

Python Programming

2023 Spring; week 15

Instructor: Cheng-Chun Chang (張正春)
Department of Electrical Engineering

Textbook: Python程式設計:從入門到進階應用(第三版) 2020

課程助教

協助 **dubug**, 禁止同學看 **code**照抄

第一排:

第二排:

第三排:

第四排:

第五排:

第六排:

打游擊:

Ch6 函式與遞迴

6-1 函式

- . 函式用於結構化程式
- . 將相同功能的程式獨立出來，經由函式的呼叫，傳入資料與回傳處理後的結果
- . 程式設計師只要將函式寫好，就可以不斷利用此函式做相同動作，同時可使程式碼不重複，而若要修改此功能，只要更改此函式。

6-1-1 函式的定義、傳回值與呼叫

- . 自訂函式需要包含兩個部分，分別是「**函式的定義**」與「**函式的呼叫**」。
 - . 「**函式的定義**」是實作函式的功能，輸入參數與回傳處理後的結果
 - . 「**函式的呼叫**」是其他程式中呼叫自訂函式，讓自訂函式真正執行

6-1-1 函式的定義、傳回值與呼叫

函式的定義

- 以def 開頭，空一個空白字元(space)，接函式名稱後，串接著一對小括號，小括號可以填入要傳入函式的參數
- 當參數有多個的時候以逗號隔開，右小括號後面須接上「:」，函式範圍以縮行固定個數的空白字元表示，縮行相同個數的空白字元的程式碼就是函式的作用範圍。

6-1-1 函式的定義、傳回值與呼叫

當函式需要傳回值使用指令`return`，表示函式回傳資料給原呼叫函式，若不需要回傳值的函式就不需要加上`return`，函式的定義與傳回值格式，如下表。


表 6-1 函式的定義

分類	函式的定義語法	範例
不回傳值的函式	<code>def</code> 函式名稱 (參數 1 , 參數 2 , ...): 函式的敘述區塊	<pre>def hi(): print('hi')</pre>
回傳值的函式	<code>def</code> 函式名稱 (參數 1 , 參數 2 , ...): 函式的敘述區塊 <code>return</code> 要傳回的變數或值	<pre>def min(a,b): if a > b: return b else: return a</pre>

Programming codes I




6-1-1 函式的定義、傳回值與呼叫

綜合前面敘述，函式定義與函式呼叫範例 ( : ch6\6-1-1-func1.py)，如下。

行號	範例	執行結果
1	def hi():	hi 2
2	print("hi")	
3	hi()	
4	def min(a,b):	
5	if a > b:	
6	return b	
7	else:	
8	return a	
9	print(min(2,4))	

6-1-1 函式的定義、傳回值與呼叫

- 以下範例用於計算長方形面積，使用者輸入長度與寬度，呼叫自訂的函式`area` 將長度與寬度傳入，回傳計算結果。

行號	函式範例程式 ( : ch6\ 6-1-1-rec.py)	執行結果
1	<code>def area(x,y):</code>	請輸入長度？ 3 請輸入寬度？ 4 長方形面積為 12
2	<code> return x*y</code>	
3	<code>a = int(input(' 請輸入長度？ '))</code>	
4	<code>b = int(input(' 請輸入寬度？ '))</code>	
5	<code>ans = area(a, b)</code>	
6	<code>print(' 長方形面積為 ', ans)</code>	

□ 函式呼叫過程的流程圖



圖 6-1 函式呼叫過程的流程圖

Programming codes 2



6-1-2 函式與變數的作用範圍

變數作用範圍分成全域變數與函式內的區域變數，宣告在最上面最外層的稱作**全域變數**，宣告在函式內的變數稱作**區域變數**，函式內若沒有那個變數就會往函式外找尋，舉例如以下範例(ch6\6-1-2a-func2.py)。

行號	範例	執行結果
1	g = 5	5
2	def f1():	
3	print(g)	
4	f1()	

6-1-2 函式與變數的作用範圍

行號	範例	執行結果
1	<code>g = 5</code>	10 5
2	<code>def f2():</code>	
3	<code> #print(g)</code>	
4	<code> g = 10</code>	
5	<code> print(g)</code>	
6	<code>f2()</code>	
7	<code>print(g)</code>	

□ 舉例如以下範例(ch6\6-1-2b-func3.py)。

行號	範例	執行結果
1	g = 5	5 10 10
2	def f():	
3	global g	
4	print(g)	
5	g = 10	
6	print(g)	
7	f()	
8	print(g)	

-
- 全域變數g 與區域變數g, 是兩個不同的變數
 - 函式內區域變數g 作用範圍在函式內, 全域變數g 作用範圍為整個檔案, 但因為函式內區域變數有相同的變數名稱, 函式會優先使用區域變數, 若找不到才去找全域變數。


Programming codes 3



6-3 函式的輸入與輸出


6-3-1 函式的輸入

函式中有預設值的輸入參數一定要放在後面，預設值要是不可以變的常數，不能為串列或字典等可以修改的資料結構。


行號	範例 ( : ch6\6-3-1-func4.py)	執行結果
1 2 3	<pre>def f(s, count=1): print(s * count) f('Hi')</pre>	Hi

6-3-1 函式的輸入


可以經由函式呼叫輸入新的數值取代原預設值，如以下範例。

行號	範例 ( : ch6\6-3-1-func4.py)	執行結果
1	<code>def f(s, count=1):</code>	HiHiHi
2	<code> print(s * count)</code>	
3	<code>f('Hi',3)</code>	

6-3-1 函式的輸入

行號	範例 ( : ch6\6-3-1-func5.py)	執行結果
1	def func(x, y, z=9):	
2	print("x=", x, "y=", y, "z=", z)	
3	func(1, 2)	x= 1 y= 2 z= 9
4	func(1, 2, 3)	x= 1 y= 2 z= 3
5	func(x=3, y=4)	x= 3 y= 4 z= 9
6	func(y=5, x=6)	x= 6 y= 5 z= 9
7	#func(x=3, z=6)	

6-3-2 函式的回傳值

行號	範例 ( : ch6\6-3-2-func6.py)	執行結果
1	<code>from datetime import datetime</code>	2018 4 30
2	<code>def ymd():</code>	
3	<code> now = datetime.now()</code>	
4	<code> return (now.year, now.month, now.day)</code>	
5	<code>y, m, d = ymd()</code>	
6	<code>print(y,m,d)</code>	

6-3-3 函式的進階輸入 — 位置引數與關鍵字引數

綜合上述範例獲得以下程式。

行號	範例 ( : ch6\6-3-3-func7.py)	執行結果
1	def func1(*args):	
2	print(' 位置引數為 ', args)	
3	func1(1,2,3)	
4	def func2(**kwargs):	位置引數為 (1, 2, 3)
5	print(' 關鍵字引數為 ', kwargs)	關鍵字引數為 {'b': 2, 'a': 1}
6	func2(a=1, b=2)	start= 1
7	def func3(start, *args, **kwargs):	位置引數為 (2, 3)
8	print("start=", start)	關鍵字引數為 {'b': 5, 'a': 4}
9	print(" 位置引數為 ", args)	
10	print(" 關鍵字引數為 ", kwargs)	
11	func3(1, 2, 3, a=4, b=5)	


Programming codes 4



6-4 函式的說明文件

- 可以在函式下方使用「`'''`」撰寫函式的說明文件，說明文件可以跨好幾行，直到找到下一個「`'''`」，使用「`'''`」會保留第2 行以後所有開頭的空格，如以下範例。

6-4

行號	範例 ( : ch6\6-4-func8.py)	執行結果
1	def min(a, b):	Help on function min in module __main__
2	""" 使用 min 可以找出 a 與 b 較小的值	min(a, b)
3	Args:	使用 min 可以找出 a 與 b 較小的值
4	a: 輸入的第一個參數	Args:
5	b: 輸入的第二個參數	a: 輸入的第一個參數
6		b: 輸入的第二個參數
7	Returns:	
8	回傳 a 與 b 中較小的值	Returns:
9	"""	回傳 a 與 b 中較小的值
10	if a > b:	
11	return b	使用 min 可以找出 a 與 b 較小的值
12	else:	Args:
13	return a	a: 輸入的第一個參數
14	help(min)	b: 輸入的第二個參數
15	print(min.__doc__)	
		Returns:
		回傳 a 與 b 中較小的值

Programming codes 5




6-5 函式視為物件

- Python 中函式視為物件，以函式名稱當成物件，函式名稱加上() 才會執行該函式，範例如下。

```
def add(a, b):  
    return a + b  
  
def run(func, x, y):  
    return func(x, y)  
  
k = run(add, 10, 20)  
print('k=', k)
```

6-5 函式視為物件


行號	範例 ( : ch6\6-5-func9.py)	執行結果
1	<code>def add(a, b):</code>	k= 30
2	<code> return a + b</code>	
3	<code>def run(func, x, y):</code>	
4	<code> return func(x, y)</code>	
5	<code>k = run(add, 10, 20)</code>	
6	<code>print('k=', k)</code>	


6-6 函式lambda

- 函式若只有一行，可以轉換成為函式lambda，函式lambda 的轉換格式如下。
 - lambda 輸入的參數: 函式的定義

原始函式	轉換為 lambda
<pre>def add(a, b): return a + b</pre>	<pre>lambda a, b: a+b</pre>

6-6 函式lambda

行號	範例 ( : ch6\6-5-func9.py)	執行結果
1	<code>def add(a, b):</code>	k= 30
2	<code> return a + b</code>	
3	<code>def run(func, x, y):</code>	
4	<code> return func(x, y)</code>	
5	<code>k = run(add, 10, 20)</code>	
6	<code>print('k=', k)</code>	

行號	範例 ( : ch6\6-6-func10.py)	執行結果
1	<code>def run(func, x, y):</code>	k= 30
2	<code> return func(x, y)</code>	
3	<code>k = run(lambda a,b: a+b, 10, 20)</code>	
4	<code>print('k=', k)</code>	

Programming codes 6



6-7 產生器(generator)

- 使用函式製作產生器，產生器可以產生一個序列的資料，產生器要使用`yield` 回傳資料，而非使用`return` 回傳資料，使用`yield` 回傳資料會紀錄上一次回傳時函式的狀態，不會從頭到尾都執行。


6-7 產生器(generator)

行號	範例 ( : ch6\6-7-func11.py)	執行結果
1	def irange(start, stop, step=1):	
2	if start < stop:	
3	i = start	
4	while i < stop:	
5	yield i	<generator object irange at
6	i = i + step	0x0000000000110A360>
7	else:	1
8	i = start	2
9	while i > stop:	3
10	yield i	4
11	i = i + step	4
12	x = irange(1,10)	3
13	print(x)	2
14	for i in irange(1, 5, 1):	
15	print(i)	
16	for i in irange(4, 1, -1):	
17	print(i)	


Programming codes 7



6-8 內部函式

行號	範例 ( : ch6\6-8-func12.py)	執行結果
1	def hello(msg):	Hello, John Hello, 你好
2	def say(text):	
3	return 'Hello,'+text	
4	print(say(msg))	
5	print(say(' 你好 '))	
6	hello('John')	

6-9 closure 函式

行號	範例 ( : ch6\6-9-func13.py)	執行結果
1	def hello(msg):	Hello,Claire Hi,Fiona
2	def say(hi):	
3	return hi+msg	
4	return say	
5	x=hello('Claire')	
6	y=hello('Fiona')	
7	print(x('Hello,'))	
8	print(y('Hi,'))	

Programming codes 8



6-11 遞迴

- . 遞迴是有趣的程式設計技巧，也是一種解題策略，函式執行過程中呼叫自己，稱作「遞迴」
- . 利用遞迴函式撰寫程式時，需明確定義遞迴關係，而這樣的自己呼叫自己，需要有終止的條件，若沒有終止的條件就會形成無窮遞迴。

範例6-11-1 求n 階乘(ch6\6-11-1-fac.py)

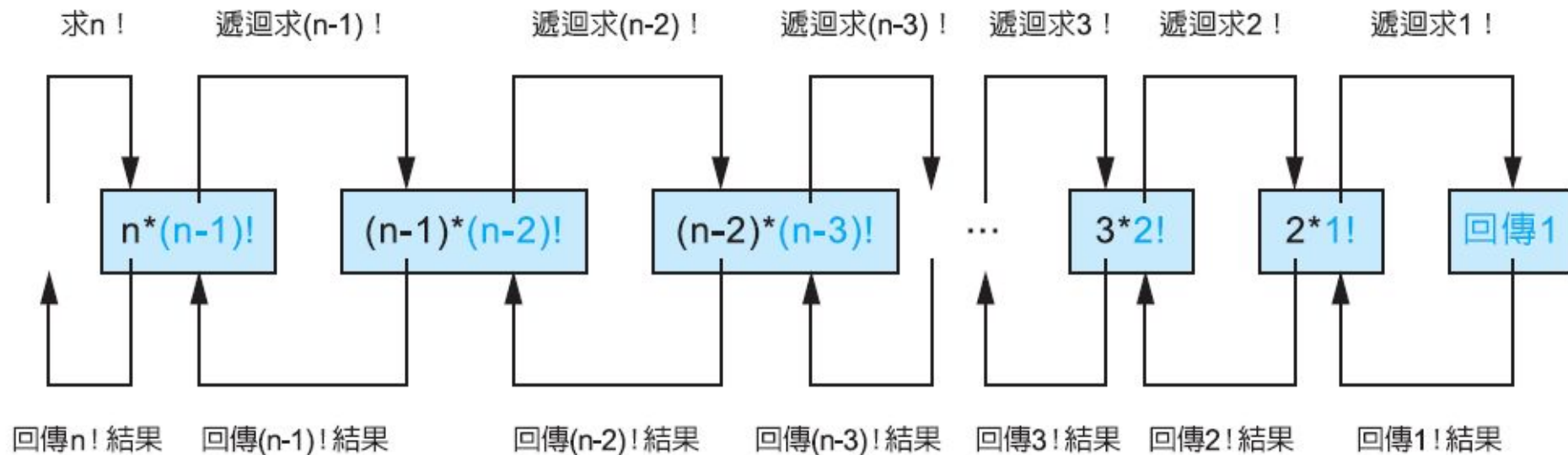



圖 6-2 求 n 階乘示意圖 1

範例6-11-1 求n 階乘(ch6\6-11-1-fac.py)

我們接下來實作求 n 階乘的程式。

行數	程式碼 ( : ch6\6-11-1-fac.py)	執行結果
1	def fac(num):	輸入 n 值，例如 5，程式執行結果如下。 請輸入 n 值？ 5 5! 為 120
2	if num == 1:	
3	return 1	
4	else:	
5	return num*fac(num-1)	
6	n = int(input(' 請輸入 n 值？ '))	
7	ans = fac(n)	
8	print(n,'! 為 ',ans,sep='')	

範例6-11-1 求n 階乘(ch6\6-11-1-fac.py)

使用圖示表示遞迴求解 5 階乘，相當於以 $f(5)$ 執行為例。

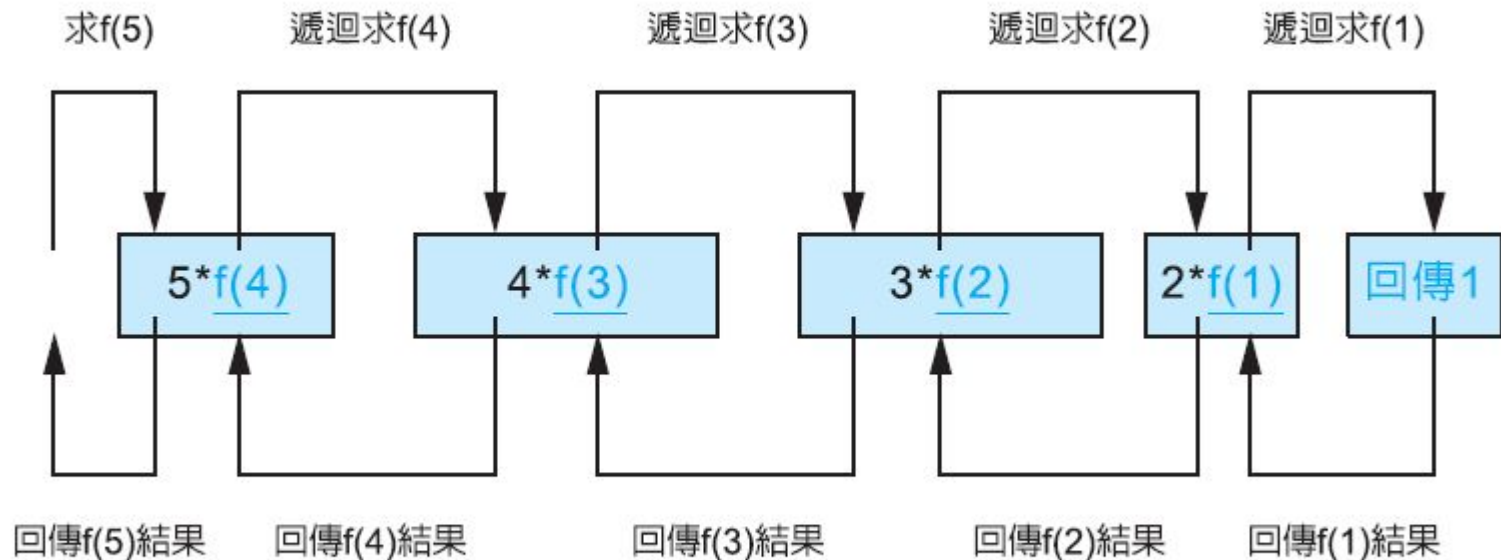


圖 6-4 求 n 階乘示意圖 3

Readings



6-10 Decorator(裝飾器)

- 裝飾器為一種函式，允許輸入一個函式，回傳另一個函式，常用於將一個自訂函式改裝成另一個函式，可以用於顯示除錯訊息，例如定義一個除錯函式如下。

```
def debug(func1):  
    def func2(*args, **kwargs):  
        print(' 正在執行函式 ', func1.__name__)  
        print(' 函式的說明文件爲 ', func1.__doc__)  
        print(' 位置引數 ', args)  
        print(' 關鍵引數 ', kwargs)  
        return func1(*args, **kwargs)  
    return func2
```

-
- 函式debug 允許輸入一個函式func1, 在函式debug 內定義另一個函式func2, 函式func2 接收傳入函式func2 的位置引數args 與關鍵字引數kwargs, 顯示函式func1的函式名稱、說明文件到螢幕上

-
- 接著顯示傳入函式func2 的位置引數與關鍵字引數在螢幕上, 回傳函式func1 以位置引數args 與關鍵字引數kwargs 為輸入的結果, 最後函式debug 執行結束回傳函式func2。

-
- 使用上可以輸入一個函式物件到裝飾器，使用變數接收裝飾器所回傳的函式物件，輸入到裝飾器的函式名稱與接收裝飾器所回傳的函式名稱，可以使用相同函式名稱

-
- 如以下範例都使用`add`，但輸入的函式物件`add` 與回傳的函式物件`add` 是指向不同的函式物件，也就是經由裝飾器可以將函式物件轉換成另一個函式物件。

```
def add(a, b):  
    ' 回傳 a 加 b 的結果 '  
    return a+b  
  
add = debug(add)  
print(add(1, b=2))
```

6-10 Decorator(裝飾器)

上述程式執行結果如下。

正在執行函式 add

函式的說明文件為 回傳 a 加 b 的結果

位置引數 (1,)

關鍵引數 {'b': 2}

3

6-10 Decorator(裝飾器)

- 也可以利用「@」簡化裝飾器的使用步驟, 在需要裝飾器的函式的上一行, 使用「@」串接裝飾器名稱, 就可以達成使用裝飾器改變下一行所定義的函式。

```
@debug
def add(a, b, c):
    ' 回傳 a+b+c 的結果 '
    return a+b+c
print(add(1, 2, c=3))
```

. 程式執行結果如下

正在執行函式 add

函式的說明文件為 回傳 $a+b+c$ 的結果

位置引數 (1, 2)

關鍵引數 {'c': 3}

6

6-10 Decorator(裝飾器)

行號	範例 ( : ch6\6-10-func14.py)	執行結果
1	def debug(func1):	
2	def func2(*args, **kwargs):	
3	print(' 正在執行函式 ', func1.__name__)	正在執行函式 add
4	print(' 函式的說明文件爲 ', func1.__doc__)	函式的說明文件爲 回傳 a 加 b
5	print(' 位置引數 ', args)	的結果
6	print(' 關鍵引數 ', kwargs)	位置引數 (1,)
7	return func1(*args, **kwargs)	關鍵引數 {'b': 2}
8	return func2	3
9	def add(a, b):	正在執行函式 add
10	' 回傳 a 加 b 的結果 '	函式的說明文件爲 回傳 a+b+c
11	return a+b	的結果
12	add = debug(add)	位置引數 (1, 2)
13	print(add(1, b=2))	關鍵引數 {'c': 3}
14	@debug	6
15	def add(a, b, c):	
16	' 回傳 a+b+c 的結果 '	
17	return a+b+c	
18	print(add(1, 2, c=3))	

□ Exercise



範例6-2-2 求質數

(ch6\6-2-2-prime.py)

- 某數的因數只有1 與自己, 沒有其他因數, 稱為質數, 寫一個程式列出 2 到 X 所有質數。

解題想法

- . 自訂判斷質數的函式，輸入一個數字，回傳是否為質數，回傳True 表示為質數，回傳False 表示為非質數
- . 接著使用迴圈結構尋找指定範圍的所有質數，將每個數輸入到判斷質數函數，判斷該數是否為質數。印出指定範圍的質數個數、所有質數到螢幕上。

範例6-2-2 求質數 (ch6\6-2-2-prime.py)

```
1 import math
2 prime_list = []
3
4 def prime(num):
5     j = 2
6     while
7
8
9
10     retu
11
12 for i in
13     if p
14
15 print('2
16 print(prime
```

Exercise I
(2 pt)

執行結果： ※ 1. 2.須完整截圖，3.擷取質數個數與部分質數即可

1. 印出 2 到 10 的質數個數與各質數：

```
2 到 10 共有 4 個質數  
[2, 3, 5, 7]
```

2. 印出 2 到 100 的質數個數與各質數：

```
2 到 100 共有 25 個質數  
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,  
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

3. 印出 2 到「學號後 4 碼」的質數個數與各質數：

```
2 到 1001 共有 168 個質數  
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,  
43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 1  
01, 103, 107, 109, 113, 127, 131, 137, 139, 149,  
151, 157, 163, 167, 173, 179, 181, 191, 193, 197,  
199, 211, 223, 227, 229, 233, 239, 241, 251, 257]
```

假設學號後 4
碼為 1001

範例6-11-2 求最大公因數 (ch6\6-11-2-gcd.py)

- 求 m 與 n 的最大公因數，數學上可以使用輾轉相除法求解，其原理為 m 與 n 的最大公因數相當於求解「 n 除以 m 的餘數」與 m 的最大公因數，這樣一層又一層遞迴下去直到「 n 除以 m 的餘數」等於0，就終止遞迴，再一層又一層往上回推。

□ 以求11 與25 的最大公因數為例

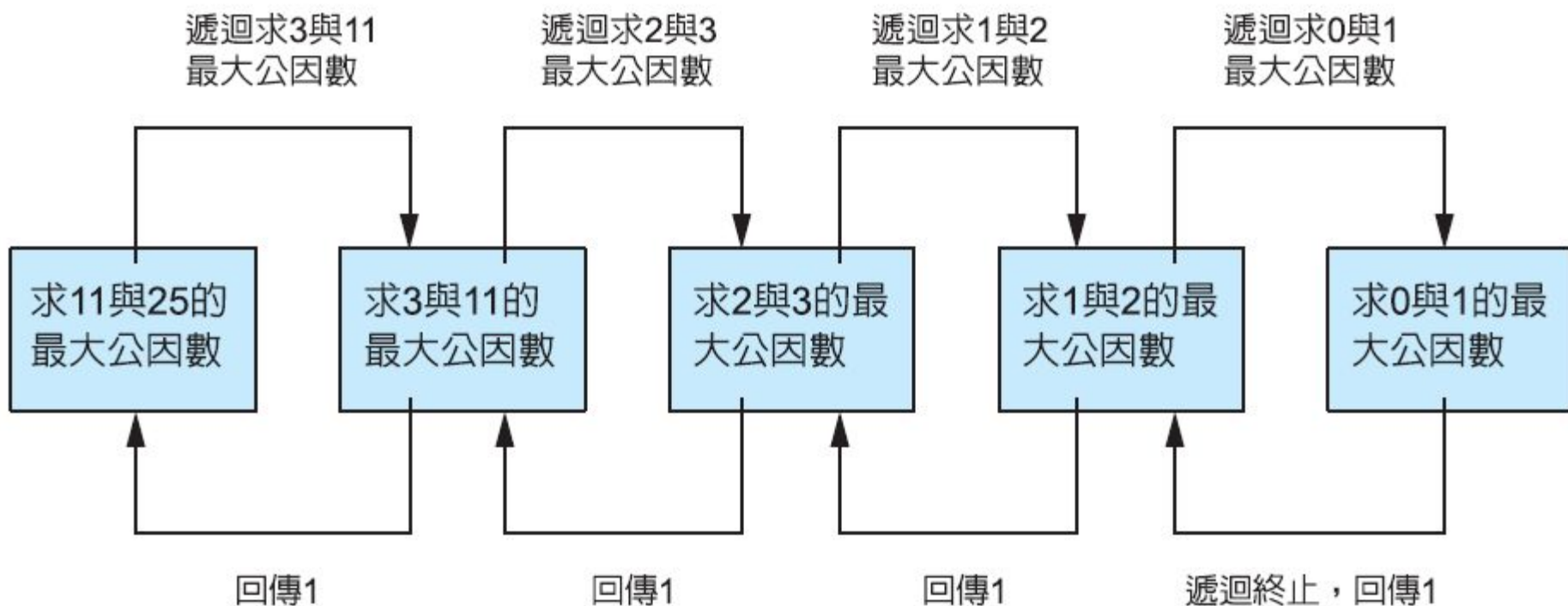


圖 6-5 求最大公因數示意圖

行數	程式碼
1	def gcd(m, n):
2	i
3	
4	
5	
6	return gcd(n % m, m)
7	m = int(input(' 請輸入 m 值? '))
8	n = int(input(' 請輸入 n 值? '))
9	ans = gcd(m, n)
10	print(m, ' 與 ', n, ' 的最大公因數為 ', ans, sep="")

Exercise 2 (2 pt)



WordPad
Document

範例6-11-2 求最大公因數(ch6\6-11-2-gcd.py)

執行結果

輸入 m 值，輸入 n 值，程式執行結果如下。

請輸入 m 值？ 11

請輸入 n 值？ 25

11 與 25 的最大公因數相當於 3 與 11 的最大公因數

3 與 11 的最大公因數相當於 2 與 3 的最大公因數

2 與 3 的最大公因數相當於 1 與 2 的最大公因數

1 與 2 的最大公因數相當於 0 與 1 的最大公因數

11 與 25 的最大公因數為 1

執行結果：

1. $m=10, n=25$:

輸入m值:10

輸入n值:25

10與25的最大公因數相當於5與10的最大公因數

5與10的最大公因數相當於0與5的最大公因數

10與25的最大公因數為5

2. $m=7, n=8$:

輸入m值:7

輸入n值:8

7與8的最大公因數相當於1與7的最大公因數

1與7的最大公因數相當於0與1的最大公因數

7與8的最大公因數為1

3. $m=\text{學號前 3 碼}, n=\text{學號後 3 碼}$:

請輸入 m 值? 109

請輸入 n 值? 001

109與1的最大公因數相當於1與109的最大公因數

1與109的最大公因數相當於0與1的最大公因數

109與1的最大公因數為1



To be continued.....

Instructor: Cheng-Chun Chang (張正春)
Department of Electrical Engineering