:00:00+00:00 albert@superstar.com 0918976588 台北市北環路2巷80號
[03/Aug/2021 15:28:35] "GET /test/ HTTP/1.1" 200 13
```

比較運算子：

=	!=	<>	<
>	<=	>=	IS NULL

AND、OR、NOT：連接多個條件式：

Ex:想要由 students 資料表中找出座號大於 5 的男生

SQL:

```
SELECT * FROM `students` WHERE `cID`>5 AND `cSex`='M';
```

+ 選項

	cID	cName	cSex	cBirthday	cEmail	cPhone	cAddr
編輯 複製 刪除	06	蔡中穎	M	1987-05-05	zhong@superstar.com	0951983366	台北市三民路1巷10號
編輯 複製 刪除	10	王燕博	M	1993-08-10	albert@superstar.com	0918976588	台北市北環路2巷80號

```
# 大於小於
    .filter(id__gt=1)              ->      > 1
    .filter(id=1)                  ->      = 1
    .filter(id__lt=1)              ->      < 1
    .filter(id__lte=1)             ->      <= 1
    .filter(id__gte=1)             ->      >= 1
    .exclude(id__gt=1)             ->      != 1  exclude 除了...與filter相反
    .filter(id__gt=1, id__lt=10)   ->      1< x <10
```

ORM:

```
# 比較，gt:>，gte:>=，lt:<，lte:<=

# isnull：isnull=True 為空，isnull=False 不為空


datas = students.objects.filter(cID__gt=5,cSex='M')

for data in datas:

   print('%s  %s  %s  %s  %s  %s  %s'  %  (data.cID,  data.cName,

data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
6 蔡中穎 M 1987-05-05 00:00:00+00:00 zhong@superstar.com 0951983366 台北市三民路1巷10號
10 王燕博 M 1993-08-10 00:00:00+00:00 albert@superstar.com 0918976588 台北市北環路2巷80號
[03/Aug/2021 15:40:05] "GET /test/ HTTP/1.1" 200 13
```

Ex:想要由 students 資料表中找出座號不大於 5 的男生

SQL:

```
SELECT * FROM `students` WHERE !(`cID`>5 AND `cSex`='M');
```

+ 選項

	cID	cName	cSex	cBirthday	cEmail	cPhone	cAddr
編輯 複製 刪除	02	黃靖輪	M	1987-07-01	jinglun@superstar.com	0918181111	台北市敦化南路93號5樓
編輯 複製 刪除	03	潘四敬	M	1987-08-11	sugie@superstar.com	0914530768	台北市中央路201號7樓
編輯 複製 刪除	04	賴勝恩	M	1984-06-20	shane@superstar.com	0946820035	台北市建國路177號6樓

Q 表達式：

通過 objects.filter 傳入多個條件參數時，多個條件之間默認爲「與」的關係，如果想要變成「或」、「非」等其他關係，就可以使用 Q 表達式。Q 表達式可以使用 &（AND）、|（OR）、~（NOT）等運算符。

```
    from django.db.models import Q       # 匯入Q模組
# 方式一：
    .filter( Q(id__gt=10) )                    ->
    .filter( Q(id=8) | Q(id__gt=10) )          -> or
    .filter( Q( Q(id=8) | Q(id__gt=10) ) & Q(caption='root') )  -> and, or
```

例項：查詢作者姓名中包含 方/少/偉/3字，書名不包含偉，並且出版社地址以山西開頭的書

```
book=models.Book.objects.filter(
                        Q(
                            Q(author__name__contains='方') |
                            Q(author__name__contains='少') |
                            Q(author__name__contains='偉') |
                            Q(title__icontains='偉')
                        ) &
                        Q(publish__addr__contains='山西')
                    ).values('title')
```

注意：Q查詢和非Q查詢混合使用，非Q查詢一定要放在Q查詢後面

ORM:

```
from django.db.models import Q

    datas = students.objects.filter( ~Q(cID__gt=5),cSex='M')


for data in datas:

   print('%s  %s  %s  %s  %s  %s  %s'  %  (data.cID,  data.cName,

data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
Django version 3.2.4, using settings 'DB.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
2 黃靖輪 M 1987-07-01 00:00:00+00:00 jinglun@superstar.com 0918181111 台北市敦化南路93號5樓
3 潘四敬 M 1987-08-11 00:00:00+00:00 sugie@superstar.com 0914530768 台北市中央路201號7樓
4 賴勝恩 M 1984-06-20 00:00:00+00:00 shane@superstar.com 0946820035 台北市建國路177號6樓
[03/Aug/2021 15:44:11] "GET /test/ HTTP/1.1" 200 13
```

Ex:想要由 students 資料表中找出座號等於 1 號或座號大於等於 9 號的人

SQL:

```
SELECT * FROM `dbapp_students` WHERE `cID`=1 OR `cID`>=9;
```

ORM:

```
from django.db.models import Q
    datas = students.objects.filter(
        Q(cID=1) | Q(cID__gte=9)
        )
for data in datas:
    print('%s %s %s %s %s %s %s' % (data.cID, data.cName,
data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
1 簡奉君 F 1987-04-04 00:00:00+00:00 elven@superstar.com 0922988876 台北市濟洲北路12號
9 林心儀 F 1988-12-01 00:00:00+00:00 peggy@superstar.com 0916456723 台北市建國北路10號
10 王燕博 M 1993-08-10 00:00:00+00:00 albert@superstar.com 0918976588 台北市北環路2巷80號
[03/Aug/2021 15:56:43] "GET /test/ HTTP/1.1" 200 13
```

設定數值篩選範圍：

EX:由 students 資料表中找出座號大於等於 4 且小於等於 6 的學生資料

SQL

```
SELECT * FROM `students` WHERE `cID` BETWEEN 4 AND 6;
```

+ 選項

	cID	cName	cSex	cBirthday	cEmail	cPhone	cAddr
編輯 複製 刪除	04	賴勝恩	M	1984-06-20	shane@superstar.com	0946820035	台北市建國路177號6樓
編輯 複製 刪除	05	黎楚寧	F	1988-02-15	ivy@superstar.com	0920981230	台北市忠孝東路520號6樓
編輯 複製 刪除	06	蔡中穎	M	1987-05-05	zhong@superstar.com	0951983366	台北市三民路1巷10號

```
# 範圍range
    .filter(id__range=[1,3])        ->        [1~3]    bettwen + and
```

ORM:

```
datas = students.objects.filter(cID__range=[4,6])

for data in datas:

    print('%s  %s  %s  %s  %s  %s  %s' % (data.cID,  data.cName,

data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
4 賴勝恩 M 1984-06-20 00:00:00+00:00 shane@superstar.com 0946820035 台北市建國路177號6樓
5 黎楚寧 F 1988-02-15 00:00:00+00:00 ivy@superstar.com 0920981230 台北市忠孝東路520號6樓
6 蔡中穎 M 1987-05-05 00:00:00+00:00 zhong@superstar.com 0951983366 台北市三民路1巷10號
[03/Aug/2021 16:00:34] "GET /test/ HTTP/1.1" 200 13
```

- IN：指定多個篩選值

Ex:想要由 students 資料表中找出座號為 1,3,5,7,9 的學生資料：

SQL:

```
SELECT * FROM `students` WHERE `students`.`cID` IN (1,3,5,7,9)
```

+ 選項

	cID	cName	cSex	cBirthday	cEmail	cPhone	cAddr
編輯 複製 刪除	01	簡奉君	F	1987-04-04	elven@superstar.com	0922988876	台北市濟洲北路12號
編輯 複製 刪除	03	潘四敬	M	1987-08-11	sugie@superstar.com	0914530768	台北市中央路201號7樓
編輯 複製 刪除	05	黎楚寧	F	1988-02-15	ivy@superstar.com	0920981230	台北市忠孝東路520號6樓
編輯 複製 刪除	07	徐佳瑩	F	1985-08-30	lala@superstar.com	0918123456	台北市仁愛路100號
編輯 複製 刪除	09	林心儀	F	1988-12-01	peggy@superstar.com	0916456723	台北市建國北路10號

```
# 範圍in
    .filter(id__in=[1,2,3])        ->        in [1,2,3]
    .exclude(id__in=[1,2,3])       ->        in [1,2,3]
```

ORM:

```
datas = students.objects.filter(cID__in=[1,3,5,7,9])

for data in datas:

    print('%s %s %s %s %s %s %s' % (data.cID, data.cName,

data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
1 簡奉君 F 1987-04-04 00:00:00+00:00 elven@superstar.com 0922988876 台北市濟洲北路12號
3 潘四敬 M 1987-08-11 00:00:00+00:00 sugie@superstar.com 0914530768 台北市中央路201號7樓
5 黎楚寧 F 1988-02-15 00:00:00+00:00 ivy@superstar.com 0920981230 台北市忠孝東路520號6樓
7 徐佳螢 F 1985-08-30 00:00:00+00:00 lala@superstar.com 0918123456 台北市仁愛路100號
9 林心儀 F 1988-12-01 00:00:00+00:00 peggy@superstar.com 0916456723 台北市建國北路10號
[03/Aug/2021 16:06:42] "GET /test/ HTTP/1.1" 200 13
```

- LIKE：設定字串比對的篩選值

SQL:

萬用字元	說明	範例
_(底線)	任何單一字元，一個中文字也代表一個字元。	條件：「LIKE '文_閣'」，「文淵閣」符合，「文藏經閣」即不符合。
%	任何含有零或多個字元的字串。	條件：「LIKE '文%閣'」，「文淵閣」符合，「文藏經閣」都符合。

ORM

語法	等同於 sql 語法
__exact	like 'aaa'
__iexact	忽略大小寫 ilike 'aaa'
__contains	包含 like '%aaa%'
__icontains	包含，忽然大小寫我喜歡 '%aaa%'，但對於 sqlite 來說，包含的作用等同於 icontains。
__startswith	以…開頭
__istartswith	以…開頭忽略大小寫
__endswith	以…结尾
__iendswith	以…結尾，忽略大小寫
__range	在…範圍內
__year	日期字段的年份
__month	日期字段的月份
__day	日期字段的日

Ex:想要由 students 資料表中，出電話號碼是「0918」開頭的學生資料

SQL:

```
SELECT * FROM `students` WHERE `cPhone` LIKE '0918%';
```

+ 選項

				cID	cName	cSex	cBirthday	cEmail	cPhone	cAddr
編輯	複製	刪除		02	黃靖輪	M	1987-07-01	jinglun@superstar.com	0918181111	台北市敦化南路93號5樓
編輯	複製	刪除		07	徐佳螢	F	1985-08-30	lala@superstar.com	0918123456	台北市仁愛路100號
編輯	複製	刪除		10	王燕博	M	1993-08-10	albert@superstar.com	0918976588	台北市北環路2巷80號

```
# 包含、開頭、結尾 __startswith, istartswith, endswith, iendswith
    .filter(name__contains="ven")
    .filter(name__icontains="ven")     # i 忽略大小寫
```

ORM:

```
datas = students.objects.filter(cPhone__istartswith='0918')

for data in datas:

    print('%s %s %s %s %s %s %s' % (data.cID, data.cName,

data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
2 黃靖輪 M 1987-07-01 00:00:00+00:00 jinglun@superstar.com 0918181111 台北市敦化南路93號5樓
7 徐佳螢 F 1985-08-30 00:00:00+00:00 lala@superstar.com 0918123456 台北市仁愛路100號
10 王燕博 M 1993-08-10 00:00:00+00:00 albert@superstar.com 0918976588 台北市北環路2巷80號
[05/Aug/2021 13:38:27] "GET /test/ HTTP/1.1" 200 13
```

Ex: 想要由 students 資料表中，找出學生的地址中有「建國」這個字的資料。

SQL:

```
SELECT * FROM `students` WHERE `cAddr` LIKE '%建國%'
```

ORM:

```
datas = students.objects.filter(cAddr__contains='建國')

for data in datas:

    print('%s %s %s %s %s %s %s' % (data.cID, data.cName,

data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
4 賴勝恩 M 1984-06-20 00:00:00+00:00 shane@superstar.com 0946820035 台北市建國路177號6樓
9 林心儀 F 1988-12-01 00:00:00+00:00 peggy@superstar.com 0916456723 台北市建國北路10號
[05/Aug/2021 13:41:09] "GET /test/ HTTP/1.1" 200 13
```

- ORDER BY：設定查詢結果的排序

ORDER BY 的功能是用來設定欄位，進行排序查詢結果，它的基本語法如下：

```
SELECT 欄位名稱
FROM 資料表名稱
ORDER BY 指定排序的欄位 排序方式
```

其中排序方式有二種：

1. **ASC(ascending)**：遞增排序，由小排到大，也是未指定時預設的排序方法。
2. **DESC(Descending)**：遞減排序，由大排到小。

Ex: 想要由 students 資料表所有同學的資料依生日遞減排序

SQL:

```
SELECT * FROM `students` ORDER BY `cBirthday` DESC;
```

+ 選項

	cID	cName	cSex	cBirthday 1	cEmail	cPhone	cAddr
編輯 複製 刪除	10	王燕博	M	1993-08-10	albert@superstar.com	0918976588	台北市北環路2巷80號
編輯 複製 刪除	09	林心儀	F	1988-12-01	peggy@superstar.com	0916456723	台北市建國北路10號
編輯 複製 刪除	05	黎楚寧	F	1988-02-15	ivy@superstar.com	0920981230	台北市忠孝東路520號6樓
編輯 複製 刪除	03	潘四敬	M	1987-08-11	sugie@superstar.com	0914530768	台北市中央路201號7樓
編輯 複製 刪除	02	黃靖輪	M	1987-07-01	jinglun@superstar.com	0918181111	台北市敦化南路93號5樓
編輯 複製 刪除	06	蔡中穎	M	1987-05-05	zhong@superstar.com	0951983366	台北市三民路1巷10號
編輯 複製 刪除	01	簡奉君	F	1987-04-04	elven@superstar.com	0922988876	台北市濟洲北路12號
編輯 複製 刪除	08	林雨煊	F	1986-12-10	crystal@superstar.com	0907408965	台北市民族路204號
編輯 複製 刪除	07	徐佳瑩	F	1985-08-30	lala@superstar.com	0918123456	台北市仁愛路100號
編輯 複製 刪除	04	賴勝恩	M	1984-06-20	shane@superstar.com	0946820035	台北市建國路177號6樓

```
# 排序
    .filter(name='seven').order_by('id')     ->  asc
    .filter(name='seven').order_by('-id')    ->  desc
```

ORM:

遞增排序:

```
datas = students.objects.all().order_by('cBirthday')
```

遞減排序:

```
datas = students.objects.all().order_by('-cBirthday')

for data in datas:

   print('%s  %s  %s  %s  %s  %s  %s'  %  (data.cID,  data.cName,

data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
10 王燕博 M 1993-08-10 00:00:00+00:00 albert@superstar.com 0918976588 台北市北環路2巷80號
9 林心儀 F 1988-12-01 00:00:00+00:00 peggy@superstar.com 0916456723 台北市建國北路10號
5 黎楚寧 F 1988-02-15 00:00:00+00:00 ivy@superstar.com 0920981230 台北市忠孝東路520號6樓
3 潘四敬 M 1987-08-11 00:00:00+00:00 sugie@superstar.com 0914530768 台北市中央路201號7樓
2 黃靖輪 M 1987-07-01 00:00:00+00:00 jinglun@superstar.com 0918181111 台北市敦化南路93號5樓
6 蔡中穎 M 1987-05-05 00:00:00+00:00 zhong@superstar.com 0951983366 台北市三民路1巷10號
1 簡奉君 F 1987-04-04 00:00:00+00:00 elven@superstar.com 0922988876 台北市濟洲北路12號
8 林雨煊 F 1986-12-10 00:00:00+00:00 crystal@superstar.com 0907408965 台北市民族路204號
7 徐佳瑩 F 1985-08-30 00:00:00+00:00 lala@superstar.com 0918123456 台北市仁愛路100號
4 賴勝恩 M 1984-06-20 00:00:00+00:00 shane@superstar.com 0946820035 台北市建國路177號6樓
[05/Aug/2021 13:55:19] "GET /test/ HTTP/1.1" 200 13
```

Ex: 想要由 students 資料表所有同學的資料依性別遞增排序，再依生日遞減排序

SQL:

```
SELECT * FROM `students` ORDER BY `cSex` ASC, `cBirthday` DESC;
```

+ 選項

	cID	cName	cSex 1	cBirthday 2	cEmail	cPhone	cAddr
編輯 複製 刪除	09	林心儀	F	1988-12-01	peggy@superstar.com	0916456723	台北市建國北路10號
編輯 複製 刪除	05	黎楚寧	F	1988-02-15	ivy@superstar.com	0920981230	台北市忠孝東路520號6樓
編輯 複製 刪除	01	簡奉君	F	1987-04-04	elven@superstar.com	0922988876	台北市濟洲北路12號
編輯 複製 刪除	08	林雨煊	F	1986-12-10	crystal@superstar.com	0907408965	台北市民族路204號
編輯 複製 刪除	07	徐佳螢	F	1985-08-30	lala@superstar.com	0918123456	台北市仁愛路100號
編輯 複製 刪除	10	王燕博	M	1993-08-10	albert@superstar.com	0918976588	台北市北環路2巷80號
編輯 複製 刪除	03	潘四敬	M	1987-08-11	sugie@superstar.com	0914530768	台北市中央路201號7樓
編輯 複製 刪除	02	黃靖輪	M	1987-07-01	jinglun@superstar.com	0918181111	台北市敦化南路93號5樓
編輯 複製 刪除	06	蔡中穎	M	1987-05-05	zhong@superstar.com	0951983366	台北市三民路1巷10號
編輯 複製 刪除	04	賴勝恩	M	1984-06-20	shane@superstar.com	0946820035	台北市建國路177號6樓

ORM:

```
datas = students.objects.all().order_by('cSex', '-cBirthday')

for data in datas:

   print('%s  %s  %s  %s  %s  %s  %s'  %  (data.cID,  data.cName,

data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
9 林心儀 F 1988-12-01 00:00:00+00:00 peggy@superstar.com 0916456723 台北市建國北路10號
5 黎楚寧 F 1988-02-15 00:00:00+00:00 ivy@superstar.com 0920981230 台北市忠孝東路520號6樓
1 簡奉君 F 1987-04-04 00:00:00+00:00 elven@superstar.com 0922988876 台北市濟洲北路12號
8 林雨煊 F 1986-12-10 00:00:00+00:00 crystal@superstar.com 0907408965 台北市民族路204號
7 徐佳螢 F 1985-08-30 00:00:00+00:00 lala@superstar.com 0918123456 台北市仁愛路100號
10 王燕博 M 1993-08-10 00:00:00+00:00 albert@superstar.com 0918976588 台北市北環路2巷80號
3 潘四敬 M 1987-08-11 00:00:00+00:00 sugie@superstar.com 0914530768 台北市中央路201號7樓
2 黃靖輪 M 1987-07-01 00:00:00+00:00 jinglun@superstar.com 0918181111 台北市敦化南路93號5樓
6 蔡中穎 M 1987-05-05 00:00:00+00:00 zhong@superstar.com 0951983366 台北市三民路1巷10號
4 賴勝恩 M 1984-06-20 00:00:00+00:00 shane@superstar.com 0946820035 台北市建國路177號6樓
[05/Aug/2021 14:18:13] "GET /test/ HTTP/1.1" 200 13
```

- LIMIT：設定查詢顯示的筆數

LIMIT 可以設定查詢後由哪一筆開始顯示，並顯示多少筆數，它的基本語法如下：

SELECT 欄位名稱
FROM 資料表名稱
LIMIT 開始顯示的筆數, 顯示多少筆資料

LIMIT 是由查詢後的結果再進行擷取資料的動作，如果與 ORDER BY 進行排序搭配可以輕易取得最前的 10 筆資料或是最後的 10 筆資料的結果。也因為如此，LIMIT 在使用時必須放置在 ORDER BY 之後。

SQL:

```
SELECT * FROM `students` ORDER BY `students`.`cSex` ASC,
`students`.`cBirthday` DESC LIMIT 5


SELECT * FROM `students` LIMIT 2
SELECT * FROM `students` LIMIT 0,2
SELECT * FROM `students` LIMIT 1,2
SELECT * FROM `students` LIMIT 4,2
SELECT * FROM `students` LIMIT 3,4
```

```
# 切片
    .all()[10:20]
    .all()[::2]
    .all()[6]     # 索引
```

ORM:

```
datas =
students.objects.all().order_by('cSex', '-cBirthday')[:5]
for data in datas:
   print('%s %s %s %s %s %s %s' % (data.cID, data.cName,
data.cSex ,data.cBirthday, data.cEmail, data.cPhone, data.cAddr))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
9 林心儀 F 1988-12-01 00:00:00+00:00 peggy@superstar.com 0916456723 台北市建國北路10號
5 黎楚寧 F 1988-02-15 00:00:00+00:00 ivy@superstar.com 0920981230 台北市忠孝東路520號6樓
1 簡奉君 F 1987-04-04 00:00:00+00:00 elven@superstar.com 0922988876 台北市濟洲北路12號
8 林雨媗 F 1986-12-10 00:00:00+00:00 crystal@superstar.com 0907408965 台北市民族路204號
7 徐佳螢 F 1985-08-30 00:00:00+00:00 lala@superstar.com 0918123456 台北市仁愛路100號
[05/Aug/2021 14:28:17] "GET /test/ HTTP/1.1" 200 13
```

```
datas = students.objects.all()[:2]
```

```
datas = students.objects.all()[0:2]
```

```
Quit the server with CTRL-BREAK.
1 簡奉君 F 1987-04-04 00:00:00+00:00 elven@superstar.com 0922988876 台北市濟洲北路12號
2 黃靖輪 M 1987-07-01 00:00:00+00:00 jinglun@superstar.com 0918181111 台北市敦化南路93號5樓
[05/Aug/2021 14:31:50] "GET /test/ HTTP/1.1" 200 13
```

```
datas = students.objects.all()[1:3]
```

```
2 黃靖輪 M 1987-07-01 00:00:00+00:00 jinglun@superstar.com 0918181111 台北市敦化南路93號5樓
3 潘四敬 M 1987-08-11 00:00:00+00:00 sugie@superstar.com 0914530768 台北市中央路201號7樓
```

```
datas = students.objects.all()[4:6]
```

```
Quit the server with CTRL-BREAK.
5 黎楚寧 F 1988-02-15 00:00:00+00:00 ivy@superstar.com 0920981230 台北市忠孝東路520號6樓
6 蔡中穎 M 1987-05-05 00:00:00+00:00 zhong@superstar.com 0951983366 台北市三民路1巷10號
```

```
datas = students.objects.all()[3:7]
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
4 賴勝恩 M 1984-06-20 00:00:00+00:00 shane@superstar.com 0946820035 台北市建國路177號6樓
5 黎楚寧 F 1988-02-15 00:00:00+00:00 ivy@superstar.com 0920981230 台北市忠孝東路520號6樓
6 蔡中穎 M 1987-05-05 00:00:00+00:00 zhong@superstar.com 0951983366 台北市三民路1巷10號
7 徐佳螢 F 1985-08-30 00:00:00+00:00 lala@superstar.com 0918123456 台北市仁愛路100號
[05/Aug/2021 14:34:59] "GET /test/ HTTP/1.1" 200 13
```

- 統計函式

MySQL 的 SQL 語法還提供許多統計函式，可以總計出一些資料表中的彙整資料。為了以下的範例說明，我們要在「class」資料庫中再加入一個儲存成績的資料表：「scorelist」，以下是這個資料表欄位的規劃：

名稱	欄位	資料型態	屬性	NULL	其他
編號	id	TINYINT(4)	USIGNED	否	主鍵，auto_increment
座號	cID	TINYINT(2)	USIGNED ZEROFILL	否	
科目	course	ENUM('國文', '英文', '數學')		否	預設為國文
分數	score	TINYINT(3)		否	

SUM():合計值

EX:想要算出全班國文、英文及數學總分

SQL:

```
SELECT SUM(`score`) FROM `scorelist`;
```

+ 選項

SUM(`score`)
2178

aggregate：

用於執行聚合函數。

ORM:

```
from DBapp.models import scorelist

from django.db.models import Sum,Count,Max,Min,Avg

datas = scorelist.objects.aggregate(Sum('score')

print(datas)
```

```
Django version 3.2.4, using settings 'DB.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
{'score__sum': 2178}
[09/Aug/2021 15:56:06] "GET /test/ HTTP/1.1" 200 13
```

EX:想要算出全班國文總分

SQL:

```
SELECT SUM(`score`) FROM `scorelist` WHERE `course`='國文';
```

+ 選項

SUM(`score`)
789

ORM:

```
from DBapp.models import scorelist

from django.db.models import Sum,Count,Max,Min,Avg

datas= scorelist.objects.filter(course='國文

').aggregate(Sum('score'))

print(datas)
```

```
[09/Aug/2021 16:03:38] "GET /test/ HTTP/1.1" 200 13
{'score__sum': 789}
[09/Aug/2021 16:03:39] "GET /test/ HTTP/1.1" 200 13
```

AVG():平均值
EX:想要算出全班國文的平均分數

SQL:

```
SELECT AVG(`score`) FROM `scorelist` WHERE `course`='國文';
```

+ 選項

AVG(`score`)
78.9000

ORM:

```
from django.db.models import Sum,Count,Max,Min,Avg

datas = scorelist.objects.filter(course='國文

').aggregate(Avg('score'))

print(datas)
```

COUNT():計次
EX:由 **students** 資料表統計全班人數

SQL:

```
SELECT COUNT(`students`.`cID`) FROM `students`
```

+ 選項

COUNT(*)
10

ORM:

```
from django.db.models import Sum,Count,Max,Min,Avg

datas = students.objects.aggregate(Count('cID'))

print(datas)
```

```
Quit the server with CTRL-BREAK.
{'cID__count': 10}
[09/Aug/2021 16:12:42] "GET /test/ HTTP/1.1" 200 13
```

MAX()、MIN():最大值、最小值

EX:找出全班國文最高分

SQL:

```
SELECT MAX(`score`) FROM `scorelist` WHERE `course`='國文'
```

ORM:

```
from django.db.models import Sum,Count,Max,Min,Avg

datas = scorelist.objects.filter(course='國文

').aggregate(Max('score'))

print(datas)
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
{'score__max': 90}
[09/Aug/2021 16:16:02] "GET /test/ HTTP/1.1" 200 13
```

EX:找出全班數學最低分

SQL:

```
SELECT MIN(`score`) FROM `scorelist` WHERE `course`='數學'
```

ORM:

```
from django.db.models import Sum,Count,Max,Min,Avg

datas = scorelist.objects.filter(course='數學

').aggregate(Min('score'))

print(datas)
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
{'score__min': 38}
[09/Aug/2021 16:17:16] "GET /test/ HTTP/1.1" 200 13
```

GROUP BY:分組排列
EX:想要顯示每個學生的總分

SQL:

```
SELECT `cID`, SUM(`score`) FROM `scorelist` GROUP BY `cID`
```

annotate：
給 QuerySet 中的每個對象在執行 SQL 時添加上一個查詢表達式（聚合函數、F 表達式、Q 表達式、Func 表達式等），表達式的查詢結果會以新欄位的方式添加到結果對象中。返回值爲 QuerySet 對象，QuerySet 中存儲的是模型對象。

ORM:

```
from django.db.models import Sum,Count,Max,Min,Avg

datas =

scorelist.objects.values_list('cID').annotate(Sum('score'))

print(datas)
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
<QuerySet [(1, 236), (2, 207), (3, 242), (4, 233), (5, 207), (6, 218), (7, 193), (8, 195),
(9, 257), (10, 190)]>
```

HAVING:GROUP BY 的條件式

若希望對 GROUP BY 的 SQL 敘述加上條件式的限制，就不能使用 WHERE 的方法，而是用 HAVING。

EX:想要顯示座號 1 到 5 同學的分數總計

SQL:

```
SELECT `cID`, SUM(`score`) FROM `scorelist` GROUP BY `cID` HAVING
`cID`<=5
```

+ 選項

←T→				cID	SUM(`score`)
☐	編輯	複製	刪除	1	236
☐	編輯	複製	刪除	2	207
☐	編輯	複製	刪除	3	242
☐	編輯	複製	刪除	4	233
☐	編輯	複製	刪除	5	207

ORM:

```
from django.db.models import Sum,Count,Max,Min,Avg
datas =
scorelist.objects.filter(cID__lte=5).values_list('cID').annota
te(Sum('score'))
print(datas)
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
<QuerySet [(1, 236), (2, 207), (3, 242), (4, 233), (5, 207)]>
[09/Aug/2021 16:34:05] "GET /test/ HTTP/1.1" 200 13
```

➢ 新增、更新、刪除

查詢雖然是資料庫中重要的功能，但是新增、更新與刪除資料的動作，才是維護資料庫內容的主要核心功能，以下將說明 SQL 語法中新增、更新與刪除資料的指令與語法。

- INSERT:新增資料

可以使用 INSERT 語法為資料表新增資料，其基本語法如下：

```
INSERT [INTO] 資料表名稱 [(欄位名稱1, 欄位名稱2, …)]
VALUES (值 1, 值 2, …);
```

EX:新增繼續新增 5 位同學進入 students 資料表，

座號	姓名	性別	生日	電子郵件	電話	地址	身高	體重
	Bill1	男	1988-02-10	Bill1@bb.com	0925932221	台北	176	89
	Bill2	男	1988-02-10	Bill2@bb.com	0925932222	新竹	170	81
	Bill3	男	1988-02-10	Bill3@bb.com	0925932223	桃園	172	84

新增一筆資料:

SQL:

```
INSERT INTO `students`
(`cName`,`cSex`,`cBirthday`,`cEmail`,`cPhone`,`cAddr`,`cHeight
`,`cWeight`) VALUES('Bill1','男
','1988-02-10','Bill1@bb.com','0925932221','台北','176','89')
```

ORM:

```
add = students(cName="bill", cSex="M", cBirthday="2021-08-08",
cEmail="bill@yahoo.com.tw", cPhone="0922222", cAddr="新竹",
cHeight=0, cWeight=0)
add.save()
```

- UPDATE:更新資料

可以使用 UPDATE 語法為資料表更新資料，其基本語法如下：

```
UPDATE 資料表名稱
SET 欄位名稱1 = 值 1, 欄位名稱2 = 值 2, ...
WHERE 條件式;
```

UPDATE 更新資料的動作可以一次更動多筆資料的內容，所以 WHERE 後加上的條件式十分重要，只要符合條件的資料內容即會進行更新的動作，要特別注意。

EX:要修改座號為 11 同學的身高體重

SQL:

```
UPDATE `students` SET `cHeight`=174, `cWeight`=92 WHERE `cID`=11
```

ORM:

```
update = students.objects.get(cID=11)

update.cHeight = 180

update.cWeight = 60

update.save()
```

- DELETE:刪除資料

可以使用 DELETE 語法為資料表刪除資料，其基本語法如下：

```
DELETE FROM 資料表名稱
WHERE 條件式;
```

DELETE 刪除資料的動作可以一次刪除多筆資料的內容，所以 WHERE 後加上的條件式十分重要，只要符合條件的資料內容即會進行刪除的動作，要特別注意。

EX:刪除座號大於 11 的同學的資料

SQL:

```
DELETE FROM `students` WHERE `cID`>11;
```

ORM:

```
delete = students.objects.filter(cID__lte=5) #(cID<=5)

delete.delete()
```

參考文獻:

https://www.geeksforgeeks.org/django-orm-inserting-updating-deleting-data/

➢ 多資料表關聯查詢

除了在一個資料表中選取欄位進行查詢，我們也可以在多個資料表中之中選取不同的欄位，進行查詢的動作。這樣的查詢方式是必須有前提的，那就是資料表之間要有一欄可以指定相關或是建立關聯。

➢ 使用 JOIN 結合資料表 by SQL

- JOIN 的基本語法

若要 JOIN 語法結合二個資料表的基本語法如下：

```
SELECT 顯示欄位…
FROM 資料表A [INNER] JOIN 資料表B
ON A.相關欄位 = 資料表 B.相關欄位
```

無論何種方式結合資料表，會顯示的資料必須在兩邊資料表有資料，只要有一方法有，即不會出現在結果中。例如有一個學生沒有登錄成績，就不會顯示結果中，如此很容易有漏失資訊的情況。

Ex:顯示出學生座號、姓名及其國文成績的查詢，使用 JOIN：

```
SELECT `students`.`cID`, `students`.`cName`, `scorelist`.`score`

FROM `students` JOIN `scorelist`

ON `students`.`cID` = `scorelist`.`cID`

WHERE `scorelist`.`course`='國文';
```

+ 選項

cID	cName	score
01	簡奉君	82
02	黃靖輪	68
03	潘四敬	78
04	賴勝恩	85
05	黎楚寧	80
06	蔡中穎	76
07	徐佳螢	90
08	林雨媗	87
09	林心儀	78
10	王燕博	65

- LEFT JOIN、RIGHT JOIN 語法

SELECT 顯示欄位..
FROM 資料表 A LEFT|RIGHT JOIN 資料表 B
ON A.相關欄位＝ B.相關欄位

EX:希望可以列出全班同學每個人的成績總分與平均。

使用結合資料表基本語法：

```
SELECT `students`.`cID`, `students`.`cName`
, SUM(`scorelist`.`score`), AVG(`scorelist`.`score`)
FROM `students`,`scorelist`
WHERE `students`.`cID` = `scorelist`.`cID`
GROUP BY `students`.`cID`, `students`.`cName`
```

	cID	cName	SUM(`scorelist`.`score`)	AVG(`scorelist`.`score`)
編輯 複製 刪除	01	簡奉君	236	78.6667
編輯 複製 刪除	02	黃靖輪	207	69.0000
編輯 複製 刪除	03	潘四敬	242	80.6667
編輯 複製 刪除	04	賴勝恩	233	77.6667
編輯 複製 刪除	05	黎楚寧	207	69.0000
編輯 複製 刪除	06	蔡中穎	218	72.6667
編輯 複製 刪除	07	徐佳螢	193	64.3333
編輯 複製 刪除	08	林雨嬉	195	65.0000
編輯 複製 刪除	09	林心儀	257	85.6667
編輯 複製 刪除	10	王燕博	190	63.3333

使用 JOIN 基本語法：

```
SELECT `students`.`cID`, `students`.`cName`
, SUM(`scorelist`.`score`), AVG(`scorelist`.`score`)
FROM `students` INNER JOIN `scorelist`
ON `students`.`cID` = `scorelist`.`cID`
GROUP BY `students`.`cID`


補充：將表別設定別名使用
SELECT `students`.`cID`, `students`.`cName`
, SUM(`scorelist`.`score`), AVG(`scorelist`.`score`)
FROM `students` st INNER JOIN `scorelist` sc
ON st.`cID` = sc.`cID`
GROUP BY st.`cID`
```

+ 選項

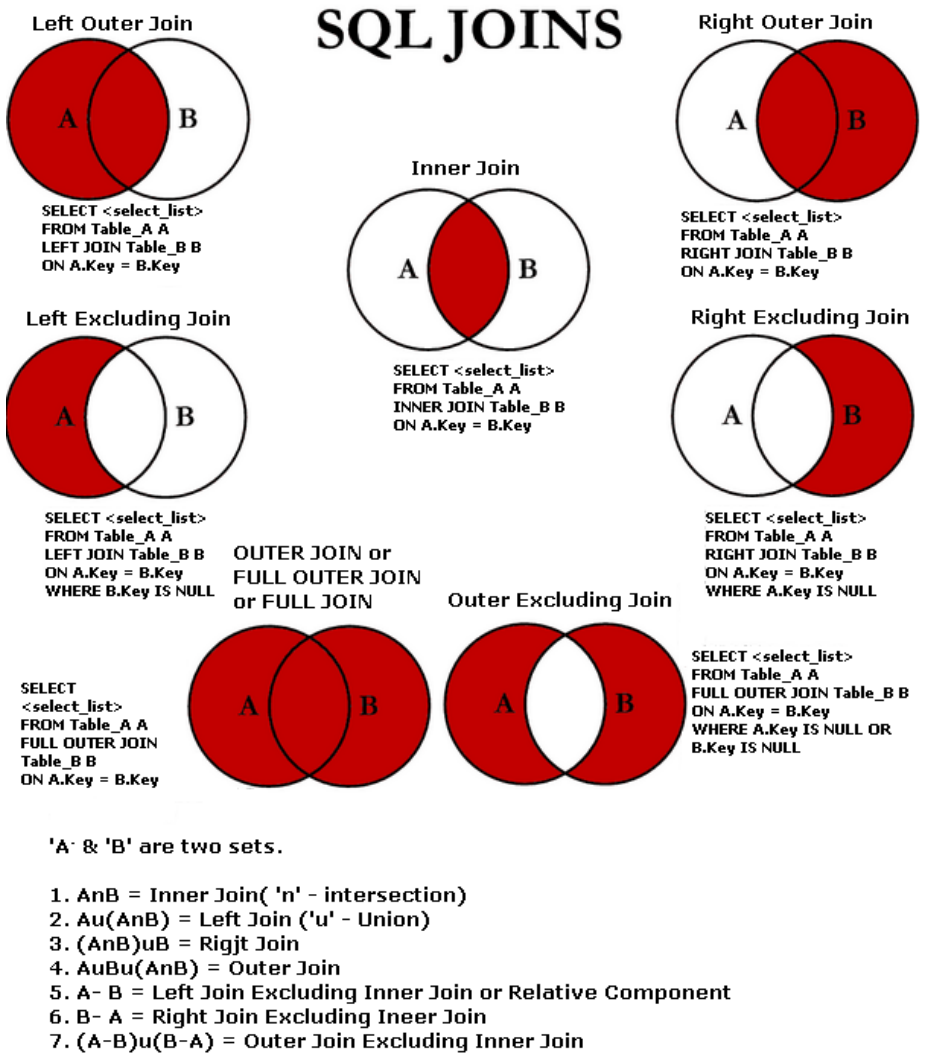
	cID	cName	SUM(`scorelist`.`score`)	AVG(`scorelist`.`score`)
編輯 複製 刪除	01	簡奉君	236	78.6667
編輯 複製 刪除	02	黃靖輪	207	69.0000
編輯 複製 刪除	03	潘四敬	242	80.6667
編輯 複製 刪除	04	賴勝恩	233	77.6667
編輯 複製 刪除	05	黎楚寧	207	69.0000
編輯 複製 刪除	06	蔡中穎	218	72.6667
編輯 複製 刪除	07	徐佳螢	193	64.3333
編輯 複製 刪除	08	林雨煊	195	65.0000
編輯 複製 刪除	09	林心儀	257	85.6667
編輯 複製 刪除	10	王燕博	190	63.3333

使用 LEFT JOIN：

```
SELECT `students`.`cID`, `students`.`cName`
, SUM(`scorelist`.`score`), AVG(`scorelist`.`score`)
FROM `students` LEFT JOIN `scorelist`
ON `students`.`cID` = `scorelist`.`cID`
GROUP BY `students`.`cID`, `students`.`cName`
```

+ 選項

	cID	cName	SUM(`scorelist`.`score`)	AVG(`scorelist`.`score`)
編輯 複製 刪除	01	簡奉君	236	78.6667
編輯 複製 刪除	02	黃靖輪	207	69.0000
編輯 複製 刪除	03	潘四敬	242	80.6667
編輯 複製 刪除	04	賴勝恩	233	77.6667
編輯 複製 刪除	05	黎楚寧	207	69.0000
編輯 複製 刪除	06	蔡中穎	218	72.6667
編輯 複製 刪除	07	徐佳螢	193	64.3333
編輯 複製 刪除	08	林雨煊	195	65.0000
編輯 複製 刪除	09	林心儀	257	85.6667
編輯 複製 刪除	10	王燕博	190	63.3333
編輯 複製 刪除	11	Bill1	NULL	NULL
編輯 複製 刪除	14	Bill2	NULL	NULL
編輯 複製 刪除	15	Bill3	NULL	NULL



'A' & 'B' are two sets.

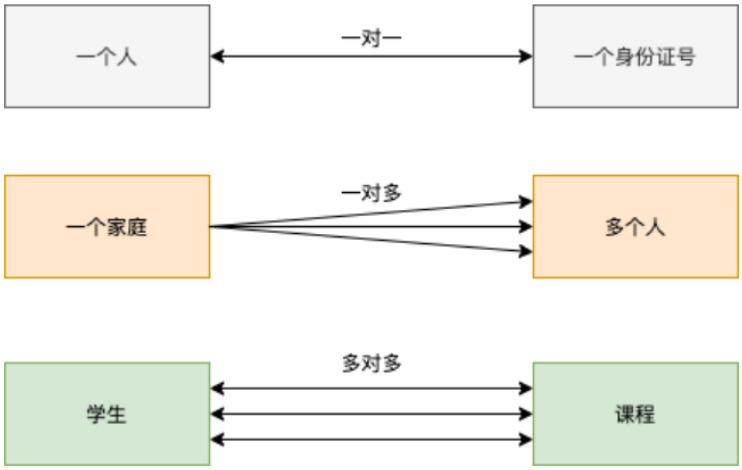
1. AnB = Inner Join('n' - intersection)
2. Au(AnB) = Left Join ('u' - Union)
3. (AnB)uB = Rigjt Join
4. AuBu(AnB) = Outer Join
5. A- B = Left Join Excluding Inner Join or Relative Component
6. B- A = Right Join Excluding Ineer Join
7. (A-B)u(B-A) = Outer Join Excluding Inner Join

➢ Django ORM 一對一、一對多、多對多

一對一：一個人對應一個身份證號碼。

一對多：一個家庭有多個人，一般會使用外鍵來實現。

多對多：一個學生有多門課程，一個課程有很多學生，一般會使用第三個表來實現關聯。



#vim DBapp/models.py

```
from django.db import models

# Create your models here.
class students(models.Model):
    cID = models.AutoField(primary_key=True)
    cName = models.CharField(max_length=20, blank=False)
    cSex =  models.CharField(max_length=1, blank=False, default='F')
    cBirthday = models.DateTimeField(auto_now_add=False)
    cEmail = models.CharField(max_length=100, blank=False)
    cPhone = models.CharField(max_length=50, blank=False)
    cAddr = models.CharField(max_length=255, blank=False)
    cHeight = models.IntegerField(blank=True, null=True)
    cWeight = models.IntegerField(blank=True, null=True)

#一個學生有多個成績(one to many)(students to scorelist)
class scorelist(models.Model):
    id = models.AutoField(primary_key=True)
    cID = models.ForeignKey('students', on_delete=models.CASCADE, null=True)
    course = models.CharField(max_length=20, blank=False)
    score = models.IntegerField(blank=False)
```

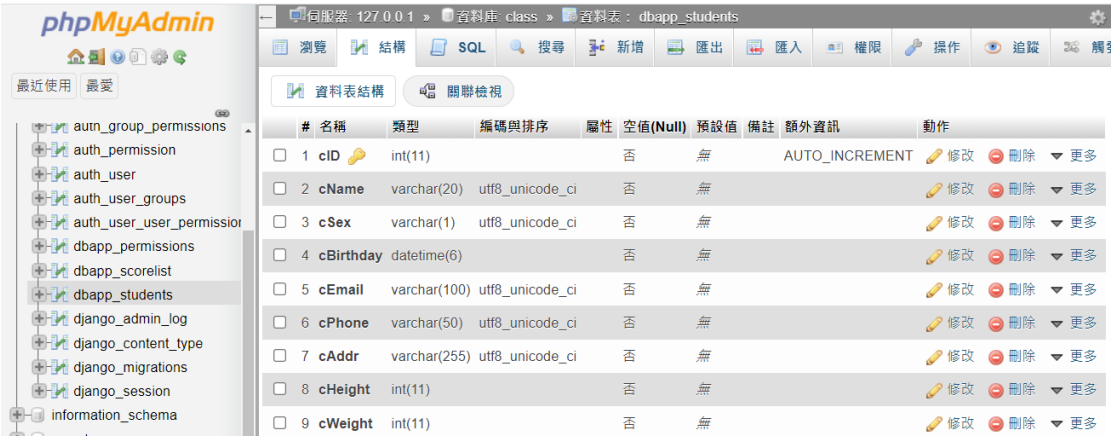
```
#一個學生有一個密碼與權限(one to one)(students to permissions)
class permissions(models.Model):
    id = models.AutoField(primary_key=True)
    cID = models.OneToOneField('students', on_delete=models.CASCADE, null=True)
    passwd = models.CharField(max_length=100, blank=False)
    level = models.CharField(max_length=2, blank=False) #0管理者 #1一般使用者

class Book(models.Model):
    name = models.CharField(max_length=32)
    authors = models.ManyToManyField(to='Author')

class Author(models.Model):
    name = models.CharField(max_length=32)
```

sudo python3 manage.py makemigrations

(建立資料庫和 Django 間的中介檔)

sudo python3 manage.py migrate

(同步更新資料庫內容)


phpMyAdmin
最近使用 最愛
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissio
dbapp_permissions
dbapp_scorelist
dbapp_students
django_admin_log
django_content_type
django_migrations
django_session
information_schema
伺服器: 127.0.0.1 » 資料庫: class » 資料表: dbapp_students
瀏覽 結構 SQL 搜尋 新增 匯出 匯入 權限 操作 追蹤
資料表結構 關聯檢視
名稱 類型 編碼與排序 屬性 空值(Null) 預設值 備註 額外資訊 動作
1 cID int(11) 否 無 AUTO_INCREMENT 修改 刪除 更多
2 cName varchar(20) utf8_unicode_ci 否 無 修改 刪除 更多
3 cSex varchar(1) utf8_unicode_ci 否 無 修改 刪除 更多
4 cBirthday datetime(6) 否 無 修改 刪除 更多
5 cEmail varchar(100) utf8_unicode_ci 否 無 修改 刪除 更多
6 cPhone varchar(50) utf8_unicode_ci 否 無 修改 刪除 更多
7 cAddr varchar(255) utf8_unicode_ci 否 無 修改 刪除 更多
8 cHeight int(11) 否 無 修改 刪除 更多
9 cWeight int(11) 否 無 修改 刪除 更多

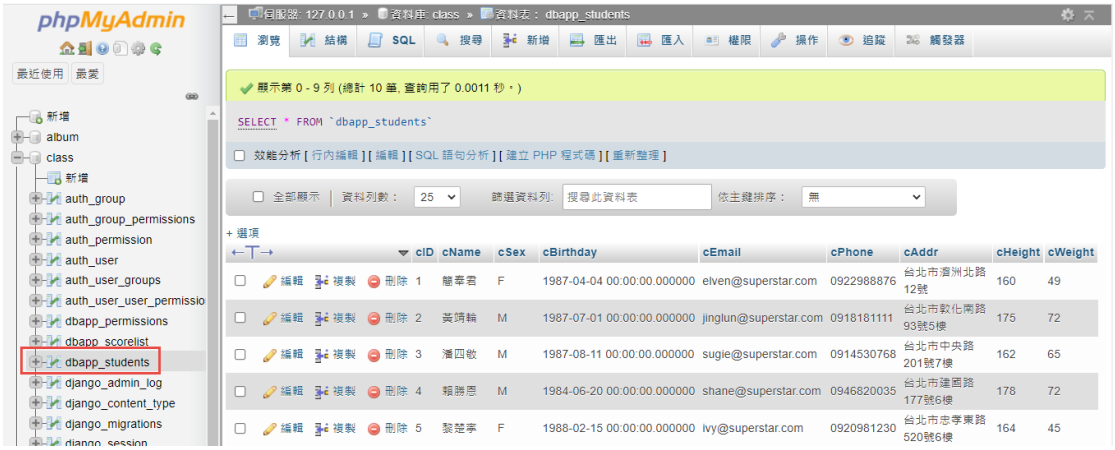
phpMyAdmin
最近使用 最愛
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissio
dbapp_permissions
dbapp_scorelist
dbapp_students
django_admin_log
django_content_type
伺服器: 127.0.0.1 » 資料庫: class » 資料表: dbapp_scorelist
瀏覽 結構 SQL 搜尋 新增 匯出 匯入 權限 操作 追蹤
資料表結構 關聯檢視
名稱 類型 編碼與排序 屬性 空值(Null) 預設值 備註 額外資訊 動作
1 id int(11) 否 無 AUTO_INCREMENT 修改 刪除 更多
2 cID_id int(11) 是 NULL 修改 刪除 更多
3 course varchar(20) utf8_unicode_ci 否 無 修改 刪除 更多
4 score int(11) 否 無 修改 刪除 更多
全選 已選擇項目： 瀏覽 修改 刪除 主鍵 獨一 索引 空間
全文搜尋 新增至中央欄位 從中央欄位移除

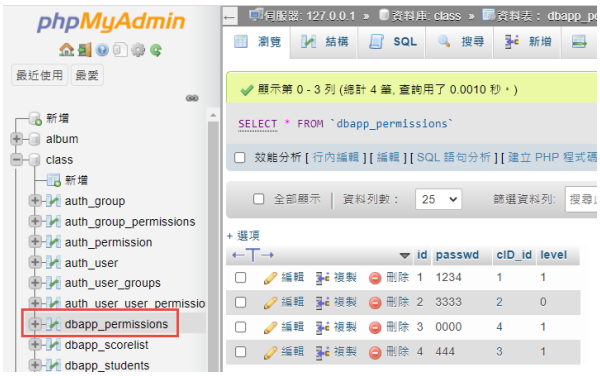
phpMyAdmin
最近使用 最愛
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissio
dbapp_permissions
dbapp_scorelist
dbapp_students
伺服器: 127.0.0.1 » 資料庫: class » 資料表: dbapp_permissions
瀏覽 結構 SQL 搜尋 新增 匯出 匯入 權限 操作 追蹤 觸發器
資料表結構 關聯檢視
名稱 類型 編碼與排序 屬性 空值(Null) 預設值 備註 額外資訊 動作
1 id int(11) 否 無 AUTO_INCREMENT 修改 刪除 更多
2 passwd varchar(100) utf8_unicode_ci 否 無 修改 刪除 更多
3 cID_id int(11) 是 NULL 修改 刪除 更多
4 level varchar(2) utf8_unicode_ci 否 無 修改 刪除 更多
全選 已選擇項目： 瀏覽 修改 刪除 主鍵 獨一 索引 空間

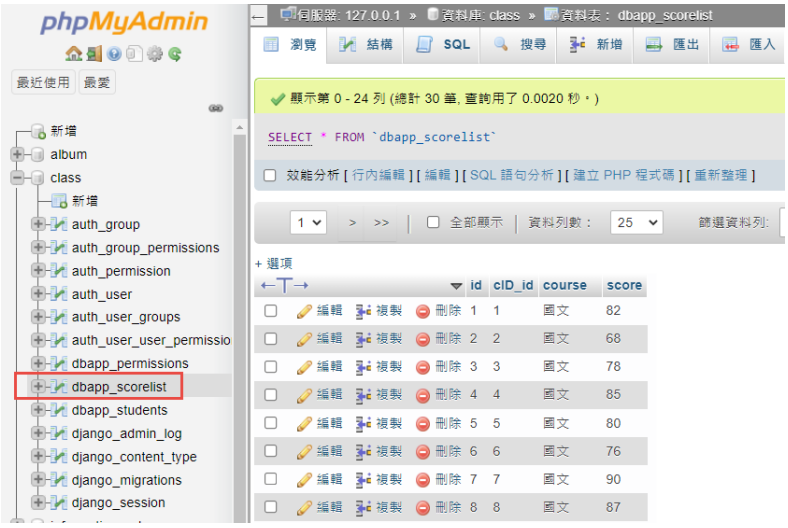
phpMyAdmin
最近使用 最愛
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissio
dbapp_author
dbapp_book
伺服器: 127.0.0.1 » 資料庫: class » 資料表: dbapp_author
瀏覽 結構 SQL 搜尋 新增 匯出 匯入 權限 操作 追蹤
資料表結構 關聯檢視
名稱 類型 編碼與排序 屬性 空值(Null) 預設值 備註 額外資訊 動作
1 id bigint(20) 否 無 AUTO_INCREMENT 修改 刪除 更多
2 name varchar(32) utf8_unicode_ci 否 無 修改 刪除 更多
全選 已選擇項目： 瀏覽 修改 刪除 主鍵 獨一 索引 空間
新增至中央欄位 從中央欄位移除

dbapp_book_authors(自動增加的表格)

➢ 匯入資料表資料







- ORM(One-To-One Relation)

```
#vim DBapp/models.py
```

```
class students(models.Model):
    cID = models.AutoField(primary_key=True)
    cName = models.CharField(max_length=20, blank=False)
    cSex =  models.CharField(max_length=1, blank=False, default='F')
    cBirthday = models.DateTimeField(auto_now_add=False)
    cEmail = models.CharField(max_length=100, blank=False)
    cPhone = models.CharField(max_length=50, blank=False)
    cAddr = models.CharField(max_length=255, blank=False)
    cHeight = models.IntegerField(blank=True, null=True)
    cWeight = models.IntegerField(blank=True, null=True)
```

```
class permissions(models.Model):
    id = models.AutoField(primary_key=True)
    cID = models.OneToOneField('students', on_delete=models.CASCADE, null=True)
    passwd = models.CharField(max_length=100, blank=False)
    level = models.CharField(max_length=2, blank=False) #0管理者 #1一般使用者
```

新增

Ex:新增一個學生

基本資料如下：

「王大明」、「M」、「2020-08-20」、「wang@yahoo.com.tw」、「099999」、「新竹」、「120」、「40」，密碼為「0000」、level 為「0」

ORM:

```
datas = students.objects.create(cName='王大明',cSex='M',cBirthday='2020-08-20', cEmail='wang@yahoo.com.tw', cPhone='09333333', cAddr='新竹', cHeight=120, cWeight=40)
```

```
permissions.objects.create(cID=datas, passwd='0000', level='0')
```

查尋

Ex:顯示學生的座號、姓名、密碼、level

ORM:

```
datas =
students.objects.values('cID','cName','permissions__passwd',
'permissions__level')
for data in datas:
   print('%s %s %s %s' % (data['cID'], data['cName'],
data['permissions__passwd'] ,data['permissions__level']))
```

```
1 簡奉君 1234 1
2 黃靖輪 3333 0
3 潘四敬 444 1
4 賴勝恩 0000 1
5 黎楚寧 None None
6 蔡中穎 None None
7 徐佳螢 None None
8 林雨媗 None None
9 林心儀 None None
10 王燕博 None None
```

刪除

Ex:刪除「王大明」的基本資料與權限資料

```
第一種方法:
students.objects.get(cName='王大明').delete()
第二種方法:
students.objects.filter(cName='王大明').delete()
```

- ORM(One-To-Many Relation)

```
#vim DBapp/models.py
```

```
class students(models.Model):
    cID = models.AutoField(primary_key=True)
    cName = models.CharField(max_length=20, blank=False)
    cSex =  models.CharField(max_length=1, blank=False, default='F')
    cBirthday = models.DateTimeField(auto_now_add=False)
    cEmail = models.CharField(max_length=100, blank=False)
    cPhone = models.CharField(max_length=50, blank=False)
    cAddr = models.CharField(max_length=255, blank=False)
    cHeight = models.IntegerField(blank=True, null=True)
    cWeight = models.IntegerField(blank=True, null=True)
```

```
class scorelist(models.Model):
    id = models.AutoField(primary_key=True)
    cID = models.ForeignKey('students', on_delete=models.CASCADE, null=True)
    course = models.CharField(max_length=20, blank=False)
    score = models.IntegerField(blank=False)
```

新增

Ex: 想新增姓名為「黃靖輪」的地理成績

ORM:

```
第一種寫法:

scorelist.objects.create(cID=students.objects.get(cName='黃靖輪
'), course='地理', score=60)


第二種寫法:

data  =  scorelist(cID=students.objects.get(cName=' 黃 靖 輪 '),
course='音樂', score=60)

data.save()
```

查尋

Ex:顯示出學生座號、姓名及其國文成績的查詢。

ORM:

```
print(students.objects.filter(scorelist__course='國文
').values('cID','cName','scorelist__course','scorelist__score'
))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
<QuerySet [{'cID': 1, 'cName': '簡奉君', 'scorelist__course': '國文', 'scorelist__score': 82}, {'cID': 2, 'cName': '黃靖輪',
 'scorelist__course': '國文', 'scorelist__score': 68}, {'cID': 3, 'cName': '潘四敬', 'scorelist__course': '國文', 'scorelist
__score': 78}, {'cID': 4, 'cName': '賴勝恩', 'scorelist__course': '國文', 'scorelist__score': 85}, {'cID': 5, 'cName': '黎楚
寧', 'scorelist__course': '國文', 'scorelist__score': 80}, {'cID': 6, 'cName': '蔡中穎', 'scorelist__course': '國文', 'score
list__score': 76}, {'cID': 7, 'cName': '徐佳螢', 'scorelist__course': '國文', 'scorelist__score': 90}, {'cID': 8, 'cName': '
林雨媗', 'scorelist__course': '國文', 'scorelist__score': 87}, {'cID': 9, 'cName': '林心儀', 'scorelist__course': '國文', 's
corelist__score': 78}, {'cID': 10, 'cName': '王燕博', 'scorelist__course': '國文', 'scorelist__score': 65}]>
```

```
datas =
students.objects.filter(scorelist__course='國文
').values('cID','cName','scorelist__course','scorelist__score'
)
for data in datas:
  print('%s %s %s %s' % (data['cID'], data['cName'],
data['scorelist__course'] ,data['scorelist__score']))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
1 簡奉君 國文 82
2 黃靖輪 國文 68
3 潘四敬 國文 78
4 賴勝恩 國文 85
5 黎楚寧 國文 80
6 蔡中穎 國文 76
7 徐佳螢 國文 90
8 林雨媗 國文 87
9 林心儀 國文 78
10 王燕博 國文 65
```

EX:希望可以列出全班同學每個人的成績總分與平均。

ORM:

```
from django.db.models import Sum,Count,Max,Min,Avg
print(students.objects.values_list('cID','cName').annotate(Sum
('scorelist__score'), Avg('scorelist__score')))
```

```
<QuerySet [(1, '簡奉君', 236, 78.6667), (2, '黃靖輪', 207, 69.0), (3, '潘四敬', 242, 80.6667), (4, '賴勝恩', 233, 77.6667),
(5, '黎楚寧', 207, 69.0), (6, '蔡中穎', 218, 72.6667), (7, '徐佳螢', 193, 64.3333), (8, '林雨媗', 195, 65.0), (9, '林心儀',
257, 85.6667), (10, '王燕博', 190, 63.3333)]>
[19/Aug/2021 14:17:10] "GET /test/ HTTP/1.1" 200 13
```

```
from django.db.models import Sum,Count,Max,Min,Avg
```

```
datas =

students.objects.values_list('cID','cName').annotate(Sum('scor

elist__score'), Avg('scorelist__score'))

for data in datas:

  print('%s %s %s %s' % (data[0], data[1], data[2] , data[3]))
```

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
1 簡奉君 236 78.6667
2 黃靖輪 207 69.0
3 潘四敬 242 80.6667
4 賴勝恩 233 77.6667
5 黎楚寧 207 69.0
6 蔡中穎 218 72.6667
7 徐佳螢 193 64.3333
8 林雨媗 195 65.0
9 林心儀 257 85.6667
10 王燕博 190 63.3333
```

修改

Ex:

將學號「03」的國文成績改成 60 分。

ORM:

```
第一種寫法:

datas = students.objects.get(cID=3)

datas.scorelist_set.filter(course='國文').update(score=60)



第二種寫法:

scorelist.objects.filter(course='國文' ,cID=

students.objects.get(cID=3).cID).update(score=60)
```

Ex:

將姓名為「簡奉君」的國文成績改成 100 分。

ORM:

```
第一種寫法:

datas = students.objects.get(cName='簡奉君')
```

```
(先查尋多對一的一那方)

datas.scorelist_set.filter(course='國文').update(score=100)

(再查尋多的那方，更新多的那方成績)


第二種寫法:

scorelist.objects.filter(course='國文' ,cID=

students.objects.get(cName='簡奉君').cID).update(score=100)

(內層為多對一的一那方的 cID 等於多的那方的外來鍵，更新多的那方成績)
```

刪除

Ex: 將姓名簡奉君的學生資料與成績全部刪除。

ORM:

```
students.objects.get(cName='簡奉君').delete()
```

- ORM(Many-To-Many Relation)

```
# vim DBapp/models.py
```

```
class Book(models.Model):
    name = models.CharField(max_length=32)
    authors = models.ManyToManyField(to='Author')

class Author(models.Model):
    name = models.CharField(max_length=32)
```

新增

Ex: 第一次新增作者「小虎」與新增書名「C++」、「PHP」

ORM:

```
第一種寫法:

Author_obj = Author.objects.create(name="小虎")


Book_obj = Book.objects.create(name="C++")

Book_obj.authors.add(Author_obj)

Book_obj = Book.objects.create(name="PHP")
```

```
Book_obj.authors.add(Author_obj)


第二種寫法

Author_obj = Author(name="小虎")

Author_obj.save()


Book_obj1 = Book(name="C++")

Book_obj1.save()

Book_obj2 = Book(name="PHP")

Book_obj2.save()


Book_obj1.authors.add(Author_obj)

Book_obj2.authors.add(Author_obj)
```

Ex: 已存在作者小虎，新增此作者的書「JAVA」、「Linux」

ORM:

```
第一種寫法:

Author_obj = Author.objects.get(name="小虎")

Book_obj = Book.objects.create(name="JAVA")

Book_obj.authors.add(Author_obj)

Book_obj = Book.objects.create(name="Linux")

Book_obj.authors.add(Author_obj)
```

第二種寫法:

```
Author_obj = Author.objects.get(name="小虎")

Book_obj1 = Book(name="JAVA")

Book_obj1.save()

Book_obj1.authors.add(Author_obj)


Book_obj2 = Book(name="Linux")

Book_obj2.save()

Book_obj2.authors.add(Author_obj)
```

Ex: 新增兩個「小明」、「大雄」均為「C#」書的作者

ORM:

第一種寫法:

```
Author_obj1 = Author.objects.create(name="小明")

Author_obj2 = Author.objects.create(name="大雄")

Book_obj = Book.objects.create(name="c#")

Book_obj.authors.add(Author_obj1, Author_obj2)


Author_obj1 = Author(name="小明")

Author_obj1.save()
```

```
Author_obj2 = Author(name="大雄")

Author_obj2.save()

Book_obj = Book(name="c#")

Book_obj.save()

Book_obj.authors.add(Author_obj1, Author_obj2)
```

查詢:

Ex:「C#」書的作者有那些?

ORM:

```
print(Book.objects.filter(name="C#").values('name','authors__name'))
```

```
Quit the server with CTRL-BREAK.
<QuerySet [{'name': 'c#', 'authors__name': '小明'}, {'name': 'c#', 'authors__name': '大雄'}]>
[23/Aug/2021 15:02:42] "GET /test/ HTTP/1.1" 200 4
```

Ex:作者「小虎」有出版那些書?

ORM:

```
第一種寫法:

print(Book.objects.filter(authors=Author.objects.get(name="小虎")).values('name','authors__name'))


第二種寫法:

print(Book.objects.filter(authors__name="小虎").values('name','authors__name'))
```

第三種寫法:

```
authors_id=Author.objects.get(name="小虎").id
print(Book.objects.filter(authors=authors_id).values('name','authors__name'))
```

```
<QuerySet [{'name': 'C++', 'authors__name': '小虎'}, {'name': 'PHP', 'authors__name': '小虎'}, {'name': 'JAVA', 'authors__name': '小虎'}, {'name': 'Linux', 'authors__name': '小虎'}]>
```

刪除:

Ex:刪除書名為「**C#**」的所有資料，但 **Author** 資料表保留

```
Book.objects.get(name="C#").delete()
```

Ex:刪除作者為「小虎」的所有資料，但 **Book** 資料表保留

```
Author.objects.get(name="小虎").delete()
```

參考文獻:

https://notes.andywu.tw/2018/%E8%B3%87%E6%96%99%E5%BA%AB-%E9%97%9C%E8%81%AF%E4%BB%8B%E7%B4%B9-%E4%B8%80%E5%B0%8D%E4%B8%80%E3%80%81%E4%B8%80%E5%B0%8D%E5%A4%9A%E3%80%81%E5%A4%9A%E5%B0%8D%E5%A4%9A/

https://www.runoob.com/django/django-orm-2.html

https://www.cnblogs.com/pythonxiaohu/p/5814247.html

➢ UPDATE using INNER JOIN

Format:

```
UPDATE T1
[INNER JOIN | LEFT JOIN] T2 ON T1.C1 = T2. C1
SET T1.C2 = T2.C2,
      T2.C3 = expr
WHERE condition
```

將 cID 為「03」的國文成績改成 60 分。

UPDATE `scorelist` SET `score`=60 WHERE `cID`=3 AND `course`='國文'

將姓名為「簡奉君」的國文成績改成 100 分。

UPDATE `students` INNER JOIN `scorelist` ON `students`.`cID`=`scorelist`.`cID` SET `scorelist`.`score`='100' WHERE `students`.`cName`='簡奉君' AND `scorelist`.`course`='國文'

➢ Delete using INNER JOIN

Format:

```
DELETE T1, T2
FROM T1
INNER JOIN T2 ON T1.key = T2.key
WHERE condition;
```

將姓名簡奉君的學生資料與成績全部刪除。

DELETE `students`,`scorelist` FROM `students` INNER JOIN `scorelist` ON `students`.`cID`=`scorelist`.`cID` WHERE `students`.`cName`="簡奉君"

參考資料:

http://www.mysqltutorial.org/mysql-update-join/

https://www.yiibai.com/mysql/update-join.html

3-14 Cookies 與 Sessions

使用者在瀏覽網頁時並不是一直與伺服器保持在連線的狀態下，事實上當瀏覽者送出需求到伺服器端處理後將結果回傳顯示，就已經結束與伺服器的連線。所以當需要新資料或是更新顯示內容時，都必須重新載入頁面或是重新送出需求。

但遇到在網站運作上有些需要「維持記憶」的狀況時，例如記住當前登入使用者的資訊，或是保持在購物車裡未結帳的商品以供下次繼續使用時，該怎麼辦？Cookie 與 Session 的存在就是為了要解決網站不能保存狀態的問題。

以一般網站最常見的會員系統來說，當會員以帳號密碼登入系統的同時，程式可以有二個方式來記住登入會員的資料：一個方法是在登入者的電腦中放入一個小檔案來記憶，這個就是 Cookie；另一個方法是在伺服器的記憶體產生一個空間來記憶，這個就是 Session。

當瀏覽者在成功登入後，在載入頁面的同時程式即可調出用戶端的 Cookie 或是伺服器端的 Session 來檢查並維特登入的狀態。當使用者要離開時，程式只要清除用戶端的 Cookie 或是伺服器端的 Session，即可將原來的狀態清除。

➢ 建立環境

- 建立 Django 專案：

```
# cd ~/dvds
# source ./dvds/dvdsenv/bin/activate (啟動虛擬環境)
# django-admin startproject CookieSession (建立 project1 專案)
```

➢ 關於 Cookie

Cookie 是儲存在瀏覽者電腦中的小檔案，可以用來識別使用者的身份或是相關資訊。因為是放置在用戶端的電腦，瀏覽者不必與伺服器溝通即可取得其中的資訊，免除與伺服器之間多餘的連線。例如瀏覽者將未結帳的商品放置在購物車中，中途可能因故離開或是關閉瀏覽器，都能藉由 Cookie 的幫忙在下一次回

到原網站操作時，調出未結帳的商品繼續購物。

Cookie 放置在瀏覽者的電腦中能保持一段較長的時間，在 Cookie 未消失前都能正確記錄資訊，搭配程式的應用即可免隱重複輸入資料的麻煩。例如當登入會員系之後，程式則將該使用者的資訊紀入在 Cookie 中，即使關閉瀏覽器後重新開啟原來的頁面，該使用者依然能夠因為 Cookie 的幫忙維持登入的狀態。

Cookie 的限制：

1. 目前每個瀏覽器最多只能儲存 300 個 cookie。
2. 每個瀏覽器對單一網站只能儲存 20 的 cookie。
3. 每個 Cookie 的大小最多僅 4k Bytes 的容量。
4. 用戶端電腦的 cookie 只要關閉就沒辦法使用。

Cookie 在使用時較讓人擔心的是資訊安全，因為它是以明確的方式儲存在使用者的電腦中，有可能被擷取進行不當的利用。所以若記錄的資訊較為機密，如帳號密碼、信用卡卡號等，就不適合。

➢ 關於 Session

當瀏覽者進入網站伺服器瀏覽時，在 Session 的開啟狀態下即會開始記錄使用者所賦予的資訊，一直到關閉瀏覽器才結束 Session 的使用。在安全性的考量下，許多人在設計程式時都會使用 Session 而不用 Cookie。因為 Session 是產生在伺服器端的，不易遭人利用進行其他的操作。在程式運作正常的考量下，Session 因為存在伺服器端，即使用戶端的瀏覽器關閉 Cookie 的使用時，Session 仍可正常運作。

➢ Cookie 的使用

Cookie 是將狀態資料記錄在用戶端電腦的技術，當瀏覽者開啟網站時即可在程式的設定下將指定的資料儲存在用戶端電腦中，並可以設定該資料的有效時間、存放路徑與有效網域。以下我們將介紹如何存取 Cooke 與設定 Cookie 的有效時間。

- 儲存 Cookie 資料

格式如下：

set_cookie(key, value=”, max_age=None, expires=None)

參數	
key	變數名稱
value	值
max_age	持續時間，單位為秒

expires	到期時間

HttpResponse.set_cookie(*key, value='', max_age=None, expires=None, path='/', domain=None, secure=False, httponly=False, samesite=None*)

Sets a cookie. The parameters are the same as in the **Morsel** cookie object in the Python standard library.

- **max_age** should be an integer number of seconds, or **None** (default) if the cookie should last only as long as the client's browser session. If **expires** is not specified, it will be calculated.
- **expires** should either be a string in the format **"Wdy, DD-Mon-YY HH:MM:SS GMT"** or a **datetime.datetime** object in UTC. If **expires** is a **datetime** object, the **max_age** will be calculated.

其中 expires 是使用 UTC(世界協調時間，Coordinated Universal Time)，在時刻上儘量接近於格林威治標準時間。

Ex:

from django.http import HttpResponse

response = HttpResponse('TestCookie')

response.set_cookie(TestCookie,”這是 cookie 內容”)

設定有效持續 1 小時：

from django.http import HttpResponse

response = HttpResponse('TestCookie')

response.set_cookie(TestCookie,”這是 cookie 內容”, max_age=3600)

- 讀取 Cookie 資料

Request.COOKIES[“名稱”]

範例：儲存並顯示 Cookie 值

新增瀏覽位置(urls.py):

```
# sudo vim CookieSession/urls.py
```

```
16  from django.contrib import admin
17  from django.urls import path
18  from CookieSessionApp import views
19
```

```
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('set_cookie/<str:key>/<str:value>', views.set_cookie),
23
24  ]
```

新增 view functions：

```
# sudo vim CookieSessionApp/views.py
```

urls.py views.py

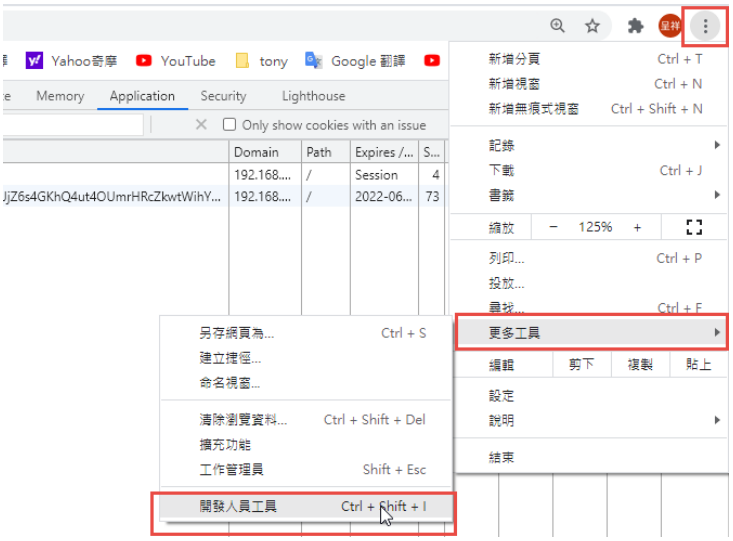
C: > Users > tony > DOCUME~1 > MobaXterm > slash > RemoteFiles > 2

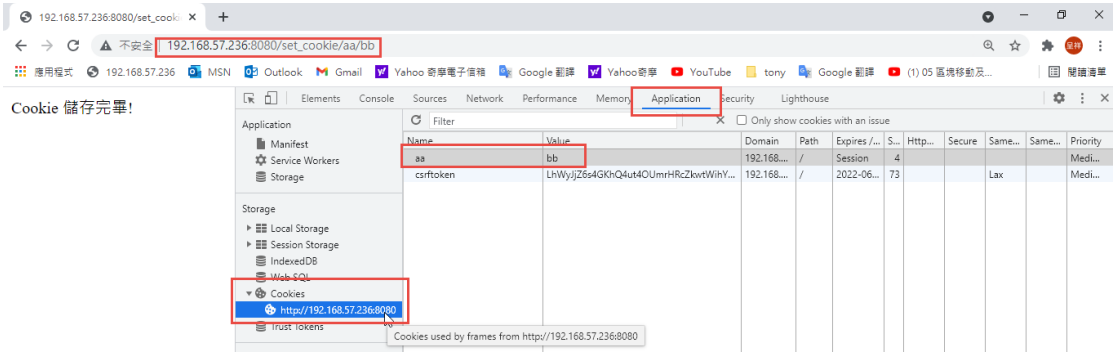
```
from django.shortcuts import render
from django.http import HttpResponse

def set_cookie(request,key=None,value=None):
    # print(key)
    # print(value)
    # return HttpResponse("ok")
    response = HttpResponse('Cookie 儲存完畢!')
    response.set_cookie(key,value)
    return response
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

按「F12」，開啟「開發人員工具」





新增瀏覽位置(urls.py):

sudo vim CookieSession/urls.py

```
16  from django.contrib import admin
17  from django.urls import path
18  from CookieSessionApp import views
19
```

```
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('set_cookie/<str:key>/<str:value>', views.set_cookie),
23      path('get_cookie/<str:key>', views.get_cookie),
24
25  ]
```

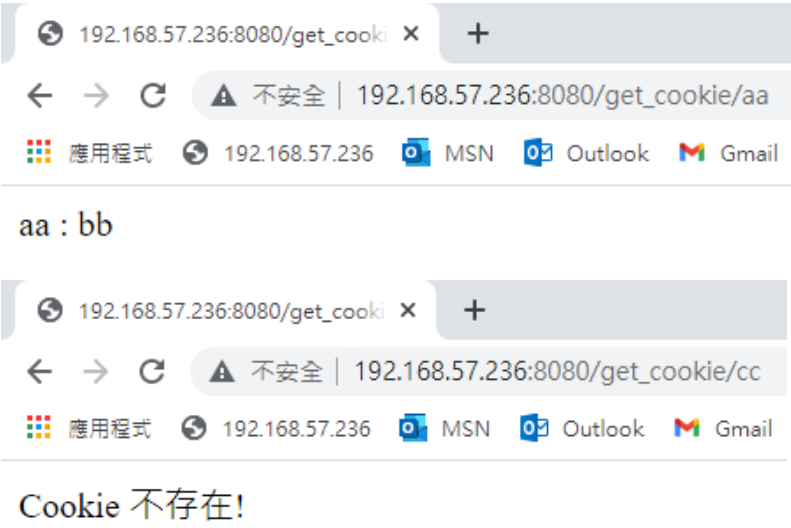
新增 view functions：

sudo vim CookieSessionApp/views.py

```
12  def get_cookie(request,key=None):
13      if key in request.COOKIES:
14          return HttpResponse('%s : %s' %(key,request.COOKIES[key]))
15      else:
16          return HttpResponse('Cookie 不存在!')
17
```

測試結果：

sudo python3 manage.py runserver 0.0.0.0:8080



Cookie 字典:

Cookie 的格式是字典，因此可以迴圈方式從字典的 items()項目中，取得字典的

key 與 value。

新增瀏覽位置(urls.py):

sudo vim CookieSession/urls.py

```
16  from django.contrib import admin
17  from django.urls import path
18  from CookieSessionApp import views
19
```

```
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('set_cookie/<str:key>/<str:value>', views.set_cookie),
23      path('get_cookie/<str:key>', views.get_cookie),
24      path('get_allcookies/', views.get_allcookies),
```

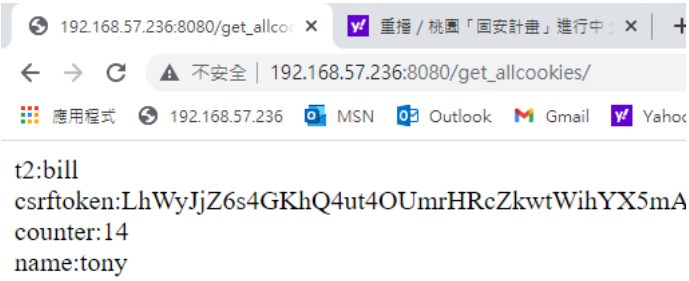
新增 view functions：

sudo vim CookieSessionApp/views.py

```
def get_allcookies(request):
    if request.COOKIES!=None:
        strcookies=""
        for key1,value1 in request.COOKIES.items():
            strcookies= strcookies + key1 + ":" + value1 + "<br>"
        return HttpResponse('%s' %(strcookies))
    else:
        return HttpResponse('Cookie 不存在!')
```

測試結果：

sudo python3 manage.py runserver 0.0.0.0:8080



- Cookie 的有效時間

當您關閉瀏覽器再重新開啟瀏覽器執行該程式，會發現原來儲存的 Cookie 值已消失，程式又必須重新加入。若想要保持 Cookie 的存在，就必須設定 Cookie 的持續時間。

設定 Cookie 的持續時間：
持續時間 1 小時：
response.set_cookie("TestCookie","內容",max_age=3600)

設定到期時間：
tomorrow = datetime.datetime.now() + datetime.timedelta(days=1)
tomorrow = datetime.datetime.replace(tomorrow, hour=0, minute=0, second=0)
expires = datetime.datetime.strftime(tomorrow, "%a, %d-%b-%Y %H:%M:%S GMT")
response.set_cookie("counter", counter, expires= expires)

刪除 Cookie：
delete_cookie[名稱]
Ex:
from django.http import HttpResponse
response = HttpResponse('Delete Cookie')
response.delete_cookie('TestCookie')

範例：設定 Cookie 的持續時間

新增瀏覽位置(urls.py):

sudo vim CookieSession/urls.py

```
16  from django.contrib import admin
17  from django.urls import path
18  from CookieSessionApp import views
19
```

```
16  from django.contrib import admin
17  from django.urls import path
18  from CookieSessionApp import views
19
20  urlpatterns = [
21      path('admin/', admin.site.urls),
22      path('set_cookie/<str:key>/<str:value>', views.set_cookie),
23      path('get_cookie/<str:key>', views.get_cookie),
24
25      path('set_cookie2/<str:key>/<str:value>', views.set_cookie2),
26      path('delete_cookie/<str:key>', views.delete_cookie),
27
28  ]
```

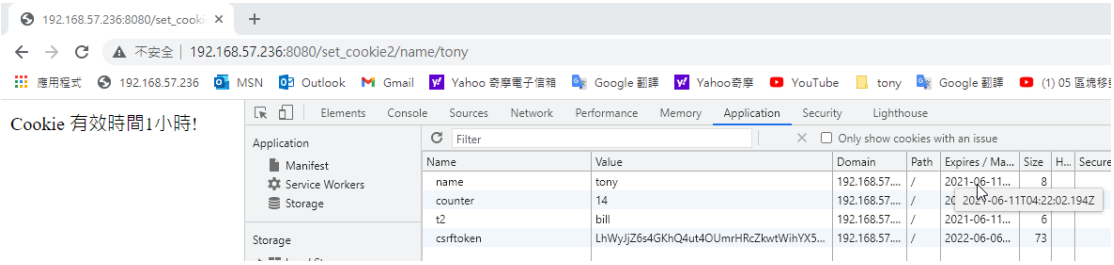
新增 view functions：

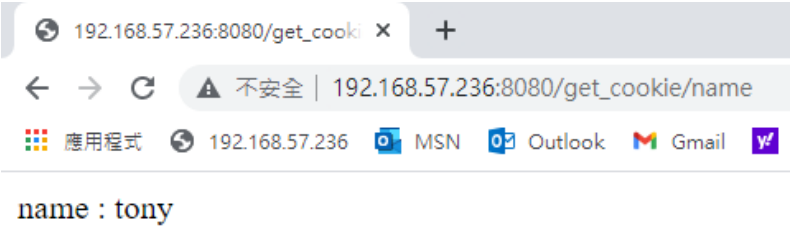
sudo vim CookieSessionApp/views.py

```python
def set_cookie2(request,key=None,value=None):
    response = HttpResponse('Cookie 有效時間1小時!')
    response.set_cookie(key,value,max_age=3600)
    return response

def delete_cookie(request,key=None):
    if key in request.COOKIES:
        response = HttpResponse('Delete Cookie: '+key)
        response.delete_cookie(key)
        return response
    else:
        return HttpResponse('No cookies:' + key)

```

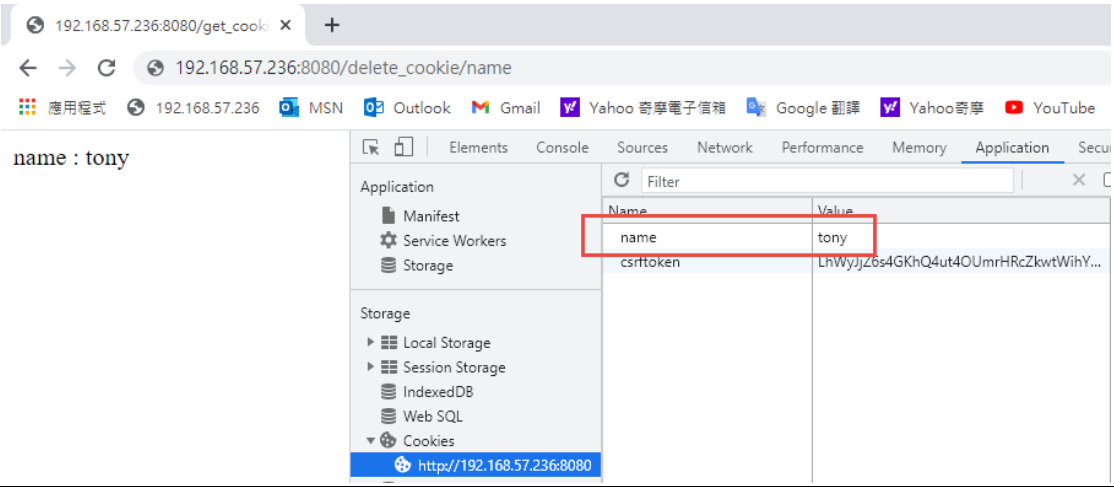
測試結果：

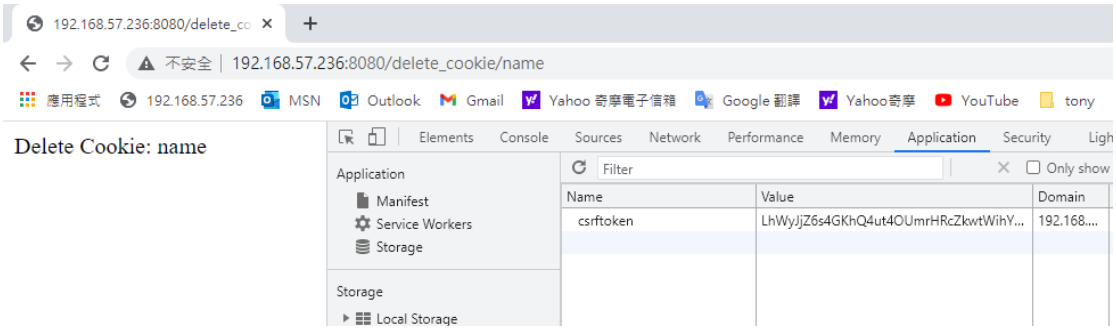
```
# sudo python3 manage.py runserver 0.0.0.0:8080
```





關閉瀏覽起後再開啟此頁，cookie 的 name 變數一樣會在





範例：顯示使用者今天瀏覽本頁面的次數

新增瀏覽位置(urls.py):

sudo vim CookieSession/urls.py

```
from django.contrib import admin
from django.urls import path
from CookieSessionApp import views

```

```
    path('', views.index),
    path('index/', views.index),

```

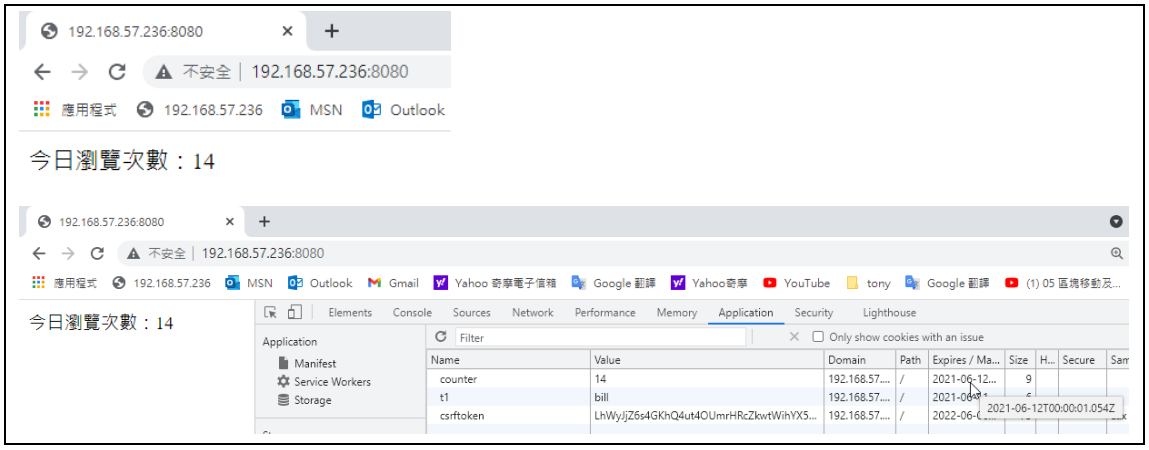
新增 view functions：

sudo vim CookieSessionApp/views.py

```
import datetime
def index(request):
    if "counter" in request.COOKIES:
        counter=int(request.COOKIES["counter"])
        counter+=1
    else:
        counter=1
    response = HttpResponse('今日瀏覽次數：' + str(counter))
    tomorrow = datetime.datetime.now() + datetime.timedelta(days = 1)
    tomorrow = datetime.datetime.replace(tomorrow, hour=0, minute=0, second=0)
    expires = datetime.datetime.strftime(tomorrow, "%a, %d-%b-%Y %H:%M:%S GMT")
    response.set_cookie("counter",counter,expires=expires)
    return response
```

測試結果：

sudo python3 manage.py runserver 0.0.0.0:8080



➢ Session 的使用

Session 是瀏覽者與伺服器連線工作期間所保持的狀態，它的使用時間是在開啟瀏覽器後進入啟動 Session 機制的網站開始，只是 Session 沒有到期，回到原網站時會發現原來的 Session 仍然有效。

Session 的運作原理：

當使用者使用瀏覽器連線到伺服網站時，網站伺服器會自動派發一個 SessionID 給這次的連線動作，網站程式即可依照這個 SessionID 分辨使用者來處理所儲存的狀態。事實上：在預設的狀態下，Session 會將伺服器所派發的 SessionID 加密處理後以 Cookie 的方式儲存在用戶端來記錄狀態，同一個站的不同網頁可以藉由這個 Cookie 的記錄來維持同一個 SessionID 的狀態。

- 安裝 Session APP

使用 Session，必須安裝 Session 的 APP。

```
# vim CookieSession/CookieSession/settings.py
```

```
31  # Application definition
32
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'CookieSessionApp',
41  ]
42
43  MIDDLEWARE = [
44      'django.middleware.security.SecurityMiddleware',
45      'django.contrib.sessions.middleware.SessionMiddleware',
46      'django.middleware.common.CommonMiddleware',
47      'django.middleware.csrf.CsrfViewMiddleware',
```

- 存取 Session 資料

可以使用 request 物件的 sesson()函式，以字典方式存取 session 資料。

儲存 Session 資料：request.session[名稱]=值
讀取 Session 資料：變數=request.session[名稱]

範例：儲存並顯示 Session 值

新增瀏覽位置(urls.py):

sudo vim CookieSession/urls.py

```
16  from django.contrib import admin
17  from django.urls import path
18  from CookieSessionApp import views
19
```

```
32      path('set_sessiion/<str:key>/<str:value>', views.set_session),
33      path('get_sessiion/', views.get_session),
34
```

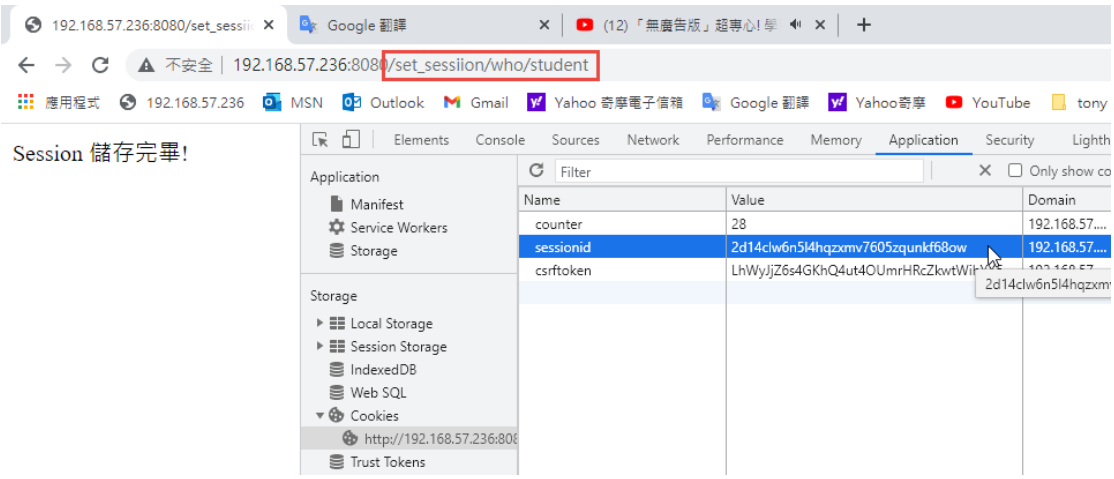
新增 view functions：

sudo vim CookieSessionApp/views.py

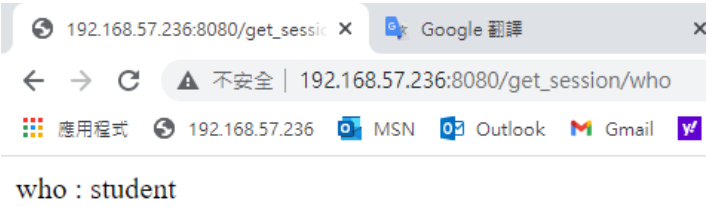
```
62  def set_session(request,key=None,value=None):
63      response = HttpResponse('Session 儲存完畢!')
64      request.session[key]=value
65      return response
66
67  def get_session(request,key=None):
68      if key in request.session:
69          return HttpResponse('%s : %s' %(key,request.session[key]))
70      else:
71          return HttpResponse('Session 不存在!')
```

測試結果：

sudo python3 manage.py runserver 0.0.0.0:8080



其中，Session 將伺服器所派發的 SessionID 加密處理後以 Cookie 的方式儲存在用戶端，它的名稱是 sessionid，內容是一個編碼過的字串。



Session 字典:
Session 的格式是字典，因此可以迴圈方式從字典的 items()項目中，取得字典的 key 與 value。

新增瀏覽位置(urls.py):

sudo vim CookieSession/urls.py

```
from django.contrib import admin
from django.urls import path
from CookieSessionApp import views

    path('set_session/<str:key>/<str:value>', views.set_session),
    path('get_session/<str:key>', views.get_session),
    path('get_allsessions/', views.get_allsessions),
```

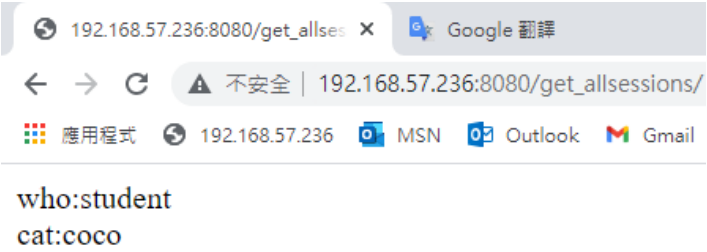
新增 view functions：

sudo vim CookieSessionApp/views.py

```
def get_allsessions(request):
    if request.session!=None:
        strsessions=""
        for key1,value1 in request.session.items():
            strsessions= strsessions + key1 + ":" + str(value1) + "<br>"
        return HttpResponse(strsessions)
    else:
        return HttpResponse('Session 不存在!')
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```



範例：利用 Session 防止灌票

利用 Session 可以防止灌票動作，可以重新整頁面幾次，或是關閉瀏覽器後重新啟動瀏覽，瀏覽的次數不會一直累加。

新增瀏覽位置(urls.py):

```
# sudo vim CookieSession/urls.py
```

```
16  from django.contrib import admin
17  from django.urls import path
18  from CookieSessionApp import views
19
```

```
35
36      path('vote/', views.vote),
37
```

新增 view functions：

```
# sudo vim CookieSessionApp/views.py
```

```
82  def vote(request):
83      if not "vote" in request.session:
84          request.session["vote"]=True
85          msg="您第一次投票!"
86      else:
87          msg="您已投過票!"
88      response = HttpResponse(msg)
89      return response
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```

- Session 的有效時間

參數	意義	預設值
SESSION_EXPIRE_AT_BROWSER_CLOSE	決定 session 是否在關閉時結束。若設為 True，在瀏覽器關閉後，該 session 將會自動結束。	False
SESSION_COOKIE_AGE	session(cookie)的有效時間	1209600 秒，兩週

EX:
注意:要將 cookie 刪除才可以測試
設定瀏覽器關閉時結束，Session 即自動結束➔
SESSION_EXPIRE_AT_BROWSER_CLOSE=True

設定 Session 最大的有效時間為 24 分➔
SESSION_COOKIE_AGE=1440

設定 Session 持續時間為 5 分➔
Request.session.set_expiry(50*60)

- 刪除 Session

刪除指定的 Session➔
del request.session[名稱]
刪除所有的 Session➔
del request.clear()

範例：設定 Session 持續時間

新增瀏覽位置(urls.py):

sudo vim CookieSession/urls.py

```
from django.contrib import admin
from django.urls import path
from CookieSessionApp import views

```

```
    path('set_session2/<str:key>/<str:value>', views.set_session2),

    path('delete_session/<str:key>', views.delete_session),
```

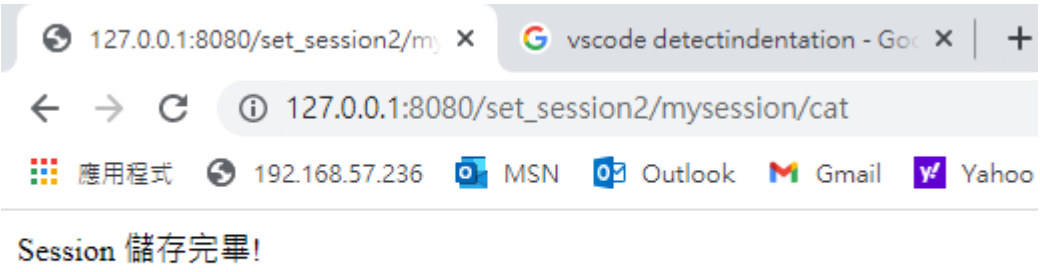
新增 view functions：

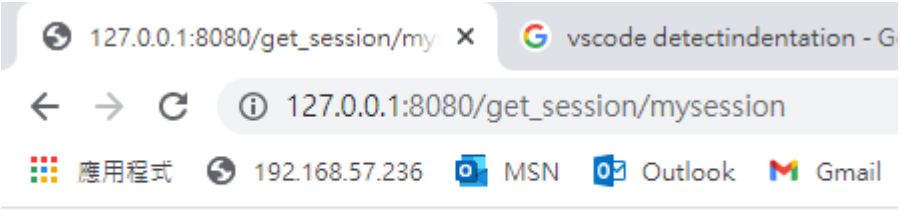
sudo vim CookieSessionAPP/views.py

```
def set_session2(request,key=None,value=None):
    response = HttpResponse('Session 儲存完畢!')
    request.session[key]=value
    request.session.set_expiry(30) # 設持續的時間為30秒
    return response

def delete_session(request,key=None):
    if key in request.session:
        response = HttpResponse('Delete Session: '+key)
        del request.session[key]
        return response
    else:
        return HttpResponse('No Session:' + key)
```

測試結果：

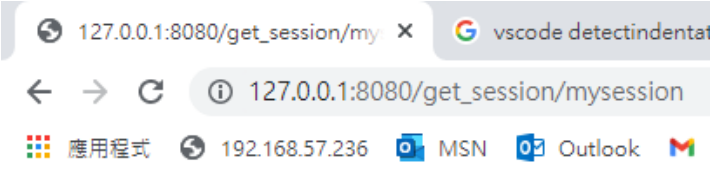
sudo python3 manage.py runserver 0.0.0.0:8080





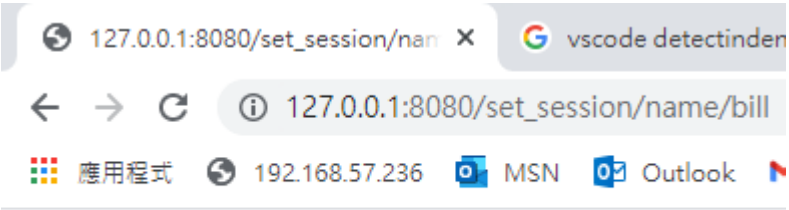
mysession : cat

30 秒後就自動消失：

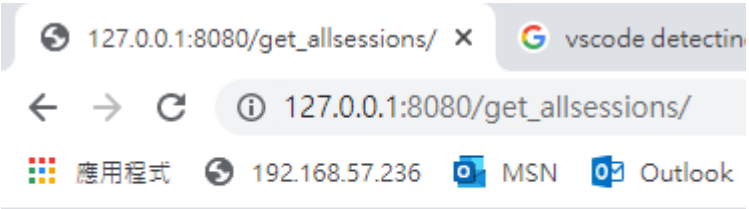


Session 不存在!

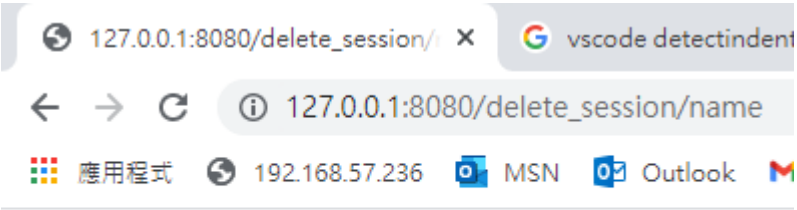
新增一個 seesion，再將 seesion 刪除



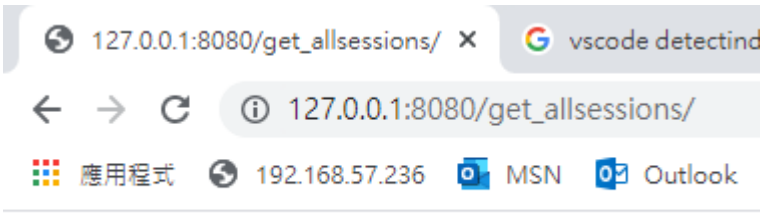
Session 儲存完畢!



name:bill



Delete Session: name



範例：會員系統的登入與登出

新增瀏覽位置(urls.py):

sudo vim CookieSession/urls.py

```
from django.contrib import admin
from django.urls import path
from CookieSessionApp import views

```

```
    path('login/',views.login),
    path('logout/',views.logout),

]
```

新增 view functions：

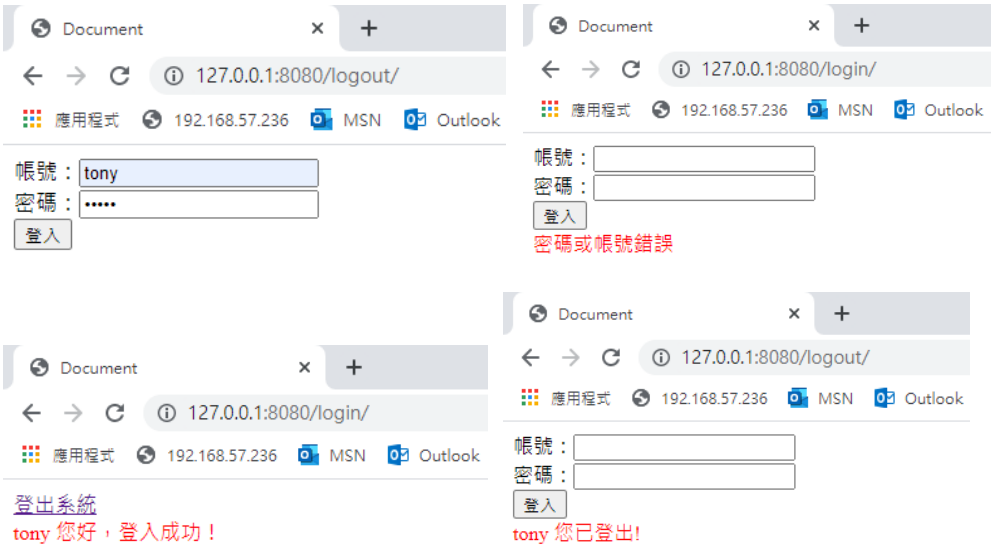
sudo vim CookieSessionAPP/views.py

```
def login(request):
    #預設帳號密碼
    username = "tony"
    password = "1234"
    if request.method == 'POST':
        if not 'username' in request.session:
            if request.POST['username']==username and request.POST['password']==password:
                request.session['username']=username #儲存Session
                message=username + " 您好，登入成功！"
                status="login"
            else:
                message="密碼或帳號錯誤"

    else:
        if 'username' in request.session:
            if request.session['username']==username:
                message=request.session['username'] + " 您已登入過了！"
                status="login"
    return render(request, 'login.html',locals())
```

```python
def logout(request):
    if 'username' in request.session:
        message=request.session['username'] + ' 您已登出!'
        del request.session['username'] #刪除Session
    return render(request, 'login.html',locals())
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```



3-15 使用者管理

在 django.contrib 套件的 auth 應用程式中，已內建，user 使用者的資料表，使用這個內建的資料表就可以記錄使用者的資訊。以 is_authenticated 可以檢使用者是否認證過。如果是 user 物件會傳回 true，而 AnonymousUser 物件則回傳 false。auth.lgoin()接收 request、user 兩個參數，登入成功後會產生一個 Session，因為這個 Session 的存在，使得該使用者可以跨頁面保存。auth.logout()可以進行登出動作，登出後，原來的 Session 將會被清除。

➢ 建立環境

- 建立 Django 專案：

```
# cd ~/dvds
# source ./dvds/dvdsenv/bin/activate (啟動虛擬環境)
# django-admin startproject login (建立 login 專案)
```

➢ 讀取 Django auth 使用者

使用 auth 就可以達到使用者驗證的動作，其中的 User 資料表就是使用者用戶的資料，可以利用 models 模組的 objects.get()和 objects.all()方法讀取資料。

範例：輸入會員姓名，判斷是否為使用者用戶

新增瀏覽位置(urls.py):

```
# sudo vim login/urls.py
```

```
from django.contrib import admin
from django.urls import path
from loginapp import views
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('search_name/<str:name>', views.search_name),
]
```

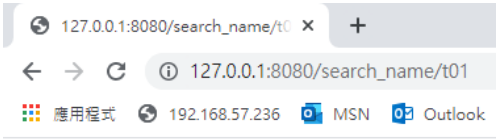
新增 view functions：

```
# sudo vim loginapp/views.py
```

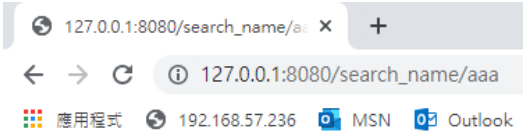
```python
from django.shortcuts import render
from django.contrib.auth.models import User
from django.http import HttpResponse

def search_name(request,name=None):
    try:
        #print(name)
        user = User.objects.get(username=name)
        #print(dir(user))
        print(user.username)
        print(user.email)
        return HttpResponse('存在')
    except:
        return HttpResponse('不存在!')
```

測試結果：

```
# sudo python3 manage.py runserver 0.0.0.0:8080
```



存在



不存在!

➢ HttpRequest.user 物件

HttpRequest 物件中包含了一個 user 屬性，利用 HttpRequest.user 可以取得一個 User 物件或是一個 AnonymousUser 物件，利用這個物件可以判斷是否已經登錄。如果是 User 物件代表該使用者已經登入，就是具名用戶，如果是 AnonymousUser 物件代表該使用者未登入，就是匿名用戶。

```python
if request.user.is_authenticated:
    ... # Do something for logged-in users.
else:
    ... # Do something for anonymous users.
```

User 物件中的常用屬性:

屬性	說明
username	使用者的帳號，由字母、數字和底線組成
is_anonymous	是否是匿名用戶，永遠回傳 False，若為 AnonymousUser 物件，則永遠回傳 True
is_authenticated	用戶是否認證過，永遠回傳 True，若為 AnonymousUser 物件，則永遠回傳 False
first_name	名字
last_name	姓氏
email	電子郵箱
password	加密(經編碼)過後的密碼
is_staff	真假值，若為 True，該用戶可登入 admin 後端
is_active	真假值，若為 True，該用戶可登入
is_superuser	真假值，若為 True，該用戶擁有全權限
last_login	用戶上一次登入的日期與時間
date_joined	用戶被創建的日期與時間

is_active 屬性可以設定該使用者是否有效，若設定為 False 該帳號將失敗，即使帳號、密碼均正確以 auth.authenticate 驗證仍會傳回 None。如果帳號不使用，建議以 is_active=False 設定讓帳號失效，而不要直接刪除帳號。

User 物件中的常用方法:

屬性	說明
get_username()	取得用戶帳號
get_full_name()	回傳完整的姓名
get_short_name()	只回傳名字
set_password(password)	設定密碼，會自動編碼加密，不包含 User 物件的儲存
check_password(password)	確認密碼，正確會回傳 True，會自動編碼加密才比較

範例：新增使用者

新增瀏覽位置(urls.py):

sudo vim login/urls.py

```
from django.contrib import admin
from django.urls import path
from loginapp import views
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('search_name/<str:name>', views.search_name),

    path('index/', views.index),

    path('adduser/', views.adduser),

]
```

新增 view functions：

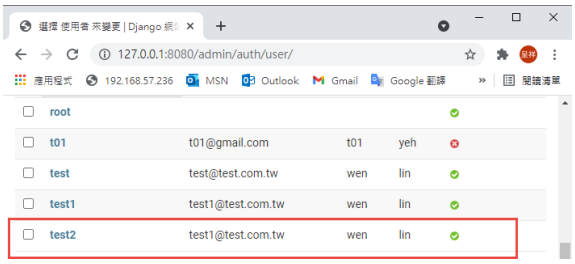
sudo vim loginapp/views.py

```
from django.shortcuts import redirect

def adduser(request):
    try:
        user=User.objects.get(username="test1")
    except:
        user=None

    if user!=None:
        message = user.username + " 帳號已建立!"
        return HttpResponse(message)
    else:   # 建立 test 帳號
        user=User.objects.create_user("test1","test1@test.com.tw","aa123456")
        user.first_name="wen" # 姓名
        user.last_name="lin"  # 姓氏
        user.is_staff=True  # 工作人員狀態
        user.save()
        return redirect('/admin/')
```

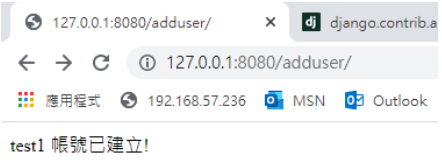
測試結果：

sudo python3 manage.py runserver 0.0.0.0:8080

登入 127.0.0.1:8080/useradd/，畫面如下



再重新整理，畫面如下：



➢ 登入和登出

登入驗證：

利用 auth 物件內的 authenticate 方法，可以完成使用者登入驗證。

user= auth.authenticate (username=帳號, password='密碼')

若密碼正確，會回傳具名的 User 物件，否則回傳 None。

user.is_active 可檢查帳戶是否有效，若 False 表示失效，即使帳號密碼正確，以 auth. authenticate 驗證仍會傳回 None。

驗證用戶

authenticate(*request=None* , ***credentials* **)**

使用**authenticate()**來驗證用戶。它使用**username**和**password**作為參數來驗證，對每個身份驗證後端(authentication backend `)進行檢查。如果後端驗證有效，則返回一個:class:`~django.contrib.auth.models.User 對象。如果後端引發**PermissionDenied**錯誤，將返回**None**。舉例：

```
from django.contrib.auth import authenticate
user = authenticate(username='john', password='secret')
if user is not None:
    # A backend authenticated the credentials
else:
    # No backend authenticated the credentials
```

request是可選的**HttpRequest**，它在身份驗證後端上的**authenticate()**方法來傳遞。

登入：

auth.login(request, user)

登入成功後，會產生 session，因為這個 session 的存在，使用該使用者可以跨頁面保存，直到登出時刪除該 session 為止。通過驗證使用者以 user 物件進行登入。

```
If user.is_active:
    auth.login(request, user)
```

登出：

auth.logout()可進行登出動作，登出後，原來的 session 即會被清除。

```
auth.logout(request)
```

範例：使用者登入和登出

新增瀏覽位置(urls.py):

sudo vim login/urls.py

```
from django.contrib import admin
from django.urls import path
from loginapp import views
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('search_name/<str:name>', views.search_name),

    path('adduser/', views.adduser),

    path('index/', views.index),
    path('login/', views.login),
    path('logout/', views.logout),
]
```

新增 view functions：

sudo vim loginapp/views.py

```
def index(request):
    #判斷用戶是否認證過
    if request.user.is_authenticated:
        name=request.user.username
    return render(request, "index.html", locals())

```

```python
from django.contrib import auth
def login(request):
    if request.method == 'POST':
        name = request.POST['username']
        password = request.POST['password']
        #利用auth物件內的authenticate方法，可以完成使用者登入驗證
        user = auth.authenticate(username=name, password=password)
        if user is not None:
            if user.is_active:
                auth.login(request,user)
                message = '登入成功！'
            else:
                message = '帳號尚未啟用！'
        else:
            message = '登入失敗！'
        return render(request, "index.html", locals())
    else:
        return render(request, "login.html", locals())

def logout(request):
    auth.logout(request)
    return redirect('/index/')
```

auth.authenticate()此方法預設，若帳號或密碼錯誤，或者帳號與密碼正確，但未啟動(user.is_active=False)，會傳回 None。因此無法判斷帳號密碼錯誤或未啟動的差異，因此做以下修改即可判斷之間的差別。

新增參數(settings.py):

```
# sudo vim login/settings.py

AUTHENTICATION_BACKENDS =

['django.contrib.auth.backends.AllowAllUsersModelBackend']
```

views.py settings.py urls.py login.html index.html

login > settings.py

AUTHENTICATION_BACKEN 1 of 1

```python
# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'


AUTHENTICATION_BACKENDS = ['django.contrib.auth.backends.AllowAllUsersModelBackend']
```

參考文獻：

https://www.cnblogs.com/yoyoketang/p/13192138.html

新增 template index.html：

sudo vim templates/index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"
    <title>首頁</title>
</head>
<body>
    <h2>文淵閣首頁</h2>
    {% if request.user.is_authenticated %}
        歡迎光臨:{{name}}
        <p><a href="/logout/">登出</a></p>
    {% else %}
        <p>您尚未登入喔~</p>
        <a href="/login/">登入</a>
    {% endif %}
    <span style="color:red">{{message}}</span>
</body>
</html>
```

新增 template login.html：

sudo vim templates/login.html

```
<html>
<form action="." method="POST" name="form1">
    {% csrf_token %}
    <div>帳號：<input type="text" name="username" id="username" autocomplete="off"></div>
    <div>密碼：<input type="password" name="password" id="password"/></div>
    <div>
    <input type="submit" name="button" id="button" value="登入" />
    </div>
    <span style="color:red">{{message}}</span>
</form>
</html>
```

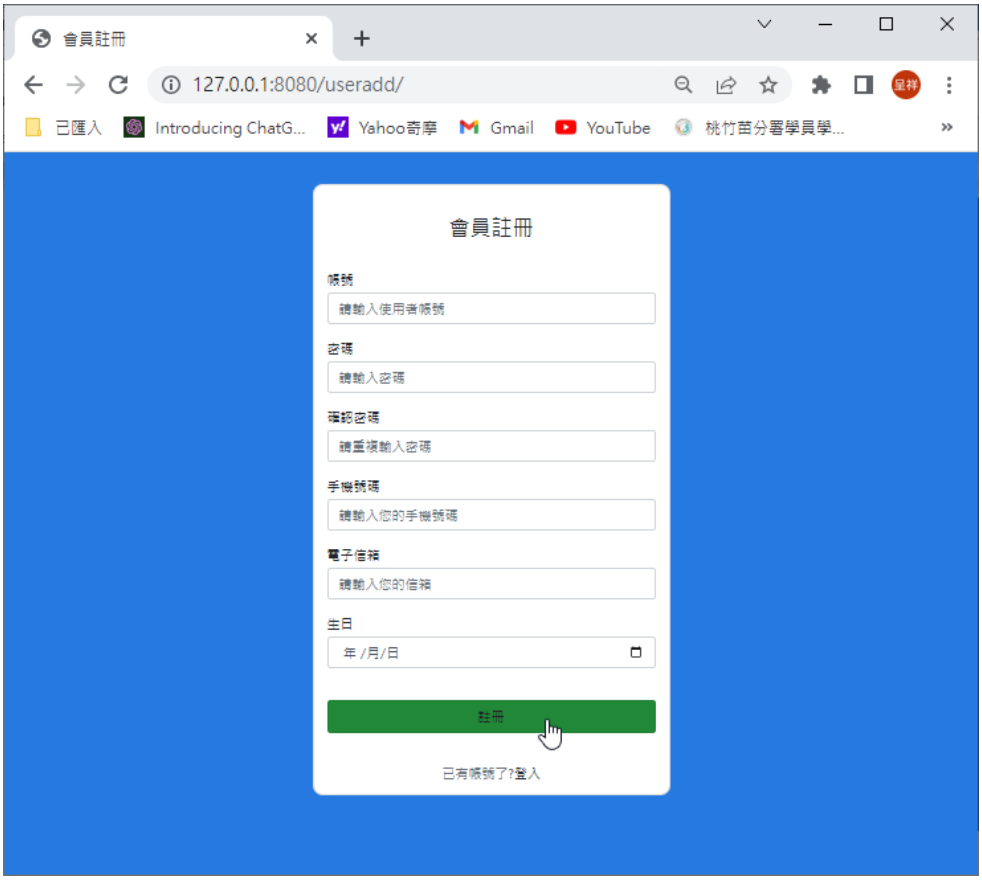
測試結果：

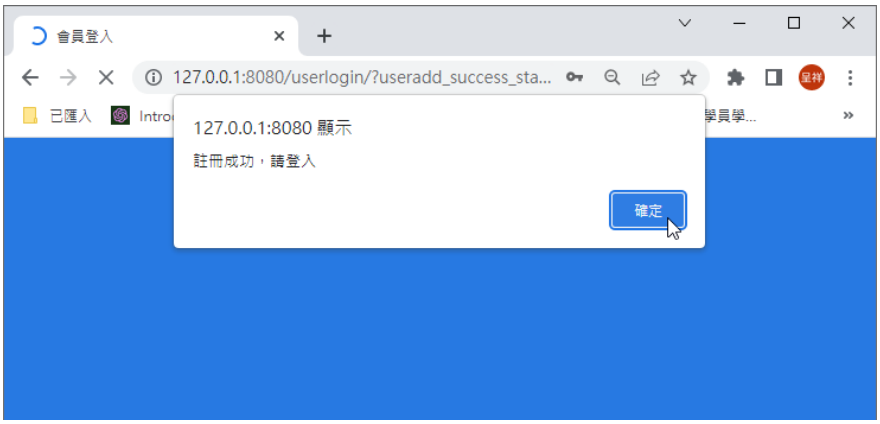
sudo python3 manage.py runserver 0.0.0.0:8080

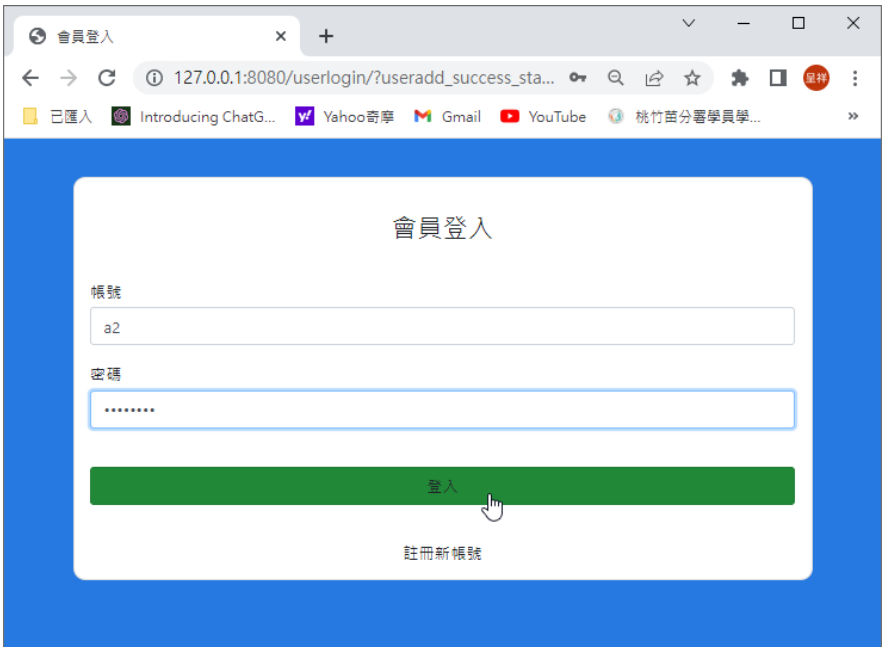
➢ Registration 小專案(會員註冊與登入)

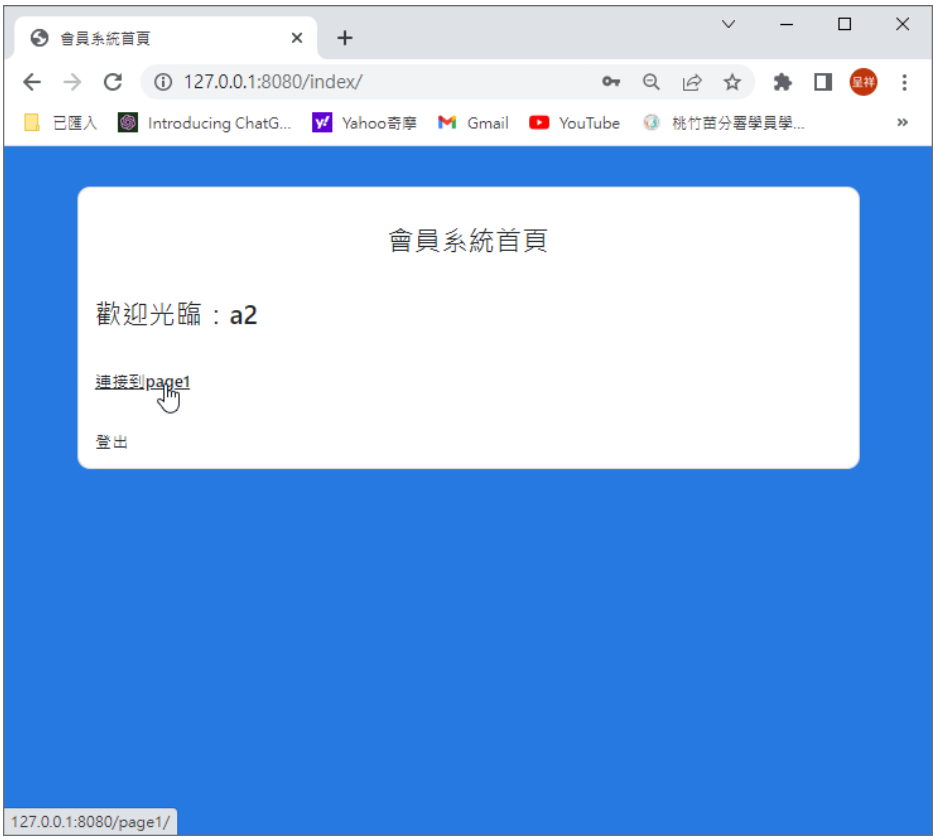
結果如下：

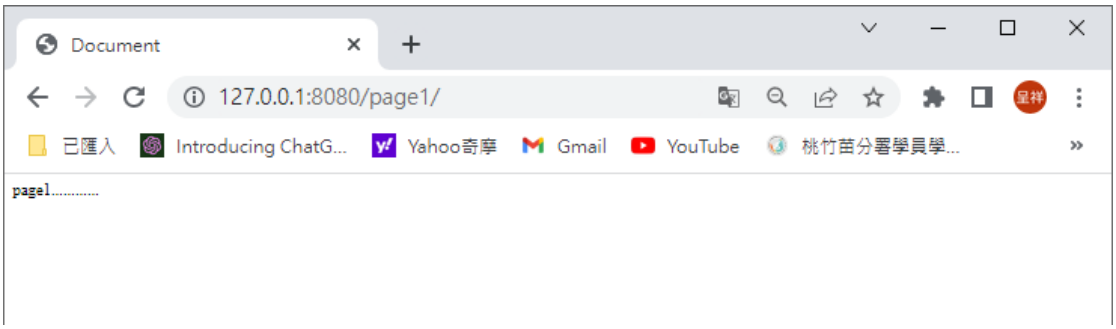
會員系統首頁
127.0.0.1:8080/index/
已匯入
Introducing ChatG...
Yahoo奇摩
Gmail
YouTube
桃竹苗分署學員學...
會員系統首頁
您尚未登入
已有帳號？登入
尚未申請帳號？加入會員


會員註冊
127.0.0.1:8080/useradd/
已匯入
Introducing ChatG...
Yahoo奇摩
Gmail
YouTube
桃竹苗分署學員學...
會員註冊
帳號
請輸入使用者帳號
密碼
請輸入密碼
確認密碼
請重複輸入密碼
手機號碼
請輸入您的手機號碼
電子信箱
請輸入您的信箱
生日
年 /月/日
註冊
已有帳號了?登入


會員登入
127.0.0.1:8080/userlogin/?useradd_success_sta...
127.0.0.1:8080 顯示
註冊成功，請登入
確定


會員登入
127.0.0.1:8080/userlogin/?useradd_success_sta...
已匯入
Introducing ChatG...
Yahoo奇摩
Gmail
YouTube
桃竹苗分署學員學...
會員登入
帳號
a2
密碼
登入
註冊新帳號





- Layout 參考

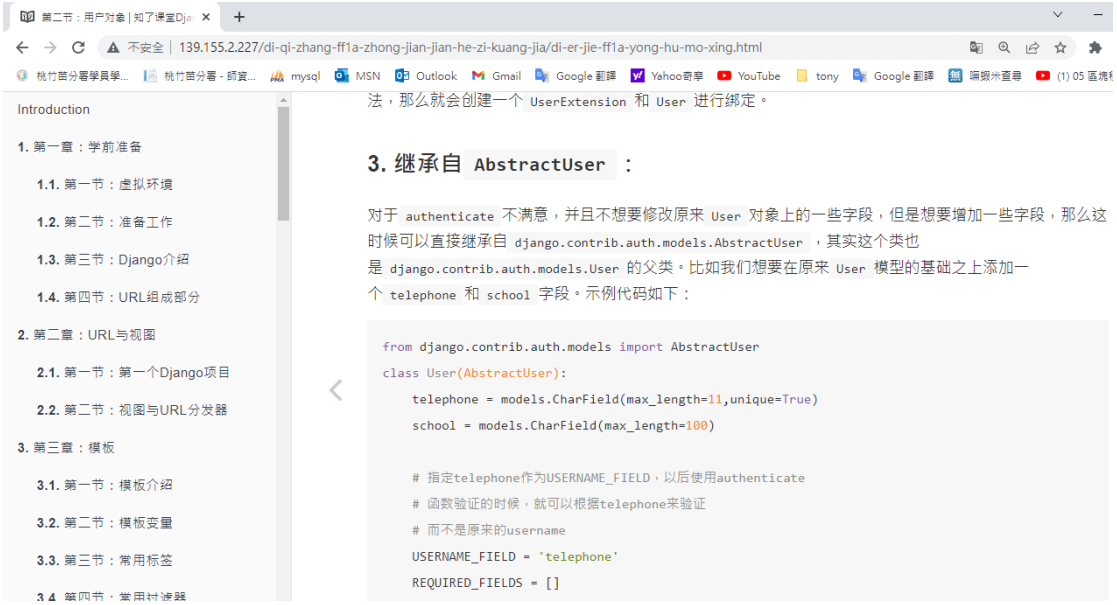
1、HTML/CSS 寫的簡單的註冊頁面

https://www.796t.com/content/1545570916.html

2、Registration form Bootstrap 5 Registration form componen

https://mdbootstrap.com/docs/standard/extended/registration/#docsTabsOverview

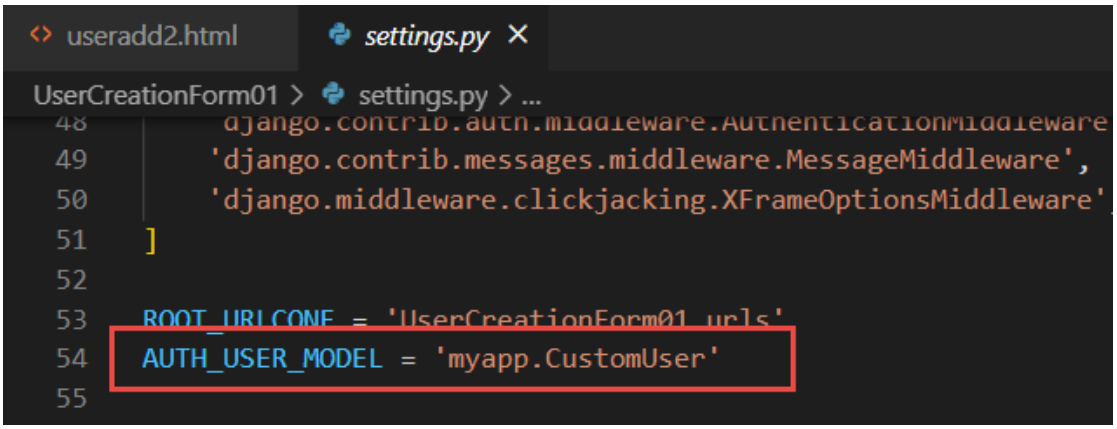
- How to add phone number to django user model?



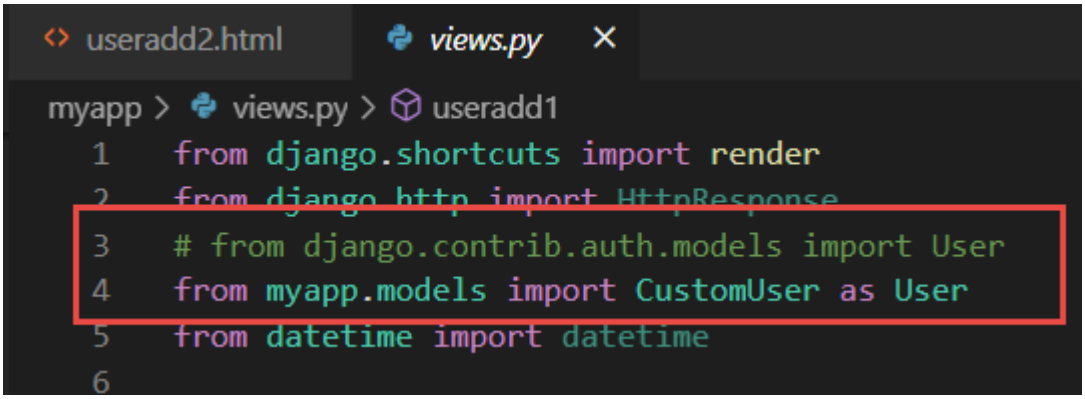
http://139.155.2.227/di-qi-zhang-ff1a-zhong-jian-jian-he-zi-kuang-jia/di-er-jie-ff1a-yong-hu-mo-xing.html

注意，自訂 User 資料表後，要新增兩個位置要新增參數:

1、



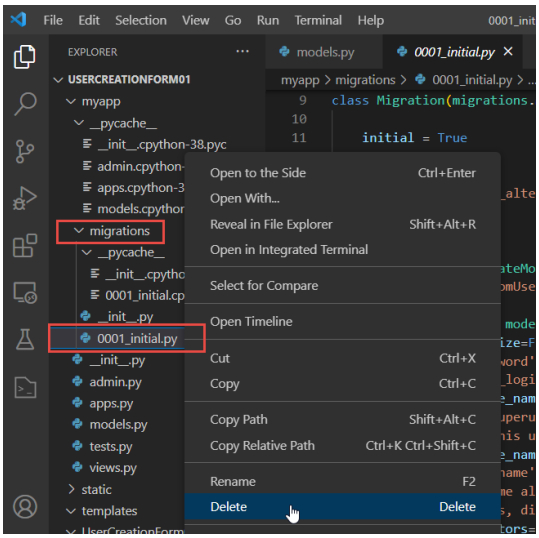
2、



- Django 變更模型(Models)過程中易出現的問題及解決方案

1、Adding custom fields to users in Django

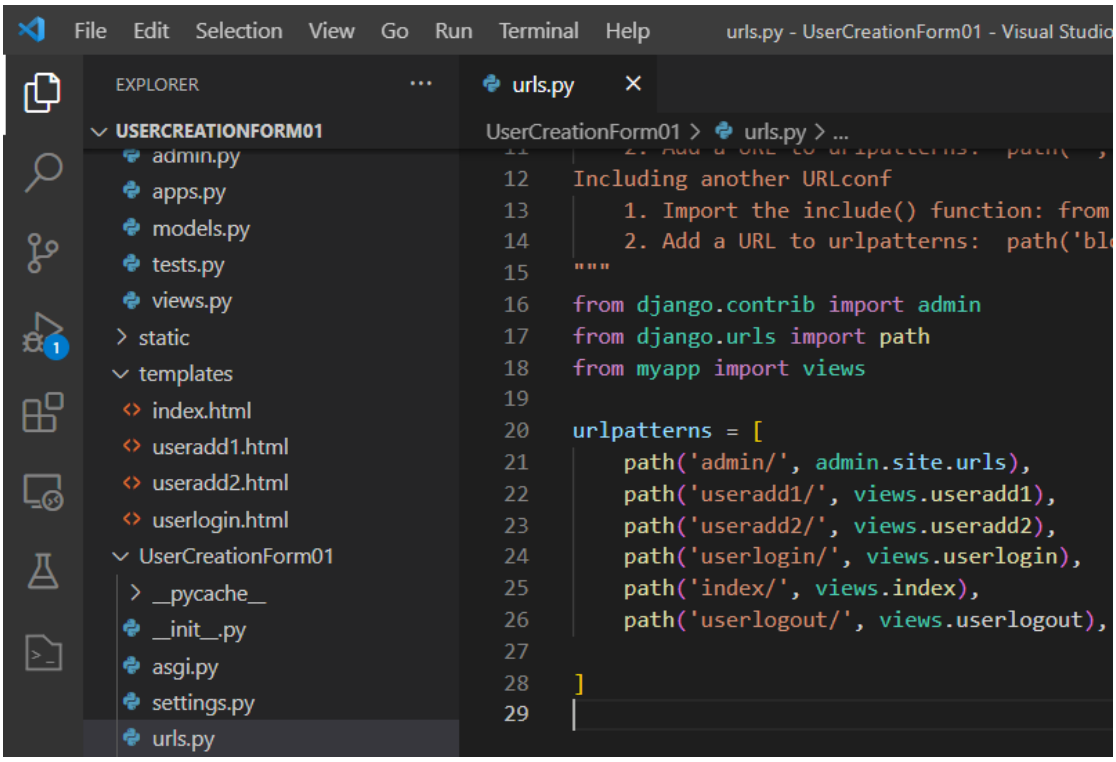
```
(dvds) c:\dvds\UserCreationForm01>python manage.py makemigrations myapp
Did you rename customuser.phone to customuser.tel (a CharField)? [y/N] y
You are trying to add a non-nullable field 'cBirthday' to customuser without a default; we can't do that (the d
atabase needs something to populate existing rows).
Please select a fix:
 1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
 2) Quit, and let me add a default in models.py
Select an option:
```



刪除資料庫，再用以下指令重建

```
(dvds) c:\dvds\UserCreationForm01>python manage.py makemigrations myapp
Migrations for 'myapp':
  myapp\migrations\0001_initial.py
    - Create model CustomUser
```

```
(dvds) c:\dvds\UserCreationForm01>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, myapp, sessions
Running migrations:
  No migrations to apply.
```



範例:專案 UserCreationForm01

Bootstrap 4:

Colors:
https://getbootstrap.com/docs/4.0/utilities/colors/

```
{% comment %}

mt-4:margin-top

mb-5:margin-bottom

text-uppercase: 本文轉換大寫字母

text-muted:用于将元素中的文本设置为灰色

d-flex：這個類表示該元素採用 Flexbox 佈局，可以用於實現靈活的佈局方案。

justify-content-center：這個類表示該元素的 Flexbox 容器採用水平居中對齊方式，使其子元素在水平方向上居中對齊。

bg-danger:表示將該元素的背景色設置為"danger"主題顏色，即紅色。
```

btn：這個類表示該元素是一個按鈕。

btn-success：這個類表示該按鈕採用定義的 "success" 主題顏色，即綠色。

btn-block：這個類表示該按鈕採用塊級元素（block element）的顯示方式，佔據整個容器的寬度，以便更好地適應不同的屏幕尺寸。

btn-lg：這個類表示該按鈕採用大號（large）的樣式。

text-body：這個類表示該按鈕中的文本採用默認的文字顏色和字體大小，通常是 Bootstrap 中定義的黑色和 16px。

{% endcomment %}