

學 Python 玩創客

Rewritten by

賴秉樑

私立極限翻轉文理電腦補習班 班主任

大學程式教師

華盛頓中學程式教師

課程網址 <https://max543.com/debugger>

書商原始投影片下載網址 <http://bit.ly/pptfm610a>



PERSONAL INFO

姓名 NAME

賴秉樑 debugger

學歷 EDUCATION

台科大資工、電子雙學位
中興資訊科學與工程碩士

經歷 EXPERIENCE

國立大學電子系、資工系講師
職業訓練、產投電腦講師
竹科半導體研發工程師

個人興趣 INTEREST

玩數學、打電腦

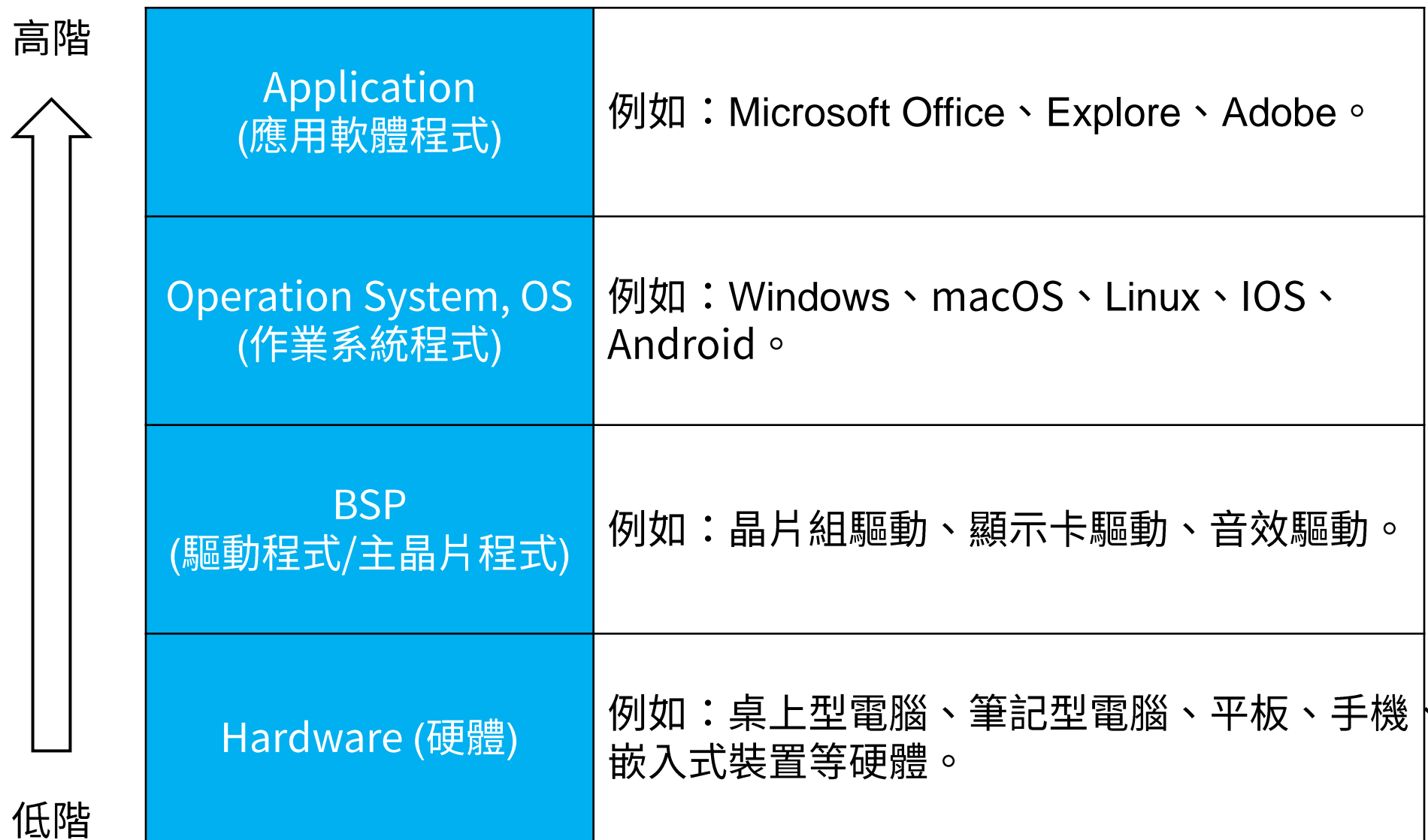
我的座右銘 MOTTO

動手做、樂趣多

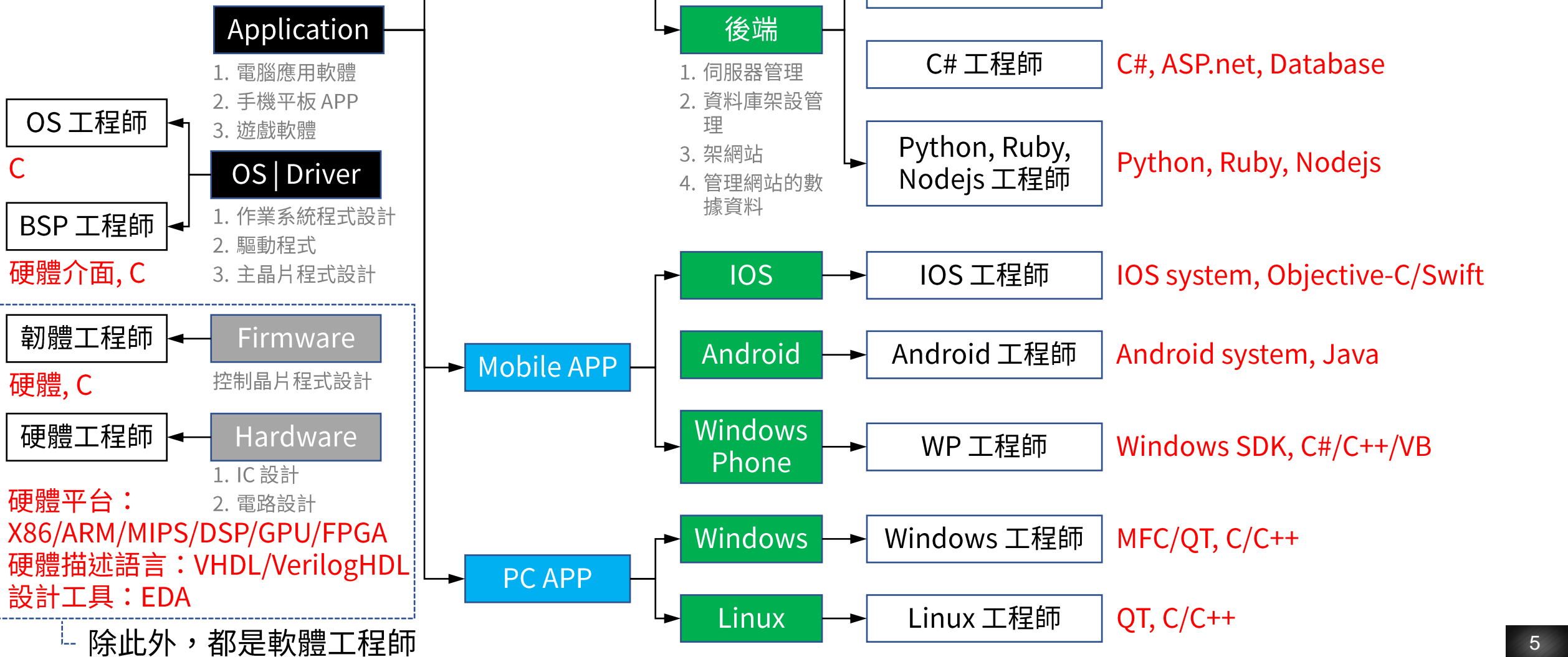
Coder, Hacker, and Maker

- 程式設計師 (programmer, 或 coder)
 - ✓ 主要透過編輯程式，簡稱編程 (coding)，它可以指在程式設計**某個專業領域的專業人士**，或是從事軟體撰寫，程式開發、維護的專業人員。
- 駭客 (hacker)
 - ✓ 除了**精通**程式設計、作業系統的人可以被視作駭客，對硬體裝置做創新的工程師通常也被認為是駭客，精通網路入侵的人也被看作是駭客。
- 創客 (maker)
 - ✓ 又稱自造者。是一群酷愛科技、熱衷實踐的人群，他們以分享技術、**激發的創造力**與交流思想為樂。

從低階硬體，到高階應用程式的分類



科技業工程師 主要分類



運算思維為何很重要？ (1/2)

- 學會了運算思維，讓我們也能擁有電腦科學家面對問題時，所持有的科學方法。各種領域都需要運算思維，例如：
- **科學與工程領域**
 - ✓ 利用運算模擬建築結構，以確認安全性。
 - ✓ 利用運算預測氣象，以增加準確性。
- **金融領域**
 - ✓ 利用運算研究經濟大數據。
 - ✓ 利用運算完成自動交易。

運算思維為何很重要？ (2/2)

- **人文與社會領域**

- ✓ 利用運算分析，並優化廣告投放策略。
- ✓ 利用運算分析人口老化趨勢，與醫療資源分布。

- **藝術領域**

- ✓ 利用運算建構三維動畫。
- ✓ 利用運算創作數位音樂。

- **工業設計**

- ✓ 利用運算實現工業 4.0。
- ✓ 利用運算實現更多的增值應用，例如：自動化與智慧化。

科技業的下一個未來

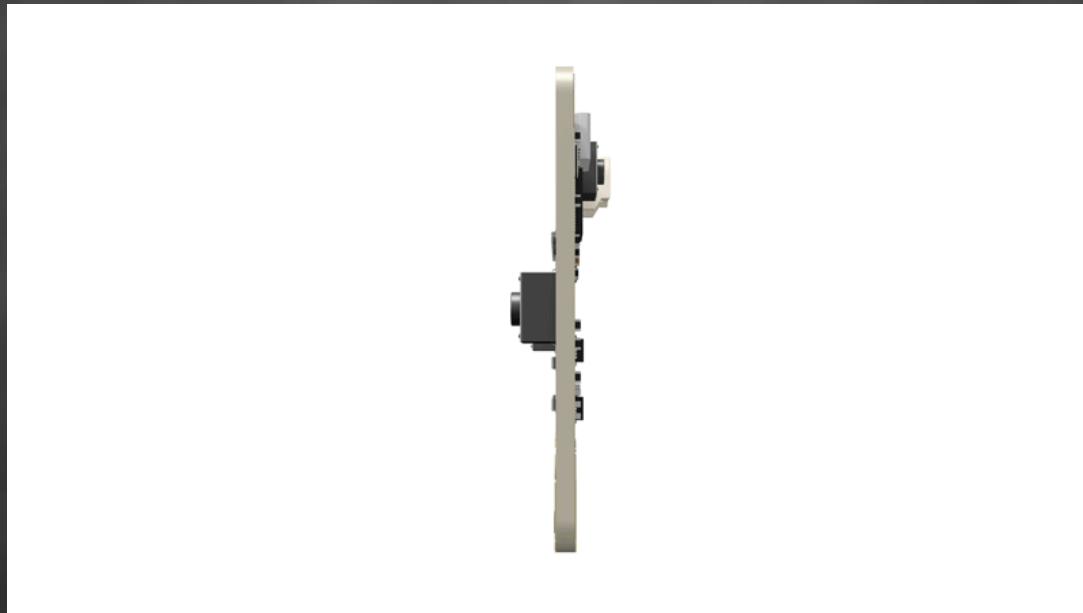
電子商務 X 社群網路 X 串流媒體



物聯網 X 人工智慧 X 區塊鏈

主流的創客材料：micro:bit (適合國小生)

- micro:bit 是一塊沒有外殼的開發板。由英國廣播公司 (BBC) 設計用於英國的青少年程式教育。具備以下特點：
 1. 體積小、耗電低、便宜，主控板市價約 450~550 元，配件也很便宜。
 2. 主控板基本功能完整，可額外結合許多硬體，創造更多樂趣。
 3. 能夠使用積木式程式 (Blocks)、JavaScript 或 MicroPython 編寫。



主流的創客材料：Arduino（適合小四 ~ 玩家）

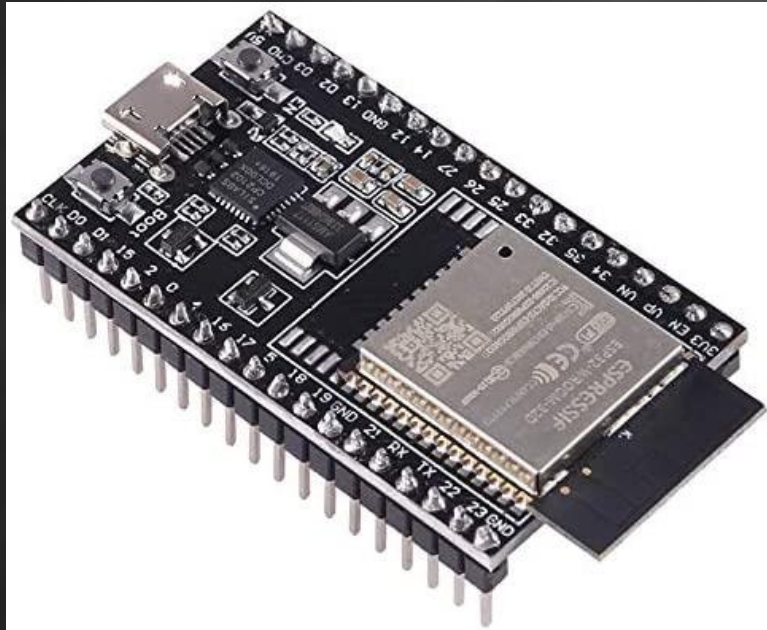
- Arduino 是一家製作開源硬體和開源軟體的公司，該公司負責設計和製造單板微控制器和微控制器套件，用於構建數位裝置和互動式物件。



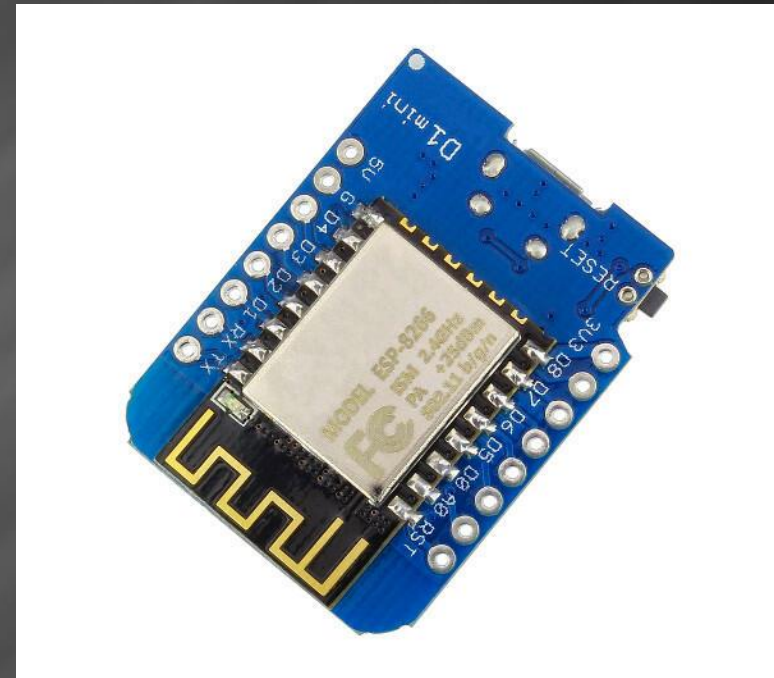
Arduino Uno SMD R3

主流的單晶片：ESP 系列（適合小四 ~ 玩家）

- ESP 系列由上海樂鑫信息科技所開發，基於這個 Wi-Fi IoT 晶片發展出的開發套件系列，這一、兩年紅透半邊天，甚至給其他通訊晶片大廠很大的壓力。



ESP32



ESP8266 (D1 mini)

主流的創客材料：樹莓派（適合專業玩家）

- 樹莓派（Raspberry Pi），簡稱 pi，是基於 Linux 的單板電腦，由英國樹莓派基金會開發。目的是以低價的硬體，及自由軟體促進學校的電腦科學教育，使得軟體開發變得非常上手。



因為有這些程式
生活更美、更好



Hello, World! 最像英文的 **Python**



Python

```
# 印出 Hello World! 字串物件  
print("Hello World!")
```

C

```
/* 印出 Hello World! 字串物件*/  
include <stdio.h>  
  
int main()  
{  
    printf("Hello, World!\n");  
    return 0;  
}
```

C++

```
//印出 Hello World! 字串物件  
#include <iostream>  
  
using namespace std;  
  
int main()  
{  
    cout << "Hello World" << endl;  
    return 0;  
}
```

Java

```
//印出 Hello World! 字串物件  
public class HelloWorld{  
  
    public static void main(String []args){  
        System.out.println("Hello World");  
    }  
}
```

程式語言：五大語法結構



Sequence

循序執行



Conditional Statements

if 判斷式



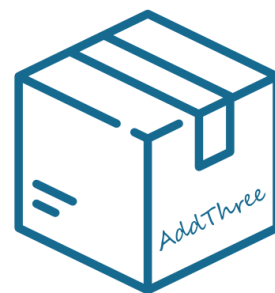
Loops

迴圈



Input / Output

輸入/輸出



Functions

函數

程式設計師是高技能的職業

- 可類比於作曲家：一首小蜜蜂與命運交響曲的差異。



今日議程

時間	主題
09:10—09:30	01：課程簡介與軟體環境設定 D1 mini 與 Tonny 開發環境
09:30—10:00	02：電子電路與 Python 基礎 電壓、電流、物件、資料型別、變數
10:00—10:30	03：控制 LED 亮暗 - 數位輸出 Python 流程控制 (while 迴圈) 與區塊縮排
10:30—10:40	休息
10:40—11:20	04：讀取按鈕 - 數位輸入 Python 流程控制 (if...else)
11:20—12:00	05：光感應自動電燈 - 類比輸入 Python 流程控制 (if...else)

今日議程

時間	主題
13:10—13:50	06：LED 呼吸燈 - 類比輸出 Python 流程控制 (for 迴圈)
13:50—14:30	07：霹靂車跑馬燈 Python 資料的容器：串列 (list)
14:30—14:40	休息
14:40—15:20	08：電子鋼琴 Python 資料的容器：字典 (dictionary)
15:20—16:00	09：氣象預報站 Python 網路爬蟲

我們要使用的套件



01 課程簡介與軟體環境設定

D1 mini 與 Tonny 開發環境

程式設計課程教學 Python 的問題

- 太難，一般程式語言書籍有滿滿語法
- 學了不知道要幹麻
- 教一堆語法學生不一定能完全吸收，讓程式語言課程淪為語法背誦課程
- Python 最像英文，最好學

Python

```
# 印出 Hello World! 字串物件  
print("Hello World!")
```

C

```
/* 印出 Hello World! 字串物件*/  
include <stdio.h>  
  
int main()  
{  
    printf("Hello, World!\n");  
    return 0;  
}
```

C++

```
//印出 Hello World! 字串物件  
#include <iostream>  
  
using namespace std;  
  
int main()  
{  
    cout << "Hello World" << endl;  
    return 0;  
}
```

Java

```
//印出 Hello World! 字串物件  
public class HelloWorld{  
  
    public static void main(String []args){  
        System.out.println("Hello World");  
    }  
}
```

本課程套件的特點

- 只要用精簡的 Python 語法，即可創造有趣的創客應用
- 軟硬搭配，程式效果立刻實體呈現
- Learning by doing 體驗式學習
- 看得到效果更能加深印象，體會其中的邏輯運作原理
- 期末學生自行發想做出生活應用
- 電子硬體接線圖以實物呈現

7個主題 + 13個實驗 + 延伸練習

期末專題

按鈕控制跑馬燈速度
光感應鬧鐘
氣象警報器

2 開發環境 / 2

3 控制 LED 亮暗 - 數位輸出 / 12

Lab 01 實作：點亮 / 熄滅 LED / 17

Lab 02 實作：閃爍 LED / 20

4 讀取按鈕 - 數位輸入 / 22

Lab 03 實作：讀取觸控按鈕的輸入值 / 23

Lab 04 實作：用觸控按鈕控制 LED / 26

5 光感應自動電燈 - 類比輸入 / 27

Lab 05 實作：讀取光敏電阻的輸入值 / 28

Lab 06 實作：光感應自動電燈 / 30

6 LED 呼吸燈 - 類比輸出 / 31

Lab 07 實作：漸亮 LED 燈 / 33

Lab 08 實作：LED 呼吸燈 / 34

7 霹靂車跑馬燈 / 36

Lab 09 實作：單向 LED 跑馬燈 / 38

Lab 10 實作：雙向 LED 跑馬燈 / 39

8 電子鋼琴 / 41

Lab 11 實作：嗡嗡翁 -- 小蜜蜂音樂 / 42

Lab 12 實作：電子鋼琴 / 43

9 網路連線 / 46

Lab 13 實作：氣象預報站 / 56

「用以致學」強過「學以致用」，問題導向式教育為發展重點

由於AI的自動化將取代許多例行性的工具，具有開創性、能夠靈活在工作上運用知識的人才變得更加珍貴。因此，林百里強調未來的教育應該以問題導向式的教學模式為主，讓學生以解決某個問題為出發點去學習，讓他們不斷與環境互動並進行決策，失敗了再繼續修正解決方案，才是培育未來AI人才的關鍵點。

翻轉 Python 的教學模式

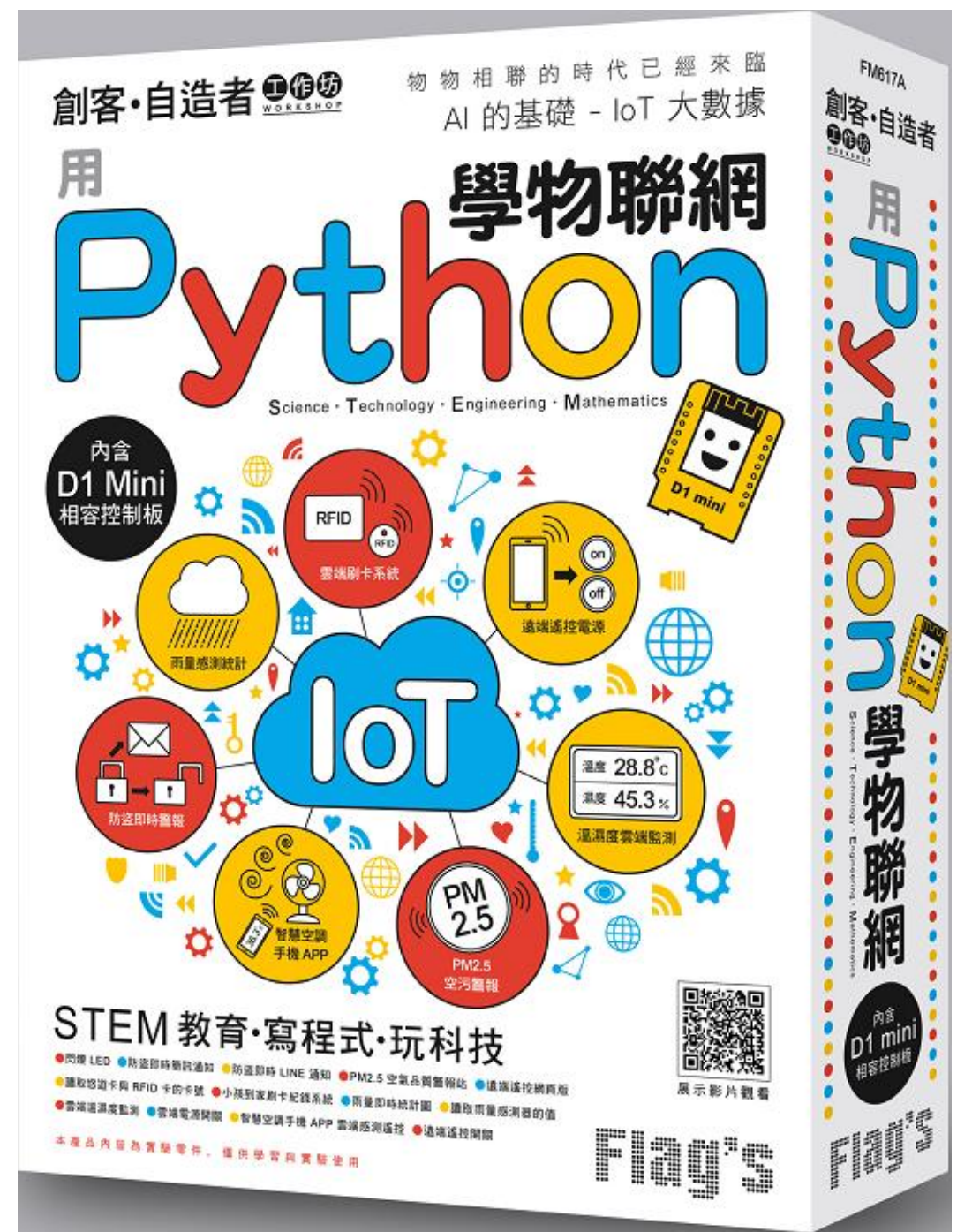


學軟體來控制硬體
讓學生搶著修Python的課

課程建議

週數	課程主題	FT707章節	FM610A章節
1	課程介紹		
2	Python 概論與 Thonny 開發環境安裝	Ch0	Ch1
3~4	資料型別、變數	Ch1、Ch2	Lab1
5~6	流程控制 (while 迴圈)	Ch7前半	Lab2
7~9	流程控制 (if 判斷式)	Ch6	Lab3~6
10	期中考		
11~12	流程控制 (for 迴圈)	Ch7後半	Lab7、8
13~14	資料結構 (list)	Ch3、Ch4	Lab9、10
15~16	資料結構 (dictionary)	Ch3、Ch4	Lab11、12
17~18	網路爬蟲	Ch10	Lab13
19~20	期末專題		

後續的課程套件



1

物聯網與 Python 簡介 / 2

Lab 01 實作：點亮 / 熄滅 LED / 12

Lab 02 實作：閃爍 LED / 14

2

防盜即時警報器 - 手機簡訊、LINE 即時通知 / 16

Lab 03 實作：讀取振動感測模組的輸入值 / 16

Lab 04 實作：防盜即時簡訊通知 / 20

Lab 05 實作：防盜即時 LINE 通知 / 24

3

PM2.5 空污警報燈 - Open Data 網路爬蟲 / 25

Lab 06 實作：PM2.5 空氣品質警報站 / 30

4

RFID 刷卡紀錄 - 雲端資料庫 / 32

Lab 07 實作：讀取悠遊卡與 RFID 卡的卡號 / 33

Lab 08 實作：小孩到家刷卡紀錄系統 / 37

5

雨量即時統計圖 - 雲端數位儀表板 / 39

Lab 09 實作：讀取雨水感測模組的值 / 40

Lab 10 實作：雨量即時統計圖 / 42

6

手機 APP 雲端資訊互通 - MQTT 發佈 / 訂閱訊息 / 46

Lab 11 實作：讀取溫濕度感測值 / 47

Lab 12 實作：雲端溫濕度監測 / 50

Lab 13 實作：雲端電源開關 / 53

7

手機 APP 感測遙控 - Blynk 自訂介面手機 APP / 56

Lab 14 實作：智慧空調手機 APP 雲端感測遙控 / 58

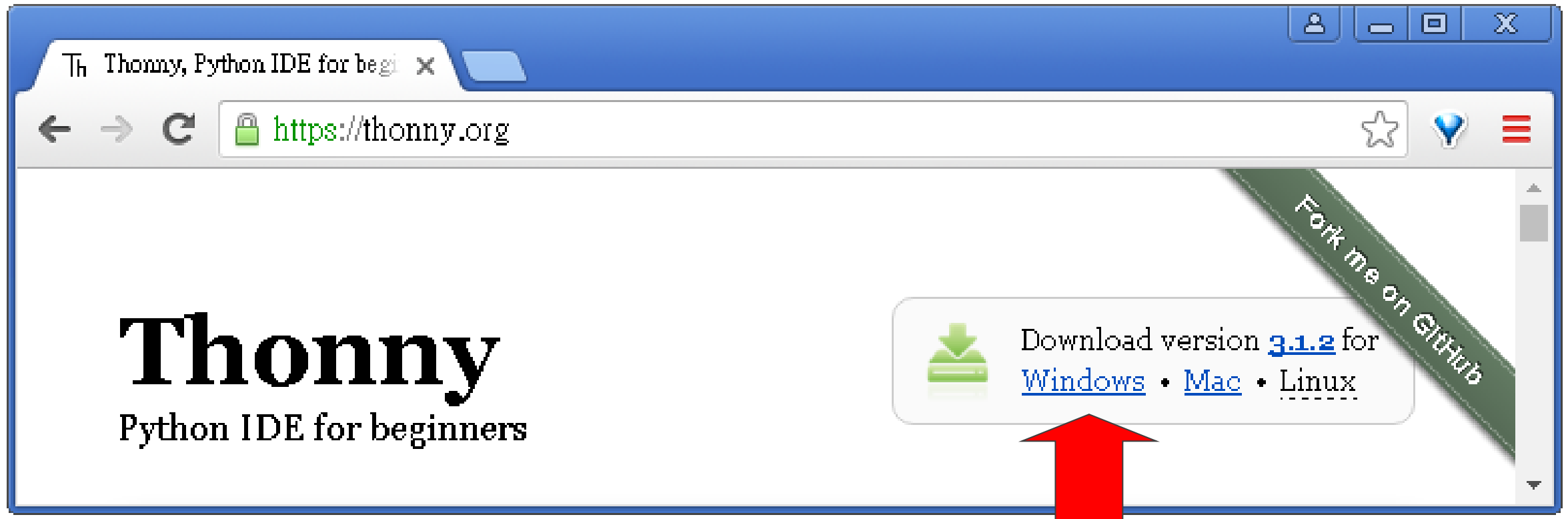
8

自製雲端平台 - 用網頁遙控家電 / 64

Lab 15 實作：遠端遙控開關 / 66

安裝 Thonny 開發環境

- 下載網址：<https://thonny.org>

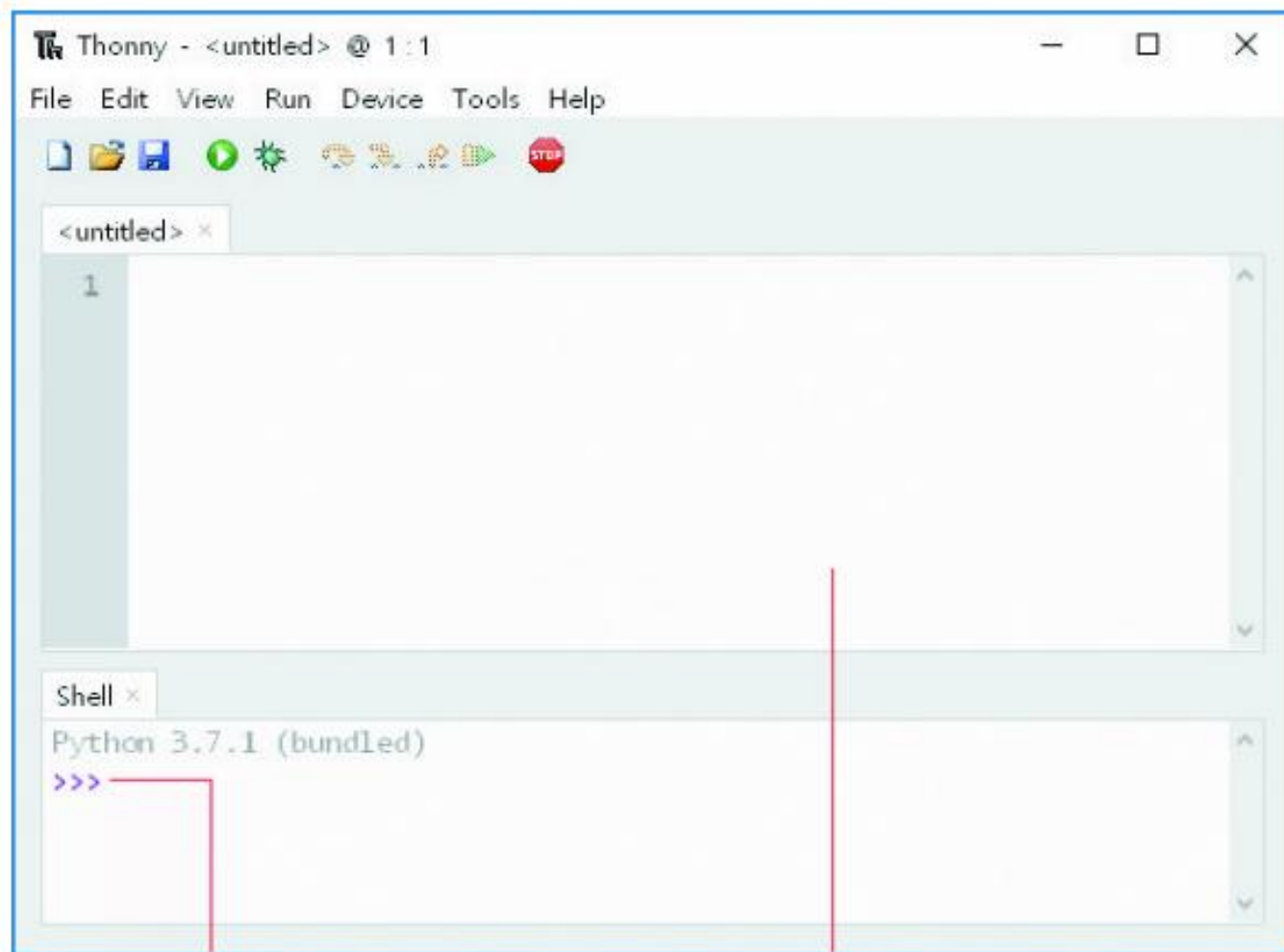


安裝 Thonny 開發環境

- 下載後請雙按執行該檔案，然後一直按 Next 即可完成安裝



開始寫第一行程式



互動性程式執行區

程式編輯區

```
Shell x  
Python 3.7.1 (bundled)  
>>> print("Hello World")
```

print("Hello World") 這個
程式是要求電腦在螢幕
印出 "Hello World"

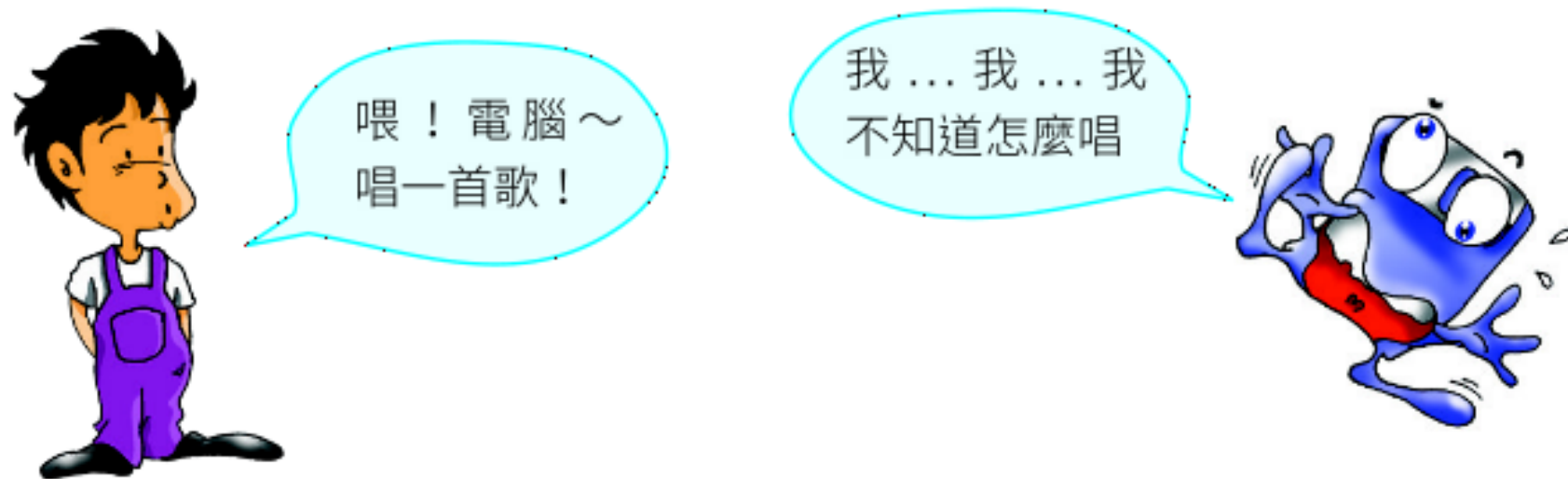
1 輸入 `print("Hello World")`,
然後按 **Enter** 鍵



```
Shell x  
Python 3.7.1 (bundled)  
>>> print("Hello World")  
  
Hello World  
  
>>>
```

2 電腦依照我們的程
式顯示 **Hello World**

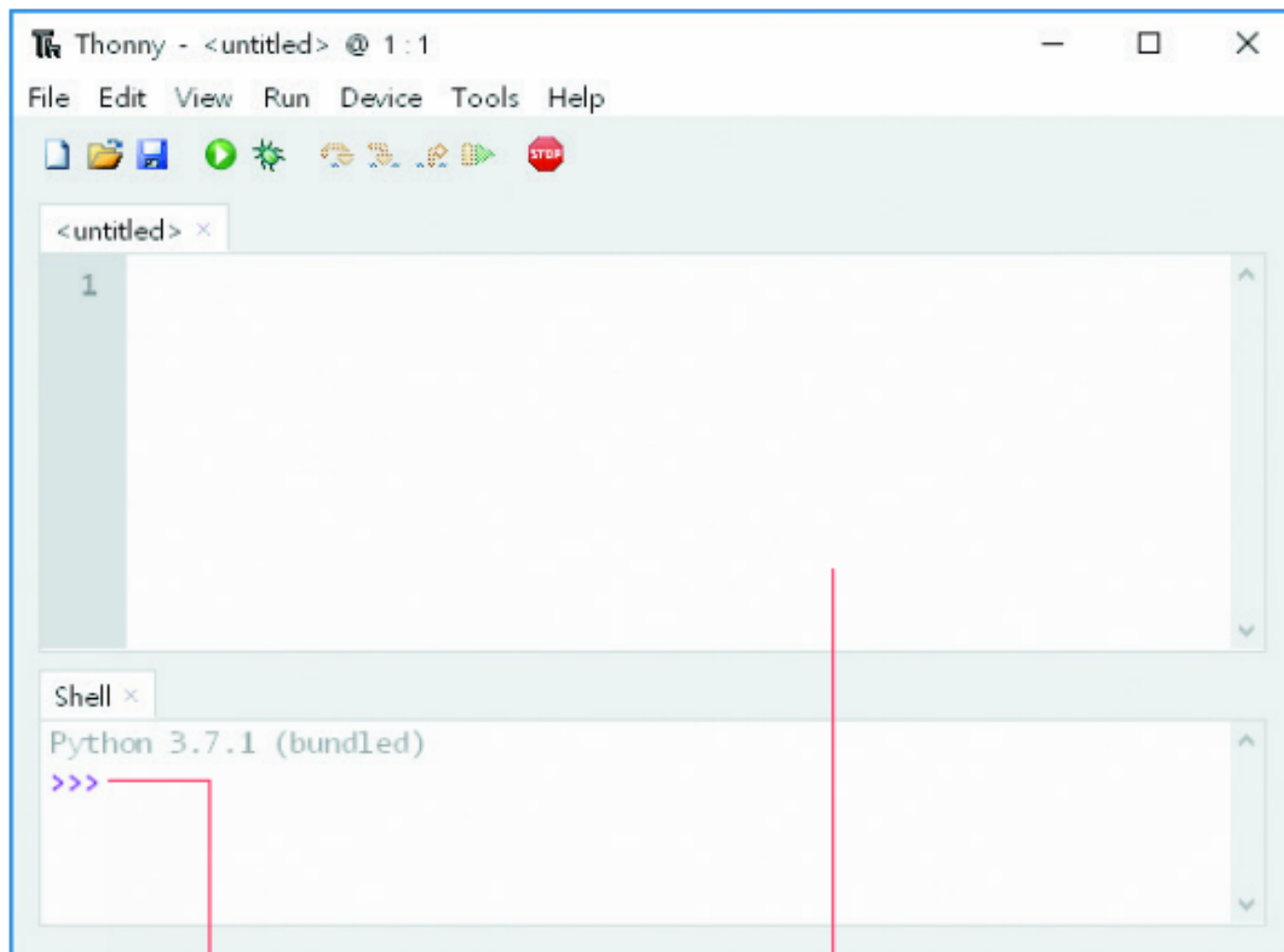
寫程式其實就像是寫劇本，寫劇本是用來要求演員如何表演，而寫程式則是用來控制電腦如何動作。



雖然說寫程式可以控制電腦，但是這個控制卻不像是人與人之間溝通那樣，只要簡單一個指令，對方就知道如何執行。您可以將電腦想像成一個動作超快，但是什麼都不懂的小朋友，當您想要電腦小朋友完成某件事情，例如唱一首歌，您需要告訴他這首歌每一個音是什麼、拍子多長才行。

所以寫程式的時候，我們需要將每一個步驟都寫下來，這樣電腦才能依照這個程式來完成您想要做的事情。

Thonny 開發環境基本操作



互動性程式執行區

程式編輯區

Thonny 開發環境基本操作

```
Thonny - <untitled> @ 8:13
File Edit View Run Device Tools Help
<untitled> * x
1 import time
2 from machine import Pin
3
4 led = Pin(15, Pin.OUT)
5
6 led.value(1)
7 time.sleep(3)
8 led.value(0)
Shell x
Python 3.7.1 (bundled)
>>> print("Hello World")
Hello World
>>>
```

在此區域撰寫程式

Thonny 開發環境基本操作

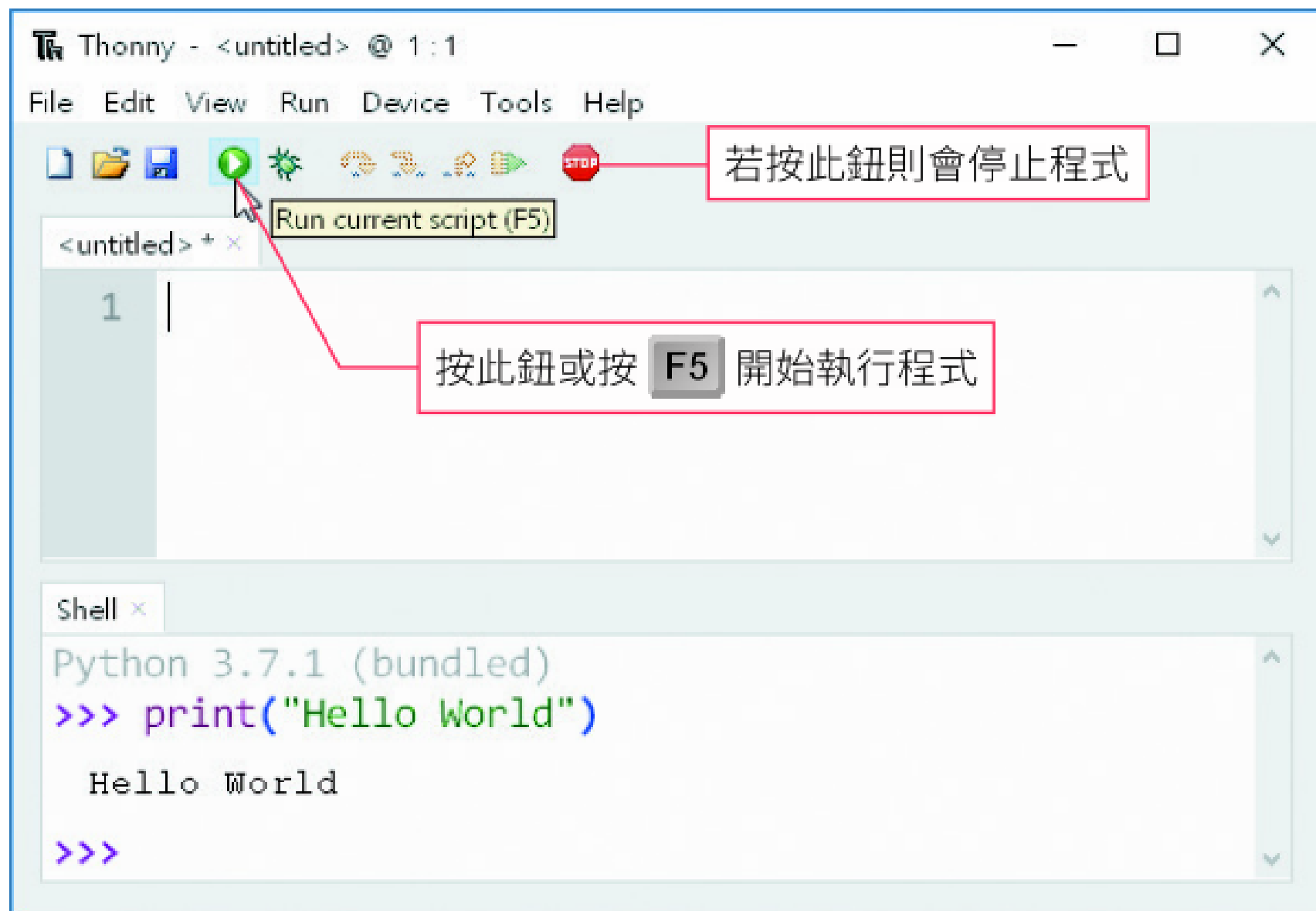
可以說，上半部程式編輯區類似稿紙，讓我們將想要電腦做的指令全部寫下來，寫完後交給電腦執行，一次做完所有指令。

而下半部 **Shell** 窗格則是一個交談的介面，我們寫下一行指令後，電腦就會立刻執行這個指令，類似老師下一個口令學生做一個動作一樣。

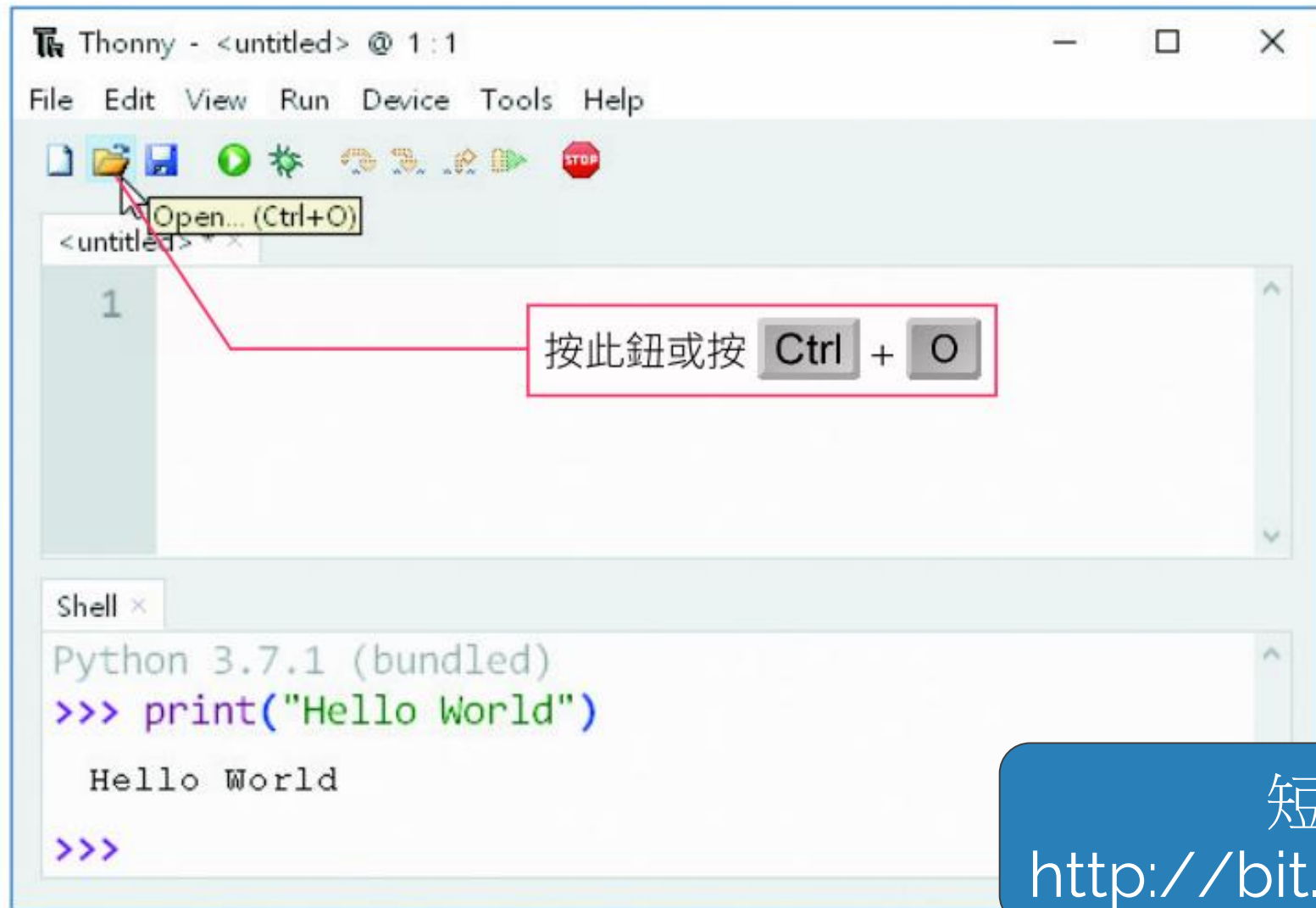
所以 **Shell** 窗格適合用來作為程式測試，我們只要輸入一句程式，就可以立刻看到電腦執行結果是否正確。

 本書後面章節若看到程式前面有 >>>，便表示是在 **Shell** 窗格內執行與測試。

如果要讓電腦執行或停止程式，請依照下面步驟：



若要打開之前儲存的程式或範例程式檔，請如下開啟：



⚠ 本套件範例程式下載網址：<http://www.flag.com.tw/maker/download/FM610A>。

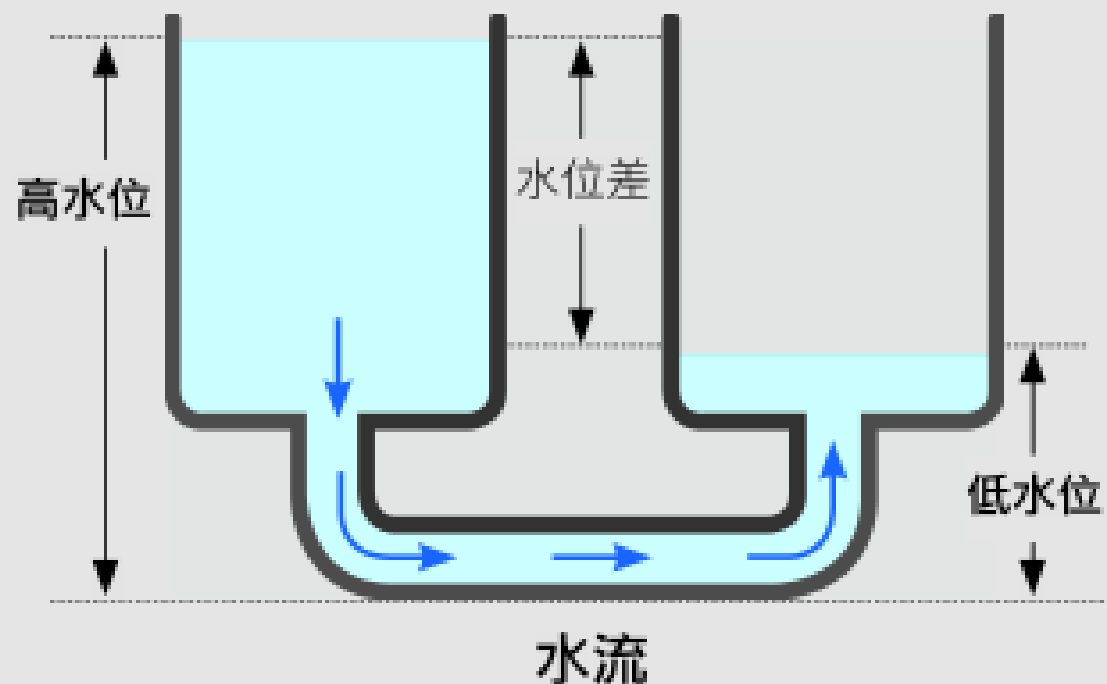
02 電子電路基礎

電壓、電流、電阻

電壓、電流

■ 電壓、電流

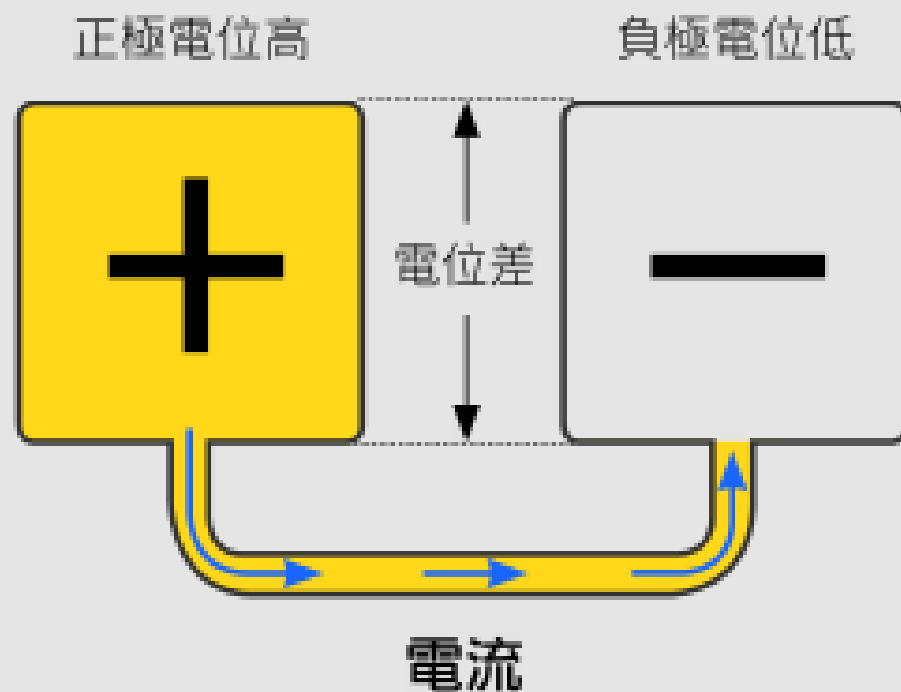
在現實世界中，水的流動稱為水流，水流的流向與大小會由水位的高低差來決定。請將同樣的觀念類推到在電子的世界，電子的流動稱為**電流**，電流的流向與大小會由電位的高低差來決定：



電壓、電流

電位的高低差則被稱為**電壓**或**電位差**，電壓的單位為伏特 (Volt, 簡稱為 V)，電流的單位是安培 (Ampere, 簡稱為 A)，電流的大小和電壓成正比。

一般我們會以大地的電位為 0，所以電子元件或裝置若標示輸出電壓 5V，表示其輸出電力的電位 - 大地電位 = 5V。而電子元件或裝置也常會以 GND 或 G (Ground 的簡稱) 來標示 0 電位點 (負極)。



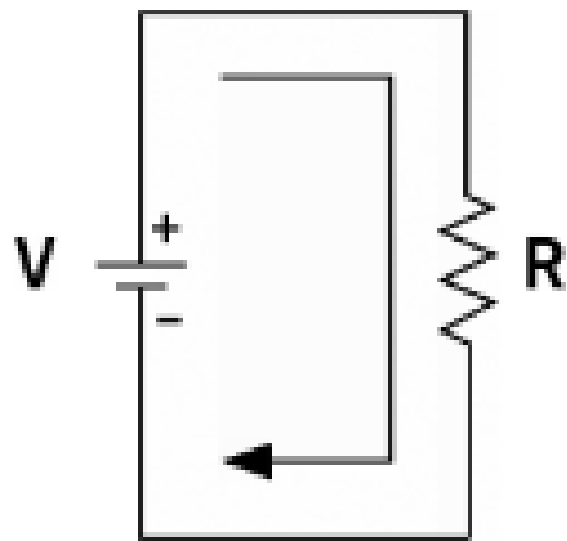
■ 電阻

電阻是物體對於電流通過的阻礙能力，在電壓固定的條件下，電阻值越高，代表阻礙能力越強，能夠通過這個物體的電流量就會越小，反之電阻值越小，可通過的電流愈大。

電阻除了是阻礙電流的能力值以外，也是一種電子元件的名字。當我們做實驗時，為了避免電流量過大而燒壞其中的零件，會額外加上名為電阻的元件，用來阻礙電流進而控制電流的大小。

電子迴路

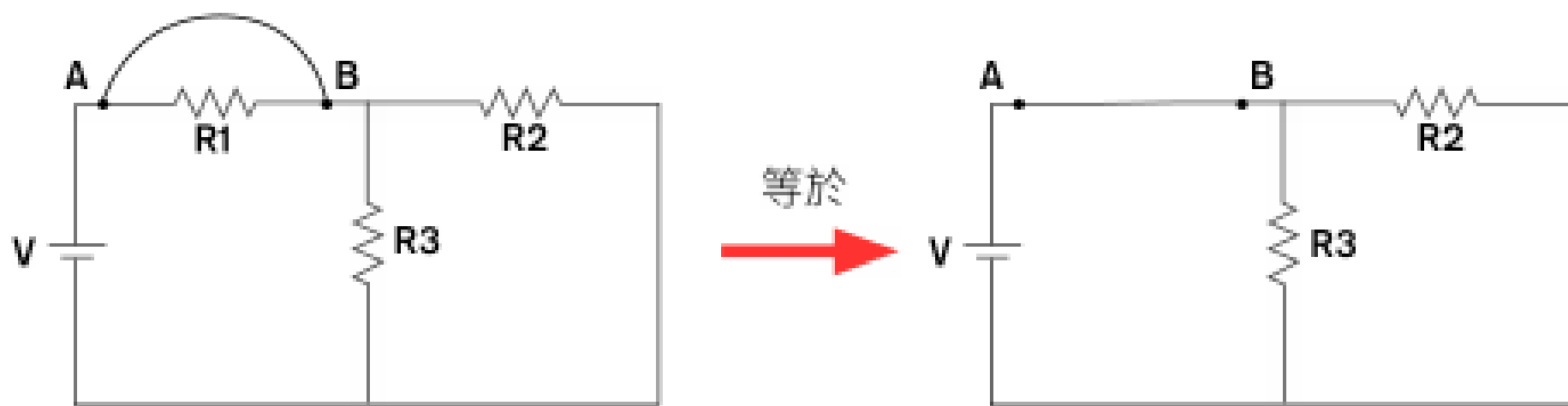
電子零件的連接必須構成迴路才能產生作用，所謂迴路指的是能夠讓電流通過的電路，最簡單的電子迴路如下：



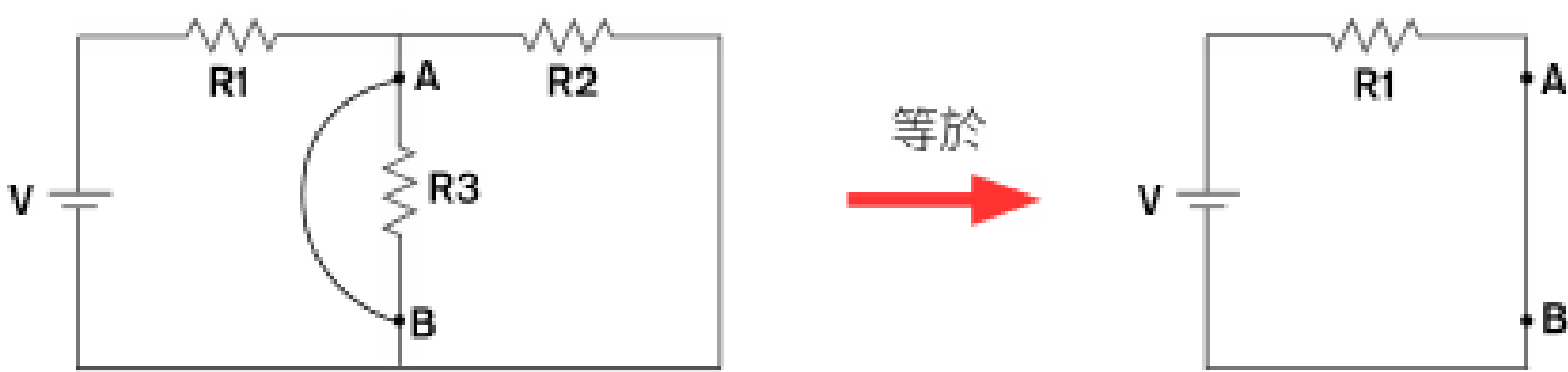
電子迴路上一定要有零件，否則會造成短路，請參見稍後說明。

短路

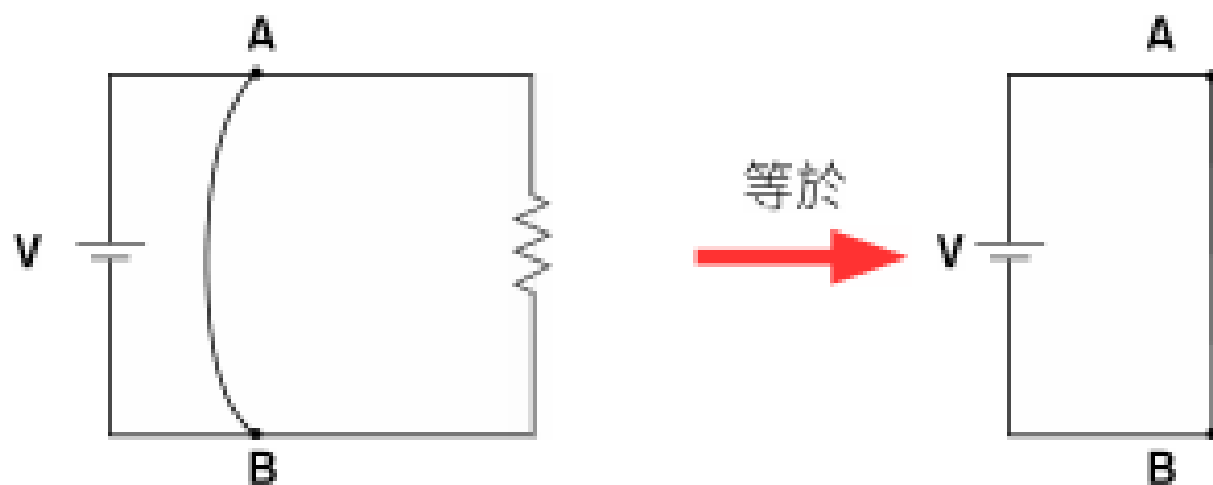
短路泛指用一導體（如：電線）接通迴路上的兩個點，因為導體的電阻幾乎為 0，絕大部分的電流會經由新接的電線流過，而不經過原來這兩個點之間的零件，如此將使得這些零件失去功能。



A、B 被短路，電流直接由 A 流到 B，R1 失去作用

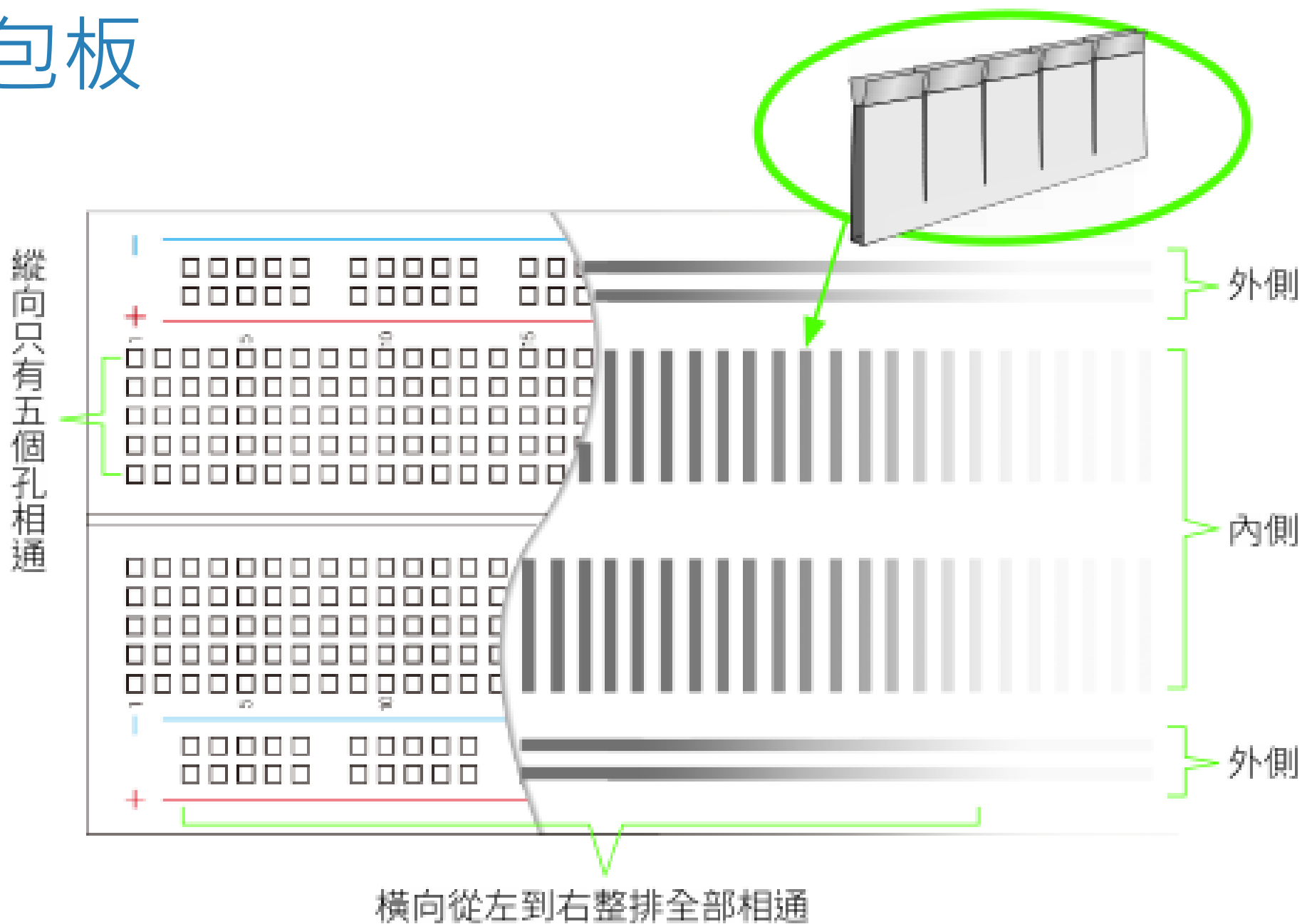


電流直接由 A 流到 B, R_3 、 R_2 都失去作用

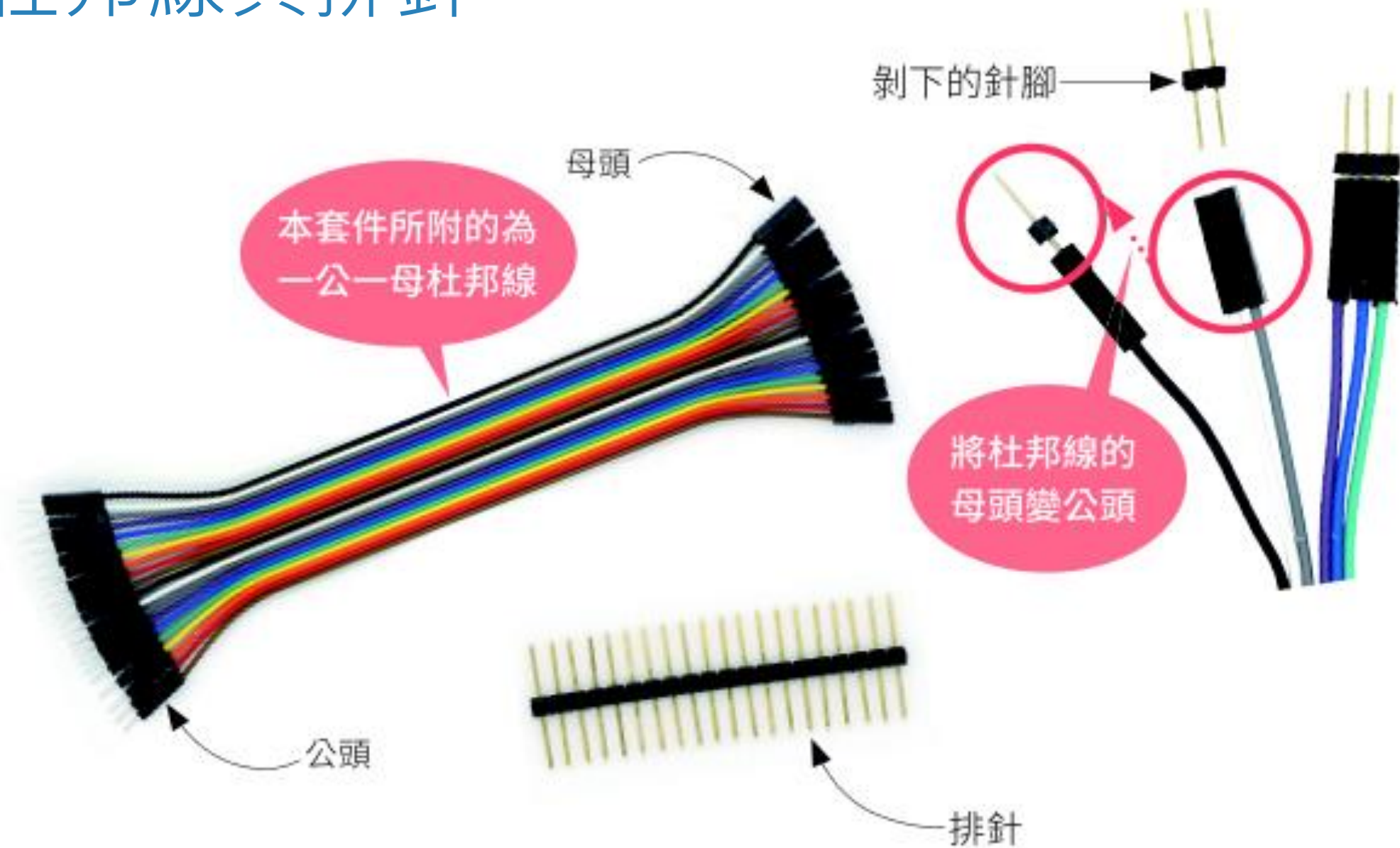


將電源短路電流直接由 A 流到 B, 因為導線電阻幾乎為 0, 電流變得很大, **電池將發熱燒毀**

麵包板



杜邦線與排針



D1 mini 控制板

D1 mini 本套件使用的主要控制板，這是一個單晶片開發板，您可以將它想成是一部小電腦，可以用來執行 Python 撰寫的程式。

後面章節的實驗中，我們會使用杜邦線，將電子元件連接到兩側的輸出入腳位，然後就可以藉由這些輸出入腳位控制外部的電子元件，或是從外部電子元件獲取資訊。



03 控制 LED 亮暗 - 數位輸出

Python 物件、資料型別、變數、匯入模組

Python 流程控制 (while 迴圈) 與區塊縮排

物件

- 英文一般寫句子時，會以名詞 + 動詞。Python 是以物件.方法來描述。

文章寫作	寫 Python 程式	
車子	<code>car</code>	car 物件
車子向前進	<code>car.start()</code>	car 物件的 start 方法

- 方法後面會加上括號()`()`，有些方法需要加入額外的參數。
 - ✓ 例如：`car.go(100)`，車子加速到 100。
- 若方法有多個參數，以逗點分隔。
 - ✓ 例如：`car.left(50, 30)`，以 50 的速度，向左轉 30 度。

練習：字串物件

- 在互動模式中，輸入下列敘述：(>>> 指的是在互動模式中，執行單行敘述)

```
>>> "abc".upper() —— 使用字串物件 "abc" 的 upper() 方法，  
                        將字串轉成大寫  
'ABC'  
  
>>> "abc".find('b') —— find() 方法尋找 'b' 的位置  
                        (從 0 開始)  
1  
  
>>> "abc".replace('b', 'z') —— replace() 方法將所有的  
                        'b' 取代成 'z'  
'azc'
```

- 不同的物件會有不同的方法。例如：字串物件與整數物件。

資料型別

- 除字串物件以雙引號或單引號來表示，寫程式常有整數與浮點數(小數)物件，例如：111 與 11.1。

```
>>> 111 + 111 —— 整數物件相加
```

```
222
```

```
>>> "111" + "111" —— 字串物件串聯
```

```
'111111'
```

- 上述 + 的運算，因物件的資料不同而產生不同的結果。物件的種類，程式語言稱之為『物件型態』或『資料型態』(Data Type)。

練習：要分清楚資料型別

- 兩個資料型別若不同，可能會導致程式錯誤。

```
>>> 111 + "111" —— 不同型別的資料相加發生錯誤
```

```
Traceback (most recent call last):
```

```
  File "<ipython-input-6-4832c22160be>", line 1, in  
<module>
```

```
    111 + "111"
```

```
TypeError: unsupported operand type(s) for +: 'int'  
and 'str'
```

型別轉換

- 兩個資料型別若要運算，可以使用型別轉換。

```
>>> str(111) + "111"  —— str() 可轉換物件為字串型別
'111111'
>>> 111 + int("111") —— int() 可轉換物件為整數型別
222
```

整數與浮點數

- 整數與浮點數的數學運算

- ✓ + (加)、- (減)、* (乘)、/ (取商)、// (取商，整數)、% (取餘數)、** (指數)。
- ✓ Python 允許整數與浮點數直接運算，執行下列程式：

加法 (+)

```
>>> 10 + 5.5  
15.5
```

減法 (-)

```
>>> 10 - 5.5  
4.5
```

乘法 (*)

```
>>> 10 * 5.5  
55.0
```

除法取商 (/)

```
>>> 10 / 5.5  
1.8181818181818181
```

除法取整數商 (//)

```
>>> 10 // 5.5  
1.0
```


取餘數 (%)

```
>>> 10 % 5.5  
4.5
```

指數 (**)

```
>>> 4 ** 0.5, 8 ** (1/3)  
(2.0, 2.0)
```

TIP

1. 整數與浮點數做運算，結果一定為浮點數。
2. 只有整數與整數做除法，結果為浮點數。

常用的變數運算

- 把整數加上特定的值：

```
>>> x = 1
>>> x = x + 1
>>> x
2
```

- 常用的簡式：

簡式	意義
$x += 2$	$x = x + 2$
$x -= 2$	$x = x - 2$
$x *= 2$	$x = x * 2$

簡式	意義
$x /= 2$	$x = x / 2$
$x //= 2$	$x = x // 2$
$x %= 2$	$x = x \% 2$

變數

- 『變數』(variable) 就像是掛在物件的名牌，幫物件取名之後，讓我們方便識別物件與操作，其語法為：

變數名稱 = 物件

- 例如：

```
>>> n1 = 123456789 —— 將整數物件 123456789 指派給變數 n1
>>> n2 = 987654321 —— 將整數物件 987654321 指派給變數 n2
>>> n1 + n2 —— 實際上是 123456789 + 987654321
1111111110
```

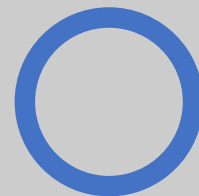
寫出可讀性高的程式 (1/2)

- 使用有意義的變數 (variable) 的名稱。

```
a = 3.1  
b = 2.2  
c = a * b * b
```



```
pi = 3.1  
radius = 2.2  
# 使用公式計算圓面積  
circle_area = pi * radius * radius
```



寫出可讀性高的程式 (2/2)

- 將程式加上註解。
 - ✓ 註解可以幫助其他人了解這個程式。
- 較佳的註解：

註解 1：使用公式計算圓面積

- 不好閱讀的註解：

註解 2：將圓周率乘半徑乘半徑

內建函式

- 『**函式**』 (function) 是一段預先寫好的程式，方便重複使用。而程式先將經常使用到的功能以函式的形式先寫好，稱為『**內建函式**』。
- 例如：print() 是最常用的顯示函數：

```
>>> print("abc") —— 顯示字串物件
```

```
abc
```

```
>>> print("abc".upper()) —— 顯示字串物件.方法的執行結果
```

```
ABC
```

```
>>> print(111 + 111) —— 顯示整數物件運算的結果
```

```
222
```

匯入模組

- 內建函式不就越多越好？若內建函式無限制增加，會導致啟動速度越來越慢，執行時佔用的記憶體越來越多。
- 『**模組**』(module)，就是將同一類的函式打包成模組，預設不會啟用。需要時，再用**匯入**(import)的方式啟用。預先寫好的稱為『**內建模組**』。

```
>>> import time —— 匯入時間相關的 time 模組
```

```
>>> time.sleep(3) —— 執行 time 模組的 sleep() 函式，暫停 3 秒
```

```
>>> from time import sleep —— 從 time 模組裡匯入 sleep() 函式
```

```
>>> sleep(5) —— 執行 sleep() 函式，暫停 5 秒
```


練習：匯入模組

程式

暫停 3 秒後，印出 Hello World! 字串物件

```
# 暫停 3 秒後，印出 Hello World!  
from time import sleep  
sleep(3)  
print("Hello World!")
```

指的是在程式編輯窗格中編輯與執行。

觀念整理

各種程式語言的語法邏輯都差不多，就像人類語言的文法。

1. 程式的每一個東西都是**物件**，有些物件有其特定的操作方法。
2. 基本物件有**資料型別**，型別不同，結果不同。甚至有時會錯誤。
3. **變數**是物件的名牌而已，也方便程式設計師操作。
4. 使用有意義的變數名稱與註解。
5. **內建函式**是經常用的函式，預先寫好的。
6. 適當的**匯入模組**，能精簡效能。

安裝與設定 D1 mini

1 連線 http://www.wch.cn/downloads/CH341SER_EXE.html

2 按此鈕下載

WCH 沁恒

产品中心 应用方案 沁恒社区 服务支持 关于沁恒

全部

CH341SER.EXE

适用范围	版本	上传时间	资料大小	
CH340G · CH340T · CH340C · CH340E · CH340B · CH341A · CH341T · CH341B · CH341C · CH341U	3.4	2016-09-28	237KB	下载

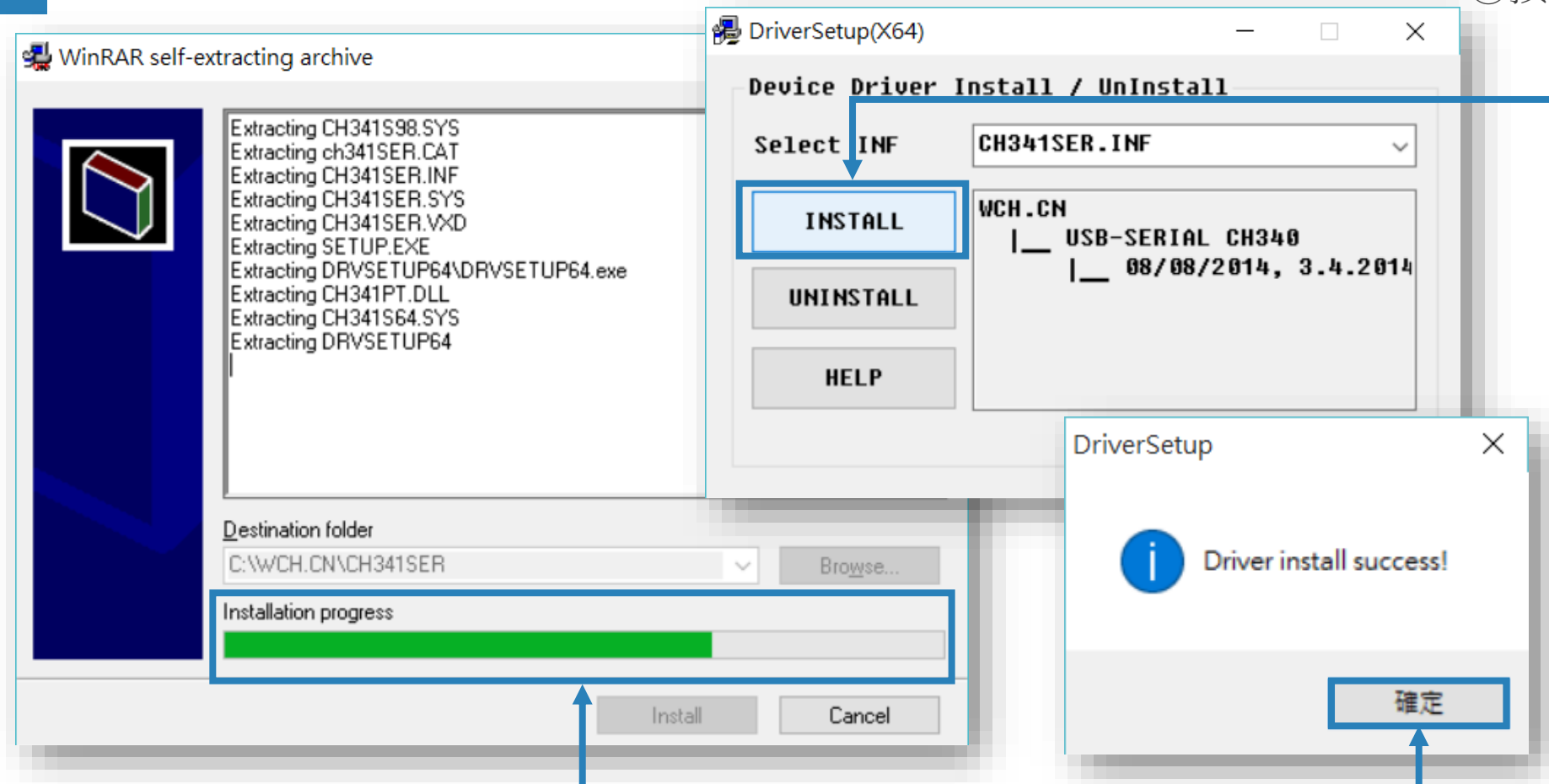
CH340/CH341USB转串口WINDOWS驱动程序。支持32/64位 Windows 10/8.1/8/7/VISTA/XP · SERVER 2016/2012/2008/2003 · 2000/ME/98 · 通过微软数字签名认证。支持USB转3线和9线串口等，用于随产品发行到最终用户。

产品手册

开发资源

驅動程式安裝步驟

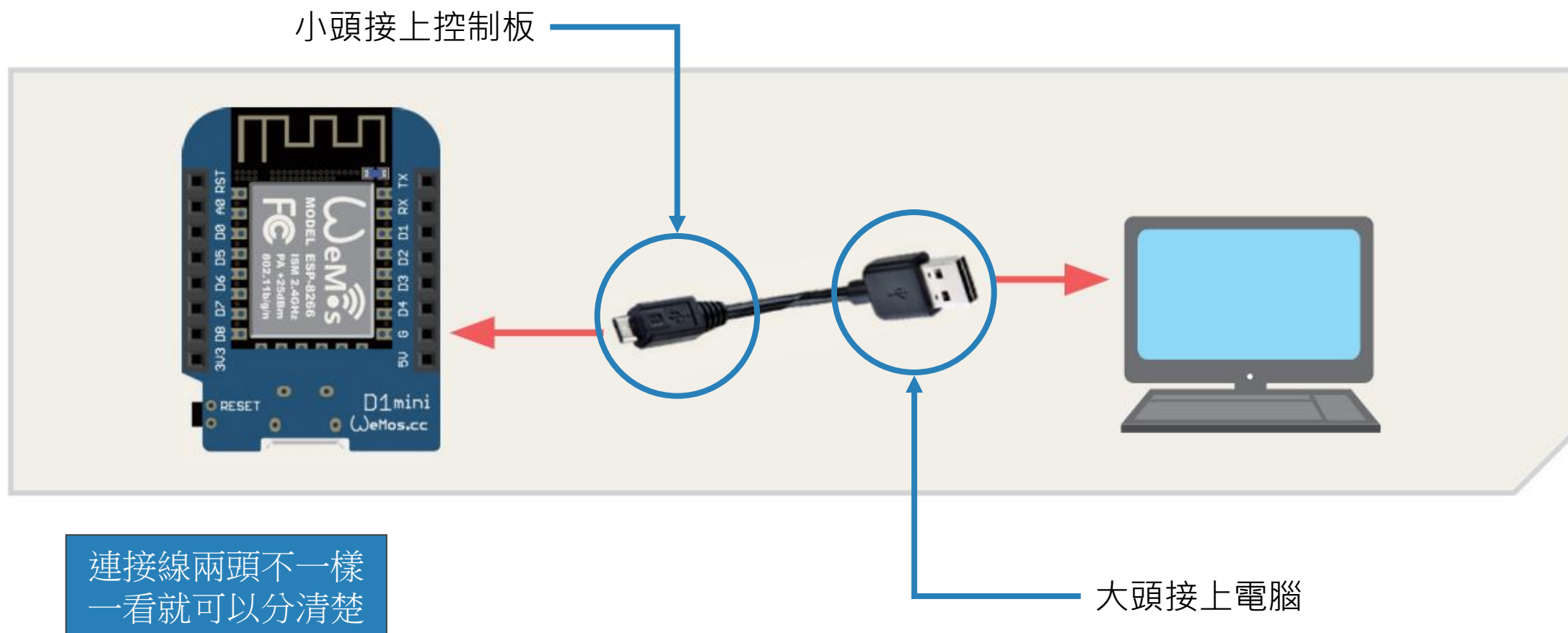
②按『INSTALL』開始安裝



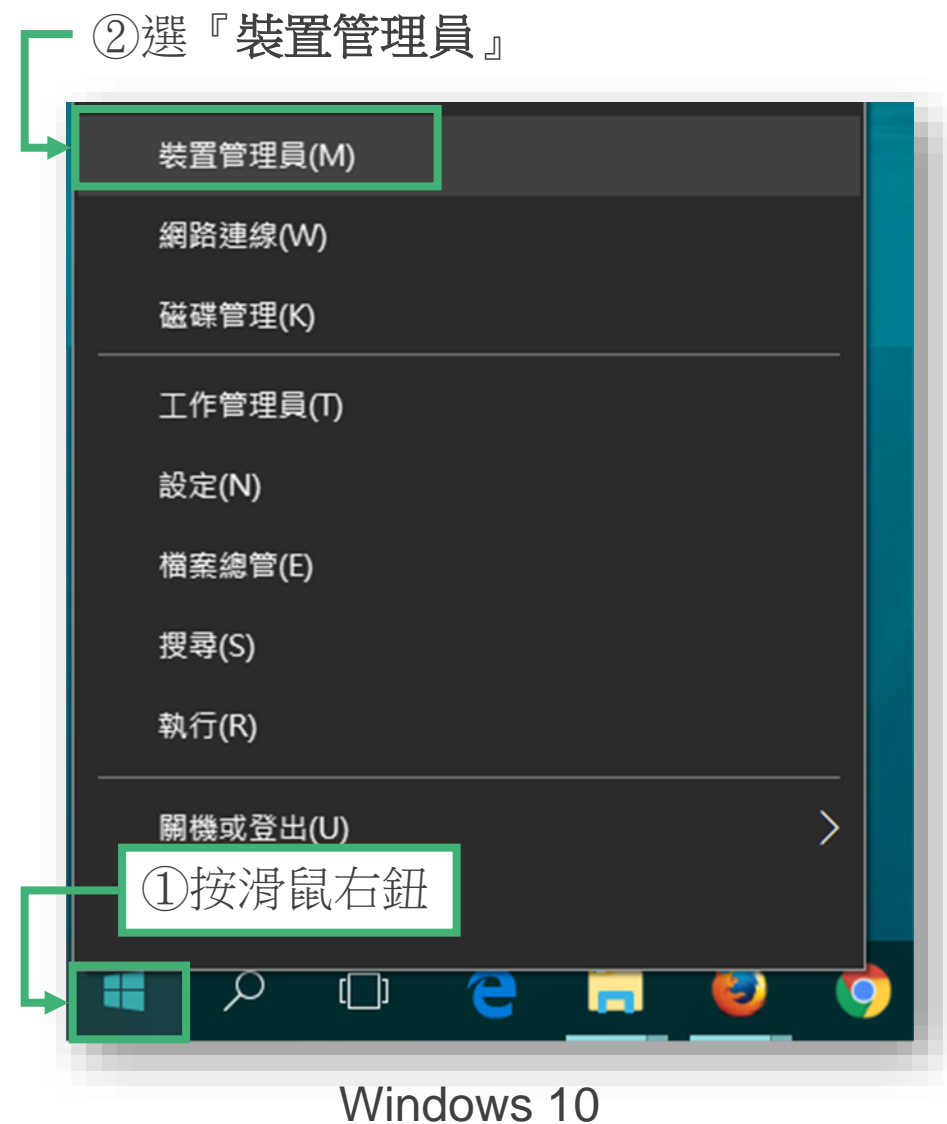
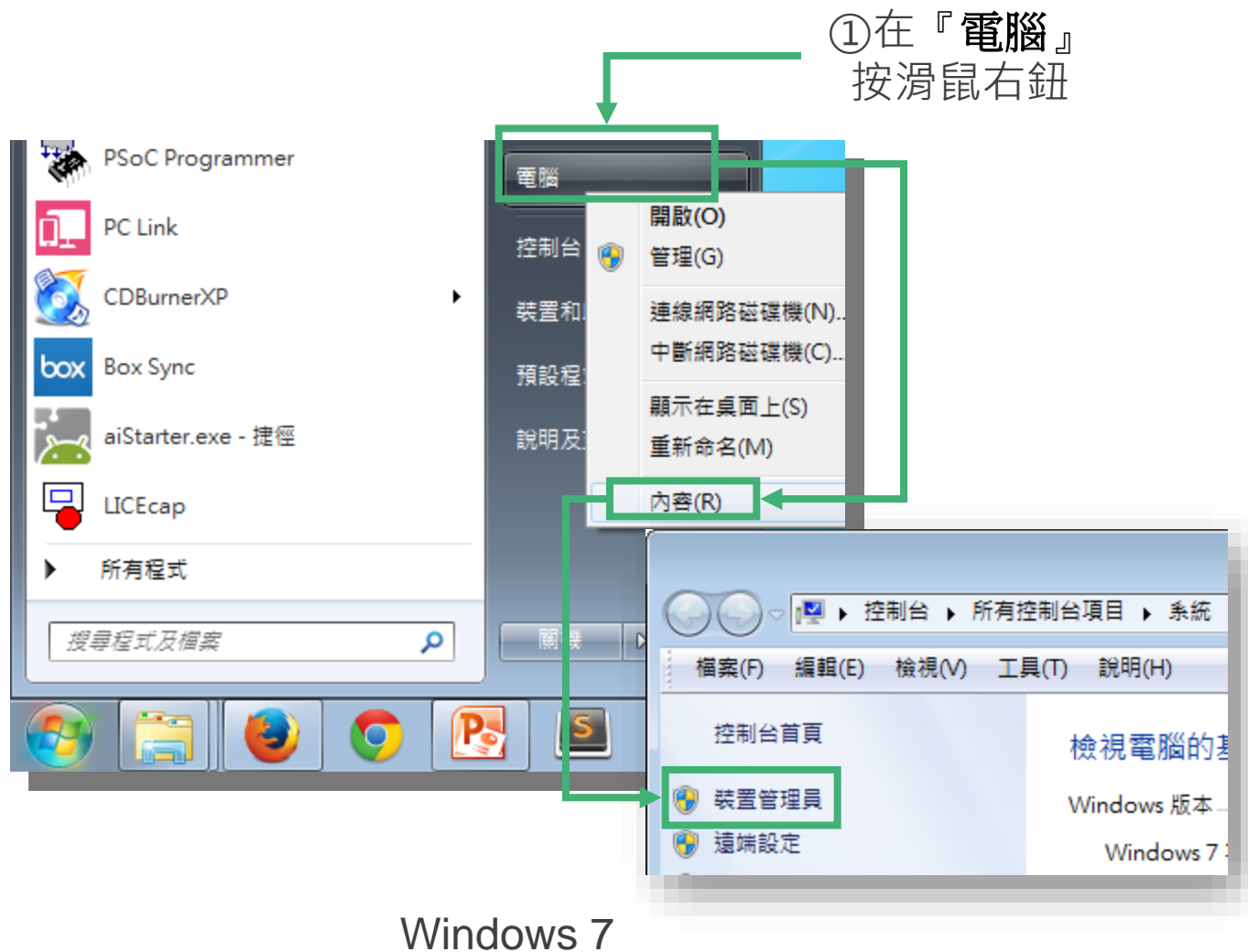
①會先解開驅動程式安裝檔案

③安裝完成

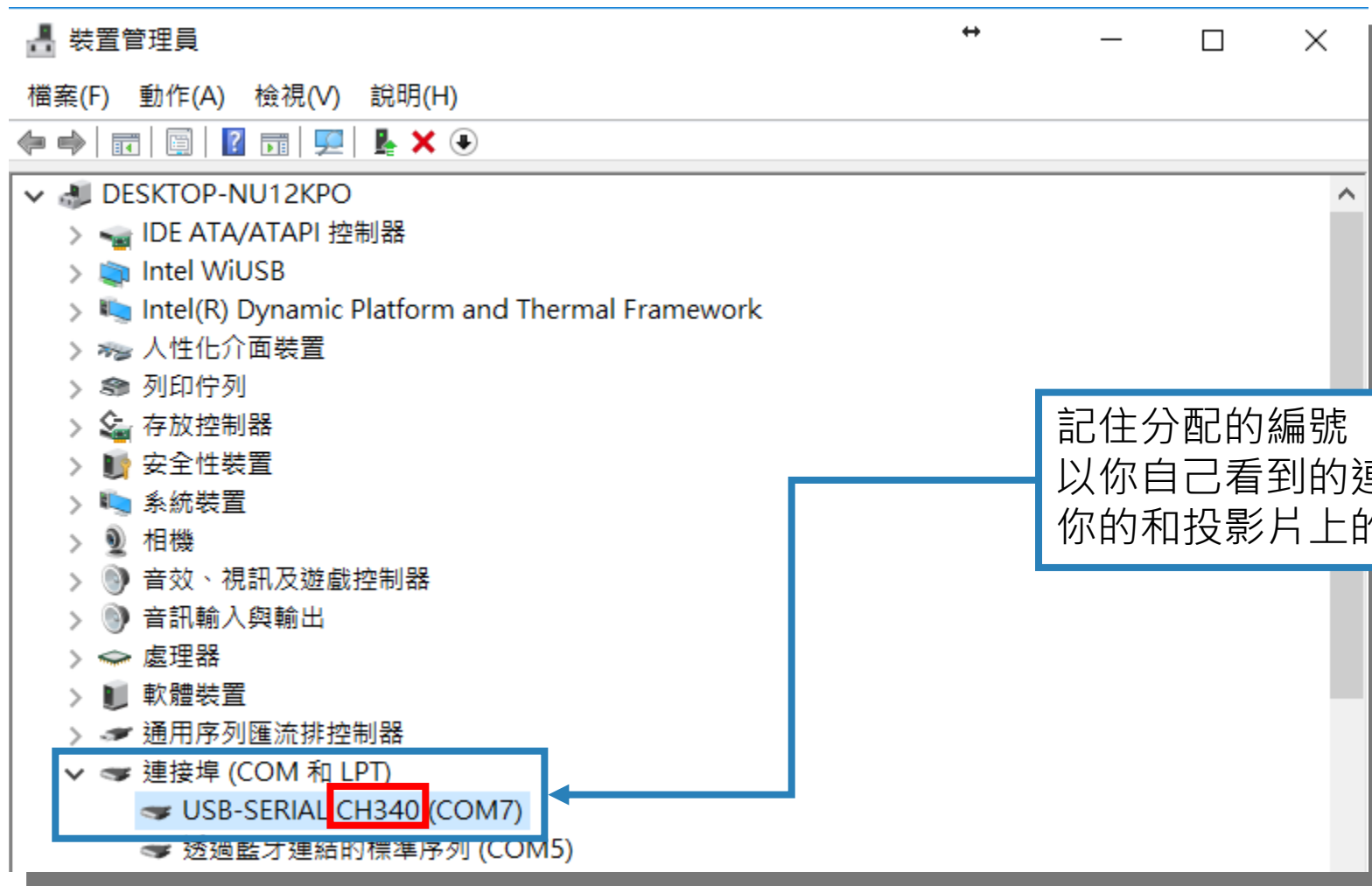
連接 PC 與 D1 mini 控制板



查看連接埠編號

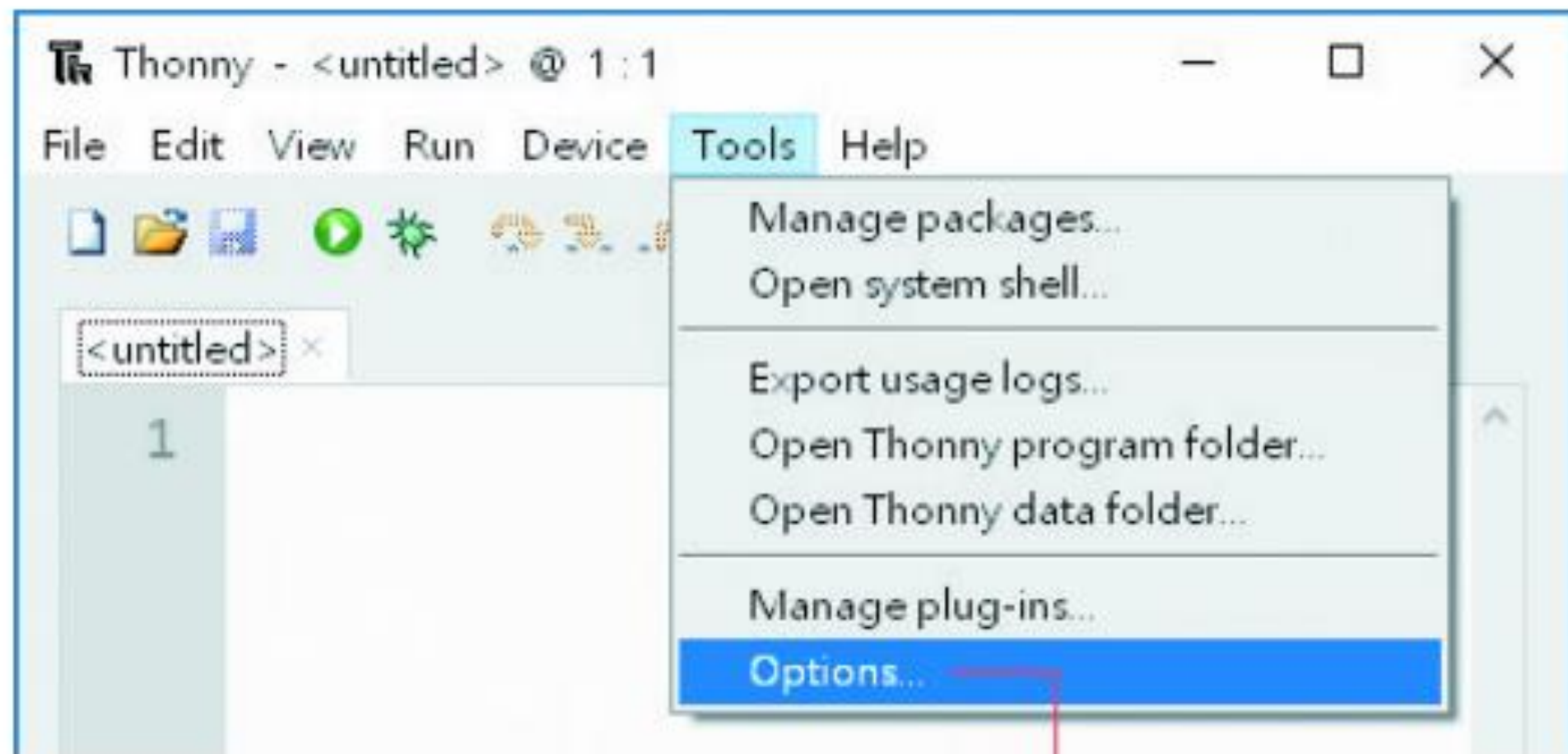


查看連接埠編號



記住分配的編號 (此例為 COM7)
以你自己看到的連接埠編號為準
你的和投影片上的很可能不一樣

找到 D1 mini 使用的序列埠後，請如下設定 Thonny 連線 D1 mini：

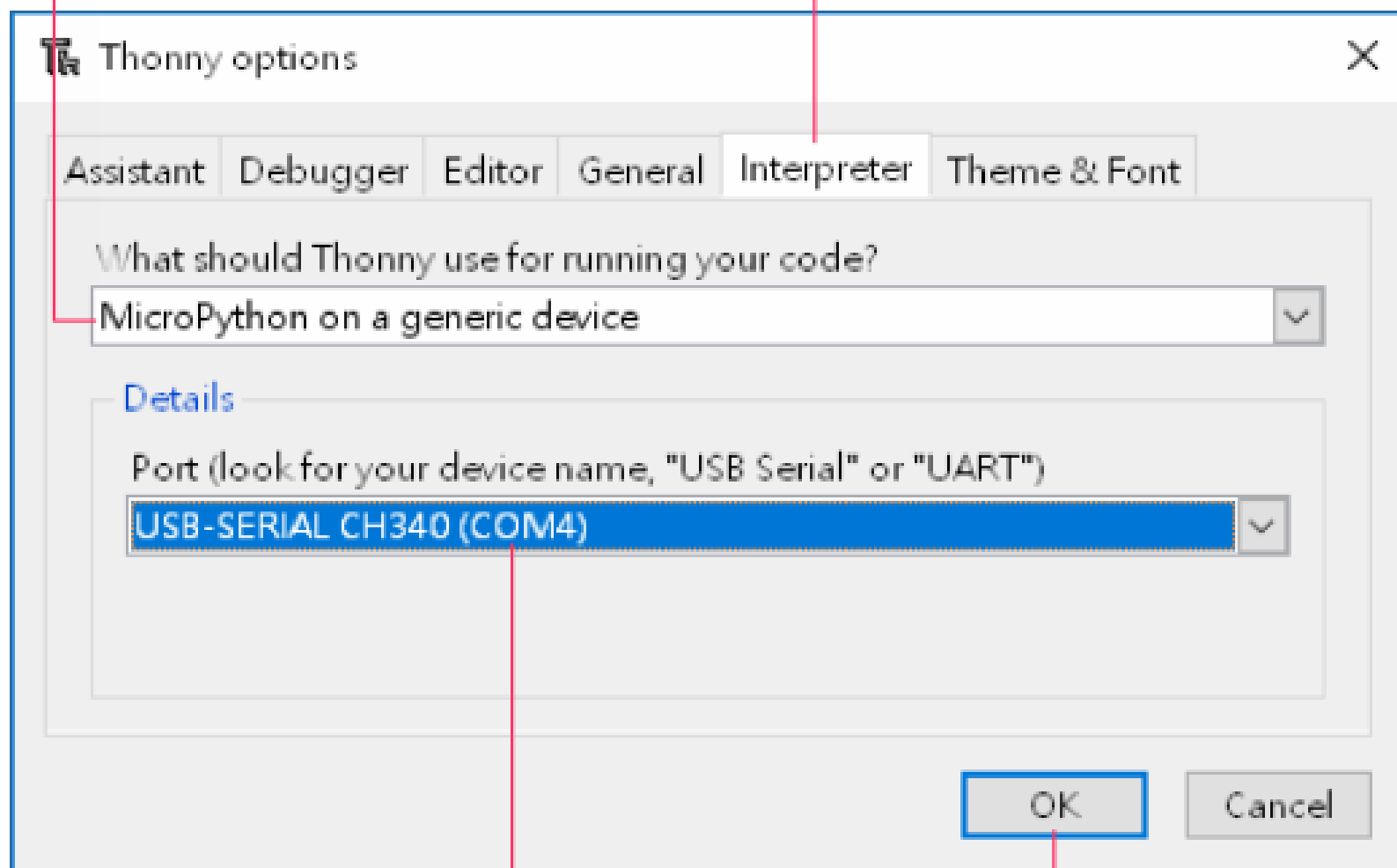


1 執行選單的『Tools/Options』命令，開啟設定視窗

3 拉下選單選擇

MicroPython on
generic device

2 切換到 Interpreter 頁面



4 拉下選單選擇剛剛
記下的序列埠編號

5 按 **Ok** 鈕儲存設定

Thonny - <untitled> @ 1:1

File Edit View Run Device Tools Help

<untitled> x

1

Shell x

```
MicroPython v1.9.4-8-ga9a3caad0 on 2018-05-11
; ESP module with ESP8266
Type "help()" for more information. [backend=
GenericMicroPython]
>>> |
```

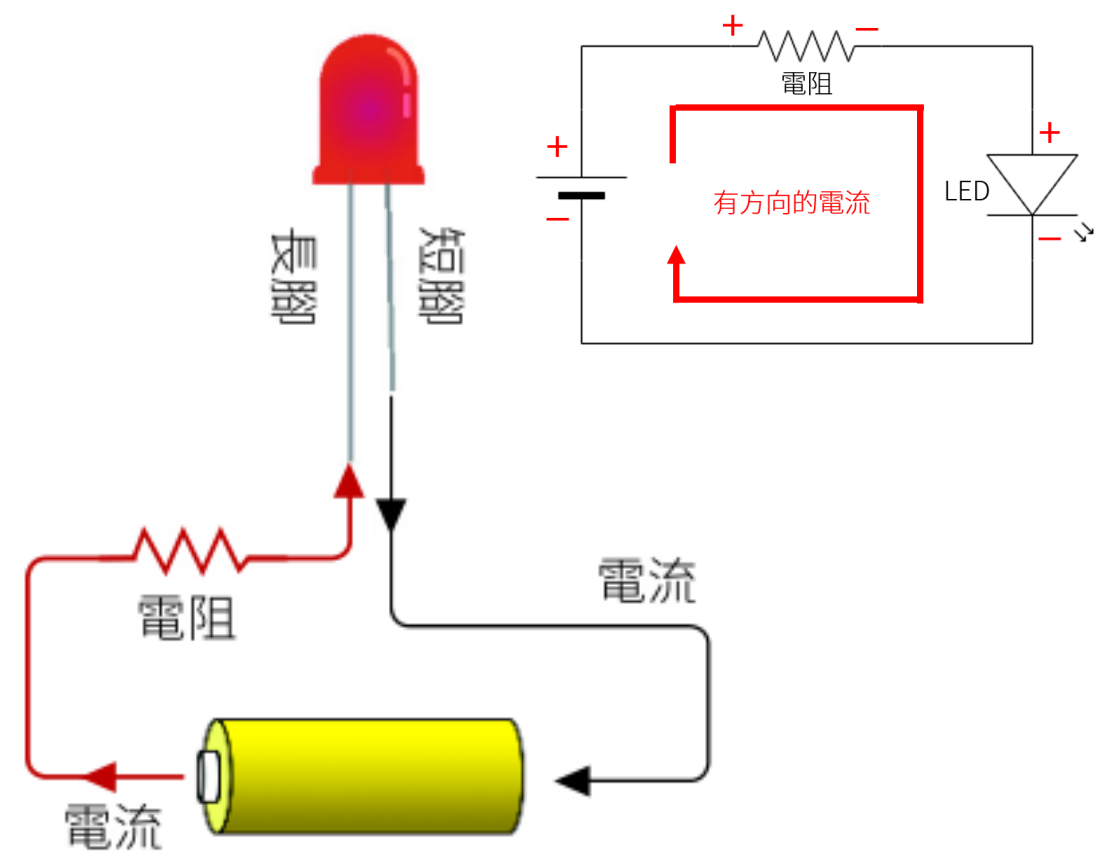
在 **Shell** 窗格看到 MicroPython 字樣便表示連線成功，若看不到請參見第 21 頁重新燒錄

! MicroPython 是特別設計的精簡版 Python，以便在 D1 mini 這樣記憶體較少的小電腦上面執行。

LED，中文為發光二極體，具有一長一短兩隻接腳，若要讓 LED 發光，則需對長腳接上高電位，短腳接低電位，像是水往低處流一樣產生高低電位差讓電流流過 LED 即可發光。LED 只能往一個方向導通，若接反則稱逆向偏壓，LED 不會發光。

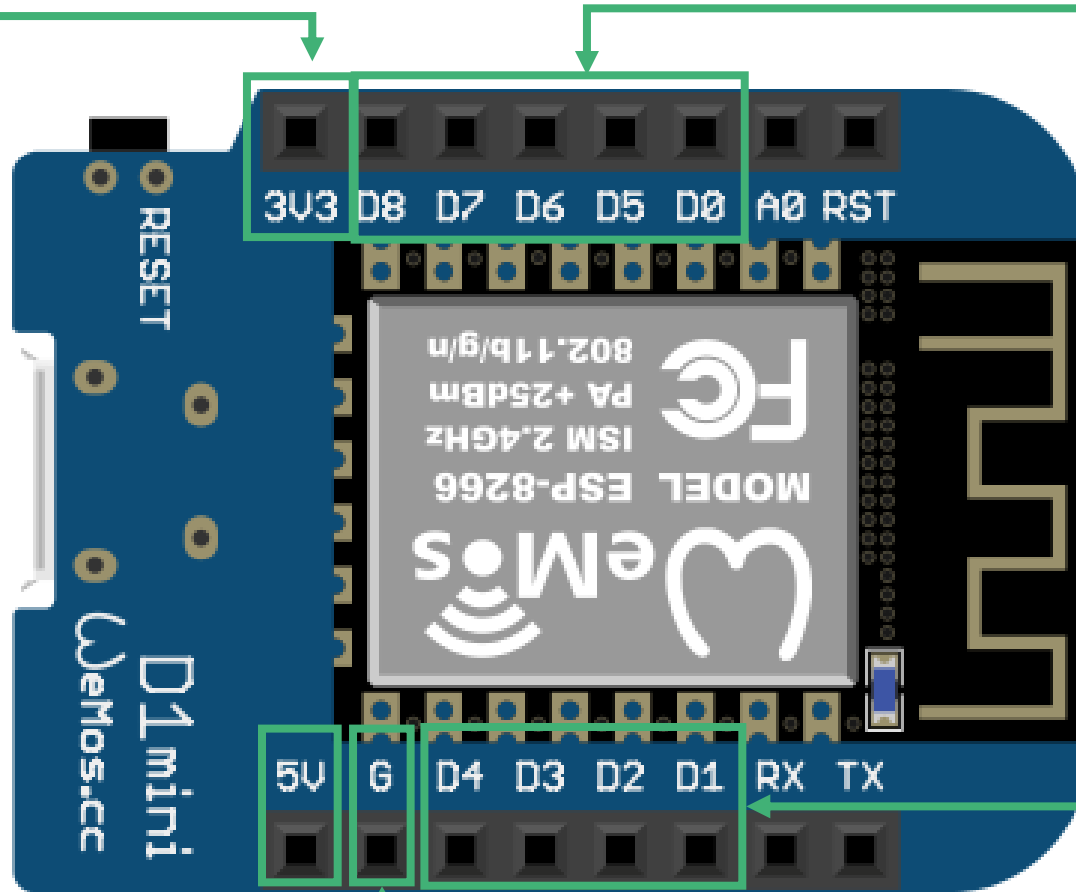
如圖所示，電池的正極代表高電位，負極代表低電位，如此接上 LED 後，電流流過 LED 即可發亮。稍後的實驗中，我們將用 D1 mini 來模擬電池的作用，讓 LED 短腳接 D1 mini 的 G 腳位 (相當於電池的負極)，另外用輸出腳位來給 LED 長腳高電位或低電位，若給高電位則導通發光，若給低電位則不會發光。

⚠ 為了避免 LED 被過大的電流燒毀，一般會加上第二章介紹過的電阻來控制電流的大小。



D1 mini 開發板上的電位電流控制

永遠在 (3.3 V)
高電位的插槽



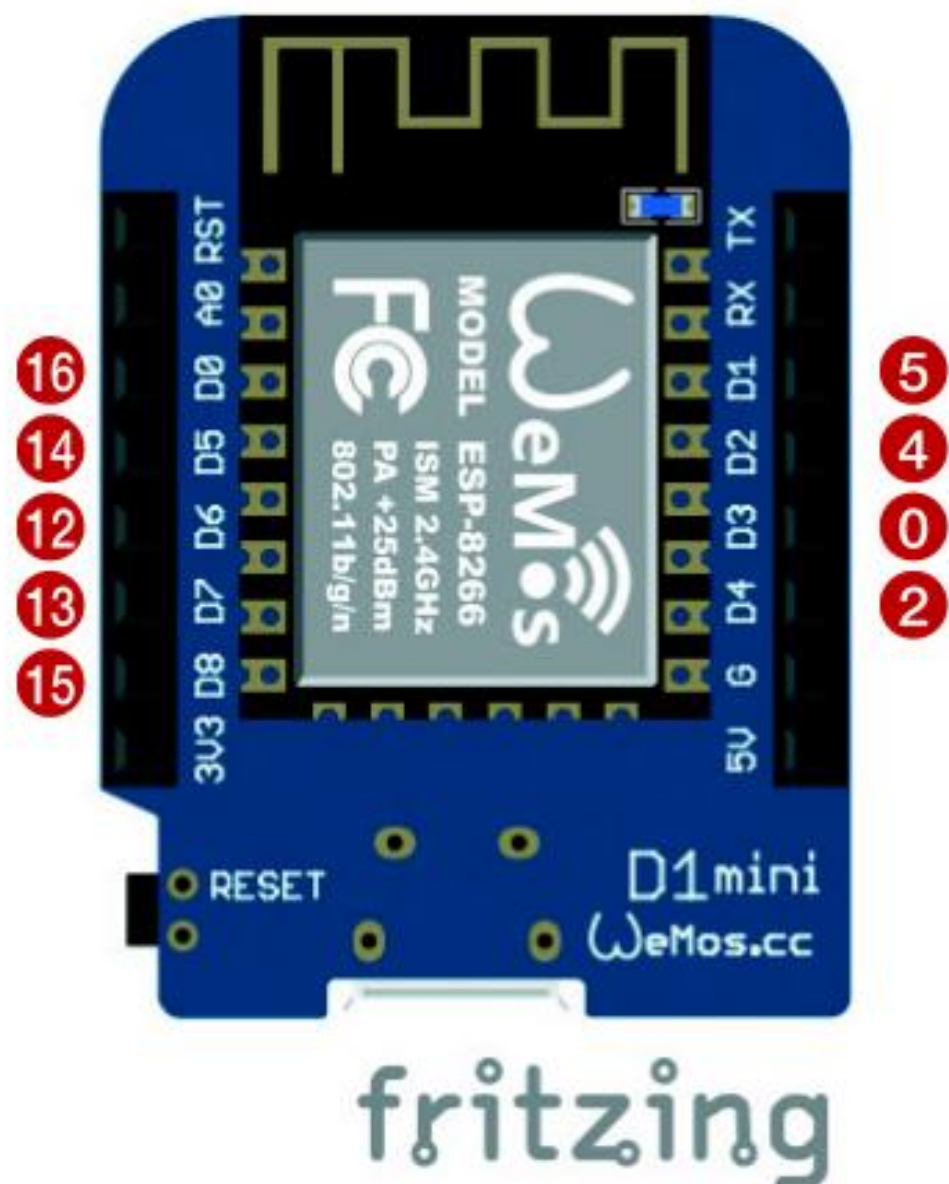
可升高、降低電位的插槽 (D0~D8)
稱為『數位 (digital) 輸出腳位』

永遠在 (5 V)
高電位的插槽

低電位的地面 (GrouND)

在程式中我們會以 1 代表有電，0 代表沒有電，所以等一下寫程式時，若設定腳位的值是 1，便表示要讓腳位有電，若設定值為 0 則表示沒有電。

D1 mini 兩側數位 IO 腳位外部的標示是 D0~D8，但是實際上在 D1 mini 晶片內部，這些腳位的真正編號並不是 0~8，其腳位編號請參見右圖：



Lab01

點亮/熄滅 LED

實驗目的

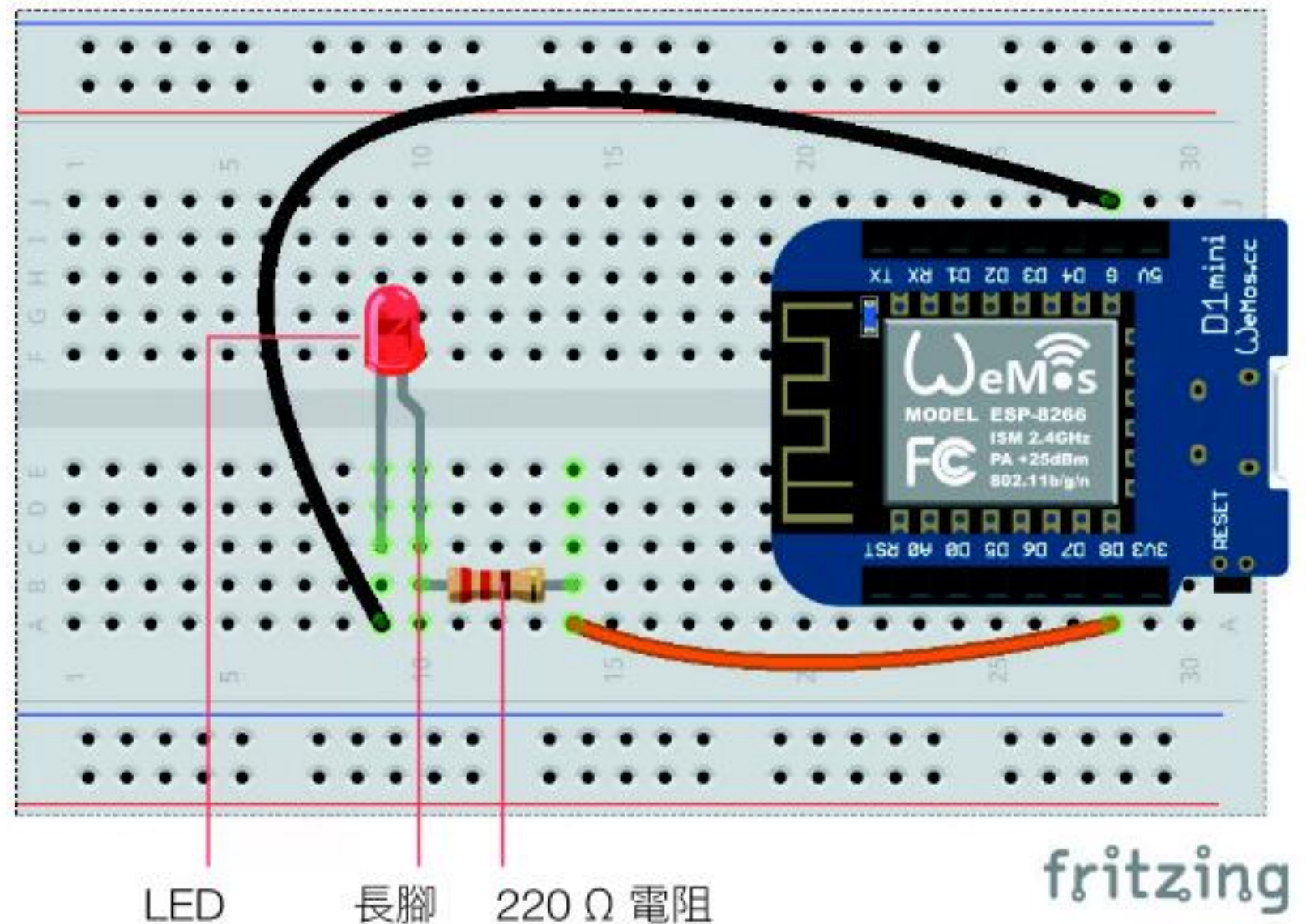
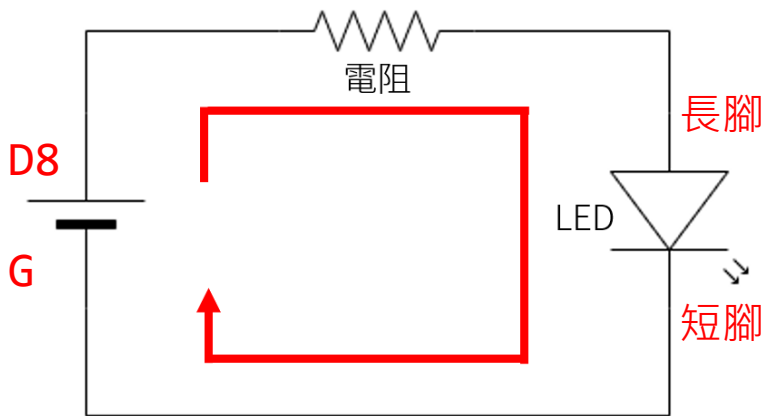
用 Python 程式控制 D1 mini 腳位, 藉此點亮或熄滅該腳位連接的 LED 燈。

材料

- D1 mini
- 紅色 LED
- 220 Ω 電阻

請依線路圖接線

- 電阻沒有方向性
- 線的顏色不必和圖一樣
- 要注意的是 LED 有分長短腳，以及腳位不要接錯



■ 設計原理

為了在 Python 程式中控制 D1 mini 的腳位，我們必須先從 machine 模組匯入 Pin 物件：

```
>>> from machine import Pin
```

前面我們接線時，連接 LED 高腳位的是 D8 腳位，這個腳位在晶片內部的編號是 15 號，所以我們可以如下建立 15 號腳位的 Pin 物件：

```
>>> led = Pin(15, Pin.OUT)
```

上面我們建立了 15 號腳位的 Pin 物件，並且將其命名為 led，因為建立物件時第 2 個參數使用了 "**Pin.OUT**"，所以 15 號腳位就會被設定為輸出腳位。

然後即可使用 value() 方法來指定腳位是否要輸出電：

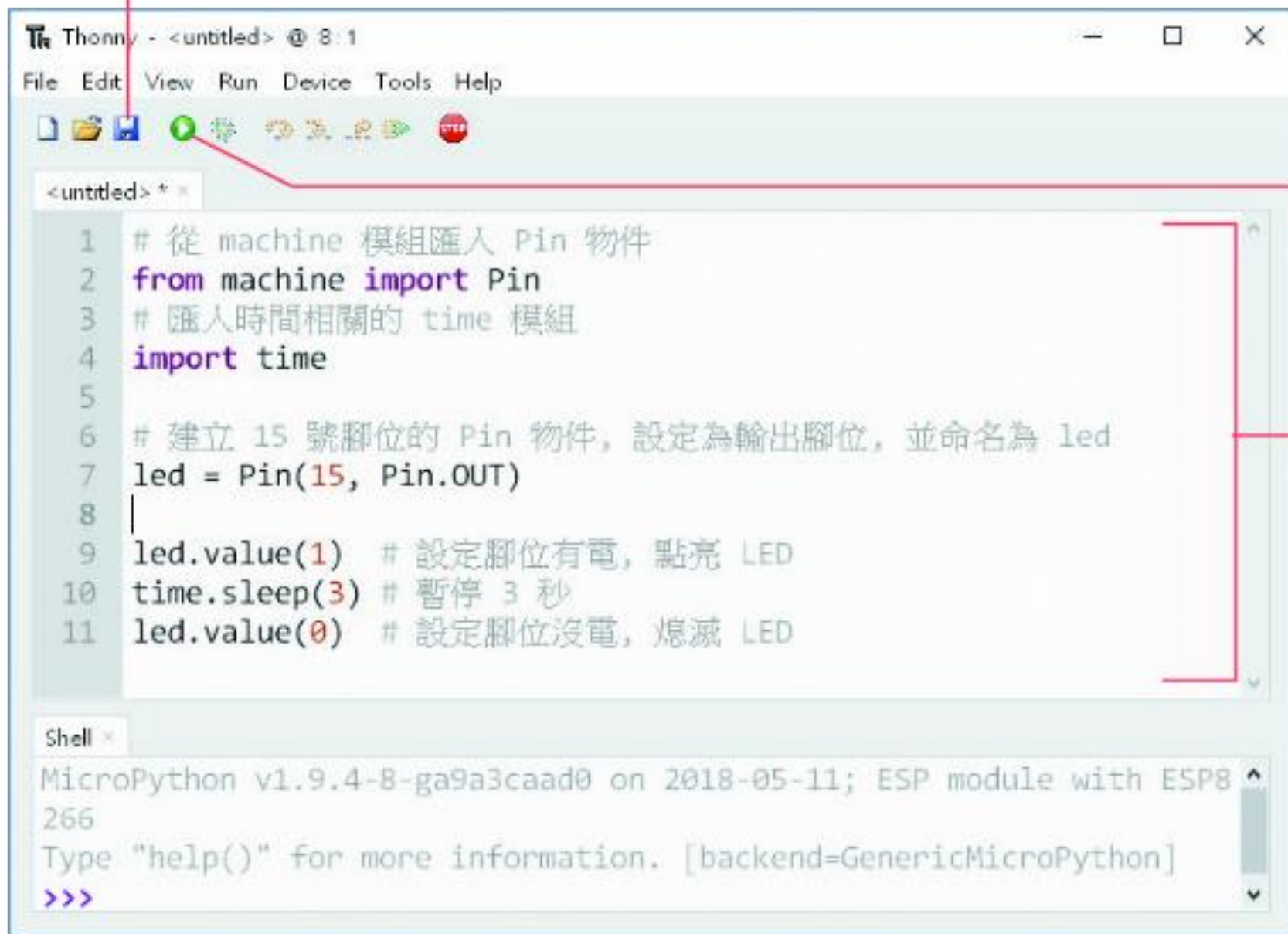
```
>>> led.value(1) ← 有電  
>>> led.value(0) ← 沒電
```


■ 程式流程圖



2 按此鈕或按 **Ctrl** + **S** 儲存檔案

Lab01.py



```
Thonny - <untitled> @ 8:1
File Edit View Run Device Tools Help
<untitled> *
1 # 從 machine 模組匯入 Pin 物件
2 from machine import Pin
3 # 匯入時間相關的 time 模組
4 import time
5
6 # 建立 15 號腳位的 Pin 物件，設定為輸出腳位，並命名為 led
7 led = Pin(15, Pin.OUT)
8 |
9 led.value(1) # 設定腳位有電，點亮 LED
10 time.sleep(3) # 暫停 3 秒
11 led.value(0) # 設定腳位沒電，熄滅 LED

Shell *
MicroPython v1.9.4-8-ga9a3caad0 on 2018-05-11; ESP module with ESP8
266
Type "help()" for more information. [backend=GenericMicroPython]
>>>
```

3 按此鈕或按

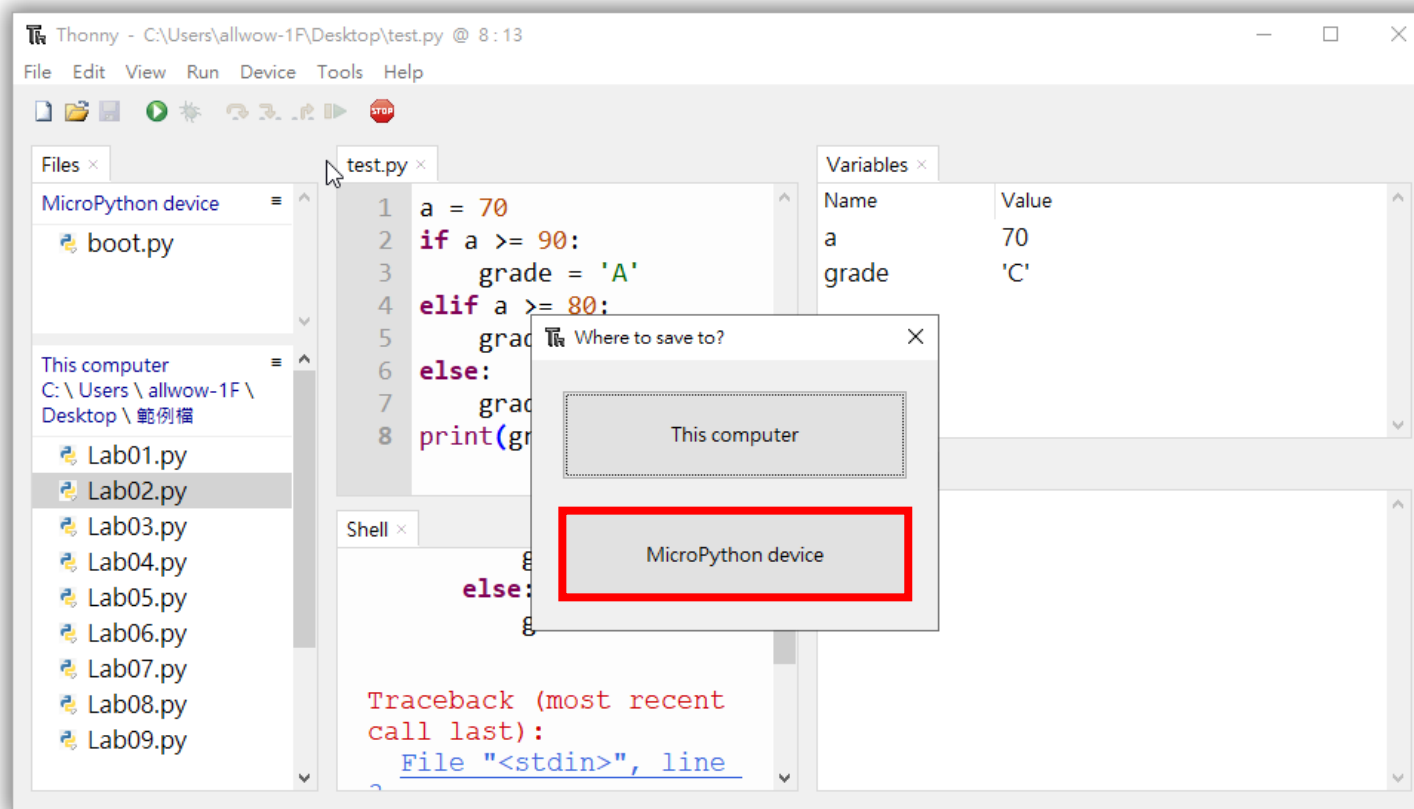
F5 執行程式

1 程式編輯區

輸入程式碼

上傳到.py檔到 D1 mini

- 上述是透過電腦直接在 D1 mini 上執行，並不會存到 D1 mini。
- 若要上傳到 D1 mini 中執行，將檔案另存到 D1 mini 中，並命名為 **main.py**。(main 是 D1 mini 開機預設執行的主程式。)

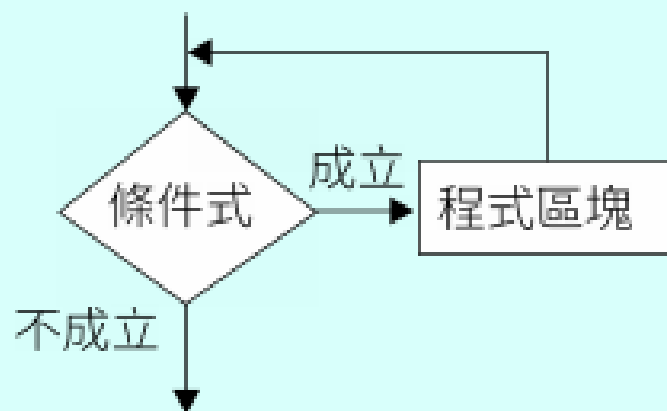


Python 流程控制 (while 迴圈)

上一個實驗我們用程式點亮 LED 3 秒後熄滅，如果我們想要做出一直閃爍的效果，該不會要寫個好幾萬行控制有電沒電的程式吧？！

當然不是！如果需要重複執行某項工作，可利用 Python 的 while 迴圈來依照條件重複執行。其語法如下：

while 條件式：
 程式區塊



Python 流程控制 (while 迴圈)

while 會先對條件式做判斷，如果條件成立，就執行接下來的程式區塊，然後再回到 while 做判斷，如此一直循環到條件式不成立時，則結束迴圈。



只要手沒斷 (條件式)
就一直重複 (while 迴
圈) 做伏地挺身 (程式
區塊)!

嗚~我要打家暴專線 ...



Python 流程控制 (while 迴圈)

通常我們寫程式控制硬體時，大多數的狀況下都會希望程式永遠重複執行，此時條件式就可以用 **True** 這個關鍵字來代替，True 在 Python 中代表『成立』的意義。

⚠ 關鍵字是 Python 保留下來有特殊意義的字。

例如我們要做出 LED 一直閃爍的效果，便可以使用以下程式碼：

```
while True:           # 一直重複執行
    led.value(1)      # 點亮 LED
    time.sleep(0.5)   # 暫停 0.5 秒
    led.value(0)      # 熄滅 LED
    time.sleep(0.5)   # 暫停 0.5 秒
```

Python 區塊縮排

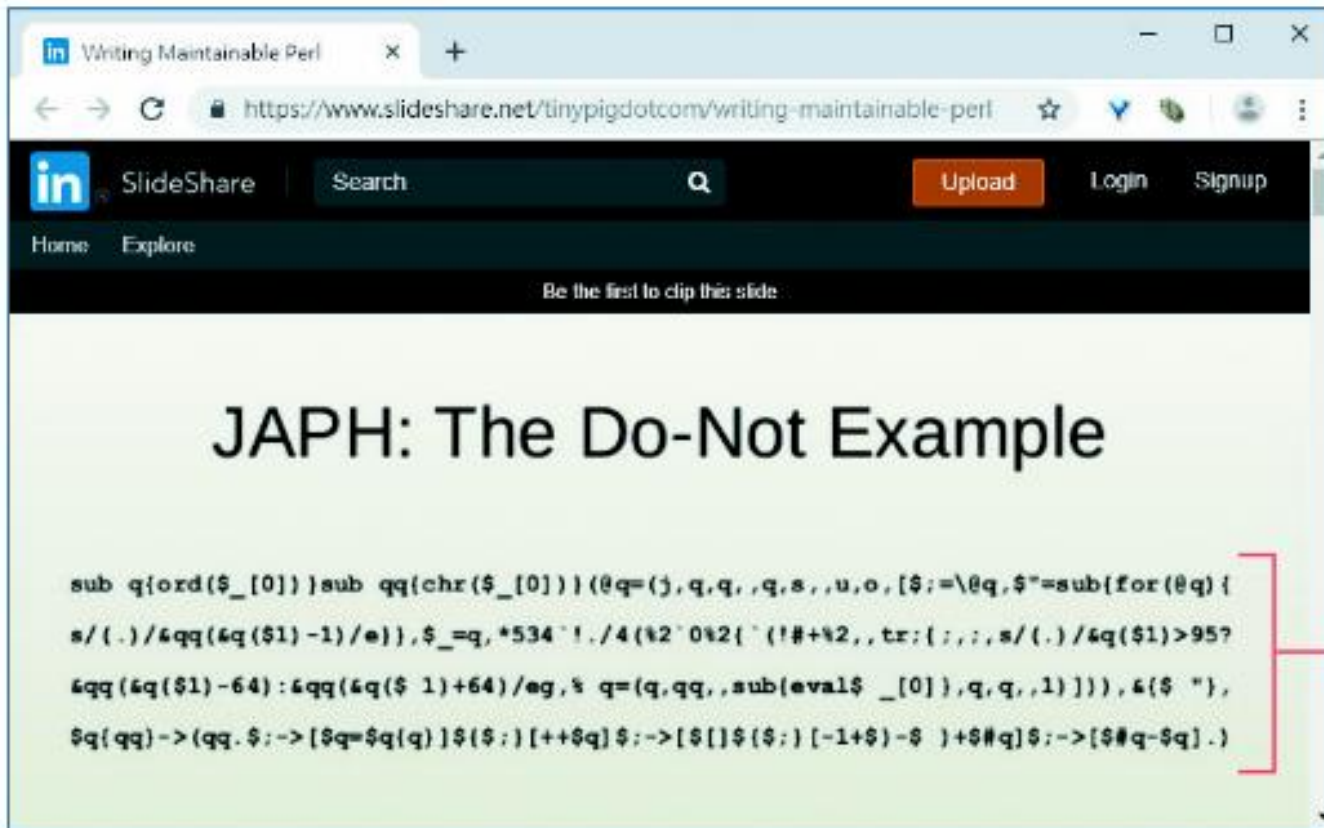
```
while True:           # 一直重複執行
    led.value(1)      # 點亮 LED
    time.sleep(0.5)   # 暫停 0.5 秒
    led.value(0)      # 熄滅 LED
    time.sleep(0.5)   # 暫停 0.5 秒
```

請注意！如上所示，屬於 `while` 的程式區塊要『以 4 個空格向右縮排』，表示它們是屬於上一行 (`while`) 的區塊，而其他非屬 `while` 區塊內的程式『不可縮排』，否則會被誤認為是區塊內的敘述。

其實 Python 允許我們用任意數量的空格或定位字元 (Tab) 來縮排，只要同一區塊中的縮排都一樣就好。不過建議使用 4 個空格，這也是官方建議的用法。

Python 區塊縮排

區塊縮排是 Python 的特色，可以讓 Python 程式碼更加簡潔易讀。其他的程式語言大多是用括號或是關鍵字來決定區塊，可能會有人寫出以下程式碼：



```
sub q{ord($_[0])}sub qq(chr($_[0])){@q=(j,q,q,,q,s,,u,o,[$;=\@q,$*=sub{for(@q){s/(.)/&qq(&q($1)-1)/e}),$_=q,*534`!./4(%2`0%2{`(!#+%2,,tr{:,:,:,s/(.)/&q($1)>95?&qq(&q($1)-64):&qq(&q($1)+64)/eg,% q=(q,qq,,sub{eval$_[0]},q,q,,1))},&($ "),$q(qq)->(qq.$:->[$q=$q(q)]$($:)[+$q]$:->[$[]$($:)[-1+$]-$ ]+$#q]$:->[$#q-$q].)}
```

沒有縮排全都擠在一起的程式碼

Lab02

閃爍 LED

實驗目的

用 Python 的 `while` 迴圈重複執行 LED 的控制程式, 使其每 0.5 秒閃爍一次。

材料

- D1 mini
- 紅色 LED
- 220 Ω 電阻



情境想一想 Consider this

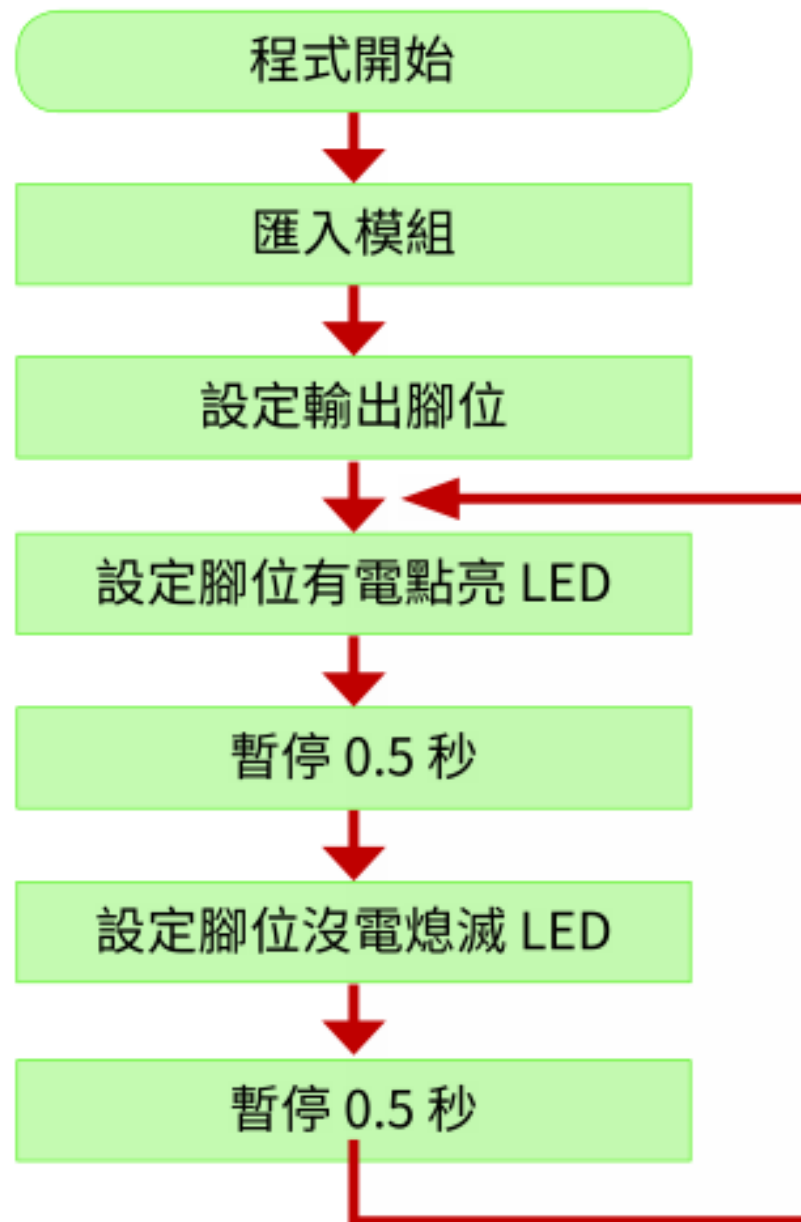
生活中常常需要重複動作的情境，請說明下列重複的動作為何，什麼條件下會停止工作，條件可能不只 1 項。

1. 登入網站時輸入密碼。
2. 聽音樂時開啟循環播放。
3. 參加球隊訓練，跑操場鍛鍊體能。
4. 上網搶購演唱會門票。
5. 玩數字賓果遊戲。

參考答案

1. 一直重複輸入帳號和密碼進行登入，直到輸入正確密碼完成登入為止；若密碼輸入次數太多，帳號會被鎖住。
2. 一直聆聽音樂，直到按下停止鈕為止。
3. 一直繞著操場跑道跑步，直到達成教練要求的圈數，或教練說停止。
4. 一直按購買選擇場次，直到買到門票或賣完了才停止。
5. 一直消去填寫的數字，直到有人消去的數字連成 5 條水平、垂直或對角線，遊戲才結束。

程式流程圖



程式設計

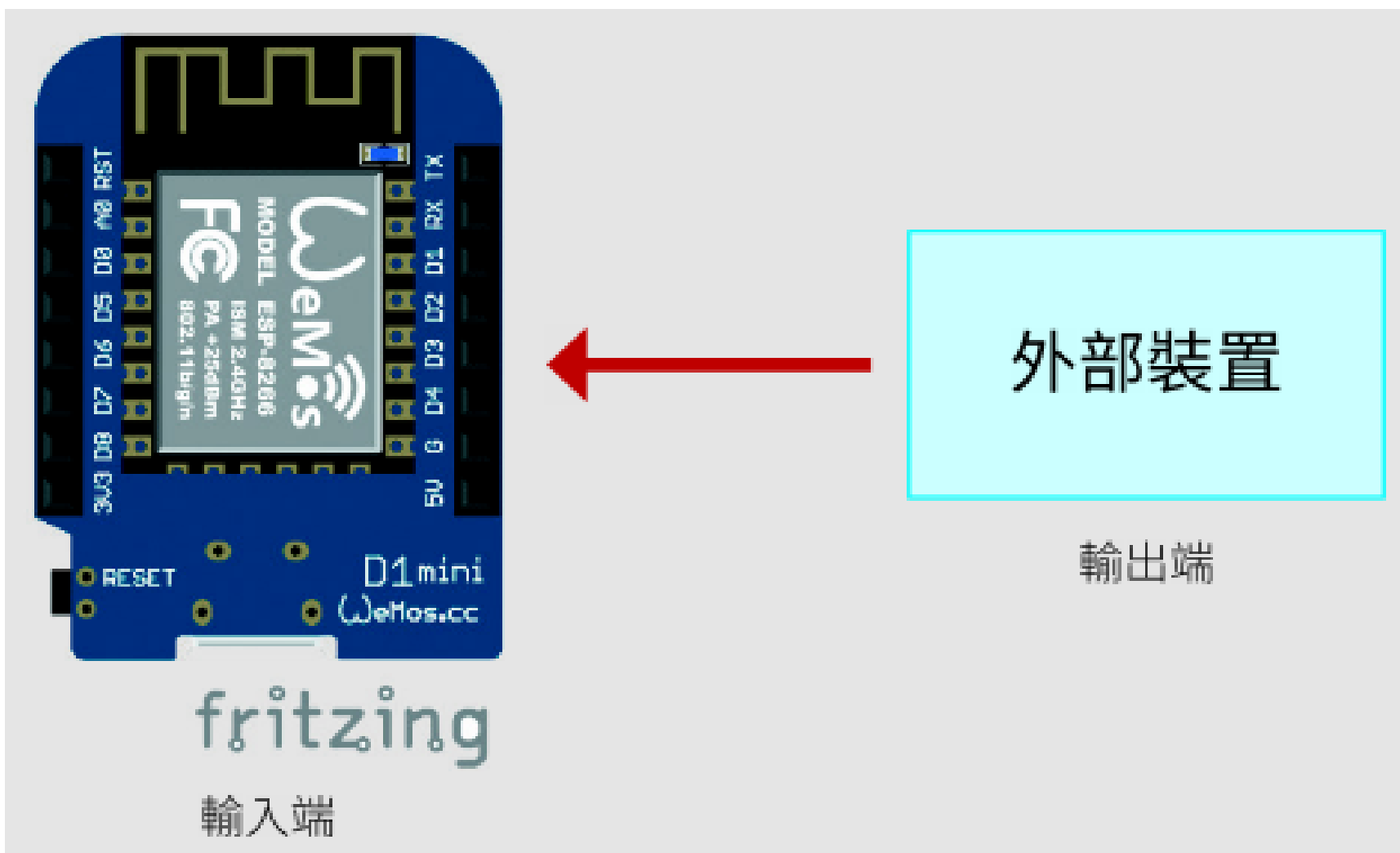
Lab02.py

```
01 # 從 machine 模組匯入 Pin 物件
02 from machine import Pin
03 # 匯入時間相關的 time 模組
04 import time
05
06 # 建立 15 號腳位的 Pin 物件，設定為輸出腳位，並命名為 led
07 led = Pin(15, Pin.OUT)
08
09 while True:                # 一直重複執行
10     led.value(1)           # 點亮 LED
11     time.sleep(0.5)        # 暫停 0.5 秒
12     led.value(0)           # 熄滅 LED
13     time.sleep(0.5)        # 暫停 0.5 秒
```

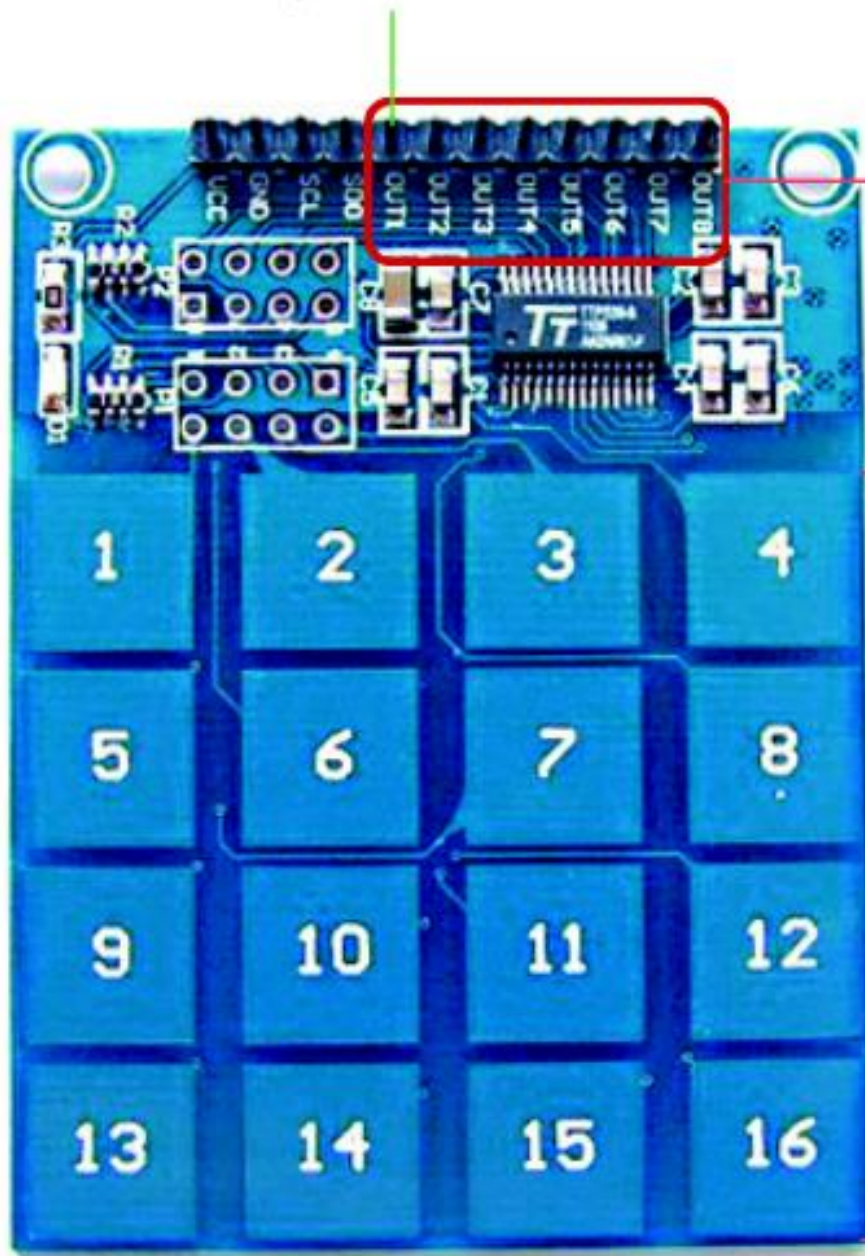
04 讀取按鈕 - 數位輸入

Python 流程控制 (if...else)

數位輸入



1 號按鈕有觸碰時，OUT1 腳位會輸出高電位，否則輸出低電位



OUT1~OUT8 是訊號輸出腳位，分別對應 1~8 號按鈕

觸摸偵測區

⚠ 本套件只會使用 1~8 號按鈕，9~16 號按鈕不會使用。

Lab03

讀取觸控按鈕的輸入值

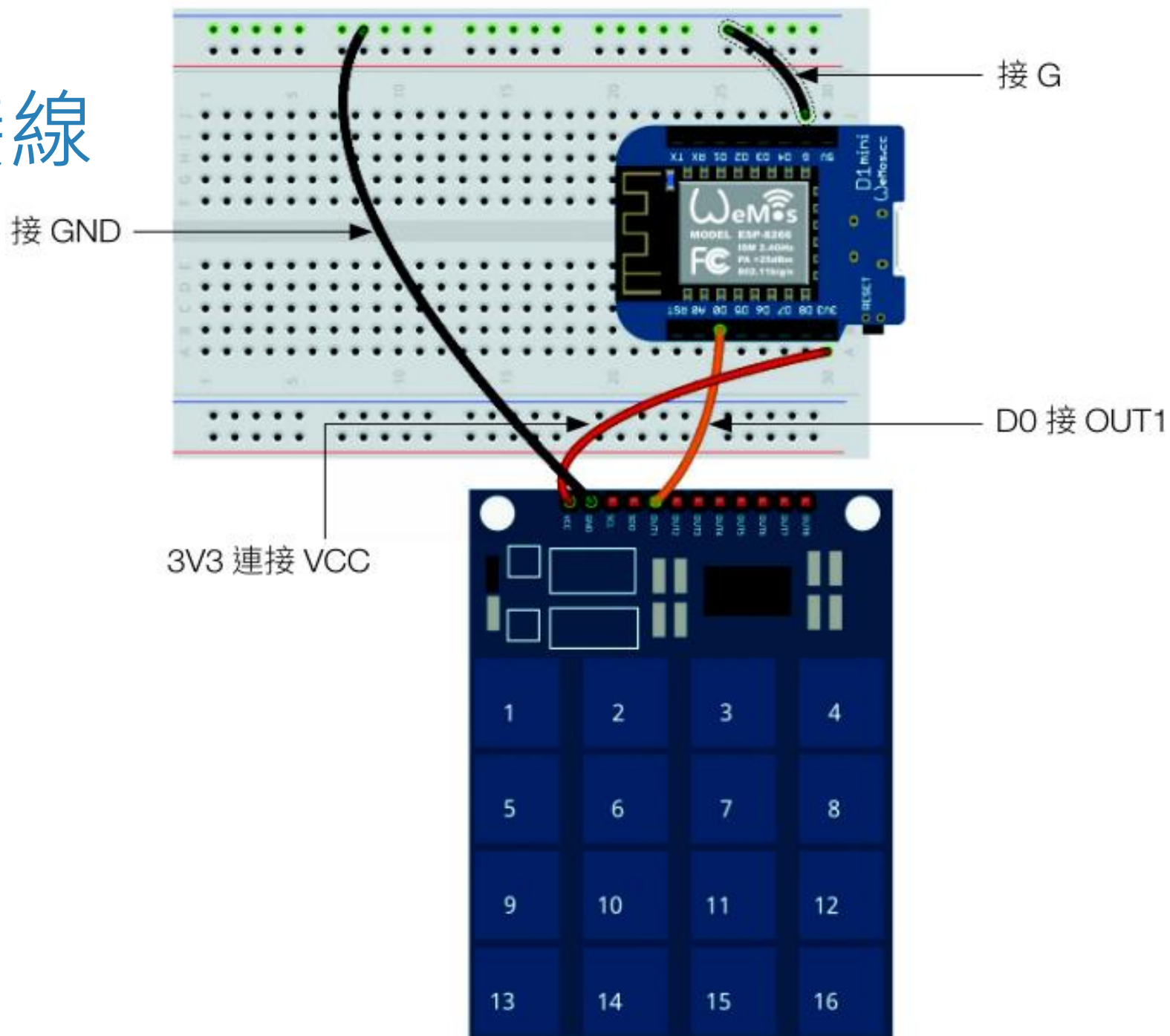
實驗目的

用程式讀取觸控按鈕的輸入值, 藉以判斷按鈕是否有被觸碰

材料

- D1 mini
- 電容式觸控按鈕模組

請依線路圖接線



■ 設計原理

當我們建立腳位的 Pin 物件時，可用 **"Pin.IN"** 作為參數，設定這個腳位為輸入腳位：

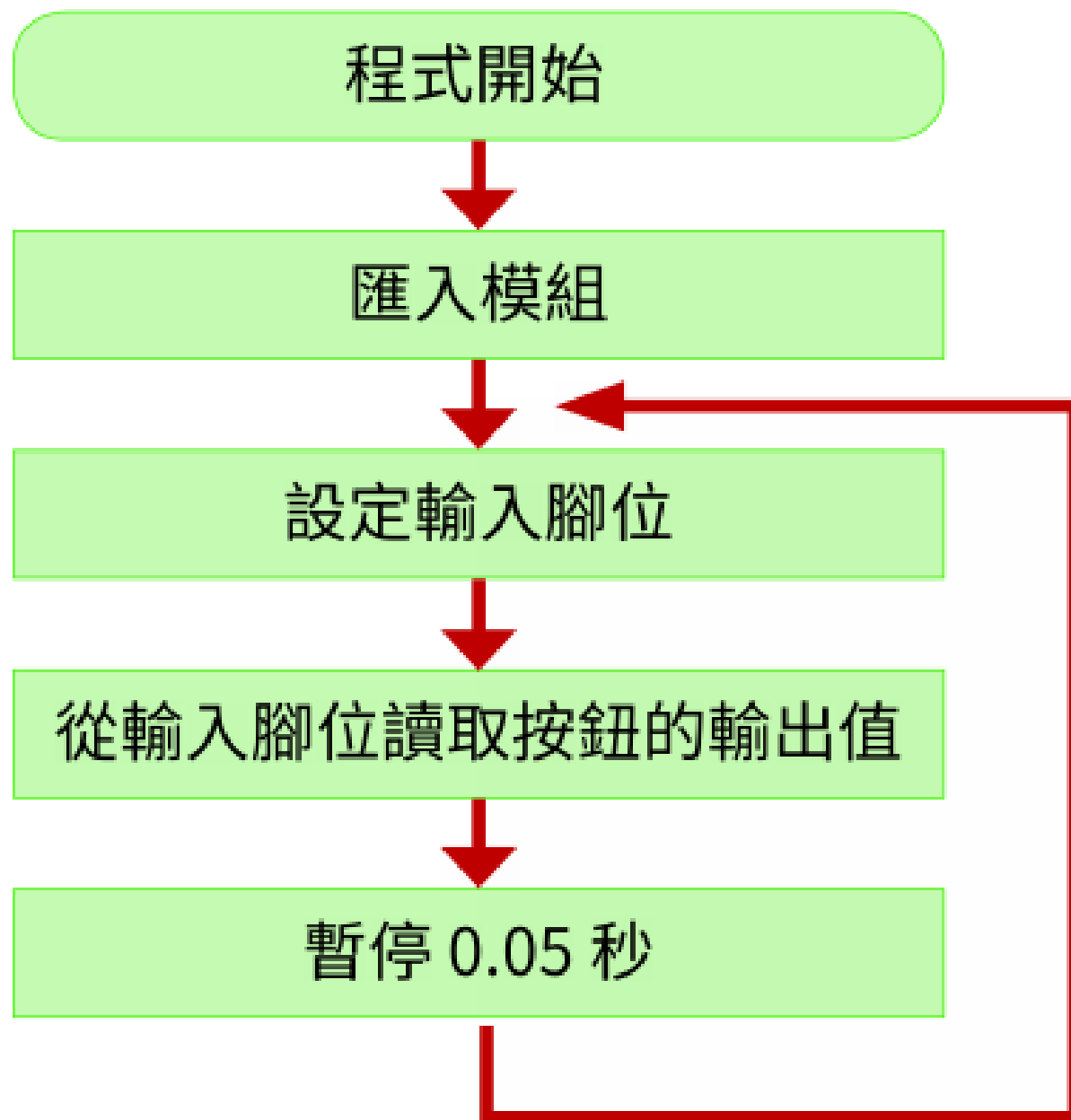
```
>>> from machine import Pin
>>> button = Pin(16, Pin.IN)
```

上面我們建立了 16 號腳位的 Pin 物件，並且將其命名為 button，因為建立物件時使用了 **"Pin.IN"** 參數，所以 16 號腳位就會被設定為輸入腳位。

建立好輸入腳位的 Pin 物件後，便可以使用 value() 方法來讀取外部裝置輸出的電位高低：

```
>>> button.value()
0           ← 讀到 0 表示外部裝置輸出低電位
>>> button.value()
1           ← 讀到 1 表示外部裝置輸出高電位
```

程式流程圖



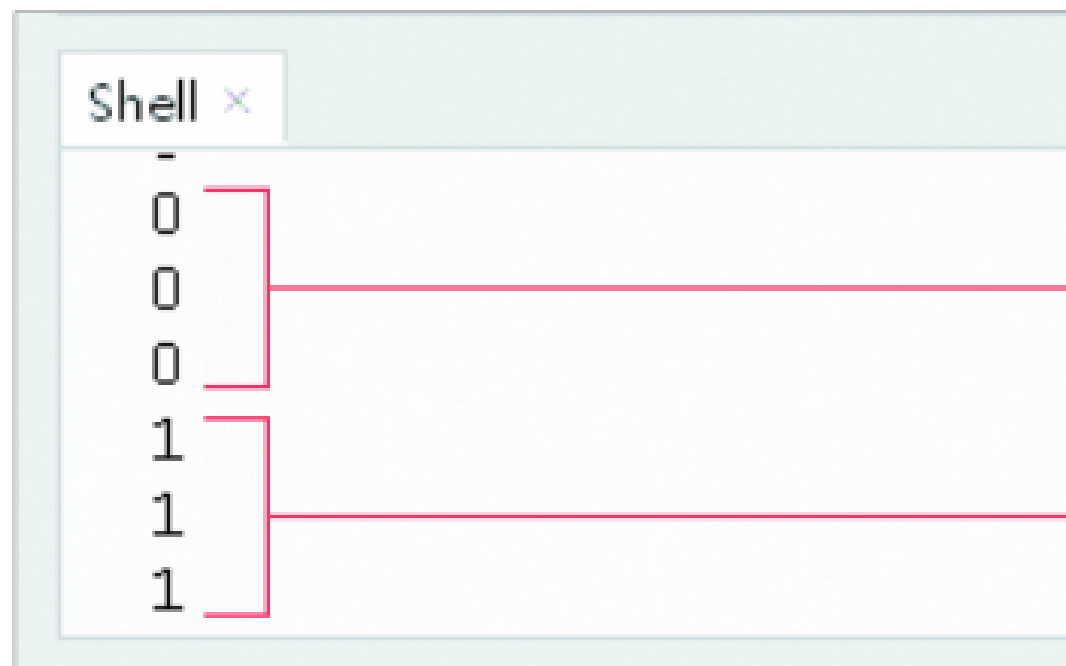
程式設計

Lab03.py

```
01 from machine import Pin
02 import time
03
04 # 建立 16 號腳位的 Pin 物件，設定為輸入腳位，並命名為 button
05 button = Pin(16, Pin.IN)
06
07 while True:
08     # 用 value() 方法從 16 號腳位讀取按鈕輸出的高低電位
09     # 然後將讀到的值用 print() 輸出
10     print(button.value())
11
12     # 暫停 0.05 秒
13     time.sleep(0.05)
```

實測

請按 **F5** 執行程式，然後用手指觸摸一下觸控按鈕模組的 1 號按鈕，在 Thonny 的 Shell 窗格觀察程式輸出的值：



The screenshot shows a Thonny Shell window with a title bar 'Shell x'. The output consists of six lines of numbers: 0, 0, 0, 1, 1, 1. Red brackets group the first three '0's and the last three '1's. Red lines connect these groups to explanatory text on the right.

```
Shell x
-
0
0
0
1
1
1
```

未觸碰 1 號按鈕時，
得到的值為 0

若觸碰 1 號按鈕，
得到的值為 1

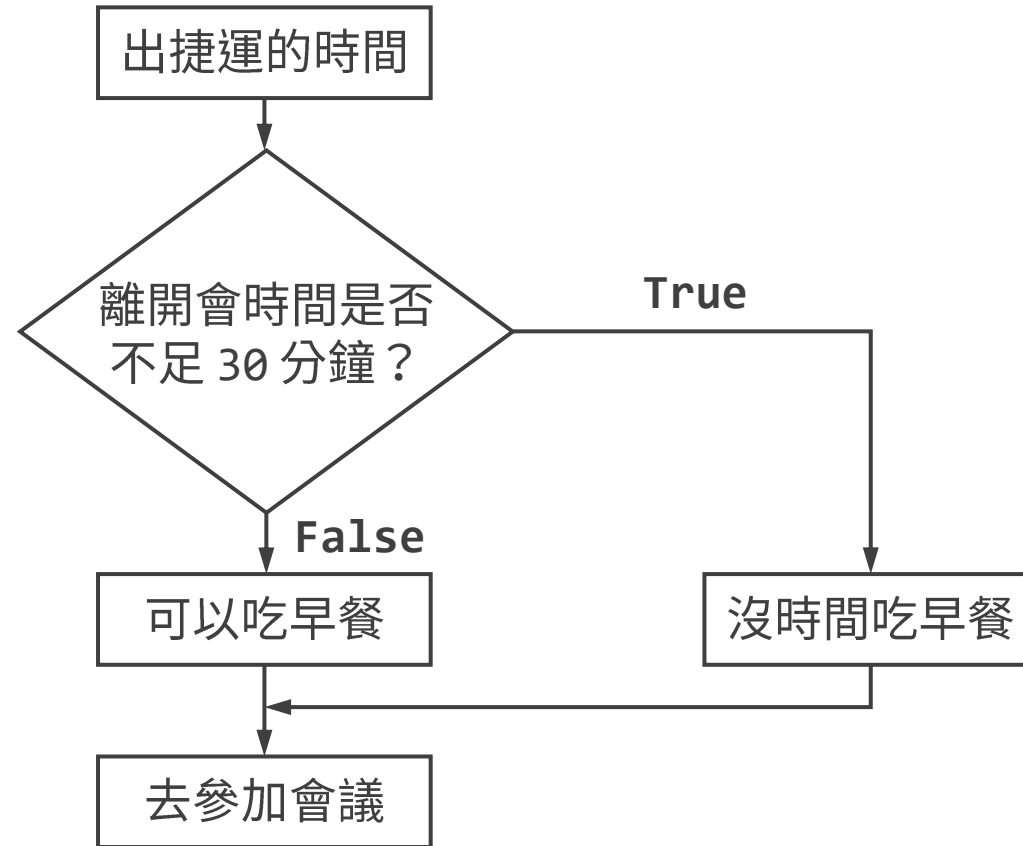
Lab04

用觸控按鈕控制 LED

實驗目的	如果觸控按鈕被觸碰，便點亮 LED 燈，否則便熄滅 LED 燈。
材料	<ul style="list-style-type: none">● D1 mini● 電容式觸控按鈕模組● 紅色 LED● 220 Ω 電阻

💡 情境想一想

星期一早上，你的第一場會議開始時間是 8:30 AM。吃早餐的時間約 30 分鐘。現在假設走出捷運看一下時間，你如何判斷使否還有時間吃早餐呢？請使用下列流程圖來判斷。



參考答案

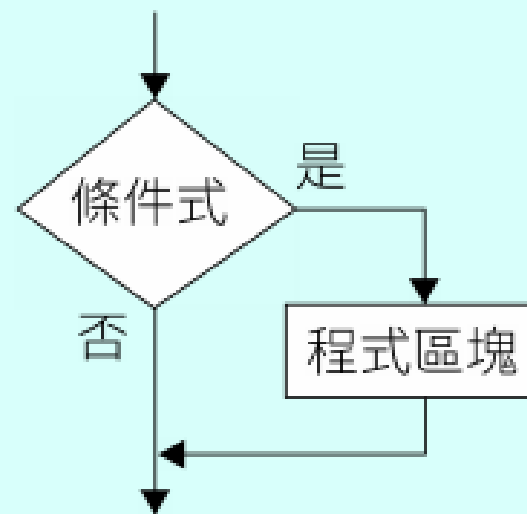
只要在 8:00 AM 前走出車站，就還有時間吃早餐。

Python 流程控制 (if..else)

if 可以在程式中做「**如果 ... 就 ...**」的判斷，寫法如下：

if 條件式： ← 注意最後要加：

程式區塊
... } ← 可以有多行程式，
每行都要向右縮排



以上就是「當**條件式**成立時就執行**程式區塊**」內的敘述，否則略過**程式區塊**。

Python 流程控制 (if..else)

```
if a < 1:
    a += 1
    b = a + 3 } ← 程式區塊
print(b) ← 接下來的程式未縮排，不屬於 if 區塊了
...
```

條件式可以用 $>$ (大於)、 \geq (大於等於)、 $<$ (小於)、 \leq (小於等於)、 $==$ (等於)、 $!=$ (不等於) 來比較。

和 While 一樣屬於 if 的程式區塊要「以 4 個空格向右縮排」，表示它們是屬於上一行 (if...:) 的區塊，而其他非區塊內的敘述則「不可縮排」，否則會被誤認為是區塊內的敘述。

Python 流程控制 (if..else)

■ if...elif...else...

如果想讓 if 多做一點事，例如「**如果 ... 就 ... 否則就 ...**」，那麼可加上 else：

if 條件式：

 程式區塊 ← 條件成立時要執行的程式

else：

 程式區塊 ← 條件不成立時要執行的程式

程式說明

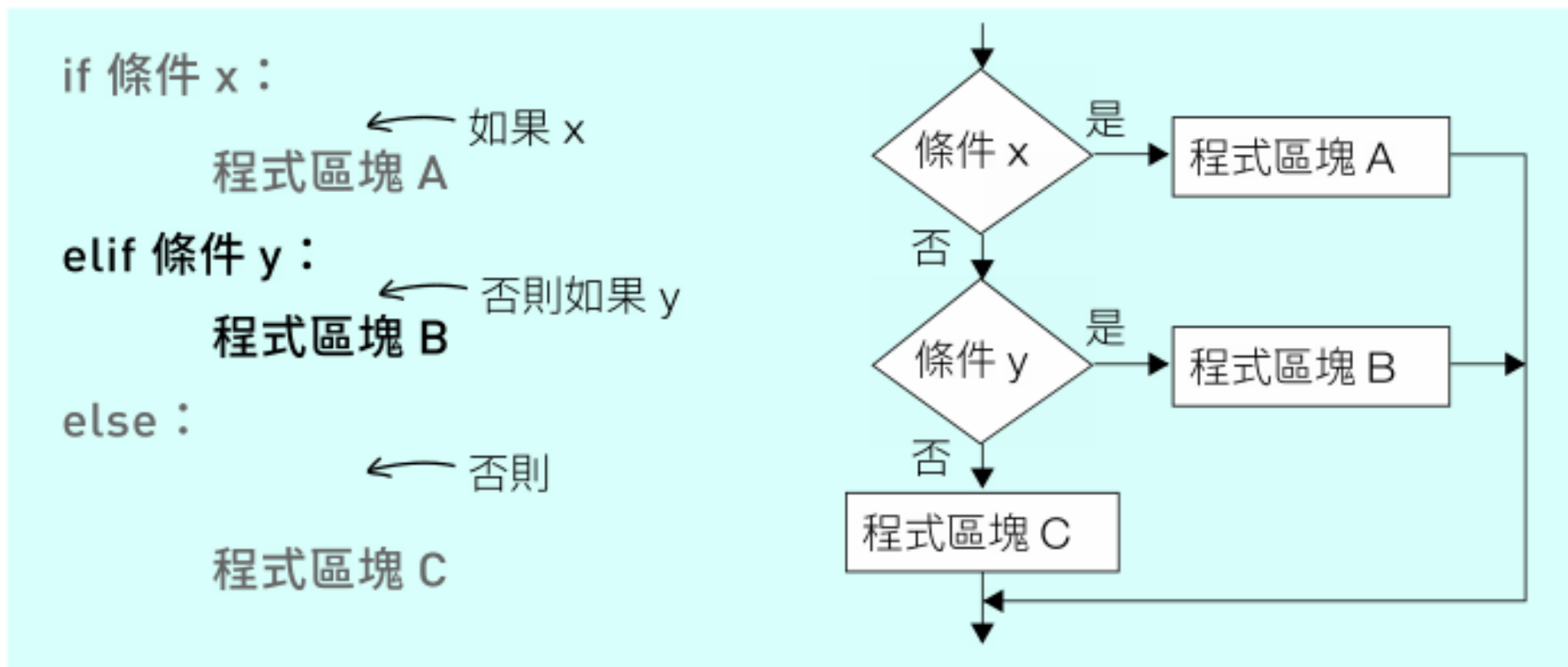
一個整數除以 2，不是 0 就是 1，適合用 if-else。

```
num = int(input("please pick a number: "))
if num % 2 == 0:
    print("Even Number")
else:
    print("Odd Number")

print("End")
```

Python 流程控制 (if..else)

又如果想做更多的判斷，例如「**如果** x 就 A **否則如果** y 就 B **否則**就 C」，則可再加上代表**否則如果**的 elif：



依序確認 if-elif-else 中的條件，只要某一個條件成立，就會執行該程式區塊，其他條件就不會再進行判斷。

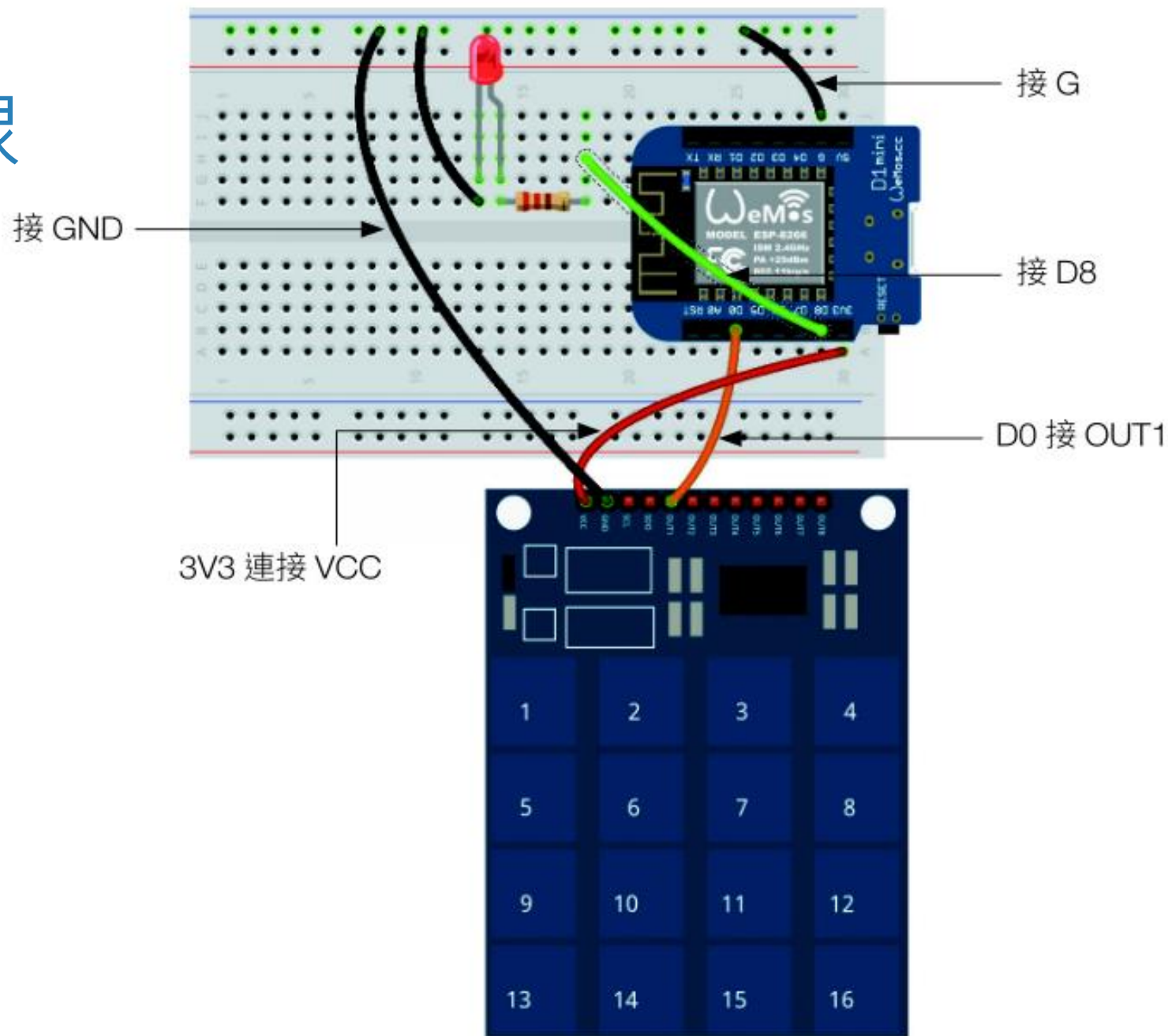
程式說明 判斷數字為正負數或 0。

```
num = int(input("Enter a number: "))
if num > 0:
    print("num is positive")
elif num < 0:
    print("num is negative")
else:
    print("num is zero")
print("End")
```

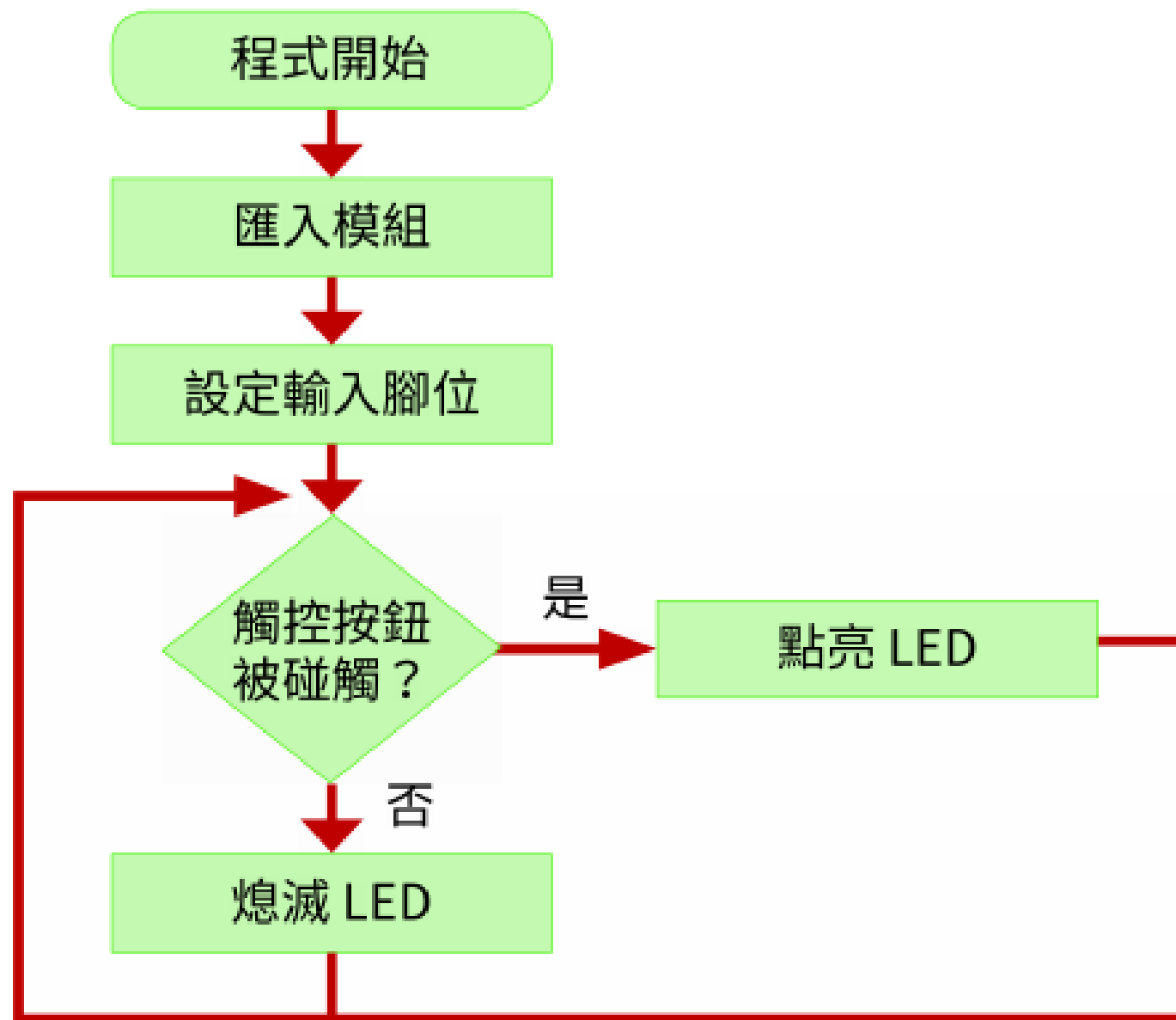
程式說明 判斷分數，給出評分。

```
a = 70
if a >= 90:
    grade = 'A'
elif a >= 80:
    grade = 'B'
else:
    grade = 'C'
print(grade)
```

請依線路圖接線



程式流程圖



程式設計

Lab04.py

```
01 from machine import Pin
02
03 # 建立 15 號腳位的 Pin 物件, 設定為輸出腳位, 並命名為 led
04 led = Pin(15, Pin.OUT)
05 # 建立 16 號腳位的 Pin 物件, 設定為輸入腳位, 並命名為 button
06 button = Pin(16, Pin.IN)
07
08 while True:
09     if button.value() == 1: # 如果觸控按鈕被碰觸
10         led.value(1)      # 點亮 LED
11     else:                  # 否則 (觸控按鈕沒有碰觸)
12         led.value(0)      # 熄滅 LED
```

實測

請按 **F5** 執行程式，然後用手指觸碰觸控按鈕模組的 1 號按鈕，即可看到 LED 被點亮，若放開則 LED 便會熄滅。

05 光感應自動電燈 - 類比輸入

Python 流程控制 (if...else)

硬體模組

認識光敏電阻

光敏電阻 (photoresistor) 是一種會因為光線明暗而改變導電效應的電阻，電阻值與光線亮度成反比，光線越亮則電阻值越小，所以我們可以利用光敏電阻來偵測目前環境的明暗度。

不過 D1 mini 控制板並沒有偵測電阻值的能力，為了取得光敏電阻的偵測結果，我們將採用電壓分配規則來計算光敏電阻的電阻變化。



使用 ADC 偵測電壓變化

在電子的世界中，訊號只分為有電跟沒電兩個值，這個稱之為**數位訊號** (0/1、High/Low、或 On/Off)，所以前面章節我們使用 D1 mini 輸出或輸入時，只能輸出 / 輸入高、低電位兩個值。

但電壓變化不是這樣的二分值，而是連續的變化，例如 1V、2.1V 等都是可能的值，這種訊號稱為**類比值**。

為了偵測光敏電阻導致的電壓變化，必須透過 **ADC (Analog-to-Digital Conversion, 類比數位轉換器)**，將電壓值轉換為電腦可以讀取的數位值。

D1 mini 控制板具備 ADC 的是 A0 腳位，當 A0 腳位有電壓輸入時，ADC 會將 0~3.2V 電壓範圍轉成 0~1024 再傳給 D1 mini。所以傳回值 1024 就是 3.2V 電壓輸入，341 表示大約 1.1V 電壓輸入。也就是說，將傳回值先除以 1024 再乘上 3.2 就可以換算成電壓。

Lab05

讀取光敏電阻的輸入值

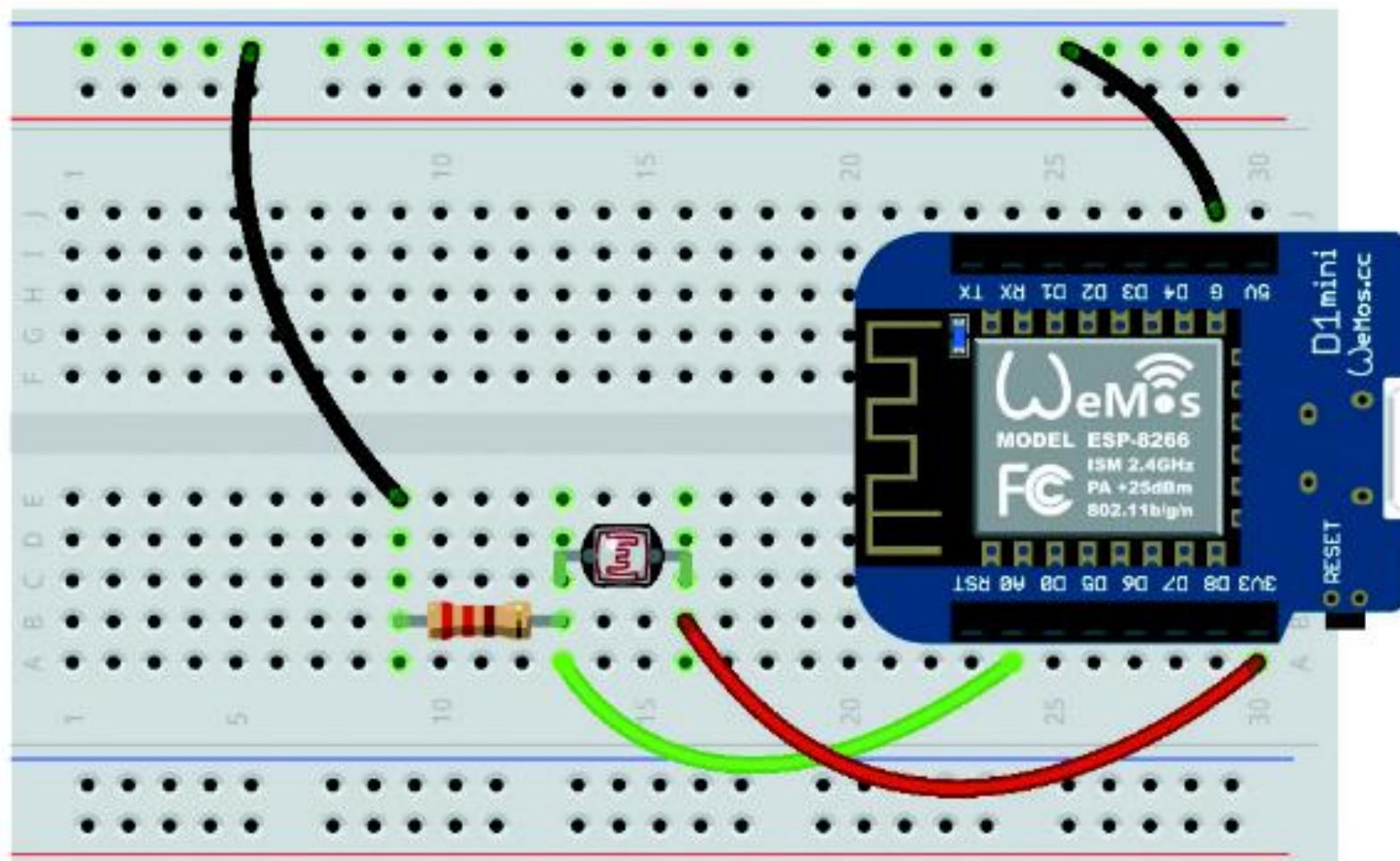
實驗目的

用程式讀取光敏電阻壓降後的電壓。

材料

- D1 mini
- 光敏電阻
- 220 Ω 電阻

請依線路圖接線



fritzing

■ 設計原理

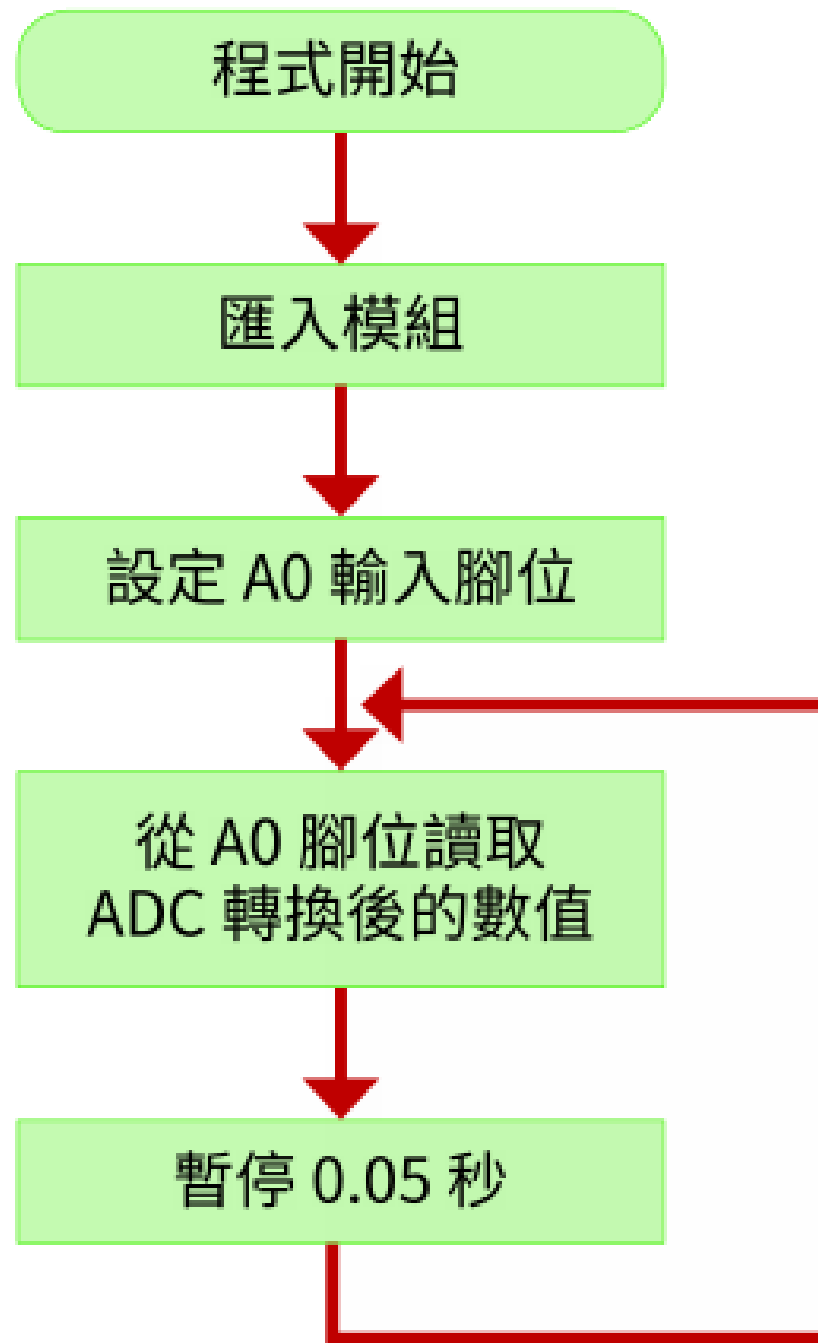
請使用以下語法建立 A0 腳位的 ADC 物件：

```
>>> from machine import ADC
>>> adc = ADC(0)
```

然後使用 `read()` 方法即可讀取 ADC 轉換後的數值，數值越大表示電壓越大：

```
>>> adc.read()
152
>>> adc.read()
168
```

程式流程圖



程式設計

Lab05.py

```
01 from machine import ADC
02 import time
03
04 # 建立 A0 腳位的 ADC 物件, 並命名為 adc
05 adc = ADC(0)
06
07 while True:
08     # 用 read() 方法從 A0 號腳位讀取 ADC 轉換後的數值
09     # 然後將讀到的值用 print() 輸出
10     print(adc.read())
11
12     # 暫停 0.05 秒
13     time.sleep(0.05)
```

實測

請按 **F5** 執行程式，然後用手放在光敏電阻上面擋住光線，在 Thonny 的 Shell 窗格觀察程式輸出的值：

```
Shell x
124
108
81
84
75
77
110
150
```

擋住光線後，光敏電阻的電阻值會變大，所以 ADC 輸入值會變小

手放開不要擋住光線，光敏電阻的電阻值會變小，所以 ADC 輸入值會變大

經過實測後，我們發現光線不足時 ADC 輸入值會小於 100，光線充足的話 ADC 輸入值會大於 100，所以接下來我們會用 100 這個數值來判斷光線是否充足。

Lab06

光感應自動電燈

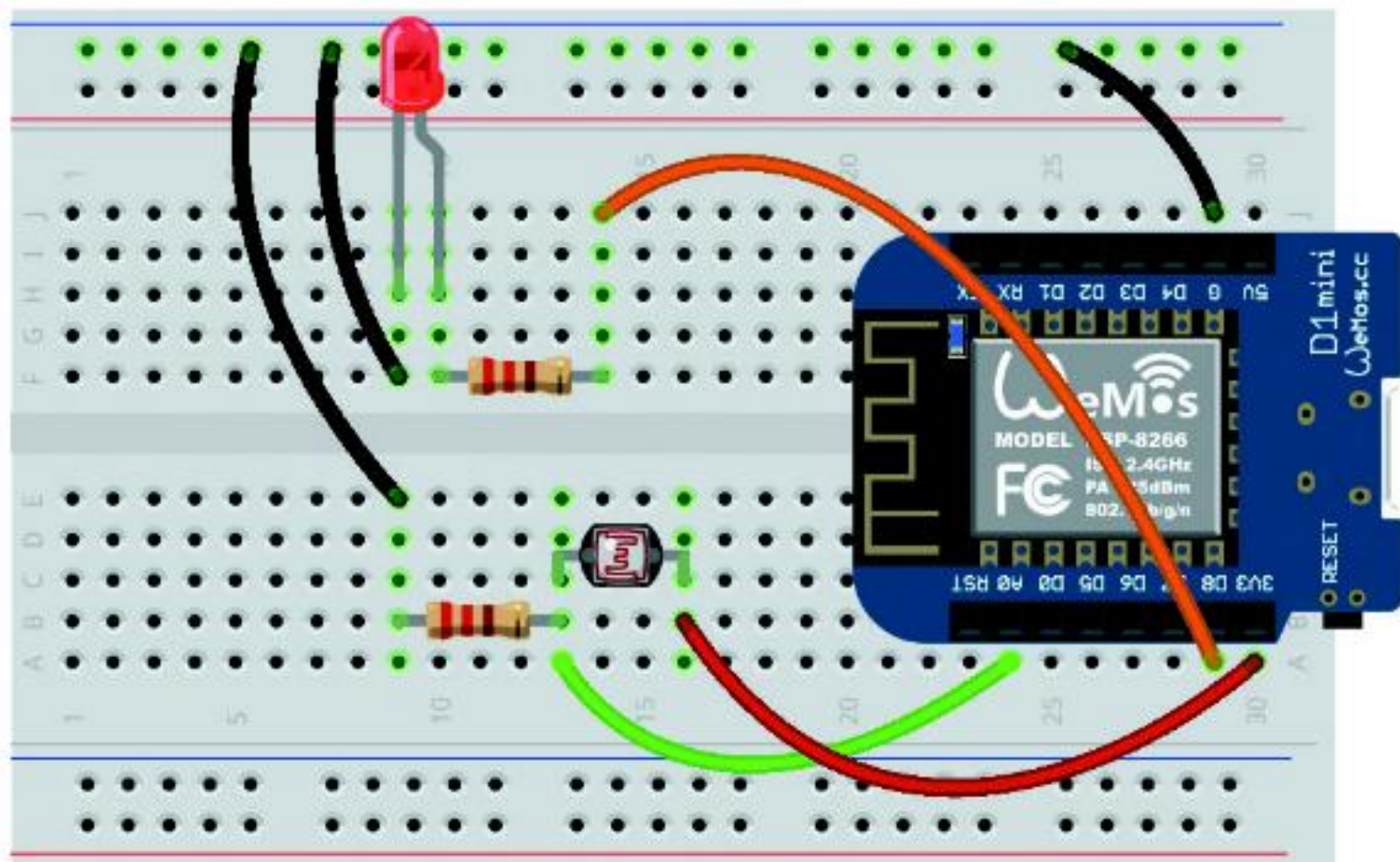
實驗目的

偵測環境光線不足自動打開電燈, 或光線充足則關閉電燈。

材料

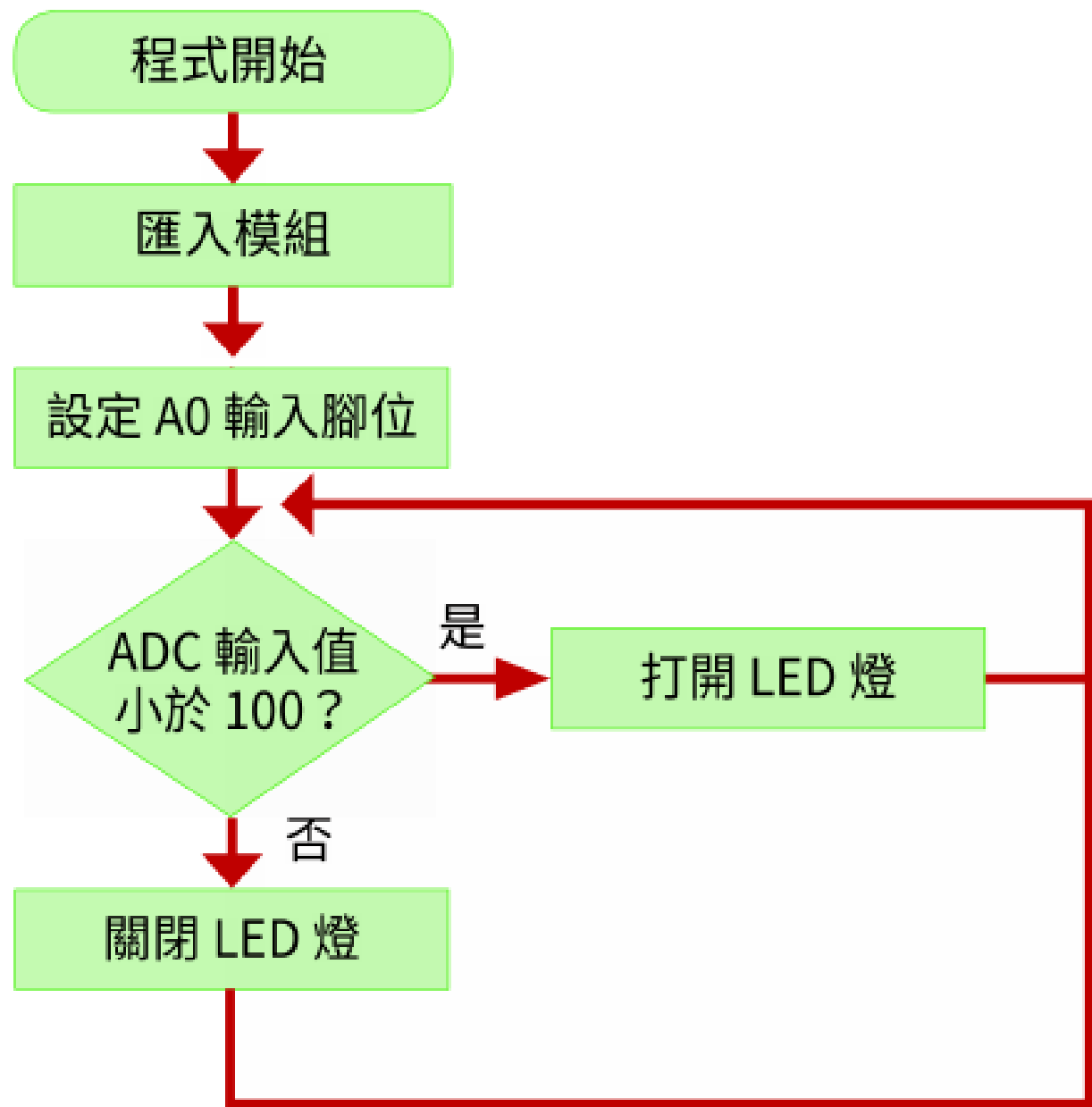
- D1 mini
- 光敏電阻
- 紅色 LED
- 220 Ω 電阻 \times 2

請依線路圖接線



fritzing

程式流程圖



程式設計

Lab06.py

```
01 from machine import ADC, Pin
02
03 # 建立 A0 腳位的 ADC 物件, 並命名為 adc
04 adc = ADC(0)
05 # 建立 15 號腳位的 Pin 物件, 設定為輸出腳位, 並命名為 led
06 led = Pin(15, Pin.OUT)
07
08 while True:
09     if adc.read() < 100: # 光線不足
10         led.value(1)    # 打開 LED 燈
11     else:                # 否則
12         led.value(0)    # 關閉 LED 燈
```

實測

請按 **F5** 執行程式，然後用手放在光敏電阻上面擋住光線，此時可以看到 LED 燈亮起，將手移走則 LED 會熄滅。

06 LED 呼吸燈 - 類比輸出

Python 流程控制 (for 迴圈)

用 PWM 類比輸出控制 LED 亮度

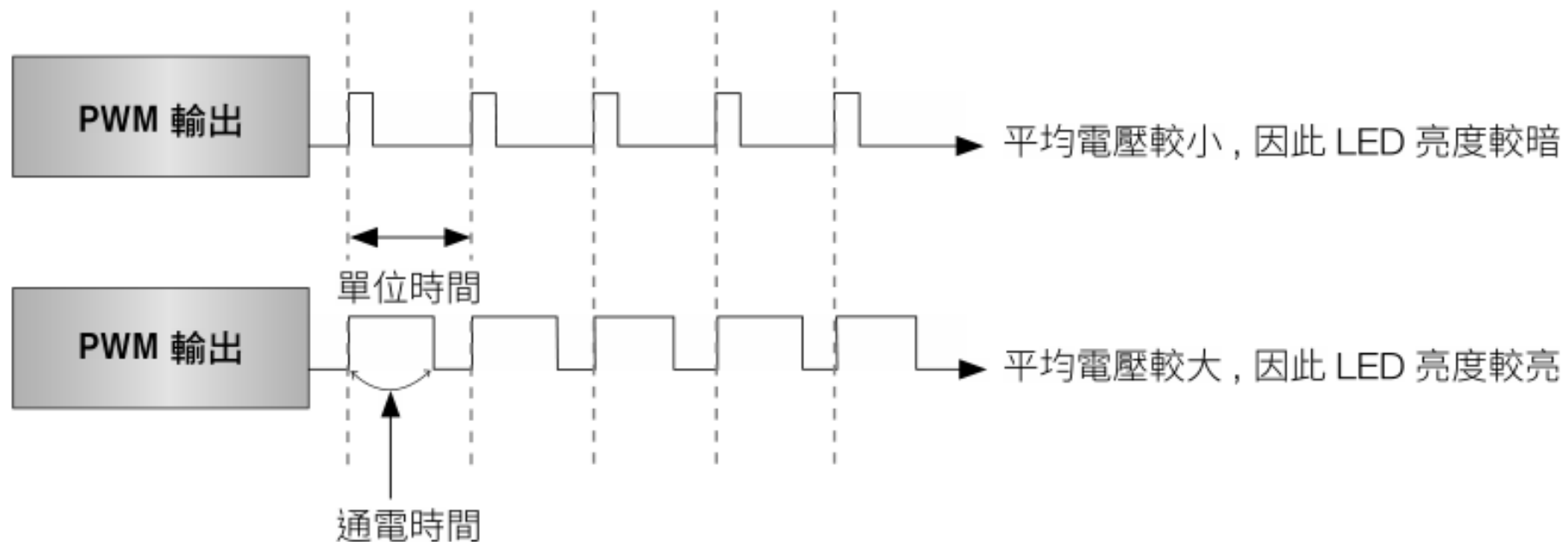
第 3 章我們曾經說明 LED 的發光方式是長腳接高電位，短腳接低電位，像水往低處流一樣產生高低電位差，讓電流流過 LED 即可發光。若是長腳連接的電壓越高，LED 發出的光就會越亮。

但是在電子數位的世界裡面，狀態只有 0/1 (無 / 有、關 / 開) 兩種，因此 D1 mini 控制板上的 IO 腳位電壓輸出只能有 0V 與 3.3V 兩種，為了要控制 LED 的亮度，我們將採用 **PWM (Pulse Width Modulation, 脈波寬度調變)**。

PWM 的概念很簡單，數位世界只有 0/1，所以只有高、低電位兩種變化，但是我們可以加上時間因素，以通電時間的長短來呈現強弱的概念。

用 PWM 類比輸出控制 LED 亮度

當同樣單位時間內 LED 通電的時間較久，LED 的亮度會較高；反之就會讓 LED 的亮度變低。也就是說只要以 PWM 改變單位時間內的通電時間，即可模擬輸出不同電壓的電流，因而讓 LED 有不同的亮度。



用 PWM 類比輸出控制 LED 亮度

由於 PWM 是不斷的在高、低電位間切換，也就是說 LED 實際上是不斷在通電、斷電間切換，若切換的速度（頻率）很快，感覺就會像是輸出連續的電力。

設定 PWM 時，PWM 是以百分比（稱為 Duty Cycle, 負載率，亦稱佔空比）來表示。例如 D1 mini 的 PWM 最大值為 1023, 若是設定 PWM 值為 818, 則負載率等於 $818 \div 1023$ 約為 80%, 表示該腳位 80% 的時間是高電位。

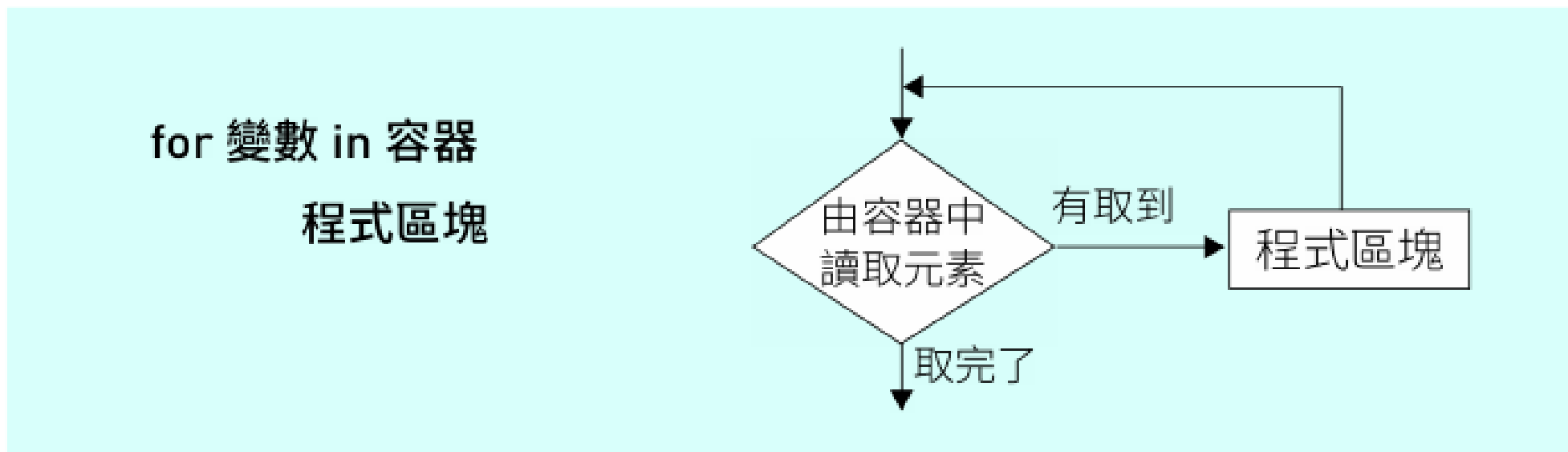
Python 流程控制 (for 迴圈)

當我們用 PWM 控制 LED 亮度時，可以使用的值為 0~1023, 所以若要寫程式控制 LED 顯示呼吸燈的效果時，最直覺的步驟如下；

```
設定 PWM 值等於 0, 控制 LED 亮度 ← 最暗 (熄滅)
設定 PWM 值等於 1, 控制 LED 亮度
設定 PWM 值等於 2, 控制 LED 亮度
設定 PWM 值等於 3, 控制 LED 亮度
設定 PWM 值等於 4, 控制 LED 亮度
設定 PWM 值等於 5, 控制 LED 亮度
...
設定 PWM 值等於 1018, 控制 LED 亮度
設定 PWM 值等於 1019, 控制 LED 亮度
設定 PWM 值等於 1020, 控制 LED 亮度
設定 PWM 值等於 1021, 控制 LED 亮度
設定 PWM 值等於 1022, 控制 LED 亮度
設定 PWM 值等於 1023, 控制 LED 亮度 ← 最亮
```

Python 流程控制 (for 迴圈)

為了解決這個問題，Python 提供了 for 迴圈的語法，for 迴圈可將容器中的元素一一讀取出來做處理，其語法如下：



⚠ 與 while 和 if 一樣，for 迴圈的程式區塊也要內縮 4 個空白。

Python 流程控制 (for 迴圈)

為了產生一個有 0~1023 數值的容器，我們還可以使用 Python 內建的 `range()` 來產生一個指定範圍的數列容器，其語法如下：

```
range(x)           ← 產生「由 0 到 x 但不包含 x」的數列  
range(x, y)        ← 產生「由 x 到 y 但不包含 y」的數列  
range(x, y, z)     ← 產生「由 x 到 y 但不包含 y, 間隔為 z」的數列
```

Python 流程控制 (for 迴圈)

for 迴圈搭配 range() 的範例如下：

```
>>> for i in range(10):  
    print(i)
```

← 產生 0 到 10 但不包含 10 的數列
← 輸出 0 1 2 3 4 5 6 7 8 9
← Thonny 會幫你自動內縮

0
1
2
3
4
5
6
7
8
9

多按一次空行才會結束 for 執行

Python 流程控制 (for 迴圈)

```
>>> for i in range(1, 11):  
        print(i)
```

← 產生 1 到 11 但不包含 11 的數列
← 輸出 1 2 3 4 5 6 7 8 9 10

1
2
3
4
5
6
7
8
9
10

 可以用「有頭無尾」的口訣來記憶 range() 會產生的數列！

Lab07

漸亮 LED 燈

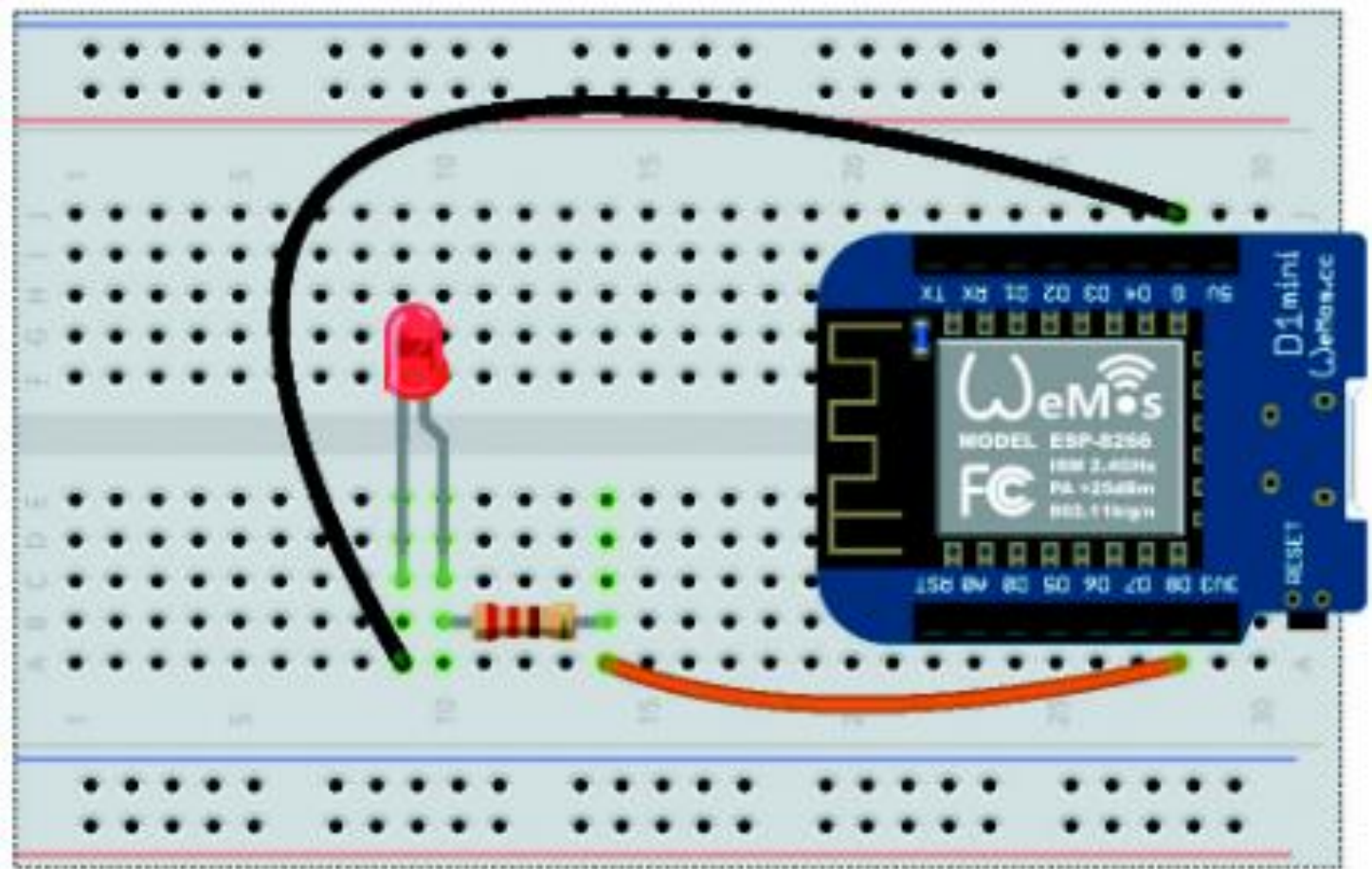
實驗目的

用 PWM 控制 LED 的亮度, 使其由暗至亮逐漸亮起。

材料

- D1 mini
- 紅色 LED
- 220 Ω 電阻

請依線路圖接線



fritzing

■ 設計原理

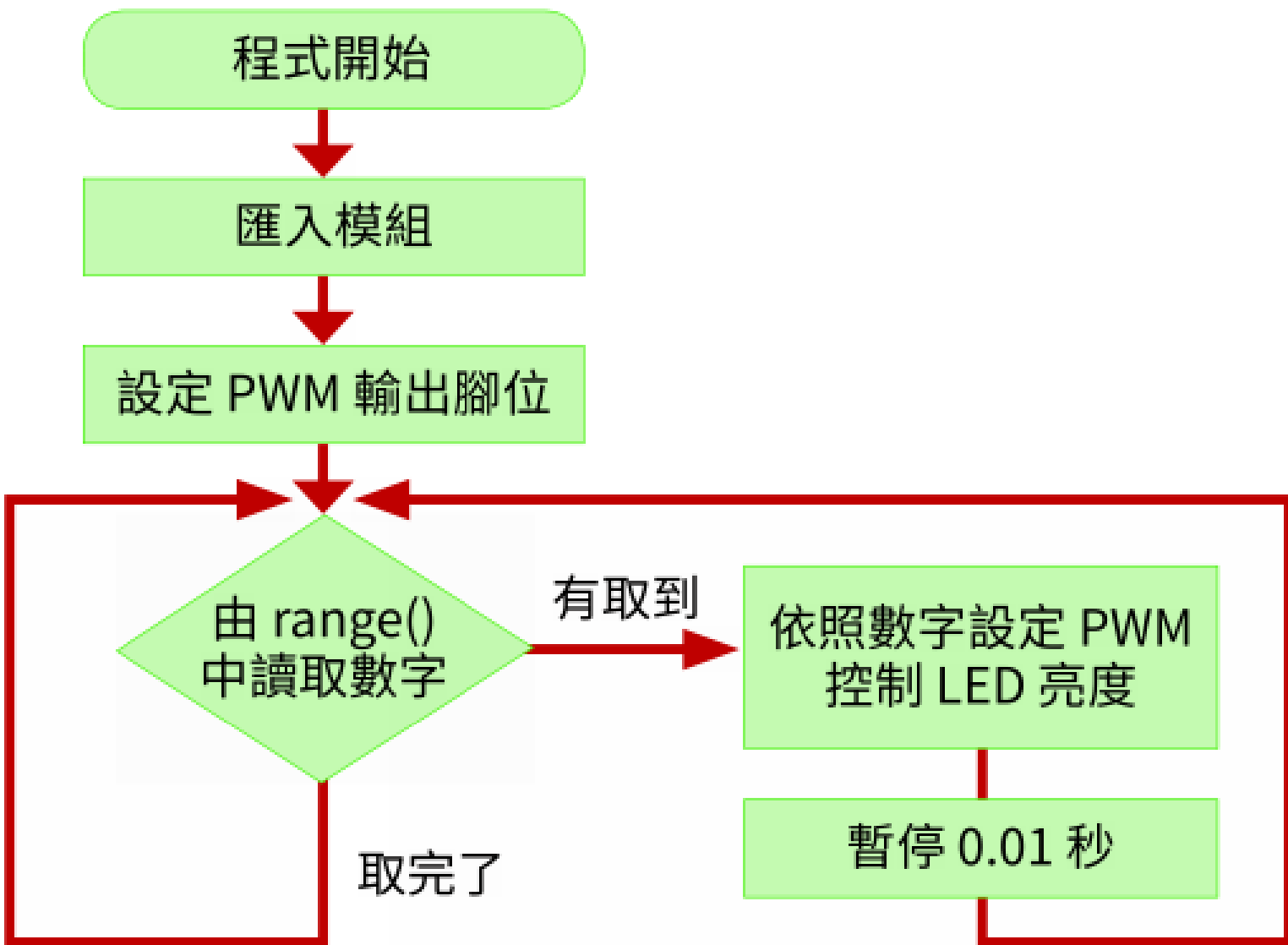
當我們建立腳位的 Pin 物件後，將這個 Pin 物件作為參數再建立 PWM 物件，便可以設定這個腳位為 PWM 輸出腳位：

```
>>> from machine import Pin, PWM
>>> led = PWM(Pin(15))
```

然後即可使用 `duty()` 方法來指定 PWM 的輸出值：

```
>>> led.duty(1023) ← 設定 PWM 輸出值為 1023 (最亮)
>>> led.duty(0)    ← 設定 PWM 輸出值為 0 (最暗)
```

程式流程圖



程式設計

Lab07.py

```
01 from machine import Pin, PWM
02 import time
03
04 # 建立 15 號腳位的 PWM 物件，並命名為 led
05 led = PWM(Pin(15))
06
07 while True:
08     # range() 會產生 0 到 1024 但不包含 1024，間隔為 10 的數列
09     for i in range(0, 1024, 10):
10         led.duty(i)    # 設定 PWM 輸出值控制 LED 亮度
11         time.sleep(0.01)
```


實測

請按 **F5** 執行程式，即可看到 LED 由暗到亮逐漸亮起。

Lab08

LED 呼吸燈

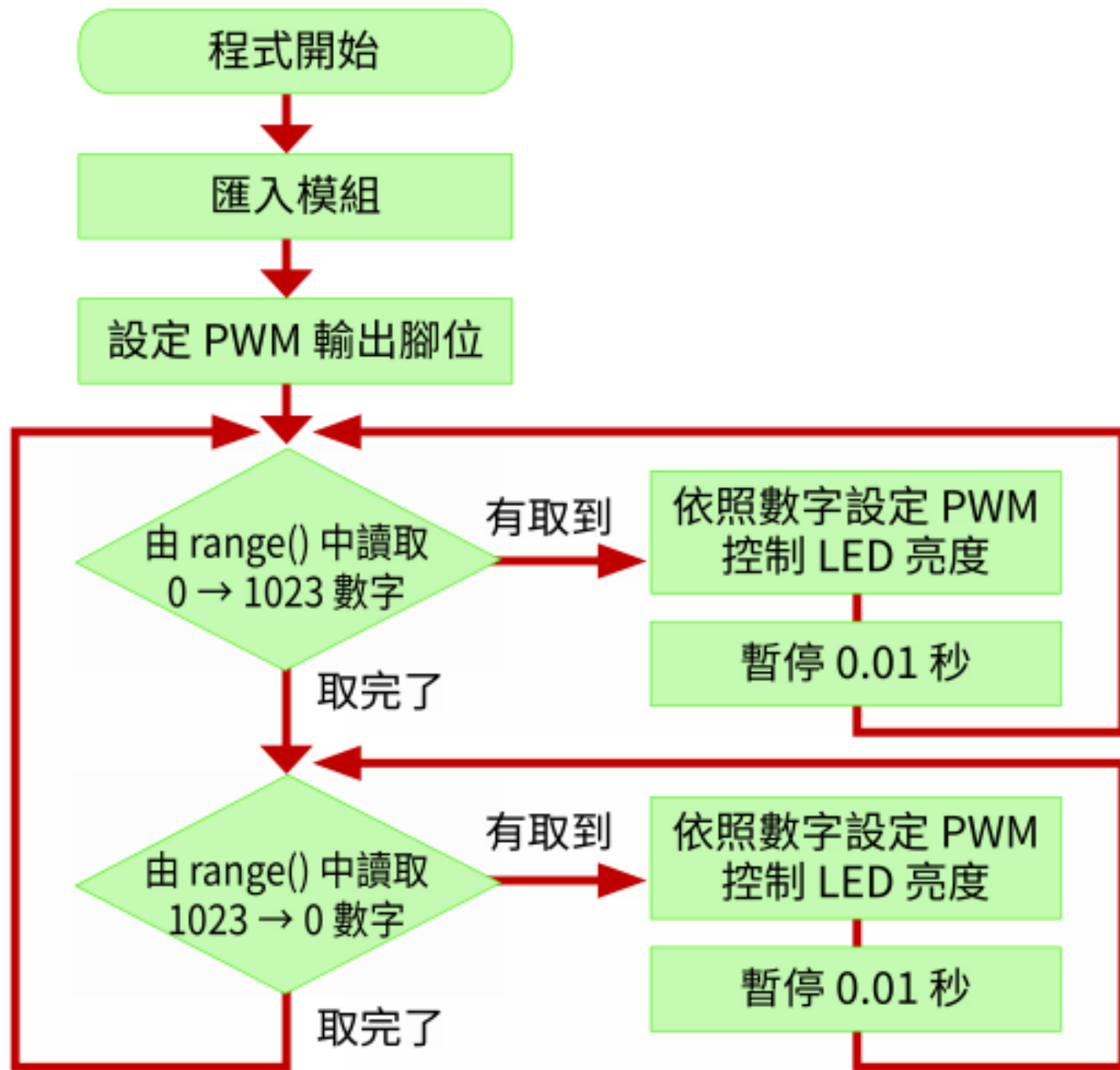
實驗目的

用 PWM 控制 LED 的亮度, 使其由暗至亮逐漸亮起, 然後由亮至暗逐漸熄滅, 產生呼吸的效果。。

材料

- D1 mini
- 紅色 LED
- 220 Ω 電阻

程式流程圖



程式設計

Lab08.py

```
01 from machine import Pin, PWM
02 import time
03
04 # 建立 15 號腳位的 PWM 物件, 並命名為 led
05 led = PWM(Pin(15))
06
07 while True:
08     # 漸亮
09     for i in range(0, 1024, 10): # 從 range() 中讀取 0→1023
10         led.duty(i)
11         time.sleep(0.01)
12
13     # 漸暗
14     for i in range(1023, -1, -10): # 從 range() 中讀取 1023→0
15         led.duty(i)
16         time.sleep(0.01)
```

實測

請按 **F5** 執行程式，即可看到 LED 由暗到亮逐漸亮起，然後由亮至暗逐漸熄滅。

07 霹靂車跑馬燈

Python 資料的容器：串列 (list)

可循序放置 / 取出資料的串列 (list)

在 Python 語言中，提供有一種特別的資料類型，叫做『**串列 (list)**』。串列就像一個容器，可以讓您隨意放置多項資料，這些資料稱為『**元素**』(element)，會依序排列放置，並且可以搭配 for 敘述循序取出個別元素。例如：

```
>>> leds = [16, 14, 12, 13, 15, 5, 4]
>>> for i in leds:
    print(i)
```

```
16
14
12
13
15
5
4
```

Lab09

單向 LED 跑馬燈

實驗目的

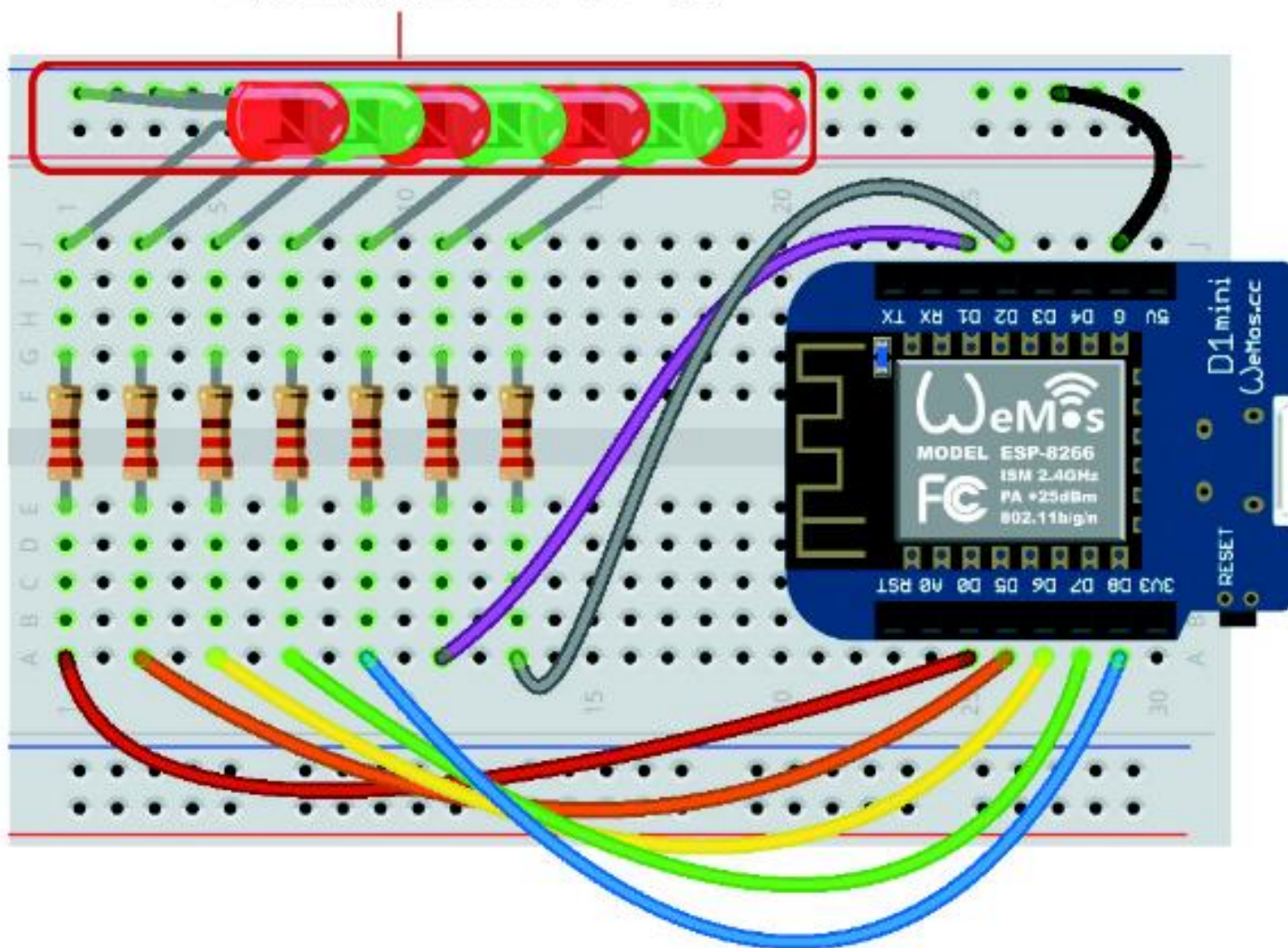
製作 7 顆 LED 的跑馬燈, 讓每顆 LED 輪流點亮、熄滅, 不斷重複。

材料

- D1 mini
- LED (顏色不拘) × 7
- 220Ω 電阻 × 7
- 杜邦線與排針若干

請依線路圖接線

LED 短腳全部接藍色的 - 這一排



```
from machine import Pin
import time

while True:
    led = Pin(16, Pin.OUT)
    led.value(1)
    time.sleep(0.1)
    led.value(0)
    led = Pin(14, Pin.OUT)
    led.value(1)
    time.sleep(0.1)
    led.value(0)
    ... ..
    led = Pin(4, Pin.OUT)
    led.value(1)
    time.sleep(0.1)
    led.value(0)
```

程式設計

Lab09.py

```
01 from machine import Pin
02 import time
03
04 # 建立串列, 依序儲存 D0、D5、D6、D7、D8、D1、D2 腳位編號
05 leds = [16, 14, 12, 13, 15, 5, 4]
06
07 while True:                                # 重複跑馬燈效果
08     for i in leds:                          # 依序取出個別腳位編號
09         led = Pin(i, Pin.OUT)              # 設定當前腳位為輸出功能
10         led.value(1)                       # 點亮對應的 LED
11         time.sleep(0.1)                   # 等待 0.1 秒
12         led.value(0)                       # 熄滅剛剛點亮的 LED
```

Lab 10

雙向 LED 跑馬燈

實驗目的

製作 7 顆 LED 的雙向跑馬燈，讓每顆 LED 輪流點亮、熄滅後再反向輪流——點亮、熄滅，不斷重複前述過程。

材料

- D1 mini
- 紅色 LED × 7
- 220Ω 電阻 × 7
- 杜邦線與排針若干

■ 設計原理

串列可以進行多種操作，例如使用 `reversed()` 函式可讓您從指定的串列中以相反的順序一一取出個別元素：

```
>>> leds = [16, 14, 12, 13, 15, 5, 4]
>>> for i in reversed(leds):
    print(i)

4
5
15
13
12
14
16
```

程式設計

```
01 from machine import Pin
02 import time
03
04 # 建立串列, 依序儲存 D0、D5、D6、D7、D8、D1、D2 腳位編號
05 leds = [16, 14, 12, 13, 15, 5, 4]
06
07 while True:                                # 重複雙向跑馬燈效果
08     for i in leds:                            # 依序取出個別腳位編號
09         led = Pin(i, Pin.OUT)                # 設定當前腳位為輸出功能
10         led.value(1)                          # 點亮對應的 LED
11         time.sleep(0.05)                      # 等待 0.05 秒
12         led.value(0)                          # 熄滅剛剛點亮的 LED
13
14     for i in reversed(leds):                  # 反方向依序取出個別腳位編號
15         led = Pin(i, Pin.OUT)                # 設定當前腳位為輸出功能
16         led.value(1)                          # 點亮對應的 LED
17         time.sleep(0.05)                      # 等待 0.05 秒
18         led.value(0)                          # 熄滅剛剛點亮的 LED
```

08 電子鋼琴

Python 資料的容器：字典 (dictionary)

認識蜂鳴器

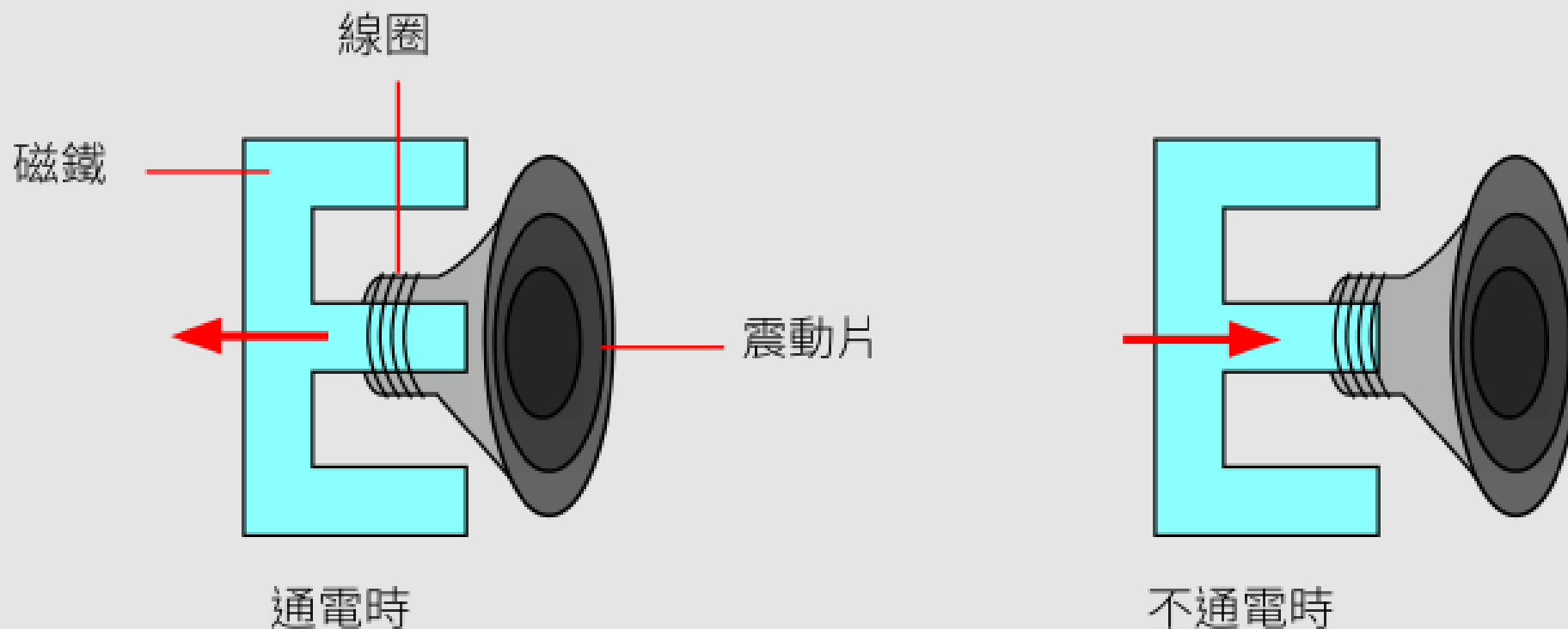
蜂鳴器是一種可讓內部銅片依據不同頻率震動發出聲音的電子元件：



蜂鳴器

認識蜂鳴器

利用電磁原理，即可吸附或是鬆開內部的震動片，造成震動：



用蜂鳴器發出音樂

- 只要使用第 6 章說明過的 PWM 類比輸出功能, 就可以控制蜂鳴器震動的頻率
- PWM 負載率 (duty cycle) 則可以控制震動的幅度, 改變音量的大小

我們日常聽見的音樂, 就是由不同震動頻率的音符所組成, 常用音符與對應的頻率 (每秒次數, Hz) 及音名的對照表如下:

音名	C	D	E	F	G	A	B
音符	Do	Re	Mi	Fa	So	La	Si
頻率	262	294	330	349	392	440	494

Lab 11

嗡嗡翁--小蜜蜂音樂

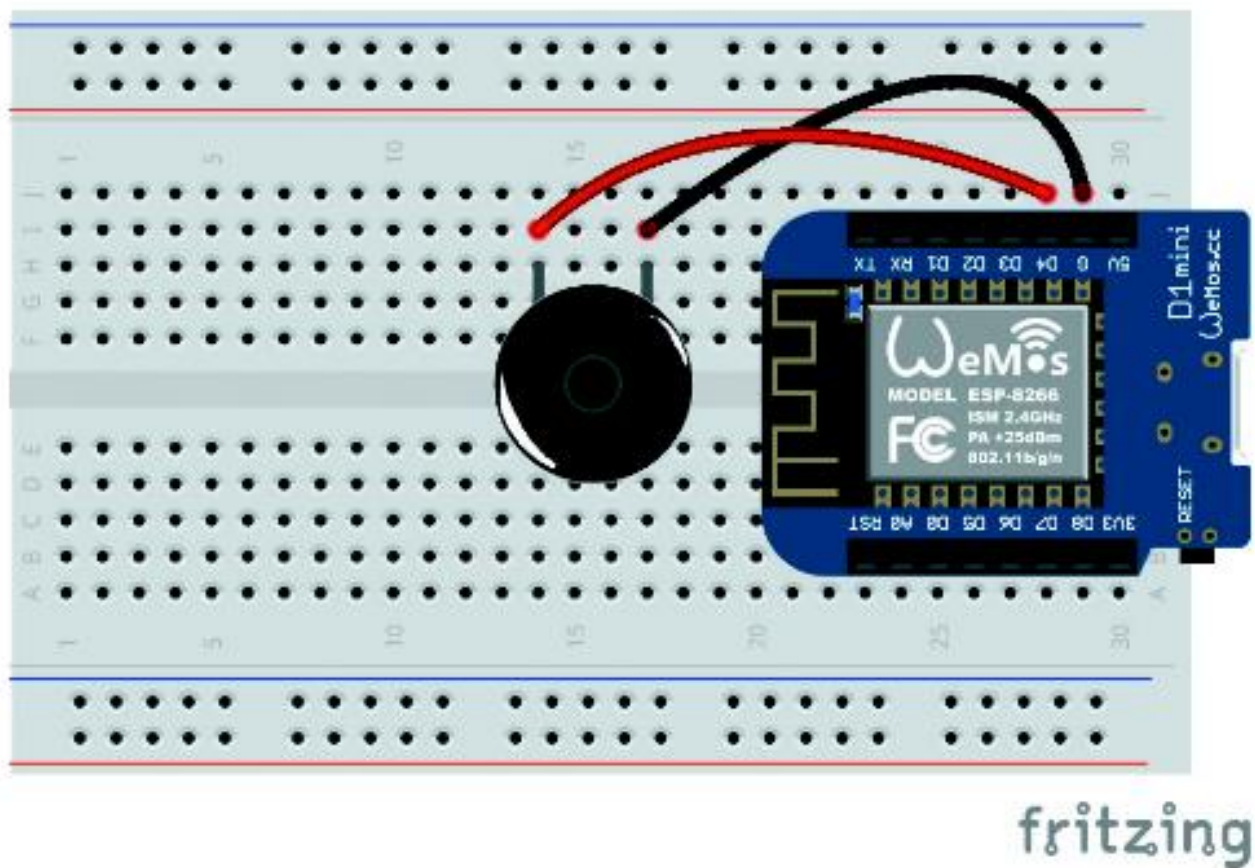
實驗目的

本實驗將利用控制蜂鳴器的震動頻率與震動時間，撰寫程式發出小蜜蜂這首歌的第一句，有興趣的讀者，也可以比照本實驗的作法，完成整首歌。

材料

- D1 mini
- 蜂鳴器
- 杜邦線及排針若干

請依線路圖接線



- ⚠ 蜂鳴器的 2 隻腳不像 LED 有區分長短腳，不需分辨方向，這 2 隻腳可以直接插入麵包板或是用杜邦線串接到控制板。

■ 設計原理

小蜜蜂歌曲的第 1 句簡譜如下：

```
| 5 3 3 - | 4 2 2 - | 1 2 3 4 5 5 5 - |
```

只要根據對應的頻率發聲即可，不發聲的地方將負載率 (duty cycle) 設為 0，蜂鳴器就會變靜音。要注意的是，每一個音符之間要加入不發聲的停頓，才能明顯聽出音符的變化，讓旋律富有韻律感。

程式設計

Lab8-1

Lab11.py

```
01 from machine import Pin, PWM
02 import time
03
04 beeper = PWM(Pin(2, Pin.OUT)) # 使用 2 號腳位控制蜂鳴器
05 # |5 3 3 -|4 2 2 -|1 2 3 4 5 5 5 -|
06 # 0 表示休止符
07 notes = [
08     392, 330, 330, 0,
09     349, 294, 294, 0,
10     262, 294, 330, 349, 392, 392, 392, 0]
11
12 for note in notes:           # 一一取出音符
13     if note == 0:           # 休止符不發音
14         beeper.duty(0)
15     else:
16         beeper.duty(512)    # 設定為一半音量
17         beeper.freq(note)  # 依照音符設定頻率
18         time.sleep(0.2)     # 讓聲音持續 02 秒
19         beeper.duty(0)     # 停止發聲
20         time.sleep(0.1)    # 持續無聲 01 秒
```

Python 資料結構: 字典 (dictionary)

在前一節的範例中，是直接以音符的頻率來表達旋律，這種方式撰寫起來很麻煩，而且很容易寫錯，只要頻率錯了，就無法正確發出聲音，讀程式的人也不容易看懂旋律。如果有簡便的方式用音符或音名幫對應的頻率命名，就可以直接用名稱來表示旋律，不但不易出錯，也更容易看懂。

在 Python 中提供有一種特別的容器，稱為『**字典 (dictionary)**』，和上一章的串列有點類似，可以隨意放置多項元素，不過每一個元素都由**名稱**（稱為『**鍵 (key)**』）與**值 (value)** 組成，要取出值時，都必須指定元素名稱（鍵）才能取出對應的值，例如：

```
>>> ages = { "Mary":13, "John":14 }
```

Python 資料結構: 字典 (dictionary)

要取出資料，必須指定字典名稱，搭配以中括號包夾的鍵，例如：

```
>>> ages["Mary"]  
13  
>>> ages["John"]  
14
```

就可以分別取出字典中名稱為 "Mary" 或是 "John" 的值。

Lab 12

電子鋼琴

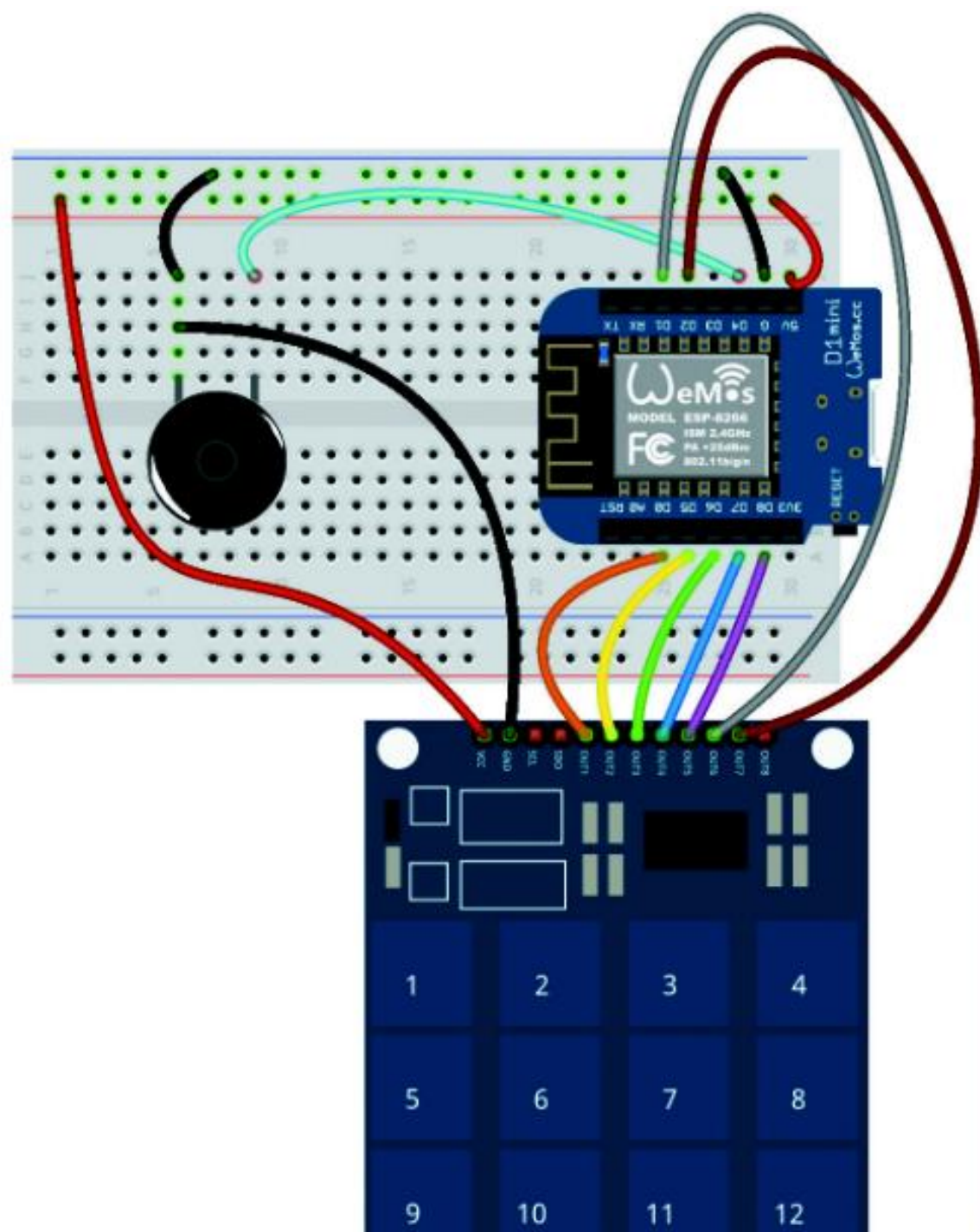
實驗目的

使用觸控開關當成琴鍵，讓使用者可彈出 Do~Si 各音符組成的旋律。

材料

- Di mini
- 蜂鳴器
- 電容式觸控開關
- 杜邦線及排針若干

線路圖



觸控開關	D1 mini
OUT1	D0
OUT2	D5
OUT3	D6
OUT4	D7
OUT5	D8
OUT6	D1
OUT7	D2
GND	G
VCC	3V3

■ 設計原理

此範例使用字典儲存音名與對應的頻率：

```
tones = {  
    'c': 262,  
    'd': 294,  
    'e': 330,  
    'f': 349,  
    'g': 392,  
    'a': 440,  
    'b': 494,  
}
```

程式會在使用者觸碰按鈕時，從 tones 字典中取出對應音名的頻率讓蜂鳴器發聲，即可達到觸碰彈奏音樂的目的。

■ 程式設計

```
01 from machine import Pin, PWM
02 import time
03
04 beeper = PWM(Pin(2, Pin.OUT)) # 用 2 號腳位控制蜂鳴器
05 button1 = Pin(16, Pin.IN) # 用 D0 讀 OUT1
06 button2 = Pin(14, Pin.IN) # 用 D5 讀 OUT2
07 button3 = Pin(12, Pin.IN) # 用 D6 讀 OUT3
08 button4 = Pin(13, Pin.IN) # 用 D7 讀 OUT4
09 button5 = Pin(15, Pin.IN) # 用 D8 讀 OUT5
10 button6 = Pin(5, Pin.IN) # 用 D1 讀 OUT6
11 button7 = Pin(4, Pin.IN) # 用 D2 讀 OUT7
12
13 tones = { # 儲存音名與頻率的字典
14     'c': 262, # Do
15     'd': 294, # Re
16     'e': 330, # Mi
17     'f': 349, # Fa
18     'g': 392, # So
```

```
19     'a': 440,           # La
20     'b': 494,           # Si
21 }
22
23 while True:             # 持續讀取觸控按鈕訊號
24     if button1.value() == 1: # 按了 1 號按鈕
25         beeper.duty(512)     # 設定一半音量
26         beeper.freq(tones['c']) # 設定 Do 的頻率
27
28     elif button2.value() == 1: # 按了 2 號按鈕
29         beeper.duty(512)     # 設定一半音量
30         beeper.freq(tones['d']) # 設定 Re 的頻率
31
32     elif button3.value() == 1: # 按了 3 號按鈕
33         beeper.duty(512)     # 設定一半音量
34         beeper.freq(tones['e']) # 設定 Mi 的頻率
```

```
35
36     elif button4.value() == 1:      # 按了 4 號按鈕
37         beeper.duty(512)           # 設定一半音量
38         beeper.freq(tones['f'])    # 設定 Fa 的頻率
39
40     elif button5.value() == 1:      # 按了 5 號按鈕
41         beeper.duty(512)           # 設定一半音量
42         beeper.freq(tones['g'])    # 設定 So 的頻率
43
44     elif button6.value() == 1:      # 按了 6 號按鈕
45         beeper.duty(512)           # 設定一半音量
46         beeper.freq(tones['a'])    # 設定 La 的頻率
47
48     elif button7.value() == 1:      # 按了 7 號按鈕
49         beeper.duty(512)           # 設定一半音量
50         beeper.freq(tones['b'])    # 設定 Si 的頻率
51
52     else:                            # 沒有按鈕
53         beeper.duty(0)             # 設定不發聲
54
55     time.sleep(0.05)
```

```
from machine import Pin, PWM
import time

beeper = PWM(Pin(2, Pin.OUT)) # 用 2 號腳位控制蜂鳴器
buttons = [
    ... ..
]

tones = [ # 儲存音名與頻率的串列
    ... ..
]

while True: # 持續讀取觸控按鈕訊號
    for i in range(len(buttons)): # 一一取出物件檢查按鈕，這裡函式 len() 為計算 buttons 的元素個數
        if buttons[i].value() == 1: # 如果有按鈕
            beeper.duty(512) # 設定一半音量
            beeper.freq(tones[i]) # 設定對應音符的音量
            break # 不需再檢查其他按鈕
        else: # 如果所有按鈕都沒按下
            beeper.duty(0) # 設定不發聲

time.sleep(0.05)
```

09 氣象預報站

Python 網路爬蟲

WiFi 連線

D1 mini 控制板除了可以控制外部裝置，或是讀取外部資訊外，也具備有無線網路的能力，可以連到網路上擷取資訊。要使用網路，首先必須匯入 **network** 模組，利用其中的 **WLAN** 類別建立控制無線網路的物件：

```
>>> import network
>>> sta_if = network.WLAN(network.STA_IF)
```

由於我們需要讓 D1 mini 連上網際網路擷取資訊，所以必須使用**工作站介面**。取得無線網路物件後，要先啟用網路介面：

```
>>> sta_if.active(True)
```

參數 True 表示要啟用網路介面；如果傳入 False 則會停用此介面。接著，就可以嘗試連上無線網路：

```
>>> sta_if.connect('無線網路名稱', '無線網路  
密碼')
```

取得網路資料

網路上有各式各樣的資訊，也有許多廠商提供公開的服務給大家取用，只要我們的程式能夠扮演瀏覽器的角色，就可以透過程式擷取網頁的內容，取出所需的資料。

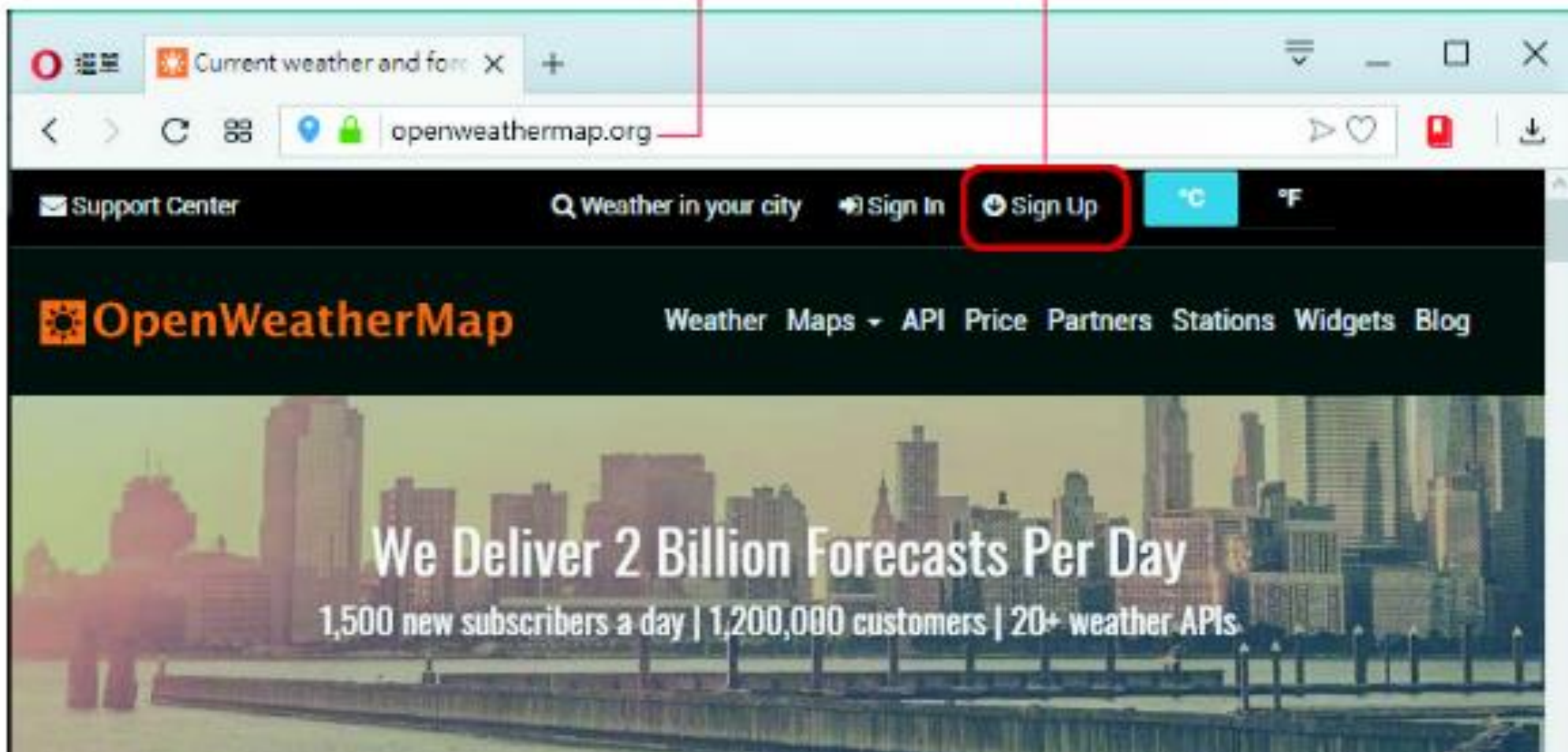
在 Python 中有個 requests 模組就可以提供瀏覽器到網路上抓取資料的功能，不過這個模組功能比較複雜，在 MicroPython 中提供的是精簡版的 urequests 模組，名稱開頭的 'u' 是 'micro' 的意思。只要匯入此模組，即可使用該模組提供的 get() 取得網站傳回的資料：

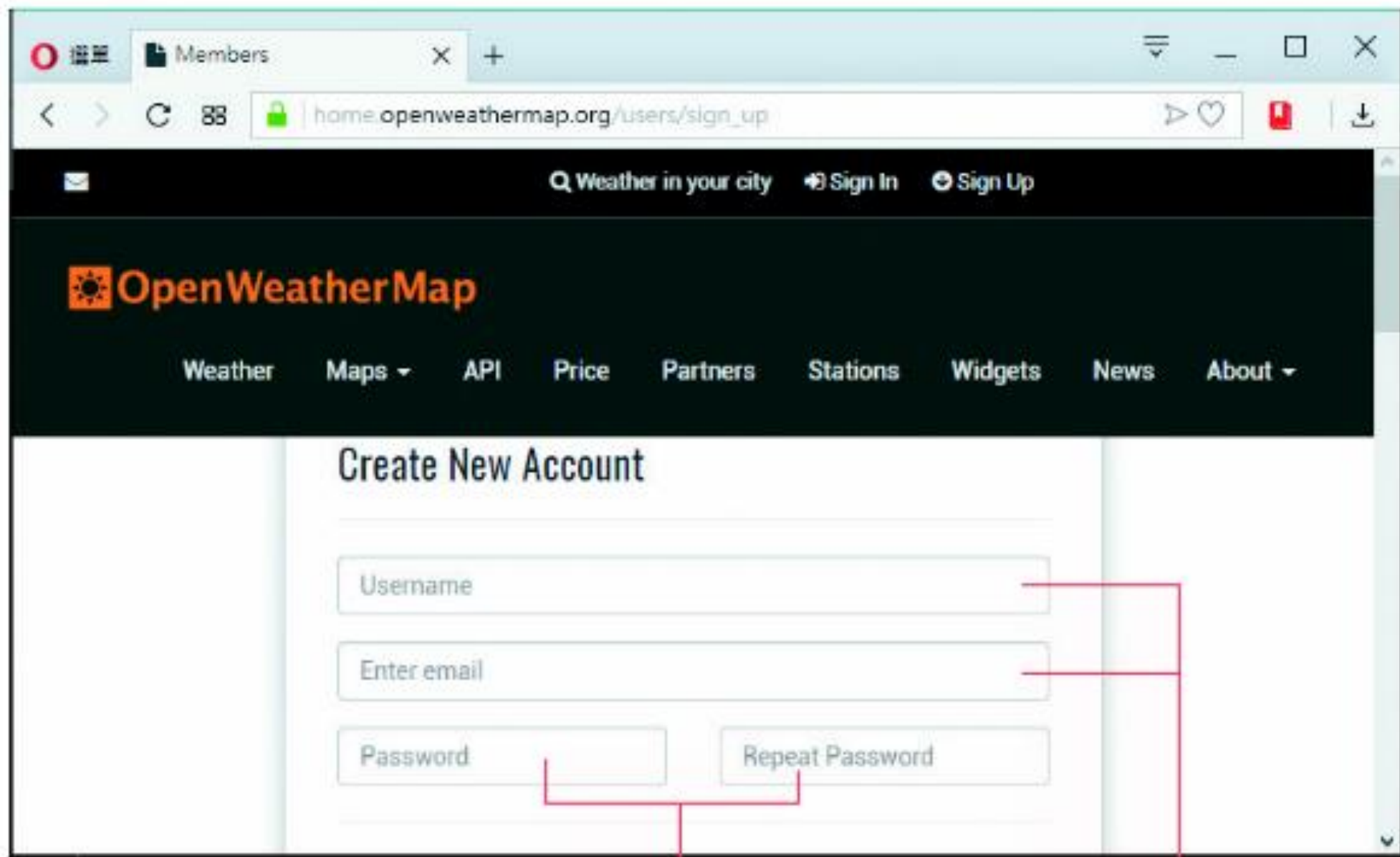
```
>>> res = urequests.get('http://www.flag.com.tw')
>>> print(res.text)
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-
```

取得 OpenWeatherMap 天氣資料

1 輸入網址 `https://openweathermap.org`

2 點選 Sign Up





4 填入要設定的密碼

3 填入帳戶名稱和電子郵件位址

Members

home.openweathermap.org/users/sign_up

OpenWeatherMap

Weather Maps API Price Partners Stations Widgets News About

We will use information you provided for management and administration purposes, and for keeping you informed by mail, telephone, email and SMS of other products and services from us and our partners. You can proactively manage your preferences or opt-out of communications with us at any time using Privacy Centre. You have the right to access your data held by us or to request your data to be deleted. For full details please see the OpenWeather [Privacy Policy](#).

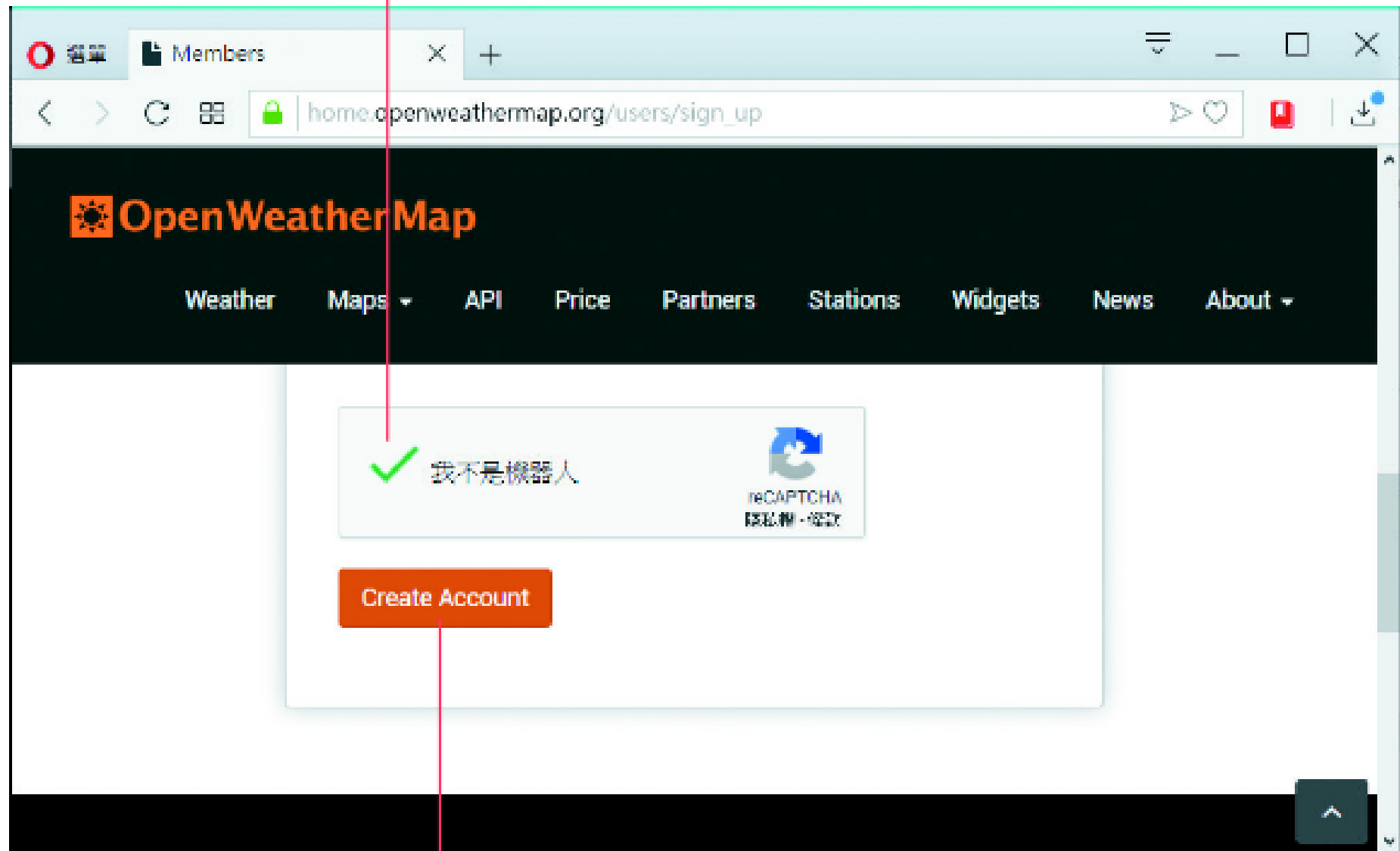
I am 16 years old and over

I agree with [Privacy Policy](#), [Terms and conditions of sale](#) and [Websites terms and conditions of use](#)

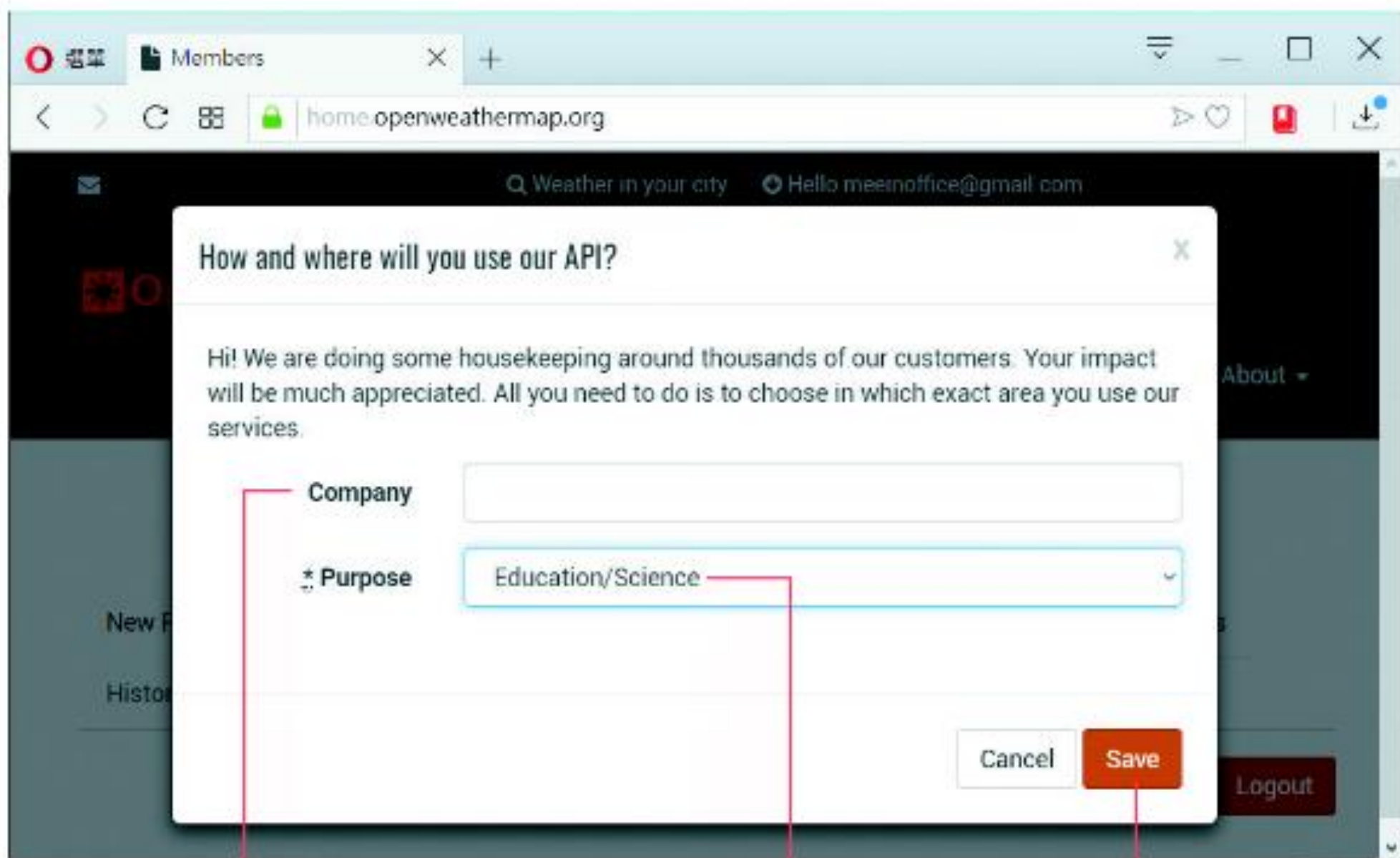
5 往下捲後
勾選確認年齡

6 勾選同意隱私權協議

7 勾選確認不是機器人程式



8 最後按下 Create Account



9 填入公司名稱

10 選取用途

11 按 **Save**

2 等待帳戶生效，會收到一封電子郵件：



1 從 OWM Team 寄來
通知帳號生效的信件

2 點選查看信件內容

Thank you for subscribing to OpenWeather API!

Dear Customer!

Thank you for subscribing to Free [OpenWeather API!](#)

API key:

- Your API key is **2fdb27801a15ec274616f7838e77ea42**
- Within the next couple of hours, it will be activated and ready to use
- You can later create more API keys on your [account page](#)
- Please, always use your API key in each API call

Endpoint:

- Please, use the endpoint [api.openweathermap.org](#) for your API calls
- Example of API call:

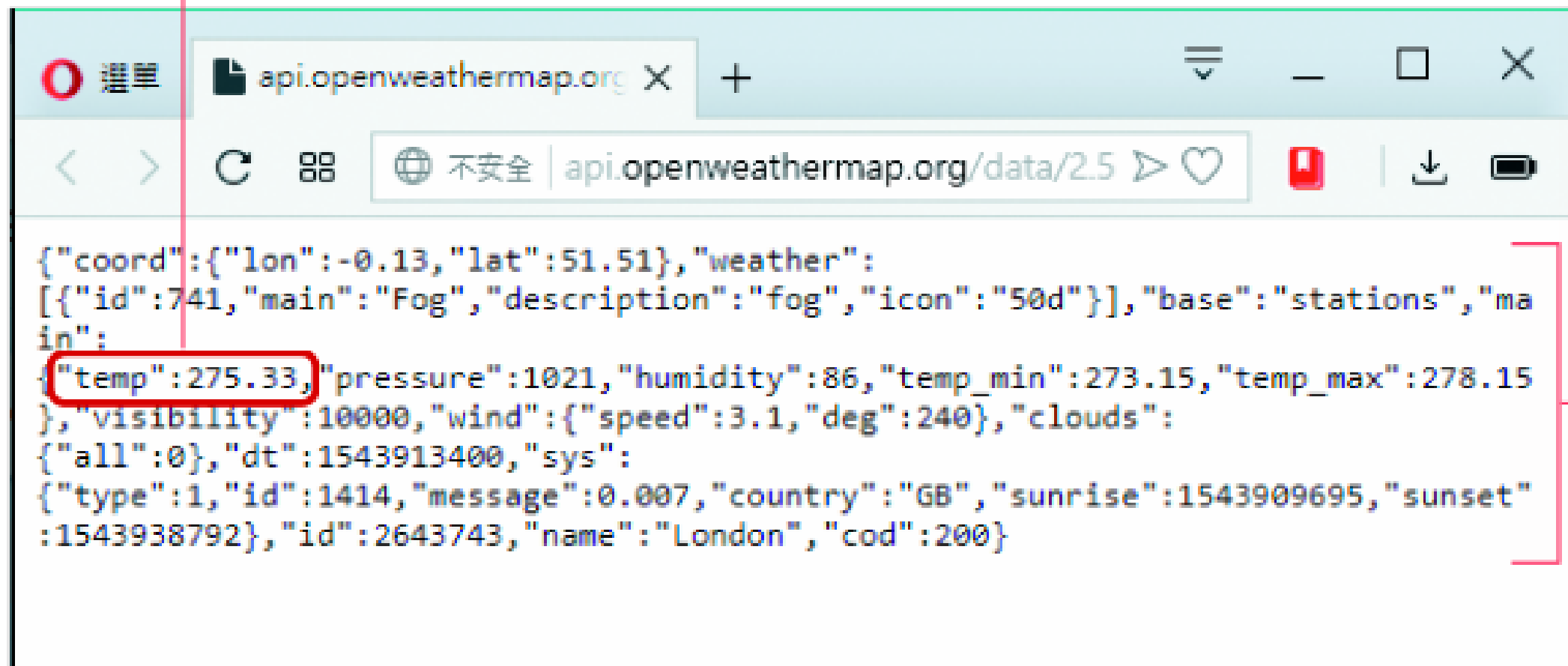
[api.openweathermap.org/data/2.5/weather?q=London,uk&APPID=2fdb27801a15ec274616f7838e77ea42](#)

3 這是你的 API key (金鑰), 代表你的個人身份, 讀取天氣資訊時需要提供此金鑰

4 此為取得天氣資訊的範例, 點選可取得倫敦的目前天氣

6 這裡是溫度，單位是絕對溫度 K，
275.33K 就是攝氏 2.18 度 (倫敦好冷)

5 點選範例後取得的倫敦目前天氣



```
["coord":{"lon":-0.13,"lat":51.51},"weather":  
[{"id":741,"main":"Fog","description":"fog","icon":"50d"}],"base":"stations","ma  
in":  
{"temp":275.33,"pressure":1021,"humidity":86,"temp_min":273.15,"temp_max":278.15  
},"visibility":10000,"wind":{"speed":3.1,"deg":240},"clouds":  
{"all":0},"dt":1543913400,"sys":  
{"type":1,"id":1414,"message":0.007,"country":"GB","sunrise":1543909695,"sunset"  
:1543938792},"id":2643743,"name":"London","cod":200}
```

⚠ 攝氏溫度 = 絕對溫度 - 273.15

3 有關提供個別資訊的網址格式，可參考 <http://api.openweathermap.org/>：

The screenshot shows the OpenWeatherMap API page. On the left, there are three vertical labels: '16 天預測', '5 天預測', and '目前天氣'. A red line connects '16 天預測' to the '16 day / daily forecast' section. Another red line connects '5 天預測' to the '5 day / 3 hour forecast' section. A third red line connects '目前天氣' to the 'Current weather data' section. At the bottom left, there is a red box with the number '1' and the text '點選資訊類別的 API doc'. A red line connects this box to the 'API doc' buttons under each of the three forecast sections.

16 天預測

5 天預測

目前天氣

1 點選資訊類別的 API doc

Support Center Weather in your city Sign In Sign Up °C °F

OpenWeatherMap Weather Maps API Price Partners Stations Widgets Blog

Weather API [Home / Weather API](#)

Please [sign up](#) and use our fast and easy-to-work weather APIs for free. Look at our [monthly subscriptions](#) for more options than Free account can provide you. Read [How to start](#) first and enjoy using our powerful weather APIs.

Current weather data	5 day / 3 hour forecast	16 day / daily forecast
API doc Subscribe	API doc Subscribe	API doc Subscribe
<ul style="list-style-type: none">• Access current weather data for any location including over 200,000 cities• Current weather is frequently updated based on global models and data from more than 40,000 weather stations	<ul style="list-style-type: none">• 5 day forecast is available at any location or city• 5 day forecast includes weather data every 3 hours• Forecast is available in JSON and XML• Available for Free and all other paid accounts	<ul style="list-style-type: none">• 16 day forecast is available at any location or city• 16 day forecast includes daily weather• Forecast is available in JSON and XML• Available for all paid accounts

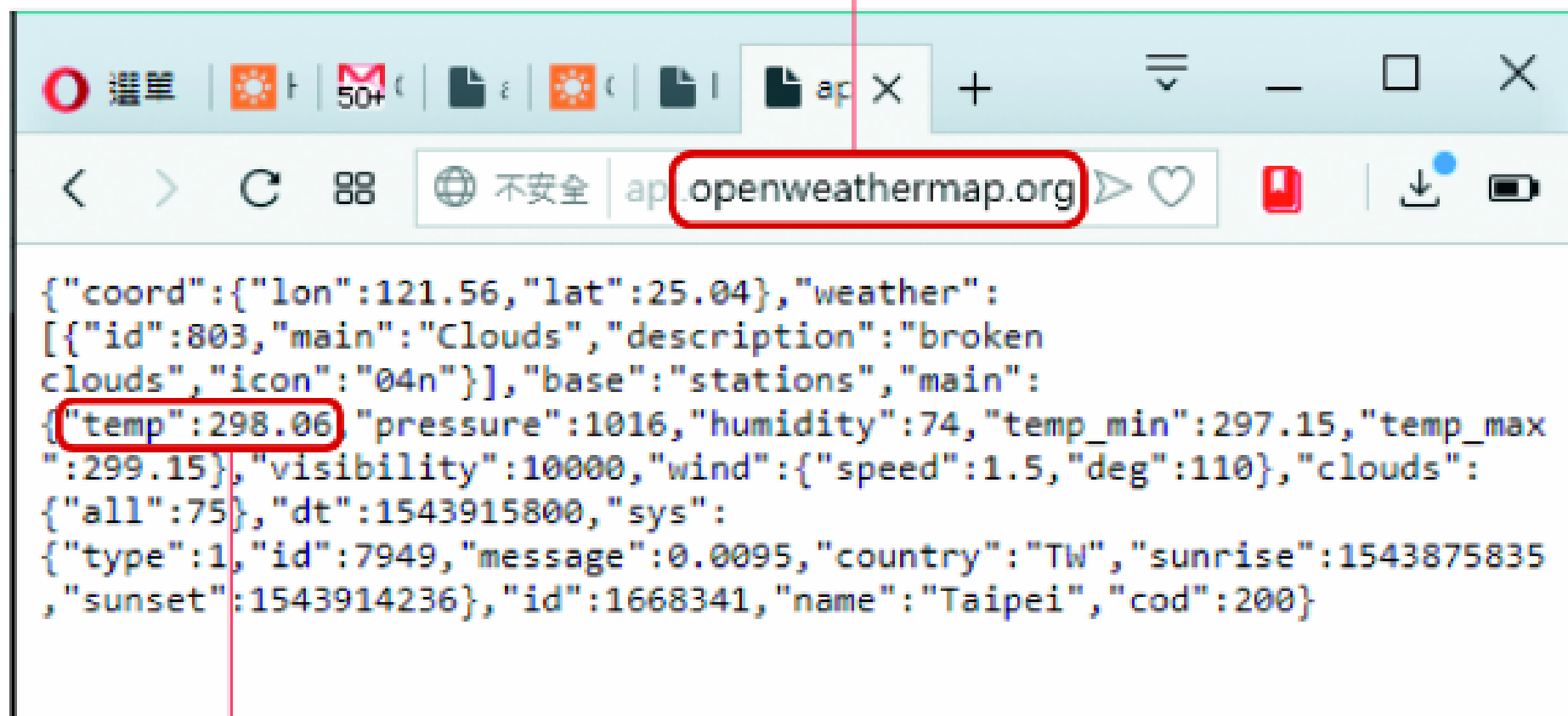
■ 讀取天氣資訊

這裡我們以取得目前天氣狀況 (Get Current Data) 為例，說明如何使用 Python 程式取得 OpenWeatherMap 的資訊。假設我們想要取得台北市的天氣，依據網站上文件的說明，格式如下：

```
http://api.openweathermap.org/data/2.5/weather?q={城市名稱},  
{國別}&appid={API key}
```

台北市的 { 城市名稱 } 為 "Taipei", { 國別 } 為 "TW", {API key} 在剛剛收到的電子郵件中就可以找到，將以上套入後輸入到瀏覽器的網址列即可得到台北的目前天氣：

輸入 `http://api.openweathermap.org/data/2.5/
weather?q=Taipei,TW&appid=你的 API key`



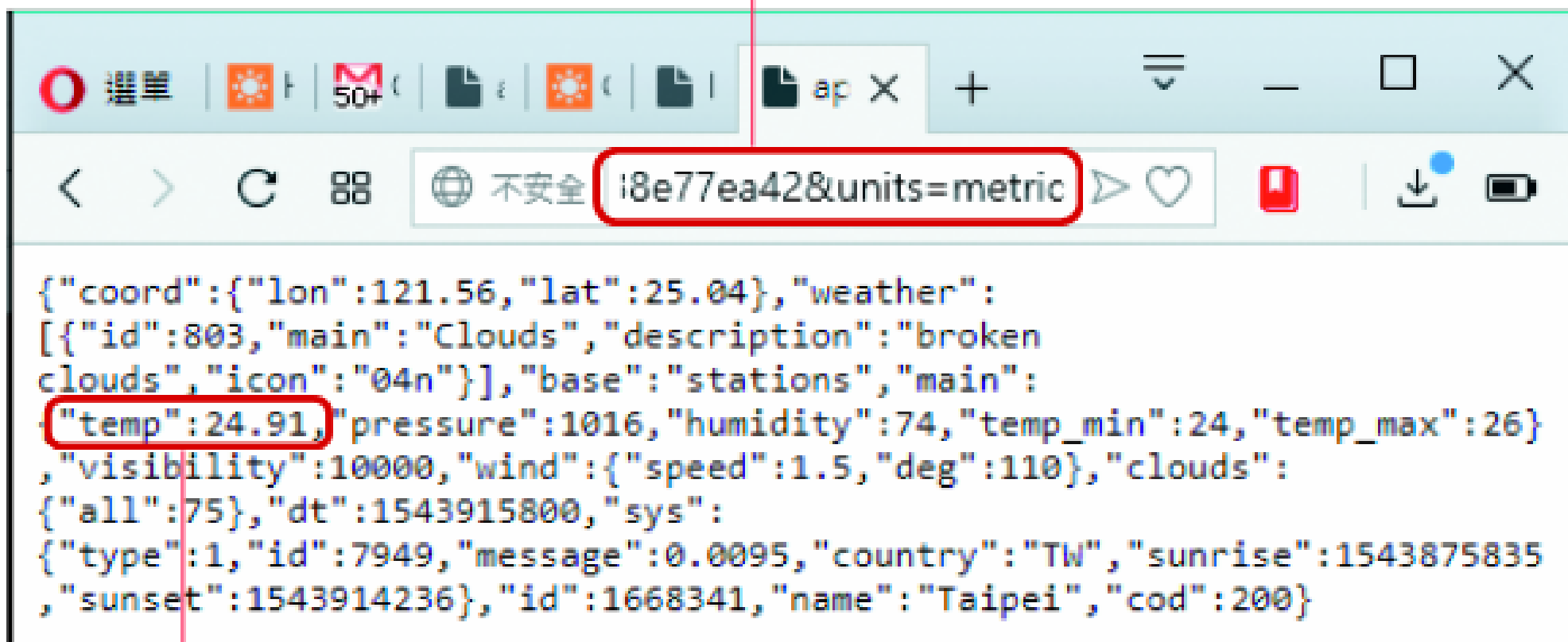
```
{
  "coord": {
    "lon": 121.56,
    "lat": 25.04
  },
  "weather": [
    {
      "id": 803,
      "main": "Clouds",
      "description": "broken clouds",
      "icon": "04n"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 298.06,
    "pressure": 1016,
    "humidity": 74,
    "temp_min": 297.15,
    "temp_max": 299.15
  },
  "visibility": 10000,
  "wind": {
    "speed": 1.5,
    "deg": 110
  },
  "clouds": {
    "all": 75
  },
  "dt": 1543915800,
  "sys": {
    "type": 1,
    "id": 7949,
    "message": 0.0095,
    "country": "TW",
    "sunrise": 1543875835,
    "sunset": 1543914236
  },
  "id": 1668341,
  "name": "Taipei",
  "cod": 200
}
```

這裡是台北的溫度

你可以看到溫度是絕對溫度 298.06K, 也就是攝氏 24.91 度。

您也可以在剛剛的網址後面加上 "&units=metric"，即可指定使用攝氏單位：

輸入 `http://api.openweathermap.org/data/2.5/
weather?q=Taipei,TW&appid=你的 API key&units=metric`



溫度改成攝氏單位了

你可以在 <http://bulk.openweathermap.org/sample/city.list.json.gz> 取得所有城市名稱與國別的資料，這是一份壓縮過的文字檔，可使用 7-Zip 或是 Winzip 等工具解開，即可用記事本打開。以下列出幾個主要城市的名稱：

城市名稱	中文
Keelung	基隆
Banqiao	板橋
Taoyuan	桃園
Hsinchu	新竹
Miaoli	苗栗
Taichung	台中
Nantou	南投
Yunlin	雲林

城市名稱	中文
Tainan	台南
Kaohsiung	高雄
Pingtung	屏東
Yilan	宜蘭
Hualien	花蓮
Taitung	台東
Penghu	澎湖
Hengchun	恆春

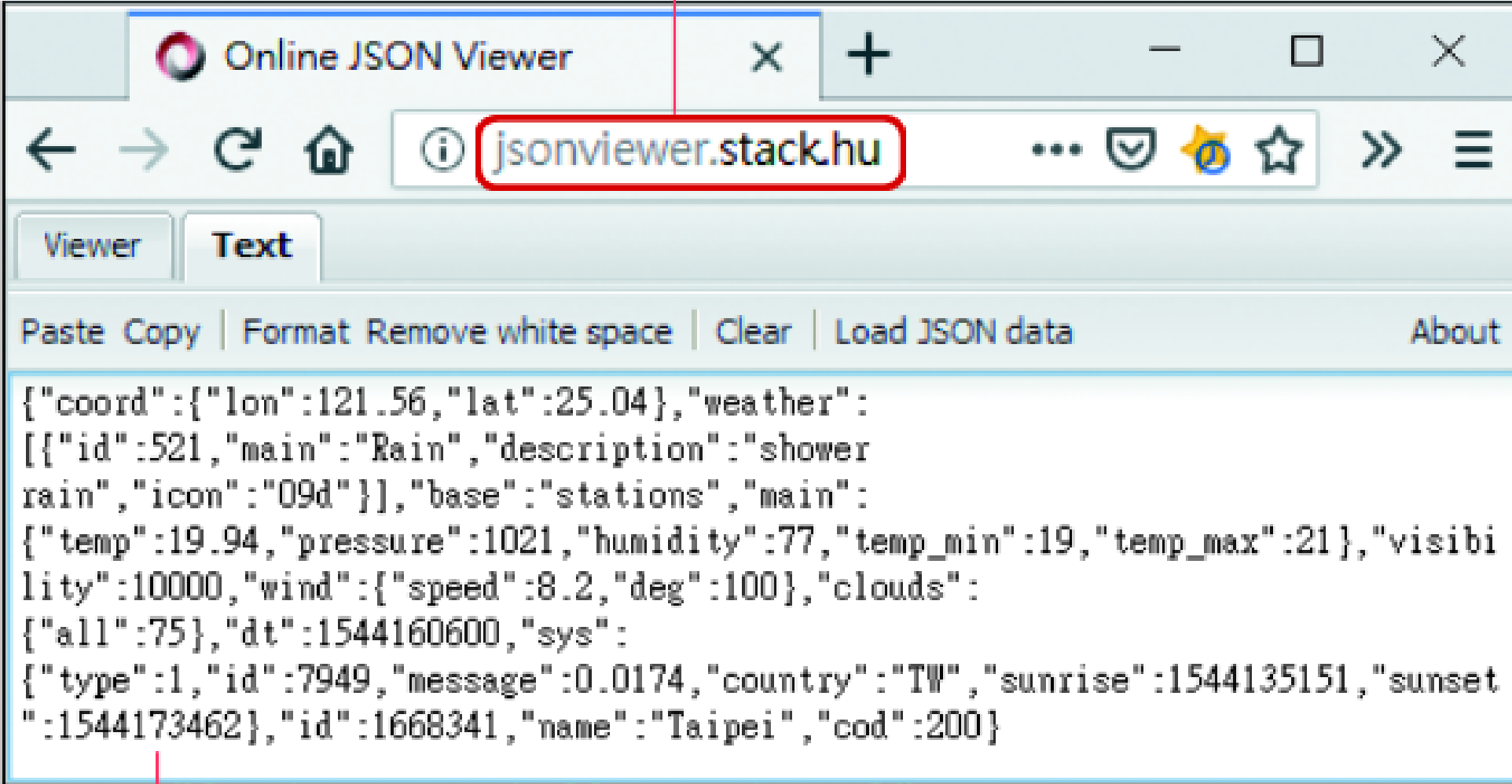
將上述網址與 9-1 節結合，就可以利用 Python 程式取得台北市的天氣狀況：

```
>>> import urequests
>>> res = urequests.get('http://api.openweathermap.
org/data/2.5/weather?q=Taipei, TW&appid=你的 API
key&units=metric')
>>> print(res.text)
{"coord":{"lon":121.56, "lat":25.04}, "weather":[{"id":802,
"main":"Clouds", "description":"scattered clouds",
"icon":"03n"}], "base":"stations", "main":{"temp":23.47,
"pressure":1018, "humidity":83, "temp_min":23, "temp_
max":24}, "visibility":10000, "wind":{"speed":5.7,
"deg":90}, "clouds":{"all":40}, "dt":1543923000,
"sys":{"type":1, "id":7949, "message":0.0048, "country":"TW",
"sunrise":1543875837, "sunset":1543914236}, "id":1668341,
"name":"Taipei", "cod":200}
>>>
```

JSON 資料格式解析

上一節從 OpenWeatherMap 取得的資料其實不只有溫度，還包含有其他豐富的資料，為了能夠適當呈現個別資料，它使用名為 JSON 的文字格式。JSON 的全名是 **JavaScript Object Notation**，原本是 JavaScript 程式語言中以文字形式描述物件內容的格式，由於簡單易用，現在變成呈現多層結構資料的常見格式。

1 網址 <http://jsonviewer.stack.hu/>

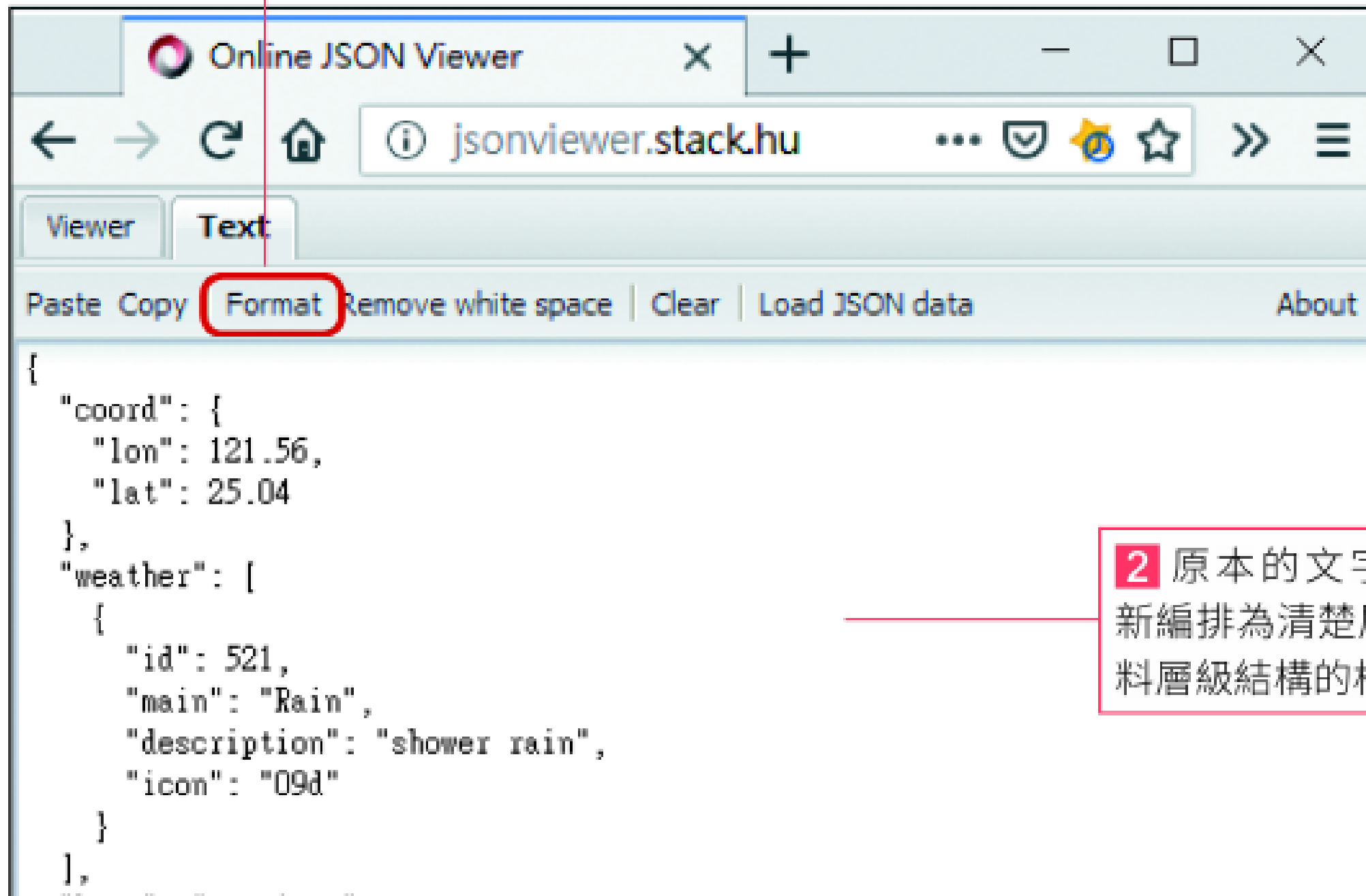


The screenshot shows a web browser window titled "Online JSON Viewer" with the URL "jsonviewer.stack.hu" in the address bar. The page has a "Text" tab selected and a menu with options: "Paste", "Copy", "Format", "Remove white space", "Clear", "Load JSON data", and "About". The main content area displays a JSON object representing weather data for Taipei.

```
{
  "coord": {
    "lon": 121.56,
    "lat": 25.04
  },
  "weather": [
    {
      "id": 521,
      "main": "Rain",
      "description": "shower rain",
      "icon": "09d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 19.94,
    "pressure": 1021,
    "humidity": 77,
    "temp_min": 19,
    "temp_max": 21
  },
  "visibility": 10000,
  "wind": {
    "speed": 8.2,
    "deg": 100
  },
  "clouds": {
    "all": 75
  },
  "dt": 1544160600,
  "sys": {
    "type": 1,
    "id": 7949,
    "message": 0.0174,
    "country": "TW",
    "sunrise": 1544135151,
    "sunset": 1544173462
  },
  "id": 1668341,
  "name": "Taipei",
  "cod": 200
}
```

2 在這裡貼上從 OpenWeatherMap 取得的 JSON 格式資料

1 點選 format



The screenshot shows a web browser window with the title 'Online JSON Viewer'. The address bar displays 'jsonviewer.stack.hu'. Below the browser window, there are two tabs: 'Viewer' and 'Text'. The 'Text' tab is active. A toolbar contains several buttons: 'Paste', 'Copy', 'Format', 'Remove white space', 'Clear', 'Load JSON data', and 'About'. The 'Format' button is highlighted with a red rectangular box. A red line points from the number '1' in the top-left corner to this 'Format' button. Below the toolbar, a JSON object is displayed in a text area. The JSON is partially visible and shows a structure with 'coord' and 'weather' properties. The 'weather' property is an array containing one object with 'id', 'main', 'description', and 'icon' fields.

```
{  
  "coord": {  
    "lon": 121.56,  
    "lat": 25.04  
  },  
  "weather": [  
    {  
      "id": 521,  
      "main": "Rain",  
      "description": "shower rain",  
      "icon": "09d"  
    }  
  ],  
}
```

2 原本的文字會重新編排為清楚展現資料層級結構的樣貌

⚠️ JSON 與 Python 字典的語法類似，但是 JSON 中字串必須以英文雙引號 "" 括起來，不能使用英文單引號 ' '。

整個資料就是一個字典，包含以下元素：

鍵	值
coord	城市經緯度
weather	晴天、多雲等天氣狀況
base	內部資料
main	溫濕度等天氣資訊
visibility	能見度
wind	風向風速
clouds	雲量
dt	資料回報時間
sys	國別及日出日落時間等
id	城市代碼
name	城市名稱
cod	內部資料

而鍵為 "weather" 的元素它的值則是一個串列：

```
"weather": [  
  {  
    "id": 803,  
    "main": "Clouds",  
    "description": "多雲",  
    "icon": "04d"  
  }  
]
```

在此例中這個串列裡面只有一個元素，這個元素又是一個字典，稍後的實驗中我們會使用其中鍵為 "id" 與 "description" 的 2 個元素，分別代表各種天氣狀況的代碼與說明文字：

代碼	意義
2XX	大雷雨
3XX	毛毛雨
5XX	下雨
6XX	下雪
7XX	霧、霾等狀況
800	晴天
8XX	陰天

 個別天氣代碼詳細意義可參考 <https://openweathermap.org/weather-conditions>。

如果以 800 為分界，就可以將小於 800 的天氣視為不佳，800 (含) 以上視為好天氣。

■ 使用 Python 解讀 JSON 資料

Python 中提供有 json 模組可以解析 JSON 格式，從文字形式轉換成 Python 的字典。MicroPython 則提供精簡版本的 ujson 模組，使用方法非常簡單，以下假設 res 是使用 urequests.get() 從 OpenWeatherMap 取回的結果：

```
>>> import ujson # 匯入 ujson 模組
>>> j = ujson.loads(res.text) # 載入 JSON 格式資料
>>> j["main"]["temp"] # 依照結構透過字典取得資料
19.94
>>> j["weather"][0]["id"] # 依結構循字典、串列取得資料
521
```

][

```
{
  ...
  "weather": [
    {
      "id": 521,
      "main": "Rain",
      ...
    }
  ],
  "base": "stations",
  "main": {
    "temp": 19.94,
    "pressure": 1021,
    ...
  },
  ...
}
```

→ j
→ j["weather"]
→ j["weather"][0]
→ j["weather"][0]["id"]
→ j["main"]
→ j["main"]["temp"]

Lab 13

氣象預報站

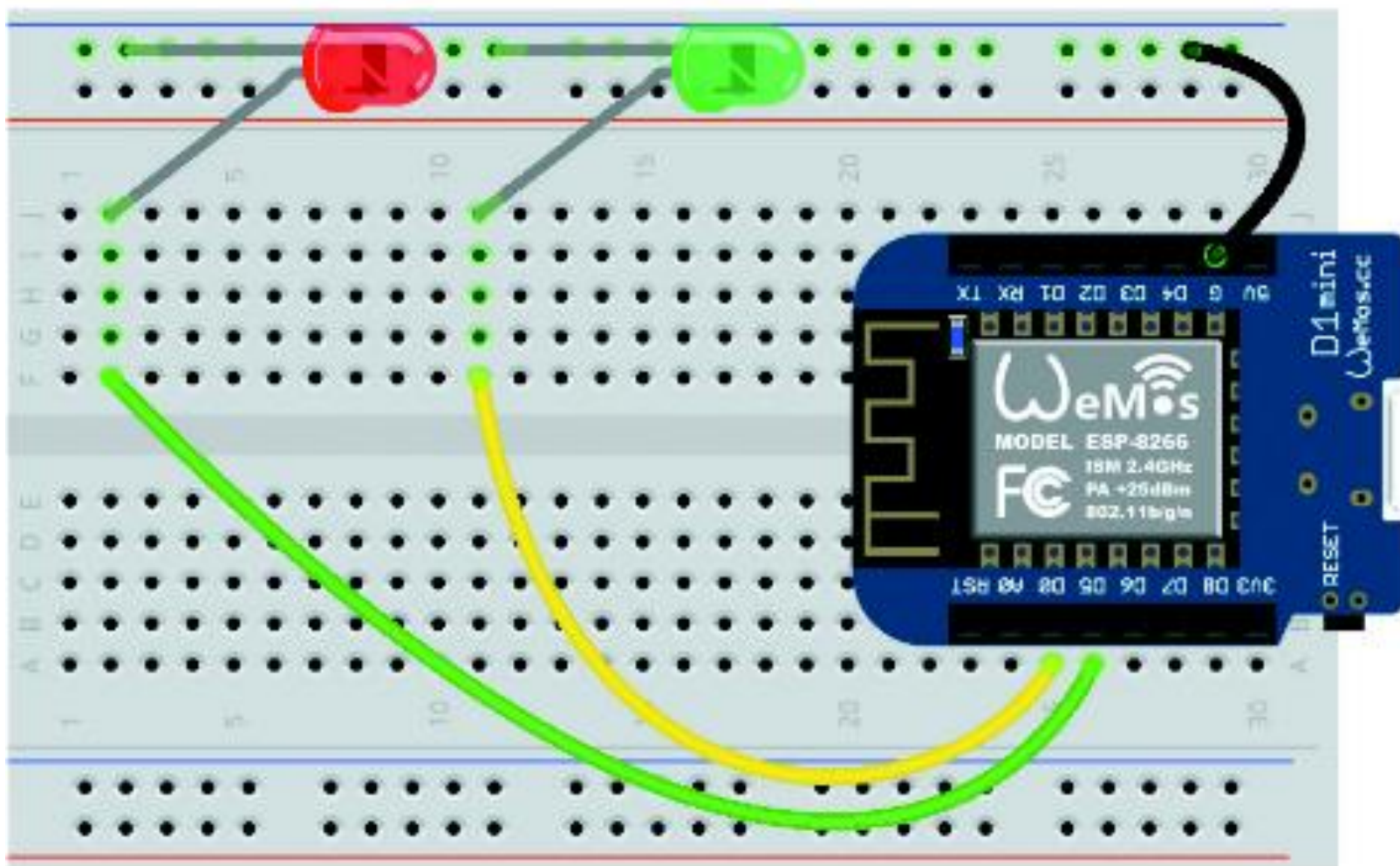
實驗目的

利用從 OpenWeatherMap 取得的天氣資料，依據天氣狀況顯示燈號，如果沒下雨就亮綠燈，有下雨就亮紅燈提醒要記得帶傘。

材料

- D1 mini
- 蜂鳴器
- 紅色 LED × 1
- 杜邦線及排針若干

線路圖



fritzing

■ 程式設計

Lab13.py

```
01 import network, urequests, ujson, machine
02 sta_if = network.WLAN(network.STA_IF)
03 sta_if.active(True)
04 sta_if.connect('FLAG-SCHOOL', '12345678')
05 while not sta_if.isconnected():
06     pass
07 print(sta_if.ifconfig()[0])           # 顯示 IP 位址
08
```

```
09 res = urequests.get(                                     # API 網址
10     "https://api.openweathermap.org/data/2.5/weather?" +
11     "q=" + "Taipei" + ", TW" +                          # 指定城市與國別
12     "&units=metric&lang=zh_tw&" +                      # 使用攝氏單位及繁中語系
13     "appid=" + # 以下填入註冊後取得的 API key
14     "XXXXXXXXXXXX")
15 j = ujson.loads(res.text);                               # 從 JSON 轉成字典
16 gLED = machine.Pin(16, machine.Pin.OUT) # 控制綠燈
17 rLED = machine.Pin(14, machine.Pin.OUT) # 控制紅燈
18 weatherID = j["weather"][0]["id"]                      # 天氣狀況代碼
19 weatherDesc = j["weather"][0]["description"] # 天氣狀況
20 if weatherID < 800:                                     # 雨天
21     rLED.value(1)                                       # 亮紅燈
22     gLED.value(0)
23 else:                                                    # 沒下雨
24     rLED.value(0)                                       # 亮綠燈
25     gLED.value(1)
26 print("目前天氣:", str(weatherID))
27 print("代碼意義:", weatherDesc )
```

實測

程式執行後可以看到 D1 mini 的 IP 位址，以及天氣代碼值與天氣狀況說明文字，同時也會點亮紅燈（下雨）或綠燈（沒雨）。

```
>>> %Run Lab13.py  
192.168.100.39  
目前天氣： 521  
代碼意義： 陣雨
```