8 布林判斷式

挑戰題目: 博客來購物金額

下面是網路購書博客來商店的運費處理方式，請寫一個程式計算消費者總共要該付多少錢？

國內處理費

付款方式	單筆訂單實際消費金額	處理費
7-ELEVEN取貨付款 7-ELEVEN取貨(線上付款、ATM轉帳)	未滿350元	20元
線上付款、 客樂得、 ATM轉帳	未滿490元	65元
	490~999元	50元
	滿1000元以上	免收處理費

```
## 博客來網路購物付款總額
buying = int(input("請輸入顧客單筆訂單實際消費金額： "))
method = input("請輸入是否由7-11取貨（是或否）： ")

if method == "是":
    if buying < 350:
        total = buying + 20
    else:
        total = buying
else:
    if buying < 490:
        total = buying + 65
    elif buying < 999:
        total = buying + 50
    else:
        total = buying
print(f"消費者含運費之總付款金額為{total}")
```

從這個例子中，同學發現有了邏輯判斷結果後，就可以做選擇了。

觀念 1. 我們會根據一件事情結果的真(True)或假(False)做決定，進而執行不同的動作。

- 根據條件的真假做出不同的決定。
  - 如果中樂透，就捐錢給世新大學。
  - 如果我考100分，就去吃大餐；如果不及格，就去跳舞。

觀念 2. 電腦會根據「布林條件式」結果的真假執行不同的程式碼，稱決策結構(條件選擇/分支)。

- 「條件敘述式」(if conditional statement)
  - 如果布林條件的判斷結果為True(真)，則執行if程式碼區塊；

- 如果布林條件的判斷結果為True(真)，則執行程式碼區塊，
  - 如果布林條件的判斷結果為False(假)，則執行其他程式碼區塊。

- 常見的有單向選擇、雙向選擇、多向選擇與巢狀選擇四種。

8-1. 電腦懂得真與假

8-1-1. 布林條件式

我們使用布林值，並不像數字或文字資料一樣，是直接輸入電腦真值或假值，而是交由電腦來判斷敘述的真假。

電腦可以決定一個敘述是真(True)還是假(False)。

- 如「24是一個偶數」這個敘述是True。
- 又如「25是一個偶數」這個敘述是False。

布林條件式

- 得到True或False結果的任何敘述式，稱為**布林條件式**。
- 布林值是布林條件式的判斷結果。
- 在判斷條件真假、或者進行資料篩選的時候，我們仰賴布林值。

=> 在布林條件式中，用來做真假值判斷的運算子，包括關係運算子、邏輯運算子，以及成員與身份運算子三種。

8-1-2 關係運算子

在 Python 語法中，我們使用**關係運算子**來判斷條件的真假。

- 常見的關係運算子一共有六個：`==`、`!=`、`>`、`>=`、`<`、`<=`。
- 由關係運算子所構成的運算式稱為**條件運算式**。
- 使用條件運算式來比較兩個運算元之間的關係時，有 True（真）或是 False（假）兩種結果。

表 4-3.1　邏輯運算子的相關運算

運算子		功能	例子	說明
1	==	等於	x==y	若 x 等於 y，則結果為真
2	!=	不等於	x!=y	若 x 不等於 y，則結果為真
3	<	小於	x<y	若 x 小於 y，則結果為真
4	>	大於	x>y	若 x 大於 y，則結果為真
5	<=	小於等於	x<=y	若 x 小於等於 y，則結果為真
6	>=	大於等於	x>=y	若 x 大於等於 y，則結果為真

範例1：判斷7和8之間的關係

```
1 print(7 == 8)
2 print(7 != 8)
```

```
print(7 > 8)
print(7 >= 8)
print(7 < 8)
print(7 <= 8)
```

```
False
True
False
False
True
True
```

【隨堂練習1】：請輸入身分證字號，並判斷尾數是否為奇數?

8-1-3 邏輯運算子

邏輯運算子將不同的判斷式結合，變成一個新的、複雜的決策條件，並用這個布林條件式的真假，做出決策。

1. 邏輯運算子有三種運算子，and（而且）、or（或）、not（相反）。

且：只有在「兩者」敘述式都True時才為True。

- 例:下雨帶傘出門
  - 「在下雨」這個條件式為True
  - （且）「我有傘」這個條件也是True
  - 那麼就可以「下雨帶傘出門」。

或：其中之一True時為True。

- 例:天冷或風大穿外套
  - 「天冷」這個條件式為True
  - （或）「風大」這個條件式是True
  - 二者只要一個條件成立，就可以「穿外套」。
  - 也就是說，如果「天冷」、如果「風大」，或如果「同時天冷又風大」，我們都會穿外套。

非真: not True為False。

- 例:天冷穿外套，天不冷則不穿外套。
  - 「天冷」這個條件是True，則「穿外套」。
  - 「天冷」這個條件是False（非真），則「不穿外套」。
  - 「天冷」是「天不冷」的非真，即相反情況。

2. 真值表: 使用邏輯運算子連結兩個布林值，此新條件式的判斷結果。

- 多個條件結果必須同時為True ，結果才會為 True 時，就使用**「and」**結合這些條件；

X and Y	Y=True	Y=False
X=True	True	False
X=False	False	False

```
## 且and:判斷 True 與 False 的交集
True and False
True and True
False and False
False and True

rain = True
unbrella = True
result = rain and unbrella
print(result)
```

```
True
```

- 只要其中之一條件為True，結果就會為True時，就使用「**or**」結合這些條件；

X or Y	Y=True	Y=False
X=True	True	True
X=False	True	False

```
## 或or ：判斷 True 與 False 的聯集
True or False
True or True
False or False
False or True
```

- 要相反的結果，就使用「not」，「**not**」必須置於該條件的前面。

	not X
X=True	False
X=False	True

```
## 非(not): 餘集的判斷
not True
not False
```

範例2: 布林：邏輯運算子

輸入數字，然後做邏輯運算。

```
x = int(input("Enter your number: ")) # 任意給個x
((x > 60) and (x < 80))
((x > 60) or (x < 80))
not(x > 60)
```

【隨堂練習2】：條件判斷式

- 去偽存真、淘砂取金，比較兩個數的大小。請輸入a和b的值，然後確認a是不是比b大。

- 處理10歲(含)以上，18歲(不含)以下的age年齡資料，條件式如何寫？

8-1-4 成員運算子

1. 成員運算子 in

[x in y] : 判斷x是否為y中的一個元素。

範例：文字是否在字串裡或文字堆裡

- 判斷H是否存在於Hello world 之中
- 判斷Hello是否存在於Hello world 之中

範例3：成員運算子

```
## 文字是否在字串裡或文字堆裡
print("H" in "Hello world")
print("Hello" in "Hello world")
```

```
True
True
```

2. 身份運算子 is (Optional)

[x is y] : 判斷變數在電腦記憶體的位置是否相同。

注意：身份運算子的判斷和關係運算子中的等號不同。

- 關係運算子==是以變數的值是否相等來判斷。
- 身份運算子is是以記憶體位置是否相同來判斷。
    - 先要用id()把變數的記憶體位置找出來
    - 再判斷位址是否相同

範例4：數字串列的判斷

```
## 數字串列
x = 1
y = [1,2,3] # (1,2,3) [1,2,3] {1,2,3}
print(x in y)
print(x not in y)

print(x is y)
print(x is not y)
```

```
True
False
False
True
```

8-2 程式設計的輔助工具

2. 程式設計的輔助工具

在用電腦解決問題的步驟二，程式設計的部分，除了配合Input-Process-Output電腦軟硬體的運作方式思考外，還可以透過撰寫「文字式的虛擬碼」(pseudocode)，或者繪製「視覺化的流程圖」(flow chart)，呈現程式設計的邏輯與步驟，幫助我們思考來解決問題。

2-1 虛擬碼

1. **虛擬碼**是將演算法執行過程的結構，用接近自然語言的形式（人話）描述出來。

- 用文字將程式設計的邏輯表達出來。
- 可以使用任何一種作者熟悉的文字來描述解決問題的方法，例如中文、英文。

範例5：請寫出計算直角三角形面積的虛擬碼

```
## 計算直角三角形面積
## Input Data:輸入底與高的值
x = 10  # 底
y = 5   # 高
## Process Data:帶公式：面積等於底乘以高除以二
area = x * y /2 # 公式
## Output Data: 印出面積的值。
print(area)
```

```
25.0
```

2. **用虛擬碼寫出程式的邏輯，可以讓思維條理化、並快速轉為程式語言。**

人的思維可能比想像中更沒有邏輯，而寫程式強調邏輯化的思考方式。因此，要先釐清腦中模糊的想法，讓腦中的思維調理化。通常，人們透過過去的經驗，「理所當然」地處理各式各樣的情況，而這些未被明確定義的表現形態，往往含有「模糊曖昧」之處。電腦無法理解這些「模糊曖昧」，所以要先去除電腦所無法處理的「模糊曖昧」，讓一無所知的電腦能夠處理的「理所當然」。

3. **寫程式≠寫程式碼**

在寫程式之前，先想好要怎麼做，整理自己的想法，再轉換成程式碼表達。

初學者寫程式

- 邊寫程式碼邊想怎麼解
- 試著套用自己以為學過的語法

會寫程式的人寫程式

- 先想解法，在腦中構思，通常會有答案，但答案的邏輯還不夠清楚。
- 用虛擬碼把想法的邏輯思路一個步驟一個步驟的弄清楚。
- 再把解法轉換成程式碼。
    - 一行程式碼只做一件事
    - 善用「循序、選擇、重複」的結構。

4. **虛擬碼是程式碼與想法之間的橋樑，把解法用抽象的方式表示。**

寫程式小技巧： 透過加註的方式寫虛擬碼，在程式中就看到並看懂程式設計的邏輯。

寫程式小技巧： 學會像電腦一樣思考。寫程式之前，先學會「看程式」，所謂的看程式碼 = 理解程式碼如何運作。即你的腦中要有自己或別人所寫程式的虛擬碼！

8-2-2 流程圖

1. 「流程圖」是使用各種不同的圖形、線條及箭頭來描述問題的解決步驟，處理的順序，讓人了解整個作業流程。

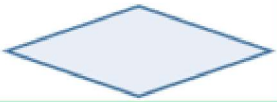
即使不懂程式語言，只用簡單的符號，也可以寫出程式執行的邏輯。因此，將流程圖當作寫程式前的準備工作，對程式設計非常有用。

2. 下面是美國國家標準學會(ANSI)於1970年公佈的流程圖符號，常用的圖形符號有五種：

- 橢圓=開始或結束
- 平行四邊形=資料輸入輸出 (Input- Output)
- 長方形 = 程序處理(Process)
- 箭頭 = 運算流程
- 菱型 = 流程控制 (選擇與重複結構)

1．流程图的基本图形及其功能

Python 语言中，在编写程序之前，都要设计算法，一般算法都可以使用流程图来表示。因为用流程图描述算法，形象、直观，更容易理解，并且不依赖于具体的计算机程序设计语言。对于一些复杂的算法，设计人员往往先用流程图描述算法，以保证程序结构的正确性，从而降低程序编写的难度。流程图基本图形是有一定的规定的，具体情况如下表如示：

图形	名称	功能
	开始/结束符	表示算法的开始或结束。一个算法只能有一个开始处，但可以有多个结束处
	输入/输出框	表示数据的输入或计算结果的输出
	判断框	表示分支情况，通常用上方的顶点表示入口，选择其余顶点中的 2 个表示出口
	流程线	指出流程控制方向，即动作的次序
	处理框	框中指出要处理的内容，该框有一个入口和一个出口

3. 能夠以流程圖的形式寫出程式處理的邏輯，可以用讓腦中的思維條理化、更清楚地表示問題的解決方法。

- 一個繪製良好的流程圖，必須符合下面原則：
  - 流程圖必須使用標準符號，以方便共同閱讀和研討分析。
  - 繪製流程圖的方向應由上而下，自左到右。

- 流程圖中的文字說明要力求簡潔，而且要明確具體可行。
- 流程圖中線條應該避免太長或交叉，此時可以使用連接符號。

【範例6】：計算十個整數的平均數

撰寫「計算十個整數之平均值」的演算法

Q 首先，想一想「計算十個整數之平均值」應該如何做？

A 求一列整數的平均值是以該列整數的和除以該列整數之個數，配合一般程式語言語法，一個演算法如下列步驟：

第一步：程式開始
第二步：讀入一個整數
第三步：將這個整數的值加入總和變數
第四步：判斷是否已讀入10個整數了？
　　若已經讀入10個整數則跳到第五步
　　否則跳到第二步（再讀入一個整數）
第五步：計算平均值（將總和變數除以10）
第六步：印出平均值
第七步：程式結束

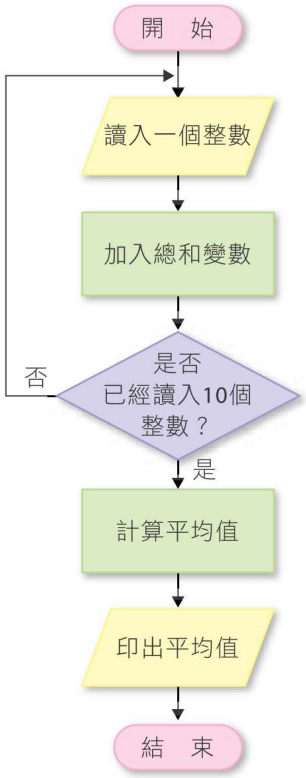


圖6-1.2 「計算10個整數之平均值」的流程圖。

```
1：  程式開始
2：  讀入一個整數
3：  加入總和變數
4：  若已經讀入10個整數
5：      計算平均值
6：      印出平均值
7：  否則
8：      回到第2步（讀入整數）
9：  結束判斷
10：  程式結束
```

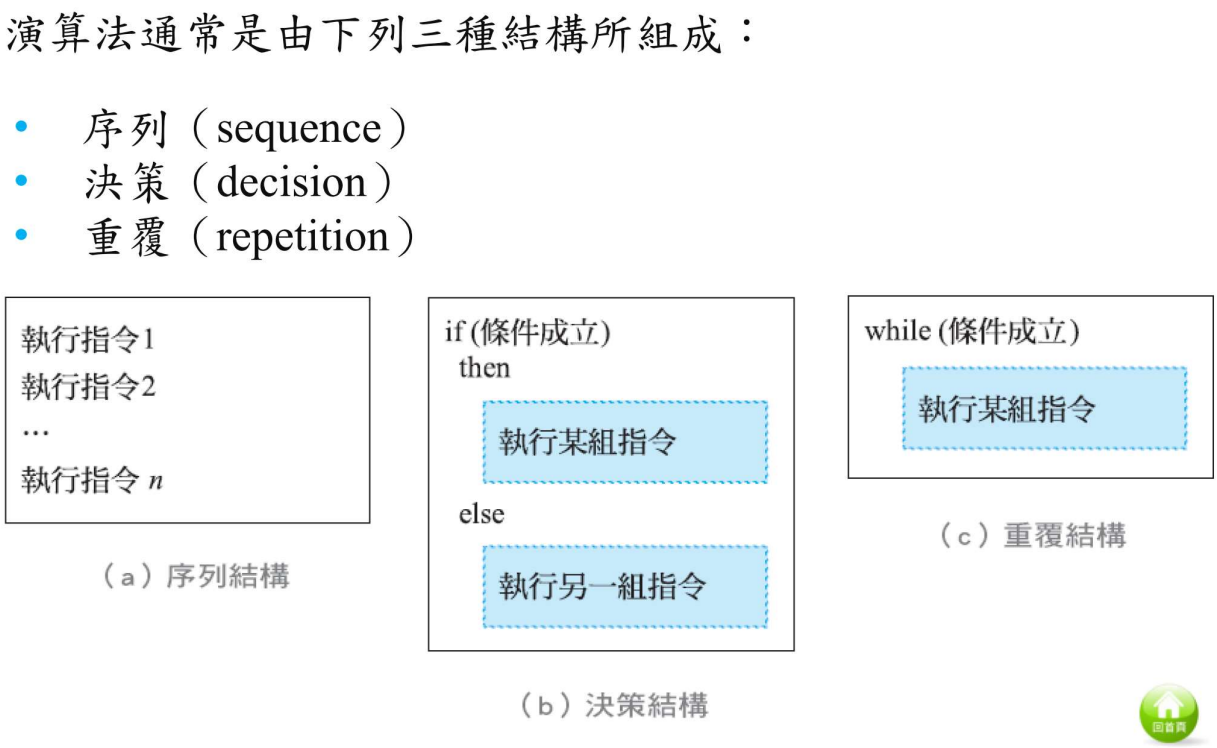
圖6-1.3 以虛擬碼表示「計算10個整數之平均值」。

```
##  虛擬碼
## Input Data
## 輸入十個整數的資料
```

```
## Process Data
## 計算十個整數的總和
## 代公式計算平均數
## Output Data
## 列印結果
```

8-3. 三種程式結構

一個程式，不管是多麼複雜，程式的流程主要有循序、選擇與重複三種結構。



- **序列結構 = 循序結構 Sequence**：是指程式指令由上到下依序執行的結構。

程式由上而下，依序一行一行逐行執行。第一行執行完畢後執行第二行，第二行執行完畢後執行第三行，直到程式執行結束。

- **決策結構 = 選擇結構 Decision**： 是指利用布林條件式的真偽來控制程式流程。

依照條件情況來選擇執行不同分支路徑。如果布林條件結果為真，就執行某些指令；如果布林條件式為偽，就執行另一些指令。選擇結構能夠讓程式選擇需要執行的程式碼，這也表示不是所有的程式碼都會被執行。

- **重覆結構 = 迴圈結構Repetition**：是以迴圈的方式，在程式的流程中反覆執行某些指令。

在程式的流程中，有些程式區塊需要重複執行若干次。可以直接指定執行的次數，或是利用布林條件式的真偽來控制執行次數。

~之前我們上的內容是「循序結構」，亦即基本程式設計的內容。現在要教「決策結構」，然後繼續教「重複結構」，這三個是結構化程式設計的基本邏輯架構。如果還有時間就會教函式（封裝），把這三個程式結構放在一起完成一個具體任務。通常，處理真實世界問題的大程式會包含很多的小程式，每一個小程式都會執行一個具體任務，這部分可以用函數處理。

例如，學校每一位學生都有你們高中與聯考的資料，也會有你們家庭背景的資料，以及大學四年就讀期間的資料。每部分的用途和功能不同，可以分開處理。

個的資料，每部分的用途和功能不同，可以分開處理。

範例7:舉出三種基本結構的例子

舉出三種基本結構的例子

Q 想一想，日常生活中有哪些事情符合以上介紹的三種結構？和同學討論一下，並分別舉出一個例子。

A 參考答案：

1. 循序結構：到醫院看病時要依序「掛號」→「候診」→「醫生診療」→「開處方」→「批價」→「領藥」。
2. 選擇結構：速食店的員工詢問顧客要點幾號套餐，並依顧客所點之餐號準備相關餐點。
3. 重複結構：電玩遊戲終結時會詢問是否再玩一次，若選擇「是」即可繼續遊戲關卡。

仔細想想，有些事情是不是必須由數種結構組合才能完成？的確如此，事實上寫程式也是類似的狀況。

補充：利用虛擬碼來表示三種基本結構

虛擬碼（pseudocode）是以近似於英文但語法更為精確的方式來描述問題的解法，下圖 是使用虛擬碼表示序列、決策及重覆等三種結構。

```
instruction 1
instruction 2
…
instruction n
```

（a）序列結構

```
if (condition)
then
  one set of instructions
else
  another set of instructions
```

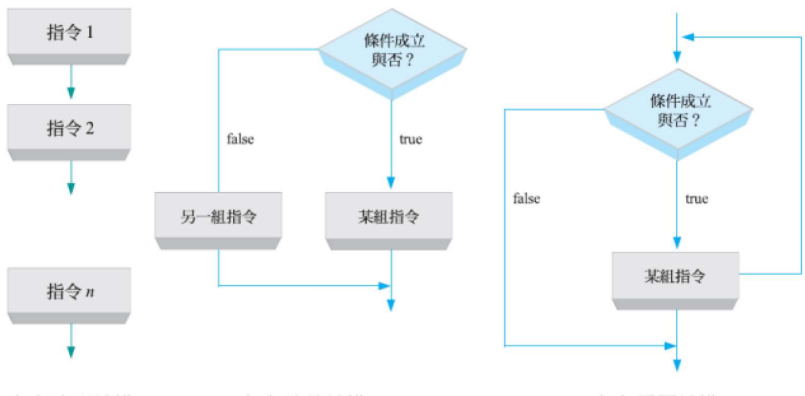
（b）決策結構

```
while (condition)
  one set of instructions
end while
```

（c）重覆結構

補充：利用流程圖來表示三種基本結構

流程圖（flowchart）是以圖形符號表示演算法，下圖是使用流程圖表示序列、決策及重覆等三種結構。




（a）序列結構
（b）決策結構
（c）重複結構