迴圈原理及應用

林隆鴻 2020/10/29

Outline

1. `if` 介紹
2. `for` 迴圈介紹
3. `while` 迴圈介紹
4. 題目練習

1. `if`:

(1)基本架構:

```
if (條件):
    如果條件成立，執行此區塊。
else:
    條件不成立，則執行此區塊。
```

EX1: 你今天想用 python 給使用者猜擲一次銅板是正面還是反面，你可以這樣寫(假 head):

In [1]:

```
def guess_func(guess):
    if guess == "tail": # ==: 關係運算子
        print("You are wrong, guess again!")
    else:
        print("You are right!")
```

In [2]:

```
guess_func("tail")
```

```
You are wrong, guess again!
```

- 盲點: 只要你不輸入 `tail` ，電腦就認定是對的，但這跟遊戲的目的不一樣:

In [3]:

```
guess_func("aaaaaaaaaaa")
```

```
You are right!
```

(2)改善: 使用 `elif(else if)` 架構:

基本架構:

```
if (條件一):
    執行結果一
elif (條件二):
    執行結果二
else:
    執行結果三
```

- 注意: 在一個 `if-else` 判斷式裡可以有無限多個 `elif` ，但只能有一個 `else`
- `else` 後面不用加條件: 意思就是只要不符合條件一或條件二，就通通執行 `el`

一樣假設正確答案是 `head`：

In [4]:

```
def guess_func2(guess):
    if guess == "head": # 回答 "head" 是正確答案
        print("You are right!")
    elif guess == "tail":
        print("You are wrong, guess again!")
    else: # 如果回答其他答案也是錯
        print("Not correct syntax, please type 'head' or 'tail'!")
```

In [5]:

```
guess_func2("aaaaaaa")
```

```
Not correct syntax, please type 'head' or 'tail'!
```

Python 的 `EAFP` (Easier to Ask Forgivness than Permission)原則:

先舉一個反例:
今天我有一個字典(`dict`)，裡面存放不同人的性別，年齡，工作等。
如果某人的所有資料都在這個字典裡，就把他的資料印出來:

In [6]:

```
person = {'name':'Bob', 'age':23, 'job':'developer'}
```

In [7]:

```
if "name" in people and 'age' in people and 'job' in people:
    print("I'm {name}. I'm {age} age years old. I am a {job}.".for
else:
    print("Missing some keys.")
```

```
---------------------------------------------------------------
NameError                                 Traceback (most recent c
<ipython-input-7-20080792d777> in <module>
----> 1 if "name" in people and 'age' in people and 'job' in peopl
      2     print("I'm {name}. I'm {age} age years old. I am a {jo
t(**person))
      3 else:
      4     print("Missing some keys.")

NameError: name 'people' is not defined
```

這種寫法沒錯，但不符合 python 的寫作風格:

In [8]:

```
# EAFP(Pythonic) 的寫作風格:
try:
    print("I'm {name}. I'm {age} age years old. I am a {job}.".for
except KeyError as e:
    print("Missing {} key".format(e))
```

```
I'm Bob. I'm 23 age years old. I am a developer.
```

2. for 迴圈

(1)基本架構:

```
for 變數 in 可迭代物件:
    執行程式碼
```

- 可迭代物件包括: 串列，元組，字典，集合， range() 等。

EX1:列印出一個串列的所有元素。

In [9]:

```
fruits = ['apple', 'banana', 'cherry', 'durian'] # fruits 是一個 li

for i in fruits:
    print(i)
```

```
apple
banana
cherry
durian
```

- range() 是一個 sequence，但可以把它當作 python 的內建函數，語法:

  range(起始值, 終止值, 間隔值)

> Rather than being a function, range is actually an immutable sequence typ
> Ranges and Sequence Types — list, tuple, range.

In [10]:

```
a = list(range(10))
a
```

Out[10]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

EX2: 叫 python 幫我們算 0 到 100 相加。

In [11]:

```
ans = 0 # 先將 ans 設為 0。

for i in range(101):
    ans = ans + i # ans += i
print(ans)
```

```
5050
```

[練習 1]: 算 1 到 100 的奇數和。

In [12]:

```
ans2 = 0
for i in range(1, 101, 2):
    ans2 += i
print(ans2)
```

```
2500
```

EX2 2: 算 2 到 250 裡 7 的倍數和:

In [14]:

```
ans3 = 0

for i in range(2, 251):
    if i % 7 == 0:
        ans3 += i
print(ans3)
```

```
4410
```

(2) 巢狀 for 迴圈:

EX1: 九九乘法表:

In [15]:

```
for i in range(1, 10):
    for j in range(1, 10):
        print(f'{i} * {j} = {i * j}')
    print(" ")
```

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45

6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
```

```
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54

7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63

8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

- 巢狀迴圈的缺點是閱讀不易。

While 迴圈

`for` 迴圈的特色之一是有固定的循環次數，那如果今天想要一個依照條件停止的迴圈，就要使用 `while` 。

(1) 基本架構:

```
while (條件判斷):
    條件成立，則執行此區域。
條件不成立，跳過 block，直接執行此區塊。
```

EX1: 用剛才水果串列的例子。

In [16]:

```
fruits = ['apple', 'banana', 'cherry', 'durian']
```

In [17]:

```
i = 0
while i < 4: # i 可以等於 0, 1, 2, 3
    print(fruits[i])
    i += 1
```

```
apple
banana
cherry
durian
```

小心 無限迴圈(infinte loop) !!

```
i = 0
while i < 4:
    print(fruits[i])
    # i += 1
```

通常在做迴圈時，都會設定一個指標(這個例子是 `i`)，它可以用來幫我們判斷甚麼時
可以自己把 `i+=1` comment 掉，就會出現無限迴圈，你的電腦就會變很吵。

In [*]:

```
i = 0
while i < 4: # i 可以等於 0, 1, 2, 3
    print(fruits[i])
```

```
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
apple
```

EX2:用 python 畫直角三角形。

In [18]:

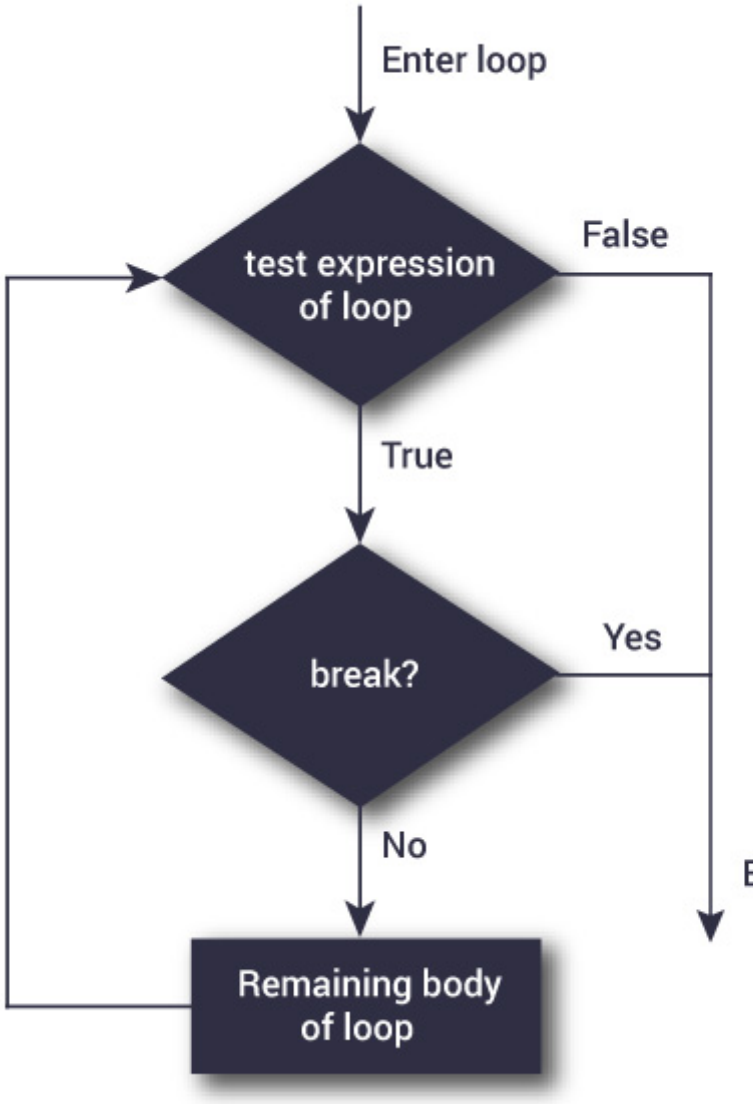
```
i = 1
while i <= 4:
    print("*" * i)
    i += 1
```

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

三個搭配語法: break , continue , 與 pass :

- 在迴圈內若出現語法 break ，則:
  強制跳出 整個 迴圈。
- 若出現的是 continue ，則:
  強制跳出 當前這一個 迴圈。
- 若出現的是 pass ，則:
  甚麼也不做。

EX0.1 break :



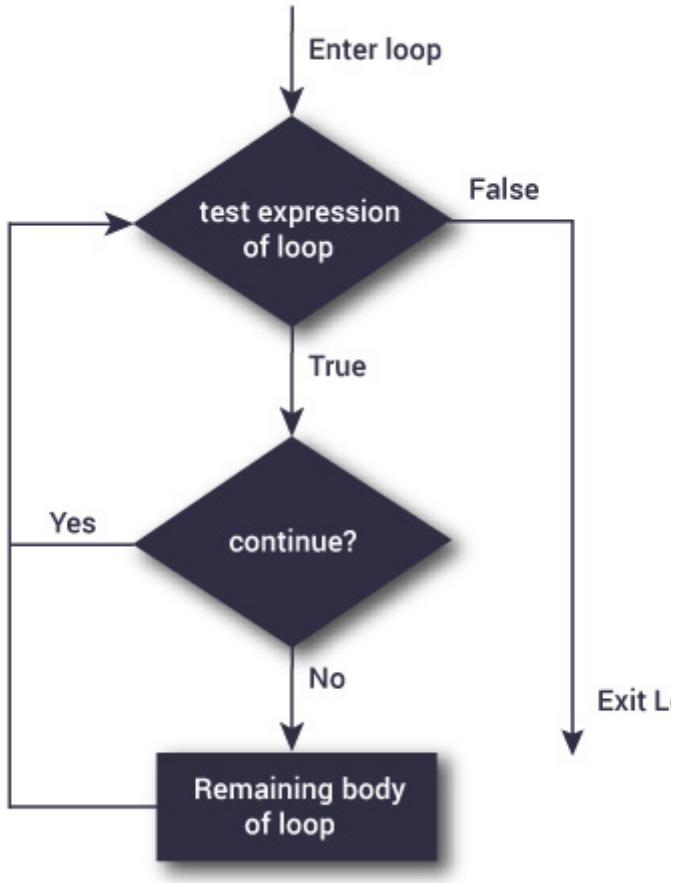
In [19]:

```
count = 0 # 指標
for char in 'content':
    count += 1
    if char == 't':
        break
    print(char)

print("")
print("結束迴圈")
print("共執行 {} 次迴圈。".format(count))
```

```
c
o
n

結束迴圈
共執行 4 次迴圈。
```

迴圈執行了 4 次，在第 4 次時 `break` 掉。

EX0.2 `continue`：



In [20]:

```
count = 0 # 指標
for char in 'content':
    count += 1
    if char == 't':
        continue
    print(char)

print("")
print("結束迴圈")
print("共執行 {} 次迴圈。".format(count))
```

```
c
o
n
e
n

結束迴圈
共執行 7 次迴圈。
```

在出現 `t` 時跳出那次的迴圈，因此不會出現 `t` 。

In [21]:

```
count = 0 # 指標
for char in 'content':
    count += 1
    if char == 't':
        pass
    print(char)

print("")
print("結束迴圈")
print("共執行 {} 次迴圈。".format(count))
```

```
c
o
n
t
e
n
t

結束迴圈
共執行 7 次迴圈。
```

[練習 2]

孤單的時候，總想要找一個人聊天，你可以寫出一個聊天機器人來幫你排解孤單嗎?
你希望他:

- 在你輸入任何話時，都輸出 "你是最棒的!"
- 在你輸入 "讓我一個人靜一靜~" 後，不輸出任何話，等你準備好了，再說話。
- 在你輸入 "我好多了，掰掰。" 時，結束這個機器人。

In [*]:

```
s = ""

while s != "我好多了，掰掰。":
    s = input()
    if s == "讓我一個人靜一靜~":
        s = ""
        continue
    print("你是最棒的!")
```

[練習 3]

給定一個串列 `l` ，裡面有 `10` 個元素。
目標是將裡面的元素由小到大排列:

[提示]
如果在 l 的第 a 個元素，知道 a 右邊(下一個)的元素比 a 小，那我就把這兩個元素
一直這樣兩兩相比，總共要比幾次?
總共要比幾輪?

In [24]:

```
l = [2, 5, 10, 7, 1, 8, 6, 4, 3, 9]
```

In [25]:

```
# bubble sort
for i in range(10):
    p = 0
    while p < 9:
        if l[p + 1] < l[p]:
            temp = l[p]
            l[p] = l[p + 1]
            l[p + 1] = temp
        p += 1
print(l)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

[補充] List comprehension:

- 今天想要一個內容物是 1-10 的 list ，可以這樣寫:

```
def gen_list(max):
  l = []
  count = 1
  for i in range(max):
      l.append(count)
      count += 1
```

但是為了一個小小的 list 寫這麼多 code很費時，而且如果 code 很多，會造
用 list comprehension:

```
l = [x + 1 for x in range(10)]
```

References:

- https://www.programiz.com/python-programming/break-continue (https://www.
programming/break-continue)

In [ ]: