迴圈與疊代

蔡尚融

2018-03-05

迴圈與疊代

- 迴圈（loop）是一個特定程式區塊（結合特定指令與一組陳述），重複執行相同工作。
- 無窮迴圈（infinite loop）：重複執行但不終止的迴圈。
- for 迴圈，while 迴圈
- break 終止迴圈執行。
- contiune 跳過此次執行，直接進行下一輪。
- 迴圈中的陳述可為迴圈或判斷式，反之亦然。

for 迴圈

- 處理固定數量的變數
- 語法：（半形冒號結束，陳述要四格半形空白縮排。）

```
for 變量 in 串列:
    陳述一
    陳述二
    ...
>>> fruit = ['apple', 'banana', 'cherry', 'guava',
'melon', 'orange']
>>> for item in fruit:
...     print(item)
...
```

- 用 range() 函式，產出一組整數序列。

  用法一：`range(stop)`

  產生 0 到小於 stop 的整數。

  用法二：`range(start, stop[, step])` 產生從 start 開始到小於 stop ，相鄰差距為 step 的整數

- 用 list() 函式轉成串列。

```
>>> list(range(5))
>>> list(range(3, 10, 2))
[3, 5, 7, 9]
```

實作範例：

計算 3 + 5 + 7 + 9 。

```
#!/usr/bin/env python
y = 0
for x in range(3, 10, 2):
    y = y + x
print(y)
```

while 迴圈

- 語法：（半形冒號結束，陳述要四格半形空白縮排。）

```
while 測試條件:
    陳述一
    陳述二
    ...
```

- 計算 3 + 5 + 7 + 9 。

```
#!/usr/bin/env python
x = 3
y = 0
while x < 10:
    y = y + x
    x = x + 2
print(y)
```

實作範例：巢狀迴圈

印出如下結果：

```
1
22
333
4444
```

方法一：用 for 迴圈。

```
#!/usr/bin/env python
for x in range(1, 5):
    for y in range(1, x + 1):
        print(x, end = "")
    print("")
```

方法二：用 while 迴圈。

```
#!/usr/bin/env python
x = 1
while x < 5:
    y = 1
    while y < x + 1:
        print(x, end = "")
        y = y + 1
    print("")
    x = x + 1
```

隨堂練習：

用 `for` 或 `while` 印出如下結果：

```
12345
1234
123
12
1
```

（提示：`range()` 函式中 `step` 的值，或嘗試 `x = x - 1`。）

實作範例：與判斷式結合

列出 1 到 100 中非偶數但為 3 的倍數，並計算個數。

```
#!/usr/bin/env python
x = 1
n = 0
while x <= 100:
    if x % 2 != 0 and x % 3 == 0:
        print(x)
        n = n + 1
    x = x + 1
print("Amount:", n)
```

隨堂練習：

列出費波那契數列（Fibonacci sequence），已知第一項與第二項為 1 ，從第三項開始定義為其前兩項合，列出前十五項數字。（1, 1, 2, 3, 5, ...）

```
1
1
2 = 1 + 1
3 = 1 + 2
5 = 2 + 3
8 = 3 + 5
```

第一項與第二項用指派（assign）給值，並設定給前兩項暫存變數，由第三項起各項，用前兩項暫存變計算，計算完成後，更新前兩項暫存變。

```
#!/usr/bin/env python
pn1 = 1
pn2 = 1
for x in range(15):
    if x == 0 or x == 1:
        fn = 1
    else:
        fn = pn1 + pn2
        pn1 = fn2
        pn2 = fn
    print(fn)
```

隨堂練習：

輾轉相除法（Euclidean algorithm）找 279 與 345 兩數的最大公因數。考慮將數字寫成商數乘除數加餘數的形式。

$$
\begin{aligned}
279 &= 0 \times 345 + 279 \\
345 &= 1 \times 279 + 66 \\
279 &= 4 \times 66 + 15 \\
66 &= 4 \times 15 + 6 \\
15 &= 2 \times 6 + 3 \\
6 &= 2 \times 3 + 0
\end{aligned}
$$

最大公因數為 3。

若餘數為 0 時，停止迴圈，此時除數為最大公因數。 若餘數為非 0 時，用餘數取代除數。

```
#!/usr/bin/env python
numa = 345
numb = 279
while numb != 0:
    tmp = numb
    numb = numa % numb
    numa = tmp
print(tmp)
```

實作範例：捨入誤差

計算 1.0×10^9 加上 1,000,000 個 1.0×10^{-9} 的值，

1. 對大數直接累加；
2. 先累加小數再加大數。

比較捨入誤差（roundoff error）對計算結果的影響。

對大數直接累加

```
#!/usr/bin/env python
n = 1000000
i = 1
x = 1.0e+9
y = 1.0e-9
z = x
while i < n:
    z = z + y
    i = i + 1

print(z)
if x == z:
    print("true")
else:
    print("false")
```

先累加小數再加大數

```
#!/usr/bin/env python
n = 1000000
i = 1
x = 1.0e+9
y = 1.0e-9
z = 0
while i < n:
    z = z + y
    i = i + 1
z = z + x
print(z)
if x == z:
    print("true")
else:
    print("false")
```

隨堂練習：黎曼和（Riemann sum）

計算函數 $f(x) = x^3 - 4x^2 + 9x$ ，在閉區間 $[a_0, a_n] = [3.1, 4.1]$ ，分割 $n = 100$ 等分的黎曼和。

每一等分長：`delta = (a_n - a_0) / n`
第 `i` 個分割左端點：`x = a_0 + i * delta`
矩形的面積：`(x ** 3 - 4 * x ** 2 + 9 * x) * delta`
計算每一分割並累加。

```
#!/usr/bin/env python
a_0 = 3.1
a_n = 4.1
n = 100
delta = (a_n - a_0) / n
y = 0
x = a_0
i = 0
while i < n:
    y = y + (x ** 3 - 4 * x ** 2 + 9 * x) * delta
    i = i + 1
    x = a_0 + i * delta
print(y)
```