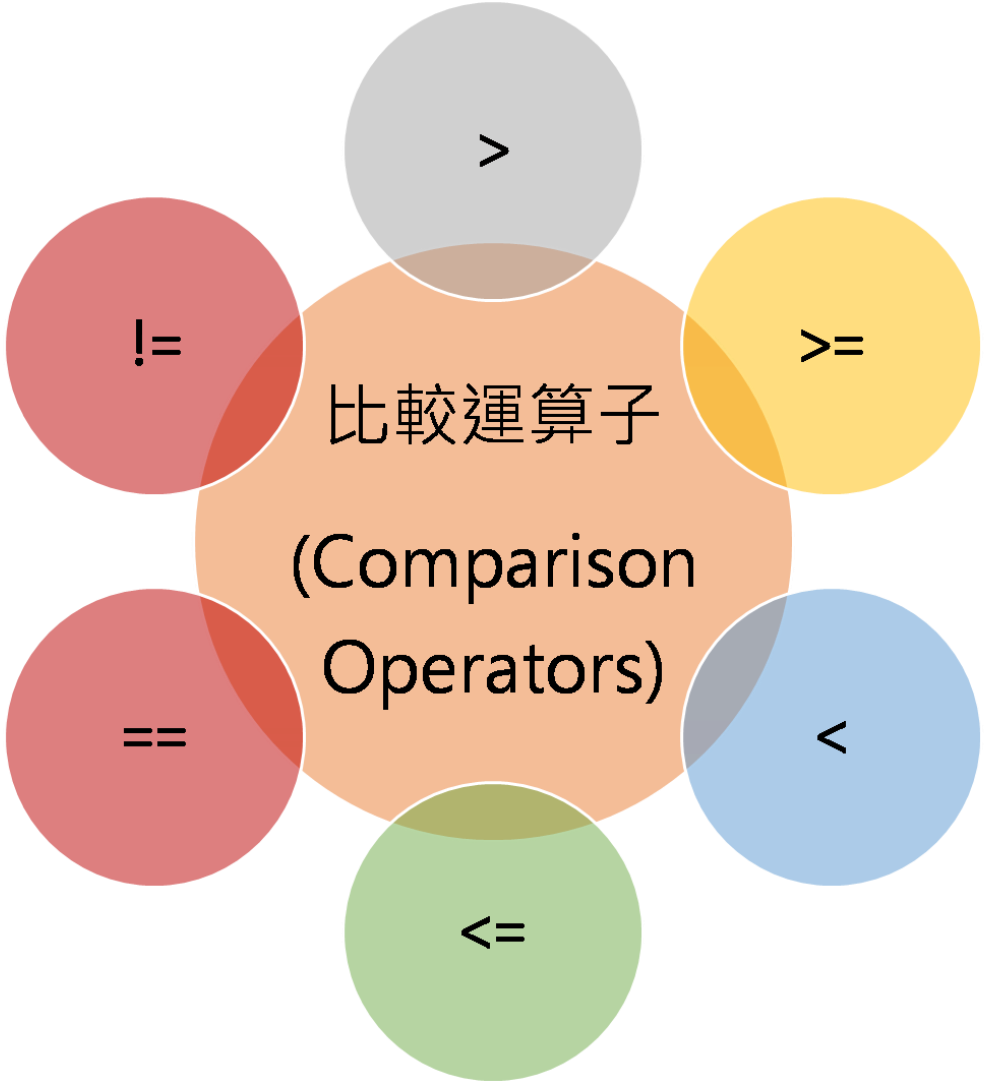
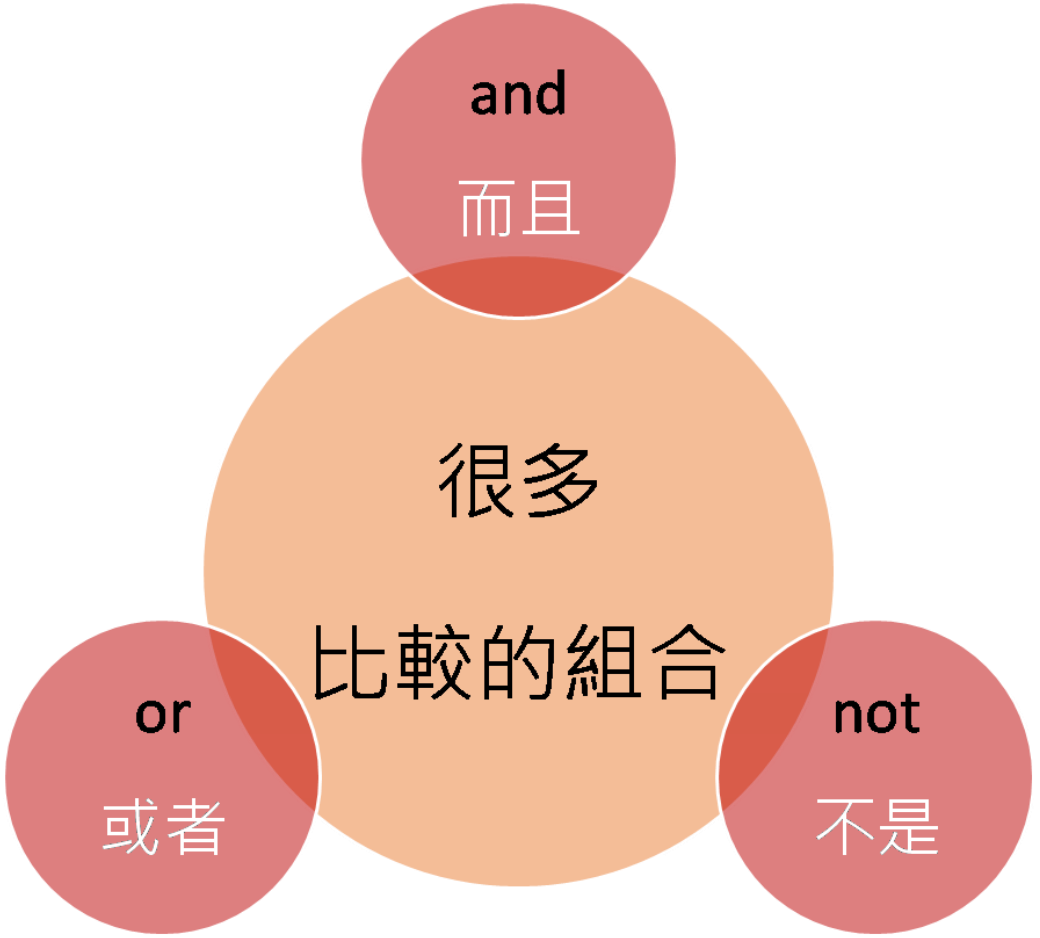
Python

判斷式if...else


不是這樣
就那樣

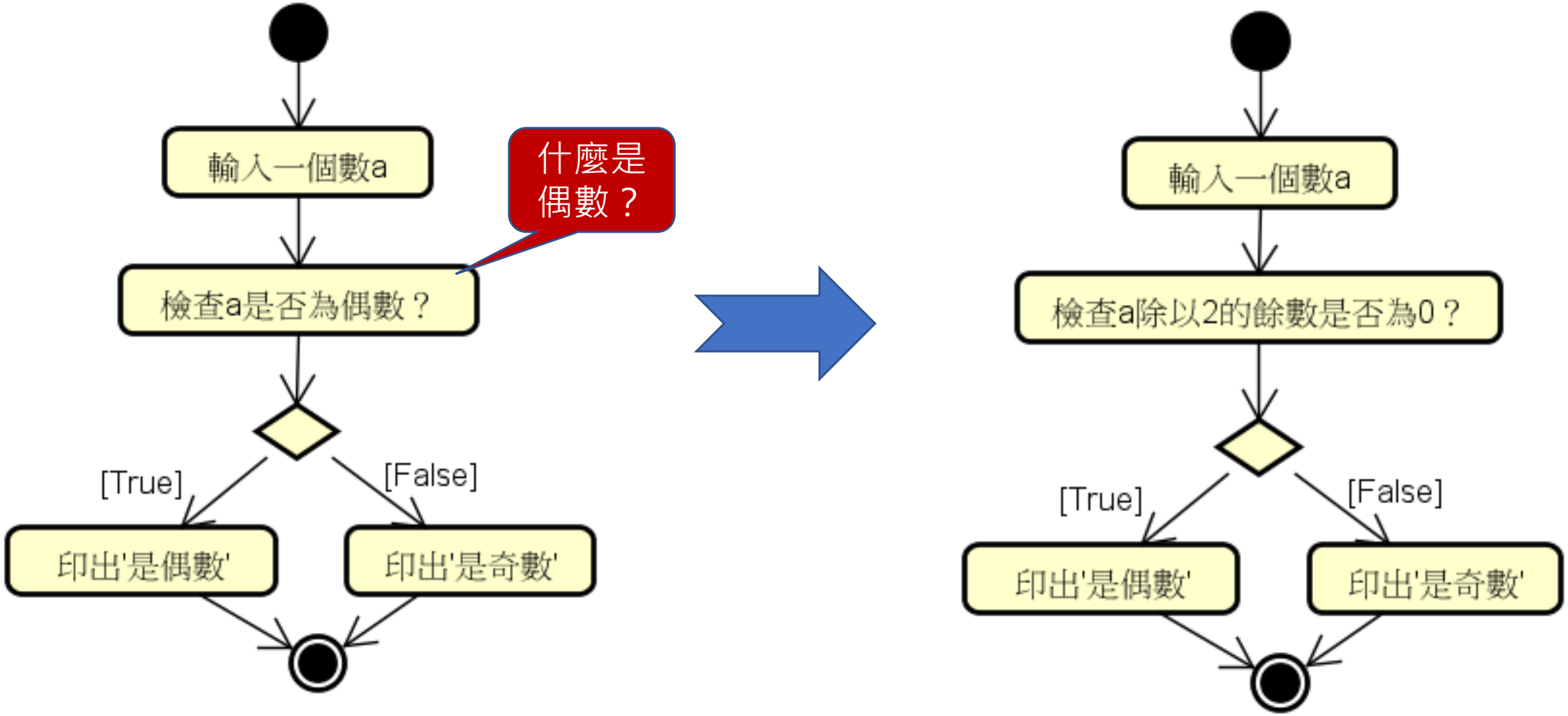
判斷式會常用到**關係(比較)運算子**與**邏輯運算子**





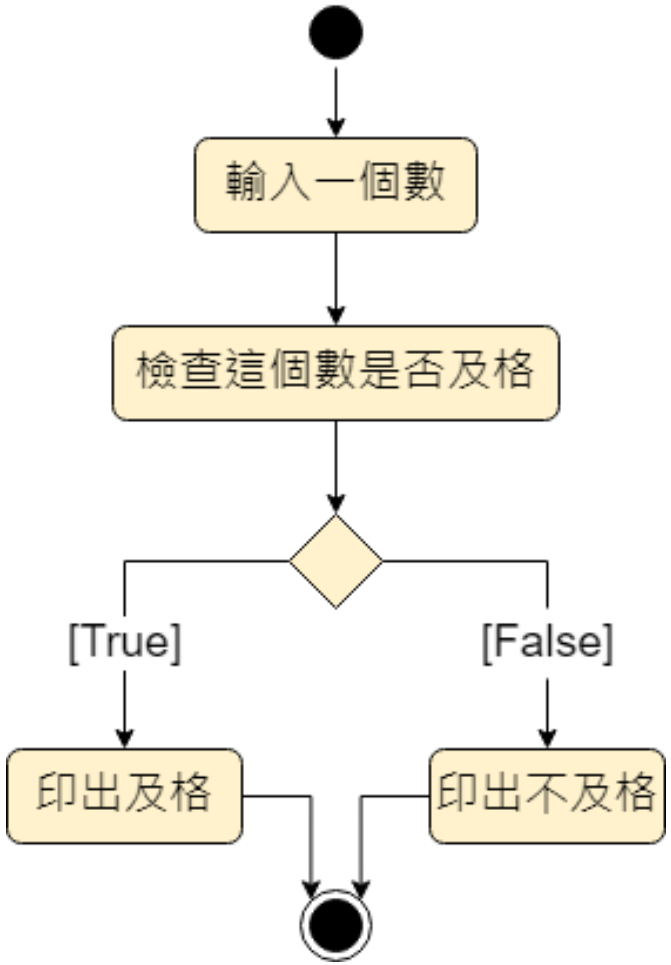
寫一個程式，判斷一個分數，是否及格

流程


輸入一個數a
什麼是偶數？
檢查a是否為偶數？
[True]
[False]
印出'是偶數'
印出'是奇數'
輸入一個數a
檢查a除以2的餘數是否為0？
[True]
[False]
印出'是偶數'
印出'是奇數'

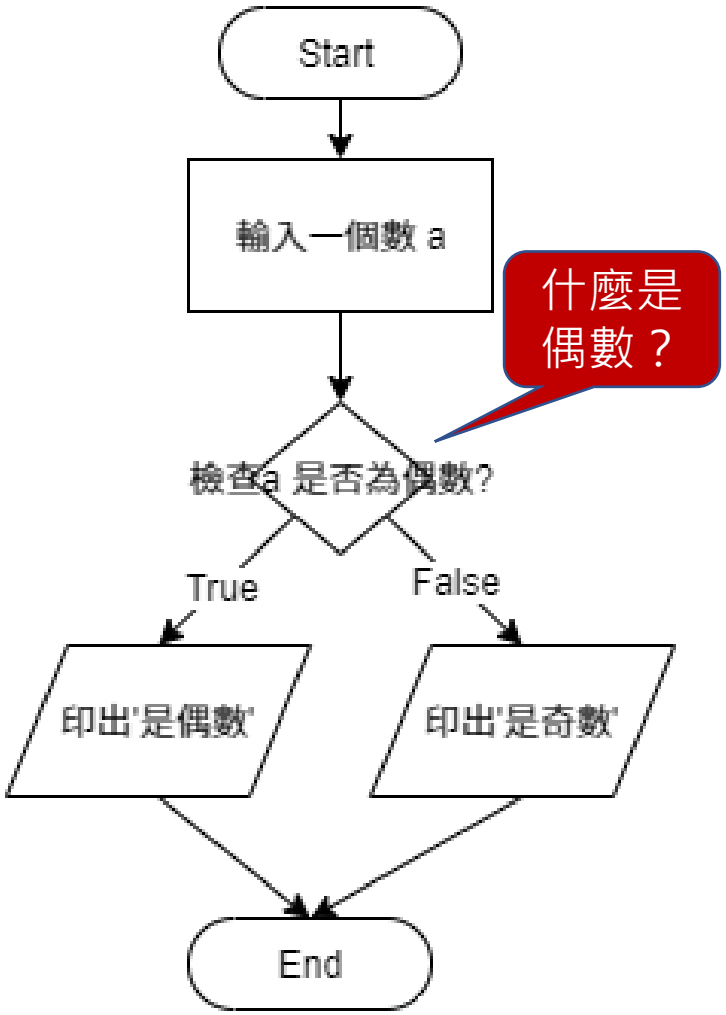
UML流程圖

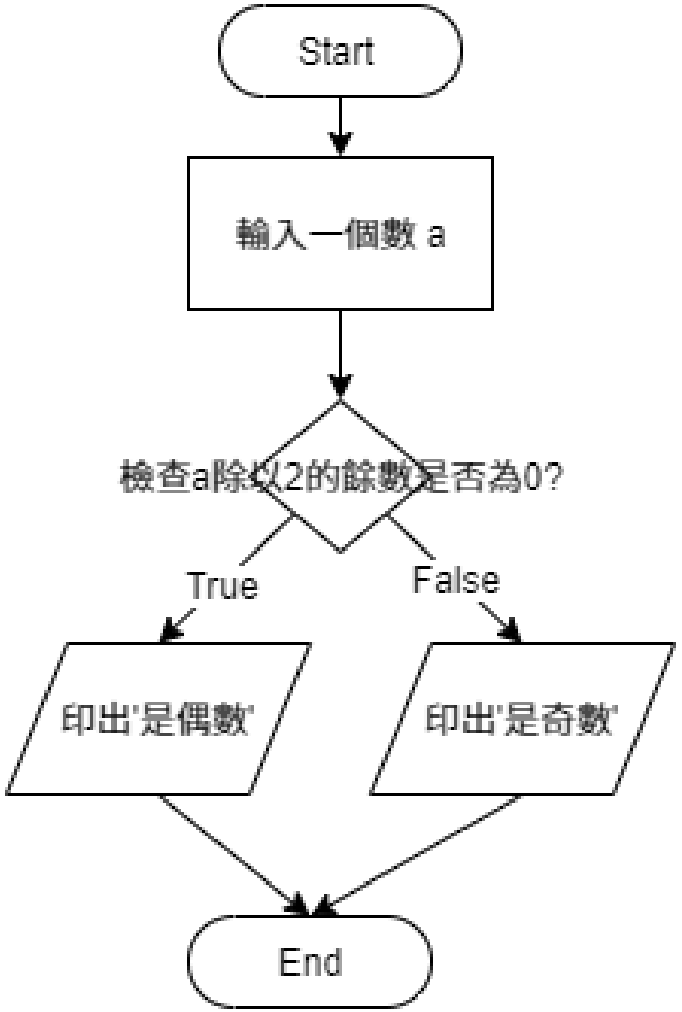
流程



UML流程圖

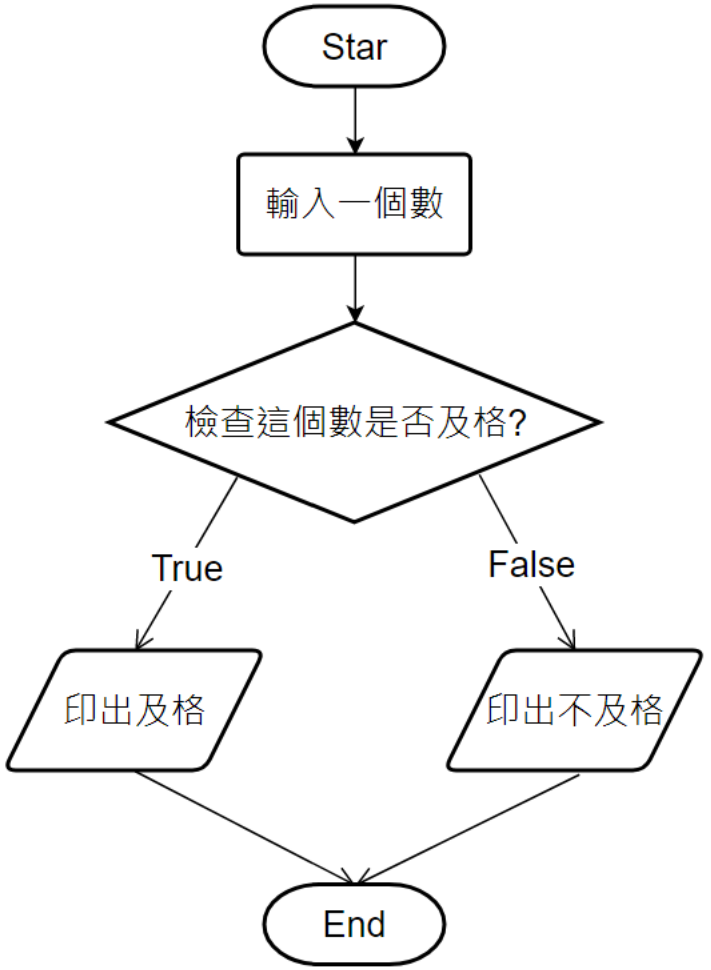
流程


Start
輸入一個數 a
什麼是偶數？
檢查a 是否為偶數?
True
False
印出'是偶數'
印出'是奇數'
End


Start
輸入一個數 a
檢查a除以2的餘數是否為0?
True
False
印出'是偶數'
印出'是奇數'
End

傳統流程圖

流程



傳統流程圖

程式碼: 是否及格

```
score = int(input())

If score >=60:
        print( “{0}分及格".format(score))
Else:
        print( “{0}分不及格” .format(score))
```

執行結果

```
75
75分及格
```

```
50
50分不及格
```

if...else語法

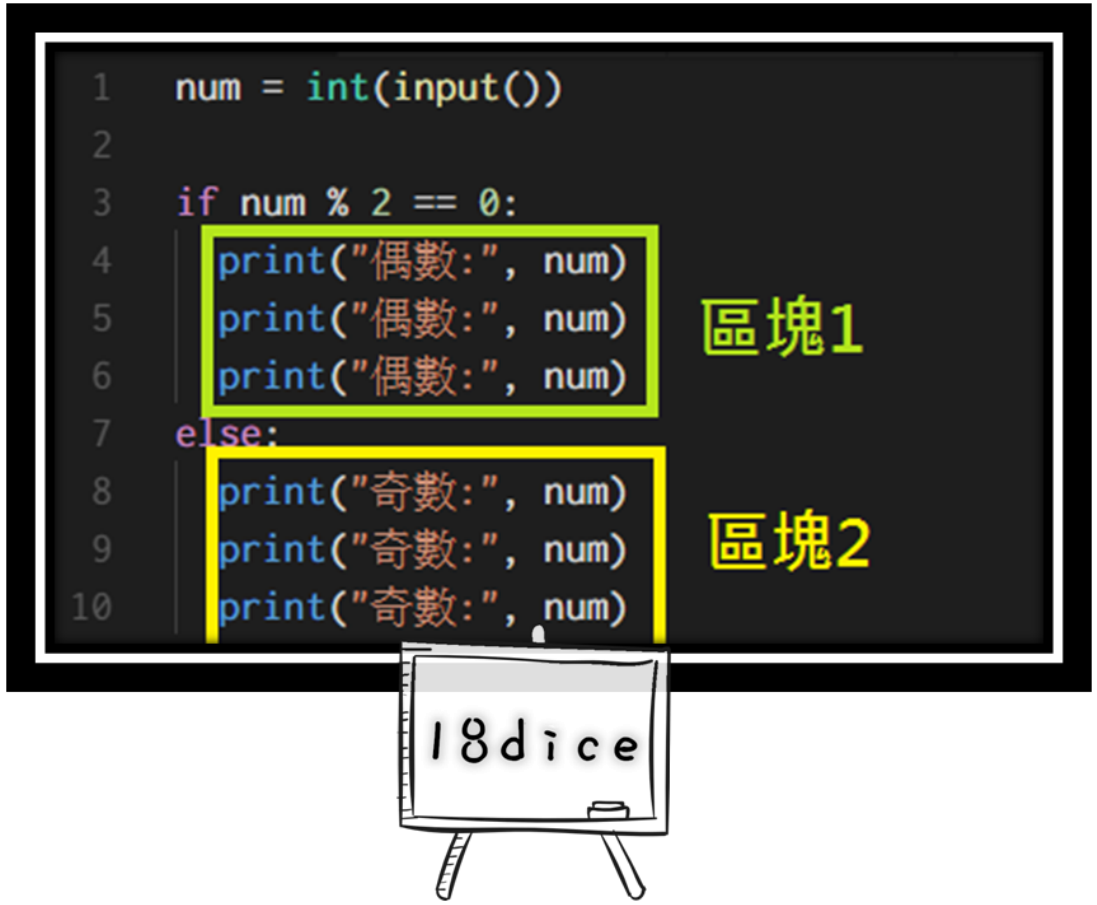
```
score = int(input())

if 條件式:
        敘述式；
else:
        敘述式；
```

PYTHON

延伸的概念

概念 1: 談談寫作風格 - 區塊



- if與else的下方分別包含一個區塊
- 連續有著同樣縮排的行數視為同一個區塊

概念 2: 再談寫作風格 - 分號 與 敘述式

```python
num = int(input())

if num % 2 == 0:
    print("偶數:", num)
    print("偶數:", num)
    print("偶數:", num)
else:
    print("奇數:", num)
    print("奇數:", num)
    print("奇數:", num)
```

一行一個敘述式

可讀性高 👍

- **敘述式：**
  - 一行一個敘述式，敘述式尾端可加也可不加分號(;)
    - 可讀性高
  - 一行多個敘述式，每個敘述式尾端用分號區格
    - 可讀性低，不建議使用

```python
num = int(input())

if num % 2 == 0:
    print("偶數:", num); print("偶數:", num); print("偶數:", num);
else:
    print("奇數:", num); print("奇數:", num); print("奇數:", num);
```

一行多個敘述式

可讀性低

- **分號：**
  - 某些程式語言(例如: C)，敘述式尾端一定要加分號(;)，Python 則不要求，但若多個敘述式放在同一行，則務必在敘述式尾端加分號，但為了程式的可讀性，建議保持一行一個敘述式，維持不使用分號的風格

概念 3: if else 也要偷懶

簡化
if else 敘述

result = a_result if **條件式** else b_result

- 條件式符合時，得到的值會是 a_result
- 條件式不符合時，得到的值會是 b_result

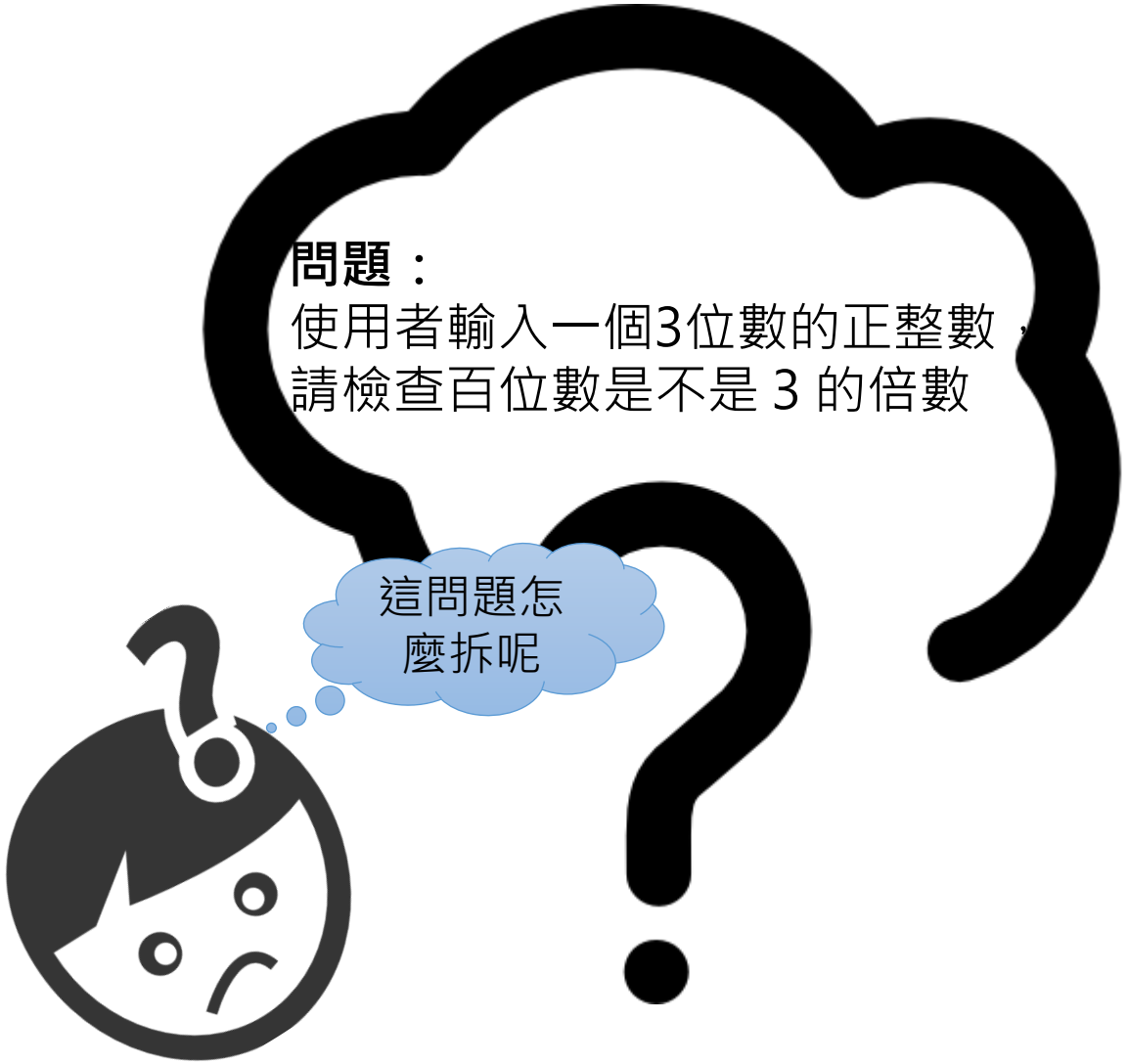
```
num = int(input())
if num > 0:
  abs = num
else:
  abs = -num
print(abs)
```

四行變一行 ☺

```
num = int(input())
abs = num if num > 0 else -num
print(abs)
```

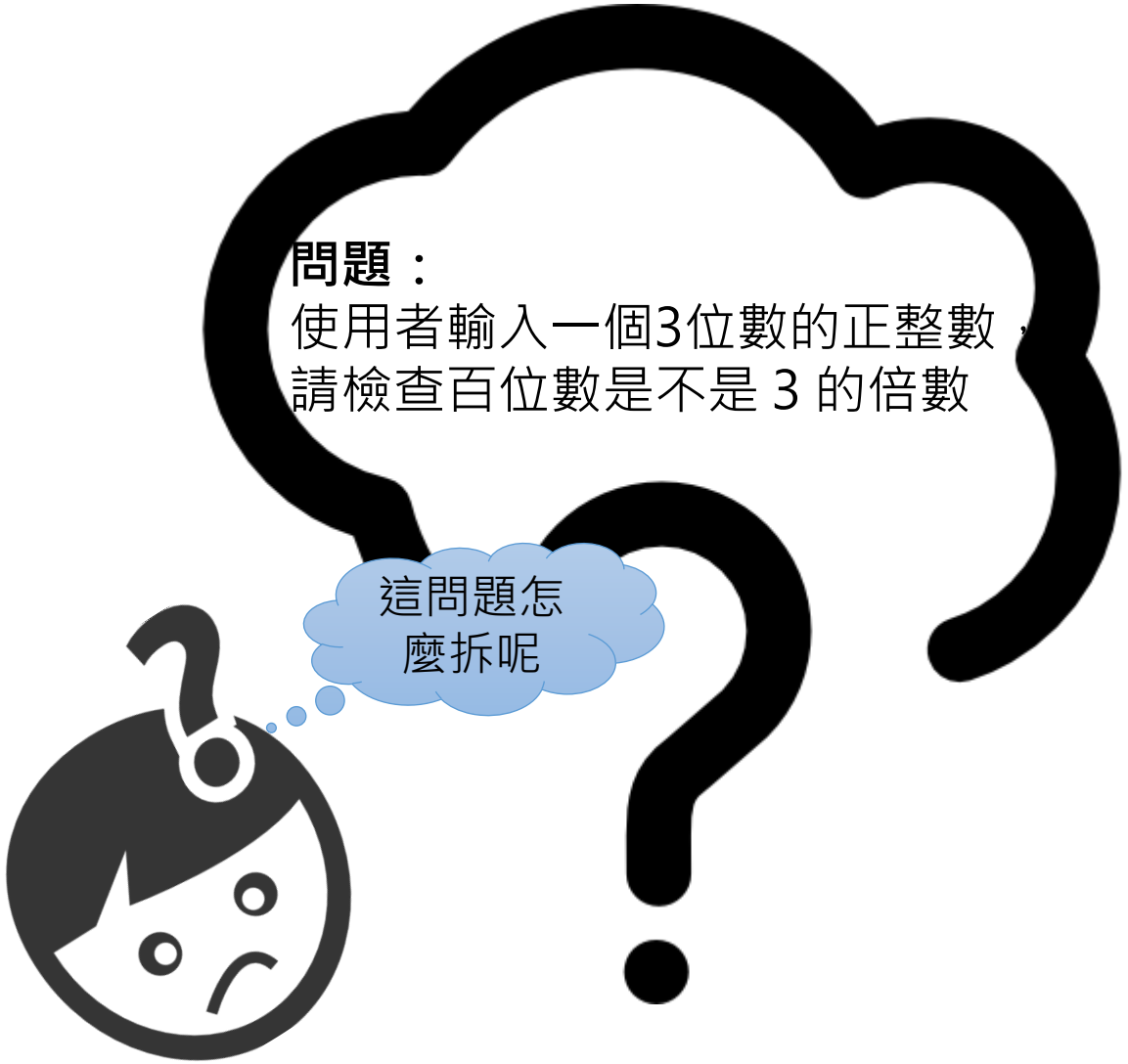
這種偷懶寫法適合用在透過 if else 判斷某資料的值，且該值只有兩種可能時。

概念 4 : if else 的實用處 – 找錯誤(1)



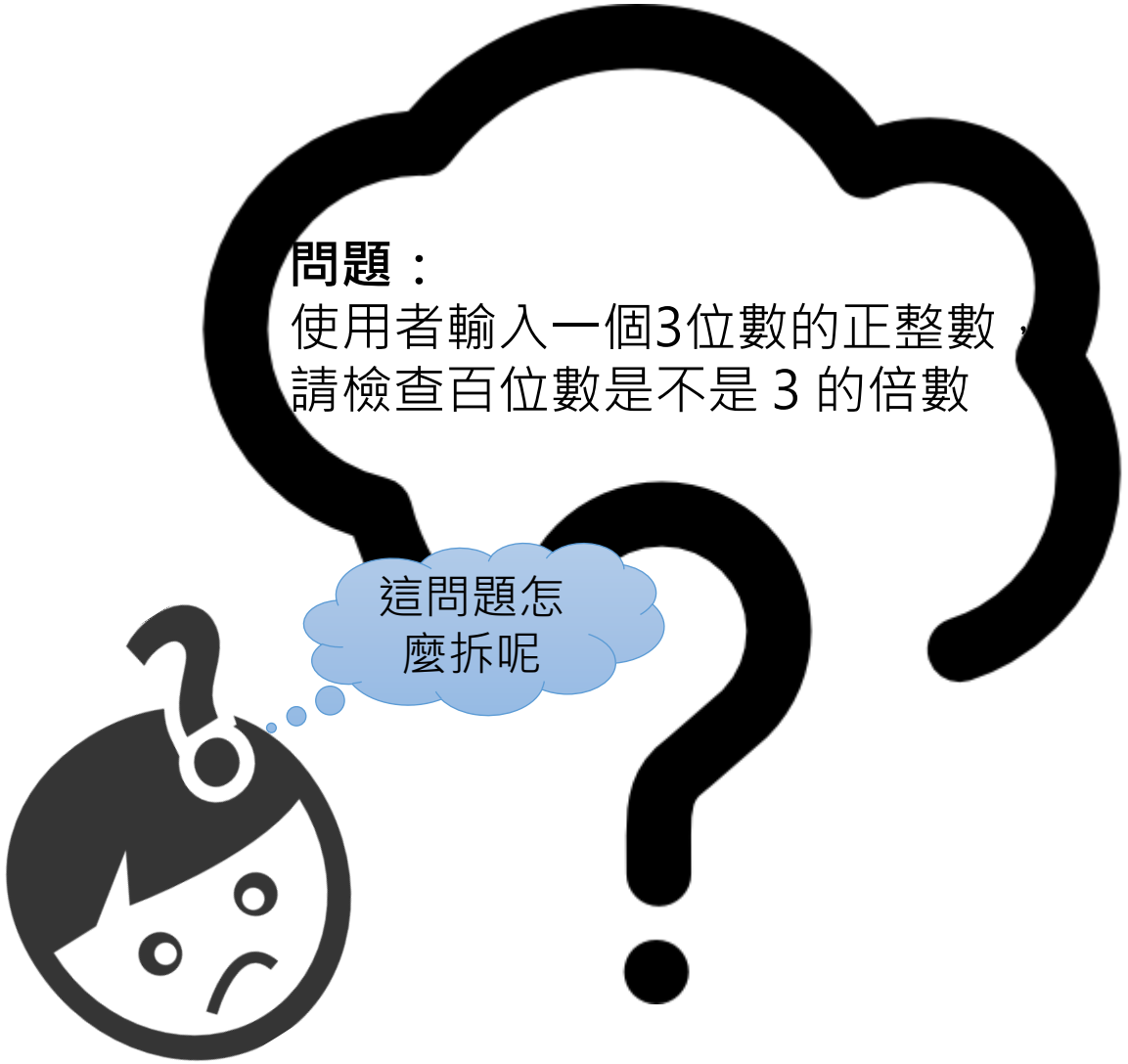
- 學會將問題縮小範圍，逐步找出錯誤
- 當問題範圍縮小後，在每個小問題使用 if else 就可以快速的找出錯誤點！

概念 4 : if else 的實用處 – 找錯誤(2)



- 解題小技巧：
  - **拆解**問題：將一個主問題分解成N個項目，一旦程式有問題，便可逐一針對各個項目確認執行結果是否有錯誤，可快速的找出錯誤！
  - 利用**註解**，分段檢視程式碼：將還沒執行到的程式，每行前方加上 #，該行 # 後面的程式碼就不會執行。

概念 4 : if else 的實用處 – 找錯誤(3)



解題，拆題，找問題！

- 問題：
  - 使用者輸入一個3位數的正整數，請檢查百位數是不是 3 的倍數

- 拆解問題：
  1. 使用者輸入正整數
  2. 取得數字的百位數的值
  3. 百位數的值是不是 3 的倍數

概念 4 : if else 的實用處 – 找錯誤(4)

1 . 輸入的數字是否正確？

```
num = int(input())
# if num < 0:
#  print('{num}不是正整數'.format(num=num))
# if num < 100:
#  print('{num}沒有百位數'.format(num=num))
hundred = num // 100
if hundred % 3 == 0:
  print('百位數{hundred}是3的倍數'.format(hundred=hundred))
else:
  print('百位數{hundred}不是3的倍數'.format(hundred=hundred))
```

2. 百位數的值是否正確？

3.百位數的值是不是 3 的倍數是否判斷正確？

當問題3的判斷結果不如預期時，就可依序執行問題1與問題2的 if 判斷檢查(虛線括號內的註解區塊)