3.8. Module

Fig. 3.8.1 Photo by Maik Kleinert on Unsplash

Note

Outline

Note

Roadmap

1. This topic: Module

Module								
import Statement	from Statement	as Statement	Module Structure	Common Module				
				math	sys	date	codec	_thread
					os			
				random	shutil	time		threading
					subprocess	datatime		

2. Course: Python 1
3. Subject: Programming
4. Field
    a. Software Engineering (SE)
    b. Computer Science and Information Engineering (CSIE)
    c. Electrical/Electronics Engineering (EE)

3.8.1. Introduction

1. 其實模組就是.py 檔，之前普羅所做過的練習，儲存起來的.py檔都是模組。
2. 而當普羅需要使用到這些模組的時候，都可以藉由import 指令來匯入現在所編輯的程式中使用。
3. 至於菲絲恩之前所import 的模組，多是Python內建的模組，以方便大部分程式設計人員所使用，而不用一再重新撰寫。
4. 具體的說，模組就是保存了程式碼的.py 檔，裡面可能定義了函數、類別、變數等內容。

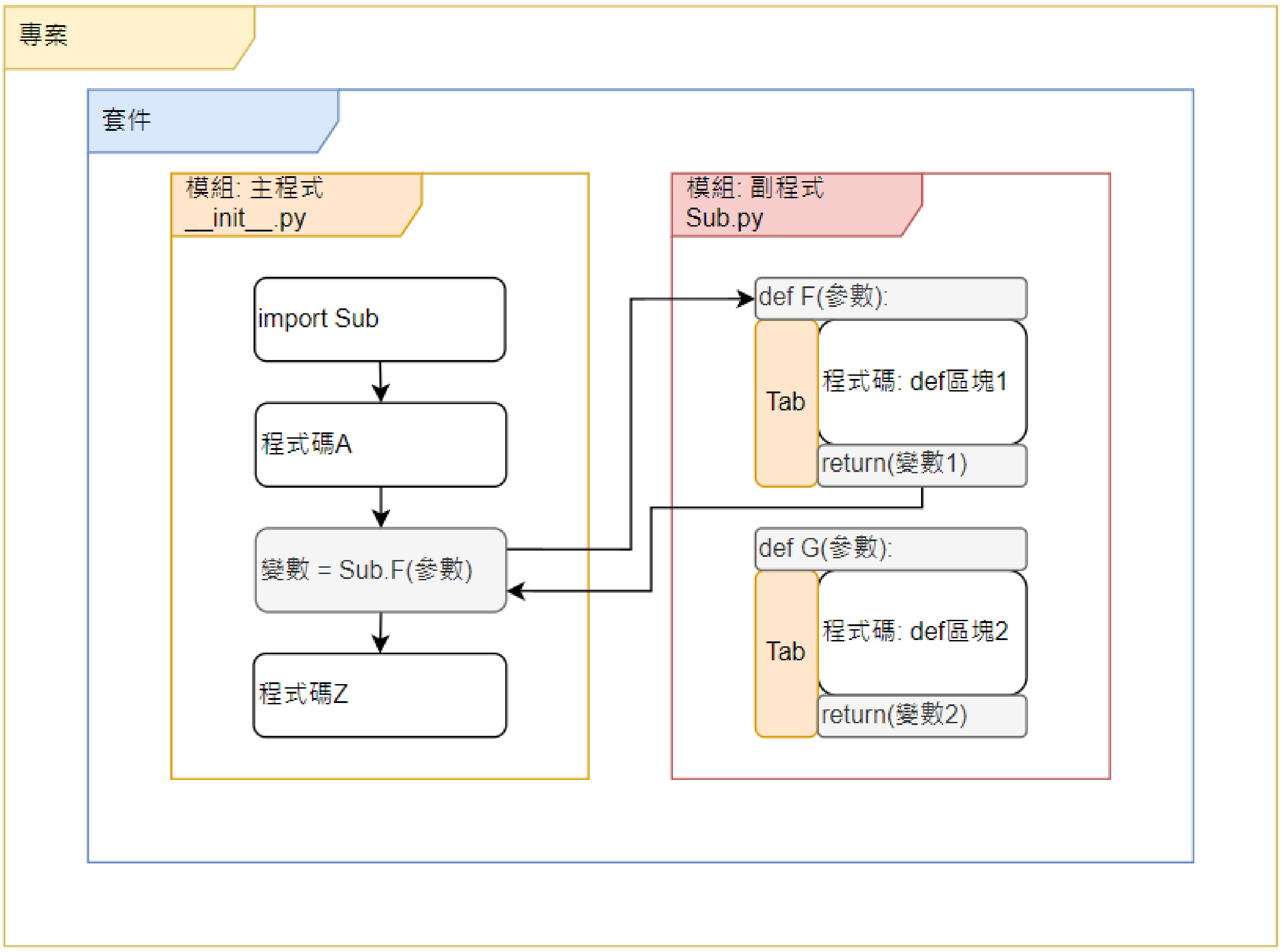
3.8.1.1. Syntax

1. 要匯入模組，則有兩個指令可以加以使用：import 跟from…import，其語法分別是

```
import module1[,module2[,... module]
```

```
from modname import name1[,name2[, ... nameN]]
```

2. 由上可知，import 可以一次匯入多個模組， 中間以「，」分隔。而from…import 則是更精確的描述想要使用的是哪個模組裡面的哪個函數。
3. 另外，在匯入的過程中，如果在這個Python 檔案的路徑底下找不到你所指定的Python 檔案，那Python 便會到sys.path 裡面加以尋找。
4. Module Relationship



3.8.2. Examples

3.8.2.1. Ex1: import

1. Source

```
import sys          # 匯入sys 模組

print(sys.path)     # 印出sys.ptah
```

2. Output

```
[‘C:\\examples\\Ch9’, ‘C:\\Python34\\Lib\\idlelib’, C:\\Windows\\system32\\python34.zip‘,
‘C:\\Python34\\DLLs‘, ‘C:\\Python34\\lib', ‘C:\\Python34', ‘C:\\Python34\\lib\\site-
packages']
```

3. 由上例可知，透過這樣簡單的方式便可知道Python 會在哪裡系統的路徑下找尋要import 的模組檔案。
4. 如果嫌import 的模組名稱太長不好打，或是不好聽而想要換名稱，則可使用import…as 這個指令。
5. 這個指令會先import 你指定的模組進來，但是在這支Python 程式裡面，所匯入的模組名稱則會變成as 後面的名稱，如下例所示：

3.8.2.2. Ex2: as

1. Source

```python
import math as ma    # import math模組，並將其重新命名為ma

print(ma.pi)         # 輸出ma模組裡的變數pi
print(math.pi)       # 輸出math模組裡的變數pi
```

2. Output

```
3.141592653589793
3.141592653589793
```

3. 使用import...as 並不會改變這個模組真正的名字，僅是在被匯入的這支程式裡面被更改名稱而已，這一點得特別注意。
4. 畢竟全世界都在用的模組名稱，如果可以讓每個人自由重新命名，那其他人還怎麼使用呢？

3.8.2.3. Ex3: from import

1. Source

Listing 3.8.2.3.1 /src/Module/Sqrt/__init__.py: Case 6a

```python
from math import sqrt

print(sqrt(3))
```

2. Output

```
1.7320508075688772
```

3.8.2.4. Ex4: Get Date/Time

1. 聽到比賽的獎品是能夠有一次機會去實現任何一個願望，普羅迫不及待能藉由這次所贏的許願機會盡快回到原來的世界。
2. 於是，當菲絲恩問普羅要不要用看看她魔法書中現成的咒語時，普羅二話不說便答應了！
3. Codes

   a. __init__.py: main program

   Listing 3.8.2.4.1 /src/Module/GetDateTime/__init__.py

```python
import ex09_05

print("現在時間: " + ex09_05.getTime('%Y-%m-%d %H:%M:%S'))
```

   b. ex09_05.py: getTime()

   Listing 3.8.2.4.2 /src/Module/GetDateTime/ex09_05.py

```python
import time

def getTime(sTimeFormat):
    dtTimeStamp = time.time()
    return(time.strftime(sTimeFormat, time.localtime(dtTimeStamp)))
```

4. Output

```
現在時間: 2014-06-21 17:43:54
```

3.8.2.5. Ex5a: Modularization

1. Original Codes

    a. We have ever write out our triangle cases in [Stars.py]

    b. Now we use [Case 6a] as our example.

Code **Output**

Listing 3.8.2.5.1 /src/Module/Stars/Ex6a.py

```python
'''
author: cph
since:  20170104
'''
def CaseSep(sCase, sMsg):
    print("----------------------------")
    print("Case %s: %s" %(sCase, sMsg))
    print("----------------------------")

if __name__ == '__main__':
    iMax = int(input("Please give me a number: "))
    CaseSep("6a", "while: string, step up, a slope")
    iCnt = 1
    while (iCnt <= iMax):
        print(" " * (iCnt - 1) + "*")
        iCnt += 1

```

2. Module Codes: Modularization

Code: i.py: for input **Output: i.py: for input** **Code: p.py: for process** **Output: p.py: for input** **Code: o.py: for output** **Output: o.py: for input** **Code: __init__.py: main program** **Output: for whole app**

Listing 3.8.2.5.2 /src/Module/Modularization/i.py: input

```python
'''
author: cph
since:  20170104
'''
def InTitle():
    iMax = int(input("Please give me a number: "))
    return(iMax)

if __name__ == '__main__':
    print(InTitle())
```

3.8.2.6. Ex5b: Modularization: Speed

1. Original Codes

a. We have ever write out our triangle cases in [Stars.py]

b. Now we use [Case 6a] as our example.

Code Output

```
Ex6a in One: 0.065928899999956242
```

2. Module Codes: Modularization

Code: i.py: for input | Output: i.py: for input | Code: p.py: for process | Output: p.py: for input | Code: o.py: for output | Output: o.py: for input | Code: __init__.py: main program | Output: for whole app

Listing 3.8.2.6.6 Modularization: output

```
__init__.py: 0.19895209999958752
```

3. In a summary, functions in one file is faster than we call functions from multiple modules.

3.8.2.7. Ex6a: __main__

1. In Python, __main__ is the name of the scope in which top-level code executes.
2. A Python file runs as a script when invoked directly rather than imported.
3. __main__ can be used to run code only when invoked as a script

    a. On import, name is set to the module name

    b. On direct execution, name is __main__

4. This allows

    a. Importing modules without running main().

    b. Writing scripts that can be both imported and executed.

    c. Protecting code that should only run at startup

5. Code+Output

Code Output

Listing 3.8.2.7.1 /src/Module/Main/main.py

```
'''
author: CPH
since:  20230805
'''
def printit():
    print("這一行被呼叫時，才會會被列印出來...")

if __name__ == '__main__':
    printit()
    print("這一行只會在直接執行時被列印出來，但若函式呼叫就不會喔...")
```

6. It can be imported and run as a script.

Code **Output**

Listing 3.8.2.7.2 /src/Module/Main/caller.py

```
'''
author: CPH
since:  20230805
'''
import main

if __name__ == '__main__':
    main.printit()
```

7. Summary

    a. __main__ is the name of the scope when a .py file runs as a script.

    b. It can be used to run initialization code safely.

Important

__name__

1. The __name__ value is a built-in variable in Python that is used to determine if a Python script is being run as the main program or if it is being imported as a module into another script.
2. We can find and use the __name__ variable in our Python code.

> **See also**
>
> Python Doc v3
>
> a. 5.4.4. Import-related module attributes, Python Documentation. [Web]

3.8.2.8. Ex6b: __main__

1. Code+Output

Code **Output**

Listing 3.8.2.8.1 /src/Module/Main2/main.py

```python
'''
author: CPH
since:  20230805
'''

if __name__ == '__main__':
    print(f'__name__: {__name__}')

    lMain = [
        "__name__ == '__main__'",
        "__name__ == '__main'",
        "__name__ == 'main__'",
        "__name__ == 'main'",
        ]

    for m in lMain:
        print(f"{m} is {eval(m)}...")

```

3.8.2.9. Ex6c: __main__

1. Code+Output

Code **Output**

Listing 3.8.2.9.1 /src/Module/Main3/main.py

```python
'''
author: CPH
since:  20230805
'''

if __name__ == '__main__':
    lsModAtt = [
        '__name__',
        '__loader__',
        '__package__',
        '__spec__',
        '__path__',
        '__file__',
        '__cached__',
        ]

    for a in lsModAtt:
        if (a in globals()):
            print(f"{a} is global variable and value is {globals()[a]}...")
        elif (a in locals()):
            print(f"{a} is local variable and value is {locals()[a]}...")
        else:
            print(f"{a} is NOT Found...")
```

Important

Summary

1. Here is a summary of built-in Python functions related to modules.

Build-in	Description
package	Package containing module
import	Imports a module into current namespace
from	Imports specific objects from a module
as	Renames a module or object on import
def	Defines a function in a module
if __name__ == '__main__'	Runs a module as a script
class	Defines a class in a module
help()	Displays docs for module, function, class
dir()	Lists names defined in a module
file	Path to module file
name	Module name, 'main' if run as script
sys.path	Search path for imported modules
sys.modules	Loaded module objects
importlib	Programmatic import interface

2. These built-ins allow creating, importing, introspecting and executing modules in Python.
3. Good modular design through functions, classes and controlled imports is important for building robust reusable code.

3.8.3. Homework

3.8.3.1. Hw1

Question **Code**

```
請載入標準模組庫中的 random 模組，
並利用其中的choice函數從52張的撲克牌中拿出5張牌。
```

3.8.3.2. Hw2

Question Code

```
請撰寫一個程式，利用random 模組輸出一份家事分配表。預設串列：
member = ["花媽", "花橘子", "花柚子", "花爸"]
housework = ["掃地", "拖地", "洗衣服", "擦窗戶"]
```

3.8.3.3. Hw3

Question Code

```
建立一個包含所有英文大寫字母的串列，
每次隨機從中挑取一個字母輸出，
共輸出5次，不可重複，也不可刪除串列中的資料。
```

Note

1. Start: 20170719

2. System Environment:

Listing 3.8.3.3.1 requirements.txt

```
sphinx==7.1.2                           # Sphinx
graphviz>=0.20.1                        # Graphviz
sphinxbootstrap4theme>=0.6.0            # Theme: Bootstrap
sphinx-material>=0.0.35                 # Theme: Material
sphinxcontrib-plantuml>=0.25            # PlantUML
sphinxcontrib.bibtex>=2.5.0             # Bibliography
sphinx-autorun>=1.1.1                   # ExecCode: pycon
sphinx-execute-code-python3>=0.3        # ExecCode
btd.sphinx.inheritance-diagram>=2.3.1   # Diagram
sphinx-copybutton>=0.5.1                # Copy button
sphinx_code_tabs>=0.5.3                 # Tabs
sphinx-immaterial>=0.11.3               # Tabs

#------------------------------------------------------
#-- Library Upgrade Error by Library Itself
#   >> It needs to fix by library owner
#   >> After fixed, we need to try it later
#------------------------------------------------------
pydantic==1.10.10                       # 2.0: sphinx compiler error, 20230701

#------------------------------------------------------
#-- Minor Extension
#------------------------------------------------------
sphinxcontrib.httpdomain>=1.8.1         # HTTP API

#sphinxcontrib-blockdiag>=3.0.0         # Diagram: block
#sphinxcontrib-actdiag>=3.0.0           # Diagram: activity
#sphinxcontrib-nwdiag>=2.0.0            # Diagram: network
#sphinxcontrib-seqdiag>=3.0.0           # Diagram: sequence

#------------------------------------------------------
#-- Still Wait For Upgrading Version
#------------------------------------------------------

#------------------------------------------------------
#-- Still Under Testing
#------------------------------------------------------
#numpy>=1.24.2                          # Figure: numpy

#------------------------------------------------------
#-- NOT Workable
#------------------------------------------------------
#sphinxcontrib.jsdemo==0.1.4    # ExecCode: Need replace add_js_file()
#jupyter-sphinx==0.4.0          # ExecCode: Need gcc compiler
#sphinxcontrib.slide==1.0.0     # Slide: Slideshare
#hieroglyph==2.1.0              # Slide: make slides
#matplotlib>=3.7.1              # Plot: Need Python >= v3.8
#manim==0.17.2                  # Diagram: scipy, numpy need gcc
#sphinx_diagrams==0.4.0         # Diagram: Need GKE access
#sphinx-tabs>=3.4.1                     # Tabs: Conflict w/ sphinx-material
```