電腦的決策

平均的危機

- 數值資料分析最常見的是統計分析，例如計數、極值、平均值和標準差等等。
- 我們都知道平均值是總和除以個數：
- `avg = total/count`
- 讓我們來寫個求平均的程式：
  - 列印程式標題
  - 輸入總和
  - 輸入個數
  - 計算平均（即總和/個數）
  - 輸出結果

這裡我們不用 sum 作為變數名稱（可以用但不建議），因為 sum 是 Python 內建函式名稱。

平均的危機

求平均的程式

```
print('計算平均的程式')
total = int(input('請輸入總和> '))
count = int(input('請輸入個數> '))
print('答案是', total/count)
```

你會如何測試你的程式？

測試案例一

```
計算平均的程式
請輸入總和> 100
請輸入個數> 5
答案是 20.0
```

測試案例二

```
計算平均的程式
請輸入總和> 42
請輸入個數> -8
答案是 -5.25
```

測試案例三

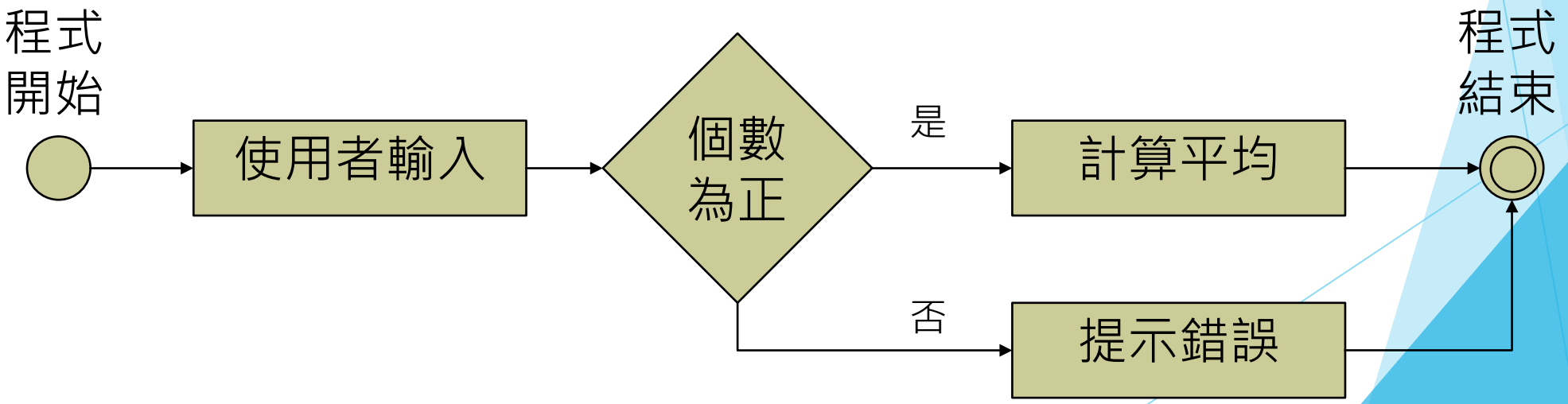
```
計算平均的程式
請輸入總和> 100
請輸入個數> 0
Traceback (most recent call last):
  File "divide by zero.py", line 4, in <module>
    print('答案是', total/count)
ZeroDivisionError: division by zero
```

平均的危機

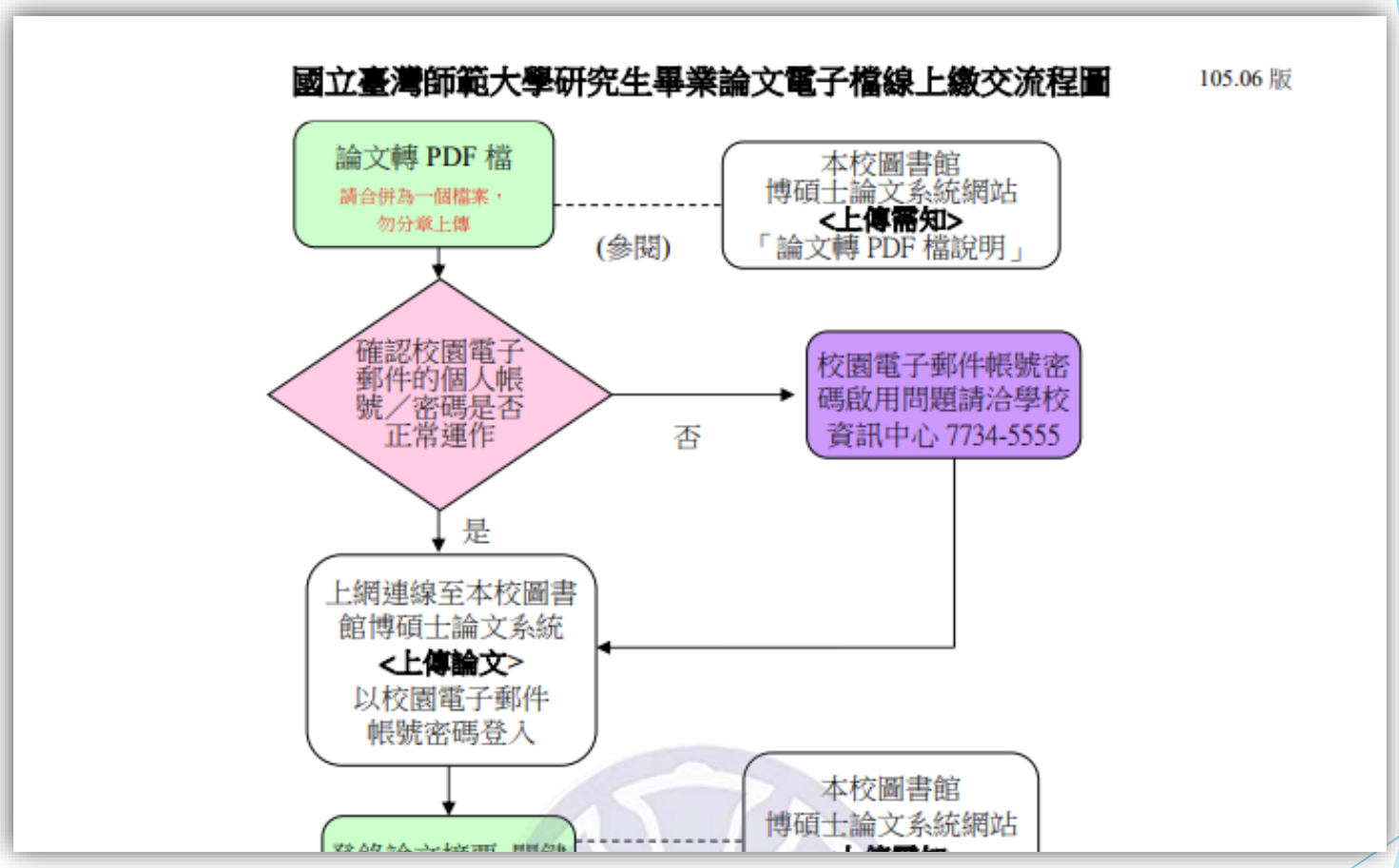
你看人家小算盤那麼聰明

除以零的錯誤

- 電腦在遇到除以零的運算時，通常都會以某種型式回報錯誤。
- 我們能否修改程式，使得：
  - 如果個數為正時，計算平均；
  - 如果個數不為正時，提示使用者輸入錯誤。



生活中的流程圖



http://etds.lib.ntnu.edu.tw/cgi-bin/gs32/gsweb.cgi/ccd=qDv7GX/fqasearch?menu=qsfta

美好的如果

除以零的錯誤也可以利用「例外處理」來解決。

- 大多數的程式語言都有個「如果」的句型
  - 中文

  如果個數為正，計算並列印平均；
  否則，提示使用者輸入錯誤。

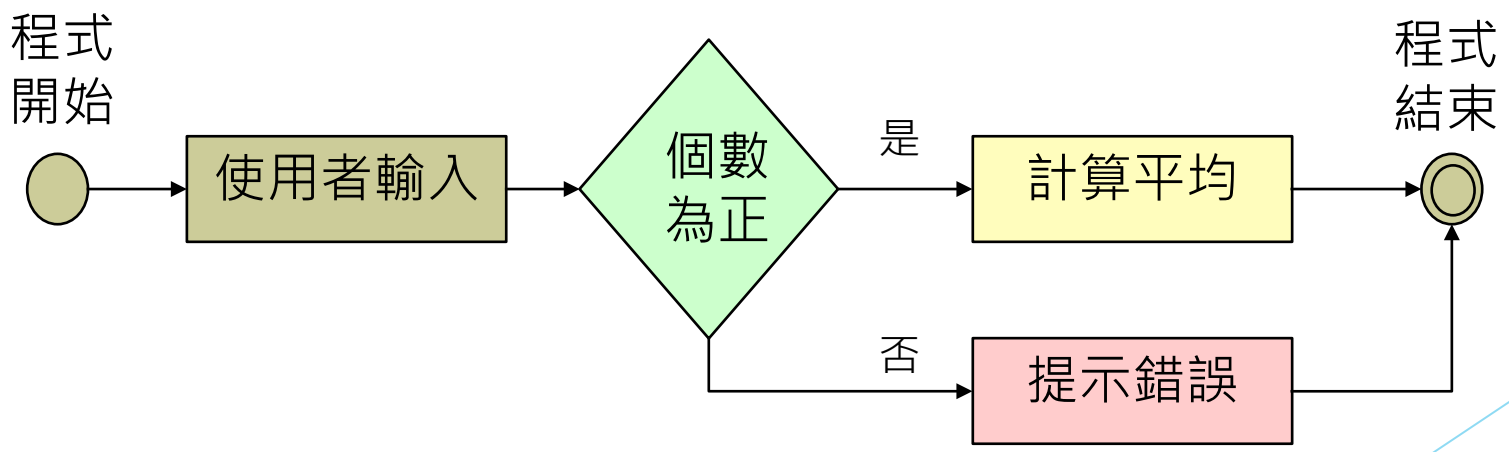
  - 英文

  If the count is positive, calculate and print the average;
  otherwise, show an error message.

美好的如果

- Python的「如果」句型

```
print('計算平均的程式')
total = int(input('請輸入總和> '))
count = int(input('請輸入個數> '))
if count>0:
    print('答案是', total/count)
else:
    print('無法計算-個數需為正值')
```

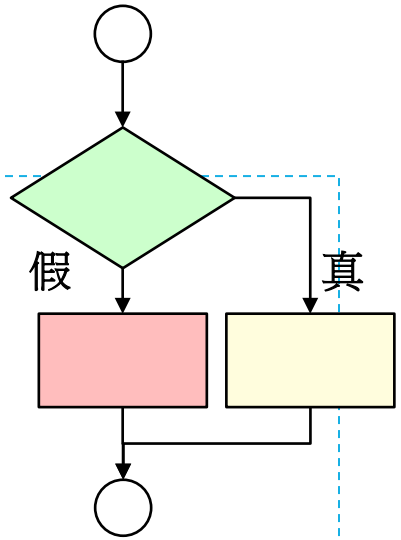


美好的如果

句型剖析

別忘了冒號喔！

```
if        條件式        :
    條件為真時的動作（可以不只一行）
else:
    條件不為真的動作（可以不只一行）
```



句型範例

```
if count>0:
    print('答案是', total/count)
else:
    print('無法計算-個數需為正值')
```

美好的如果

- 句型排版格式規定
  - Python 對於「如果」句型有嚴格的排版格式規定，若不符合規定，會出現 expected an indented block 的語法錯誤。

```
if count>0:
print('答案是', total/count)
else:
print('無法計算-個數需為正值')
```

- 被控制的程式碼必須比主控程式碼內縮。
  - 慣例是內縮四格空白。
  - 大部份開發工具都會自動排版了 – 即你寫完 `if ...:` 之後按下 ENTER 鍵，會自動跳到下一行內縮四個空白處。

美好的如果

- 如果條件不成立的時候沒有特別的工作，可以省略掉 else: 以下的部份。

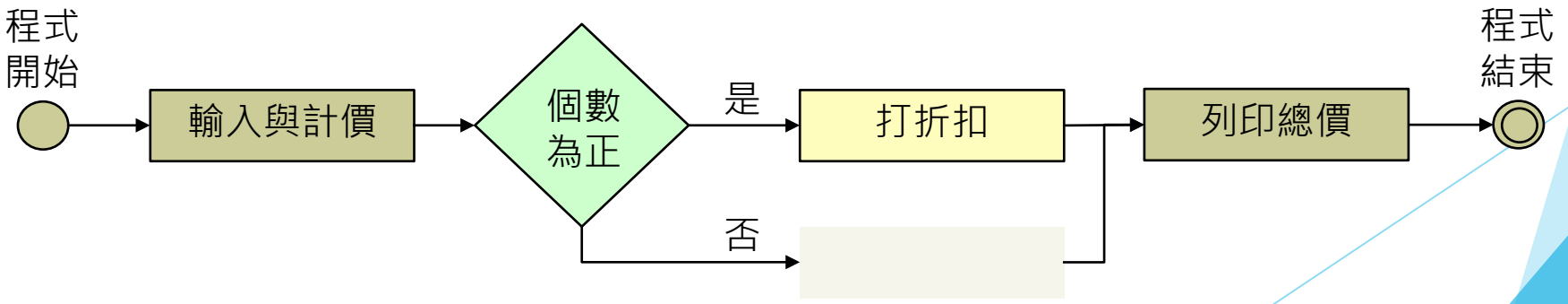
```
print('薩莉亞餐廳的程式')
school = input('請輸入就讀學校> ')
count = int(input('請輸入人數> '))
price = count*100

if school == '臺師大':
    price = price*0.9 #師大人打九折

print('總價', int(price), '元')
```

```
薩莉亞餐廳的程式
請輸入就讀學校> 臺師大
請輸入人數> 10
總價 900 元
```

```
薩莉亞餐廳的程式
請輸入就讀學校> 某大
請輸入人數> 10
總價 1000 元
```



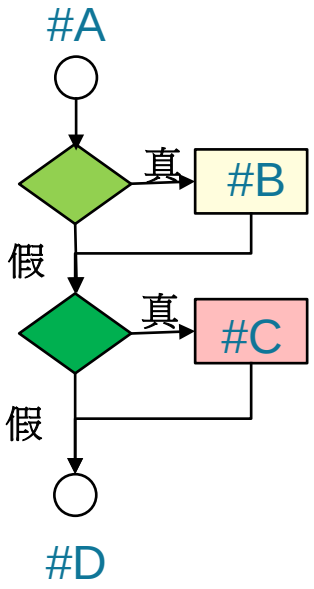
美好的如果

- 可讀性：撰寫讓人一眼看懂的程式碼

```
if count > 0:
    # action B
if count <= 0:
    # action C
```

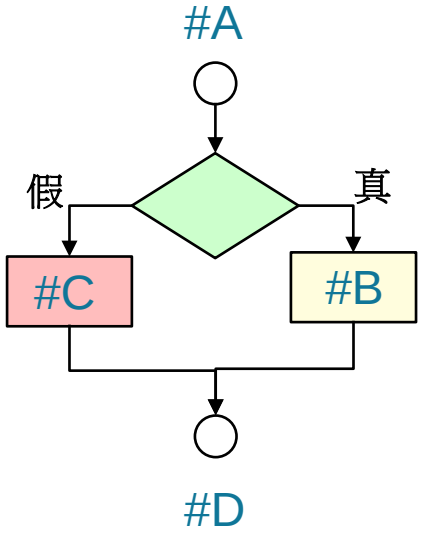
修改後更簡單明瞭

```
if count > 0:
    # action B
else:
    # action C
```



本質上有四種執行可能性：

AD
ABD
ACD
ABCD



本質上有兩種執行可能性：

ABD
ACD

關係運算子

- 「如果」句型以條件式的運算結果
  - 是真 (True)
  - 是假 (False)

  來控制該執行哪一段程式碼。

True 和 False 是布林 (bool, boolean) 型態唯二的數值。

```
if True:
    print('True')
else:
    print('False')

if False:
    print('True')
else:
    print('False')
```

```
True
False
```

關係運算子

- 關係運算子是二元運算子，其結果經常作為「如果」句型的條件式。
  - 二元運算子：運算時需要兩個運算元。
  - 英文時間：運算子 operator，運算元 operand。

意義	關係運算子	結果為真 True	結果為假 False
相等	==	3 == 3	3 == 4
不相等	!=	3 != 4	4 != 4
大於	>	4 > 3	3 > 4
大於等於	>=	4 >= 4	3 >= 4
小於	<	3 < 4	4 < 3
小於等於	<=	3 <= 4	5 <= 4

關係運算子

- 運算次序由高至低
  - 次方
  - 乘/除/餘
  - 加/減
  - 關係運算

Python 程式碼	運算次序	運算結果
3*3==9	(3*3)==9	True
2+3>1+4	(2+3)>(1+4)	False

關係運算子

挑戰時刻


```
x, y = 11, 20

print('A')
if 100 > 3:
    print('B')
if 100 < 200:
    print('C')
if x/2 > 5:
    print('D')
print('E')
if x+y < 40:
    print('F')
if x*2 == y:
    print('G')
if x <= y:
    print('H')
if x+3*y >= x*3+y:
    print('I')
if x != y:
    print('J')
```

```
A
B
C
D
E
F
H
I
J
```

關係運算子

- Snakify 平台主題

  3 Conditions: if, then, else 的習題

  - **Minimum of two numbers**：
    輸入兩個整數，輸出較小的值。

挑戰時刻

- 修改「求平均」程式以達下述功能
  - 如果使用者第一次輸入的個數為正，計算並列印平均。
  - 如果使用者第一次輸入的個數不為正，提示使用者並讓其再輸入一次個數。
    - 如果使用者第二次輸入的個數為正，計算並列印平均。
    - 如果使用者第二次輸入的個數不為正，提示使用者並結束程式。

原程式

```
print('計算平均的程式')
total = int(input('請輸入總和> '))
count = int(input('請輸入個數> '))
if count>0:
    print('答案是', total/count)
else:
    print('無法計算-個數需為正值')
```

挑戰時刻

- 修改「求平均」程式以達下述功能
  - 如果使用者第一次輸入的個數為正，計算並列印平均。
  - 如果使用者第一次輸入的個數不為正，提示使用者並讓其再輸入一次個數。
    - 如果使用者第二次輸入的個數為正，計算並列印平均。
    - 如果使用者第二次輸入的個數不為正，提示使用者並結束程式。

```
print('計算平均的程式')
total = int(input('請輸入總和> '))
count = int(input('請輸入個數> '))
if count>0:
    print('答案是', total/count)
else:
    count = int(input('個數需為正值，請輸入個數> '))
    if count>0:
        print('答案是', total/count)
    else:
        print('無法計算-個數需為正值')
```

缺點：
1. 計算平均的程式片段重覆了。
2. 控制結構複雜（由一層變成兩層）。

挑戰時刻

- 修改「求平均」程式以達下述功能
  - 如果使用者第一次輸入的個數不為正，提示使用者並讓其再輸入一次個數。
  - 如果個數為正，計算平均。
  - 如果個數不為正，提示使用者並結束程式。

```
print('計算平均的程式')
total = int(input('請輸入總和> '))
count = int(input('請輸入個數> '))
if count<=0:
    count = int(input('個數需為正值，請輸入個數> '))

if count>0:
    print('答案是', total/count)
else:
    print('無法計算-個數需為正值')
```

調整控制流程可以得到精簡的程式碼：
1. 沒有重覆片段。
2. 控制結構簡單（一層）。

挑戰時刻

- 這個程式共有幾種可能的執行流程？

```
print('計算平均的程式')
total = int(input('請輸入總和> '))
count = int(input('請輸入個數> '))  #A
if count<=0:
    count = int(input('個數需為正值，請輸入個數> '))#B

if count>0:
    print('答案是', total/count) #C
else:
    print('無法計算-個數需為正值') #D
```

編號	執行流程	測試案例
1	AC	依序輸入 100 3
2	ABC	依序輸入 100 0 3
3	ABD	依序輸入 100 0 0

辨識程式可能的執行流程並完整測試是程式員的基本且重要任務。

如果還有如果

巢狀控制

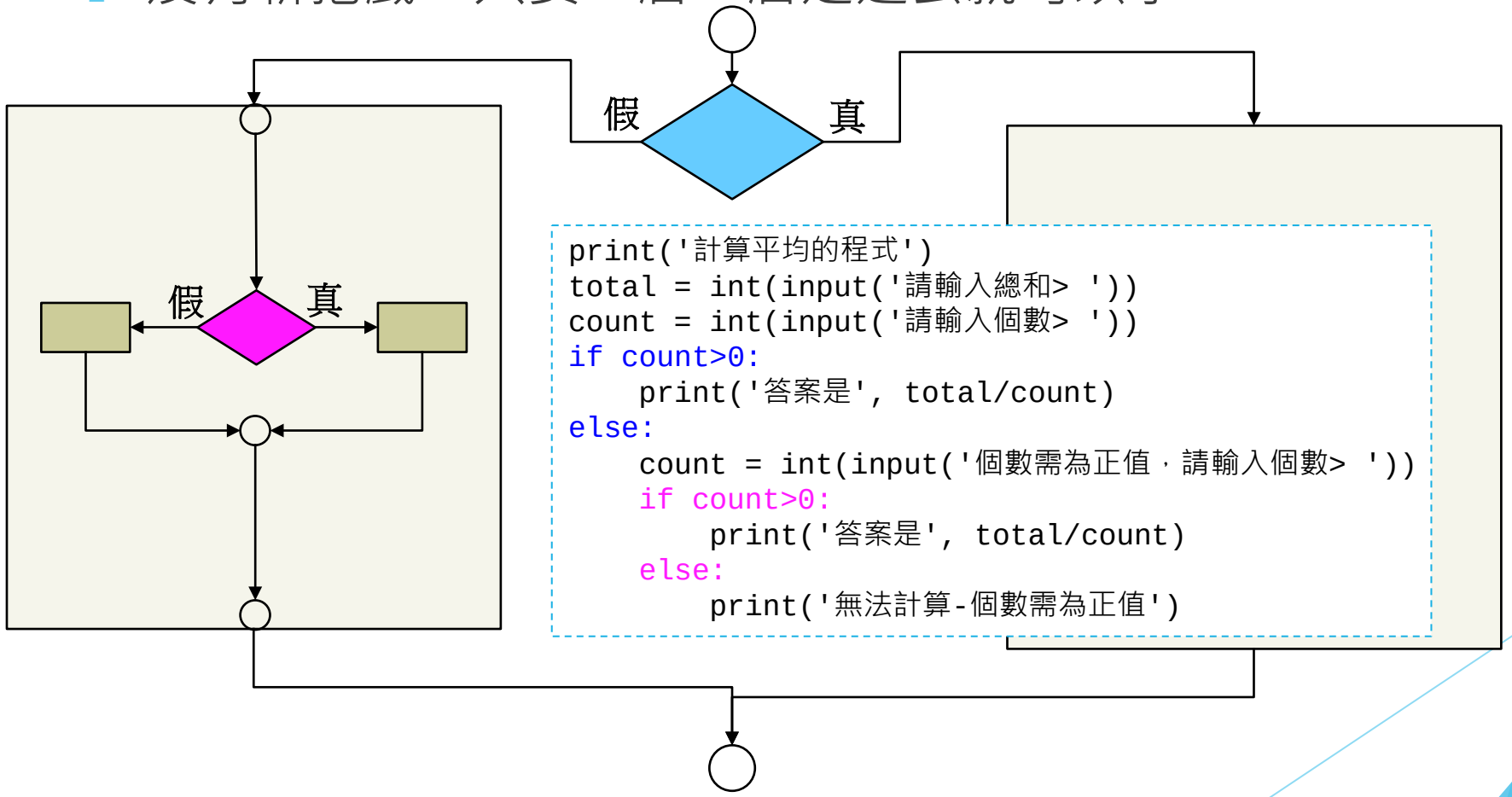
- 在前面的挑戰時刻，我們看到了「如果」還有「如果」的程式碼。
- 我們通常稱這種多於一層的程式為巢狀 (nested) 控制結構。

```
print('計算平均的程式')
total = int(input('請輸入總和> '))
count = int(input('請輸入個數> '))
if count>0:
    print('答案是', total/count)
else:
    count = int(input('個數需為正值，請輸入個數> '))
    if count>0:
        print('答案是', total/count)
    else:
        print('無法計算-個數需為正值')
```

如果還有如果

巢狀控制

- 沒有新把戲，只要一層一層走進去就可以了。




```
print('計算平均的程式')
total = int(input('請輸入總和> '))
count = int(input('請輸入個數> '))
if count>0:
    print('答案是', total/count)
else:
    count = int(input('個數需為正值，請輸入個數> '))
    if count>0:
        print('答案是', total/count)
    else:
        print('無法計算-個數需為正值')
```

如果不只兩種可能 .....?

- 在前面的例子中，我們判斷個數是不是正值。
- 如果我們想判斷正值、零和負值，分別作不同的動作，該怎麼作呢？

這是巢狀的作法

```
v = int(input('請輸入一個整數值> '))
if v>0:
    print('正值')
else:
    if v==0:
        print('零')
    else:
        print('負值')
```

如果不只兩種可能 .....?

- 對同一個(組)資料進行多重判斷時，我們通常會將「如果」句型維持在同一個層級。

```
v = int(input('請輸入一個整數值> '))
if v>0:
    print('正值')
else:
    if v==0:
        print('零')
    else:
        print('負值')
```

兩層

```
v = int(input('請輸入一個整數值> '))
if v>0:
    print('正值')
elif v==0:
    print('零')
else:
    print('負值')
```

一層

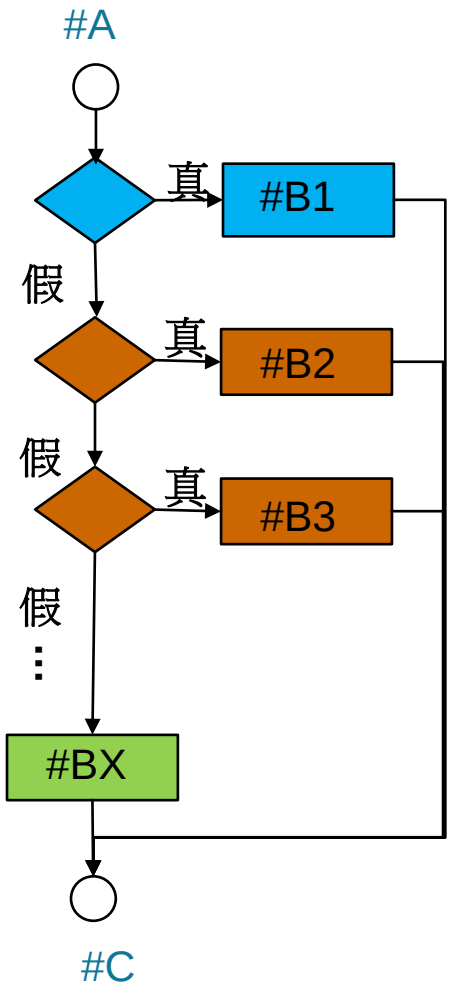
Snakify 平台主題

3 Conditions: if, then, else 的習題

Sign function：輸入一個整數，
若為正數，輸出 1；
若為負數，輸出 -1；
若為零，輸出 0。

如果不只兩種可能 .....?

- if … elif … else 句型




```
A
if condition1:
    B1
elif condition2:
    B2
elif condition3:
    B3
...
else:
    BX
C
```

這段程式有幾種可能的執行流程？

如果不只兩種可能 .....?

最常見的多分法 – 單值

```
grade = int(input('你的年級 >'))
if grade == 1:
    print('一年級')
elif grade == 2:
    print('二年級')
elif grade == 3:
    print('三年級')
elif grade == 4:
    print('四年級')
else:
    print('大大級')
```

```
bt = input('你的血型 >')
if bt == 'O':
    print('O 型人')
elif bt == 'A':
    print('A 型人')
elif bt == 'B':
    print('B 型人')
elif bt == 'AB':
    print('AB 型人')
else:
    print('外星人')
```

雖然 if 句型不一定要有 else 的部份，但建議保留此部份作為檢查時的最後一道關卡。

如果不只兩種可能 .....?

- 另一種常見的多分法：區間判斷

```
price = 10
number = int(input())
if number>=12:
    price = price*number*0.9
elif number>=6:
    price = price*number*0.95
elif number>=2:
    price = price*number*0.99
else:
    price = price*number

print(number, price)
```

- 12 人以上
- 6-11 人
- 2-5 人
- 1 人

要特別注意「由上而下」的判斷順序。

如果不只兩種可能 .....?

- 多分法練習：請撰寫一個程式，使用者輸入一個分數，請依以下區間輸出等第。

```
score = int(input('請輸入分數> '))
if 90<=score:
   print(score, 'A+')
elif 85<=score:
    print(score, 'A')
elif 80<=score:
    print(score, 'A-')
elif 77<=score:
    print(score, 'B+')
elif 73<=score:
    print(score, 'B')
elif 70<=score:
    print(score, 'B-')
```

分數區間	等第
90~100	A+
85~89	A
80~84	A-
77~79	B+
73~76	B
70~72	B-

這個程式的小缺點：當使用者輸入其它範圍的分數時（如 500 或 65），程式沒有輸出錯誤訊息。
這個程式你至少要測試過幾次?

如果不只兩種可能 .....?

補強版 A

```
score = int(input('請輸入分數> '))

if 100<score:
    print(score, '不在本程式接受範圍')
elif 90<=score:
   print(score, 'A+')
elif 85<=score:
    print(score, 'A')
elif 80<=score:
    print(score, 'A-')
elif 77<=score:
    print(score, 'B+')
elif 73<=score:
    print(score, 'B')
elif 70<=score:
    print(score, 'B-')
else:
    print(score, '不在本程式接受範圍')
```

補強版 B

```
score = int(input('請輸入分數> '))

if 90<=score<=100:
   print(score, 'A+')
elif 85<=score<=89:
    print(score, 'A')
elif 80<=score<=84:
    print(score, 'A-')
elif 77<=score<=79:
    print(score, 'B+')
elif 73<=score<=76:
    print(score, 'B')
elif 70<=score<=72:
    print(score, 'B-')
else:
    print(score, '不在本程式接受範圍')
```

`90<=score<=100` 在 Python 中是 `90<=score and score<=100` 的意思。我們很快就會談到邏輯運算子 `and`。

總結：「如果」句型

- 一個「如果」句型是由
  - 一個 `if`
  - 零個以上的 `elif`
  - 零個或一個 `else`

  組成的。

- 巢狀：「如果」句型裡面可以繼續使用「如果」句型。

邏輯運算子：如果句型的好幫手

- 邏輯運算子可以簡化「如果」句子。

```
if month == 8:
    if day == 15:
        print('中秋節快樂!')
    else:
        print('平常日')
else:
    print('平常日')
```

使用邏輯運算子 and

```
if month == 8 and day == 15:
    print('中秋節快樂')
else:
    print('平常日')
```

```
if age <= 12:
    price *= 0.9
    normal = False
if age >= 65:
    price *= 0.9
    normal = False
if normal:
    price -= 50
```

使用邏輯運算子 or

```
if age <= 12 or age >= 65:
    price *= 0.9
else:
    price -= 50
```

邏輯運算子 and

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

```
if month == 8 and day == 15:
    print('中秋節快樂!')
else:
    print('平常日')
```

- 非 0 的數值都視為 True。

```
a = 3.14
if a:
    print('這行會印出來')
```

- 若 a 為 False，無論 b 是 True 或是 False，a and b 的結果都是 False。

在 Python 中，若 a 為 False，a and b 運算並不會評估 b 的值。此稱為短路 (short-circuit)。

```
a = []
if False and a[0]>1:
    print('沒有索引範圍錯誤')
```

邏輯運算子 or

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

若 a 為 True，無論 b 是 True 或是 False，a or b 的結果都是 True。

在 Python 中，若 a 為 True，a or b 運算並不會評估 b 的值。此稱為短路 (short-circuit)。

```
a = []
if True or a[0]>1:
    print('沒有索引範圍錯誤')
```

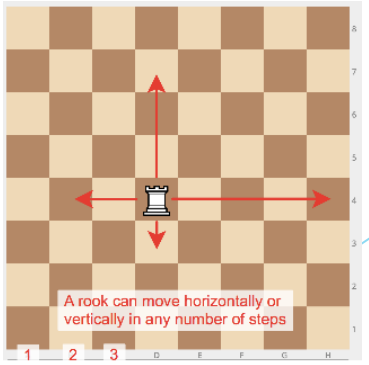
```
if score >= 60 or times_absence <
3:
    print('及格了')
else:
    print('不及格')
```

注意：這只是個範例。

Snakify 平台主題

3 Conditions: if, then, else 的習題

Rook Move



邏輯運算子 not

a	not a
False	True
True	False

```
if not (month == 8 and day == 15):
    print('平常日')
else:
    print('中秋節快樂!')
```

```
if not month == 8 and day == 15:
    print('平常日')
else:
    print('中秋節快樂!')
```


一年有幾天中秋節?

「如果」句型與「分類」

- **請描述/想像你在修完本課程後，會使用程式進行什麼樣的資料分析/自動化工作？如果可以，請進一步說明你想處理的資料格式與樣貌。**

「可以將資料分群、分類，因為在處理地理資料時分群分類很重要 。」

-- 文學院同學

「透過簡單的問卷(像是google表單)調查出一些想知道的數據，
然後用程式分析出那些數據再加以分類。」

-- 理學院同學

「設定關鍵字或關係式去自動化分區域或類別，
可能比較適合處理大量(重複性高)的資料。」

-- 理學院同學

「如果」句型與「分類」

- 我們已經會使用「如果」來「分類」了耶！

```
grade = int(input('你的年級 >'))
if grade == 1:
    print('一年級')
elif grade == 2:
    print('二年級')
elif grade == 3:
    print('三年級')
elif grade == 4:
    print('四年級')
else:
    print('大大級')
```

年級分類

```
bt = input('你的血型 >')
if bt == 'O':
    print('O 型人')
elif bt == 'A':
    print('A 型人')
elif bt == 'B':
    print('B 型人')
elif bt == 'AB':
    print('AB 型人')
else:
    print('外星人')
```

血型分類

```
score = int(input('請輸入分數> '))

if 90<=score<=100:
   print(score, 'A+')
elif 85<=score<=89:
    print(score, 'A')
elif 80<=score<=84:
    print(score, 'A-')
elif 77<=score<=79:
    print(score, 'B+')
elif 73<=score<=76:
    print(score, 'B')
elif 70<=score<=72:
    print(score, 'B-')
else:
    print(score, '不在本程式接受範圍')
```

分數區間分類

「如果」句型與「分類」

- 以下兩群數字怎麼分類？


4 16 4 10 2 6 8 12 20 14 18

3 5 17 19 1 15 9 21 11 7 13

44 26 64 21 65 40 68 81 82 24 8

34 52 79 37 57 11 96 14 91 74 30

84 86 89 88 80 81 87 82 0 85 83

-34 52 79 37 57 11 -96 -14 91 -74 -30

```
if num%2==0:
    #blue
else:
    #pink
```

```
if (num//10)%2==0:
    #blue
else:
    #pink
```

```
if num==0 or 80<=num<=89:
    #blue
else:
    #pink
```

「如果」句型與「分類」

- 以下兩群數字怎麼分類？

```
if num%4 == 0:
    if num%100!=0:
        #左上
    else:
        #左下
else:
    #右邊
```

1996 12 2108
4 8 96
800
2400 400
100 300
500 200
2100 2700

34 65 65
1887 19 773
799 13 82
21 443
27
5 317 159 191
19

```
if num%4 == 0:
    #左邊
else:
    #右邊
```

「如果」句型與「分類」

Snakify 平台主題

3 Conditions: if, then, else 的習題

Leap year 四年一閏，逢百不閏，四百又閏

```
if y%4 == 0:
    print('LEAP')
    if y%100 == 0:
        if y%400 == 0:
        else:
    else:
else:
```

```
if y%400 == 0:
    print('LEAP')
elif y%100 == 0:
elif y%4 == 0:
else:
```

上面的程式碼使用三層巢狀結構，下面的程式碼僅使用一層。

不過，有些人會覺得下面的程式碼不好懂。

想想看如何運用邏輯運算子寫出簡潔好懂的程式碼。

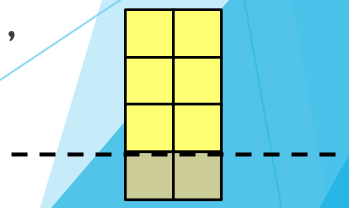
Snakify 線上練習平台：作業二

- 請完成 Snakify 平台主題
  3 Conditions: if, then, else 的習題。

- 計分方式（以題數計）：
  - 第 1-4 題，每題 +20%
  - 第 5-12 題，每題 +5%

答對任四題：80分
答對任六題：90分

- 提示
  - 課堂上已示範四題，至少 80 分了。 ☺
  - **Chocolate bar** 的題意是給一片有 n × m 小塊的巧克力，折一次，能不能變成含 k 小塊的巧克力。



4×2 的巧克力，沿虛線折會變成 6 小塊的巧克力。

更多範例：「如果」句型

- 多分法：籃球賽最後 10 秒的戰術

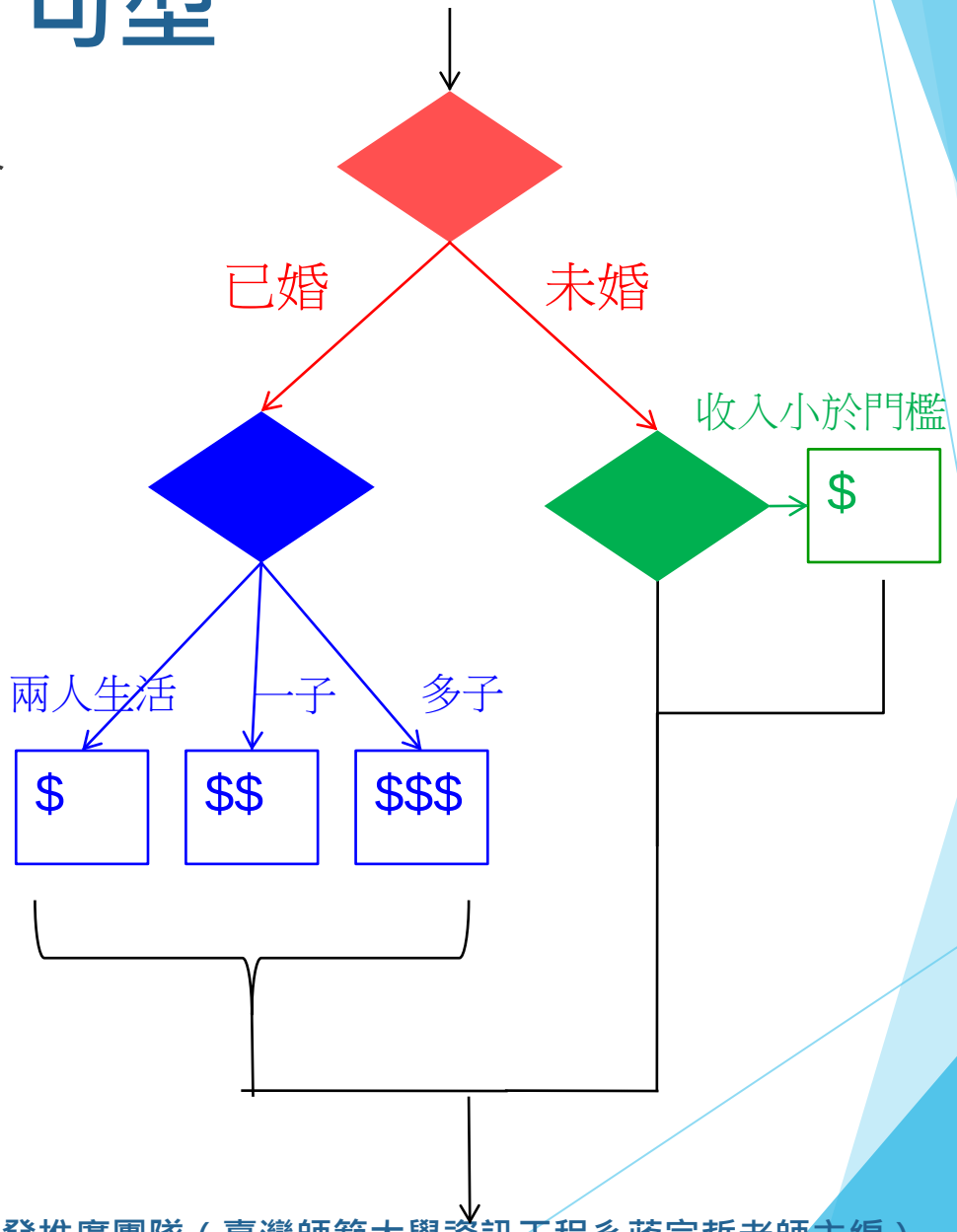
```
if our_score > opp_score: #贏球
    print('把球給罰球命中率高的，拖完時間')
elif our_score >= opp_score-2: #差距 2 分以內
    print('進攻禁區')
elif our_score == opp_score-3: #差 3 分
    print('設計三分球戰術')
else: # 差 4 分以上
    print('快速進攻，能進就好')
```

更多範例：「如果」句型

- 巢狀與多分法：調薪策略

```
if married:
    if num_kids == 0:
        bonus = 1000
    elif num_kids == 1:
        bonus = 2000
    else: #more than one kid
        bonus = 4000

else: #single
    if salary <= 50000:
        bonus = 500
```

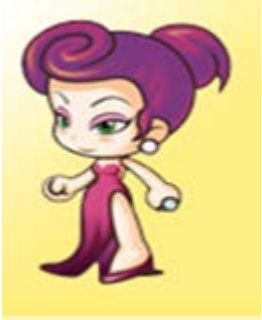


更多範例：「如果」句型

- 巢狀與多分法：大富翁遊戲-升級土地

```
if money>=price:
    print('You can upgrade your land to one of these buildings:\n'
          '(1) Department store\n'
          '(2) Park\n'
          '(3) Business building\n')
    choice = int(input('Which type of building do you want?...>'))

    if choice == 1:
        # 升級為百貨公司
    elif choice == 2:
        # 升級為公園
    elif choice == 3:
        # 升級為商辦
else:
    # 不夠錢
```



大富翁:
http://rich.joypark.com.tw/player/player.aspx

更多範例：「如果」句型

- 巢狀與多分法：角色扮演遊戲-戰鬥選單

```
action = int(input('選擇行動...>'))
if action == 1: #物理攻擊
    weapon = int(input('選擇武器...>'))
    if weapon == 1:
        print('使用長劍')
    elif weapon == 2:
        print('使用拳頭')

elif action == 2:
    #法術攻擊
elif action == 3:
    #使用物品
```

更多範例：「如果」句型

- 巢狀與多分法：角色扮演遊戲-接觸 NPC

```
if not visited:
    print('Welcome, adventurer. I will give you some gift.')

    if job == 1: #warrior
        attack += 1
    elif job == 2: #archer
        dexterity += 1
    ...

else: #already visited
    print('Good day')
```