UNIT 6

python™

Tuple, List, Set, Dictionary

張傑帆 Chang, Jie-Fan

OUTLINE

- 序對
- 串列
- 字典
- 集合

序對 TUPLE

- 序對（tuple）會在( ) 裡面放置資料，這些資料有順序但是不能更改。
  - tuple 內的資料是從0 開始編號，也可使用[ ] 來取出特定 index 值的資料
  - 此外，Python 中有順序的資料型態（如list、str 等）的 index 值也都可以這樣使用
- 換言之，如果t 不是空的，那第一個元素必定為t[0]
  - 如果要找到tuple 裡面的最後一筆資料，可以使用負數索引
  - 也就是，如果t 不是空的，則最後一個元素必定是t[-1]

序對

- tuple 只有兩個方法，因為當tuple 被建立後，就不能被修改
- 所以只有index 跟count 兩種方法。

```
01  # 設定t 的值
02  t = (‘a’, ‘b’, ‘mary’, ‘john’, ‘a’, ‘mary’, ‘a’, ‘b’)
03  print(t.index(‘a’, 1, 5))
04                  # 在t 的index 1 到5 範圍中尋找'a' 第一次出現的index 值。
05  print(t.count(‘mary’))          # 計算在'mary' 在t 當中出現過幾次
06  print(t.count(‘b’, 0, 5))       # tuple 的count 沒有辦法接收三個值
```

```
>>>
4
2
Traceback (most recent call last):
  File “C:\Users\John\Desktop\Google 雲端硬碟\Python
Wizard\examples\Ch7\ex07_01.py”, line 6, in <module>
```

序對

```
>>>
4
2
Traceback (most recent call last):
  File "C:\Users\John\Desktop\Google 雲端硬碟\Python
Wizard\examples\Ch7\ex07_01.py", line 6, in <module>
    print(t.count('b',0,5))        #tuple 的count 沒有辦法接收三個值
TypeError: count() takes exactly one argument (3 given)
>>>
```

- str.count() 跟tuple.count() 能接受的參數就不太相
- str.count() 可以指定index 範圍，在範圍內計數
- tuple.count() 只能接收單一值來做統計，沒辦法指定範圍

小練習

- **請取出tuple中第3個位置的資料**

```
01  boxes = (23, 45, 65, 23, 47)        # 這是一個tuple
02  print(boxes    …                     # 取出第3個位置中的資料
```

- **執行結果**

```
>>>
65
>>>
```

串列 LIST

- **串列（list）**會在[ ]裡面放置資料，而這些資料是有順序的，且具有可以更改的型態。

```
01  a = []                          # 建立空串列
02  b = [1, 2.0, '3', [4], (5)]     # 建立具有五個不同型態的元素
03  print(b[1])                     # 印出b 的第1 個元素 2.0
04  print(b[3])                     # 印出b 的第3 個元素 [4]
05  print(b)                        # 印出b
```

```
>>>
2.0
[4]
[1, 2.0, '3', [4], 5]
>>>
```

串列LIST

- tuple 跟list 很相似
- 差異在於list 可以增加、刪除與修改，tuple 不行
- tuple 就像是客製化的模板，你可以隨意設定這個 tuple 要多大，裡面要裝甚麼東西；但是一旦建立好就不能再更動
- 反之，list 就會比tuple 更加具有更動的彈性

串列

- 既然list 具有的彈性較高，又可以修改資料內容，那為何還要有tuple 的存在呢？
- 因為tuple 雖然沒有append、extend、remove 這些方法，但是因為不能修改，所以在存取的速度上比起list 要來得快。
- 此外，由於tuple 屬於不可變動值，所以可以拿來當作字典（dict）的key，故在使用上還是有list 所無法取代的部分

串列

- 在這裡列出一些常用的list 的函數：

函數	描述
list.append(object)	在list的最後面加入object。
list.extend(I)	將另一個具有順序的物件I附加在此list的最後面。
list. insert(index, object)	將元素object插入在index值為i的位置。
list.remove(value)	將list裡面的第一個符合value之元素刪除。
list.pop(i)	將list裡面index值為i的元素刪除並回傳。如果沒有給定i值，則刪除並回傳最後一個

串列

函數	描述
list.clear()	將整個list清空。
list.index(value, [start, [stop]])	查詢list當中第一個value出現的index值並回傳。
list.count(value)	計算list中value出現的次數。
list.sort()	將list裡面的元素做排序並儲存。
list.reverse()	將list裡面的元素順序顛倒並儲存。
list.copy()	將list直接複製一份。

串列

- 下面逐項介紹list 的常用函數：
  - **list.append(x)**
    append(x) 會將圓括號裡面的物件x 加在list 的最後端
  - Python 裡面所有東西都是物件，因此所有東西都可以用append() 加入list 的最後端

串列

- **list.pop(i)**
- i 接受int 型態的值為參數，然後會將index 值為i 的元素回傳到呼叫此函數的程式碼，並且將此元素從list 中移除，不寫i則移出最後一筆資料

- **list.clear()**
- 此函數沒有傳入參數
- 呼叫此函數會將list 的元素全部清空。

範例

```
01 list1 = [1, 2, 3, 4]     # 給定list1 初始值
02 print(list1)
03 list1.append(10) #最後加入10
04 print(list1)
05 list1.append('abc')) #加入'abc'
06 print(list1)
07 print(list1.pop()) #取出最後一筆
08 print(list1)
09 print(list1.pop()) #取出最後一筆
10 print(list1)
```

```
>>>
[1, 2, 3, 4]
[1, 2, 3, 4, 10]
[1, 2, 3, 4, 10, 'abc']
'abc'
[1, 2, 3, 4, 10]
10
[1, 2, 3, 4]
>>>
```

小練習

- 有一 shelf 書櫃裡的第三本書掉到地上了
- 請將書本放回去並按照次序排列整齊
- **程式碼**

```
01  shelf = [“Book1”, “Book2”, “Book6”, “Book4”, “Book5”]
                                  # 這是一個list 串列
02  shelf.a                       # 將shelf 中第3 個元素加到並新增到串列最後
03  shelf[                        # 將shelf 中第3 個元素更改為Book3
04  print(shelf)                  # 輸出結果
```

- **執行結果**

```
>>>
['Book1', 'Book2', 'Book3', 'Book4', 'Book5', 'Book6']
>>>
```

小練習

- 請利用迴圈及串列的append和pop功能：
- 將「資料i (i=1, …, 10)」倒序放入串列folder中，例如(10,9,8...)
- 然後，再將資料順序取出，例如(1,2,3...)

```
>>>
資料1
資料2
資料3
資料4
資料5
資料6
資料7
資料8
資料9
資料10
>>>
```

串列

- **list.extend(I)**
- 將有順序物件中的元素個別加到list 的最後端
- extend() 只接受有順序的物件為參數，也就是有index 值的物件，例如list、tuple、str 都可以
- 不過，如果以str 為參數傳入的話，會將str 的每個字元拆開當作個別元素加到list 的最後端
- 以下舉例說明：

```
01  list1 = [1, 2, 3, 4]      # 給定list1 初始值
02  print(list1)
03
04  list2 = ['a', 'b', 'c']   # 給定list2 初始值
05  list1.extend(list2)       # 將list2 的元素用extend 逐項加到list1 最後端
06  print(list1)
07
08  list1.extend('test')      # 將'test' 的各個字元拆，開逐項加到list1 最後端
09  print(list1)
```

```
>>>
[1, 2, 3, 4]
[1, 2, 3, 4, 'a', 'b', 'c']
[1, 2, 3, 4, 'a', 'b', 'c', 't', 'e', 's', 't']
>>>
```

串列

- **list.insert(index, object)**
- insert() 需要兩個傳入參數，分別是index 跟object
- index 用來指定要插入object 的位置
- object 只要是物件都可以利用insert 插入list
  以下舉例說明：

```
01  list1 = [1, 2, 3, 4]            # 給定list1 初始值
02  print(list1)
03
04  list1.insert(2, ‘insert’)        # 將‘isert’ 插入到index = 2 的地方
05  print(list1)
06
07  list1.insert(7, ‘insert2’)       # 將‘isert’ 插入到index = 7 的地方！！？？
08  print(list1)
10  list1.insert(6, ‘insert3’)       # 將‘isert’ 插入到index = 6 的地方！！？？
11  print(list1)
```

```
>>>
[1, 2, 3, 4]
[1, 2, 'insert', 3, 4]
[1, 2, 'insert', 3, 4, 'insert2']
[1, 2, 'insert', 3, 4, 'insert2', 'insert3']
>>>
```

串列

- **list.remove(value)**
- value 接受所有可能的值當作傳入參數，並將list裡面的第一個符合value之資料刪除
- 由於list 屬於容器資料型別，因此可以存放所有型態的物件。
- 只要輸入的value 符合Python 所定義的物件表現值，就可能被list.remove()所接受
  以下舉例說明：

```
01 list1 = [1, 'a', (12.4), [True], range(10)]  # 將不同型別的物件放入list1
02 print(list1)
03
04 list1.remove(1)                # 將1 從list1 移除
05 print(list1)
06
07 list1.remove('a')              # 將'a' 從list1 移除
08 print(list1)
```

```
10 list1.remove(range(10))   # 將range(10) 從list1 移除
11 print(list1)
12
13 list1.remove((12.4))      # 將(12.4) 從list1 移除
14 print(list1)
15
16 list1.remove([True])      # 將[True] 從list1 移除
17 print(list1)
```

```
>>>
[1, 'a', (12.4), [True], range(0, 10)]
['a', (12.4), [True], range(0, 10)]
[(12.4), [True], range(0, 10)]
[(12.4), [True]]
[[True]]
[]
>>>
```

串列

- **list.index(value, [start, [stop]])**
- index()有三個參數，其中value是必須傳入的，只要是Python所定義的值都可以接受
- start跟stop是選擇性選項，接受型態為int的值
- index()會將list中符合value的第一個元素index值回傳給呼叫此函數的程式碼。
- 不管start的給定值，所回傳的index 值都是以原本的索引順序為準
- 以下舉例說明：

串列

```
01  # 給定list1 初始值，為讓index 的查詢功能更明顯，給予較多值
02  list1 = [1, 2, 3, 5, 1, 47, 65, 45, 14, 2, 4, 5, 7, 4, 32, 1, 8, 4, 6]
03  print(list1)
04  print(list1.index(2))              # 在list1 裡面找尋2，
05                                     # 並回傳第一個2 的index 值
06  print(list1.index(2, 7))           # 從list1[7] 之後找尋2，
07                                     # 並回傳第一個2 的index 值
08  print(list1.index(4, 5, 15))       # 在list1[5:15] 裡面找尋4，
09                                     # 並回傳第一個4 的index 值
```

```
>>>
[1, 2, 3, 5, 1, 47, 65, 45, 14, 2, 4, 5, 7, 4, 32, 1, 8, 4, 6]
1
9
10
>>>
```

串列

- **list.count(value)**
- 跟index() 與remove() 一樣，count() 接受所有 Python 內部定義的值
- count() 會回傳在list 中和value 值相同的元素個數，並以int 型態回傳值

串列

- **list.sort()**
- sort() 不需要傳入參數
- 呼叫sort() 的list 會將其所有元素做排序
- 如果list 裡面的元素是不同型態的話，則將無法彼此進行比較
- 而如果是字串型別，會比較字串中第一個字母的 Unicode 碼大小；
- 如果是list 或tuple，則會比較 index = 0 的元素 (作為比較的元素還是需要是同資料型態)。

串列

- **list.reverse()**
- reverse() 不需要傳入參數
- 呼叫reverse() 的list 會將所有元素的順序進行反轉

- **list.copy()**
- copy() 不需要傳入參數
- 呼叫copy() 的程式碼會得到一個跟list 完全一樣的複製清單
- 使用此函數所得的清單跟原本的list 所指向的物件並不相同

串列

- 示範程式碼

```
>>>
['john', 'marry', 'ben', 'adam']
['john', 'marry', 'ben', 'adam', 'ramsay']
['john', 'marry', 'ben', 'adam', 'ramsay', 'dion', 'carl', 'john']
```

```
01  # 給定stu 初始值並印出，確認stu 初始資料
02  stu = [‘john’, ‘marry’, ‘ben’, ‘adam’]
03  print(stu)
04
05  stu.append(‘ramsay’)              # 將‘ramsay’加到stu 的最後面
06  print(stu)
07
08  stu2 = (‘dion’, ‘carl’, ‘john’)   # 建立一個名為stu2 的序對
09  stu.extend(stu2)                  # 將stu2 的資料加到stu 的最後面
10  print(stu)
```

串列

- 示範程式碼

```
12  stu.extend('test')          # 將'test' 加到stu 後面
13  print(stu)
14
15  stu.remove('john')          # 找尋stu 裡面第一個'john'，並將它移除
16  print(stu)
17
18  print(stu.pop(2))           # 印出並刪除stu 中index 值為2 的元素
19  print(stu)
```

```
>>>
['john', 'marry', 'ben', 'adam', 'ramsay', 'dion', 'carl', 'john', 't', 'e', 's', 't']
['marry', 'ben', 'adam', 'ramsay', 'dion', 'carl', 'john', 't', 'e', 's', 't']
adam
['marry', 'ben', 'ramsay', 'dion', 'carl', 'john', 't', 'e', 's', 't']
```

```
21 stu.insert(0, ‘dion’)          # 在stu 的index 為0 的位置插入‘dion’
22 print(stu)
23
24 print(stu.count(‘dion’))       # 計算‘dion’ 在stu 當中出現幾次
25 stu.sort()                     # 將stu 裡面的元素依Unicode 大小排序
26 print(stu)
```

```
['dion', 'marry', 'ben', 'ramsay', 'dion', 'carl', 'john', 't', 'e', 's', 't']
2
['ben', 'carl', 'dion', 'dion', 'e', 'john', 'marry', 'ramsay', 's', 't', 't']
```

```
28 stu.reverse()                  # 反轉stu 裡面的元素順序
29 print(stu)
30
31 stu3 = stu                     # 將stu 的物件參照指派給stu3
32 stu4 = stu.copy()              # 使用copy() 將stu 複製一份給stu4
33 print(stu == stu3)             # 使用is 跟== 運算子驗證stu、stu3、stu4 的關係
34 print(stu is stu3)
35 print(stu == stu4)
36 print(stu is stu4)
```

```
['t', 't', 's', 'ramsay', 'marry', 'john', 'e', 'dion', 'dion', 'carl', 'ben']
```

```
True
True
True
False
```

關係運算子

- 除了and、or、not 這三個關鍵字被拿來當作邏輯運算子外，還有is 與in 這兩個關鍵字也是。
- is 是判斷兩個運算元是否「本質上」相等，並回傳bool 值
- 而所謂「本質上」的相等，若指的是變數，則代表是否指向同一個物件。
- 故兩個變數可以有完全相同的值，但是卻指向不同的物件，例如：

關係運算子

- 示範程式碼

```
01  list1 = [1, 'test']      # 將一樣的值
02  list2 = [1, 'test']      # 賦予到兩個不同的變數上面
03
04  print(list1 == list2)    # 測試list1 跟list2 的值是否相等
05  print(list1 is list2)    # 測試list1 跟list2 的指向目標物件是否為同一個
06
07  list1 = list2            # 將list1 指向list2 所指向的物件
08
```

關係運算子

- 示範程式碼

```
09  print(list1 == list2)   # 測試list1 跟list2 的值是否相等
10  print(list1 is list2)    # 測試list1 跟list2 的指向目標物件是否為同一個
11
12  string1 = ‘test’         # 同樣將一樣的值
13  string2 = ‘test’         # 賦予到兩個不同的變數上面
14
15  print(string1 == string2)    # 測試string1 跟string2 的值是否相等
16  print(string1 is string2)     # 測試string1 跟string2 指向目標物件是否為同一個
```

關係運算子

- 執行結果

```
>>>
True
False
True
True
True
True
>>>
```

小練習

- 請讓使用者輸入一數列，以append加入list當中(-1結束，list中不含-1)

- 將其由小到大排序後印出
- 在第四個位置插入一整數10後印出
- 印出list中有幾個整數10
- 將其由大到小排序後印出

字典

- **字典（dict），為dictionary 的縮寫**
- 具有Key-Value 對應的型態
- 字典是種配對型態，且具有多筆資料的物件，key 就是存取該筆value 的索引值
- 同樣要建立一個dict，有四種可以用的方法：

```
d1 = {1: 'a', 2: 'b'}
d2 = dict({1: 'a', 2: 'b'})
d3 = dict(zip((1, 2), ( 'a', 'b')))
d4 = dict([[2, 'b'], [1, 'a']])
```

字典

- 這四種方法所建立出來的dict 是完全一樣的，用「==」檢驗會得到True，不過用is 就不會
- 因為is 檢驗的項目是「是否指向同一物件」

```
01  d1 = {1: 'a', 2: 'b'}
02  d2 = dict({1: 'a', 2: 'b'})
03  d3 = dict(zip((1, 2), ( 'a', 'b')))
04  d4 = dict([[2, 'b'], [1, 'a']])
05  d5 = d1
06  print(d1 is d2)
07  print(d1 is d3)
08  print(d1 == d2)
09  print(d1 == d3)
10  print(d1 == d4)
11  print(d5 is d1)
```

```
>>>
False
False
True
True
True
True
>>>
```

字典

- 而dict的讀取、刪除、回傳與判斷等方式則如下面內容所示。

計算	描述
d[x]	從d中取得x所對應的值
d[x]=y	將d中x所對應的值指定為y 若d中沒有x這個key則新增一組
del d[x]	刪除d中x所屬的組合
x in d	判斷x是否在d的key值中
x not in d	判斷x是否不在d的key值中
iter(d)	回傳由d的key值所建立的迭代器
len(d)	回傳d的資料組數

```
01 d = dict(zip((1, 2, 3), ( ‘a’, ‘b’, ‘c’)))      # 建立dict
02 print(type(d))                                  # 印出d 的型別
03 print(d[3])                                     # 印出d[3]
04 d[4] = ‘d’                                      # 因為原本沒有4 這組key，
05                                                 # 因此此行新增一個key[4]，對應值為d
06 print(d)
07
08 del d[1]                                        # 刪除d[1]
09 print(d)
10 print(3 in d)                 # 檢驗3 是否在d 的key 值中
11 print(2 not in d)             # 檢驗2 是否不在d 的key 值中
12
13 for i in iter(d):    # 建立for 迴圈。以iter(d) 為key 值所建立的迭代器
14     print(i, end = ‘ ’)       # 印出i
15 print(len(d))                 # 印出d 的配對組數
```

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

小練習

- 當使用者分別輸入「P」、「M」或「H」時，請利用字典(dict)的方法分別輸出「Pikachu」、「Mickey Mouse」或「Hello kitty」

小練習

- 密碼簿及一串文字(5-4-2.2-‘1’-“Six”)，解開了這道密碼

```
01  passbook = {‘1’:“ Wor”, 2.2:“gic”, “Three”:“rd”, 4:“ Ma”, “Six”:“ld.”,
    5:“A”}
02                                          # 這是一個字典（密碼簿）
03  print(                                  # print 出內容
04  print(
05  print(                  ……
06  print(
07  print(
```

```
>>>
A Ma        …
>>>
```

集合

- **集合（set）**會在{}裡面放置元素，類似數學裡面的集合概念
- 由於集合是可變動的，有幾個方法是可以對**set** 作更動的
- 下表將**set** 的函數分別列出作介紹：

集合

函數	功能
set.pop()	將set的隨意一個元素回傳給呼叫的程式碼之後將此元素刪除。
set.add(e)	將元素e加入set。
set.remove(e)	將元素e從set中移除。
set.clear()	將set的所有元素清空。
set.copy()	回傳一份複製的set。
set.discard(e)	將e從set中捨棄。
set.difference(set2)	將兩個set作差集運算，並回傳一個新的set
set.difference_update(set2)	將兩個set作差集運算，並將結果更新到set

集合

函數	功能
set.intersection(set2)	將兩個set作交集運算，並回傳一個新的set
set.intersection_update(set2)	將兩個set作交集運算，並將結果更新到set
set.union(set2)	將兩個set作聯集運算，並回傳一個新的set
set.update(set2)	將兩個set作聯集運算，並將結果更新到set
set.issubset(set2)	判斷set是否為set2的子集合，回傳bool值
set.issuperset(set2)	判斷set2是否為set的子集合，回傳bool值

集合

- set.remove()跟set.discard()的功能是一樣的
- 不過，當所傳入的參數不在set裡面時，set.remove會回傳KeyError，而set.discard則不會做任何回傳的動。
- set.difference()、set.difference_update()、set.intersection()、set.intersection_update()、set.union() 與set.update() 等可歸類成三類
- 每一類的差異都只是一個會回傳一個新的set，而另一個則是將運算後的結果更新到原本的set 而已

以下舉例說明：

```
01  continents = {‘California’, ‘New York’}
02  cities = {‘New York’, ‘Phoenix’, ‘Chicago’}
03  print(continents, ‘\n’, cities, ‘\n’, sep = ‘’)
04                          # 給定兩組set 的值，並且印出以確定初始狀態
05  continents.clear()      # 將continents 清空
06  print(continents, ‘\n’)
07  continents.add(‘Texas’)
08  continents.add(‘California’)
09  continents.add(‘New York’)      # 把三個元素加到continents
10  print(continents, ‘\n’)
```

```
>>>
{'New York', 'California'}
{'Chicago', 'Phoenix', 'New York'}

set()

{'New York', 'Texas', 'California'}
```

```
11  cities.remove('Chicago')          # 將'Chicago' 從cities 中移除
12  print(cities, '\n')
13  continents.discard('California')  # 將'California' 從continents 中移除
14  print(continents, '\n')
15  print(cities.union(continents))   # 取聯集，會將結果回傳一個新的set
16  print(cities)
17  print(cities.update(continents))  #取聯集， 以union 跟update 做代表，
18                                    # 示範回傳新set 和更新到set 的差異
19  print(cities)
```

```
{'Phoenix', 'New York'}

{'New York', 'Texas'}

{'Phoenix', 'New York', 'Texas'}
{'Phoenix', 'New York'}
None
{'Phoenix', 'New York', 'Texas'}
```

集合

- 由於集合這個資料型態沒有順序，因此
- **就算是同樣的一支程式執行數次後，**
  出現的集合元素印出順序也可能會不相同
- 不過仔細觀看後會發現，內容物都是一樣的，
  只是順序不同而已
- 所以如果範例程式執行出來的結果跟投影片上的不同，
  請檢查是否只是順序不同，還是連集合內的元素都不一樣

小練習

- 從這兩大堆物品中找出沒有重複的東西
- **程式碼**

```
01  itemsA = {“ 蘋果”, “ 香蕉”, “ 鳳梨”, “ 芭樂”}    # 這是第一個集合
02  itemsB = {“ 鳳梨”, “ 蘋果”, “ 水梨”, “ 蓮霧”}    # 這是第二個集合
03  print(i                  …                  把沒有重複的項目挑出來
```

- **執行結果**

```
>>>
{' 芭樂', '          …
>>>
```

- ※批改系統中須先轉成list並排序後再輸出

回家作業

請使用集合功能來完成以下問題：

- 米花市帝丹小學一年级B班正舉辦期中考試
- 數學及格的有：柯南、灰原、步美、美環、光彥
- 英文及格的有：柯南、灰原、丸尾、野口、步美
- 請分別列出
  - 數學及格且英文不及格的同學名單
  - 數學不及格且英文及格的同學名單
  - 兩者皆及格名單

- **Hint:減法、交集**

矩陣

- 因為Python沒有陣列
- 但與其相類似功能的則是list
- 因此可用list來做成二維陣列的效果
- 也可由此來模擬矩陣的操作

二維LIST的例子

```
>>>
list2x3 = [[1, 2, 3], [4, 5, 6]]
print(list2x3[1][2])
>>>
```

矩陣相乘

- 宣告三個二維 list來模擬數學矩陣相乘
- 矩陣a和矩陣b相乘
- 將所得乘積置入矩陣c

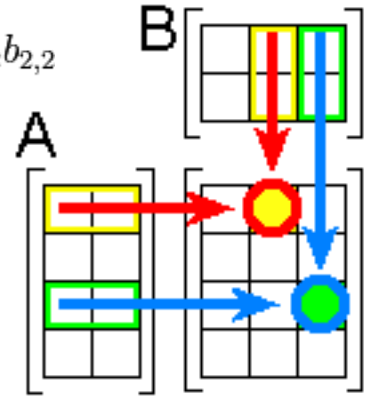
```
Matrix a =  [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
Matrix b =  [[2, 4, 6], [8, 10, 12], [14, 16, 18]]
Matrix c = Matrix a x matrix b.
The Result matrix c =  [[60, 72, 84], [132, 162, 192], [204, 252, 300]]
>>>
```

矩陣相乘的公式如下：

$$(AB)_{1,2} = \sum_{r=1}^{2} a_{1,r} b_{r,2} = a_{1,1} b_{1,2} + a_{1,2} b_{2,2}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix} = \begin{bmatrix} 60 & 72 & 84 \\ 132 & 162 & 192 \\ 204 & 252 & 300 \end{bmatrix}$$

$$(AB)_{3,3} = \sum_{r=1}^{2} a_{3,r} b_{r,3} = a_{3,1} b_{1,3} + a_{3,2} b_{2,3}$$



$$\begin{bmatrix} a[0][0] & a[0][1] & a[0][2] \\ a[1][0] & a[1][1] & a[1][2] \\ a[2][0] & a[2][1] & a[2][2] \end{bmatrix} \times \begin{bmatrix} b[0][0] & b[0][1] & b[0][2] \\ b[1][0] & b[1][1] & b[1][2] \\ b[2][0] & b[2][1] & b[2][2] \end{bmatrix} = \begin{bmatrix} c[0][0] & c[0][1] & c[0][2] \\ c[1][0] & c[1][1] & c[1][2] \\ c[2][0] & c[2][1] & c[2][2] \end{bmatrix}$$

c[i][j] += a[i][k]xb[k][j]

c[0][0]=a[0][0]*b[0][0]+a[0][1]*b[1][0]+a[0][2]*b[2][0]

c[0][1]=a[0][0]*b[0][1]+a[0][1]*b[1][1]+a[0][2]*b[2][1]

c[0][2]=a[0][0]*b[0][2]+a[0][1]*b[1][2]+a[0][2]*b[2][2]

c[1][0]=a[1][0]*b[0][0]+a[1][1]*b[1][0]+a[1][2]*b[2][0]

c[1][1]=a[1][0]*b[0][1]+a[1][1]*b[1][1]+a[1][2]*b[2][1]

c[1][2]=a[1][0]*b[0][2]+a[1][1]*b[1][2]+a[1][2]*b[2][2]

c[2][0]=a[2][0]*b[0][0]+a[2][1]*b[1][0]+a[2][2]*b[2][0]

c[2][1]=a[2][0]*b[0][1]+a[2][1]*b[1][1]+a[2][2]*b[2][1]

c[2][2]=a[2][0]*b[0][2]+a[2][1]*b[1][2]+a[2][2]*b[2][2]

矩陣相乘

```
a = [[1,2,3], [4,5,6], [7,8,9]]
b = [[2,4,6], [8,10,12], [14,16,18]]

print("Matrix a = ", a)
print("Matrix b = ", b)

#c = matrix(a, b)
c = []
for i in range(3):
    c.append([])
    for j in range(3):
        t=0
        for k in range(3):
            t+=a[i][k]*b[k][j]
        c[i].append(t)

print("Matrix c = Matrix a x matrix b.")
print("The Result matrix c = ", c)
```