Python 程式設計　講義

字典、鍵值映射與雜湊表

滄海《Python 程式設計初學指引》二版 · CH06

§ 1　名詞速查表

中文	English	一句話定義	例	易混
字典	dict	用**鍵**索引**值**的可變容器，元素是鍵值組合	`{'Java聖經': 780}`	用鍵取值，不是序號
鍵	key	字典中用來索引的標籤，**唯一**、通常用簡單型別	`'Java聖經'`	重複鍵會被覆蓋
值	value	鍵對應的資料，可為複雜型別（串列、字典）	`780`	值可重複
鍵值組合	key-value pair	一個鍵配一個值的單位，`len()` 算的就是它的個數	`'Java聖經': 780`	—
映射	mapping	由鍵找值的對應關係（字典的數學名稱）	—	—
雜湊表	hash table	字典的底層結構，讓「鍵→值」能快速直達	—	值→鍵無此捷徑
KeyError	—	用 `[鍵]` 取一個不存在的鍵時拋出的例外	`books['不存在']`	`.get()` 不會丟
無序	unordered	字典不依輸入順序、不能用序號取值（課本立場）	—	與串列序列不同
`.get()`	—	安全取值：找不到鍵回傳預設值（預設 `None`），不報錯	`books.get(k, 0)`	不改變字典
`.setdefault()`	—	鍵不存在才新增；存在則不動，回傳該鍵的值	`books.setdefault(k,v)`	get 不新增、它會

§ 2 核心概念

核心概念

字典 (dict) 是用「鍵 (key)」去索引「值 (value)」的容器，每個元素是一組**鍵值組合 (key-value pair)**，數學上叫**映射 (mapping)**。和串列、元組最大的不同：串列用**序號**（位置）取值，字典用**鍵**取值。

字典底層是**雜湊表 (hash table)**：鍵經過雜湊運算直接定位到值，所以「**鍵→值**」非常快。但反過來「**值→鍵**」沒有這條捷徑，只能用迴圈逐一比對（資料量大時很慢）。這也是為什麼鍵要**唯一**（重複鍵後者覆蓋前者）、且通常用字串或數字這類簡單、不可變的型別；值則可以是串列、字典等複雜型別。

第三點延續 CH05 的「變數是名牌」：字典是**可變物件**，`newbooks = books` 只是貼第二張名牌（別名），兩者連動；要真正複製出獨立的字典，得用 `.copy()`。

§ 3 主要內容

3.1 建立字典：三種寫法

```
# 1. 大括號 { 鍵: 值 }（最常用）
books = {'C程式設計入門': 450, 'Java聖經': 780, 'Python程式設計手冊': 520}

# 2. dict() 傳入鍵值對的串列
books = dict([['Python程式設計手冊', 520], ['Java聖經', 780]])

# 3. 多行寫法（鍵值多時可讀性較好）
books = {
    'C程式設計入門': 450,
    'Java聖經': 780,
    'Python程式設計手冊': 520,
}
```

字典**無序**：印出時不保證照輸入順序（課本立場），所以不能、也不需要用序號取值。

3.2 取值：`[鍵]` 與 KeyError、`.get()`

```
books['Java聖經']          # 780  用鍵取值
books['不存在的書']        # KeyError！鍵不存在就拋例外
books.get('不存在的書')          # None   找不到回 None、不報錯
books.get('不存在的書', '查無')  # '查無'  找不到回第二引數
```

取值兩條路

`[鍵]` 找不到鍵會 **KeyError** 中斷程式；`.get(鍵, 預設值)` 找不到只回傳預設值（沒給第二引數時回 `None`），程式繼續跑。不確定鍵在不在時用 `.get()` 較安全。

3.3 新增、修改、刪除

```
books['新書'] = 600          # 鍵不存在 → 新增；鍵已存在 → 修改該值
books.setdefault('Java聖經', 900)  # 鍵已存在 → 不動（仍 780）、回傳 780
books.setdefault('新書2', 300)     # 鍵不存在 → 新增 300

del books['Java聖經']        # 刪除一組鍵值
books.clear()                # 清空成 {}
del books                    # 連整個變數刪除（之後再用 → NameError）
```

兩個陷阱

`del books`（沒給鍵）會刪掉**整個字典變數**，之後用到會 `NameError`。`.setdefault()` 和 `.get()` 像，但 `.get()` 純取值不改字典、`.setdefault()` 在鍵不存在時**會新增**。

3.4 字典的長度與列舉

`len(字典)` 回傳鍵值組合的個數。用 `for` 走訪時，預設走訪的是**鍵**：

```
for name in books:           # 等同 for name in books.keys()，拿到「鍵」
    print(name)
for price in books.values():  # 拿到「值」
    print(price)
for k, v in books.items():    # 拿到 (鍵, 值)
    print(k, v)
```

`.keys()` 回傳 `dict_keys([...])`、`.values()` 回 `dict_values([...])`、`.items()` 回 `dict_items([(k,v),...])`。

3.5 招牌一：字典是雜湊表（鍵→值快、值→鍵慢）

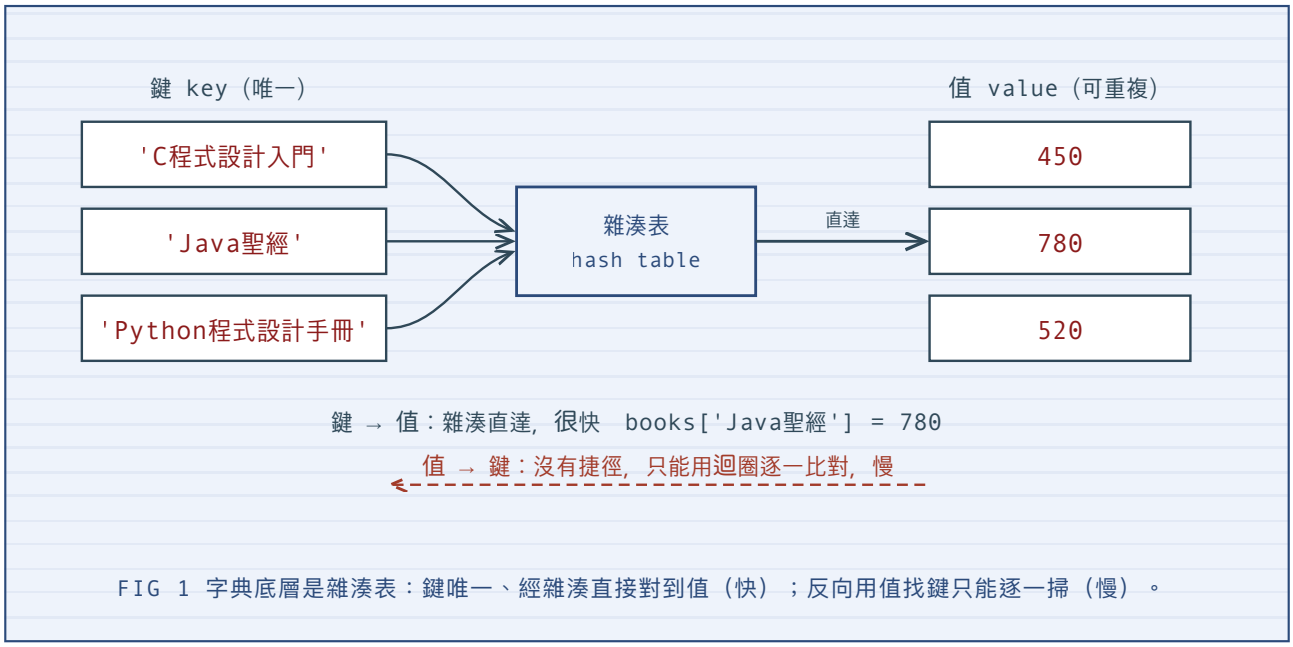

FIG 1 字典底層是雜湊表：鍵唯一、經雜湊直接對到值（快）；反向用值找鍵只能逐一掃（慢）。

反向找鍵的寫法（逐一比對）：

```
target = 520
for name in books:
    if books[name] == target:
        print(name)        # 找到 → 印出鍵
```

3.6 招牌二：複製——`=` 是別名，`.copy()` 才獨立

延續 CH05 的物件參照：字典是可變物件，`=` 只是再貼一張名牌。

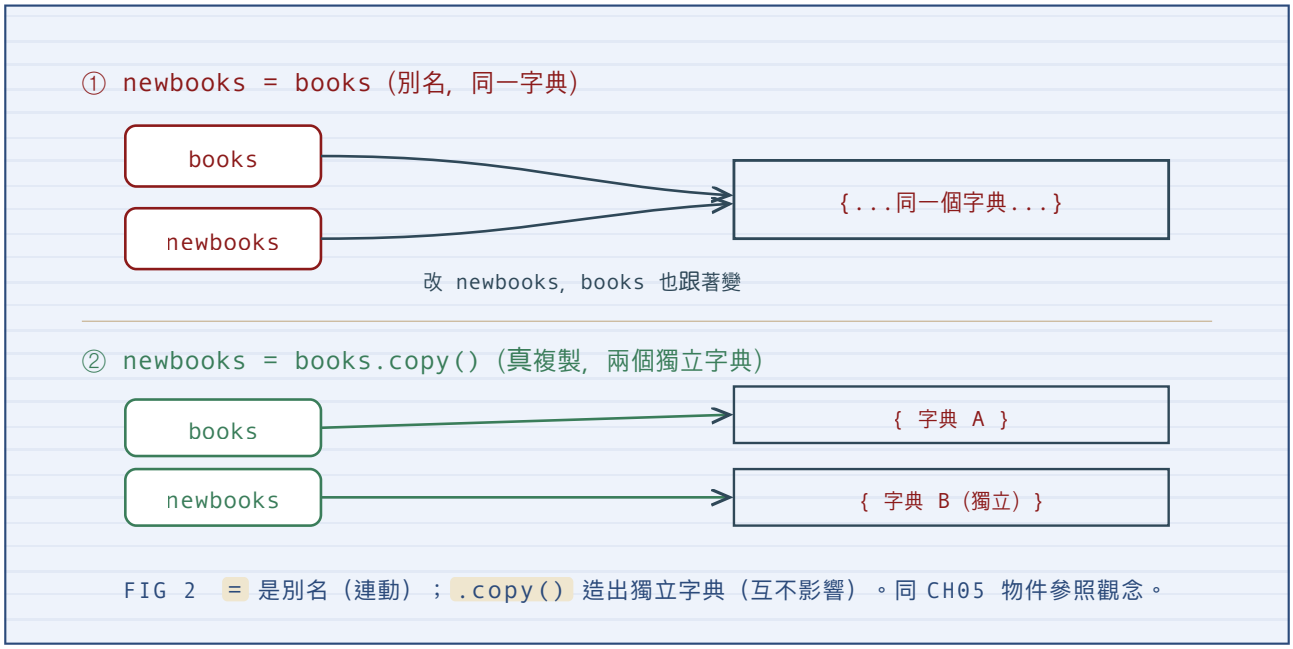

FIG 2 `=` 是別名（連動）；`.copy()` 造出獨立字典（互不影響）。同 CH05 物件參照觀念。

3.7 合併與巢狀複雜值

```
books.update(newbooks)      # 把 newbooks 併進 books，重複的鍵以 newbooks 為準

# 值可以是串列或字典（鍵保持簡單型別）
shelf = {'程式設計': {'Python手冊': 520, 'Java聖經': 780},
         '文學': {'紅樓夢': 600}}
shelf['程式設計']['Python手冊']    # 520  巢狀逐層取
```

§ 4 語法與方法速查

```
# 建立
{鍵: 值, ...}           dict([[k, v], ...])

# 取值
d[鍵]                    # 找不到 → KeyError
d.get(鍵)                # 找不到 → None
d.get(鍵, 預設值)        # 找不到 → 預設值

# 增刪改
d[鍵] = 值               # 不存在新增、存在修改
d.setdefault(鍵, 值)     # 不存在才新增
del d[鍵]                del d        d.clear()

# 列舉
len(d)    for k in d:    d.keys()    d.values()    d.items()

# 複製 / 合併
b = d                  # 別名（連動）
b = d.copy()           # 真複製（獨立）
d.update(其他字典)   # 合併、重複鍵覆蓋
```

方法	找不到鍵	改變字典？
`d[k]`	**KeyError**	否（取值）
`d.get(k)`	回 `None`	否
`d.get(k, x)`	回 `x`	否
`d.setdefault(k, x)`	新增 `k:x` 並回 `x`	**是**（新增時）

§ 5 常見錯誤

常見錯誤

- `d[不存在的鍵]` **直接丟 KeyError**：不確定鍵在不在時用 `d.get(鍵, 預設值)`。
- `del d` **（漏給鍵）刪掉整個字典變數**：之後用到會 `NameError`；要刪一組用 `del d[鍵]`。
- **誤把字典當有序**：字典不能用序號 `d[0]` 取「第一個」（除非剛好有鍵是 `0`）；要走訪用 `for`。
- `.get()` **與** `.setdefault()` **混用**：`.get()` 純取值不改字典；`.setdefault()` 在鍵不存在時會新增。
- `=` **當成複製**：`b = d` 是別名，改 `b` 連帶改 `d`；要獨立用 `d.copy()`。
- **想用值反查鍵卻直接** `d[值]`：字典只能鍵→值；值→鍵要 `for` 迴圈逐一比對。
- **重複鍵**：`{'a':1, 'a':2}` 只會留 `'a':2`（後者覆蓋）。
- **拿可變物件（串列）當鍵**：鍵必須是不可變型別，`{[1,2]: 'x'}` 會 `TypeError`。

§ 6 練習題

練習 1（一般題）：建立與取值

> **引導步驟**
>
> 1. 用大括號建一個 `prices`，三本書對三個價格。
> 2. 用 `[鍵]` 印出某本書價格。
> 3. 印一個不存在的鍵會怎樣？改用 `.get(鍵, '查無')` 再試。

解答

```
prices = {'Python手冊': 520, 'Java聖經': 780, 'C入門': 450}
print(prices['Java聖經'])            # 780
# print(prices['VB'])                # KeyError
print(prices.get('VB', '查無'))      # 查無
```

練習 2（一般題）：列舉 keys / values / items

引導步驟

1. 用 `for ... in d:` 印出所有鍵。
2. 用 `.values()` 把所有價格加總（`sum`）。
3. 用 `.items()` 一行印出「書名: 價格」。

解答

```
for name in prices: print(name)
print(sum(prices.values()))                  # 1750
for k, v in prices.items(): print(f"{k}: {v}")
```

練習 3（重要題）：別名 vs .copy()

引導步驟

1. `a = {'x': 1}`，`b = a`（別名）、`c = a.copy()`（複製）。
2. 對 `a` 做 `a['y'] = 2`。
3. 印 `b`、`c`，預測哪個跟著多了 `'y'`。

解答

```
a = {'x': 1}
b = a           # 別名
c = a.copy()    # 真複製
a['y'] = 2
print(b)    # {'x': 1, 'y': 2}  別名連動
print(c)    # {'x': 1}          複製獨立
```

別名 `b` 與 `a` 是同一個字典；`.copy()` 的 `c` 是獨立字典。

練習 4（重要題）：反向找鍵

引導步驟

1. 字典只能鍵→值，值→鍵要怎麼做？
2. 用 `for` 走訪鍵，比對 `d[鍵] == 目標值`。
3. 找到就印出鍵。為什麼這比鍵→值慢？

解答

```
prices = {'Python手冊': 520, 'Java聖經': 780, 'C入門': 450}
target = 780
for name in prices:
    if prices[name] == target:
        print(name)        # Java聖經
```

鍵→值有雜湊表直達；值→鍵沒有，只能逐一比對，資料量大時慢。

練習 5（一般題）：setdefault 與 update

引導步驟

1. `setdefault` 對「已存在的鍵」會不會改值？對「不存在的鍵」呢？
2. `update` 合併時，重複的鍵以誰為準？

解答

```
d = {'a': 1}
d.setdefault('a', 99)    # 'a' 已存在 → 不變，仍 1
d.setdefault('b', 2)     # 'b' 不存在 → 新增 2
print(d)                 # {'a': 1, 'b': 2}
d.update({'b': 20, 'c': 3})  # 重複鍵 b 以參數為準
print(d)                 # {'a': 1, 'b': 20, 'c': 3}
```

§ 7　自我檢核

- ☐ 能說出字典用「鍵」取值、串列用「序號」取值的差別。
- ☐ 能用大括號與 `dict()` 兩種方式建立字典。
- ☐ 知道 `d[不存在鍵]` 會 KeyError，會改用 `.get(鍵, 預設)` 安全取值。
- ☐ 能用 `[鍵]=值` 新增/修改、`del d[鍵]` 刪除，並知道 `del d` 會刪整個字典。
- ☐ 能用 `.keys()` / `.values()` / `.items()` 與 `for` 列舉字典。
- ☐ 能解釋字典是雜湊表：鍵→值快、值→鍵要逐一比對。
- ☐ 能寫出反向找鍵的迴圈。
- ☐ 能分辨 `=`（別名連動）與 `.copy()`（獨立複製）。

- [ ] 知道 `.update()` 合併時重複鍵會被覆蓋，鍵必須是不可變型別。