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：

```
if (score>=60) {
   grade = "及格";
}
sum = sum + score
```

程式區塊

下一列程式

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：

```
if score>=60:
    grade = "及格"
sum = sum + score
```

程式區塊

下一列程式

4 個空白鍵縮排

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：

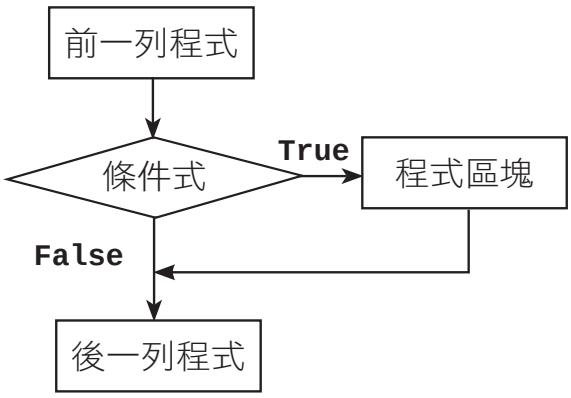
```
if 條件式：
    程式區塊
```

「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：

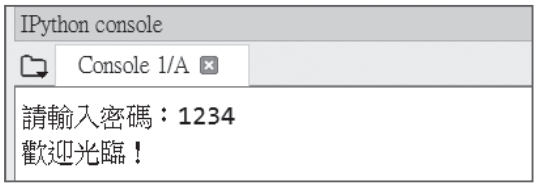
```
if 條件式 : 程式碼
```

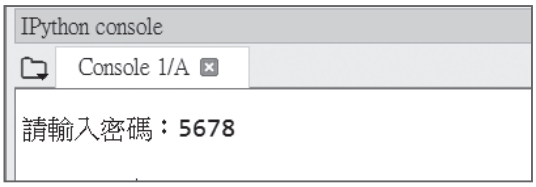
以下是單向判斷式流程控制的流程圖：



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）

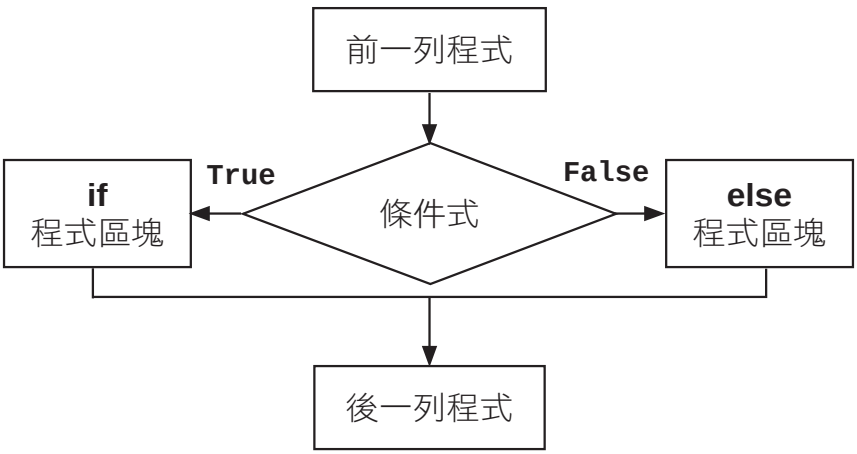
感覺上「if」語法並不完整，因為如果條件式成立就執行程式區塊內的內容，如果條件式不成立也應該做某些事來告知使用者。例如密碼驗證時，若密碼錯誤應顯示訊息告知使用者，此時就可使用「if…else…」雙向判斷式。

「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
```

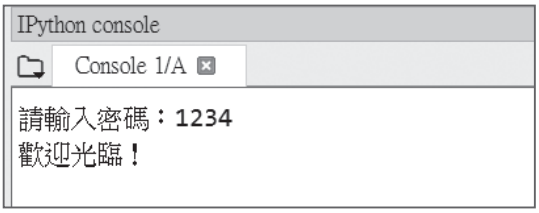
當條件式為 True 時，會執行 if 後的程式區塊一；當條件式為 False 時，會執行 else 後的程式區塊二，程式區塊中可以是一列或多列程式碼，如果程式區塊中的程式碼只有一列，可以合併為一列。

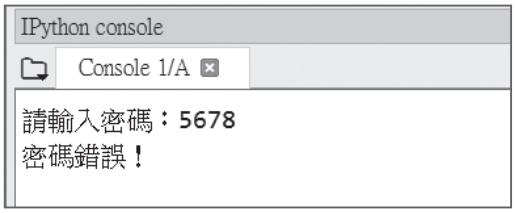
以下是雙向判斷式流程控制的流程圖：



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>)





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>)

7.4 亂數模組：random

Python 最為人稱道的優勢就是擁有許多模組 (module)，使得功能可以無限擴充。Python 的亂數模組：random 功能非常強大，不但可以產生整數或浮點數的亂數，還可以一次取得多個亂數，甚至可以為串列洗牌。

7.4.1 import 模組

模組只要使用「import」命令就可匯入，import 命令的語法為：

```
import 模組名稱
```

例如亂數模組的模組名稱為 random，匯入亂數模組的程式為：

```
import random
```

通常模組中有許多函式供設計者使用，使用這些函式的語法為：

```
模組名稱 . 函式名稱 ( 參數 )
```

例如 random 模組有 randint、random、choice 等函式，使用 randint 函式的程式語法為：

```
random.randint( 參數 )
```

如果確認只使用模組內的某個函式，可以在宣告時就設定好，語法為：

```
from 模組名稱 import 函式名稱
函式名稱 ( 參數 )
```

例如只要使用 random 模組中的 randint 函式，設定語法為：

```
from random import randint
randint( 參數 )
```

如果不想使用模組函式都要輸入模組名稱，也可以利用「*」萬用字元將其下的函數全部載入，語法為：

```
from 模組名稱 import *
```

以此種語法匯入模組後，使用模組函式就不必輸入模組名稱，直接使用函式即可，例如：

```
from random import *
randint( 參數 )
```

此種方法雖然方便，卻隱藏著極大風險：每一個模組擁有眾多函式，若兩個模組具有相同名稱的函式，由於未輸入模組名稱，使用函式時可能造成錯誤。為兼顧便利性及安全性，可為模組名稱另取一個簡短的別名，語法為：

```
import 模組名稱 as 模組別名
模組別名 . 函式名稱 ( 參數 )
```

這樣一來，使用函式時就用「模組別名 . 函式名稱」呼叫，既可避免輸入較長的模組名稱，又可避免不同模組中相同函式名稱問題，例如：

```
import random as r
r.randint( 參數 )
```

除了模組可以設定別名外，模組中的函數也能設定別名，語法為：

```
from 模組名稱 import 函式名稱 as 函式別名
函式別名 ( 參數 )
```

這樣一來，使用函式時就可以直接用「別名 . 函式名稱」呼叫，例如：

```
from random import randint as rt
rt( 參數 )
```

7.4.2 亂數模組函式整理

在下表中「r」為亂數模組的別名，str1="abcdefg", list1=["ab", "cd", "ef"]，常用的亂數模組函式有：

函式	功能	範例	範例結果
randint(n1,n2)	由 n1 到 n2 之間隨機取得一個整數	r.randint(1,10)	7
random()	由 0 到 1 之間隨機取得一個浮點數	r.random()	0.893398…
randrange(n1,n2,n3)	由 n1 到 n2 之間每隔 n3 的數隨機取得一個整數	r.randrange (0,11,2)	8 (偶數)
uniform(f1,f2)	由 f1 到 f2 之間隨機取得一個浮點數	r.uniform(1,10)	6.351865…
choice(字串)	由字串中隨機取得一個字元	r.choice(str1)	b
sample(字串 ,n)	由字串中隨機取得 n 個字元	r.sample(str1,3)	['c', 'a', 'd']
shuffle(串列)	為串列洗牌	r.shuffle(list1)	['ef', 'ab', 'cd']

7.4.3 產生整數或浮點數的亂數函式

randint 函式

randint 函式的功能是由指定範圍產生一個整數亂數，語法為：

```
random.randint( 起始值 , 終止值 )
```

執行後會產生一個在起始值 (含) 和終止值 (含) 之間的整數亂數，注意產生的亂數可能是起始值或終止值，例如：

```
import random
for i in range(5):   # 執行 5 次，產生 5 個整數亂數
    print(random.randint(1,10), end=",")   #9,8,1,10,4,
```

上例中，1 與 10 都是可能產生的亂數。

randrange 函式

randrange 函式的功能與 randint 雷同，也是產生一個整數亂數，只是其多了一個遞增值，語法為：

```
random.randrange( 起始值 , 終止值 [, 遞增值 ])
```

執行後會產生一個在起始值 (含) 和終止值 (不含) 之間，且每次增加遞增值的整數亂數，遞增值非必填，預設值為 1。特別注意產生的亂數可能是起始值，但不包含終止值，例如：

```
import random
for i in range(5):   # 執行 5 次，產生 5 個整數亂數
    print(random.randrange(0,12,2), end=",")
    #8,0,10,6,6,
```

由於從 0 開始到 12，但不包含 12，每次遞增 2，所以產生的亂數是「0、2、4、6、8、10」六個數其中之一。

random 函式

random 函式的功能是產生一個 0 到 1 之間的浮點數亂數，語法為：

```
random.random()
```

例如：

```
import random
print(random.random())   #0.5236730771512399
```

uniform 函式

uniform 函式的功能是產生一個指定範圍的浮點數亂數，語法為：

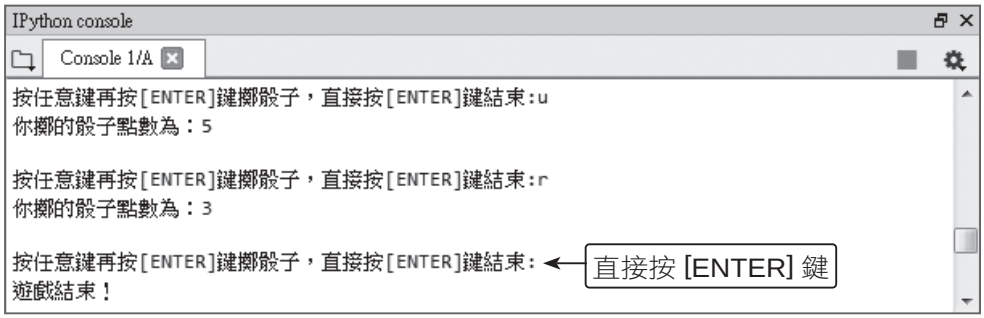
```
random.uniform( 起始值 , 終止值 )
```

執行後會產生一個在起始值和終止值之間的整數亂數，例如：

```
import random
print(random.uniform(3,10))   #6.063374013178429
```

範例實作：擲骰子遊戲

阿寶想玩擲骰子遊戲，但手邊沒有骰子，設計程式讓阿寶按任意鍵再按 **Enter** 鍵擲骰子，會顯示 1 到 6 之間的整數亂數代表骰子點數，直接按 **Enter** 鍵會結束遊戲。(<randint.py>)



程式碼：ch07\randint.py

```
 1 import random
 2
 3 while True:
 4     inkey = input("按任意鍵再按 [ENTER] 鍵擲骰子，直接
                      按 [ENTER] 鍵結束 :")
 5     if len(inkey) > 0:
 6         num = random.randint(1,6)
 7         print("你擲的骰子點數為:" + str(num))
 8     else:
 9         print("遊戲結束！")
10         break
```

程式說明

- 1　匯入亂數模組。
- 3-10　以無窮迴圈讓使用者擲骰子。
- 5-7　使用者按任意鍵再按 **Enter** 鍵就取得 1 到 6 之間的亂數顯示。
- 8-10　使用者直接按 **Enter** 鍵就顯示「結束遊戲」訊息並跳出迴圈。

Appendix

A

ITS 資訊科技專家
國際認證模擬試題

IT Specialist Certification - Python

一、使用資料類型和運算子執行操作

1. 你編寫了以下的程式碼：

對應章節 02

```
list_1 = [1, 2, 3]
list_2 = [4, 5, 6]
list_3 = list_1 + list_2
list_4 = list_3 * 2
print(list_4)
```

執行程式碼的輸出值是？

( )A. [[1, 2, 3], [4, 5, 6], [1, 2, 3], [4, 5, 6]]

( )B. [4, 10, 18]

( )C. [1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6]

( )D. [[1, 2, 3, 4, 5, 6], [1, 2, 3, 4, 5, 6]]

2. 你是運動 App 的程式設計師。你必須製作一個函式為跑者計算步速，所謂步速就是每公里所花的時間。輸出結果必須盡可能精準。要如何完成程式碼？請在回答區中選擇適當的程式碼片段。其中距離須轉換為浮點數，時間的輸入值都要轉換為整數。

對應章節 02

```
# 步速計算器
distance =___(1)___(input("Enter the distance traveled in meters"))
distance_kms = distance / 1000 # convert to kilometers
time_minute =___(2)___(input("Enter the time elapsed in minutes"))
time_sec =____(3)____(input("Enter the time elapsed in seconds"))
time = time_minute * 60 + time_sec
pace = time / distance_kms
print("The average velocity is" ,str((pace//60))+ ": str((pace%60)))
```

以上空格分別要填入的函式名為：

( )(1) A. int B. string C. float

( )(2) A. int B. string C. float

( )(3) A. int B. string C. float

3. 高年級的老師要製作一份報表來顯示這次考試班上所有學生的平均分數。報表必須去除平均分數的小數部分。每個正確的答案都提供了一個完整的解決方案。你應該使用哪兩個程式碼片段？

對應章節 02

( )A. 平均分數 = float(全班總分 // 全班人數)

( )B. 平均分數 = int(全班總分 / 全班人數)

( )C. 平均分數 = float(全班總分 ** 全班人數)

( )D. 平均分數 = 全班總分 // 全班人數

4. 你正在編寫一個 Python 程式用來記錄客戶資料並將其儲存在資料庫中。這個程式處理各種各樣的資料。以下的變數宣告後它們的資料類別是？請將適合的程式碼片段⊙連到正確的回答區⊙。

對應章節 02

程式碼片段		回答區
int	⊙	⊙ age = 12
bool	⊙	⊙ minor = False
str	⊙	⊙ name = "David"
float	⊙	⊙ weight = 64.5
		⊙ zip = "545"

5. 你編寫了以下的程式碼：

對應章節 02

```
a = 24
b = 7
ans = (a % b * 100) // 2.0 ** 3.0 - b
print(ans)
```

執行程式碼的輸出值是？

( )A. 30

( )B. 30.5

( )C. 457

( )D. 語法錯誤