Python 程式設計

林奇賦　daky1983@gmail.com

Outline

- 定義類別、建立物件
- 類別中的各種方法
- 封裝
- 繼承
- 多型

類別 Class

- Python中所有東西都是物件，凡是物件都有屬性 (attribute) 跟方法 (method) 。
- 所謂的屬性雷同變數 (variable) ，專屬於物件。
- 方法類似函數 (function) ，同樣專屬於物件。

定義類別

- 定義類別使用關鍵字 (keyword) **class** ，通常類別的命名第一個字習慣使用大寫(大駝峰)，形式如下：

```
class Class_Name:
    # class 內容
    # 記得縮排
```

類別內容

- 以學生物件為例 Student，
- 含有**屬性**：
  - name：學生姓名
  - sid：學號
  - grades：分數列表
- 含有**方法**：
  - add(分數)：新增成績

範例程式

```
1  class Student:
2      def __init__(self, name, sid):
3          self.name = name
4          self.sid = sid
5          self.grades = []
6
7      def add(self, grade):
8          self.grades.append(grade)
9
```

屬性

方法

建立物件

- 格式：
  - 類別名稱(值)

視類別建構子__init__之定義傳入

__init__( )

- 利用建構子 (constructor) 建立的物件被稱為實體 (instance)
- 初始化一個新實例，控制這個初始化的過程，比如添加一些屬性， 做一些額外的操作，發生在類實例被創建完以後
- 自行定義的類別會有預先定義好的 __init__( ) ，我們也可以改寫 (override) 成 我們自己需要的
- 改寫方式就是再定義一次，方法的定義與函數 (function) 類似，兩者同樣使用關鍵字 (keyword) def

類別屬性、實體屬性

- 類別屬性
  - 同一個類別所產生出來的不同實體，會共同存取到同一個類別屬性
  - 初始值定義在類別的區塊中
- 實體屬性
  - 每一個實體自己獨立擁有的屬性
  - 初始值定義在 __init__( ) 函數中
  - self.屬性名稱

內建方法

- **__call__**
  - 變成callable的實例，此方法內定義當實例被呼叫時要做的動作
- **__del__**
  - 當時實例被刪除的時候會被呼叫的方法
- **__str__**
  - 轉型為str型態時，若有實作此方法，會將其回傳值當作轉型後的結果

Python 類別中的各種方法

- instance method 實體方法
  - 必須要帶預設參數 self，當作此實體
- class method 類別方法
  - 必須要帶預設參數 cls，當作此類別
- static method 靜態方法
  - 不用帶任何預設參數 (self、cls)
- class method、 static method 都不需要產生實例，即可直接使用類別呼叫方法

類別方法

- 類別方法需要一個特別的參數 (parameter) ，習慣上使用 cls ，這與實體方法的 self 類似，不同的是 cls 用來存取類別的屬性 (attribute)
- 使用時需要在方法的定義之前加入 @classmethod
- 不需要產生實體，即可直接使用類別呼叫方法

靜態方法

- 不用帶任何預設參數 (self、cls)
- 使用時需要在方法的定義之前加入 @staticmethod
- 與類別方法一樣，不需要產生實體，即可直接使用類別呼叫方法

Homework 5

- 試寫一個名為 Student 的類別
- 其中屬性包含:
  - name, gender, grades
- 函數包含:
  - avg: 回傳grades list的平均值
  - add(grade): 新增成績到grades list中
  - fcount: 回傳不及格(<60)的總數
  - 分別將每個學生的成績平均、不及格的的數目印出
- 寫一個名為top的**類別函數**:
  - 傳入值為多個學生物件 (使用不定個數)
  - 將平均分數最高的學生回傳

封裝

- 封裝對象：屬性、方法
- 目的：
  - 讓屬性只能在類別中建立及修改。
  - 讓方法只能在類別中被呼叫
- 在Python中，私有化屬性或者方法，可以在屬性、方法名字前加上雙底線。

@property

- 改寫的屬性實際存放名稱在前方加上一個_(底線)
- 在設定屬性值的方法前加上@property修飾
- 在取出屬性值的方法前加上@屬性名稱.setter的修飾. 之前加上底線的屬性名稱, 是為了和設定及取用方法的名稱有所區別
- 讓屬性的讀取與寫入更嚴謹

繼承

- 繼承是物件導向程式設計的主要特性之一
- 當我們定義一個class的時候，可以從某個現有的class繼承，新的class稱為子類（Subclass），而被繼承的class稱為父類別（Superclass）
- 繼承可以把父類的所有功能都直接拿過來，這樣就不必重零做起，子類別只需要新增自己特有的方法，也可以把父類不適合的方法覆蓋重寫

繼承

- 繼承的格式如下

```
class SubDemo(Demo):
        # class 內容
        # 記得縮排
```

- 這是從 SubDemo 類別去繼承 Demo ，注意類別名稱後的小括弧中註明父類別。

子類別方法改寫

- 子類別也可依需要改寫 (override) 父類別的方法
- 子類別需要用到與父類別具有相同名稱的方法， 但是子類別需要的功能有所修改、擴充或增加，因此當子類別裡頭定義與父類別相同名稱的方法時，就會改寫父類別的方法。經過改寫，子類別的方法完全屬於子類別所有。
- 簡單說就是**重新定義同樣名稱的方法，但實作不同的內容**

super()

- 利用內建函數 (function) super()，呼叫 (call) 父類別的方法

isinstance()

- isinstance() 可以判斷某一個物件 是否為某一個類別所建構的實體 (instance) ，若真則 回傳 True ，否則回傳 False。

issubclass()

- issubclass() 則可以判斷某一個類別是 否為另一個類別的子類別，同樣的，若真則回 傳 True ，否則回傳 False 。

多重繼承

- 設計類別 (class) 時，父類別 (superclass) 可以有多個 ，這是說子類別 (subclass) 能夠繼承 (inherit) 多個父類別，使子類別可以有多種特性。
- 這裡須注意一點，當子類別繼承 (inheritance) 超過一個來源的時候，會以寫在最左邊的父類別優先繼承， 這是說，多個父類別如果有相同名稱的屬性 (attribute) 與方法 (method) ，例如 __init__() 、 __str__() 等，就會以最左邊的父類別優先。

多型

- 多型 (polymorphism) 是物件導向程式語言 (object-oriented programming language) 的一項主要特性，使物件 (object) 的使用更具彈性。簡單來說，多型可使物件的型態具有通用的效力
- 多個類別可以定義相同的函式名稱，而相同函式名稱在不同類別可以定義各自特有的功能，經由呼叫物件的函式名稱，傳入不同的物件都定義此相同函式名稱而產生不同的功能，稱作「多型」

範例

```
d1 = "12345"
d2 = [1, 2, 3, 4, "5"]
print(d1.count("4"))
print(d2.count("4"))
```

說明

- d1 為字串 (string) ， d2 為串列 (list) ，兩者皆屬於序列 (sequence) 的複合資料型態 (compound data type) ，有通用的 count() 方法，可計算某元素 (element) 累計出現的次數。
- 多型的應用很多，例如串列中可接受不同型態的物件當元素，或是方法可用不同型態的參數等