

---

# 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

(1,4);(1,3);(1,2);(1,1),  
(2,4)  
(3,4)  
(4,1)

# for 巢狀(nest)迴圈

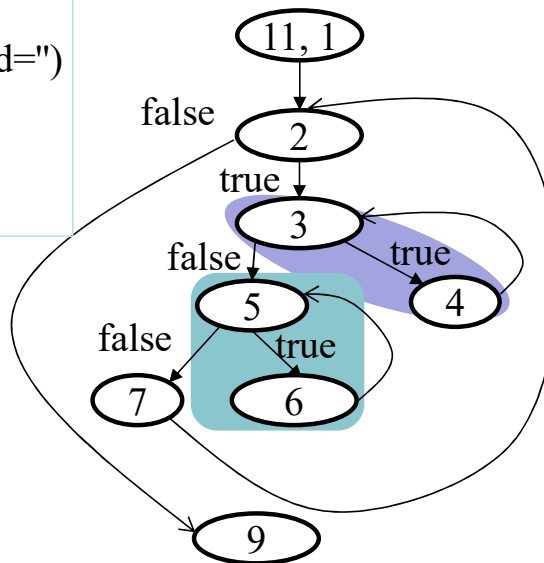
□ 迴圈內還有迴圈，變數的值如何改變?程式流程如何變化?

```
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)
```

```
(1,4);(1,3);(1,2);(1,1),  
(2,4);(2,3);(2,1),(2,2),(2,3),  
(3,4);(3,1),(3,2),(3,3),(3,4),(3,5),  
(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

劃出流程圖  
寫下執行編號順序  
寫下輸出內容



```
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

```
(1,4);(1,3);(1,2);(1,1),  
(2,4);(2,3);(2,1),(2,2),(2,3),  
(3,4);(3,1),(3,2),(3,3),(3,4),(3,5),  
(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

```
###1  
##13  
#135  
1357
```

1. 找出規則
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
2. 設計第二層迴圈，每一層印 N-1, N-2, ..., 0 個
3. 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
2. 設計第二層迴圈，每一層印 1, 2, ..., N
3. 設計第二層迴圈要列印的資料，1 開頭，每隔 2

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

### 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()
```

# for 巢狀(nest)迴圈

□ 迴圈內還有迴圈，分成兩個function

```
###1  
##13  
#135  
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 N-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

```
###1
##13
#135
1357
```

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

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

```
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="")
```

把兩種圖形整合  
1. 外層迴圈要一致  
2. 依序放內層迴圈

```
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)
```

## for 巢狀(nest)迴圈

□ 把兩個 function 整合成一個

```
###1  
##13  
#135  
1357
```

```
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)
```

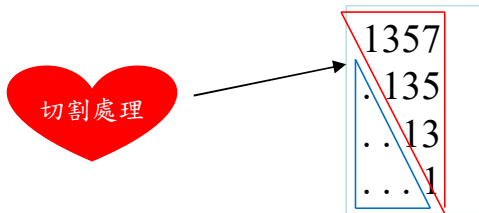
```
def PrintMark(Z, mark):  
    for y in range(Z):  
        if mark.isdigit():  
            print(y*2+1, end='')  
        else:  
            print(mark, end='')  
  
def myPrint1(N):  
    for x in range(N):  
        PrintMark(N-x-1, '#')  
        PrintMark(x+1, '0')  
        print()  
  
myPrint1(4)
```

多一個變數，控制列印哪一種圖形



# for - range

## □ 要印出



### 找出規則

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)
```

## for 巢狀(nest)迴圈

□ 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)
```

## for 巢狀(nest)迴圈

### ❑ 【將兩層迴圈改成一層迴圈】

```
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)
```

# Exercise

- 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)
```



```
6  
#  
#  
#
```

# Exercise

□ 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
```

```
  _1_
 _212_
_32123_
4321234
```

```
4321234
 _32123_
  _212_
   _1_
```

# Exercise

- 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()
```

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

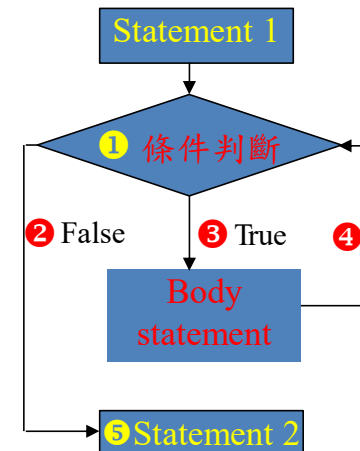
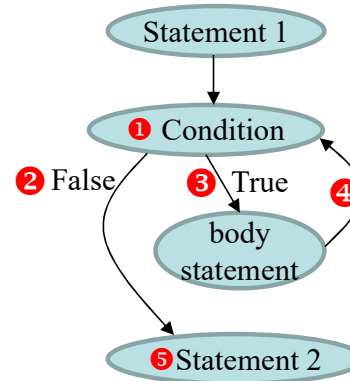
# while

❑ 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
```

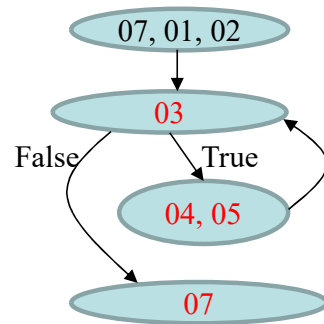


# while

## □ for in, while 對應

```
01 def myPrint():  
02     i = 0;  
03     while (i<10):  
04         print(i)  
05         i = i+1  
06  
07 myPrint()
```

```
def myPrint():  
    for i in range(10):  
        print(i)  
  
myPrint()
```





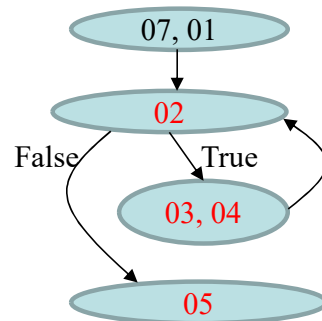
# while

## ❑ 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)

```
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)
```



# 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， $n*3+1$

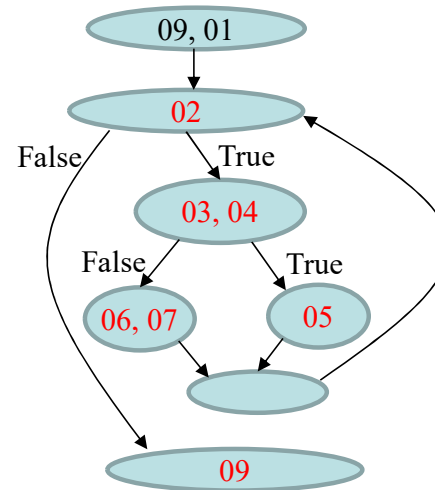
- 若 `sequence(3)`

- 輸出 3, 10, 5, 16, 8, 4, 2

- $n$  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
09 sequence(3)
```



# 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?

```
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  
09     sequence(?)
```

# while

## □ 正序印出 hi, python.

```
def myPrint01():  
    tmp = "hi, python."  
    print(tmp)  
    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)
```

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

```
#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
```

# 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

# 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 指令跳出循環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!')
```

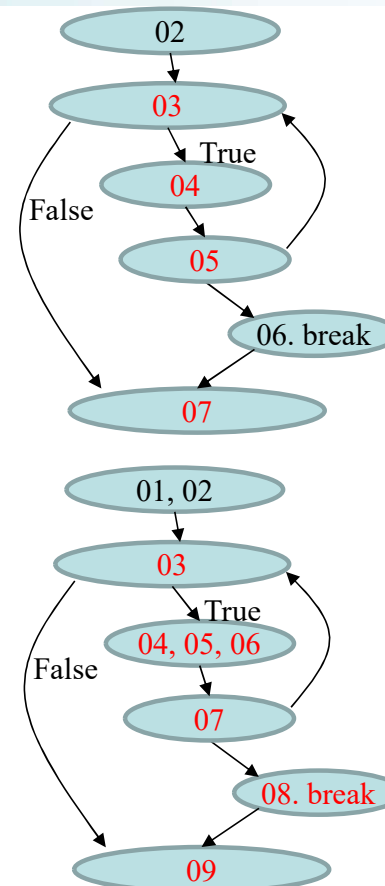
```
> hello
hello
> done
Finish!
```

# break

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

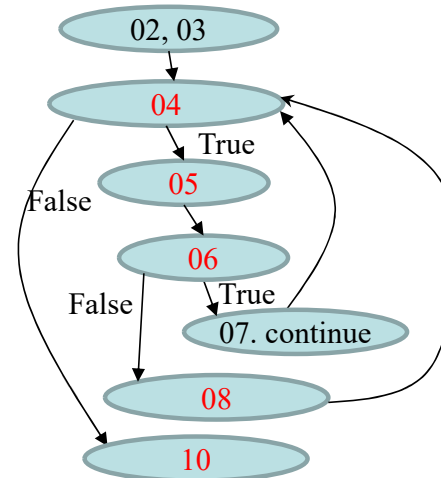
```
01 def test02():
02     sum = 0
03     while (True):
04         inputOrder =input()
05         sum = sum + i
06         print(inputOrder, sum)
07         if (inputOrder==-1):
08             break
09     print('end')
```

利用break在任何時候跳出迴圈



# 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

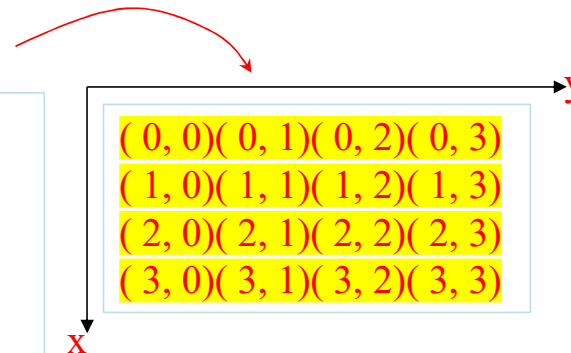


# Exercise

□ 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()
```

```
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

□ 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

□ 使用一個 loop + if 印出 0~n×n-1 數字

```
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()
```

將 x, y 轉成 i

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

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

```
i = x*n + 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)


x

y

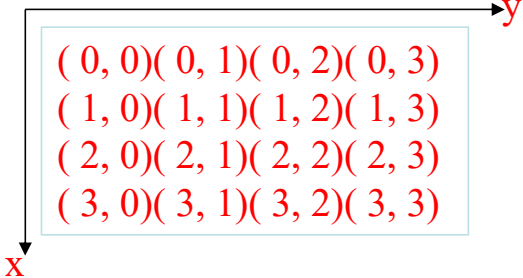
# Exercise

- 使用一個 loop + if 印出 0~n×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

□ 使用一個for loop + if 印出 0~n×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()
```

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

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

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

x ↓ y →

$x_1y_1-x_2y_2$
0 0 - 3 0
0 1 - 3 1
0 2 - 3 2
0 3 - 3 3

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

1 0 - 2 0
1 1 - 2 1
1 2 - 2 2
1 3 - 2 3

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

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

# Exercise

□ 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

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

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

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

$1\ 0-3\ 1$   
 $1\ 1-2\ 1$   
 $1\ 2-1\ 1$   
 $1\ 3-0\ 1$

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

□ 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'

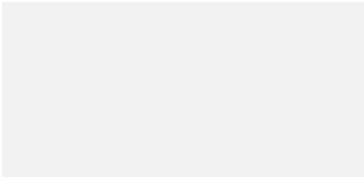
```
class Const: SIZE = 5  
  
def isSize(grid, mark, N):  
    for i in range(Const.SIZE):  
        if grid[i].count(mark) == N:  
            print(i, grid[i])  
  
def isCountinus(data, mark, N):  
    num = 0  
    for i in range(Const.SIZE):  
        if data[i] == mark: num += 1  
        if num == 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

## □ 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()
```

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

```
def printTriangle3(n):  
    star=1  
    level=1  
    while level<=n:  
          
        print()
```

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



# Exercise

## □ 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), '0', '0')  
        printOneRow(2, '0', '0')  
        printOneRow(i, 0, -1, '#')  
    print()
```

```
myPrint(4)
```

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

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

# Exercise

□ 使用 while loop + if continue 印出

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

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

# Exercise

## □ 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	

商 被除數 除數 被除數 除數 商

被除數 除數 被除數 除數 商

被除數 除數 被除數 除數 商

# Exercise

## □ 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
```

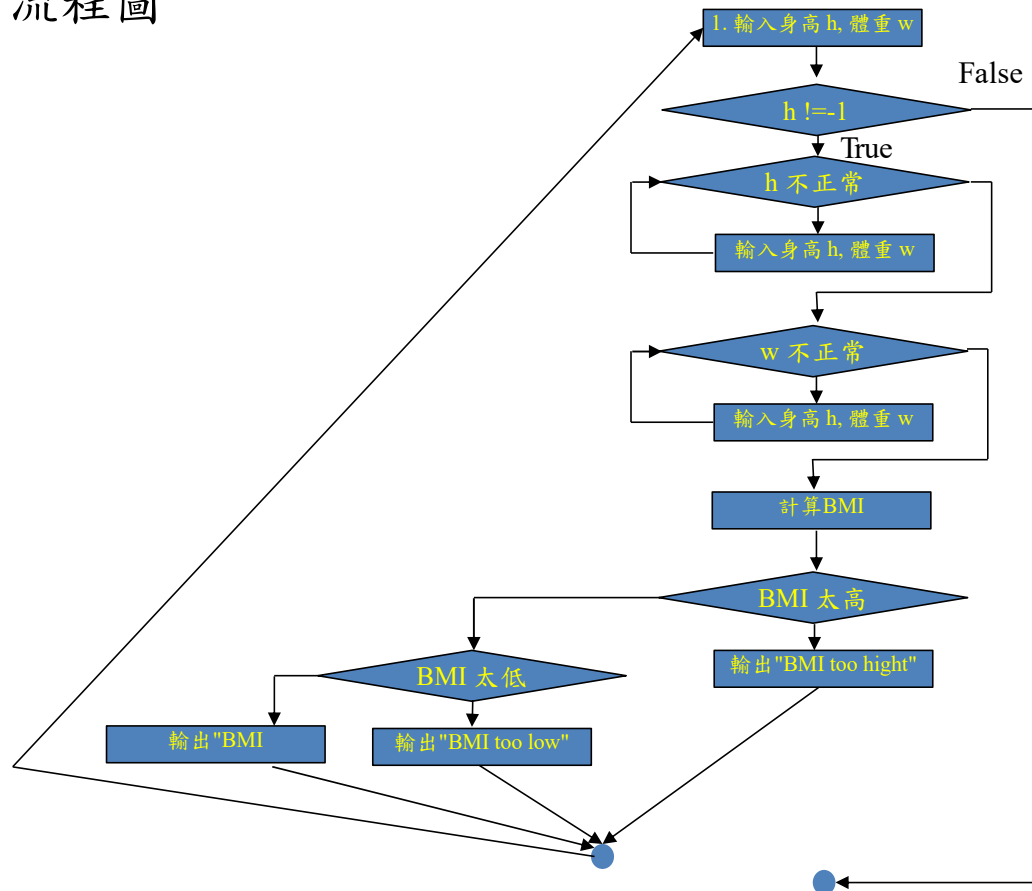
# Exercise

## □ 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

## □ 計算BMI值的流程圖



# Exercise

```
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.
- 2.
- 3.

# Exercise

```
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()
```

這樣對嗎?



# Exercise

```
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

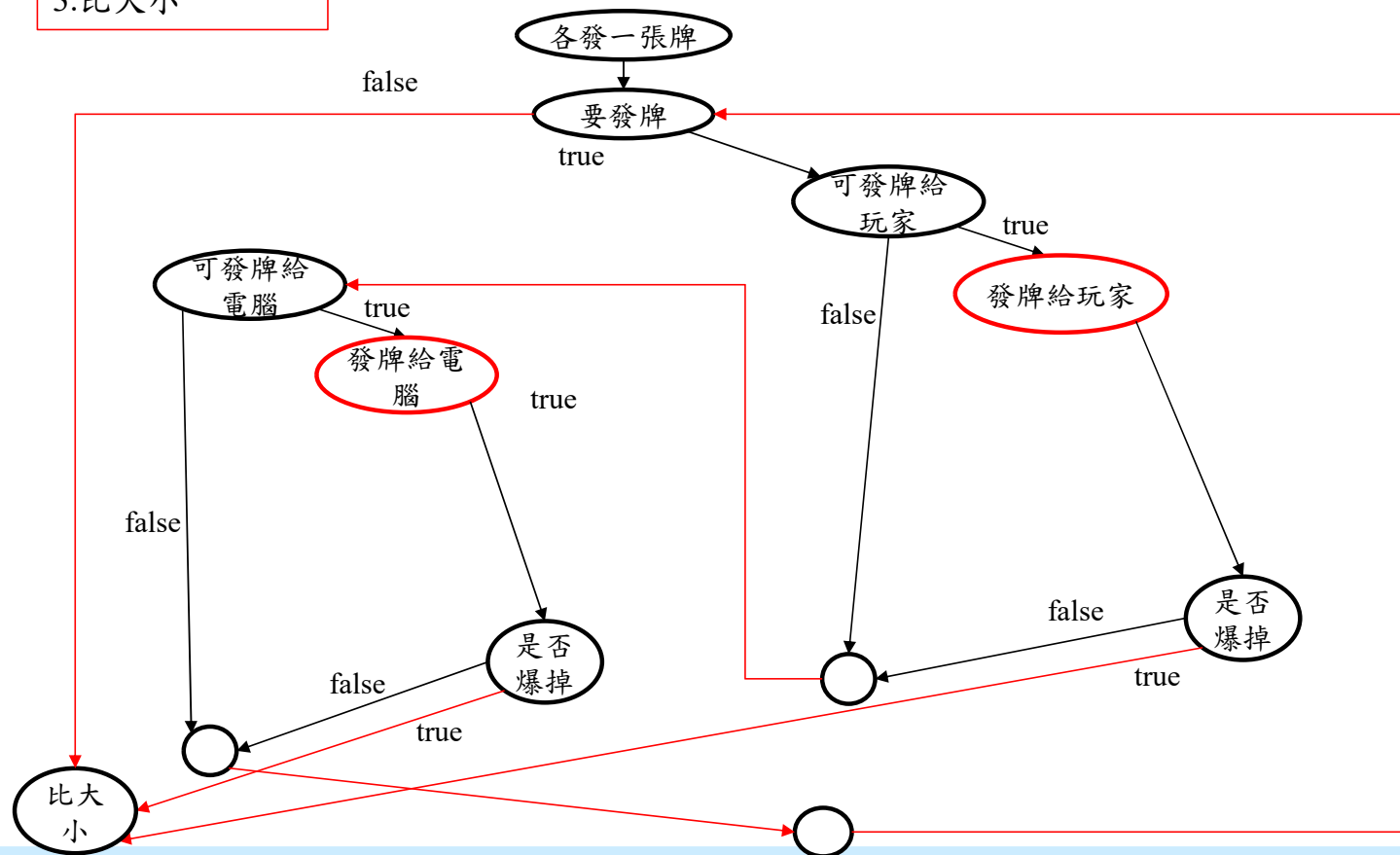
### □ 輸入範例說明

- 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)

# 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



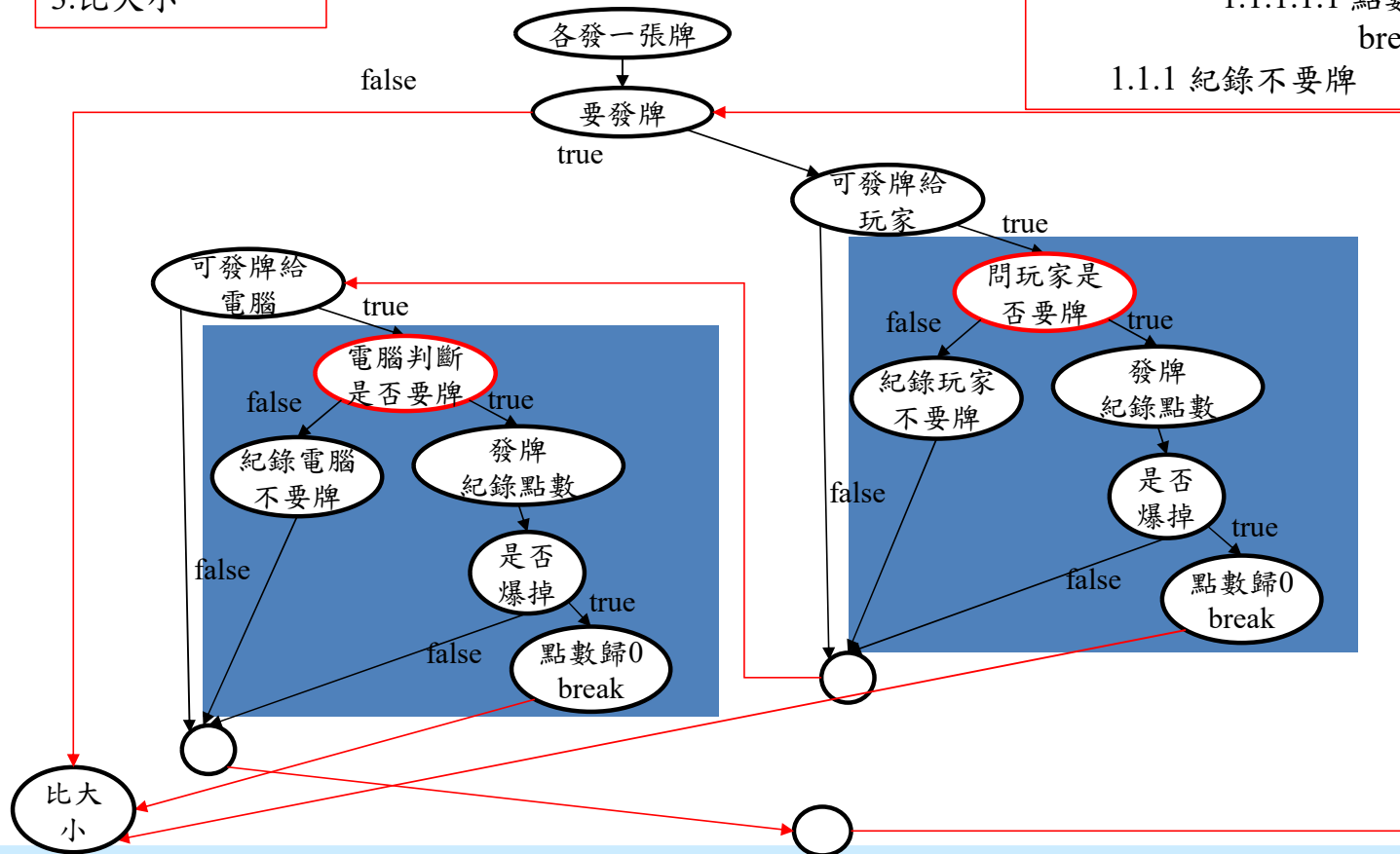
# 撲克牌比大小

## 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



# 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)

# 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

□ 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
- ❑ 重複結構while和for都支援else敘述
  - 迴圈不是因break, return或例外終止時(正常中止)，else\_suite會被執行

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

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

---

END

---

