Chapter 6

字典資料結構

/ 蔣宗哲、江政杰

第三章我們曾經學習過列表 (list) 資料結構，本章我們將講述第二種資料結構－字典 (dictionary) 和它的衍生結構 Counter，它可以建立兩個資料間的對應關係，常用於查詢和計數；舉例來說，可以記錄中英文的對照表以及計算一堆字詞的出現次數。雖然這些功能也能透過列表完成，不過字典在程式簡潔度和執行效率上都有其優勢。學習完本章後，讀者將能快速完成資料儲存、查詢、計數以及排序應用。

6-1 以字串作鍵值的查詢

回顧：列表的查詢

到目前為止，我們已經相當熟悉列表資料結構，它可以容納多個（不同型態）的資料（甚至是另一個列表），而且可以透過索引值來取用其中的資料。若有一個列表 x，從索引值 i 與列表資料 x[i] 的配對來看，我們可以說列表維護了一種整數值 0, 1, 2, … 和資料的對應關係。

以程式 E6-1-1.py 為例，我們首先建立一個可存放 100 個資料的列表 scores。接著，詢問使用者全班人數，以 n 代表。程式第 03-04 行會讀入 n 個學生成績，以學生座號作為索引值，列表 scores 的 score[i] 代表第 i 號學生的成績。輸入完成後，第 05-09 行程式可以讓使用者輸入學生座號，然後查詢該生的成績。

E6-1-1.py: 以列表記錄座號與分數關係並查詢

```
01  scores = [0]*100
02  n = int(input('全班人數>'))
03  for i in range(n):
04      scores[i+1] = int(input('學生'+str(i+1)+'分數>'))
05  stu_id = int(input('你想知道幾號學生的分數>'))
06  if 1<=stu_id<=n:
07      print(scores[stu_id], '分')
08  else:
09      print('查無此人')
```

執行結果：

```
全班人數>5
學生1分數>100
學生2分數>90
學生3分數>80
學生4分數>95
學生5分數>88
你想知道幾號學生的分數>4
95 分
```

上述範例建立了學生座號與學生成績的對應關係，由於座號數字連續，所以把成績儲存在列表中以供後續查詢是一個不錯的作法。現在，讓我們來思考另一個查詢資料的例子。

假設我們想要以國際標準書號 ISBN 來查詢書名，若以列表來建立 ISBN 和書名的關係，雖然 ISBN 是一個整數值，可以作為索引值，但 ISBN 有 13 個數字，要建立一個可容納 10^{13} 個資料的列表會需要很大的記憶體空間；再者，圖書資料庫中不一定會存放所有的書籍資料，所以即使造出這麼大的列表，當中也可能有很多空間是沒有使用到而浪費的。

程式 E6-1-2.py 示範了以列表來完成 ISBN 查詢書名的作法，這裡我們假設只記錄 978986 開頭的書籍資料，所以只用了 ISBN 的後 7 碼。已知 9789863123798 的書號是對應到「Python 程式設計超入門」這本書，因此程式第 02 行以 books 列表的 books[3123798] 代表該書書名。

E6-1-2.py: 以列表記錄書號與書名關係並查詢

```
01  books = ['']*(10**7)
02  books[3123798] = 'Python 程式設計超入門'
03  books[4342549] = 'Python 程式設計實例入門'
04  isbn = int(input('輸入 ISBN >')) - 9789860000000
05  if books[isbn] != '':
06      print(books[isbn])
07  else:
08      print('查無此書')
```

執行結果：

```
輸入 ISBN >9789863123798
Python 程式設計超入門
```

如果我們要以書名反查 ISBN，以列表能作得到嗎？下面我們示範兩種可能的作法，兩種方法都以二維列表來存放書名及 ISBN。

程式 E6-1-3 將書名及 ISBN 兩兩存放在一個列表，然後再把這些兩個元素的列表存成一個更大的二維列表，如程式第 06-07 行所示。在函式 GetBook() 中，第 02 行依序檢查二維列表 books 中的每個一維列表 e，如果發現列表 e 的第一個元素（即書名資料）符合欲查詢的書名 name，即回傳該一維列表 e，此時程式第 11 行的條件式會判定為真，第 12 行便會列印出列表的第二個元素（即 ISBN）；如果 books 中沒有符合的，此函式的回傳值為 None，此時程式第 11 行的條件式會判斷為假，執行第 14 行，列印「查無此書」。

E6-1-3.py：書名反查 **ISBN**，方法一

```
01  def GetBook(books, name):
02      for e in books:
03          if e[0] == name:
04              return e
05
06  books = [['Python 程式設計超入門', 9789863123798],
07           ['Python 程式設計實例入門', 9789864342549]]
08
09  name = input('輸入書名>')
10  b = GetBook(books, name)
11  if b:
12      print(b[1])
13  else:
14      print('查無此書')
```

執行結果：

```
輸入書名>Python 程式設計實例入門
9789864342549
```

程式 E6-1-4 將所有書名存在一個一維列表，這些書籍的 ISBN 「依序」存在另一個一維列表，然後以這兩個列表形成一個二維列表，如第 06-07 行所示。函式 GetBook() 的第 03 行呼叫了列表的內建方法 index()，找出書名在 books 中的索引值並以 ind 表示之。第 04 行就回傳書名與書號。由於 books 的第一個元素是書名列表，第二個元素是書號列表，因此程式第 04 行以 books[1][ind] 從 books[1] 中取得和欲查詢書名 name 同索引值 ind 的書號。

E6-1-4.py：書名反查 ISBN，方法二

```
01  def GetBook(books, name):
02      if name in books[0]:
03          ind = books[0].index(name)
04          return name, books[1][ind]
05
06  books = [['Python 程式設計超入門', 'Python 程式設計實例入門'],
07           [9789863123798, 9789864342549]]
08
09  name = input('輸入書名>')
10  b = GetBook(books, name)
11  if b:
12      print(b[1])
13  else:
14      print('查無此書')
```

執行結果：

```
輸入書名>Python 程式設計實例入門
9789864342549
```

從前面的例子中，我們可以看到列表的侷限性，雖然列表在資料儲存時非常適合搭配迴圈的處理，但是還是會像上面書名反查書號的例子般碰到些許麻煩。本章將介紹 Python 的另一種利器：字典，讓我們在資料處理時更加方便。

6-2 字典 (dictionary) 資料結構

這一節裡我們來介紹一種新的資料結構，稱為 dictionary，中文是字典的意思。如同列表可以記錄整數索引值和資料的對應關係，字典也可以記錄一對鍵 (key) 與值 (value) 的對應關係，而且字典的鍵值不僅可以是整數，還可以是小數或字串。哇！那不就可以解決前一節我們遇到的以書名反查書號的問題了嗎，真是太好了！

在實際解決這個問題前，我們先以下列程式快速理解字典的用法。程式的第 01 行代表以 rec 作為一個空的字典。（還記得嗎？如果是一對中括號，那代表的是一個空的列表，請見程式 E3-3-1.py。）程式第 02-04 行分別建立起 'milk' 與 100、123 與 4、及 '臺灣' 與 'Taiwan' 的對應關係，其中每組關係的第一個資料（如 'milk'）稱為「鍵」，第二個（如 100）稱為「值」。在建立對應關係後，我們便可以用下標運算子取得「鍵」對應的「值」，如程式第 05 行會取得 100 然後列印出來。如同列表的索引值可以是變數，字典的鍵也可以是變數，如程式第 07-08 行所示。注意，字典中查無鍵值，會發生 KeyError 的錯誤。

E6-2-1.py: 字典的建立與查詢

```
01  rec = {}
02  rec['milk'] = 100
03  rec[123] = 4
04  rec['臺灣'] = 'Taiwan'
05  print(rec['milk'])
06  print(1+rec[123])
07  place = '臺灣'
08  print(rec[place])
```

執行結果：

```
100
5
Taiwan
```

來一個符合「字典」這個名稱的例子，程式 E6-2-2 實現一個「英翻中」的程式，其中第 01-04 行建立一個空字典，然後記錄三個英文單字與中文翻譯的對應關係。第 05 行令 query 代表使用者輸入的字串。第 06 行判斷字典中是否存在 query 代表的字串，如果有，便列印出對應的中文翻譯；否則，列印「未收錄此字」。

這個程式有兩個地方值得留意，第一是為了避免鍵不存在所引發的 KeyError，我們必須先確定鍵存在才使用下標運算子去取用對應的值；第二，字典和列表一樣，都可以用 in 查詢，不過列表是查詢某個值存不存在，但字典是查詢某個鍵存不存在。

E6-2-2.py： 簡易英翻中字典範例

```
01 mydict = {}
02 mydict['book'] = '書'
03 mydict['computer'] = '電腦'
04 mydict['program'] = '程式'
05 query = input('英翻中>')
06 if query in mydict:
07     print('解釋:', mydict[query])
08 else:
09     print('未收錄此字')
```

執行結果 1：	執行結果 2：
英翻中>book 解釋： 書	英翻中>csie 未收錄此字

更多字典的初始化與記錄方法

除了像前面兩個程式先建立空字典再以下標運算子逐一建立「鍵」-「值」關係外，底下我們再介紹幾種直接初始化字典的語法。讀者可以參考使用。其中方法 B 的程式片段很輕鬆地完成前一節中以書名反查書號的要求。

方法 **A**：以大括號初始化字典

```
01  eng = 'apple'
02  chinese = '蘋果'
03  mydict = {'book':'書', 'computer':'電腦', 'program':'程式',
04            eng: chinese}
05  print(mydict['book'])
```

方法 **B**：以 n×2 的列表初始化字典

```
01  books = [['Python 程式設計超入門', 9789863123798],
02           ['Python 程式設計實例入門', 9789864342549]]
03  mydict = dict(books)
04  print(mydict['Python 程式設計超入門'])
```

字典中每個鍵只能出現一次，對應一個值。記錄鍵值後，無法修改鍵的內容，但可以修改其對應值的內容。如程式 E6-2-3 在第 06 行把 'computer' 對應的中文翻譯由第 03 行設定的 '電腦' 改為 '計算機'。

E6-2-3.py：字典內的鍵－值關係可以更改

```
01  mydict = {}
02  mydict['book'] = '書'
03  mydict['computer'] = '電腦'
04  mydict['program'] = '程式'
05  query = input('英翻中>')
06  mydict['computer'] = '計算機'
07  if query in mydict:
08      print('解釋:', mydict[query])
09  else:
10      print('未收錄此字')
```

執行結果：

```
英翻中>computer
解釋: 計算機
```

當我們分別擁有鍵的列表與值的列表時，我們可以經由一個迴圈來建立它們兩兩的關係，如程式 E6-2-4 所示。程式第 01 行是數字一到九的中文國字，第 02 行是阿拉伯數字。第 04-05 行依序走訪這兩個列表，然後建立起一和 1、二和 2、三和 3，直到九和 9 的對應關係。程式第 06-08 行便可將使用者輸入的一串中文字轉變成阿拉伯數字。（注意：此處我們省略判斷鍵是否存在，以便讓程式較為精簡。）

E6-2-4.py：從中文轉換成阿拉伯數字

```
01  chinese = '一二三四五六七八九'
02  digit = '123456789'
03  mydict = {}
04  for i in range(len(chinese)):
05      mydict[ chinese[i] ] = digit[i]
06  query = input('輸入由一、二、..... 九的中文字>')
07  for c in query:
08      print(mydict[c], end='')
```

執行結果：

```
輸入由一、二、..... 九的中文字>九九八一
9981
```

有個 zip() 函式，有興趣的讀者可以試試它有什麼妙用。

多維字典

如同列表中的內容可以是列表（因而形成多維列表），字典的值也可以是字典或是列表。程式 E6-2-5 示範了以列表作為字典的值（第 02、03 行），因此第 06 行的 scores['張三'] 會得到列表 [100, 90, 92]，而 scores['張三'][0] 便會得到 100。

E6-2-5.py：以列表作為字典的值

```
01  scores = {}
02  scores['張三'] = [100, 90, 92]
03  scores['李四'] = []
04  scores['李四'] += [85]
05  scores['李四'] += [75]
06  print(scores['張三'][0])
07  print(scores['李四'][1])
```

執行結果：

```
100
75
```

程式 E6-2-6 示範了以字典作為字典的值，第 02 行令 students['張三'] 是一個空字典，第 03-05 行將三組鍵值記錄到 students['張三'] 中。因此，如果第 06 行使用者輸入 '張三'，第 07 會判斷出 students 中有 '張三' 這個鍵，然後第 08 行會列印出 students['張三'] 對應的內容，也就是存有三組鍵值關係的字典。

E6-2-6.py：以字典作為字典的值

```
01  students = {}
02  students['張三'] = {}
03  students['張三']['出生地'] = '台北'
04  students['張三']['體重'] = 70
05  students['張三']['考試分數'] = [100, 80, 85]
06  stu = input('學生姓名>')
07  if stu in students:
08      print(students[stu])
```

執行結果：

```
學生姓名>張三
{'出生地': '台北', '體重': 70, '考試分數': [100, 80, 85]}
```

字典應用：計數

字典有個常見應用是計算每一個資料出現的次數，我們把想要計次的資料作為字典的鍵，把次數作為字典的值。程式 E6-2-7 是一個計票程式範例，第 01 行 votes 列表代表得票者的姓名，第 02 行建立空字典 counts。第 03-07 行走訪 votes 中所有姓名，若姓名首次出現，會在字典中建立該姓名對應的值為 1（代表得到 1 票）；若姓名已出現過，則會將該姓名對應的票數加 1。第 08-12 行則為查詢的程式碼。

E6-2-7.py：計票程式範例一

```
01  votes = ['張三', '李四', '張三', '張三', '李四']
02  counts = {}
03  for v in votes:
04      if v not in counts:
05          counts[v] = 1
06      else:
07          counts[v] = counts[v]+1
08  query = input('候選人姓名>')
09  if query in counts:
10      print('得票數:', counts[query])
11  else:
12      print('查無此人')
```

執行結果：

```
候選人姓名>張三
得票數: 3
```

程式 E6-2-7 需要第 04 行的判斷，否則若鍵值不存在，直接執行第 07 行會產生錯誤。如果覺得多加這個判斷很麻煩，字典型態有提供 get() 函式，可以設定鍵不存在時的回傳值。如下列程式碼所示。程式執行第 02 行時，mydict 尚無 '張三' 這個鍵，此時 get('張三', 0) 便會回傳 0。

```
01  mydict = {}
02  print(mydict.get('張三', 0))
03  mydict['張三'] = 4
04  print(mydict.get('張三', 0))
```

執行結果：

```
0
4
```

若以 get() 函式來改寫程式 E6-2-7，可精簡成為程式 E6-2-8。主要差異為程式 E6-2-7 的第 04-07 行縮短為程式 E6-2-8 的第 04 行。

E6-2-8.py：計票程式範例二

```
votes = ['張三', '李四', '張三', '張三', '李四']
counts = {}
for v in votes:
    counts[v] = counts.get(v, 0) + 1
query = input('候選人姓名>')
if query in counts:
    print('得票數:', counts[query])
else:
    print('查無此人')
```

執行結果：

```
候選人姓名>張三
得票數: 3
```

隨堂練習

6.2.1　請嘗試以字典重寫 4.2.1 的隨堂練習

6-3 走訪字典內容

以字典來計數真的很方便，對吧！那我們該怎麼取出所有計數結果呢？之前我們只學到以一個特定的鍵來查詢其對應的值，本節我們來看看如何以 for 句型來走訪字典中的資料。（是不是覺得似曾相識？在學習列表時，一開始我們也學習以下標運算子取得列表中的單一筆資料，後來再學習以 for 句型走訪整個列表。）

E6-3-1.py：走訪字典

```
01 mydict = {'book':'書', 'computer':'電腦', 'program':'程式'}
02 for e in mydict:
03     print(1, e)
04 for e in mydict.keys():
05     print(2, e)
06 for e in mydict.items():
07     print(3, e)
08 for e in mydict.values():
09     print(4, e)
```

執行結果：

```
1 book
1 computer
1 program
2 book
2 computer
2 program
3 ('book', '書')
3 ('computer', '電腦')
3 ('program', '程式')
4 書
4 電腦
4 程式
```

程式 E6-3-1 有四個 for 句子，其中第 02 行 in mydict 和第 04 行 in mydict.keys() 的意義相同，都是走訪字典中的所有鍵。第 06 行的 mydict.items() 代表同時取出 (鍵 , 值)，而第 08 行的 mydict.values() 則表示僅取出值。程式在列印時加上數字 1、2、3 和 4 是為了讓讀者更容易看出程式碼與列印結果的對應關係。

學會走訪後，我們將計票程式 E6-2-8 修改如 E6-3-2，該程式在計票後會列印出每位候選人的得票數。

E6-3-2.py：計票並列印所有結果

```
01 votes = ['張三', '李四', '張三', '張三', '李四']
02 counts = {}
03 for v in votes:
04     counts[v] = counts.get(v, 0) + 1
05 for e in counts:
06     print(e, '得了', counts[e], '票')
```

執行結果：

```
張三 得了 3 票
李四 得了 2 票
```

以 for 句型走訪字典，走訪順序如同字典新增鍵值的順序。以程式 E6-3-1 為例，鍵的順序為 'book'、'computer'、'program'，以程式 E6-3-2 為例，鍵的順序為 '張三'、'李四'。以下兩個程式將示範如何以鍵或值的大小關係排序後走訪。程式 E6-3-3 的第 04 行以 keys() 取出字典 books 中所有的鍵，然後以 sorted() 函式排序（預設由小到大），最後以 for 句型走訪排序後的鍵。因此，原本第 01-03 行建立字典時，三本書並未依書號排序（9789864342549 在 9789863123798 後但在 9789862487488 前），但經過第 04 行的處理再走訪，第 05 行的列印結果已由小到大排序。

E6-3-3.py：以鍵排序後走訪字典

```
01 books = {9789863123798: 'Python 程式設計超入門',
02           9789864342549: 'Python 程式設計實例入門',
03           9789862487488: '書呆與阿宅'}
04 for e in sorted(books.keys()):
05     print(e, books[e])
```

執行結果：

```
9789862487488 書呆與阿宅
9789863123798 Python 程式設計超入門
9789864342549 Python 程式設計實例入門
```

以值排序走訪比較複雜一點，我們雖然能以 values() 函式取出字典中所有的值，但我們沒辦法從值反查鍵。因此，我們的作法是以 items() 同時取出鍵 - 值對，然後自訂函式來訂定比較欄位為值，再以 sorted() 函式來排序。

程式 E6-3-4 中，第 07 行先取出 counts.items()，在程式 E6-3-1 中，我們已看到這會是所有鍵 - 值的資料，每一對鍵 - 值以元組的方式表示，[0] 為鍵而 [1] 為值。以 sorted() 函式對 counts.items() 排序時，每次會比較兩對鍵 - 值，預設的情況會先比鍵再比值。因為此處我們要以值來比較大小，我們另訂一個函式 cmp()，該函式傳回傳入物 x 的第二個資料 x[1]，也就是鍵 - 值中的值。在 sorted() 中指定 reverse=True 代表由大到小排序。

E6-3-4.py：以值排序後走訪字典

```
01 votes = ['王五', '張三', '李四', '張三', '張三', '李四']
02 counts = {}
03 for v in votes:
04     counts[v] = counts.get(v, 0) + 1
05 def cmp(x):
06     return x[1]
07 for e in sorted(counts.items(), key=cmp,
08 reverse=True):
       print(e[0], '得了', e[1], '票')
```

執行結果：

```
張三 得了 3 票
李四 得了 2 票
王五 得了 1 票
```

程式的結果會以票數由高到低列印出來。如果不特別處理，如下列程式碼，列印的結果會依照新增鍵的順序。

```
votes = ['王五', '張三', '李四', '張三', '張三', '李四']
counts = {}
for v in votes:
    counts[v] = counts.get(v, 0) + 1
for e in counts:
    print(e, '得了', counts[e], '票')
```

執行結果：

```
王五 得了 1 票
張三 得了 3 票
李四 得了 2 票
```

大家可以發現，列表與字典的使用，有點像又有點不一樣。建議大家可以多熟悉列表與字典的操作，搭配迴圈的運作，可以解決非常多樣的困難問題。

6-4 Counter 資料結構

說到計數，字典已經夠方便了，不過，還有更方便的！登登！讓我們來看看 Counter。Counter 是一種特別的字典，它可以拿需計數的資料來初始化，並在初始化後就得到每個資料的次數。對於不存在的鍵，它有預設值 0；它可以取出次數最高的項目，而且它還支援計數器的運算。

使用前，我們需從 collections 模組中匯入 Counter 型態，如下列程式第 01 行所示。第 02 行我們以字串 'abcdabcaa' 初始化 Counter，並以 c 指稱之。從第 03 行列印的結果，我們可以看到 c 已經算好每個字母的次數。Counter 也是一個字典，因此我們可以像使用字典一樣，以鍵查值，如第 04 行所示。

```
01  from collections import Counter
02  c = Counter('abcdabcaa')
03  print(c)
04  print(c['a'])
```

執行結果：

```
Counter({'a': 4, 'b': 2, 'c': 2, 'd': 1})
4
```

運用 Counter，我們可以把程式 E6-3-2 改寫成 E6-4-1。

E6-4-1.py：計票並列印所有結果：以 **Counter** 實作

```
from collections import Counter
votes = ['張三', '李四', '張三', '張三', '李四']
counts = Counter(votes)
print(counts)
for e in counts:
    print(e, '得了', counts[e], '票')
```

執行結果：

```
Counter({'張三': 3, '李四': 2})
張三 得了 3 票
李四 得了 2 票
```

當然，我們也可以使用字典原有的初始化語法來初始化 Counter。

```
01 from collections import Counter
02 c = Counter({'張三':3, '李四':2})
03 print(c)
04 print(c['張三'])
```

```
執行結果：
Counter({'張三': 3, '李四': 2})
3
```

下列程式左邊使用 Counter，右邊使用字典。當鍵不存在時，Counter 會傳回 0 值，但字典會造成 KeyError。

```
01 from collections import Counter
02 c = Counter()
03 print(c['a'])
```

```
執行結果：
0
```

```
01 d = dict()
02 print(d['a'])
```

```
執行結果：
KeyError: 'a'
```

Counter 有個 most_common(n) 函式可以傳回次數最高的前 n 個鍵 - 值對，如程式 E6-4-2 所示。若次數相同，會依建立的鍵順序。若未傳入 n 值或 n 值過大，會傳回所有的鍵 - 值對。

E6-4-2.py：Counter 的 most_common() 函式

```
01 from collections import Counter
02 c = Counter('acbcdabcaab')
03 print(c.most_common(1))
04 print(c.most_common(2))
05 print(c.most_common())
```

```
執行結果：
[('a', 4)]
[('a', 4), ('c', 3)]
[('a', 4), ('c', 3), ('b', 3), ('d', 1)]
```

Counter 還可以支援一些運算，如表 6-4-1 所示。程式 E6-4-3 可以更清楚了解運算的結果。第 02 和 03 行建立兩個 Counter，a 和 b，並在第 04 和 05 行列印其內容。第 06 行列印相加的結果，舉例來說，鍵 'c' 對應的次數為 3+4=7。第 07 行列印相減的結果，舉例來說，鍵 'c' 對應的次數為 3–4=-1，

因為次數不為正，所以就不再有 'c' 這個鍵了。第 08 行列印交集的結果，鍵 'c' 對應的次數為 3 和 4 中的較小值，因此變成了 3。第 09 行列印聯集的結果，鍵 'c' 對應的次數為 3 和 4 中的較大值，因此變成了 4。

表 6-4-1: Counter 的運算

運算	運算式	運算結果
加法	x+y	將次數相加。
減法	x-y	x 次數減 y 的次數，刪除次數非正的鍵值。
交集	x&y	取兩者次數的較小值。
聯集	x|y	取兩者次數的較大值。

E6-4-3.py：Counter 的常見運算

```
01 from collections import Counter
02 x = Counter('abccdabcaab')
03 y = Counter('abccccd')
04 print('  x', x)
05 print('  y', y)
06 print('x+y', x+y)
07 print('x-y', x-y)
08 print('x&y', x&y)
09 print('x|y', x|y)
```

執行結果：

```
  x Counter({'a': 4, 'b': 3, 'c': 3, 'd': 1})
  y Counter({'c': 4, 'a': 1, 'b': 1, 'd': 1})
x+y Counter({'c': 7, 'a': 5, 'b': 4, 'd': 2})
x-y Counter({'a': 3, 'b': 2})
x&y Counter({'c': 3, 'a': 1, 'b': 1, 'd': 1})
x|y Counter({'a': 4, 'c': 4, 'b': 3, 'd': 1})
```