屏東縣第 60 屆國中小學科學展覽會

作品說明書

科　　別：數學科

組　　別：國中組

作品名稱：如何看透迴圈─以 python 為例

關 鍵 詞：等差級數、迴圈、程式語言

編號：

摘要

本研究乃因應在資訊科技課程中，將學習程式語言經常碰到的迴圈，從數學角度發展相對的變化並探討不同圖形所需要的迴圈層數。我們利用現行廣受歡迎並較為親切的 Python 語言，透過所需輸出的圖形分層來討論之。從單層的迴圈結構到多數層的巢狀迴圈結構研究的過程中，發現輸出圖形的規律，分析變數（iterator，又稱疊代子）的數學式子來轉化為程式語言。最後發展出能快速辨別迴圈層的個數及所需的變數種類。

壹、 研究動機

在學習程式語言中，迴圈指令通常是初學著碰到的首要瓶頸，如果能夠突破並更加熟悉迴圈的指令，很多程式設計問題都可以迎刃而解。而老師讓我們練習最多的，花的時間也最多的地方，就是迴圈。電腦每秒鐘可以執行幾億次的指令，而迴圈更是可以讓電腦重覆執行某個程式區塊好多次，就好比我剛開始學程式語言打的一行”hello！”，電腦就可以列印一次出來，有了迴圈的指令，要電腦列印”hello！”幾百次幾千次都沒有問題。因此老師為了讓我們都熟悉迴圈的程式指令運作，出了好幾題星星或數字的排列圖形，要我們能利用迴圈的特性輸出。

在寫程式的過程中，可以感受到圖形的變化及排列，跟可以利用的迴圈層數可能有相關性，因此便和數學老師去討論圖形中相關的數學問題。

貳、 研究目的

我們欲研究指定輸出圖形的規律，利用數學的概念，探討相關的程式語言中，迴圈該如何分佈才會成功輸出。最後整理出如何從不同的圖形中，其變化的規律來判斷出對應的迴圈。

參、 研究設備及器材

電腦、Python 程式語言、Eclipse 開發平台、excel 軟體、紙、筆

肆、　研究過程或方法

在寫程式的過程中，利用迴圈的時機主要有下列幾點：一、用來重覆直行程式的片段，二、許多程式問題通常有著規則性，例如：1 1 1 2 2 2 3 3 3 …或 1 2 2 3 3 3 4 4 4 4…；我們必須從指定輸出的格式中，模擬問題並使用迴圈重覆執行的特性，加入適當迴圈內執行的式子。

我們在此研究中，為了輸出指定的格式或圖形，可能在變數達到某一個值才需要執行別的程式指令，因此我們須先了解 Python 中的判斷指令，其語法介紹如下：

if A:

|→ B

(備註：|→代表縮排，tab 鍵)

如果 A 為真，則執行程式區塊 B。

另外，此研究還需瞭解 Python 中迴圈的其中一種指令：for 迴圈，經常與 range 合併使用，其語法介紹如下：

for i in range(起始值,終止值的前一項,間隔):

執行程式區塊

…

…

其中變數 i，每執行完一次程式區塊，若程式沒有特別指定，會換行來輸出下一圈，並且根據 range 裡面的間隔式疊代，若有小於終止值，則再進入迴圈執行程式區塊，直到大於或等於終止值，就會跳出迴圈。

起始值可省略不寫，則從 0 開始；間隔也可省略不寫，預設值為 1。

舉例而言：

一、

example1

```
1 for i in range(1,6):
2     print("*")
3
```

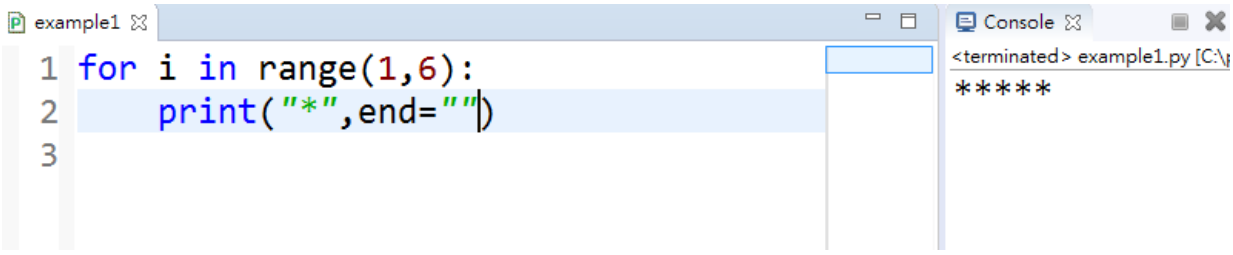
Console

<terminated> example1.py [C:\py

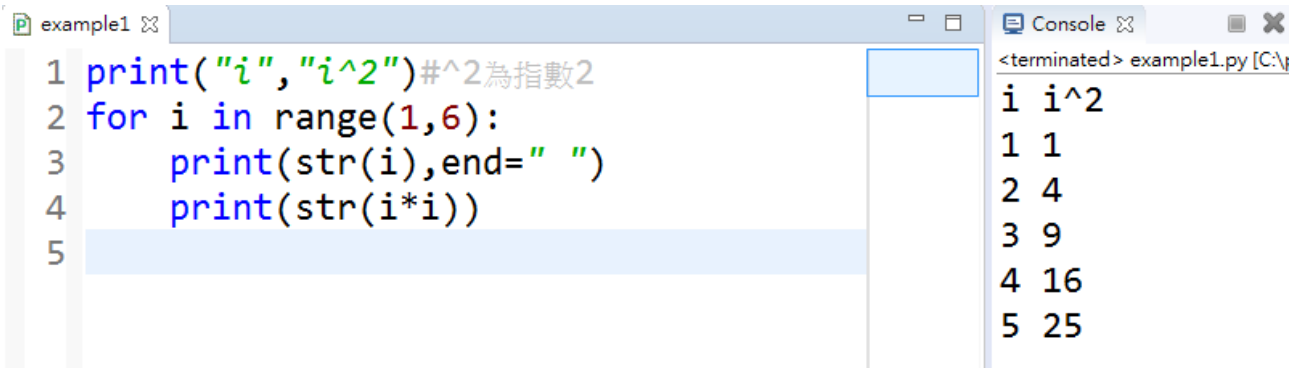
```
*
*
*
*
*
```

每印完一次「*」即完成一圈程式，因此會換行，所列印出來的圖形是一行共 5 顆星星的圖形，而變數共執行了 5 次，分別是從 i=1~5（5 為終止值 6 的前一個數）。

二、我們可以發現，若在 print 指令中，指定結尾不換行，就會形成一列 5 顆星星。如下：



三、除了列印星星圖形之外，我們也可以直接列印數字，來發展變化，例如輸出完全平方數：



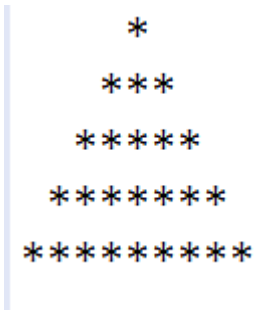
我們欲研究輸出圖形與程式語言之間的關係，為了記錄變數的變化，我們會隨著記錄並製成表格。

當拿到題目時，我們需要開始分析圖形的複雜度，嘗試先用紙筆思考，並觀察規律性變化的趨勢，猜測可能需要的迴圈層次，並記錄迴圈中變數（疊代子）的遞增或遞減，還有增或減的幅度，就能著手釐清程式的架構。此外，一剛開始寫程式通常會出錯，可以依據編譯器執行錯誤的訊息一一去修正，有可能迴圈的位置安排錯誤，也有可能是變數的數學式子理解錯誤，就會導致整個圖形走偏了，只要經常練習，嘗試錯誤，通常題目就在這些過程中迎刃而解了。

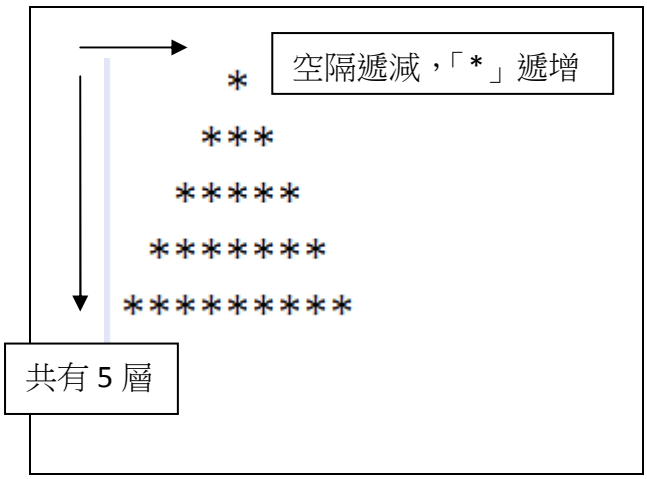
因此，我們在此研究將圖形分類為：「單層迴圈」、「數個單層迴圈」、「雙層迴圈」、「三

層迴圈」、「四層迴圈」。以下一一探討：

一、單層迴圈

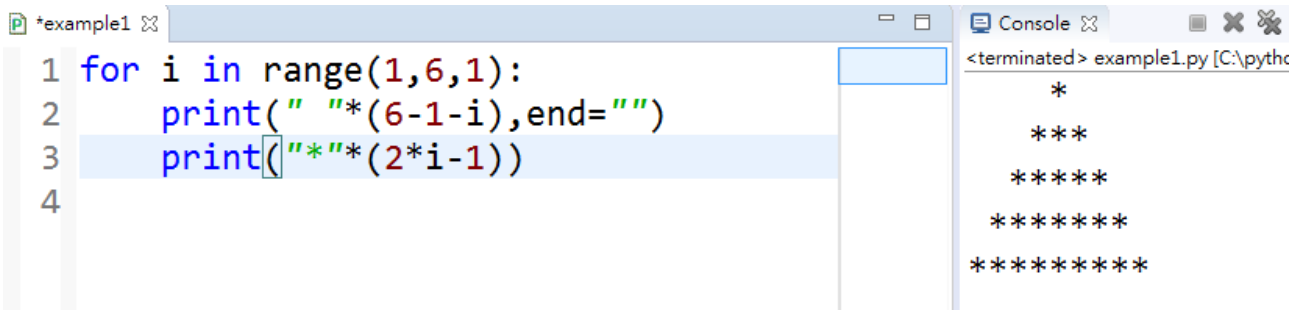
輸出左列圖形：	

在上列圖形中，要注意到空格的數量變化是有規律的，也要考量進去。在圖形上我們觀察的規律有：



列數	空隔數	星星數
1	4	1
2	3	3
3	2	5
4	1	7
5	0	9

由上表可發現，列數+空隔數=5，而星星數亦為奇數的變化，因此若只用一層迴圈，一種變數（i）作變化，就可以輸出此圖形，如下：



我們將此輸出圖形一般化，若指定列印共 n 層上三角型圖案，由於 5 對應到 n，則

只需將 6 改成 n+1，即可：

example1

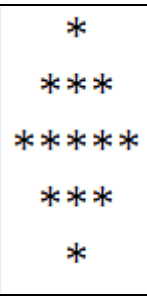
```
while True:
    iLevel=int(input("輸入你要的層數："))
    for i in range(1,iLevel+1):
        print(" "*(iLevel+1-1-i),end="")
        print("*"*(2*i-1))
```

Console

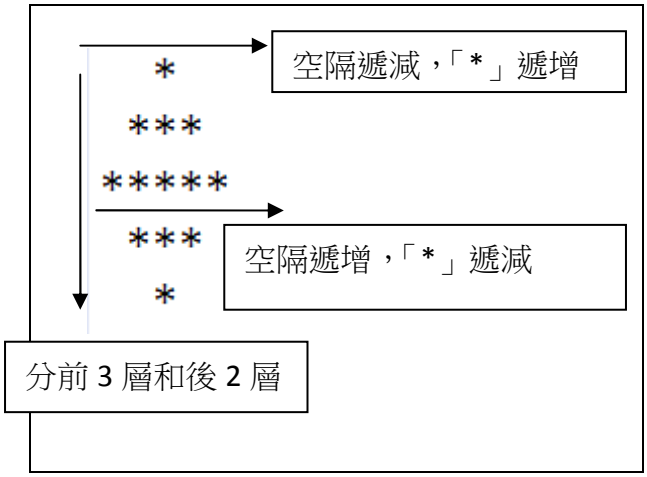
```
example1.py [C:\python\python.exe]
輸入你要的層數：5
    *
   ***
  *****
 *******
*********
輸入你要的層數：7
      *
     ***
    *****
   *******
  *********
 ***********
*************
```

註：程式中的 iLevel 即為 n

二、數個單層迴圈

輸出左列圖形：	

在上面圖形中，顯然是上下對稱的，星星的數量在前三層為遞增，後兩層為遞減；反之空格的數量前三層遞減，後兩層為遞增。更進一步記錄如下：



列數	空隔數	星星數
1	2	1
2	1	3
3	0	5
4	1	3
5	2	1

若要將此輸出圖形一般化，指定列印共 n 層上三角型圖案結合 n-1 層下三角形圖案，由於 5 對應到 n，則只需將 6 改成 n+1，即可：

```
while True:
    iLevel=int(input("輸入你要的層數："))
    for i in range(1,iLevel+1):
        print(" "*(iLevel+1-1-i),end="")
        print("*"*(2*i-1))
    for j in range(iLevel-1,0,-1):
        print(" "*(iLevel-j),end="")
        print("*"*(2*j-1))
```

```
example1.py [C:\python\python.
輸入你要的層數：4
   *
  ***
 *****
*******
 *****
  ***
   *
輸入你要的層數：5
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

註：iLevel 即為 n

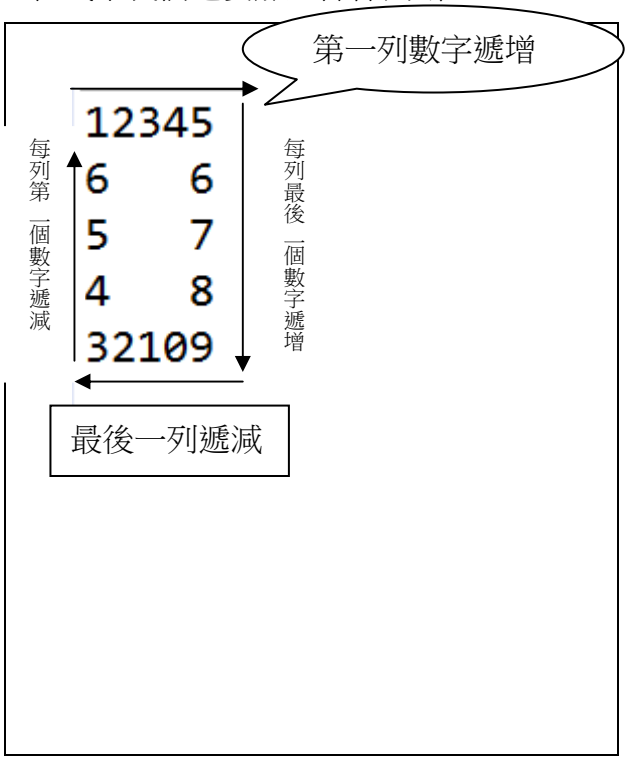
上述圖形在程式中，用了兩次單層迴圈。

三、雙層迴圈

輸出左列圖形：	12345 6 6 5 7 4 8 32109

在上面圖形中，乍看之下，數字的呈現為順時針遞增，每 0~9 一個循環，除了第一列與最後一列分別有遞增與遞減的規則外，總共要列印 5 層，其餘的列在每一列的頭跟尾有輸出數字外，中間為空格，且空格的數量為固定，明顯有三種情形要考量，因此在

程式中我們還要加入條件判斷：if…elif…elif…else。更進一步記錄如下：



列數	空格數	數字變化	輸出的位置與數字
1	0	遞增	1,2,3,4,5
2	3	頭+尾=12	位置 1，輸出 6
			位置 5，輸出 6
3	3	頭+尾=12	位置 1，輸出 5
			位置 5，輸出 7
4	3	頭+尾=12	位置 1，輸出 4
			位置 1，輸出 8
5	0	遞減	位置 1，輸出 3
			位置 2，輸出 2
			位置 3，輸出 1
			位置 4，輸出 0
			位置 5，輸出 9

由於在程式中迴圈指令無法中途要求疊代子從遞增改為遞減，且程式在跑一列要做的工作裡面，還要再依序輸出數字，可見在這一圈迴圈中，還有一層迴圈要用來處理輸出數字。因此我們考慮，在第 1 列(圈)及第 5 列(圈)中加入一個負責列印數字變化的迴圈。

在第一列中，輸出的數字正好為 1~5，而最後一列也就是第 5 列的最後一個數字，是接續第 1 列的最後一個數字往下數 4 列，因此再加 4。並且往左邊數 4 個數，可找出第 5 列的第 1 個數字，我們視為起始值 5+4+4，終止值 5+4，疊代數為－1。

討論完第 1 列跟第 5 列後，中間的 2~4 列格式較為固定，但不難發現最後一個數字正好為 5+(列數)－1；而第 1 個數字為第 5 列的第 1 個數字 5+4+4+(5-列數)。中間的空格數為 5－2＝3 格。

程式如下：

```python
for i in range(1,6):
    if i==1:
        for j in range(1,6):
            print(str(j),end="")
        print("")
    elif i==5:
        for j in range((5+4+4),(5+4-1),-1):
            print(str(j%10),end="")
        print("")
    else:
        print(str((5+4+4+(5-i))%10)+" "*(5-2)+str(5+i-1))
```

```
12345
6   6
5   7
4   8
32109
```

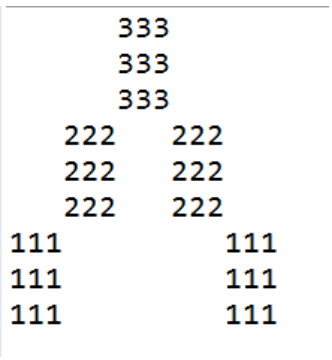
若要將此輸出圖形一般化，指定列印共 n 列的螺旋外圍矩陣形狀，由於 5 對應到 n，並將 4 改成 n－1，即可：

```python
while True:
    n=int(input("輸入:"))
    for i in range(1,(n+1)):
        if i==1:
            for j in range(1,n+1):
                print(str(j),end="")
            print("")#換行
        elif i==n:
            for j in range((n+n-1+n-1),(n+n-1-1),-1):
                print(str(j%10),end="")
            print("")#換行
        else:
            print(str((n+n-1+n-1+(n-i))%10)+" "*(n-2)+str((n+i-1)%10))
```

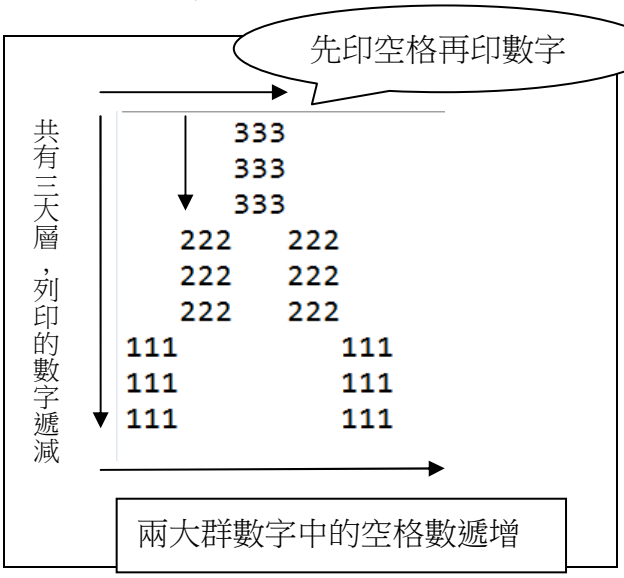
程式執行畫面：

```
0225.py [C:\python\python
輸入:4
1234
2  5
1  6
0987
輸入:6
123456
0    7
9    8
8    9
7    0
654321
```

四、三層迴圈

輸出左列圖形：	

此輸出圖形觀察，雖然有 9 列需要輸出，但其實是有 3 大層，每一大層又有 3 列要處理，而這當中的每一列就是列印同樣的數字 3 次或者，3 個同樣數字後接著空格又 3 個同樣數字。



列數	第一區空格數	第一區數字	第二區空格數	第二區數字
1~3	3×2	3 個 3	無	無
4~6	3×1	2 個 2	3×1 格	3 個 2
7~9	3×0	3 個 1	3×3 格	3 個 1
	遞減	遞減	遞增	遞減

三大層具有同樣形式的輸出，在最外圍用一層迴圈處理這三大層，變數由 1 遞增到 3；而在每一大層中，又有 3 列同樣並重複性的輸出，因此再安排一層迴圈，在這一層迴圈中，先處理第一區的空格數，是可以隨著這一層迴圈遞減的。而每一列當中，更有一層迴圈要列印第一區的數字，而第二區空格數及第二區數字也可能隨著設定。

程式如下：

```
1  for i in range(1,4):
2      for j in range(1,4):
3          print(" "*(3*(3-i)),end="")
4          for k in range(1,4):
5              print(str(4-i),end="")
6          if i !=1:
7              print(" "*(3*(2*i-3))+str(4-i)*3)
8          else:
9              print("")
10
```

```
    333
    333
    333
 222   222
 222   222
 222   222
111       111
111       111
111       111
```

若要將此輸出圖形一般化，假設 n 為輸出的最大數字，由於 3 對應到 n，並將 4 改成 n＋1，空格數的 3*(3-i)則改為 n*(n-i)，程式如下圖：

```
while True:
    n=int(input("輸入最大整數:"))
    for i in range(1,n+1):
        for j in range(1,n+1):
            print(" "*(n*(n-i)),end="")
            for k in range(1,n+1):
                print(str(n+1-i),end="")
            if i !=1:
                print(" "*(n*(2*i-3))+str(n+1-i)*n)
            else:
                print("")
```

其輸出結果如下圖：

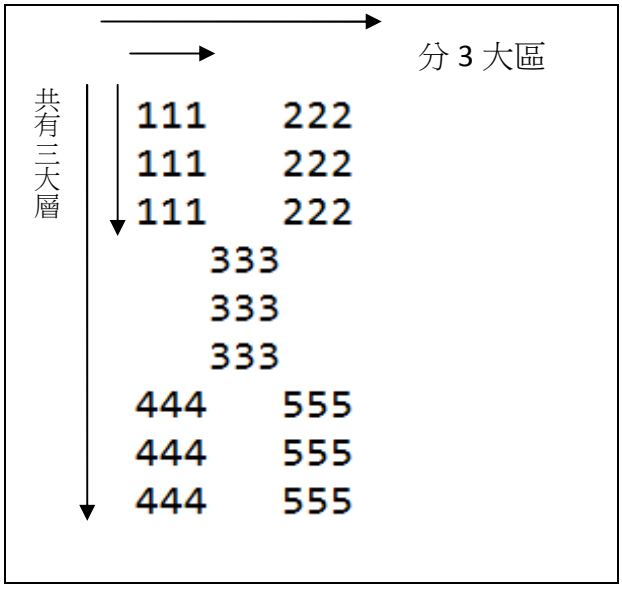
```
輸入最大整數:3
      333
      333
      333
   222   222
   222   222
   222   222
111         111
111         111
111         111
輸入最大整數:4
            4444
            4444
            4444
            4444
        3333    3333
        3333    3333
        3333    3333
        3333    3333
    2222            2222
    2222            2222
    2222            2222
    2222            2222
1111                    1111
1111                    1111
1111                    1111
1111                    1111
```

```
輸入最大整數:5
                55555
                55555
                55555
                55555
                55555
           44444    44444
           44444    44444
           44444    44444
           44444    44444
           44444    44444
       33333            33333
       33333            33333
       33333            33333
       33333            33333
       33333            33333
   22222                    22222
   22222                    22222
   22222                    22222
   22222                    22222
   22222                    22222
11111                            11111
11111                            11111
11111                            11111
11111                            11111
11111                            11111
```

五、四層迴圈

輸出左列圖形：	111　　222 111　　222 111　　222 　　333 　　333 　　333 444　　555 444　　555 444　　555

初步觀察此圖，和上一題類似，雖然有 9 列需要輸出，但其實是有 3 大層，每一大層又有 3 列要處理，而這當中在第 1 層與第 3 層的每一列就是列印同樣的數字 3 次或者列印 3 個空格；唯獨第二層的每一列是先列印 3 次空格再列印 3 次數字。



列數	大層變數	大層重複列數	橫列分區數	區重複動作
1~3	1	3	3	3
4~6	2	3	3	3
7~9	3	3	3	3
迴圈	迴圈 A	迴圈 B	迴圈 C	迴圈 D

此輸出圖形，從最外圍縱向來看，有 3 大層變化（迴圈 A），而每一大層中又有 3 列具有同樣形式的輸出（迴圈 B），分析完縱向變化後，再觀察每一橫列的輸出變化：也是分 3 大區（迴圈 C），每一區再重複列印 3 次元素（迴圈 D）。大致上只有 2 種變化，第 1 種是印完數字再空格接著又印數字；第 2 種是先印空格再印數字，我們可以使用條件判斷處理之。

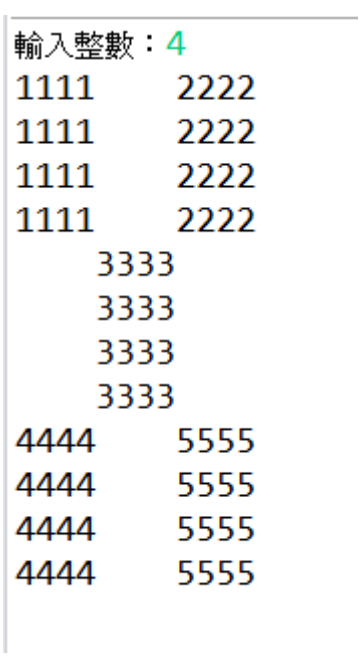
程式如下：

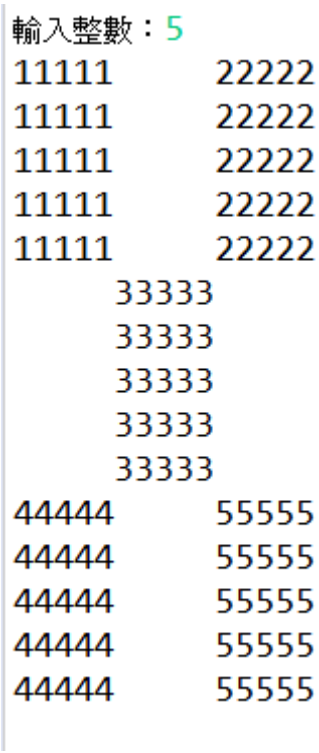
```
for i in range(1,4):
    for j in range(1,4):
        for k in range(1,4):
            for z in range(1,4):
                if i==1:
                    if k==1:
                        print("1",end="")
                    elif k==2:
                        print(" ",end="")
                    else:
                        print("2",end="")
                if i==2:
                    if k==1 or k==3:
                        print(" ",end="")
                    else:
                        print("3",end="")
                if i==3:
                    if k==1:
                        print("4",end="")
                    elif k==2:
                        print(" ",end="")
                    else:
                        print("5",end="")
        print("")
```

上述題目若要一般化，意即每一個 3×3 的矩陣放大或縮小到 n×n，除了迴圈 C 的 range 仍為(1,4)之外，其他迴圈可以將原本程式中的 range 終止值改為 n+1，程式其餘區塊（含條件判斷）皆不變，省時很多，這就是寫程式的樂趣之一。程式如下：

```
while True:
    n=int(input("輸入整數："))
    for i in range(1,n+1):
        for j in range(1,n+1):
            for k in range(1,4):
                for z in range(1,n+1):
                    if i==1:
                        if k==1:
                            print("1",end="")
                        elif k==2:
                            print(" ",end="")
                        else:
                            print("2",end="")
                    if i==2:
                        if k==1 or k==3:
                            print(" ",end="")
                        else:
                            print("3",end="")
                    if i==3:
                        if k==1:
                            print("4",end="")
                        elif k==2:
                            print(" ",end="")
                        else:
                            print("5",end="")
            print("")
```

輸出結果如下：





伍、 研究結果

在程式語言中，「迴圈」扮演很重要的角色，我們分析以上五種圖形輸出之後，可以發現迴圈使用的時機點就是：有規則的重複步驟。而每一個圖形的程式就是由一些有規則的與數學式子交錯組合而成，只要發現有規則的步驟，就用迴圈代替，沒有規則則用一般的判斷或數學式子。

這些程式題目中，通常隱藏著很多數學思考在裡面，乍看之下總是不好釐清，此研究當中我們發現，可以遵循下列步驟：

一、簡化題目，固定範圍（當問題有些複雜且找不出規律時，試著簡化成可完成的圖形中某部分小圖形）。

二、觀察圖形中重複性及規律性的地方。

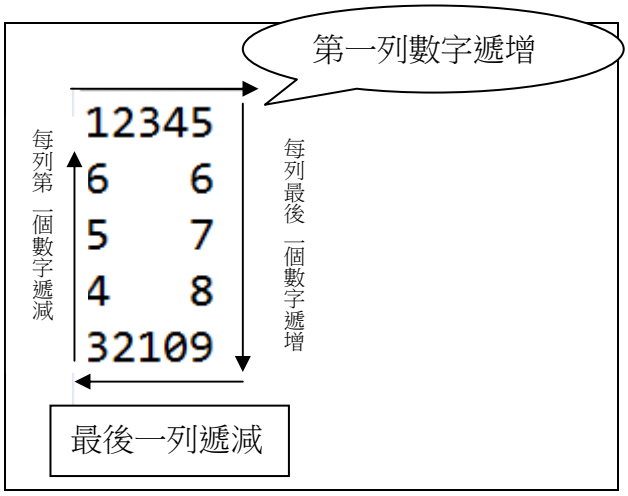
三、設定變數起始值，藉由表格記錄變數規律變化的趨勢。

四、從表格中決定迴圈的配置

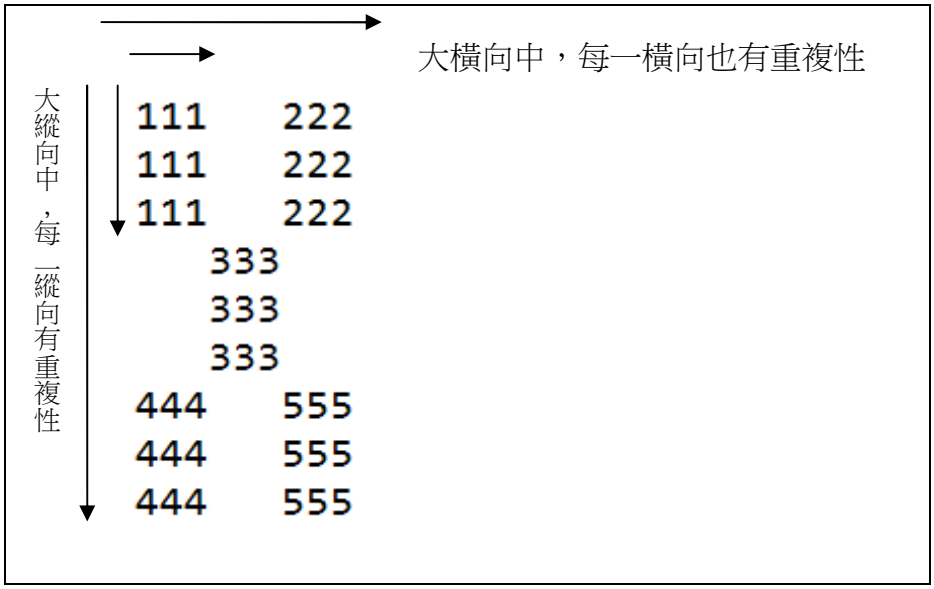
五、著手寫程式，漸進修改程式。

陸、 討論

第三題圖形開始，轉為矩陣的樣式呈現，開始需要雙層迴圈。如下：



在第五題的圖形中，觀察大縱向變化後，發現縱向又有重複次數；而在大橫向變化中，每橫向也有重複次數，因此使用了四層迴圈，如圖所示：



我們可以歸納出需要多層迴圈的時機為：

（1）圖形呈矩陣方向變化

（2）變數有交錯變化

柒、 結論

在本研究中只利用程式語言的條件判斷指令，與 for 迴圈來解題。其實要解類似規

律性、重複性的題目，還有 while 迴圈可以使用；若對程式有更進一步的興趣，還可以學習用串列（list），配合數學思考來解決。而且 python 內建了很多強大的數學相關模組，未來可以更廣泛開發運用。

捌、 參考資料

吳維漢（2018）。**簡明 python 學習講義**。台北市：遠流出版社。

健康資訊工程實驗室（2016）。**菲絲恩教您學會 Python 第二版**。新北市：博碩文化。

康軒版　國中數學第四冊第一章第一節 數列