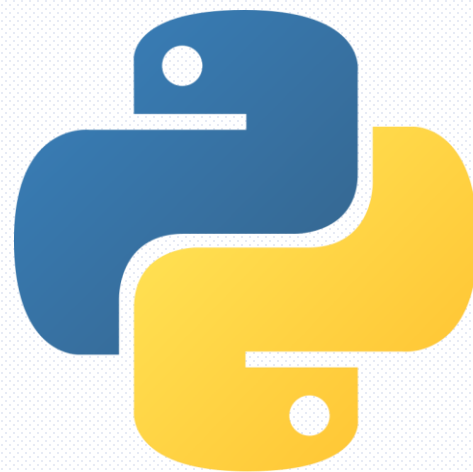


if 後面隱藏了bool函數，判斷True還是False，  
0、 false、None、空

# 流程控制 - 條件控制

Control Flow - Conditional statements

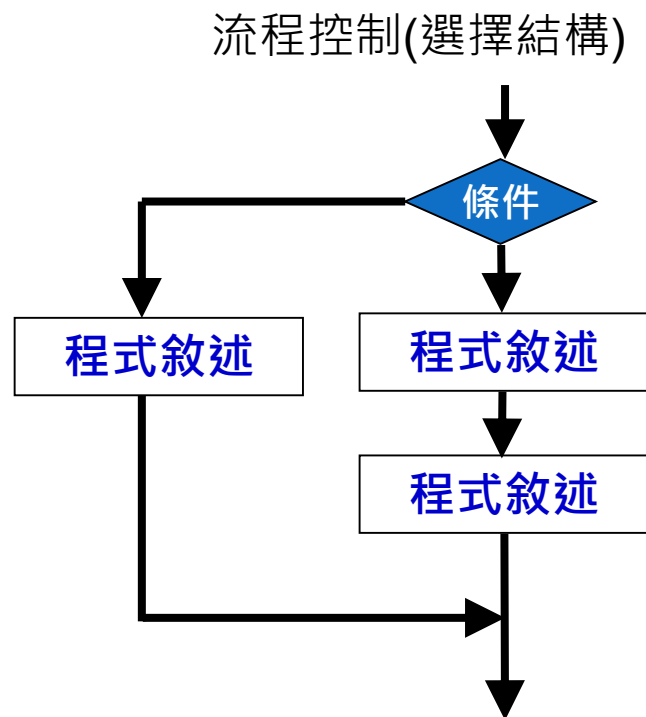
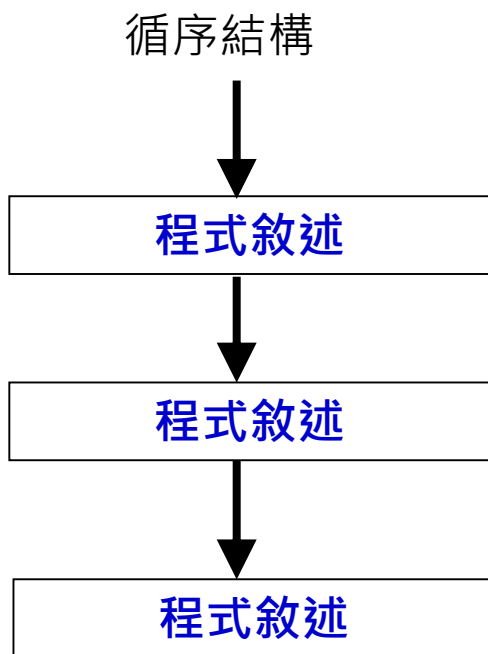


Python Fundamental



# 流程控制簡介

- ◆ 單純的程式執行流程，都是從第一行開始，往下依序一行一行的執行，不會跳行、轉彎。
  - ◆ 然而，實際上大多數的程式執行流程，可能需要針對不同的情況下，進行不同的處理，以便完成更複雜的工作。
- 也因此，就需要流程控制 (Flow Control) 來協助程式的進行。





# 流程控制類型

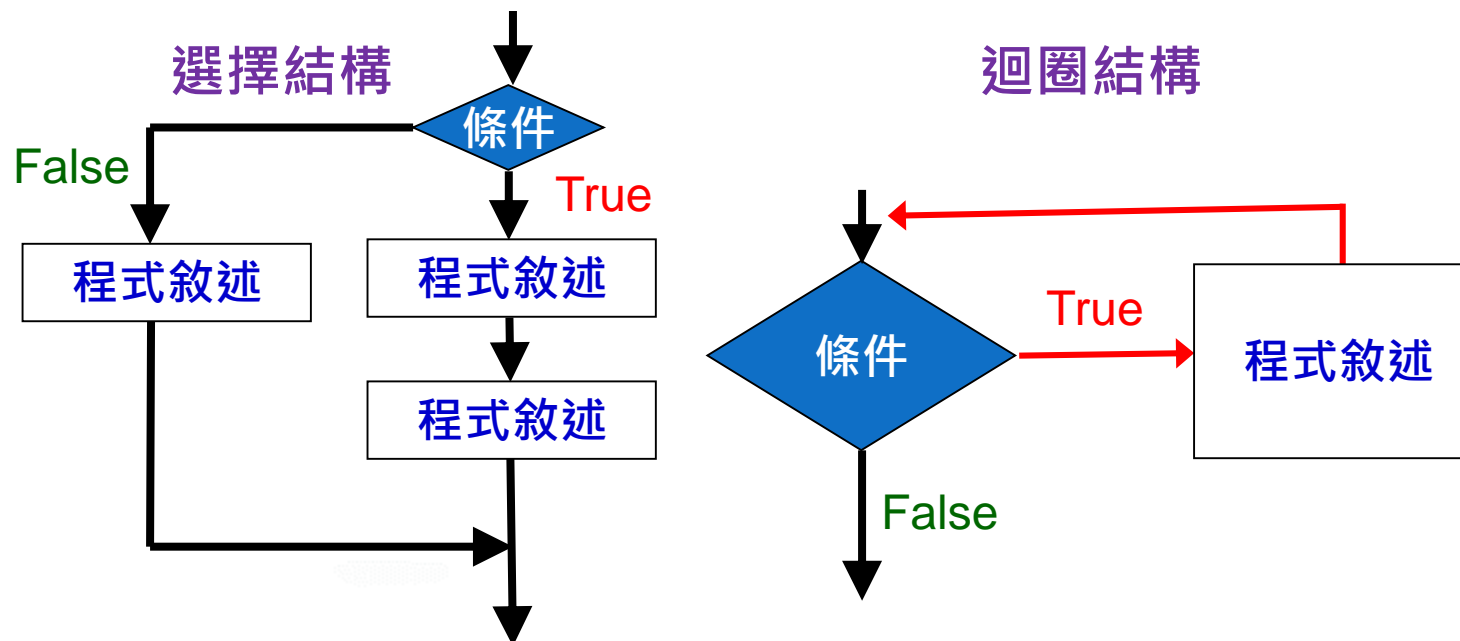
◆ Python 的流程控制分為下列兩種類型：

➤ 選擇/條件結構 ( **Select structure** )：

根據條件的結果為 True 或 False 來執行不同的程式敘述。

➤ 迴圈結構 ( **Loop structure** )：

根據條件的結果為 True 或 False 來重複執行某些敘述，直到滿足特定條件才停止。



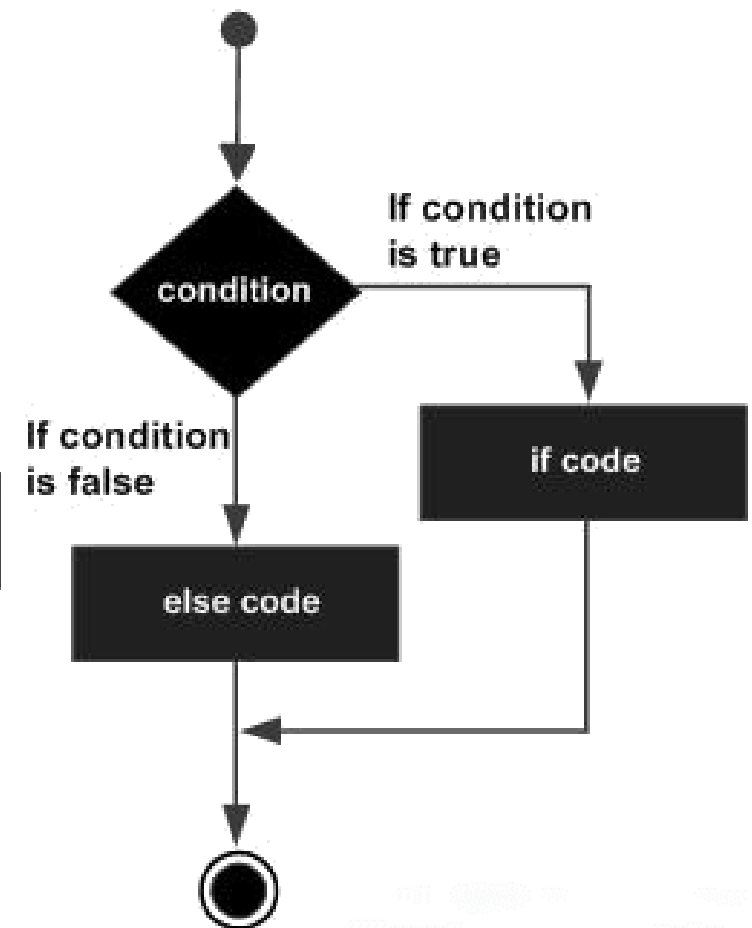
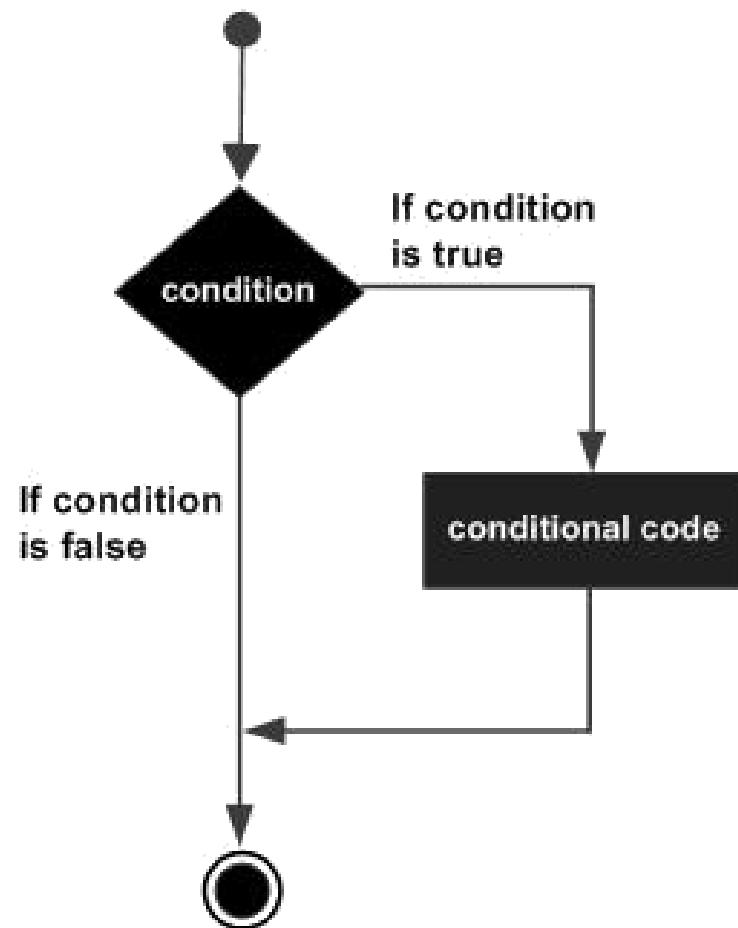
# 選擇結構



# 選擇結構類型

◆ 首先我們先談選擇結構。選擇結構可以分成以下幾種類型：

- 「單向 if」
- 「雙向 if ... else」
- 「多向 if...elif...else」
- 「巢狀 if」





# 選擇結構

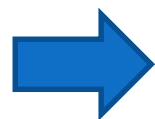
- ◆ 選擇結構可以用來檢查條件式，根據條件的結果為 **True** 或 **False** 而執行不同的敘述。
- ◆ 程式區塊使用冒號「:」開頭，之後同一個區塊範圍都要有相同的縮排。
- ◆ 不可混用不同空白數量，也不可混用空白與 Tab。

```
a=100
```

```
b=200
```

```
if a<b:
```

```
    print(a)
```



100

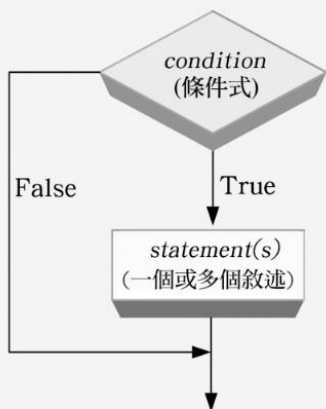


# if - 單向 if (如果是真的...就做...)

- ◆ 「if...」是單向選擇的語法，意思是「如果是真的(True)，就做縮排裡面的敘述.....」。
- ◆ 若條件式成立 (True)，就執行縮排裡面的敘述；若條件式不成立 (False)，就不執行縮排裡面的敘述。
- ◆ statements(s) 必須向右縮排至少一個空白 (預設是四個)；縮排要對齊，表示這些敘述是在區塊內。
- ◆ 由於 Python 使用縮排來劃分程式的執行區塊，因此程式不能隨意的縮排。

if condition :

statement(s)



```
x = 15
```

```
y = 10
```

```
if x > y:
```

```
    z = x - y
```

```
    print("x比y大", z)
```

#判斷 2的10次方是否等於1024

```
if 2**10 == 1024:
    print("2**10 == 1024")
```

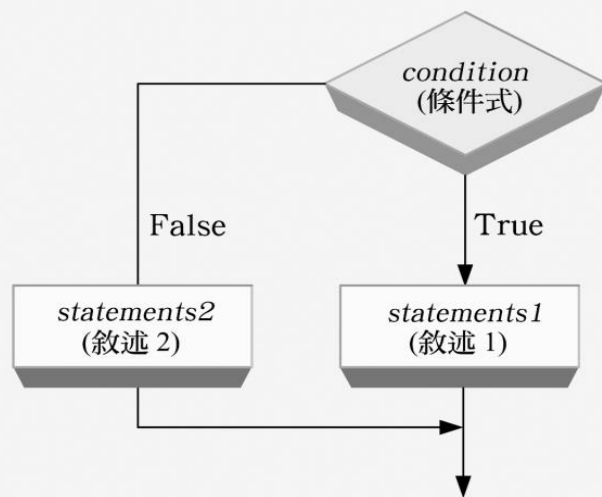


# if...else - 雙向 if (2選1)

- ◆ 「if...else」是雙向選擇的語法，意思是「如果是真的(True)，...就執行 if 區塊的內容，如果不是真的(False)，就執行 else 區塊的內容。

- 若條件式成立，就執行statements1，不執行statements2。
- 若條件式不成立，就執行statements2，不執行statements1。

```
if condition:  
    statements1  
else:  
    statements2
```



```
score = eval(input("請輸入數學分數 (0 ~ 100) : "))  
if score >= 60:  
    print("及格！")  
else:  
    print("不及格！")
```

請輸入數學分數 (0 ~ 100) : 85  
及格！

請輸入數學分數 (0 ~ 100) : 55  
不及格！

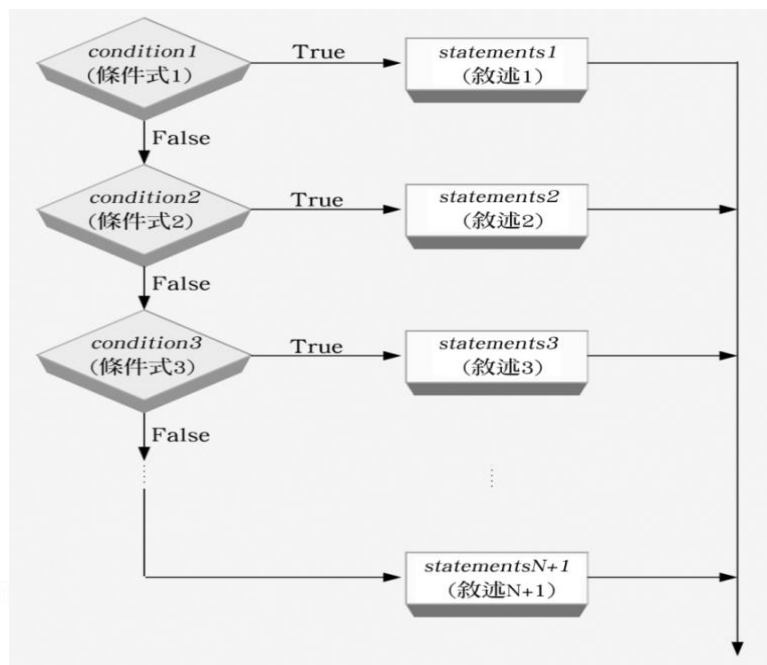




# if...elif...else - 多向 if (多選1)

- ◆ 「if...elif...else」是多向選擇的語法，意思是「如果第一個是真的，就執行敘述；如果不是，那如果下一個是真的...就執行敘述」，依此類推。
- ◆ elif 關鍵字是 else if 的簡寫，可以沒有、一個或多個；相較之下，else 可以沒有或只能有一個，而且一定是放置在最後。
- ◆ 因為是多選 1，所以 statements1~~~ statmentsN+1 只有一組會被執行。

```
if condition1:  
    statements1  
elif condition2:  
    statements2  
elif condition3:  
    statements3  
...  
else:  
    statementsN+1
```





```
print("不及格")
```

不及格



# if - 巢狀 if

- ◆ 巢狀 if 指的是 if 敘述裡面包含其它的 if 敘述，而且沒有深度的限制。
- ◆ 因為沒有深度的限制，因此縮排層級一定要正確，才不會發生錯誤。
- ◆ 要避免深度過深，造成許多縮排，導致不易閱讀，建議還是採取多向 if...elif...else 的方式處理。

```
score = eval(input("請輸入數學分數 (0 ~ 100) : "))
if score >= 90:
    print("優等")
else:
    if score >= 80:
        print("甲等")
    else:
        if score >= 70:
            print("乙等")
        else:
            if score >= 60:
                print("丙等")
            else:
                print("不及格")
```

請輸入數學分數 (0 ~ 100) : 80  
甲等

請輸入數學分數 (0 ~ 100) : 60  
丙等

請輸入數學分數 (0 ~ 100) : 40  
不及格

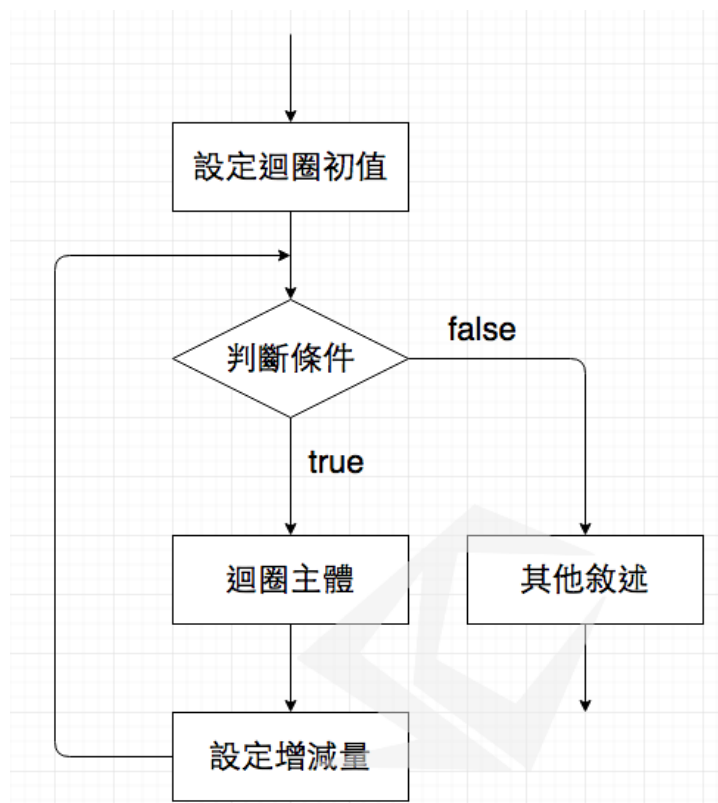
# 迴圈結構



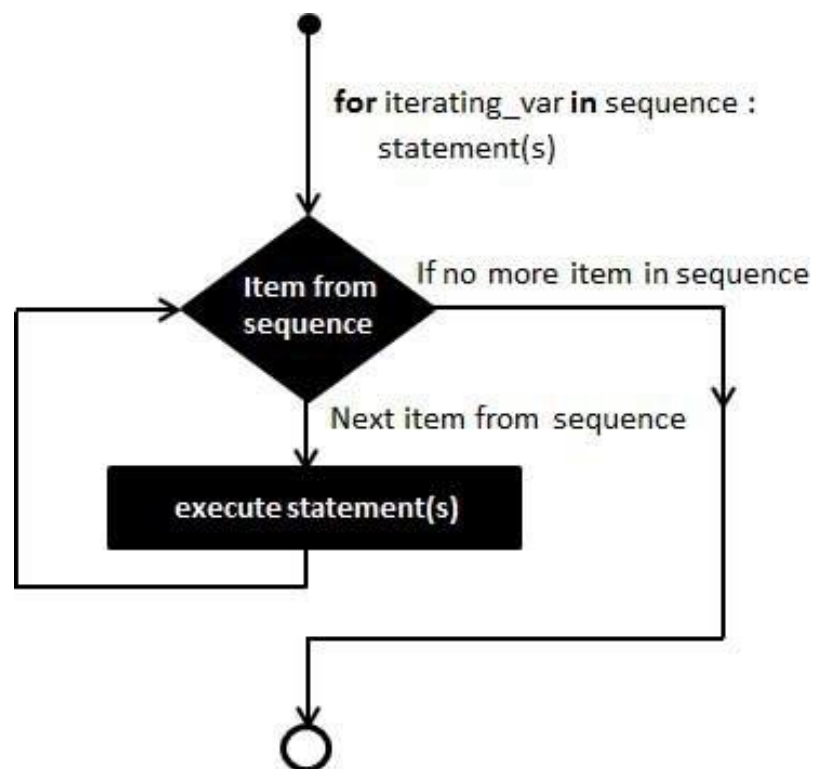
# 迴圈結構使用時機

- ◆ 當重複的次數難以清楚計算時 → 使用 `while` 迴圈。
- ◆ 當重複的次數可以清楚計算時 → 使用 `for` 迴圈。

## while 迴圈



## for 迴圈



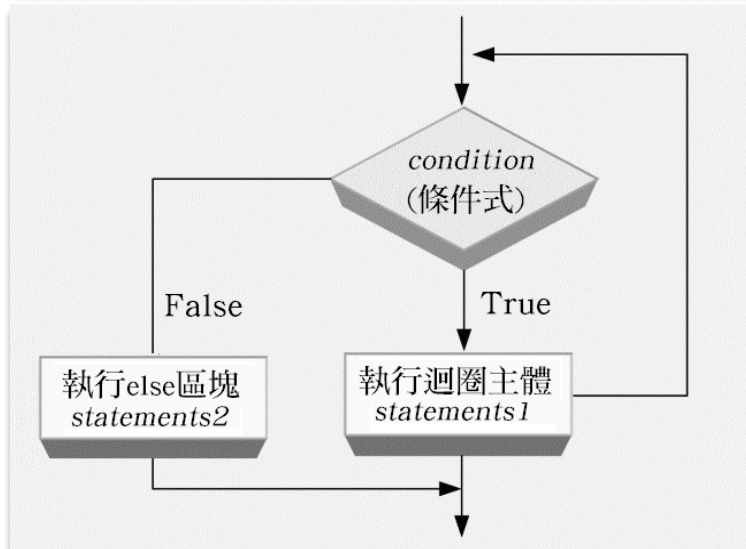


# while 迴圈

◆ 當重複的次數難以清楚計算時使用 `while` 迴圈。

◆ `while` 的語法如下。

```
while condition:  
    statements1  
[else:  
    statements2]
```



◆ 例如：

```
i = 0  
while i < 5:  
    print(i)  
    i = i + 1
```

0  
1  
2  
3  
4



# while 迴圈

```
answer = input("請輸入「快樂」的英文：")  
  
while answer.upper() != "HAPPY":  
    answer = input("答錯了，請重新輸入「快樂」的英文：")  
else:  
    print("答對了！")
```

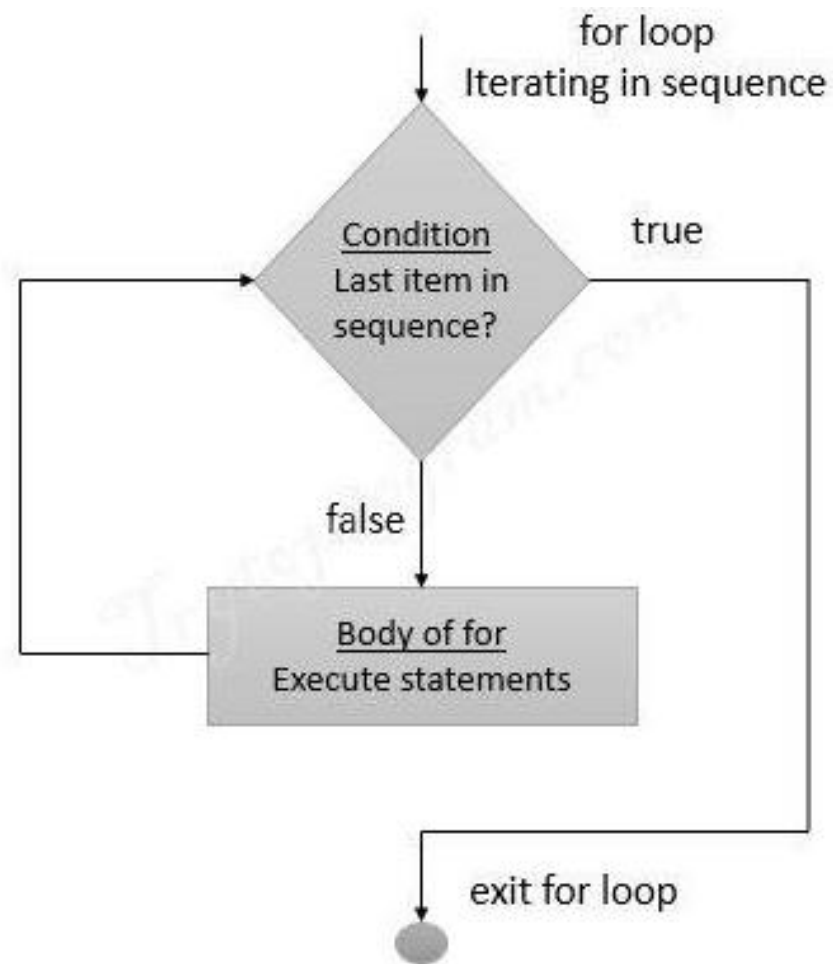
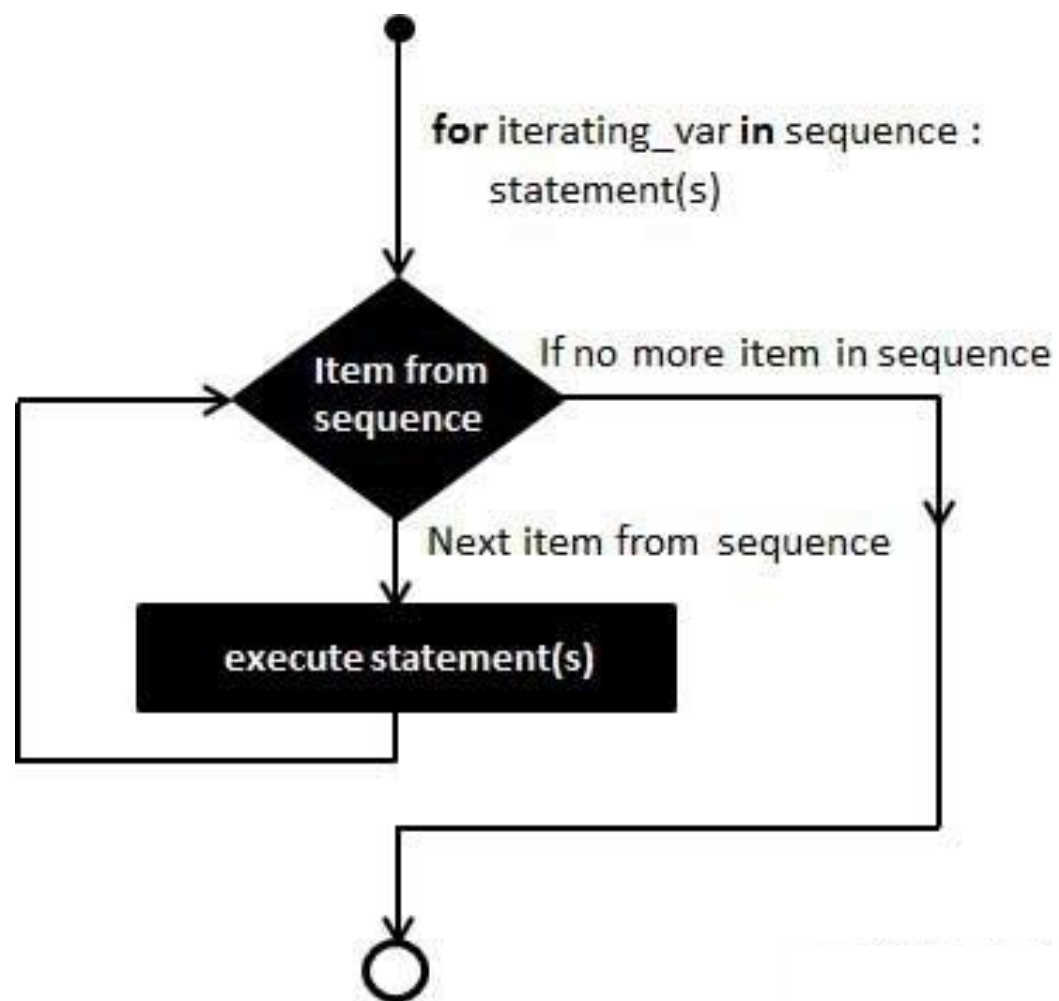
請輸入「快樂」的英文：mad

答錯了，請重新輸入「快樂」的英文：happy  
答對了！



# for 迴圈

- ◆ 當重複的次數可以清楚計算時使用 for 迴圈，所以 for 迴圈又稱為「計數迴圈」。







# for 迴圈

- ◆ for 迴圈語法如下，`iterator` 是有順序、可反覆運算的物件，例如 `range()` 函式或字串、串列等有順序的序列。

```
for var in iterator:  
    statements
```



```
range(stop)  
range(start, stop [, step])
```

- ◆ 迴圈主體 `statements1` 必須以 `for` 關鍵字為基準向右縮排，表示在 `for` 區塊內。
- ◆ `else` 子句為選擇性敘述，可以設定或省略。



# for 迴圈 - range() 函式應用

◆ 我們可以使用 Python 內建的 range() 函式來產生 range 物件：

1. range(stop)
2. range(start, stop)
3. range(start, stop, step)

# 起始值為0，終止值為5 (不含)，間隔值為1的整數序列

```
list(range(5))
```

```
[0, 1, 2, 3, 4]
```

# 起始值為1，終止值為10 (不含)，間隔值為1的整數序列

```
list(range(1, 10))
```

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

# 起始值為10，終止值為-10 (不含)，間隔值為-2的整數序列

```
list(range(10, -10, -2))
```

```
[10, 8, 6, 4, 2, 0, -2, -4, -6, -8]
```



# for 迴圈 - range() 函式應用

# 當 i 尚未等於終止值5時，就印出i；當i等於終止值5時，就跳出迴圈。

```
for i in range(5):  
    print(i)
```

0  
1  
2  
3  
4

```
name='Bob'  
  
for i in range(len(name)):  
    print(i, name[i])
```

0 B  
1 o  
2 b

```
name='Bob'  
  
for i in name:  
    print(i)
```

B  
o  
b



# break & continue

- ◆ 若在迴圈中有需要檢查其它的條件，一旦符合該條件，就強制離開迴圈，此時可以使用 **break** 敘述。
- ◆ **break** 敘述可強制離開迴圈，通常配合條件判斷使用。

```
answer = input("請輸入「好」的英文：")

while answer.upper() != "GOOD":
    if answer.upper() == "QUIT":
        print("我不玩了！")
        break
    print("Bye ! ")
    answer = input("答錯了，請重新輸入「好」的英文：")
else:
    print("答對了！")
```

請輸入「好」的英文：bad

答錯了，請重新輸入「好」的英文：quit  
我不玩了！



# break & continue

- ◆ 除了break敘述，Python還提供了另一個經常使用於迴圈的continue敘述。
- ◆ continue 敘述用來在迴圈內跳過後面的敘述，直接返回迴圈的開頭。

```
i = 0
while i < 10:
    i = i + 1
    if i % 3 != 0:
        continue
    i = i + 2
    print(i)
```

3

6

9

Q & A