程式設計概論
Programming 101
—模組與套件

授課老師：邱淑怡

Date:1/6/2021

Outline

- Python的包裝機制
  - Function
  - Module
  - Class
  - Package

Function

- 當發現程式碼片段相似只有參數會變化時，這時可以考慮將程式碼包成函數使用，也是一種抽象化的實作，將執行的細節(加法的動作)隱藏起來

```
def add(a,b):
    return a+b

ab=add(1,4)
print(ab)
```

模組(module)

- 模組：就是一個.py檔案
  - 是Python程式的基本單位
  - 包含了相關性較高的程式碼
  - 隨著應用程式的開發規模越來越大，我們不可能把所有的程式碼都寫在同一份Python檔案中，一定會將關聯性較高的程式碼抽出來放在不同的檔案中來形成模組(Module)，主程式再透過引用的方式來使用。
  - 模組可以提高程式碼的重用性(Reusable)且易於維護。
- 模組定義function, class, variable等等，在使用前需要 import modulename

Module

- 假設需要做一個兩數運算的簡易數學模組，增加一組數學相關函數存成一個檔案命名為math_module.py，即完成一個模組。這也是Python裡面最直接直覺的一個建立模式

```
# Save as math_module.py
def add(a, b):
  return a+b
def sub(a, b):
  return a-b
def p(content):
    return print(content)
```

Module(cont.)

- 如何使用該模組?
  - 在相同目錄下建立一個math1.py當作測試該模組

```
# test math_module 可用 jpynb執行之
import math_module as my1

print(my1.add(1,3)) #3
my1.p(my1.sub(2, 5))  #-1
```

如何匯入?

- import package_name [as new_name]
  - import pandas as pd
- import package_name.sub_package_name.module_name [as new_name]
  - import matplotlib.pyplot as plt
- from package_name import module_name [as new_name]
  - from scipy import stats
- from package_name.module_name import variable_name [as new_name]
  - from random import randint
  - print(randint(0,5))

class(類別)

- 當將某些**狀態**與**功能**黏在一起時，適合用類別
- 多數主流的答案可能會是可實現物件導向設計，而物件導向在嘗試解決的其實是易用性（Usability）與重用性（Reusability）
- 讓程式碼一直被重複使用
- OOP的重點依然是考量到未來的擴展與嘗試讓複雜的系統重用/易用提高
  - 封裝
  - 繼承
  - 多型

套件 (package)

- 當一堆模組（一堆.py檔案）分別分工實現某些功能
- Package將模組以資料夾形式進行分組管理的方法
- 當彼此有共通性成為一種類別後，應該要怎麼管理？這時套件就可以利用上了
- 就是一個容器(資料夾或目錄)，包含了一個或多個的模組(Module)，並且擁有__init__.py檔案，其中可以撰寫套件初始化的程式碼

Package架構: 舉例說明

```
package/
    __init__.py
    subpackage1/
        __init__.py
        moduleX.py
        moduleY.py
    subpackage2/
        __init__.py
        moduleZ.py
    moduleA.py
```

- Import同一個package底下的sibling module ‘module Y’
  - from package.subpackage1 import moduleY
  - (or) from . import moduleY

Package架構: 舉例說明 (cont.)

```
package/
    __init__.py
    subpackage1/
        __init__.py
        moduleX.py
        moduleY.py
    subpackage2/
        __init__.py
        moduleZ.py
    moduleA.py
```

- __init__.py:是package內必須存在的檔案，會在package載入時自動呼叫，無論以什麼方式匯入，都會執行上一級所有package的__init__.py檔案
  - __init__.py內的__all__列表: 用於記錄此包中有哪些模組需要匯入，需要自行建立用from xxx import *時，只查詢__all__列表中的module 或subpackage
  - 當在subpackage中呼叫package中的函式時，用相對匯入
    .表示本級(目前)目錄
    ..上級目錄
    ...上兩級目錄

模組的相關操作: 以Python提供的 sys為例

```
import sys
dir(sys)  # 檢視模組中可用的方法
sys.modules  #已經被載入的模組欄位， key是模組名，value是模組
sys.path   #模組所使用的路徑
sys.platform  #系統平臺
sys.stdin    #標準輸入流
sys.stdout    #標準輸出流
sys.version  #獲取Python解釋程式的版本資訊
sys.argv   #命令列的引數，包括指令碼名稱
sys.exit   #返回異常
sys.modules.keys()  # 返回所有已經匯入的模組列表
```

常見第三方函式庫

- Numpy: 矩陣與資料運算，線性代數、傅立葉轉換
- Matplotlib: 2D 視覺化工具
- Scipy: 科學計算，最佳化與求解、矩陣運算、傅立葉轉換
- Pandas: 資料處理及分析
- Django, Pyramid, Web2py, Flask: web框架，開發網站
- Kivy, Flexx, Pywin32, PyQt, WxPython: GUI程式開發
- PyGame: 多媒體與遊戲軟體開發
- Requests: 存取網際網路資料
- Scrapy, Beautifulsoup: 網路爬蟲
- Scikit-learn, TensorFlow, Keras: 機器學習與深度學習