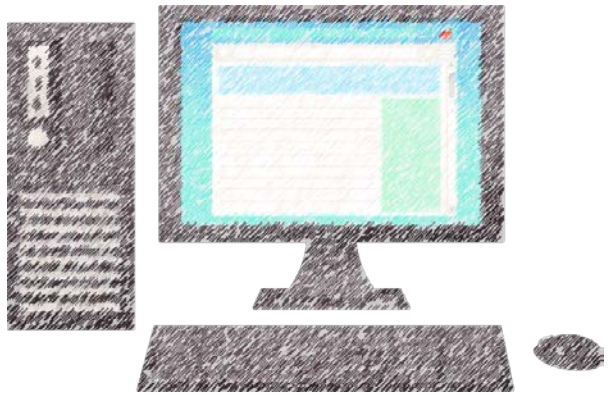
桃園市政府勞動局 112 年勞工學苑產業應用班
Python 程式設計：從零基礎入門到進階

第 4 單元

基本資料結構



林柏江老師
元智大學 電機工程學系 助理教授
pclin@saturn.yzu.edu.tw

預計課程進度

週次	日期	上午課程內容 (09:00 ~ 12:00)	下午課程內容 (13:00 ~ 16:00)
1	2023/07/23	01. 運算思維簡介	02. Python 快速上手
2	2023/07/30	03. Python 基礎	04. Python 基本資料結構
3	2023/08/06	05. 字串	06. 字典
4	2023/08/13	07. 流程控制	08. 函式
5	2023/08/20	09. 模組與作用域	10. Python 程式檔
6	2023/08/27	11. 檔案系統的使用與檔案讀寫	12. 例外處理
7	2023/09/03	13. 類別與物件導向程式設計	14.初探資料結構與演算法
8	2023/09/10	15. 陣列	16. 鏈結串列
9	2023/09/17	17. 堆疊與佇列	18. 圖形結構
	2023/09/24, 2023/10/01, 2023/10/08 (共三周) 放假		
10	2023/10/15	19. 樹狀結構	20. 分治法
11	2023/10/22	21. 動態規劃	22. 貪婪演算法
12	2023/10/29	23. 回溯	24. 分支定界法

課程大綱

1. list (串列)
2. list 的索引與切片
3. list 常用的方法與操作
4. list 的排序
5. 其他常見的 list 操作
6. 多層 list 和深層拷貝 (deepcopy)
7. tuple
8. set (集合)

課程大綱

1. list (串列)
2. list 的索引與切片
3. list 常用的方法與操作
4. list 的排序
5. 其他常見的 list 操作
6. 多層 list 和深層拷貝 (deepcopy)
7. tuple
8. set (集合)

list (串列)

- Python 的 list 有點類似其他程式語言的陣列 (array)，但是 list 比 array 更靈活、功能更強大。
- list 是由 [ ] 把元素括起來，元素之間以逗號隔開。
- 不需要事先宣告 list 元素的型別或是 list 的大小，會根據需要自動調整。

```
>>> x = [1, 2, 3]
>>> x
[1, 2, 3]
```

Python 的 list 可以包含不同型別的元素

list 的元素可以是任何 Python 物件。

```
>>> x = [2, "two", [1, 2, 3]]
>>> x
[2, 'two', [1, 2, 3]]
```

使用 len() 取得 list 中的元素數量

len() 只計算 list 第一層的元素數量。

```
>>> x = [2, "two", [1, 2, 3]]
>>> len(x)
3
```

課程大綱

1. list (串列)
2. list 的索引與切片
3. list 常用的方法與操作
4. list 的排序
5. 其他常見的 list 操作
6. 多層 list 和深層拷貝 (deepcopy)
7. tuple
8. set (集合)

list 的索引 (index)

- 從 Python 的 list 中取得元素值的方式與 C 語言類似，都是使用索引 [n] 來取值。
- Python 的索引是從 0 開始計數。
- 索引值 0 回傳 list 的第 0 個元素。
- 索引值 1 回傳 list 的第 1 個元素。
- …

```
>>> x = ["first", "second", "third", "fourth"]
>>> x[0]
'first'
>>> x[2]
'third'
```

負數的索引值

- Python 的索引值如果是負數，表示從 list 尾端開始計數。
- -1 是 list 中的最後一個位置。
- -2 是 list 中的倒數第二個位置。
- ...

```
>>> x = ["first", "second", "third", "fourth"]
>>> a = x[-1]
>>> a
'fourth'
>>> x[-2]
'third'
```

list 的切片 (slicing)

- Python 可以一次指定 list 中的多個元素，稱作切片。
- `[index1:index2]` 用來指定 list 中 `index1` 到 `index2` 之間 (不包括 `index2`) 的元素。

```
>>> x = ["first", "second", "third", "fourth"]
>>> x[0:3]
['first', 'second', 'third']
>>> x[1:3]
['second', 'third']
>>> x[1:-1]
['second', 'third']
>>> x[-2:-1]
['third']
```

list 的切片 (slicing) (續)

在 list 的切片中，如果第二個索引值的位置在第一個索引值的前面，會回傳一個空的 list。

```
>>> x = ["first", "second", "third", "fourth"]
>>> x[-1:2]
[]
```

list 的切片 (slicing) (續)

- 對 list 切片時，可以省略 `index1` 或 `index2`。
- 省略 `index1` 表示「從 list 的開頭」。
- 省略 `index2` 表示「到 list 的最尾端」。

```
>>> x = ["first", "second", "third", "fourth"]
>>> x[:3]
['first', 'second', 'third']
>>> x[2:]
['third', 'fourth']
```

list 的切片 (slicing) (續)

- 如果切片時省略 index1 以及 index2 兩個索引，會產生一個從原本 list 的開頭到結尾的新 list。也就是複製整個 list。
- 適用於想要在不影響原本 list 的情況下，產生一個副本來修改內容。

```
>>> y = x[:]
>>> y[0] = '1 st'
>>> y
['1 st', 'second', 'third', 'fourth']
>>> x
['first', 'second', 'third', 'fourth']
>>> y = x
>>> y[0] = '1 st'
>>> x
['1 st', 'second', 'third', 'fourth']
```

更改 list 的元素

list 索引不只是可以用來取得元素值，
也可以用來更改元素值。

```
>>> x = [1, 2, 3, 4]
>>> x[1] = "two"
>>> x
[1, 'two', 3, 4]
```

把位於索引 **1** 的元素值改為 **"two"**。

更改 list 的元素 (續)

可以使用切片來更改元素值。
a[m:n] = b 會導致 list a 的 m 與 n 之間 (不包括 n)
所有元素被 list b 中的元素替換。

```
>>> x = [0, 1, 2, 3]
>>> x[1:3] = ["one", "two"]
>>> x
[0, 'one', 'two', 3]
```

更改 list 的元素 (續)

使用 a[m:n] = b 替換元素時，b 的元素個數不一定要等於 m 到 n 之間的元素個數，可以更多或更少。這種情況下，a 的長度會被修改。

```
>>> x = [0, 1, 2, 3]
>>> x [1:3] = ["one", "two", "three"]
>>> x
[0, 'one', 'two', 'three', 3]
>>> x[0:4] = ["a", "b"]
>>> x
['a', 'b', 3]
```

更改 list 的元素 (續)

可利用上一頁的技巧，把多個值附加在 list 最後面、
插入到最前面、或是直接刪除多個元素。

```
>>> x = [1, 2, 3, 4]
>>> x[len(x):] = [5, 6, 7]
>>> x
[1, 2, 3, 4, 5, 6, 7]
>>> x[:0] = [-1, 0]
>>> x
[-1, 0, 1, 2, 3, 4, 5, 6, 7]
>>> x[1:-1] = []
>>> x
[-1, 7]
```

於 list 最後面附加多個值。

於 list 最前面插入多個值。

移除 list 中多個元素。

課程大綱

1. list (串列)
2. list 的索引與切片
3. list 常用的方法與操作
4. list 的排序
5. 其他常見的 list 操作
6. 多層 list 和深層拷貝 (deepcopy)
7. tuple
8. set (集合)

方法 (Method)

- 把物件 (object) 想像成實體的物品，則方法 (method) 就是操作該物品的動作。
- 以物件為主詞，方法為動詞。

寫文章	寫 Python 程式	
車子	`car`	← car 物件
車子向前進	`car.go()`	← car 物件的 go 方法

- 在後面的單元，我們會再來討論物件與方法。

附加 list 的元素：append()

append() 用來把單一個元素附加到 list 尾端。

```
>>> x = [1, 2, 3]
>>> x.append("four")
>>> x
[1, 2, 3, 'four']
```

若是用 append() 把一個 list 附加到另一個 list 尾端，會變成多層 list。

```
>>> x = [1, 2, 3, 4]
>>> y = [5, 6, 7]
>>> x.append(y)
>>> x
[1, 2, 3, 4, [5, 6, 7]]
```

附加 list 的元素：extend()

若想要把 list y 裡面所有元素附加到另一個 list x 的尾端，可使用 extend() 方法。

```
>>> x = [1, 2, 3, 4]
>>> y = [5, 6, 7]
>>> x.extend(y)
>>> x
[1, 2, 3, 4, 5, 6, 7]
```

插入 list 的元素：insert()

insert() 方法可在兩個元素之間或是 list 最前端插入新的元素。
insert() 方法有兩個參數：

1. 新元素應插入的索引位置
2. 新元素本身

```
>>> x = [1, 2, 3]
>>> x.insert(2, "hello")
>>> x
[1, 2, 'hello', 3]
>>> x.insert(0, "start")
>>> x
['start', 1, 2, 'hello', 3]
```

插入 list 的元素：insert() (續)

insert() 方法可以處理負的索引值。

```
>>> x = [1, 2, 3]
>>> x.insert(-1, "hello")
>>> print(x)
[1, 2, 'hello', 3]
```

插入 list 的元素：insert() (續)

以下左右兩段程式碼，執行結果是否相同？

```
>>> x = [1, 2, 3, 4, 5, 6]
>>> x.insert(3, 'hi')
>>> x
```

```
>>> x = [1, 2, 3, 4, 5, 6]
>>> x[3:3] = ['hi']
>>> x
```

左右這兩段程式碼，哪一種的可讀性比較高？

刪除元素：del

del 可用來刪除 list 元素或切片。

```
>>> x = ['a', 2, 'c', 7, 9, 11]
>>> del x[1]
>>> x
['a', 'c', 7, 9, 11]
>>> del x[:2]
>>> x
[7, 9, 11]
```

del 不是 list 的方法，而是 Python 的關鍵字指令。

刪除元素：del (續)

以下左右兩段程式碼，執行結果是否相同？

```
>>> x = [1, 2, 3, 4, 5, 6]
>>> del x[3]
>>> x
```

```
>>> x = [1, 2, 3, 4, 5, 6]
>>> x[3:4] = []
>>> x
```

左右這兩段程式碼，哪一種的可讀性比較高？

在 list 中找到指定值並刪除：remove()

remove() 方法會在 list 中找到第一個內容與指定值相同的元素，並從 list 中刪除該元素。

```
>>> x = [1, 2, 3, 4, 3, 5]
>>> x.remove(3)
>>> x
[1, 2, 4, 3, 5]
>>> x.remove(3)
>>> x
[1, 2, 4, 5]
>>> x.remove(3)
Traceback (innermost last):
  File "<stdin>", line 1, in ?
ValueError: list.remove(x): x not in list
```

remove() 無法找到要刪除的元素，所以引發錯誤。

反轉 list 的元素：reverse()

reverse() 方法用來把一個 list 的順序反轉。

```
>>> x = [1, 3, 5, 6, 7]
>>> x.reverse()
>>> x
[7, 6, 5, 3, 1]
```

課程大綱

1. list (串列)
2. list 的索引與切片
3. list 常用的方法與操作
4. list 的排序
5. 其他常見的 list 操作
6. 多層 list 和深層拷貝 (deepcopy)
7. tuple
8. set (集合)

list 的元素排序：sort() 方法

sort() 方法用來把一個 list 的元素排序。

```
>>> x = [3, 8, 4, 0, 2, 1]
>>> x.sort()
>>> x
[0, 1, 2, 3, 4, 8]
```

sort() 方法是原地排序 (in-place sort)，也就是會改變原本 list 的內容。
若不想改變原本 list 的內容，有兩種方法：

1. 使用 sorted() 函式 (稍後會介紹)。
2. 複製 list，再對副本再進行排序。

複製 list，再對副本再進行排序

```
>>> x = [2, 4, 1, 3]
>>> y = x[:]
>>> y.sort()
>>> y
[1, 2, 3, 4]
>>> x
[2, 4, 1, 3]
```

sort() 方法的排序規則

若 list 內的元素是字串，sort() 方法會依照字母順序進行排序。

- 字母大寫小於小寫。
- 'a' < 'z'。
- 先比較第一個字母，若相同，則比較第二個字母，依此類推。

```
>>> x = ["Life", "Is", "Enchanting"]
>>> x.sort()
>>> x
['Enchanting', 'Is', 'Life']
```

```
>>> x = ["Life", "Love", "Liberty"]
>>> x.sort()
>>> x
['Liberty', 'Life', 'Love']
```

sort() 方法的排序規則 (續)

使用 sort() 方法時，list 中的所有元素必須是可以互相比較的型別。
在包含數字和字串的 list 上使用 sort() 會引發錯誤。

```
>>> x = [1, 2, 'hello', 3]
>>> x.sort()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '<' not supported between
instances of 'str' and 'int'
```

sort() 方法的排序規則 (續)

若 list 的元素是 list，也可以進行排序。

```
>>> x = [[3, 5], [2, 9], [2, 3], [4, 1], [3, 2]]
>>> x.sort()
>>> x
[[2, 3], [2, 9], [3, 2], [3, 5], [4, 1]]
```

遞減排序

sort() 預設是遞增排序。
它有一個選用的 (optional) 參數 reverse，當 reverse = True 時，
會以遞減的順序進行排序。

```
>>> x = [0, 1, 2]
>>> x.sort(reverse=True)
>>> x
[2, 1, 0]
```

自定義排序

我們在後續的單元會詳細介紹函式。

list 的 sort() 方法使用內建的 Python 比較函式來排定順序。
也可使用 sort() 方法的參數 key，自行定義一個比較函式。

```
>>> def compare_num_of_chars(string1):
...     return len(string1)
>>> word_list = ['Python', 'is', 'better', 'than', 'C']
>>> word_list.sort()
>>> print(word_list)
['C', 'Python', 'better', 'is', 'than']
>>> word_list = ['Python', 'is', 'better', 'than', 'C']
>>> word_list.sort(key=compare_num_of_chars)
>>> print(word_list)
['C', 'is', 'than', 'Python', 'better']
```

預設是依照字母順序來排序。

使用自定義的比較函式，來讓 sort() 排序。

sorted() 函式

- list 的 sort() 方法會直接修改原本的 list (原地排序，in-place sort)。
- sorted() 函式不會修改原本的 list，而是回傳一個新的已排序 list。
- sorted() 函式一樣可以使用 key 以及 reverse 參數。

```
>>> x = [4, 2, 1, 3]
>>> y = sorted(x)
>>> y
[1, 2, 3, 4]
>>> z = sorted(x, reverse=True)
>>> z
[4, 3, 2, 1]
>>> x
[4, 2, 1, 3]
```

x 不會被修改。

list 的 sort() 方法
與 sorted() 函式比較

	list 的 sort() 方法	sorted() 函式
使用方式	list.sort()	sorted(list)
排序方式	直接修改原本 list。 (原地排序)	不修改原本 list，回傳一個新的已排序 list。
哪些物件可以用	只有 list 可以用 sort() 方法。	list、tuple、set、dictionary 等資料結構都可以用。

動手做

假設有一個 list，其中每一個元素也是一個 list：`[[1, 2, 3], [2, 1, 3], [4, 0, 1]]`

假如你想按照每個 list 元素中的第二個元素 (索引為 1) 的大小進行排序，得到的結果為：`[[4, 0, 1], [2, 1, 3], [1, 2, 3]]`

請寫出一個比較函式當作 key 參數，來傳給 sort() 方法。

```
>>> x = [[1, 2, 3], [2, 1, 3], [4, 0, 1]]
>>> def compare_second_element(my_list):
...     return 
...
>>> x.sort(key = compare_second_element)
>>> x
[[4, 0, 1], [2, 1, 3], [1, 2, 3]]
```

請在這裡寫出你的自訂比較函式。

動手做的參考答案

```
>>> x = [[1, 2, 3], [2, 1, 3], [4, 0, 1]]
>>> def compare_second_element(my_list):
...     return my_list[1]
...
>>> x.sort(key = compare_second_element)
>>> x
[[4, 0, 1], [2, 1, 3], [1, 2, 3]]
```

課程大綱

1. list (串列)
2. list 的索引與切片
3. list 常用的方法與操作
4. list 的排序
5. 其他常見的 list 操作
6. 多層 list 和深層拷貝 (deepcopy)
7. tuple
8. set (集合)

使用 in 算符判斷某個元素值是否在 list 中

in 算符可用來判斷某個元素值是否在 list 中，它會回傳一個布林值。
not in 算符則可用來判斷某個元素值是否不在 list 中。

```
>>> 3 in [1, 3, 4, 5]
True
>>> 3 not in [1, 3, 4, 5]
False
>>> 3 in ["one", "two", "three"]
False
>>> 3 not in ["one", "two", "three"]
True
```

使用 + 算符來串聯 list

+ 算符可把兩個 list 串聯起來，建立一個新的 list。
參與 + 運算的兩個 list 保持不變。

```
>>> z = [1, 2, 3] + [4, 5]
>>> z
[1, 2, 3, 4, 5]
```

```
>>> x = [1, 2, 3]
>>> y = [4, 5]
>>> z = x + y
>>> z
[1, 2, 3, 4, 5]
>>> x
[1, 2, 3]
>>> y
[4, 5]
```

使用 * 算符把 list 初始化

* 算符可用來建立一個特定大小的 list，並把該 list 初始化為特定值。

```
>>> z = [None] * 4
>>> z
[None, None, None, None]
```

使用 * 算符來複製 list 內容

可使用任何 list 搭配 * 算符來複製 list 內容，
並把所有副本連接起來，建立一個新的 list。

```
>>> z = [3, 1] * 2
>>> z
[3, 1, 3, 1]
```

使用 min() 以及 max() 函式找出 list 中最小以及最大的元素

Python 內建的 min() 以及 max() 函式可找出 list 中最小以及最大的元素。
若 list 中的元素無法互相比較，嘗試找出最小以及最大的元素將會引發錯誤。

```
>>> min([3, 7, 0, -2, 11])
-2
>>> max([4, "Hello", [1, 2]])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: '>' not supported between
instances of 'str' and 'int'
```

使用 index()方法 找出某個元素在 list 中的索引值

index() 方法會在 list 中搜尋指定的元素值，並回傳該元素在 list 中的索引值。
若元素值不存在 list 中，會引發錯誤。

```
>>> x = [1, 3, "five", 7, -2]
>>> x.index(7)
3
>>> x.index(5)
Traceback (innermost last):
 File "<stdin>", line 1, in ?
ValueError: 5 is not in list
```

可先用 in 算符判斷要找的元素值是否在 list 中。

使用 count() 方法
找出某個元素在 list 中出現的次數

count() 方法會在 list 中搜尋指定的元素值，並回傳該元素在 list 中出現的次數。

```
>>> x = [1, 2, 2, 3, 5, 2, 5]
>>> x.count(2)
3
>>> x.count(5)
2
>>> x.count(4)
0
```

list 操作總結

list 操作	說明	範例
`[]`	建立一個空 list	`x = []`
`len()`	回傳 list 長度	`len(x)`
`append()`	在 list 後面附加一個元素	`x.append('y')`
`extend()`	在 list 後面附加另一個 list 內所有值	`x.extend(['a', 'b'])`
`insert()`	在 list 指定位置之前插入一個元素	`x.insert(0, 'y')`
`del`	刪除 list 元素或切片	`del x[0]`
`remove()`	搜尋 list 並刪除指定值	`x.remove('y')`
`reverse()`	把 list 順序反轉	`x.reverse()`
`sort()`	對 list 原地排序	`x.sort()`
`sorted()`	把 list 排序後回傳一個新 list	`sorted(x)`

list 操作總結 (續)

list 操作	說明	範例
`+`	把兩個 list 串接成新 list	`x1 + x2`
`*`	複製 list	`x = ['y'] * 3`
`min()`	回傳 list 中最小的元素	`min(x)`
`max()`	回傳 list 中最大的元素	`max(x)`
`index()`	回傳某個值在 list 中的索引位置	`x.index('y')`
`count()`	計算 list 中出現某個值的次數	`x.count('y')`
`sum()`	求出 list 中所有 (可相加的) 項目總和	`sum(x)`
`in`	回傳某個元素值是否在 list 中	`'y' in x`

動手做

假設一個 list 中有 10 個元素，如何把最後三個元素從 list 尾端移動到開頭，並且保持它們原來的順序呢？

課程大綱

1. list (串列)
2. list 的索引與切片
3. list 常用的方法與操作
4. list 的排序
5. 其他常見的 list 操作
6. 多層 list 和深層拷貝 (deepcopy)
7. tuple
8. set (集合)

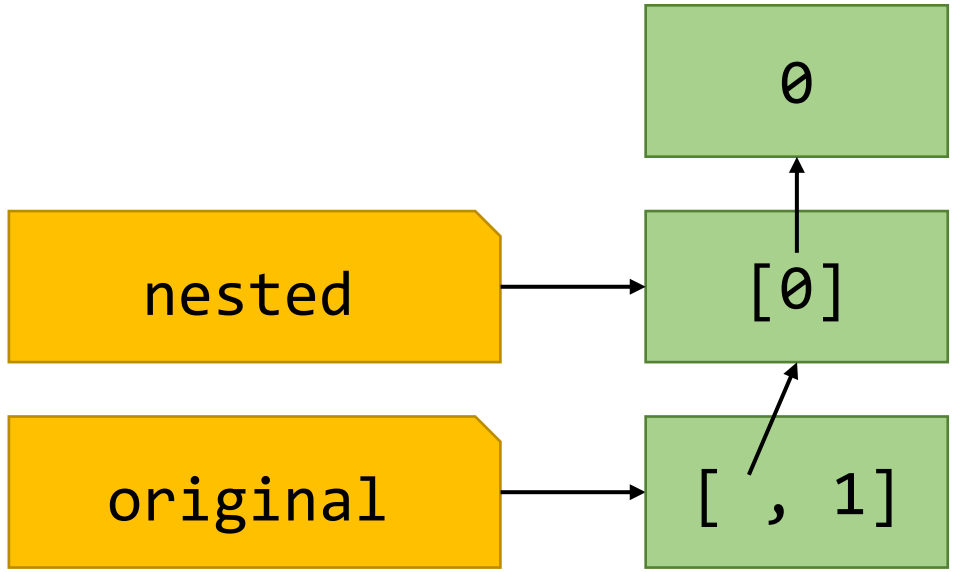
多層 list

多層 list (也稱作巢狀 list，nested list) 應用之一是用來表示數學中的二維矩陣。可使用二維索引來存取。
這種方式可以擴展到任意維度。

```
>>> m = [[0, 1, 2], [10, 11, 12], [20, 21, 22]]
>>> m[0]
[0, 1, 2]
>>> m[0][1]
1
>>> m[2]
[20, 21, 22]
>>> m[2][2]
22
```

多層 list 的物件參照

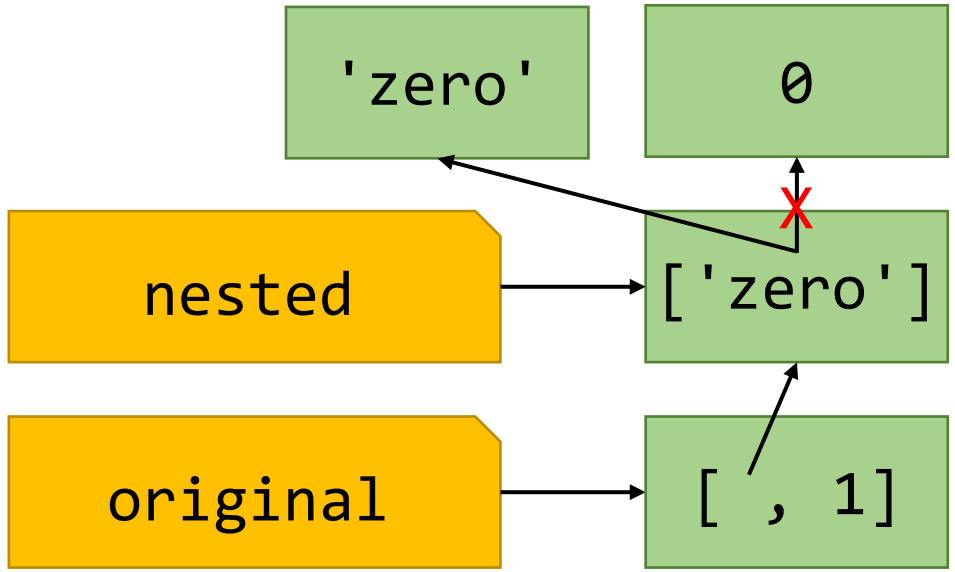
```
>>> nested = [0]
>>> original = [nested, 1]
>>> original
[[0], 1]
```



多層 list 的物件參照 (續)

使用 nested 變數或是 original 變數都可以改變第二層 list 中的值。

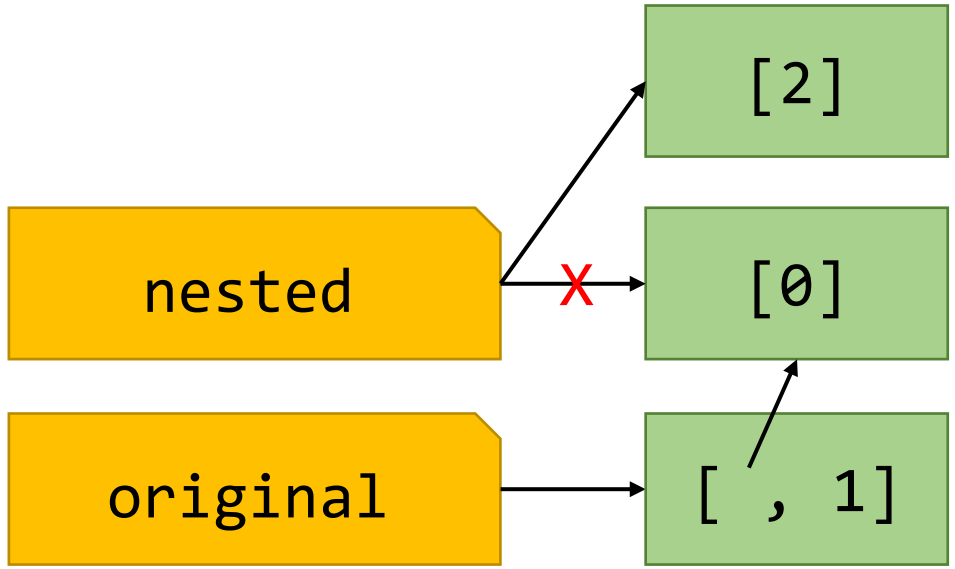
```
>>> nested[0] = 'zero'
>>> original
[['zero'], 1]
>>> original[0][0] = 0
>>> nested
[0]
>>> original
[[0], 1]
```



多層 list 的物件參照 (續)

如果把 nested 變數改設定為另一個 list，則它們之間的連結會被破壞。

```
>>> nested = [2]
>>> original
[[0], 1]
```



深層拷貝 deepcopy()

- 一般情況下，我們只需要 list 淺層拷貝 (shallow copy)，可使用以下三種運算其中一種：
  - 整個切片 (例如 x[:])
  - + 算符 (例如 x + [])
  - * 算符 (例如 x * 1)
- 若 list 中有第二層以上的 list 時，可能需要進行深層拷貝 (deep copy)。
  - 可使用 copy 模組中的 deepcopy 函式來進行深層拷貝。

深層拷貝 deepcopy() 範例

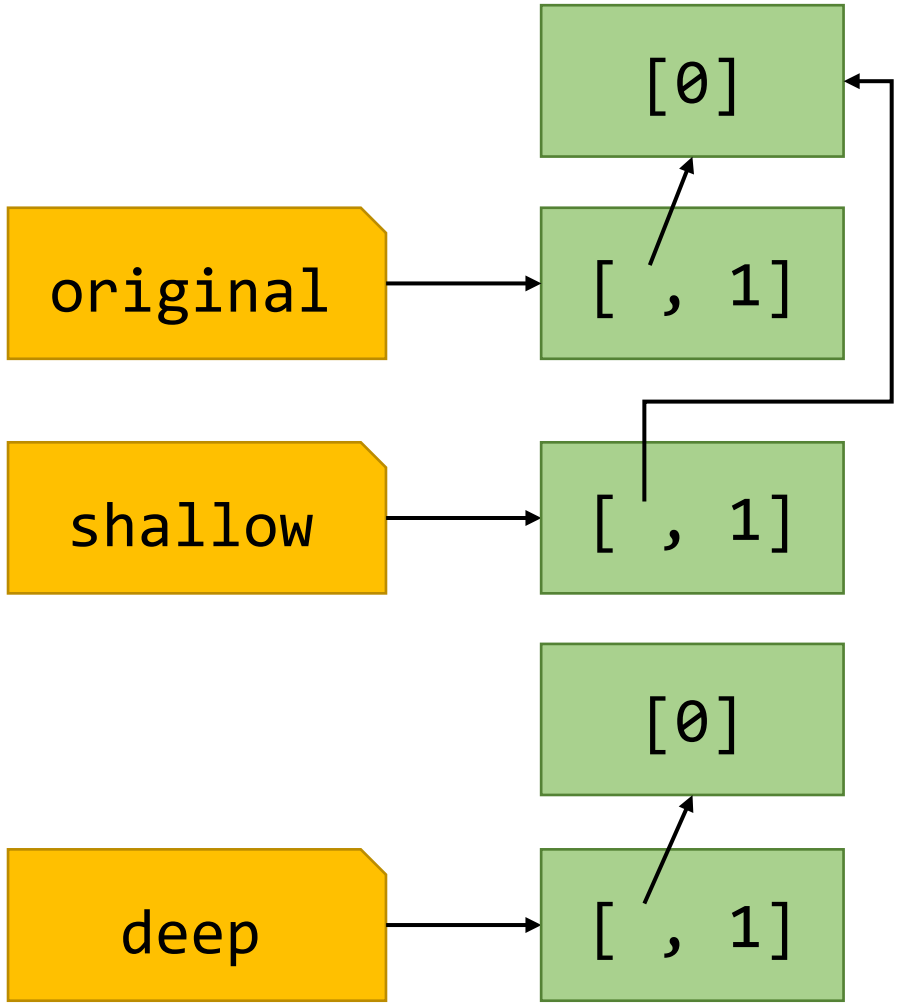
```
>>> original = [[0], 1]
>>> shallow = original[:]
>>> import copy
>>> deep = copy.deepcopy(original)
```

```
>>> shallow[1] = 2
>>> shallow
[[0], 2]
>>> original
[[0], 1]
>>> shallow[0][0] = 'zero'
>>> original
[['zero'], 1]
```

淺層拷貝與原始 list 的第二層 list 是同一個。
透過其中一個變數更改第二層 list 的值，會影響到另一個。

```
>>> deep[0][0] = 5
>>> deep
[[5], 1]
>>> original
[['zero'], 1]
```

深層拷貝獨立於原始 list。
更改深層拷貝的 list 對原始的 list 不會有任何影響。



課程大綱

1. list (串列)
2. list 的索引與切片
3. list 常用的方法與操作
4. list 的排序
5. 其他常見的 list 操作
6. 多層 list 和深層拷貝 (deepcopy)
7. tuple
8. set (集合)

tuple

- tuple 是跟 list 非常類似的資料結構。
- tuple 只能夠被建立，而不能夠被修改。
- 可把 tuple 視為無法變更內容的 list。

建立 tuple

list 是由 [ ] 把元素括起來。
tuple 是由 ( ) 把元素括起來。

```
>>> x = ('a', 'b', 'c')
>>> x
('a', 'b', 'c')
```

使用 tuple

list 與 tuple 中的元素是依照順序排列的。這種有順序的資料結構在 Python 中稱作序列 (sequence) 型別。
序列型別都是用 [n] 做索引來取得第 n 個位置的元素。
tuple 雖然使用 () 來建立，但是取值時請使用 []。

```
>>> x[2]
'c'
>>> x[1:]
('b', 'c')
>>> len(x)
3
>>> max(x)
'c'
>>> min(x)
'a'
>>> 5 in x
False
>>> 5 not in x
True
```

tuple 是不可變的型別

```
>>> x[2] = 'd'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support
item assignment
```

使用現有 tuple 建立新的 tuple

```
>>> x[:2]
('a', 'b')
>>> x + x
('a', 'b', 'c', 'a', 'b', 'c')
>>> x * 2
('a', 'b', 'c', 'a', 'b', 'c')
>>> x + (1, 2)
('a', 'b', 'c', 1, 2)
```

tuple 的切片、+ 算符、* 算符使用方式與 list 相同。

單一元素的 tuple 須加上逗號

由於 () 也用在數學運算的優先性，
加上逗號可消除歧異。

```
>>> x = 3
>>> y = 4
>>> (x + y)    # x 與 y 相加
7
>>> (x + y,)   # 因為有逗號，所以結果會產生單一元素的 tuple。
(7,)
>>> ()         # 產生空的 tuple
()
```

tuple 的自動解包、自動打包

Python 允許元素是變數的 tuple 出現在 = 符號的左邊。
tuple 中的變數會從 = 符號右邊的 tuple 接收相對應位置的值。

```
>>> (one, two, three, four) =  (1, 2, 3, 4)
>>> one
1
>>> two
2
```

也可以用更簡單的方式來寫。

```
>>> one, two, three, four = 1, 2, 3, 4
```

指派時，會自動解包 (unpacking) 資料，
然後一一分配指派給各個變數。

被打包 (packing) 成 tuple

交換變數的值

交換變數的值，通常需要靠第三個變數來暫存變數值。

```
>>> a = 1
>>> b = 2
>>> temp = a
>>> a = b
>>> b = temp
>>> a
2
>>> b
1
```

Python 可以用以下的簡單方式。

```
>>> a = 1
>>> b = 2
>>> a, b = b, a
>>> a
2
>>> b
1
```

list 與 tuple 的轉換

使用 `list()` 函式可以把 tuple 轉換成 list。
使用 `tuple()` 函式可以把 list 轉換成 tuple。

```
>>> list((1, 2, 3, 4))
[1, 2, 3, 4]
>>> tuple([1, 2, 3, 4])
(1, 2, 3, 4)
```

把字串分解為字元

使用 `list()` 或 `tuple()` 函式可以把字串分解為字元。

```
>>> list('Hello')
['H', 'e', 'l', 'l', 'o']
>>> tuple('hello')
('h', 'e', 'l', 'l', 'o')
```

`list()` 與 `tuple()` 函式可以適用於任何 Python 序列型別。字串就是字元序列。

課程大綱

1. list (串列)
2. list 的索引與切片
3. list 常用的方法與操作
4. list 的排序
5. 其他常見的 list 操作
6. 多層 list 和深層拷貝 (deepcopy)
7. tuple
8. set (集合)

set (集合)

- list、tuple 是由有序的資料所組成。
- set 是由無序的資料所組成的可變型別。
- set 中重複的資料會被自動刪除。
- set 主要用於：
  - 想知道一群資料中是否存在某個物件
  - 想要保持各元素的唯一性
- set 中的元素必須是不可變的資料型別。
  - 可作為 set 中的元素：整數、浮點數、字串、tuple
  - 不可作為 set 中的元素：list、set、dictionary
- set 使用 {} 來建立。

set 的操作

```
>>> x = {1, 2, 1, 2, 1, 2}
>>> x
{1, 2}
>>> x = set([1, 2, 3, 1, 3, 5])
>>> x
{1, 2, 3, 5}
>>> x.add(6)
>>> x
{1, 2, 3, 5, 6}
>>> x.remove(5)
>>> x
{1, 2, 3, 6}
```

```
>>> 1 in x
True
>>> 4 in x
False
>>> y = set([1, 7, 8, 9])
>>> x | y
{1, 2, 3, 6, 7, 8, 9}
>>> x & y
{1}
>>> x ^ y
{2, 3, 6, 7, 8, 9}
```

frozenset

- set 是可變型別。
- set 中的元素必須是不可變的資料型別。
- 所以 set 不能做為另一個 set 的元素。
- frozenset 像是一個 set，但是不可變，建立後無法更改，因此可以做為另一個 set 的元素。

frozenset 的範例

```
>>> x = set([1, 2, 3, 1, 3, 5])
>>> z = frozenset(x)
>>> z
frozenset({1, 2, 3, 5})
>>> z.add(6)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'frozenset' object has no attribute 'add'
>>> x.add(z)
>>> x
{1, 2, 3, 5, frozenset({1, 2, 3, 5})}
```

重點整理

	有序	無序
可變更 (mutable)	list	set
不可變更 (immutable)	tuple	

	適用度	呼叫方式
函式 (function)	比較通用	函式()、 模組.函式()
方法 (method)	類別專屬	物件.方法()

- 善用 Python 的資料結構。
- 要熟悉 list 的切片操作。
- 多層 list 若要完整複製，需要使用 deepcopy。