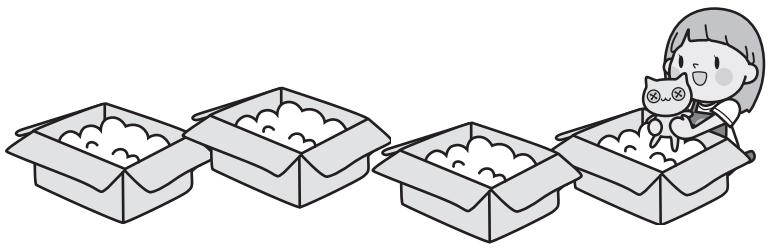
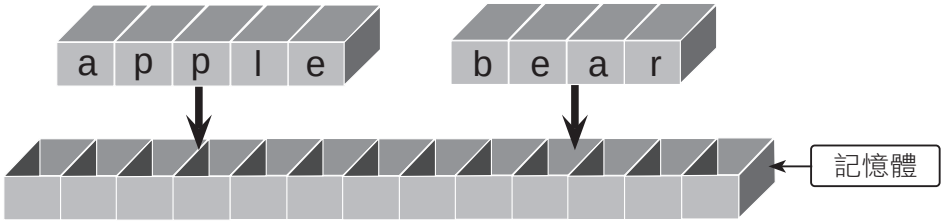
2.1 變數

「變數」顧名思義，是一個隨時可能改變內容的容器名稱，就像家中的收藏箱可以放入各種不同的東西。



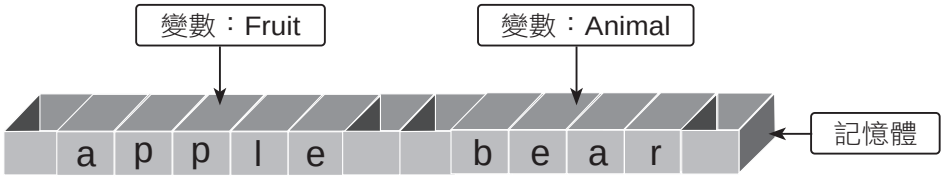
2.1.1 認識變數

應用程式執行時必須先儲存許多資料等待進一步處理，例如在英文單字教學應用程式中，許多英文單字必須先儲存在電腦內，等到要使用時再將其取出。那麼電腦將這些資料儲存在哪裡呢？事實上，電腦是將資料儲存於「記憶體」中，等到需要使用特定資料時，就到記憶體中將該資料取出。



▲ 資料儲存於記憶體

當資料儲存於記憶體時，電腦會記住該記憶體的位置，以便要使用時才可以取出。但電腦的地址是一個複雜且隨機的數字，例如「65438790」，程式設計者怎麼可能會記得此地址呢？更何況有很多地址要記憶。解決的方法是給予這些地址一個有意義的名稱，取代無意義的數字地址，就可輕鬆取得電腦中的資料了！這些取代數字地址的名稱就是「變數」。



▲ 以變數取代記憶體地址

2.1.2 建立變數

當建立一個變數時，應用程式就會配置一塊記憶體給此變數使用，並以變數名稱做為辨識此塊記憶體的標誌，設計者就可在程式中將各種值存入該變數中。

新增變數

Python 變數不需宣告就可以使用，語法為：

```
變數名稱 = 變數值
```

例如建立變數 score 的值為 80：

```
score = 80
```

使用變數時不必指定資料型態，Python 會根據變數值設定資料型態。例如上述 score 變數，系統會設定其資料型態為整數 (int)。又如：

```
fruit = "香蕉"  #fruit 的資料型態為字串
```

如果多個變數具有相同變數值，可以一起指定變數值，例如變數 a、b、c 的值皆為 20，其宣告方式為：

```
a = b = c = 20
```

也可以在同一列中指定多個變數，「變數」之間以「,」分隔，「值」之間也以「,」分隔。例如變數 age 的值為 18，name 的值為「林大山」：

```
age, name =  18, "林大山"
```

刪除變數

如果變數不再使用，可以將變數刪除以節省記憶體。刪除變數的語法為：

```
del 變數名稱
```

例如刪除變數 score：

```
del score
```

2.1.3 變數命名規則

為變數命名必須遵守一定的規則，否則在程式執行時會產生錯誤。Python 變數的命名規則為：

- 變數名稱只能由大小寫英文字母、數字、_、中文組成變數名稱。
- 變數名稱的第一個字母不能是數字，必須是大小寫字母、_ 及中文。
- 英文字母大小寫視為不同變數名稱。
- 變數名稱不能與 Python 內建的保留字相同。Python 常見的保留字有 ：

acos	and	array	asin	assert	atan
break	class	close	continue	cos	Data
def	del	e	elif	else	except
exec	exp	fabs	float	finally	floor
for	from	global	if	import	in
input	int	is	lambda	log	log10
not	open	or	pass	pi	print
raise	range	return	sin	sqrt	tan
try	type	while	write	zeros	

雖然 Python 3.x 的變數名稱支援中文，但建議最好不要使用中文做為變數命名，不但在撰寫程式時輸入麻煩，而且會降低程式的可攜性。

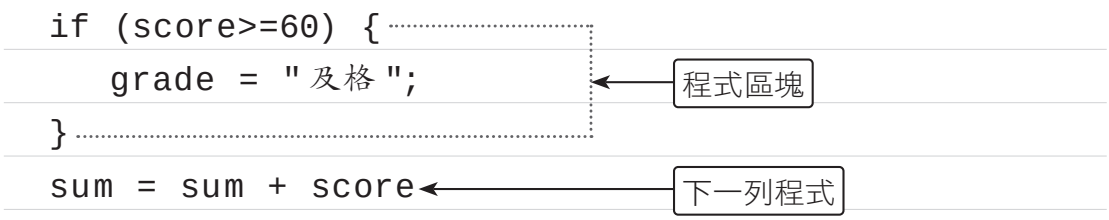
下表是一些錯誤變數名稱的範例實作：

屬性	說明
7eleven	第一個字元不能是數字
George&Mary	不能包含特殊字元「&」
George Mary	不能包含空白字元
if	Python 的保留字

3.1 Python 程式碼縮排

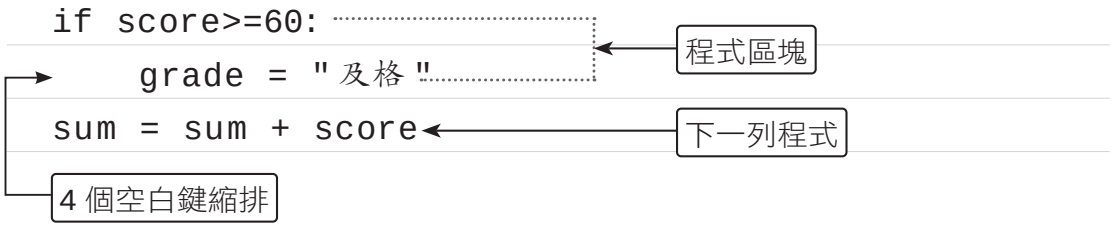
程式語言以縮排方式表示是一組相同的程式區塊。

大部分語言如 C、Java 等，都是以一對大括號「{}」來表示程式區塊，例如：



3.1.1 Python 程式碼縮排格式

Python 語言以冒號「:」及縮排來表示程式區塊，縮排建議使用 4 個空白鍵，例如：



在 Python 中建議的縮排方式是用 4 個空白鍵，但許多人卻習慣使用 Tab 鍵，在不同的編輯器讀取時可能就會產生不一樣的效果。

3.1.2 絕對不要混用 Tab 鍵和空白鍵

其實只要以相同的 Tab 鍵或相同字元的空白鍵整齊排列，即可達到同一程式區塊程式碼縮排的效果，但同一個程式區塊中絕對不要混用 Tab 鍵和空白鍵，官方建議以 4 個空白鍵做為縮排。

混用 Tab 鍵和空白鍵來縮排的程式碼，應該轉成只用空白鍵。在呼叫 Python 直譯器時加上「-t」選項，它會對混用 Tab 鍵和空白鍵的程式發出警告。若使用「-tt」選項，則會發出錯誤。

如果您使用的是 Spyder 或 Jupyter NoteBook 編輯器，可以按 4 個空白鍵，即使用 Tab 鍵也會自動轉換為 4 個空白鍵，避免這個問題。

3.2 判斷式

在日常生活中，我們經常會遇到一些需要做決策的情況，然後再依決策結果進行不同的事件，例如：暑假到了，如果所有學科都及格的話，媽媽就提供經費讓自己與朋友出國旅遊；如果有某些科目當掉，暑假就要到校重修了！程式設計也一樣，常會依不同情況進行不同處理方式，這就是「判斷式」。

3.2.1 程式流程控制

程式的執行方式有循序式及跳躍式兩種，循序式是程式碼由上往下依序一列一列的執行，到目前為止的範例都是這種模式。程式設計也和日常生活雷同，常會遇到一些需要做決策的情況，再依決策結果執行不同的程式碼，這種方式就是跳躍式執行。

Python 流程控制命令分為兩大類：

- **判斷式**：根據關係運算或邏輯運算的條件式來判斷程式執行的流程，若條件式結果為 True，就執行跳躍。判斷式命令只有一個：

```
if…elif…else
```

- **迴圈**：根據關係運算或邏輯運算條件式的結果為 True 或 False 來判斷，以決定是否重複執行指定的程式。迴圈指令包括下列兩種：（迴圈將在第 4 章詳細說明）

```
for
while
```

3.2.2 單向判斷式（if…）

「if…」為單向判斷式，是 if 指令中最簡單的型態，語法為：

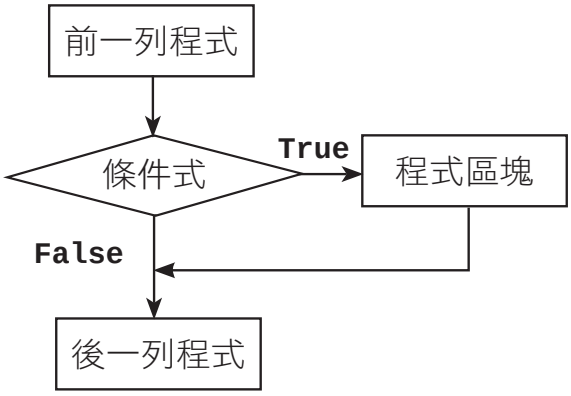
```
if 條件式：
    程式區塊
```

「條件式」允許加上括號，即「if (條件式):」。當條件式為 True 時，就會執行程式區塊的敘述；當條件式為 False 時，則不會執行程式區塊的敘述。

條件式可以是關係運算式，例如：「x>2」；也可以是邏輯運算式，例如：「x>2 or x<5」，如果程式區塊只有一列程式碼，也可以將兩列合併為一列，直接寫成：

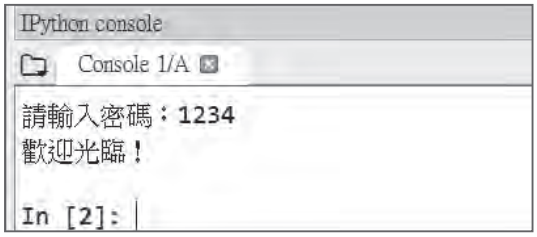
```
if 條件式 : 程式碼
```

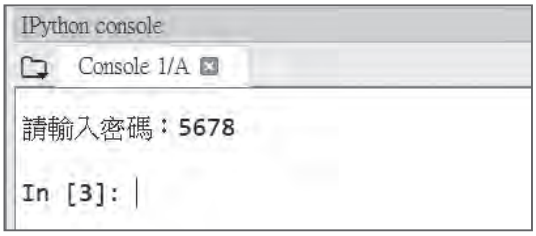
以下是單向判斷式流程控制的流程圖：



範例實作：密碼輸入判斷

小杰設計了一個通關密碼的程式，訪客必須輸入正確密碼才能登入，如果輸入的密碼正確（1234），會顯示「歡迎光臨！」；如果輸入的密碼錯誤，則不會顯示任何訊息。(<password1.py>)





程式碼：ch03\password1.py

```
1 pw = input("請輸入密碼:")
2 if pw=="1234":
3     print("歡迎光臨!")
```

程式說明

- 2-3　預設的密碼為「1234」，若輸入的密碼正確，就執行第 3 列程式列印「歡迎光臨！」訊息；若輸入的密碼錯誤就結束程式。
- 3　if 條件成立的程式區塊，必須以 Tab 鍵或空白鍵向右縮排，本例是以 4 個空白鍵做縮排。

因為此處 if 程式區塊的程式碼只有一列，所以第 2-3 列可改寫為：

```
if pw=="1234" : print("歡迎光臨！")
```

3.2.3 雙向判斷式（if…else）

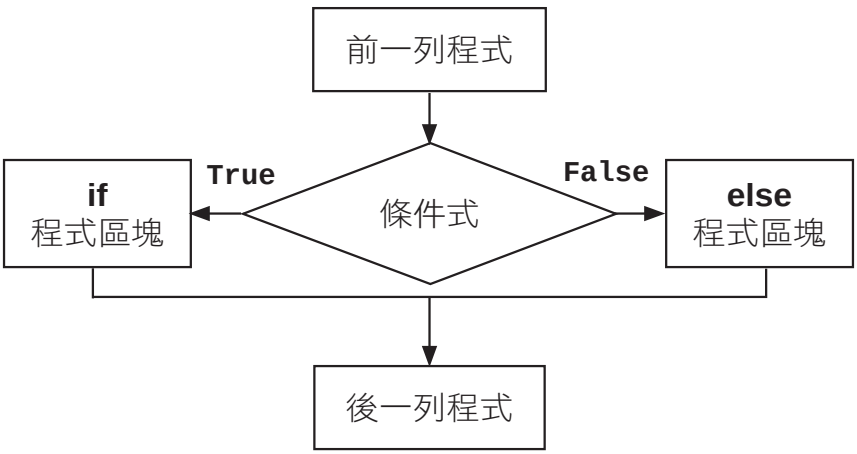
感覺上「if」語法並不完整，因為如果條件式成立就執行程式區塊內的內容，如果條件式不成立也應該做某些事來告知使用者。例如密碼驗證時，若密碼錯誤應顯示訊息告知使用者，此時就可使用「if…else…」雙向判斷式。

「if…else…」為雙向判斷式，語法為：

```
if 條件式：
    程式區塊一
else:
    程式區塊二
```

當條件式為 True 時，會執行 if 後的程式區塊一；當條件式為 False 時，會執行 else 後的程式區塊二，程式區塊中可以是一列或多列程式碼，如果程式區塊中的程式碼只有一列，可以合併為一列。

以下是雙向判斷式流程控制的流程圖：



範例實作：進階密碼判斷

小杰程式設計的功力進步許多，現在他改進了通關密碼程式，如果訪客輸入的密碼正確（1234），會顯示「歡迎光臨！」；如果訪客輸入的密碼錯誤，則會顯示「密碼錯誤！」。(<password2.py>)

```
IPython console
Console 1/A
請輸入密碼：1234
歡迎光臨！

In [18]:
```

```
IPython console
Console 1/A
請輸入密碼：5678
密碼錯誤！

In [19]:
```

程式碼：ch03\password2.py

```
1  pw = input("請輸入密碼:")
2  if pw=="1234":
3      print("歡迎光臨！")
4  else:
5      print("密碼錯誤！")
```

程式說明

- 2-3　若輸入的密碼正確，就執行第 3 列程式，顯示歡迎訊息。
- 4-5　若輸入的密碼錯誤，就執行第 5 列程式，顯示密碼錯誤訊息。注意第 4 列要由開頭處輸入「else:」。

延伸練習

資訊小楷模阿梅幫老師設計一個程式，讓老師輸入學生的成績，若學生成績大於等於 60 分，顯示「讚，成績及格!」，否則顯示「成績不及格，加油喔!」。(<score.py>)

```
IPython console
Console 1/A
請輸入成績：90
讚，成績及格!

In [13]:
```

```
IPython console
Console 1/A
請輸入成績：58
成績不及格，加油喔!

In [14]:
```

5.1 串列的使用

程式中的資料通常是以變數來儲存，如果有大量資料需要儲存時，就必須宣告龐大數量的變數。例如：某學校有 500 位學生，每人有 10 科成績，就必須有 5000 個變數才能完全存放這些成績，程式設計者要如何宣告 5000 個變數呢？在程式中又如何明確的存取某一特定的變數呢？

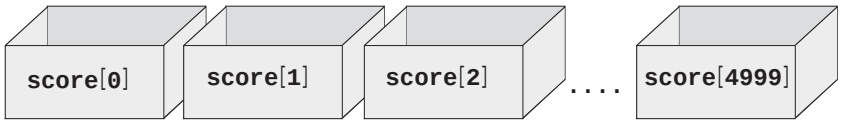
```
score1=90
score2=80
...
score5000=100
```

宣告 5000 個變數

5.1.1 何謂串列 (List)

串列 (又稱為「清單」或「列表」)，與其他語言的「陣列 (Array)」相同，其功能與變數相類似，是提供儲存資料的記憶體空間。每一個串列擁有一個名稱，做為識別該串列的標誌；串列中每一個資料稱為「元素」，每一個串列元素相當於一個變數，如此就可輕易儲存大量的資料儲存空間。

可以把串列想成是有許多相同名稱的箱子，連續排列在一起，這些箱子可以儲存資料，而每個箱子依序給予索引編號，如果要存取箱子中的資料，只要指定編號即可存取對應箱子內的資料。



▲ 串列元素配置

5.1.2 串列宣告

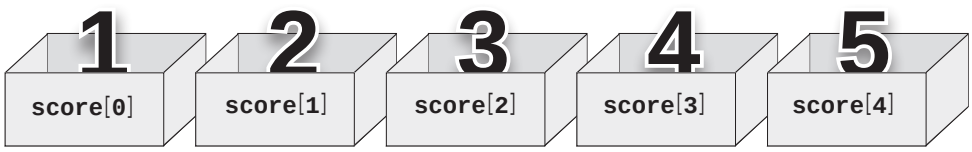
串列的功能與變數類似，其使用方法也類似。首先，在使用之前需先宣告，宣告時要指定識別字做為串列名稱，以便未來可用此名稱來存取這個串列。

一維串列宣告

一維串列的宣告方式是將元素置於中括號 ([]) 中，每個元素之間以逗號分隔，語法為：

```
串列名稱 = [ 元素 1, 元素 2,……]
```

例如：宣告 score 串列，其元素內容為 [1, 2, 3, 4, 5]。



以 print(score) 可以顯示 score 串列的內容。

```
print(score)  # [1, 2, 3, 4, 5]
```

串列中各個元素資料型態可以相同，也可以不同，例如：

```
list1 = [1, 2, 3, 4, 5]              # 元素皆為整數
list2 = ["香蕉", "蘋果", "橘子"] # 元素皆為字串
list3 = [1, "香蕉", True]            # 包含不同資料型態元素
```

空串列

如果宣告時省略串列中的元素，該串列即為一個空的串列。例如：

```
list4=[]
```

多維串列宣告

串列的元素可以是另一個串列，這樣就形成多維串列。多維串列元素的存取是使用多個中括號組合，例如下面是二維串列的範例，其串列元素是帳號、密碼組成的串列：

```
list5=[["joe","1234"],["mary","abcd"], ["david","5678"]]
print(list5[1])     #["mary","abcd"]
print(list5[1][1])  #abcd
```

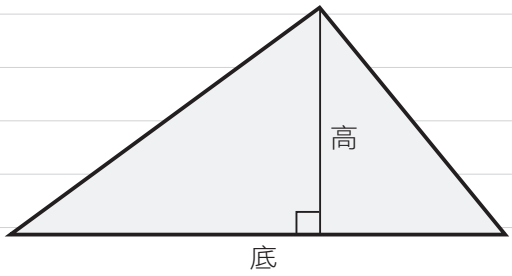
8.1 認識演算法

做任何事情都有一定的步驟。簡單的說，演算法 (Algorithms) 就是為了解決一個問題而採取的方法和步驟，演算法通常會以虛擬碼來表示，再以熟悉的語言來實現，本書當然是以 Python 來完成。 如果是較複雜的問題，有時還會繪製流程圖加以輔助。

以虛擬碼表示

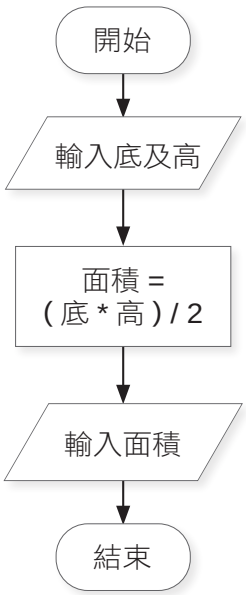
例如我們想要撰寫「求三角形面積」這個問題的演算法並繪製流程圖。以下將這個任務以虛擬碼表示：

```
程式開始
    輸入 底
    輸入 高
    面積 = ( 底 X 高 ) / 2
    輸出 面積
程式結束
```



演算法和流程圖

1. 程式開始
2. 輸入三角形的底和高
3. 計算三角形的面積 = (底 × 高)÷2
4. 輸出計算後所得三角形的面積
5. 程式結束



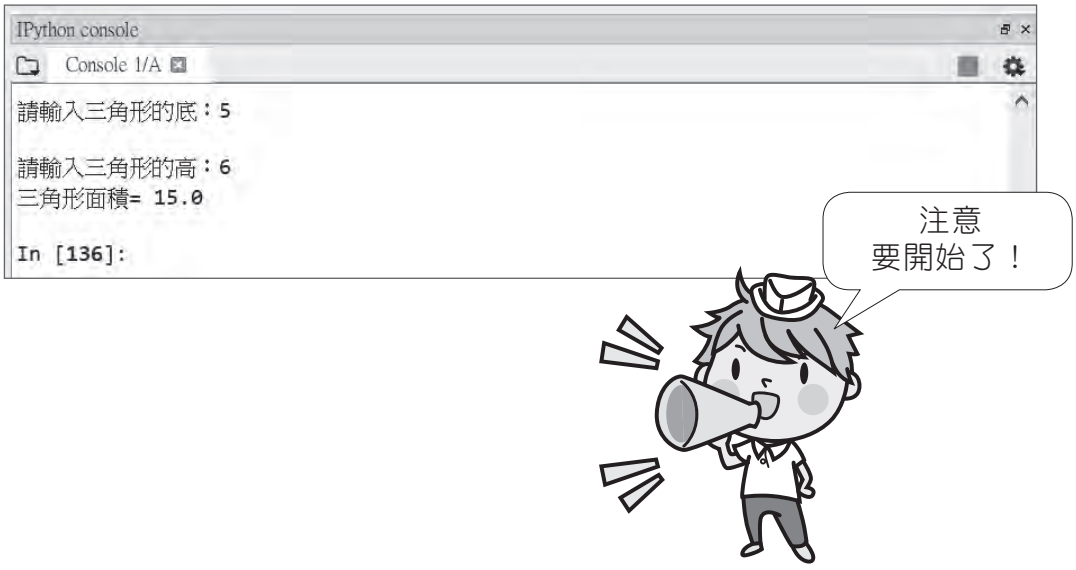
▲ 計算三角形面積流程圖

撰寫程式碼

程式碼：ch08\triangleArea.py

```
bottom = int(input("請輸入三角形的底:"))
height = int(input("請輸入三角形的高:"))
area=(bottom*height)/2
print("三角形面積 =",area)
```

觀察執行結果



8.3 搜尋

資料搜尋是串列另一個最常使用的功能。搜尋資料常用的搜尋方法有循序搜尋和二分搜尋，循序搜尋是依序逐一搜尋；使用二分搜尋則可以提昇速度，但程式較複雜，且搜尋前資料必須先進行排序。

8.3.1 循序搜尋

循序搜尋是從串列中第一個串列元素開始，依序逐一搜尋，方法很簡單，但是缺點是較沒有效率。

循序搜尋的演算法

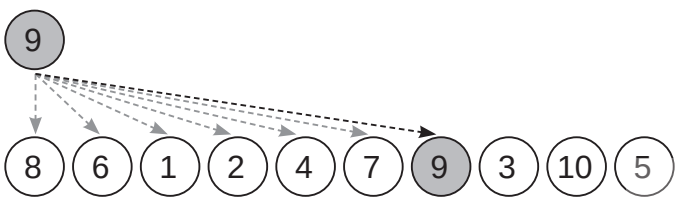
循序搜尋資料可以不用排序，演算法的運作如下：

1. 從串列中第一個串列元素開始搜尋。
2. 如果找到目標，結束搜尋。
3. 如果沒有找到目標，繼續搜尋下一個串列元素，直到串列元素全部搜尋完為止。

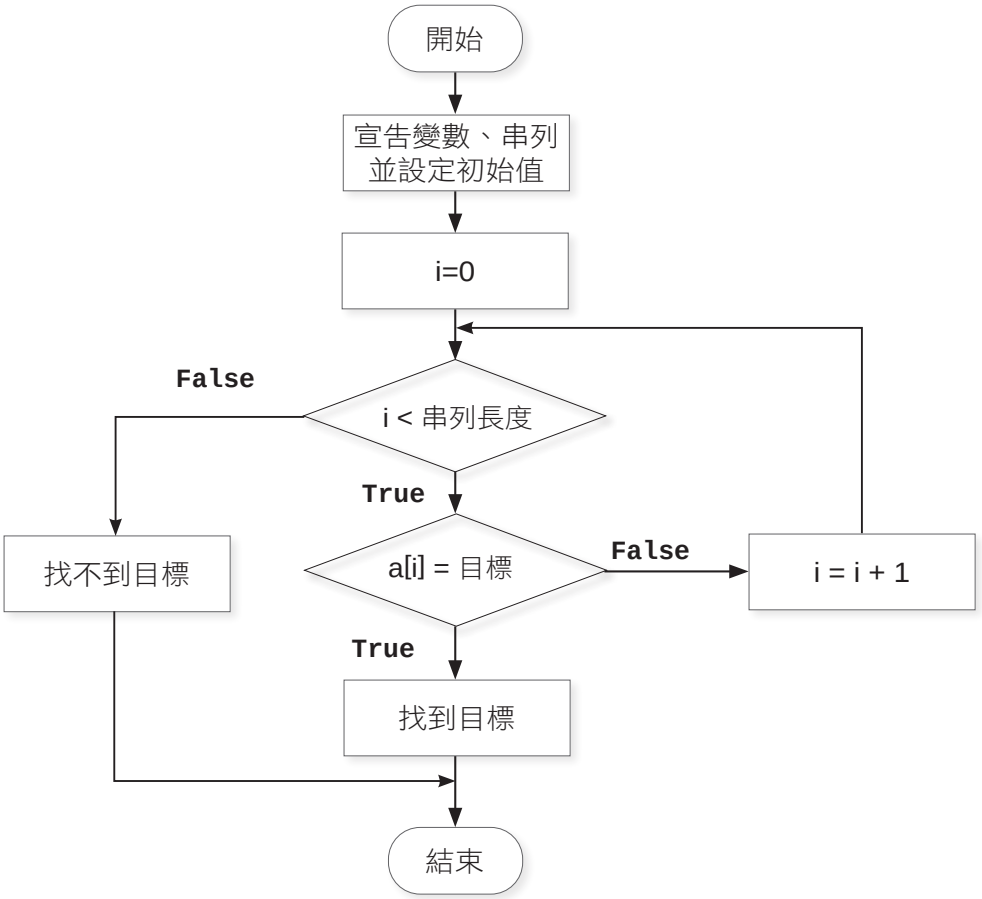
循序搜尋是從第一個串列元素開始，依序逐一搜尋，如果串列元素有 n 個，循序搜尋最快一次就可以搜尋到， 最慢則需 n 次才能搜尋到。程式中刻意加入「比對次數」訊息，目的是要讓使用者了解實際的搜尋次數（實際應用程式中可將此部分移除）：觀察執行結果的第二項，當查詢資料不存在時，循序搜尋須從頭到尾搜尋一次，本範例中有 8 筆資料，所以顯示比對 8 次。實際應用時資料動輒數十萬筆，一次搜尋就要比對數十萬次，非常沒有效率，將造成系統很重的負擔，且搜尋時間很長！

循序搜尋的運作實例

例如有一序列的值是 8, 6, 1, 2, 4, 7, 9, 3, 10, 5，若想要在其中找出 9 這個值所在的位置，循序搜尋的過程如下，要找到第 7 次才能找到：



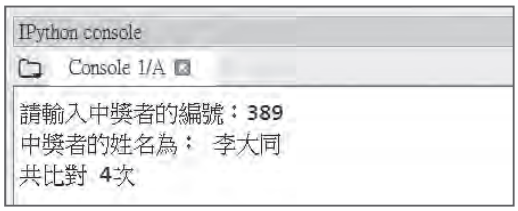
循序搜尋的流程圖

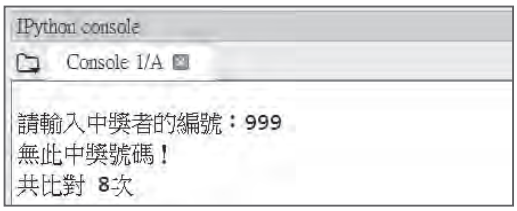


▲ 循序搜尋流程圖

範例實作：中獎者姓名 (循序搜尋)

百貨公司舉辦週年抽獎活動，將顧客的抽獎編號及姓名分別儲存於串列中，使用者輸入編號，程式會搜尋出該編號的姓名並顯示；若查詢不到也會顯示無此編號的訊息。(<sequential.py>)





程式碼：ch08\sequential.py

```
1   num=[256,731,943,389,142,645,829,945]
2   name=["林小虎","王中森","邵木淼","李大同","陳子孔",
          "鄭美麗","曾溫柔","錢來多"]
3   no = int(input("請輸入中獎者的編號:"))
4
5   IsFound=False
6   for i in range(len(name)):   #逐一比對搜尋
7       if (num[i]==no):         #號碼相符
8           IsFound=True         #設旗標為 True
9           break                #結束比對
10
11  if (IsFound==True):
12      print("中獎者的姓名為:",name[i])
13  else:
14      print("無此中獎號碼!")
15  print("共比對 %d 次 " %(i+1))
```

程式說明

- 1-2　分別建立編號及姓名的對應串列。
- 3　輸入中獎者的編號 no。
- 5　宣告 IsFound 並預設為 False，如果在 6-9 列的搜尋有找到查詢資料，就設 IsFound=True。在程式設計中，這個觀念稱為旗標，也就是如果有找到就將設旗標 IsFound=True，否則 IsFound 的值為 False，最後判斷旗標 IsFound 就可得知資料是否有找到。
- 6-9　逐一比對資料是否相符。
- 9　如果找到查詢資料就離開 for 迴圈。
- 11-14　根據判斷旗標 IsFound 以得知資料是否有找到來顯示訊息。
- 15　因為 i 的值是由 0 開始，所以比對次數是將其加 1 才是真正的比對次數。