Ch08 整個網路都是我的感測器 – JSON 網路資料爬蟲

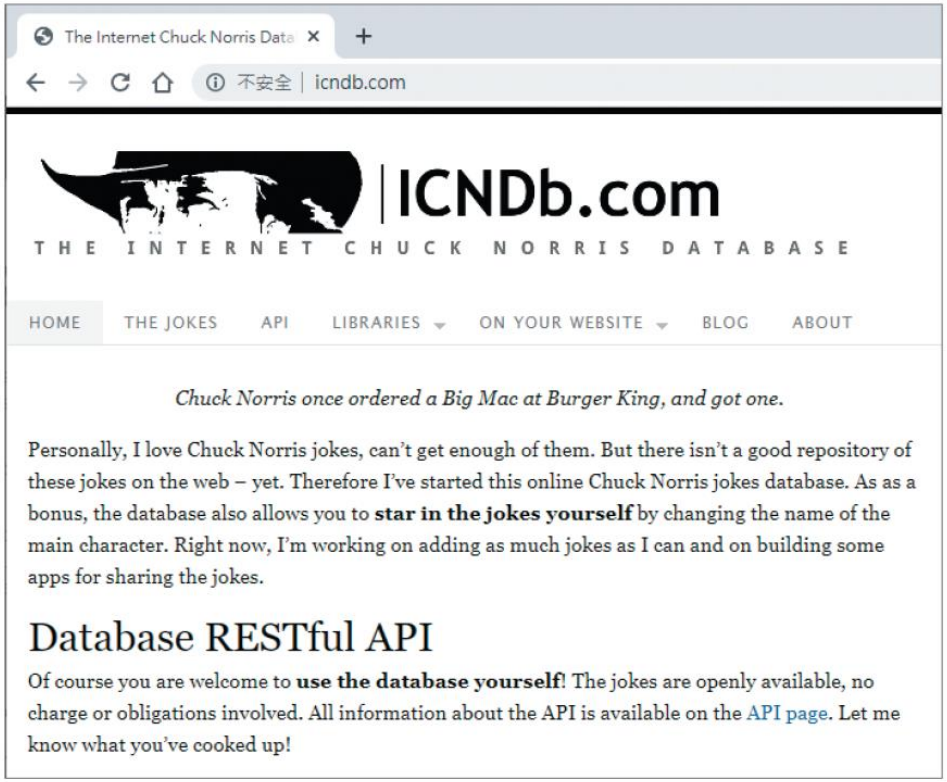
Python 的資料結構（容器）

Python 的資料結構

- 字串容器：由字元組成。
  - 例如：string = "52python"。
- tuple 容器：由資料物件組成。
  - 例如：tuple = (1, '2', 3)。
- 串列容器：由資料物件組成。
  - 例如：list = [1, '2', 3]。
- 集合容器：由資料物件組成。
  - 例如：set = {1, '2', 3}。
- 字典容器：由資料物件組成，以鍵：值表示。
  - 例如：dick = {'A':1, 'B':'2', 'C':3}。

8-1 抓取和解析 JSON 格式資料

JSON是一種電腦文字資料交換格式 (與 python 的容器相同)，容易被人和程式讀取。網路上有許多 API 和一般網站一樣有個網址，客戶端呼叫它後會傳回 JSON 格式的資料。



8-1 抓取和解析 JSON 格式資料

此 API 來自網站 The Internet Chuck Norris Database ，它的結構簡單，用來隨機查詢羅禮士笑話。

```
# D1 mini 得先連上 WiFi - 見第 7 章
url = "https://api.icndb.com/jokes/random" # API 網址
response = urequests.get(url)   # 呼叫 API 和取得回應資料
parsed = response.json()        # 把回應資料轉換成 JSON 格式
print(parsed)                   # 印出 JSON 內容
```

若執行以上程式，會在編輯器互動環境窗格中看到類似以下的結果：

```
{'value': {'id': 344, 'joke': 'Aliens DO indeed exist. They
just know better than to visit a planet that Chuck Norris is
 on.', 'categories': []}, 'type': 'success'}
```

8-1 抓取和解析 JSON 格式資料

把以上這段排版過，會得到類似以下結果：

```
{
   'value':{
      'id':344,
      'joke':'Aliens DO indeed exist. They just know better
 than to visit a planet that Chuck Norris is on.',
      'categories':[
      ]
   },
   'type':'success'
}
```

線上排版器

http://jsonviewer.stack.hu/

8-1 抓取和解析 JSON 格式資料

可看到 JSON 資料是以鍵 (資料名稱) : 值 (資料內容) 的方式配對的多層結構 :

```
value = parsed["value"]
joke = value["joke"]
# 上面這兩句也可合併成 joke = parsed["value"]["joke"]
print(joke)
```

這會在編輯器互動環境視窗印出以下字串 :

```
Aliens DO indeed exist. They just know better than to visit a
 planet that Chuck Norris is on.
# 翻譯: 外星人的確存在, 只是沒膽造訪查克羅禮士所在的星球。
```

8-2 國際太空站什麼時候經過頭上？

- **Lab22**

國際太空站查詢器	
實驗目的	從 API 取得 ISS 掠過你所在地頭上的時間。
材料	• D1 mini • OLED 模組
API 網址	http://api.open-notify.org/iss-pass.json

- **接線圖**

與第 2 章 Lab 03 相同。

ex8-5

8-2 國際太空站什麼時候經過頭上？

- **使用 API**

要使用此 API，網址後面必須加上所在地的經緯度參數：

```
http://api.open-notify.org/iss-pass.json?lat=緯度&lon=經度
```

以台北為例，經緯度是 (25.066667, 121.516667)，
API 參數為"?lat=25.066667&lon=121.516667"。

呼叫後傳回內容範例如下：

ex8-6

8-2 國際太空站什麼時候經過頭上？

```
{
  "message": "success",       # 查詢成功與否
  "request": {                # 查詢參數
    "altitude": 100,          # 觀察者高度預設為 100(公尺)
    "datetime": 1575511953,   # 查詢時間
    "latitude": 25.066667,    # 緯度
    "longitude": 121.516667,  # 經度
    "passes": 5               # 此次查詢傳回的國際太空站經過次數
  },
  "response": [               # 查詢結果
    {                         # 第1次掠過 (第1筆資料)
      "duration": 282,        # 經過時間 (秒)
      "risetime": 1576477844  # 開始時間 (Unix 時間戳)
    },
    {                         # 第2次掠過 (第2筆資料)
      "duration": 633,        # 經過時間 (秒)
      "risetime": 1576483518  # 開始時間 (Unix 時間戳)
    },
# …下略
  ]
}
```

https://www.cadch.com/article/timestamp/index.php

ex8-7

8-2 國際太空站什麼時候經過頭上？

- **設計原理**

此 API 會將 ISS 最近會掠過指定經緯度的時刻和時間長度放入鍵 response 的值中：

```
data = parsed["response"][0] # 取得 response 下的第 1 筆資料
time = data["risetime"]
print(time)
```

若想取得第 2 筆資料，要寫 data = parsed["response"][1] 。

8-2 國際太空站什麼時候經過頭上？

time.localtime() 方法會傳回 (2019, 12, 16, 14, 30, 44, 0, 350) 這樣類似串列的資料組，可存取當中每一項日期或時間資訊：

項目編號	0	1	2	3	4	5	6	7
值	2019	12	16	14	30	44	0	350
意義	年	月	日	時	分	秒	星期幾 (星期一為 0, 星期日為 6)	今年第幾日

```
pass_localtime = time.localtime(Unix 時間戳秒數)
year = pass_localtime[0] # 取得年份
hour = pass_localtime[3] # 取得小時
```

如果呼叫 time.localtime() 時沒有給 Unix 時間戳參數，它就會傳回控制板目前的系統時間 。

ex8-8

8-2 國際太空站什麼時候經過頭上？

- 程式設計

```
import network, urequests, utime
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C

oled = SSD1306_I2C(128, 64, I2C(scl=Pin(5), sda=Pin(4)))

ssid = "你的 WiFi 名稱"
pw = "你的 WiFi 密碼"
url = "http://api.open-notify.org/iss-pass.json?lat=25.066667
&lon=121.516667"

print("連接 WiFi...")
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(ssid, pw)
while not wifi.isconnected():
    pass
print("已連上")

response = urequests.get(url)
```

ex8-8

8-2 國際太空站什麼時候經過頭上？

```
if response.status_code == 200:

    parsed = response.json()
    print("JSON 資料查詢成功:")
    print("")

    print("國際太空站下次掠過時間:")

    # 取得 response 底下第 1 筆資料
    data = parsed["response"][0]

    # 計算正確 Unix 時間戳秒數
    pass_time = int(data["risetime"]) - 946684800 + 28800
    # 換算成本地時間
    pass_localtime = utime.localtime(pass_time)
    # 取得年, 月, 日, 時, 分, 秒
    year = str(pass_localtime[0])
    month = str(pass_localtime[1])
    day = str(pass_localtime[2])
    hour = str(pass_localtime[3])
    minute = str(pass_localtime[4])
    second = str(pass_localtime[5])
```

ex8-8

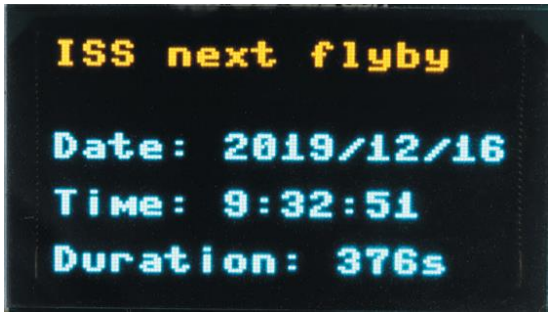
8-2 國際太空站什麼時候經過頭上？

```
# 取得經過時間
duration = str(data["duration"])

# 把時間資料組合成字串
date = str(year) + "/" + str(month) + "/" + str(day)
time = str(hour) + ":" + str(minute) + ":" + str(second)

# 在編輯器輸出資料
print(date + " " + time + " (為時 " + duration + " 秒)")

# 在 OLED 模組顯示資料
oled.fill(0)
oled.text("ISS next flyby", 0, 0)
oled.text("Date: " + str(year) + "/" +
          str(month) + "/" + str(day), 0, 24)
oled.text("Time: " + str(hour) + ":" +
          str(minute) + ":" + str(second), 0, 40)
oled.text("Duration: " + str(duration) + "s", 0, 56)
oled.show()
```

8-2 國際太空站什麼時候經過頭上？

- **實測**

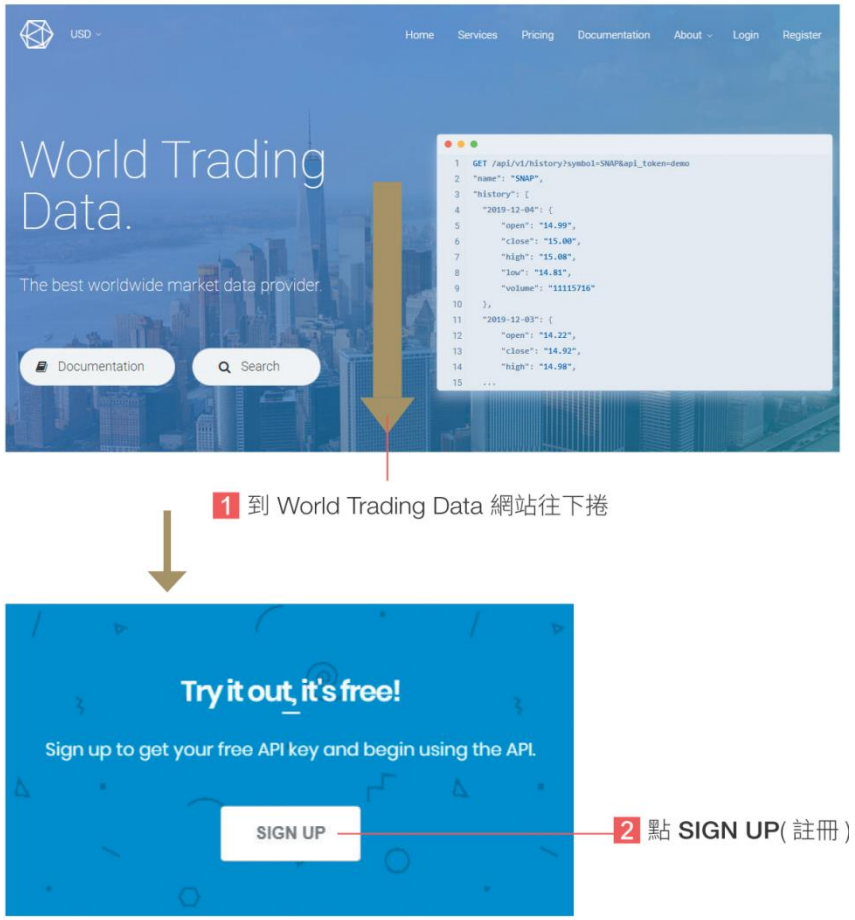
修改程式中的 WiFi 名稱與密碼，然後執行程式，編輯器互動環境窗格和 OLED 一會兒後就會顯示查詢到的結果：

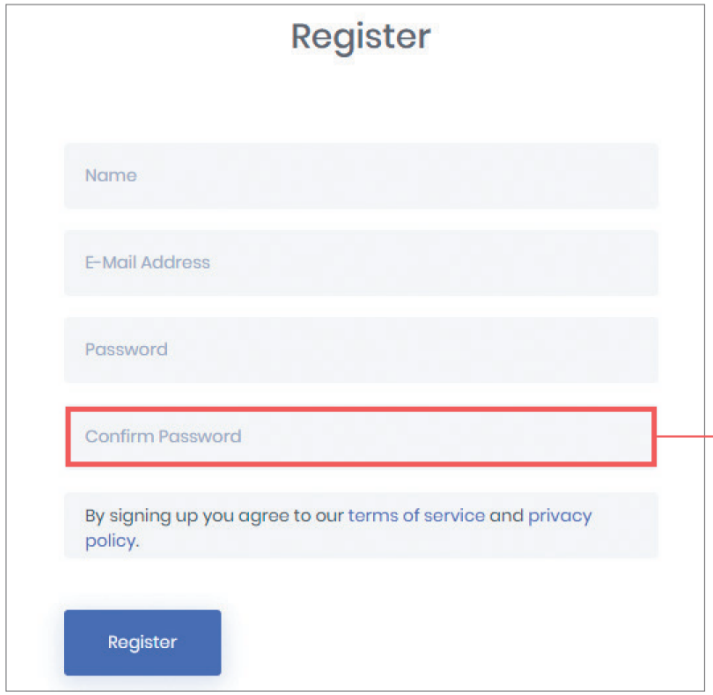



```
連接 WiFi...
已連上
JSON 資料查詢成功:

國際太空站下次掠過時間:
2019/12/16 9:32:51 (為時 376 秒)
```

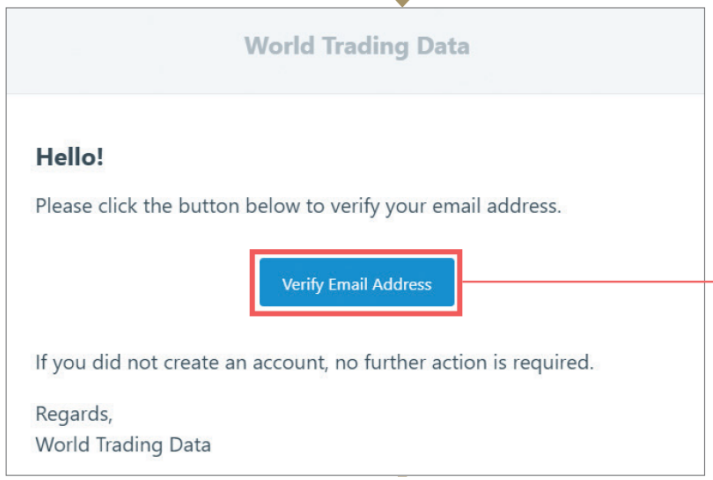
8-3 股價查詢

- 申請帳號與取得 **API** 金鑰



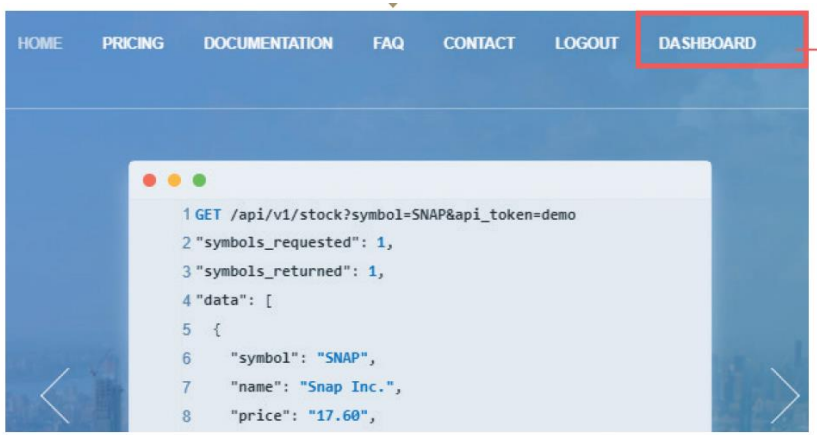


3 填寫信箱、帳戶名稱與密碼後點 Register (註冊)

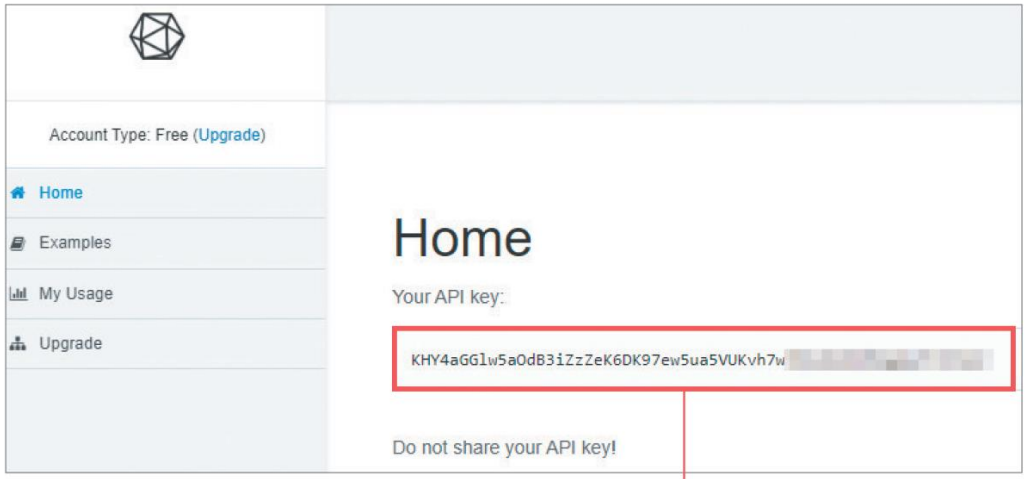


4 系統會寄送驗證信到你的 email 信箱，點信件中的 **Verify Email Access** (驗證信箱)。

8-3 股價查詢



5 驗證帳戶後，你應該會跳轉到 https://www.worldtradingdata.com/home 看到你的帳戶相關資訊。日後要重新進入時，在首頁點 **Dashboard**（會員區）。



6 不同於 Lab 22 的 API, 此 API 在呼叫時需要加上驗證金鑰。在帳戶頁面即可看到你個人的金鑰。

8-3 股價查詢

- **Lab23**

簡易股票機	
實驗目的	從 API 取得上市股票的相關交易資訊。
材料	• D1 mini • OLED 模組
API 網址	https://api.worldtradingdata.com/api/v1/stock

- **接線圖**

與 Lab 03 相同。

8-3 股價查詢

- 使用 **API**

申請好帳號後，查詢股價的 API 如下：

```
https://api.worldtradingdata.com/api/v1/stock?symbol=股票代號&
 api_token=金鑰
```

公司	蘋果	亞馬遜	特斯拉	任天堂	台積電
股票名稱	AAPL	AMZN	TSLA	NTDOY	TSM

8-3 股價查詢

此 API 也允許一次同時查詢多個股票：

```
stock?symbol= AAPL,TSLA,TSM # 查詢蘋果, 特斯拉和台積電股價
```

回傳的 JSON 資料範例如下：

```
{
   "symbols_requested":3,    # 查詢的股票數量
   "symbols_returned":3,     # 傳回的股票數量
   "data":[                  # 傳回的資料
      { # 第 1 筆
         "symbol":"AAPL",
         "name":"Apple Inc.",
         "currency":"USD",
         "price":"275.15",
         # …下略
      },
```

8-3 股價查詢

```
        { # 第 2 筆
           "symbol":"TSLA",
           "name":"Tesla, Inc.",
           "currency":"USD",
           "price":"358.39",
           # …下略
        },
        { # 第 3 筆
           "symbol":"TSM",
           "name":"Taiwan Semiconductor Manufacturing Company
                   Limited",
           "currency":"USD",
           "price":"58.25",
           # …下略
        }
    ]
}
```

8-3 股價查詢

- **設計原理**

和 Lab 22 一樣，這裡只會查詢一支股票的資訊，並顯示在 OLED 模組上。在此 Lab 想查詢的資料如下：

鍵 data 底下的子鍵	意義
name	公司全名
symbol	股票名稱
price	股價
currency	股價幣值
change_pct	股價漲跌幅 (百分比)
pe	市盈率或本益比
last_trade_time	最後交易時間

8-3 股價查詢

- 程式設計

```
import network, urequests
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C

oled = SSD1306_I2C(128, 64, I2C(scl=Pin(5), sda=Pin(4)))

ssid = "你的 WiFi 名稱"
pw = "你的 WiFi 密碼"

stock_name = "TSM" # 欲查詢的股票名稱

key = "你的查詢金鑰"
# 產生查詢網址
url = "https://api.worldtradingdata.com/api/v1/stock?symbol="
       + stock_name + "&api_token=" + key
```

ex8-10

8-3 股價查詢

```
print("連接 WiFi...")
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(ssid, pw)
while not wifi.isconnected():
    pass
print("已連上")

response = urequests.get(url)

if response.status_code == 200:

    parsed = response.json()
    print("JSON 資料查詢成功:\n")

    # 取得第1筆股票資料
    stock = parsed["data"][0]

    # 取得資料中的特定項目
    name = stock["name"]
    symbol = stock["symbol"]
    price = stock["price"] + " " + stock["currency"]
    change = stock["change_pct"] + "%"
```

ex8-10

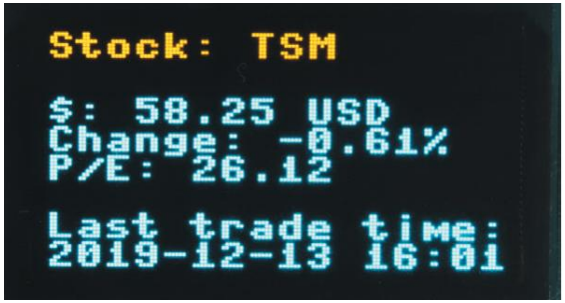
8-3 股價查詢

```
pe = stock["pe"]
trade_time = stock["last_trade_time"]

# 在編輯器輸出資料
print("公司名稱: " + name + " (" + symbol + ")")
print("股價: " + price)
print("漲跌幅: " + change)
print("市盈率: " + pe)
print("最後交易時間: " + trade_time)

# 在 OLED 模組顯示資料
oled.fill(0)
oled.text("Stock: " + symbol, 0, 0)
oled.text("$: " + price, 0, 16)
oled.text("Change: " + change, 0, 24)
oled.text("P/E: " + pe, 0, 32)
oled.text("Last trade time:", 0, 48)
oled.text(trade_time, 0, 56)
oled.show()
```

8-3 股價查詢

- **實測**

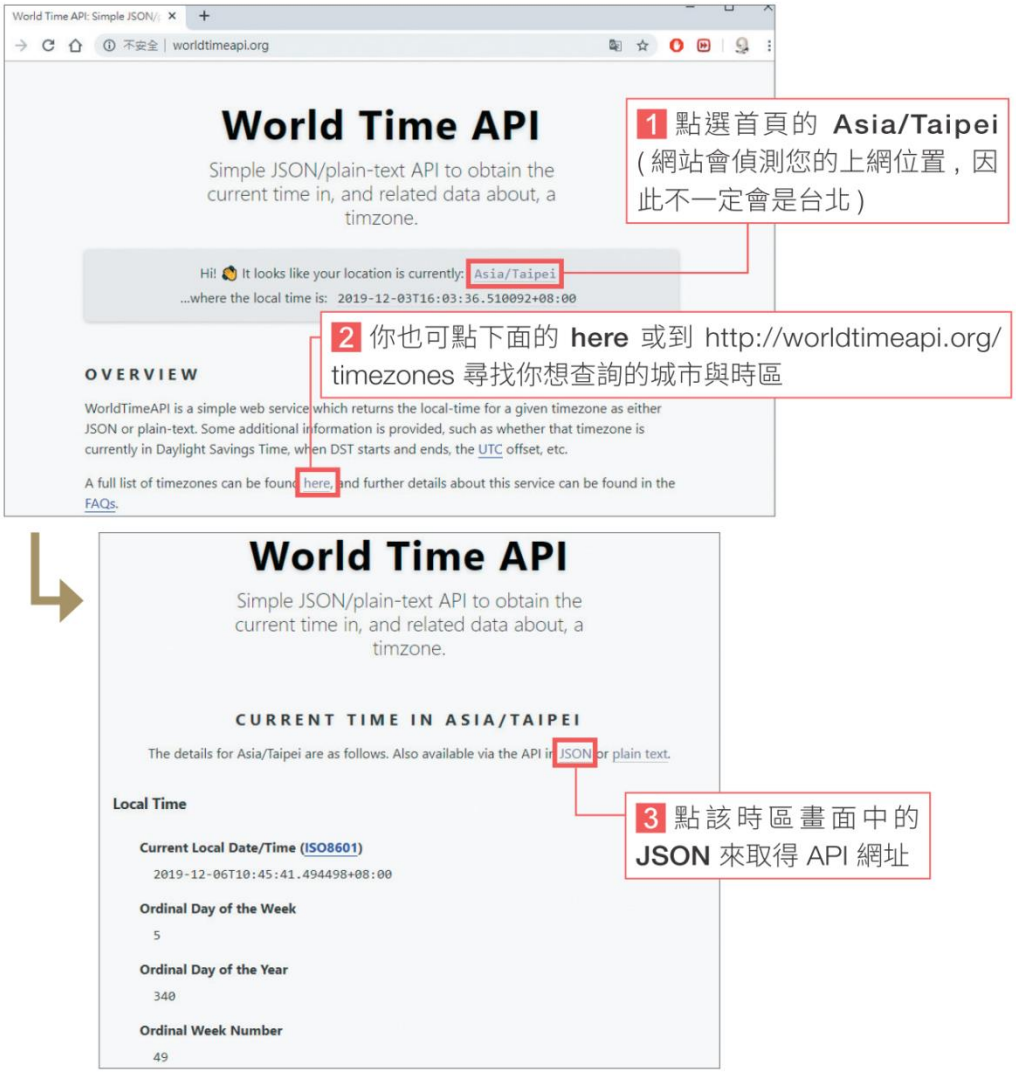
執行程式後，編輯器互動環境窗格會看到類似以下的結果：



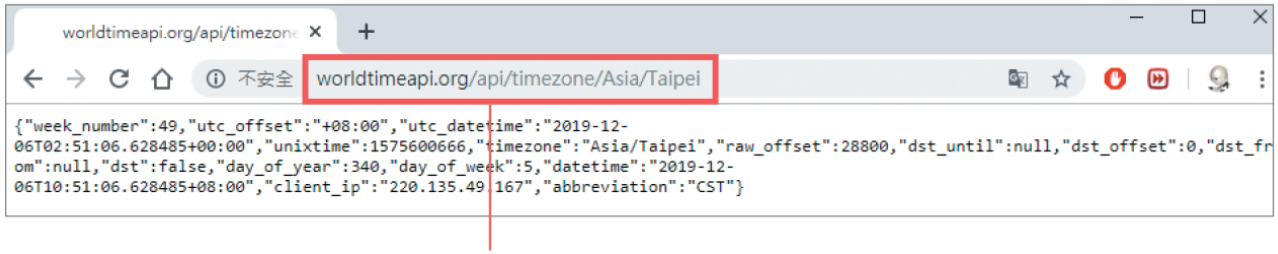

```
連接 WiFi...
已連上
JSON 資料查詢成功:

公司名稱: Taiwan Semiconductor Manufacturing Company Limited (TSM)
股價: 58.25 USD
漲跌幅: -0.61%
市盈率: 26.12
最後交易時間: 2019-12-13 16:01:36
```

8-4 網路對時

- 取得 API



8-4 網路對時



呼叫後傳回的範例結果如下：(只顯示部分內容)

```
{
   "week_number":51,              # 目前為當年第幾周
   "utc_offset":"+08:00",         # 時差 (小時)
   "utc_datetime":"2019-12-16T04:24:47.715637+00:00",
                                  # UTC 標準時間
   "unixtime":1576470287,         # UTC 時間的 Unix 時間戳
   "timezone":"Asia/Taipei",      # 時區/城市名稱
   "raw_offset":28800,            # 時差 (秒)
   "day_of_year":350,             # 目前為當年第幾日
   "day_of_week":1,         # 星期幾 (星期一為 1, 星期日為 7)
   "datetime":"2019-12-16T12:24:47.715637+08:00", # 本地時間
   "abbreviation":"CST"     # 時區簡稱
}
```

8-4 網路對時

- **Lab24**

自動對時鐘	
實驗目的	從 API 取得時間，用來更新 D1 mini 系統時間並顯示在 OLED 模組上。
材料	• D1 mini • OLED 模組
API 網址	http://worldtimeapi.org/api/timezone/Asia/Taipei

- **接線圖**

與 Lab 03 相同。

8-4 網路對時

- **設計原理**

呼叫 API 後從傳回結果中的鍵 datetime 可得類似以下的字串：

原始字串	2019-	12-	06	T	10:	51:	06.	628485	+	08:00
意義	年	月	日	T	時	分	秒	秒小數位	+	時差
位置編號	0~4	5~7	8~9	10	11~13	14~6	17~19	20~25	26	27~31

由於這是一整個字串，Pythop 可以用擷取串列內元素的方式抽取出個別資料。字串擷取方式如下：

```
擷取結果 = 字串[擷取起點:擷取終點後一格]
```

8-4 網路對時

因此我們可以擷取出特定的時間資料：

```
datetime_str = str(parsed["datetime"]) # 取得鍵 datetime 的值
year = int(datetime_str[0:4])   # 抽取年份 (位置 0 到 3, 共 4 位數)
hour = int(datetime_str[11:13]) # 抽取小時 (位置 11 到 12, 共 2 位數)
```

我們會拿這時間來更新控制板系統時間，然後平時讀取系統時間就可以了。

```
from machine import RTC
rtc = RTC() # 建立 RTC 物件
print(rtc.datetime()) # 顯示系統時間
# 設定系統時間
rtc.datetime((年, 月, 日, 星期幾, 時, 分, 秒, 秒小數點))
```

8-4 網路對時

注意 RTC 傳回的時間格式會像是 (2019, 12, 16, 0, 9, 48, 0, 89), 和 Lab 22 的 time.localtime() 傳回的格式不太一樣：

項目編號	0	1	2	3	4	5	6	7
值	2019	12	16	0	9	48	0	89
意義	年	月	日	星期幾 (星期一為 0, 星期日為 6)	時	分	秒	秒小數點

透過 RTC 更新系統時間後，我們就能取得個別的時間資料：

```
year = rtc.datetime()[0]
hour = rtc.datetime()[4]
```

8-4 網路對時

- 程式設計

```
from machine import Pin, I2C, RTC
from ssd1306 import SSD1306_I2C
import network, urequests, utime

ssid = "你的 WiFi 名稱"
pw = "你的 WiFi 密碼"
url = "http://worldtimeapi.org/api/timezone/Asia/Taipei"
web_query_delay = 600000 # 查詢間隔設為 10 分鐘

# 星期幾的對照字典
weekday_name = {0:"Monday", 1:"Tuesday", 2:"Wednesday",
 3:"Thursday", 4:"Friday", 5: "Saturday", 6:"Sunday"}

oled = SSD1306_I2C(128, 64, I2C(scl=Pin(5), sda=Pin(4)))
rtc = RTC()
```

```
print("連接 WiFi...")
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(ssid, pw)
while not wifi.isconnected():
    pass
print("已連上")

# 把原本的系統時間減去查詢間隔時間, 好讓開機後就會先查一次
update_time = utime.ticks_ms() - web_query_delay

while True:

    # 當系統過了間隔時間時
    if utime.ticks_ms() - update_time >= web_query_delay:

        response = urequests.get(url) # 呼叫 API

        if response.status_code == 200:

            parsed = response.json()
            print("JSON 資料查詢成功")
```

ex8-12

```
# 取得鍵 datetime 的內容
datetime_str = str(parsed["datetime"])
# 解析日期與時間資料
year = int(datetime_str[0:4])
month = int(datetime_str[5:7])
day = int(datetime_str[8:10])
hour = int(datetime_str[11:13])
minute = int(datetime_str[14:16])
second = int(datetime_str[17:19])
subsecond = int(round(int(datetime_str[20:26]) /
                          10000))
# 取得星期幾的資料
weekday = int(parsed["day_of_week"]) - 1

# 將時間資料寫入 RTC, 更新系統時間
rtc.datetime((year, month, day, weekday, hour,
              minute, second, subsecond))
print("系統時間已更新:")
print(rtc.datetime())

# 更新查詢時間
update_utime = utime.ticks_ms()
```

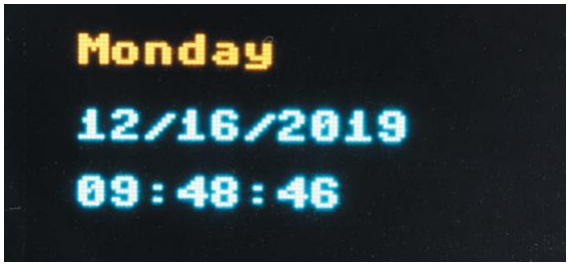
8-4 網路對時

```
    else:
        print("JSON 資料查詢失敗")

# 設定要顯示在 OLED 的文字 (從 RTC 查詢)
weekday_str = " " + weekday_name[rtc.datetime()[3]]
date_str = " {:02}/{:02}/{:4}".format(rtc.datetime()[1],
                  rtc.datetime()[2], rtc.datetime()[0])
time_str = " {:02}:{:02}:{:02}".format(rtc.datetime()[4],
                  rtc.datetime()[5], rtc.datetime()[6])

# 於 OLED 顯示日期時間
oled.fill(0)
oled.text(weekday_str, 0, 8) # 星期幾
oled.text(date_str, 0, 24)   # 日期
oled.text(time_str, 0, 40)   # 時間
oled.show()

utime.sleep(0.1)
```

8-4 網路對時

- **實測**

執行程式後，可看到 OLED 模組
像真實時鐘一樣顯示出時間：



編輯器互動環境窗格則會在更新時間時顯示一點資訊：

```
連接 WiFi...
已連上
JSON 資料查詢成功
系統時間已更新:
(2019, 12, 16, 0, 9, 48, 0, 89)
```