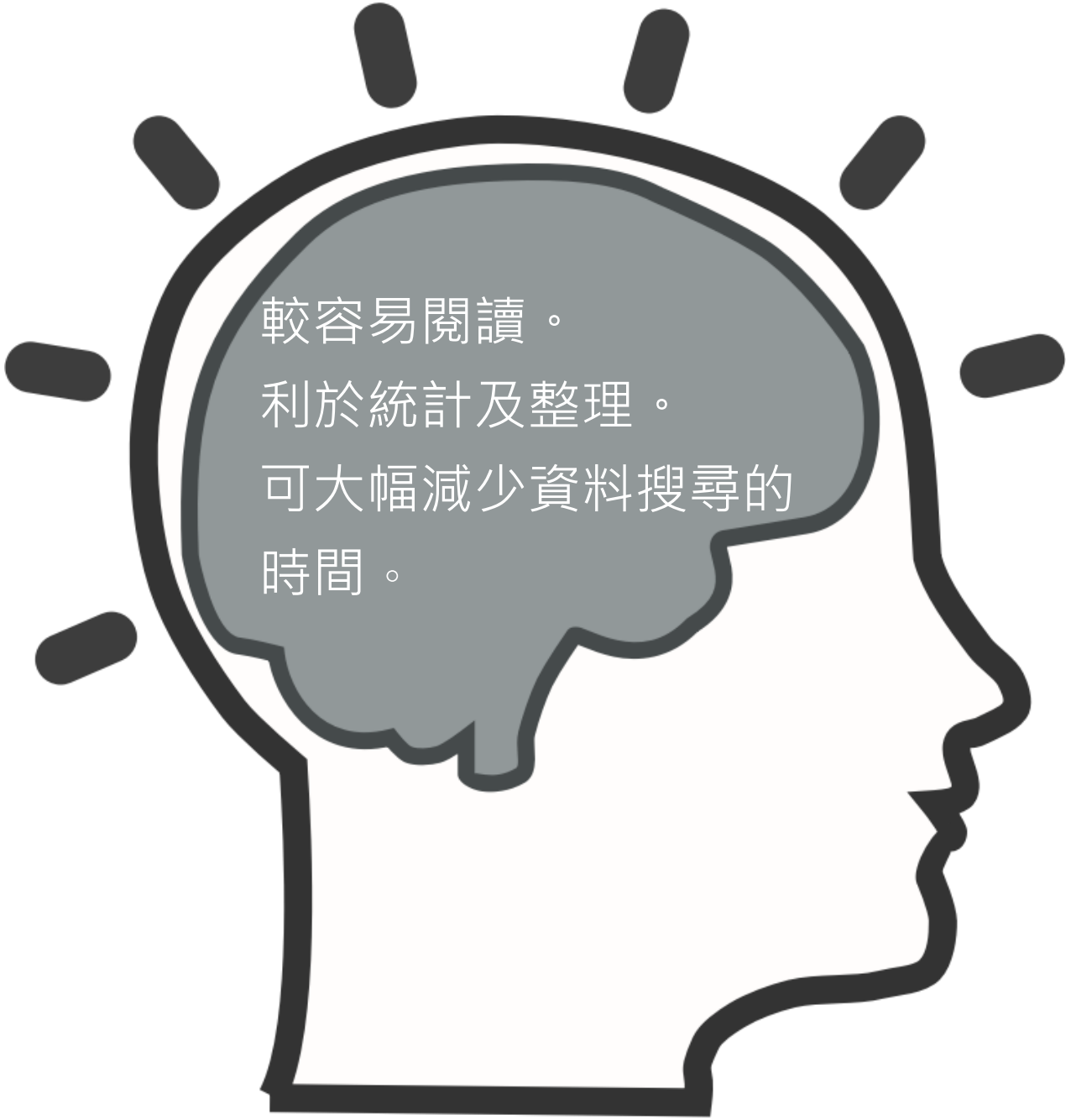
# Python

排序

# 排序的目的



5個整數，
由小到大排列。

**問題**

# 先想想簡單的

只有兩個數字時，就比較這兩個:

•前面數字比較大就跟後面的交換
•前面數字比較小就維持不動


**原始資料**

- [22, 7]

**第 1 回合**

- 比較第 1 個跟第 2 個數字
- 需要交換: [22, 7] => [7, 22]

**答案**

- [7, 22]

# 複雜一點點呢?

若有三個數字時，維持兩個數字排序的原則，兩個數字比較，前面數字比較大就跟後面的交換，當所有兩個數字的組合都比較過後，結果自然就是排序後的組合


**原始資料**
- [66, 22, 7]

**第 1 回合**
- 比較第 1 個跟第 2 個數字
- 需要交換: [66, 22, 7] => [22, 66, 7]

**第 2 回合**
- 比較第 2 個跟第 3 個數字
- 需要交換: [22 66, 7] => [22, 7, 66]

**第 3 回合**
- 第2個數字在第2回合有被移動過，重新比較第 1 跟第 2 個數字
- 需要交換: [22, 7, 66] => [7, 22, 66]

**第 4 回合**
- 第2個數字在第3回合可能有移動過，重新比較第 2 跟第 3 個數字
- 需要交換: [7, 22, 66] => [7, 22, 66]

**結果**
- [7, 22, 66]

# 我們知道

- 排序的中心原則就是由小到大（或由大到小）
- **兩兩數字**比較，前面比較大就跟後面數字交換
- 只要確定所有兩兩數字的配對組合都有比較過，就可得到想要的排序順序

# 問題是

兩個數字
=> 第 1 個跟第 2 個數字比

三個數字
=>第 1 個跟第 2 個數字比
=>第 2 個跟第 3 個數字比
=>第 1 個跟第 3 個數字比

一堆數字
=> 要怎麼組合阿？？？

# 換個方式想

- 兩個數字相比，大的數字會被換到後面位置
- N個數字時，從第 1 個位置開始：
  - 第 1 回合：第 1 個位置的數字跟第 2 個位置的數字比，第 1 個位置的數字比較大就跟第 2 個位置的數字交換位置
  - 第 2 回合：第 2 個位置的數字跟第 3 個位置的數字比，第 2 個位置的數字比較大就跟第 3 個位置的數字交換位置
  - 第 3 回合：第 3 個位置的數字跟第 4 個位置的數字比，第 3 個位置的數字比較大就跟第 4 個位置的數字交換位置
  - 第 4 回合、第 5 回合、第 6 回合....
  - 第 N-1 回合：第 N-1 個位置的數字跟第 N 個位置的數字比，第 N-1 個位置的數字比較大就跟第 N 個位置的數字交換位置

# 也就是說

- 經過第 1 次的 N-1 回合，第 1 大的數字會被推到最後 1 個位置
- 經過第 2 次的 N-1 回合，第 2大的數字會被推到倒數第 2 個位置
- 經過第 3 次的 N-1 回合，第 3 大的數字會被推到倒數第 3 個位置
- .....
- .....
- .....
- 經過第 N 次的 N-1 回合，第 N 大的數字會被推到倒數第 N 個位置

  ⇒ 也就是最小的數字會被放到最前面的方式

  ⇒ 所以，最後就產生由小到大排列的數字組合了！！！

# 就這樣比...

| 原始資料 | 第一回合 | | | | 第二回合 | | | | 第三回合 | | | | 第四回合 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 繼續往下比... | | |
| 3 | 4 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 | 2 | | | |
| 2 | 2 | 4 | 4 | 4 | 4 | 4 | 1 | 1 | 1 | 3 | 3 | 3 | 3 | | | |
| 5 | 5 | 5 | 5 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | | | |
| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | | | |

方法:
比4大回合，
每一回合又比4小回合
兩兩相比，大的往後推

# 將方法變成程式碼

```
intList = []
numCnt = 5

for i in range (0, numCnt):
    intList.append(int(input()))

for i in range(0, numCnt): # 比四大回合
    for j in range(0, numCnt-1): # 比四小回合
        f intList[j] > intList[j+1]:
          tmp = intList[j]
          intList[j] = intList[j+1]
          intList[j+1] = tmp

print(intList)
```

輸入資料

前一個比後一個大，
就做交換

輸出資料

第一大回合

第二大回合

第三大回合

第四大回合

第五大回合

```
排序前:          [4, 3, 2, 5, 1]
i: 0, j: 0       [3, 4, 2, 5, 1]
i: 0, j: 1       [3, 2, 4, 5, 1]
i: 0, j: 2       [3, 2, 4, 5, 1]
i: 0, j: 3       [3, 2, 4, 1, 5]
i: 1, j: 0       [2, 3, 4, 1, 5]
i: 1, j: 1       [2, 3, 4, 1, 5]
i: 1, j: 2       [2, 3, 1, 4, 5]
i: 1, j: 3       [2, 3, 1, 4, 5]
i: 2, j: 0       [2, 3, 1, 4, 5]
i: 2, j: 1       [2, 1, 3, 4, 5]
i: 2, j: 2       [2, 1, 3, 4, 5]
i: 2, j: 3       [2, 1, 3, 4, 5]
i: 3, j: 0       [1, 2, 3, 4, 5]
i: 3, j: 1       [1, 2, 3, 4, 5]
i: 3, j: 2       [1, 2, 3, 4, 5]
i: 3, j: 3       [1, 2, 3, 4, 5]
i: 4, j: 0       [1, 2, 3, 4, 5]
i: 4, j: 1       [1, 2, 3, 4, 5]
i: 4, j: 2       [1, 2, 3, 4, 5]
i: 4, j: 3       [1, 2, 3, 4, 5]
排序後:          [1, 2, 3, 4, 5]
```


第一大回合結束後，
**最大的值**會被推到**最下面**

第二大回合結束後，
**次大的值**會被推到**倒數第二層**

第三大回合結束後，
**第三大的值**會被推到**倒數第三層**

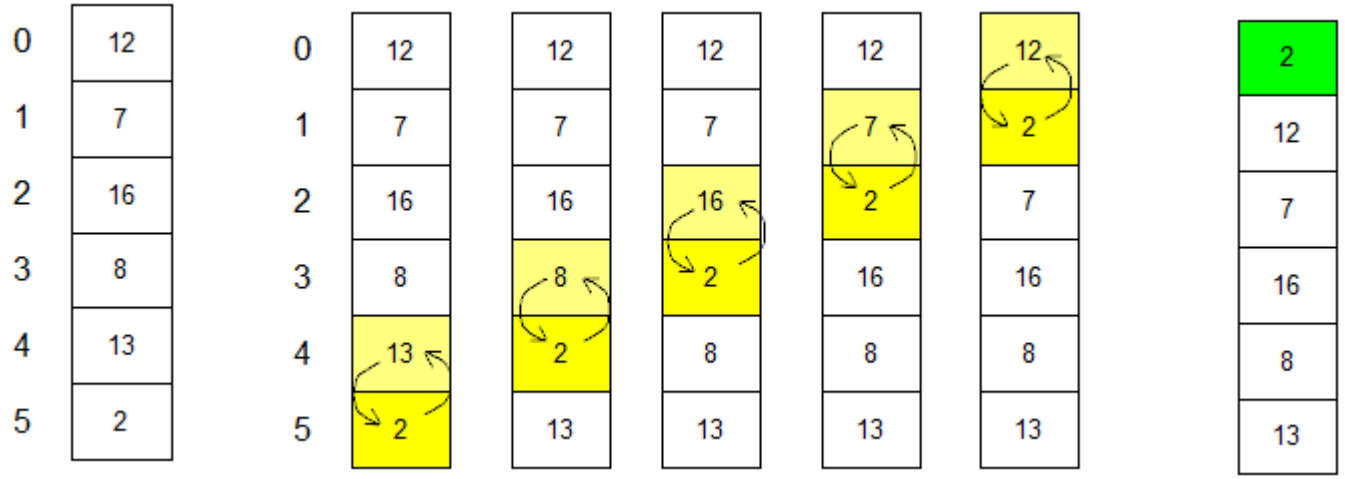
第四大回合結束後，
**第四大的值**會被推到**倒數第四層**

第五大回合結束後，
**第五大（最小）的值**會被推到**倒數第五層（最上層）**

# 這是泡沫排序法

- 重複地走訪要排序的數列，一次比較兩個元素，如果他們的順序錯誤就把他們交換過來。
- 走訪數列的工作是重複地進行直到沒有再需要交換，也就是說該數列已經排序完成。
- 越小的元素會經由交換慢慢「浮」到數列的頂端。
- 越大的元素會經由交換慢慢「沉」到數列的底端。

CS50說氣泡



還有其他方法嗎?

問題

# 也可以這樣比

**原始資料**
- [4, 3, 2, 5, 1]

**Step 1**
- 找出最小數字所在的位置 => 第 5 個位置
- 最小數字所在的位置 (第 5 個位置)跟第 1 個位置的數字交換
- 需要交換: [4, 3, 2, 5, 1] => [1, 3, 2, 5, 4]

**Step 2**
- 找出第 2 小數字所在的位置 => 第 3 個位置
- 第 2 小數字所在的位置(第 3 個位置) 跟第 2 個位置的數字交換
- 需要交換: [1, 3, 2, 5, 4] => [1, 2, 3, 5, 4]

**Step 3**
- 找出第 3 小數字所在的位置 => 第 3 個位置
- 第 3 小數字已經在第 3 個位置
- 不需要交換: [1, 2, 3, 5, 4]

**Step 4**
- 找出第 4 小數字所在的位置 => 第 5 個位置
- 第 4 小數字所在的位置(第 5 個位置) 跟第 4 個位置的數字交換
- 需要交換: [1, 2, 3, 5, 4] => [1, 2, 3, 4, 5]

**Step 5**
- 找出第 5 小數字所在的位置 => 第 5 個位置
- 第 5 小數字已經在第 5 個位置
- 不需要交換: [1,2, 3, 4, 5]

**答案**
- [1, 2, 3, 4, 5]

方法:
在每一回合中找出最小值

# 將方法變成程式碼

歡迎參考 edX 上開授的 CS 50 課程示範影片:

```
intList = []
numCnt = 5
for i in range (0, numCnt):
  intList.append(int(input()))
print(intList)

for i in range(0, numCnt):
  minIndex = i
  minNum = intList[i]
  for j in range(i+1, numCnt):
    if intList[j] < minNum:
      minIndex = j
      minNum = intList[j]
  if i != minIndex:
    intList[minIndex] = intList[i]
    intList[i] = minNum
  print('i: {i}, 數字第 {index} 小的位置: {minIndex}, 第 {index} 小的數字值: {minNum}, 列表: {intList}'.format(i=i, index=i+1, minIndex=minIndex,
minNum=minNum, intList=intList))

print(intList)
```


執行結果

```
[4, 3, 2, 5, 1]
i: 0, 數字第 1 小的位置: 4, 第 1 小的數字值: 1, 列表: [1, 3, 2, 5, 4]
i: 1, 數字第 2 小的位置: 2, 第 2 小的數字值: 2, 列表: [1, 2, 3, 5, 4]
i: 2, 數字第 3 小的位置: 2, 第 3 小的數字值: 3, 列表: [1, 2, 3, 5, 4]
i: 3, 數字第 4 小的位置: 4, 第 4 小的數字值: 4, 列表: [1, 2, 3, 4, 5]
i: 4, 數字第 5 小的位置: 4, 第 5 小的數字值: 5, 列表: [1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
```

# 這是選擇排序法

- 在所有的資料中：
  - 當由大至小排序，則將最大值放入第一位置；
  - 若由小至大排序時，則將最小值放入第一位置。
- 例如當N筆資料需要由大至小排序時，先假設第一個位置的資料是最大值，利用一個變數記錄最大值的位置(即假設的第一個位置)，依次向2、3、4 ...N個位置的資料作比較。
- 如果資料大於或等於其中一個位置，則記錄最大值位置的變數值不變；若小於其中一個位置，則記錄最大值位置的變數值改為有更大值數字的位置值。

# 排序，還有很多種方法呢!

**內部排序：**

排序的資料量小，可以完全在記憶體內進行排序。

- 氣泡排序法 (bubble sort)
- 選擇排序法 (selection sort)
- 插入排序法 (insertion sort)
- 快速排序法 (quick sort)
- 堆積排序法 (heap sort)
- 謝爾排序法 (shell sort)
- 基數排序法 (radix sort)

**外部排序：**

排序的資料量無法直接在記憶體內進行排序，而必須使用到輔助記憶體

- 直接合併排序法(Direct Merge Sort)
- k路合併法(k-Way Merge)
- 多相合併法(Polyphase Merge)