函式與遞迴

蔡尚融

2018-03-26

函式

- 函式（function），由一些循序的陳述式組成，用以執行特定運算的程式片段。
- 函式名稱（function name）。
- 函式呼叫（function calls）

```
>>> str1 = 'This is a string.'
>>> vartyp = type(str1)
```

- 輸入（input），引數（argument）。
- 輸出（output），回傳值（return value）。

為何要用函式？

- 提供開發者對特定一組陳述式的命名，用以表達此程式區塊的用途或成果與目的。
- 減少重複陳述式編寫，減輕程式的維護成本。
- 方便程式碼的閱讀與分段除錯。
- 亦可透過適當安排組合出新功能。
- 設計良好的函式，較易於移植到其他程式，有利於重新利用。

函式定義

- 函式定義（function defintion）：
  給定名稱與循序的陳述式，表明執行時的運作。

```
def show_test():
    print("A function shows a test string.")
    print("Test.")
```

- 第一行稱為頭行（header），以半形冒號（:）結束。
- 其餘的是函式的身體（body），必須縮排建議以四個半形空白（space）。
- 函式定義完成後會產生函式物件（function object）。

引數與參數

- 引數（argument），用以輸入提供函式執行用的資料（Python 物件，包含數值與函式）。

```
>>> math.cos(1.1)
```

此處 1.1 是函式引數。

- 參數（parameter），函式括號中儲存輸入資料的變數。

```
def cook(food):
    print(food)
```

- 函式的參數與變數是區域變數，只在函式身體裡可用。

```
def cook(food):
    cook_food = 'cook ' + food
    print(cook_food)
```

函數回傳值

- 函數回傳用 return 關鍵字作開始，後接變數或表達式。
- 若無撰寫 return 回傳，預設會回傳 NoneType 物件。
- 可以有多個回傳值，用逗號分隔。

```
#/usr/bin/env python

def multireturn():
    foo = 1
    bar = 2
    return foo, var

rv1, rv2 = multireturn()
print(rv1)
print(rv2)
```

範例：

找 279 與 345 兩數的最大公因數與最小公倍數。

```
#!/usr/bin/env python

def gcd(numa, numb):
    while numb != 0:
        tmp = numb
        numb = numa % numb
        numa = tmp
    return tmp
```

```
def lcm(numa, numb):
    gcd_ab = gcd(numa, numb)
    qa = numa // gcd_ab
    qb = numb // gcd_ab
    return gcd_ab * qa * qb

print(gcd(345, 279))
print(lcm(345, 279))
```

隨堂練習：

將之前介紹的牛頓法改寫成函數，輸入項有計算 $f(x)$ 的函式、計算 $f'(x)$ 的函式，起始值、最大的疊代次數，殘值絕對值的要求；輸出項目為數值解（近似值）、實際疊代的次數、殘值絕對值。

以 $f(x) = x^3 - 5$ 為測試例子，起始值 $x_0 = 2$。

```
def newton_method(fn, df, x0, maxiter, atol):
    ...
    return sol, iter, fval
```

數學模組物件與函式

- Python 有提供一個處理基本數學相關的模組物件 `math`，以 `import` 指令載入。

```
>>> import math
```

- 三角函數與反三角函數。

```
math.sin(), math.cos(), ..., math.asin(), ...
```

- 指數與對數。

```
math.exp(), math.log(), math.log2(), ...
```

- 其他： `math.pow()`, `math.floor()`, ...
- 預先定義的常數：

```
e = 2.718281828459045     pi = 3.141592653589793
inf = inf                 nan = nan
```

遞迴（Recursive）

- 將問題重複分解為同類型的子問題，再解決問題的方法。
- 資料是按遞迴定義，如費氏數列；1, 1, 2, 3, 5, 8, ...。
- 遞迴演算法；河內塔（Tower of Hanoi）問題。
- 優點：程式相對簡潔明確。
- 缺點：消耗較多計算資源，花費長時間執行。

遞迴的設計

- 函式的自我呼叫（call）。

```
def testfunc(arg1, ...):
    ...
    testfunc(...)
    ...
```

- 函式停止自我呼叫的條件。
- 函式的輸入值。

範例：

列出費波那契數列（Fibonacci sequence），已知第零項為 0，一、二項為 1，從第三項起定義為其前兩項和，列出第十五項。

```
#!/usr/bin/env python

def fibonacci(item):
    if item == 0:
        return 0
    elif item == 1 or item == 2:
        return 1
    else:
        return fibonacci(item - 2) + \
            fibonacci(item - 1)

print(fibonacci(15))
```

範例：組合數

若正整數 n 大於正整數 $k \geq 0$，則從 n 個物品中任取 k 個物品的組合記為 C_k^n，計算公式如下：

$$C_k^n = \frac{n!}{k!(n-k)!}$$

另一個常見的遞迴公式為：

$$C_k^n = C_{k-1}^{n-1} + C_k^{n-1}$$

計算 C_5^{20} 的值。

```
#!/usr/bin/env python

def choose(numn, numk):
    if numk > numn:
        return 0
    elif numk == 0 or numk == numn:
        return 1
    else:
        return choose(numn - 1, numk - 1) + \
            choose(numn - 1, numk)

print(choose(20, 5))
```

隨堂練習：

用遞迴計算 $2+3+4+5+6+7+8+9$，函數名稱與輸入為 `compute_sum(head, tail)`，其中 `head = 2`，`tail = 9`。

匿名函式

- 以關鍵字 `lambda` 建立匿名函數（anonymous function）。
- 語法：`lambda` 引數一, 引數二, ... : 表達式

```
>>> amount = lambda numa, numb: numa + numb
>>> amount(10, 15)
25
```

- 匿名函數表達式中不能有區塊，適用於簡單的運算任務。

範例：

作為函式輸入引數。

```
#!/usr/bin/env python

def compare(rfnc, numa, numb):
    if rfnc(numa, numb):
        return numa
    else:
        return numb

print(compare(lambda na, nb: na > nb, 15, 17))
print(compare(lambda na, nb: na < nb, 15, 17))
```