



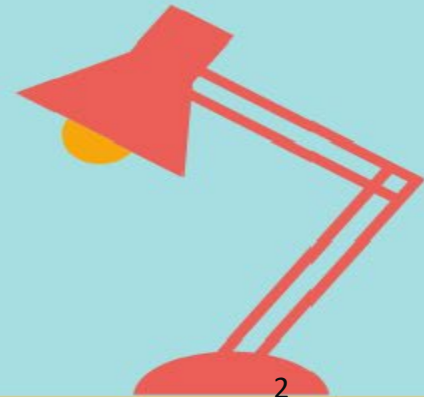
113-1基礎程式設計(15)

亞大資工系

課程大綱

- W1-Python簡介及程式工具
- W2
- W3-變數和運算
- W4-迴圈和格式化輸出
- W5-判斷式和容器
- W6-字串處理和輸出入
- W7-字典容器
- W8-檔案處理
- **W9-期中測驗**

- W10-函數
- W11-進階流程控制
- W12-進階運算和生成器
- W13-進階函數
- W14-類別
- W15-進階類別
- W16-模組和套件
- W17-進階設計
- W18-期末測驗



本週主題-進階類別(Class)

Python 三大器:
迭代器 iterator
生成器 generator (comprehension)
裝飾器 decorator

- Week15-類別(Class)的各種方法
 - Topic 1(主題1)-實例方法/類別方法
 - Topic 2(主題2)-靜態方法
 - Topic 3(主題3)-decorator (裝飾器)
 - Topic 4(主題4)-流程控制的複習



Topic 1-類別實例方法

- derived class (衍生類別)

```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname

    def printname(self):
        print(self.firstname, self.lastname)

class Student(Person):
    pass
```



super()方法

Python 還有一個 `super()` 函數，可以讓子類繼承其父類的方法和屬性

```
class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname

    def printname(self):
        print(self.firstname, self.lastname)

class Student(Person):
    def __init__(self, fname, lname):
        super().__init__(fname, lname)
```



Topic 1-類方法

```
class Person:

    def __init__(self, name):
        self.name = name

    def greet(self):
        print('Hello, my name is {}'.format(self.name))

    @classmethod
    def info(cls):
        print('調用了類方法')
```



Topic 2-靜態方法的調用

```
Person.info()
```

```
person_b = Person('Xiao Hong')  
person_b.info()
```



Topic 2-靜態方法

```
class Person:
    def __init__(self, name):
        self.name = name
    def greet(self):
        print('Hello, my name is {}'.format(self.name))

    @classmethod
    def info(cls):
        print('調用了類方法')

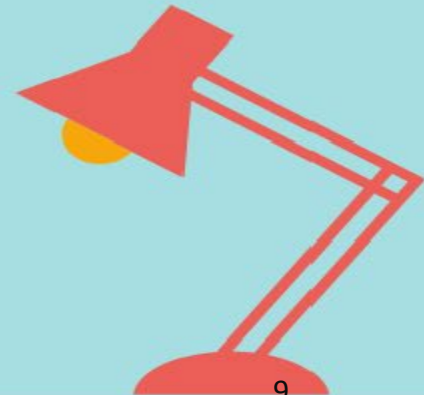
    @staticmethod
    def calculate(a, b):
        print('調用靜態方法計算兩數之和')
        print('{}+{}={}'.format(a, b, a+b))
```



Topic 2-類方法的調用

```
Person.calculate(1, 2)
```

```
person_c = Person('Pang Hu')  
person_c.calculate(1, 2)
```



Topic 3- decorator (裝飾器)

由於 Python 的基本語法為了簡單好用簡潔，讓 Python 的語法變得愈來愈繁複。裝飾器是一個函式，它會回傳另一個函式，通常它會使用 `@wrapper` 語法，被應用為一種函式的變換 (function transformation)。裝飾器的常見範例是 `classmethod()` 和 `staticmethod()`。

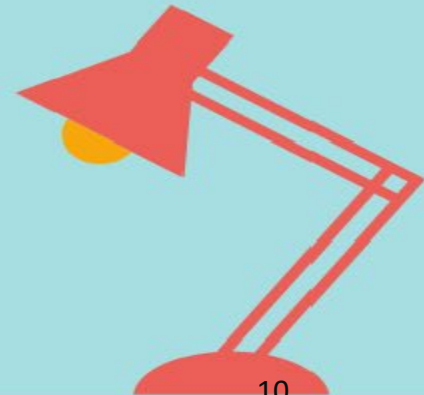
裝飾器語法只是語法糖。

```
def f(...):  
    ...  
f = staticmethod(f)  
  
@staticmethod  
def f(...):  
    ...
```

```
def my_decorator(func):  
    print('In A')  
    return func
```

```
@my_decorator  
def my_func():  
    print('In B')
```

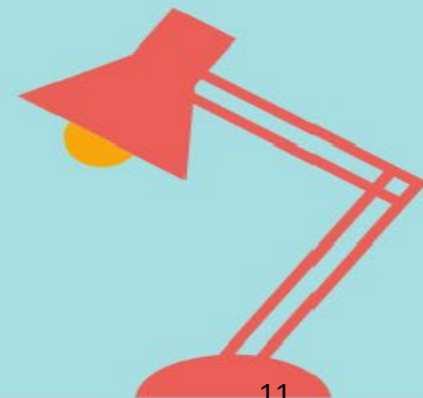
```
my_func()
```



Topic 4-日誌(logging)

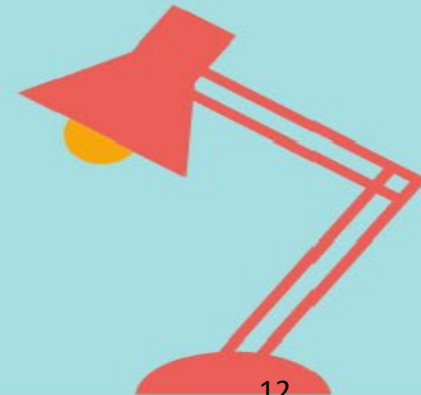
Python 的 logging 可以將 log 輸出到 console 與輸出到日誌檔裡
Python logging 的 log level 有五種等級，分別為DEBUG、INFO、
WARNING、ERROR、CRITICAL，預設 log level 等級是
WARNING。log level 等級與分別對應的 api 如下：

- DEBUG : logging.debug()
- INFO : logging.info()
- WARNING : logging.warning()
- ERROR : logging.error()
- CRITICAL : logging.critical()



Topic 5-作業系統(os)

- 提供了一種使用與操作系統相關的功能的便捷式途徑。
 - `os.system()` #括號中加入CMD指令，即可用Python執行(例如:`os.system(ls)`)
 - `os.walk()` #遍歷資料夾或路徑
 - `os.path()` #主要用於獲取資料夾or檔案屬性或資訊
 - `os.environ.get('PATH')` #取得環境變數內容
 - `os.rename(原檔名,新檔名)` #更改文件檔名
 - `os.remove(檔名)` #刪除檔案
 - `os.getcwd()` #獲取當前目錄
 - `os.mkdir(“資料夾名稱”)` #建立資料夾
 - `os.path.isdir(檔名)` #檢查是不是目錄
 - `os.path.expanduser('~')` #取得家目錄路徑



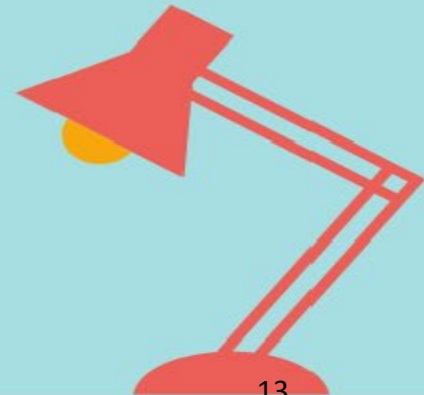
Topic 6-檔名匹配(glob)

星號匹配

```
import glob
for name in glob.glob('sample_data/*'):
    print(name)
```

字元區間匹配

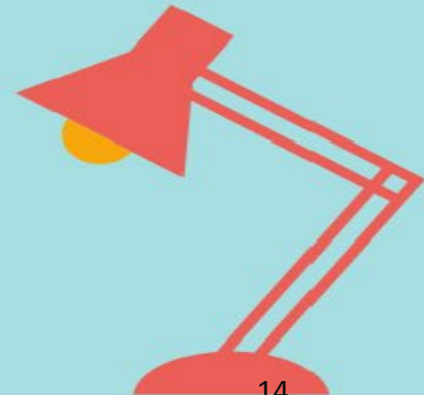
```
import glob
for name in glob.glob('sample_data/[a-c]*'):
    print(name)
```



Topic 7-sys系統相關(1) sys.argv

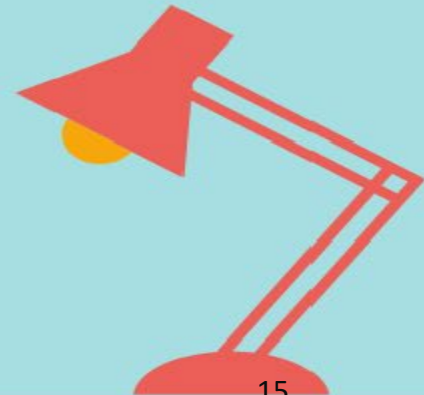
sys.argv – command arguments

```
import sys
if __name__ == '__main__':
    if len(sys.argv) < 2:
        print('no argument')
        sys.exit()
    print(len(sys.argv))
    for arg in sys.argv:
        print(arg)
```



Topic 7-sys系統相關(2) sys info

- `sys.platform` 執行的平台
- `sys.version_info` 當前 Python 的版本
- `sys.path` Python 的執行路徑





Thanks!

Q&A

