



Python程式語言

part2

劉和師

2018/12/08

Ch7.讀取檔案

▶ 關於檔案，要注意下面幾點：

- ▶ 1. 要利用磁碟內的檔案，必須先將檔案複製到記憶體，假設檔案叫做test.txt，要讀取這個檔案，使用以下指令

```
f = open("test.txt","r")
```

- ▶ 2. open是開啟檔案的指令。
- ▶ 3. f可自行命名，它僅是一個暫存區，程式結束或關機就不存在了。
- ▶ 4. 必要時須加上路徑名稱，否則會找不到檔案。
- ▶ 5. 在程式結束的地方或不再使用此檔時，需下達關檔的指令。

```
f.close()
```

Ch7.讀取檔案

▶ 檔案開啟模式參數：

參數	說 明
r	讀取模式(預設)
w	寫入模式，會先清空檔案內容
x	只在檔案不存在時才建立新檔，並開啟為寫入模式，若檔案已存在會引發FileExistsError錯誤
a	附加模式，若檔案已存在，寫入的內容會附加至檔案尾端
b	二進位模式

Ch7.讀取檔案

► 讀取檔案並印出資料

```
f = open("test.txt", "r")
```

```
for line in f:  
    print(line)
```

```
f.close()
```

以唯讀方式開啟test.txt，
暫存檔名稱爲f

For迴圈會自檔案f中一次讀
入一行放入line變數，然後
用print()印出來

程式結束要關閉檔案

Ch7.讀取檔案

- ▶ 讀取一個檔案內的數字，求其平均值。

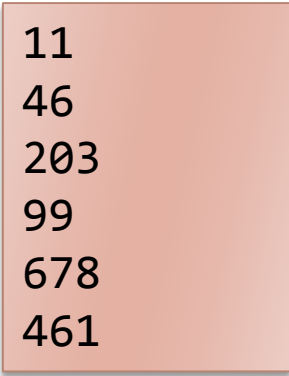
```
sum = 0
Alen = 0

f = open("Adata.txt","r")
print("Adata.txt:")

for line in f:
    sum += int(line)
    print(int(line))
    Alen += 1

print("Average: ",str(sum/Alen))
f.close()
```

Adata.txt內容



```
11
46
203
99
678
461
```

Ch7.讀取檔案

- ▶ 將兩個檔案中的數字加起來，再將結果存入第三個檔案。(為方便測試，Adata.txt與Bdata.txt內容相同)
- ▶ `f.readline()`一次讀入一整行

Adata.txt內容

```
11
46
203
99
678
461
```

+

Bdata.txt內容

```
11
46
203
99
678
461
```

=

Cdata.txt內容

```
22
92
406
198
1356
922
```

Ch7.讀取檔案

▶ 參考程式：

```
def open_and_print_file(filename):
    f = open(filename, "r")
    print(filename)
    for line in f:
        print(int(line))
    f.close()

def addfile(x,y,z):
    f1 = open(x, "r")
    f2 = open(y, "r")
    f3 = open(z, "w") # w是寫入檔案
    u = f1.readline()
    v = f2.readline()
    while u and v:
        sum = int(u) + int(v)
        f3.write(str(sum)+"\n")
        u = f1.readline()
        v = f2.readline()
    f1.close()
    f2.close()
    f3.close()

#main
open_and_print_file("Adata.txt")
open_and_print_file("Bdata.txt")
addfile("Adata.txt", "Bdata.txt", "Cdata.txt")
open_and_print_file("Cdata.txt")
```

Ch7. 習題

- 7-1. 寫一程式，從檔案中讀入 n 及 a_1, a_2, \dots, a_n 。 n 和所有的 a_i 都要是正整數，計算出 $a_1^2, a_2^2, \dots, a_n^2$ 。再將 $a_i, a_i^2, i=1$ 到 $i=n$ 寫到另一個檔案上。

ex7-1內容

10
1
2
3
4
5
6
7
8
9
10

ex7-1a內容

1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

Ch7. 習題

▶ 7-1. 參考程式：

```
f1 = open("ex7-1.txt", "r")
f2 = open("ex7-1a.txt", "w")

n = int(f1.readline())
for i in range(n):
    u = int(f1.readline())
    v = u**2
    print(str(u)+"\t"+str(v)) #顯示在螢幕
    f2.write(str(u)+"\t"+str(v)+"\n") #寫入檔案

f1.close()
f2.close()
```

Ch7. 習題

- 7-2. 寫一程式，從檔案中讀入 n 及 a_1, a_2, \dots, a_n 。 n 和所有的 a_i 都要是正整數，計算出 $b_i = \sqrt{a_i} \times 10$ ，將 $a_i, b_i, i=1$ 到 $i=n$ 寫到另一個檔案上去。

ex7-2內容

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

ex7-2a內容

```
1      10.0
2      14.142135623730951
3      17.32050807568877
4      20.0
5      22.360679774997898
6      24.49489742783178
7      26.457513110645905
8      28.284271247461902
9      30.0
10     31.622776601683796
```

Ch7. 習題

▶ 7-2. 參考程式：

```
f1 = open("ex7-2.txt", "r")
f2 = open("ex7-2a.txt", "w")

n = int(f1.readline())
for i in range(n):
    u = int(f1.readline())
    v = (u**0.5)*10
    print(str(u)+"\t"+str(v))
    f2.write(str(u)+"\t"+str(v)+"\n")

f1.close()
f2.close()
```

Ch7. 習題

- ▶ 寫一程式，從檔案中讀入一個3x5的矩陣，求此矩陣的transpose(轉置)，然後將此結果寫到另一個檔案。
- ▶ 檔案內數字以跳格符號(Tab)分開。

ex7-3內容

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

ex7-3a內容

1	6	11
2	7	12
3	8	13
4	9	14
5	10	15

Ch7. 習題

► 7-3. 參考程式：

```
f1 = open("ex7-3.txt", "r")      #開啟檔案
f2 = open("ex7-3a.txt", "w")
A = [[0 for i in range(5)] for i in range(3)]  #宣告3x5陣列
B = [[0 for i in range(3)] for i in range(5)]

def print_matrix(x,a,b):  #印出陣列副程式
    for i in range(a):
        for j in range(b):
            print(str(x[i][j])+"\t",end="")
        print("")

#read file into matrix A
i = 0
for line in f1:
    line = line.replace("\n","")  #將換行符號消去
    A[i] = line.split("\t")  #以跳格當分界，將數字存入陣列
    i += 1
print_matrix(A,3,5)
print("")  #印一個換行
```

Ch7. 習題

▶ 7-3. 參考程式：

```
#transposed matrix and write to file
for j in range(5):
    s = ""
    for i in range(3):
        B[j][i] = A[i][j]
        s += str(B[j][i])+"\t"
    f2.write(s+"\n")

print_matrix(B,5,3)    #顯示置換後結果

f1.close()    #關閉檔案
f2.close()
```

Ch7. 習題

- ▶ 7-4. 寫一程式，從一檔案中，讀入一組一元二次方程式 $ax^2+bx+c=0$ 的係數 a ， b ， c ，其中 $(b^2-4ac) \geq 0$ 。解此組方程式，並將中間過程及最後答案寫入另一檔案。
- ▶ (本題略過)

Ch7. 習題

- 7-5. 寫一程式，從檔案中讀入 n 及 a_1, a_2, \dots, a_n 。 n 和所有的 a_i 都要是正整數，將 a_n, a_{n-1}, \dots, a_1 寫到另一個檔案上去。

ex7-5內容

10
1
2
3
4
5
6
7
8
9
10

ex7-5a內容

10
9
8
7
6
5
4
3
2
1

Ch7. 習題

▶ 7-5. 參考程式：

```
f1 = open("ex7-5.txt", "r")      #開啟檔案
f2 = open("ex7-5a.txt", "w")

A = [0 for i in range(n)]        #宣告陣列
n = int(f1.readline())

for i in range(n):               #讀入檔案
    A[i] = int(f1.readline())
    print(A[i])

for i in range(n-1, -1, -1):     #倒過來寫入檔案
    f2.write(str(A[i])+"\n")
    print(A[i])

f1.close()                      #關閉檔案
f2.close()
```

休息一下~



Ch8. 配置

- ▶ 配置(configuration)有點類似C語言的結構(structure)。
- ▶ 一個配置會有一個[名稱]，和一些資料欄位，我們需要import configparser來使用。例如一個學生的課業資料：

```
[劉和師]  
mathscore = 80  
chinesescore = 99  
engscore = 97
```

- ▶ 這個配置名稱就叫studentdata好了，我們把它放在檔案內，檔名叫student_score.txt，經由配置，我們可以很方便的來分析、搜尋資料。

Ch8. 配置

- ▶ 相關指令：

- ▶ 1. 啟用配置

```
import configparser
```

- ▶ 2. 建立一個叫studentdata的配置

```
studentdata = configparser.ConfigPaeser()
```

- ▶ 3. 配置內有許多section，例如以下就是一個section：

```
[Wayne]  
mathscore = 80  
chinesescore = 99  
engscore = 97
```

- ▶ 每一個section有一個標題，這裡叫name好了



Ch8. 配置

- ▶ 新增一個section：

```
studentdata.add_section("Billy")  
    或使用變數  
studentdata.add_section(name)
```

- ▶ 在section內加入資料("mathscore"是欄位名稱)：

```
studentdata.set("Billy", "mathscore", "99")  
    或使用變數  
studentdata.set(name, "mathscore", score)
```

- ▶ 查詢某section是否已存在：

```
studentdata.has_section(name)
```

Ch8.配置

- ▶ 將配置寫入檔案f：

```
studentdata.write(f)
```

- ▶ 將已經配置好的檔案資料讀入：

```
studentdata.read(filename)  
或  
config.read(filename)
```

- ▶ 取得所有section的內容成為一個陣列：

```
SectionArray = studentdata.sections()
```

- ▶ 假如要取得某學生的數學成績：

```
studentdata.get(name, "mathscore")
```



Ch8. 配置

- ▶ 在磁碟內建立一個配置，輸入並儲存班級學生各科成績，內容如下。

[學生姓名]

(數學)欄位名稱 = 成績

(國文)欄位名稱 = 成績

(英文)欄位名稱 = 成績

```
[A]
mathscore = 77
chinesescore = 90
englishscore = 90
```

```
[B]
mathscore = 67
chinesescore = 76
englishscore = 57
```

```
[C]
mathscore = 78
chinesescore = 90
englishscore = 78
```

student_score.txt內容

Ch8. 配置

▶ 參考程式：

```
import configparser  #要先引入configparser

#建立一個配置，名稱為studentdata
def create_studentdata():
    studentdata = configparser.ConfigParser()
    return studentdata

#定義並輸入資料到一個配置
def define_and_add_student(studentdata,name,mathscore,
chinesescore,englishscore):
    studentdata.add_section(name)
    studentdata.set(name,"mathscore",mathscore)
    studentdata.set(name,"chinesescore",chinesescore)
    studentdata.set(name,"englishscore",englishscore)
```


Ch8. 配置

▶ 參考程式：

```
def input_studentdata(studentdata):    #輸入資料
    numOfStudent = int(input("請輸入學生人數:"))
    for i in range(numOfStudent):
        name = input("請輸入學生姓名:")
        if studentdata.has_section(name):
            print(name+"已經存在了!")
        else:
            mathscore = input("請輸入"+name+"的數學成績:")
            chinesescore = input("請輸入"+name+"的國文成績:")
            englishscore = input("請輸入"+name+"的英文成績:")
            define_and_add_student(studentdata,name,mathscore,
                                    chinesescore,englishscore)
            f = open("student_score.txt","w")
            studentdata.write(f)
            f.close()

#main
studentdata = create_studentdata()
input_studentdata(studentdata)
```

Ch8. 配置

- ▶ 求班級各科成績平均。
- ▶ 說明：
 - ▶ 因student_score.txt已經是一個以配置型態存檔的純文字檔，所以用 `config.read(filename)` 指令來開啟即可使用。
 - ▶ 我們將所有section讀入到陣列，然後再用迴圈來處理就很方便：`SectionArray = studentdata.sections()`
 - ▶ 例如讀出第*i*個學生的數學成績：

```
math = int(studentdata.get(SectionArray[i], "mathscore"))
```

Ch8. 配置

▶ 參考程式：

```
#引入configparser
import configparser

#建立配置
def create_studentdata():
    studentdata = configparser.ConfigParser()
    return studentdata

#讀入檔案到配置中
def load_config(config,filename):
    config.read(filename)
```

Ch8. 配置

▶ 參考程式：

```
def score_average(filename): #求各科成績平均
    studentdata = create_studentdata()
    load_config(studentdata,filename)
    SectionArray = studentdata.sections()
    numOfStudent = len(SectionArray)
    math_sum = 0
    chi_sum = 0
    eng_sum = 0
    for i in range(numOfStudent):
        math = int(studentdata.get(SectionArray[i],"mathscore"))
        math_sum += math
        chi = int(studentdata.get(SectionArray[i],"chinesescore"))
        chi_sum += chi
        eng = int(studentdata.get(SectionArray[i],"englishscore"))
        eng_sum += eng
    print("數學平均:"+str(math_sum/numOfStudent))
    print("國文平均:"+str(chi_sum/numOfStudent))
    print("英文平均:"+str(eng_sum/numOfStudent))
#main
score_average("student_score.txt")
```

Ch8. 配置

- ▶ 新增一位學生資料。

```
import configparser

def create_studentdata():
    studentdata = configparser.ConfigParser()
    return studentdata

def load_config(config,filename): #讀入已存在的資料
    config.read(filename)

def define_and_add_student(studentdata,name,mathscore,
                           chinesescore,englishscore):
    studentdata.add_section(name)
    studentdata.set(name,"mathscore",mathscore)
    studentdata.set(name,"chinesescore",chinesescore)
    studentdata.set(name,"englishscore",englishscore)
```

Ch8. 配置

- ▶ 新增一位學生資料。

```
#main
name = input("請輸入欲新增的學生姓名:")
studentdata = create_studentdata()
load_config(studentdata, "student_score.txt")
if studentdata.has_section(name):
    print("這名學生已存在")
else:
    mathscore = input("請輸入"+name+"的數學成績:")
    chinesescore = input("請輸入"+name+"的國文成績:")
    englishscore = input("請輸入"+name+"的英文成績:")
    define_and_add_student(studentdata, name, mathscore,
                           chinesescore, englishscore)

    f = open("student_score.txt", "w")
    studentdata.write(f)
    f.close()
```

Ch8. 配置

► 修改一位學生的資料。

```
import configparser
def modify_student(studentdata,name):
    if studentdata.has_section(name):
        select = int(input("請輸入"+name+"要修改成績的科目(1.數學 2.國文 3.英文):"))
        if select == 1:
            mathscore = input("請輸入"+name+"新的數學成績:")
            old = studentdata.get(name,"mathscore")
            studentdata.set(name,"mathscore",mathscore)
            print(name+"的數學成績已由"+old+"分改為"+mathscore+"分")
        if select == 2:
            chinesescore = input("請輸入"+name+"新的國文成績:")
            old = studentdata.get(name,"chinesescore")
            studentdata.set(name,"chinesescore",chinesescore)
            print(name+"的國文成績已由"+old+"分改為"+chinesescore+"分")
        if select == 3:
            englishscore = input("請輸入"+name+"新的英文成績:")
            old = studentdata.get(name,"englishscore")
            studentdata.set(name,"englishscore",englishscore)
            print(name+"的英文成績已由"+old+"分改為"+englishscore+"分")
    else:
        print("找不到這名學生")
```

Ch8. 配置

- ▶ 修改一位學生的資料。

```
def create_studentdata():  
    studentdata = configparser.ConfigParser()  
    return studentdata  
  
def load_config(config,filename):  
    config.read(filename)  
  
#main  
studentdata = create_studentdata()  
load_config(studentdata,"student_score.txt")  
name = input("請輸入欲修改的學生姓名:")  
modify_student(studentdata,name)  
  
f = open("student_score.txt","w")  
studentdata.write(f)  
f.close()
```


Ch8. 配置

- ▶ 查詢一位學生的資料。

```
import configparser

def create_studentdata():
    studentdata = configparser.ConfigParser()
    return studentdata

def load_config(config,filename):
    config.read(filename)

def search_student(studentdata,name):
    if studentdata.has_section(name):
        math = studentdata.get(name,"mathscore")
        chi = studentdata.get(name,"chinesescore")
        eng = studentdata.get(name,"englishscore")
        print("數學:"+math+" 國文:"+chi+" 英文:"+eng)
    else:
        print("找不到這名學生")

#main
studentdata = create_studentdata()
load_config(studentdata,"student_score.txt")
name = input("請輸入欲查詢學生姓名:")
search_student(studentdata,name)
```

Ch8. 配置

- ▶ 刪除一位學生的資料。

```
import configparser

def create_studentdata():
    studentdata = configparser.ConfigParser()
    return studentdata

def load_config(config,filename):
    config.read(filename)

def remove_student(studentdata,name):
    if studentdata.has_section(name):
        studentdata.remove_section(name)
        print("刪除成功")
    else:
        print("沒有這名學生")

#main
studentdata = create_studentdata()
load_config(studentdata,"student_score.txt")
name = input("請輸入欲刪除的學生姓名:")
remove_student(studentdata,name)
f = open("student_score.txt","w")
studentdata.write(f)
f.close()
```

Ch8.配置

- ▶ 簡易借還書程式。
- ▶ 這個程式有以下功能：
 - ▶ 1.新增書籍
 - ▶ 2.借書
 - ▶ 3.還書
 - ▶ 4.結束程式
- ▶ 配置的規格如下：
 - ▶ 1.[書名]
 - ▶ 2.出版社
 - ▶ 3.作者
 - ▶ 4.頁數
 - ▶ 5.借閱人(為none時表示尚無人借閱)



Ch8.配置

- ▶ 程式執行後，檔案booklist.txt的內容參考。

[迷霧之子I]

出版社 = 皇冠出版社

作者 = 布蘭登山德勒

頁數 = 684

借閱人 = 劉和師

[哈利波特I]

出版社 = CROWN

作者 = JK羅琳

頁數 = 1532

借閱人 = none

[西遊記]

出版社 = 三民書局

作者 = 吳承恩

頁數 = 1515

借閱人 = none

Ch8. 配置

▶ 參考程式： (新增書籍)

```
import configparser

def addBook(): #新增一本書籍
    books = configparser.ConfigParser()
    books.read("booklist.txt")
    title = input("請輸入欲新增書籍的書名:")
    if books.has_section(title):
        print("這本書已經存在了")
    else:
        books.add_section(title)
        com = input("請輸入"+title+"的出版社:")
        books.set(title,"出版社",com)
        writer = input("請輸入"+title+"的作者:")
        books.set(title,"作者",writer)
        pages = input("請輸入"+title+"的頁數")
        books.set(title,"頁數",pages)
        #新增書本，尚無人借閱，故填入none
        books.set(title,"借閱人","none")
        f = open("booklist.txt","w")
        books.write(f)
        f.close()
```

Ch8. 配置

▶ 參考程式： (借書)

```
def borrowBook():
    books = configparser.ConfigParser()
    books.read("booklist.txt")
    title = input("請輸入欲借閱的書名:")

    if books.has_section(title):
        borrow = books.get(title, "借閱人")
        if borrow != "none":
            print("這本書已經被借走了")
        else:
            borrow = input("請輸入借閱人姓名:")
            books.set(title, "借閱人", borrow)
            print("借書成功")
            f = open("booklist.txt", "w")
            books.write(f)
            f.close()
    else:
        print("沒有這本書!")
```

Ch8.配置

▶ 參考程式：

▶ (還書)

```
def returnBook():
    books = configparser.ConfigParser()
    books.read("booklist.txt")
    bookName = input("請輸入歸還的書名:")
    if books.has_section(bookName):
        borrow = input("請輸入借閱人姓名:")
        if borrow == books.get(bookName, "借閱人"):
            books.set(bookName, "借閱人", "none")
            print("還書成功")
            f = open("booklist.txt", "w")
            books.write(f)
            f.close()
        else:
            print(bookName+"不是被"+borrow+"所借走的!")
    else:
        print("沒有這本書!")
```

Ch8.配置

▶ 參考程式： (主程式)

```
#main
while True:
    print()
    print("----請選擇----")
    print("1. 新增書籍")
    print("2. 借書")
    print("3. 還書")
    print("4. 結束程式")
    print("-----")
    sel = int(input("請選擇:"))

    if sel == 1:
        addBook()
    elif sel == 2:
        borrowBook()
    elif sel == 3:
        returnBook()
    elif sel == 4:
        break
    else:
        print("錯誤，請輸入 1 ~ 4")
```


Ch8. 習題

- ▶ 我們要替一家醫院處理一些病患資料如下：
 - ▶ 1. 健保號碼(也是section)
 - ▶ 2. 姓名
 - ▶ 3. 性別
 - ▶ 4. 電話
 - ▶ 5. 醫療科別
- ▶ 寫一程式，在磁碟內建立一個配置，輸入並儲存病人資料，要有以下功能：
 - ▶ 1. 新增一位病人資料
 - ▶ 2. 修改一位病人資料
 - ▶ 3. 查詢一位病人資料
 - ▶ 4. 刪除一位病人資料
 - ▶ 5. 列印所有女性病人姓名
 - ▶ 6. 結束程式



Ch8. 習題

- ▶ 程式執行後，檔案patientlist.txt的內容參考。

[1001]

姓名 = 張三

性別 = 男

電話 = 03-8312225

醫療科別 = 內科

[1002]

姓名 = 李四

性別 = 男

電話 = 03-8312227

醫療科別 = 外科

[1003]

姓名 = 隋棠

性別 = 女

電話 = 02-32569874

醫療科別 = 婦科

[1004]

姓名 = 林志玲

性別 = 女

電話 = 02-98745632

醫療科別 = 家醫科

Ch8. 習題

▶ 參考程式： (新增)

```
import configparser

#新增一位病人資料
def addPatient():
    patient = configparser.ConfigParser()
    patient.read("patientlist.txt")
    PID = input("請輸入新病患的健保號碼:")

    if patient.has_section(PID):
        print("本病患資料已存在了")
    else:
        patient.add_section(PID)
        name = input("請輸入"+PID+"的姓名:")
        patient.set(PID, "姓名", name)
        sex = input("請輸入"+PID+"的性別:")
        patient.set(PID, "性別", sex)
        tel = input("請輸入"+PID+"的電話:")
        patient.set(PID, "電話", tel)
        division = input("請輸入"+PID+"的醫療科別:")
        patient.set(PID, "醫療科別", division)
        f = open("patientlist.txt", "w")
        patient.write(f)
        f.close()
        print("新增成功")
```

Ch8. 習題

▶ 參考程式： (修改)

#修改一位病人資料

```
def editPatient():
    patient = configparser.ConfigParser()
    patient.read("patientlist.txt")
    PID = input("請輸入欲修改的病患的健保號碼:")
    if patient.has_section(PID):
        print("請輸入新資料，空白則不修改")
        name = patient.get(PID, "姓名")
        name = input("原姓名:"+name+", 新姓名:")
        if name != "":
            patient.set(PID, "姓名", name)
        sex = patient.get(PID, "性別")
        sex = input("原性別:"+sex+", 新性別:")
        if sex != "":
            patient.set(PID, "性別", sex)
        tel = patient.get(PID, "電話")
        tel = input("原電話:"+tel+", 新電話:")
        if tel != "":
            patient.set(PID, "電話", tel)
        division = patient.get(PID, "醫療科別")
        division = input("原醫療科別:"+division+", 新醫療科別:")
        if division != "":
            patient.set(PID, "醫療科別", division)
        f = open("patientlist.txt", "w")
        patient.write(f)
        f.close()
        print("資料修改完畢")
    else:
        print("查無本病患資料")
```

Ch8. 習題

▶ 參考程式： (查詢)

```
#查詢一位病人資料
def searchPatient():
    patient = configparser.ConfigParser()
    patient.read("patientlist.txt")
    PID = input("請輸入欲查詢病患的健保號碼:")

    if patient.has_section(PID):
        name = patient.get(PID, "姓名")
        print("姓名: "+name)
        sex = patient.get(PID, "性別")
        print("性別: "+sex)
        tel = patient.get(PID, "電話")
        print("電話: "+tel)
        division = patient.get(PID, "醫療科別")
        print("醫療科別: "+division)
    else:
        print("查無本病患資料")
```

Ch8. 習題

▶ 參考程式： (刪除)

```
#刪除一位病人資料
def deletePatient():
    patient = configparser.ConfigParser()
    patient.read("patientlist.txt")
    PID = input("請輸入欲刪除病患的健保號碼:")

    if patient.has_section(PID):
        patient.remove_section(PID)
        f = open("patientlist.txt", "w")
        patient.write(f)
        f.close()
        print("本筆資料已刪除")
    else:
        print("查無本病患資料")
```

Ch8. 習題

- ▶ 參考程式：
(列印)

```
#列印所有女性病患姓名
def printFemalePatient():
    patient = configparser.ConfigParser()
    patient.read("patientlist.txt")
    IDs = patient.sections()
    numOfID = len(IDs)

    for i in range(numOfID):
        if patient.get(IDs[i], "性別") == "女":
            name = patient.get(IDs[i], "姓名")
            print("健保號碼: "+IDs[i]+" 姓名: "+name)
```

Ch8. 習題

▶ 參考程式： (主程式)

```
while True:  #main
    print()
    print("-----請選擇-----")
    print("1. 新增一位病人資料")
    print("2. 修改一位病人資料")
    print("3. 查詢一位病人資料")
    print("4. 刪除一位病人資料")
    print("5. 列印所有女性病人姓名")
    print("6. 結束程式")
    print("-----")
    sel = int(input("請選擇:"))
    if sel == 1:
        addPatient()
    elif sel == 2:
        editPatient()
    elif sel == 3:
        searchPatient()
    elif sel == 4:
        deletePatient()
    elif sel == 5:
        printFemalePatient()
    elif sel == 6:
        break
    else:
        print("錯誤，請輸入1 ~ 6")
```


休息一下~



Ch9.遞迴(recursive)

- ▶ 副程式可以被呼叫，當副程式裡又呼叫自己時，這種情況稱為遞迴。
- ▶ 有些問題用遞迴可以得到更簡潔的程式，但不一定會有高效率。
- ▶ 使用遞迴時要注意：
 - ▶ 1.傳入的參數：要能符合要解決的問題。
 - ▶ 2.終止條件：錯誤或沒有終止條件，將造成無盡迴圈甚至當機
- ▶ 經典題型：最大公因數 (GCD)、費波納西數列 (Fibonacci Sequence)、河內塔 (Hanoi Tower)、N 個字元的排列組合等。



Ch9.遞迴(recursive)

- ▶ 求 $1 + 2 + 3 + \dots + N$ 的和。

```
def sum(N):  
    if N == 1:  
        value = 1  
    else:  
        value = N + sum(N-1)  
    return value  
  
#main  
x = int(input("計算1到N的總和，請輸入N:"))  
y = sum(x)  
print("1到"+str(x)+"的總和為"+str(y))
```

終止條件

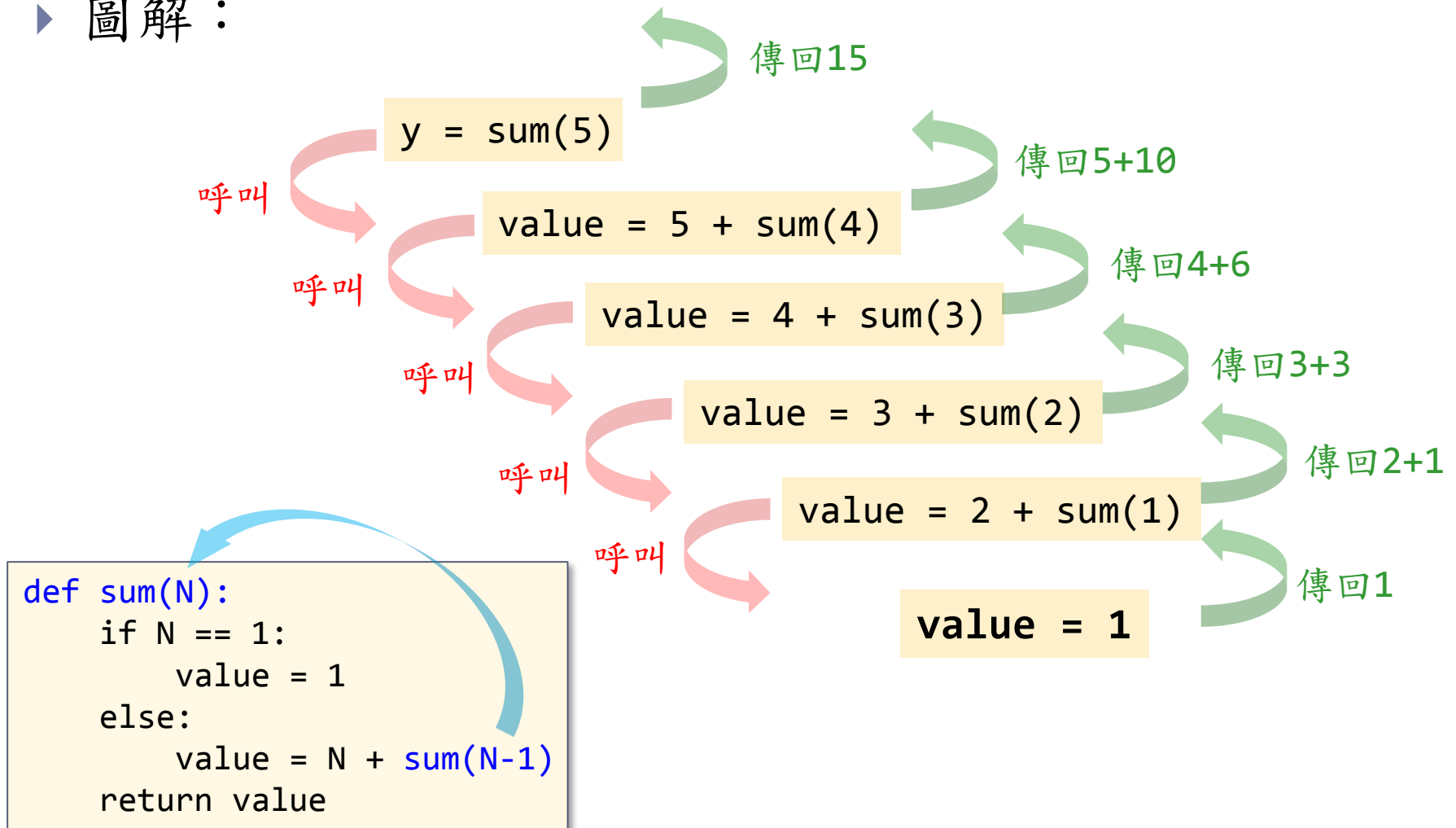
結束呼叫，回傳確定值

傳遞參數，呼叫自己

- ▶ 假設輸入5

Ch9.遞迴(recursive)

► 圖解：



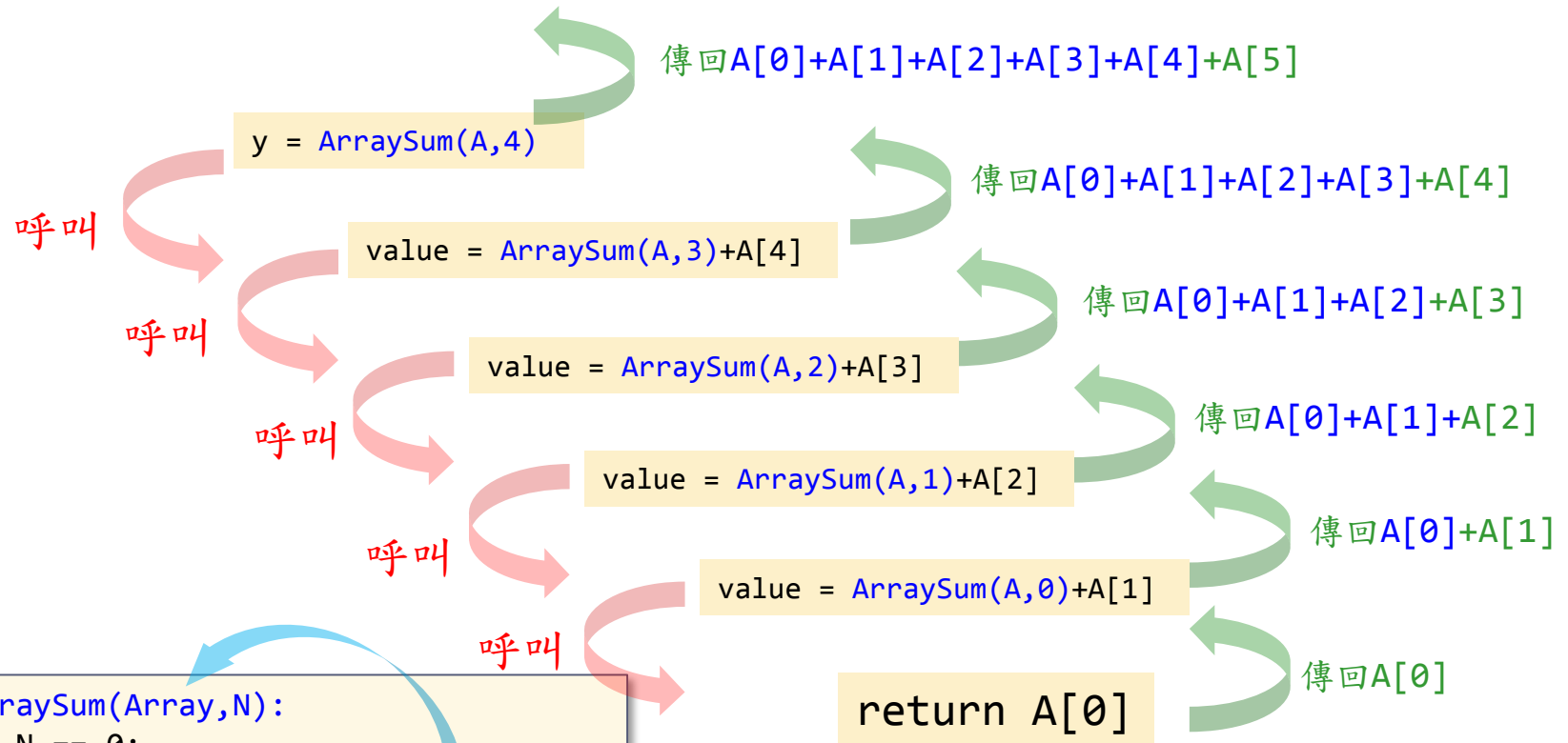
Ch9.遞迴(recursive)

- ▶ 求陣列A中所有數值的和。

```
def ArraySum(Array,N):  
    if N == 0:  
        return A[0]  
    else:  
        value = ArraySum(Array,N-1)+A[N]  
        return value  
  
#main  
A = []  
s = int(input("請輸入陣列大小:"))  
for i in range(s):  
    x = int(input("請輸入A["+str(i)+"]的值:"))  
    A.append(x)  
  
y = ArraySum(A,s-1)  
print("Array內所有數值的總和為"+str(y))
```

Ch9. 遞迴(recursive)

► 圖解：(假設N=5)



```
def ArraySum(Array, N):  
    if N == 0:  
        return A[0]  
    else:  
        value = ArraySum(Array, N-1) + A[N]  
        return value
```

Ch9.遞迴(recursive)

▶ 計算N!。

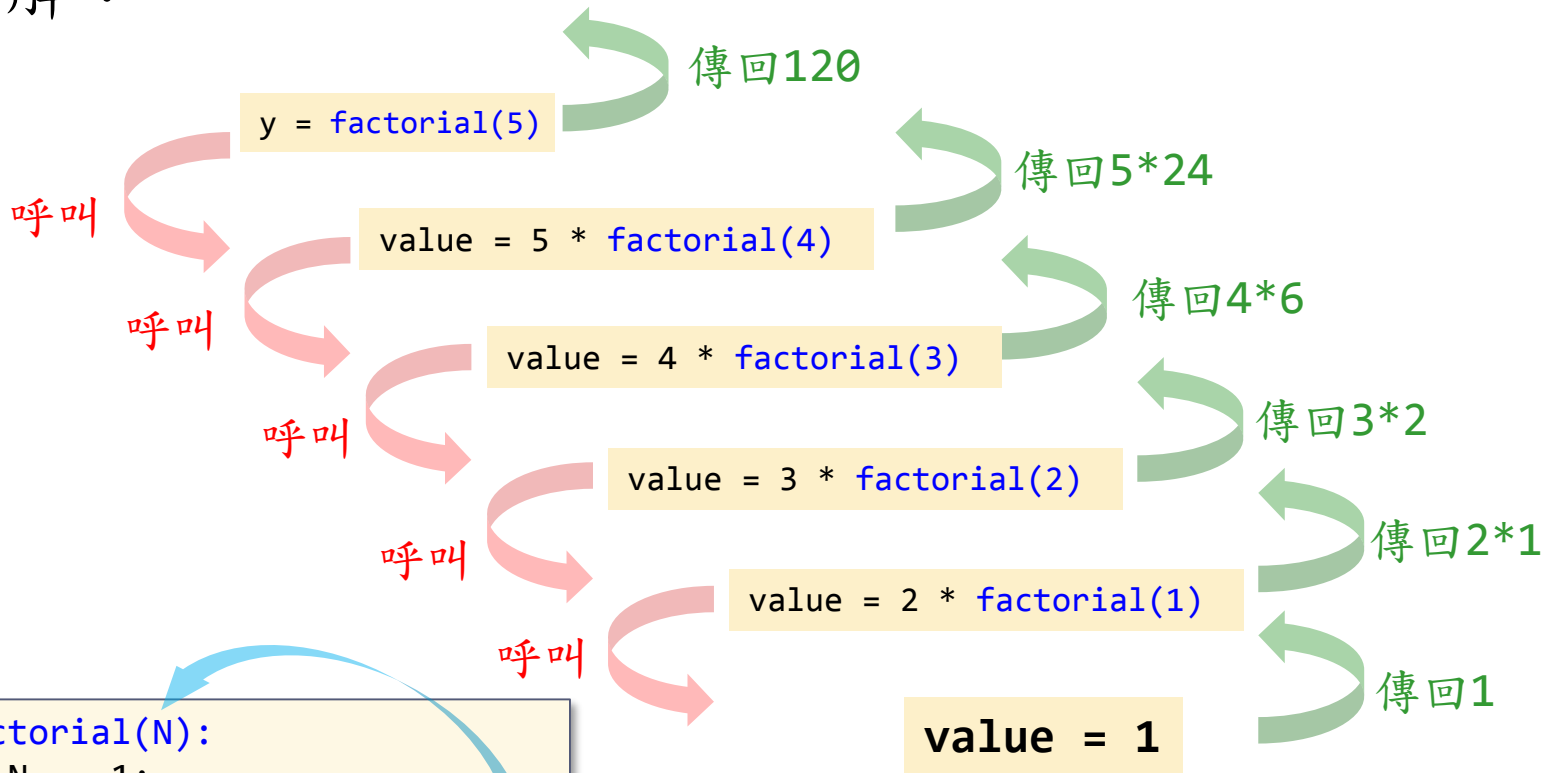
```
def factorial(N):  
    if N == 1:  
        value = 1  
    else:  
        value = N * factorial(N-1)  
    return value  
  
#main  
x = int(input("要計算幾階層"))  
y = factorial(x)  
print(str(x)+"! = "+str(y))
```

▶ 假設輸入5



Ch9.遞迴(recursive)

► 圖解：



```
def factorial(N):  
    if N == 1:  
        value = 1  
    else:  
        value = N * factorial(N-1)  
    return value
```


Ch9.遞迴(recursive)

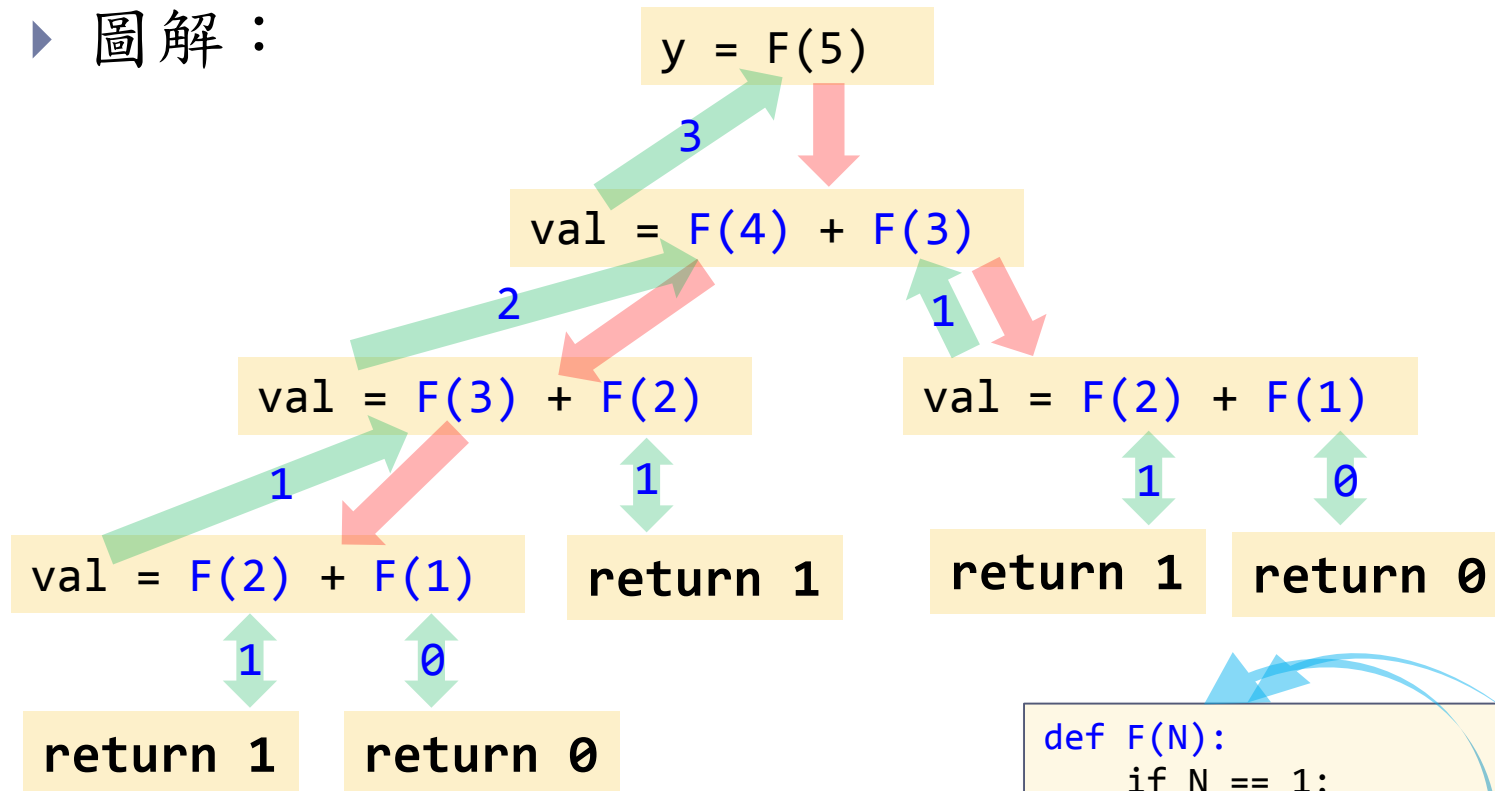
- ▶ 求費氏數列 $F(n)$ 。

```
def F(N):  
    if N == 1: 終止條件1  
        return 0  
    elif N == 2: 終止條件2  
        return 1  
    else:  
        val = F(N-1) + F(N-2)  
        return val  
  
#main  
x = int(input("請輸入求F(N)的N為:"))  
y = F(x)  
print("F("+str(x)+") = "+str(y))
```

- ▶ 假設輸入5

Ch9. 遞迴(recursive)

► 圖解：



► 最後 $y = 3$

```
def F(N):  
    if N == 1:  
        return 0  
    elif N == 2:  
        return 1  
    else:  
        val = F(N-1) + F(N-2)  
        return val
```

Ch9. 習題

- ▶ 9-1. 以遞迴方式求 $1+b+b^2+\dots+b^n$ 。
- ▶ 9-2. 以遞迴方式求 $1\times 2+2\times 3+\dots+n(n+1)$ 。
- ▶ 9-3. 以遞迴方式求 $a+(a+b)+(a+2b)+\dots+(a+(n-1)b)$ 。
(注意b的係數在增加)



Ch9. 習題

- 9-1. 以遞迴方式求 $1+b+b^2+\dots+b^n$ 。

```
def nF(b,n):  
    if n == 0:  
        return 1  
    else:  
        return b**n + nF(b,n-1)  
  
#main  
b = int(input("請輸入基底b:"))  
n = int(input("請輸入次方n:"))  
y = nF(b,n)  
print("1+b+b**2+.....+b**n= "+str(y))
```

Ch9. 習題

- 9-2. 以遞迴方式求 $1 \times 2 + 2 \times 3 + \dots + n(n+1)$ 。

```
def f(n):  
    if n == 2:  
        return 2  
    else:  
        return n*(n-1) + f(n-1)  
  
#main  
n = int(input("請輸入N:"))  
x = f(n+1)  
print("(1*2)+(2*3)+...+n*(n+1)= "+str(x))
```

Ch9. 習題

- 9-3. 以遞迴方式求 $a + (a+b) + (a+2b) + \dots + (a+(n-1)b)$ 。
(注意 b 的係數在增加)

```
def f(a,b,n):  
    if n == 0:  
        return a  
    else:  
        return a+n*b + f(a,b,n-1)  
  
#main  
a = int(input("請輸入a:"))  
b = int(input("請輸入b:"))  
n = int(input("請輸入n:"))  
x = f(a,b,n-1)  
print("結果為:"+str(x))
```

休息一下~



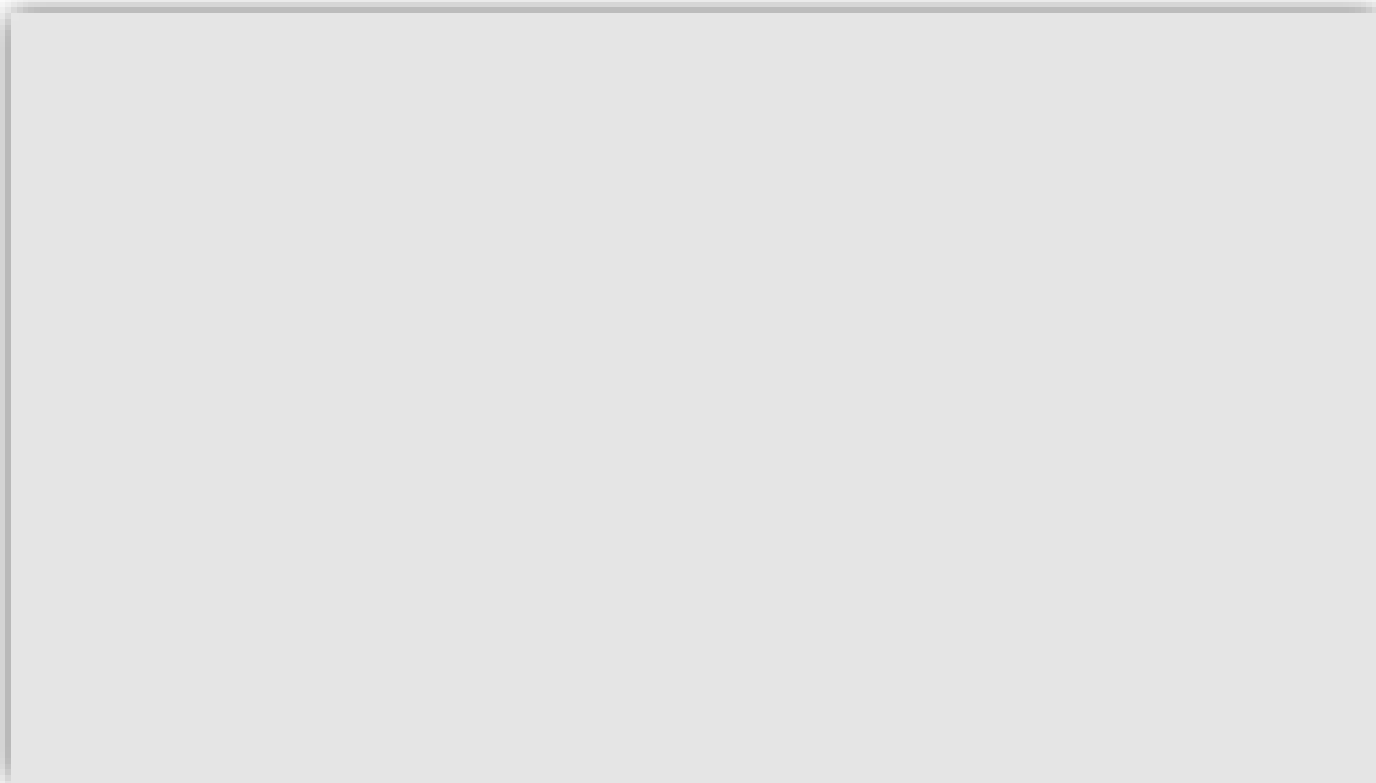
Ch.10 排序

- ▶ 排序(Sort)是將資料依某種規則重新安排其先後順序。
- ▶ 最常見的是依大小或字母排序。
- ▶ 常見排序方法：
 - ▶ 1.氣泡排序法(Bubble Sort)
 - ▶ 2.選擇排序法(Selection Sort)
 - ▶ 3.插入排序法(Insertion Sort)
 - ▶ 4.快速排序法(Quick Sort)
 - ▶ 5.合併排序法(Merge Sort)
- ▶ 不同的排序方法效率不同，應用的地方也不同。



Ch10. 氣泡排序法(Bubble Sort)

- ▶ 是一種簡單的排序演算法。它重複地走訪過要排序的數列，每次比較相鄰的兩個元素，如果他們的順序錯誤就把他們交換過來。



Ch10. 氣泡排序法(Bubble Sort)

▶ 參考程式：

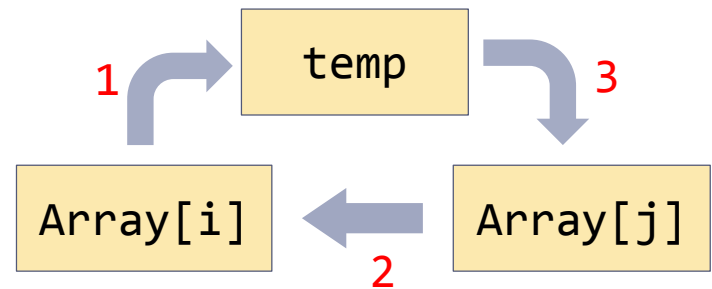
```
def bubble(Array,i,N):  #氣泡排序一個巡迴
    k = N-1
    while k>i:
        if Array[k-1]>Array[k]:
            Array[k-1],Array[k] = Array[k], Array[k-1]
        k = k-1

#main
A = []
N = int(input("請輸入陣列大小:"))
#輸入數字並存入陣列
for i in range(N):
    x = int(input("請輸入第"+str(i)+"個數字:"))
    A.append(x)
#呼叫排序副程式
for i in range(N):
    print(A)  #每一巡迴就印出陣列現況
    bubble(A,i,N)
print(A)
```

Ch10. 氣泡排序法(Bubble Sort)

- ▶ 兩變數互換：
- ▶ 1. 其他語言需要藉助另一個變數當中介來交換兩變數。

```
def SWAP(Array,i,j):  
    temp = Array[i]  
    Array[i] = Array[j]  
    Array[j] = temp
```



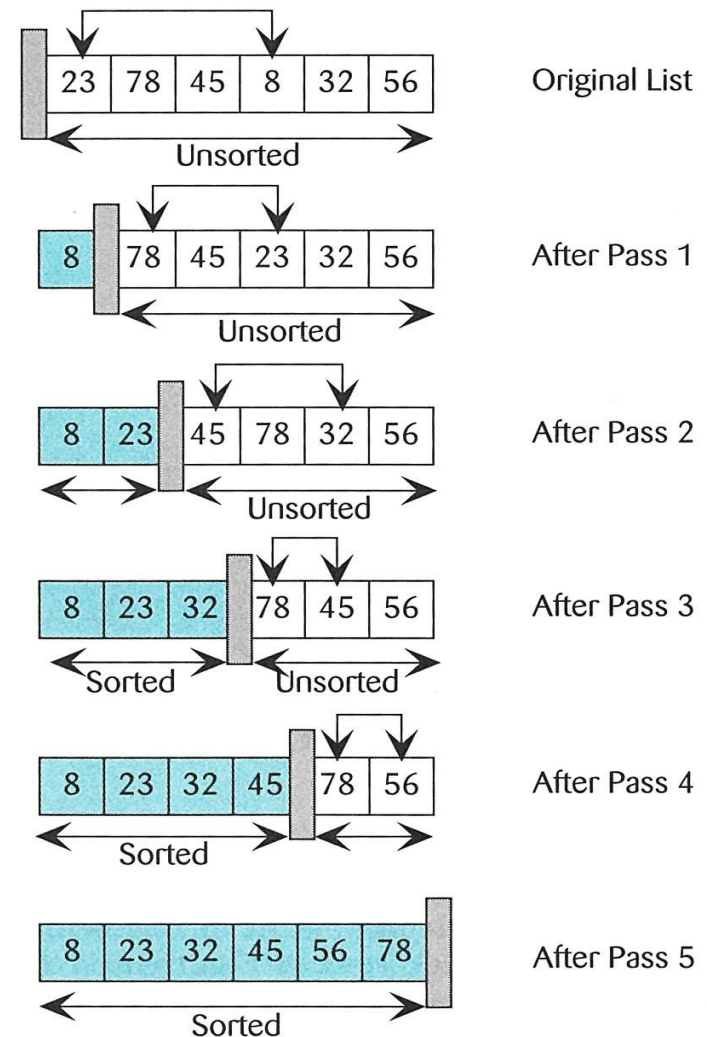
- ▶ 2. 在Python只要一行就可以了。

```
Array[i] , Array[j] = Array[j] , Array[i]
```

即 $a, b = b, a$

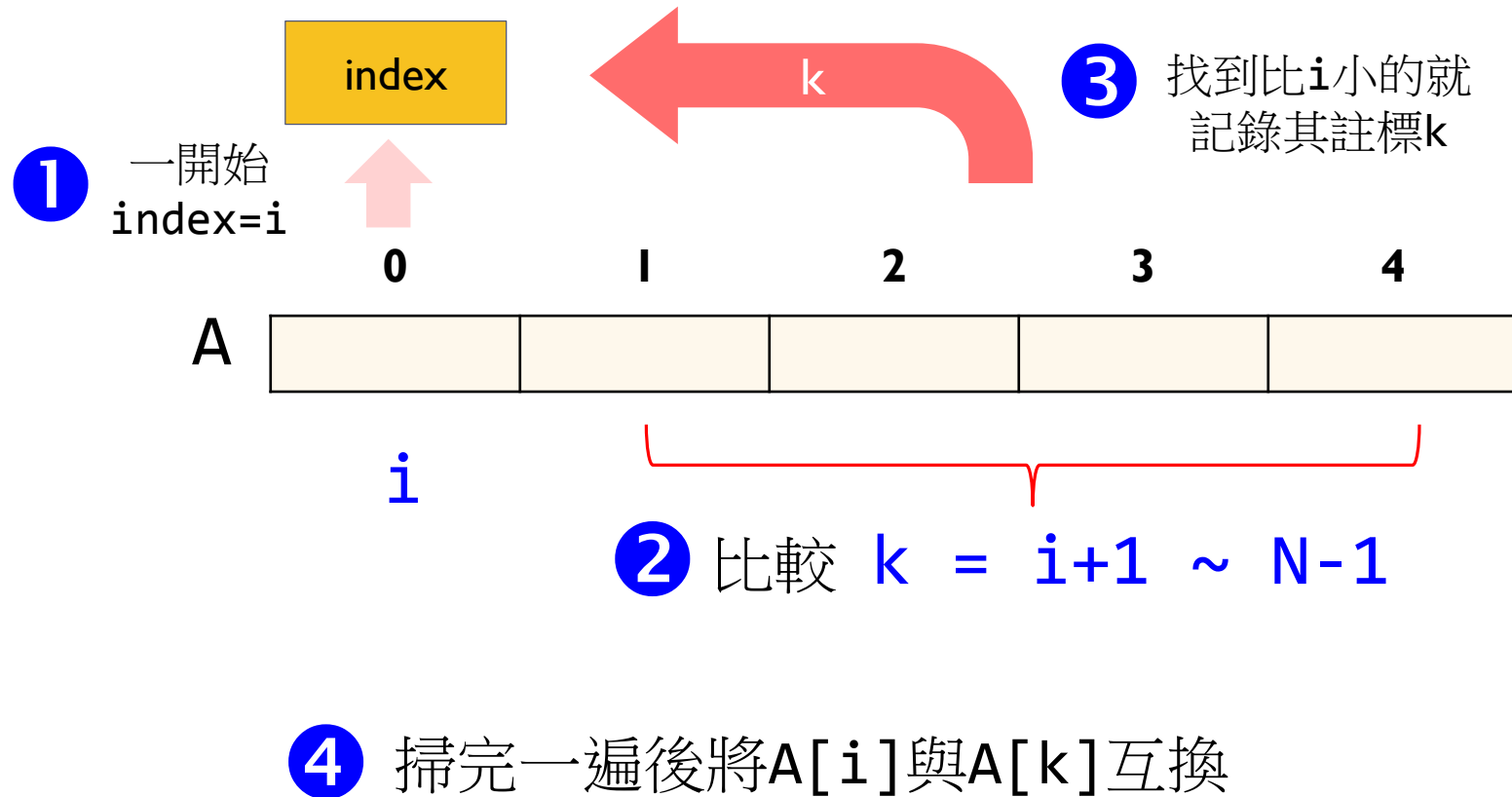
Ch10.選擇排序法(Selection Sort)

- ▶ 先假設第一個元素是最小的，依序向後掃描個元素，若有更小的就記錄起來，最後和第一個元素互換位置，完成一個巡迴，然後第二個元素，依此類推。



Ch10.選擇排序法(Selection Sort)

► 圖示說明：



Ch10.選擇排序法(Selection Sort)

▶ 參考程式：

```
def selection(Array,i,N):
    index = i    #現在位置
    k = i+1      #尋找範圍：現在位置之後到陣列結尾
    while k < N:  #尋找比現在元素更小的值，並記錄位置
        if Array[k] < Array[index]:
            index = k
        k += 1
    if index != i: #如果有記錄到更小的值就交換，否則略過
        Array[i],Array[index] = Array[index],Array[i]

#main
A = []
N = int(input("請輸入陣列大小:"))
for i in range(N):
    x = int(input("請輸入第"+str(i)+"個數字:"))
    A.append(x)
#依每個元素位置呼叫選擇排序副程式
for i in range(N):
    print(A)
    selection(A,i,N)
print(A)
```

Ch10.選擇排序法(Selection Sort)

▶ 執行結果：

請輸入陣列大小:6
請輸入第0個數字:9
請輸入第1個數字:4
請輸入第2個數字:7
請輸入第3個數字:2
請輸入第4個數字:8
請輸入第5個數字:3
[9, 4, 7, 2, 8, 3]

Pass 1 [2, 4, 7, 9, 8, 3]



Pass 2 [2, 3, 7, 9, 8, 4]



Pass 3 [2, 3, 4, 9, 8, 7]



Pass 4 [2, 3, 4, 7, 8, 9]



Pass 5 [2, 3, 4, 7, 8, 9]



最後一次無須交換

result [2, 3, 4, 7, 8, 9]

Ch10.插入排序法(Insertion Sort)

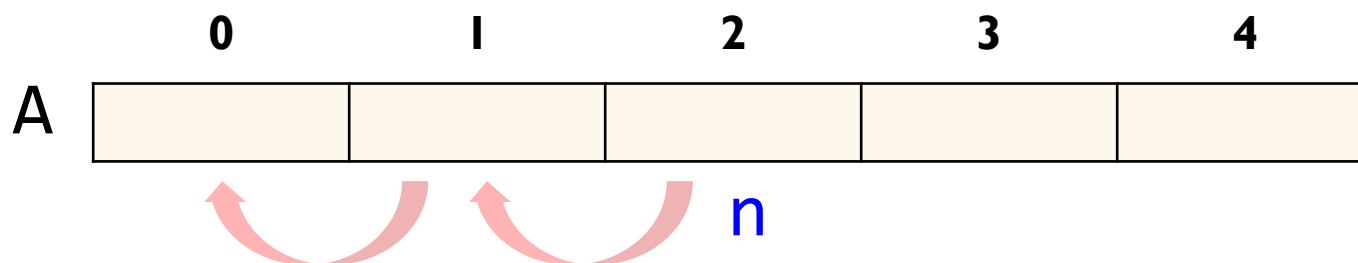
- ▶ 假設由小到大排序，每個元素和其左邊比較，若比左側小則交換，直到比左側大為止。

Insertion Sort							
35	97	19	4	57	27	98	36
0	1	2	3	4	5	6	7



Ch10.插入排序法(Insertion Sort)

▶ 交換原則：



比較 $A[n]$ 和 $A[n-1]$

- ❶ if $A[n] < A[n-1] \rightarrow$ 交換
- ❷ 直到 $A[1]$ 或 $A[n] \geq A[n-1] \rightarrow$ 停止
- ❸ 下一個，重複直到陣列最後一個元素

Ch10.插入排序法(Insertion Sort)

▶ 參考程式：

```
def insert(Array,i):
    k = i
    while k >= 1: #從第i個位置開始向前依序比較
        if Array[k] < Array[k-1]: #比左側小則交換兩元素
            Array[k-1],Array[k] = Array[k],Array[k-1]
        else: #否則結束本pass
            break
        k = k - 1

#main
A = []
N = int(input("請輸入陣列大小:"))
for i in range(N):
    x = int(input("請輸入第"+str(i)+"個數字:"))
    A.append(x)
k = 1
while k < N: #從第1個元素開始依序呼叫排序副程式
    print(A,"\n")
    insert(A,k)
    k += 1
print(A) #印出最後結果
```

Ch10.插入排序法(Insertion Sort)

▶ 執行結果：

請輸入陣列大小:6
請輸入第0個數字:9
請輸入第1個數字:4
請輸入第2個數字:1
請輸入第3個數字:5
請輸入第4個數字:13
請輸入第5個數字:8

Pass 1 [9, 4, 1, 5, 13, 8]


Pass 2 [4, 9, 1, 5, 13, 8]



Pass 3 [1, 4, 9, 5, 13, 8]


Pass 4 [1, 4, 5, 9, 13, 8] 無須交換

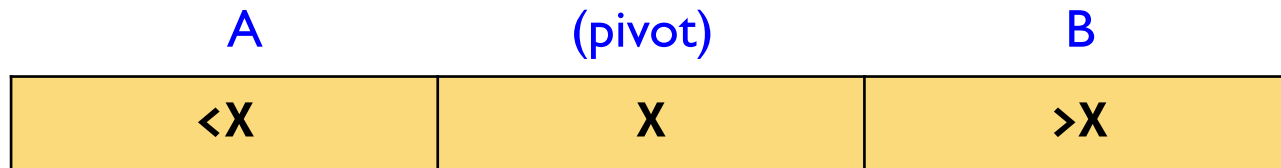

Pass 5 [1, 4, 5, 9, 13, 8]



result [1, 4, 5, 8, 9, 13]

Ch10.快速排序法(Quick Sort)

- ▶ 快速排序法的原理是選定一個數X(稱「基準」pivot)，然後設法將數列分成三份，如圖：

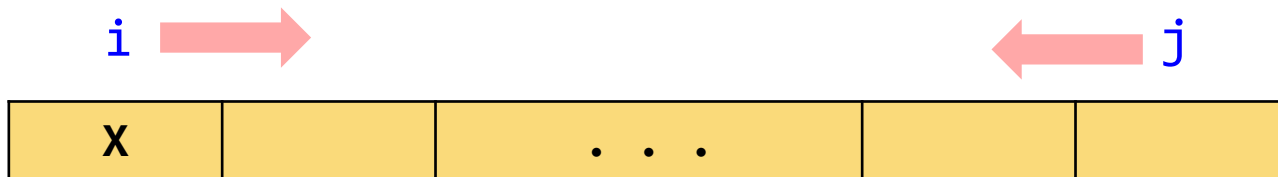


- ▶ 使用遞迴，以一樣的方式再分別處理A段和B段，直到所有數字都處理完畢。
- ▶ 遞迴到最底部時，數列的大小是零或一，也就是已經排序好了。
- ▶ 快速排序通常明顯比其他演算法更快。

Ch10.快速排序法(Quick Sort)

▶ 演算過程如下：

- ▶ 1. 令數列中最左邊的數為 $X(\text{pivot})$ 。
- ▶ 2. 設立兩個指標 i 和 j 。
- ▶ 3. 將 i 往右移，直到找到第一個 $A[i] \geq X$ 為止。
- ▶ 4. 將 j 往左移，直到找到第一個 $A[j] \leq X$ 為止。
- ▶ 5. 將 $A[i]$ 與 $A[j]$ 互換。
- ▶ 6. 重複以上動作直到 $i \geq j$ 為止。
- ▶ 7. 將 X 與 $A[i]$ 互換。



Ch10.快速排序法(Quick Sort)

▶ 原理示範說明：

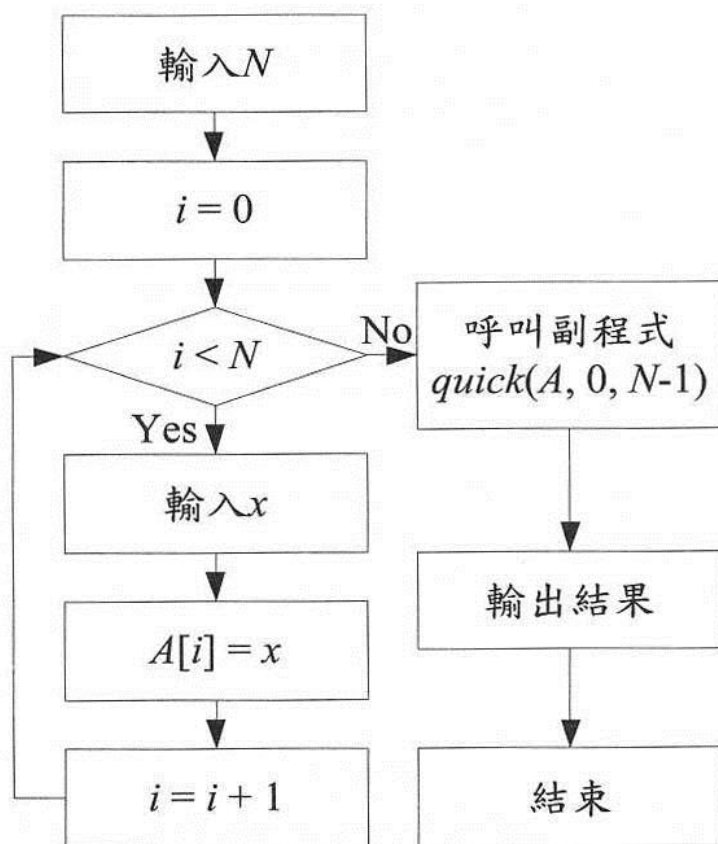
arr



Ch10.快速排序法(Quick Sort)

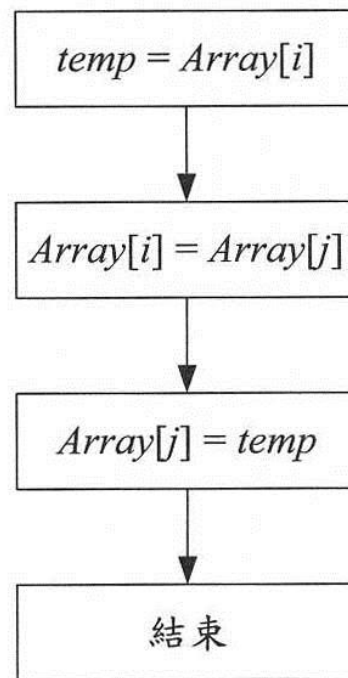
► 流程圖：

主程式



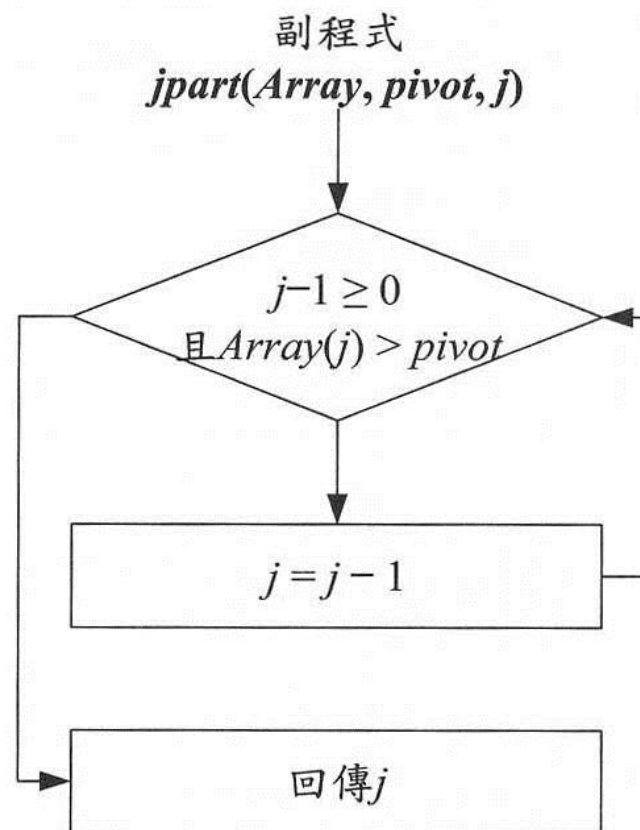
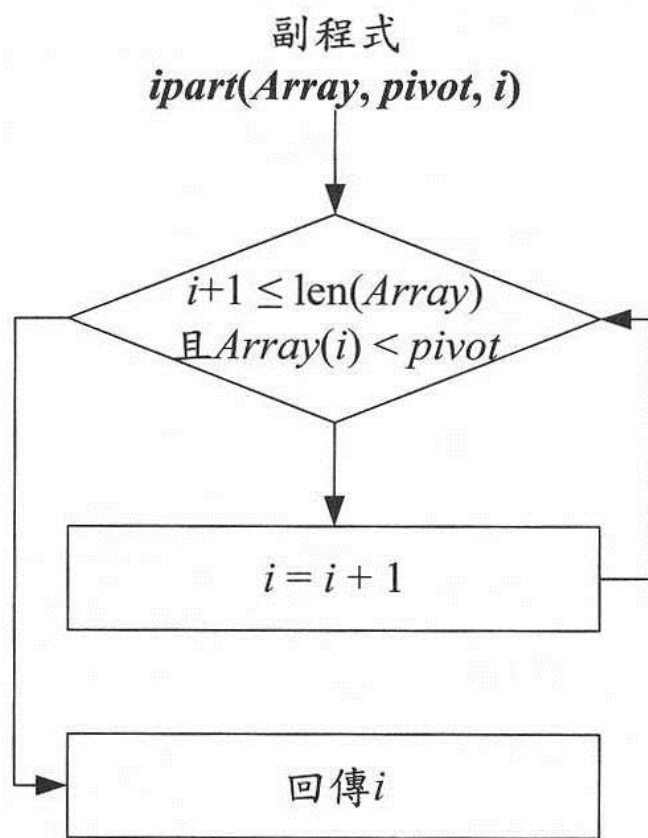
副程式

swap(Array, i, j)



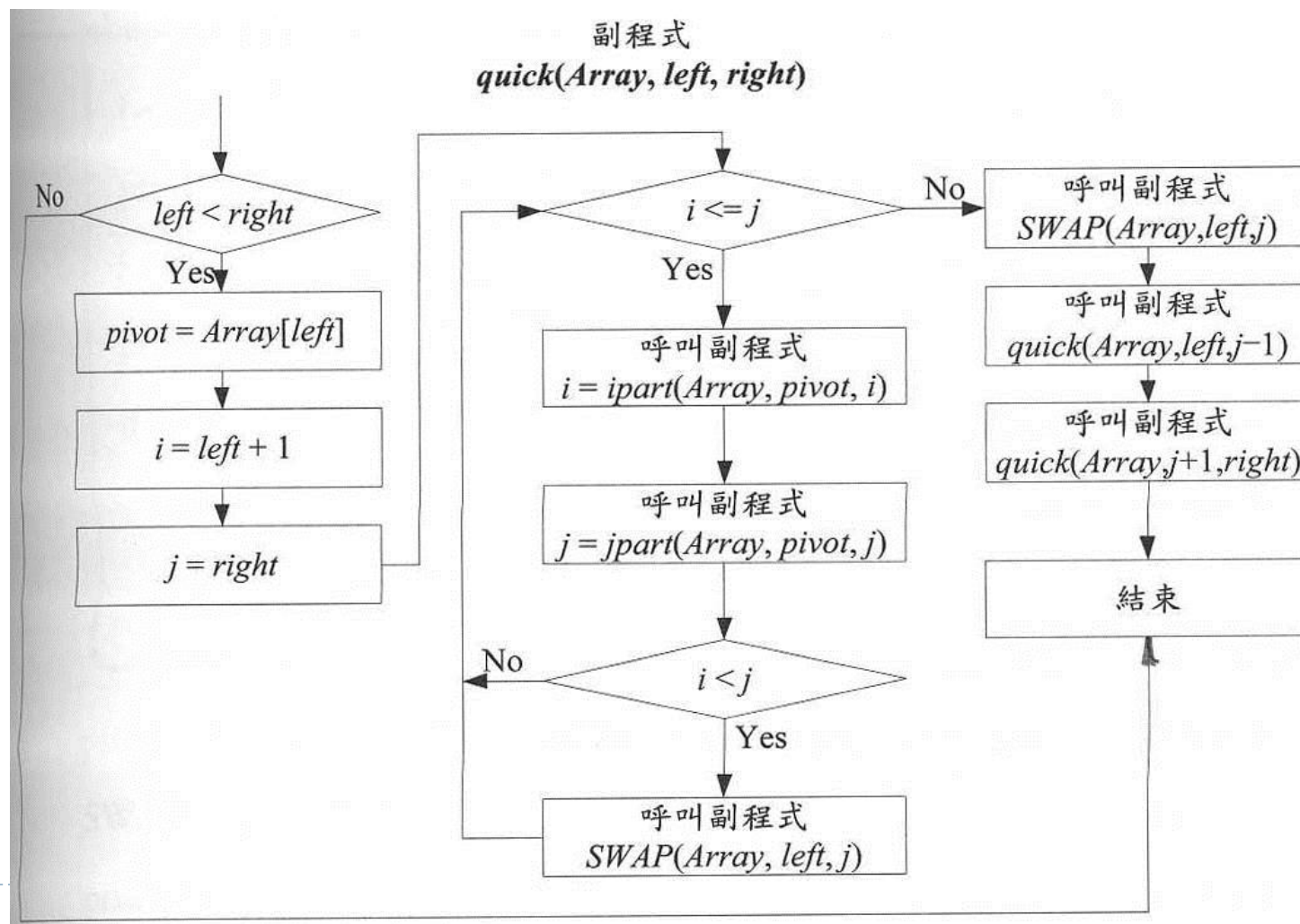
Ch10.快速排序法(Quick Sort)

► 流程圖：



Ch10.快速排序法(Quick Sort)

► 流程圖：



Ch10.快速排序法(Quick Sort)

▶ 參考程式：

```
#交換兩元素
def SWAP(Array,i,j):
    temp = Array[i]
    Array[i] = Array[j]
    Array[j] = temp
#將i往右移，直到找到第一個A[i]>=X為止
def ipart(Array,pivot,i):
    while i+1 <= len(Array) and Array[i]<pivot:
        i += 1
    return i
#將j往左移，直到找到第一個A[j]<=X為止
def jpart(Array,pivot,j):
    while j-1 >= 0 and Array[j]>pivot:
        j -= 1
    return j
```

Ch10.快速排序法(Quick Sort)

▶ 參考程式：

說明：

程式較冗長是因為
要印出執行的過程。

```
def quick(Array,left,right):
    print("Q: left=",left,"right=",right)
    if left < right:
        pivot = Array[left]
        print("pivot=A[",left,"]= ",pivot)
        i = left +1
        j = right

        while i <= j:
            i = ipart(Array,pivot,i)
            print("i=",i)
            j = jpart(Array,pivot,j)
            print("j=",j)
            if i < j:
                print("i<j，執行交換A[",i,"]和A[",j,"]")
                SWAP(Array,i,j)
                print(A)
            print("交換pivot和A[",j,"]")
            SWAP(Array,left,j)
            print(A)
        quick(Array,left,j-1)
        quick(Array,j+1,right)
```

Ch10.快速排序法(Quick Sort)

▶ 參考程式：

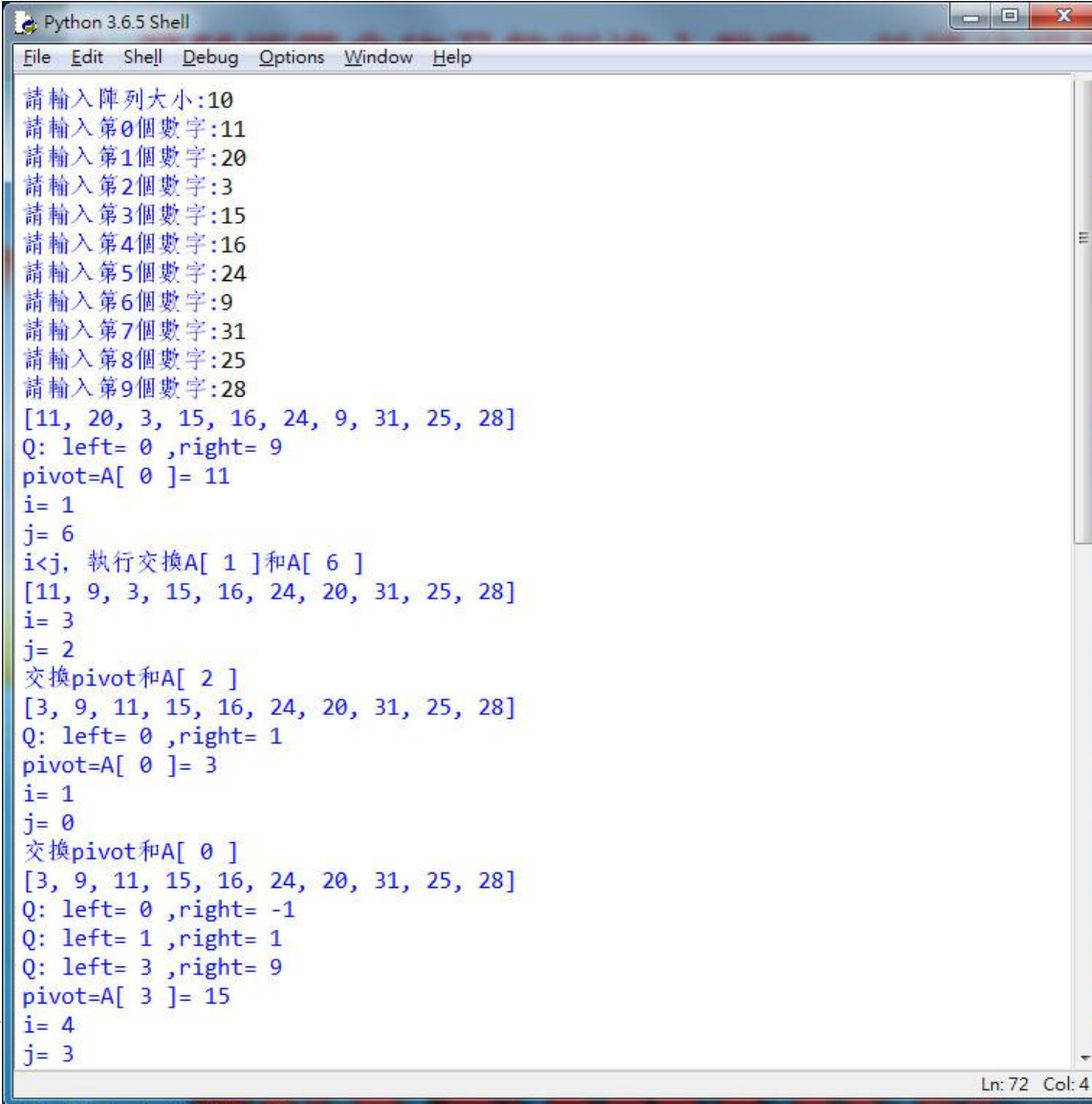
```
#main
A = []
N = int(input("請輸入陣列大小:"))

for i in range(N):
    x = int(input("請輸入第"+str(i)+"個數字:"))
    A.append(x)

print(A)
quick(A,0,N-1)
print(A)
```

Ch10.快速排序法(Quick Sort)

▶ 執行結果：



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help

請輸入陣列大小:10
請輸入第0個數字:11
請輸入第1個數字:20
請輸入第2個數字:3
請輸入第3個數字:15
請輸入第4個數字:16
請輸入第5個數字:24
請輸入第6個數字:9
請輸入第7個數字:31
請輸入第8個數字:25
請輸入第9個數字:28
[11, 20, 3, 15, 16, 24, 9, 31, 25, 28]
Q: left= 0 ,right= 9
pivot=A[ 0 ]= 11
i= 1
j= 6
i<j, 執行交換A[ 1 ]和A[ 6 ]
[11, 9, 3, 15, 16, 24, 20, 31, 25, 28]
i= 3
j= 2
交換pivot和A[ 2 ]
[3, 9, 11, 15, 16, 24, 20, 31, 25, 28]
Q: left= 0 ,right= 1
pivot=A[ 0 ]= 3
i= 1
j= 0
交換pivot和A[ 0 ]
[3, 9, 11, 15, 16, 24, 20, 31, 25, 28]
Q: left= 0 ,right= -1
Q: left= 1 ,right= 1
Q: left= 3 ,right= 9
pivot=A[ 3 ]= 15
i= 4
j= 3

Ln: 72 Col: 4
```

Ch10.快速排序法(Quick Sort)

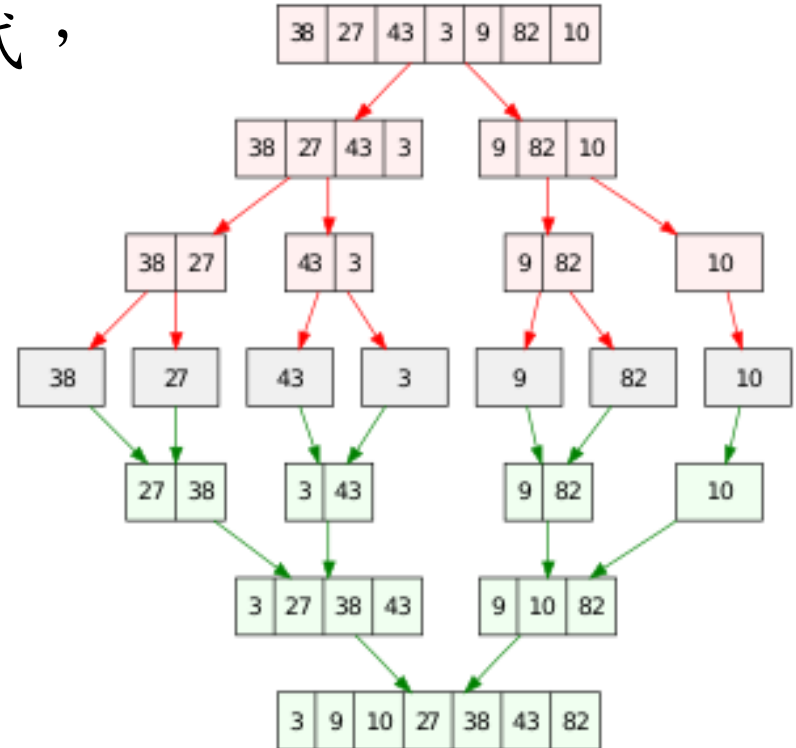
▶ 執行結果：

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
交換pivot和A[ 3 ]
[3, 9, 11, 15, 16, 24, 20, 31, 25, 28]
Q: left= 3 ,right= 2
Q: left= 4 ,right= 9
pivot=A[ 4 ]= 16
i= 5
j= 4
交換pivot和A[ 4 ]
[3, 9, 11, 15, 16, 24, 20, 31, 25, 28]
Q: left= 4 ,right= 3
Q: left= 5 ,right= 9
pivot=A[ 5 ]= 24
i= 7
j= 6
交換pivot和A[ 6 ]
[3, 9, 11, 15, 16, 20, 24, 31, 25, 28]
Q: left= 5 ,right= 5
Q: left= 7 ,right= 9
pivot=A[ 7 ]= 31
i= 10
j= 9
交換pivot和A[ 9 ]
[3, 9, 11, 15, 16, 20, 24, 28, 25, 31]
Q: left= 7 ,right= 8
pivot=A[ 7 ]= 28
i= 9
j= 8
交換pivot和A[ 8 ]
[3, 9, 11, 15, 16, 20, 24, 25, 28, 31]
Q: left= 7 ,right= 7
Q: left= 9 ,right= 8
Q: left= 10 ,right= 9
[3, 9, 11, 15, 16, 20, 24, 25, 28, 31]
>>> |
```

Ln: 72 Col: 4

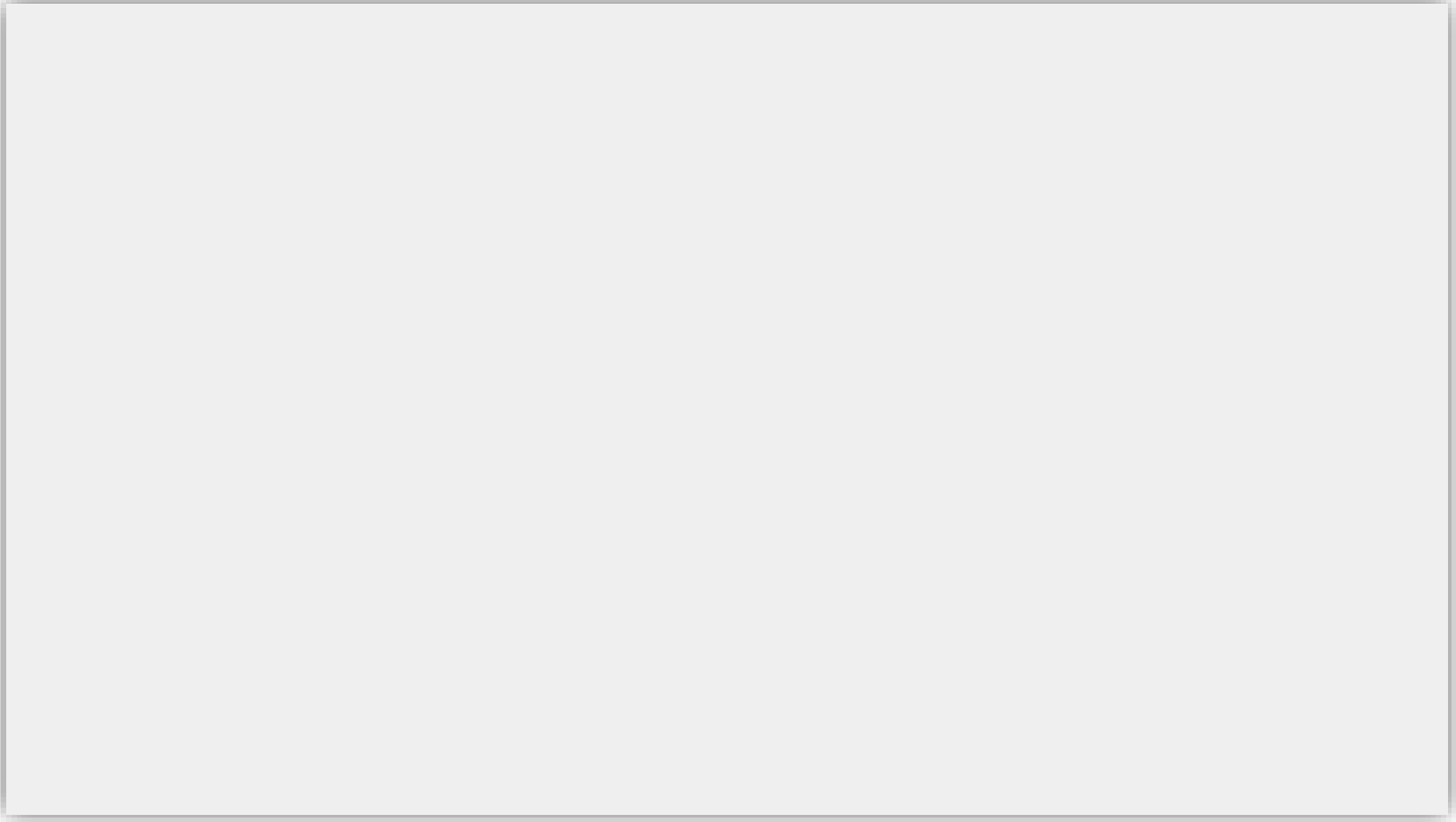
Ch10. 合併排序法(Merge Sort)

- ▶ 合併排序法是依據”合併”來排序的，假設我們有兩個已經排好的數列，我們就可以很容易的將兩個數列合併成一個排序好的數列。
- ▶ 採用先打散，在合併的方式，稱為divide and conquer。
- ▶ 一般都以遞迴來實現。



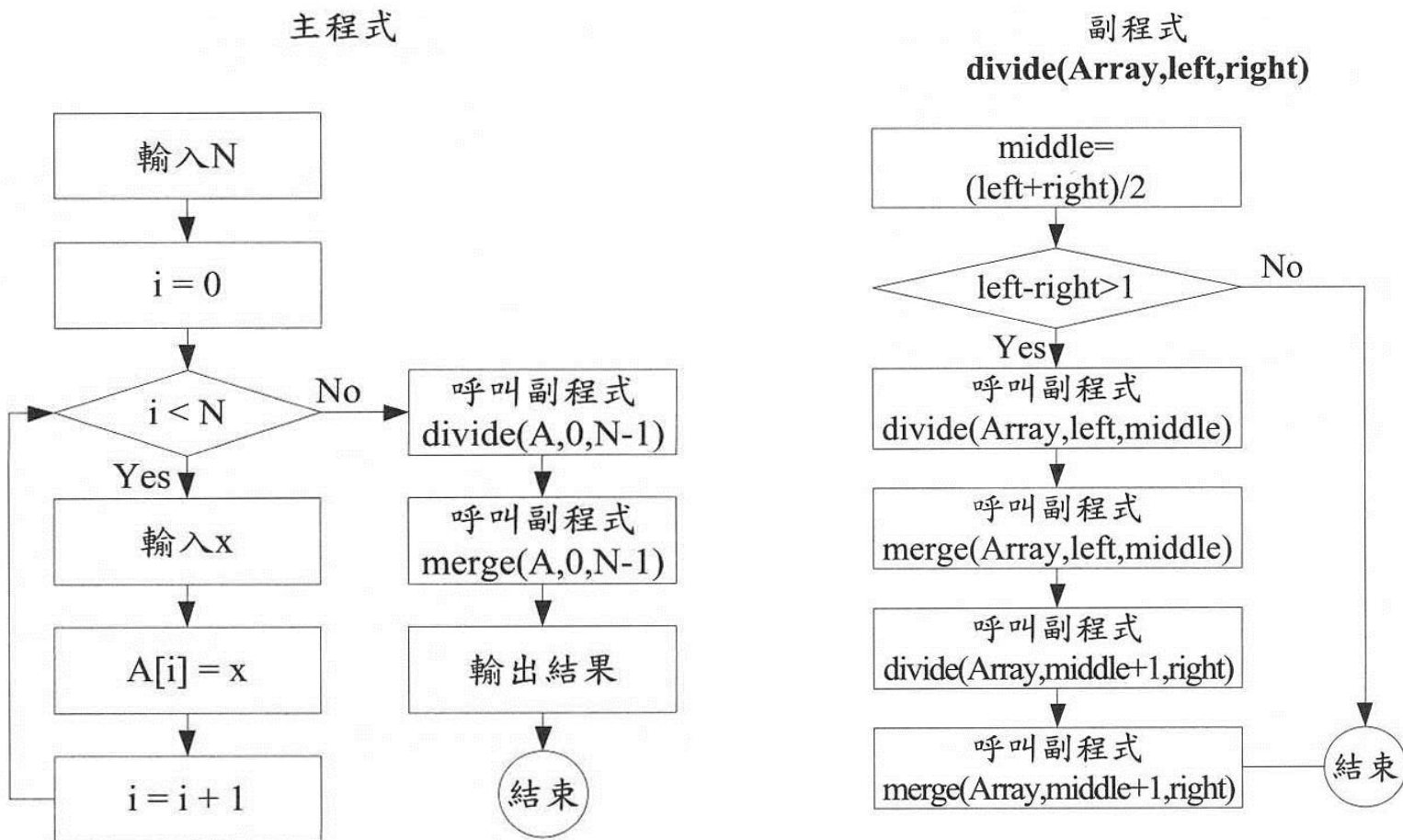
Ch10. 合併排序法(Merge Sort)

▶ 原理示範說明：



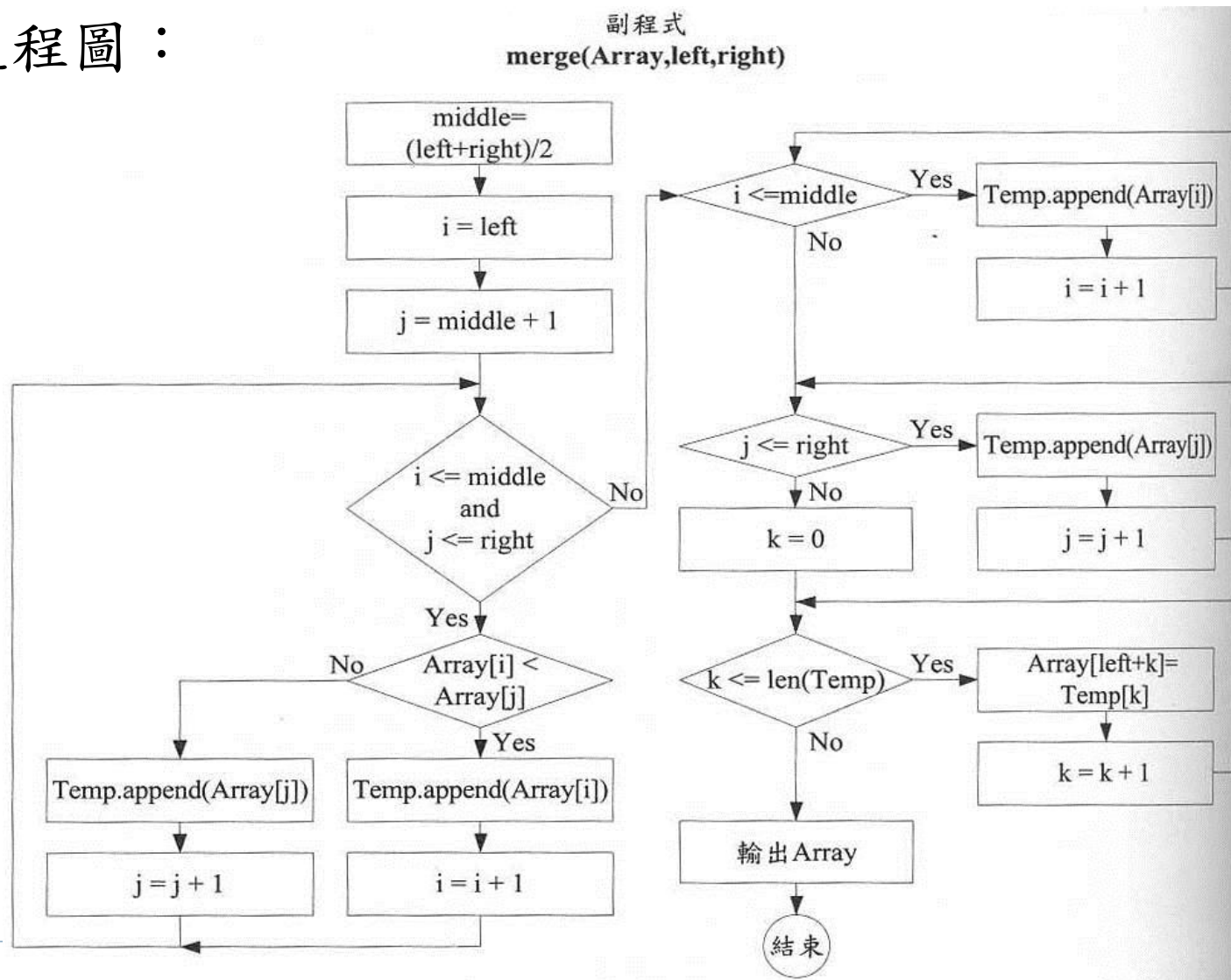
Ch10. 合併排序法(Merge Sort)

► 流程圖：



Ch10. 合併排序法(Merge Sort)

▶ 流程圖：



Ch10.合併排序法

▶ 參考程式：

```
def merge(Array,left,right):
    print("M: left=",left," right=",right)
    middle = int((left+right)/2)
    i = left
    j = middle + 1
    print("i=",i," j=",j)
    Temp = []
    while i <= middle and j <= right:
        if Array[i] < Array[j]:
            Temp.append(Array[i])
            i += 1
        else:
            Temp.append(Array[j])
            j += 1
    print("Temp=",Temp)
    while i <= middle:
        Temp.append(Array[i])
        i += 1
    while j <= right:
        Temp.append(Array[j])
        j += 1
    for k in range(len(Temp)):
        Array[left+k] = Temp[k]
    print(Array,end="")
    #將排序好的內容印出
    print("(",end="")
    leftRange = left
    while leftRange <= right:
        print(Array[leftRange],end="")
        leftRange += 1
    print(")排序好\n")
```

Ch10.合併排序法(Merge Sort)

▶ 參考程式：

```
def divide(Array,left,right):
    print("D:left=",left,"right=",right)
    middle = int((left+right)/2)
    if (right-left) > 1:
        divide(Array,left,middle)
        merge(Array,left,middle)
        divide(Array,middle+1,right)
        merge(Array,middle+1,right)
    #印出合併的提示訊息，左半邊要與右半邊合併
    print("接著將左邊(",end="")
    leftRange = left
    while leftRange <= middle:
        print(Array[leftRange],end="")
        leftRange += 1
    print(")及右邊(",end="")
    leftRange = middle + 1
    while leftRange <= right:
        print(Array[leftRange],end="")
        leftRange += 1
    print(")合併...")
```

Ch10.合併排序法(Merge Sort)

▶ 參考程式：

```
#main
N = int(input("請輸入陣列大小："))
A = [0 for i in range(N)]
for i in range(N):
    x = int(input("請輸入第"+str(i)+"個數字"))
    A[i] = x
print(A)
divide(A,0,N-1)
merge(A,0,N-1)
print("排序完成：",A)
```

Ch10.合併排序法(Merge Sort)

▶ 執行結果：

```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
請輸入陣列大小: 7
請輸入第0個數字6
請輸入第1個數字3
請輸入第2個數字9
請輸入第3個數字5
請輸入第4個數字2
請輸入第5個數字7
請輸入第6個數字1
[6, 3, 9, 5, 2, 7, 1]
D:left= 0 ,right= 6
D:left= 0 ,right= 3
D:left= 0 ,right= 1
M: left= 0 right= 1
i= 0 j= 1
Temp= [3]
[3, 6, 9, 5, 2, 7, 1](36)排序好

D:left= 2 ,right= 3
M: left= 2 right= 3
i= 2 j= 3
Temp= [5]
[3, 6, 5, 9, 2, 7, 1](59)排序好

接著將左邊(36)及右邊(59)合併...
M: left= 0 right= 3
i= 0 j= 2
Temp= [3, 5, 6]
[3, 5, 6, 9, 2, 7, 1](3569)排序好

D:left= 4 ,right= 6
D:left= 4 ,right= 5
M: left= 4 right= 5
i= 4 j= 5
```

```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Temp= [2]
[3, 5, 6, 9, 2, 7, 1](27)排序好

D:left= 6 ,right= 6
M: left= 6 right= 6
i= 6 j= 7
Temp= []
[3, 5, 6, 9, 2, 7, 1](1)排序好

接著將左邊(27)及右邊(1)合併...
M: left= 4 right= 6
i= 4 j= 6
Temp= [1]
[3, 5, 6, 9, 1, 2, 7](127)排序好

接著將左邊(3569)及右邊(127)合併...
M: left= 0 right= 6
i= 0 j= 4
Temp= [1, 2, 3, 5, 6, 7]
[1, 2, 3, 5, 6, 7, 9](1235679)排序好

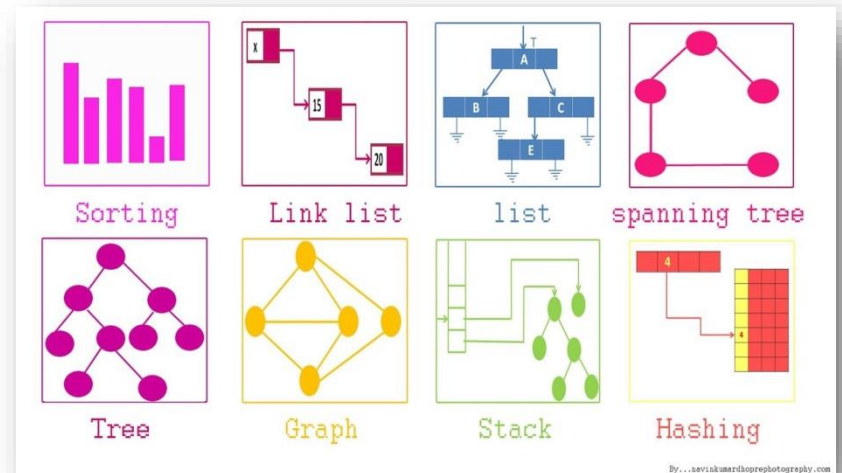
排序完成: [1, 2, 3, 5, 6, 7, 9]
>>>
```


休息一下~



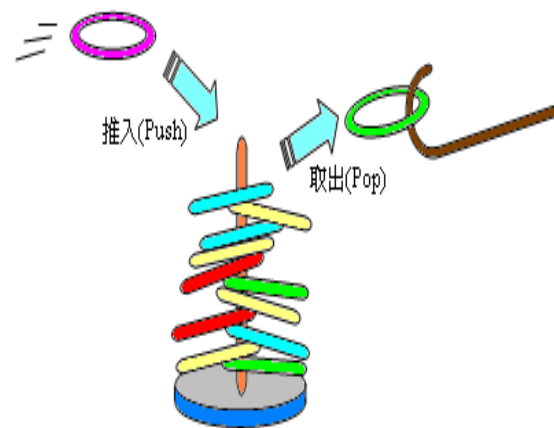
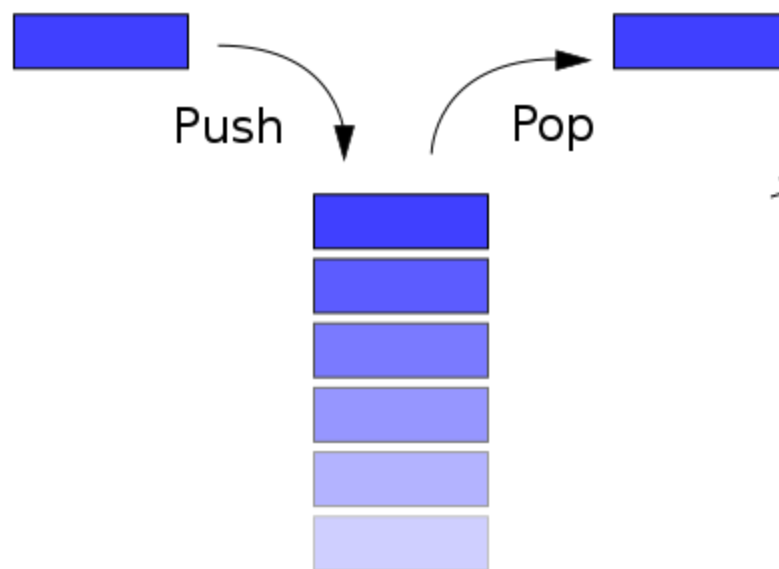
Ch11. 資料結構

- ▶ 資料結構(data structure)是電腦中儲存、組織資料的方式。
- ▶ 正確的資料結構選擇可以提高演算法的效率。在電腦程式設計的過程中，選擇適當的資料結構是一項重要工作。
- ▶ 常見的資料結構：
 - ▶ 陣列 (Array)
 - ▶ 堆疊 (Stack)
 - ▶ 佇列 (Queue)
 - ▶ 連結串列 (Linked List)
 - ▶ 樹 (Tree)
 - ▶ 圖 (Graph)
 - ▶ 堆積 (Heap)
 - ▶ 雜湊表 (Hash)



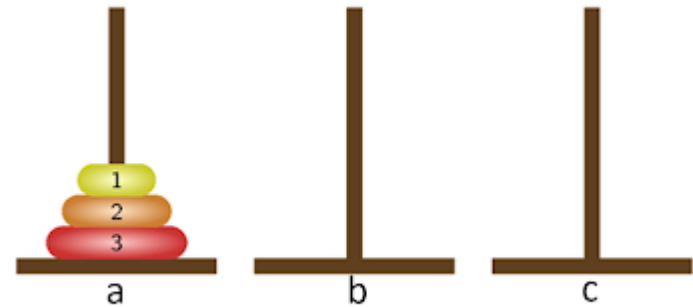
Ch11. 資料結構 – 堆疊(Stack)

- ▶ 在日常生活中，把物品(如餐盤、書本)由桌面一個一個向上疊放，取用時由最上面一個個向下拿去，這種觀念稱為堆疊(stack)
- ▶ 存入稱**PUSH**，取出稱**POP**



Ch11. 資料結構 – 堆疊(Stack)

- ▶ 堆疊觀念：先進後出(First In Last Out, **FILO**)或稱為後進先出(Last In First Out, **LIFO**)
- ▶ 堆疊觀念在計算機中使用很廣，例如主副程式間訊息傳送，CPU中斷處理、遞迴程式的呼叫及返還
- ▶ 最著名的問題就是：河內塔



Ch11. 資料結構 – 堆疊(Stack)

- ▶ 使用堆疊結構，寫一程式，使用者可以有四種選擇：
 - ▶ 1.加入數字
 - ▶ 2.取出數字
 - ▶ 3.查看Stack
 - ▶ 4.離開程式
- ▶ (一般都是用陣列來做。)

- ▶ 將資料y存入堆疊s，使用指令：

```
s.append(y)
```

- ▶ 將最後一個資料pop出來：

```
y = s.pop(len(s)-1)
```

Ch11. 資料結構 – 堆疊(Stack)

▶ 參考程式：

```
#加入一個資料到堆疊
def addNumberToStack(s):
    y = int(input("請輸入要加進stack的數字:"))
    s.append(y)
#從堆疊取出一個資料
def popNumberFromStack(s):
    if len(s) != 0:
        y = s.pop(len(s)-1)
        print("從stack裡取出的數字為:"+str(y))
    else:
        print("Stack現在是空的!")
#印出堆疊內容
def showStack(s):
    if len(s) != 0:
        print("Stack的內容為: "+str(s))
    else:
        print("Stack現在是空的!")
```

Ch11. 資料結構 – 堆疊(Stack)

▶ 參考程式：

```
#main
S = []
while True:
    x = int(input("[STACK] 1.加入數字 2.取出數字 3.查看  
4.離開程式 || 請選擇功能:"))

    if x == 1:
        addNumberToStack(S)
    elif x == 2:
        popNumberFromStack(S)
    elif x == 3:
        showStack(S)
    elif x == 4:
        break
```

Ch11. 資料結構－堆疊(Stack)

▶ 執行結果：

```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\MyDocs\MyProgram\專門為中學生寫的程式語言設計\Python\ch11\
11-1.py =====
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:1
請輸入要加進stack的數字:16
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:1
請輸入要加進stack的數字:88
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:1
請輸入要加進stack的數字:49
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:3
Stack的內容為: [16, 88, 49]
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:2
從stack裡取出的數字為:49
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:3
Stack的內容為: [16, 88]
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:2
從stack裡取出的數字為:88
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:3
Stack的內容為: [16]
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:2
從stack裡取出的數字為:16
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:3
Stack現在是空的!
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:2
Stack現在是空的!
[STACK] 1. 加入數字 2. 取出數字 3. 查看 4. 離開程式 || 請選擇功能:4
>>> |
```

Ln: 28 Col: 4

Ch11. 資料結構 – 佇列(Queue)

- ▶ 顧名思義是一種像排隊一樣的概念
- ▶ 在這個模式下我們可以知道它是一種先進先出(First-In-First-Out, **FIFO**)的排程
- ▶ 存入稱**Enqueue**，取出稱**Dequeue**



Ch11. 資料結構 – 佇列(Queue)

▶ 使用佇列結構，寫一程式，使用者可以有四種選擇：

- ▶ 1. 加入數字
- ▶ 2. 取出數字
- ▶ 3. 查看Queue
- ▶ 4. 離開程式

▶ 將資料y存入佇列q，使用指令：

```
q.append(y)
```

▶ 將最前面的資料取出來：

```
y = q.pop(len(0))
```

▶ 在Python中我們仍用pop方法來取出資料，只是從陣列最前頭取出。

Ch11. 資料結構 – 佇列(Queue)

▶ 參考程式：

```
#加入一個資料到佇列
def addNumberToQueue(q):
    y = int(input("請輸入要加進Queue的數字:"))
    q.append(y)
#從佇列取出一個資料
def popNumberFromQueue(q):
    if len(q) != 0:
        y = q.pop(0)
        print("從Queue裡取出的數字為:"+str(y))
    else:
        print("Queue現在是空的!")
#印出佇列內容
def showQueue(q):
    if len(q) != 0:
        print("Queue的內容為: "+str(q))
    else:
        print("Queue現在是空的!")
```

Ch11. 資料結構 – 佇列(Queue)

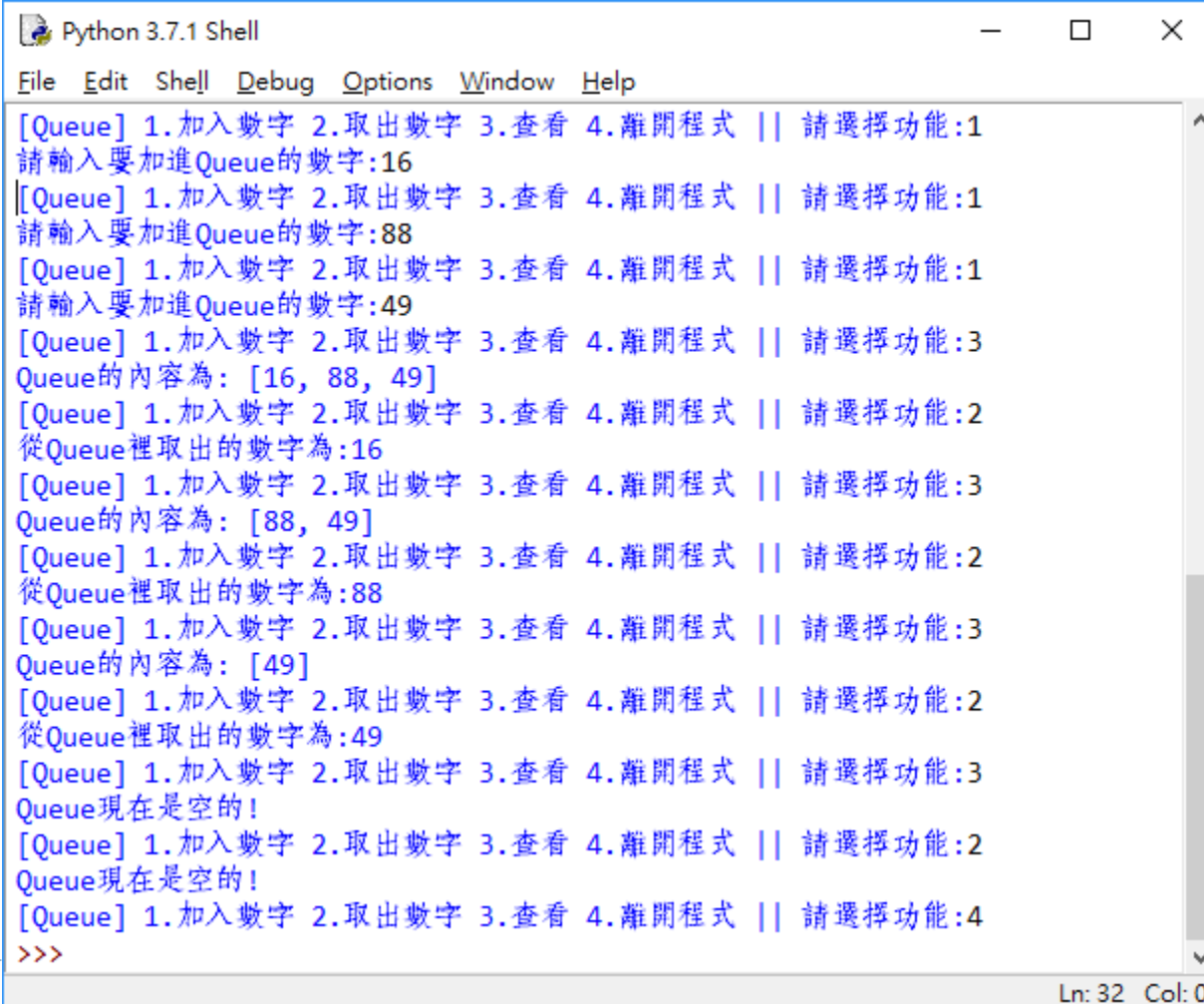
▶ 參考程式：

```
#main
q = []
while True:
    x = int(input("[Queue] 1.加入數字 2.取出數字 3.查看  
4.離開程式 || 請選擇功能:"))

    if x == 1:
        addNumberToQueue(q)
    elif x == 2:
        popNumberFromQueue(q)
    elif x == 3:
        showQueue(q)
    elif x == 4:
        break
```

Ch11. 資料結構 – 佇列(Queue)

▶ 執行結果：

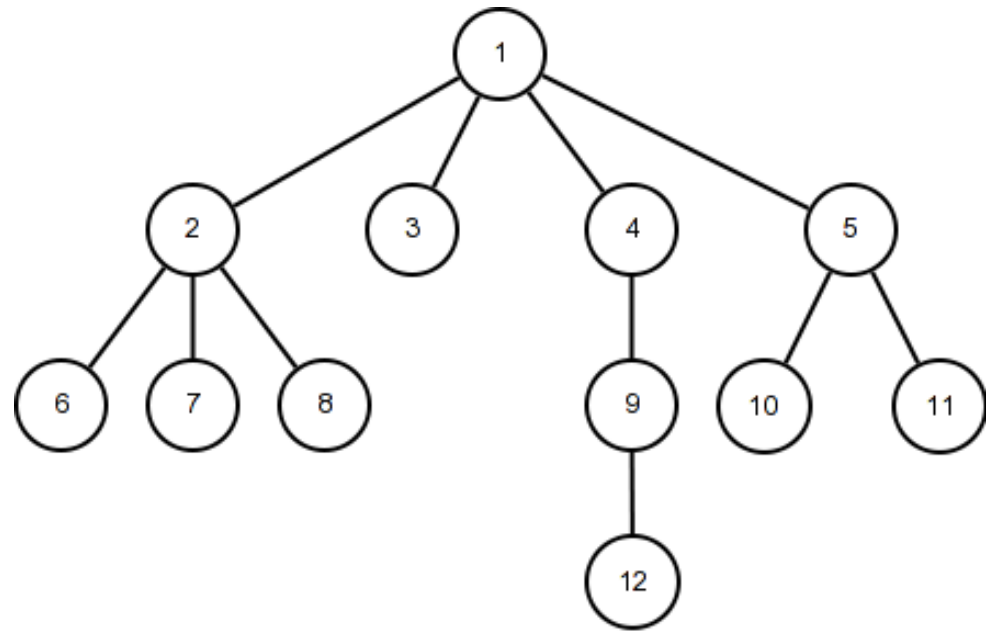
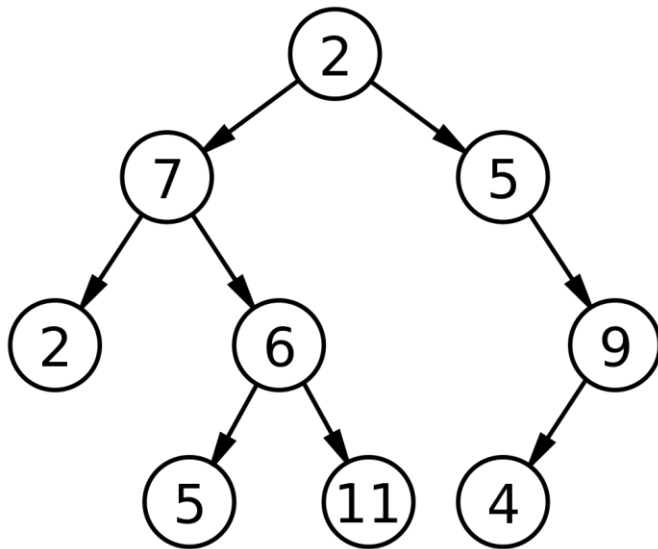


```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:1
請輸入要加進Queue的數字:16
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:1
請輸入要加進Queue的數字:88
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:1
請輸入要加進Queue的數字:49
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:3
Queue的內容為: [16, 88, 49]
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:2
從Queue裡取出的數字為:16
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:3
Queue的內容為: [88, 49]
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:2
從Queue裡取出的數字為:88
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:3
Queue的內容為: [49]
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:2
從Queue裡取出的數字為:49
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:3
Queue現在是空的!
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:2
Queue現在是空的!
[Queue] 1.加入數字 2.取出數字 3.查看 4.離開程式 || 請選擇功能:4
>>>
```

Ln: 32 Col: 0

Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

- ▶ 樹是一種資料結構方式，使用連結串列(Link List)來組成資料，因為連接的方式像一棵樹，故稱之。
- ▶ 這都是「樹」：



Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

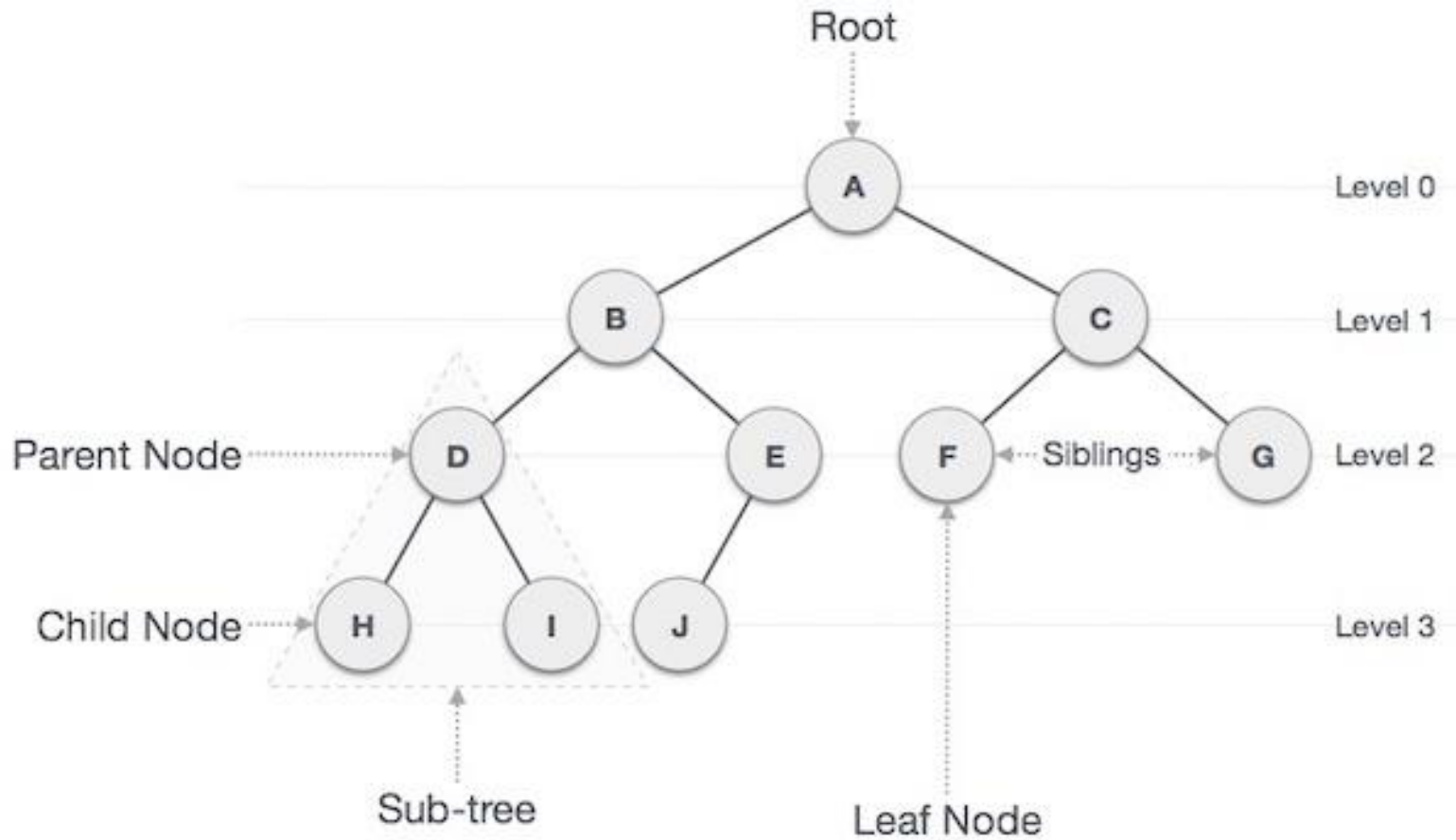
▶ 樹的相關定義：

- ▶ 1. 一個樹包含了一組有限的元素，稱為節點(Node)。
- ▶ 2. 一組有限的方向線段，稱分支(Branch)，連接到結點上。
- ▶ 3. 和這個節點有關的分支數目稱為此節點的degree。
- ▶ 4. 當一個分支指向一個節點時，稱此分支為indegree。
- ▶ 5. 當一個分支離開一個節點時，稱此分支為outdegree。
- ▶ 6. indegree和outdegree的總和等於該節點的degree。
- ▶ 7. 第一個節點稱為根(Root)，除了根節點外，任何節點都必須至少有一個indegree，但outdegree則沒有限制。



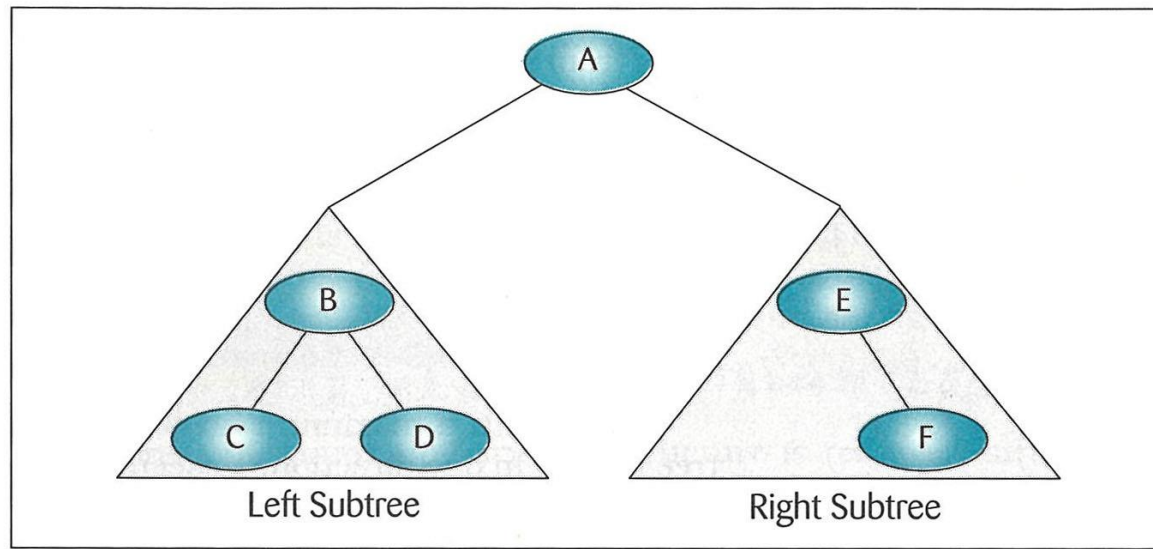
Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

► 樹的一些專有名詞：



Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

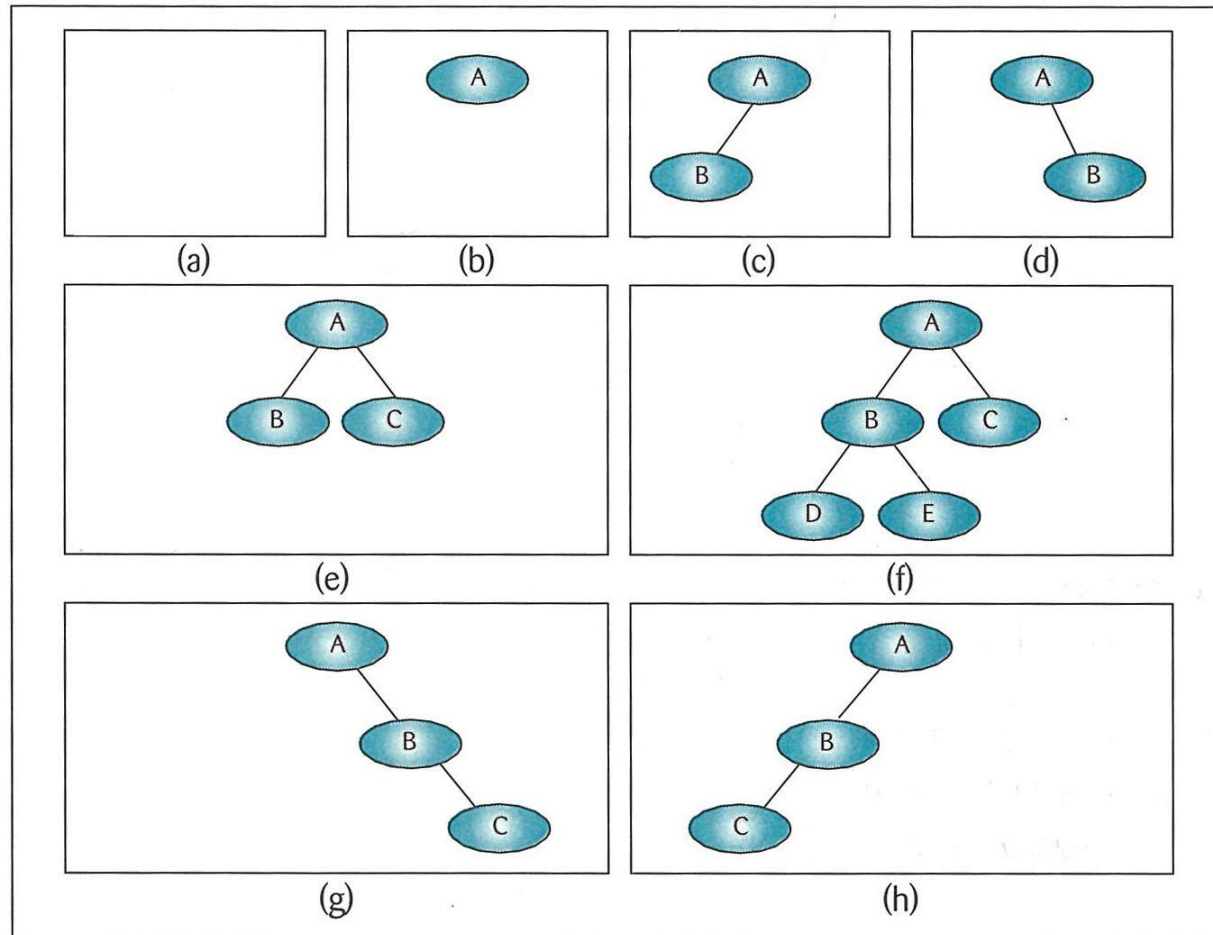
- ▶ 二元樹定義：
- ▶ 1.所有的節點(node)至多只能有兩個子樹，稱二元樹。
- ▶ 2.左邊稱為左子樹，右邊稱為右子樹。
- ▶ 3.樹不一定要對稱。



Binary tree

Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

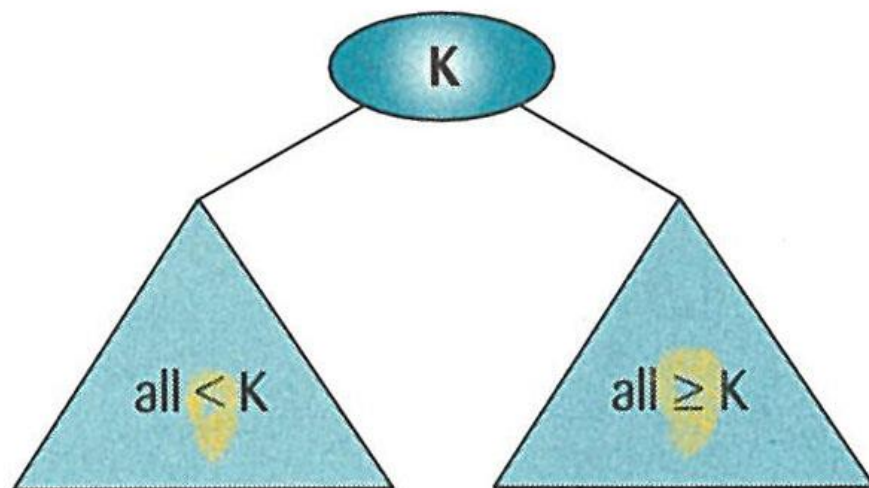
► 一些二元樹的例子：



A collection of binary trees

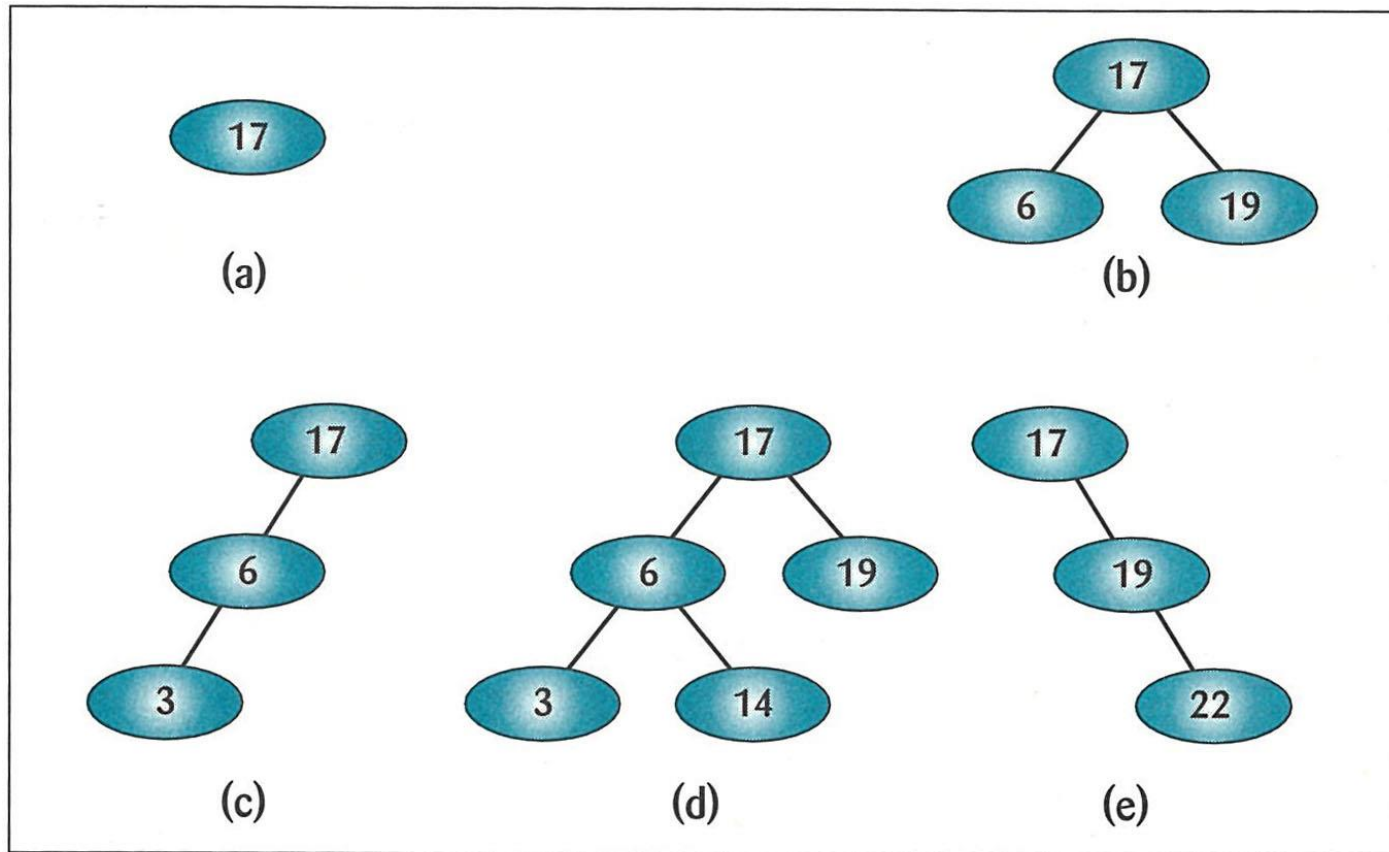
Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

- ▶ 二元搜尋樹(Binary Search Tree, BST)是一個二元樹，並具有下列性質：
- ▶ 1.任一左子樹都比它的根來的小。
- ▶ 2.任一右子樹都大於或等於它的根。
- ▶ 3.每一個子樹亦是一個二元搜尋樹。



Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

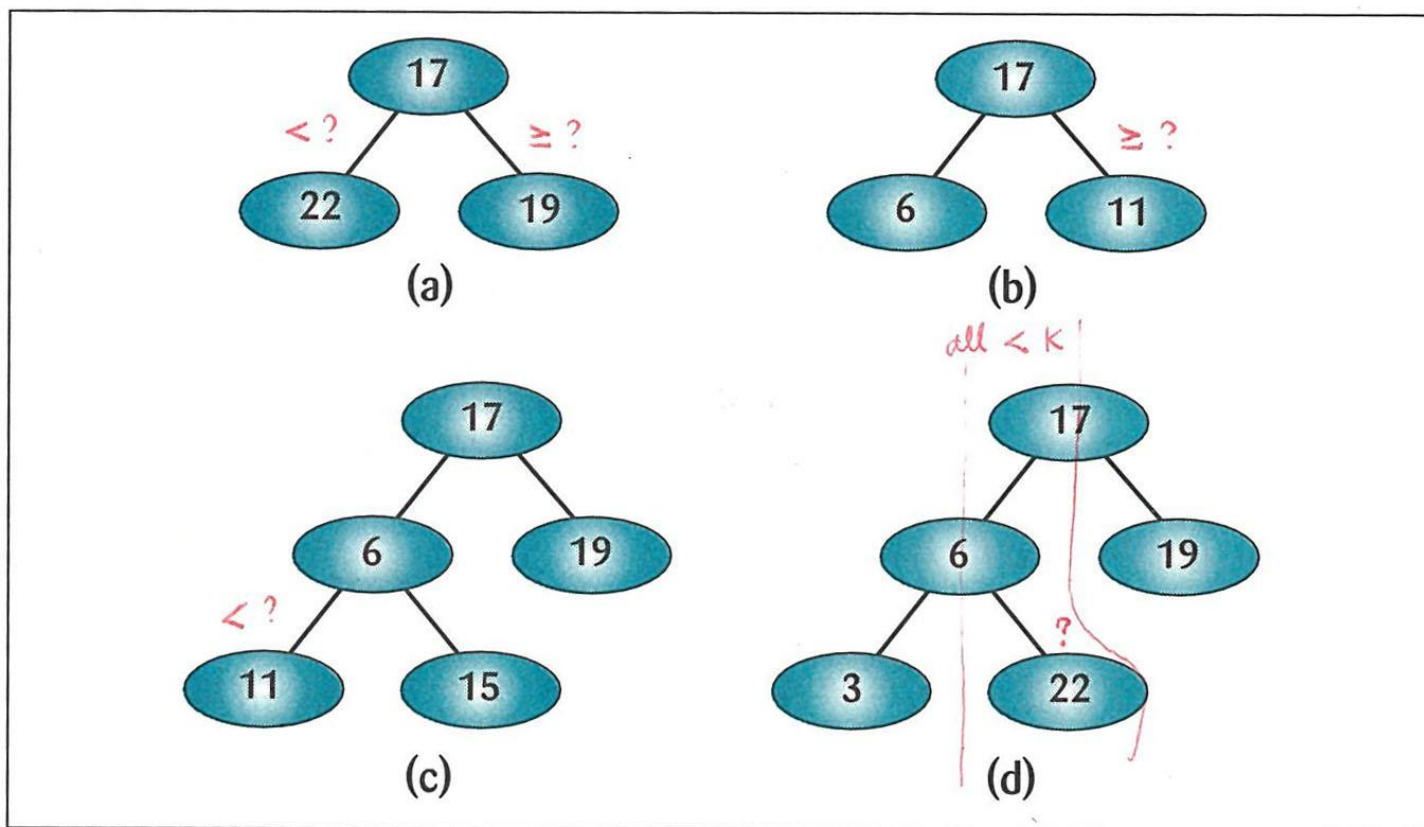
► 一些二元搜尋樹的例子：



Binary search trees

Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

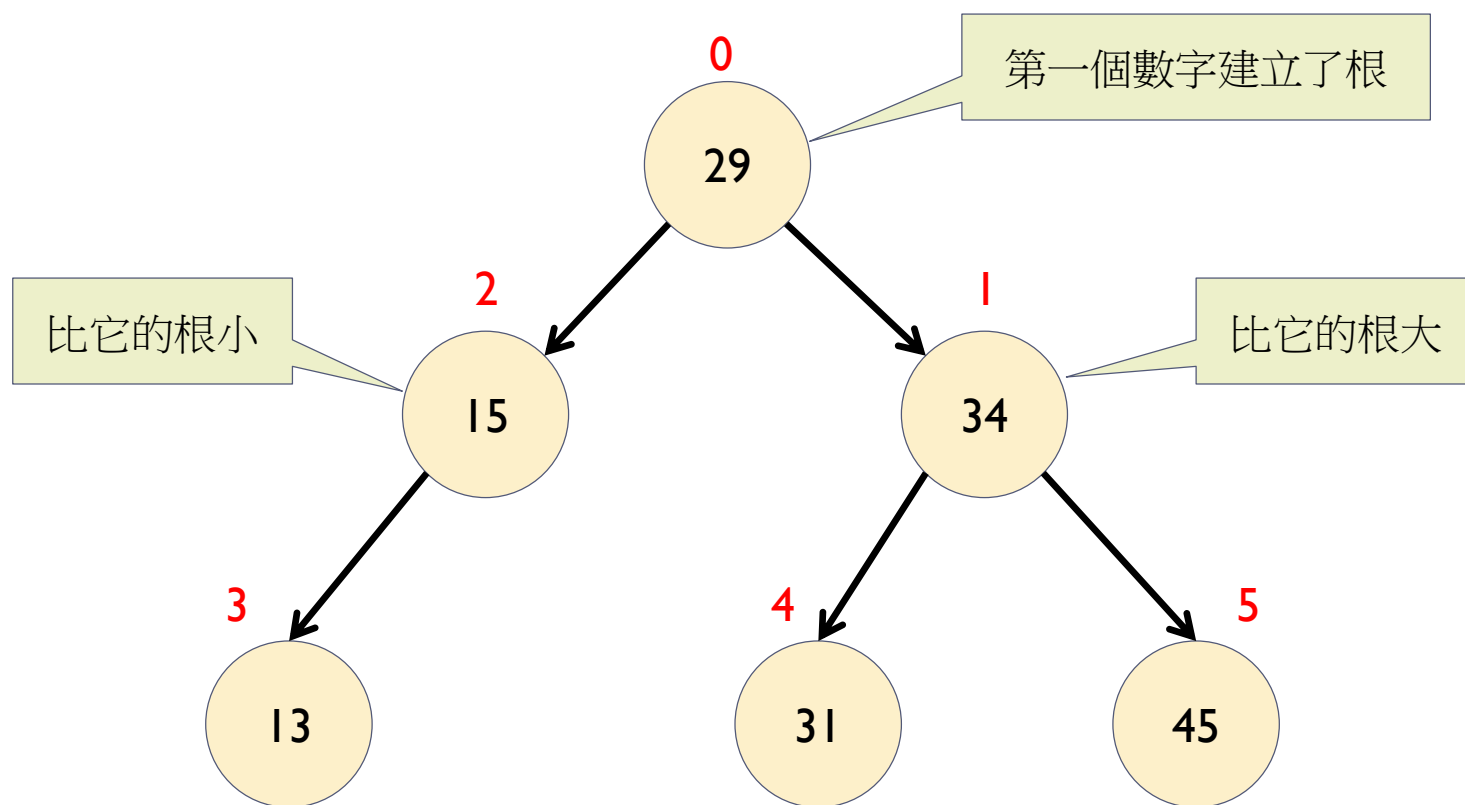
- 一些無效的二元搜尋樹的例子：



Invalid binary search trees

Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

- ▶ 如何建立一個二元搜尋樹
- ▶ 假設輸入數字順序如下：29、34、15、13、31、45



Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

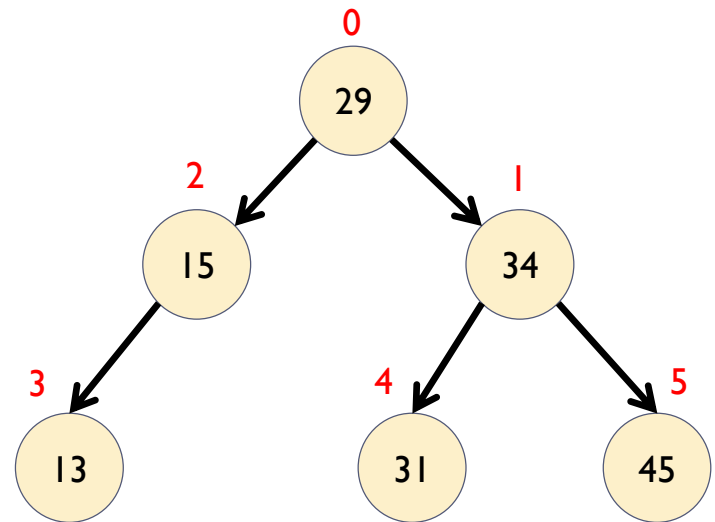
- ▶ 在這裡我們使用陣列來表示一個節點：

Node $i[0]$ = 節點的值
Node $i[1]$ = 節點的編號
Node $i[2]$ = 節點的左子節點編號
Node $i[3]$ = 節點的右子節點編號

- ▶ 所以一棵樹可用一個二維陣列來表示：

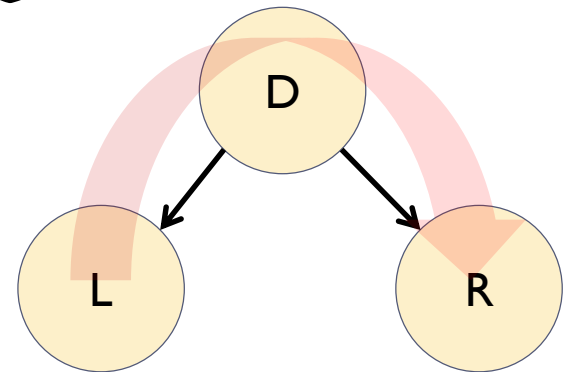
Tree[0] = [29, 0, 2, 1]
Tree[1] = [34, 1, 4, 5]
Tree[2] = [15, 2, 3, None]
Tree[3] = [13, 3, None, None]
Tree[4] = [31, 4, None, None]
Tree[5] = [45, 5, None, None]

- ▶ None表示無子節點。



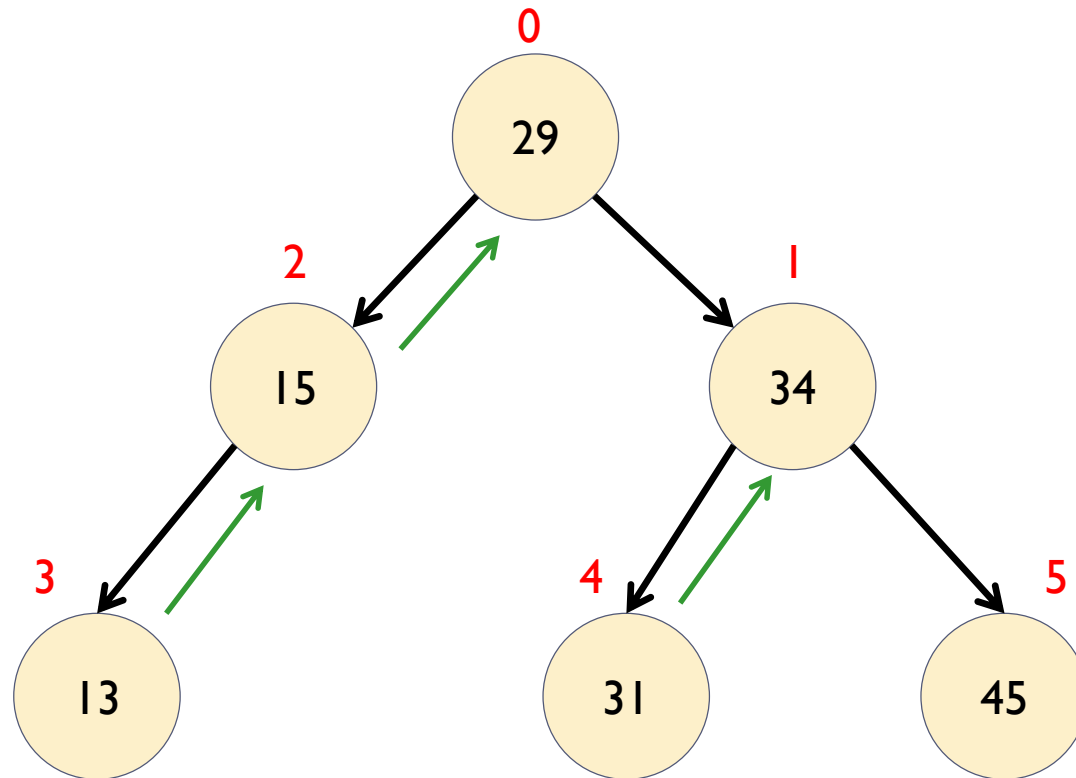
Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

- ▶ 二元搜尋樹走訪(列出所有資料)，有三種方式：
 - ▶ DLR: 前序走訪 (Preorder Traversal)
 - ▶ **LDR: 中序走訪(Inorder Traversal)**
 - ▶ LRD: 後序走訪(Postorder Traversal)
 - ▶ (D: root, L: 左子樹, R: 右子樹)
- ▶ 以LDR為例，是先走訪左子樹(直到沒有左子樹為止)，根結點，然後右子樹。如果右子樹底下還有左子樹，擇期左子樹要優先訪，這是一個遞迴。



Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

► LDR 中序走訪 (Inorder Traversal) :



輸出結果： 13 15 29 31 34 45

Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

- ▶ 二元搜尋樹的輸入：將一個數字input_value輸入到某一節點index。

```
def insert(index,input_value,tree):
    #若輸入的數字比此node所儲存的值還小，則針對left chile tree
    if input_value < tree[index][0]:
        #若left chile tree存在，試著在left sub-tree中找到適合的位置插入輸入的數字
        if tree[index][2] != None:
            insert(tree[index][2],input_value,tree)
        #若left chile tree不存在，將此node產生一個新的left chile node並儲存此輸入的數字
        else:
            #產生一個新的node
            newNode = [input_value,len(tree),None,None]
            #將此新的node加入tree中
            tree.append(newNode)
            #tree[index][2]記錄此新加入node的index
            tree[index][2] = len(tree) - 1
            print("加入新node:NODE["+str(len(tree)-1)+
                  "](此node為NODE["+str(tree[index][1])+"]的left chile node).")
    #若輸入的數字比此node所儲存的值還大，則針對right chile tree
    elif input_value >= tree[index][0]:
        if tree[index][3] != None:
            insert(tree[index][3],input_value,tree)
        #若left chile tree不存在，將此node產生一個新的left chile node並儲存此輸入的數字
        else:
            #產生一個新的node
            newNode = [input_value,len(tree),None,None]
            #將此新的node加入tree中
            tree.append(newNode)
            #tree[index][3]記錄此新加入node的index
            tree[index][3] = len(tree) - 1
            print("加入新node:NODE["+str(len(tree)-1)+
                  "](此node為NODE["+str(tree[index][1])+"]的right chile node).")
```

Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

▶ 二元搜尋樹的搜尋：

```
def searchValueFromTree(index,input_value,tree):
    print("NODE["+str(tree[index][1])+"]所儲存的數字為"+str(tree[index][0]))
    #若比此node所儲存的數字還小，則找left sub-tree
    if input_value < tree[index][0]:
        print("尋找NODE["+str(tree[index][1])+
              "]的left child node:NODE["+str(tree[index][2])+"]")
        if tree[index][2] == None:
            print("NODE["+str(tree[index][1])+
                  "]的left child node不存在，找不到"+str(input_value)+"這個數字!")
        else:
            searchValueFromTree(tree[index][2],input_value,tree)
    #若比此node所儲存的數字還大，則找right sub-tree
    elif input_value > tree[index][0]:
        print("尋找NODE["+str(tree[index][1])+
              "]的right child node:NODE["+str(tree[index][3])+"]")
        if tree[index][3] == None:
            print("NODE["+str(tree[index][1])+
                  "]的right child node不存在，找不到"+str(input_value)+"這個數字!")
        else:
            searchValueFromTree(tree[index][3],input_value,tree)
    #此node所儲存的數字與input_value相同
    else:
        print(str(input_value)+"這個數字存在tree中(位置在NODE["+str(tree[index][1])+"])!")
```

Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

▶ 二元搜尋樹的讀取資料：

```
def inorderShowTree(index,tree):  
    if tree[index][2] != None:  
        inorderShowTree(tree[index][2],tree)  
    print("NODE["+str(tree[index][1])+  
          " ]所儲存的數字為"+str(tree[index][0])+  
          ",left chile node為NODE["+str(tree[index][2])+  
          " ],right chile node為NODE["+str(tree[index][3])+"]")  
    if tree[index][3] != None:  
        inorderShowTree(tree[index][3],tree)
```

▶ 要讀出整棵樹裡面各節點的數字，我們用採中序式走訪。

Ch11. 資料結構 – 二元搜尋樹(Binary Search Tree)

▶ 主程式：

```
#main
T = []
while True:
    x = int(input("[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:"))
    if x == 1:
        y = int(input("請輸入要加入的數字:"))
        if len(T) == 0:
            #root[0]代表data, root[1]代表左child的index, root[2]代表右child的index
            root = [y, len(T), None, None]
            T.append(root) #將此node加入binary tree T中
        else:
            #試著找到適合的地方, 將輸入的數字插入
            insert(0, y, T)
    elif x == 2:
        y = int(input("請輸入要搜尋的數字:"))
        print("從NODE["+str(T[0][1])+"]開始尋找...")
        searchValueFromTree(0, y, T)
    elif x == 3:
        inorderShowTree(0, T)
    elif x == 4:
        break
```

Ch11. 資料結構 —

► 執行結果：

```
*Python 3.5.2 Shell*
File Edit Shell Debug Options Window Help

[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:1
請輸入要加入的數字:16
[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:1
請輸入要加入的數字:88
加入新node:NODE[1](此node為NODE[0]的right chile node).
[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:1
請輸入要加入的數字:49
加入新node:NODE[2](此node為NODE[1]的left chile node).
[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:1
請輸入要加入的數字:8
加入新node:NODE[3](此node為NODE[0]的left chile node).
[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:1
請輸入要加入的數字:65
加入新node:NODE[4](此node為NODE[2]的right chile node).
[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:1
請輸入要加入的數字:99
加入新node:NODE[5](此node為NODE[1]的right chile node).
[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:1
請輸入要加入的數字:4
加入新node:NODE[6](此node為NODE[3]的left chile node).
[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:2
請輸入要搜尋的數字:10
從NODE[0]開始尋找...
NODE[0]所儲存的數字為16
尋找NODE[0]的left child node:NODE[3]
NODE[3]所儲存的數字為8
尋找NODE[3]的right child node:NODE[None]
NODE[3]的right child node不存在, 找不到10這個數字!
[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:2
請輸入要搜尋的數字:8
從NODE[0]開始尋找...
NODE[0]所儲存的數字為16
尋找NODE[0]的left child node:NODE[3]
NODE[3]所儲存的數字為8
8這個數字存在tree中(位置在NODE[3])!
[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:3
NODE[6]所儲存的數字為4,left chile node為NODE[None],right chile node為NODE[None]
NODE[3]所儲存的數字為8,left chile node為NODE[6],right chile node為NODE[None]
NODE[0]所儲存的數字為16,left chile node為NODE[3],right chile node為NODE[1]
NODE[2]所儲存的數字為49,left chile node為NODE[None],right chile node為NODE[4]
NODE[4]所儲存的數字為65,left chile node為NODE[None],right chile node為NODE[None]
NODE[1]所儲存的數字為88,left chile node為NODE[2],right chile node為NODE[5]
NODE[5]所儲存的數字為99,left chile node為NODE[None],right chile node為NODE[None]
[Tree] 1.加入數字 2.搜尋數字 3.中序查看Tree 4.離開程式 || 請選擇功能:4
>>> |
```


下課~

