

# 程式語言基本概念

## 學習綱要

- ⊞ 1-1 程式語言與程式設計的演算法
- ⊞ 1-2 程式語言開發環境的操作
- ⊞ 1-3 物件導向程式設計的認識
- ⊞ 1-4 程式設計相關議題
- ⊞ 1-5 本章內容摘要

## 學習表現

- ⊞ 1. 了解程式語言在生活上的應用與對科技創新的影響。
- ⊞ 2 能思辨勞動法令規章與相關議題，省思自我的社會責任。

## 1-1 程式語言與程式設計的演算法

我們人人手上的手機與電腦之所以讓人愛不釋手，主要是這些設備功能非常豐富，現代人每天生活都已經離不開手機與電腦的程式應用服務之中，例如：

1. 手機與電腦的作業系統與應用程式。隨著資訊科技的進步，現在已經是人人有手機，家家有電腦或筆電，手機與電腦的作業系統與應用程式都是程式的應用。
2. 通訊軟體。隨著網路的普及，在家用手机與電腦，搭配通訊軟體 Facebook、Line 等等，就能得到全世界所有的資料、影片，這些網路傳輸軟體也是程式應用。例如，可以聊天、互傳訊息、看線上球類比賽、網紅直播等。
3. 工業自動化（Industrial Automation）。現代的工業生產線，已經大部分由機器人取代人力，機器人可自動焊接、組裝、運送，機器人與機器人也自動溝通，整個生產線的人力僅維護這些機器人的運轉，這些機器人的運轉與溝通也都是程式應用。
4. 商業自動化。現在所有的商業進銷存運轉、股票買賣、銀行金流、線上轉帳、網路購物等都是靠程式應用。圖 1-1 是新光銀行的轉帳系統，讓您不用出門就可以辦理查詢交易明細、轉帳等功能。



★ 圖1-1 銀行網路轉帳系統

5. 醫院自動化。現代醫院也不用人工傳送病歷了，所有的檢驗報告、醫囑、處方、收費、掛號等也都是程式應用。圖 1-2 是高雄榮總的網路掛號系統。讓您不用出門就可以完成看診預約。



★ 圖1-2 醫院網路掛號系統

6. 個人穿戴式裝置。隨著智慧型裝置的快速發展，手機將不是被動裝置，日後手機將會協助監控我們的身體健康，發出訊息提醒自己、或直接將訊息傳給醫療單位，作為協助就醫的依據等，這也是程式應用。圖 1-3 是一些常用個人穿戴式裝置。



★ 圖1-3 是一些常用個人穿戴式裝置。（摘自PCHOME購物網）

7. 物聯網（Internet of Things，簡稱 IoT）時代來臨。現在每個人都有手機，人與人都可溝通，下一步將是物與物直接溝通，冰箱東西吃完了，冰箱會直接叫貨，機器人可自駕，也可直接操作電梯將東西送到我們家；地上髒了、機器人也可以自動打掃等等，這些也都是程式應用。

能讓手機與電腦有如此豐富的功能，表示這些設備有預先安裝應用程式，這些應用程式隨時可依照使用者的需求提供服務，有如孔明的錦囊妙計。而這些應用程式都需要程式語言與程式設計，此即為本章的重點，以下說明什麼是程式語言、程式設計與程式設計演算法。

## ⚙️ 程式語言（Programming language）

人與人之間溝通的工具稱為語言，人類是由不同民族組成，因其發源地不同，所以就有許多語言。例如，華語、英語及德語等。其次，人與電腦溝通的工具，則稱為電腦程式語言。那麼為什麼沒有電視語言、冰箱語言或冷氣語言呢？那是因為這些機器的功能較為簡單，只要幾個按鈕就能發揮其功能。但是電腦的功能就非常多，能處理所有的數值與字串計算、可以同人類有判斷功能、可以無限重複或依某些條件重複某些事件等等智慧功能，而完成以上銀行轉帳、醫院掛號、股票買賣、穿戴式裝置等應用程式，這些功能多到連用整個鍵盤的所有按鍵都無法表現其功能，所以必須使用一些類似單字所組成的片語與敘述來發揮其所有功能，這些單字與片語的集合就稱為電腦程式語言，簡稱程式語言。就如同人類也無法用 26 個字母表達所有感受與思維，必須藉助這些字母的排類組合，先組成單字，再由單字組合成片語與句子，才能充分表達其思維。

## ⚙️ 程式設計（Programming）

使用程式語言命令電腦工作稱為程式設計。

## ⚙️ 程式設計的演算法

演算法 (Algorithm) 是指電腦程式完成一項工作所需要『步驟』的集合。演算法嚴謹定義如下：

在有限 (finite) 的步驟 (step) 所構成的集合中，依照給定輸入 (input)，依序執行每個明確 (definite) 且有效 (effective) 的步驟，以便能夠解決特定的問題，而且步驟的執行必定會終止 (terminate)，並產生輸出 (output)。

## ⚙️ 演算法的流程表示

常見的演算法流程表示有三種，分別是自然語言、虛擬碼與流程圖。分別說明如下：

### ► 自然語言 (Nature Language)

自然語言就是使用我們日常生活的文字表示或已經熟悉的數學語言。例如，以下是求 9 開根號的自然語言演算法。

1. 使用迴圈從 1 到 9 分別求其平方。例如，1 的平方是 1，2 的平方是 4，3 的平方是 9...，此稱為循序法。
2. 若其平方大於等於 9，此數即為所求。例如 3 的平方已經大於等於 9，3 即為所求。

### ► 虛擬碼 (Pseudo Code)

在自然語言中，有時嵌入一些慣用程式敘述，如 if、switch 或 match 代表決策（第四章介紹）for、while 代表迴圈（第五章介紹），如此可縮短文字長度，也會讓敘述更加清楚易懂。例如，以下是求 9 開根號的虛擬碼演算法：

```
for i=1 to 9{ #1,2,3,4,5,6,7,8,9
    y=i*i
    if y>=9 then
        print(i)
}
```

### ► 流程圖 (Flow Chart)

流程圖是利用各種方塊圖形、線條及箭頭等符號來表達演算流程的順序，常用的流程符號如表 (1-1)。

表1-1 常用流程圖符號表

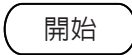

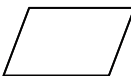

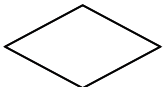






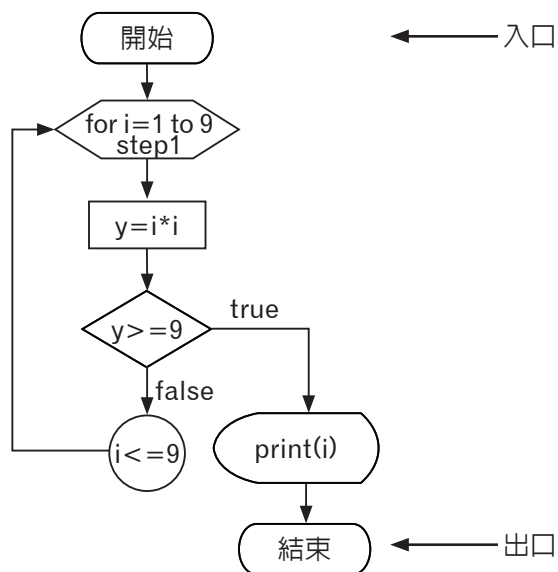
編號	符號	意義
1	 	起迄符號。 代表流程圖的開始或結束，此符號若是開始，則僅有一出口，若是結束則僅有一入口。
2		輸入與輸出符號。 用來填入輸入與輸出的符號，此符號有一入口與一出口。
3		處理符號。 用來填入處理程序，此符號有一入口與一出口。

表1-1 常用流程圖符號表 (續)

編號	符號	意義
4		決策符號。 用來表示決策分歧點，此符號的出口至少有兩個，分別是 true 或 false。
5		重複符號。 聲明重複指令的起點與離開條件，通常配合下面的連結符號表示重複的範圍。
6		連結符號。 可配合上面的重複符號或移至另一頁的起點。
7		副程式。(副程式又稱函式) 用來表示另一個副程式。
8		程式流向符號。 用來連結以上符號，說明程式的流向。
9		代表輸出至螢幕。 將資料由螢幕輸出。
10		代表輸出至印表機。 將資料由印表機輸出。

例如，圖 1-4 是求 9 開根號的流程圖演算法。



★ 圖1-4 開根號求解流程圖

### ► 演算法比較

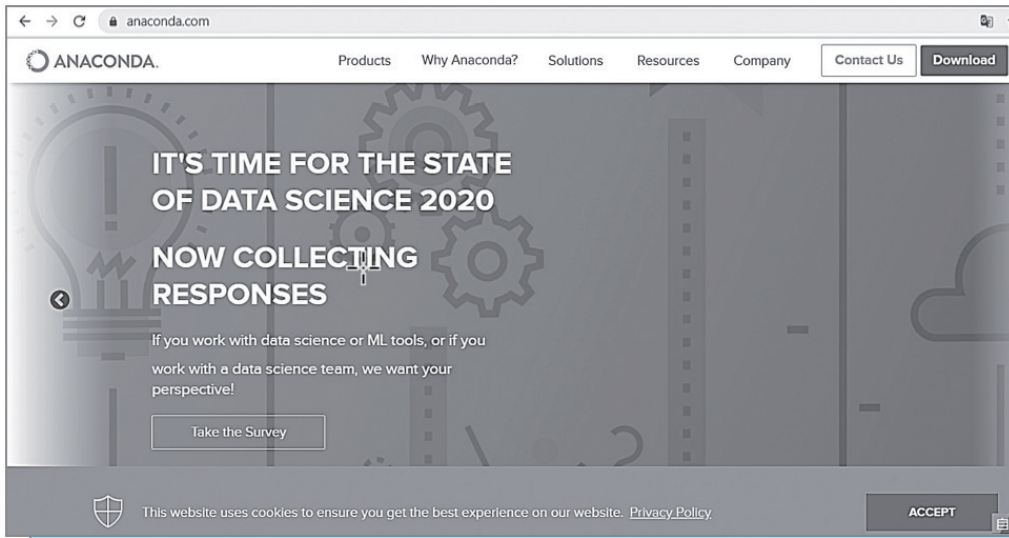
使用自然語言文字描述演算法，適合已經有程式設計經驗者，因為隨著問題的複雜度增加，初學者較難體會精簡的自然語言描述。虛擬碼使用類似程式碼的文字描述演算法，但不能夠直接執行，雖然能快速轉換成程式碼，適合已經有程式基礎的人員，因為有時過於精簡，初學者也難體會。流程圖使用鉅細靡遺的流程圖符號，雖然較耗時，但最能精準表現程式運算與流程，最適合初學者學習程式設計。

## 1-2 程式語言開發環境的操作

目前流行的程式語言，依其產出年份排列，有 C、C++、Visual Basic、Java、C#、Python 等語言。因為 Python 是目前最新的物件導向程式設計語言，且其抽象化等級最為先進、語言規定最少、功能也最進步、性能價值比(C/P 值)最高、初學者也最容易入門。所以本書選擇 Python 作為技術型高中商管群『程式語言與設計』的語言。

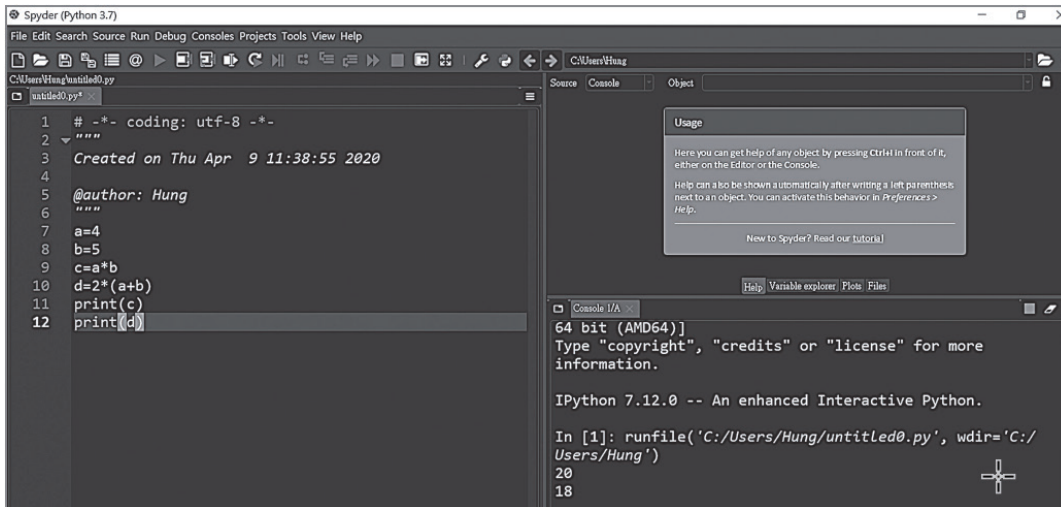
### ⚙️ Python 整合開發環境的操作

所謂的整合程式開發環境，是指可以在同一個畫面鍵入程式、存檔、執行程式、看到執行結果，若有錯誤也可在同一畫面除錯、再執行，直到滿意為止。Anaconda 雖不是 Python 官方正式出版的整合程式開發環境，但卻是目前最實用、最強的 Python 外掛整合開發環境，所以本書選用此編譯程式。Anaconda 官網([anaconda.com](http://anaconda.com))如圖 1-5，請依下圖點選「Download」下載安裝執行檔。另外，因為是執行檔，所以只要按照指示即可完成安裝。



★ 圖1-5 Anaconda官網首頁

以上執行檔安裝完成後，會在程式集出現「Anaconda」資料夾，請點選資料夾裡面的「Spyder」即可進入整合開發環境，如圖 1-6。以下是 Spyder 開新檔案（點選功能表的「File/NewFile」）的畫面。左邊窗格是程式編寫區，右上方窗格是輔助說明區，右下方窗格是程式執行輸出區。



★ 圖1-6 Spyder整合視窗

左邊的程式編寫區中，第一行的井號「#」與兩個「三雙引號（""""）」之間都是註解，此註解記載此程式開發的時間與作者姓名。這些文字僅給人看，電腦都不解譯。

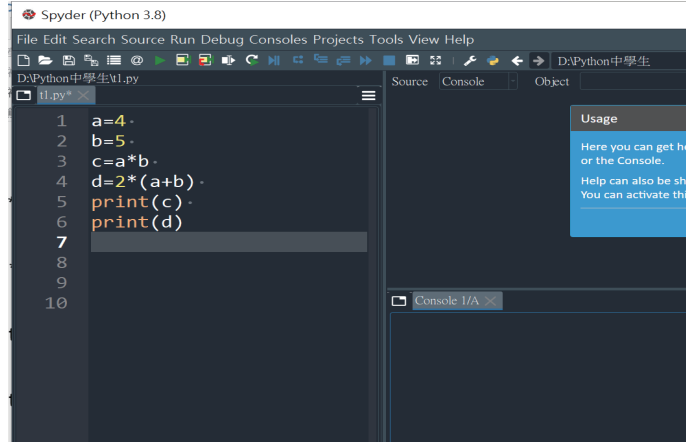


## ⚙️ 程式編輯

圖 1-7 是一個已知長方形邊長，計算面積與周長的程式，請同學練習鍵入這些程式敘述。圖 1-8 則是寫入此程式的畫面。


```
a=4
b=5
c=a*b
d=2*(a+b)
print(c)
print(d)
```

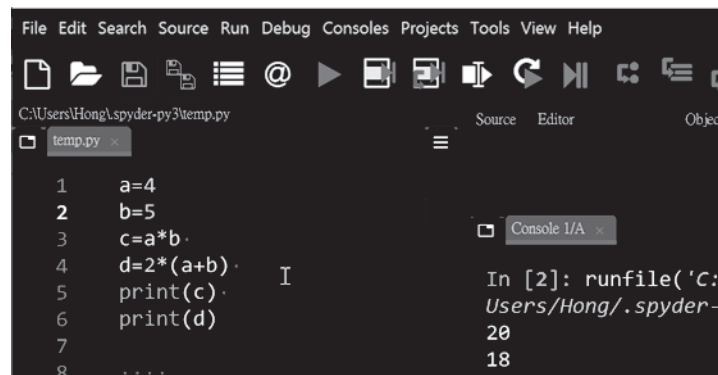
✦ 圖1-7 計算長方形面積程式



✦ 圖1-8 Spyder編輯視窗

## ⚙️ 程式的執行

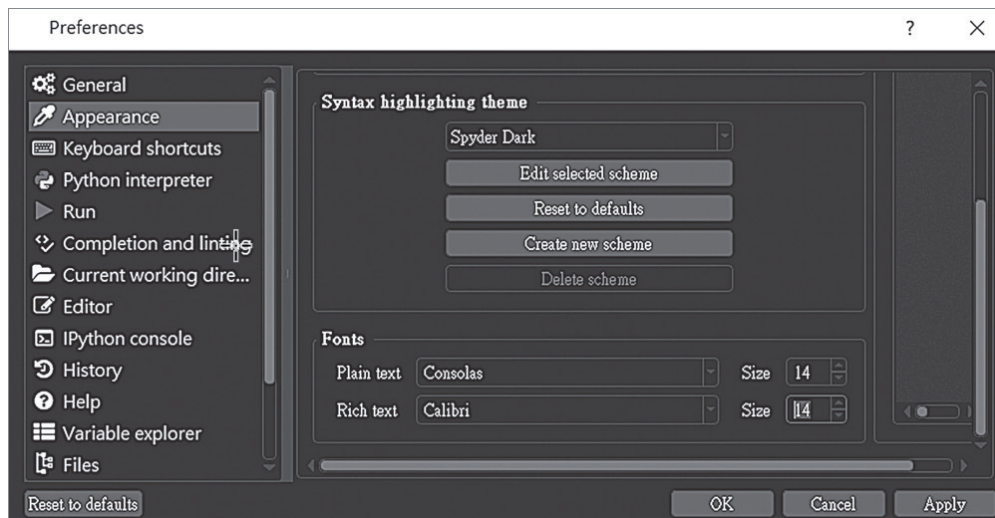
圖 1-9 是點選工具列的  「Run file (F5)」的畫面，或點選功能表的 Run/Run」亦可執行程式。因為是第一次執行，電腦會出現存檔對話框，要求先存檔，請點選資料夾，填入主檔名就好（例如可填入 t1），附檔名預設為「.py」，不用鍵入。執行結果就在右下窗格。若有任何錯誤，請自己修改，然後再執行程式即可。



✦ 圖1-9 Spyder執行結果畫面

## ⚙️ 修改編輯環境

點選功能表的「Tools/Preferences/Appearance」，畫面如圖 1-10，可在此修改程式字型的大小。



★ 圖1-10 Preferences微調視窗

## 👉 自我練習

以下是解一元二次方程式  $x^2+4x-5=0$  的程式，請自行使用 Python 整合開發環境輸入，並觀察輸出結果是否為 1 與 -5。

```
a = 1 ; b = 4 ; c = - 5
d = ( b ** 2 - 4 * a * c ) ** ( 1 / 2 )
x1 = ( - b + d ) / ( 2 * a )
x2 = ( - b - d ) / ( 2 * a )
print ( x1 )
print ( x2 )
```

## 1-3 物件導向程式設計的認識

人類之所以會是萬物之靈，其中一個主要原因是人類可以在錯誤中成長。物件導向的程式設計（Object Oriented Programming, OOP）也是人類在程式語言中逐漸累積的成果，這個觀念在 1970 年代就已提出，只是當時時機未到，未受到任何程式語言支持。現在，OOP 則已是所有程式語言的基本設計理念，為了說明 OOP 大行其道的原因，我們將程式語言的發展分為 3 個時期，分別是非程序導向、程序導向及物件導向，三者分別說明如下。

### ⚙️ 非程序導向（NonProcedure Oriented）

早期的程式語言，並沒有內儲副程式（Subroutine）。當開發新的應用程式時，如果某一功能與之前寫過的程式相近，則會將此段已完成的程式整段複製，並稍加修改即可重新加以利用。但是，這些程式的分身包括本尊，自從複製出來以後就開始以各自的方式發展，結果造成各版本的差異越來越大，這些程式很難分辨誰複製誰，彼此之間也難再共用某些程式碼，當遇到錯誤或欲新增功能時，更是很難逐一修改所有程式。

例如：以組合計算  $C_n^m$  為例， $C_n^m = \frac{m!}{n!(m-n)!}$ ， $m!=1*2*3*\dots*m$ （此為階乘計算），因為非程式導向時代沒有函式，階乘計算共要寫 3 次，所以程式如下：

```
m=5
n=2
#計算階乘
s1=1
for i in range(1,m+1):
    s1=s1*i
#計算階乘
s2=1
for i in range(1,n+1):
    s2=s2*i
#計算階乘
s3=1
for i in range(1,m-n+1):
    s3=s3*i
print(s1/(s2*s3))#10
```

以上計算階乘共要寫 3 次，程式有點冗長，若要修改階乘運算，更要在 3 個地方同時修改，若沒有完全修改，將造成錯誤結果。

### ⚙️ 程序導向 (Procedure Oriented)

為了解決以上程式共用問題，各編譯器廠商便將一些常用功能寫成函式。較有規模的軟體設計公司亦會將常用的函式集中在一個函式庫，旗下的軟體產品一律呼叫這些標準的函式庫，而不是從函式庫複製出來修改，此即為程序導向的程式設計。程序導向與非程序導向相比，的確解決了程式共用的問題，但是，人們並不以此為自滿，有些問題還是不夠順暢。例如：有些函式庫會隨著人們需求的增加而有不同版本，當某些函式功能增加時，只好重新取函式庫的函式加以修改，並賦予新的名稱。如此日積月累，我們的函式庫已有許多函式，這些函式有的功能相近、有的是前後版本不同、有的函式裡的變數來龍去脈不明，造成使用者的混亂。為了突破以上瓶頸，於是有物件導向的發展，以解決以上程序導向的不足。

以前面計算  $C_n^m$  為例，階乘計算共 3 次，則先以函式完成階乘計算，往後所有程式都可使用此函式，這樣若要修改函式內容，只要統一在函式內修改，所有程式都有相同結果。以上程序導向程式如下：

```
#先以函式完成階乘計算
def fsum(a):
    s=1
    for i in range(1,a+1):
        s=s*i
    return s
#主程式
m=5
n=2
s1=fsum(m) #呼叫fsum函式
s2=fsum(n) #呼叫fsum函式
s3=fsum(m-n) #呼叫fsum函式
print(s1/(s2*s3))#10.0
```

## ⚙️ 物件導向 (Object Oriented)

程序導向中的函式，存有許多解決問題的函式（函式在目前的物件導向程式設計中稱為『方法』），它是偏重在方法的解決。但是，人類的生活方式不僅是單純的行為描述，更存在著屬性的記載。例如：當在描述一個人時，通常描述如：『他的名字是洪國勝，身高 172、體重 70，具有滾進、游泳及跑步的能力』；又例如：描述一輛車子時，其描述如：『它的名字是 SENTRA，排氣量是 1600c.c.，耗油量是每公里 0.1 公升，且具有每小時 120 公里的移動能力』。以上人與車即稱為『物件』，名字、身高、排氣量則稱為『屬性』或稱『資料』，而滾進、游泳、跑步、移動則稱為『方法』。既然真實世界的事件都有『資料』與『方法』，程式設計亦不應侷限在狹隘的函式，僅偏重解決問題的方法，而是應以物件的宏觀角度撰寫程式。所以，基於物件導向的新觀念，程式開發工具即制訂一種新的型態，此型態稱為『類別』。每一個類別都有屬於自己的『資料』與『方法』。有了『類別』，我們可將眾多的函式依照類別存放，如此可解決目前日益龐大的函式命名與函式取用的困擾。例如：程序導向的時代，關於開門的函式即有『電梯開門』、『汽車開門』、『房子開門』等數種開門的『方法』，如此徒增命名與取用的困擾。但在物件導向的領域裡，開門這個方法是附在相對應的類別裡。例如：於電梯類別裡有電梯的開門，汽車類別有汽車的開門，房子類別裡有房子的開門方法，大家的方法名稱都叫『開門』，撰寫程式時也是『電梯.開門』，『汽車.開門』，或是『房子.開門』（補充說明：物件與方法、屬性之間以點「.」運算子連結），如此既可簡化程式的撰寫，亦可減少程式出錯的機會。但是，在程序導向的領域裡，所有的函式都集中，就有可能用錯方法。例如：用開電梯門的方法去開汽車的門，結果當然是錯的。

## ⚙️ 物件 (Object)

物件導向的程式設計是先建立類別，就如同先建立產品模型，然後我們使用類別建立物件，並且可以將此物件指派給變數，物件亦稱為類別的實現或類別的樣例化 (Instance)。也就是說，類別就像是一個模型，在這些模型裡面已設計好他所具備的資料與方法，當您需要物件時，只要依照這些模型樣例去產生一個或多個物件即可。

再以前面計算 $C_m^n$ 為例，函式導向僅在乎函式，物件導向則包含資料成員與函式成員，這樣才能完整展現此物件樣貌。因為階乘的計算，包括輸入、輸出等資料成員與計算階乘的函式成員，所以程式如下：

```
#建立 Factorial 類別
class Factorial:
    a=1 #資料成員 階乘輸入
    b=1 #資料成員 階乘輸出
    def sum(self): #函式成員，計算階乘
        s=1
        b=self.a
        for i in range(1,b+1):
            s=s*i
        self.b=s #將計算結果指派給b
#主程式
m=5
n=2
c=Factorial() #以c變數樣力產生1個物件
c.a=m #指派輸入值
c.sum() #計算階乘
s1=c.b #傳回結果
c.a=n #指派輸入值
c.sum() #計算階乘
s2=c.b #傳回結果
c.a=m-n #指派輸入值
c.sum() #計算階乘
s3=c.b #傳回結果
print(s1/(s2*s3)) #10.0
```

## ⚙️ 抽象化 (Abstraction)

什麼是抽象化呢？例如，拍照與攝影是一種具體的表現，但是繪畫就需要抽象化。因為人類不可能、也沒有需要一五一十的具體描繪，所以需要抽象化。優秀的抽象化，更要能以很簡潔的描述就能夠傳達繪畫者的意念。所以畢卡索的抽象畫就很精簡，但卻能讓普羅大眾讚賞，甚至不同層面的士農工商與販夫走卒都能感受到自己想要的喜悅。程式設計也是相同的道理，亦要依照不同的需求給予抽象化，讓程式設計者都能很簡單的描述，編譯器就會依照不同的需求編譯與連結不同的程式，而完成不同使用

者的需求。物件導向所提供的多型(Polymorphism)、封裝(Encapsulation)、繼承(Inheritance)等，都可以實現不同等級的抽象化，以解決程序導向的不足。以上多型、封裝、繼承說明如下：

### ▶ 多型

同一種形式，但編譯器能依照使用者的需求，進行適當的處理。例如：以下同樣是加法運算子「+」，但編譯器卻能依照使用者的需求，找到對應的編譯方式，完成數值相加或字串串接，此即為編譯器多型的表現。

```
print(3+4)      #結果是7
print("3"+"4") #結果是字串串接34
```

### ▶ 封裝

程式設計的開發與其它的工業，如汽車、電視、收音機等機械、電子產品相較，可說是起步較晚的領域，在汽車、電視及收音機等產品上，我們發現這些產品按鈕或旋扭有不同的封裝等級，有些是留給使用者操作用，所以就放在明顯的地方讓使用者操作機器，有些則是出廠微調後就隱藏或稱封裝起來，僅供日後技術人員維修用，這樣才不會被使用者任意操作而破壞原先功能。程式設計也是這樣，有些功能可讓使用者操作，有些功能則應封裝，才不會被任意破壞。

### ▶ 繼承

任何工業或電子產品的開發，均不是無中生有，而是從舊有的產品中繼承某些特性，再加入新的零件或修改部份零件而成一項新的產品。例如：SENTRA180 正是繼承 NEW SENTRA 而來，只是排氣量提高了、內裝豪華了，但是原來的輪胎、方向盤及座椅還是用原來 SENTRA 的東西，這就是繼承的道理，使得新產品的開發得以縮短時程。程式設計也是相同的道理，先建立類別，往後都可以繼承一個或數個類別，再新增或修改方法，這樣才能節省程式開發流程。

## 1-4 程式設計相關議題

### ⚙️ 程式語言對科技創新的影響

電動車、無人駕駛等技術日益成熟、線上購物網路交易金額日益蓬勃發展、單晶全球鬧缺貨、5G 網路日益普及，這些創新科技都需要程式設計的支援，所以學會程式設計就能站在科技創新的尖端。

### ⚙️ 資料保護及資訊安全

網路的發展可說無遠弗界，全世界的電腦都網網相連，國家發展委員會為規範個人資料之蒐集、處理及利用，以避免人格權受侵害，並促進個人資料之合理利用，特制定「個人資料保護法」，如圖 1-11 所示：



▲ 圖1-11 個人資料保護法（摘自法規資料庫）

例如，您寫程式時，將會面對很多客戶的資料，這些資料程式設計人員也不得任意使用、公開、販賣，請資訊相關從業人員，應該先行瀏覽，才能保護自己，以免執業過程不甚觸法，得不償失。



## ⚙️ 妨害電腦使用罪

我國一位某大學工學院的學生，曾經因為在 1998 年設計了名為「CIH」的電腦病毒，這種病毒在當年造成全球 6000 萬台電腦受到破壞，該名學生當年散佈電腦病毒後「逍遙法外」，主要因為當時我國刑法並未針對「散佈電腦病毒」之行為加以明確規範。

隨著資訊科技日新月異的發展，利用電腦及相關設備的犯罪也日益增加。因此，我國於民國九十二年六月二十五日增定刑法第三十六章「妨害電腦使用罪」，以便有效赫阻類似的犯罪行為，其中第三百六十條之「干擾他人電腦使用罪」，便是明確處罰散佈「電腦病毒」之規定，該條規定：「無故以電腦程式或其他電磁方式干擾他人電腦或其相關設備，致生損害於公眾或他人者，處三年以下有期徒刑、拘役或科或併科十萬元以下罰金。」

次外，鑑於電腦病毒這種惡意的電腦程式，對於電腦系統安全性危害甚鉅，往往造成重大的財產損失，致生損害於公眾或他人，鉅額損失往往難以估計，因此，實有必要對於此類電腦病毒的程式設計者，加以處罰的必要，故刑法第三百六十二條規定：「製作專供犯本章之罪之電腦程式，而供自己或他人犯本章之罪，致生損害於公眾或他人者，處五年以下有期徒刑、拘役或科或併科二十萬元以下罰金。」。刑法第三十六章妨害電腦使用罪如圖 1-12（以上摘自教育部資訊教育司 <https://depart.moe.edu.tw/>）



★ 圖1-12 妨害電腦使用罪（摘自全國法規資料庫）

## ⚙️ 勞動基準法

勞動部為規定勞工勞動條件最低標準，保障勞工權益，加強勞雇關係，促進社會與經濟發展，特制定勞動基準法，請開啓全國法規資料庫 (<https://law.moj.gov.tw/LawClass/LawAll.aspx?PCode=N0030001>)，如圖 1-13。

內有許多勞工勞動權益，請同學自己瀏覽，才能保障自己勞動權益。例如，程式設計容易被歸納為「包工制」，有時有些突發狀況，上班時間是否過長、長時間沒有休假等，這都會影響身體健康，請從事程式設計工作人員翻閱勞基法，適度的向公司反應。



★ 圖1-13 勞動基準法 (摘自全國法規資料庫)

## 1-5 本章內容摘要

1. 常見的演算法流程表示有三種，分別是自然語言、虛擬碼與流程圖，其中以流程圖最適合初學者。
2. 程式語言的發展分為 3 個時期，分別是非程序導向、程序導向及物件導向。
3. 物件導向的開發程序是先建立類別，再由類別產生 1 個或多個物件。
4. 物件導向所提供的多型（Polymorphism）、封裝（Encapsulation）、繼承（Inheritance）等，都可以實現不同等級的抽象化，以解決程序導向的不足。
5. 程式設計師設計程式時，必然看到很多個人隱私資料，請瀏覽「個人資料保護法」，不能散播、販賣個人資料，以免觸法。
6. 刑法第三十六章，有訂定「妨害電腦使用罪」，內有一些妨害他人使用電腦的刑法。
7. 勞動基礎法內有保障勞工基本勞動安全的規章。

## 課後評量

### 一、填充題

1. 寫出 3 種常用演算法的表示。\_\_\_\_\_，\_\_\_\_\_，\_\_\_\_\_。
2. 寫出流程圖符號的開始符號。\_\_\_\_\_
3. 寫出流程圖符號的輸出、輸入符號。\_\_\_\_\_
4. 寫出流程圖符號的運算處理符號。\_\_\_\_\_
5. 寫出流程圖符號的決策符號。\_\_\_\_\_
6. 寫出流程圖符號的迴圈重複符號。\_\_\_\_\_
7. 寫出流程圖符號的連結符號。\_\_\_\_\_
8. 同樣是「+」運算子，但編譯器能依照使用者的需求，進行數值相加或字串串接，此種特性在物件導向稱為\_\_\_\_\_。
9. 開發一個方法，但某些參數不想讓使用者任意更動，在物件導向裡稱為\_\_\_\_\_。
10. 開發新的類別不是從零開始，而是從現有類別新增與修改方法，在物件導向程式設計稱為\_\_\_\_\_。