

# PYTHON

## CONTAINER: LIST, TUPLE, DICTIONARY, AND SET

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

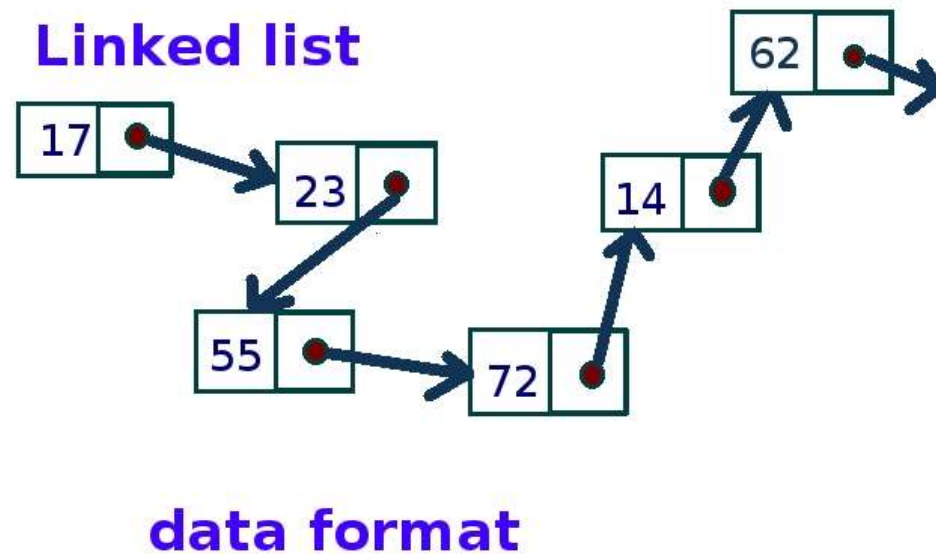


# 串列的使用

---

- 何謂串列 (List)

- 串列 (又稱為「清單」或「列表」)，功能與其他語言的「陣列 (Array)」雷同，但使用方法與變數類似
- 提供儲存一堆資料的記憶體空間。



# 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：

串列變數 = [資料1, 資料2, ..., 資料N]

- 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=["文字1", '數字', '69115908']  
print(list1)  
list2=[True, "somewhat", 69115908]  
print(list2)
```

```
>> ['文字1', '數字', '69115908']  
>> [True, 'somewhat', 69115908]
```

# 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 = [69115908, 'ncku', True]  
print(list1[1])
```

```
>> ncku
```

# 串列宣告

---

- 空串列

- 例如：

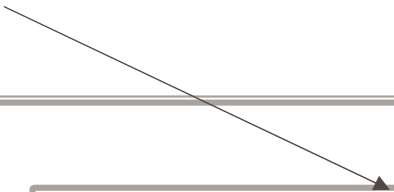
```
list1 = []
```

- 多維串列宣告 (類似表格)

- 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：

```
#Multidimension
list1 = [[0,1], ["ok", "fine"], [69115908, "how"]]
print(list1[1])
print(list1[2][0])
```

```
>> ['ok', 'fine']
>> 69115908
```



```
list1 = [[0,1],
          ["ok", "fine"],
          [69115908, "how"]]
```

# 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：

```
list1 = [69115908, 'ncku', True]  
print(list1[3])
```

```
>> IndexError: list index out of range
```

- 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：

```
print(list1[-1])  
print(list1[-4])
```

```
>> True
```

```
>> IndexError: list index out of range
```

# Practice 1

---

- Try once:

- 輸入：三科不同科目成績存在List中 (科目不一定需要存list)
- 輸出：依序把串列裡面每一個元素列印出來

科目國文分數為94

科目英文分數為87

科目 Python程式設計 分數為 0

# 串列資料存取

---

- 改變串列元素

- 語法為：

串列名稱[索引] = 新的資料

- 如：將 list1 串列中索引為 0 (第 1 個元素) 的元素內容，改成字串。

```
list1 = [True, 'somewhat', 69115908]
list1[0] = "ncku"
list1[1] = list1[0]
print(list1)
```

```
>> ['ncku', 'ncku', 69115908]
```



# 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：

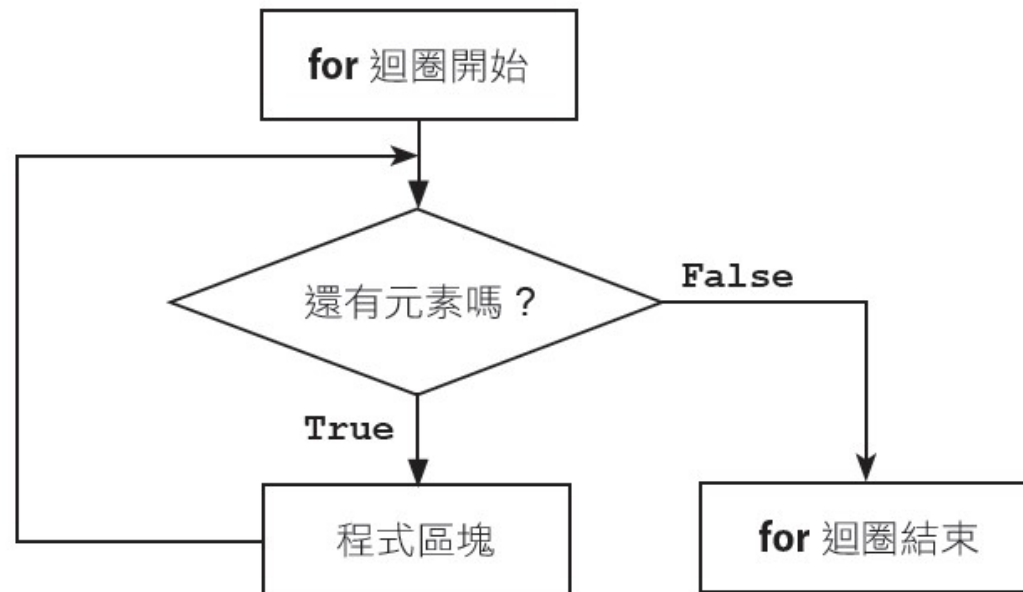
```
list1=[11, 69115908, 22]  
for s in list1:  
    print(s, end=',')
```

```
>> 11,69115908,22,
```

# 連續串列存取

---

- for迴圈的運作流程如下 (關乎邏輯)



## Practice 2

---

- 練習一下如何利用 For loop + List的資料型態，一次列印出全部資料
  - 輸入：內建一個 list然後裡面包含三種不同元素
  - 輸出：利用 for-loop 列印出裡面的值

```
國文 type= <class 'str'>  
69115908 type= <class 'int'>  
True tvpe= <class 'bool'>
```

## 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。

```
list1=['國文', 69115908, True]  
print(len(list1))    # 顯示3
```

- 以 for in range 迴圈讀取串列

```
for idx in range(len(list1)):  
    print(list1[idx])
```

```
>> 國文  
>> 69115908  
>> True
```

## Practice 3

---

- 請同學建立 subjects 和 scores 串列，每個串列包含三個元素，即：subjects=["國文","數學","英文"]、scores = [85, 79, 93]，請以 for 迴圈顯示subjects 和scores 串列的元素

☞ 科目 國文之分數: 85  
科目 數學之分數: 79  
科目 英文之分數: 93

# 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
```

## Practice 4

---

- 請設計一個輸入三個成績的程式，並且存到List裡面，最後印出總分與平均
- (optional) 請設計一個輸入成績的程式，將學生成績存入串列做為串列元素，如果輸入「-1」表示成績輸入結束，最後顯示班上總成績及平均成績。

請輸入學生分數:100

請輸入學生分數:40

請輸入學生分數:60

請輸入學生分數:-1

共 3 位學生，總分 200 平均 66

# 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]
```

# Practice

---

- 設計一個練習刪除的程式，並顯示剩下的元素，如果輸入 Enter 就結束輸入。
  - 輸入：內建一個五個元素的 List
  - 輸出：持續詢問什麼時候要刪除什麼，若找不到，就輸出“不在串列中”，若找到便刪除
- (optional) 輸入：手動連續輸入，不輸入並按下enter即為停止輸入
- (optional) 輸出：輸入完立即持續詢問什麼時候要刪除什麼，若找不到，就輸出“不在串列中”，若找到便刪除、若串列已經沒有元素程式也會結束、或是什麼都不打按下Enter也會結束

```
請輸入data:a
請輸入data:b
請輸入data:c
請輸入data:100
請輸入data:
你的資料 ['a', 'b', 'c', '100']
請輸入要刪除哪筆資料b
經刪除後，資料為: ['a', 'c', '100']
請輸入要刪除哪筆資料10
找不到阿，開什麼玩笑!
請輸入要刪除哪筆資料100
經刪除後，資料為: ['a', 'c']
請輸入要刪除哪筆資料a
經刪除後，資料為: ['c']
請輸入要刪除哪筆資料c
經刪除後，資料為: []
已經空了
```

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

## 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



TUPLE: 快一點的 LIST

# 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。

## 重點整理

---

- 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」。



---

---

# DICTIONARY

---

---

# 字典基本操作

---

- Python 中「字典」資料型態與國語字典結構類似，其元素是以「鍵- 值」對方式儲存，運作方式為利用「鍵」來取得「值」。
  - 建立字典
  - 字典的結構，其元素是以「鍵- 值」對方式儲存，這樣就可使用「鍵」來取得「值」。有多種方式可以建立字典，第一種方式為將元素置於一對大括號「{ }」中，其語法為：

```
字典名稱 = { 鍵 1: 值 1, 鍵 2: 值 2, …… }
```

- 字串、整數、浮點數等皆可做為「鍵」，但以字串做為「鍵」的情況最多。

## 字典基本操作

---

- 建立字典的第二種方式是使用 dict 函式，再將鍵- 值對置於中括號「[]」中，語法為：

```
字典名稱 = dict([[ 鍵 1, 值 1], [ 鍵 2, 值 2], ……])
```

- 建立字典的第三種方式也是使用 dict 函式，只要將鍵與值以等號連接起來即可，語法為：

```
字典名稱 = dict( 鍵 1= 值 1, 鍵 2= 值 2, ……)
```

- 第三種建立字典的方式相當簡潔，但特別注意此種方式建立的字典「鍵」不能使用數值，否則執行時會產生錯誤，例如：

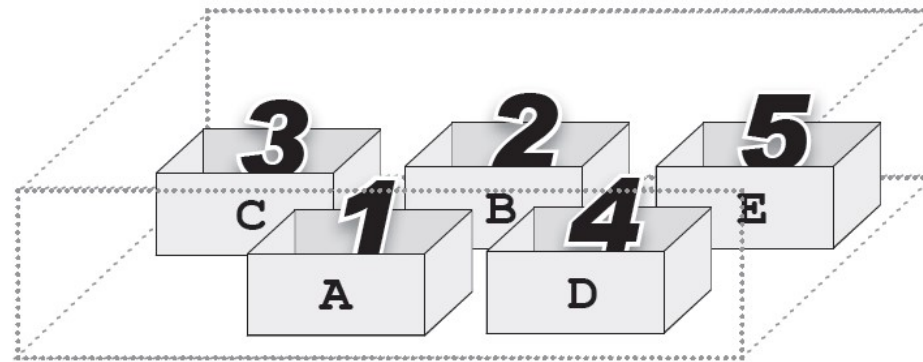
```
dict1 = dict(1=" 林大明 ", 3=" 李美麗 ", 5=" 陳品言 ")  
print(dict1[3])    # 錯誤，字典取值的說明參考下一小節
```

# 字典存取資料

---

- 字典取值

- 字典，與串列最大的不同在於串列元素在記憶體中是依序排列，而字典元素則是隨意放置，沒有一定順序。



- 基本取值方式

- 取得字典元素值的方法是以「鍵」做為索引來取得「值」，語法為：

字典名稱 [ 鍵 ]

# 字典存取資料

---

- 當字典的鍵重複時

- 字典是使用「鍵」做為索引來取得「值」，所以「鍵」必須是唯一，而「值」可以重覆。如果「鍵」重覆的話，則前面的「鍵」會被覆蓋，只有最後的「鍵」有效，例如：

```
dict2 = {"香蕉":20, "蘋果":50, "橘子":30, "香蕉":25}  
print(dict2["香蕉"])    #25, 「"香蕉":20」被覆蓋
```



# 字典存取資料

---

- 當字典的鍵不存在時

- 元素在字典中的排列順序是隨機的，與設定順序不一定相同，例如：

```
dict1 = {"香蕉":20, "蘋果":50, "橘子":30}
```

```
print(dict1)    # 結果：{"蘋果":50, "香蕉":20, "橘子":30}
```

- 由於元素在字典中的排列順序是隨機的，所以不能以位置數值做為索引。另外，若輸入的「鍵」不存在也會產生錯誤，此種字典取值方式當「鍵」不存在時會因錯誤而讓程式中斷，因此 Python 另外提供了 `get` 方法可以取得字典元素值，即使「鍵」不存在也不會產生錯誤，語法為：

```
字典名稱.get(鍵[, 預設值])
```

## 字典存取資料

---

- 預設值可有可無。根據是否有傳入預設值及「鍵」是否存在可分為四種情形：

字典名稱.get( 鍵 [, 預設值 ] )

| 預設值狀況   | 「鍵」是否存在 | 返回值            |
|---------|---------|----------------|
| 沒有傳入預設值 | 「鍵」存在   | 返回鍵對應的值        |
|         | 「鍵」不存在  | 返回 <b>None</b> |
| 有傳入預設值  | 「鍵」存在   | 返回鍵對應的值        |
|         | 「鍵」不存在  | 返回預設值          |

# 字典的修改

---

- 字典維護

- 修改字典

- 修改字典元素值與在字典中新增元素的語法相同：

字典名稱 [ 鍵 ] = 值

- 如果「鍵」存在就是修改元素值，新元素值會取代舊元素值，如果「鍵」不存在就是新增元素。

# 字典的修改

---

- 刪除字典

- 刪除字典則有三種情況。第一種是刪除字典中特定元素，語法為：

```
del 字典名稱 [ 鍵 ]
```

- 第二種是刪除字典中所有元素，語法為：

```
字典名稱 .clear()
```

- 第三種是刪除字典，字典刪除後該字典就不存在，語法為：

```
del 字典名稱
```

# 字典進階操作

## ■ 字典進階功能整理

- 字典常用的進階功能整理於下表：( 表中 dict1={"joe":5,"mary":8}，n 為整數，b 為布林變數)

| 方法                        | 意義                                  | 範例及結果                                               |
|---------------------------|-------------------------------------|-----------------------------------------------------|
| len(dict1)                | 取得字典元素個數                            | n=len(dict1)<br>n=2                                 |
| dict1.copy()              | 複製字典                                | dict2=dict1.copy()<br>dict2={"joe":5, "mary":8}     |
| 鍵 in dict1                | 檢查「鍵」是否存在                           | b="joe" in dict1<br>b=True                          |
| dict1.items()             | 取得以「鍵 - 值」組為元素的組合                   | item1=dict1.items()<br>dict2=[("joe":5),("mary":8)] |
| dict1.keys()              | 取得以「鍵」為元素的組合                        | key1=dict1.keys()<br>key1=["joe", "mary"]           |
| dict1.setdefault( 鍵 , 值 ) | 與 get() 類似，若「鍵」不存在就以參數的「鍵 - 值」建立新元素 | n=dict1.setdefault("joe")<br>n=5                    |
| dict1.values()            | 取得以「值」為元素的組合                        | value1=dict1.values()<br>value1=[5,8]               |

# 字典Key檢查

---

- in 功能

- 許多字典功能傳送「鍵」做為參數時，若做為參數的「鍵」不存在就會產生錯誤而讓程式中斷執行。in 功能會檢查字典中的「鍵」是否存在，語法為：

鍵 in 字典名稱

- 如果「鍵」存在就傳回 True，「鍵」不存在就傳回 False。
- in 功能可在執行如果「鍵」不存在就會產生錯誤的程式之前進行檢查，確定「鍵」存在才執行該程式。

## keys 及 values 功能

---

- 字典的 `keys()` 功能可取得字典中所有「鍵」，資料型態為 `dict_keys`，雖然 `dict_keys` 資料型態看起來像串列，但它不能以索引方式取得元素值：

```
dict1 = {"香蕉":20, "蘋果":50, "橘子":30}
key1 = dict1.keys()
print(key1[0])    # 產生錯誤，不能以索引方式取得元素值
```

- 必須將 `dict_keys` 資料型態以 `list` 函式轉換為串列才能取得元素值：

```
dict1 = {"香蕉":20, "蘋果":50, "橘子":30}
key1 = list(dict1.keys())
print(key1[0])    # 香蕉
```

# Keys and Values

---

- 注意的是，一定要把Keys 跟 Values 轉換成 List，不然直接存取會有問題

`TypeError: 'dict_keys' object is not subscriptable`

- `values()` 功能可取得字典中所有「值」，資料型態為 `dict_values`。  
`dict_values` 資料型態的用法與 `dict_keys` 完全相同，不再贅述。
- 如果同時要對兩個 (keys, values) 進行存取如何做？
  - Recall: `len()` function + `range`，然後利用index分別對keys/values做存取



## 更多存取功能

---

- items 功能

- items() 功能可同時取得所有「鍵- 值」組成的組合，資料型態為 dict\_items。
- 將 dict\_items 資料型態以 list 函式轉換為串列後相當於二維串列，可以取得個別元素值。

```
items[i][0], items[i][1]
```

- 分別會顯示第  $i$  個key ( $[i][0]$ ) 以及其 Value ( $[i][1]$ ) 的結果

## 更多存取功能

---

- 兩個以上的List若有相同長度，一樣可以用單一for Loop來搞定
  - Zip function: `zip(串列1, 串列2,..)`
  - 搭配 For loop, we have

```
for x, y in zip(list1, list2):
```

- 其中 x, y 分別會在同一個index中從 list1, list2 取出元素並放到x, y中
- 哦? 那這麼說來 items 不就是原生兩個 list的意思? (二維list)
  - 所以也可以

```
for x, y in dict1.items():
```

## setdefault 功能：取值/新增

- setdefault 功能的使用方式、功能及傳回值與 get 功能雷同，不同處在於是否改變字典的內容。get 功能不會改變字典的內容；setdefault 功能可能改變字典的內容。
- setdefault 功能的語法為：

```
字典名稱.setdefault( 鍵 [, 預設值 ] )
```

- 預設值可有可無。根據是否有傳入預設值及「鍵」是否存在可分為四種情形：

| 預設值狀況   | 「鍵」是否存在 | 返回值     | 字典            |
|---------|---------|---------|---------------|
| 沒有傳入預設值 | 「鍵」存在   | 返回鍵對應的值 | 沒有改變          |
|         | 「鍵」不存在  | 返回 None | 加入元素「鍵 :None」 |
| 有傳入預設值  | 「鍵」存在   | 返回鍵對應的值 | 沒有改變          |
|         | 「鍵」不存在  | 返回預設值   | 加入元素「鍵：預設值」   |

## 重點整理

---

- Dictionary 有下列的重點：

- 建立字典的第一種方式為將元素置於一對大括號「{ }」中，其語法為：

```
字典名稱 = { 鍵 1: 值 1, 鍵 2: 值 2, ..... }
```

- 建立字典的第二種方式是使用 dict 函式，再將鍵 - 值對置於中括號「[]」中，語法為：

```
字典名稱 = dict([ [ 鍵 1, 值 1 ], [ 鍵 2, 值 2 ], ..... ])
```

- 建立字典的第三種方式也是使用 dict 函式，只要將鍵與值以等號連接起來即可，語法為：

```
字典名稱 = dict( 鍵 1= 值 1, 鍵 2= 值 2, ..... )
```

- 取得字典元素值的方法是以「鍵」做為索引來取得「值」，語法為：

```
字典名稱 [ 鍵 ]
```

## 重點整理

---

- `get` 可以取得字典元素值，即使「鍵」不存在也不會產生錯誤，語法為：

```
字典名稱.get(鍵[, 預設值])
```

- 修改字典元素值與在字典中新增元素的語法相同，語法為：

```
字典名稱[鍵] = 值
```

- `in` 功能會檢查字典中的「鍵」是否存在，語法為：

```
鍵 in 字典名稱
```

- 字典的 `keys()` 功能可取得字典中所有「鍵」，`values()` 功能可取得字典中所有「值」，`items()` 功能可同時取得所有「鍵- 值」組成的組合。

- `setdefault` 功能的語法為：

```
字典名稱.setdefault(鍵[, 預設值])
```