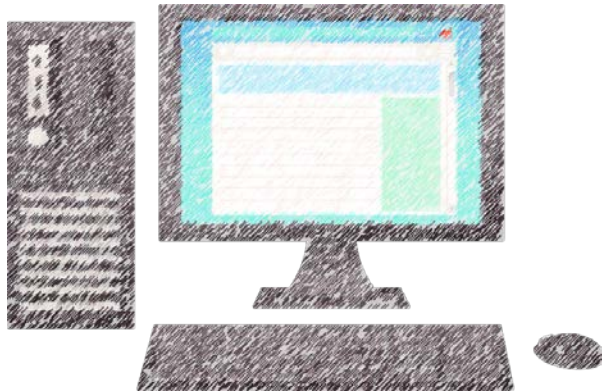


桃園市政府勞動局 112 年勞工學苑產業應用班
Python 程式設計：從零基礎入門到進階

第 6 單元

字典



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. 字典簡介
2. 字典的操作
3. 字典應用範例：字數統計
4. 什麼資料型別可以當作字典的鍵來使用？
5. 稀疏矩陣
6. 以字典作為快取
7. 字典的效率

課程大綱

1. 字典簡介
2. 字典的操作
3. 字典應用範例：字數統計
4. 什麼資料型別可以當作字典的鍵來使用？
5. 稀疏矩陣
6. 以字典作為快取
7. 字典的效率

字典 (Dictionary)

- Python 的字典是一種以鍵 (key):值 (value) 對應的資料結構。
 - 使用 key 來查詢對應的 value。
 - 字典裡每一組資料稱作一組鍵值對 (key-value pair)。
 - 類似字典裡以單字查解釋的方式。
- 字典中的鍵必須是唯一的。
- 字典使用大括號 {} 來標示。
- 建立字典的語法為：
{key1:value1, key2:value2, ...}
大括號 {} 裡面放零組以上的鍵值對。

字典的建立

可以從建立一個空字典開始。

```
>>> ages = {}  
>>> ages  
{}  
>>> type(ages)  
<class 'dict'>
```

字典的建立 (續)

也可以在建立字典時，同時指定初始的鍵-值對應。

```
>>> ages = {0:13, 1:14, 2:15}
>>> ages
{0: 13, 1: 14, 2: 15}
>>> type(ages)
<class 'dict'>
```

這個範例使用整數作為鍵。

字典的建立 (續)

整數的鍵不一定要由 0 開始，
也不一定要照順序排列。

```
>>> ages = {65:13, 12:14, 43:15}  
>>> ages  
{65: 13, 12: 14, 43: 15}
```



字典的建立 (續)

除了整數之外，字串也可以
作為字典的鍵。

```
>>> ages = {"Mary":13, "John":14, "Tony":15}  
>>> ages  
{ 'Mary': 13, 'John': 14, 'Tony': 15 }
```

字典的建立 (續)

鍵必須是唯一的。

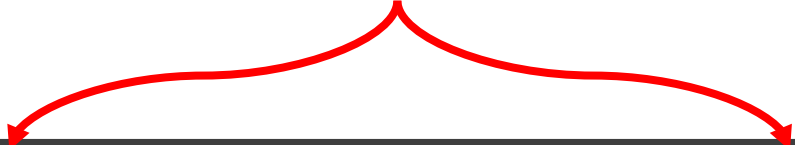


```
>>> ages = {"Mary":13, "John":14, "Mary":15}  
>>> ages  
{ 'Mary': 15, 'John': 14}
```

若使用相同的鍵，會導致前面設定好的值被覆蓋。

字典的建立 (續)

不同的鍵可以對應到相同的值。



```
>>> ages = {"Mary":13, "John":14, "Tony":13}  
>>> ages  
{ 'Mary': 13, 'John': 14, 'Tony': 13 }
```

使用鍵來存取字典

```
>>> ages = {"Mary":13, "John":14, "Tony":13}
>>> ages
{'Mary': 13, 'John': 14, 'Tony': 13}
>>> ages["Mary"] ← 取值。
13
>>> ages["Tony"] = 15 ← 修改一筆資料。
>>> ages
{'Mary': 13, 'John': 14, 'Tony': 15}
>>> ages["Peter"] = 12 ← 新增一筆資料。
>>> ages
{'Mary': 13, 'John': 14, 'Tony': 15, 'Peter': 12}
```

字典與 list 的比較

- list 與字典都可以儲存任何類型的物件。
- list 的索引表示元素在 list 中的順序。
 - list 的索引是連續的整數。
 - list 中的元素已經依照其索引來排序。
- 字典透過鍵作為索引來存取資料。
 - 字典的鍵可以是整數、字串、或其他 Python 物件。
 - 字典內的資料並不會自動依照鍵來排序。

範例：字典與 list 的比較

```
>>> x = [] ← 建立一個空的 list。
>>> y = {} ← 建立一個空的字典。
>>> y[0] = 'Hello' ← 把 0:'Hello' 存到字典 y。
>>> y[1] = 'Goodbye' ← 把 1:'Goodbye' 存到字典 y。
>>> x[0] = 'Hello' ← 不可使用 list 中不存在的索引來賦值。
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
>>> print(y[0])
Hello
>>> y[1] + ", Friend."
'Goodbye, Friend.'
```

對於字典的儲存，若鍵不存在，字典會自動以 `key:value` 的方式來儲存新的對照關係。

對字典取值後，可進行運算

```
>>> y = {}  
>>> y["two"] = 2  
>>> y["pi"] = 3.14  
>>> y["two"] * y["pi"]  
6.28
```

若鍵不存在

若使用不存在的鍵來讀取字典，會發生錯誤。

```
>>> ages = {"Mary":13, "John":14, "Tony":13}
>>> ages["Kevin"]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'Kevin'
```


課程大綱

1. 字典簡介
2. 字典的操作
3. 字典應用範例：字數統計
4. 什麼資料型別可以當作字典的鍵來使用？
5. 稀疏矩陣
6. 以字典作為快取
7. 字典的效率

len() 函式

len() 函式可用來查詢字典中鍵值對的組數。

```
>>> english_to_french = {'red': 'rouge', 'blue': 'bleu', 'green': 'vert'}  
>>> len(english_to_french)  
3
```

字典的 `keys()` 方法

`keys()` 方法可取得字典中所有的鍵。
可轉成 `list`，以便於後續的資料處理。

```
>>> english_to_french.keys()  
dict_keys(['red', 'blue', 'green'])  
>>> list(english_to_french.keys())  
['red', 'blue', 'green']
```

從 Python 3.6 開始，字典會保留鍵的建立順序，所以 `keys()` 方法回傳的鍵，順序就代表鍵的建立順序。

字典的 `values()` 方法

`values()` 方法可取得字典中所有的值。
可轉成 `list`，以便於後續的資料處理。

```
>>> english_to_french.values()  
dict_values(['rouge', 'bleu', 'vert'])  
>>> list(english_to_french.values())  
['rouge', 'bleu', 'vert']
```

字典的 `items()` 方法

`items()` 方法可取得字典中所有的鍵值對。
每一組鍵值對會放在一個 `tuple` 裡。
可轉成 `list`，以便於後續的資料處理。

```
>>> english_to_french.items()
dict_items([('red', 'rouge'), ('blue', 'bleu'), ('green', 'vert')])
>>> list(english_to_french.items())
[('red', 'rouge'), ('blue', 'bleu'), ('green', 'vert')]
```

字典的方法回傳的視圖 (View) 物件

- 字典的 `keys()`、`values()`、`items()` 方法回傳的是視圖 (View) 物件，而不是 `list`。
- 視圖物件類似前面介紹過的序列 (sequence) 型別，也可使用 `for` 迴圈走訪 (下個單元會介紹)、使用 `in` 來檢查成員資格等。
- 當字典的內容有變動，視圖就會動態更新。

```
>>> x = {0: 'zero', 1: 'one'}
>>> my_keys = x.keys()
>>> my_keys
dict_keys([0, 1])
>>> x[2] = 'two'
>>> my_keys
dict_keys([0, 1, 2])
```

del 敘述

`del` 敘述可用來刪除字典中的項目。

```
>>> list(english_to_french.items())  
[('red', 'rouge'), ('blue', 'bleu'), ('green', 'vert')]  
>>> del english_to_french['green']  
>>> list(english_to_french.items())  
[('red', 'rouge'), ('blue', 'bleu')]
```

in 關鍵字

若使用不存在的鍵來讀取字典，會發生錯誤。
可先用 **in** 關鍵字來測試某個鍵是否存在字典中。

```
>>> 'red' in english_to_french
True
>>> 'orange' in english_to_french
False
```


字典的 `get()` 方法

字典的 `get()` 方法讓我們可透過鍵來讀取該字典。
若字典中含有該鍵，則 `get()` 方法會回傳該鍵對應的值。
若字典中沒有該鍵，則回傳 `None`，或是回傳 `get()` 方法的第二個參數值。

```
>>> english_to_french.get('blue')
'bleu'
>>> english_to_french.get('yellow')
>>> print(english_to_french.get('yellow'))
None
>>> english_to_french.get('blue', 'No translation')
'bleu'
>>> english_to_french.get('yellow', 'No translation')
'No translation'
```

字典的 `setdefault()` 方法

字典的 `setdefault()` 方法讓我們可透過鍵來讀取該字典。
若字典中含有該鍵，則 `setdefault()` 方法會回傳該鍵對應的值。
若字典中沒有該鍵，則會自動新增一組鍵值對，其鍵為該鍵，值為 `None`，或是 `setdefault()` 方法的第二個參數值。

```
>>> english_to_french.setdefault('blue')
'bleu'
>>> english_to_french.setdefault('yellow')
>>> english_to_french
{'red': 'rouge', 'blue': 'bleu', 'yellow': None}
>>> english_to_french.setdefault('pink', 'No translation')
'No translation'
>>> english_to_french
{'red': 'rouge', 'blue': 'bleu', 'yellow': None, 'pink': 'No translation'}
```

字典的 `copy()` 方法

字典的 `copy()` 方法會製造該字典的一個淺層拷貝副本。

```
>>> x = {0: 'zero', 1: 'one'}
>>> y = x.copy()
>>> y
{0: 'zero', 1: 'one'}
>>> y[1] = 'first'
>>> y
{0: 'zero', 1: 'first'}
>>> x
{0: 'zero', 1: 'one'}
```

字典的淺層拷貝與深層拷貝

若字典的值包含 list 或字典等可變物件，可能會需要使用深層拷貝。

```
>>> x = {0:[1, 2]}
>>> y = x.copy()
>>> y
{0: [1, 2]}
>>> x
{0: [1, 2]}
>>> y[0][0] = 0
>>> y
{0: [0, 2]}
>>> x
{0: [0, 2]}
```

淺層拷貝

```
>>> import copy
>>> x = {0:[1, 2]}
>>> y = copy.deepcopy(x)
>>> y
{0: [1, 2]}
>>> x
{0: [1, 2]}
>>> y[0][0] = 0
>>> y
{0: [0, 2]}
>>> x
{0: [1, 2]}
```

深層拷貝

字典的 update() 方法

字典的 `update()` 方法會用其他字典的內容來更新該字典。
若鍵為兩個字典共有，其他字典中的值會覆蓋掉該字典的值。
若其他字典中的鍵在該字典中不存在，則會在該字典新增這個鍵值對。
可把 `update()` 方法視為兩個字典的合併。

```
>>> z = {1: 'One', 2: 'Two'}
>>> x = {0: 'zero', 1: 'one'}
>>> x.update(z)
>>> x
{0: 'zero', 1: 'One', 2: 'Two'}
>>> z
{1: 'One', 2: 'Two'}
```

字典相關操作的整理

字典操作	說明	範例
<code>{}</code>	建立一個空字典	<code>x = {}</code>
<code>len()</code>	回傳字典中項目的個數	<code>len(x)</code>
<code>keys()</code>	回傳字典中所有鍵的視圖	<code>x.keys()</code>
<code>values()</code>	回傳字典中所有值的視圖	<code>x.values()</code>
<code>items()</code>	回傳字典中所有鍵與值的視圖	<code>x.items()</code>
<code>del</code>	從字典中刪除一個項目	<code>del x[key]</code>
<code>in</code>	測試某一個鍵是否存在於字典中	<code>'y' in x</code>
<code>get()</code>	如果鍵存在字典中，回傳該對應值；否則回傳 <code>None</code> 或是第二個參數設定的值	<code>x.get('y', None)</code>
<code>setdefault()</code>	如果鍵存在字典中，回傳該對應值；否則回傳 <code>None</code> 或是第二個參數設定的值，並以傳入的引數來新增該鍵值對	<code>x.setdefault('y', None)</code>
<code>copy()</code>	製作一個淺層副本	<code>y = x.copy()</code>
<code>update()</code>	合併兩個字典	<code>x.update(z)</code>

字典操作的完整列表請見 <https://docs.python.org/3/library/stdtypes.html#dict>

課程大綱

1. 字典簡介
2. 字典的操作
3. 字典應用範例：字數統計
4. 什麼資料型別可以當作字典的鍵來使用？
5. 稀疏矩陣
6. 以字典作為快取
7. 字典的效率

字典應用範例：字數統計

```
>>> sample_string = "To be or not to be"
>>> occurrences = {}
>>> for word in sample_string.split():
...     occurrences[word] = occurrences.get(word, 0) + 1
...
>>> for word in occurrences:
...     print("The word", word, "occurs", occurrences[word], \
...           "times in the string")
...
The word To occurs 1 times in the string
The word be occurs 2 times in the string
The word or occurs 1 times in the string
The word not occurs 1 times in the string
The word to occurs 1 times in the string
```


課程大綱

1. 字典簡介
2. 字典的操作
3. 字典應用範例：字數統計
4. 什麼資料型別可以當作字典的鍵來使用？
5. 稀疏矩陣
6. 以字典作為快取
7. 字典的效率

什麼資料型別可以當作字典的鍵來使用？

- 我們已經知道，整數以及字串都可以做為字典的鍵。
- Python 允許以任何**不可變且可雜湊**的物件做為字典的鍵。
- Python 中任何可以修改的物件，都稱為可變物件。
 - list、字典都可以添加、更改、或刪除元素，它們都是可變物件。
 - 數值、字串是不可變物件。
- list 不能做為字典的鍵。
- 若想要使用具有多個元素的物件做為鍵，可以使用 tuple。
 - 例如：人事資料以姓、名做為鍵、地圖資料以經度、緯度做為鍵。

可雜湊 (Hashable)

- 在 Python 中，一個物件是**可雜湊 (hashable)** 的，指的是說該物件具有一個雜湊值 (hash value)，在該物件的生命週期中，此雜湊值不會改變，而且該物件可以與其他物件做比較。
- 雜湊 (Hashing) 是電腦科學中一種對資料的處理方法，通過某種特定的函式/演算法 (稱為雜湊函式/演算法) 把要檢索的項與用來檢索的索引 (稱為雜湊，或者雜湊值) 關聯起來，產生一種便於搜尋的資料結構 (稱為雜湊表)。
- 若想要使用具有多個元素的物件做為鍵，可以使用 tuple，但必須確保 tuple 底下各層的元素都是不可變的。
 - 若 tuple 的元素是可變物件 (例如 list)，只要元素值一改變，該 tuple 的雜湊值就會改變，這樣的 tuple 是不可雜湊的，無法做為字典的鍵。

Python 各種型別是否可做為字典的鍵？

Python 型別	是否為不可變？	是否可雜湊？	是否可當作字典的鍵？
int	是	是	是
float	是	是	是
boolean	是	是	是
complex	是	是	是
str	是	是	是
bytes	是	是	是
bytearray	否	否	否
list	否	否	否
tuple	是	有時可以	有時可以
set	否	否	否
frozenset	是	是	是
dictionary	否	否	否

課程大綱

1. 字典簡介
2. 字典的操作
3. 字典應用範例：字數統計
4. 什麼資料型別可以當作字典的鍵來使用？
5. 稀疏矩陣
6. 以字典作為快取
7. 字典的效率

矩陣

- 在數學術語中，矩陣是二維數字陣列，如下所示：

$$\begin{bmatrix} 3 & 0 & -2 & 11 \\ 0 & 9 & 0 & 0 \\ 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & -5 \end{bmatrix}$$

- 在 Python 中，可以使用兩層的 list 來表示：

```
>>> matrix = [[3, 0, -2, 11], [0, 9, 0, 0], [0, 7, 0, 0], [0, 0, 0, -5]]
```

- 透過列號與行號的索引，可存取矩陣中的元素：

```
>>> matrix[2][1]
7
```

稀疏矩陣

- 在某些應用中，矩陣大部分的元素值是零。為了節省記憶體，這種矩陣通常只儲存非零元素，這種表示法稱為稀疏矩陣 (sparse matrix)。
- 在 Python 中，可使用具有 tuple 索引的字典，來實現稀疏矩陣：

```
>>> matrix = {(0, 0): 3, (0, 2): -2, (0, 3): 11,  
... (1, 1): 9, (2, 1): 7, (3, 3): -5}  
>>> matrix  
{(0, 0): 3, (0, 2): -2, (0, 3): 11, (1, 1): 9, (2, 1): 7, (3, 3): -5}
```

$$\begin{bmatrix} 3 & 0 & -2 & 11 \\ 0 & 9 & 0 & 0 \\ 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & -5 \end{bmatrix}$$

稀疏矩陣的存取

```
>>> rownum = 2
>>> colnum = 1
>>> if (rownum, colnum) in matrix:
...     element = matrix[(rownum, colnum)]
... else:
...     element = 0
...
>>> element
7
```


稀疏矩陣的存取 (續)

稀疏矩陣的另一種存取方式是使用字典的 `get()` 方法。
若 `get()` 無法在字典中找到鍵，則回傳 `0`。

```
>>> rownum = 2
>>> colnum = 1
>>> element = matrix.get((rownum, colnum), 0)
>>> element
7
```

課程大綱

1. 字典簡介
2. 字典的操作
3. 字典應用範例：字數統計
4. 什麼資料型別可以當作字典的鍵來使用？
5. 稀疏矩陣
6. 以字典作為快取
7. 字典的效率

以字典作為快取

假設我們需要一個名為 `sole` 的函式，它需要三個整數參數，並回傳計算後的結果。

```
def sole(m, n, t):  
    #... 將 m, n, t 進行一些計算 ...  
    return(result)
```

若這個函式的計算非常耗時，而且它需要被呼叫很多次，我們可以把計算結果儲存起來，以避免反覆重新計算。

以字典作為快取 (續)

假設程式需要呼叫 `sole(12, 20, 6)` 很多次，我們可以使用 `(12, 20, 6)` 這個 `tuple` 做為鍵，把 `sole(12, 20, 6)` 的計算結果儲存在字典中。只有第一次呼叫 `sole(12, 20, 6)` 需要計算，之後的呼叫直接從字典中取值即可。

```
sole_cache = {}  
def sole(m, n, t):  
    if (m, n, t) in sole_cache:  
        return sole_cache[(m, n, t)]  
    else:  
        #... 否則對 m, n, t 進行一些計算 ...  
        sole_cache[(m, n, t)] = result  
    return result
```

課程大綱

1. 字典簡介
2. 字典的操作
3. 字典應用範例：字數統計
4. 什麼資料型別可以當作字典的鍵來使用？
5. 稀疏矩陣
6. 以字典作為快取
7. 字典的效率

字典的效率

- Python 的字典採用雜湊表 (hash table)，搜尋的平均時間複雜度是 $O(1)$ 。
 - 需要占用記憶體，以空間換取時間。
 - 當鍵的數量增加時，雜湊的碰撞機率也變高。
- Python 的字典 (以及其他資料結構) 經過大量的實現優化，運行速度很快。
- 不需要花時間在意哪一種資料結構效率比較好。
- 如果使用字典比起使用其他資料結構 (例如 list) 能夠更容易、更清楚地解決問題，那就使用字典。

重點整理

	有序	無序	從 Python 3.7 開始， 字典保持鍵建立時的 順序
可變更 (mutable)	list	set	字典
不可變更 (immutable)	tuple、 字串		

	list 串列	dict 字典
可儲存的資料	任何類型的物件	任何類型的物件
索引	連續的整數	以鍵做為索引