
Python 迴圈 Loop (II)

臺北科技大學資訊工程系

for 巢狀(nest)迴圈

□ 迴圈內還有迴圈，變數的**值如何改變**?程式**流程如何變化**?

```
01 def myPrint1(num):  
02     for x in range(1, num+1):  
03         for y in range(num, x, -1):  
04             print('%d,%d;' % (x, y), end="")  
05         for y in range(1, 2*x, 1):  
06             print('%d,%d;' % (x, y), end="")  
07         print()  
08  
09 myPrint1(4)
```

劃出流程圖
寫下執行編號順序
寫下輸出內容

```
01 (1,4);(1,3);(1,2);(1,1),  
02 (2,4);(2,3);(2,1),(2,2),(2,3),  
03 (3,4);(3,1),(3,2),(3,3),(3,4),(3,5),  
04 (4,1),(4,2),(4,3),(4,4),(4,5),(4,6),(4,7),
```

for 巢狀(nest)迴圈

□ 迴圈內還有迴圈，變數的值如何改變?程式流程如何變化?

```
01 def myPrint1(num):
02     for x in range(1, num+1):
03         for y in range(num, x, -1):
04             print('%d,%d;' % (x, y), end='')
05         for y in range(1, 2*x, 1):
06             print('%d,%d;' % (x, y), end='')
07         print()
08
09 myPrint1(4)
```

劃出流程圖
寫下執行編號順序
寫下輸出內容

```
01 (1,4);(1,3);(1,2);(1,1),
02 (2,4);(2,3);(2,1),(2,2),(2,3),
03 (3,4);(3,1),(3,2),(3,3),(3,4),(3,5),
04 (4,1),(4,2),(4,3),(4,4),(4,5),(4,6),(4,7),
```

灰色區域放 #
白色區域放 y

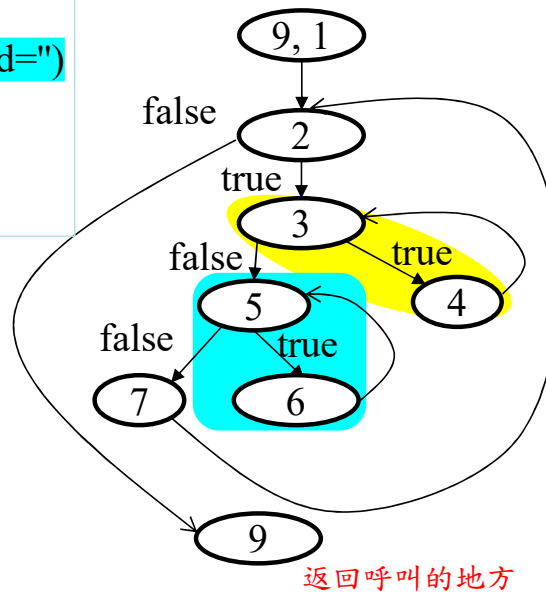
```
9, 1,
2, # x = 1~4, x = 1
3, # x=1, y = 4~2, 內層迴圈1
4,3 # x=1, y = 4
4,3 # x=1, y = 3
4,3 # x=1, y = 2
5, # x=1, y = 1~1, 內層迴圈2
6,5 # x=1, y = 1
7,
2, # x = 1~4, x = 2
3, # x=2, y = 4~3, 內層迴圈1
4,3 # x=2, y = 4
4,3 # x=2, y = 3
5, # x=2, y = 1~3, 內層迴圈2
6,5 # x=2, y = 1
6,5 # x=2, y = 2
6,5 # x=2, y = 3
```

for 巢狀(nest)迴圈

□ 迴圈內還有迴圈，變數的**值如何改變**? 程式**流程如何變化**?

```
01 def myPrint1(num):
02     for x in range(1, num+1):
03         for y in range(num, x, -1):
04             print('%d,%d;' % (x, y), end='')
05         for y in range(1, 2*x, 1):
06             print('%d,%d;' % (x, y), end='')
07         print()
08
09 myPrint1(4)
```

劃出流程圖
寫下執行編號順序
寫下輸出內容



```
01 (1,4);(1,3);(1,2);(1,1),
02 (2,4);(2,3);(2,1),(2,2),(2,3),
03 (3,4);(3,1),(3,2),(3,3),(3,4),(3,5),
04 (4,1),(4,2),(4,3),(4,4),(4,5),(4,6),(4,7),
```

灰色區域放 #
白色區域放 y

```
9, 1,
2, # x = 1~4, x = 1
3, # x=1, y = 4~2, 內層迴圈1
4,3 # x=1, y = 4
4,3 # x=1, y = 3
4,3 # x=1, y = 2
5, # x=1, y = 1~1, 內層迴圈2
6,5 # x=1, y = 1
7,
2, # x = 1~4, x = 2
3, # x=2, y = 4~3, 內層迴圈1
4,3 # x=2, y = 4
4,3 # x=2, y = 3
5, # x=2, y = 1~3, 內層迴圈2
6,5 # x=2, y = 1
6,5 # x=2, y = 2
6,5 # x=2, y = 3
```

for 巢狀(nest)迴圈

□ 迴圈內還有迴圈，印出數字金字塔

```
def myPrint1(num):
```

```
    for x in range(1, num+1):
```

```
        for y in range(num, x, -1):
```

```
            print('#', end="")
```

```
        for y in range(1, 2*x, 1):
```

```
            print(y, end="")
```

```
        print()
```

```
myPrint1(4)
```

第一層迴圈，控制層數

第二層迴圈，控制每一層個數，個數需要根據第一層迴圈的索引值變數計算

若是有兩種圖，需要有第二個第二層迴圈

並且兩種圖形，中間不能換行

每印出一層，要有換行

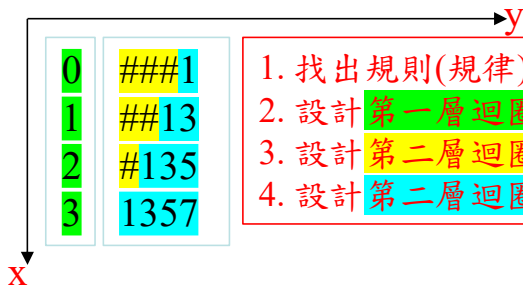
```
01 (1,4);(1,3);(1,2);(1,1),
02 (2,4);(2,3);(2,1),(2,2),(2,3),
03 (3,4);(3,1),(3,2),(3,3),(3,4),(3,5),
04 (4,1),(4,2),(4,3),(4,4),(4,5),(4,6),(4,7),
```

灰色區域放 #
白色區域放 數字 y

```
###1
##123
#12345
1234567
```

for 巢狀(nest)迴圈

□ 迴圈內還有迴圈，印出數字金字塔




0	###1
1	##13
2	#135
3	1357

1. 找出規則(規律)
2. 設計 **第一層迴圈**，圖形有幾層
3. 設計 **第二層迴圈**，每一層印幾個
4. 設計 **第二層迴圈** 要列印的資料

第一種圖形

1. 規則，N層，**N-1, N-2, ..., 3, 2, 1, 0**
2. 設計第一層迴圈，圖形有N層
3. 設計第二層迴圈，每一層印 **N-1, N-2, ..., 0** 個
4. 設計第二層迴圈要列印的資料，'#'




```
def myPrint1(N):  
    for x in range(N):  
        for y in range(N-x-1):  
            print('#', end='')  
        print()
```

第二種圖形

1. 規則，N層，**1, 13, 135, 1357, ...**
2. 設計第一層迴圈，圖形有N層
3. 設計第二層迴圈，每一層印 **1, 2, ..., N**
4. 設計第二層迴圈要列印的資料，1 開頭，每隔 2


把兩種圖形整合

1. **外層迴圈**要一致
2. 依序放 **內層迴圈**



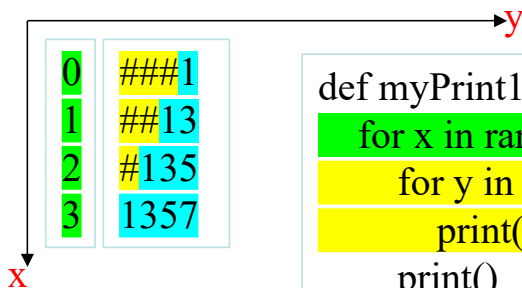
```
def myPrint1(N):  
    for x in range(N):  
        for y in range(N-x-1):  
            print('#', end='')  
        for y in range(x+1):  
            print(y*2+1, end='')  
        print()
```

```
def myPrint1(N):  
    for x in range(N):  
        for y in range(x+1):  
            print(y*2+1, end='')  
        print()
```



for 巢狀(nest)迴圈

□ 迴圈內還有迴圈，分成兩個function

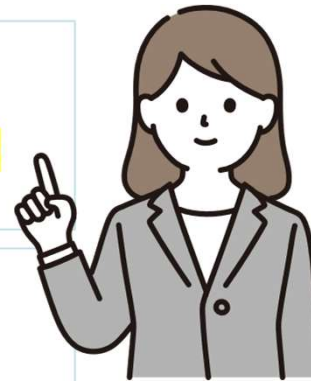


0	###1
1	##13
2	#135
3	1357

```
def myPrint1(N):  
    for x in range(N):  
        for y in range(N-x-1):  
            print('#', end='')  
        print()
```

```
def myPrint1(N):  
    for x in range(N):  
        PrintMark01(N-x-1)  
        print()
```

```
def PrintMark01(Z):  
    for y in range(Z):  
        print('#', end='')
```



1. 把第二個迴圈抽出成一個 function
2. 第一個迴圈呼叫第二個迴圈做成的function
3. 提供正確的呼叫參數 $N-x-1$

```
def myPrint1(N):  
    for x in range(N):  
        for y in range(x+1):  
            print(y*2+1, end='')  
        print()
```

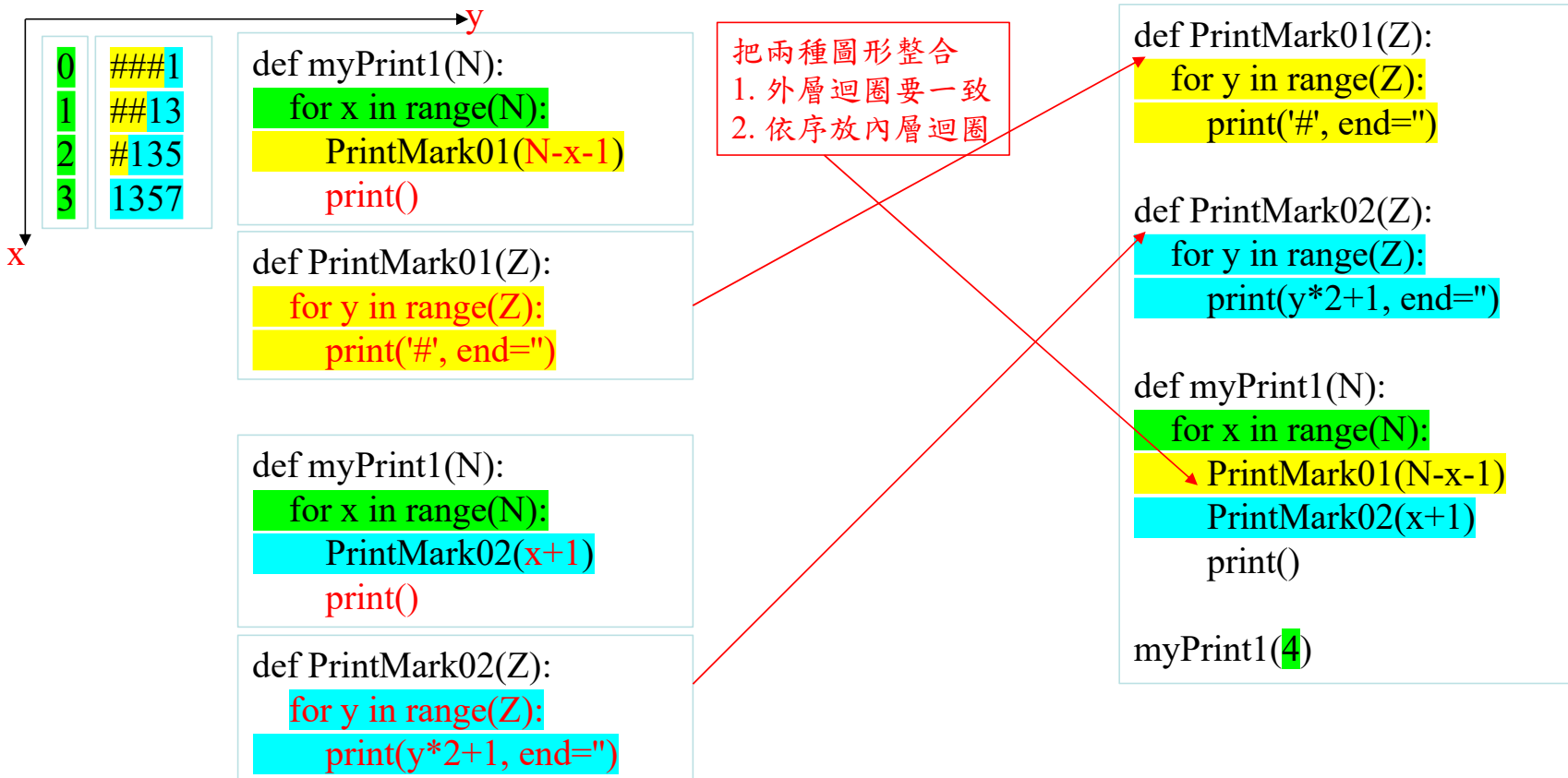
```
def myPrint1(N):  
    for x in range(N):  
        PrintMark02(x+1)  
        print()
```

```
def PrintMark02(Z):  
    for y in range(Z):  
        print(y*2+1, end='')
```

1. 把第二個迴圈抽出成一個 function
2. 第一個迴圈呼叫第二個迴圈做成的function
3. 提供正確的呼叫參數 $x+1$

for 巢狀(nest)迴圈

□ 迴圈內還有迴圈，印出數字金字塔，分成兩個function



for 巢狀(nest)迴圈

□ 把兩個 function 整合成一個

Diagram illustrating the initial state with two separate functions and a character.

Character: A girl wearing a yellow hard hat and a pink shirt, carrying a yellow bag.

Code Snippets:

```
def PrintMark01(Z):  
    for y in range(Z):  
        print('#', end='')  
  
def PrintMark02(Z):  
    for y in range(Z):  
        print(y*2+1, end='')  
  
def myPrint1(N):  
    for x in range(N):  
        PrintMark01(N-x-1)  
        PrintMark02(x+1)  
    print()  
  
myPrint1(4)
```

Output for myPrint1(4):

x	PrintMark01(N-x-1)	PrintMark02(x+1)
0	###1	
1	##13	
2	#135	
3	1357	

多一個變數，控制列印哪一種圖形

Code Snippet for PrintMark:

```
def PrintMark(Z, mark):  
    for y in range(Z):  
        if mark.isdigit():  
            print(y*2+1, end='')  
        else:  
            print(mark, end='')
```

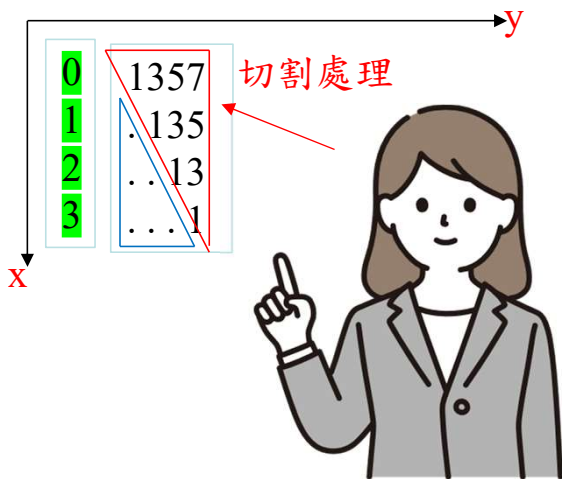
Code Snippet for myPrint1:

```
def myPrint1(N):  
    for x in range(N):  
        PrintMark(N-x-1, '#')  
        PrintMark(x+1, '0')  
    print()  
  
myPrint1(4)
```



for - range

□ 要印出



找出規則(規律)

1. 先出現符號. 再出現數字

2. 符號部分

(1) 一開始0個, 每行多一個, 共n行

3. 數字部分

(1) 從1開始, 間隔2

(2) 一開始n個, 每行少一個, 共n行

寫成程式

```
def printOneRow(m, n, s, mark):  
    for y in range(m, n, s):  
        if mark.isdigit():  
            print(y, end="")  
        else:  
            print(mark, end="")
```

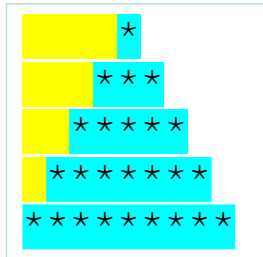
```
def myPrint1(n):  
    for x in range(0, n):  
        printOneRow(0, x, 1, '!')  
        printOneRow(1, 2*(n-x)+1, 2, '0')  
        print()  
  
myPrint1(4)
```

Exercise 1

□ 將下面 code 寫成二個 function，如何寫？

- 每一個 function 只能有一層迴圈 【將兩層迴圈改成一層迴圈】
- 其中有一個 function，其迴圈內呼叫另一個 function

```
def printGold(num):  
    for x in range(1, num+1):  
        for y in range(num, x, -1):  
            print(' ',end="")  
        for y in range(0, 2*x-1, 1):  
            print('*',end="")  
        print()  
  
def main():  
    num = 5  
    printGold(num)
```



```
def printOneRow(m, n, s, mark):  
    for y in range(m, n, s):  
        print(mark, end="")
```

```
def printGold(num):  
    for x in range(1, num+1):  
        printOneRow(num, x, -1, ' ') # 空格  
        printOneRow(0, 2*x-1, 1, '*') # 星星  
        print()
```

```
def main():  
    num = 5  
    printGold(num)
```

```
main()
```

解答

Exercise 2

- 將 Code 寫成 二個 function，每一個 function 使用 一層迴圈
- 寫出以下 code 的 output?

```
def printOneRow(m, n, s, mark):  
    for y in range(m, n, s):  
        if mark.isdigit():  
            print(y, end="")  
        else:  
            print(mark, end="")  
  
def myPrint1(n):  
    for x in range(0, n):  
        printOneRow(0, x, 1, '#')  
        printOneRow(2*(n-x-1), -1, -2, '0')  
        printOneRow(2, 2*(n-x), 2, '0')  
        print()  
  
myPrint1(4)
```

解答

```
6420246  
#42024  
##202  
###0
```

Exercise 3

□ 將 Code 寫成 二個 function，每一個 function 使用 一層迴圈

○ code?

1 12 123 1234 12345	54321 4321 321 21 1	1 22 333 4444 55555	1 121 12321 1234321 123454321	<u> 1 </u> <u> 212 </u> <u>32123 </u> 4321234	4321234 <u> 32123 </u> <u> 212 </u> <u> 1 </u>
---------------------------------	---------------------------------	---------------------------------	---	--	---

Exercise 4

□ 使用一個 **for loop** + **if** 印出 $n \times n$ 數字

```
def printSquare(n):  
    for i in range(1, n*n+1):  
        print('%3d'%i, end="")  
        if i%n == 0:  
            print()
```

解答

```
printSquare(4)
```

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

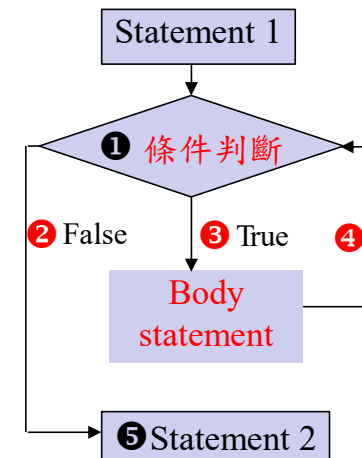
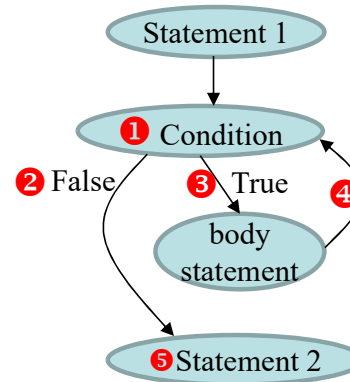
while

❑ 電腦程式常常需要重複執行工作任務，for, while

- 1. 執行條件 **Condition** 判斷，
- 2. **False** 不執行 body 指令，結束 loop (跳出 loop)
 - 繼續執行 while 後面指令 (5. Statement 2)
- 3. **True** 繼續執行 body 指令
- 4. 上面 3. 執行後 **回到 1.**
 - loop

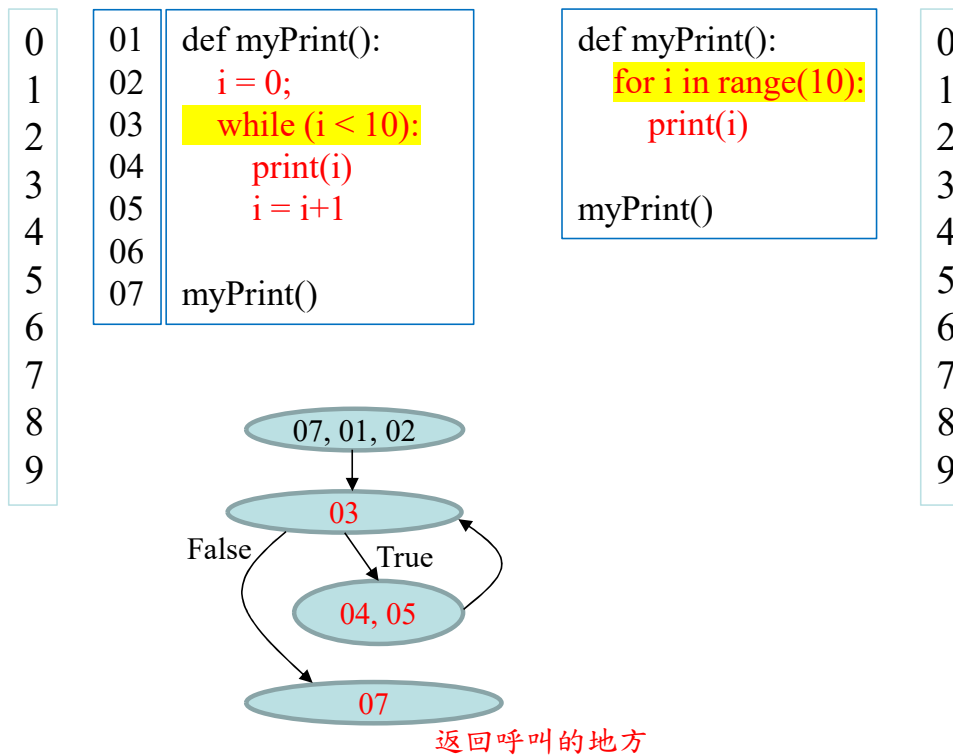
❑ while 適用於 **loop 圈數未知**

01	Statement 1
02	while Condition :
03	Body Statement
04	Statement 2



while

□ for in, while 對應



while

❑ 火箭發射 (blast off) 倒數計時 (countdown)

- 當不為 0 或 -1，繼續執行 loop
- 當倒數計時 **n 到 0** 時發射 (blast off)

```
10
9
8
7
6
5
4
3
2
1
blast off
```

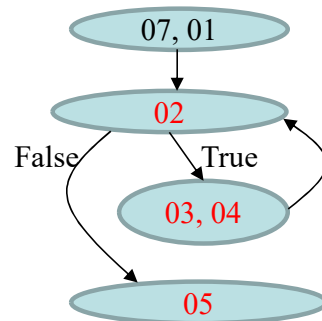
```
01 def countdown(n):
02     while (n>0):
03         print(n)
04         n = n-1
05     print('blast off')
06
07 countdown(10)
```

```
def countdown(n):
    for i in range(n,0,-1):
        print(i)

    print('blast off')

countdown(10)
```

```
10
9
8
7
6
5
4
3
2
1
blast off
```



while

□ While 適用於 loop 圈數未知

- 也可能造成 infinite loop

□ 此程式無法確認執行幾次

- $N \neq 1$ 持續執行 loop

- 印出 n
- N 是偶數， $n / 2$
- N 是奇數， $n * 3 + 1$

- 若 `sequence(3)`

- 輸出 3, 10, 5, 16, 8, 4, 2, 1

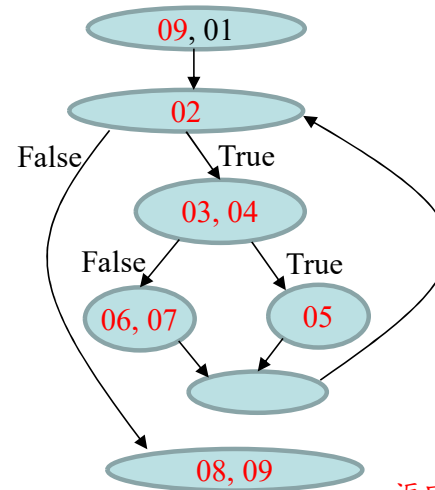
- n 有時增加，有時減少

- 無法確認 n 何時等於 1

```
01 def sequence(n):
02     while n != 1:
03         print(n)
04         if n % 2 == 0:      # n is even
05             n = n // 2
06         else:               # n is odd
07             n = n * 3 + 1
08     print(n)
09     sequence(3)
```

3
10
5
16
8
4
2
1

離散數學課本中的一個定理
當 n 為正整數，
若 n 為偶數，取半；
若 n 為奇數，乘上3後加1；
則最後會變成 1。



返回呼叫的地方

while

- 程式邏輯錯誤造成 infinite loop

```
01 def countdown(n):  
02     while (n > 0):  
03         print(n)  
04         n = n + 1  
05         print('blast off')  
06  
07     countdown(10)
```

- sequence(?)會造成 infinite loop?

$n \leq 0$

```
01 def sequence(n):  
02     while n != 1:  
03         print(n)  
04         if n % 2 == 0:      # n is even  
05             n = n / 2  
06         else:              # n is odd  
07             n = n*3 + 1  
08         print(n)  
09     sequence(?)
```

while

□ 正序印出 hi, python.

h
i
,

p
y
t
h
o
n
.

```
def myPrint01():  
    tmp = "hi, python."  
    i = 0  
    while(i < len(tmp)):  
        print(tmp[i])  
        i = i + 1
```

```
def myPrint02():  
    tmp = "hi, python."  
    i = 0  
    for c in tmp:  
        print(c)
```

h
i
,

p
y
t
h
o
n
.

編號每一行程式
劃出流程圖
寫下執行編號順序
寫下輸出內容

while

□ 正序印出 hi, python.

```
# 處理特別的字串
def myPrint03():
    tmp = "hi, python."
    i = 0
    while (i < len(tmp)):
        if tmp[i] == 'h':
            print('###')
        else:
            print(tmp[i])
        i = i + 1
```

```
###
i
,

p
y
t
###
o
n
.
```

編號每一行程式
劃出流程圖
寫下執行編號順序
寫下輸出內容

while

```
def mySum(m, n):  
    i = m  
    sumValue = 0  
    while (i <= n):  
        sumValue = sumValue + i  
        i = i + 1  
    print(sumValue)  
    return sumValue
```

```
def main():  
    minValue = int(input("Input a min number: "))  
    maxValuE = int(input("Input a max number: "))  
    main_sum = mySum(minValue, maxValuE)  
    print('sum (%d ~ %d) = %d' %(minValue, maxValuE, main_sum))  
main()
```

編號每一行程式
劃出流程圖
寫下執行編號順序
寫下輸出內容

```
Input a min number: 1  
Input a max number: 6  
21  
sum (1 ~ 6) = 21
```

break

❑ 使用 break 時機

- 當 loop 結束的時機無法在 loop 一開始決定
- 在 loop 的 body 內任一個地方，執行到某一條件/情境觸發，結束 loop
- 使用 break 指令跳出循 loop

❑ 例如，從 user 取得輸入，直到輸入 "done"

- 條件判斷 True，永遠為 True
- 若 user 輸入 'done'，break 跳出 loop
- 若 user 輸入其他資料，程式回到 loop 最開始

❑ 在 loop 的 body 內任一個地方

- 某一停止條件成立，break 跳出 loop

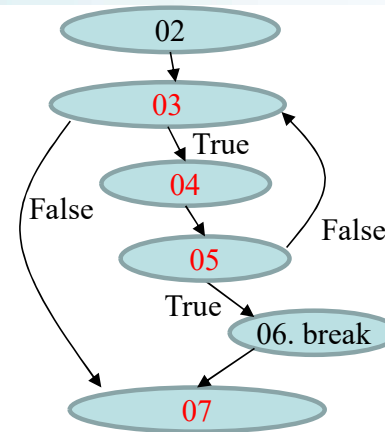
```
01 #當輸入"done"就離開 loop
02 def testBreak():
03     while True:
04         line = input()
05         if line == 'done':
06             break
07         print('line')
08     print('Finsh!')
```

```
> hello
hello
> done
Finish!
```

break

```
01 #當 i 數到 5 時就不做
02 def test01():
03     for i in range(1,10):
04         print(i)
05         if (i == 5):
06             break
07         print('end')
```

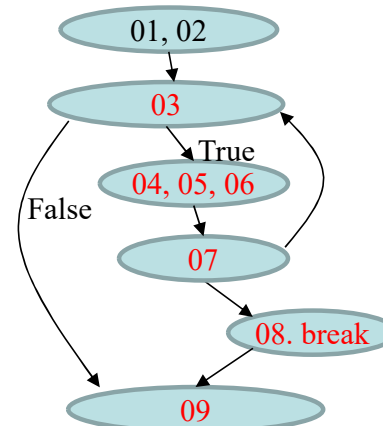
```
1
2
3
4
5
end
```



break

```
01 def test02():  
02     sum = 0  
03     while (True):  
04         inputOrder = int(input('Input: '))  
05         sum = sum + i  
06         print(inputOrder, sum)  
07         if (inputOrder == -1): # 當輸入 -1 就跳出  
08             break  
09     print('end')
```

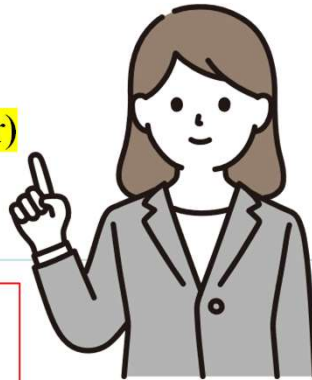
利用 **break** 在任何時候跳出迴圈



```
Input: 6  
6 6  
Input: 9  
9 15  
Input: -1  
-1 14  
end
```

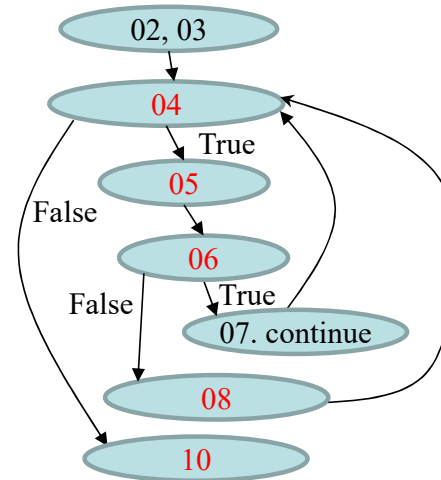
continue

```
01 #當number 沒超過 20 不印@，超過印@
02 def test03():
03     number = 0
04     for i in range(1,10):
05         number = number + i
06         if (number < 20):
07             continue
08         print('@', i, number)
09
10     print(i, number)
```



利用 continue 在任何時候
略過本次迴圈剩餘的運算

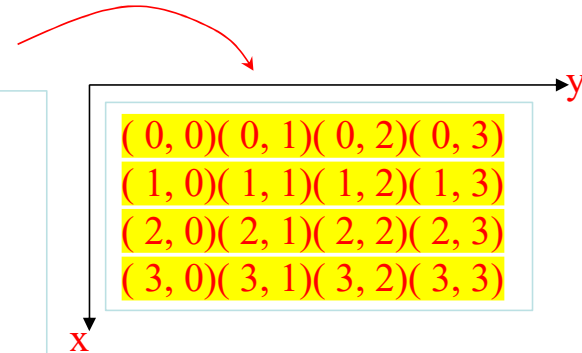
```
@ 6 21
@ 7 28
@ 8 36
@ 9 45
9 45
```



Exercise 1

□ 使用一個 loop + if 印出 $n \times n$ 座標

```
def printSquare(n):  
    x = y = 0  
    while x < n:  
        print('%2d,%2d' % (x, y), end="")  
        y = y + 1  
        if y >= n:  
            y = 0  
            x = x + 1  
        print()
```



```
(0,0)(0,1)(0,2)(0,3)  
(1,0)(1,1)(1,2)(1,3)  
(2,0)(2,1)(2,2)(2,3)  
(3,0)(3,1)(3,2)(3,3)
```

```
def printSquare(n):  
    i = 0  
    while i < n * n:  
        print('%2d,%2d' % (i//n, i%n), end="")  
        i = i + 1  
        if i%n == 0:  
            print()
```

將 i 轉成 x, y
 $x = i // n$
 $y = i \% n$
 $i = x * n + y$

i 值的變化			
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Exercise 2

- 使用一個 loop + if 印出 $n \times n$ 座標

```
def printTriangle(n):  
    x = y = 0  
    while x < n:  
        print('%2d,%2d' %(x, y), end="")  
        y = y + 1  
        if y > x:  
            y = 0  
            x = x + 1  
        print()
```

```
( 0, 0)  
( 1, 0) ( 1, 1)  
( 2, 0) ( 2, 1) ( 2, 2)  
( 3, 0) ( 3, 1) ( 3, 2) ( 3, 3)
```

Exercise 3

□ 使用一個 loop + if 印出 $0 \sim n \times n - 1$ 數字

```
# 0 <= x <= 3
# 0 <= y <= 3
def printSquare1(n):
    x = y = 0
    while x < n:
        print('%3d' % (x*n + y), end='')
        y = y + 1
        if y >= n:
            y = 0
            x = x + 1
        print()
    printSquare1(4)
```

將 x, y 轉成 i 也就是 $(x*n + y)$

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

將 i 轉成 x, y

```
x = i // n
y = i % n
```

將 x, y 轉成 i 也就是 $(x*n + y)$

```
i = x*n + y
```


x y

(0, 0)	(0, 1)	(0, 2)	(0, 3)
(1, 0)	(1, 1)	(1, 2)	(1, 3)
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

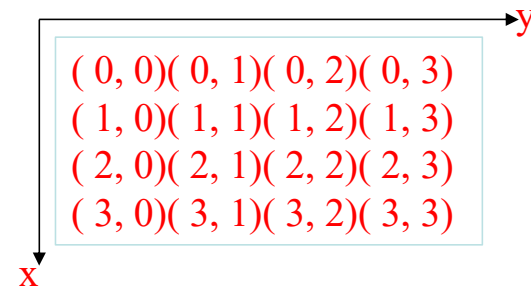
Exercise 4

□ 使用一個 loop + if 印出 $0 \sim n \times n - 1$ 數字

```
def printSquare2(n):  
    i = 0  
    while i < n*n:  
        print('%3d' % i, end="")  
        i = i + 1  
        if i % n == 0:  
            print()
```



i				
0	1	2	3	
4	5	6	7	
8	9	10	11	
12	13	14	15	



(0, 0)	(0, 1)	(0, 2)	(0, 3)
(1, 0)	(1, 1)	(1, 2)	(1, 3)
(2, 0)	(2, 1)	(2, 2)	(2, 3)
(3, 0)	(3, 1)	(3, 2)	(3, 3)

Exercise 5

□ 使用一個for loop + if 印出 $0 \sim n \times n - 1$ 數字與上下翻轉數字

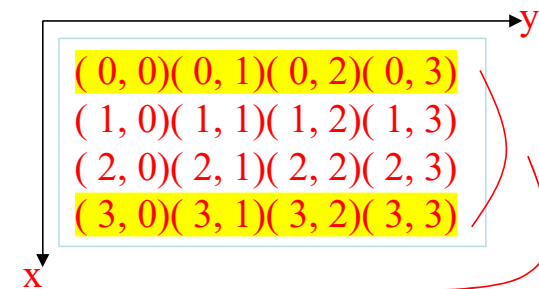
```
def t(i, n):  
    x = i//n  
    y = i%n  
    x = n-x-1  
    return (x*n + y)
```

```
def printSquare1(n):  
    i = 0  
    while i < n*n:  
        print('%3d' %i, end="")  
        i = i + 1  
        if i%n==0:  
            print()
```

```
def printSquare2(n):  
    i = 0  
    while i < n*n:  
        print('%3d' %t(i,n), end="")  
        i = i + 1  
        if i%n==0:  
            print()
```

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3



$x_1y_1 \rightarrow x_2y_2$
0 0 \rightarrow 3 0
0 1 \rightarrow 3 1
0 2 \rightarrow 3 2
0 3 \rightarrow 3 3

1 0 \rightarrow 2 0
1 1 \rightarrow 2 1
1 2 \rightarrow 2 2
1 3 \rightarrow 2 3

$x_2 = 3 - x_1$
 $y_2 = y_1$

$x_2 = 3 - x_1$
 $y_2 = y_1$

$x_2 = 4 - 1 - x_1$
 $x_2 = n - 1 - x_1$

Exercise 6

□ 使用一個for loop + if 印出 0 ~ n×n-1 數字與順時鐘翻轉數字

```
def t(i, n):
    x1 = i//n
    y1 = i%n
    y2 = x1
    x2 = n-y1-1
    return (x2*n+y2)
```

```
def printSquare1(n):
    i = 0
    while i<n*n:
        print('%3d' %i, end=" ")
        i = i + 1
        if i%n==0:
            print()
```

```
def printSquare2(n):
    i = 0
    while i<n*n:
        print('%3d' %t(i,n), end=" ")
        i = i + 1
        if i%n==0:
            print()
```

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

12	8	4	0
13	9	5	1
14	10	6	2
15	11	7	3

$(0,0)(0,1)(0,2)(0,3)$
 $(1,0)(1,1)(1,2)(1,3)$
 $(2,0)(2,1)(2,2)(2,3)$
 $(3,0)(3,1)(3,2)(3,3)$

$(0,0)(0,1)(0,2)(0,3)$
 $(1,0)(1,1)(1,2)(1,3)$
 $(2,0)(2,1)(2,2)(2,3)$
 $(3,0)(3,1)(3,2)(3,3)$

$x_1y_1 \rightarrow x_2y_2$
 $00 \rightarrow 30$
 $01 \rightarrow 20$
 $02 \rightarrow 10$
 $03 \rightarrow 00$

$10 \rightarrow 31$
 $11 \rightarrow 21$
 $12 \rightarrow 11$
 $13 \rightarrow 01$

0 換成 12
 $y_2 = x_1$
 $x_2 = 3-y_1$

$y_2 = x_1$
 $x_2 = 3-y_1$
 $x_2 = 4-1-y_1$
 $x_2 = n-1-y_1$

Exercise 7

解答: 所有 row 中，
是否有連續四個 'w'

- 針對 N 個 list，
- 每一個元素是一個 N 個 'b' 或 'w', 或 'e' 的 list。
- 例如 N = 5

```
grid = [['b', '.', 'w', 'b', 'w'],
        ['w', 'b', 'w', 'b', '.'],
        ['w', '.', 'w', 'w', 'w'],
        ['.', 'w', 'w', 'w', 'w'],
        ['.', 'b', 'w', 'b', 'w']]
```

00	01	02	03	04
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34

- 計算所有 column 中，是否有 4 個 'b'
- 計算所有 row 中，是否有 4 個 'b'
- 計算所有 column 中，是否有連續 4 個 'b'
- 計算所有 row 中，是否有連續 4 個 'b'
- 計算所有斜對角 中，是否有 4 個 'b'
- 計算所有斜對角 中，是否有連續 4 個 'b'

```
2 ['w', '.', 'w', 'w', 'w']
3 ['.', 'w', 'w', 'w', 'w']
cont => ['.', 'w', 'w', 'w', 'w']
```

```
class Const: SIZE = 5
```

```
def isSize(grid, mark, N):
    for i in range(Const.SIZE):
        if grid[i].count(mark) == N:
            print(i, grid[i])
```

```
def isCountinus(data, mark, N):
    num = 0
    for i in range(Const.SIZE):
        if data[i] == mark: num += 1
        if num == N: return True
        if data[i] != mark: num = 0
    return False
```

```
def compute():
    grid = [['b', '.', 'w', 'b', 'w'],
            ['w', 'b', 'w', 'b', '.'],
            ['w', '.', 'w', 'w', 'w'],
            ['.', 'w', 'w', 'w', 'w'],
            ['.', 'b', 'w', 'b', 'w']]
    isSize(grid, 'w', 4)
    for i in range(Const.SIZE):
        if isCountinus(grid[i], 'w', 4) == True:
            print('cont=>', grid[i])
```

```
compute()
```

Exercise 8

□ 使用 while loop + if 印出

```
def printTriangle1(n):  
    star=0  
    level=1  
    while level<=n:  
        print('*',end="")  
        star = star + 1  
        if star>=level:  
            star = 0  
            level = level + 1  
        print()
```

```
*  
**  
***  
****  
*****
```

```
1  
1 2  
1 2 3  
1 2 3 4  
1 2 3 4 5
```

```
def printTriangle3(n):  
    star=1  
    level=1  
    while level<=n:  
        print('%2d' %star,end="")  
        star = star + 1  
        if star>level:  
            star = 1  
            level = level + 1  
        print()
```

Exercise 9

□ 使用 while loop + if 印出

```
def printOneRow(m, n, s, mark):
    i = m
    if (m==n): return
    while (i!=n):
        if (mark=='0'): print("%d" %i, end="")
        else: print("%c" %mark, end="")
        i = i + s

def myPrint(n):
    for i in range(n):
        printOneRow(i, 0, -1 , '#')
        printOneRow(2*(n-i-1), -2, -2 , '0')
        printOneRow(2, 2*(n-i), 2 , '0')
        printOneRow(i, 0, -1 , '#')
        print()

myPrint(4)
```

```
6420246
#42024#
##202##
###0###
```

myPrint(4)

```
864202468
#6420246#
###42024##
####202###
#####0#####
```

myPrint(5)

Exercise 10

□ 使用 while loop + if continue 印出

```
def printTriangle2(n):  
    star = 0  
    level = n  
    while level > 0:  
        print('*',end="")  
        star = star + 1  
        if star < level:  
            continue  
        star = 0  
        level = level - 1  
        print()
```

```
*****  
****  
***  
**  
*
```

Exercise 11 輾轉相除法

□ 計算最大公因數

$x = 42$
 $y = 75$

② → 1	42	75	1 ← ①
	33	42	
④ → 1	9	33	3 ← ③
	6	27	
	3	6	2 ← ⑤
		6	
		0	

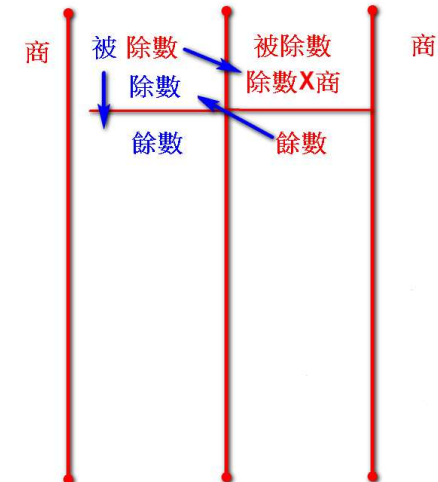
GCD

1. 以較大數 (75) 為被除數，較小數 (42) 為除數， $75 / 42 = 1$ 餘 33
2. 以前一步驟的除數為被除數，餘數為除數， $42 / 33 = 1$ 餘 9
3. $33 / 9 = 3$ 餘 6
4. $9 / 6 = 1$ 餘 3
5. $6 / 3 = 2$ 餘 0，除數 3 為最大公因數

```
def gcd(x, y):
    while (x>0) and (y>0):
        if (x>y):
            x = x%y
        else:
            y = y%x
    return (x if x>y else y)
```

```
print(gcd(18, 24))
print(gcd(90, 36))
```

1	123	321	2
	75	246	
1	48	75	1
	27	48	
3	21	27	1
	18	21	
	3	6	2
		6	
		0	



Exercise 12

□ 計算BMI值的function

- BMI值計算公式: $BMI = \text{體重(公斤)} / \text{身高}^2(\text{公尺}^2)$
- 例如：一個52公斤的人，身高是155公分，則BMI為：
- $52(\text{公斤}) / 1.55^2(\text{公尺}^2) = 21.6$
- 正常範圍為 BMI=18.5~24
- 輸入身高、體重，輸出BMI值。
- 身高正常範圍 0.5~2.50 公尺，體重正常20~300 公斤，若輸入不在正常範圍，輸出 "Input Error (0.5~2.50)"/ "Input Error (20~300)"，請重新輸入。
- 若BMI值太高，輸出"BMI too high"，太低輸出"BMI too low"。
- 可以接受不斷輸入計算，直到輸入-1停止。
- 不會有身高與體重皆不正常之情況。

Sample input :

3 20

1.55 52

2.4 299

-1

Sample output :

Input Error 0.5~2.50

21.64

BMI too high

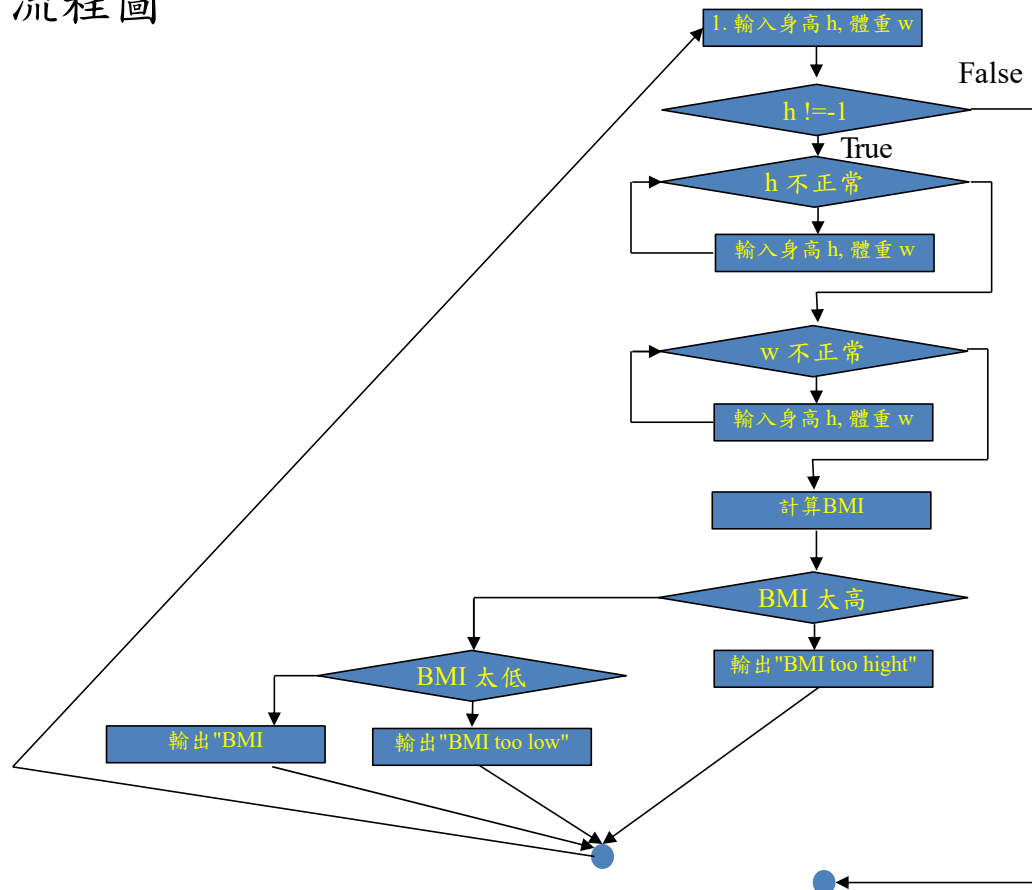
Exercise 12

□ 計算BMI值的流程說明

- 1. 輸入身高 h , 體重 w
- 2. 假如身高 $h == -1$ 停止程式
- 3. 假如身高不在正常範圍 $0.5 \sim 2.50$
 - 輸出 "Input Error (0.5~2.50)"
 - 輸入身高 h , 體重 w
- 4. 假如體重不在正常範圍 $20 \sim 300$
 - 輸出 "Input Error (20~300)"
 - 輸入身高 h , 體重 w
- 5. 計算 $BMI = \text{體重(公斤)} / \text{身高}^2(\text{公尺}^2)$
- 6. 假如 BMI 太高, 輸出 "BMI too high"
- 7. 假如 BMI 太低, 輸出 "BMI too low"。
- 8. 回到步驟 1

Exercise 12

□ 計算BMI值的流程圖



Exercise 12

```
def bmi_input():
    x=input()
    x=x.split()
    CM=float(x[0])
    KG=float(x[1])
    CM=float(input())
    KG=float(input())
    BMI=round((KG//CM**2),2)
    print(BMI)
    if BMI > 24:
        print('BMI too high')
    if BMI < 18.5:
        print('BMI too low')
    if BMI >=18.5 and BMI <=24:
        print(BMI)
    while(CM!=-1):
        break
    if CM>=0.5 and CM<=2.5:
        print('Input Error 0.5~2.5')
    if KG >=20 and KG<=300:
        print('Input Error 20~300')
    bmi_input()
```

錯在哪裡？

1. while 要在外圍
2. BMI 判斷要使用 if-elif-else
3. CM, KG判斷要往前

Exercise 12

這樣對嗎？

```
def check(CM, KG):
    if CM<0.5 or CM>2.5:
        print('Input Error 0.5~2.5')
        return 0
    if KG <20 or KG>300:
        print('Input Error 20~300')
        return 0
    return 1

def bmi_input():
    while(True):
        x=input()
        x=x.split()
        if x[0]=='-1':
            break
        CM=float(x[0])
        KG=float(x[1])
        if (check(CM, KG)==0):
            continue
        BMI=round((KG/CM**2),2)
        print(BMI)
        if BMI > 24:
            print('BMI too high')
        if BMI < 18.5:
            print('BMI too low')
        if BMI >=18.5 and BMI <=24:
            print(BMI)

bmi_input()
```

Exercise 12

```
def inputBMI():  
    x=input().split()  
    stop=0  
    if x[0]=='-1':  
        stop = -1  
    CM=float(x[0])  
    KG=float(x[1])  
    return stop, CM, KG
```

```
def check(CM, KG):  
    if CM<0.5 or CM>2.5:  
        print('Input Error 0.5~2.5')  
        return 0  
    if KG <20 or KG>300:  
        print('Input Error 20~300')  
        return 0  
    return 1
```

```
def output(CM,KG):  
    BMI=round((KG/CM**2),2)  
    print(BMI)  
    if BMI > 24:  
        print('BMI too high')  
    if BMI < 18.5:  
        print('BMI too low')  
    if BMI >=18.5 and BMI <=24:  
        print(BMI)
```

```
def computeBMI():  
    while(True):  
        stop, CM, KG = inputBMI()  
        if stop=='-1':  
            break  
        if (check(CM, KG)==0):  
            continue  
        output(CM, KG)
```

```
computeBMI()
```

跟上頁code差別在哪裡?
function 模組化
優點?

Exercise 13 撲克牌比大小

□ 撲克牌

- A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
- A~10 點數為 1~10，J, K, Q 為 0.5。

□ 電腦與玩家，各隨機發撲克牌，加總點數接近 10.5 則贏。

- 超過 10.5 爆掉分數為 0。

□ 程式

- 電腦隨機發X撲克牌，使用者可選擇要牌或不要牌。
- 電腦隨機發電腦撲克牌，電腦判斷是否停發牌。
- 輸出電腦與玩家的點數，以及電腦贏或玩家贏或平手。

Exercise 13 撲克牌比大小

□ 點數

- A~10 點數 1~10，J, K, Q 為 0.5。

□ 玩法

- 電腦與玩家各隨機發撲克牌，加總點數接近 10.5 則贏。
- 超過 10.5 爆掉分數為 0 且該方不得繼續要牌。
- 任一回合並未要牌的一方，失去要牌權利。
- 程式發一張撲克牌給玩家，玩家可選擇要牌或不要牌。
- 程式發一張撲克牌給電腦，電腦判斷是否停發牌。

□ 電腦判斷要牌：

- 1. 總點數比玩家小 或
- 2. 總點數 8 點以下(含)

□ 輸出電腦與玩家點數，電腦贏或玩家贏或平手(Tie)輸出：It's a tie)

Exercise 13 撲克牌比大小

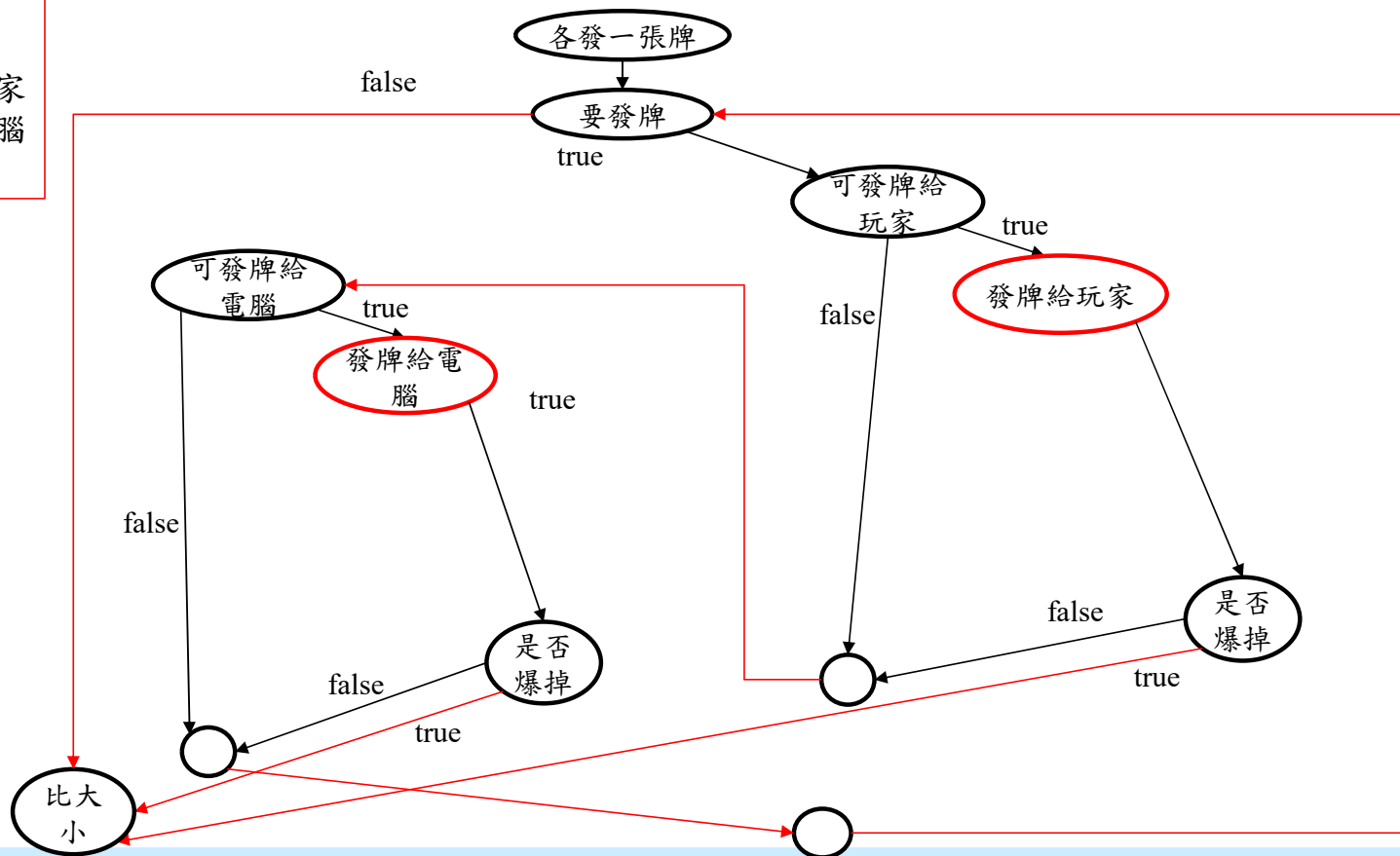
□ 輸入範例說明

- A 先發一張給給玩家(玩家獲得 A)
- J 再發一張給電腦(電腦獲得 J)
- Y 玩家選擇要牌
- 9 發一張給玩家(玩家獲得 9)
- 8 電腦牌面 0.5 點，未超過 8 點，再發一張給電腦(電腦獲得 8)
- N 玩家選擇不要牌
- 5 電腦牌面 8.5，低於玩家的 10，因此再抽(獲得 5)

Exercise 13 撲克牌比大小

流程

1. 各發一張牌
2. 發牌迴圈
 - 2.1. 發牌給玩家
 - 2.2. 發牌給電腦
3. 比大小



Exercise 13 撲克牌比大小

流程

1. 各發一張牌
2. 發牌迴圈
 - 2.1. 發牌給玩家
 - 2.2. 發牌給電腦
3. 比大小

發牌給(玩家/電腦)

1. 可發牌

1.1 是否發牌

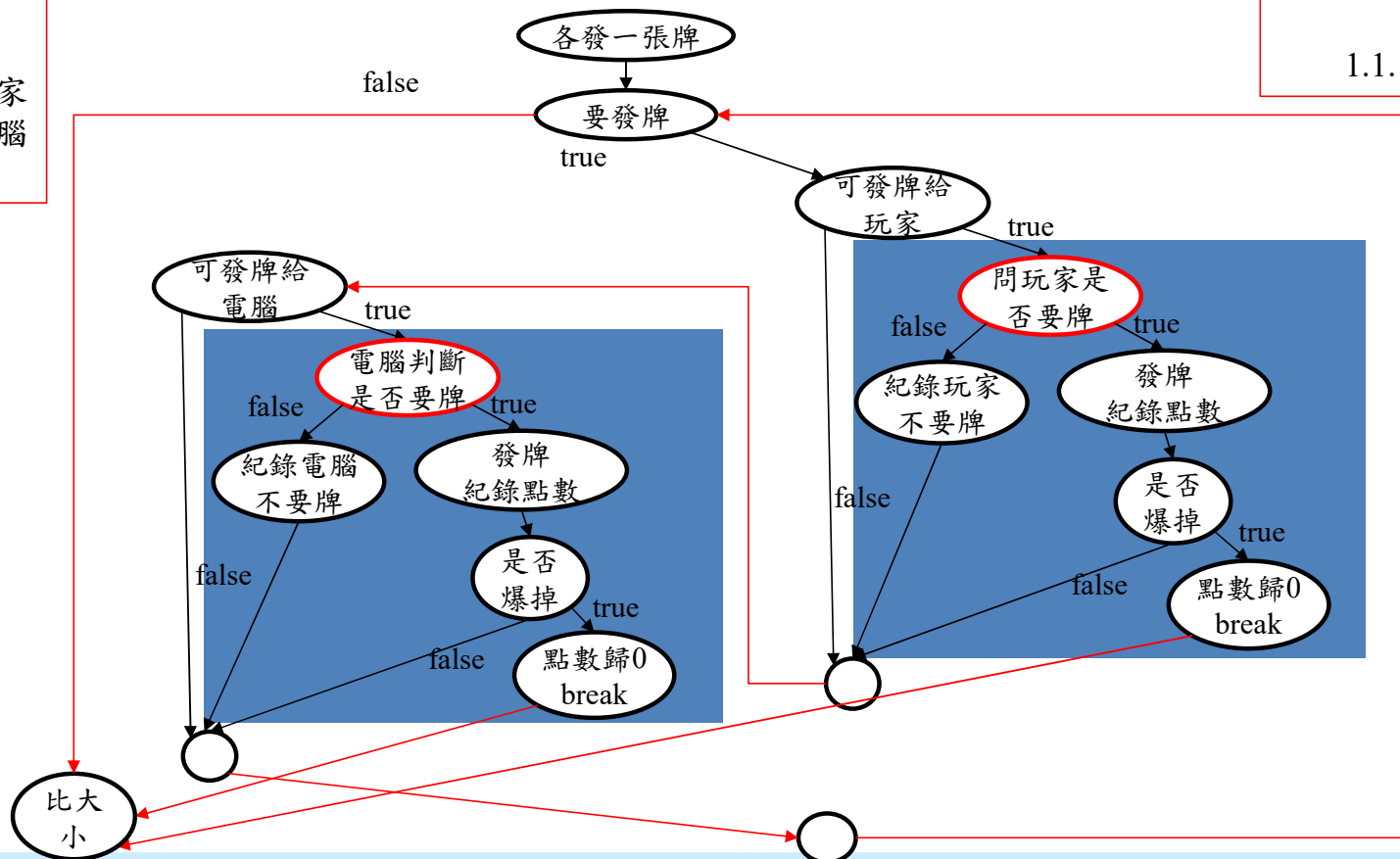
1.1.1 發牌、紀錄點數

1.1.1.1 是否爆掉

1.1.1.1.1 點數歸0

break

1.1.1 紀錄不要牌



Exercise 13 撲克牌比大小

流程

1. 各發一張牌
2. 發牌迴圈
 - 2.1. 發牌給玩家
 - 2.2. 判斷是否結束
 - 2.3. 發牌給電腦
 - 2.4. 判斷是否結束
3. 比大小

發牌給玩家 (deal, 參數- role, myPoint, bPoint, judge_func, 回傳值- role, point, over

1. 可發牌(設定變數 - role)

1.1 是否發牌(輸入變數-wantCard)

變成function判斷是否要牌

1.1.1. 發牌、紀錄點數(變數-myPoint)

1.1.1.1 是否爆掉(function-isExplode)

1.1.1.1.1 點數歸0 (myPoint=0) break (game over)

變成function調整點數

1.1.1 紀錄不要牌(role=false)

Exercise 13 撲克牌比大小

```
def playerJudge(a, b):  
    isWant=input()  
    if (isWant=='Y'):  
        return True  
    else:  
        return False
```

```
def computerJudge(a, b):  
    if (a<b):  
        return True  
    elif (a<8):  
        return True  
    else:  
        return False
```

```
def justPoint(point):  
    over = False  
    if (point>10.5):  
        point=0  
        over=True  
    return over, point
```

```
def deal(myPoint, bPoint, judge):  
    over = False  
    isWant=judge(myPoint, bPoint)  
    if (isWant==True):  
        myPoint=myPoint+transferPoint(input())  
        over, myPoint=justPoint(myPoint)  
    return isWant, myPoint, over
```

```
def game():  
    over = False  
    computer=player=True  
    playerPoint=transferPoint(input())  
    computerPoint=transferPoint(input())  
    while (player or computer):  
        if (player==True):  
            player, playerPoint, over=deal(playerPoint, computerPoint, playerJudge)  
        if (over==True):  
            break  
        if (computer==True):  
            computer, computerPoint, over=deal(computerPoint, playerPoint, computerJudge)  
        if (over==True):  
            break  
    print(playerPoint, computerPoint)
```

初始化與初始輸入

Exercise 13 撲克牌比大小

□ 遊戲規則修改

- 可以有**多位玩家**，電腦當莊家
- 玩家可以只在某一輪放棄要牌
- 電腦判斷是否**要牌**，要考慮**多位玩家**

Exercise 14

□ 猜數字，隨機(外部輸入)產生一個介於1~10的答案，使用者猜中則停止輸入，根據使用者輸入提示以下訊息：

- 1.猜太大
- 2.猜太小
- 3.猜中了

```
import random
def myFunction():
    ans = random.randint(1,10)
    while True:
        inputData = int(input("Guess 1~10: "))
        if (inputData == ans):
            print("Right")
            break

def main():
    myFunction()
```

for/while 的使用時機

- 需重複進行運算的時候使用迴圈(for/while)
 - 重複次數可清楚計算或當疊代明顯時，使用 for 迴圈
 - 重複次數難以計算，但條件清楚，或有條件的重複時，使用 while 迴圈

for/while 的使用時機

□ 重複結構 while 和 for 都支援 else 敘述

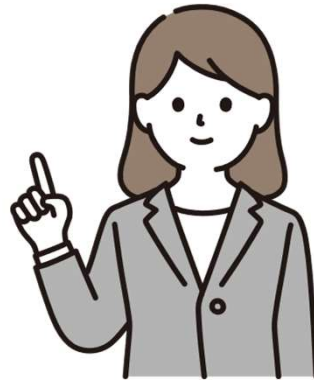
- 當迴圈 **非因為** break, return 或例外終止時，(正常中止)，else_suite 是 **會被執行**
- 當迴圈 **是因為** break, return 或例外終止時，(非正常中止)，else_suite 是 **不會被執行的！**

```
while Condition :  
    while_body  
else:  
    else_suite
```

```
for var in Condition :  
    for_body  
else:  
    else_suite
```

當迴圈因為
break 或 return
中斷時，

下面的 else 是
不會執行的！



```
chance = 5  
fruits = {"apple", "banana", "cherry", "durian"}
```

```
while chance:  
    ans = input('Guess the fruit: ')  
    print(f'Your guess is {ans}')
```

if ans in fruits:

```
    print("You win!")  
    break  
    chance -= 1  
else:  
    print("You lose!")
```

字串格式語法

```
person = 'Sean'  
print("My name is %s" %person)
```

```
person = 'Sean'  
print("My name is {}".format(person))
```

- ❑ f-strings 作為字串格式化的方式。
- ❑ f-Strings 又稱為「string interpolation」。

```
person = 'Sean'  
print(f"My name is {person}")  
# My name is Sean
```

END

