AIoT

用Python學智慧聯網

Chapter 04：迴歸問題 – 體溫監測站

旗標創客

旗標創客

踏入 AIoT 的世界

旗標創客

1 何謂 AIoT

2 用Python 玩轉 AI

3 AI 的小大腦 – 微控制器

4 迴歸問題 – 體溫監測站

5 IoT 應用 – 體溫通報器

6 二元分類 – 雲端步頻紀錄儀

7 多元分類 – 無線體感鍵盤

8 CNN – 智慧聲控燈

旗標創客

前言

旗標創客

- 體溫是一種確認身體狀況的簡易依據，藉由 AI 和 ESP32 來打造個人專屬的溫度計，讓你隨時關心自己的健康。

旗標創客

前言

旗標創客

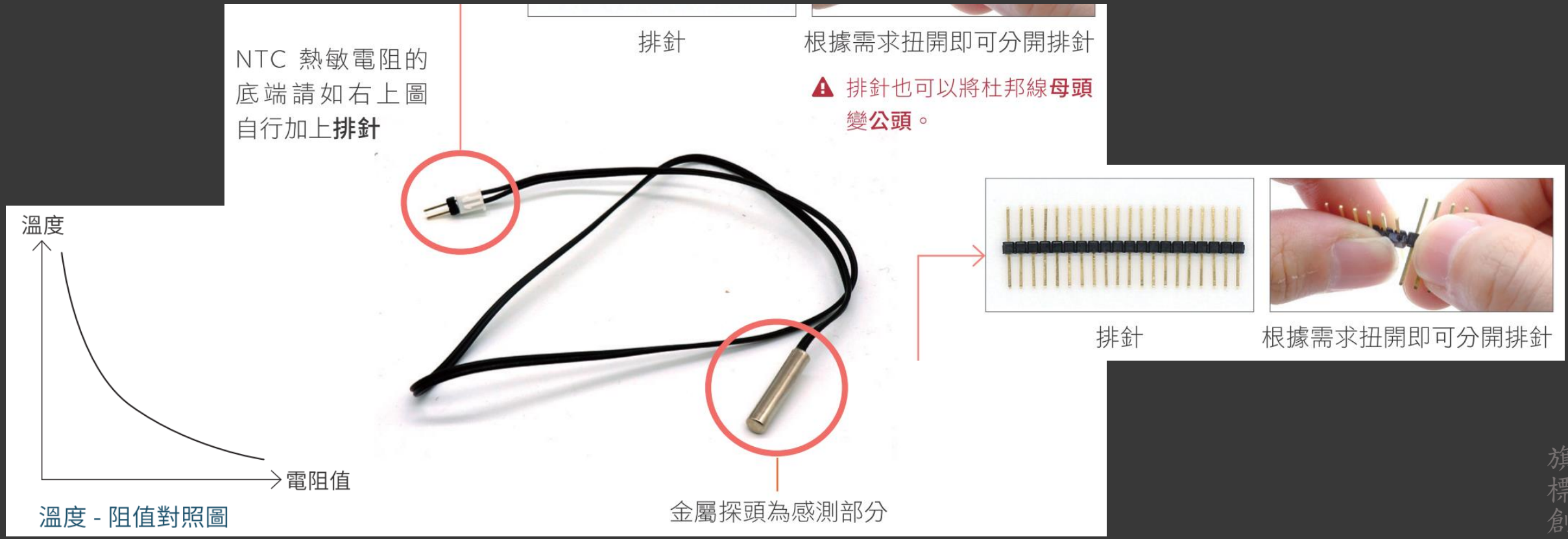
- 4-1 NTC 熱敏電阻
- 4-2 分壓電路
- 4-3 ESP32 類比輸入
  - LAB02 量測 ADC 值
- 4-4 資料輸入與檔案處理
  - LAB03 輸入資料並存檔
- 4-5 實作：體溫監測站
  - LAB04 實作：體溫監測站 – 蒐集資料
  - LAB05 實作：體溫監測站 – 預測溫度

旗標創客

4-1 NTC 熱敏電阻

旗標創客

- NTC 熱敏電阻有**溫度越高, 電阻值越低**特性。



旗標創客

4-2 分壓電路

旗標創客

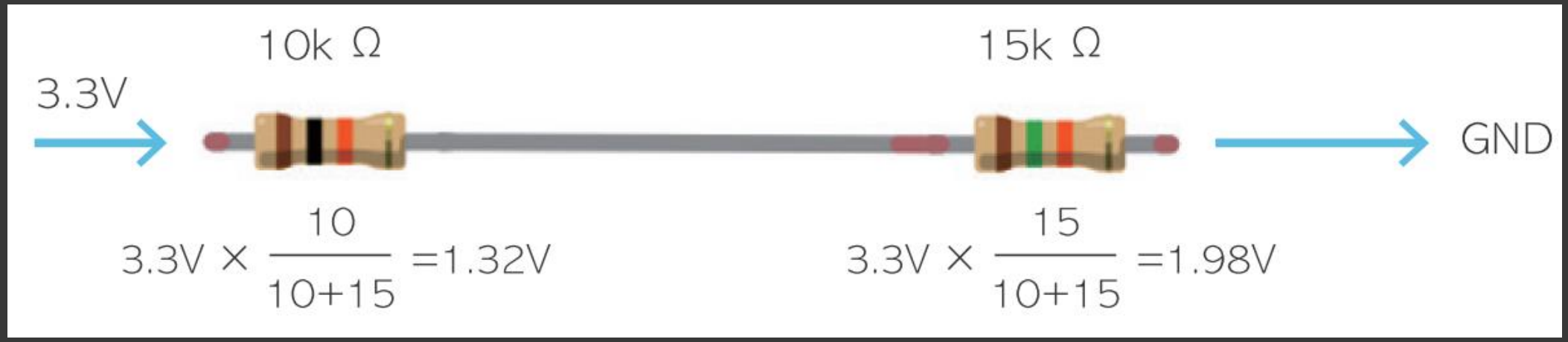
- 熱敏電阻的電阻值與溫度有相對應的關係, ESP32 沒有量測電阻值能力 , 但可量測**電壓**。
- 電壓與電阻可用『分壓電路』解釋兩者關係。

旗標創客

分壓電路

旗標創客

- 2 個電阻串聯時,電阻與電壓的關係呈正相關：



旗標創客

分壓電路

旗標創客

- 從尋找**電阻與溫度**關係改成**電壓與溫度**關係

旗標創客

4-3 ESP32 類比輸入

旗標創客

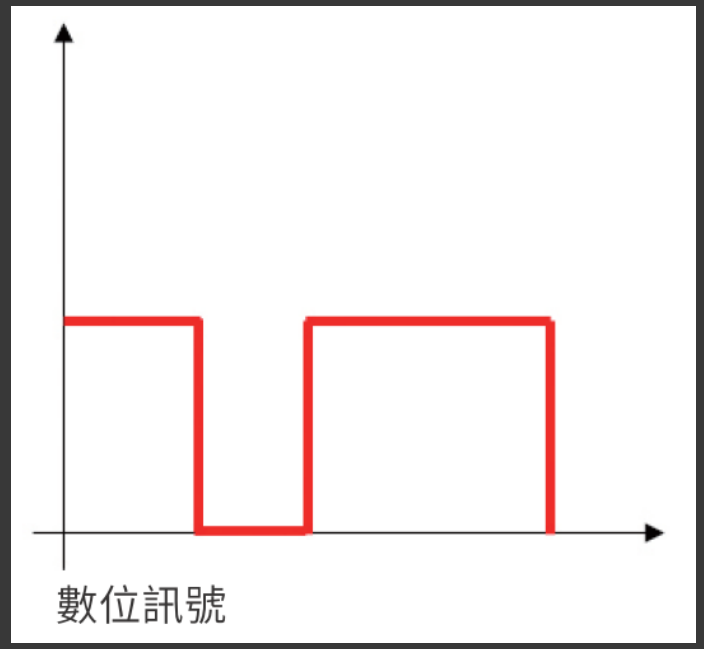
- GPIO 腳位可輸出電流與輸入訊號
- ESP32 可藉由感測器輸出的電壓變化了解目前感測器的狀態。

旗標創客

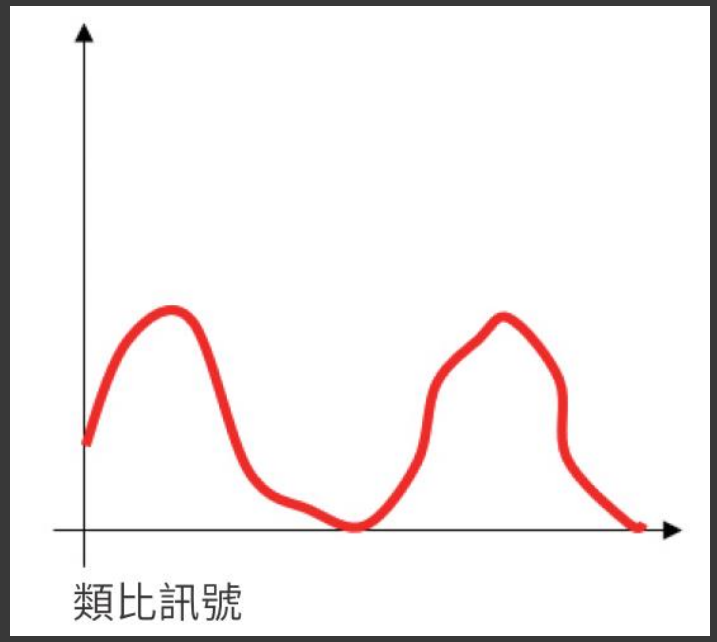
ESP32 類比輸入

旗標創客

不連續的訊號變化：



連續的訊號變化：



旗標創客

ESP32 類比輸入

旗標創客

- ESP32 無法讀取類比訊號, 需要透過 ADC 轉換成數位訊號。
- 使用 ADC 前 , 需要先選擇解析度。

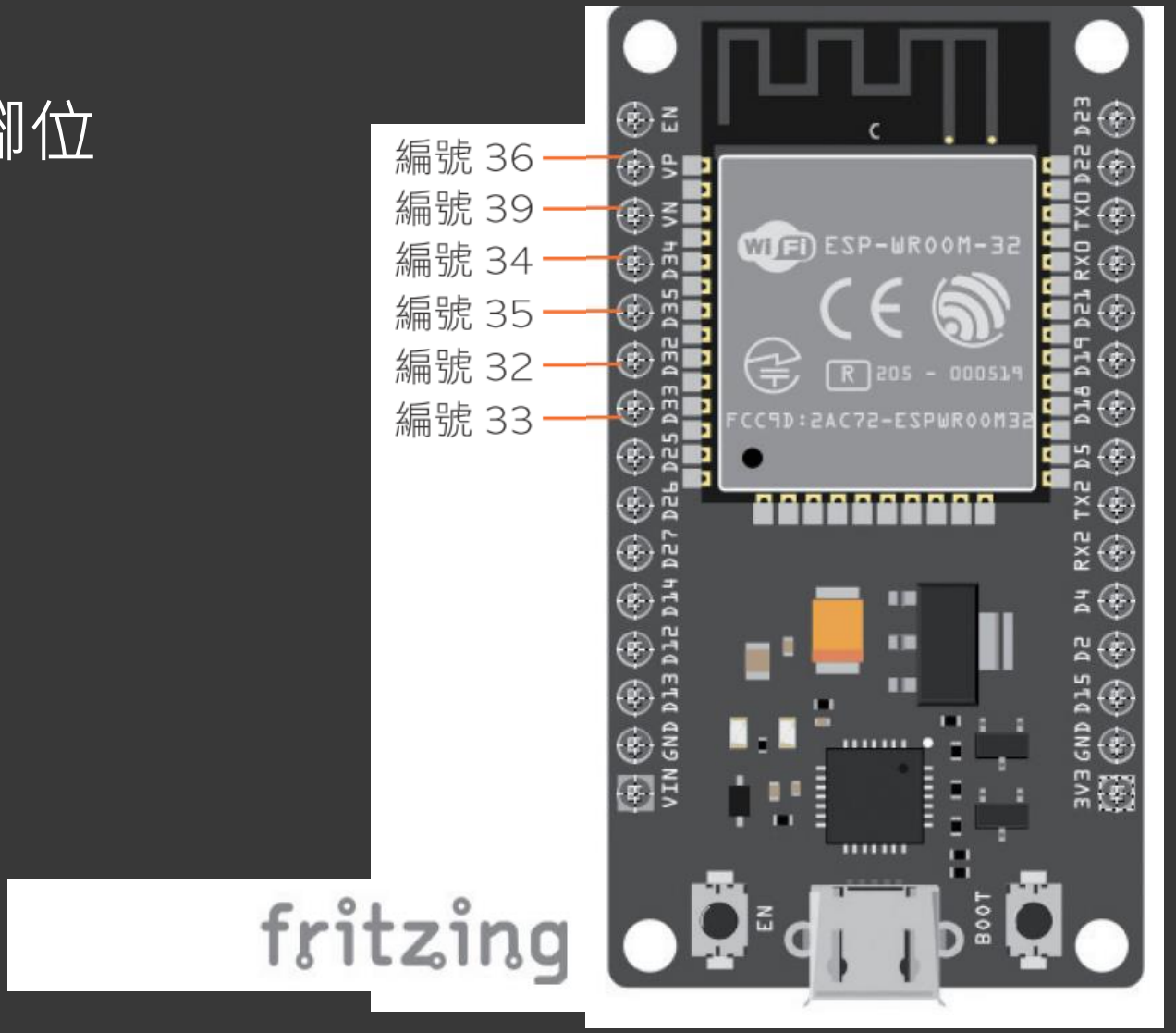
- ESP32 可以選擇 9bit、10bit、11bit 和 12bit 的解析度。當選擇 9bit(9 位元) 時, 就是將 0V~3.3V 轉換成 0~511, 所以讀取值為 170(≅ 511/3) 時, 大約就等於 3.3*V*/3 = 1.1*V* 的電壓。

旗標創客

ESP32 類比輸入

旗標創客

- ESP32 只有幾個腳位可以使用 ADC：



旗標創客

LAB02 量測 ADC 值

實驗目的	藉由類比腳位讀取熱敏電阻的 ADC 值。
材料	ESP32 NTC 熱敏電阻 排針 10k Ω 電阻 單芯線 1 條 杜邦線若干條 麵包板

⚠ 本套件中有 2 種電阻 (10k Ω、1k Ω)，可以使用電阻貼紙上的印章確認。

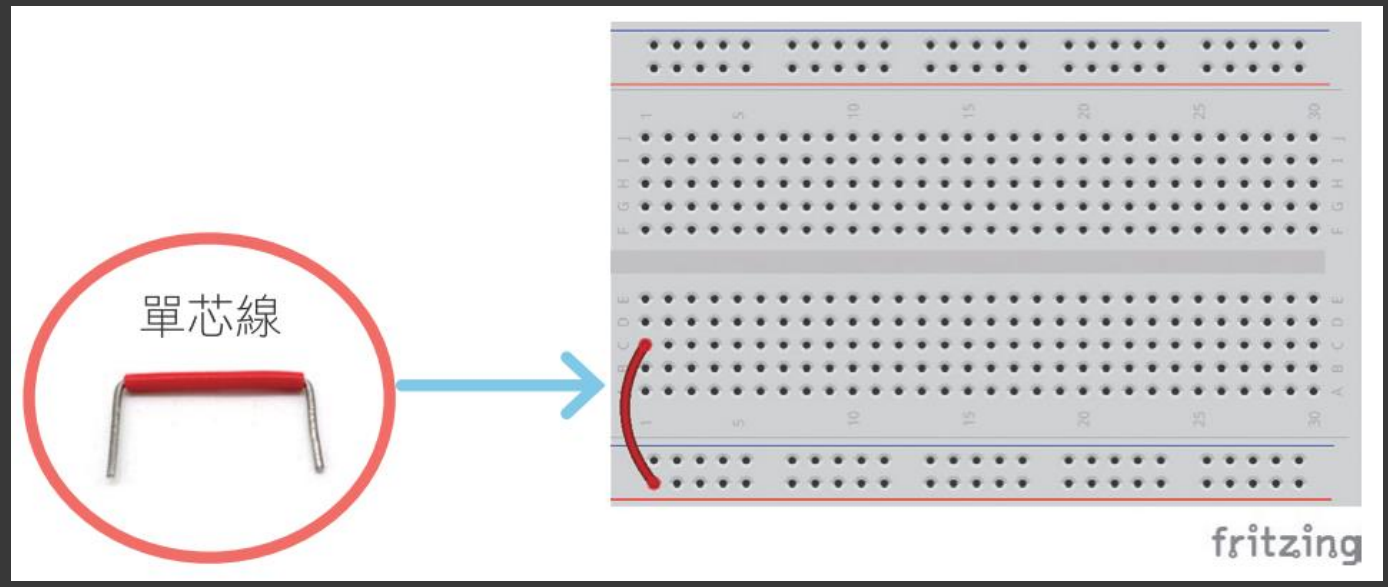
開發環境 Thonny

旗標創客
旗標創客

線路圖

旗標創客

- 1

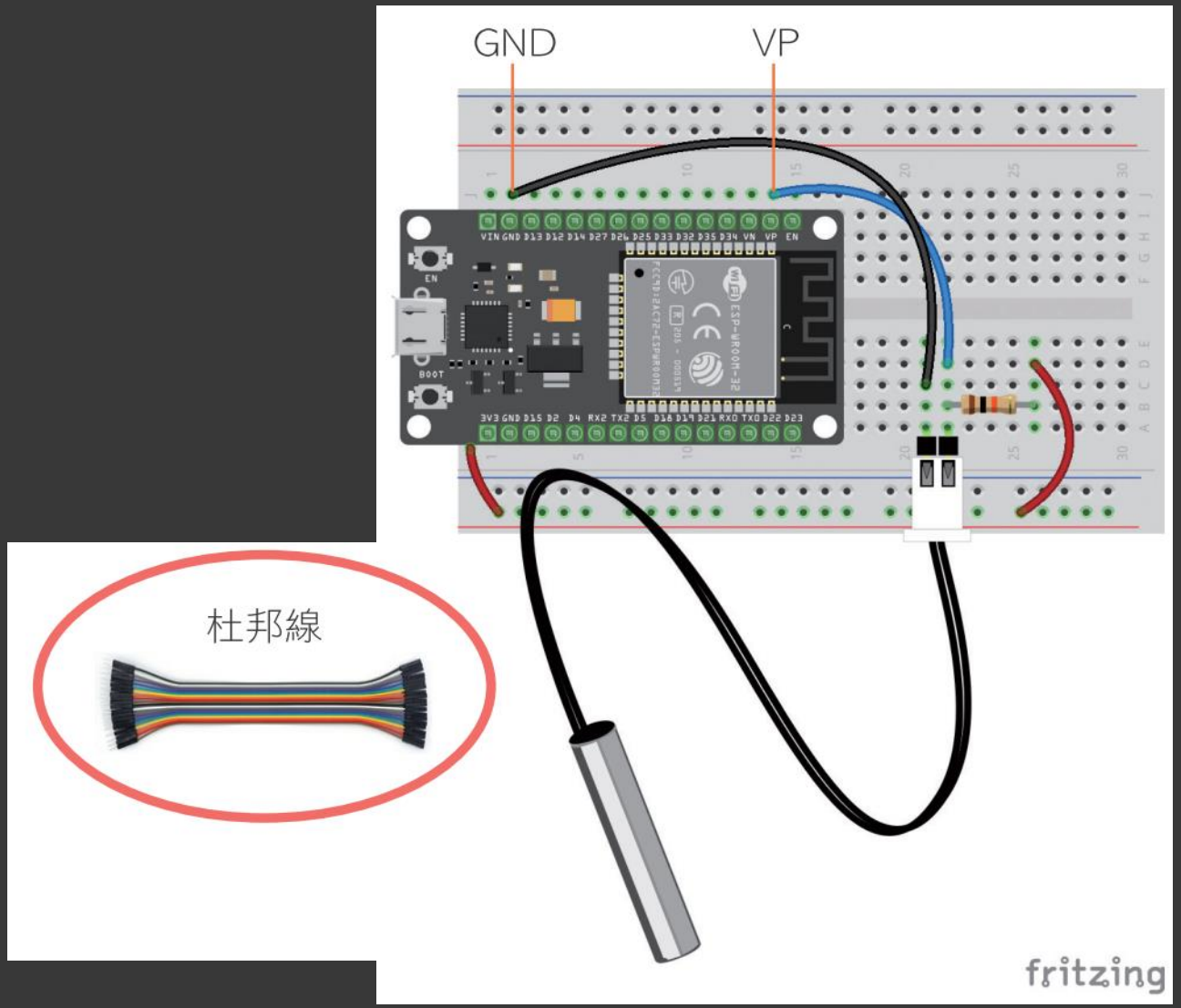


旗標創客

線路圖

旗標創客

- 2



旗標創客

實驗原理

旗標創客

- ESP32 可將熱敏電阻與 10kΩ 電阻串聯，
  由類比腳位讀取代表分壓的 ADC 值即可反應電阻的變化。

旗標創客

ADC 物件

旗標創客

Thonny

```
from machine import Pin, ADC
adc_pin=Pin(36)                 # 36 是 ESP32 的 VP 腳位
adc = ADC(adc_pin)              # 建立 ADC 物件
```

旗標創客

最大感測電壓

旗標創客

- ESP32 中預設的最大感測電壓為1V, 只要接收超過 1V 的電壓 , 得到的 ADC 值就是**峰值**。

參數	最大感測電壓
ADC.ATTN_0DB	1V
ADC.ATTN_2_5DB	1.34V
ADC.ATTN_6DB	2V
ADC.ATTN_11DB	3.6V

旗標創客

ADC 設定與讀取

旗標創客

Thonny

```
adc.width(ADC.WIDTH_9BIT)  # 設定 ADC 範圍。9BIT 代表範圍是 0~511
adc.atten(ADC.ATTN_11DB)   # 將最大感測電壓設定成 3.6V。
```

Thonny

```
adc.read()
```

旗標創客

程式設計

ex4-1

旗標創客

LAB02.py 量測 ADC 值 Thonny

```
1   from machine import Pin, ADC
2   import time
3
4   adc_pin=Pin(36)
5   adc = ADC(adc_pin)
6   adc.width(ADC.WIDTH_9BIT)
7   adc.atten(ADC.ATTN_11DB)
8   while True:
9       print(adc.read())   # 顯示 ADC 值
10      time.sleep(0.5)     # 暫停 0.5 秒
```

旗標創客

測試程式

旗標創客

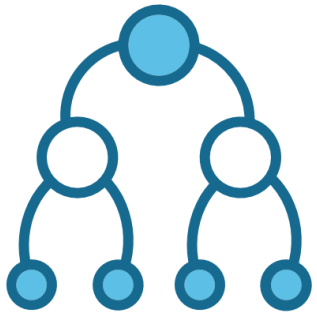
- 按 F5 執行程式 ：
- 嘗試握住熱敏電阻的金屬探頭，觀察 ADC 值之變化。

旗標創客

程式語言：五大語法結構

Sequence

循序執行

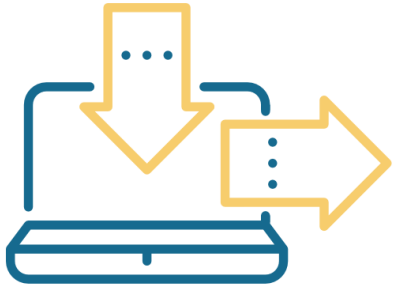


Conditional Statements

`if` 判斷式

Loops

迴圈



Input / Output

輸入/輸出

Functions

函數

4-4 資料輸入與檔案處理

旗標創客

- 輸入資料
- 儲存資料到檔案中

旗標創客

輸入資料

ex4-2

旗標創客

Thonny

```
>>>tem=input('現在溫度:')
現在溫度:
```

```
現在溫度:30 ←── 輸入 30
>>>tem
'30'
```

旗標創客

儲存資料到檔案中

旗標創客

- Python 在儲存資料到檔案時的步驟：

❶ 開啟檔案
❷ 寫入資料
❸ 關閉檔案

旗標創客

儲存資料到檔案中

ex4-3

旗標創客

❶ 開啟檔案

Thonny

```
f=open('檔案名稱', 'w')
```

❷ 寫入資料

Thonny

```
f.write(str(123))
```

❸ 關閉檔案

Thonny

```
f.close()
```

旗標創客

LAB03 輸入資料並存檔

實驗目的	學習將 ADC 值與使用者輸入的內容儲存到檔案中。
材料	同 LAB02

開發環境

Thonny

線路圖

同 LAB02

旗標創客
旗標創客

實驗原理

旗標創客

- 藉由 open() 開啟 **test.txt** 來寫入資料，並使用 input() 輸入**現在溫度**。
- 等輸入完畢後，write() 會將 "ADC 值" 和 "現在溫度" 一同儲存到 test.txt 中。

旗標創客

程式設計

ex4-4

LAB03.py 輸入資料並儲存 Thonny

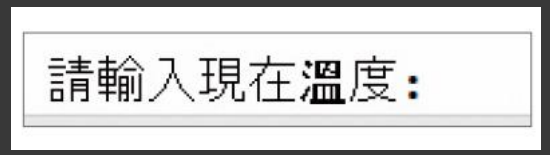
```
1   from machine import Pin, ADC
2
3   f=open('test.txt', 'w')
4
5   adc_pin=Pin(36)
6   adc = ADC(adc_pin)
7   adc.width(ADC.WIDTH_9BIT)
8   adc.atten(ADC.ATTN_11DB)
9   tem=input('請輸入現在溫度:')
10
11  f.write(str(adc.read())+' '+tem) # 以空格隔開 ADC 值與溫度值
12  f.close()
```

旗標創客
旗標創客

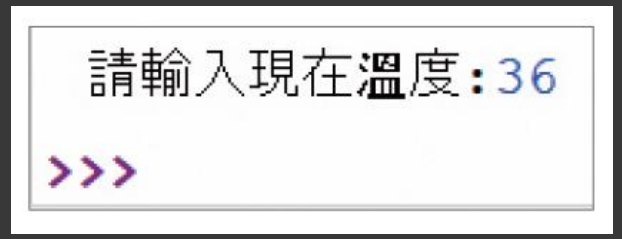
測試程式

旗標創客

- 執行程式：



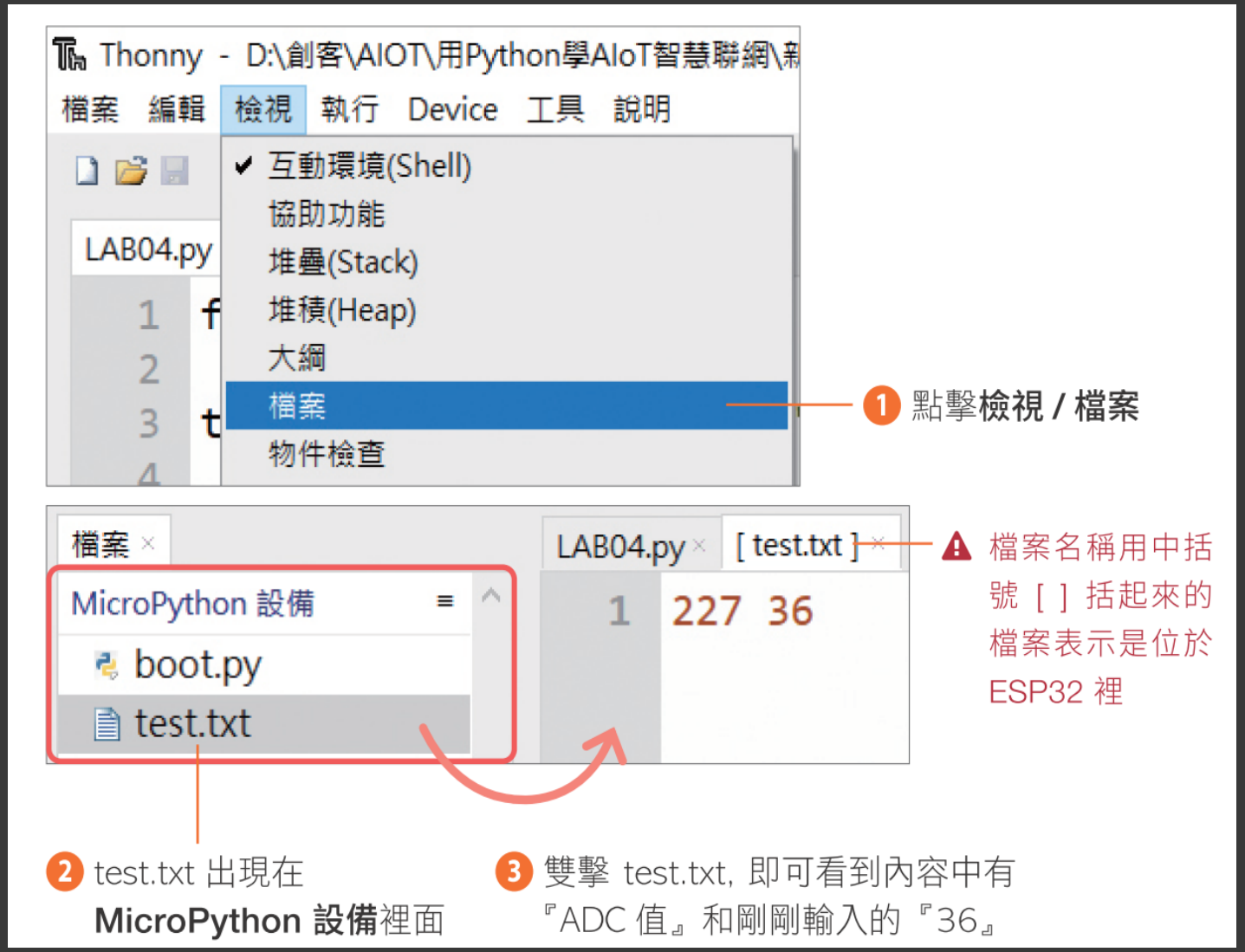
- 在後方欄位填入數值：



旗標創客

測試程式

旗標創客



旗標創客

4-5 實作：體溫監測站

旗標創客

- 使用熱敏電阻的 ADC 值預測體溫，根據以下流程說明：

❶ 蒐集資料

❷ 建立神經網路訓練模型

❸ 使用訓練好的模型進行預測

旗標創客

對應的 Python 開發環境

旗標創客

功能	Python 開發環境	理由
蒐集資料	Thonny	需藉由 ESP32 讀取感測器資料
建立神經網路訓練模型	Colab	需神經網路的函式庫及大量運算的能力
使用訓練好的模型進行預測	Thonny	需藉由 ESP32 讀取感測器資料

旗標創客

❶ 蒐集資料：量測及記錄 ADC 值與實際溫度

旗標創客

- 在蒐集資料前，先將 " 自製溫度計 " 和 "水銀溫度計 " 放在一起，才能確保兩者讀取到同一個溫度。
- 蒐集**範圍選擇正確**且**不重複**的資料。

旗標創客

LAB04 實作：體溫監測站 – 蒐集資料

旗標創客

- 實驗目的：
  蒐集多筆 ADC 值 (特徵) 及實際溫度 (標籤) 到『temperature.txt』供訓練模型使用
- 材料：同 LAB02、水銀溫度計
- 線路圖：同 LAB02
- 開發環境：Thonny

旗標創客

實驗原理

旗標創客

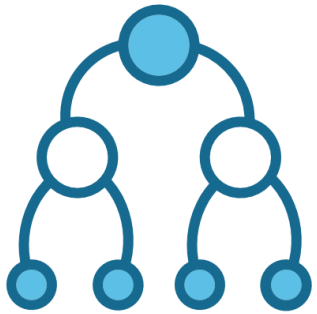
- 採每 0.01 秒讀一次 ADC 值，每 20 筆 ADC 值取平均值的方式來降低數值浮動的影響。
- 記錄過程中，使用 while True 迴圈不斷重複執行程式，等資料足夠時，輸入 end 就可以使用 break 跳出 while 迴圈來結束程式。

旗標創客

程式語言：五大語法結構

Sequence

循序執行

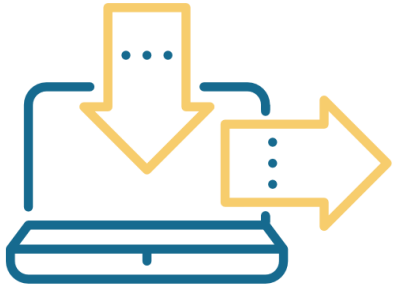


Conditional Statements

`if` 判斷式

Loops

迴圈



Input / Output

輸入/輸出

Functions

函數

程式設計

ex4-5

旗標創客

LAB04.py　實作：體溫監測站–蒐集資料　Thonny

```
1   from machine import Pin,ADC
2   import time
3
4   adc_pin = Pin(36)
5   adc = ADC(adc_pin)
6   adc.width(ADC.WIDTH_9BIT)
7   adc.atten(ADC.ATTN_11DB)
8
9   data=0                              # 資料總和
10  ti=1                                # 資料筆數
11  f=open('temperature.txt','w')       # 開啟txt檔
12
13  print(adc.read())                   # 確認數值是否正常
14
15  while True:
16      print('第'+str(ti)+'筆')       # 顯示紀錄第幾筆
```

旗標創客

程式設計

ex4-5

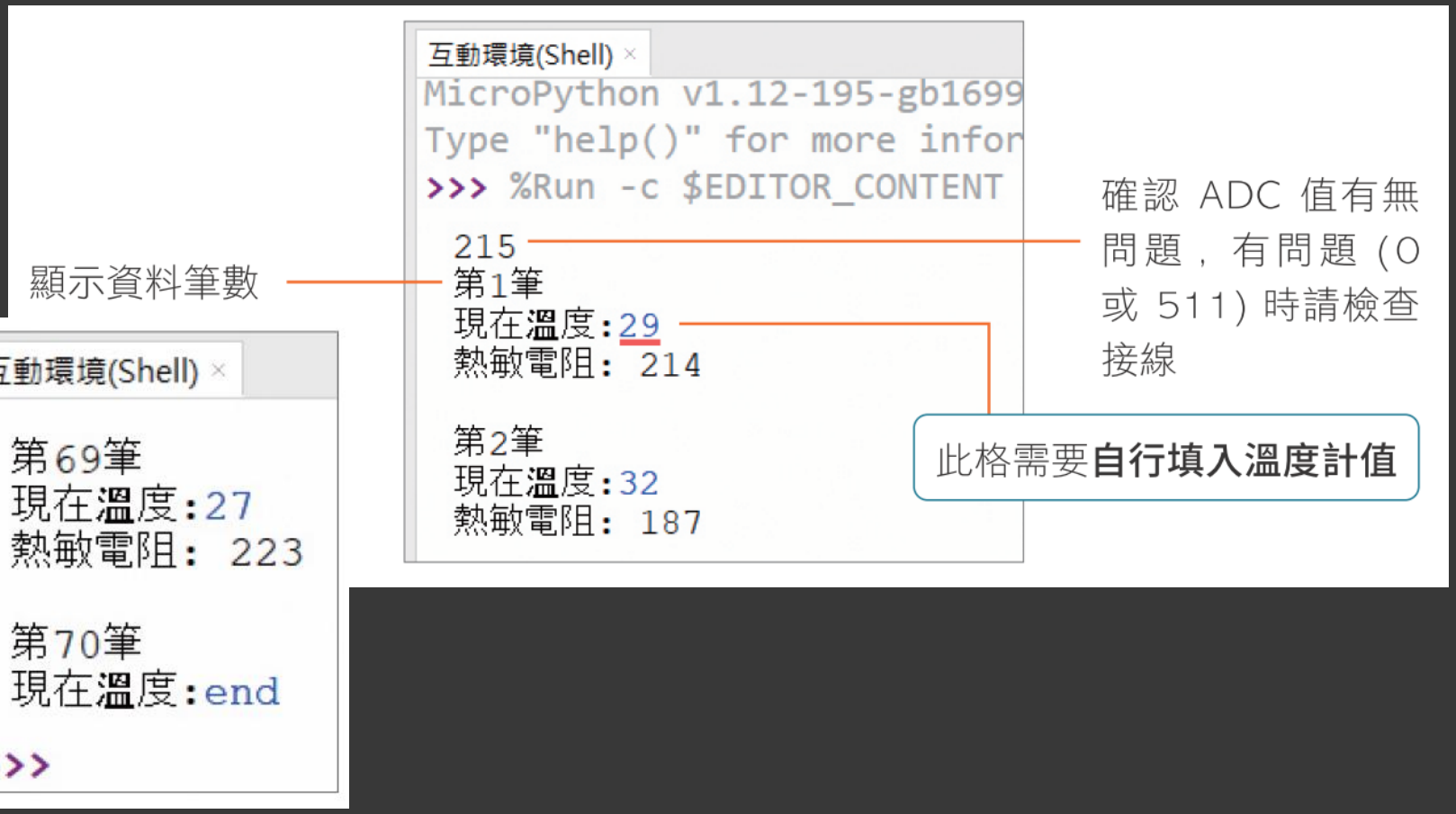
旗標創客

```
17      tem=input("現在溫度:")          # 輸入實際溫度
18
19      if(tem=='end'):
20          break
21      else:
22          for i in range(20):          # 重複20次
23              thermal=adc.read()       # ADC值
24              data=data+thermal        # 加總至data
25              time.sleep(0.01)
26
27          data=int(data/20)                    # 取平均
28          print('熱敏電阻:',data)
29          print('')                            # 多空一行
30          f.write(str(data)+' '+tem+'\n')      # data存到檔案中
31
32          data=0                               # 總和歸0
33          ti+=1                                # 次數加1
34  f.close()
```

旗標創客

測試程式

旗標創客

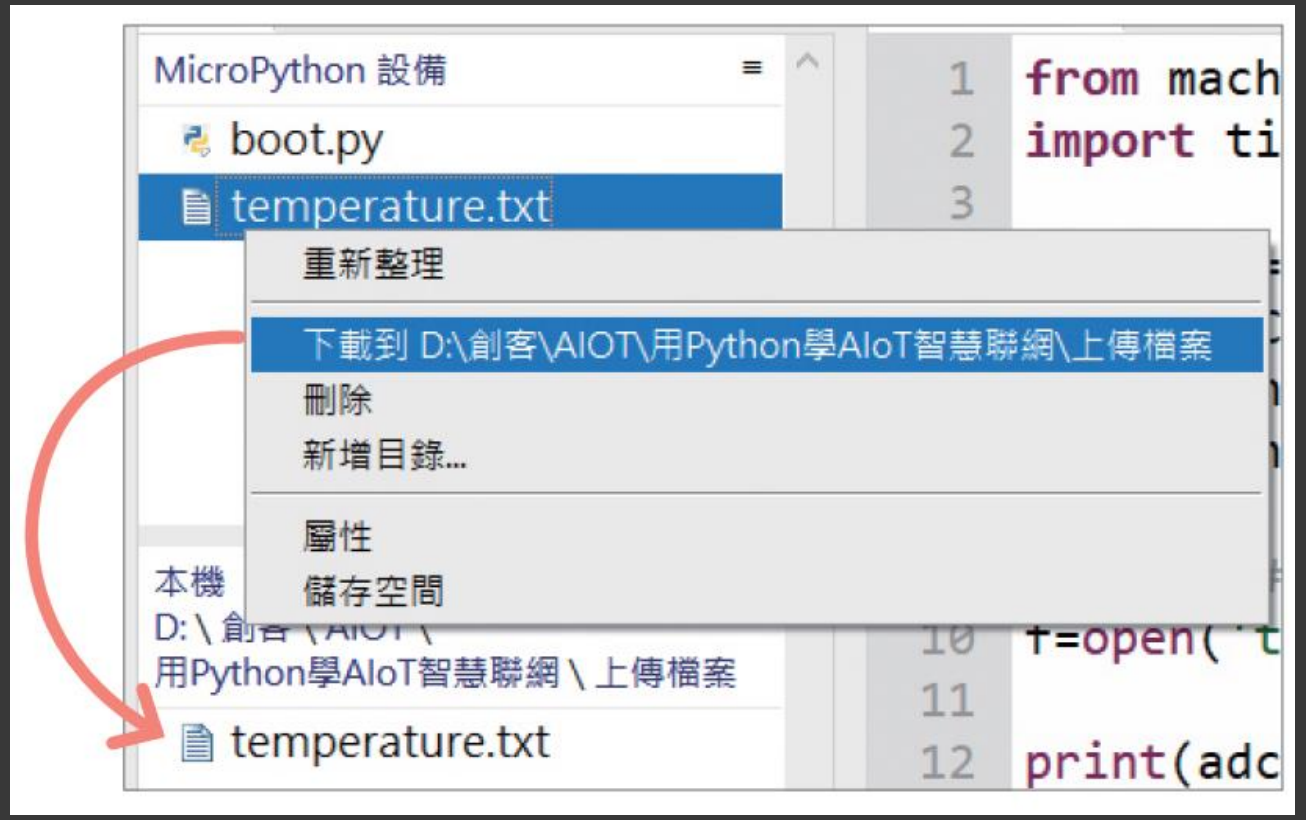


旗標創客

測試程式

旗標創客

- 蒐集完畢後將 ESP32 上的『temperature.txt』下載到電腦端。



旗標創客

❷ 建立神經網路：迴歸模型

旗標創客

- ❶ 讀取檔案
- ❷ 資料預處理
- ❸ 建立神經網路架構
- ❹ 編譯及訓練模型
- ❺ 測試模型
- ❻ 儲存模型

旗標創客

❷ 建立神經網路：迴歸模型

ex4-6

旗標創客

- 開發環境：Colab
- 讀取檔案：先將 temperature.txt 上傳至 Colab 才能開始訓練神經網路。

```
from google.colab import files
uploaded = files.upload()          # 匯入檔案
```

旗標創客

讀取檔案

旗標創客

ex4-7

temperature_model.ipynb(續) 匯入 keras_lite_convertor 模組 Colab

```
uploaded = files.upload()      # 匯入 keras_lite_convertor
```

temperature_model.ipynb(續) 讀取 temperature.txt Colab

```
import keras_lite_convertor as kc
path_name = 'temperature.txt'
Data_reader = kc.Data_reader(path_name, mode='regression')
data, label = Data_reader.read()
```

旗標創客

資料預處理

ex4-8

旗標創客

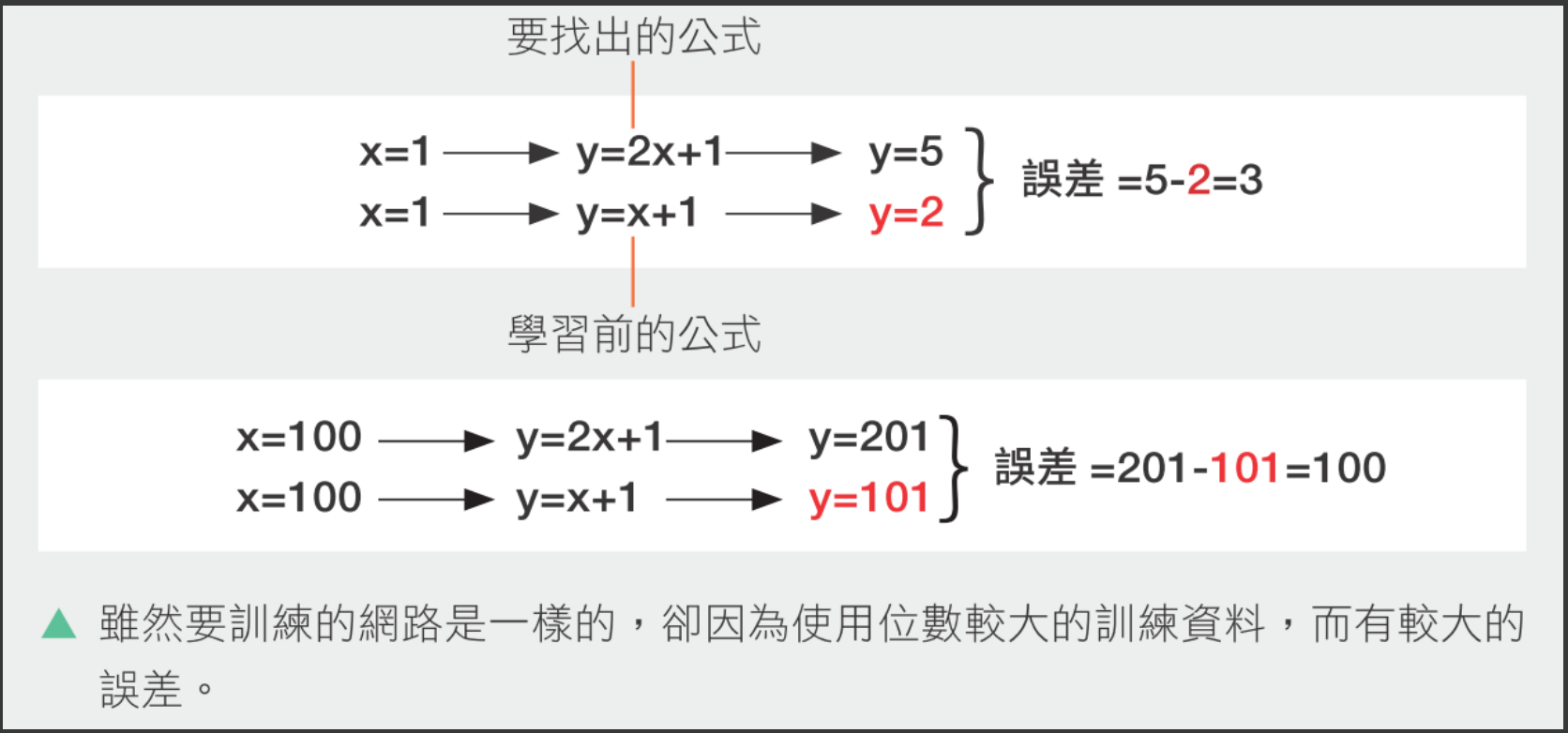
temperature_model.ipynb(續) 資料分割-訓練集 Colab

```
split_num = int(len(data)*0.85)
train_data=data[:split_num]          # 訓練用資料
train_label=label[:split_num]        # 訓練用標籤
```

旗標創客

資料預處理

旗標創客



▲ 雖然要訓練的網路是一樣的，卻因為使用位數較大的訓練資料，而有較大的誤差。

旗標創客

資料預處理

旗標創客

temperature_model.ipynb(續) 資料正規化 Colab ex4-9

```
mean = train_data.mean() # 平均數
data -= mean             # data 減掉平均數
std = train_data.std()   # 標準差
data /= std              # data 除以標準差
label /= 100             # 將 label 範圍落在 0~1 (label 正規化)
```

temperature_model.ipynb(續) 資料分割-驗證集、測試集 Colab ex4-10

```
# 驗證集
validation_data=data[split_num:-5]      # 驗證用資料
validation_label=label[split_num:-5]    # 驗證用標籤
# 測試集
test_data=data[-5:]                     # 測試用資料
test_label=label[-5:]                   # 測試用標籤
```

旗標創客

建立神經網路架構

ex4-11

旗標創客

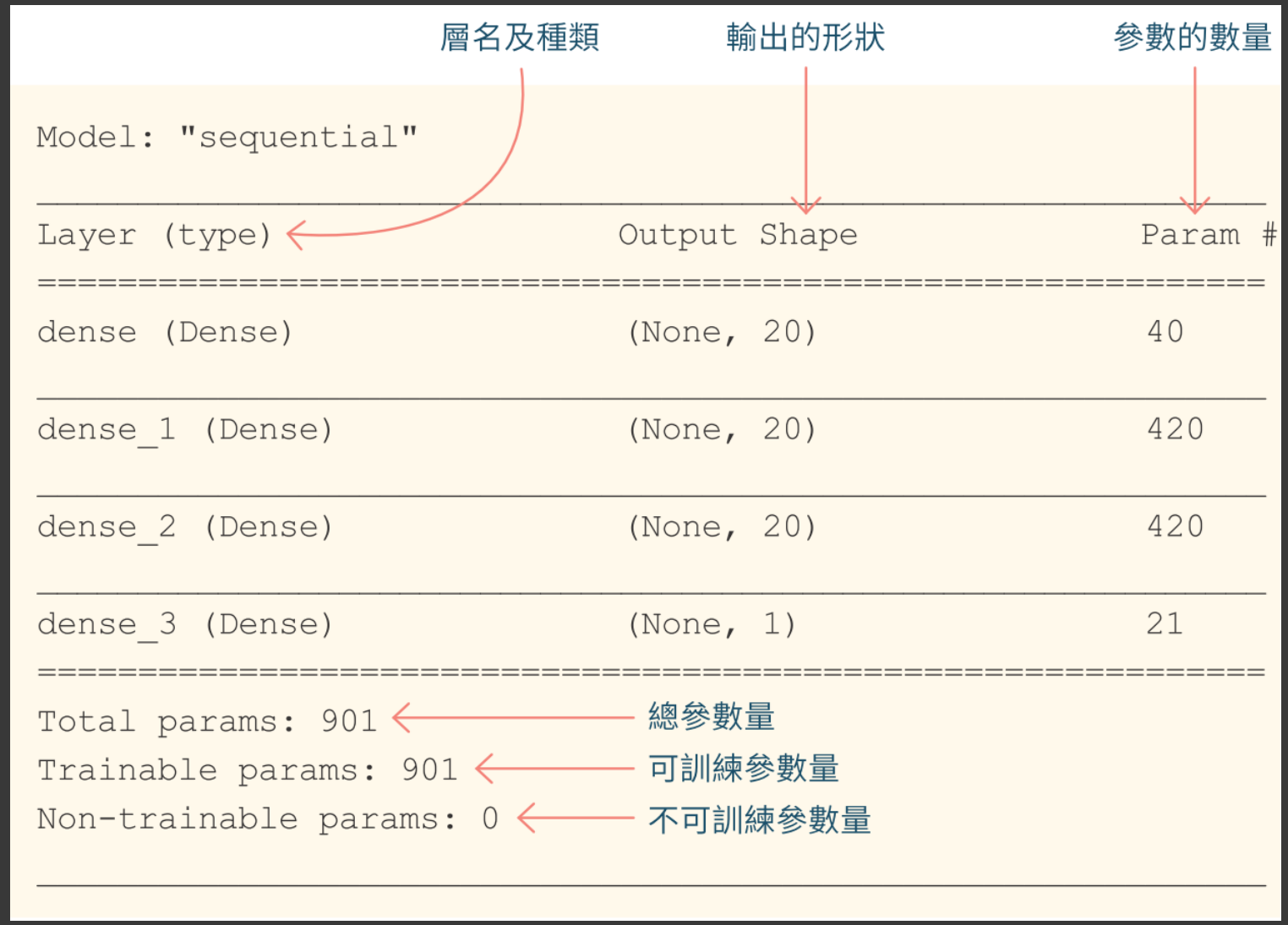
temperature_model.ipynb(續) 建立神經網路架構 Colab

```
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
model = Sequential()
model.add(layers.Dense(20,activation = 'relu',
                       input_shape=(1,)))
model.add(layers.Dense(20, activation = 'relu'))
model.add(layers.Dense(20, activation = 'relu'))
model.add(layers.Dense(1))
model.summary()      # 顯示模型資訊
```

旗標創客

建立神經網路架構

旗標創客



旗標創客

編譯及訓練模型

ex4-12

旗標創客

temperature_model.ipynb(續) 編譯及訓練模型 Colab

```
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
train_history = model.fit(train_data, train_label,
          validation_data=(validation_data, validation_label),
          epochs=1000)
```

```
Epoch 1/1000
2/2 ... - loss: 0.1781 - mae: 0.3778 ... - val_mae: 0.4690
Epoch 2/1000
2/2 ... - loss: 0.1549 - mae: 0.3481 ... - val_mae: 0.4397
Epoch 3/1000
2/2 ... - loss: 0.1357 - mae: 0.3211 ... - val_mae: 0.4099
...
Epoch 998/1000
2/2 ... - loss: 9.2451e-06 - mae: 0.0019 ... - val_mae: 0.0021
Epoch 999/1000
2/2 ... - loss: 8.3551e-06 - mae: 0.0018 ... - val_mae: 0.0023
Epoch 1000/1000
2/2 ... - loss: 9.0325e-06 - mae: 0.0020 ... - val_mae: 0.0023
```

旗標創客

測試模型

ex4-13

旗標創客

temperature_model.ipynb(續) 測試模型 Colab

```
# 預測值
print('predict:')
print(model.predict(test_data))
print()
# 實際值
print('real:')
print(test_label)
```

```
predict:
[[0.7242295 ]
 [0.4094786 ]
 [0.5608362 ]
 [0.05896593]
 [0.5899766 ]]

real:
[[0.73]
 [0.41]
 [0.56]
 [0.05]
 [0.59]]
```

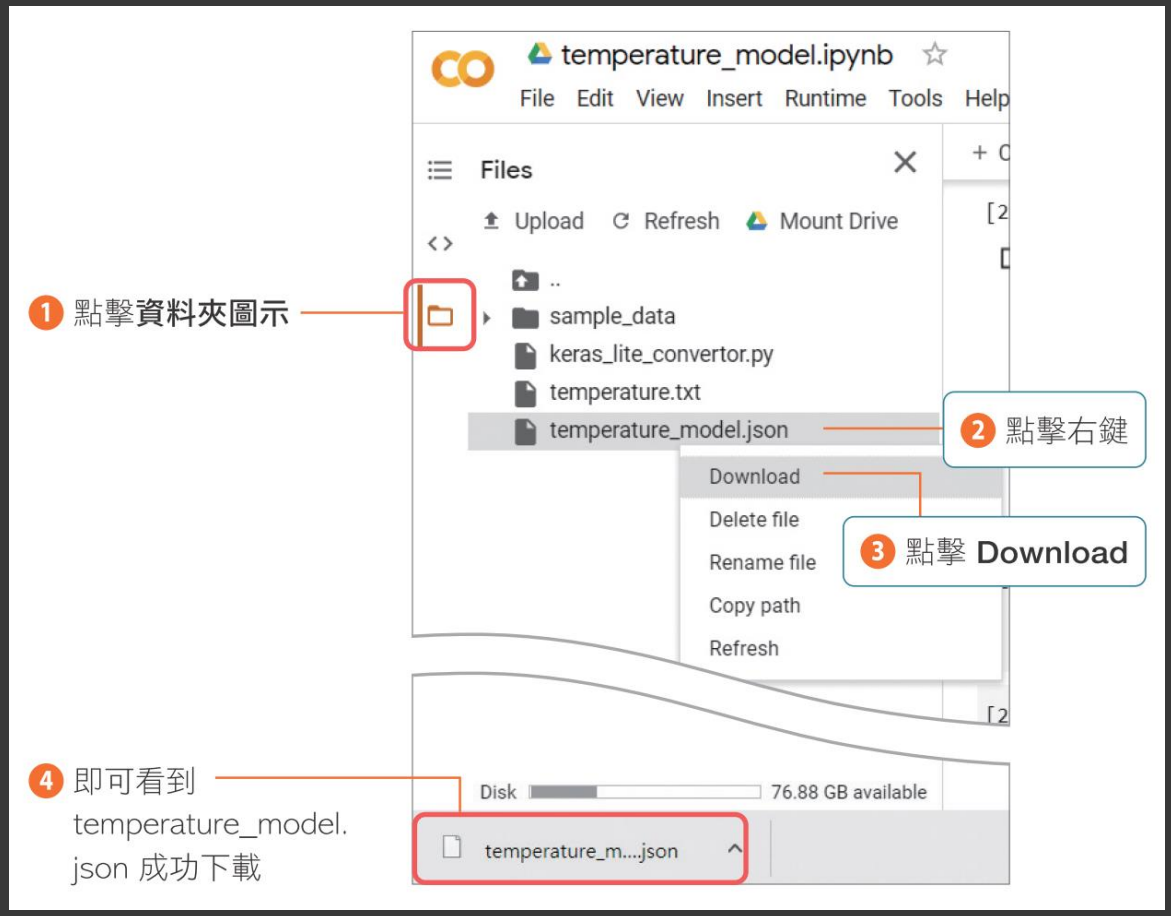
旗標創客

儲存模型

ex4-14

temperature_model.ipynb(續) 儲存模型 Colab

```
kc.save(model, 'temperature_model.json')
```



旗標創客
旗標創客

顯示正規化相關資訊

ex4-15

旗標創客

temperature_model.ipynb(續) 顯示資訊 Colab

```
print('mean=', mean)
print('std=', std)
```

```
mean= 170.98275862068965
std= 90.31162360353873
```

旗標創客

❸ 使用訓練好的模型進行溫度預測

旗標創客

- 預測溫度 需要熱敏電阻的 ADC 值，
  所以會重新移至 ESP32 上進行。

旗標創客

LAB05 實作：體溫監測站 – 預測溫度

旗標創客

- 實驗目的：
  將ADC值帶入訓練好的模型中進行溫度預測
- 材料、線路圖：同 LAB02
- 開發環境：Thonny

旗標創客

LAB05 實作：體溫監測站 – 預測溫度

旗標創客

- 實驗原理：
  將 Colab 中訓練好的模型上傳至 ESP32, 並將熱敏電阻蒐集到的資料正規化後帶入模型進行預測。最後預測出來的值需要乘上 100 才會是實際溫度。

旗標創客

實驗過程

旗標創客

- 上傳模型至 ESP32

旗標創客

實驗過程

ex4-16

旗標創客

Thonny

```
from keras_lite import Model
model = Model('temperature_model.json')    # 建立模型物件
```

Thonny

```
mean=170.98275862068965    # 請複製 Colab 上的 mean
std=90.31162360353873      # 請複製 Colab 上的 std
```

Thonny

```
data = np.array([int(data)])    # 將整數 data 轉換成 array 格式
data = data-mean                # data 減掉平均數
data = data/std                 # data 除以標準差
```

旗標創客

實驗過程

旗標創客

Thonny

```
tem = model.predict(data)
```

Thonny

```
tem[0]          # 藉由位置 0 取出預測值
```

Thonny

```
tem = round(tem[0]*100, 1)
```

旗標創客

程式設計

旗標創客

LAB05.py 實作:體溫監測站-預測溫度 Thonny

```
from machine import Pin, ADC
import time
from keras_lite import Model
import ulab as np

model = Model('temperature_model.json')

#增加神經網路的參數與模型
mean = 170.98275862068965
std = 90.31162360353873

```

旗標創客

程式設計

旗標創客

```
12  adc_pin = Pin(36)
13  adc = ADC(adc_pin)
14  adc.width(ADC.WIDTH_9BIT)
15  adc.atten(ADC.ATTN_11DB)
16
17  data=0
18
19  while True:
20
21      for i in range(20):
22          thermal=adc.read()
23          data=data+thermal
24          time.sleep(0.01)
25
26      data=data/20
27
```

旗標創客

程式設計

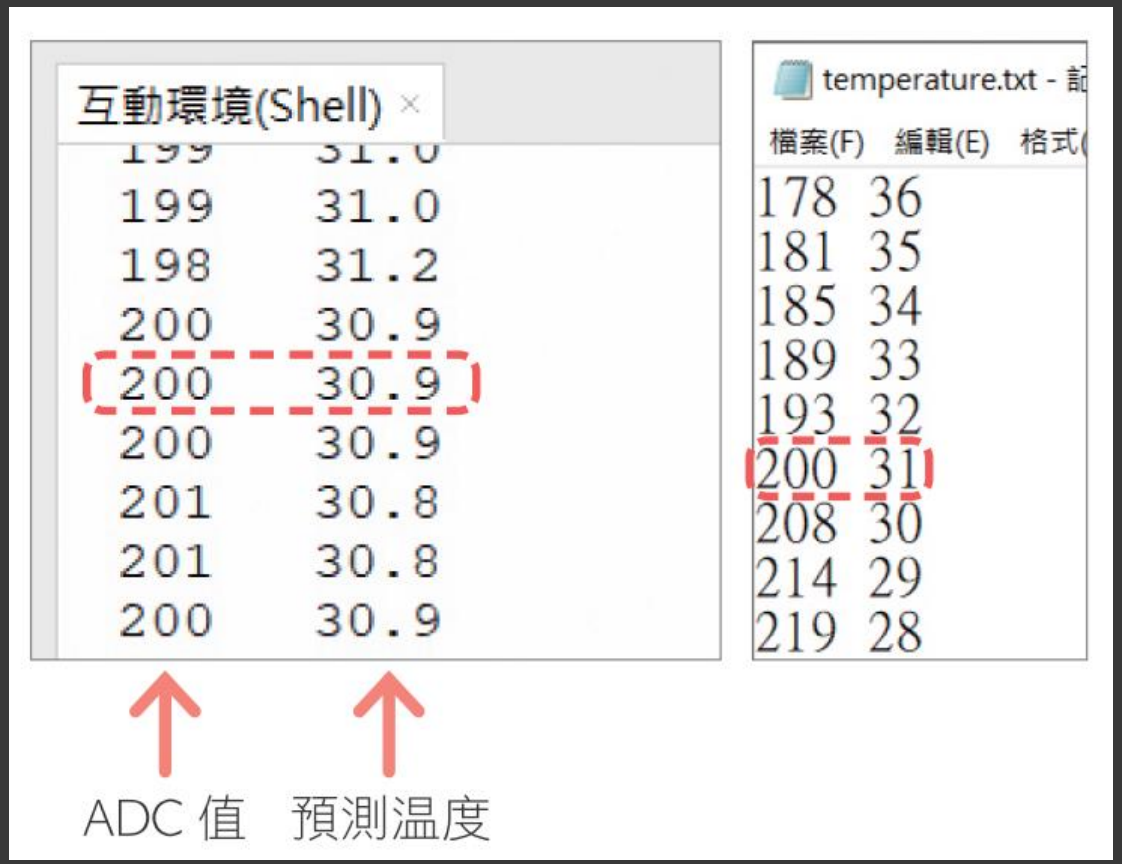
旗標創客

```
28      print(int(data), end='   ')    # 顯示 ADC 值;end=''代表不換行
29
30      data = np.array([int(data)])
31      data = data-mean
32      data = data/std
33      tem = model.predict(data)
34      tem = round(tem[0]*100, 1)
35      print(tem)              # 顯示實際溫度
36
37      data=0
38      time.sleep(1)           # 暫停 1 秒
```

旗標創客

測試程式

旗標創客



旗標創客