06

函 式

本章綱要

- 定義形式參數(formal parameters)的函式(§6.2)
- 以實際參數呼叫函式(§6.3)
- 函式有無回傳值之間的差異(§6.4)
- 使用位置引數或關鍵字引數來呼叫函式(§6.5)
- 以傳送參考來呼叫函式(§6.6)
- 開發可重複使用的程式碼，它是模組並易於閱讀、除錯和維護(§6.7)
- 對重複使用的函式建立模組(§§6.7~6.8)

本章綱要

- 定義變數的範圍(§6.9)
- 定義預設參數的函式(§6.10)
- 定義回傳多個值的函式(§6.11)
- 應用函式的萃取於軟體的開發(§6.12)
- 使用逐步細緻化(stepwise refinement)的方式，設計與實作函式(§6.13)
- 使用可重複使用的函式來畫圖(§6.14)

6.1 開放式問題

- 請分別取得 1 到 10，20 到 37，以及 35 到 49 間整數的總和，你會怎麼做?

6.1 開放式問題

```
sum = 0
for i in range(1, 10):
    sum += i
print("Sum from 1 to 10 is", sum)

sum = 0
for i in range(20, 37):
    sum += i
print("Sum from 20 to 37 is", sum)

sum = 0
for i in range(35, 49):
    sum += i
print("Sum from 35 to 49 is", sum)
```

開放式問題

```
sum = 0
for i in range(1, 10):
    sum += i
print("Sum from 1 to 10 is", sum)

sum = 0
for i in range(20, 37):
    sum += i
print("Sum from 20 to 37 is", sum)

sum = 0
for i in range(35, 49):
    sum += i
print("Sum from 35 to 49 is", sum)
```

解答

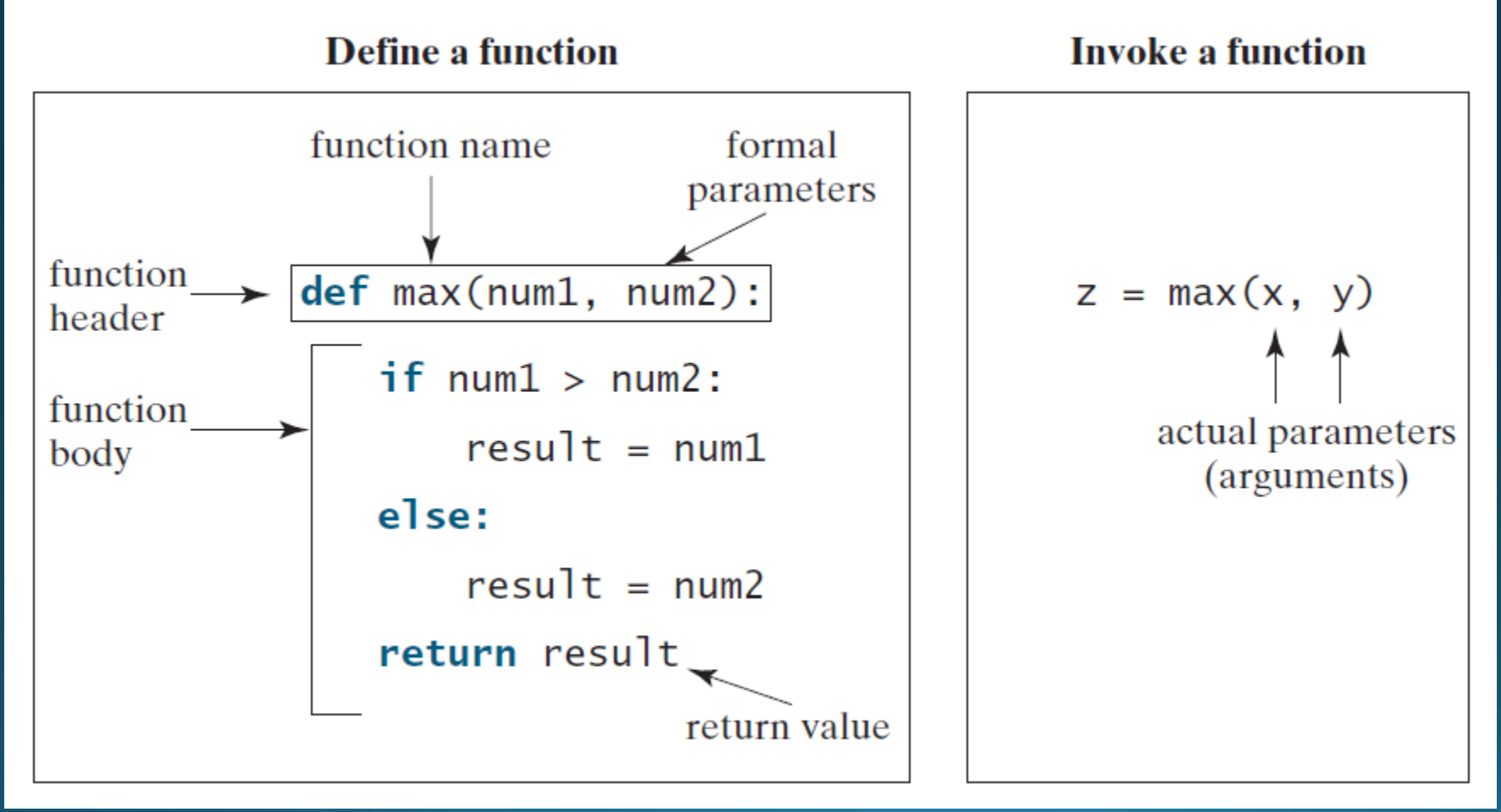
```
def sum(i1, i2):
    result = 0
    for i in range(i1, i2):
        result += i
    return result

def main():
    print("Sum from 1 to 10 is", sum(1, 10))
    print("Sum from 20 to 37 is", sum(20, 37))
    print("Sum from 35 to 49 is", sum(35, 49))

main() # Call the main function
```

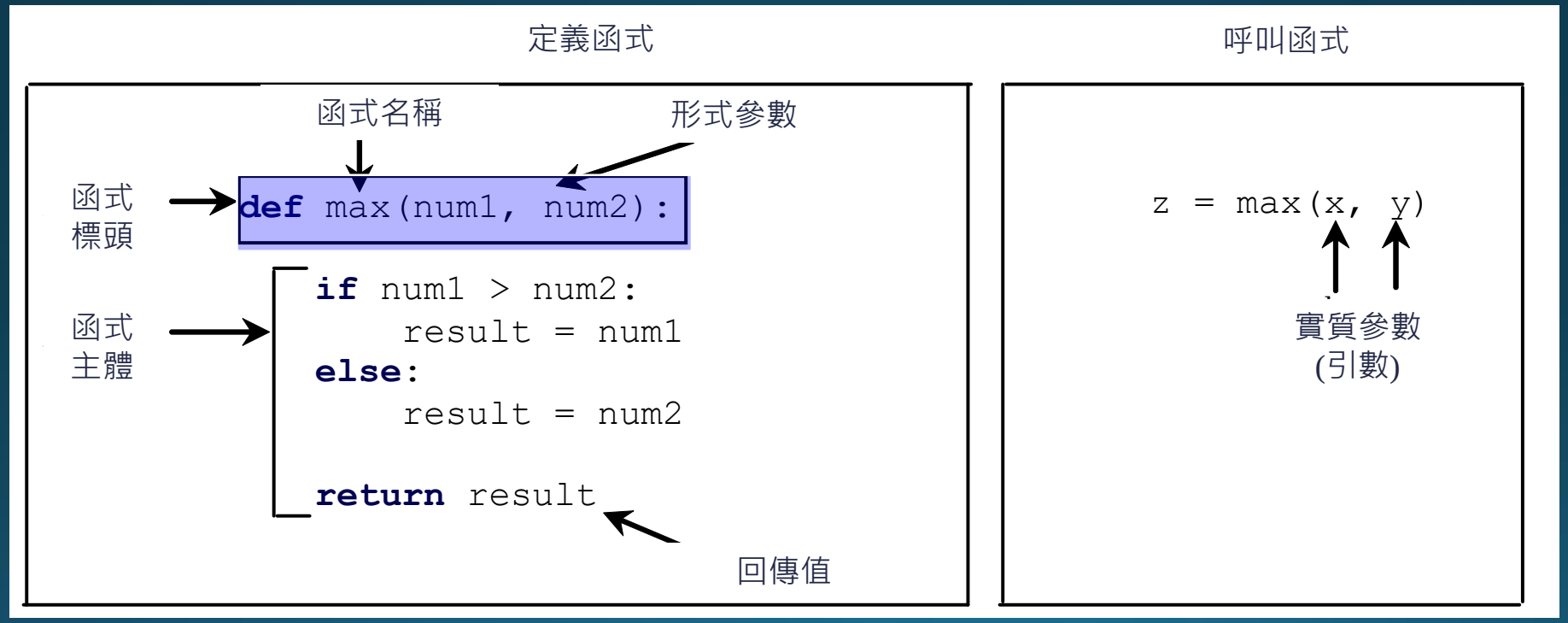
6.2 定義函式

- 函式定義由函式名稱、參數，以及主體內容所組成。



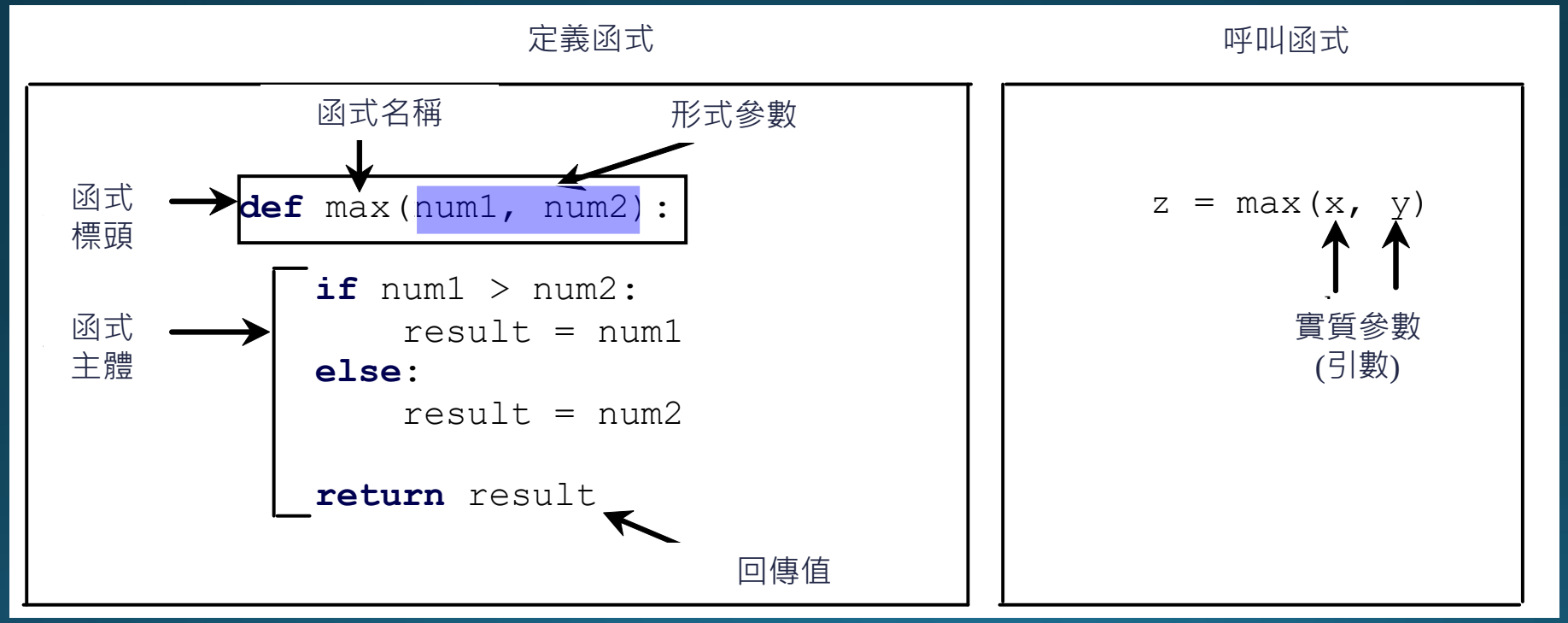
函式標頭

- 函式包含標頭和主體。標頭(header)起源於def關鍵字，後接函式名稱和參數，最後以冒號結尾



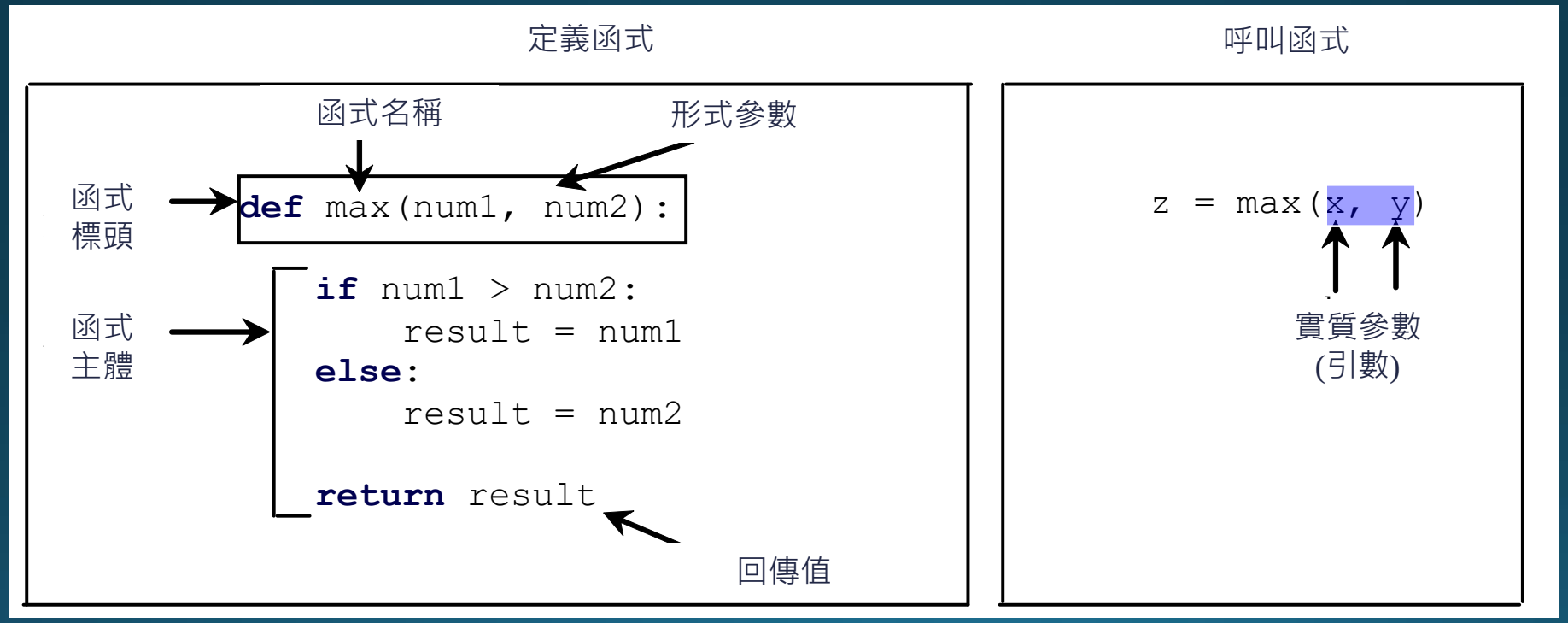
形式參數

- 定義於函式標頭的變數則稱為形式參數



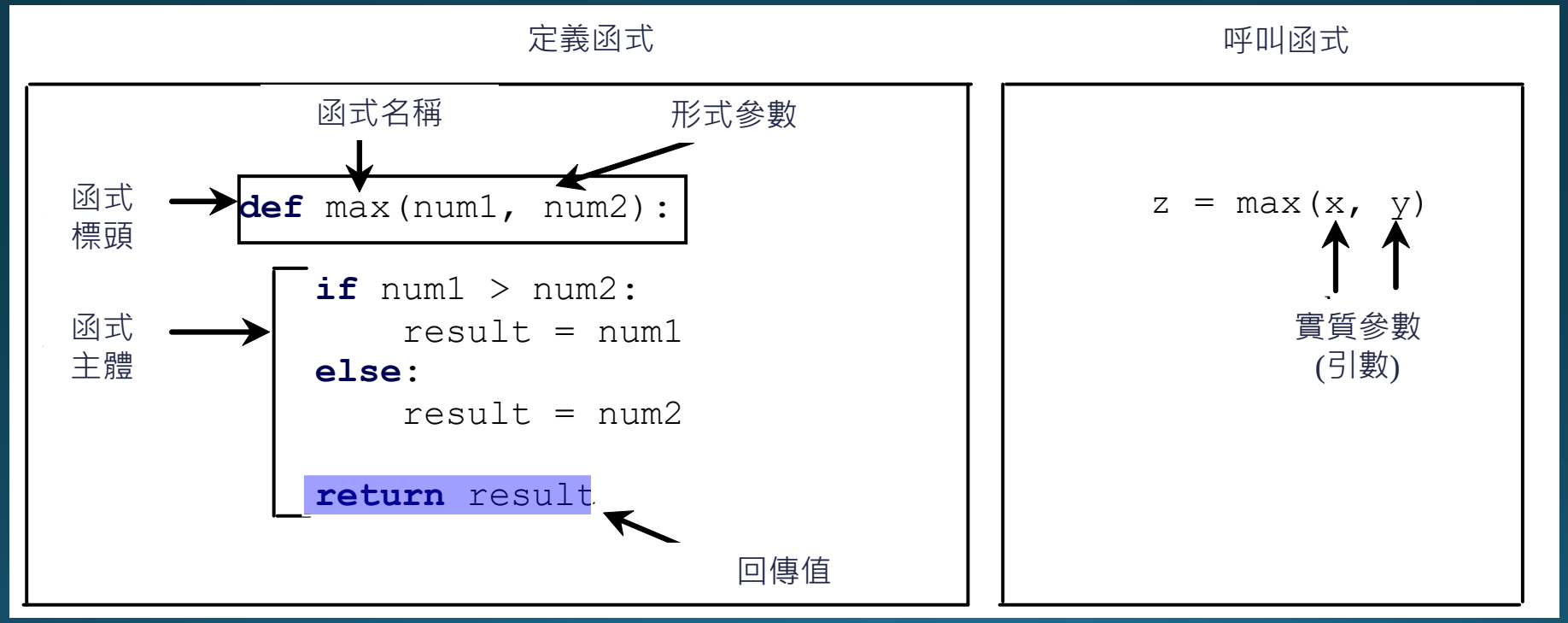
實質參數(引數)

- 當函式被呼叫時，你必須給予的參數則被稱為實質參數(引數)



回傳值

- 函式會使用關鍵字return 回傳一個結果數值，此數值稱為回傳值。



6.3 呼叫函式

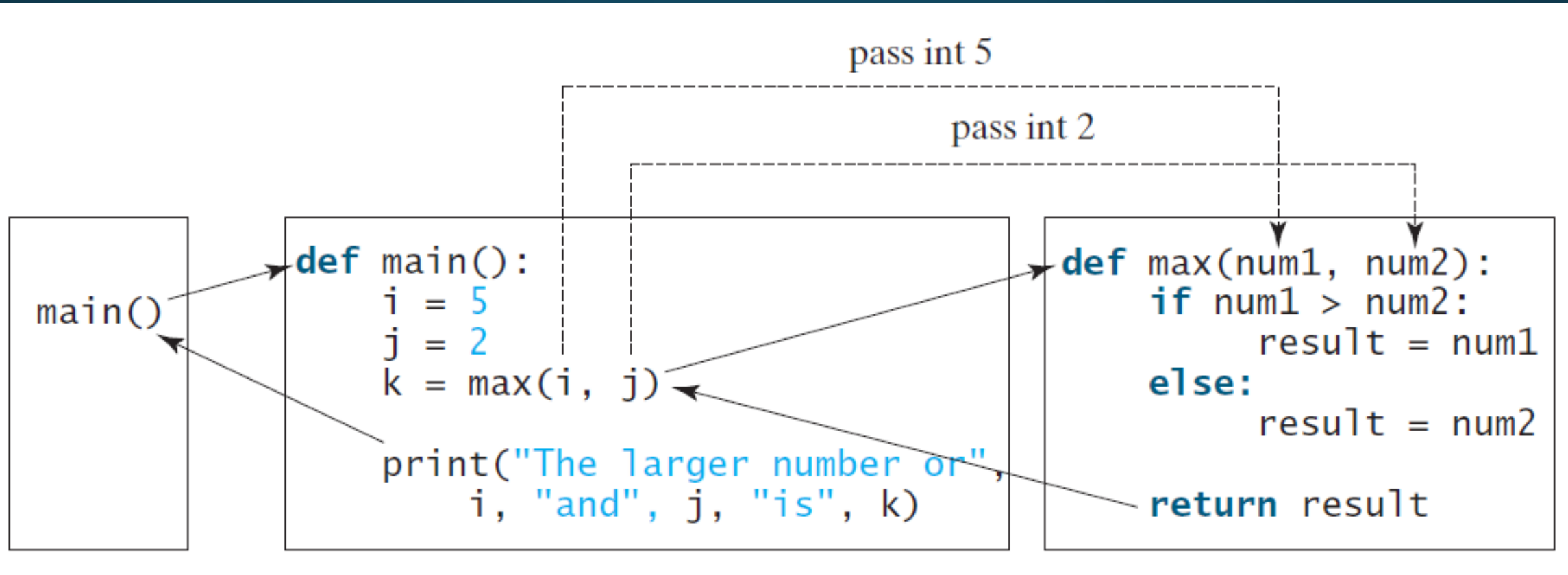
- 於函式定義裡，我們定義該函式的功能為何。要使用函式，就得呼叫(call或invoke)該函式。
- 範例程式6.1為一完整程式，用來測試max函式。

TestMax.py > ...

```python
def max(num1,num2):
    if num1 > num2:
        result = num1
    else:
        result = num2
    return result
def main():
    i = 5
    j = 2
    k = max(i,j)
    print('The larger number of ',i,' and ',j,' is ',k)
main()
```

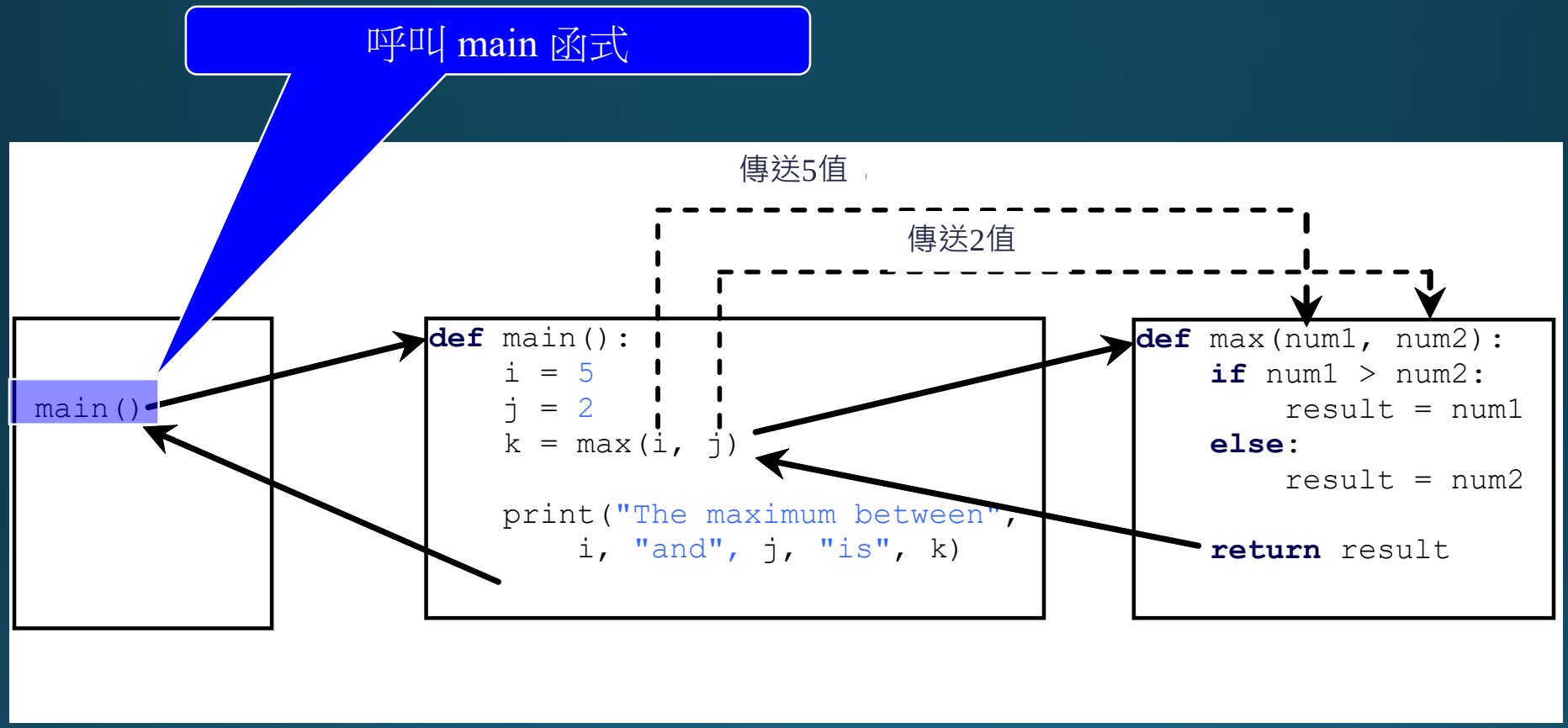
animation

呼叫函式



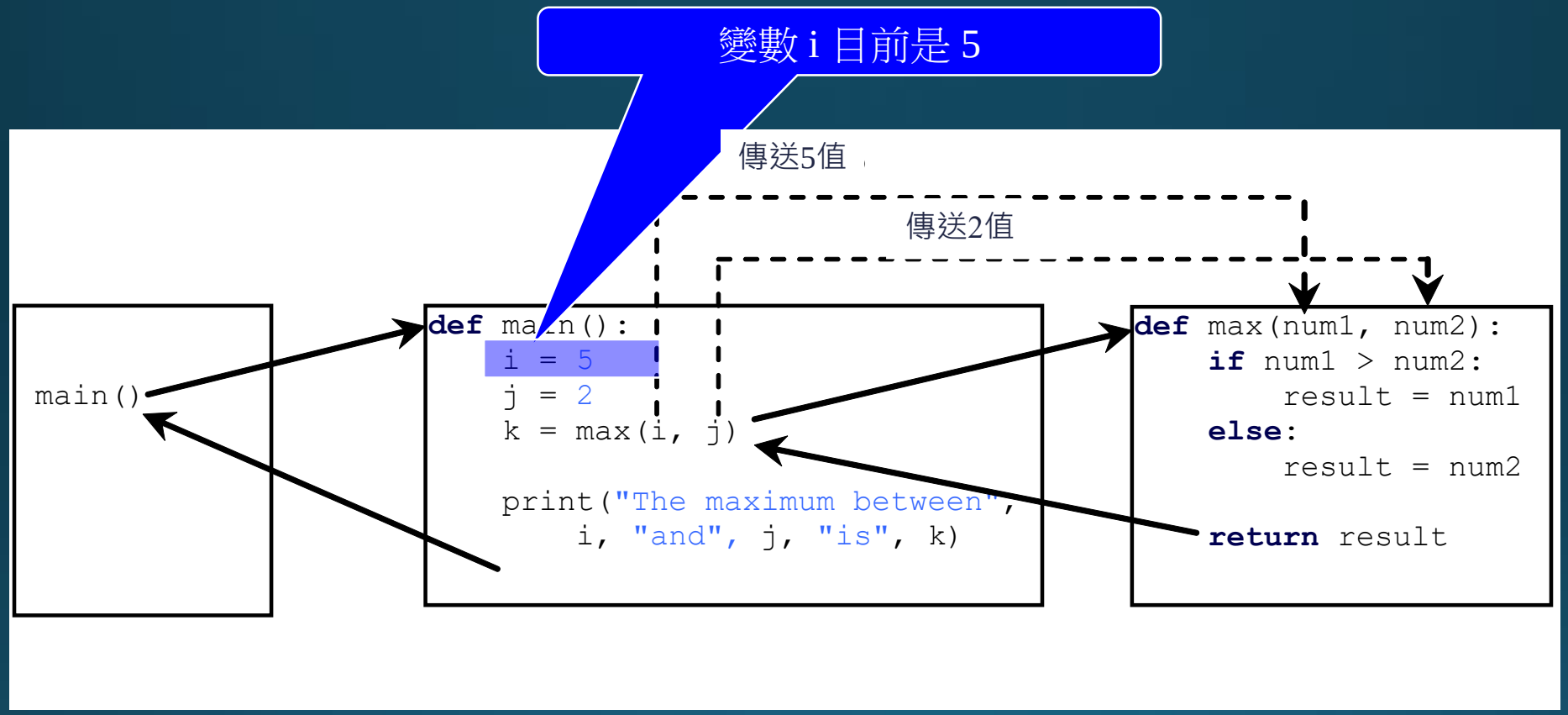
animation

呼叫函式



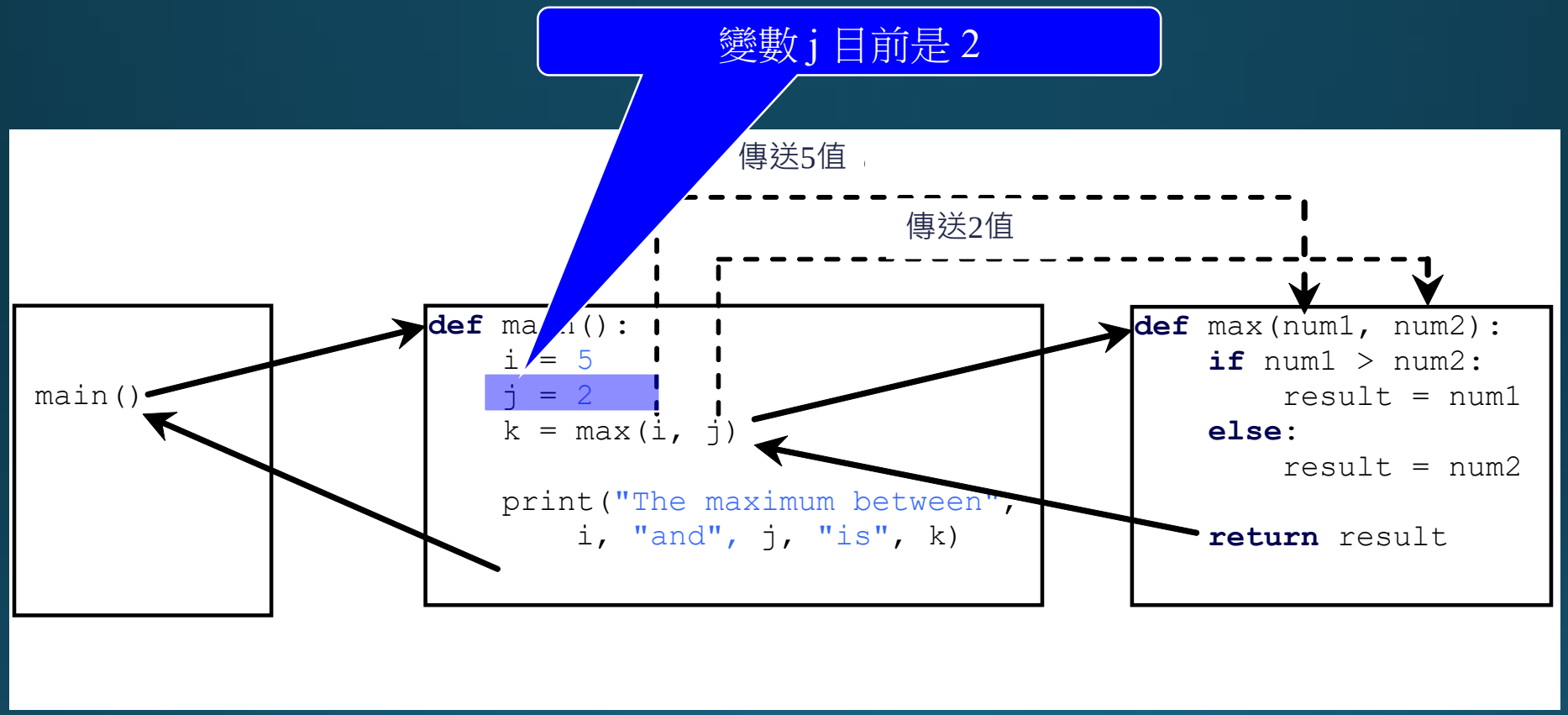
animation

呼叫函式



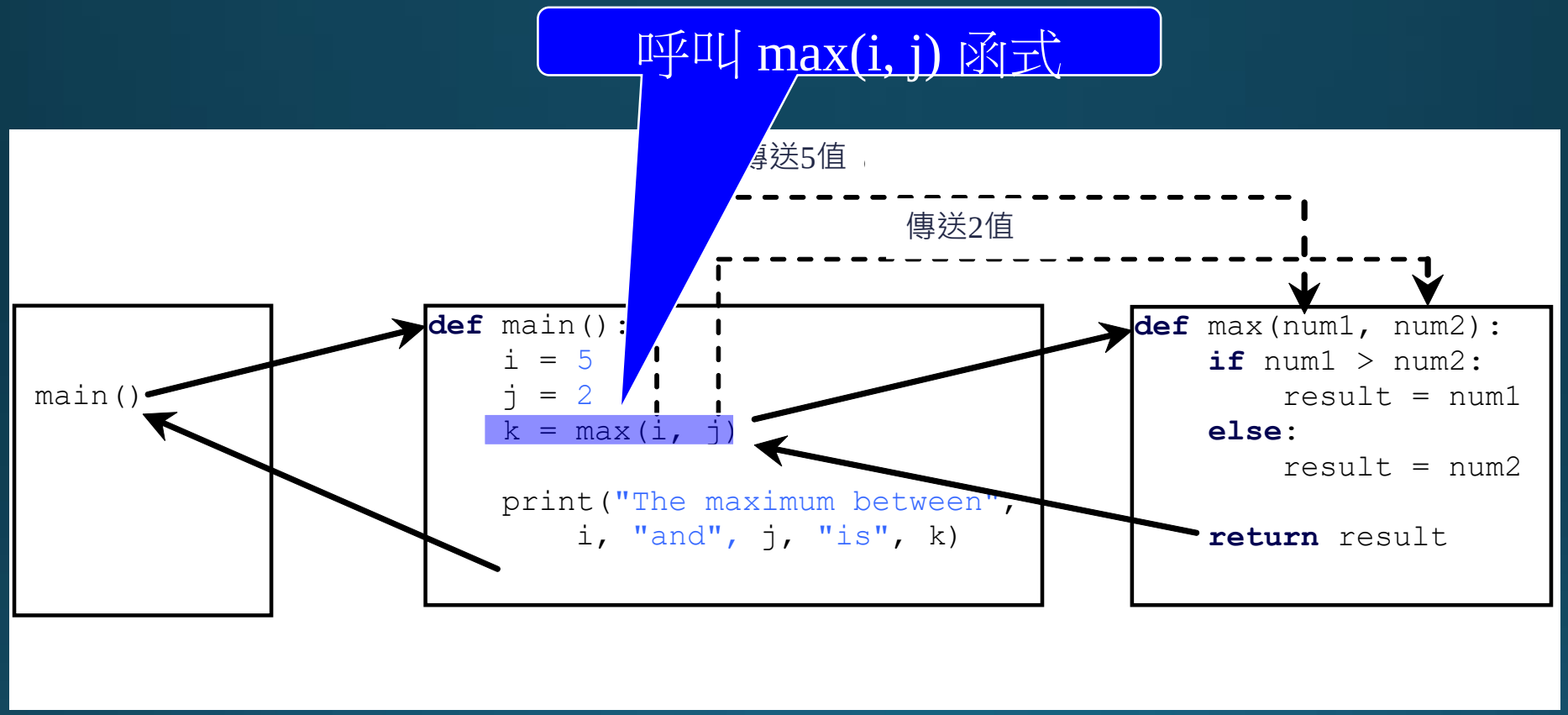
animation

呼叫函式



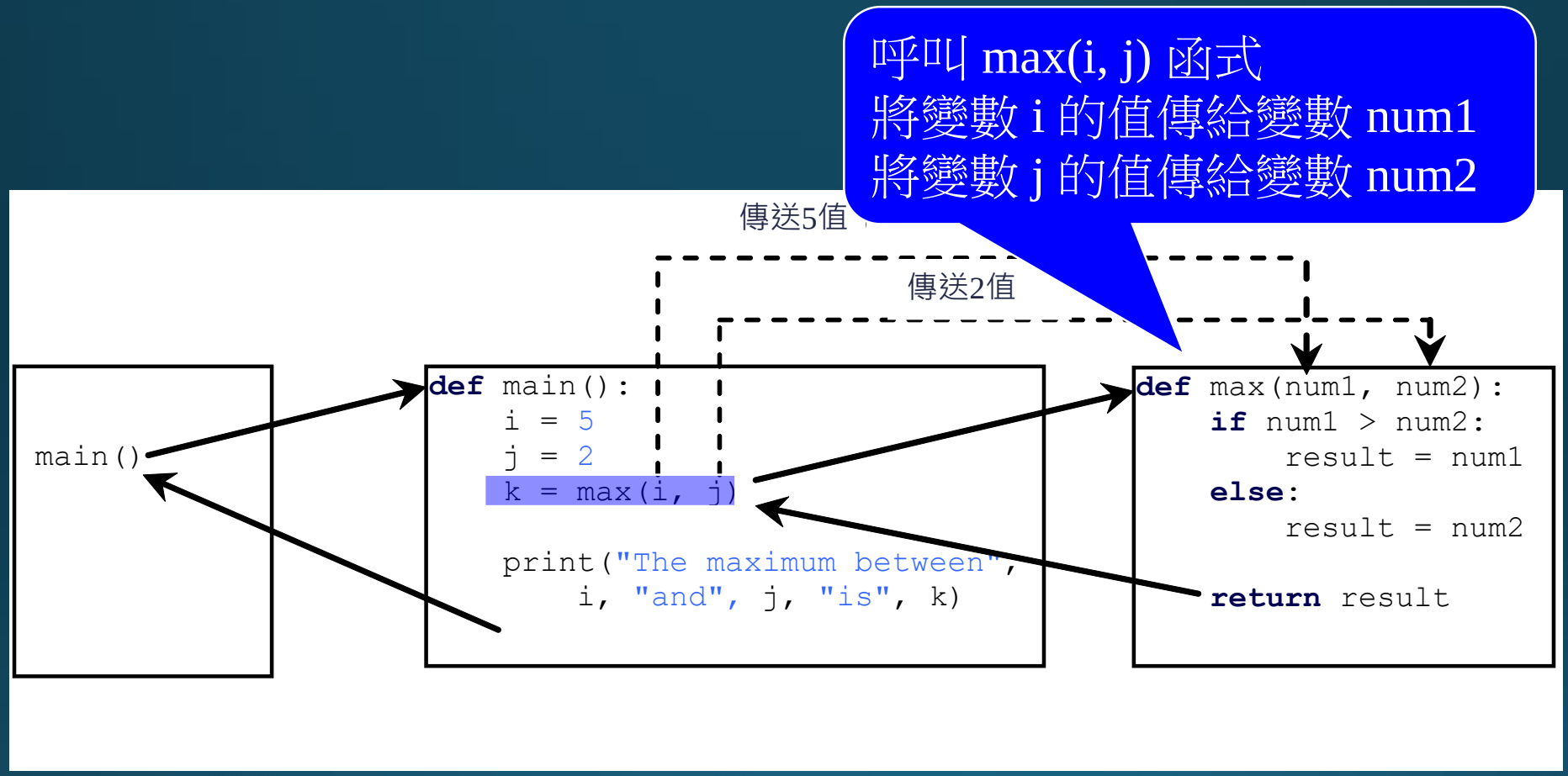
animation

呼叫函式



animation

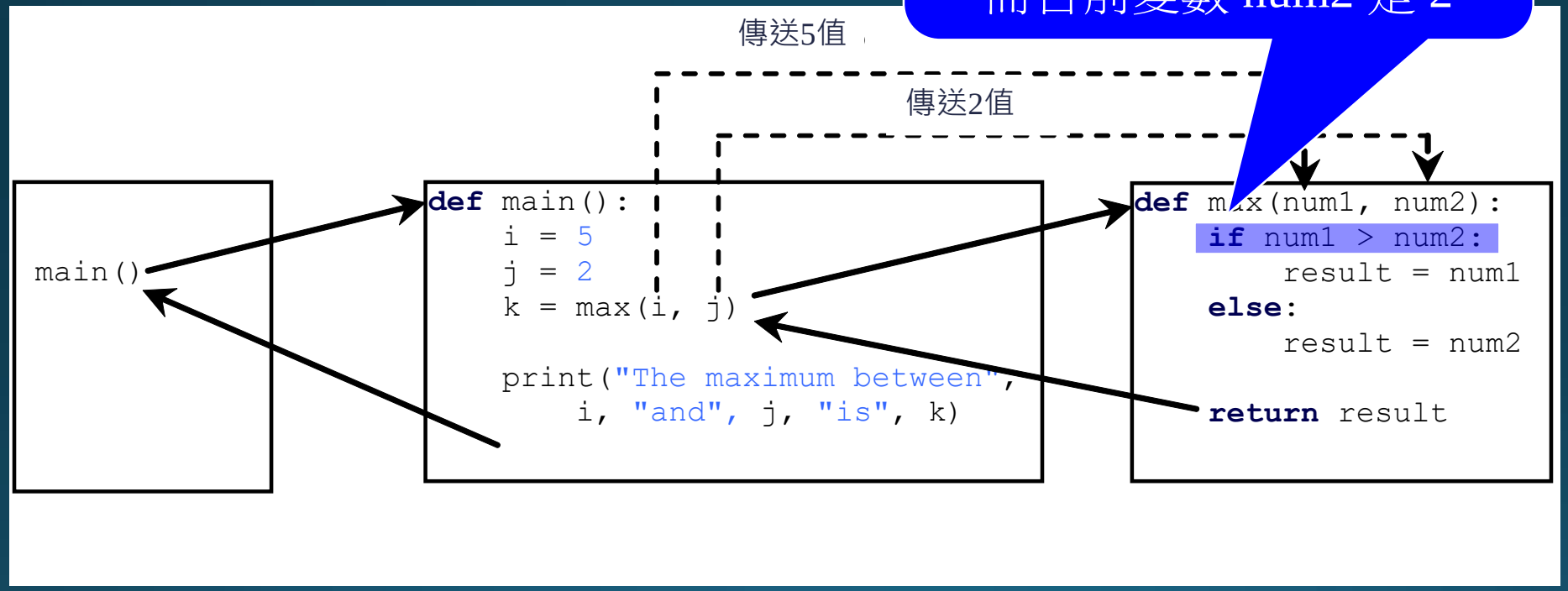
呼叫函式



animation

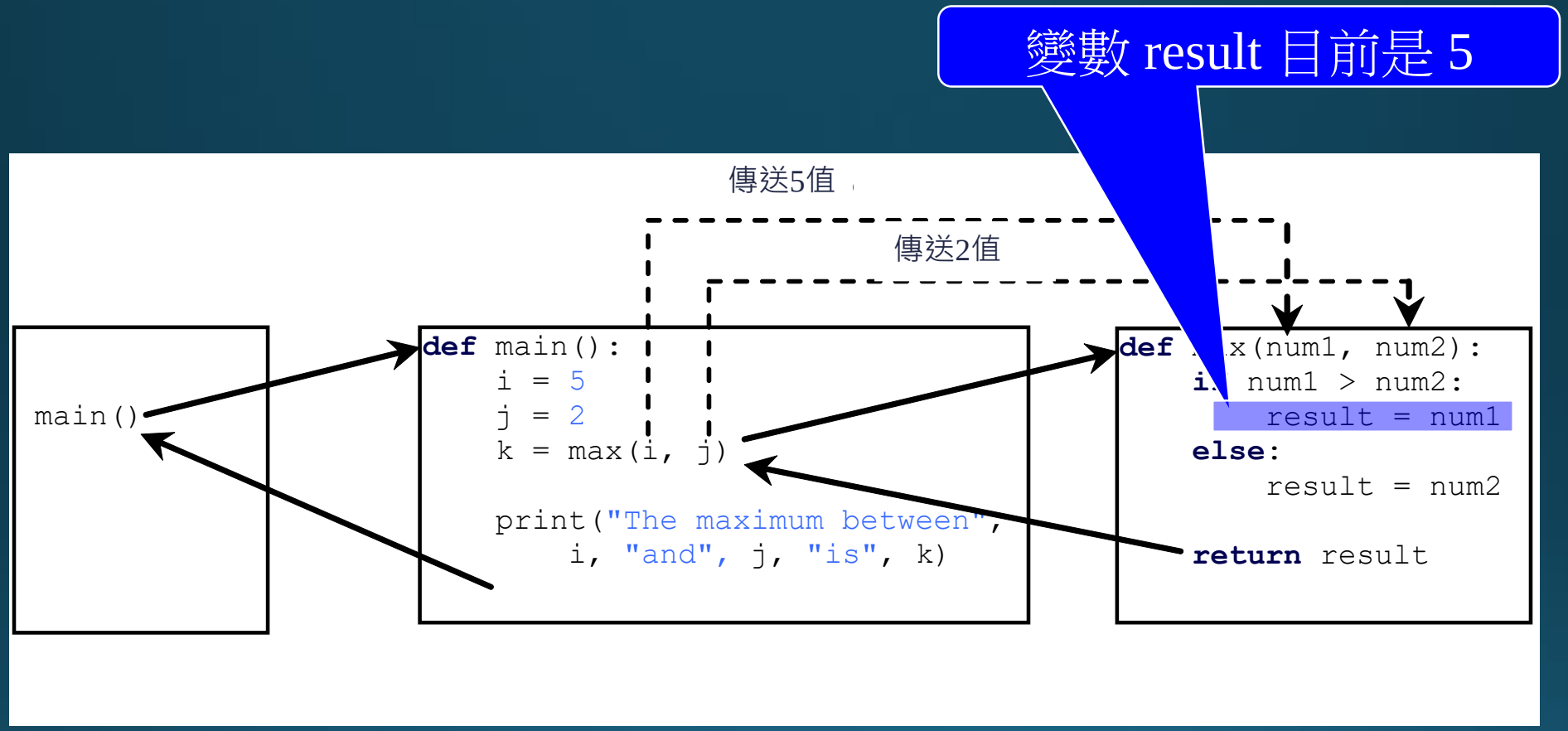
呼叫函式

(num1 > num2) 假設為 true
因為目前變數 num1 是 5
而目前變數 num2 是 2



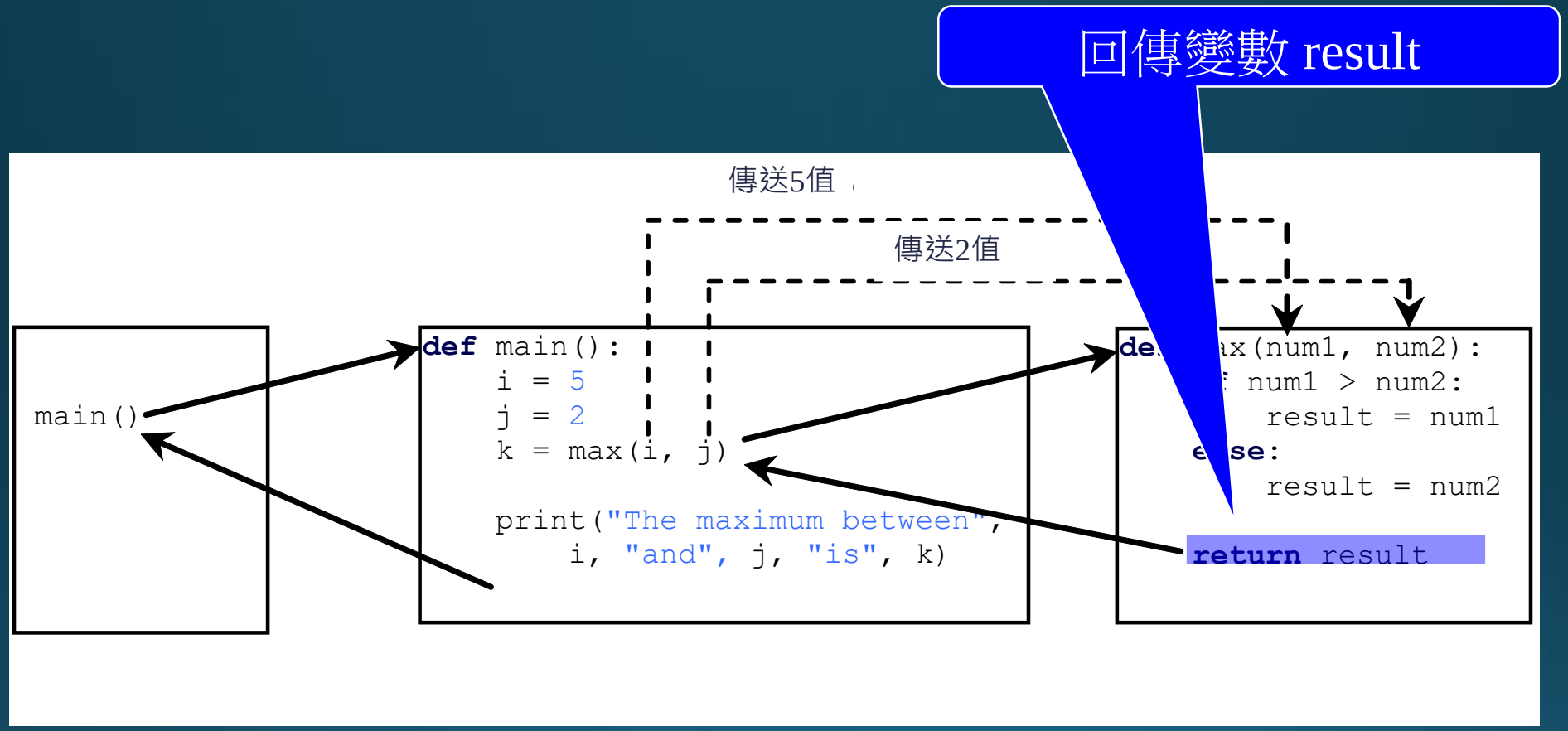
animation

呼叫函式



animation

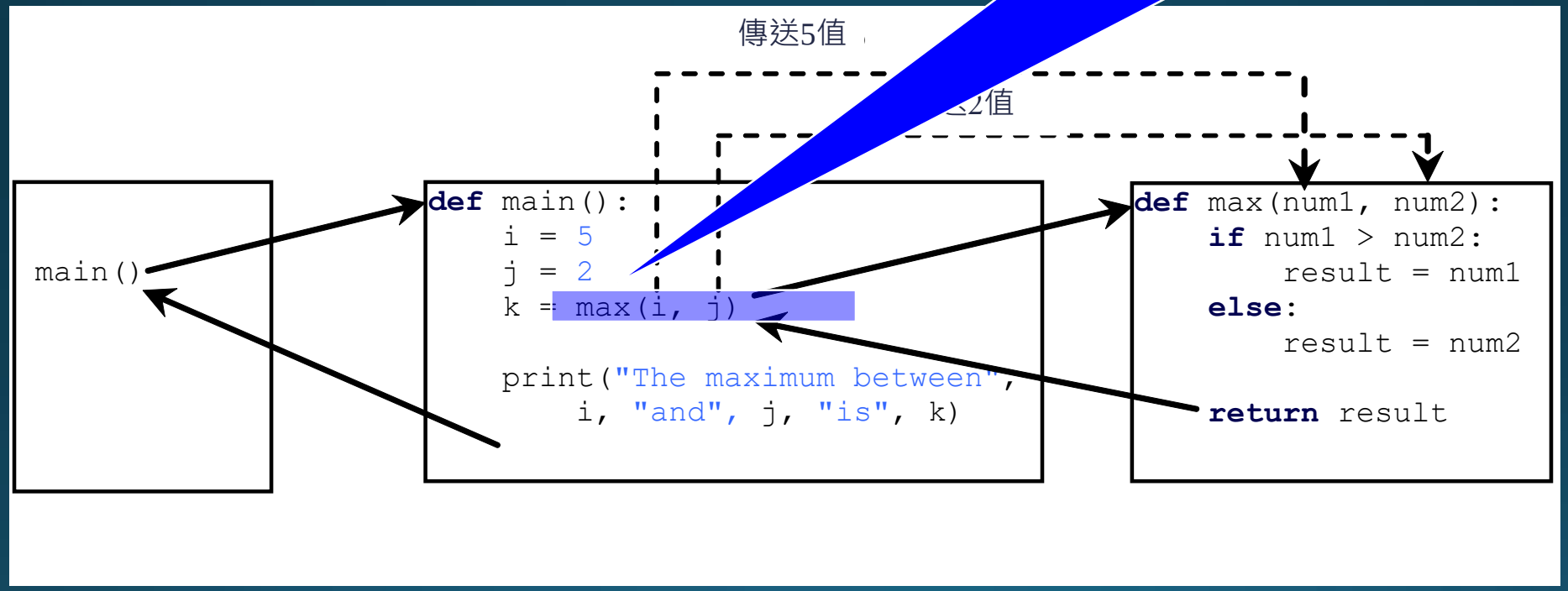
呼叫函式



animation

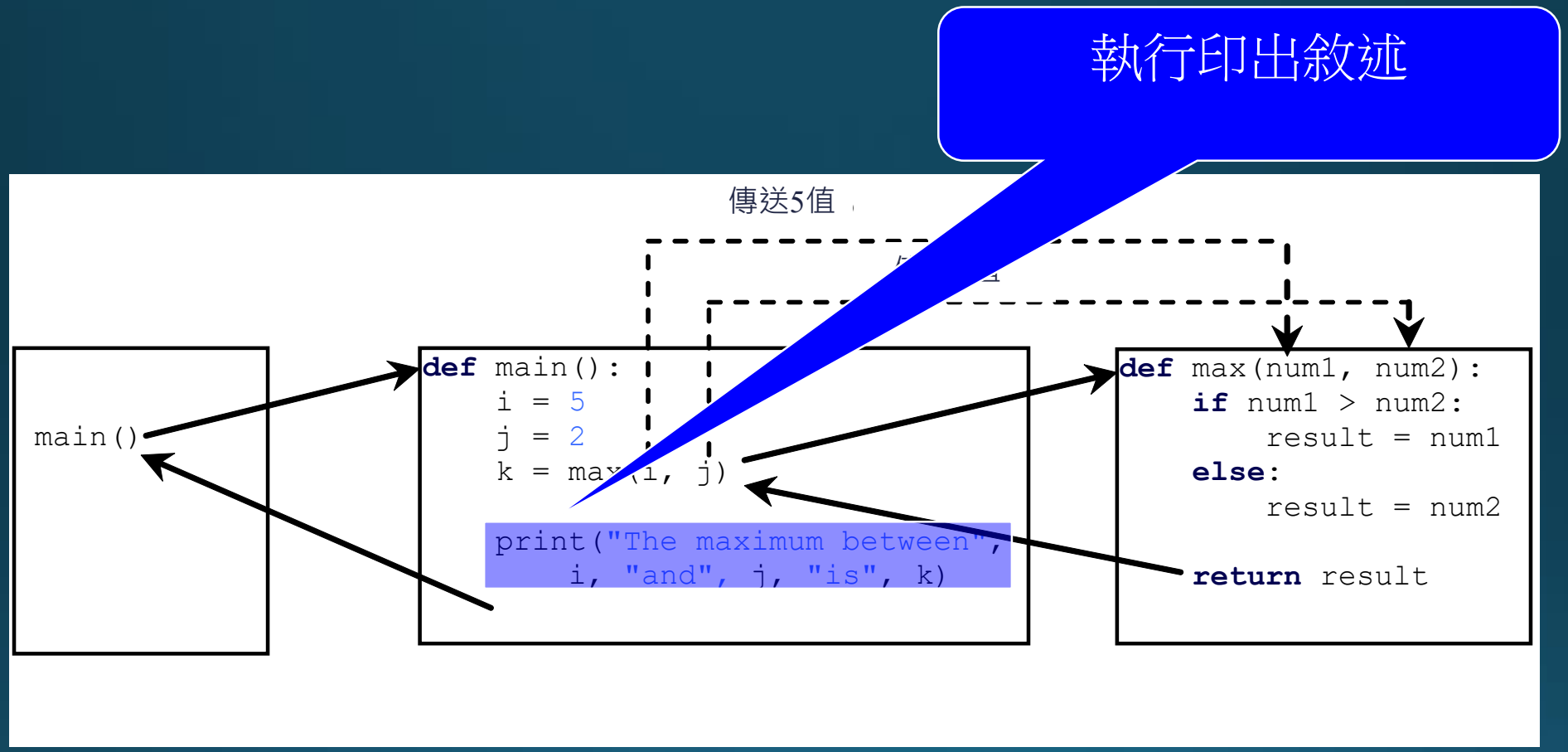
呼叫函式

回到 max(i, j) 並將回傳值指定給變數 k



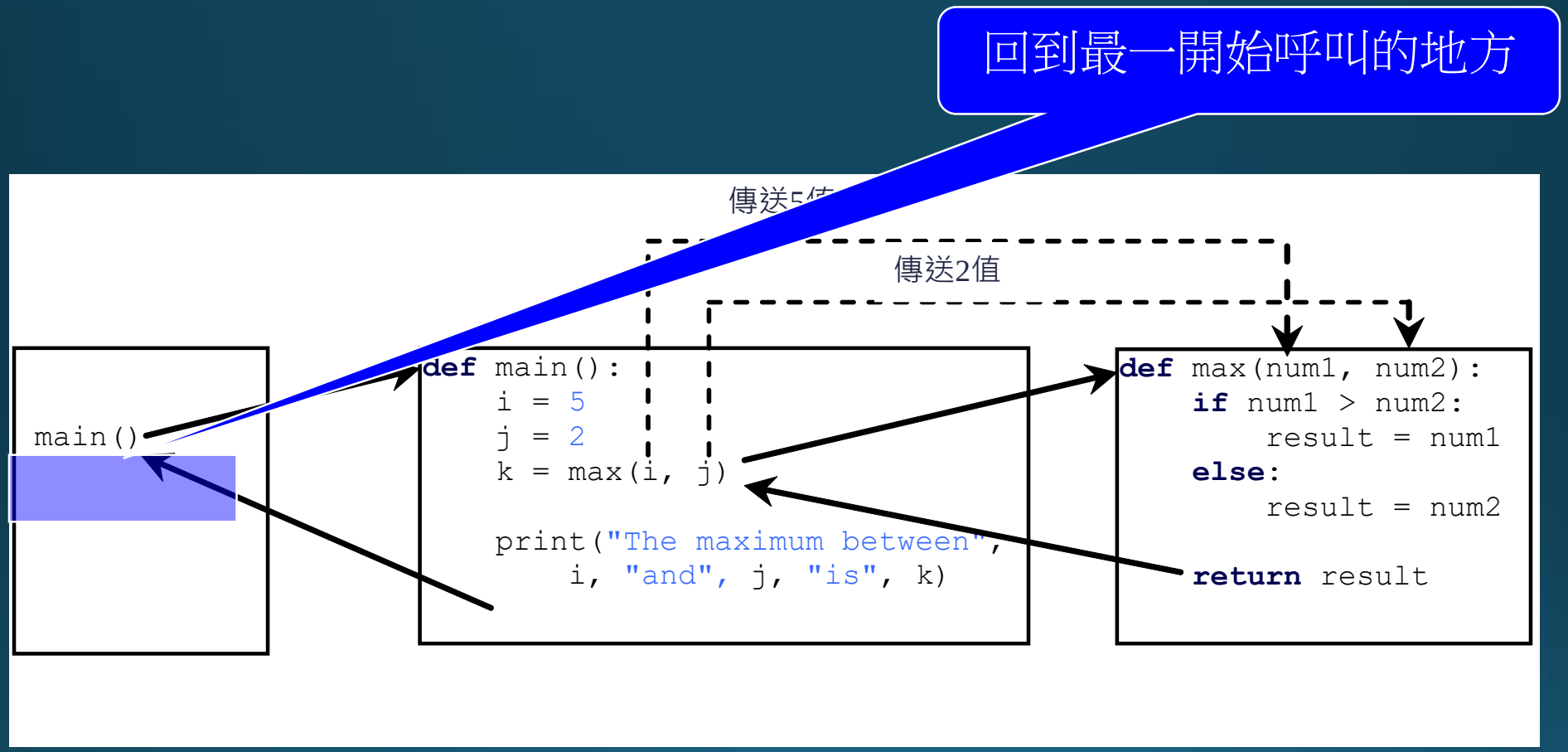
animation

呼叫函式

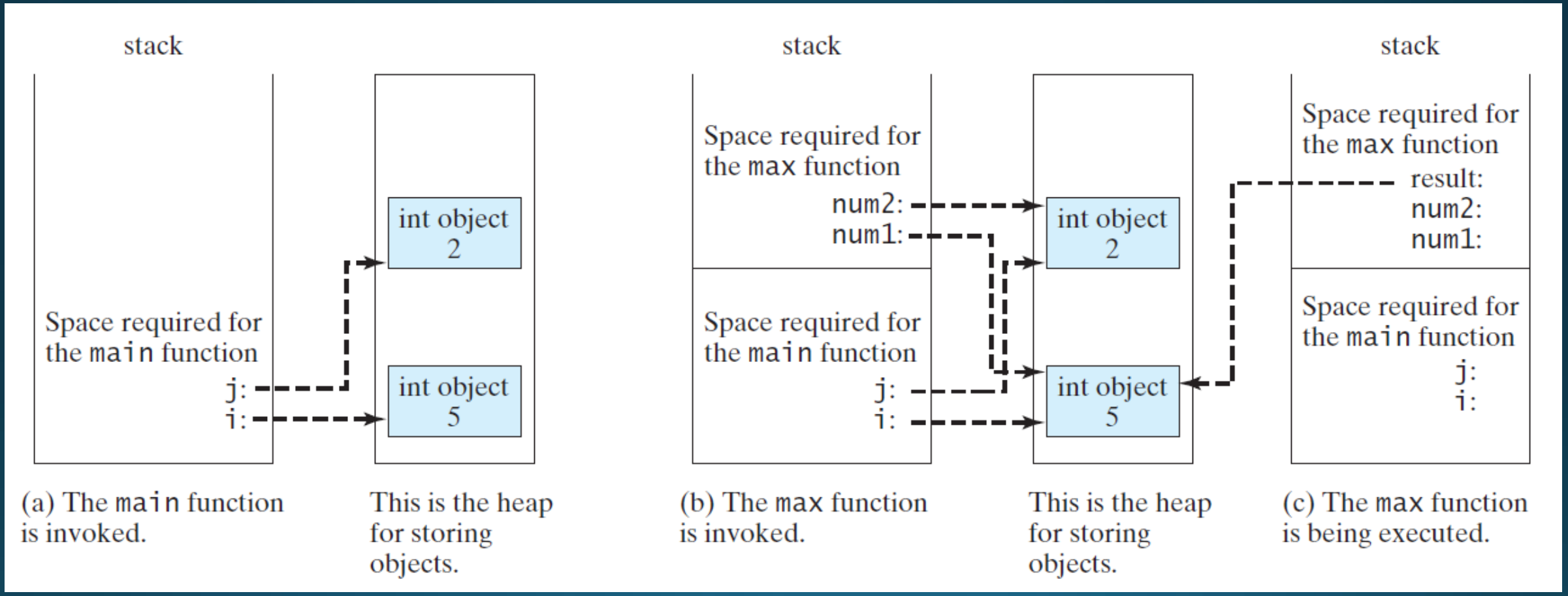


animation

呼叫函式



呼叫堆疊



(a) The `main` function is invoked.

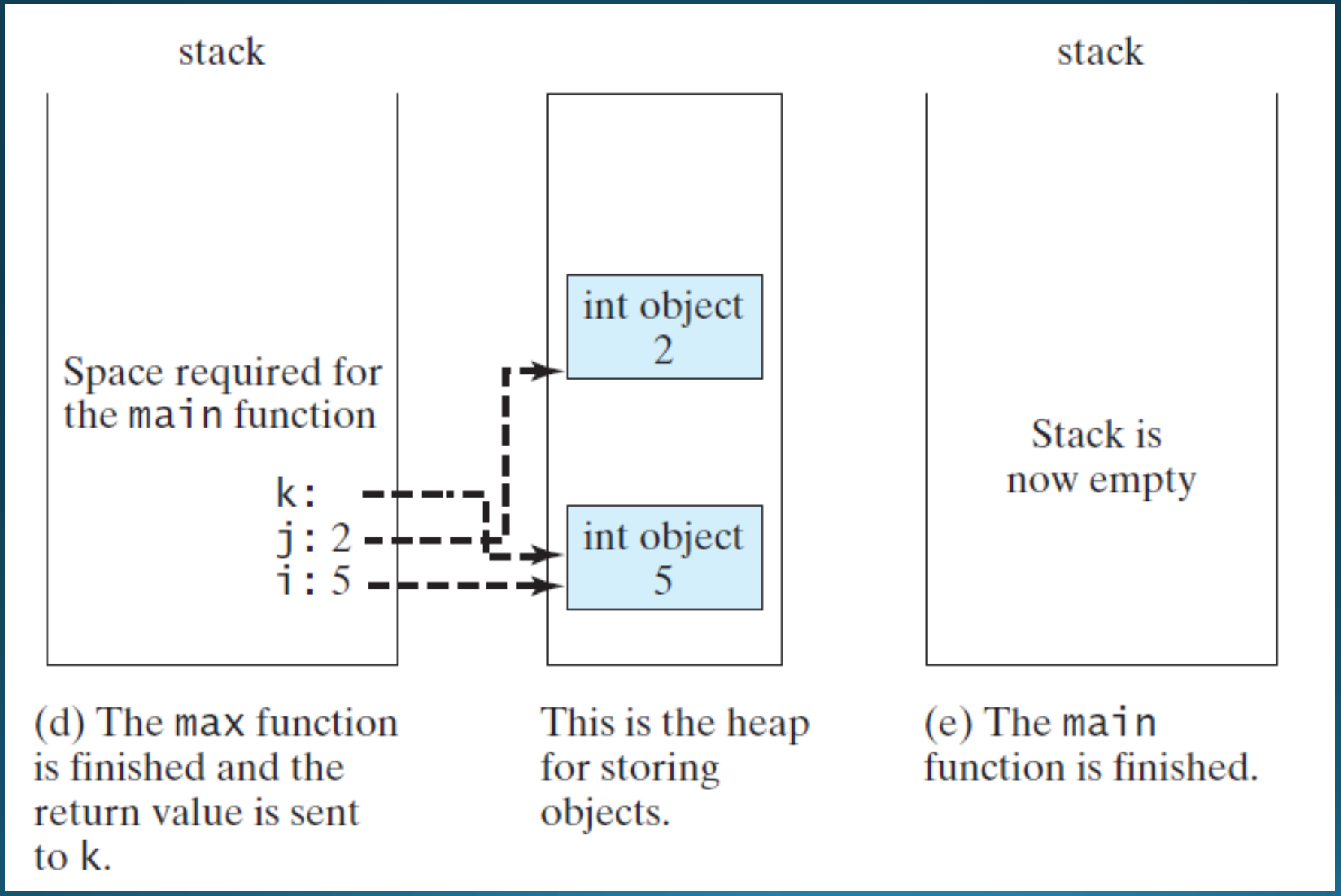
This is the heap for storing objects.

(b) The `max` function is invoked.

This is the heap for storing objects.

(c) The `max` function is being executed.

Call Stacks



(d) The max function is finished and the return value is sent to k.

This is the heap for storing objects.

(e) The main function is finished.

6.4 有無回傳值的函式

- 前一小節介紹了回傳值函式。本小節將說明如何定義與呼叫沒有回傳值的函式。像此類的函式在一般的述語稱之為void 函式。

範例程式6.2 PrintGradeFunction.py

PrintGradeFunction.py > printGrade

```
1  def printGrade(score):
2      if score >= 90:
3          print('A')
4      elif score >= 80:
5          print('B')
6      elif score >= 70:
7          print('C')
8      elif score >= 60:
9          print('D')
10     else:
11         print('F')
12 def main():
13     score = eval(input('Enter a score: '))
14     print('The grade is ',end = '')
15     printGrade(score)
16 main()
```

```
Enter a score: 90
The grade is A
```

```
Enter a score: 80
The grade is B
```

```
Enter a score: 70
The grade is C
```

```
Enter a score: 60
The grade is D
```

```
Enter a score: 59
The grade is F
```

範例程式6.3 ReturnGradeFunction.py

ReturnGradeFunction.py > ...

```
1  def getGrade(score):
2      if score >= 90:
3          return 'A'
4      elif score >= 80:
5          return 'B'
6      elif score >= 70:
7          return 'C'
8      elif score >= 60:
9          return 'D'
10     else:
11         return 'F'
12 def main():
13     score = eval(input('Enter a score: '))
14     print('The grade is ',getGrade(score))
15 main()
```

註釋

- 技術上而言，在Python的函式是否有無回傳值，端看函式是否有無return敘述。若函式沒有回傳值，預設是回傳一None特殊值。基於此理由，沒有回傳值的函式也稱為None函式。None值可以指定給一變數，表示此變數沒有參考到任何的物件。例如您執行以下的程式

```
>>> def sum(number1,number2):
        total = number1+number2

>>> print(sum(1,2))
None
```

returnNone.py

```
1  def printGrade(score):
2      if score < 0 or score > 100 :
3          print("Invalid score")
4          return
5      if score >= 90 :
6          print("A")
7      elif score >= 80 :
8          print("B")
9      elif score >= 70 :
10         print("C")
11     elif score >= 60 :
12         print("D")
13     else:
14         print("F")
15 def main():
16     scoreValue = eval(input("Enter your score: "))
17     printGrade(scoreValue)
18 main()
```

checkpoint 6.2

- 如何定義函式?如何呼叫函式?

```
1  def max(num1,num2):
2      if num1 > num2:
3          result = num1
4      else:
5          result = num2
6      return result
7  def main():
8      i = 5
9      j = 2
10     k = max(i,j)
11     print('The larger number of ',i,' and ',j,' is ',k)
12 main()
```

checkpoint 6.3

- 如何使用條件運算子，簡化範例程式6.1 的max函式?

checkPoint6_03.py > ...

```
1  def max(num1,num2):
2      return num1 if num1 > num2 else num2
3  def main():
4      i = 5
5      j = 2
6      k = max(i,j)
7      print('The larger number of ',i,' and ',j,' is ',k)
8  main()
```

```
The larger number of  5  and  2  is 5
```

checkpoint 6.4

- True or False？呼叫 None 函式總是為一敘述(True)，但呼叫有回傳值的函式總是為運算式的元素?(False)

checkpoint 6.5

- 在 None 函式中可以有 return 敘述? 在下列的函式return 敘述會產生語法錯誤?

checkPoint6_04.py > ...

```
1  def xFunction(x , y):
2      print(x + y)
3      return
4  xFunction(2,3)
```

checkpoint 6.8

- 試更正以下程式碼的錯誤?

checkPoint6_08.py > ...

```
1  def function1(n,m):
2      function2(3.4)
3  def function2(n):
4      if n > 0:
5        return 1
6    elif n == 0:
7        return 0
8    elif n < 0:
9      return -1
10 print(function1(2,3))
11 print(function2(3.4))
```

checkPoint6_08.py > ...

```
1  def function1(n,m):
2      function2(3.4)
3  def function2(n):
4      if n > 0:
5          return 1
6      elif n == 0:
7          return 0
8      elif n < 0:
9          return -1
10 print(function1(2,3))
11 print(function2(3.4))
```

checkpoint 6.9

- 請問以下程式碼的輸出結果?

checkPoint6_09.py > ...

```
1  def main():
2      print(min(5,6))
3  def min(n1,n2):
4      smallest = n1
5      if n2 < smallest :
6          smallest = n2
7  main()
```

None

checkpoint 6.10

- 當執行以下程式碼時，將會出現何種錯誤?

checkPoint6_10.py > ...

```
1  def main():
2      print(min(min(5,6),min(51,6)))
3  def min(n1,n2):
4      smallest = n1
5      if n2 < smallest :
6          smallest = n2
7  main()
```

```
TypeError: '<' not supported between instances of 'NoneType' and
'NoneType'
```

6.5 位置引數

```
def nPrintln(message, n):
    for i in range(0, n):
        print(message)
```

- 假設你使用下列敘述來呼叫函式 nPrintln(“Welcome to Python”, 5)
- 你會得到什麼?
- 假設你使用下列敘述來呼叫函式 nPrintln(“Computer Science”, 15)
- 你會得到什麼?
- 然而，不可以使用以下的敘述呼叫：nPrintln(4, “Computer Science”)

關鍵字引數

```
def nPrintln(message, n):
    for i in range(0, n):
        print(message)
```

- **下列敘述錯在哪：**

nPrintln(4, “Computer Science”)

- **那這樣ok嗎?**

nPrintln(n = 4, message = “Computer Science”)

checkpoint 6.12

- 當執行以下程式碼時，將會出現何種錯誤?

checkPoint6_12.py > ...

```
1  def f(p1,p2,p3,p4):
2      return
3  print("Result is:",f(1,p2=3,p3=3,p4=4))
4  # print("Result is:",f(1,p2=3,4,p4=4))      SyntaxError
5  # print("Result is:",f(p1=1,p2=3,4,p4=4))   SyntaxError
6  print("Result is:",f(p1=1,p2=3,p3=4,p4=4))
7  print("Result is:",f(p4=1,p2=3,p3=4,p1=4))
```

```
Result is: None
Result is: None
Result is: None
```

6.6 以參考傳遞參數

- Python的所有資料皆為物件，物件的變數其實就是參考(reference)到物件。當引發一含有引數的函式時，每一引數的參考將傳送給參數。此稱為傳參考呼叫(pass-by-reference)。
- 若引數是一數值或字串，即使在函式內更改了參數，引數也不會改變。

範例程式6.4 Increment.py

Increment.py > ...

```
1  def main():
2      x = 1
3      print('Before the call, x is',x)
4      increment(x)
5      print('After the call, x is',x)
6  def increment(n):
7      n += 1
8      print('\tn inside the function is',n)
9  main()
```

```
Before the call, x is 1
        n inside the function is 2
After the call, x is 1
```

數值與字串是不可變更物件(immutable object)

```
>>> x = 4
>>> y = x
>>> id(x)
140736867120048
>>> id(y)
140736867120048
>>>
>>> y = y+1
>>> id(y)
140736867120080
```

checkpoint 6.15(a)

- 當執行以下程式碼時其輸出結果為何?

checkPoint6_15a.py > ...

```
1  def main():
2      max = 0
3      getMax(1, 2, max)
4      print(max)
5  def getMax(value1, value2, max):
6      if value1 > value2:
7          max = value1
8      else:
9          max = value2
10 main()
```

```
0
```

checkpoint 6.15(b)

- 當執行以下程式碼時其輸出結果為何?

checkPoint6_15b.py > ...

```
1  def main():
2      i = 1
3      while i <= 6:
4          print(function1(i, 2))
5          i += 1
6  def function1(i, num):
7      line = ""
8      total =0
9      for j in range(1, i):
10         line += str(num) + " "
11         num *= 2
12         total +=j
13     return line
14 main()
```

```
2
2 4
2 4 8
2 4 8 16
2 4 8 16 32
```

checkpoint 6.15(c)

- 當執行以下程式碼時其輸出結果為何?

```
Before the call, variable times is 3

n =  3

Welcome to CS!

n =  2

Welcome to CS!

n =  1

Welcome to CS!

After the call, variable times is 3
```

checkpoint 6.15(c)

checkPoint6_15c.py > ...

```
1  def main():
2      times = 3
3      print("Before the call, variable","times is", times)
4      nPrint("Welcome to CS!", times)
5      print("After the call, variable","times is", times)
6  def nPrint(message, n):
7      while n > 0:
8          print("n = ", n)
9          print(message)
10         n -= 1
11 main()
```

checkpoint 6.15(d)

- 當執行以下程式碼時其輸出結果為何?

```
i is 1
 1
i is 2
 2 1
i is 3
and then it is an infinite loop.
```

checkpoint 6.15(d)

checkPoint6_15d.py > ...

```python
def main():
    i = 0
    while i <= 4:
        function1(i)
        i += 1
        print("i is", i)
def function1(i):
    line = " "
    while i >= 1:
        if i % 3 != 0:
            line += str(i) + " "
            i -= 1
    print(line)
main()
```

6.7 模組化程式

- 函式可用來減少多餘的程式碼，允許程式碼重複使用，還可以用來模組化程式碼，提升程式品質。

範例程式6.5 GDCFunction.py

GCDFunction.py > ...

```
1  def gcd(n1,n2):
2      gcd = 1
3      k = 2
4      while k <= n1 and k <= n2:
5          if n1 % k == 0 and n2 % k == 0 :
6              gcd = k
7          k += 1
8      return gcd
```

範例程式6.6 TestGDCFunction.py

TestGCDFunction.py > ...

```
1  from GCDFunction import gcd
2  n1 = eval(input('Enter the first integer: '))
3  n2 = eval(input('Enter the second integer: '))
4  print('The greatest common divisor for',n1,'and',n2,'is',gcd(n1,n2))
```

```
Enter the first integer: 45
Enter the second integer: 75
The greatest common divisor for 45 and 75 is 15
```

範例程式6.7 PrimeNumberFunction.py

```
The first 50 prime numbers are:
   1   2   3   4   5   6   7   8   9  10
 ----------------------------------------
   2   3   5   7  11  13  17  19  23  29
  31  37  41  43  47  53  59  61  67  71
  73  79  83  89  97 101 103 107 109 113
 127 131 137 139 149 151 157 163 167 173
 179 181 191 193 197 199 211 223 227 229
```

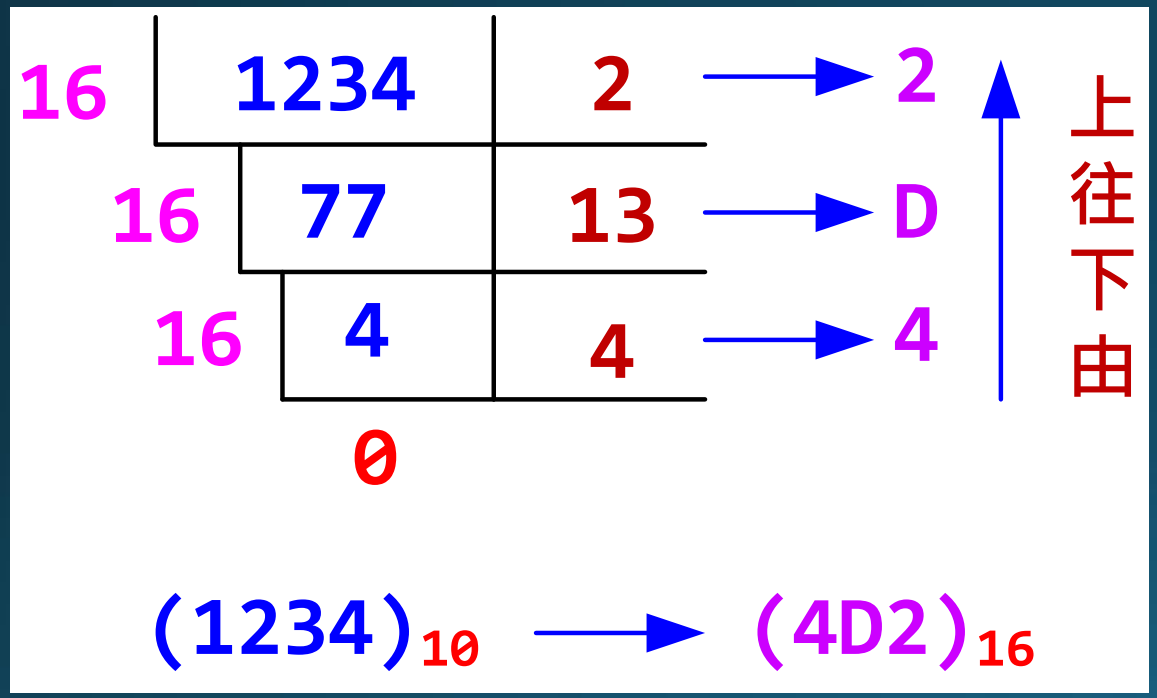
範例程式6.7 PrimeNumberFunction.py

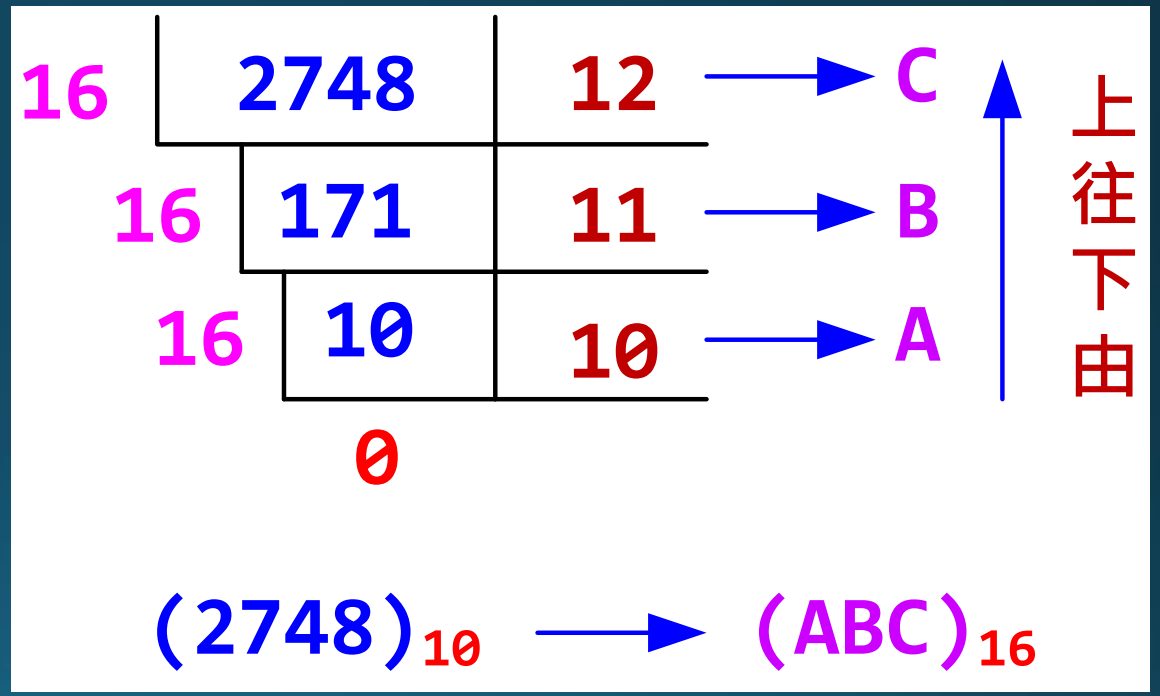
PrimeNumberFunction.py > isPrime

```
1  def isPrime(number):
2      divisor = 2
3      while divisor <= number /2:
4          if number % divisor == 0:
5              return False
6          divisor += 1
7      return True
8  def printPrimeNumbers(numberOfPrimes):
9      numberOfPrimePerLine = 10
10     count = 0
11     number = 2
```

```
12      while count < numberOfPrimes:
13          if isPrime(number):
14              count += 1
15              print(format(number,'4d'),end ='')
16              if count%numberOfPrimePerLine == 0:
17                  print()
18          number += 1
19  def main():
20      print('The first 50 prime numbers are:')
21      for i in range(1,11):
22          print(format(i,'4d'),end='')
23      print('\n','-'*40)
24      printPrimeNumbers(50)
25  main()
```

6.8 個案研究：將十進位轉換為十六進位





範例程式6.8 Decimal2HexConversion.py

```
Enter a decimal number: 1234
The hex number for decimal 1234 is 4D2


Enter a decimal number: 2748
The hex number for decimal 2748 is ABC
```

Decimal2HexConversion.py > decimalToHex

```python
def decimalToHex(decimalValue):
    hex = ''
    while decimalValue != 0:
        hexValue = decimalValue % 16
        hex = toHexChar(hexValue) + hex
        decimalValue = decimalValue // 16
    return hex
def toHexChar(hexValue):
    if 0<= hexValue <=9 :
        return chr(hexValue + ord('0'))
    else:
        return chr(hexValue -10 + ord('A'))
def main():
    decimalValue = eval(input('Enter a decimal number: '))
    print('The hex number for decimal',
            decimalValue, 'is',decimalToHex(decimalValue))
main()
```

個案研究：將十進位轉換為十六進位

	line#	decimalValue	hex	hexValue	toHexChar(hexValue)
	21	1234			
	3		""		
iteration 1	6			2	
	7		"2"		"2"
	8	77			
iteration 2	6			13	
	7		"D2"		"D"
	8	4			
iteration 3	6			4	
	7		"4D2"		"4"
	8	0			

6.9 變數的有效範圍

- 變數的有效範圍(Scope)：表示變數在程式可參考的範圍。
- 宣告於函式的變數被稱為區域變數(local variable)，何者只能在該函式內存取。
- 區域變數的有效範圍起始於宣告點，結束於包含該變數的區段結尾。
- 在 Python 中，你也可以使用全域變數(global variables)，何者宣告於所以得函式之外，可以在所有函式被存取。

變數的有效範圍: 範例一

scope-exm1.py > ...

```
1  globalVar = 1
2  def f1():
3      localVar = 2
4      print(globalVar)
5      print(localVar)
6  f1()
7  localVar = 0
8  print(globalVar)
9  print(localVar)
```

```
1
2
1
0
```

變數的有效範圍: 範例二

scope-exm2.py > ...

```
1  x = 1
2  def f1():
3      x = 2
4      print(x)
5  f1()
6  print(x)
```

```
2

1
```

變數的有效範圍: 範例三

scope-exm3.py > ...

```
1  x = eval(input('Enter a number: '))
2  if x > 0 :
3      y = 4
4  print(y)
```

```
Enter a number: 6
4
```

```
Enter a number: -6
print(y)
NameError: name 'y' is not defined
```

變數的有效範圍: 範例四

scope-exm4.py > ...

```
1  sum = 0
2  for i in range(5):
3      sum += i
4      print('第',i+1,'次總和 = ',sum)
5  print('最後一次的 i= ',i)
6  print('總和 = ',sum)
```

```
第 1 次總和 =  0
第 2 次總和 =  1
第 3 次總和 =  3
第 4 次總和 =  6
第 5 次總和 =  10
最後一次的 i=  4
總和 =  10
```

變數的有效範圍: 範例五

scope-exm5.py > ...

```
1  x = 1
2  def increase():
3      global x
4      x = x+1
5      print(x)
6  increase()
7  print(x)
```

```
2

2
```

checkpoint 6.17(a)

correct6-17a.py > ...

```python
def function(x):
    print('Function\'s argument x =',x)
    x = 4.5
    y = 3.4
    print('Local variable y = ',y)
x = 2
y = 4
function(x)
print('Main function variable x =',x)
print('Main function variable y =',y)
```

```
Function's argument x = 2
Local variable y =  3.4
Main function variable x = 2
Main function variable y = 4
```

checkpoint 6.17(b)

checkPoint6_17b.py > ...

```
1  def f(x,y = 1,z = 2):
2      return x + y + z
3  print(f(1,1,1))
4  print(f(y = 1, x = 2, z = 1))
5  print(f(1, z = 3))
```

```
3

4

5
```

checkpoint 6.18

checkPoint6_18.py > ...

```
1  x = y = 0
2  def function():
3      x = 4.5
4      y = 3.4
5      print(x)
6      print(y)
7  function()
8  print(x)
9  print(y)
```

x and y are not defined outside the function.

checkpoint 6.19

- 以下程式碼可以執行嗎? 若可以，請問其輸出結果為何?

checkPoint6_19.py > ...

```
1  x = 10
2  if x < 10 :
3      y = -1
4  else:
5      y = 1
6  print('y is ',y)
```

```
y is  1
```

6.10 預設參數

- Python允許您定義含有預設參數值的函式。當呼叫函式時，若沒有傳送參數，此時將使用預設參數值。

範例程式6.9 DefaultArgumentDemo.py

```
width: 1          height: 2          area: 2
width: 4          height: 2.5        area: 10.0
width: 3          height: 5          area: 15
width: 1.2        height: 2          area: 2.4
width: 1          height: 6.2        area: 6.2
```

範例程式6.9 DefaultArgumentDemo.py

DefaultArgumentDemo.py > ...

```
1  def printArea(width = 1, height = 2):
2      area = width * height
3      print('width:',width,'\theight:',height,'\tarea:',area)
4  printArea()
5  printArea(4,2.5)
6  printArea(height=5,width=3)
7  printArea(width = 1.2)
8  printArea(height=6.2)
```

checkpoint 6.20

- 請顯示以下程式碼的輸出結果：

checkPoint6_20.py > ...

```
1  def f(w = 1, h = 2):
2      print("w=",w,"\th=",h)
3  f()
4  f(w = 5)
5  f(h = 24)
6  f(4,5)
```

```
w=  1       h=  2
w=  5       h=  2
w=  1       h=  24
w=  4       h=  5
```

checkpoint 6.21

- 請將下列程式碼加以除錯：

checkPoint6_21.py > ...

```
1  def main():
2      nPrintln(5)
3  def nPrintln(message="Welcome to Python!" ,n):
4      for i in range(n):
5          print(i+1,'time:',message)
6  main()
```

```
SyntaxError: non-default argument follows default argument
```

checkpoint 6.21 (修正後)

- 請將下列程式碼加以除錯：

checkPoint6_21.py > ...

```
1  def main():
2      nPrintln("Welcome to Python!",5)
3  def nPrintln(message,n):
4      for i in range(n):
5          print(i+1,'time:',message)
6  main()
```

```
1 time: Welcome to Python!
2 time: Welcome to Python!
3 time: Welcome to Python!
4 time: Welcome to Python!
5 time: Welcome to Python!
```

6.11 回傳多個值

- Python允許函式回傳多個值。範例程式6.10定義一接收兩個數字的函式，並將這兩個數字以由小到至大加以回傳。

```
Enter number1: 456
Enter number2: 123
n1 is  123
n2 is  456
```

範例程式6.10 MultipleReturnValueDemo.py

MultipleReturnValueDemo.py > sort

```
1  def sort(number1,number2):
2      if number1 < number2:
3          return number1,number2
4      else:
5          return number2,number1
6  inNum1 = eval(input('Enter number1: '))
7  inNum2 = eval(input('Enter number2: '))
8  n1,n2 = sort(inNum1,inNum2)
9  print('n1 is ',n1)
10 print('n2 is ',n2)
```

checkpoint 6.23

- 函式可以傳回多個值嗎? 請顯示下列程式碼輸出結果：

checkPoint6_23.py > ...

```
1  def f(x, y):
2      return x+y, x - y, x * y, x /y
3  t1, t2, t3, t4 = f(9,5)
4  print(t1,t2,t3,t4)
```

```
14 4 45 1.8
```

6.12 個案研究：隨機產生ASCII字元

```
>>> from random import randint
>>> ord('A')
65
>>> ord('Z')
90
>>> chr(randint(65,90))
'Y'
>>> ord('a')
97
>>> randint(ord('a'),ord('z'))
110
>>> chr(randint(ord('a'),ord('z')))
'b'
```

範例程式6.11 RandomCharacter.py

RandomCharacter.py > ...

```python
from random import randint
def getRandomCharacter(ch1,ch2):
    return chr(randint(ord(ch1),ord(ch2)))
def getRandomLowerCaseLetter():
    return getRandomCharacter('a','z')
def getRandomUpperCaseLetter():
    return getRandomCharacter('A','Z')
def getRandomDigitCharacter():
    return getRandomCharacter('0','9')
def getRandomASCIICharacter():
    return chr(randint(0,127))
```

範例程式6.12 TestRandomCharacter.py

```
i e t s k q l b u d k g k d s c f n a m o e u k e
l m z v s f v q d r e k c e u s u q k v w e k j a
d n d y n m v j t u l i p v j u h y r k m f h e d
l j q w x j y i q h f s q r f t q u d a f u a y m
j y i h e k l f f v y y f e v c z b b t t e g n k
u l f d r h c x r v j b p r w v n e o p y y w f c
y q d p l f s y n b p g r i k s u u o d o l s m u
```

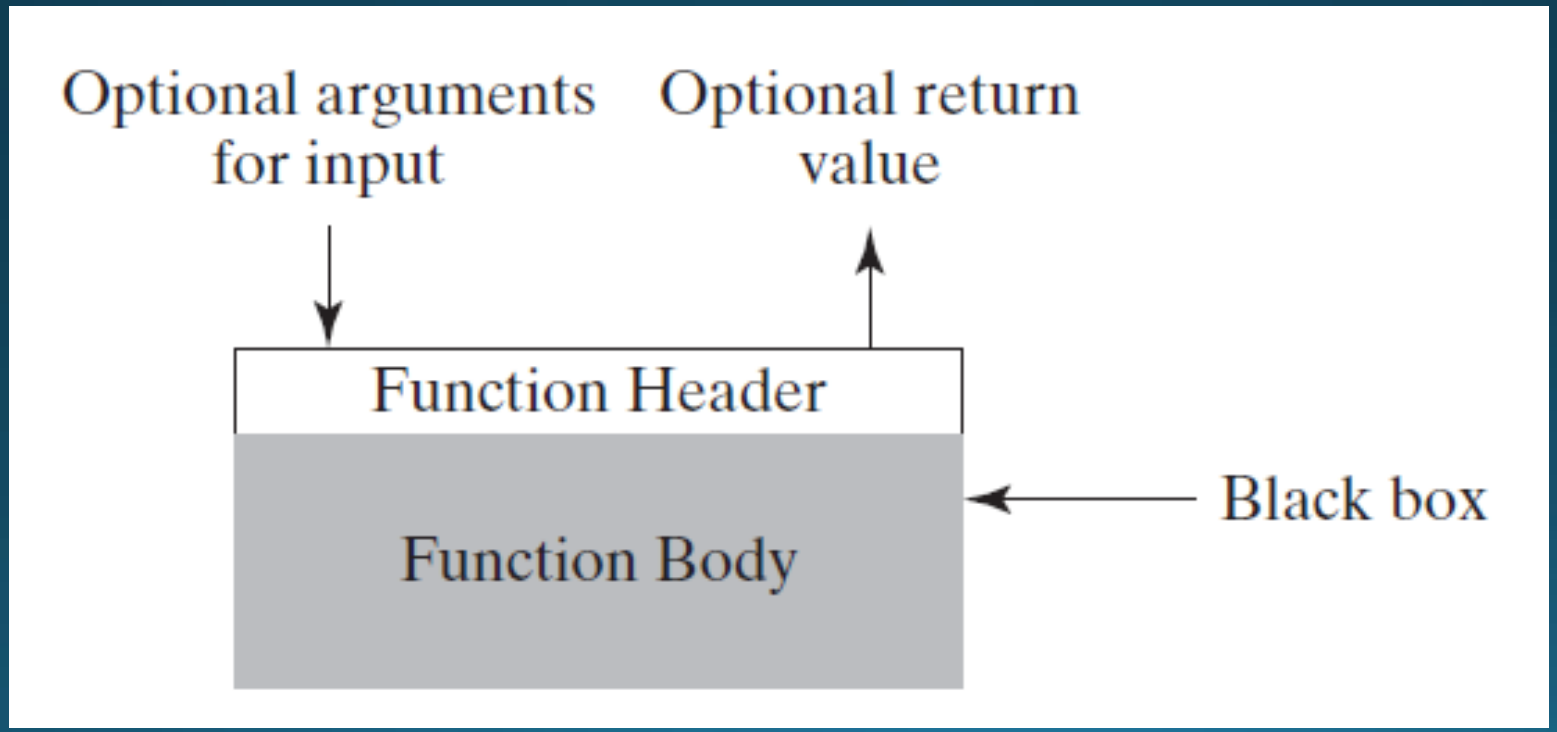
範例程式6.12 TestRandomCharacter.py

TestRandomCharacter.py > ...

```
1  import RandomCharacter
2  NUMBER_OF_CHARS = 175
3  CHARS_PER_LINE = 25
4  for i in range(NUMBER_OF_CHARS):
5      print(RandomCharacter.getRandomLowerCaseLetter() , end = " ")
6      if (i + 1) % CHARS_PER_LINE == 0:
7          print()
```

6.13 函式萃取

- 您可以將函式想像為“黑盒子(black box)”，函式實作內容以以此形式對使用者隱藏。



撰寫一程式來顯示某一年指定月份的日曆

```
Enter full year (e.g., 2001): 2011 ↵Enter
Enter month as number between 1 and 12: 9 ↵Enter

         September 2011
------------------------------
 Sun  Mon  Tue  Wed  Thu  Fri  Sat
                     1    2    3
   4    5    6    7    8    9   10
  11   12   13   14   15   16   17
  18   19   20   21   22   23   24
  25   26   27   28   29   30
```

函式的好處

- 撰寫一函式且重複呼叫使用。
- 對使用者隱藏資訊。
- 降低複雜性。

逐步細緻化

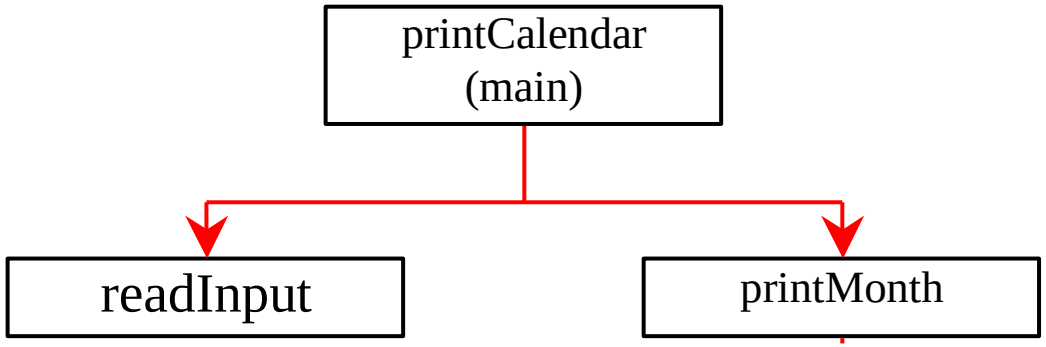
- 函式萃取的概念可應用在開發程式的過程中。當撰寫大型程式時，可使用 "分治法(divide-and-conquer)" 策略，又被稱作逐步細緻化(stepwise refinement)，將程式分解成好幾個子問題。各個子問題又可進一步分解成更小，使得管理更容易。

範例：印出月曆

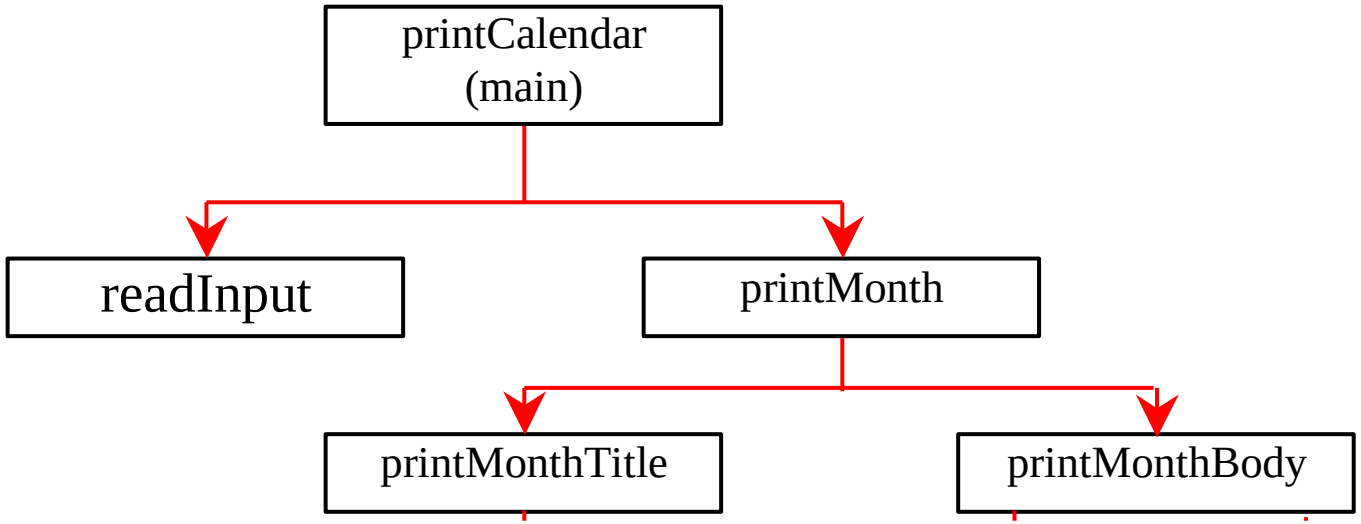
- 假設我們要撰寫一程式，顯示某一年指定月份的日曆。此程式會提示使用者輸入年份與月份，接著顯示該月份的完整日曆，如下圖所示：

```
            December   2016
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
                   1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30  31
```

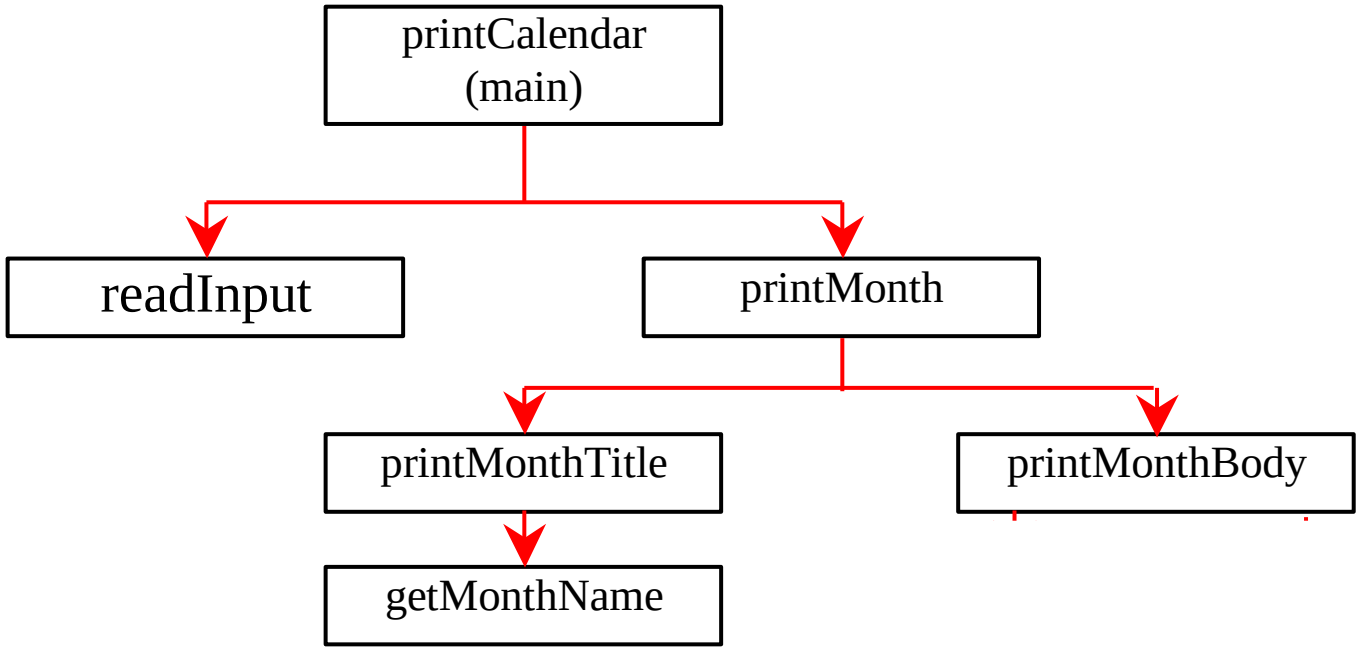
結構圖



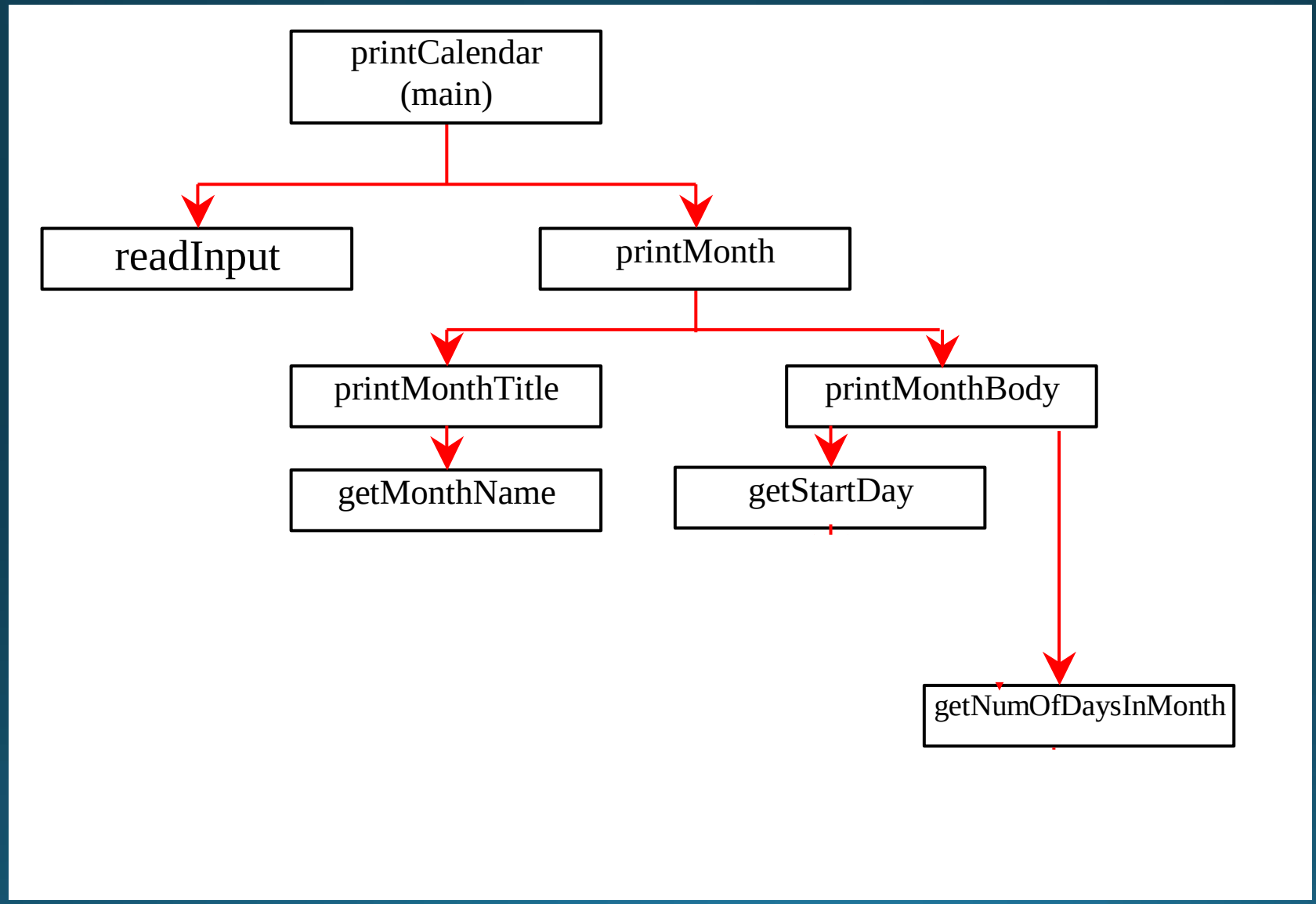
結構圖


printCalendar
(main)
readInput
printMonth
printMonthTitle
printMonthBody

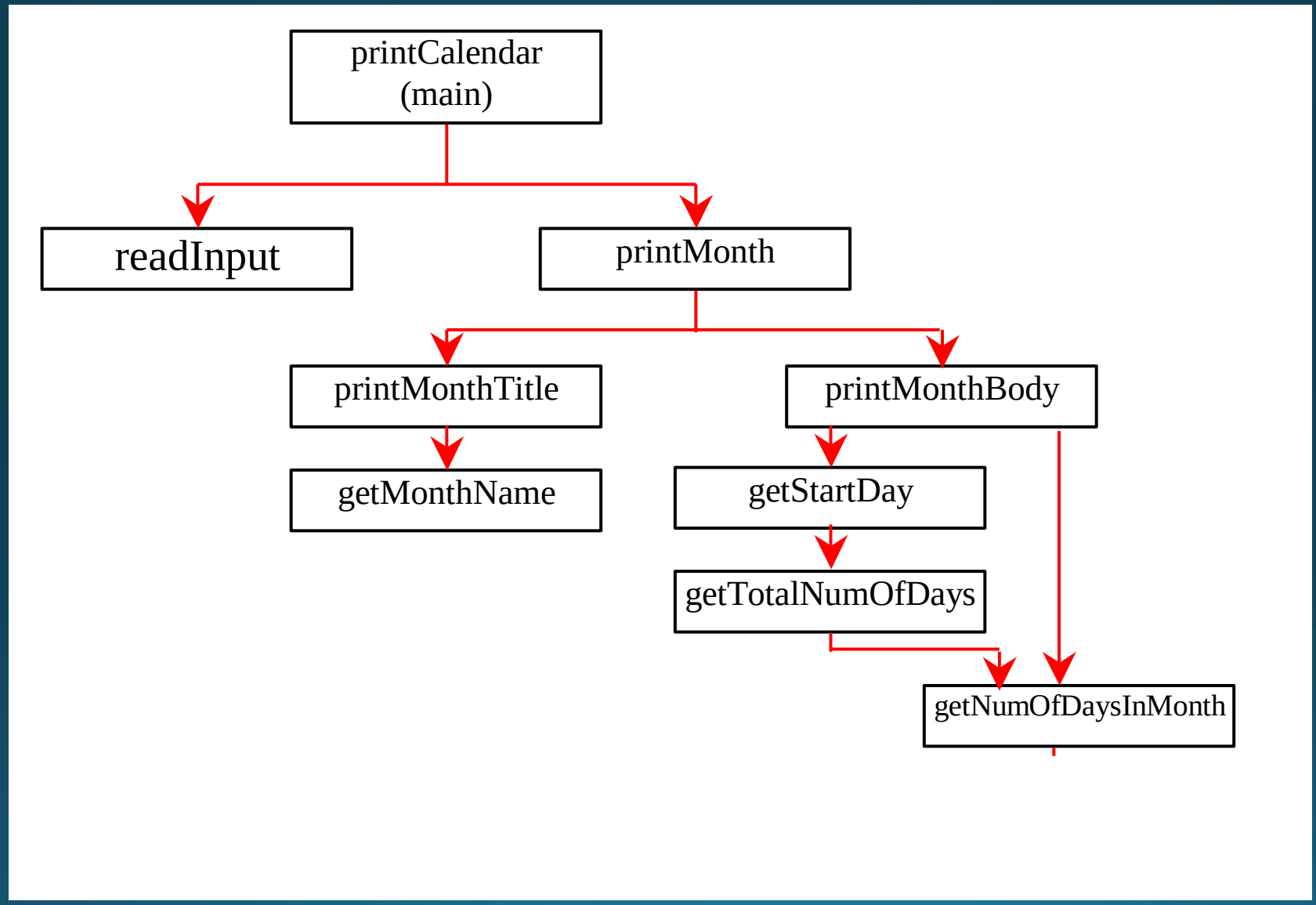
結構圖


printCalendar
(main)
readInput
printMonth
printMonthTitle
printMonthBody
getMonthName

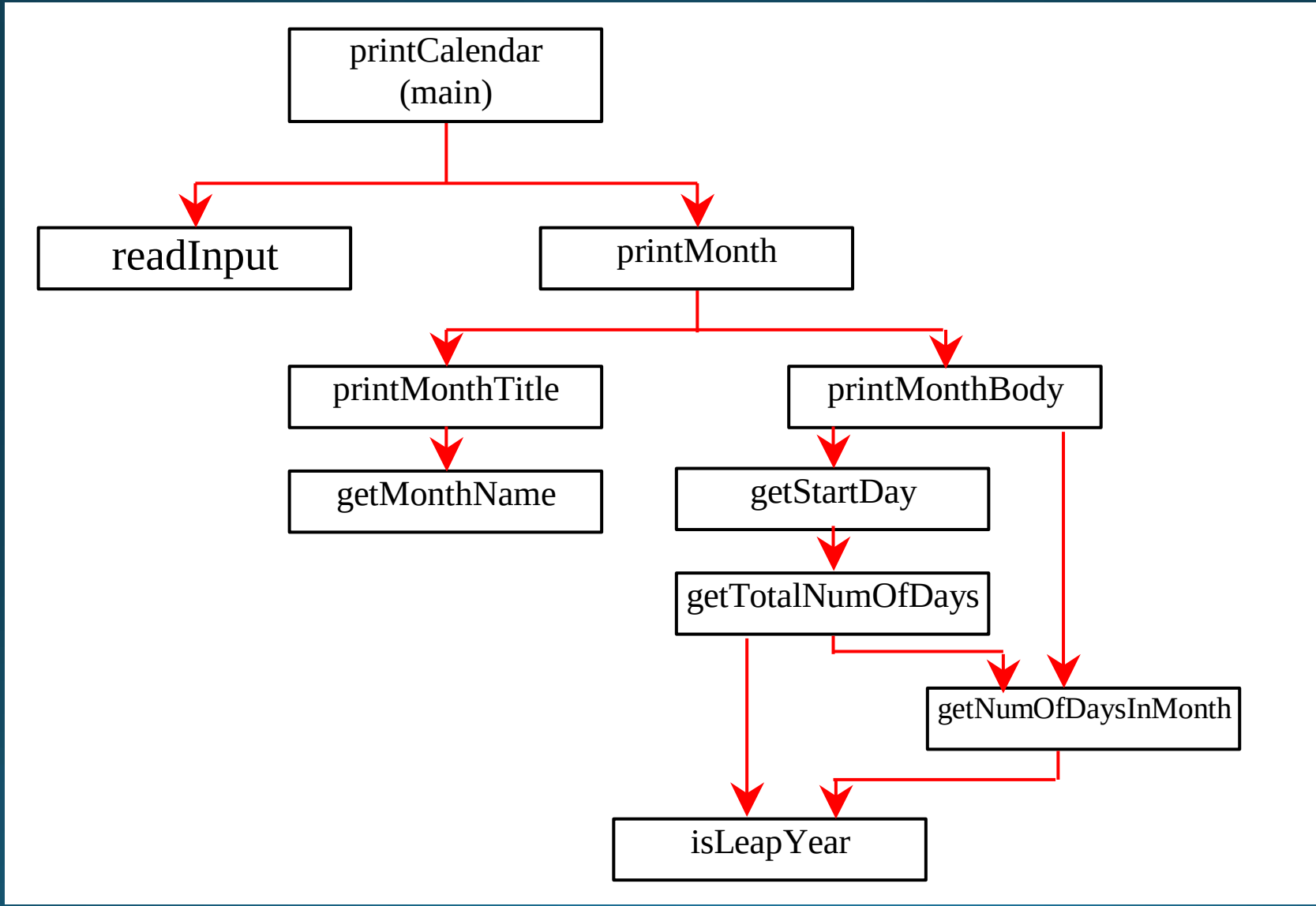
結構圖


printCalendar
(main)
readInput
printMonth
printMonthTitle
printMonthBody
getMonthName
getStartDay
getNumOfDaysInMonth

結構圖


printCalendar
(main)
readInput
printMonth
printMonthTitle
printMonthBody
getMonthName
getStartDay
getTotalNumOfDays
getNumOfDaysInMonth

結構圖


printCalendar
(main)
readInput
printMonth
printMonthTitle
printMonthBody
getMonthName
getStartDay
getTotalNumOfDays
getNumOfDaysInMonth
isLeapYear

範例程式6.13 PrintCalendar.py

PrintCalendar.py > ...

```
1  def printMonth(year,month):
2      printMonthTitle(year,month)
3      printMonthBody(year,month)
4
5  def printMonthTitle(year,month):
6      print('          ',getMonthName(month),' ',year)
7      print('---------------------------')
8      print(' Sun Mon Tue Wed Thu Fri Sat')
9
```

```
10  def printMonthBody(year,month):
11      startDay = getStartDay(year,month)
12      numberOfDaysInMonth = getNumberOfDaysInMonth(year,month)
13      i = 0
14      for i in range(startDay):
15          print('     ',end = '')
16      for i in range(1,numberOfDaysInMonth+1):
17          print(format(i,'4d'),end='')
18          if (i+startDay) % 7 == 0:
19              print()
20
```

```
21  def getMonthName(month):
22      if month == 1:
23          monthName = 'January'
24      elif month == 2:
25          monthName = 'February'
26      elif month == 3:
27          monthName = 'March'
28      elif month == 4:
29          monthName = 'Aprial'
30      elif month == 5:
31          monthName = 'May'
32      elif month == 6:
33          monthName = 'June'
34      elif month == 7:
35          monthName = 'July'
36      elif month == 8:
37          monthName = 'August'
38      elif month == 9:
39          monthName = 'September'
40      elif month == 10:
41          monthName  11: int er'
42      elif month == 11:
43          monthName = 'November'
44      else:
45          monthName = 'December'
46      return monthName
```

```
48  def getStartDay(year,month):
49      startDayForJan_1_1800 = 3
50      totalNumberOfDays = getTotalNumberOfDays(year,month)
51      return (totalNumberOfDays+startDayForJan_1_1800) % 7
52
53  def getTotalNumberOfDays(year,month):
54      total = 0
55      for i in range(1800,year):
56          if isLeapYear(i):
57              total = total + 366
58          else:
59              total = total + 365
60      for i in range(1,month):
61          total = total + getNumberOfDaysInMonth(year,i)
62      return total
```

```
64 def getNumberOfDaysInMonth(year,month):
65     if(month ==1 or month == 3 or month == 5 or month == 7 or
66         month == 8 or month == 10 or month == 12):
67         return 31
68     if month == 4 or month == 6 or month == 9 or month == 11:
69         return 30
70     if month == 2:
71         return 29 if isLeapYear(year) else 28
72     return 0
73
74 def isLeapYear(year):
75     return year % 400 == 0 or (year % 4 == 0 and year % 100 != 0)
76
77 def main():
78     year = eval(input('Enter full year(e.g. 2001):'))
79     month = eval(input('Enter month as number between 1 and 12: '))
80     printMonth(year,month)
81
82 main()
```

```
Enter full year(e.g. 2001):2016
Enter month as number between 1 and 12: 12
        December   2016
-------------------------------
 Sun Mon Tue Wed Thu Fri Sat
                   1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30  31
```

由上而下的設計 (Top-Down Design)

- 要如何開始進行此程式的開發呢？您會立即開始撰寫程式碼嗎？新手程式設計師常會試著在一開始就對所有細節進行撰寫。雖然細節對最終的程式內容很重要，但在開發初期就著手於細節部分會阻礙解決問題的過程。為了讓問題解決過程更流暢，此範例一開始會使用函式萃取將細節從設計中分離出來，之後再進行細節的實作。

由上往下及/或由下往上實作

- 接下來讓我們將注意力移到實作部分。一般來說，子問題對應到實作內容裡的函式，雖然有些對應過於簡單以至於不必要。您必須決定要實作哪些模組函式，以及哪些要併入其他函式。這類的決定取決於整體程式是否會因為所做的決定而易於閱讀。
- 您可以選擇使用"由上往下(top-down)"或"由下往上(bottom-up)"函式。

個案研究：可重複使用的圖形函式

```
def drawLine(x1, y1, x2, y2):

def writeString(s, x, y):

def drawPoint(x, y):

def drawCircle(x = 0, y = 0, radius = 10):

def drawRectangle(x = 0, y = 0, width = 10, height = 10):
```

範例程式6.14 UsefulTurtleFunctions.py

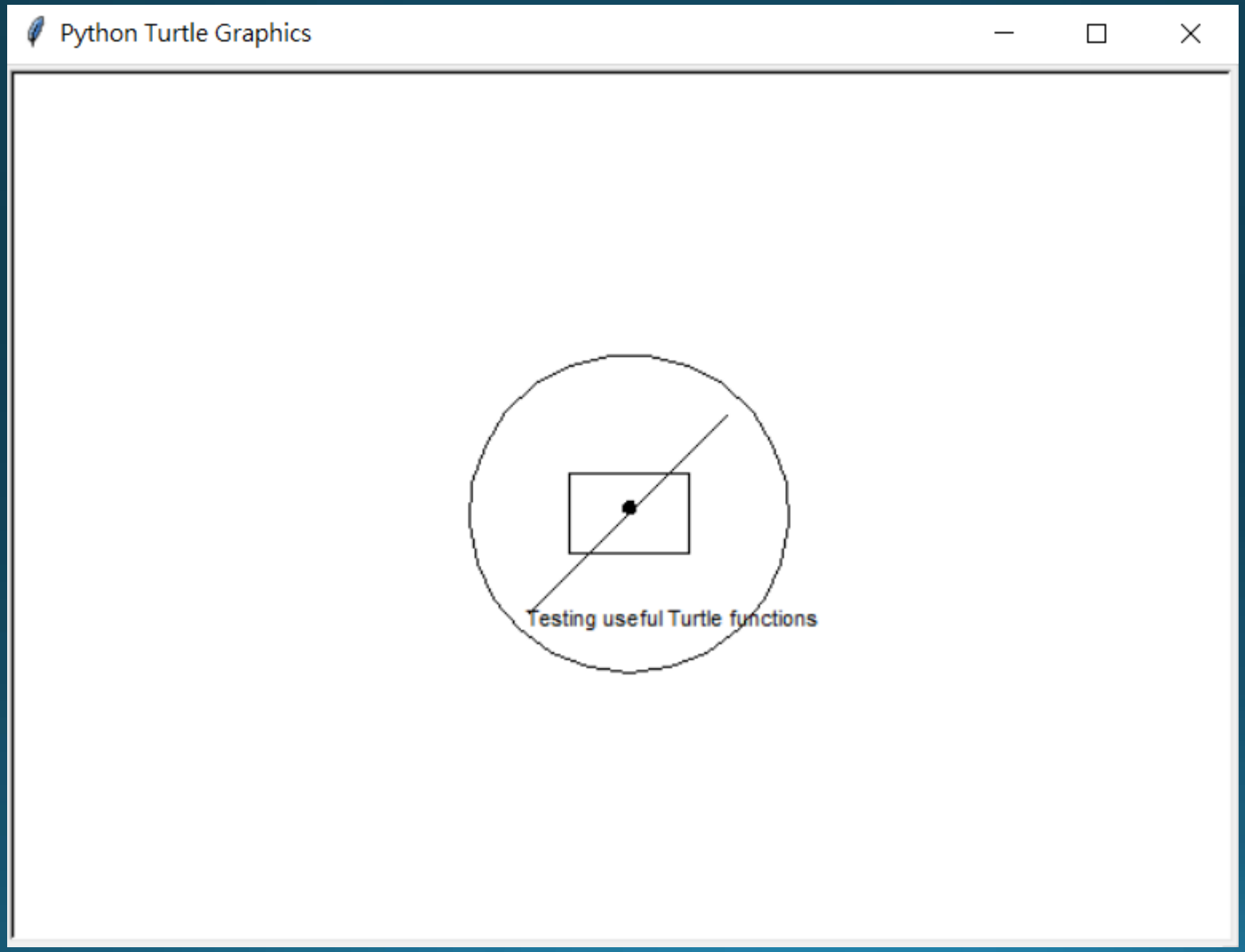
usefulTurtleFunction.py > ...

```
1  import turtle
2  myTurtle = turtle.Turtle()
3  def drawLine(x1,y1,x2,y2):
4      myTurtle.penup()
5      myTurtle.goto(x1,y1)
6      myTurtle.pendown()
7      myTurtle.goto(x2,y2)
8  def writeText(s,x,y):
9      myTurtle.penup()
10     myTurtle.goto(x,y)
11     myTurtle.pendown()
12     myTurtle.write(s)
```

```
13  def drawPoint(x,y):
14      myTurtle.penup()
15      myTurtle.goto(x,y)
16      myTurtle.pendown()
17      myTurtle.begin_fill()
18      myTurtle.circle(3)
19      myTurtle.end_fill()
20  def drawCircle(x,y,radius):
21      myTurtle.penup()
22      myTurtle.goto(x,y-radius)
23      myTurtle.pendown()
24      myTurtle.circle(radius)
```

```
25  def drawRectangle(x,y,width,height):
26      myTurtle.penup()
27      myTurtle.goto(x+width/2,y+height/2)
28      myTurtle.pendown()
29      myTurtle.right(90)
30      myTurtle.forward(height)
31      myTurtle.right(90)
32      myTurtle.forward(width)
33      myTurtle.right(90)
34      myTurtle.forward(height)
35      myTurtle.right(90)
36      myTurtle.forward(width)
```

範例程式6.15 UseCustomTurtleFunctions.py



UseCustomTurtleFunction.py > ...

```
1  import turtle
2  from usefulTurtleFunction import *
3  myTurtle = turtle.Turtle()
4  drawLine(-50,-50,50,50)
5  writeText('Testing useful Turtle functions',-50,-60)
6  drawPoint(0,0)
7  drawCircle(0,0,80)
8  drawRectangle(0,0,60,40)
9  myTurtle.hideturtle()
10 turtle.done()
```