Python 程式設計：模組

目錄

1. 基本概念

想像一下，您正在寫一個越來越大的程式。起初您把所有的功能、類別都寫在同一個 .py 檔案裡，但很快這個檔案就變得數千行長，難以閱讀、修改和維護。模組 (Module) 就是為了解決這個問題而誕生的。

在 Python 中，任何一個 .py 檔案都可以被視為一個模組。將相關的函式 (functions)、類別 (classes) 和變數組織在一個獨立的檔案中，以達到程式碼的組織化與重複使用 (reuse)。另外，下面所教的引入方式，是適用於 module 跟 class 歐

一個模組就像一個「工具箱」。您可以建立一個專門放「美食相關」工具的 makefood.py 工具箱，再建立一個專門放「銀行相關」工具的 banks.py 工具箱。當您需要用到某個工具時，只需要「匯入 (import)」對應的工具箱即可，而不需要把所有工具都散落在同一個地方。

2. 建立模組檔案

我們來建立一個名為 makefood.py 的檔案。這個檔案本身就是我們的模組。

```
# makefood.py
def make_icecream(toppings):
    """ 列出製作冰淇淋的配料 """
    print("製作冰淇淋的配料如下：")
    for topping in toppings:
        print("--- ", topping)

# 定義製作飲料的函式
def make_drink(size, drink):
    """ 輸入飲料規格與種類，然後輸出飲料 """
```

```
    print("--- 客製化飲料 ---")
    print("尺寸：", size)
    print("飲料：", drink)
```

3. 在其他程式中應用自己建立的函式

3.1 方法一：導入整個模組 import 模組名稱

```
├── makefood.py
└── main.py
```

```
# main.py
import makefood

# 呼叫函式時，必須在前面加上 "模組名稱."
makefood.make_icecream(['草莓醬', 'OREO 餅乾', '巧克力碎片'])
makefood.make_drink('large', 'coke')
```

這種方式的好處是，程式碼非常清楚，任何人一看就知道 make_icecream 這個函式是來自 makefood 模組，不會搞混。

3.2 方法二：從模組中導入特定函式 from ... import ...

```
# main.py
from makefood import make_icecream, make_drink

# 因為已經明確導入，可以直接呼叫函式名稱
make_icecream(['草莓醬', 'OREO 餅乾', '巧克力碎片'])
make_drink('large', 'coke')
```

3.3 方法三：使用 as 給模組或函式取一個「別名」

有時候模組名稱太長，或是導入的函式名稱與現有程式碼衝突，我們可以用 as 來取一個好記又簡短的別名。

- 語法 (模組別名)：import 模組名稱 as 別名
- 語法 (函式別名)：from 模組名稱 import 函式名稱 as 別名

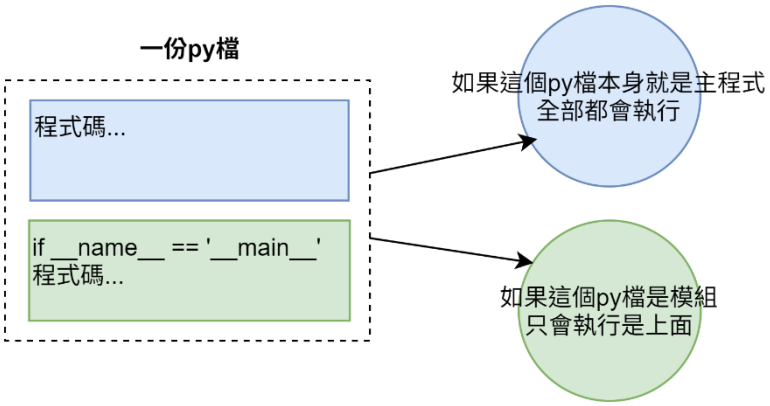
```
# main.py
import makefood as m  # 將 makefood 取一個簡短的別名 m

m.make_icecream(['草莓醬'])
m.make_drink('small', 'tea')
```

4. if __name__ == '__main__'

首先，我們需要理解 Python 直譯器在執行任何程式碼檔案（.py 檔案）時，會自動為該模組定義一個特殊的內建變數 __name__。

- 當你直接執行一個 .py 檔案時（例如，在終端機中輸入 python your_script.py），該檔案會被視為主程式。此時，Python 會將該模組的 __name__ 變數賦予字串 '__main__'。
- 當你從另一個 .py 檔案中匯入（import）這個檔案作為一個模組時，Python 會將該模組的 __name__ 變數賦予模組本身的名稱，也就是檔名（不含 .py 副檔名）。例如，如果你匯入一個名為 my_module.py 的檔案，其 __name__ 變數的值將是 'my_module'。



這邊以一個簡單的數學工具模組

```python
# math_tools.py

def calculate_area(length, width):
    """
    計算長方形的面積
    """
    return length * width

# 在這裡，我們用 if __name__ == '__main__': 來測試我們的函式
if __name__ == '__main__':
    print("--- 正在以主程式模式執行 ---")

    # 這裡的程式碼只在直接執行 math_tools.py 時才會運作
    length = 5
    width = 8
    area = calculate_area(length, width)
    print(f"長度 {length}，寬度 {width} 的面積為: {area}")
```

4.1 情境一：直接執行 math_tools.py

```
(base) PS D:\github\python-tutor> python math_tools.py
--- 正在以主程式模式執行 ---
長度 5，寬度 8 的面積為: 40
```

4.2 情境二：在另一個檔案中匯入 math_tools.py

假設我們有另一個檔案 main_program.py，需要使用 calculate_area 函式。

```
├── math_tools.py
└── main_program.py
```

```python
# main_program.py
import math_tools

print("--- 正在匯入 math_tools 模組 ---")
my_area = math_tools.calculate_area(10, 20)
print(f"從 main_program 計算的面積為: {my_area}")
```

```
(base) PS D:\github\python-tutor> python math_tools.py
--- 正在匯入 math_tools 模組 ---
從 main_program 計算的面積為: 200
```