

Chapter 7

TUPLE, LIST, SET, AND DICTIONARY

序對 (Tuple)

要怎麼宣告一個序對 (tuple) 變數：使用 “ () ” (小括號) 來放入資料

tuple 裡的資料不能**做任何修改**

tuple 是從 0 開始索引，可用 “ [] ” (中括號) 來取出特定 index 值的資料

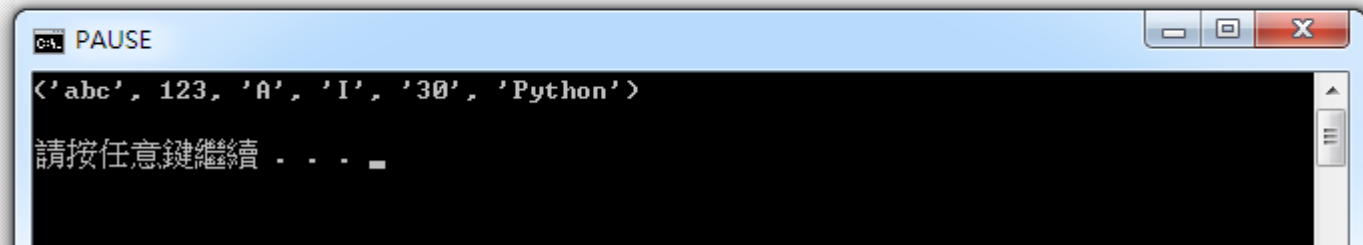
如果一個 tuple 不為空，那第一個元素必定為 `tuple_name [0]`

當然了，我們也可以用負數索引，則最後一個元素可以是 `tuple_name [-1]`

序對 (Tuple)

tuple 裡的資料可以是任何資料型態，沒有規定一定要統一格式

```
icy = ( "abc", 123, 'A', 'I', "30", "Python" )  
  
print(icy)
```



序對 (Tuple)

tuple 裡的資料也可以用索引來找值或是使用 **len** 函數來計算有多少元素

```
icy = ( "abc", 123, 'A', 'I', "30", "Python" )
```

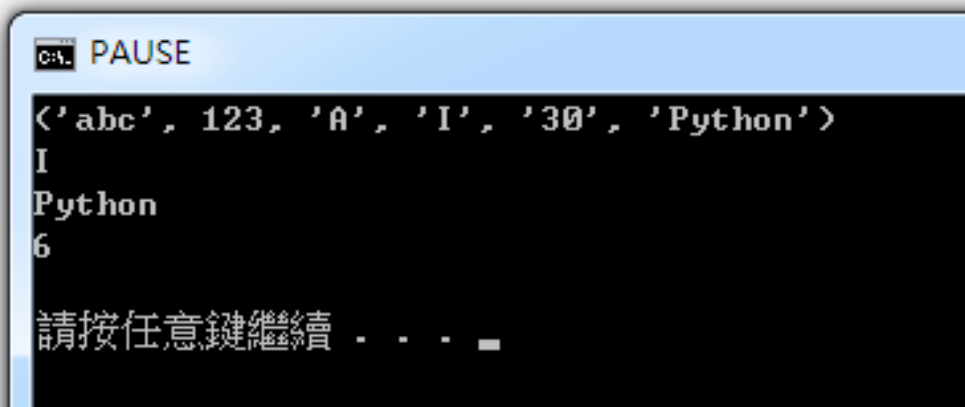
```
#print(icy)
```

```
print(icy[:])
```

```
print(icy[3])
```

```
print(icy[5])
```

```
print(len(icy))
```



The screenshot shows a Python interpreter window with a blue title bar and a black background. The title bar contains a small icon and the text "PAUSE". The window displays the output of the code: the tuple elements are printed as a list of strings, the element at index 3 is printed as a string, the element at index 5 is printed as a string, and the length of the tuple is printed as an integer. The prompt "請按任意鍵繼續" is visible at the bottom.

```
<'abc', 123, 'A', 'I', '30', 'Python'>
I
Python
6
請按任意鍵繼續 . . .
```

序對 (Tuple)

tuple 裡的索引也可以使用負數索引

```
print(icy[:])  
  
#print(icy[3])  
  
#print(icy[5])  
  
print(icy[-1::-1])
```

PAUSE

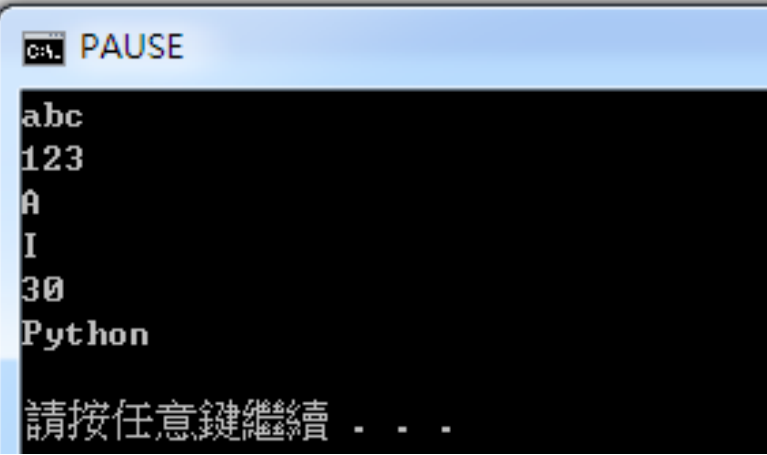
```
<'abc', 123, 'A', 'I', '30', 'Python'>  
<'Python', '30', 'I', 'A', 123, 'abc'>
```

請按任意鍵繼續 . . .

序對 (Tuple)

tuple 裡的資料也可以使用迴圈印出所有元素

```
icy = ( "abc", 123, 'A', 'I', "30", "Python" )  
  
for i in icy :  
    print(i)
```

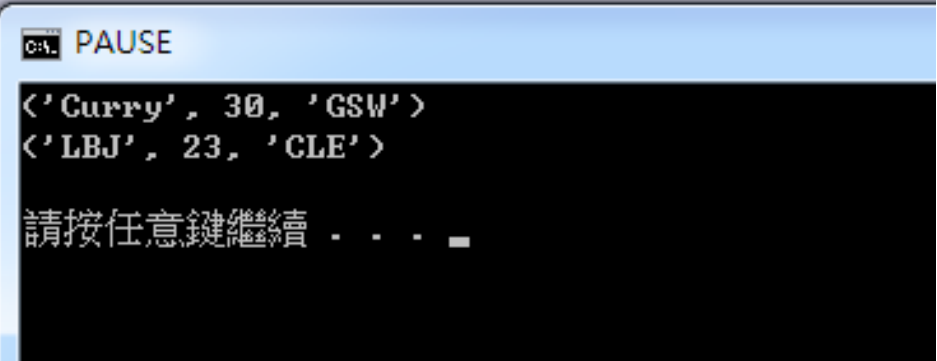


The screenshot shows a Python interpreter window with a blue title bar. The window title is "PAUSE". The output of the code is displayed in a black area with white text, showing the elements of the tuple: "abc", 123, A, I, 30, and Python. At the bottom of the window, there is a prompt "請按任意鍵繼續 . . ." (Press any key to continue . . .).

序對 (Tuple)

tuple 裡的資料也可以用類似集合的方式儲放

```
icy = ( ("Curry", 30, "GSW") , ("LBJ", 23, "CLE") )  
  
for i in icy :  
    print(i)
```



The image shows a Python code snippet and its execution output. The code defines a tuple named 'icy' containing two elements: ("Curry", 30, "GSW") and ("LBJ", 23, "CLE"). A for loop iterates over this tuple, printing each element. The output shows the two tuples as they are stored in memory, with single quotes around the strings. The console also displays a 'PAUSE' message and a prompt to press any key to continue.

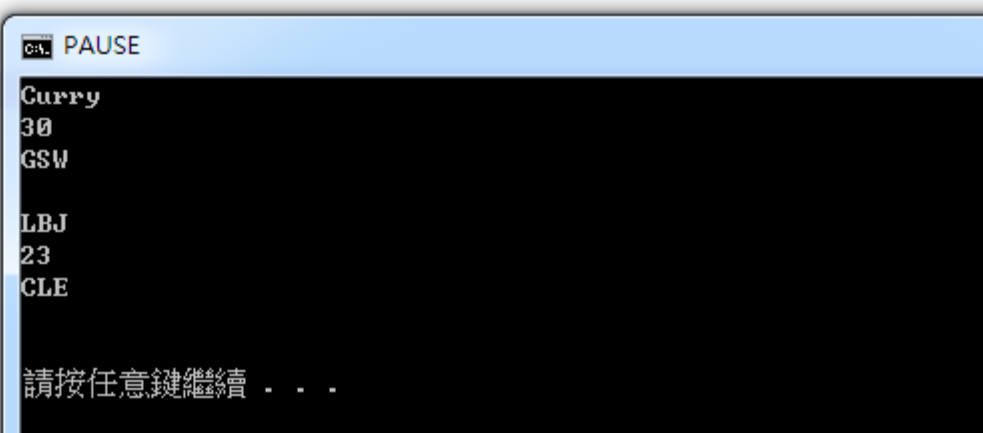
```
>>> PAUSE  
<'Curry', 30, 'GSW'>  
<'LBJ', 23, 'CLE'>  
請按任意鍵繼續 . . . _
```

序對 (Tuple)

tuple 裡的成對的資料也可以逐一取出，

```
icy = ( ("Curry", 30, "GSW") , ("LBJ", 23, "CLE") )

#icy[1] = ("Jordan", 23, "CHI")
for i, c, y in icy :
    print(i)
    print(c)
    print(y)
    print("")
```



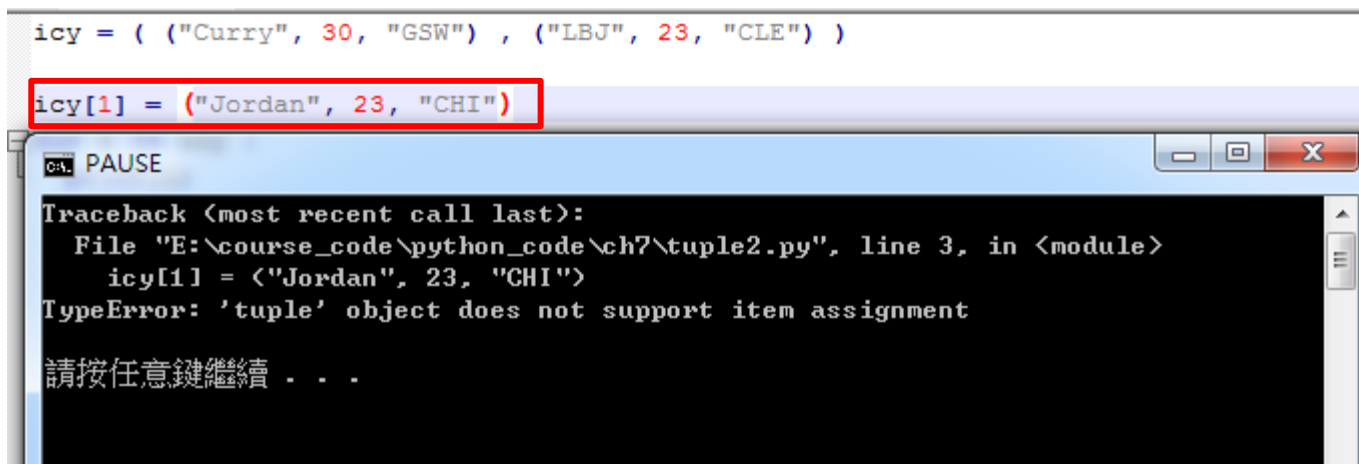
Curry
30
GSW

LBJ
23
CLE

請按任意鍵繼續 . . .

序對 (Tuple)

Note. tuple 裡的資料 ***** ”是不能改變的” *****



The screenshot shows a Python IDE with a script editor and a console window. In the script editor, the following code is visible:

```
icy = ( ("Curry", 30, "GSW") , ("LBJ", 23, "CLE") )  
icy[1] = ("Jordan", 23, "CHI")
```

The second line, `icy[1] = ("Jordan", 23, "CHI")`, is highlighted with a red rectangular box. Below the script editor, the console window displays a traceback error:

```
PAUSE  
Traceback (most recent call last):  
  File "E:\course_code\python_code\ch7\tuple2.py", line 3, in <module>  
    icy[1] = ("Jordan", 23, "CHI")  
TypeError: 'tuple' object does not support item assignment  
請按任意鍵繼續 . . .
```

序對 (Tuple)

tuple 是一個很簡單的資料型態，由於不能修改資料及順序

基本上的用法單純的用來搜尋，所以適合用來儲存不太會變動的資料

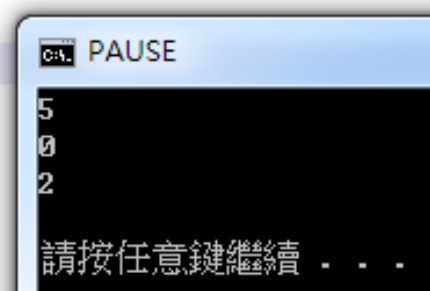
所以 tuple 可以使用的函數就少少的 2 個：**index** 以及 **count**

index(element, start, end)	回傳 元素第一次出現的索引值 start, end 是尋找的起點跟終點
count(element)	計算元素出現的次數

序對 (Tuple)

來看看 index, count 要怎麼使用囉

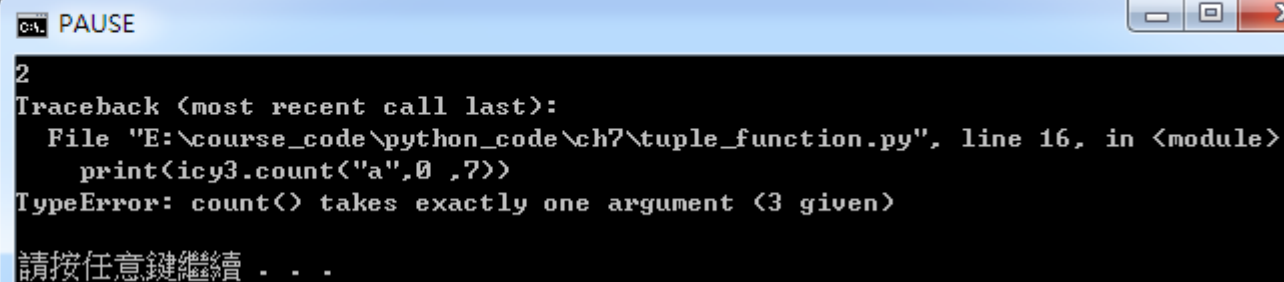
```
icy = ( "abc", 123, 'A', 'I', "30", "Python" )  
  
icy2 = ( ("Curry", 30, "GSW") , ("LBJ", 23, "CLE") )  
  
icy3 = ( "a" , "b" , "c" , "d" , "c" , "b", "a" )  
  
print(icy.index("Python"))  
  
print(icy2.index( ("Curry", 30, "GSW") ))  
  
print(icy3.count("a"))
```



序對 (Tuple)

序對的 count 不能設起點跟終點

```
icy3 = ( "a" , "b" , "c" , "d" , "c" , "b" , "a" )  
icy4 = "an apple a day, keeps a doctor a way "  
print(icy4.count("a",0 ,7))  
print(icy3.count("a",0 ,7))
```



The screenshot shows a Python interpreter window with a blue title bar and standard window controls. The main area is black with white text. It displays a 'Traceback' error for a 'TypeError: count() takes exactly one argument (3 given)'. The error message indicates the problem is in 'tuple_function.py' at line 16, specifically in the 'print' statement for 'icy3'. The prompt '請按任意鍵繼續 . . .' is visible at the bottom.

```
PAUSE  
2  
Traceback (most recent call last):  
  File "E:\course_code\python_code\ch7\tuple_function.py", line 16, in <module>  
    print(icy3.count("a",0 ,7))  
TypeError: count() takes exactly one argument (3 given)  
請按任意鍵繼續 . . .
```

序對 (Tuple)

Unpacking (把 tuple 轉字串)

```
CCF = ("52","123")
print(type(CCF))

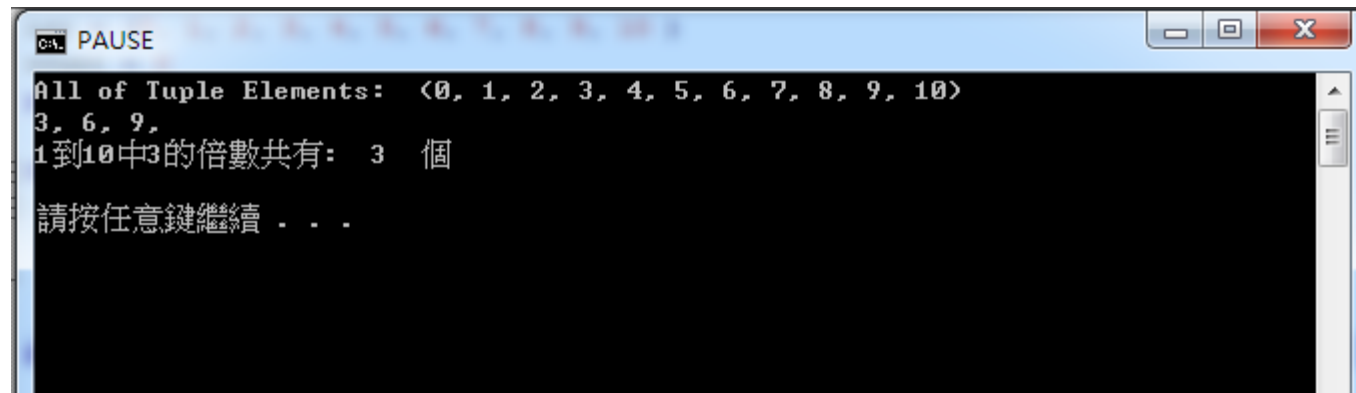
five, two = CCF
print(five)
print(two)
print(type(five))
```

PAUSE

```
<class 'tuple'>
52
123
<class 'str'>
請按任意鍵繼續 . . .
```

隨堂練習

1. 宣告一個 tuple ex. (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
2. 寫一個迴圈去印出 tuple 中是 3 的倍數的值



```
PAUSE
All of Tuple Elements: <0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10>
3, 6, 9,
1到10中3的倍數共有: 3 個
請按任意鍵繼續 . . .
```

串列 (List)

要怎麼宣告一個串列 (list) 變數：使用 “ [] ” (中括號) 來放入資料

List 裡的資料**可以改變**

List 是從 0 開始索引，可用 “ [] ” (中括號) 來取出特定 index 值的資料

如果一個 List 不為空，那第一個元素必定為 **List_name [0]**

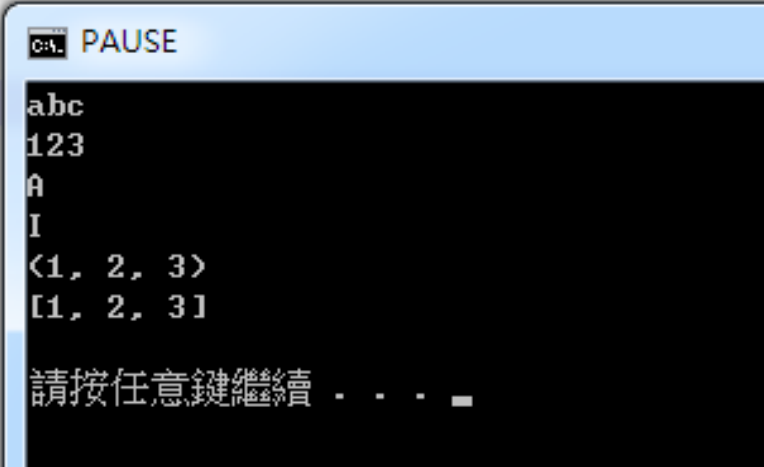
當然了，我們也可以用負數索引，則最後一個元素可以是 **List_name [-1]**

串列 (List)

List 裡的資料可以是任何資料型態，沒有規定一定要統一格式

Note : list 裡面可以有 tuple 資料，也可以有 list or tuple 的資料哦

```
icy = [ "abc", 123, 'A', 'I', (1,2,3) , [1,2,3] ]  
for i in icy :  
    print(i)
```



```
C:\> PAUSE  
abc  
123  
A  
I  
(1, 2, 3)  
[1, 2, 3]  
請按任意鍵繼續 . . .
```


串列 (List)

List 裡的資料也可以用索引來找值或是使用 **len** 函數來計算有多少元素

```
icy = [ "abc", 123, 'A', 'I', (1,2,3) , [1,2,3] ]  
  
print(icy[0])  
print(icy[3])  
print(icy[5])  
print(len(icy))
```

PAUSE

abc

I

[1, 2, 3]

6

請按任意鍵繼續 . . .

串列 (List)

List 裡的索引也可以使用負數索引

```
icy = [ "abc", 123, 'A', 'I', (1,2,3) , [1,2,3] ]
```

```
print(icy[0])  
print(icy[3])  
print(icy[5])  
print(len(icy))
```

```
print(icy[-1::-1])  
print(icy[-1:-3:-1])
```

PAUSE

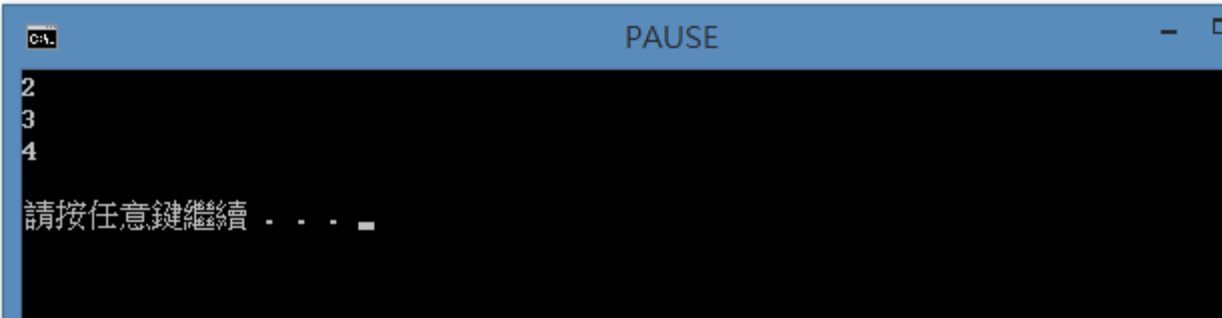
```
abc  
I  
[1, 2, 3]  
6  
[[1, 2, 3], <1, 2, 3>, 'I', 'A', 123, 'abc']  
[[1, 2, 3], <1, 2, 3>]
```

請按任意鍵繼續 . . .

串列 (List)

如果 List 不要重頭開始輸出要怎麼處理

```
list = [1,2,3,4]
for i in list[1::] :
    print (i)
```



2
3
4
請按任意鍵繼續 . . .

串列 (List)

和 Tuple 資料型態不同的是，List 裡的資料 ***** ”是可以改變的” *****

```
icy = [ "abc", 123, 'A', 'I', (1,2,3) , [1,2,3] ]
```

```
print(icy)
```

```
icy [0] = "Allen"
```

```
icy [1] = "Iverson"
```

```
icy [2] = "The Answer"
```

```
icy [3] = 3
```

```
print("\n", icy)
```

Ctrl PAUSE

```
['abc', 123, 'A', 'I', <1, 2, 3>, [1, 2, 3]]
```

```
['Allen', 'Iverson', 'The Answer', 3, <1, 2, 3>, [1, 2, 3]]
```

請按任意鍵繼續 . . .

串列 (List)

List 可以修改元素資料、改變元素順序及新增刪除的資料型態
所以… 可使用的函數就比 tuple 多了

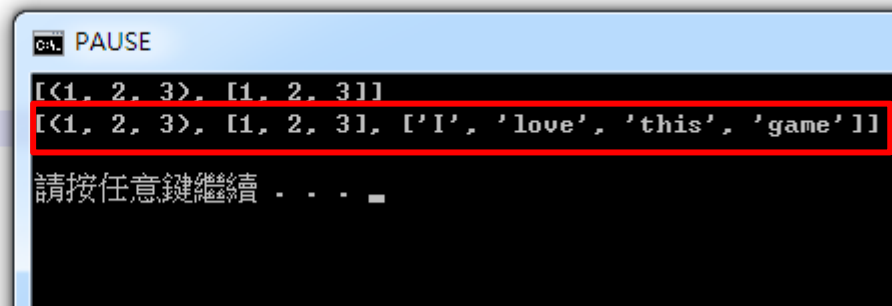
函數	描述
List.append()	在 List 的最後面加入物件
List.extend()	在 List 的最後面加入 List
List.insert()	將元素插入 List
List.remove(X)	刪除 List 裡第一個符合 X 的元素
List.pop()	刪除並回傳 List 裡最後一個元素

串列 (List)

List.append()

將物件 (如，整數、浮點數、小數…等) 插入在整個 List 的最後面

```
icy2 = [ (1,2,3) , [1,2,3] ]  
  
icy3 = [ "I", "love", "this", "game" ]  
  
print(icy2)  
  
icy2.append(icy3)  
  
print(icy2)
```



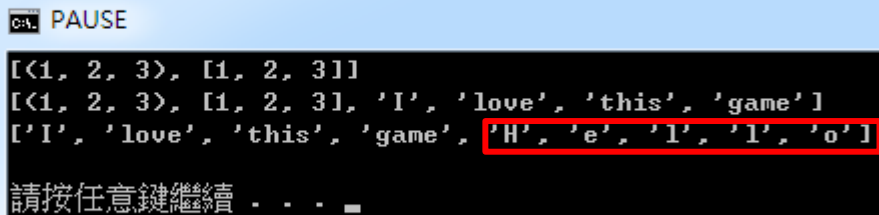
```
C:\ PAUSE  
[(1, 2, 3), [1, 2, 3]]  
[(1, 2, 3), [1, 2, 3], ['I', 'love', 'this', 'game']]  
請按任意鍵繼續 . . .
```

串列 (List)

List.extend()

跟 append() 很像，但是只能插入有順序的物件 (如 tuple, string, or list)

```
icy2 = [ (1,2,3) , [1,2,3] ]  
  
icy3 = [ "I", "love", "this", "game" ]  
  
print(icy2)  
  
icy2.extend(icy3)  
  
print(icy2)  
  
icy4 = "Hello"  
  
icy3.extend(icy4)  
  
print(icy3)
```



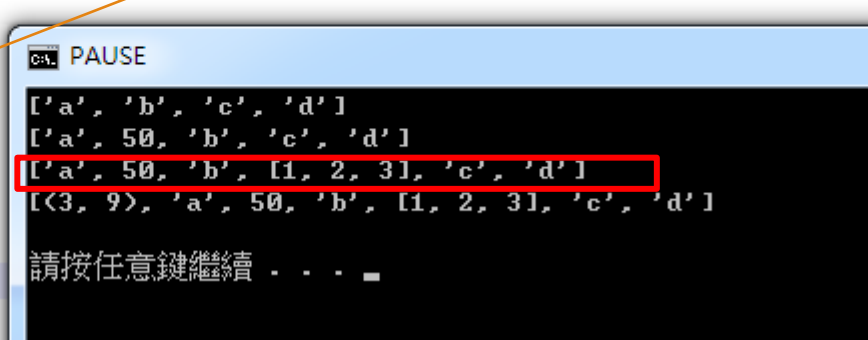
```
PAUSE  
[<1, 2, 3>, [1, 2, 3]]  
[<1, 2, 3>, [1, 2, 3], 'I', 'love', 'this', 'game']  
['I', 'love', 'this', 'game', 'H', 'e', 'l', 'l', 'o']  
請按任意鍵繼續 . . .
```

串列 (List)

List.insert()

insert() 需要兩個參數，分別是 **index(索引位置)** 跟物件資料 (**任意資料型態皆可**)

```
icy3 = [ "a", "b", "c", "d" ]  
  
icy4 = 50  
  
icy5 = [1,2,3]  
  
icy6 = (3,9)  
  
print(icy3)  
icy3.insert(1, icy4)  
print(icy3)  
icy3.insert(3, icy5)  
print(icy3)  
icy3.insert(0, icy6)  
print(icy3)
```



```
PAUSE  
['a', 'b', 'c', 'd']  
['a', 50, 'b', 'c', 'd']  
['a', 50, 'b', [1, 2, 3], 'c', 'd']  
[(3, 9), 'a', 50, 'b', [1, 2, 3], 'c', 'd']  
請按任意鍵繼續 . . .
```


串列 (List)

List.insert()

index(索引位置): 如果 index 值超過現有範圍的時，只會將物件放在最後

```
icy7 = [ 1 , 2 , 3 ]  
icy7.insert(5, 5)  
print(icy7)  
icy7.insert(5, 4)  
print(icy7)
```

PAUSE

```
[1, 2, 3, 5]  
[1, 2, 3, 5, 4]
```

請按任意鍵繼續 . . .

串列 (List)

List.remove()

可使用 index 或物件資料來刪除資料

```
icy = [ "abc", 123, 'A', 'I', (1,2,3) , [1,2,3] ]  
  
print(icy)  
  
icy.remove('A')  
print(icy)  
  
icy.remove(icy[0])  
print(icy)  
  
icy.remove(icy[3])  
  
icy.remove((1,2,3))  
print(icy)
```

PAUSE

```
['abc', 123, 'A', 'I', (1, 2, 3), [1, 2, 3]]  
['abc', 123, 'I', (1, 2, 3), [1, 2, 3]]  
[123, 'I', (1, 2, 3), [1, 2, 3]]  
[123, 'I']
```

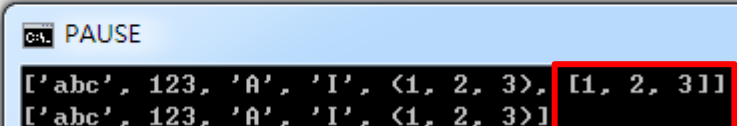
請按任意鍵繼續 . . .

串列 (List)

List.pop()

pop 是拿掉最後一個元素

```
icy = [ "abc", 123, 'A', 'I', (1,2,3) , [1,2,3] ]  
print(icy)  
  
icy.pop()  
print(icy)
```



The image shows a screenshot of a Python IDLE console. The code defines a list 'icy' with elements: "abc", 123, 'A', 'I', (1,2,3), and [1,2,3]. It prints the list, then calls 'icy.pop()' to remove the last element, and prints the list again. The output shows the list after removal: ['abc', 123, 'A', 'I', <1, 2, 3>]. The last element, [1, 2, 3], is highlighted with a red box in the original image.

串列 (List)

List.pop()

pop 也可以指定 index 位置來拿元素

```
icy = [ "abc", 123, 'A', 'I', (1,2,3) , [1,2,3] ]  
print(icy)
```

```
icy.pop()  
print(icy)
```

```
icy.pop(2)  
print(icy)
```

PAUSE

```
['abc', 123, 'A', 'I', <1, 2, 3>, [1, 2, 3]]  
['abc', 123, 'A', 'I', <1, 2, 3>]  
['abc', 123, 'I', <1, 2, 3>]
```

請按任意鍵繼續 . . .

串列 (List)

其他方法

函數	描述
List.clear()	清空 list
List.index(x)	回傳 list 中 x 元素的 index 值
List.count(x)	計算 list 中 x 出現次數
List.sort()	將 list 的元素做排序
List.reverse()	將 list 的元素順序顛倒 (反轉)
List.copy()	複製 list 的所有元素

串列 (List)

List.clear()

清空 list 的所有資料

```
icy = [ "abc", 123, 'A', 'I', (1,2,3) , [1,2,3] ]  
  
print(icy)  
  
icy.clear()  
  
print(icy)
```

PAUSE

['abc', 123, 'A', 'I', (1, 2, 3), [1, 2, 3]]
[]

請按任意鍵繼續 . . .

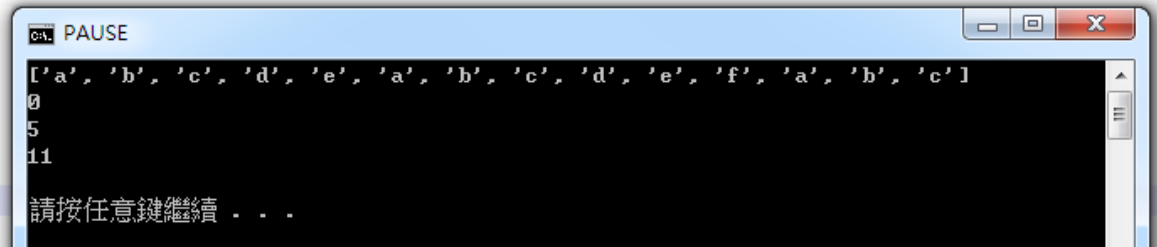
串列 (List)

List.index(X)

index() 會將 list 中符合 x 的第一個元素的索引值回傳

index() 另外有二個選擇參數，分別是 start(索引起點) 跟 end(索引終點)

```
icy = [ "a", "b", "c", "d", "e", "a", "b", "c", "d", "e", "f", "a", "b", "c" ]  
  
print(icy)  
  
print(icy.index("a"))    #回傳第一個出現a的索引值  
print(icy.index("a", 3)) #從icy[3] 後開始找"a"  
print(icy.index("a", 7, 13)) #從icy[4] ~ 找到 icy[13]
```



```
PAUSE  
['a', 'b', 'c', 'd', 'e', 'a', 'b', 'c', 'd', 'e', 'f', 'a', 'b', 'c']  
0  
5  
11  
請按任意鍵繼續 . . .
```

串列 (List)

List.count(x)

count 資料用 **index** 或物件資料，來計算某資料出現了幾次

```
icy = [ "a", "b", "c", "d", "e", "a", "b", "c", "d", "e", "f", "a", "b", "c" ]
```

```
a = icy.count("a")  
print("a: ", a)
```

```
b = icy.count(icy[3])  
print(icy[3], ": ", b)
```

C:\ PAUSE

a: 3
d : 2

請按任意鍵繼續 . . .

串列 (List)

List.sort()

將 list 所有資料排序

note. 如果使用字串，則會比較第一個字母的值

```
icy = [ "a", "b", "c", "d", "e", "a", "b", "c", "d", "e", "f", "a", "b", "c" ]  
print(icy)
```

```
icy.sort()  
print(icy)
```

```
print("")
```

```
icy2 = [3, 9, 3, 9, 8, 8, 9, 1, 2, 3, 4, 5, 6, 7]  
print(icy2)
```

```
icy2.sort()  
print(icy2)
```

PAUSE

```
['a', 'b', 'c', 'd', 'e', 'a', 'b', 'c', 'd', 'e', 'f', 'a', 'b', 'c']  
['a', 'a', 'a', 'b', 'b', 'b', 'c', 'c', 'c', 'd', 'd', 'e', 'e', 'f']  
  
[3, 9, 3, 9, 8, 8, 9, 1, 2, 3, 4, 5, 6, 7]  
[1, 2, 3, 3, 3, 4, 5, 6, 7, 8, 8, 9, 9, 9]
```

串列 (List)

List.reverse()

將 list 內的所有資料反轉

```
icy2 = [3, 9, 3, 9, 8, 8, 9, 1, 2, 3, 4, 5, 6, 7]
print(icy2)

icy2.sort()
print(icy2)

print("")
icy2.reverse()
print(icy2)
```

PAUSE

```
[3, 9, 3, 9, 8, 8, 9, 1, 2, 3, 4, 5, 6, 7]
[1, 2, 3, 3, 3, 4, 5, 6, 7, 8, 8, 9, 9, 9]

[9, 9, 9, 8, 8, 7, 6, 5, 4, 3, 3, 3, 2, 1]
請按任意鍵繼續 . . .
```

串列 (List)

List.copy()

將 list copy 一份，但是所指向的物件不相同

```
icy2 = [3, 9, 3, 9, 8, 8, 9, 1, 2, 3, 4, 5, 6, 7]
#print(icy2)

icy2.sort()
#print(icy2)

#print("")
icy2.reverse()
print(icy2)

icy3 = icy2.copy()

print(icy3)
```

C:\ PAUSE

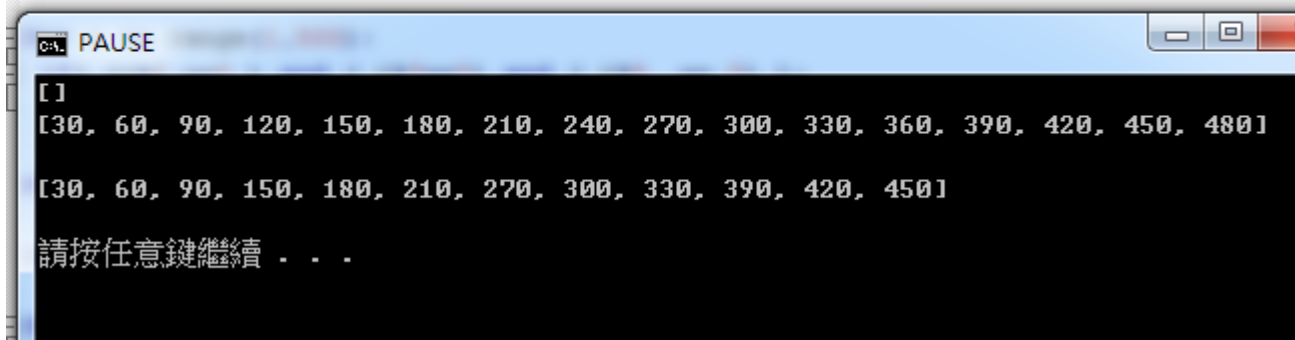
```
[9, 9, 9, 8, 8, 7, 6, 5, 4, 3, 3, 3, 2, 1]
[9, 9, 9, 8, 8, 7, 6, 5, 4, 3, 3, 3, 2, 1]
```

請按任意鍵繼續 . . .

隨堂練習

1. 建立一個空的 list[]
2. 寫一個迴圈去將 1-500 中是 2 是 3 也是 5 的倍數放入 list 中
3. 印出此 list 的目前所有元素
4. 再將此 list 是 8 的倍數拿掉
5. 最後再印出此 list 所有元素

```
icy = []  
  
print(icy)
```



```
PAUSE  
[ ]  
[30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360, 390, 420, 450, 480]  
[30, 60, 90, 150, 180, 210, 270, 300, 330, 390, 420, 450]  
請按任意鍵繼續 . . .
```

集合 (Set)

要怎麼宣告一個集合 (set) 變數：使用 ” {} ” (大括號) 來放入資料

set 裡的資料**可以改變**

set 變數就把他當成 高中的時候在學的集合的概念就對了

集合 (Set)

函數	功能
<code>set.add(x)</code>	將元素 x 加入 set
<code>set.remove(x)</code>	將元素 x 從移除
<code>set.clear()</code>	清空 set 所有元素
<code>set.copy()</code>	複製一份 set
<code>set.discard(e)</code>	將 e 從 set 中捨棄
<code>a.issubset(b)</code>	判斷 set a 是否為 set b 的子集合

集合 (Set)

```
a = { 1, 2, 3, 4, 5, 6 }  
b = { 1, 3, 5, 7, 9, 11 }
```

```
a.add(7)  
print(a)
```

```
a.remove(7)  
print(a)
```

```
a.discard(7)  
print(a)
```

```
print(a.issubset(b))
```

PAUSE

<1, 2, 3, 4, 5, 6, 7>

<1, 2, 3, 4, 5, 6>

<1, 2, 3, 4, 5, 6>

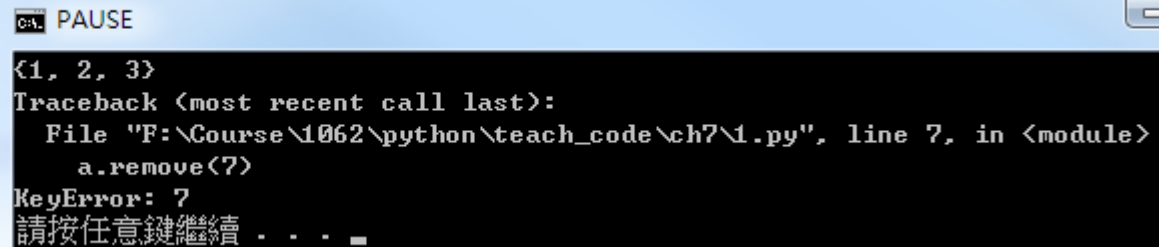
False

請按任意鍵繼續 . . .

集合 (Set)

Remove vs discard

```
1 a = {1,2,3}
2 b = {4,5,6}
3
4 a.discard(7)
5 print(a)
6
7 a.remove(7)
8 print(a)
```

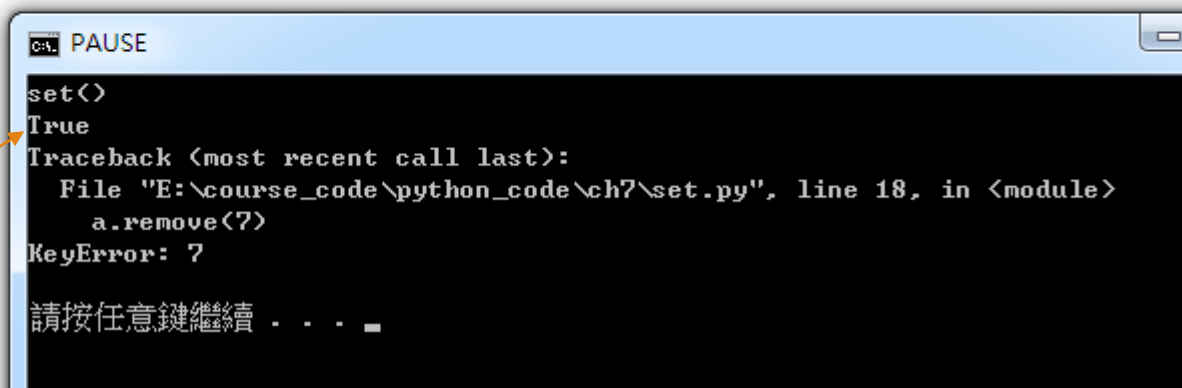


The screenshot shows a Python interpreter window with a blue title bar. The window title is "PAUSE". The output shows the set {1, 2, 3} followed by a traceback. The traceback indicates a KeyError: 7 at line 7 of the file "F:\Course\1062\python\teach_code\ch7\1.py". The error message is "KeyError: 7" and "請按任意鍵繼續" (Press any key to continue).

```
<1, 2, 3>
Traceback (most recent call last):
  File "F:\Course\1062\python\teach_code\ch7\1.py", line 7, in <module>
    a.remove(7)
KeyError: 7
請按任意鍵繼續 . . .
```


集合 (Set)

```
a = { 1, 2, 3, 4, 5, 6 }  
b = { 1, 3, 5, 7, 9, 11 }  
  
a.add(7)  
#print(a)  
  
a.remove(7)  
#print(a)  
  
a.discard(7)  
#print(a)  
  
a.clear()  
print(a)  
  
print(a.issubset(b))  
  
a.remove(7)  
print(a)
```



The screenshot shows a Python interpreter window titled "PAUSE". The output of the code execution is as follows:

```
set()  
True  
Traceback (most recent call last):  
  File "E:\course_code\python_code\ch7\set.py", line 18, in <module>  
    a.remove(7)  
KeyError: 7  
  
請按任意鍵繼續 . . .
```

An orange arrow points from the `print(a.issubset(b))` line in the code block to the `set()` output in the interpreter window.

集合 (Set)

函數	功能
<code>set.difference(set2)</code>	作差集運算，結果需回傳至另一個 set
<code>set.difference_update(set2)</code>	作差集運算，結果直接更新
<code>set.intersection(set2)</code>	作交集運算，結果需回傳至另一個 set
<code>set.intersection_update(set2)</code>	作交集運算，結果直接更新
<code>set.union(set2)</code>	作聯集運算，結果需回傳至另一個 set
<code>set.update(set2)</code>	作聯集運算，結果直接更新

集合 (Set)

```
a = { 1, 2, 3, 4, 5, 6 }  
b = { 1, 3, 5, 7, 9, 11 }  
  
print("a: " , a)  
print("b: " , b)
```

```
c = a.difference(b)  
print(c)  
  
d = a.union(b)  
print(d)  
  
e = a.intersection(b)  
print(e)
```

```
print("-----")  
print("我是分隔線")  
print("-----")
```

```
print("a: " , a)  
print("b: " , b)
```

```
a.difference_update(b)  
print(a)  
  
a.update(b)  
print(a)  
  
a.intersection_update(b)  
print(a)
```

C:\ PAUSE

```
a: {1, 2, 3, 4, 5, 6}  
b: {1, 3, 5, 7, 9, 11}  
{2, 4, 6}  
{1, 2, 3, 4, 5, 6, 7, 9, 11}  
{1, 3, 5}
```

我是分隔線

```
a: {1, 2, 3, 4, 5, 6}  
b: {1, 3, 5, 7, 9, 11}  
{2, 4, 6}  
{1, 2, 3, 4, 5, 6, 7, 9, 11}  
{1, 3, 5, 7, 9, 11}
```

請按任意鍵繼續 . . .

集合 (Set)

frozenset()

使用 frozenset() 函數建立不可變的集合

```
a =frozenset( { 1, 2, 3, 4, 5, 6 } )  
b =frozenset( { 1, 3, 5, 7, 9, 11 } )
```

```
print(type(a))  
print(a)
```

```
print("-----")  
print("""
```

```
c = a | b  
print(c)
```

```
d = a - b  
print(d)
```

```
e = a & b  
print(e)
```

PAUSE

```
<class 'frozenset'>  
frozenset(<1, 2, 3, 4, 5, 6>)
```

```
-----  
frozenset(<1, 2, 3, 4, 5, 6, 7, 9, 11>)  
frozenset(<2, 4, 6>)  
frozenset(<1, 3, 5>)
```

請按任意鍵繼續 . . .

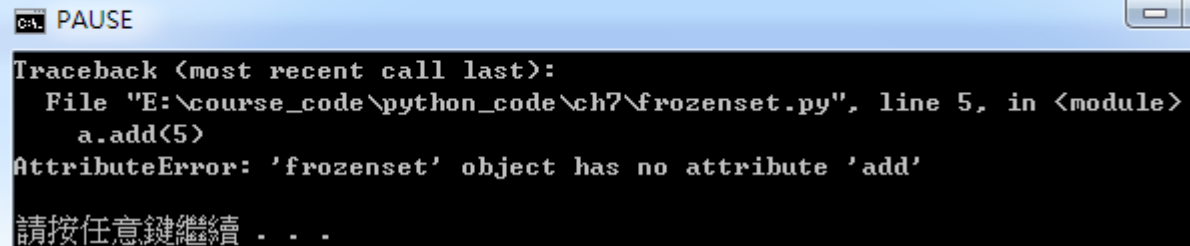
集合 (Set)

frozenset()

使用 frozenset() 函數建立不可變的集合，無法修改資料

```
a =frozenset( { 1, 2, 3, 4, 5, 6 } )  
b =frozenset( { 1, 3, 5, 7, 9, 11 } )
```

```
a.add(5)  
a.remove(5)
```



PAUSE

```
Traceback (most recent call last):  
  File "E:\course_code\python_code\ch7\frozenset.py", line 5, in <module>  
    a.add(5)  
AttributeError: 'frozenset' object has no attribute 'add'  
請按任意鍵繼續 . . .
```

字典 (Dictionary)

宣告一個字典 (dict) 變數：使用 ” {} ” (大括號) 來放入資料

dict 裡的資料是一種 **Key-Value 對應的型態**

key 就是存取該筆 value 的索引值

```
a = {1: 'a', 2: 'b', 3: 'c', 4: 'd' }
```

```
print(a)
```

PAUSE

```
<1: 'a', 2: 'b', 3: 'c', 4: 'd'>
```

請按任意鍵繼續 . . .

字典 (Dictionary)

也可以這樣宣告

```
a = {1: 'a', 2: 'b', 3: 'c', 4: 'd' }  
  
print(a, type(a))  
  
a = dict({1: 'a', 2: 'b', 3: 'c', 4: 'd' })  
print(a, type(a))  
  
a = dict( zip( (1,2,3,4) , ('a','b','c','d') ) )  
print(a, type(a))
```

PAUSE

```
<1: 'a', 2: 'b', 3: 'c', 4: 'd'> <class 'dict'>  
<1: 'a', 2: 'b', 3: 'c', 4: 'd'> <class 'dict'>  
<1: 'a', 2: 'b', 3: 'c', 4: 'd'> <class 'dict'>
```

請按任意鍵繼續 . . .

字典 (Dictionary)

Function	meaning
<code>d[x]</code>	從字典 d 中取得 x 所對應的值
<code>d[x]=y</code>	將字典 d 中 x 所對應的值改為 y 若字典中沒有 x 這個 key 則會新增一組資料
<code>del d[x]</code>	刪除字典 d 中 x 的配對
<code>x in d</code>	判斷 x 是否在字典 d 的 key 值中
<code>x not in d</code>	判斷 x 是否不在字典 d 的 key 值中
<code>len(d)</code>	回傳字典 d 共有多少資料組數。

字典 (Dictionary)

```
icy = {1: 'a', 2: 'b', 3: 'c', 4: 'd' }
```

```
print(icy[1])
```

```
print(icy[3])
```

```
icy[5] = 'e'
```

```
print(icy)
```

```
c = len(icy)
```

```
print(c)
```

```
print( 1 in icy )
```

```
print( 1 not in icy)
```

PAUSE

a

c

{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}

5

True

False

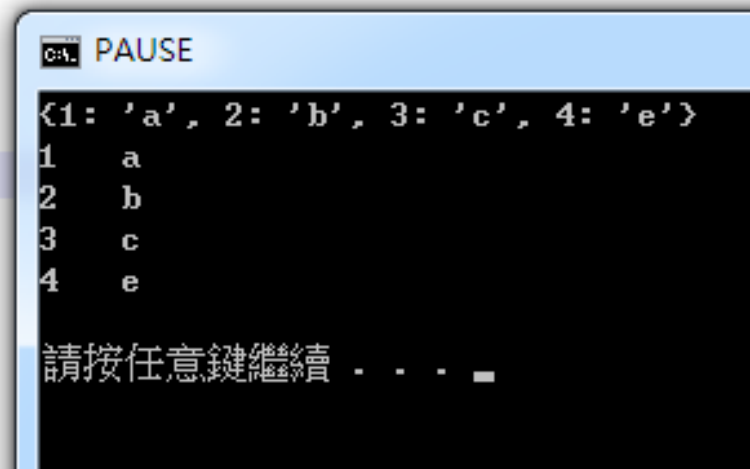
請按任意鍵繼續 . . .

字典 (Dictionary)

Function	meaning
d.update()	更新字典 d 的內容
d.clear()	清空字典 d
dic.copy()	Copy 字典 d
iter(d)	回傳一組由字典 d 的 key 值所建立的疊代器

字典 (Dictionary)

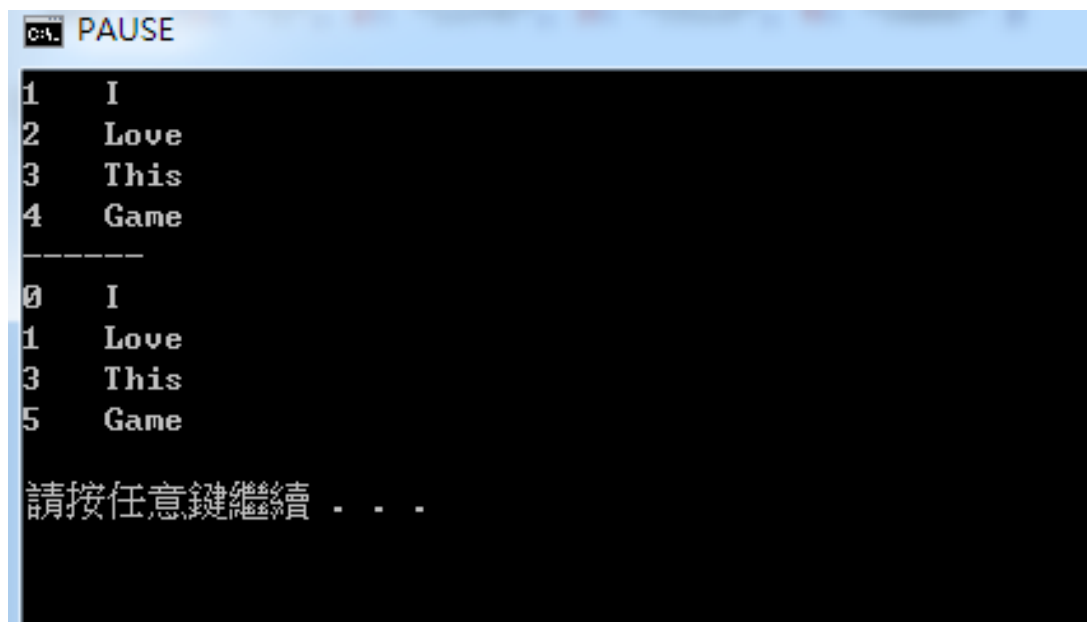
```
icy = {1: 'a', 2: 'b', 3: 'c', 4: 'd' }  
  
icy.update( {4: 'e'})  
print(icy)  
  
for i in iter(icy):  
    print(i, " ", icy[i])
```



```
PAUSE  
{1: 'a', 2: 'b', 3: 'c', 4: 'e'}  
1 a  
2 b  
3 c  
4 e  
請按任意鍵繼續 . . .
```

隨堂練習

1. 建立一個 dict {1: xxx , 2: xxx 3: xxx 4: xxx 5:xxx }
2. 寫一個迴圈輸此 dict 內容
3. 修改此 dict 內容並寫一個迴圈輸出新的 dict 的內容如下



```
C:\> PAUSE
1 I
2 Love
3 This
4 Game
-----
0 I
1 Love
3 This
5 Game
請按任意鍵繼續 . . .
```

Any Questions !?