Modularization
模組化設計

Python 編程課程 第 3 課

- 模組化設計 Modularization
- 匯入 import
- 套件 Package

熱身：生活比喻——筆袋

當你在上課時，你想**記錄筆記**（原子筆）
當你在考試時，你想**作答試卷**（鉛筆、橡皮擦）
當你在講解時，你想**畫圖**（馬克筆、間尺）
……

這些事情需要不同文具，它們都在你的**筆袋**裡

換言之：
帶一個**筆袋**可以幫助你完成不同工作時
調用不同文具，更簡單、更快速的完成工作。

模組的概念 - 工具箱

模組就取自於一個的概念。

因應**功能性**的不同，工具箱也有不同種類

例如：

- **筆袋**是放**文具**的工具箱
- **零件箱**是放**維修**用的工具箱
- **書架**是放**書**的工具箱。

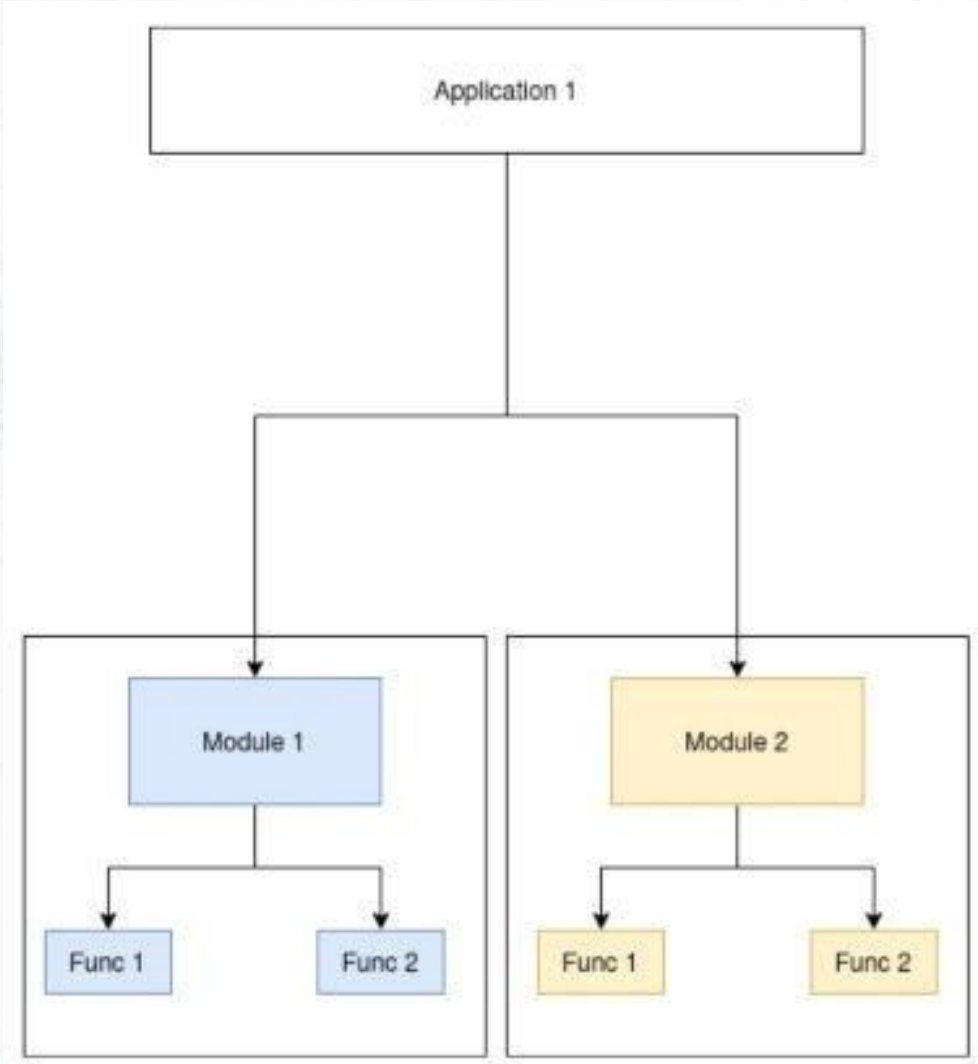
程式中的模組

將關聯性較高的程式碼抽出來放在一個模組(Module)，主程式再匯入所需模組去調用功能。

你會選擇.....

a. **分工合作**：一個負責設計，一個施工，一個檢測

b. **一個人負責所有** 設計、施工、檢測



重用代碼

Python 的一個很大的優勢是**大用戶基數**，
因此有**大量現成工具（代碼）**：

Python 標準函數庫／模組	Python 套件
• 預設已經安裝在 Python	• 不是 Python 自帶
• 超過 200 個自帶模組 Module	• 其他人開發

匯入 - import

使用**關鍵字 import** 可以匯入 Python 的程式碼(.py)當作模組使用，繼而調用其功能(函數)。

匯入方法 有兩種:

- 匯入**整個模組**:import <模組名稱>(模組名稱不包含「.py」)
- 匯入模組中的**某一段程式**:from <模組名稱> import <方法>

例子：random模組

random模組：

- 生成隨機亂數
- 打亂列表排序
- 隨機抽取項目

使用**關鍵字 import**
匯入模組

程式碼

```
import random

print(random.randint(1, 10))
```

.randint(1, 10)：
生成1到10之間（包括1和10）的任意整數

輸出

```
8
```

程式碼

```
from random import choices

name = choices(["Sam", "Ben", "Rocky"])
print(name)
```

從 random 模組指定匯入 choices
choices([“Sam”, “Ben”, “Rocky”])：
從列表中隨機選擇一個元素

輸出

```
['Rocky']
```

別名 - as

使用**關鍵字 as**（如同）去賦予「別名」給一個模組／函數

import <模組> as <別名>

如果匯入的模組名稱和原本程式碼相同

程式碼

```
import random

num = random.randint(1, 10))
print(num)
```

輸出

```
8
```

程式碼

```
import random as r

num = r.randint(1, 10))
print(num)
```

輸出

```
8
```

堂課 - 請打開Thonny一起做

匯入 random 模組

粉紅色斜體:題目
黑色正常字體:提供的程式碼
紅色底線:請填入程式碼

程式碼

```
# 匯入 random 模組
_______ random
# 建立列表變量 names，初始賦值如下：
names = ["alan", "ben", "chris", "danny", "eric"]


# 使用 .choices()從 names 列表抽出一個名字賦 值給 player
player = _________._______name_
# 使用 .randint()隨機生成數字（1-10），賦值給 rep
rep = _________._____________


# 列印
print(f"{player} please do push-up for {rep} times !")
```

輸出

```
alan please do push-up for 9 times !
```

匯入 math 模組

粉紅色斜體:題目
黑色正常字體:提供的程式碼
紅色底線:請填入程式碼

程式碼

```
# 匯入 math 模組
_______ math

# 建立變量 num，用 input() 賦值
num = _______

# 使用 math.sqrt(num) 計算 num 的開方值，賦值給變量 sqrt_value
sqrt_value = _____.sqrt(____)

# 列印
print(f"Square root of {num} is {_________}")
```

輸出

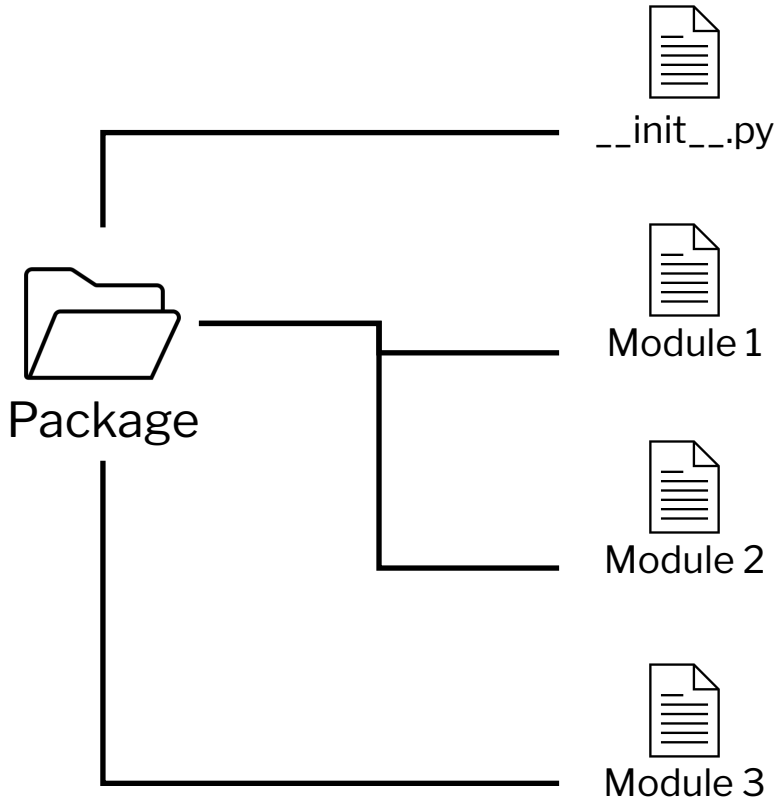
```
9
Square root of 9 is 3
```

套件(Package) vs 模組(Module)

有時程式碼規模太大，不想塞在同一個檔案中

就可以用**套件(package)**的形式**整合**多個目的類近的**模組**

檔案結構



總結

```
# 匯入模組 module
import module
module.say_morning('LP')


# 匯入模組中的特定函數
from module import say_morning
say_morning('LP')


# 別名
import random as r
```

檔案結構