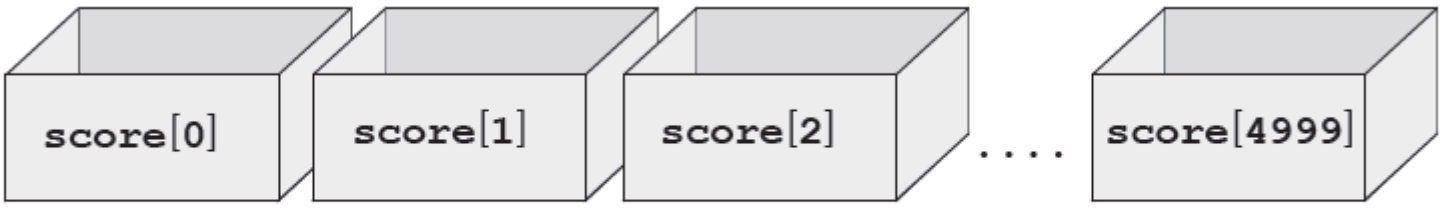
PYTHON
CONTAINER: LIST, TUPLE, DICTIONARY, AND SET

Chih-Chung Hsu (許志仲)
Institute of Data Science
National Cheng Kung University
https://cchsu.info

串列的使用

- 何謂串列 (List)
  - 串列 (又稱為「清單」或「列表」)，與其他語言的「陣列 (Array)」相同，其功能與變數相類似，是提供儲存資料的記憶體空間。



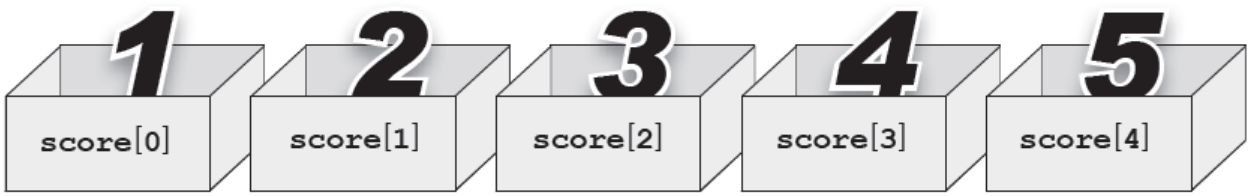
▲ 串列元素配置

標準套路

- 串列宣告
  - 一維串列宣告
    - 一維串列的宣告方式是將元素置於中括號 ([]) 中，每個元素之間以逗號分隔，語法為：

```
串列名稱 = [元素1, 元素2,……]
```

    - 例如：宣告 score 串列，其元素內容為 [1, 2, 3, 4, 5]。
    - 串列中各個元素資料型態可以相同，也可以不同，例如：




```
list1 = [1, 2, 3, 4, 5]                  # 元素皆為整數
list2 = ["香蕉", "蘋果", "橘子"]          # 元素皆為字串
list3 = [1, "香蕉", True]                 # 包含不同資料型態元素
```

串列宣告

- 空串列
  - 例如：

```
list4=[]
```

- 多維串列宣告
  - 例如下面是二維串列的範例，其串列元素是帳號、密碼組成的串列：

```
list5=[["joe","1234"],["mary","abcd"], ["david","5678"]]
print(list5[1])     #["mary","abcd"]
print(list5[1][1])   #abcd
```

串列宣告

- 串列元素的存取
  - 讀取串列元素
    - 讀取串列元素的語法為：
    - 例如：

```
串列名稱 [ 索引 ]
```

    - 取得前面 list1 串列之中索引為 0 (第 1 個元素) 的元素內容，得到結果為 1。

```
list1 = [1, 2, 3, 4, 5]
print(list1[0])   #1
```

串列宣告

- 注意索引值是從 0 開始計數：第一個元素索引值為 0，第二個元素索引值為 1，依此類推。索引值不可超出串列的範圍，否則執行時會產生「list index out of range」錯誤。例如：

```
list4 = ["香蕉", "蘋果", "橘子"]
print(list4[1])   # 蘋果
print(list4[2])   # 橘子
print(list4[3])   # 錯誤，索引值超過範圍
```

- 索引值可以是負值， 表示由串列的最後向前取出，「-1」表示最後一個元素，「-2」表示倒數第二個元素， 依此類推。同理， 負數索引值不可超出串列的範圍，否則執行時會產生錯誤。例如：

```
list4 = ["香蕉", "蘋果", "橘子"]
print(list4[-1])   # 橘子
print(list4[-3])   # 香蕉
print(list4[-4])   # 錯誤，索引值超過範圍
```

串列資料存取

- 改變串列元素
  - 語法為：

```
串列名稱 [ 索引 ]= 元素內容
```

  - 如：將 list1 串列中索引為 0 (第 1 個元素) 的元素內容，由 1 改變為 9。

```
list1 = [1, 2, 3, 4, 5]
print(list1[0])   # 1
list1[0]=9        # 更改為 9
print(list1[0])   # 9
```

連續串列存取

- 使用 for ... 迴圈讀取串列
  - 使用 for 變數 in 串列讀取串列
    - 使用 for 迴圈可以讀取串列的元素，它相當於其他語言的 for ~ each，其基本語法結構為：

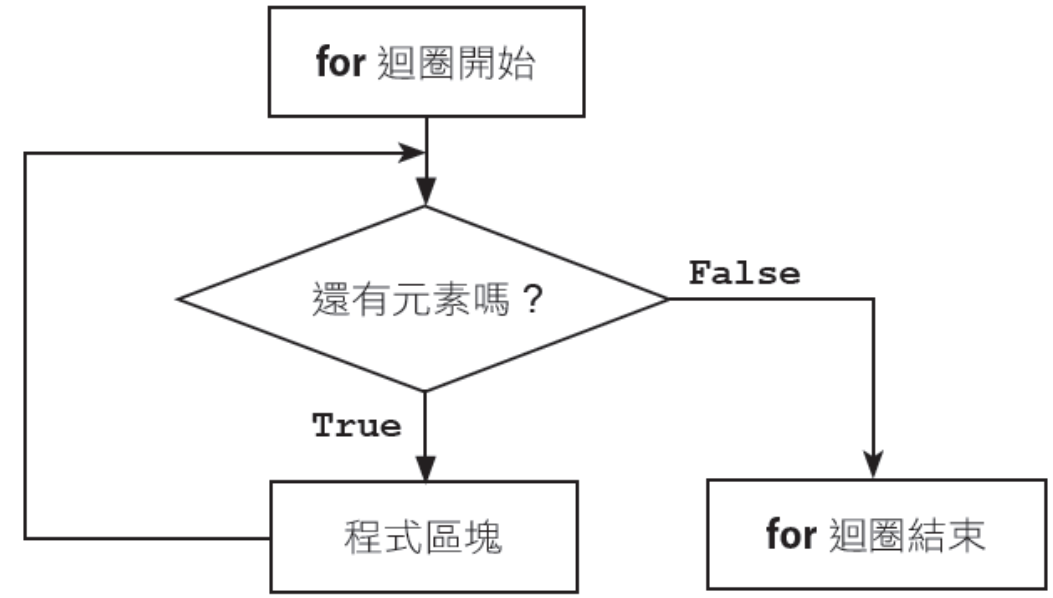
```
for 變數 in 串列:
    程式區塊
```

    - 以實例解說：

```
1 list1 = ["香蕉", "蘋果", "橘子"]
2 for s in list1:
3     print(s, end=",")   #執行結果為：香蕉,蘋果,橘子,
```

連續串列存取

- for迴圈的流程如下：



如果想要取得長度，並利用索引?

- 使用 for ... range 迴圈讀取串列
  - 取得串列長度
    - 迴圈中 range() 函式的範圍通常會利用 len() 函式計算串列的長度。例如：計算 scores 串列的長度，顯示結果為 3。

```
scores = [85, 79, 93]
print(len(scores)) # 3
```

  - 以 for in range 迴圈讀取串列

```
scores = [85, 79, 93]
for i in range(len(scores)):
    print(scores[i])
```

如果想要取得長度，並利用索引?

- enumerate function
  - 範例用法
    - For index, item in enumerate(your_list):

```
for  index,  sc  in  enumerate(scores):
```

    - 記得這邊的 index 是對應到某個list裡面的索引值，並且 item是對應到某個 list裡面的第 index 筆資料

串列搜尋與計次

- index() 搜尋
  - 語法：

```
索引值 = 串列名稱 .index( 串列元素 )
```

  - 例如：

```
list1 = ["香蕉","蘋果","橘子"]
n = list1.index("蘋果")   #n=1
m = list1.index("梨子") #ValueError: '梨子' is not in list
```

List內建功能：計次

- count() 計算次數
  - 語法：

```
次數 = 串列名稱 .count( 串列元素 )
```

  - 例如：

```
list1 = ["香蕉","蘋果","橘子"]
n = list1.count("橘子")   #n=1
m = list1.count("梨子")   #m=0
```

串列元素新增和刪除

- 增加串列元素
  - append() 方法
    - 語法：

```
串列名稱 .append( 元素值 )
```

    - 例如：在 list1 串列最後面增加一個串列元素「金榜」。

```
list1 = [1,2,3,4,5,6]
list1.append("金榜")   #list1=[1,2,3,4,5,6,'金榜']
print(list1[6])    # 金榜
print(len(list1))  #7
```

串列元素新增和刪除

- insert() 方法
  - 語法：

```
串列名稱 .insert( 索引值 , 串列元素 )
```

  - 例如：在 list1 串列索引 3 的位置插入一個串列元素「紅榜」。

```
list1 = [1,2,3,4,5,6]
list1.insert(3," 紅榜 ")   #list1=[1,2,3," 紅榜 ",4,5,6]
print(list1[3])            # 紅榜
print(len(list1))          #7
```

串列元素新增和刪除

- 如果索引值大於或等於串列元素個數，將如同 append() 方法一樣將串列元素加在最後面。
- 索引值也可以是負值，表示由串列的最後向前推算，「-1」表示最後一個元素，「-2」表示倒數第二個元素，依此類推。
- 例如：在 list1 串列索引第 -1、12 的位置插入串列元素。

```
list1 = [1,2,3,4,5,6]
list1.insert(-1, "愛")   #list1=[1, 2, 3, 4, 5, '愛', 6]
list1.insert(12, "台灣") #list1=[1, 2, 3, 4, 5, '愛', 6, '台灣']
print(list1)             #[1, 2, 3, 4, 5, '愛', 6, '台灣']
print(len(list1))        #8
```

刪除串列元素

- 刪除串列元素
  - remove() 方法
    - 語法：

```
串列名稱 .remove( 串列元素 )
```

    - 例如：刪除 list1 串列中「夏天」的串列元素。

```
list1 = ["春天","夏天","秋天","冬天"]
list1.remove("夏天")
print(list1)   #['春天', '秋天', '冬天']
```

  - pop() 方法
    - 語法：

```
串列名稱 .pop([index])
```

    - 例如：

```
list1 = [1,2,3,4,5,6]
n = list1.pop()   #n=6, list1=[1,2,3,4,5]
n = list1.pop(2)  #n=3, list1=[1,2,4,5]
```

刪除串列元素

- del 刪除串列元素
  - del 刪除串列單一元素語法：

```
del 串列名稱(n1)
```

  - del 刪除串列指定範圍元素的語法：

```
del 串列名稱(n1:n2[:n3])
```

  - 例如：

```
list1 = [1,2,3,4,5,6]
del list1[1]
print(list1) #[1,3,4,5,6]
list2=[1,2,3,4,5,6]
del list2[1:5:2]   # 刪除索引第 1、3 的串列元素
print(list2) #[1,3,5,6]
```

串列排序

- sort() 由小到大排序
  - 語法：

```
串列名稱 .sort()
```

  - 例如：將 list1 串列由小到大排序。

```
list1=[3,2,1,5] #[3, 2, 1, 5]
list1.sort()
print(list1) #[1, 2, 3, 5]
```

串列排序

- reverse() 反轉串列順序
  - 語法

```
串列名稱 .reverse()
```

  - 例如：將 list1 串列順序反轉。

```
list1=[3,2,1,5] #[3, 2, 1, 5]
list1.reverse()
print(list1) #[5, 1, 2, 3]
```

串列排序

- 由大到小排序
  - 語法：

```
串列名稱 .sort()
串列名稱 .reverse()
```

  - 例如：將 list1 串列由大到小排序。(<sort1.py>)

```
list1=[3,2,1,5] #[3, 2, 1, 5]
list1.sort()
print(list1) #[1, 2, 3, 5]
list1.reverse()
print(list1) #[5, 3, 2, 1]
```

串列排序

- sorted() 排序
  - 語法：

```
串列名稱 2=sorted( 串列名稱 1,reverse=True)
```

  - 例如：將 list1 串列由大到小排序，並儲存在 list2 串列。

```
list1=[3,2,1,5]  #[3,  2,  1,  5]
list2=sorted(list1,reverse=True)
print(list2)      #[5,  3,  2,  1]
print(list1)      #[3,  2,  1,  5]   #原串列不變
```

串列常用方法列表

- 下表為串列的常用方法：(表中 list1=[1,2,3,4,5,6])

方法	意義	範例	範例結果
list1[n1:n2]	取出 n1 到 n2-1 元素。	list2=list1[1:4]	list2=[2,3,4]
list1[n1:n2:n3]	同上，取出間隔為 n3。	list2=list1[1:4:2]	list2=[2,4]
del list1[n1:n2]	刪除 n1 到 n2-1 元素。	del list1[1:4]	list1=[1,5,6]
del list1[n1:n2:n3]	同上，刪除間隔為 n3。	del list1[1:4:2]	list1=[1,3,5,6]
n=len(list1)	取得串列元素數目。	n=len(list1)	n=6
n=min(list1)	取得元素最小值。	n=min(list1)	n=1
n=max(list1)	取得元素最大值。	n=max(list1)	n=6
n=list1.index(n1)	第 1 次 n1 元素的索引值。	n=list1.index(3)	n=2
n=list1.count(n1)	n1 元素出現的次數。	n=list1.count(3)	n=1

串列常用方法列表

方法	意義	範例	範例結果
list1.append(n1)	將 n1 做為元素加在串列最後。	list1.append(8)	list1=[1,2,3,4,5,6,8]
list1.insert(n,n1)	在位置 n 加入 n1 元素。	list1.insert(3,8)	list1=[1,2,3,8,4,5,6]
n=list1.pop()	取出最後 1 個元素並由串列中移除元素。	n=list1.pop()	n=6, list1=[1,2,3,4,5]
list1.remove(n1)	移除第 1 次的 n1 元素。	list1.remove(3)	list1=[1,2,4,5,6]
list1.reverse()	反轉串列順序	list1.reverse()	list1=[6,5,4,3,2,1]
list1.sort()	將串列由小到大排序。	list1.sort()	list1=[1,2,3,4,5,6]

多個 List 的處理

- 用運算子 + 可用於兩個串列的 “串接”
  - 同樣的用在字串上也是兩個字串的 “串接”

```
shoplist1 = [' 牛奶 ', ' 蛋 ', ' 咖啡豆 ']
shoplist2 = [' 西瓜 ', ' 鳳梨 ']
shoplist_all = shoplist1 + shoplist2
print(shoplist_all)
```

```
[' 牛奶 ', ' 蛋 ', ' 咖啡豆 ', ' 西瓜 ', ' 鳳梨 ']
```

  - 使用 [:] 與函式copy 拷貝串列，會將串列複製一份與原來串列不同，是兩個不同的物件，佔有不同的記憶體空間，而使用等號= 指向同一個位置

元組 (Tuple)

- 建立元組
  - 語法為：

```
元組名稱 = (元素1, 元素2, ……)
```

  - 例如：

```
tuple1 = (1, 2, 3, 4, 5)    # 元素皆為整數
tuple2 = (1, "香蕉", True)  # 包含不同資料型態元素
```

  - 元組的使用方式與串列相同，但不能修改元素值，否則會產生錯誤，例如：

```
tuple3 = ("香蕉", "蘋果", "橘子")
print(tuple3[1])     # 蘋果
tuple3[1] = "芭樂"   # 錯誤，元素值不能修改
```

串列和元組互相轉換

- 串列和元組互相轉換
  - Python 提供 list 命令將元組轉換為串列，tuple 命令將串列轉換為元組。
  - 元組轉換為串列的範例實作：

```
tuple1 = (1,2,3,4,5)
list1 = list(tuple1)    # 元組轉換為串列
list1.append(8)         # 正確，在串列中新增元素
```

  - 串列轉換為元組的範例實作：

```
list2 = [1,2,3,4,5]
tuple2 = tuple(list2)    # 串列轉換為元組
tuple2.append(8)         # 錯誤，元組不能增加元素
```

重點整理

- 在本章中介紹了下面的重點：
  - 可以把串列想成是有許多相同名稱的箱子，連續排列在一起，這些箱子可以儲存資料，而每個箱子有不同編號，只要指定編號即可存取對應箱子內的資料。
  - 一維串列的宣告的語法：

```
串列名稱 = [元素1, 元素2,……]
```

  - 存取串列元素的語法為：

```
串列名稱[索引]
```

  - 索引值是從 0 開始計數：第一個元素值索引值為 0，第二個元素值索引值為 1，依此類推。索引值不可超出串列的範圍，否則執行時會產生「list index out of range」錯誤。

- for 迴圈讀取串列的方法有下列兩種。
  - for 變數 in 串列:
  - for 變數 in range():
- index() 方法可以搜尋指定串列元素的索引值，count() 方法可以計算指定串列元素出現的次數。
- append() 方法是將元素加在串列最後面，insert() 方法是將元素插入在串列中指定索引位置。
- remove() 方法是刪除串列中第一個指定的串列元素，pop() 方法的功能是由串列中取出元素，同時串列會將該元素移除。
- del 可以刪除變數、串列、也可以刪除串列元素。
- sort() 方法將指定串列由小到大排序，reverse() 方法將指定串列順序反轉。
- sorted() 方法將指定的串列排序，原來的串列不會被改變。
- 元組的結構與串列完全相同，不同處在於元組的元素個數及元素值皆不能改變，而串列則可以改變，所以一般將元組說成是「不能修改的串列」。