

Python程式設計

模組(Modules)

高師大數學系

葉倚任

模組(Modules)



實際應用程式的程式碼數量很多，
只用一個 .py 原始檔來寫，程式碼管理
會很混亂。

管理程式碼方式

- 依職責將程式碼劃分在不同的模組
- 相近或者彼此輔助的模組用套件來管理

使用模組範例

- 每個 .py 檔案本身就是一個模組

用 **import** 重用先前撰寫好的.py 檔案

撰寫了一個 mymath.py

```
mymath.py
1  pi = 3.14159
2  e = 2.71828
3  |
```

```
mydemo.py
1  import mymath
2
3  print(mymath.pi)
4
```

```
YehYi-Rens-MacBook-Pro:jupyter_ex Ian$ python3 mydemo.py
3.14159
```

再來看一個範例

一樣之前撰寫了一個 myhello.py

myhello.py

```
1 name = input('Your name: ')
2 print('Hi!', name)
3
```

- 每個 .py 檔的主檔名就是模組名稱
- 需用 import 關鍵字匯入模組名稱
- 取用模組中定義的名稱，需要加一個「.」
- 被 import 模組中程式碼會被執行，接著才執行 import 之後的程式碼

在 mydemo 中 import myhello

mydemo.py

```
1 import myhello
2
3 print('How are you doing?', myhello.name)
4
```

```
YehYi-Rens-MacBook-Pro:jupyter_ex Ian$ python3 mydemo.py
Your name:Ian
Hello~ Ian
How are you doing? Ian
```

來看一下 Python 中的 math 模組

mydemo.py

```
1 import math
2
3 print(math.pi)
4 print(math.sin(math.pi/2))
5
```

- import 某個模組 python 直譯器會為它建立一個 module 實例，並建立一個模組名稱來參考它。
- dir(math) 是查詢 math 名稱參考的實例上，有哪些屬性(attribute)名稱可以取用。

```
YehYi-Rens-MacBook-Pro:jupyter_ex Ian$ python3 mydemo.py
3.141592653589793
1.0
```

想要知道一個模組中有哪些名稱，可以在 python 的環境中使用 dir() 函式。

```
>>> dir(math)
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

正式寫一個小模組：bank

bank.py

```
1 def account(name, number, balance):
2     return {'name': name, 'number': number, 'balance': balance}
3
4 def deposit(acct, amount):
5     if amount <= 0:
6         print('存款金額不得為負')
7     else:
8         acct['balance'] += amount
9
10 def withdraw(acct, amount):
11     if amount > acct['balance']:
12         print('餘額不足')
13     else:
14         acct['balance'] -= amount
15
16 def desc(acct):
17     return 'Account:' + str(acct)
18
```

← 建立帳戶

← 存款

← 提款

← 描述帳戶狀態

import bank

mydemo.py

```
1 import bank
2
3 acct = bank.account('Justin', '123-4567', 1000)
4 bank.deposit(acct, 500)
5 bank.withdraw(acct, 200)
6
7 print(bank.desc(acct))
8
```

```
YehYi-Rens-MacBook-Pro:jupyter_ex Ian$ python3 mydemo.py
Account: {'name': 'Justin', 'number': '123-4567', 'balance': 1300}
```

模組補充說明

- 如果有多個模組需要 import，除了逐行 import 之外，也可以在單一行中使用逗號「,」來區隔模組
 - `import random, math`
- 在 `__builtins__` 模組中的函式、類別等名稱，都可以不用 import 直接取用，而且不用加上模組名稱作為前置

```
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'ValueError', 'Warning', 'ZeroDivisionError']
```

練習

- 設計一個小模組
 - 功能：為創造角色，包含名字、性別、血量、魔法能量
 - 功能：增加血量與減少血量
 - 功能：增加魔法能量與減少魔法能量
 - 功能：描述目前角色狀態

使用套件管理模組

- 模組也應該分門別類加以放置，也就是套件(package)

A 部門寫了一個 myhello.py

```
myhello.py
1 name = input('Your name:')
2 print('Hello~',mystr)
3
```

B 部門寫了一個 myhello.py

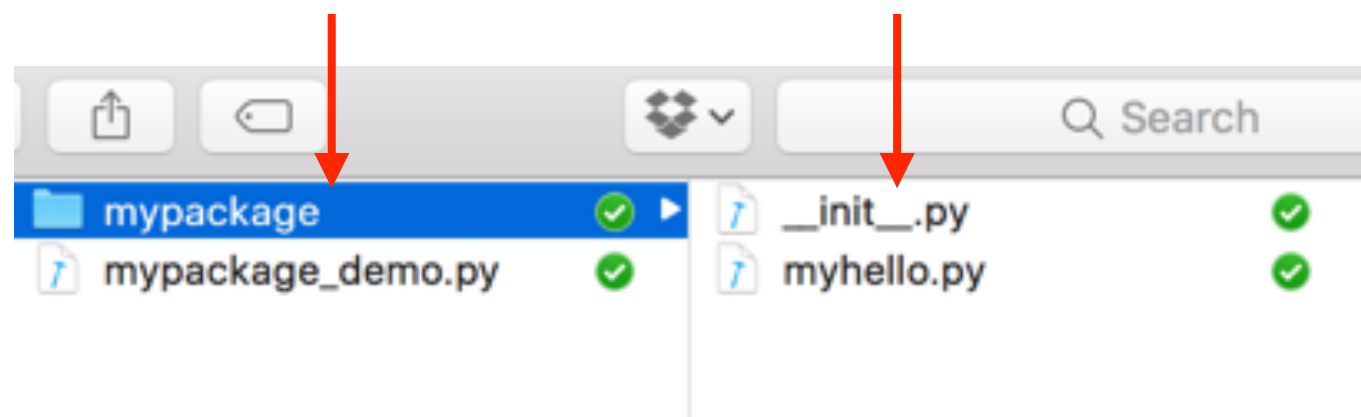
```
myhello.py
1 print('Hi~ World!')
2
```

名字一樣但是功能不一樣，放在一起會有檔案覆蓋的問題
使用套件來管理

套件建立方式

1. 建立一個目錄夾

2. 在目錄夾裡建立一個 `__init__.py` 檔



⇒ mypackage 套件

```
mypackage_demo.py
1  import mypackage.myhello
2  print('How are you doing?', mypackage.myhello.name)
3
```

```
YehYi-Rens-MacBook-Pro:hello_prj Ian$ python3 mypackage_demo.py
Your name: Ian
Hi! Ian
How are you doing? Ian
```

套件名稱會成為名稱空間的一部份

練習

- 將前面 A 部門與 B 部門的程式整理成兩的套件吧
 - 可以命名為 DepA 與 DepB

import as 與 from import

mypackage_demo.py

```
1 import mypackage.myhello
2 print('How are you doing?', mypackage.myhello.name)
3
```

名字很長！

import as 重新命名

可以用 import as 或者 from import 來解決

mypackage_demo.py

```
1 import mypackage.myhello as hello
2 print('How are you doing?', hello.name)
3
```

import as 與 from import

可以使用 from import 直接將模組中的指定名稱匯入

mypackage_demo.py

```
1 from mypackage.myhello import name
2 print('How are you doing?', name)
3
```

from import *

- 使用 from import 時，若結尾是 * 則會將匯入模組中的所有變數

```
mypackage_demo.py
1  from math import *
2
3  print('PI is',pi)
4  print('e is',e)
5
```

小心使用，很容易造成名稱衝突

```
YehYi-Rens-MacBook-Pro:hello_prj Ian$ python3 mypackage_demo.py
PI is 3.141592653589793
e is 2.718281828459045
```

限制 from import *

- 如果有些變數，並不想被 from import * 建立同名變數，可以用底線作為開頭

```
mymath.py  
1 pi = 3.14159  
2 e = 2.71828  
3 _myvar = 30  
4
```

用底線當開頭

```
mydemo.py  
1 from mymath import *  
2  
3 print('PI is',pi)  
4 print('e is',e)  
5 print('myvar is',_myvar)  
6
```

```
YehYi-Rens-MacBook-Pro:jupyter_ex Ian$ python3 mydemo.py  
PI is 3.14159  
e is 2.71828  
Traceback (most recent call last):  
  File "mydemo.py", line 5, in <module>  
    print('myvar is',_myvar)  
NameError: name '_myvar' is not defined
```


__all__ 清單

- 定義 __all__ 清單，使用字串列出可被 from import * 的名稱:

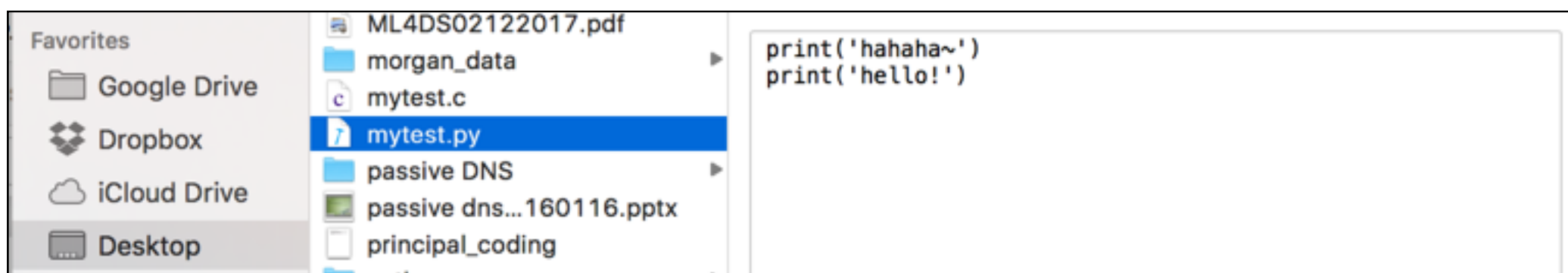
```
mymath.py
1  __all__ = ['pi','e']
2
3  pi = 3.14159
4  e = 2.71828
5  z = 9527
6
```

```
mydemo.py
1  from mymath import *
2
3  print('PI is',pi)
4  print('e is',e)
5  print('z is',z)
6
```

```
YehYi-Rens-MacBook-Pro:jupyter_ex Ian$ python3 mydemo.py
PI is 3.14159
e is 2.71828
Traceback (most recent call last):
  File "mydemo.py", line 5, in <module>
    print('z is',z)
NameError: name 'z' is not defined
```

設定 PYTHONPATH

- 若想用寫好的模組，一定要將 .py 檔案放到目前的工作資料夾嗎？



```
import mytest
```

```
-----  
ImportError                                Traceback (most recent call last)  
<ipython-input-3-39d1a25523aa> in <module>()  
----> 1 import mytest  
  
ImportError: No module named 'mytest'
```

不用，可以利用 `sys.path` 來解決這個問題!

利用 sys.path 查尋目前模組尋找路徑

sys.path 為尋找模組路徑之清單

```
import sys  
print(sys.path)
```

```
['', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python35.zip', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/plat-darwin', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/lib-dynload', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages/IPython/extensions', '/Users/Ian/.ipython']
```

sys.path 為一 list ，裡面為路徑清單

利用 sys.path 新增路徑

既然 sys.path 為一個 list，那就可以用 append 新增路徑囉！

```
import sys
sys.path.append( '/Users/Ian/Desktop' )
import mytest
```

```
hahaha~
hello!
```

```
print(sys.path)
```

```
['', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python35.zip', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/plat-darwin', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/lib-dynload', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages', '/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages/IPython/extensions', '/Users/Ian/.ipython', '/Users/Ian/Desktop']
```