

# 物件導向程式設計簡介

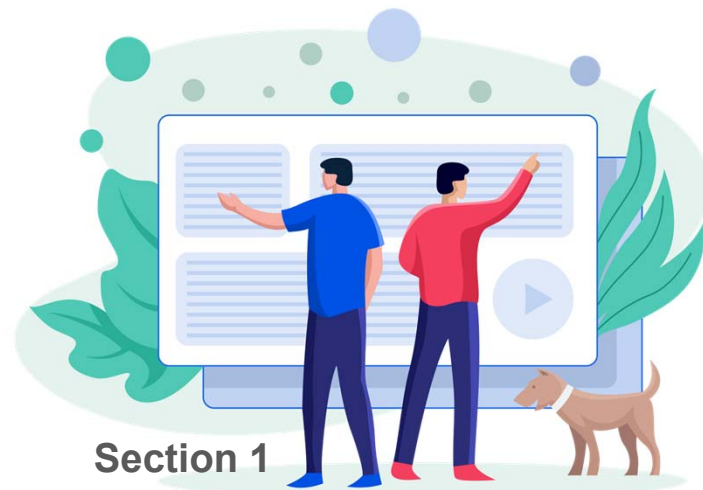
# 學習目標

- 物件導向簡介
- 物件導向相關名詞
- 介紹 Python 類別(Class)



# 物件導向概念

- 簡介物件導向特性
- 物件導向常用名詞

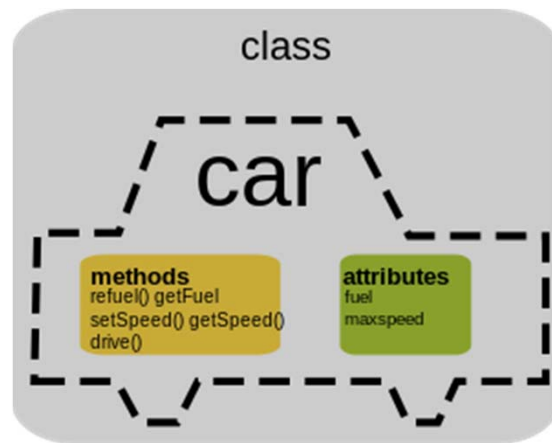
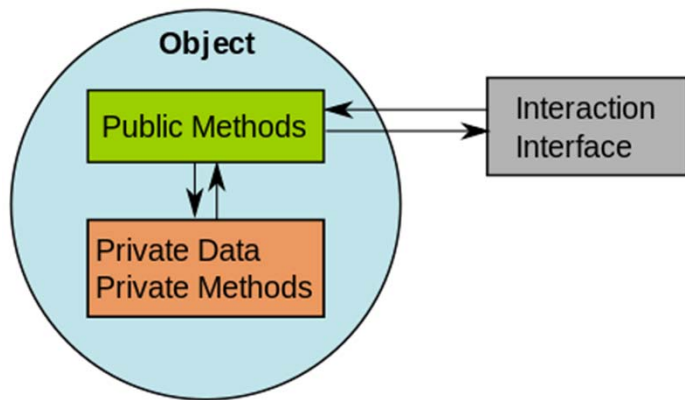


# 類別(Class)

- 類別提供一種將資料與函數綁在一起的機制
- 建立新類別(Class) 會建立一個新型態的物件(Object)，然後可以建立該物件的新實例(Instance)
- 與其他程式語言相比，Python 的類別機制使用最少新語法和語義，它是源自 C++ 和 Modula-3 的類別機制
- Python 類別提供物件導向程式設計的標準功能，例如繼承機制允許繼承多個父類別，衍生類別可以覆蓋(Override) 父類別的方法(Method)，方法(Method) 內可以呼叫父類別同名的方法

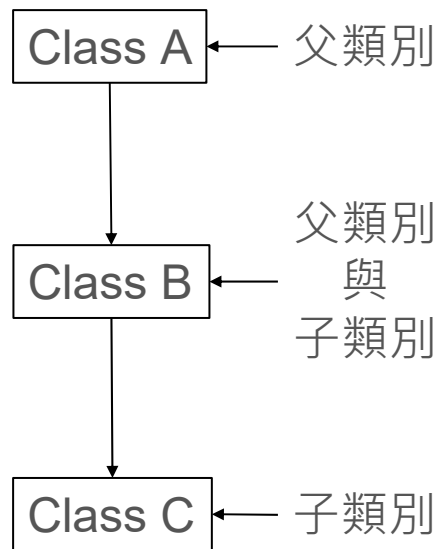
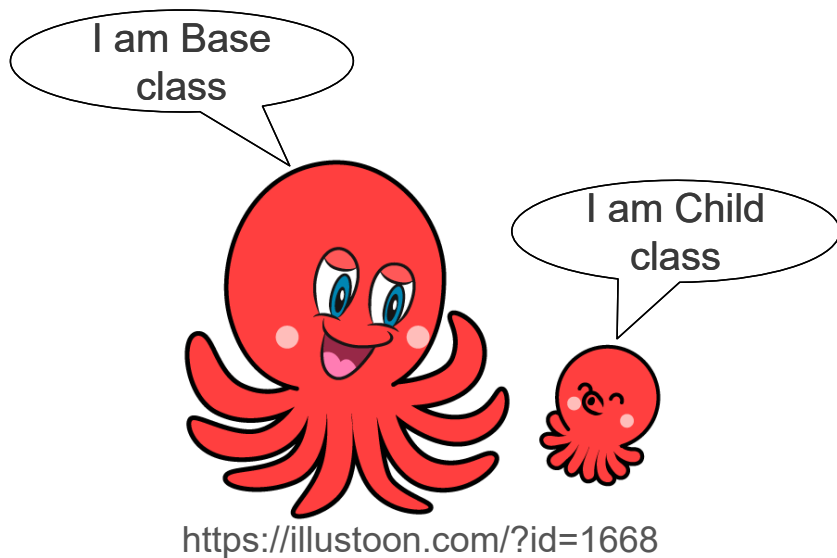
# 封裝(Encapsulation)

- 隱藏類別內的資料狀態或是內容值，避免外部直接存取資料，達到資料的安全性與一致性



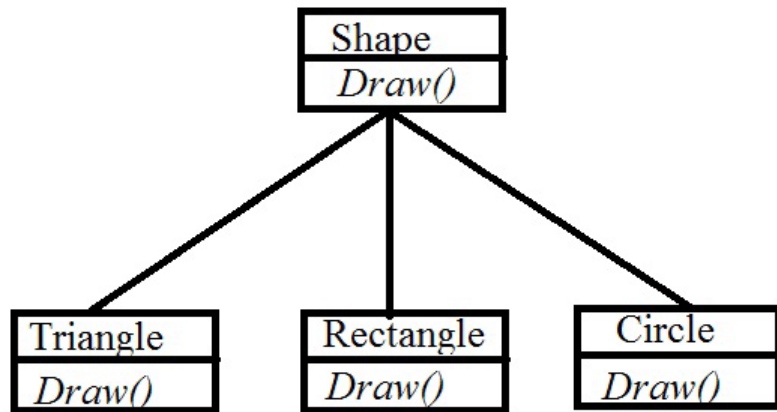
# 繼承(Inheritance)

## ■ 從已存在的類別定義新的類別



# 多形(Polymorphism)

- 一個介面(Interface) , 多個方法(Methods)



# 物件導向名詞 -1

## ■類別(Class)

- 使用者自訂的物件原型(prototype)，定義一系列屬性紀錄該類別物件的特徵
- 屬性有二，一為資料成員(data members)：有類別變數(class variables) 與實例變數(instance variables)，另一為方法(methods)：透過逗點方式(dot notation) 存取

## ■物件(Object)

- 由類別定義的資料結構所產生的實例(instance)
- 物件是由資料成員與方法組成

## ■實例(Instance)

- 某類別的單一物件，例如有一物件 obj 是數與 Circle 類別，則 obj 就是 Circle 類別的一個實例



# 物件導向名詞 -2

## ■ 實例化(Instantiation)

- 建立一個類別的實例

## ■ 類別變數(Class variable)

- 所有類別實例共用的變數
- 類別變數定義於類別內，但是在類別方法之外
- 類別變數不像實例變數(instance variables) 那麼常用

## ■ 實例變數(Instance variable)

- 定義在方法內的變數，而且只屬於目前類別實例擁有

# 物件導向名詞 -3

## ■資料成員(Data member)

- 類別變數(class variable) 或是實例變數(instance variable) 儲存類別共用資料或是個別物件資料

## ■方法(Methods)

- A special kind of function that is defined in a class definition

## ■函數重載(Function overloading)

- 一個函數，可以指定多個行為(behavior)
- 會涉及到引數的個數與資料型態
- 目前 Python 不支援函數重載，因為 Python 不用宣告變數(例如 C 語言的 `int cnt`)

# 物件導向名詞 -4

## ■ 運算子重載(Operator overloading)

- 指定某個運算子有不同的功能

## ■ 繼承(Inheritance)

- 將父類別(superclass) 的特徵傳遞給子類別(subclass)
- Python 允許多個父類別繼承

## ■ 方法覆寫(Method overriding)

- 有繼承關係的類別之中，覆蓋父類別同名的方法，重新定義該方法

# Python類別

## ■ 定義與使用類別



Section 2

# 定義類別 -1

- class 敘述可以定義新的類別

- 語法

- class ClassName:  
    <statement-1>  
    .  
    .  
    .  
    <statement-N>

- statement-1 ~ statement-N 包含所有類別的元件，例如類別成員，資料屬性與函數

## 定義類別 -2

### ■範例

```
■class Employee:
    emp_count = 0
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.emp_count += 1
    def display_count(self):
        print ('Total Employee {}'.
              .format(Employee.emp_count))
    def display_employee(self):
        print('Name : ', self.name, ', Salary: ',
              self.salary)
```

## 定義類別 -3

- `emp_count` 變數是類別變數(class variable)，該值是所有實例(instances) 共用
  - 可以使用 `Employee.emp_count` 方式，在類別內外存取類別變數
- 類別內的第一個方法 `__init__()` 是一個特殊的方法，稱為類別建構子(class constructor) 或初始化方法(initialization method)
  - 當產生類別的實例時，Python 會自動呼叫類別建構子
- 用一般函數宣告的方式定義方法，唯一的不同是第一個引數是 “self”
  - 當呼叫方法時，Python 會自動加入 “self” 引數

## 使用類別 -1

```
■ emp1 = Employee('Lisk', 22000)
  emp2 = Employee('Mary', 55000)

emp1.display_employee()
emp2.display_employee()

emp1.display_count()
emp2.display_count()

print('emp_count=', Employee.emp_count)
```



# 使用類別 -2

## ■ 建立實例物件(Instance Objects)

- 要建立類別的實例(instances) , 要使用類別名稱並傳入必要的引數 , Python 會將相關引數傳給類別 `__init__` 方法
- `emp1 = Employee('Lisk', 22000)`
- `emp2 = Employee('Mary', 55000)`

## ■ 存取屬性

- 使用物件逗點運算子來存取物件屬性
  - `emp1.display_employee()`
  - `emp2.display_count()`
- 類別變數可以使用類別名稱來存取
  - `Empolyee.emp_count`

# Python類別(Demo)

## ■ 定義與使用 Python 類別



# 本章重點精華回顧

- 物件導向相關概念
- Python 類別的定義與使用



# Lab: 類別-物件導向程式設計簡介

## ■ Lab01: 定義 Python 類別

# Lab01: 定義Python類別

1. 啟動Python IDLE環境，做以下練習
2. 使用 “File/Open...” 開啟 “baseclass.py” 程式，了解基礎類別的定義
3. 關閉baseclass.py程式視窗