Python 程式設計：函數

目錄

1. 基本概念

在 Python 中，函數 是一個可以重複使用的程式碼區塊。它的主要目的是將一段特定的功能打包起來，當你需要執行這個功能時，只要呼叫（call）這個函數就行了。

使用函數有以下幾個主要優點：

- 重複使用程式碼：避免重複寫相同的程式碼，讓程式更簡潔、有效率。
- 模組化：將大型程式拆解成一個個獨立的小功能，每個函數只負責單一任務，讓程式更容易閱讀、理解和維護。
- 提高可讀性：透過有意義的函數名稱，一眼就能知道這段程式碼的功能。

2. 定義函數

要定義一個函數，需要使用 def 關鍵字，後面跟著函數名稱、括號 ()，以及冒號 :。括號內可以用來定義參數。

```
def my_function():
    # 這是函數內部的程式碼
    print("這是一個簡單的函數。")
```

3. 呼叫函數

定義好函數後，只需要寫下它的名稱和括號 () 就可以呼叫它。

```
# 呼叫上面定義的函數
my_function()
```

4. 函數的參數（Parameters）

參數就像是函數的輸入。你可以在呼叫函數時，傳遞資料給它，讓函數可以根據這些資料執行不同的動作。

4.1 單一參數

```
def greeting(name):
    print("嗨，", name, "早安！")

greeting("Nelson")
# 這裡的 name 就是參數。當你呼叫 greeting("Nelson") 時，"Nelson" 這個字串就被賦予給 name，然後在函數內部使用。
```

4.2 多個參數

```
def subtract(x1, x2):
    result = x1 - x2
    print("x1-x2=", result)

# 呼叫函數並傳入兩個數值
subtract(7, 3)
```

4.3 關鍵字參數（Keyword Arguments）

```
def interest(interest_type, subject):
    print("主詞是:", subject)
    print("興趣是:", interest_type)

# 使用關鍵字參數，順序可以不固定
interest(subject="程式設計", interest_type="Python")
```

4.4 函數參數是串列（list）

在 Python 中，你可以將串列（list） 作為參數傳遞給函數。這讓函數可以處理一組資料，而不是單一的數值。

```
def product_msg(customers):
    # 這裡的 customers 是一個串列
    for customer in customers:
        msg = 'str2' + customer + 'str2'
        print(msg)

customers = ['Damon', 'Peter', 'Mary']
product_msg(customers)
# 這裡的 customers 串列被傳遞給 product_msg 函數，函數內部再透過 for 迴圈來處理串列中的每一個元素。
```

4.5 傳遞任意數量的參數：*args 與**kwargs

- 任意位置參數（*args）

使用 * 加上一個參數名（通常用 *args），可以讓函數接收任意數量的位置參數。這些參數會被打包成一個元組（tuple）。

```
def make_icecream(toppings):
    print("冰淇淋的配料：")
    for topping in toppings:
        print(topping)

make_icecream("草莓醬", "OREO 餅乾", "巧克力碎片")
```

- 任意關鍵字參數（**kwargs）

使用 ** 加上一個參數名（通常用 **kwargs），可以讓函數接收任意數量的關鍵字參數。這些參數會被打包成一個字典（dictionary）。

```
def build_dict(name, **kwargs):
    kwargs['Name'] = name
    return kwargs
```

```
player_dict = build_dict('James', Age='32', City='Cleveland', State='Ohio')
print(player_dict)
```

5. 函數的回傳值（Return Value）

函數可以將計算結果或資料回傳給呼叫它的地方，這就是 回傳值。要回傳資料，需要使用 return 關鍵字。

5.1 單一回傳值

```
def greeting(name):
    return "嗨，" + name + "早安！"

result = greeting("Helena")
print(result)
# greeting() 函數將一個字串回傳，然後我們將這個字串存到 result 變數中並印出
```

5.2 多個回傳值

Python 的函數可以同時回傳多個值，這些值會被打包成一個 元組（tuple）。圖片中 ch11_13.py 就是一個例子。

```
def multifunc(x1, x2):
    addresult = x1 + x2
    subresult = x1 - x2
    return addresult, subresult

# 呼叫函數並接收回傳的多個值
add, sub = multifunc(10, 2)
print("加法結果：", add)
print("減法結果：", sub)
```

6. 函數是物件（Object）

在 Python 中，函數本身就是一種物件，這意味著你可以將函數賦予給變數，或者作為參數傳遞給另一個函數。

```
def add(x, y):
    return x + y

def mul(x, y):
    return x * y

def running(func, arg1, arg2):
    return func(arg1, arg2)

result = running(add, 5, 10)
print(result)
```

7. 函數可以回傳函數

一個函數也可以回傳另一個函數，這在進階程式設計中很常用。

```
def create_multiplier(number):
    def multiplier(num):
        return number * num
    return multiplier

double_function = create_multiplier(2)
triple_function = create_multiplier(3)

print("兩倍數字：", double_function(5))
print("三倍數字：", triple_function(5))
```

8. Docstring (文件字串)

在函數定義的第一行，你可以使用三個引號 """ 來撰寫文件字串（Docstring），用來描述函數的功能。這是一個很好的程式習慣，可以幫助自己或其他人了解函數的用途。

```
def greeting(name):
    """
    這是一個打招呼的函數，
    參數 name 是要打招呼的對象名稱
    """
    print("Hi,", name, "Good Morning!")

# 可以用 __doc__ 屬性來讀取文件字串
print(greeting.__doc__)
```

9. 匿名函數 (Lambda)

匿名函數，顧名思義就是沒有名稱的函數。它們通常只用於一個簡單的、一次性的任務。

```
lambda 參數: 運算式
# lambda 是定義匿名函數的關鍵字。
# 參數 是輸入，可以有多個，用逗號分隔。
# 運算式 是函數體，只能有一個運算式，且其結果會被自動回傳，不用 return
```

```
square = lambda x: x ** 2
print(square(10))
```

10. 高階函數 (Higher-Order Functions)

高階函數是指可以接收其他函數作為參數，或將函數作為回傳值的函數。匿名函數 (lambda) 經常與高階函數一起使用，因為它能讓程式碼更精簡。這裡介紹三個最常見的高階函數：filter()、map() 和 reduce()。

10.1 filter()

filter()函數用於過濾串列中的元素，只保留符合特定條件的。

```
filter(函數, iterable)
# 函數：一個判斷條件的函數，回傳 True 或 False。
# iterable：可迭代的物件，如串列、元組等。
```

```
myList = [10, 15, 20, 25, 30]

# 使用 lambda 篩選出偶數
oddList = list(filter(lambda x: (x % 2 == 0), myList))
print(oddList)
```

10.2 map()

map() 函數用於對串列中的每個元素執行相同的操作，並回傳新的結果。

```
map(函數, iterable)
# 函數：一個處理每個元素的函數。
```

10.3 reduce()

reduce() 函數用於對串列中的元素進行累積運算，將所有元素歸納成一個單一值。

```
from functools import reduce


reduce(函數, iterable)
# 函數：必須有兩個參數，分別是累積值和下一個元素。
```

```
from functools import reduce

string = '54387'
x = reduce(lambda x, y: x + y, string)
print(x)
```

11. 變數作用域 (Scope)

變數的作用域決定了它在程式碼的哪些地方是可見的。

- 區域變數 (Local Variables)：在函數內部定義的變數，只能在該函數內部使用。
- 全域變數 (Global Variables)：在所有函數之外定義的變數，可以在程式的任何地方使用。如果你想在函數內部

修改全域變數，需要使用 global 關鍵字。

```
global_msg = "Global Variable"

def printMsg():
    # 可以在函數內部讀取全域變數
    print("函數內可以讀取:", global_msg)

print("主程式可以讀取:", global_msg)
printMsg()
```

12. pass 與函數

在 Python 中，pass 是一個空語句。它什麼都不做，只是一個佔位符。在設計程式時，你可能會先規劃好所有函數的名稱和結構，但還沒來得及寫具體的程式碼。這時，就可以在函數內部使用 pass，讓程式碼不會因為空的函數而報錯。

```
def fun(arg):
    pass
```

13. 裝飾器 (Decorator)

裝飾器 是一個非常強大的功能，它讓你可以在不修改原始函數程式碼的情況下，增加額外的功能。簡單來說，裝飾器就是一個特殊的函數，它接收另一個函數作為參數，然後回傳一個被修改過的函數。

13.1 @ 裝飾器語法

使用 @ 符號是應用裝飾器最常見的方式，它讓程式碼更簡潔、易讀。

@ 符號放在函數定義上方，等同於：my_function = decorator(my_function)。

```
def upper(func):
    def wrapper(args):
        result = func(args)
        return result.upper()
    return wrapper

@upper
def greeting(string):
    return string

print(greeting('Hello! iPhone'))
```

13.2 迭代器 (Iterator) 與生成器 (Generator)

當處理大量資料時，如果一次將所有資料載入記憶體，會佔用大量資源並降低效能。這時，迭代器 和 生成器 就派

上用場了。

- 迭代器 是一個可以被迭代的物件，它提供了 __iter__ 和 __next__ 方法
  - iter() 函數將可迭代物件（如串列）轉換成迭代器。
  - next() 函數可以從迭代器中逐一取出下一個值。

```
my_list = [1, 3, 5]
my_iterator = iter(my_list)

print(next(my_iterator))
print(next(my_iterator))
print(next(my_iterator))
```

- 生成器 (Generator)

生成器是一種特殊的函數，它會回傳一個迭代器。生成器函數使用 yield 關鍵字而不是 return。yield 每次執行時會回傳一個值，然後暫停函數的執行，並保留狀態。當下一次呼叫 next() 時，函數會從上次暫停的地方繼續執行。

```
def list_square(n):
    for i in range(1, n + 1):
        yield i ** 2

for data in list_square(5):
    print(data)

# list_square 函數是一個生成器。當 for 迴圈開始時，它會呼叫 list_square(5)，第一次執行到 yield 1**2 時，會回傳 1 並暫停。第二次迭代時，函數會從上次暫停的地方繼續，執行 yield 2**2，回傳 4，依此類推。
```

優點：

- 記憶體效率高：生成器一次只產生一個值，不會將所有結果一次性存入記憶體，特別適合處理龐大的資料集。
- 延遲計算 (Lazy Evaluation)：只有在需要時才計算下一個值，可以節省運算資源。

14. 練習

14.1 Problem: 質數判定器 (Prime Number Checker)

Problem Description: 質數（Prime Number）是指在一個大於 1 的自然數中，除了 1 和此整數自身外，無法被其他自然數整除的數。請編寫一個名為 isPrime(n) 的函數，該函數接收一個整數 n 作為參數，如果 n 是質數，則回傳 True，否則回傳 False。主程式需讀取一個整數，並使用此函數判斷其是否為質數，最後印出對應的結果。	
Input: 輸入包含一個正整數 n ($1 < n \leq 1{,}000{,}000$)。	Output: 如果該數是質數，請輸出 "Prime"。如果不是，請輸出 "Not Prime"。
Sample Input:	Sample Output:

13	Prime
Sample Input: 91	Sample Output: Not Prime
Answer:	

14.2 Problem: 最大公因數與最小公倍數 (GCD and LCM)

Problem Description: 請實作兩個函數：gcd(a, b) 和 lcm(a, b)。gcd 函數使用輾轉相除法（Euclidean algorithm）計算兩個正整數的最大公因數（Greatest Common Divisor）。lcm 函數則利用 gcd 的結果來計算最小公倍數（Least Common Multiple）。	
Input: 輸入包含兩個正整數 a 和 b ($1 \le a, b \le 1{,}000{,}000$)，以空白分隔。	Output:
Sample Input: 8 12	Sample Output: 4 24
Sample Input: 40 16	Sample Output: 8 80
Answer:	

14.3 Problem: 歌詞單字計數器 (Lyric Word Counter)

Problem Description: 請設計一個程式來計算一段英文歌詞中每個單字出現的次數。你需要撰寫兩個函數： ● clean_text(text): 此函數接收一個字串，將其中所有的標點符號（例如 . 和 ,）替換成空白，並將所有字母轉為小寫，最後回傳清理過的字串。 ● word_count(text): 此函數接收清理過的字串，計算每個單字出現的次數，並回傳一個字典（dictionary）。 主程式需讀取一行字串，依序呼叫上述兩個函數，最後按照**單字字母順序**印出所有單字的出現次數。	
Input: 輸入為一行包含英文字母、空白與標點符號的字串。	Output: 按照單字的字母順序，逐行輸出每個單字及其出現次數，格式為 word: count。
Sample Input: Are you sleeping, are you sleeping, Brother John, Brother John?	Sample Output: are: 2 brother: 2 john: 2 sleeping: 2 you: 2
Sample Input:	Sample Output:

Ding dong ding, ding dong ding.	ding: 4 dong: 2
Answer:	

14.4 Problem: 夏令營學生名單 (Summer Camp Attendee Lists)

Problem Description: 給定兩個集合，分別代表參加數學夏令營（math）和物理夏令營（physics）的學生名單。請編寫一個函數 process_lists(math_set, physics_set)，該函數接收兩個集合作為參數，並回傳一個元組（tuple），元組中包含兩個新的集合： ● 同時參加兩個夏令營的學生名單（交集）。 ● 只參加其中一個夏令營（但非兩個都參加）的學生名單（對稱差集）。 主程式讀取兩行學生名單，建立集合並呼叫此函數，最後分別印出兩個結果。	
Input: 輸入有兩行。第一行是參加數學夏令營的學生名單，以空白分隔。第二行是參加物理夏令營的學生名單，以空白分隔。	Output: 輸出有兩行。第一行印出交集名單，第二行印出對稱差集名單。輸出時請將集合排序，以確保輸出順序一致。格式參考範例。
Sample Input: Kevin Peter Eric Kevin Eric Tim	Sample Output: Intersection: ['Eric', 'Kevin'] Symmetric Difference: ['Peter', 'Tim']
Sample Input: A B C B D E	Sample Output: Intersection: ['B'] Symmetric Difference: ['A', 'C', 'D', 'E']
Answer:	

14.5 Problem: 數字過濾與轉換 (Number Filtering and Mapping)

Problem Description: 請使用高階函數 filter() 和 map() 搭配 lambda 匿名函數來處理一個數字串列。首先，從串列中篩選出所有大於 50 的數字，然後將這些篩選出來的數字每個都平方，最後印出結果串列。	
Input: 輸入一行由空白分隔的整數。	Output: 輸出一行處理過後的新串列。
Sample Input: 10 55 20 70 95 40	Sample Output: [3025, 4900, 9025]
Sample Input: 100 25 60	Sample Output: [10000, 3600]
Answer:	

14.6 Problem: 數字列表的累積乘積 (Cumulative Product with reduce)

Problem Description: 費波那契數列（Fibonacci sequence）的定義可以找一下網路 請編寫一個名為 fib_generator(limit) 的生成器（Generator）函數。此函數接收一個上限值 limit，並使用 yield 關鍵字依序產生小於 limit 的費波那契數。	
Input: 輸入一個正整數作為上限 limit	Output: 在一行中輸出所有小於 limit 的費波那契數，以空白分隔。
Sample Input: 50	Sample Output: 0 1 1 2 3 5 8 13 21 34
Sample Input: 200	Sample Output: 0 1 1 2 3 5 8 13 21 34 55 89 144
Answer:	

14.7 Problem: 執行時間裝飾器 (Execution Time Decorator)

Problem Description: 裝飾器（Decorator）可以在不修改原函數程式碼的情況下，為其增加額外的功能。請編寫一個名為 timer 的裝飾器，它會計算並印出被裝飾函數的執行時間（單位：秒）。接著，定義一個 long_running_task(n) 函數，該函數計算從 1 到 n 的總和，並使用 @timer 裝飾它。	
Input: 輸入一個正整數 n ($1 \le n \le 5{,}000{,}000$)。	Output: 輸出有兩行。第一行為裝飾器印出的執行時間（由於執行環境不同，時間的小數點後位數可能不同，但格式應類似），第二行為 long_running_task 的計算結果。
Sample Input: 1000000	Sample Output: Execution time: 0.0468... s Sum: 500000500000
Sample Input: 5000000	Sample Output: Execution time: 0.2343... s Sum: 12500002500000
Answer:	