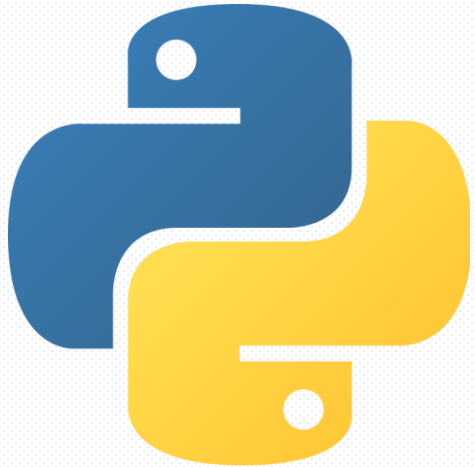
容器型態 (串列、數組、集合、字典)

Container Type (list、tuple、set、dict)

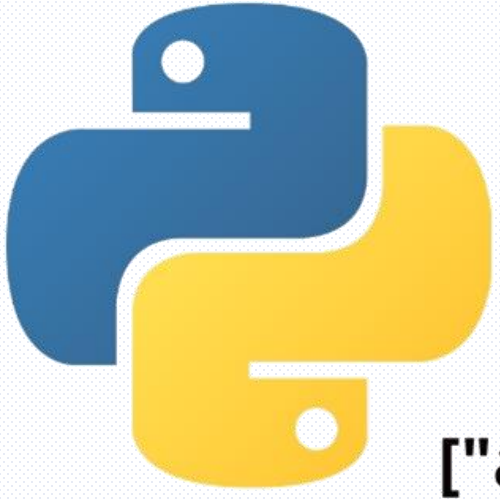


Python Fundamental

容器型態 - Container Type

- 容器型態可用來裝入多個不同型態資料，因此當程式需要處理大量資料時，容器型態就會相當實用。
- 容器型態有以下幾種：
  - 串列 (list)
  - 數組 (tuple)
  - 集合 (set)
  - 字典 (dict)

list 串列



Lists In Python

Example

["apple", "banana", "cherry"]

容器型態 - list (串列)

- 串列，list，指的是由一連串資料所組成、有順序、而且可以改變內容的序列。
- 串列以中括號「[ ] 」表示，裡面的資料以逗點隔開，資料型態可以不同。
- 我們可以使用中括號「[ ]」建立串列或是使用同名函式 list() 轉換成串列。

```
lis=[]
print(lis)

lis=[1,2.3,True,'ABC']
print(lis)

lis=list()
print(lis)

lis=list({1,2.5,False,'XYZ'})
print(lis)
```

```
[]
[1, 2.3, True, 'ABC']
[]
[False, 1, 2.5, 'XYZ']
```

容器型態 - list (串列)

- 串列是最常用的資料結構，以中括號 [ ] 表示。
- 串列裡面的資料稱為元素；元素的資料型態可以不同，我們使用索引位置可以存取元素。
- 串列裡面的元素是有順序的，順序不一樣就是不一樣。

```
In [1]: list1=[1, 2.5, 'xyz', False]
```
不同型態的資料可以被收集在同一個串列中

```
In [2]: print(type(list1))
<class 'list'>
```
顯示 list1 型態

```
In [3]: print(list1[0])
1
```
顯示 list1的第一個元素 (索引值由零開始)

```
In [4]: print(list1[2])
xyz
```
顯示 list1的第三個元素 (索引值由零開始)

容器型態 - list (串列)

- 我們可以用 del 從串列中刪除指定索引的元素。
- 以下面的例子為例，del L[2] 表示從串列 L 中刪除索引為 2 的元素，也可就刪除第三個元素。

```
In [1]: L=[3,6,9,12,15,18,21,24,27,30]

In [2]: del L[2]

In [3]: del L[3]

In [4]: print(L)
[3, 6, 12, 18, 21, 24, 27, 30]
```

容器型態 - list (串列)

◆ 任意物件的串列

- a = [1, 5, 9] # 整數串列 (An integer list)
- b = [2, 3.5, 6.5, 'Hello'] # 混合資料型態的串列 (A mixed list)
- c = [] # 空串列 (An empty list)
- d = [18, [a, b]] # 多維串列 (A list containing a list)
- e = a + b # 串列結合 (join two lists)

◆ 串列的操作

- b[0] = 23 # 更新元素 (change an element)
- x = a[2] # 取得元素內容 (get 2nd element)
- y = b[1:3] # 回傳子串列 (return a sub-list)
- z = d[1][1][1] # 巢狀串列 (Nested lists)

list (串列) 相關運算子

- **+** 連接運算子：用來連接串列。

```
print([1,2,3]+[4,5,6])

print([1,2,3]+['X','Y','Z'])

print([1,2,3]+[7,'A',True])

print([1,2,3]+True)
```

```
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 'X', 'Y', 'Z']
[1, 2, 3, 7, 'A', True]
Traceback (most recent call last):

  File "C:\Users\USER\Desktop\temp.py", line 7,
in <module>
    print([1,2,3]+True)

TypeError: can only concatenate list (not
"bool") to list
```

- ***** 重複運算子：用來重複串列。

```
print([5,6,7]*3)

print([5,6,7]*0)

print([5,6,7]*-3)

print([5,6,7]*2.5)

print([5,6,7]*-2.5)
```

```
[5, 6, 7, 5, 6, 7, 5, 6, 7]
[]
[]
Traceback (most recent call last):

  File "C:\Users\USER\Desktop\temp.py", line 15, in <module>
    print([5,6,7]*2.5)

TypeError: can't multiply sequence by non-int of type 'float'
```

list (串列) 相關運算子

- 比較運算子：(>、<、>=、<=、==、!=) 可以用來比較兩個串列的大小或相等與否。

```
print([1,3,5]==[5,3,1])
print(['A','B','C']>=['A','b','C'])
print(['x','y','z']!=['X','y','z'])
```

```
False
False
True
```

- in 與 not in 運算子：檢查某個元素是否存在/不存在於串列。

```
print(3 in [1,2,3,4,5])
print('B' in ['AB','BC','CD'])
print('AB' in ['AB','BC','CD'])
```

```
True
False
True
```

list (串列) 相關運算子

- 索引與片段運算子
  - 我們可以使用索引運算子取得串列的元素。假設變數 L的值為 [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]，存放順序如下。
  - L[0]、L[1]……L[9] 表示5、10、…、50。
  - L[-1]、L[-2]……L[-10] 表示50、45、…、5。

索引	0	1	2	3	4	5	6	7	8	9
內容	5	10	15	20	25	30	35	40	45	50
索引	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- 我們可以使用片段運算子 ([start:end]) 指定索引範圍。

```
In [1]: L = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

In [2]: L[1:5]
Out[2]: [10, 15, 20, 25]

In [3]: L[2:-2]
Out[3]: [15, 20, 25, 30, 35, 40]
```

list (串列) 建構

- **串列生成、串列建構、串列理解、串列解析 (list comprehension)**
  - 串列生成提供了一種較為簡潔的方式來建立串列。
  - 串列生成會有一個 for 敘述，後面跟著 0 個、1 個或多個 for 或 if 敘述。串列的元素就是運算式產生的結果。

```
In [1]: listc=[i for i in range(10)]

In [2]: print(listc)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [3]: listc2=[j+3 for j in range(10)]

In [4]: print(listc2)
[3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

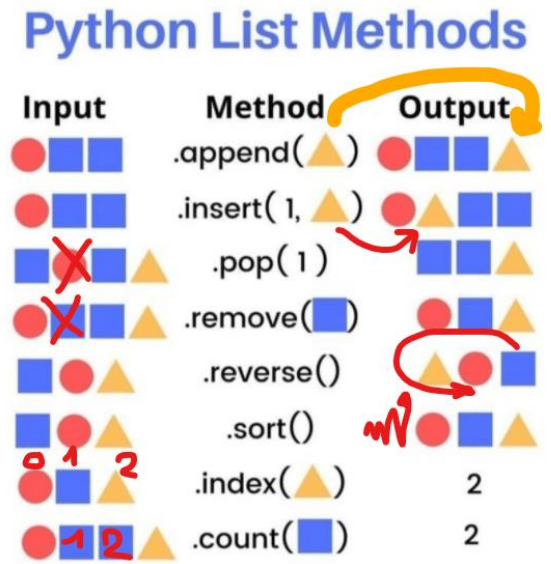
list (串列) 相關函式

- Python 的內建函式有些適用於串列，例如：len()、max()、min()、sum()等。

```
In [1]: print(len([1,3,5,7,9]))
5

In [2]: print(max([1,3,5,7,9]))
9

In [3]: print(min([1,3,5,7,9]))
1

In [4]: print(sum([1,3,5,7,9]))
25
```

list (串列) 相關函式

方法	描述
list.append(x)	將 x 附加到 list 的最後
list.extend(x)	將 x 中的元素附加到 list 的最後
list.count(x)	計算 list 中 x 出現的次數
list.index(x[, i[, j]])	回傳 x 在 list 最小的索引值
list.insert(i, x)	將 x 插入 list 索引值 i 的地方
list.pop([i])	取出 list 中索引值為 i 的元素，預設是最後一個
list.remove(x)	移除 list 中第一個 x 元素
list.reverse()	倒轉 list 中元素的順序
list.sort([key[, reverse]])	排序 list 中的元素




```
In [1]: num1=[1,3,5,7,9]

In [2]: num2=[2,5,8,11,14]

In [3]: num1.append(11)

In [4]: print(num1)
[1, 3, 5, 7, 9, 11]

In [5]: num1.extend(num2)

In [6]: print(num1)
[1, 3, 5, 7, 9, 11, 2, 5, 8, 11, 14]
```

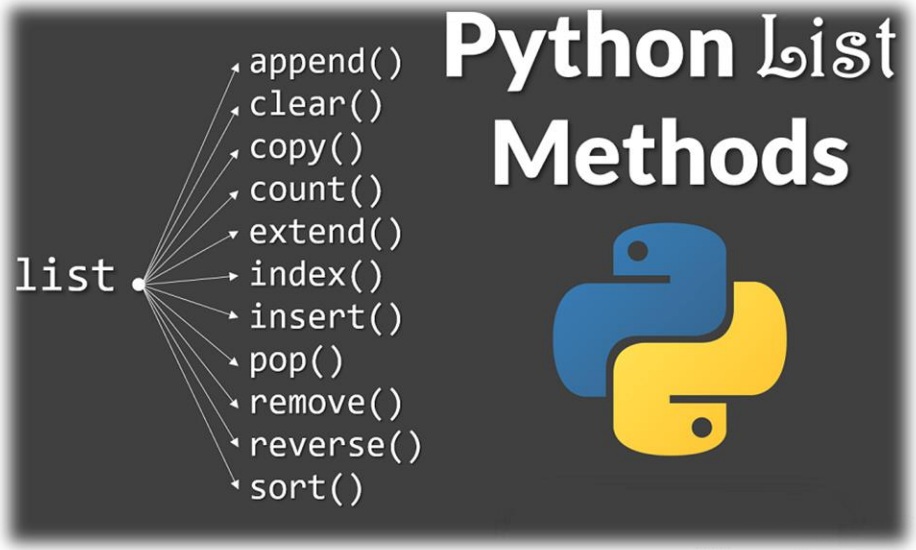
```
In [7]: num1.count(11)
Out[7]: 2

In [8]: num1.index(9)
Out[8]: 4

In [9]: num1.insert(3,999)

In [10]: print(num1)
[1, 3, 5, 999, 7, 9, 11, 2, 5, 8, 11, 14]

In [11]: num1.pop()
Out[11]: 14

In [12]: print(num1)
[1, 3, 5, 999, 7, 9, 11, 2, 5, 8, 11]
```

```
In [13]: num1.remove(5)

In [14]: print(num1)
[1, 3, 999, 7, 9, 11, 2, 5, 8, 11]

In [15]: num1.reverse()

In [16]: print(num1)
[11, 8, 5, 2, 11, 9, 7, 999, 3, 1]

In [17]: num1.pop(5)
Out[17]: 9

In [18]: print(num1)
[11, 8, 5, 2, 11, 7, 999, 3, 1]
```

list (串列) 排序

- Python 提供兩種內建排序的函式：sort() 和 sorted()，這兩個函式都可以用來排序串列。
- sort() 會直接修改原始的串列並排序完成。sorted() 則會回傳一個排序好的新串列。
- 如果要保留原本的串列順序，可以使用 sorted() 產生一個排序好的新串列，但是對原本的串列沒有影響。
- 如果不需要保留原始的串列順序，可以使用 sort() 對原本的串列直接進行排序。

```
In [1]: X=[1,3,2,4,5]

In [2]: print(sorted(X))
[1, 2, 3, 4, 5]

In [3]: print(X)
[1, 3, 2, 4, 5]

In [4]: X.sort()

In [5]: print(X)
[1, 2, 3, 4, 5]
```



tuple 數組

Tuple In

Python

("apple", "banana", "cherry")

tuple (數組)

- tuple，數組，也稱為元組。發音可以唸成 too-pull 也可唸成 tub-pull。指的是由一連串資料組成、有順序而且不可改變內容的序列。
- tuple 以小括號「( ) 」表示，裡面的元素以逗號隔開，資料型態可以不同。
- 我們可以使用同名函式 tuple() 或是小括號「( )」建立數組。

```
In [1]: tup=tuple()

In [2]: print(tup)
()

In [3]: tup2=tuple((1,3,5))

In [4]: print(tup2)
(1, 3, 5)
```

```
In [5]: tup3=()

In [6]: print(tup3)
()

In [7]: tup4=(2,4,6)

In [8]: print(tup4)
(2, 4, 6)
```

tuple (數組)

- tuple 可以收集不同資料型態的元素。
- tuple 中的元素允許 tuple 或是其他的容器型態。
- tuple 的元素是有順序的 (誰前誰後有關係)。
- tuple 的特徵是小括號()，使用索引位置可以存取元素 。
- tuple 與 list 主要的差異在於 tuple 是一種唯讀且不可變更的資料結構，所以不可取代 tuple 中的任何元素。

tuple (數組) 相關運算子 & 相關函式

- 適用於串列且不會涉及變更元素的運算子均適用於數組。
  - 連接運算子 (+)
  - 重複運算子 (*)
  - 比較運算子 (>、<、>=、<=、==、!=)
  - in 與 not in運算子
  - 索引運算子 ([])
  - 片段運算子 ([start:end])
- len()、max()、min() 、sum() 、 index()、 count()等內建函式均適用。

tuple (數組)

- a = (3, 5, 7, 6, 8, 9) # 整數數組 A tuple of integers
- b = () # 空數組 An empty tuple
- c = (2,[4,6],(10,11,12)) # 混合資料型態數組 A tuple containing mixed objects
- d = a[2] **d = 7**
- e = a[3:5] **e = (6,8)**
- f = c[2][1] **f = 11**

tuple (數組)

- 我們可以拆解（Unpack）串列或數組中的元素。

```
In [1]: tup=(3, False, 'XYZ')

In [2]: tup=a,b,c
Traceback (most recent call la

  File "<ipython-input-2-e0364
    tup=a,b,c

NameError: name 'a' is not def


In [3]:

In [3]: a,b,c=tup

In [4]: print(a)
3

In [5]: print(b)
False

In [6]: print(c)
XYZ
```

```
In [1]: a, *b=(1,3,5,7,9)

In [2]: print(a)
1

In [3]: print(b)
[3, 5, 7, 9]

In [4]: x, *y, z=(2,4,6,8,10)

In [5]: print(x)
2

In [6]: print(y)
[4, 6, 8]

In [7]: print(z)
10
```

set 集合

Python Sets and

Sets Function

Example

set (集合)

- set 集合，指的是沒有順序、不可重覆，可改變內容的多個資料。
- 集合前後以大括號「{ }」表示，裡面的元素以逗號隔開，資料型態可以不同。
- 想建立空集合，必須使用 set() 函式。不可使用{ } (因為{ }視為空字典(dict))。

```
In [1]: A=set()

In [2]: A.add(3)

In [3]: A.add('ABC')

In [4]: A.add(True)

In [5]: print(A)
{True, 3, 'ABC'}

In [6]: print(3 in A)
True
```

```
In [1]: X={'a','b',1,2}

In [2]: Y={1,2,'a','b'}

In [3]: print(X==Y)
True

In [4]: print(X)
{'b', 1, 2, 'a'}

In [5]: print(Y)
{'b', 1, 2, 'a'}
```

```
# 包含4個元素的集合
# 元素相同但順序不同，仍是相同集合
```

set (集合)

- set 的元素是無順序的 (誰前誰後沒關係)

```
In [1]: S1={1,1,2,2,2,2,3,3,5,5,6,6,6}

In [2]: S2={6,3,1,5,2,2,5,3,1,2}

In [3]: print(S1==S2)
True

In [4]: print(S1)
{1, 2, 3, 5, 6}

In [5]: print(S2)
{1, 2, 3, 5, 6}
```

沒有重複的元素， 而且設定時的順序也不影響

set (集合) 相關運算子

- 集合支援 in 與 not in 運算子，用來檢查指定的元素是否存在於集合。

- 由於集合中的元素沒有順序之分，因此不支援與順序相關運算，諸如連接運算子 (+)、重複運算子 (*)、索引運算子 ([ ])、片段運算子 ([start:end]) 等。

```
In [1]: S1={1,2,3,4,5}

In [2]: S2={3,4,5,6,7}

In [3]: S3=S1+S2
Traceback (most recent call last):

  File "<ipython-input-3-041fd543711a>", line
1, in <module>
    S3=S1+S2

TypeError: unsupported operand type(s) for +:
'set' and 'set'


In [4]: print(S1-S2)
{1, 2}
```

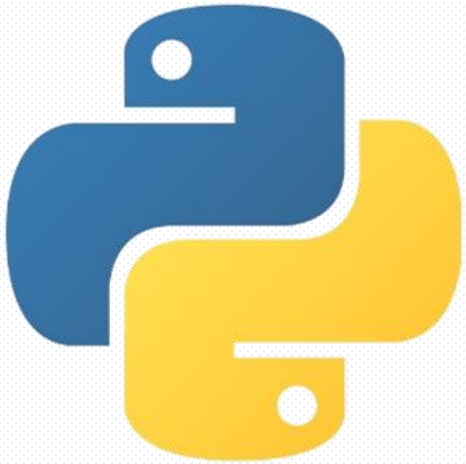
```
In [5]: print(S1[1:3])
Traceback (most recent call last):

  File "<ipython-input-5-7bab60fef15c>", line
1, in <module>
    print(S1[1:3])

TypeError: 'set' object is not subscriptable


In [6]:

In [6]: print(S2[2])
Traceback (most recent call last):

  File "<ipython-input-6-4c435194bc73>", line
1, in <module>
    print(S2[2])

TypeError: 'set' object does not support
indexing
```

set (集合) 相關函式

- len()、max()、min() 和sum() 等內建函式均適用於集合：

```
In [1]: A={1,5,3,7,9,5,9,7,3}

In [2]: print(len(A))
5

In [3]: print(max(A))
9

In [4]: print(min(A))
1

In [5]: print(sum(A))
25
```

dictionary 字典



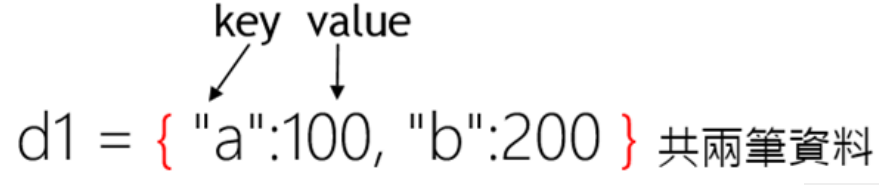
Python

Dictionary

{'name': 'john', age: '28'}

dict (字典)

- dict 字典，指的是沒有順序、不可重覆，可改變內容的多個「鍵：值」 (key：value pair) 的對映方式。
- dict 型態屬於對映型態 (mapping type)，也就是以鍵 (key) 做為索引來存取字典裡面的值。
- dict 的前後以大括號表示，裡面的鍵：值以冒號隔開，若有多筆鍵/值資料再以逗號區隔開。

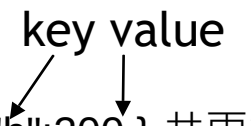



```
In [1]: X={'A':567, 'B':789}

In [2]: Y={'B':789, 'A':567}

In [3]: print(X==Y)
True

In [4]: print(X)
{'A': 567, 'B': 789}

In [5]: print(Y)
{'B': 789, 'A': 567}
```

```
# 包含2個鍵:值對的字典
# 鍵：值對相同但順序不同，仍是相同字典
```

dict (字典)

- dict (大括號{ })裡頭以key : value 為配對的資料項目，若有多筆資料再以逗號區隔開。



- 例如 d1 = { "a":100, "b":200 } 共兩筆資料。
- dict 的元素是無順序的(誰前誰後沒關係)。
- dict 沒有索引，因此無法以索引位置(無索引) 存取元素，而是改以 「key」來存取元素。(Key 不是索引！)
- dict 中的 key 必須是不可變(immutable) 的資料型態，如數字、字串 (string)。
- dict 中的 value 則沒有限制。

dict (字典) 基本處理

◆ 建立字典

我們可以使用Python內建的dict() 函式或 {} 建立字典，例如下面的前四個敘述會建立包含相同鍵:值對的字典。

```
In [1]: A={'one':1, 'two':2, 'three':3}

In [2]: B=dict({'three':3, 'one':1, 'two':2})

In [3]: C=dict(one=1, two=2, three=3)

In [4]: D=dict([('two', 2),('one', 1), ('three',3)])
```

```
In [5]: print(A==B==C==D)
True

In [6]: print(A)
{'one': 1, 'two': 2, 'three': 3}

In [7]: print(B)
{'three': 3, 'one': 1, 'two': 2}

In [8]: print(C)
{'one': 1, 'two': 2, 'three': 3}

In [9]: print(D)
{'two': 2, 'one': 1, 'three': 3}
```

dict (字典) 基本處理

- **取得、新增、變更或刪除鍵：值**

  - 我們可以使用 dict[key] = value 新增鍵和值。

```
In [1]: A={}

In [2]: A['1']=10

In [3]: A['2']=20

In [4]: A['3']=30

In [5]: print(A)
{'1': 10, '2': 20, '3': 30}
```

  - 我們可以使用 del 敘述刪除鍵為 key 的鍵和值，其語法如下。

```
In [1]: A={'one':1, 'two':2, 'three':3}

In [2]: del A['one']

In [3]: print(A)
{'two': 2, 'three': 3}
```

dict (字典) 基本處理

- **取得、新增、變更或刪除鍵：值**

  - 建立字典後，我們可以透過鍵來取得對映的值

```
In [1]: A={'one':1, 'two':2, 'three':3}

In [2]: X=A['one']

In [3]: print(X)
1
```

  - 我們也可以使用 dictName[key] = value的語法來新增或變更鍵值。

```
In [4]: A['two']=200

In [5]: print(A['two'])
200
```

dict (字典) 相關運算子

- 字典支援 == 和 != 兩個比較運算子，也支援 in 與 not in 運算子，用來檢查指定的鍵是否存在於字典。

```
In [1]: A={"a":1,"b":2,"c":3}

In [2]: B={"a":7,"b":2,"c":9}

In [3]: print(A['a']==B['a'])
False

In [4]: print(A['b']==B['b'])
True

In [5]: print(A['c']!=B['c'])
True
```

```
In [6]: print(1 in A)
False

In [7]: print(a in A)
Traceback (most recent call last):

  File "<ipython-input-7-e8aeacafe694>", line 1, in
<module>
    print(a in A)

NameError: name 'a' is not defined


In [8]: print('a' in A)
True
```

dict (字典) 相關運算子

- 由於字典中的鍵：值 沒有順序之分，因此，字典不支援連接運算子 (+)、重複運算子 (*)、索引運算子 ([ ])、片段運算子 ([start:end]) 或其它與順序相關的運算。

```
In [1]: A={"a":1,"b":2,"c":3}

In [2]: B={"a":7,"b":2,"c":9}

In [3]: print(A+B)
Traceback (most recent call last):

  File "<ipython-input-3-d4231d815df3>", line 1, in <module>
    print(A+B)

TypeError: unsupported operand type(s) for +: 'dict' and 'dict'


In [4]: print(A*2)
Traceback (most recent call last):

  File "<ipython-input-4-c4cae0d057c1>", line 1, in <module>
    print(A*2)

TypeError: unsupported operand type(s) for *: 'dict' and 'int'
```

```
In [5]: print(A[0])
Traceback (most recent call last):

  File "<ipython-input-5-332f5adb1257>", line 1, in <module>
    print(A[0])

KeyError: 0


In [6]: print(A['a':'c'])
Traceback (most recent call last):

  File "<ipython-input-6-5075d8223b19>", line 1, in <module>
    print(A['a':'c'])

TypeError: unhashable type: 'slice'
```

dict (字典) 相關函式

- **keys()、values() 、items()**

```
In [1]: dic={'X':123,'Y':456,'Z':789}

In [2]: print(dic)
{'X': 123, 'Y': 456, 'Z': 789}

In [3]: print(dic.keys())
dict_keys(['X', 'Y', 'Z'])

In [4]: print(dic.values())
dict_values([123, 456, 789])

In [5]: print(dic.items())
dict_items([('X', 123), ('Y', 456), ('Z', 789)])
```

迴圈與容器型態處理

```python
lista=[1,5,3]
tupleb=(2,6,4)
setc={7,9,8}
dictd={"a":123,"b":456,"c":789}

for i in lista:
    print(i)
print()

for j in tupleb:
    print(j)
print()

for k in setc:
    print(k)
print()

for z in dictd:
    print(z)
print()

for x,y in dictd.items():
    print(x,y)
```

```
1
5
3

2
6
4

8
9
7

a
b
c

a 123
b 456
c 789
```

enumerate()函式與容器型態

- enumerate() 函式：遍歷一個序列的同時追踪當前元素的索引。

```
lista=[1,5,3]
tupleb=(2,6,4)
setc={7,9,8}
dictd={"a":123,"b":456,"c":789}

for i in enumerate(lista):
    print(type(i))
    print(i)

for i,j in enumerate(lista):
    print(i,j)
```

```
<class 'tuple'>
(0, 1)
<class 'tuple'>
(1, 5)
<class 'tuple'>
(2, 3)
0 1
1 5
2 3
```

zip()函式與容器型態

- zip()函式用於將可迭代的對像作為參數，將對像中對應的元素打包成一個個元組，然後返回由這些元組組成的列表。
- 如果各個迭代器的元素個數不一致，則返回列表長度與最短的對象相同，利用* 號操作符，可以將元組解壓為列表。

```
lista=[1,5,3]
tupleb=(2,6,4)
setc={7,9,8}
dictd={"a":123,"b":456,"c":789}

for i,j,k,l in zip(lista, tupleb, setc, dictd.values()):
    print(i,j,k,l)
```

```
1 2 8 123
5 6 9 456
3 4 7 789
```

map()函式與容器型態

```
lista=[1,5,3]
tupleb=(2,6,4)
setc={7,9,8}
dictd={"a":123,"b":456,"c":789}

mapob=map(float,lista)
print(mapob)
mapfloat=list(map(float, lista))
mapbool=tuple(map(bool, tupleb))
mapstr=set(map(str,setc))

for i,j,k in zip(mapfloat,mapbool,mapstr):
    print(i,j,k)
```

```
<map object at 0x0000016796457460>
1.0 True 7
5.0 True 8
3.0 True 9
```

容器型態歸納

容器型態	list (串列)	tuple (數組)	set (集合)	dict (字典)
前後符號	[ ]	( )	{ }	{ }
有無順序	有	有	無	無
內容更動	可	不可	可	可

Q & A