05 函式

黃彬華編撰

- 什麼叫函式？
- 函式的參數與回傳值
- 遞迴概論
- 河內塔
- 不是只有變數，函式也可以指派
- 函式也可以當做參數
- 函式也可以被回傳
- 函式界的無名英雄 - Lambda
- Decorator - 取用修飾函式

什麼叫函式？

黃彬華編撰

- 函式 (function)
  - 自成一個程式區塊 (block)
  - 執行特定功能或特定計算
- 函式有
  - 名稱
    - 方便被呼叫
  - 參數 (parameter)
    - 可以不定義參數；有定義參數則方便傳遞不同的變量，處理不同情況
  - 回傳值 (return value)
    - 可以沒有回傳值；但有定義回傳值可將計算完畢結果回傳

函式的參數與回傳值 - 1

黃彬華編撰

- 無參數、無回傳值
  - 定義：**def** sayHello():
  - 呼叫：sayHello()
- 有參數、無回傳值
  - 定義：def showInfo(**name**, **price**):
  - 呼叫：showInfo("**Python**", **500**)
    - 呼叫時給予引數值 (argument)

函式的參數與回傳值 - 2

黃彬華編撰

- 參數有預設值（default arguments）
  - 定義：def showBookInfo(name, price, **discount=0.2**):
    - 一個參數有預設值，其後的參數也必須有預設值，否則產生 "non-default parameter follows default parameter" 錯誤
  - 呼叫：showBookInfo("Python", 500)
    - 參數有預設值，呼叫時不傳值代表使用預設值
  - 呼叫：showBookInfo("Python", 500, 0.3)
    - 呼叫時也可傳值代表不使用預設值

函式的參數與回傳值 - 3

黃彬華編撰

- Python 語法較寬鬆，函式若有回傳值，定義或不定義回傳類型皆可，但內容必須加上 return
  - 有回傳值但沒有定義回傳類型
    - def getInfo(name, price):
  - 有回傳值，使用「->」定義回傳類型
    - def getInfo(name, price) -> str:
  - 呼叫
    - result = getInfo("Python", 500)

函式的參數與回傳值 - 4

黃彬華編撰

- 有參數，多個回傳值 (其實是回傳 tuple)

```
def getSaleInfo(name, price, discount):
    salePrice = price * (1 - discount)
    info = f"書名: {name}\n定價: {price}"
    return salePrice, info

# 使用索引取值
result = getSaleInfo("Python", 500, 0.2)
print(f"{result[1]}\n特價: {result[0]}")
# 使用變數取值
salePrice, info = getSaleInfo("Python", 500, 0.2)
print(f"info:\n{info}\nsalePrice: {salePrice}")
```

函式的參數與回傳值 - 5

黃彬華編撰

- 任意引數 (arbitrary arguments，簡稱為「*args」) 的參數為tuple（類似List），可以接受多個引數
- 一個函式只能定義一個任意引數，否則錯誤
  - 定義：def showNames(*names):
  - 呼叫：showNames("Python", "Java", "JS")

函式的參數與回傳值 - 6

黃彬華編撰

- 呼叫函式時可以指定參數名稱（keyword arguments，簡稱為「kwargs」）
- 因為指定名稱，所以引數可以不按照參數順序
  - 定義：def showBook(name, price, discount):
  - 呼叫：showBook(**name**="Python", **discount**=0.2, **price**=500)

範例

黃彬華編撰

- FunctionsDemo

練習 5-1

黃彬華編撰

- 定義1個 calculate(numbers)，共有 2 個回傳值
  - 計算numbers所有元素的總和與平均值並回傳 (回傳總和與平均)
- 讓使用者輸入一個數列，輸入完畢後呼叫上述 calculate 函式運算，並將回傳的總和與平均顯示在畫面上

```
請輸入整數數列(空白分隔): 1 2 3 4 5
數列為: 1 2 3 4 5
總和 = 15
平均 = 3.0
```

遞迴函式概論

黃彬華編撰

- 重複不斷做同樣的事情，可以透過重複不斷呼叫同一個函式以達到程式重複利用目的，此為遞迴函式 (recursive function) 的應用
- 需要有終止條件，否則會一直遞迴下去無法中止
- 遞迴函式功能通常也可使用迴圈功能取代

範例 RecursiveFunctionDemo

- 分別使用遞迴函式與迴圈功能列印 10 到 1

```
10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
```

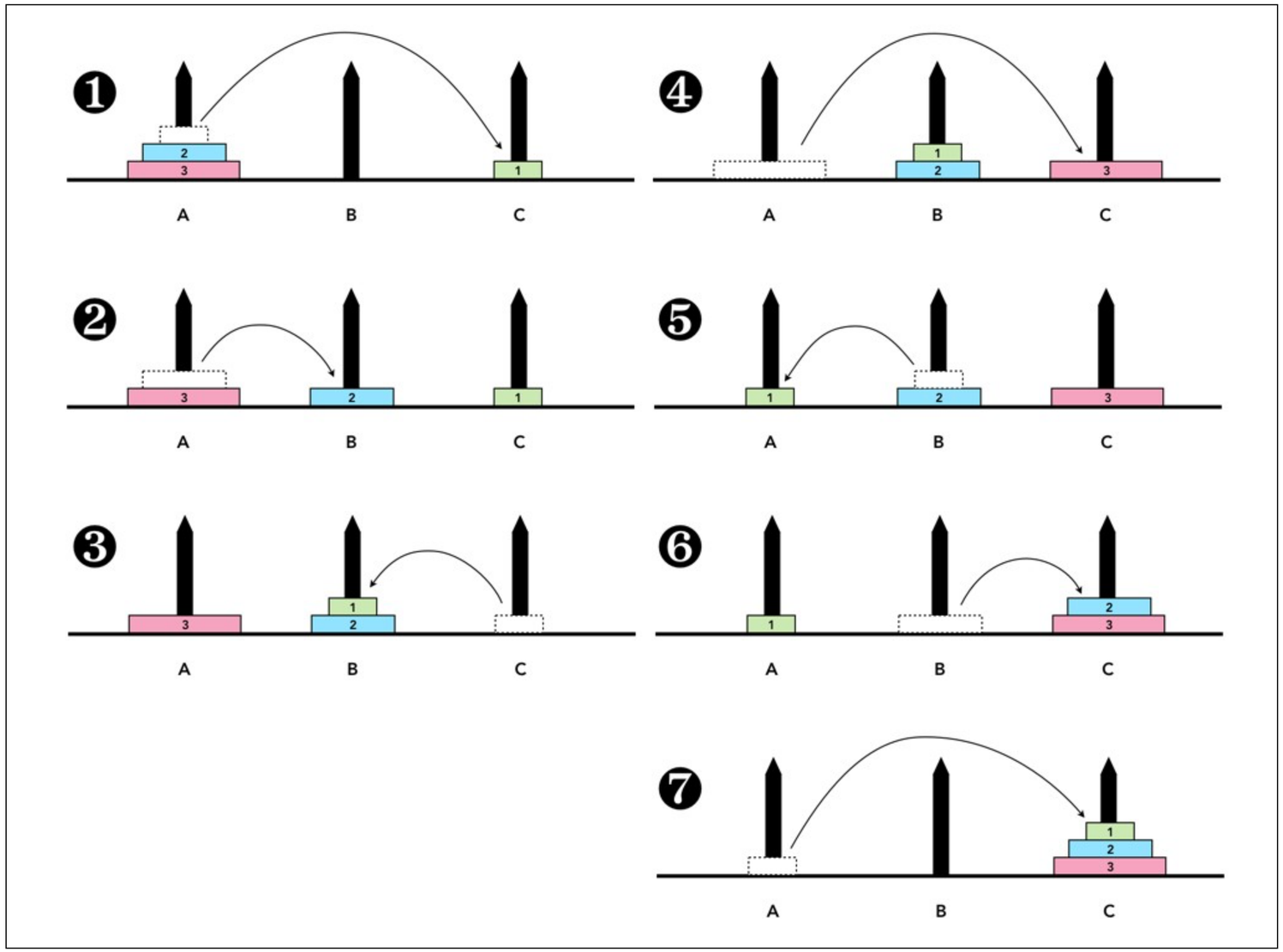
遞迴函式經典範例：河內塔

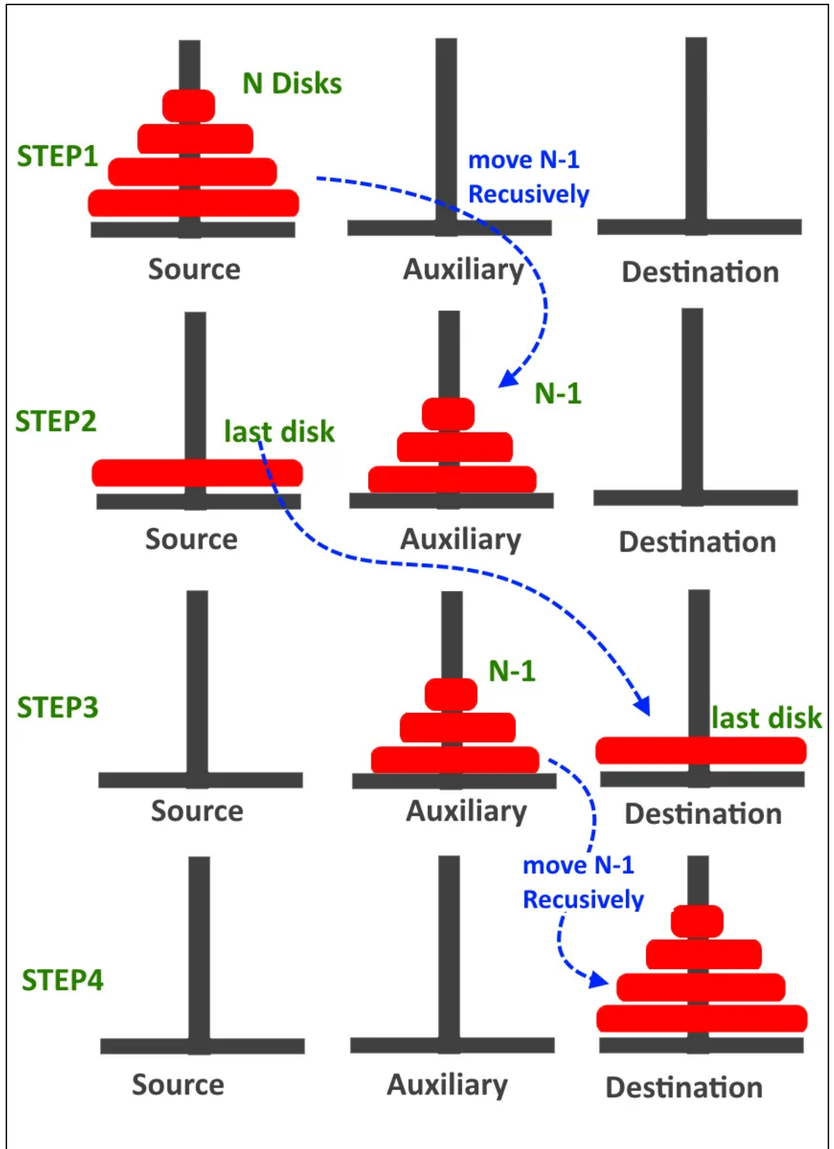
黃彬華編撰

- 題目：河內塔 (Tower of Hanoi)
- 說明：
  - 有 a, b, c 三根柱，n 個圓盤 (由使用者輸入) 放在 a 柱，要將所有圓盤從 a 柱移動到 c 柱，並計算總共移動幾次
- 條件：
  - 柱子的小圓盤在上，大圓盤在下
  - 圓盤可移動到任一根柱子
  - 每次只能移動一個圓盤, 而且只能從最上面圓盤開始移動

河內塔 - 圖解

黃彬華編撰


1
2
3
A
B
C
4
5
6
7


N Disks
STEP1
move N-1
Recusively
Source
Auxiliary
Destination
STEP2
last disk
N-1
STEP3
last disk
STEP4
move N-1
Recusively

範例 TowerOfHanoiDemo

- 基本上就是使用遞迴來解
  - 把 a 柱 (source) 的 n-1 個圓盤移動至 b 柱 (auxiliary)
  - 再把 a 柱 (source) 剩下的 1 個大圓盤移動至 c 柱 (destination)
  - 最後把 b 柱 (auxiliary) 的 n-1 個圓盤移動至 c 柱 (destination)

```
圓盤數量: 3
a -> c
a -> b
c -> b
a -> c
b -> a
b -> c
a -> c
搬運次數: 7
```

不是只有變數，函式也可以指派

黃彬華編撰

- 定義
  - def divide(dividend, divisor):
- 指派
  - 像變數指派一樣，將一個函式指派給其他函式
    - mathDivide = divide
- 呼叫
  - 呼叫被指派的函式就等於呼叫原來的函式
    - mathDivide(10, 2)

範例

- FunctionAssignDemo

函式也可以當做參數

- 定義
  - def showDivide(divideFunc, num1, num2):
- 呼叫
  - showDivide(myDivide, 10, 2)
    - 將 myDivide 函式指派給 divideFunc 參數

範例

- FunctionAsParamDemo

練習 5-2

黃彬華編撰

- 定義 1 個函式: calculate(numbers, myFunction)
  - numbers: 一個數列
  - myFunction: 一個函式會計算 numbers 所有元素的平均值後顯示在畫面上
- 讓使用者輸入一個數列，會將平均顯示在畫面上

```
請輸入整數數列(空白分隔): 1 2 3 4 5
數列為: 1 2 3 4 5
平均 = 3.0
```

函式也可以被回傳

- 定義
  - 可以定義函式的回傳值為另一個函式

```
def math(isAdd):
    return add if isAdd else subtract


def add(num1, num2):
    return f"{num1} + {num2} = {num1 + num2}"


def subtract(num1, num2):
    return f"{num1} - {num2} = {num1 - num2}"
```

- 呼叫

```
calculate = math(True)
result = calculate(5, 3)
```

範例

- FunctionAsReturnDemo

函式界的無名英雄 - Lambda

黃彬華編撰

- lambda 是一個沒有名字、極其簡化的函式
  - 必須是運算式，所以無需加上return
  - 定義：
    - lambda *parameters* : expression
      - parameters：參數，可以多個
      - expression：運算式
- 可以將 lambda 指派給變數
- 常用於函式型參數的傳遞

範例

- LambdaDemo

Decorator - 取用修飾函式

黃彬華編撰

- 使用 decorator 目的就是在不影響原來函式情況下，加上其他功能(稱為修飾)，兼具重複利用與模組化
- 在函式上面加上 "@" 就是加上decorator，而 "@" 所指定的名稱就是修飾用的函式名稱

範例

- DecoratorDemo