

【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



精選簡報・教師專用  
博碩文化・版權所有  
www.drmaster.com.tw

# 第五章程式設計

- 5-1 邏輯判斷
- 5-2 重覆迴圈
- 5-3 Spyder 除錯應用

# 邏輯判斷

邏輯判斷是程式設計最基本、最常用的指令。Python提供的邏輯判斷指令，分為三種用法：

(1)單純的if指令

(2)if else指令

(3)巢狀if else

主要功能是用於判斷不同情境採取不同的動作。在 if 後面加上判斷條件，並且在同一列最後加上「：」符號。指令區的指令必須要**內縮**，Python 會檢查指令區內的指令，若沒有內縮會發生列縮錯誤（IndentationError）。

# if 判斷

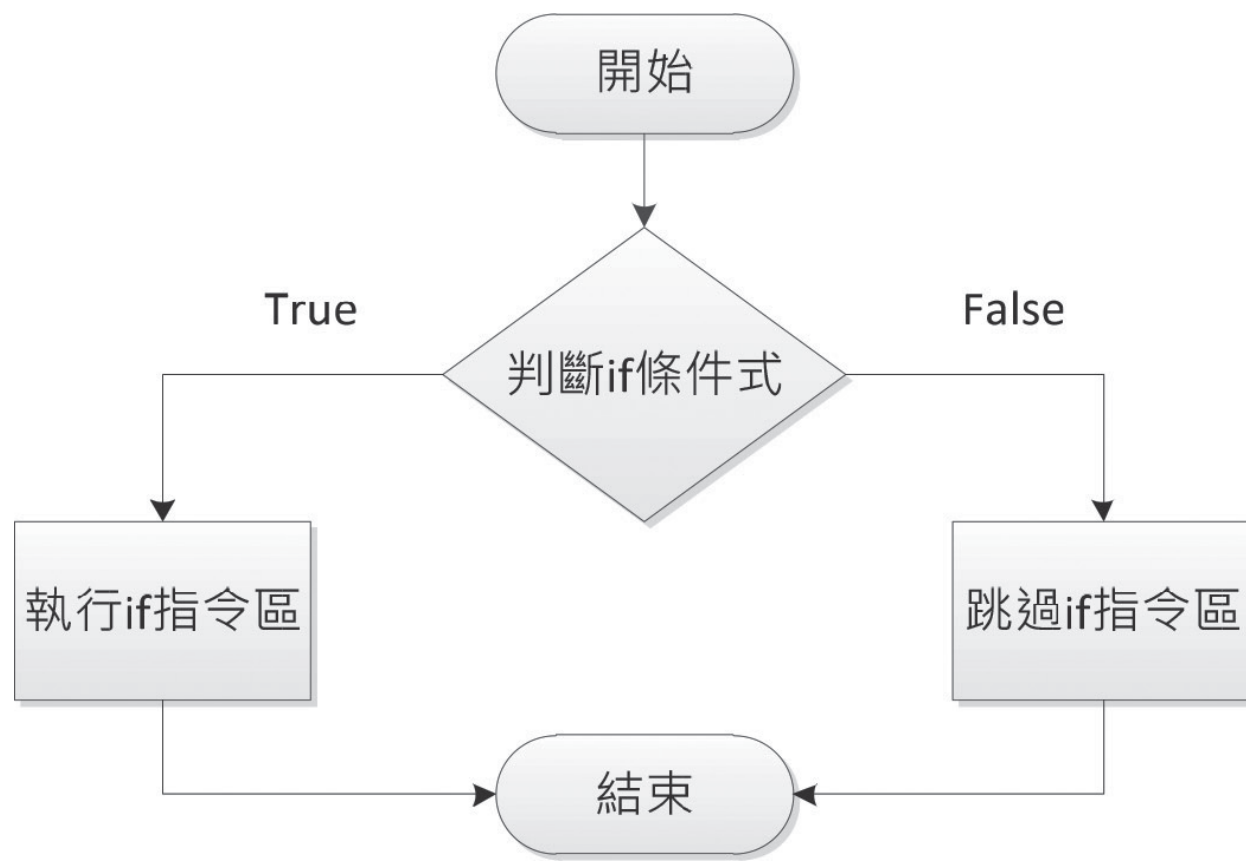


圖5-1 if 判斷流程圖

**範例 5-1** 輸入一個數值，判斷若小於 50 則開根號乘以 10。

### 示範程式碼

```
1 #E_5_1: 輸入一個數值，判斷若小於 50 則開根號乘以 10
2 import math
3 num=int(input(' 請輸入任一數 '))
4 Tt=num
5 if num<50:
6     Tt=math.sqrt(num)*10
7 print(num,Tt)
```

### 執行結果

```
1 請輸入任一數 40
2 40 63.24555320336759
```

# if else 判斷

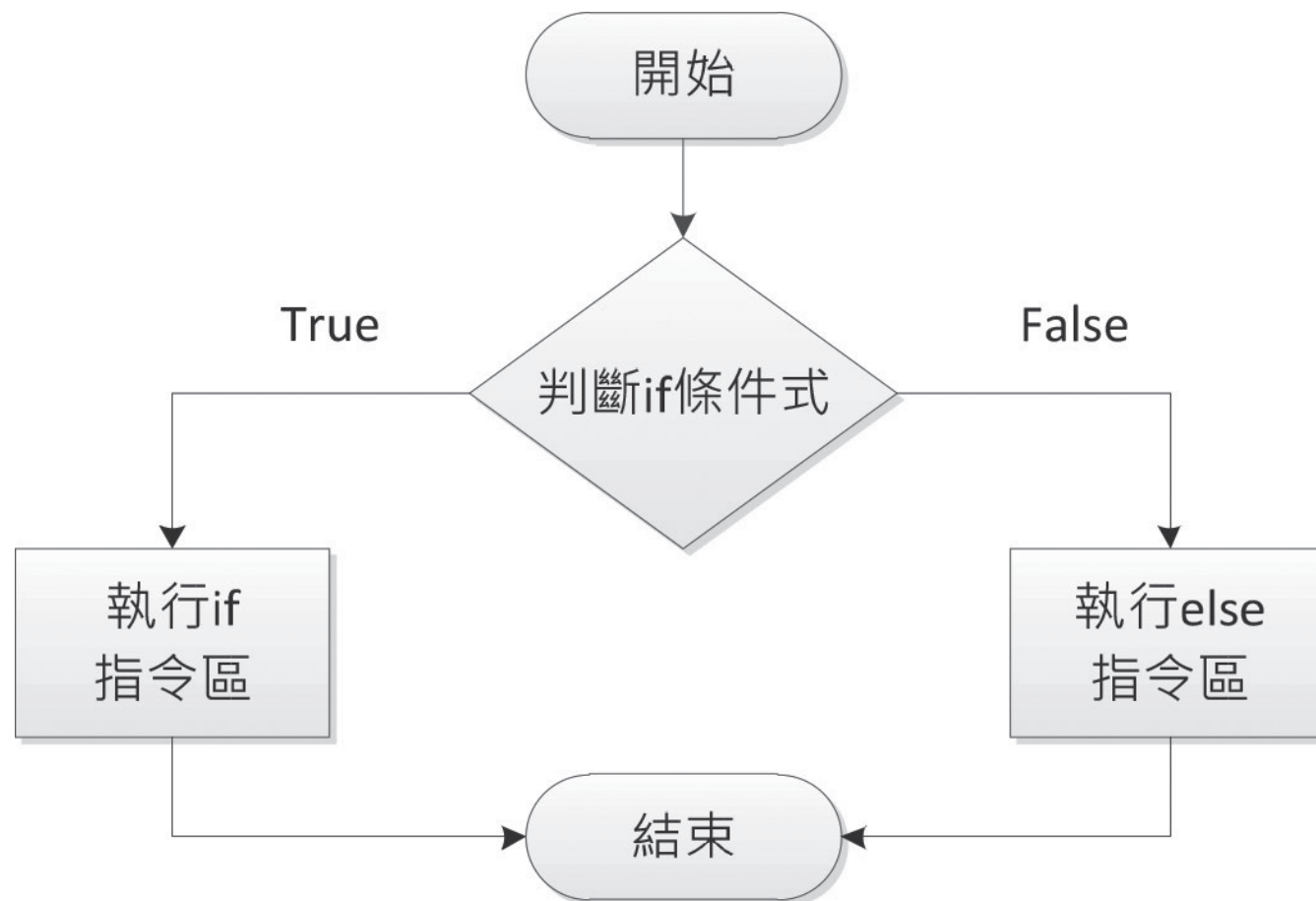


圖5-3 if else 判斷流程圖

**範例 5-2** 輸入一個數值，判斷若小於 50 則開根號乘以 10，否則加 10。

### 示範程式碼

```
1 #E_5_2: 輸入一個數值，判斷若小於 50 則開根號乘以 10，否則加 10。  
2 import math  
3 num=int(input(' 請輸入數據 1-100: '))  
4 if num<50:  
5     total=math.sqrt(num)*10  
6 else:  
7     total=num+10  
8 print('total=%d' % round(total))
```

### 執行結果

```
1 請輸入數據 1-100: 60  
2 total=70
```

# *if elif else* 判斷

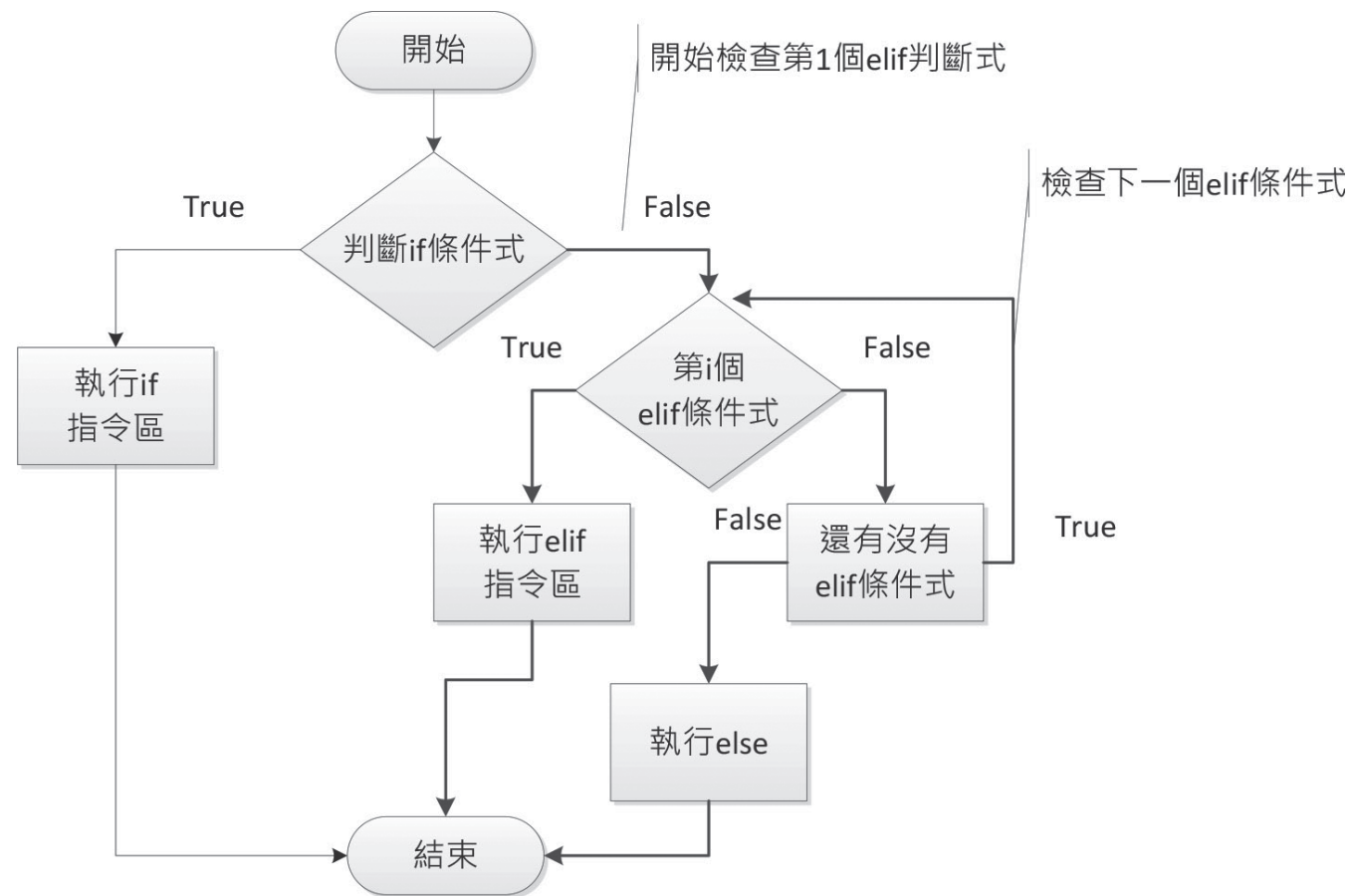


圖5-4 `if elif else` 判斷流程圖



### 範例 5-3 成績分等級

設計一個成績分等級的程式，利用 `if elif else` 指令將成績分成五個等級，成績 90 到 100 為甲等，成績 80 到 89 為乙等，成績 70 到 79 為丙等，成績 60 到 69 為丁等，成績 0 到 59 為戊等。

#### 示範程式碼

```
1 #E_5_3: 成績分等級
2 score=int(input(' 請輸入成績 0-100 分 : '))
3 grade=' 輸入 '
4 if score<60: # 條件 1
5     grade=' 戊等 '
6 elif score <70: # 條件 2
7     grade=' 丁等 '
8 elif score<80: # 條件 3
9     grade=' 丙等 '
10 elif score<90: # 條件 4
11     grade=' 乙等 '
12 elif score<=100: # 條件 5
13     grade=' 甲等 '
14 else: # 條件 6
15     grade=' 輸入錯誤 '
16 print (grade)
```



### 提示

多組級距的數值判斷，可以選最小或最大的一邊開始判斷，再一組一組往上或往下判斷，其結果是一樣的。

### 執行結果

```
1 請輸入成績 0-100 分 : 50
2 戊等
3 請輸入成績 0-100 分 : 91
4 甲等
5 請輸入成績 0-100 分 : 70
6 丙等
7 請輸入成績 0-100 分 : 101
8 輸入錯誤
```

### 結果說明

按「F5」執行程式時，IPython console 視窗（在 Spyder 右下角）會出現提示字串「請輸入成績 0-100 分 :」，使用者在冒號後面點滑鼠左鍵一次，即可輸入數值，假設輸入 50（見執行結果第 1 列），則符合條件 1 印出「戊等」（見執行結果第 2 列），以此類推（見執行結果第 3-8 列）。



# 重覆迴圈

迴圈每執行完一次，就會檢查是否到達停止條件。本節將介紹：

(1)單一for 迴圈

(2)巢狀for

(3)while

(4)break 和 continue

(5)range()函數無法處理浮點數的序列

(6)Spyder除錯應用

# 單一for迴圈

## 5-2-1 單一 for 迴圈

```
for 計數器 in range (start, end, step) :
```

```
    指令 1
```

```
    ...
```

```
    指令 n
```

- 步驟 1 會先藉由 range 產生可迭代的元素，並指派給計數器當初始值。
- 步驟 2 進入檢查迴圈是否到達終止條件，亦即檢查序列內是否還有存在元素，若有則條件成立為 True，表示序列內還有元素，繼續執行。
- 步驟 3 取出序列的元素，往下執行指令區的指令。再回到步驟 2. 檢查是否達到終止條件，若序列內沒有任何元素了，元素存在條件不成立為 False，即滿足終止條件，結束迴圈。



# 單一for 迴圈

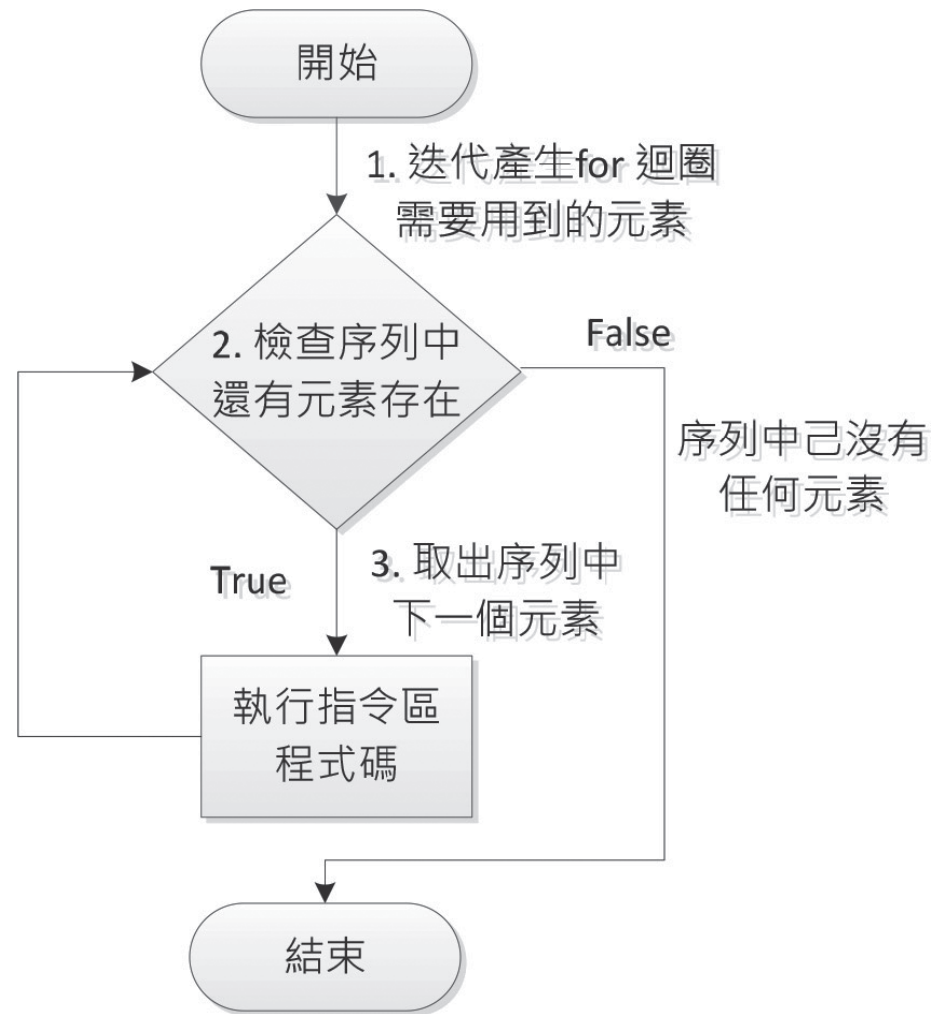


圖5-6 單一 for 迴圈流程圖

### 範例 5-6 計算 1 累加到 5 的總和。

#### 示範程式碼

```
1 #E_5_6 功能：計算 1 累加到 5 的總和
2 sumi=0
3 n=5
4 for i in range(1,n+1):
5     sumi=sumi+i
6     print(sumi)
7 print('sumi =',sumi)
```

#### 程式說明

`sumi` 變數是用來累加因子的總和（見第 2 列），`n` 設定為 5 是從 1 累加到 5（見第 3 列），接著開始執行迴圈，計數器為 `i`，產生 1 至 `n+1` 的序列，每次遞增加 1，可以省略不寫，`n+1=6`，是因為 `range` 的 `end` 只會產生到 `n-1`。因此，若要累加到 `n`，`range` 的 `end` 參數必須設 `n+1`，才會產生 1 至 5 的序列（見第 4 列）。



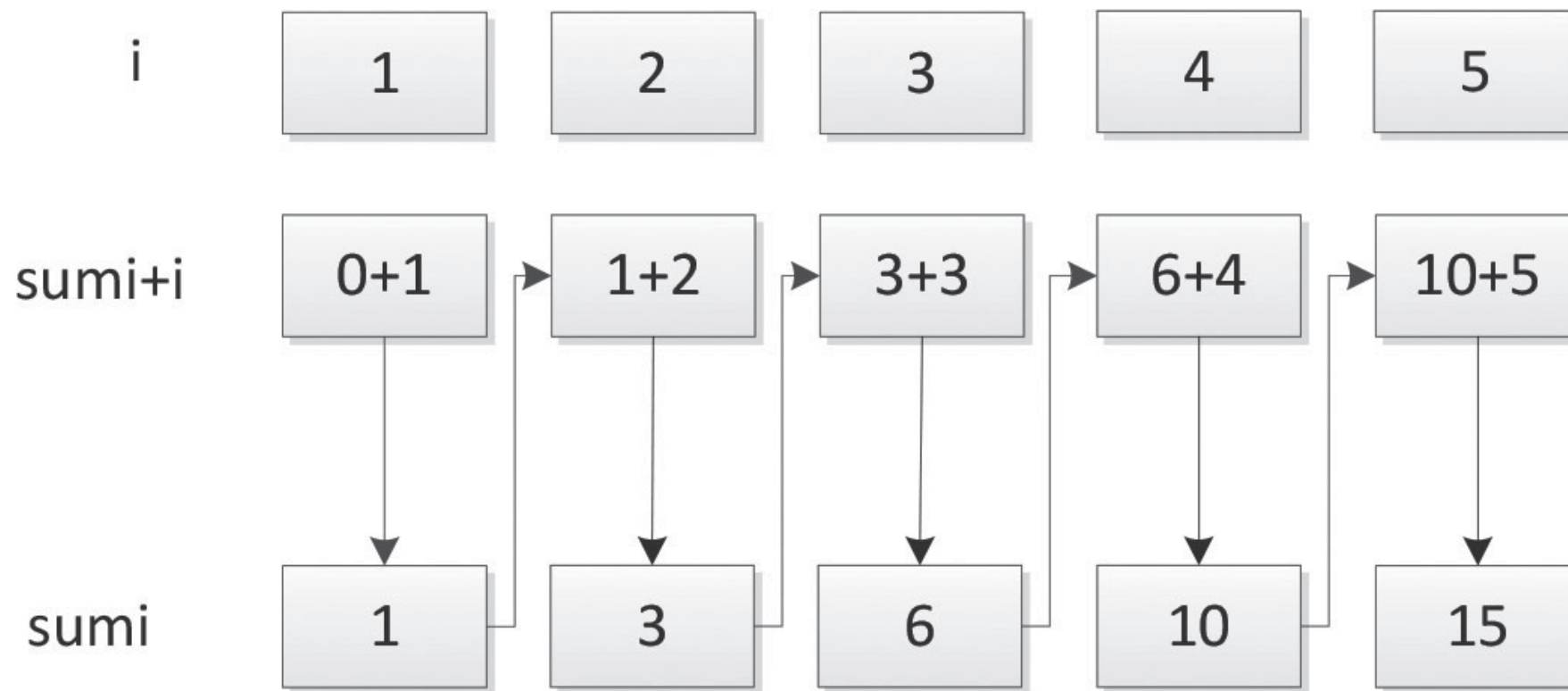


圖5-7 1 加到 5 的示意圖



### 提示

程式最重要是從問題中找出規則以及計算的流程，再設計程式碼。

### 執行結果

```
1  1
2  3
3  6
4 10
5 15
6 sumi = 15
```

### 結果說明

印出第 1 圈 `sumi` 加總後的執行結果對照圖 5-7 的最後一列值為 1 ( $0+1$ )，兩者的值是一樣的；第 2 圈 `sumi` 的累加結果為 3 ( $1+2$ )，第 3 圈 `sumi` 的累加結果為 6 ( $3+3$ )，以此類推見執行結果 2 到 5 列。最後一列是印出進開迴圈後，印出最後的 `sumi` 的結果（見執行結果第 6 列）。





# 巢狀for迴圈

## 5-2-2 巢狀 for

所謂巢狀迴圈是指迴圈內包含另一個迴圈，若有必要它可以組合 2 個以上的迴圈。執行順序是外圈執行 1 圈，內圈要執行所有圈數（內圈要執行 1 遍），指令的語法如下：

```
for 計數器 in range (start, end, step) :  
    for 計數器 in range (start, end, step) :  
        指令區  
    指令區
```

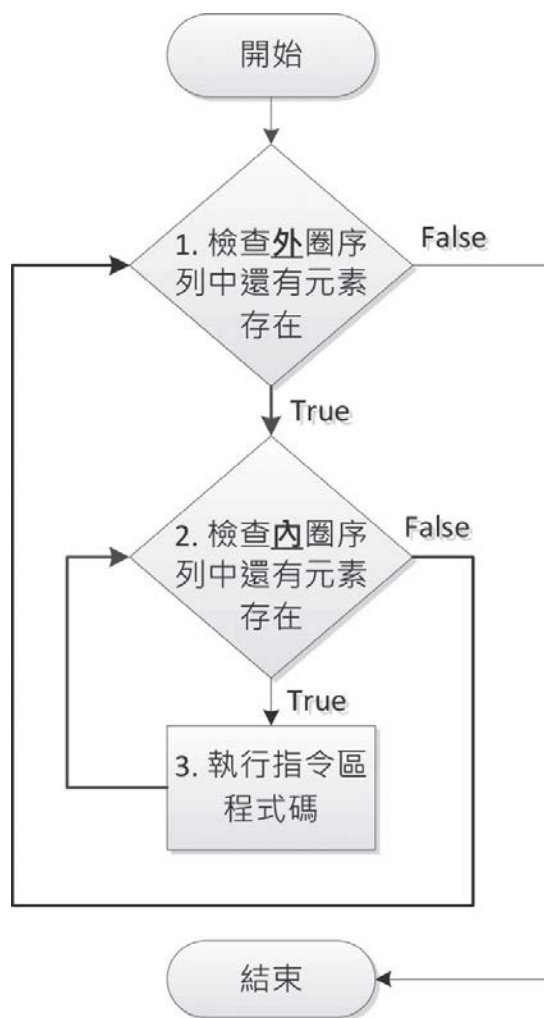


圖5-9 巢狀迴圈流程圖



### 提示

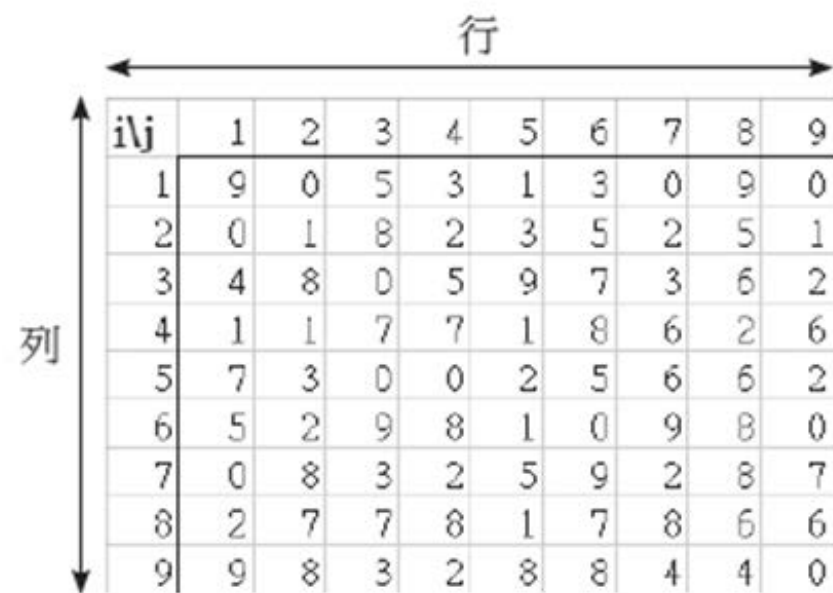
外圈執行一圈（取出一個元素），內圈要把所有圈數都執行完畢（即所有元素取完），才回到外圈繼續執行下一圈，再進內圈，直到外圈結束為止。

**範例 5-9** 以雙迴圈（9 列，9 行）產生亂數（0 到 9）模擬二維陣列。

### 示範程式碼

```
1 #E_5_9 功能：以雙迴圈 (9 列，9 行) 產生亂數 (0 到 9) 模擬二維陣列
2 import random as rd
3 for i in range(1, 10): # 外圈
4     for j in range(1, 10): # 內圈
5         num=rd.randint(0,9)
6         print('%3d'%(num), end="")
7     print('\n')
```

## 執行結果



行									
i\j	1	2	3	4	5	6	7	8	9
1	9	0	5	3	1	3	0	9	0
2	0	1	8	2	3	5	2	5	1
3	4	8	0	5	9	7	3	6	2
4	1	1	7	7	1	8	6	2	6
5	7	3	0	0	2	5	6	6	2
6	5	2	9	8	1	0	9	8	0
7	0	8	3	2	5	9	2	8	7
8	2	7	7	8	1	7	8	6	6
9	9	8	3	2	8	8	4	4	0

圖 5-10 雙迴圈的行與列

## *while* 迴圈

若要執行的重覆性工作，但是不確定要執行多少次時，則不能使用 for 迴圈，可改用 while 迴圈。while 迴圈可使用邏輯條件式做為判斷結束的條件。當邏輯條件為真（True）時，即進入迴圈內執行指令；直到邏輯條件為假（False）時結束迴圈，指令的語法如下：

while 邏輯條件式：

指令 1

...

指令 n



# while 迴圈

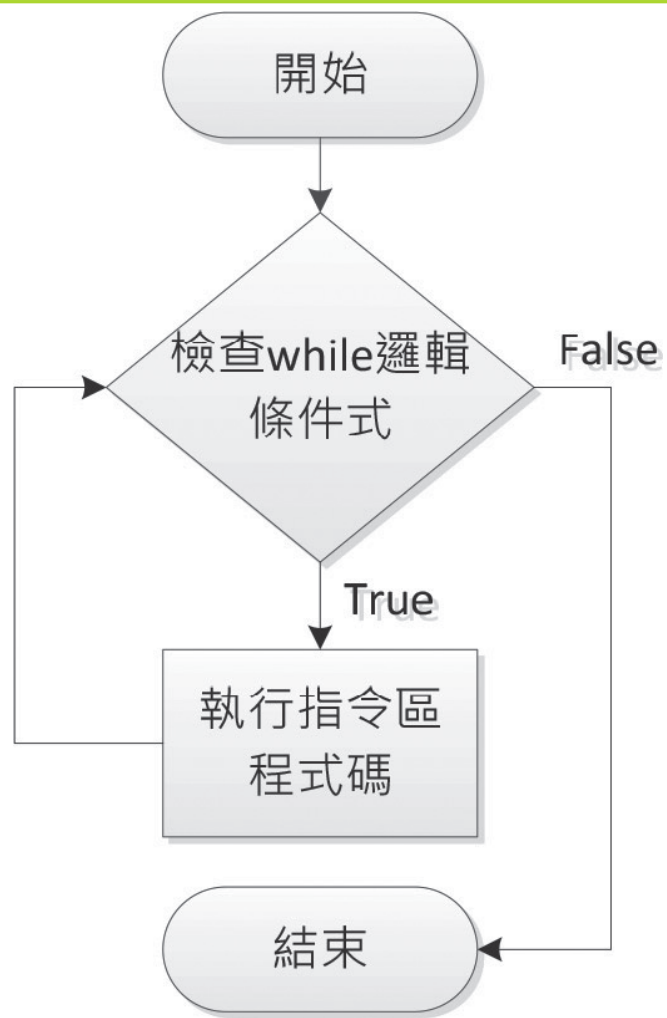


圖5-12 while 迴圈流程圖

**範例 5-11** 重覆產生亂數（0 至 9 的整數）直到產生的值是零時結束，印出每次產生的亂數值，並計算個數。

### 示範程式碼

```
1  #E_5_11 功能：重覆產生亂數 (0 至 9 的整數) 直到產生的值是零時結束。
2  import random as rd
3  num=rd.randint(0,9)
4  count=1
5  while num!=0:
6      print(num)
7      num=rd.randint(0,9)
8      count+=1
9  print(num)
10 print('%s%d%s' %('共產生亂數 ',count,'次'))
```





### 提示

若發生進入無窮迴圈現象時，可以在 IPython console 點選左選後，按 Ctrl+C 中止程式，訊息會出現 KeyboardInterrupt 的提示字串。

### 執行結果

```
1 4
2 1
3 6
4 3
5 2
6 4
7 8
8 0
9 共產生亂數 8 次
```



# 中斷迴圈

## 5-2-4 break 和 continue

有時候迴圈執行到一半時需要判斷，符合某種條件情境時要中斷迴圈，可以使用 `break` 與 `continue` 指令。`break` 指令用於中斷迴圈，忽略以下的指令不執行後，跳離該迴圈；`continue` 也是中斷迴圈，但忽略 `continue` 以下的指令，回到迴圈的第 1 列繼續執行。`while` 與 `for` 均可以搭配 `break` 與 `continue` 做控制，以 `while` 為例，其指令的語法如下：

while 邏輯條件式

if 判斷條件

break



while 邏輯條件式

if 判斷條件

continue



`while` 要使用 `break` 或 `continue` 要搭配 `if` 的條件式判斷是否滿足條件。以 `break` 為例，若 `if` 判斷條件式為真（`True`），則放棄執行以下的程式離開迴圈。若 `if` 判斷條件式為假（`False`），則繼續執行指令區其餘程式碼。指令區的程式碼執行完畢後，再回到 `while` 檢查是否滿足結束條件（見圖 5-13）。

另以 `continue` 為例，若 `if` 判斷條件式為真（`True`），一樣放棄執行以下的程式碼，回到迴圈開始，繼續執行下一圈；若 `if` 判斷條件式為假（`False`），則往下執行其餘的程式碼，再回到迴圈的開頭，直到迴圈滿足結束條件（見圖 5-14）。

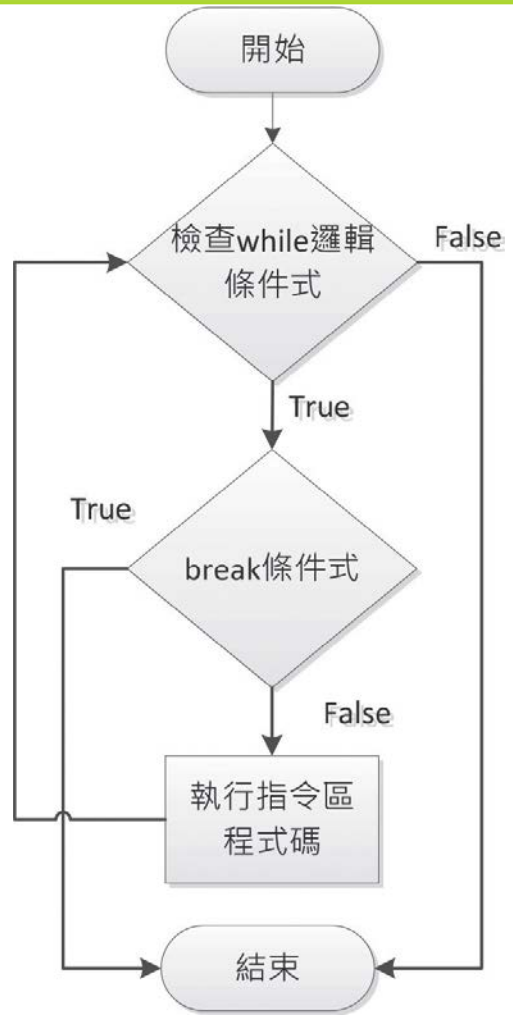


圖5-13 while 迴圈配合 break 流程圖

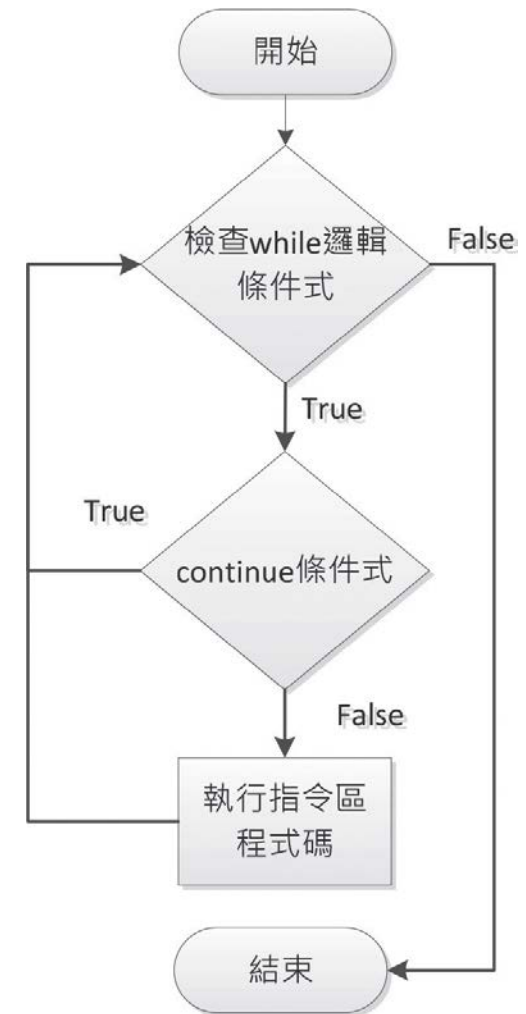


圖5-14 while 迴圈配合 continue 流程圖

**範例 5-12** 以 while 迴圈產生 9\*9 乘法表結合 break 與 continue，遇到第 4 列跳過，遇到第 7 行就結束。

### 示範程式碼

```
1  #E_5_12 功能：以 while 迴圈產生 9*9 乘法表結合 break 與 continue。
2  i=1
3  j=1
4  while i<=9: # 外圈
5      if i==4:
6          i+=1
7          continue
8      while j<=9: # 內圈
```

```
9      if j==7:
10          break
11          print("%d*%d=%2d"%(i,j,i*j), end=" ")
12          j+=1
13      j=1
14      i+=1
15      print("\n")
```

## 印出格式補充說明

印出的格式：`'%d*%d=%2d'%(i,j,i*j)` 代表的意思是三個參數都是用整數格式化輸出三個 `%d`，中間安插 `*` 與 `=` 是固定的符號，會照實印出。第 3 個 `%d` 用 2 個位元的位置是因為 `i*j` 相乘時數字較大的會有位數（見圖 5-11）。

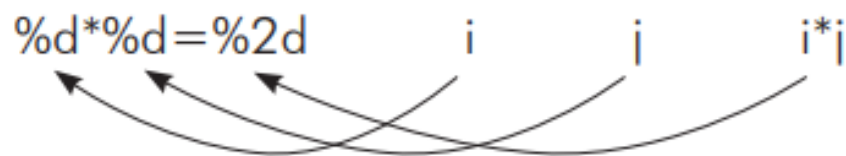


圖 5-11 print 格式化輸出與參數的對映關係圖

## 執行結果

1*1= 1	1*2= 2	1*3= 3	1*4= 4	1*5= 5	1*6= 6
2*1= 2	2*2= 4	2*3= 6	2*4= 8	2*5=10	2*6=12
3*1= 3	3*2= 6	3*3= 9	3*4=12	3*5=15	3*6=18
5*1= 5	5*2=10	5*3=15	5*4=20	5*5=25	5*6=30
6*1= 6	6*2=12	6*3=18	6*4=24	6*5=30	6*6=36
7*1= 7	7*2=14	7*3=21	7*4=28	7*5=35	7*6=42
8*1= 8	8*2=16	8*3=24	8*4=32	8*5=40	8*6=48
9*1= 9	9*2=18	9*3=27	9*4=36	9*5=45	9*6=54

## 結果說明

列的部分少了第 4 列，共只印出列 1 到 3 與 5 到 9。因為使用 `continue`，所以第 5 列開始仍會印出，共 8 列；行只印到第 6 行。因為，內圈裡面邏輯判斷只要遇到 7 就停（`break` 指令），所以不會印出第 7 行（含）以後的資料。



# Spyder 除錯

## 5-3 Spyder 除錯應用

程式要寫得好寫得快，學會除錯技巧是很重要的。除錯（debug）是指一種能用於偵測程式碼錯誤的工具。除錯可以檢查程式碼執行狀況、選擇性的執行指定的部分程式碼，尤其迴圈與邏輯判斷指令。當程式撰寫過程中遇到瓶頸或找不出錯誤時，除錯技術可以協助找出錯誤，提升撰寫程式的效能。

要說明 Spyder 如何協助程式碼除錯，可帶入一個範例，說明程式除錯時如何追蹤變數值的變化軌跡，協助改正程式碼。

### 範例 5-16 排列組合

排列組合是計算排列和組合後可能出現的情況總數，即輸入整數  $m$  與  $n$ ，計算  $C_n^m$ 。計算公式如下：

$$C_n^m = \frac{m!}{n! \times (m - n)!}$$





## 示範程式碼

```
1  # E_5_16 功能：排列組合。
2  m=int(input(' 輸入 m= '))
3  n=int(input(' 輸入 n= '))
4  prodi=1 # 計算 m!
5  prodj=1 # 計算 n!
6  prodk=1 # 計算 m-n!
7  for i in range(m,1,-1):
8      prodi=prodi*i
9  for j in range(n,1,-1):
10     prodj=prodj*j
11     k=m-n
12     for k in range(k,1,-1):
13         prodk=prodk*k
14     number=prodi/(prodj*prodk)
15     print('%s %d %s' %(' 排列組合共有 ', number, ' 方法 '))
```



## 程式說明

輸入公式要用到的  $m$  與  $n$  (見第 2 到 3 列)，再設定 3 個累乘變數初始值為 1:  $prodi$ (計算  $m!$ )， $prodj$ (計算  $n!$ )， $prodk$ (計算  $(m-n)!$ ) (見第 4 到 6 列)。進第 1 個 for 迴圈計算  $m!$ ，for 迴圈的計數器  $i$  是由  $m$  遞減到 1 (見第 7 列)，執行迴圈的指令  $prodi=prodi*i$ ，累乘後指派給新值給  $prodi$ 。第 9 到 10 列是計算  $n!$ ，並將  $n!$  累乘結果指派新值給  $prodj$ 。 $m-n$  指派給  $k$  (見第 11 列)，進第 3 個迴圈計算  $k!$  (見第 12 到 13 列)。最後將  $prodi$  除以  $prodj*prodk$ ，再印出結果 (見第 14 到 15 列)。

## 執行結果

- 1 輸入  $m=10$
- 2 輸入  $n=7$
- 3 排列組合共有 120 方法

輸入  $m=10$ ， $n=7$  的排列組合共有 120 方法。



## 除錯操作

以【範例 5-16】為例，本節所討論的除錯操作，已在第 2 章初步介紹過，本節將配合範例的程式碼，實際說明除錯的步驟。

### 1. 中斷點

若要偵測某一行，可以在那一行最左邊（列號左邊）點選左鍵一次，會出現紅點，即為中斷點（●），使用迴圈撰寫程式的過程中，使用中斷點搭配變數追蹤視窗可以及時發現運行中的錯誤。本例，將中斷點設在第 8 行（見圖 5-16）。

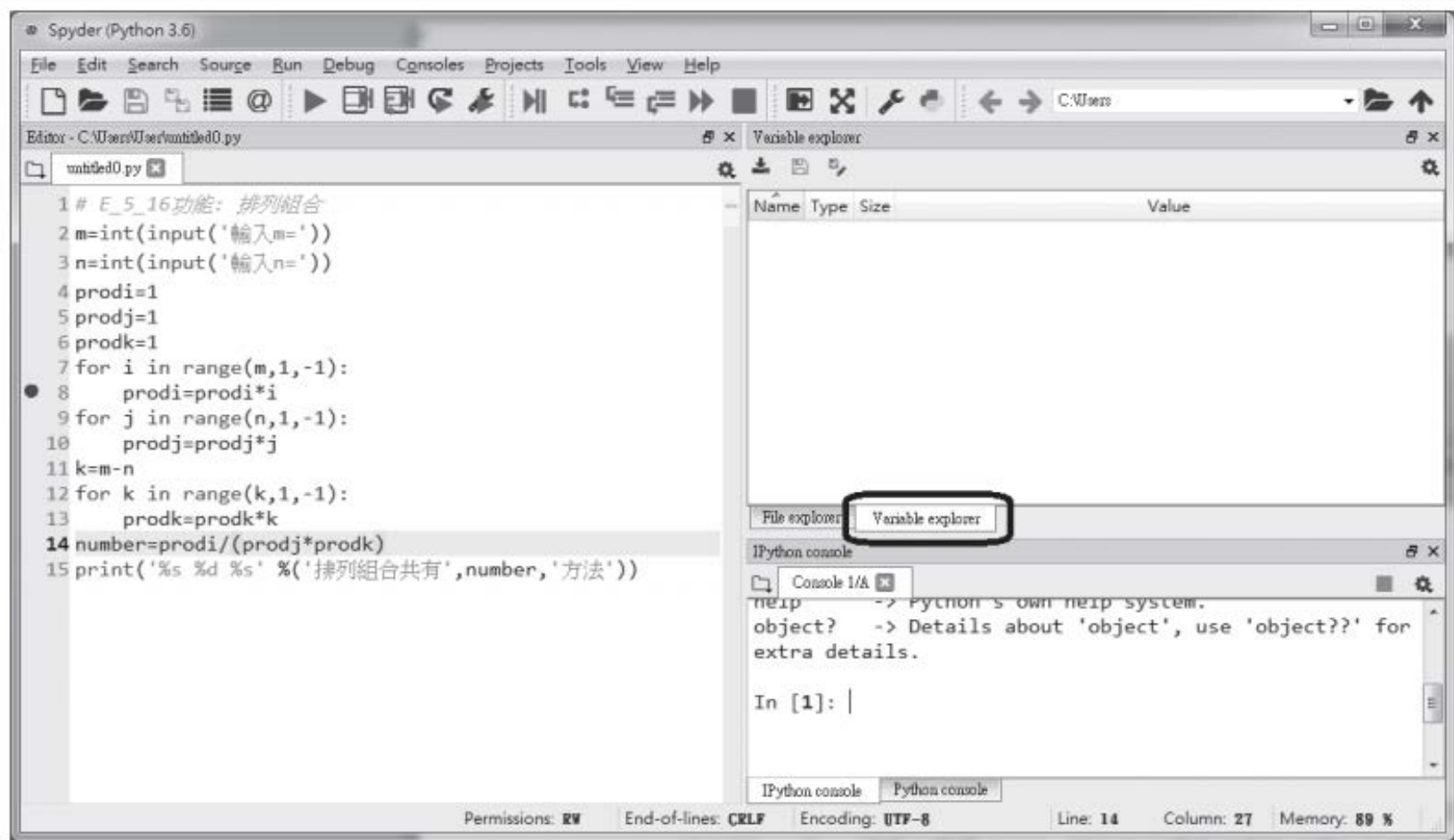



圖 5-16 設定中斷點



## 2. 變數追蹤視窗 (Variable explorer)

Spyder 開啟時右上角預設是 Object Inspector (指令查詢視窗)，點選右邊的 Variable explorer 視窗 (見圖 5-16)，可以在此視窗檢視程式碼執行過程中變數的名稱 (Name)、型別 (Type)、大小 (Size) 以及內容值 (Value) 的改變。第一次執行 Variable explorer 是空白的，執行過程式碼後所有變數即會存在，即使執行別一支程式，之前執行過的變數仍會存在視窗中，一直到離開 Spyder，所有變數才會消失。或者點選該變數後，再點選右鍵後按 remove，即可刪除單一變數。

## 3. Debug file

Python 是直譯式語言，不需要編譯器 (Compiler) 即可執行程式碼。在 Spyder 的功能視窗按下 Debug file ，即會從第 1 列執行到中斷點，除非有 input() 函數等待使用者輸入資料後，即會繼續往執行直到中斷點後暫停。停在中斷點這一系列時，該列並未執行。此時可以看到 IPython console 會提示目前執行到第 8 列之前暫停 (見圖 5-17) 右下角視窗，並用箭頭指向未執行的那一系列如下：



```
7 for i in range(m,1,-1):  
1---> 8 prodi=prodi*i  
9 for j in range(n,1,-1):
```

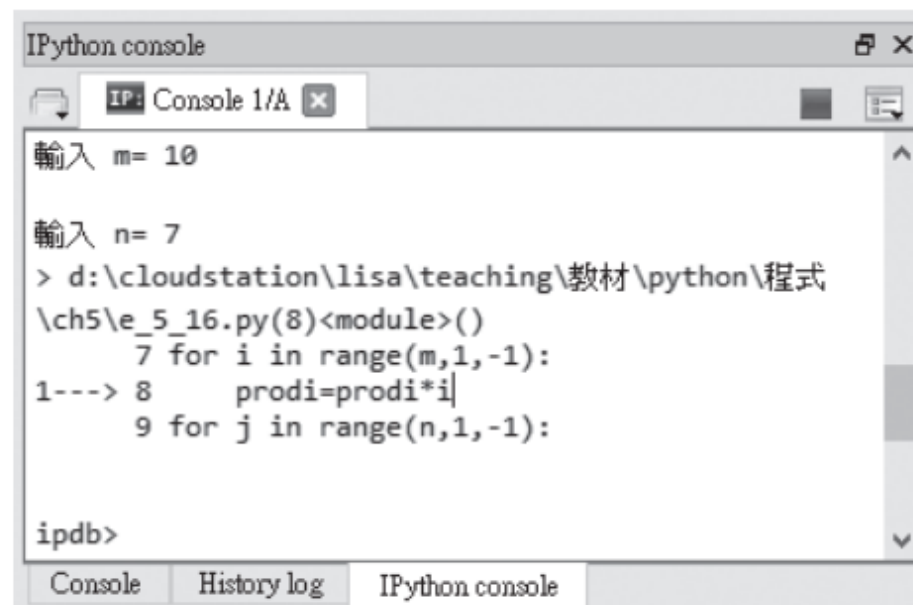



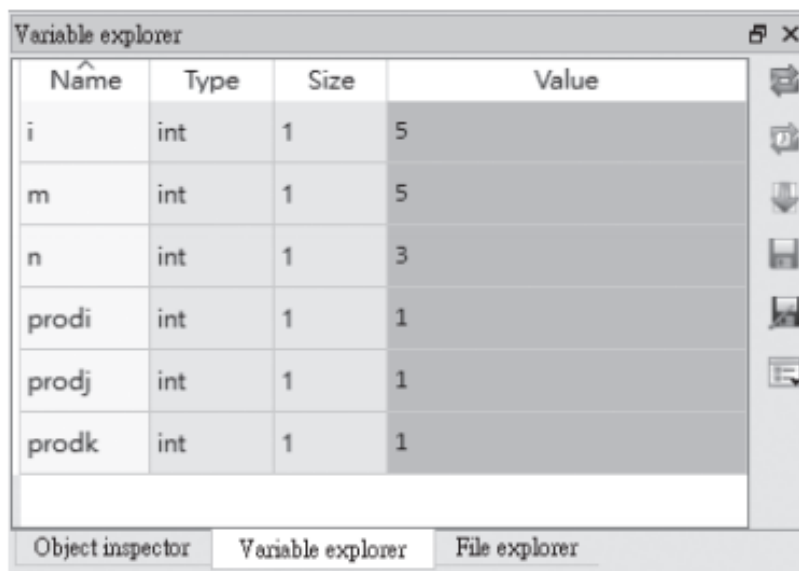
圖 5-17 IPython console 視窗中斷提醒



此時可以追蹤 Variable explorer 視窗，prodi, prodj, prodk 目前還是初始值 1，i 是第 8 列 for 迴圈的計數器初始值是 5，m 與 n 是第 2 到 3 列的 input 函數使用者輸入的值已轉成整數（見圖 5-18）。

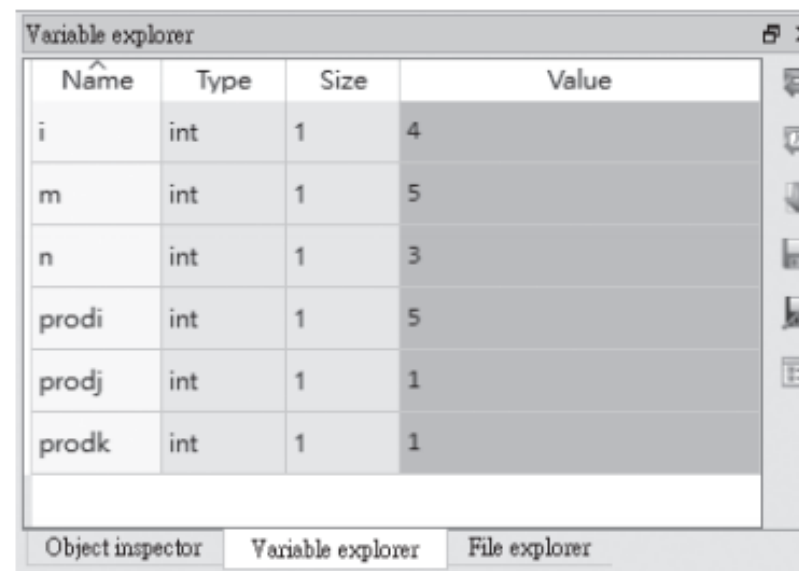
#### 4. 繼續執行直到下個中斷點

本節只試算第 7 列的迴圈，第 1 圈 i 的初始值 5 之後進迴圈，因為停在第 8 列，此時第 8 列並未執行  $\text{prodi}=\text{prodi}*i$ 。因此，prodi 的值仍為初始值 1（第 7 列未執行過的結果），必須點選  後，才會實際執行第 8 列後回到迴圈，繼續下一圈。第 2 圈時 i 的值已更新為 4，prodi 更新為 5（第 1 圈執行過的結果  $1*5=5$ ），第 2 圈尚未執行）（見圖 5-19）以此類推，往下追蹤。



Name	Type	Size	Value
i	int	1	5
m	int	1	5
n	int	1	3
prodi	int	1	1
prodj	int	1	1
prodk	int	1	1

圖 5-18 追蹤變數視窗 (1)



Name	Type	Size	Value
i	int	1	4
m	int	1	5
n	int	1	3
prodi	int	1	5
prodj	int	1	1
prodk	int	1	1

圖 5-19 追蹤變數視窗 (2)

## 5. 離開除錯模式

如果發現變數的內容值與預期不同，有錯時則可以進行修改。在修改程式之前，必須先離開除錯模式，點選 (■) 即可。



# 本章講解完畢

現場同學們如有不懂的地方，  
請提出問題。

