

【南區Fintech研習營】Python 程式設計基礎：
Google finance股價爬蟲應用
講者：林萍珍



DrMaster

精選簡報・教師專用
博碩文化・版權所有

www.drmaster.com.tw

第八章 物件導向程式設計

8-1 物件導向程式設計簡介

8-2 類別、物件與實體

8-6 錯誤與異常

8-7 模組與套件

8-8 實務案例



物件導向程式設計

- 物件導向程式設計（**object oriented programming, OOP**）分析的主角是物件，這些物件有哪些功能（方法）？
- 物件需要與外界做什麼資料交換？
- OOP是將資料與方法包裝成物件，分成多個獨立小單位，每個單位可以被重覆使用，並且透過介面互相傳遞訊息（見圖8-3）。

OOP 訊息傳遞

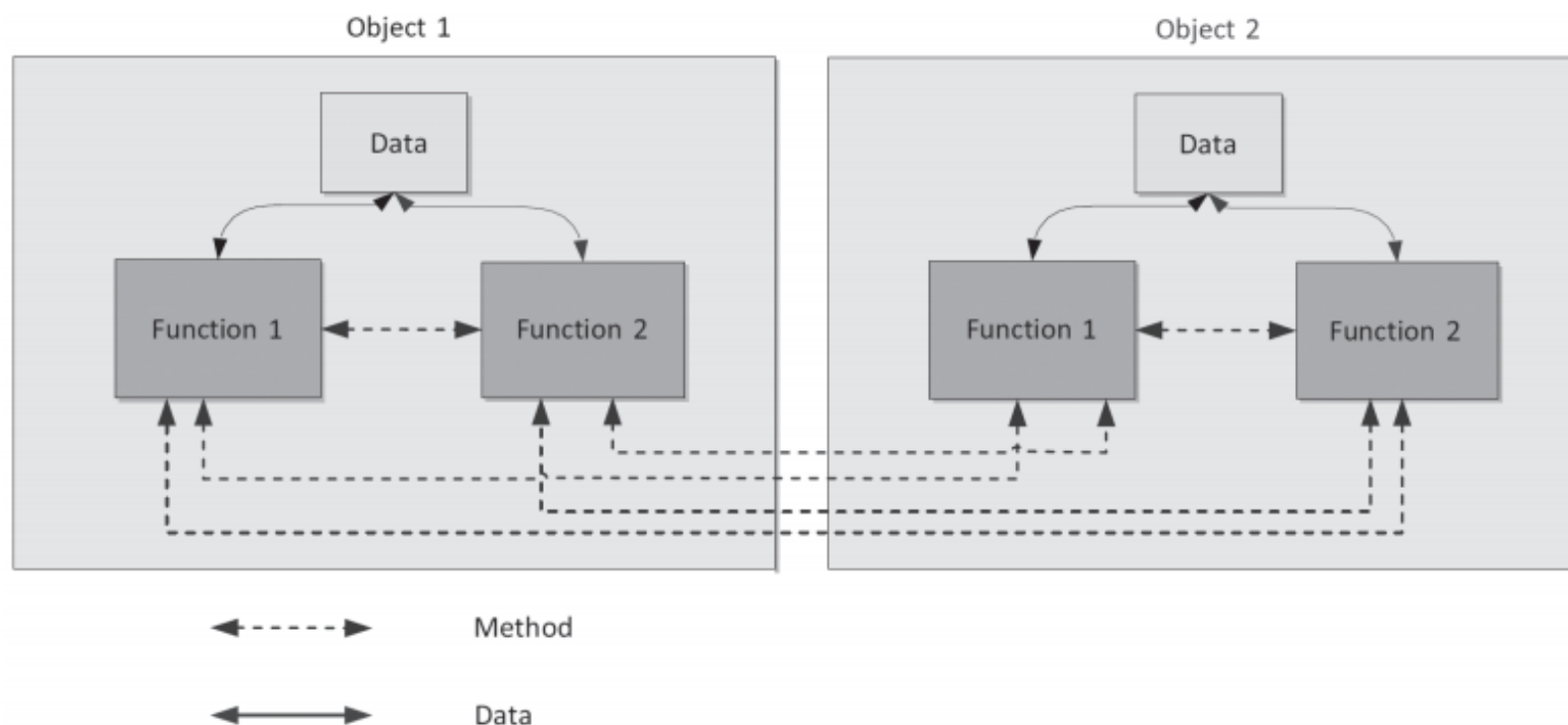


圖 8-3 物件導向程式設計

OOP 最基本的觀念包含：物件（object）、類別（class）、方法（method）、封裝（encapsulation）、繼承（inheritance）、多型（polymorphism），本章將對這些物件導向設計的觀念與用法做介紹。

何謂物件

- 將資料與函數組裝在一起成為一個基本的資料運算單元，透過對外溝通的介面——方法（methods）或函數才能操作它，即稱為物件（見圖 8-4）。

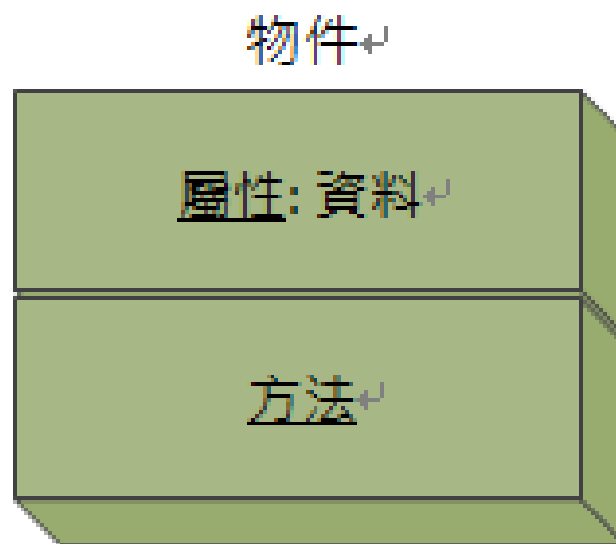
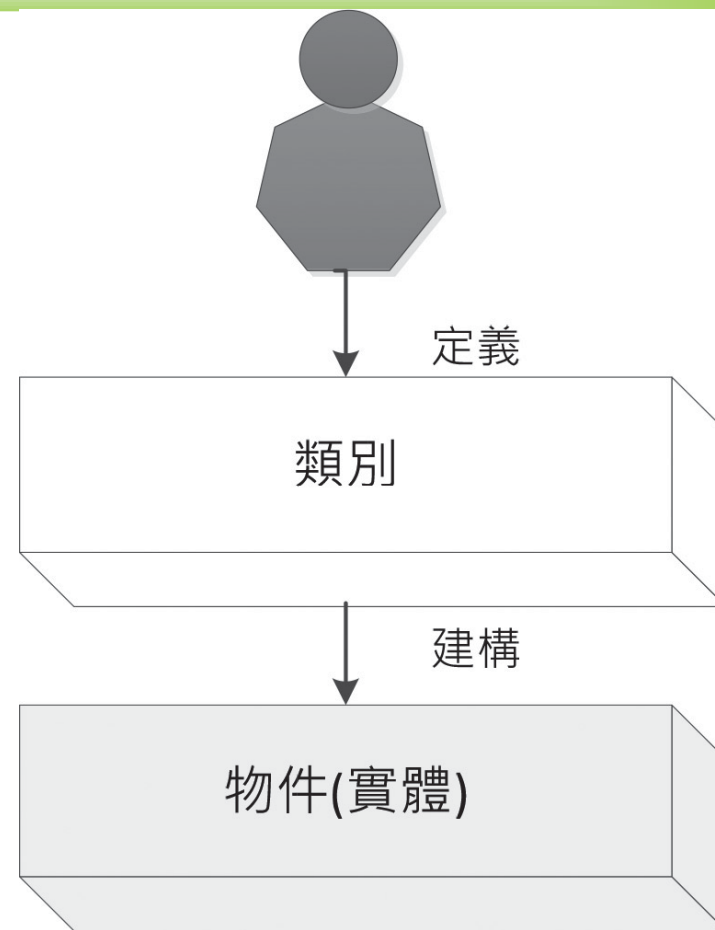


圖 8-4 物件

何謂類別

- 類別用以定義物件的屬性與方法，物件為類別的實例（見圖 8-5）。
- 它為程式提供物件的版型與結構，以利程式碼的重複利用。
- 類別所定義的物件，其成員即是內部的屬性與方法（見圖 8-4）。
- 類別是比較抽象資料型別，只有依類別建構出來的才是物件實體。



何謂實體

透過類別可以建構實體，類別是抽象的，實體才是具體的物件。Python 的類別與物件實體的關係見圖 8-6，可以用製造車子來比喻。

☑ 概念階段（發想）

類別就像製造車子之前，設計師依他的理想概念畫成設計圖，那裡有車門、輪胎、安全氣囊、車燈、方向盤等，這些能都是在腦子裡的抽象概念描述出來的。

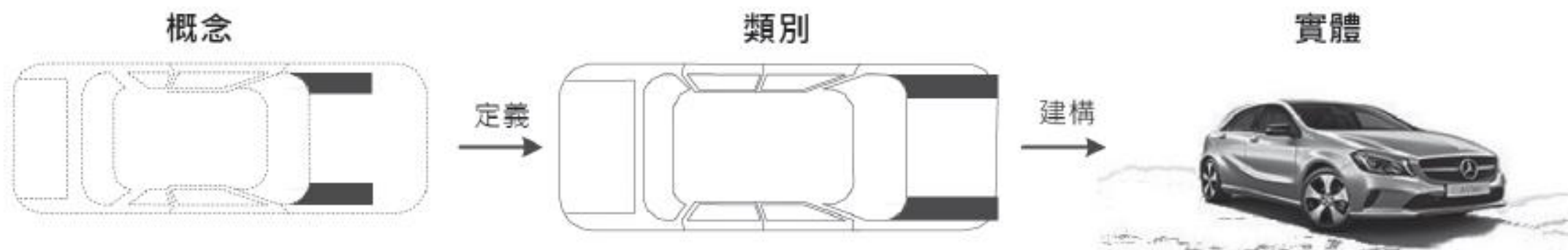
☑ 設計階段（類別）

工程師依據設計師的設計圖（概念），做出一個模型或樣版出來，要有詳細的組裝流程以及配件與配料等，例如：顏色、輪子的型式、車內應有配備等。

何謂實體

☑ 實作階段（實體）

工廠的技師，再依備組裝流程說明、配料與配件組裝出一台有形的車子實體。



簡易銀行存提款功能採用程序導向程式設計模式

範例 8-2 簡易銀行存提款功能採用物件導向程式設計模式

程式功能要求同【範例 8-1】。

示範程式碼

```
1  # E_8_2: 銀行存提款功能採用物件導向程式設計模式
2  # 定義新增銀行帳戶類別
3  class CreateBankAccount:
4      # 定義新增帳戶方法，額餘設定為 0
5      def __init__(self, id, name):
6          self.id = id
7          self.name = name
8          self.balance = 0
9      # 定義提款方法
10     def withdraw(self, amount):
11         self.balance -= amount
12         return self.balance
13     # 定義存款方法
```



```
14     def deposit(self, amount):
15         self.balance += amount
16         return self.balance
17     def __str__(self):
18         return(' 帳號 : '+self.id+ '\n 姓名 : ' + self.name + '\n 期初餘額 : ' + str(self.balance))
19 customer_a = CreateBankAccount('c001', 'Jemery Lin')
20 customer_b = CreateBankAccount('c002', 'Barack Obama')
21 print(customer_a.__str__())
22 print(' 存款後餘額 : ',customer_a.deposit(100))
23 print(customer_b.__str__())
24 print(' 存款後餘額 : ',customer_b.deposit(50))
25 print(customer_a.__str__())
26 print(' 提款後餘額 : ',customer_a.withdraw(10))
27 print(customer_b.__str__())
28 print(' 提款後餘額 : ',customer_b.withdraw(10))
```



程式說明

步驟 1 新增帳戶

呼叫新增帳戶函數 `CreateBankAccount()` 新增兩個帳戶（見第 19 到 20 列），代入帳號（`id`）與姓名（`name`）。`CreateBankAccount()` 是一個類別型別，預設功能是呼叫 `__init__(self)` 方法，新增帳戶帳號與姓名預設為傳入的參數，並將餘額設定為零（見第 6 到 8 列）。第 5 到 6 列的寫法待下一節說明。第 19 到 20 列呼叫類別，藉由類別所定義的物件 `__init__(self)` 產生的回傳值為物件實體分別為 `customer_a` 與 `customer_b`。

步驟 2 計算存款後餘額

呼叫物件實體（`customer_a`）的方法 `deposit()`，實體在前；方法在後；中間加句號，`customer_a.deposit()` 代入存入的帳戶（`customer_a`）與本次存款金額 100（見第 22 列）。方法內的形式變數是 `amount` 為本次存款金額加上傳入帳戶自己本身（`self`）的累計餘額後回傳（見第 15 到 16 列）。b 客戶同樣存入 50（見第 24 列）。

步驟 3 計算提款後餘額

呼叫物件實體 (`customer_a`) 的方法 `withdraw()`，代入提款帳戶 (`customer_a`) 與本次提款金額 10 (見第 26 列)，計算傳入帳戶的累計餘額減掉本次提款金額 (`amount`) 後回傳 (見第 11 到 12 列)。b 客戶同樣提款 10 (見第 28 列)。

◆ `__str__()` 方法

`__str__()` 方法是處理回傳物件所描述的字串，用來協助物件回傳以及與使用者溝通的說明文字。使用語法為物件實體 `.__str__()` 方法 (見第 21, 23, 25, 27 列)，呼叫前必須在所屬類別事先定義 (見第 17 到 18 列)，印出帳號、姓名與期初餘額。其中，期初餘額 (`balance`) 是數值資料，必須使用內建函數 `str()` 轉成字串才能印出。

執行結果

- 1 帳號 : c001
- 2 姓名 : Jemery Lin



3 期初餘額：0
4 存款後餘額：100
5 帳號：c002
6 姓名：Barack Obama
7 期初餘額：0
8 存款後餘額：50
9 帳號：c001
10 姓名：Jemery Lin
11 期初餘額：100
12 提款後餘額：90
13 帳號：c002
14 姓名：Barack Obama
15 期初餘額：50
16 提款後餘額：40



圖例說明

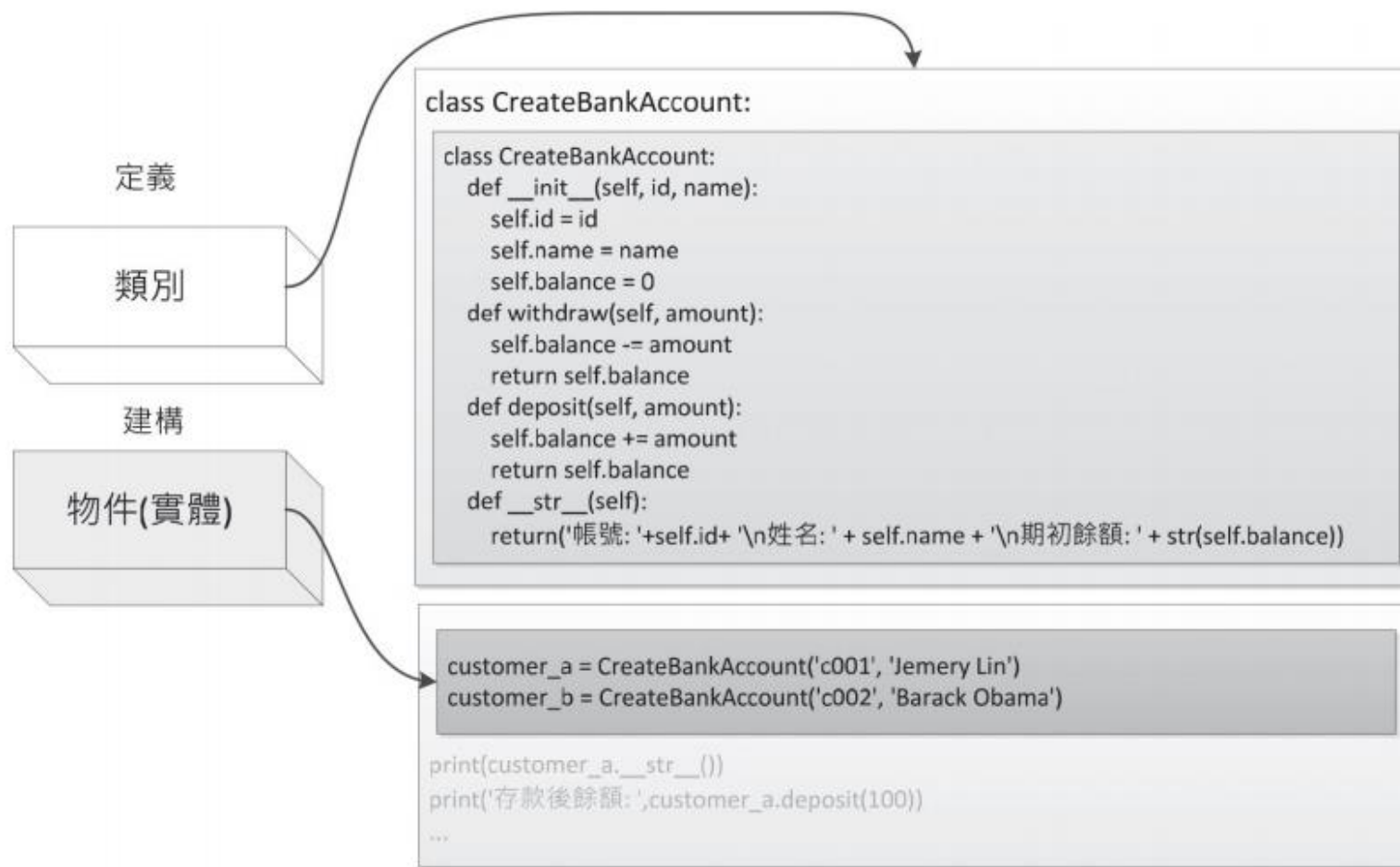


圖 8-7 類別、物件與實體對應程式碼說明



物件初始化

- 物件初始化，即建立物件的實體，通常會用到：建構（或稱建構子, constructor）：`__init__()` 與 `self` 來做初始化的動作。
- `__init__()`
 - Python 的物件是由建構子所建立的，類別定義出來的物件是可以被呼叫（callable）的，而呼叫類別即是物件建構子。建構子可為類別定義的物件建立實體。即當呼叫此類別名稱時即為該物件建構實體。
 - 只有在物件的實體建構完成後，Python 才會為被建構的實體配置記憶體，再依物件內部的設定做初始化，此即 `__init__()` 方法。

物件初始化

self

- `__init__()` 方法的主要功能是建立物件初始化，此方法的第一個形式參數是 `self`，表示要建立物件實例。
- 類別物件的定義中，第一個方法與參數必須為建構實體之用。所以，要冠上 `self` 的名稱。
- Python 習慣性用法是以 `self` 為 `__init__()` 方法的第一個命名參數，代表可存取實體本身的屬性。



方法

- 存取物件內部資料的函數稱為方法，物件是透過方法與外界溝通交換訊息，方法是物件對外溝通的管道。
- 方法在 Python 的運作方式，可以將方法視為函數，此方法是儲存在類別裡。
- 所有類別的成員包含方法與屬性都是公開（public）的成員，意指每個物件都可以使用它。
- 使用語法
 - 類別名稱.方法名稱()

封裝

- OOP 主要特性之一是封裝（encapsulation），其目的是為了隱藏物件的一些內部資訊，僅公開（公有）一些允許的介面讓外界存取。
- 封裝的好處
 - 為了隱藏或保護程式實作的細節
 - 可以讓程式碼重複利用；
 - 降低程式碼的維護成本（maintenance cost），即需求變更時，程式碼只要修改物件類別。如此，程式碼修改幅度可以降到最低。

繼承

- 類別可以繼承（inheritance）自父類別（parent class）的所有的特性，繼承者即為子類別（child class），一個子類別可以繼承自多個父類別。

- **super() 方法**

- 子類別可藉由 super() 方法繼承父類別，不必再重覆撰寫程式碼，此即稱為繼承。

- **改寫方法（override）**

- 若對類別所提供的方法，其需求有不足之處時可新增修改方法，進一步提供客製化的使用介面。
 - 子類別可視需要改寫父類別的方法，而且用到與父類別相同名稱的方法。但子類別需要的功能父類別同名的方法無法完全提供時，可以修改或擴充稱為改寫方法。即使經過修改或擴充，仍完全屬於子類別所有。



支票帳戶類別繼承的應用

範例 8-5 支票帳戶類別繼承的應用

在 Python 的世界，所有類別都繼承自 object 父類別。延用【範例 8-2】簡易銀行存提款的範例，改寫為支票存款戶子類別 `CheckingAccount` 的功能，其繼承自父類別 `CreateBankAccount` 一般帳戶的取款功能。Python 繼承的語法，是在類別名稱的括號內註明要繼承的父類別。

示範程式碼

```
1  # E_8_5: 支票存款繼承的應用
2  # 定義新增支票帳戶繼承自 CreateBankAccount
3  class CreateBankAccount:
4      # 定義新增帳戶方法，額餘設定為 0
5      def __init__(self, id, name):
6          self.id = id
7          self.name = name
8          self.balance = 0
9      ...
```



```
14     def __deposit(self, amount):
...     ...
19 class CheckingAccount(CreateBankAccount):
20     def __init__(self, id, name):
21         super().__init__(id, name) # 呼叫父類別 __init__()
22         self.overdraftlimit = 30000 # 透支額度
23         # 考慮透支額度的提款方法
24     def withdraw(self, amount):
25         if amount <= self.balance + self.overdraftlimit:
26             self.balance -= amount
27         else:
28             raise ValueError(' 超出信用 ')
29         return self.balance
30     def __str__(self):
31         return (super().__str__() +
```



```
32         '\n 透支額度 \t' + str(self.overdraftlimit));
33 customer_c = CheckingAccount('k001', 'Angela Merkel')
34 print(customer_c.__str__())
35 #print(' 存款後餘額:',customer_c.deposit(150))
36 print(' 存款後餘額:',customer_c._CreateBankAccount__deposit(150))
37 print(' 提款後餘額:',customer_c.withdraw(50))
```

程式說明

▶ 父類別 CreateBankAccount

子類別與父類別可以在同一支程式，程式碼第 1 到第 18 列同【範例 8-3】。

▶ 子類別 CheckingAccount

接著定義 CheckingAccount 子類別，其形式參數為 CreateBankAccount，即代表所有的屬性與方法可以繼承自 CreateBankAccount（見第 19 到 32 列）。super() 方法的說明見下一段。設定透支額度（overdraftlimit）期初金額為 30000（見第 22 列）。



► 方法改寫 override

子類別重新定義一個與父類別同名 `withdraw()` 的方法，即考慮透支額度的提款方法（見第 24 列）。判斷傳進來的提款金額（`amount`）若小於等於餘額（`balance`）加上透支額度為真時，則餘額扣掉提款金額後回傳（見第 25 到 26 列）；判斷結果為否時，即提款金額高於餘額加上透支額度，便啟動例外機制印出 `ValueError` 的提示字串（見第 27 到 28 列）。

► 繼承父類別 `super()`

`super()` 的用途是呼叫父類別，直接繼承父類別的方法（見第 21 與 31 列）。以 21 列為例，使用 `super()` 方法繼承父類別的 `__init__()` 的 `id`, `name` 與 `balance` 屬性；第 31 列則繼承父類別的 `__str__()`，印出帳號、姓名、初期餘額再加上子類別另外定義的透支額度等資訊。

► 私有方法

請將第 35 列的註解解開，同時將第 36 列加上註解（最左邊加入 `#`），重新執行程式即會出現屬性錯誤 `AttributeError: 'CheckingAccount' object has no attribute`



'deposit'，表示無法順利繼承父類別的私有方法。但是，Python 並不是完全無法呼叫私有方法或私有屬性，只要知道完整的類別名稱與方法名稱。即寫明物件實體 `_` 類別方法名稱，類別名稱前加一個底線，方法名稱前加兩個底線。一樣可以執行父類別私有方法（見第 36 列）。

► 建構實體

呼叫 `CheckingAccount` 類別傳入實際參數含帳號與姓名，其回傳值是物件實體 `customer_c`（見第 33 列），呼叫實體 `customer_c` 的 `__str__()` 方法（見第 34 列），實際是執行程式碼的第 30 到 32 列）。其中，第 32 列會先多重繼承 `super()` 父類別的 `__str__()` 印出帳號、姓名、期初餘額之後再加上透支額度。呼叫實體的存款方法傳入 150（見第 36 列），呼叫實體的提款方法傳入 50（見第 37 列）。



執行結果

```
1 帳號 : k001
2 姓名 : Angela Merkel
3 期初餘額 : 0
4 透支額度    30000
5 存款後餘額 : 150
6 提款後餘額 : 100
...
ValueError: 超出信用額度
```

結果說明

呼叫 `customer_c.__str__()` 印出帳號、姓名、期初餘額以及透支額度（見執行結果第 1 到 4 列）。呼叫實體的存款方法傳入 150，印出存款後餘額（見第 5 列）；呼叫實體的提款方法傳入 50，印出存款後餘額 100（見第 6 列）。

若將程式碼第 36 列的提款金額（即實際參數）代入大於餘額加透支額度（例如：50000）時，才會印出 `ValueError: 超出信用額度`。



8-6 錯誤與異常

程式設計最常面對的是錯誤（指語法錯誤，syntax error）處理。程式中的異常（exception）是指程式執行期間出現錯誤。Exception 是程式設計師本身可以處理的錯誤，可以用 try 及 except 處理異常發生時相對應的程序與結果。



範例 8-8 引發異常

計算整數與字串相加引發資料型別錯誤（TypeError）的處理機制。

示範程式碼

```
1  # E_8_8: Raise 引發異常
2  try:
3      a=100
4      b='50'
5      num=a+b
6      raise TypeError
7  except TypeError as e:
8      print('字串不能算術運算:' + str(e))
9      num=int(a)+int(b)
10     print('num =', num)
```



程式說明

`try` 指令區放了運算式 `a+b` (見第 3 到 5 列); 安插可能引發資料型別錯誤的異常 (見第 6 列); 若發生異常則進行 `except` 的比對 (見第 7 列); 若為 `TypeError` 則進入指令區進行型別轉換後的運算後, 印出結果 (見第 8 到 10 列)。

執行結果

```
1 字串不能算術運算: unsupported operand type(s) for +: 'int' and 'str'  
2 num = 150
```

結果說明

引發異常資料型別錯誤, 印出提示字串 (見執行結果第 1 列)。進入 `except` 指令區, 進行型別轉換加總後印出結果 (見第 2 列)。



範例 8-9 將累加的程式，加入異常處理機制

定義一個累加的程式，代入的參數 (1) 必須是整數，而且 (2) 要大於零，加入異常處理機制。

示範程式碼

```
1  # E_8_9: 計累加的程式，加入異常處理機制
2  def sumn(n):
3      sn=0
4      if n<=0:
5          raise ValueError(' 輸入的參數必須大於零哦 ')
6      else:
7          for i in range(1, n+1):
8              sn=sn+i
9          return(sn)
10 while True:
```



```
11     try:
12         num=int(input(' 請輸入正整數 : '))
13         result=sumn(num)
14     except ValueError as v:
15         print(' 數值錯誤 , ' + str(v))
16     except:
17         print(' 未知錯誤 ')
18     else:
19         print('1 加到 %d = %d' %(num, result))
20     break
```



程式說明

▶ sumn 函數

檢測傳進來的參數是否小於等於零，若是則引發數值異常機制（見第 4 到 5 列）。計算累加功能見 7 到 9 列）。

▶ while 迴圈

程式進入點為 **while** 迴圈，其條件式設為 **True**，等於是無條件進入迴圈，停止條件必須設在迴圈內，而且必須確認一定會被執行，否則會進入無窮迴圈（見第 10 列）。**try** 指令區內先輸入要累加的整數，回傳值儲存於 **num**（見第 12 列）。代入 **num** 呼叫 **sumn** 自訂函數，回傳值為 **result**（見第 13 列）。進入 **sumn** 函數執行，若沒有異常則執行累加（見第 7 到 9 列）。若發生異常，則回到迴圈比對 **except** 內的異常物件，比對符合則進入指令區顯示錯誤訊息。反之，沒有發生異常則進入 **else** 指令區，將 **result** 的結果印出後離開迴圈，此即停止條件（見第 18 到 20 列）。



執行結果

```
1 請輸入正整數 : d
2 數值錯誤，invalid literal for int() with base 10: 'd'
3 請輸入正整數 : 2.3
4 數值錯誤，invalid literal for int() with base 10: '2.3'
5 請輸入正整數 : 5
6 1 加到 5 = 15
```

結果說明

執行程式若輸入非數值例如：d，則會引發 `ValueError` 異常，顯示提示字串（見執行結果第 1 到 2 列）。若輸入 2.3 實數結果相同（見第 3 到 4 列）。輸入 5，正整數則會正常執行印出累加的結果後離開迴圈結束程式（見第 5 到 6 列）。

自訂套件

實務案例 8-3 以類別為單位組織套件

本範例將以類別為單位組織套件，這是比較推薦的自訂套件的設計方法。類別內可以收納相同性質的方法，一個類別儲存成 + 一個 .py 的檔案。也就是一個套件內可以有多個類別；每個類別可以有多個方法。本案例的套件名稱設定為 PCLin_Finance，資料夾下有 __init__.py 之外，另外兩個資料夾分別為 TimeValueClass 與 InterestsClass。此兩個資料夾下除有 __init__.py 外，另外有 TimeValue.py 與 Interest.py。

- ▶ TimeValue 類別：TimeValue.py 檔案的內容是定義 TimeValue 類別，此類別內有 fvfix 與 pvfix 兩個有關貨幣時間價值的方法。
- ▶ Interest 類別：Interests.py 檔案的內容是定義 Interest 類別，此類別內有 earfix 與 ccrfix 兩個有關利率的方法，套件的架構圖見圖 8-18。

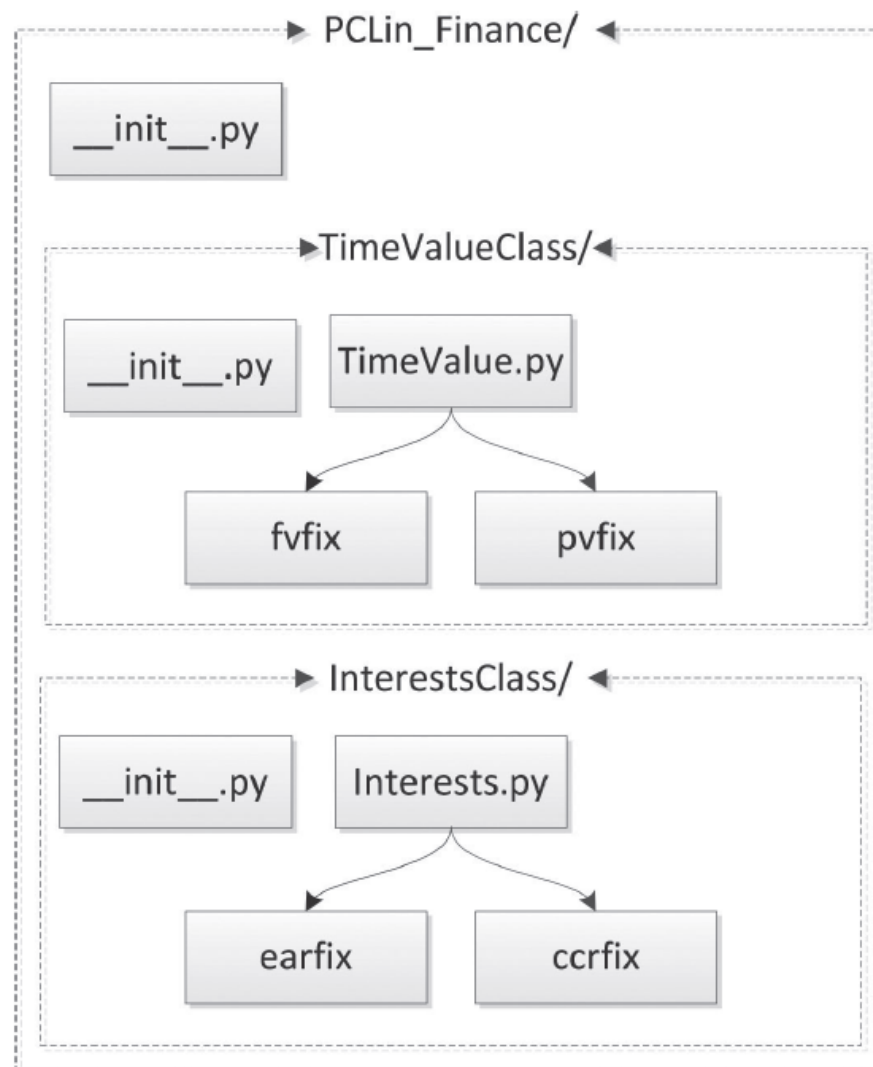


圖 8-18 PCLin_Finance 套件的架構圖

► 連續複利

貨幣時間價值的類別參考【實務案例 8-1】。有效利率的範例參考【實務案例 8-2】，以下說明連續複利（ccrfix）簡介。

連續複利指每期現金流量依利率與期間，先計算本金加利息合計（本利和），再以此做為計算下期計算利息的基底，重複計算每一期的利息，加總後即為這段貨幣期間的價值，即為俗稱的利滾利。連續複利計算公式如下：

$$ccr = m * \ln\left(1 + \frac{rn}{m}\right)$$

其中， rn 為名目利率； m 為 1 年中的複利次數。



示範程式碼

```
1  #RC_8_3: 將貨幣時間價值相關的方法建立成套件
2  import PCLin_Finance.TimeValueClass.TimeValue as tv
3  import PCLin_Finance.InterestsClass.Interests as rt
4  # 設定初始值
5  pv=100
6  fv=115.92740743
7  i=0.03
8  n=5
9  rn=0.06
10 m=int(input(' 請輸入年複利次數 : 1, 2, 4, 12, 52, 365 = '))
11 # 呼叫 TimeValue 類別，建構物實體
12 tv1=tv.TimeValue()
```



```
13 # 呼叫物件實體的方法
14 print('%d 年後的終值 = %10.4f' %(n, tv1.fvfix(pv, i, n)))
15 print('%d 年後的現值 = %10.4f' %(n, tv1.pvfix(fv, i, n)))
16 # 呼叫 TimeValue 類別，建構物實體
17 rt1=rt.Interest()
18 # 呼叫物件實體的方法
19 print('有效利率 = %8.4f%%' %(rt1.earfix(i, m)*100))
20 print('連續複利 = %8.4f%%' %(rt1.ccrfix(i, m)*100))
```

程式說明

► 匯入套件與模組

匯入貨幣時間價值與利率兩個套件模組，分別取別名為 tv 與 rt（見第 2 到 3 列）。

► 設定初始值

貨幣時間價值套件會用到的初始值（見第 5 到 8 列）；名目利率設定為 0.03（見第 9 列）；請使用者輸入一年的複利次數回傳給 m（見第 10 列）。



► 建構物件實體與呼叫實體的方法

- (1) 呼叫 tv 套件模組別名 `.TimeValue` 類別，建構物件實體 `tv1`（見第 12 列）。呼叫物件實體的方法 `tv1.fvfix()` 與 `tv1.pvfix()`，將回傳值印出（見第 14 到 15 列）。
- (2) 呼叫 rt 套件模組別名 `.Interest` 類別，建構物件實體 `rt1`（見第 17 列）。呼叫物件實體的方法 `rt1.earfix()` 與 `rt1.ccrfix()`，將回傳值印出（見第 19 到 20 列）。



示範程式碼

```
1  #TimeValue: 定義貨幣相關方法的類別
2  class TimeValue:
3      # 終值
4      def fvfix(self, pv, i, n):
5          #fvfix: 計算終值公式
6          fv=pv*(1+i)**n
7          return(fv)
8      # 現值
9      def pvfix(self, fv, i, n):
10         #pvfix: 計算現值公式
11         pv=fv/((1+i)**n)
12         return(pv)
```

程式說明

此為 TimeValue.py 的程式內容，內容是一個類別 TimeValue，類別內含 fvfix() 與 pvfix() 兩個方法，程式碼內容說明參考【實務案例 8-1】。



示範程式碼

```
1  #Interests: 定義利率相關方法的類別
2  class Interest:
3      # 有效利率
4      def earfix(self, rn, m):
5          #earfix: 計算有效利率公式
6          ear=((1+(rn/m))**m)-1
7          return(ear)
8      # 連續複利
9      def ccrfix(self, rn, m):
10         import math
11         #ccrfix: 計算連續複利公式
12         ccr=m*math.log(1+(rn/m))
13         return(ccr)
```



程式說明

此為 `Interests.py` 的程式內容，內容是一個類別 `Interest`，類別內含 `earfix()` 與 `ccrfix()` 兩個方法。`earfix()` 內容公式說明參考【實務案例 8-2】。

`ccrfix()` 內的程式碼，需要運算 \ln 自然對數，所以要引用 `math` 模組（見第 10 列），依連續複利公式撰寫的程式碼，其回傳值為 `ccr`（見第 12 到 13 列）。

執行結果

- 1 請輸入年複利次數：1, 2, 4, 12, 52, 365 = **365**
- 2 5 年後的終值 = 115.9274
- 3 5 年後的現值 = 100.0000
- 4 有效利率 = 3.0453%
- 5 連續複利 = 2.9999%

結果說明

請使用者輸入年複利的次數：假設輸入的值是 365（見執行結果第 1 列）。先計算呼叫 **TimeValue** 類別內的兩個方法終值與現值，計算結果分別為 115.9274 與 100（見第 2 到 3 列）。再著呼叫 **Interest** 類別內的兩個方法有效利率與連續複利，計算結果分別為 3.0453% 與 2.9999%（見第 4 到 5 列）。



提示

1. 類別名稱要與檔案名稱相同，例如：Class TimeValue: 與 TimeValue.py 相同。
2. 若將 PCLin_Finance 資料夾剪下，貼到使用者於前節所說的環境變數下，例如：C:\Users\user\Anaconda3\Lib\PCLin_Finance。即可做到同一台電腦，不同程式在不同路徑下也可以自由匯入套件。



pip 安裝第三方套件

若 Python 內建的標準函數庫內沒有提供程式設計時需要用到的模組，可以安裝第三方套件的模組。到 PyPI（Python Package Index）查詢是否有合適功能的套件（<https://pypi.python.org/pypi>）。

安裝 Python 第三方套件的方法有好幾種，本文介紹最簡單最好用的 `pip` 指令。可以在 windows 命令視窗使用 `pip` 指令自動安裝第三方套件，語法如下：

```
pip install 套件名稱
```

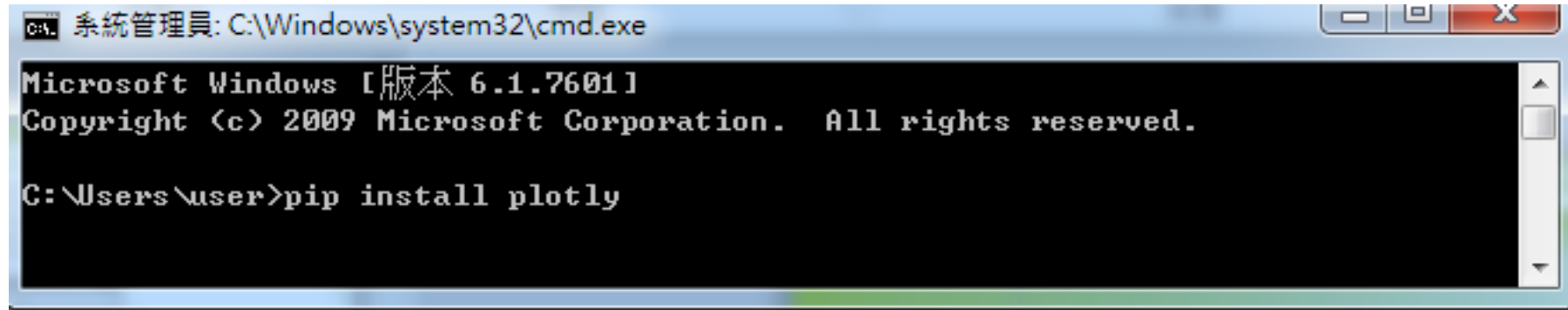


提示

要在 windows 的命令視窗（在開始程式集輸入 `cmd` 按 `Enter`）中執行 `pip`。而不是在 Python 的命令視窗中執行。

安裝 *plotly*

- 假設，要 **安裝 *plotly*** 第三方套件，可在 windows 命令視窗輸入 `pip install plotly` 後按「Enter」，



```
系統管理員: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\user>pip install plotly
```

- Python `setup.py` intall

本章講解完畢

現場同學們如有不懂的地方，
請提出問題。

