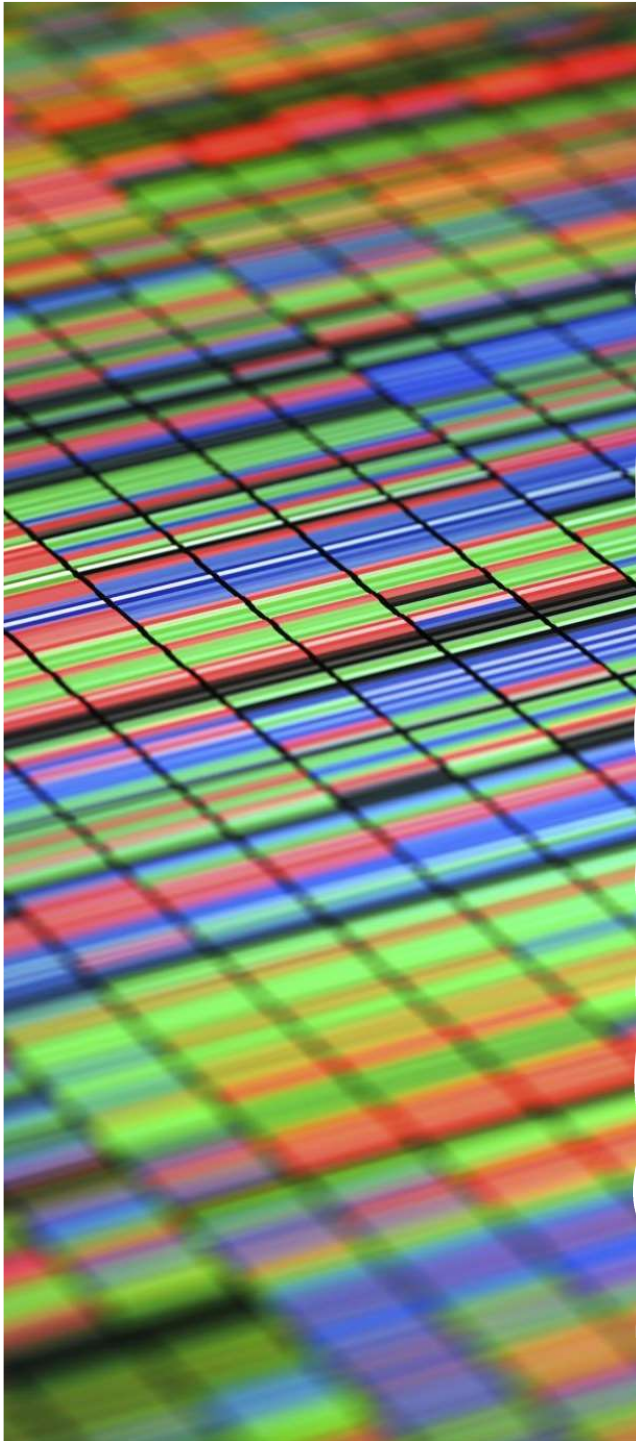


# 動態網站處理

劉維立 編著

【版權所有，不得任意拷貝或引用】



# 從範例學 Python語法 #1

---

# [Python] 綜合實 作練習 #1

📋 練習題目：

請寫一段程式能夠完成以下任務：

1. 抓取一個公開網站
2. 回傳 `status code == 200` 才繼續
3. 取回 HTML 內容並印出前 100 字

📖 語法提示：

```
requests.get(url)
if res.status_code == 200:
    res.text[:100]
```



# [Python] 綜合實作 練習 #1

```
import requests

site_name = "Yahoo"
url = "https://tw.yahoo.com/"
headers = {
    "User-Agent": "Mozilla/5.0"
}
print("正在抓取網站：", site_name)

try:
    res = requests.get(url, headers=headers)
    if res.status_code == 200:
        html = res.text
        print("前 100 字 HTML 原始碼：")
        print(html.strip()[:100])
    else:
        print("連線失敗，狀態碼：",
res.status_code)
except:
    print("請求過程發生錯誤")
```

# [Python] 發送 HTTP 請求

```
import requests  
res = requests.get("https://tw.yahoo.com")  
print(res.status_code)
```

📖 語法重點：

- import 模組
- 發送 HTTP GET 請求
- 印出回應狀態碼

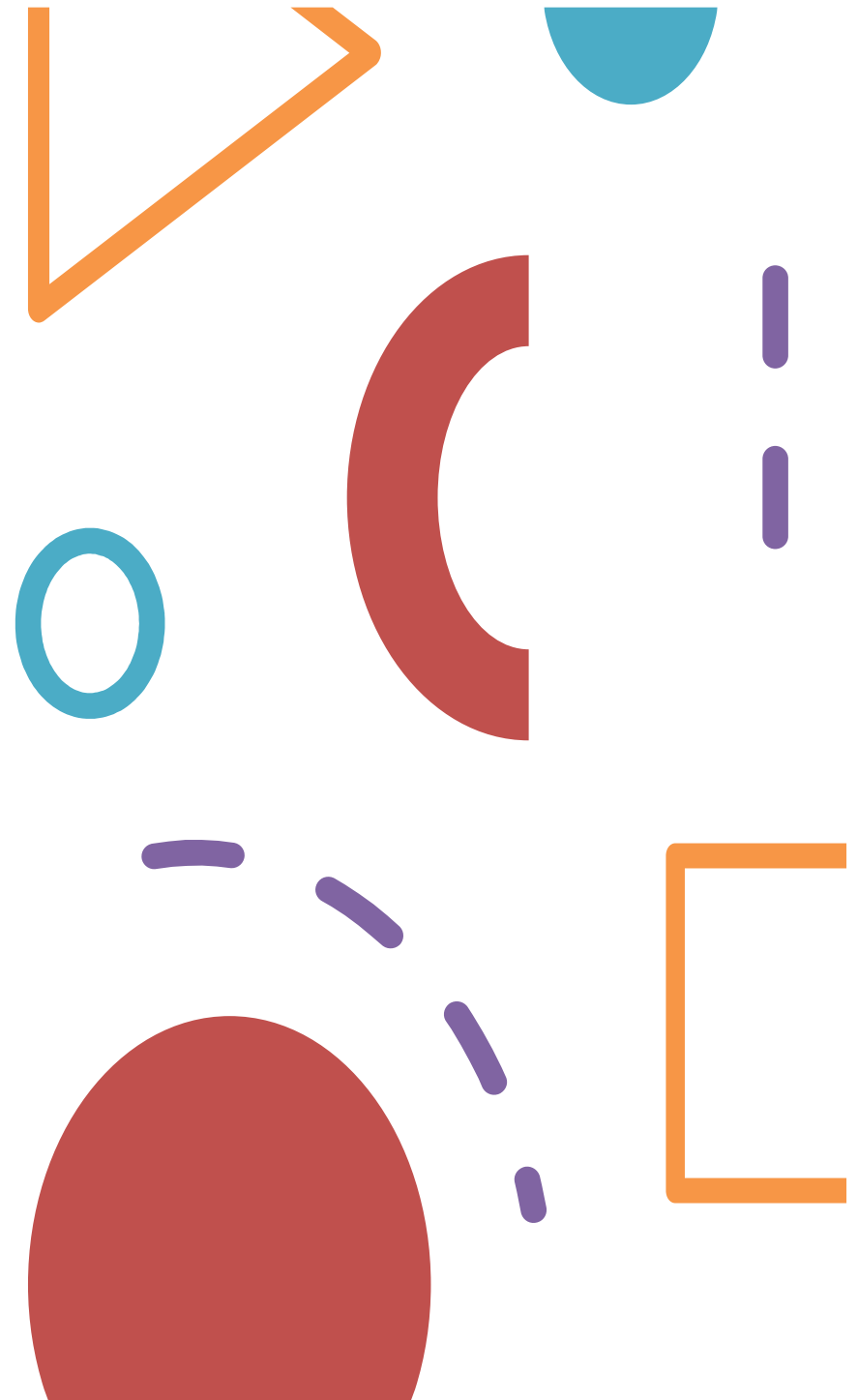
# [Python]

## 變數與型別

```
site_name = "Yahoo"  
count = 10  
is_public = True
```

📖 語法重點：

- 字串 `str`、整數 `int`、布林 `bool`
- 動態型別：不用宣告類型



# [Python] 條件判斷 if/else

```
if res.status_code == 200:  
    print("成功連線")  
else:  
    print("連線失敗")
```

📖 語法重點：

- if 條件：判斷邏輯
- == 等於、else 例外處理

# [Python] Status Code 字串切片 (陣列切片)

## 📖 語法提示

- 檢查 HTTP 回應狀態是否為 200 (成功)  
`res.status_code == 200`
- 回傳網頁的 HTML 文字內容 (字串型別)  
`res.text`
- 使用字串切片：取出 HTML 的前 100 個字元，用於預覽資料  
`res.text[:100]`

## ✅ 補充

- `res.status_code` 常見數值有：
  - 200：成功
  - 404：找不到網頁
  - 403：權限被拒
- `[:100]` 是字串或列表的切片語法，表示從第 0 字元取到第 99 字元 (共 100 個)



# [Python] 字串操作

```
raw = " \n 這是內容 \n"  
cleaned = raw.strip()  
print(cleaned)
```

📖 語法重點：

- `.strip()`：移除空白或換行符號
- `.replace()`、`.lower()` 亦常用



# Python語法介紹

陣列 Array

清單 list

字典 dict

迴圈 for

函式 function

+

0

# [Python] list 基本 用法

```
fruits = ["apple", "banana",  
"cherry"]
```

```
print(fruits[0])      # apple  
print(len(fruits))    # 3
```

📖 語法重點：

- list[index]：用索引取得元素
- append()：新增
- remove()：刪除
- for 迴圈：逐筆處理

# [Python] dict 基本 用法

```
person = {  
    "name": "Alice",  
    "age": 25,  
    "city": "Taipei"  
}
```

```
print(person["name"])      # Alice
```

## 📖 語法重點：

- 用 "key" 存取 value
- dict.get("key")：安全取值
- dict["key"] = value：修改或新增
- .items()：取得所有鍵值對

# [Python] list vs dict 比較

list：有順序、用索引

dict：無順序、用鍵值存取

範例：

list ➤ [1, 2, 3]

dict ➤ {"name": "Amy", "age": 18}

▣ 應用情境：

- list：適合儲存一群人、東西
- dict：適合儲存一個人的屬性、設定

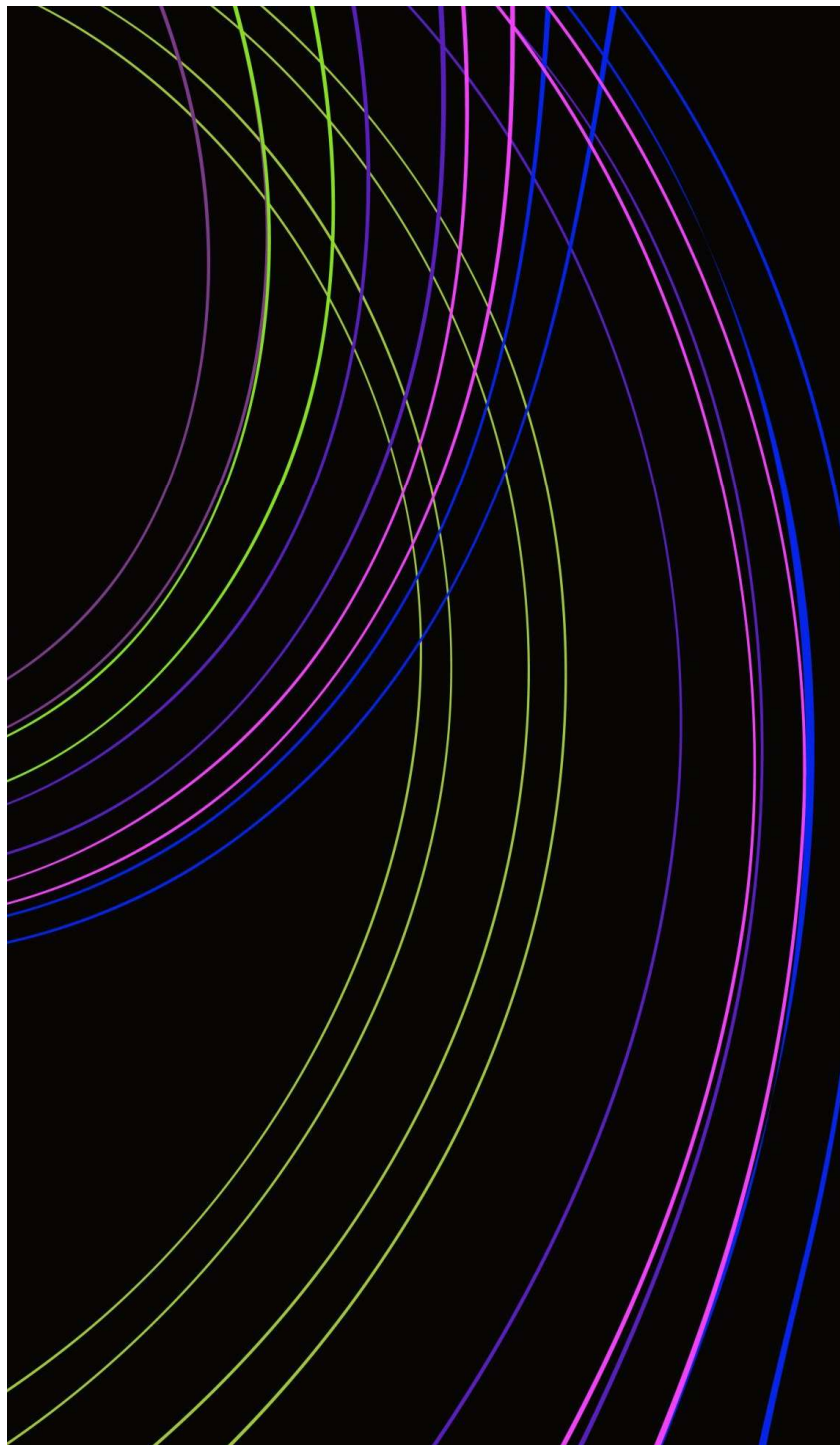
# [Python] 迴圈 for

```
names = ["小明", "小美", "阿強"]
```

```
for name in names:  
    print("哈囉", name + ", 歡迎  
加入 Python 課程!")
```


📖 語法重點：

- for 迴圈：依序處理清單內容





# [Python] list 裡面 放 dict



```
students = [  
    {"name": "小明", "score": 90},  
    {"name": "小華", "score": 85}  
]
```

```
for s in students:  
    print(s["name"], s["score"])
```

📖 應用技巧：

- 資料表格式常用 list + dict 組合
- 可搭配 for 迴圈印出報表

# [Python] 函式 function

```
import requests

def fetch_page(url):
    res = requests.get(url)
    return res.text

html =
fetch_page("https://httpbin.org/html")
print(html[:200])  # 預覽前 200 字
```

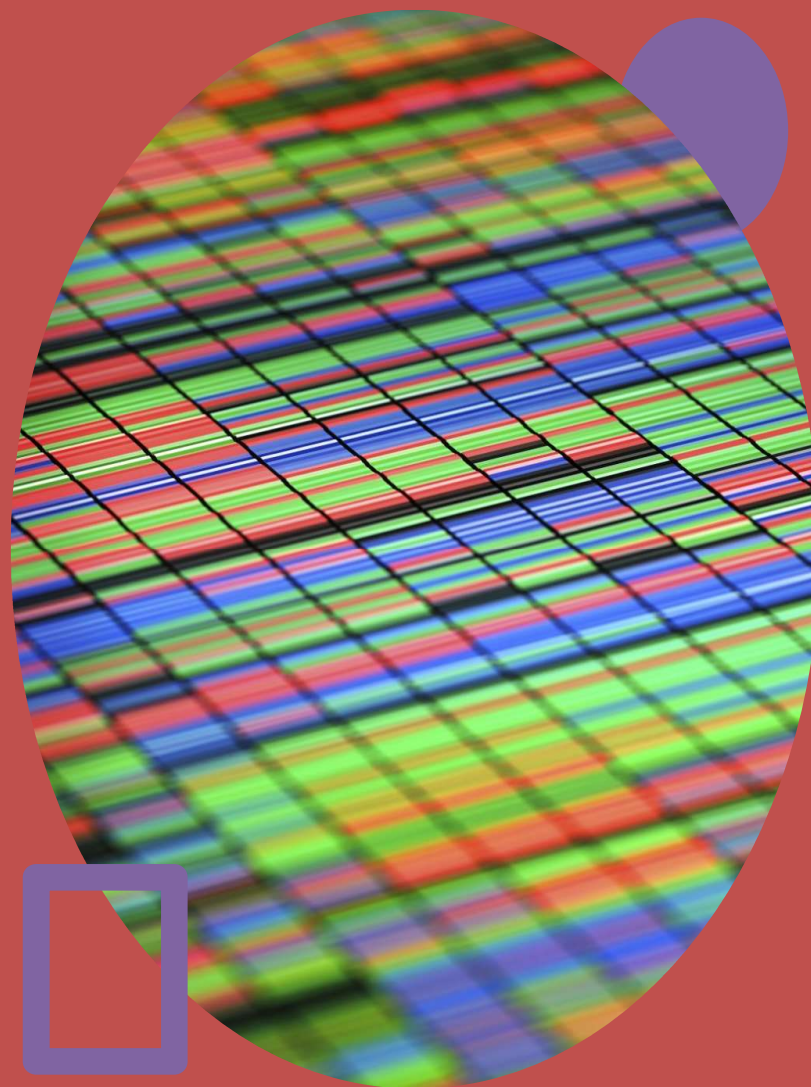
■ 語法重點：

- def 宣告函式
- 傳入參數，return 回傳結果






# 從範例學 Python語法 #2



# 歷史颱風爬蟲範例-使用Playwright

 步驟一：安裝必要模組

打開終端機，執行：

```
pip install playwright pandas beautifulsoup4
```

```
playwright install
```

```
pip install openpyxl
```

# 歷史颱風爬蟲程式

```
from playwright.sync_api import sync_playwright
from bs4 import BeautifulSoup
import pandas as pd

with sync_playwright() as p:
    browser = p.chromium.launch(headless=True)
    page = browser.new_page()
    page.goto("https://rdc28.cwa.gov.tw/TDB/public/warning_typhoon_list/",
timeout=60000)
    page.wait_for_timeout(5000) # 等待 JS 載入
    table_html = page.inner_html("table")
    browser.close()
# 解析 HTML 表格
soup = BeautifulSoup(table_html, "html.parser")
rows = soup.find_all("tr")[1:] # 忽略表頭
data = []
for row in rows:
    cells = [td.get_text(strip=True) for td in row.find_all("td")]
    if cells and len(cells) >= 8:
        data.append(cells[:8])
columns = ["年度", "編號", "名稱", "侵臺類型", "警報期間", "近臺強度", "最低氣壓(hPa)", "最大風速(m/s)"]
df = pd.DataFrame(data, columns=columns)
# 儲存為 Excel
df.to_excel("歷年有發布警報颱風列表.xlsx", index=False)
print("✅ 資料已儲存為 Excel: 歷年有發布警報颱風列表.xlsx")
```

# [Python] 套件匯入

```
from playwright.sync_api import  
sync_playwright  
from bs4 import BeautifulSoup  
import pandas as pd
```

## 📖 說明：

- playwright：控制瀏覽器模擬人類操作
- BeautifulSoup：HTML 解析套件
- pandas：資料表處理、儲存成 Excel



# [Python] 啟動瀏覽器 並打開 網頁

```
with sync_playwright() as p:
    browser =
p.chromium.launch(headless=True)
    page = browser.new_page()

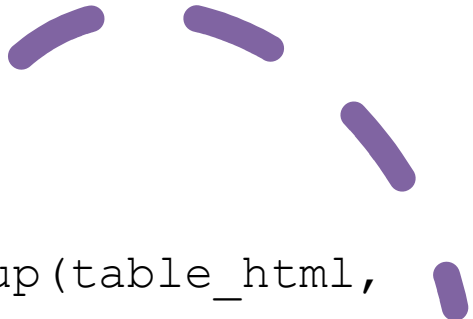
page.goto("https://rdc28.cwa.gov.tw/TDB/
public/warning_typhoon_list/",
timeout=60000)
    page.wait_for_timeout(5000)
    table_html =
page.inner_html("table")
    browser.close()
```

## ■ 說明：

- headless=True：無視覺介面背景執行
- wait\_for\_timeout(ms)：等待 JS 資料載入
- inner\_html("table")：抓取表格 HTML 內容



## [Python] 使用 BeautifulSoup 解析 HTML



```
soup = BeautifulSoup(table_html,  
"html.parser")  
rows = soup.find_all("tr")[1:]
```

▣ 說明：

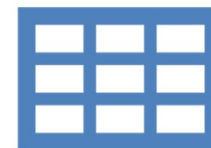
- 建立 HTML 解析器物件
- `find_all("tr")`：找出所有表格列
- `[1:]`：忽略表頭

# [Python] 擷取每列資料

```
data = []
for row in rows:
    cells = [td.get_text(strip=True) for td in
row.find_all("td")]
    if cells and len(cells) >= 8:
        data.append(cells[:8])
```

## 📖 說明：

- `get_text(strip=True)`：取得純文字內容並去除空白
- `append(cells)`：加入到資料清單中



# [Python] 建立 DataFrame 並輸出 Excel

```
columns = ["年度", "編號", "名稱", "侵臺類型", "警報期間", "近臺強度", "最低氣壓(hPa)", "最大風速(m/s)"]
```

```
df = pd.DataFrame(data,  
columns=columns)
```

```
df.to_excel("歷年有發布警報颱風列表.xlsx", index=False)
```

■ 說明：

- `pd.DataFrame(...)`：建立表格格式
- `to_excel(..., index=False)`：輸出為 Excel，不加索引





# [Python] 印出提示 訊息

```
print("✅ 資料已儲存為 Excel :  
歷年有發布警報颱風列表.xlsx")
```


📖 說明：

- print(...)：顯示程式執行結果  
或提示訊息

# [Python] 擷取表格儲存格內容

```
cells = [td.get_text(strip=True) for td in  
          row.find_all("td")]
```

---

 用途：

---

使用 BeautifulSoup 將 HTML 表格中每列 (<tr>) 的所有儲存格 (<td>) 擷取成文字清單。

---

這行是「列表生成式」的寫法，用來快速整理資料。

# [Python] 語法逐段解析

```
cells = [td.get_text(strip=True)  
for td in row.find_all("td")]
```

---

`row.find_all("td")` → 找出所有 `<td>` 標籤 ( 儲存格 )  
→ 一群 `<td>` 元素的清單

---

`for td in ...` → 對清單中每個儲存格做處理

---

`td.get_text(strip=True)` → 擷取文字並去除前後空白

---

`[...]` → 產生一個新的清單 `cells = [欄位1, 欄位2, ...]`

[Python] 等價傳統寫法

```
cells = [td.get_text(strip=True)  
for td in row.find_all("td")]
```

```
cells = []  
for td in row.find_all("td"):  
    text = td.get_text(strip=True)  
    cells.append(text)
```

📖 說明：

這段就是原本那行列表生成式的「完整寫法」，更容易讓初學者理解每步驟

## [Python] HTML 範例與輸出

```
cells = [td.get_text(strip=True) for td in  
          row.find_all("td")]
```

HTML 範例：

```
<tr>  
  <td> 2023 </td>  
  <td> 05 </td>  
  <td> 小犬 </td>  
</tr>
```

執行結果：

```
cells = ['2023', '05', '小犬']
```

# [Python] strip=True 是什麼意思？

```
cells = [td.get_text(strip=True)  
for td in row.find_all("td")]
```

`get_text(strip=True)` → 擷取儲存格文字並清除左右空白

例子：

`<td> 小犬 </td>` → "小犬" (自動清除空白)

若不加 `strip=True`，結果可能是 " 小犬 "，不方便分析

# [Python] 小結：語法重點

```
cells = [td.get_text(strip=True)  
for td in row.find_all("td")]
```

- 📌 `find_all("td")` → 擷取所有儲存格
- 📌 `get_text(strip=True)` → 擷取乾淨文字
- 📌 `[ ... for ... in ... ]` → 建立新清單 ( 列表生成式 )

✅ 實用於 HTML 表格擷取與爬蟲資料整理