嘉義市政府
Chiayi City Government

107年度物聯網應用試辦暨資料收集分析案 教育訓練

Python 基礎和數值計算

授課講師 孟璇
授課日期 108/4/25

北祥股份有限公司
PERSHING SYSTEMS CORPORATION

大綱

- Python簡介
- 安裝python
- 基本語法
- 資料分析與科學計算視覺化

Python簡介

- 創始人為吉多·范羅蘇姆 (Guido van Rossum)
- 以BBC喜劇Monty Python's Flying Circus命名

Python簡介

- Python 語言的官方介紹如下：

Python 語言是一種**簡單易學**、**功能強大**的程式語言。它具有高效率的高階資料結構、簡單而有效的物件導向程式設計方式。 Python 語言具有簡潔的語法、動態的型別、和直譯式語言的本質。由於擁有這些特質，使得 Python 語言成為多數作業系統平台上，與眾多的應用領域裡，用作處理程式腳本、**快速開發**應用程式的理想程式語言。

Python簡介

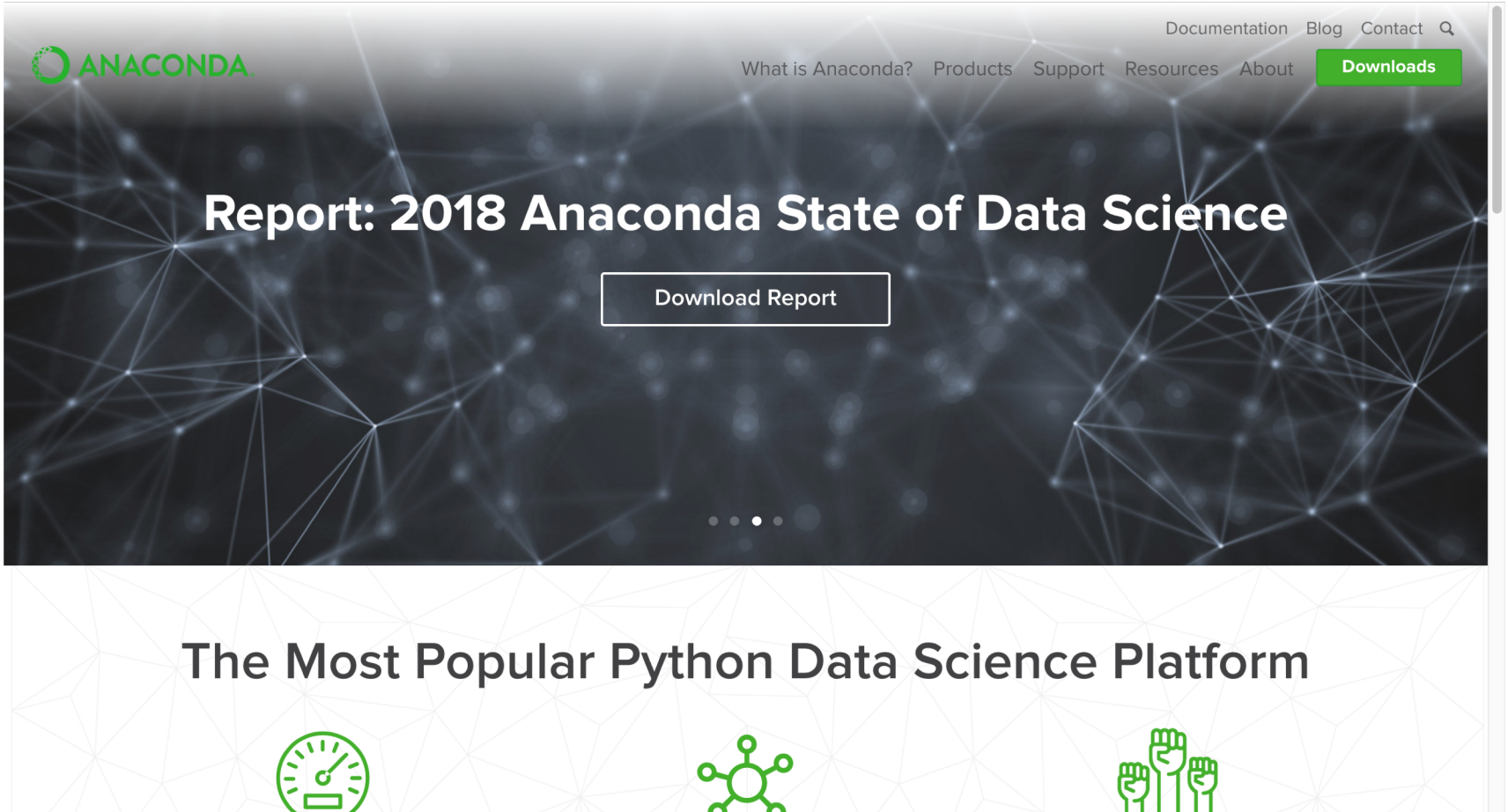
- Python語言的特色：
  - 簡單
  - 易學
  - 豐富的函式庫
  - 相容與嵌入性

安装python

- Window
- Mac
- Linux

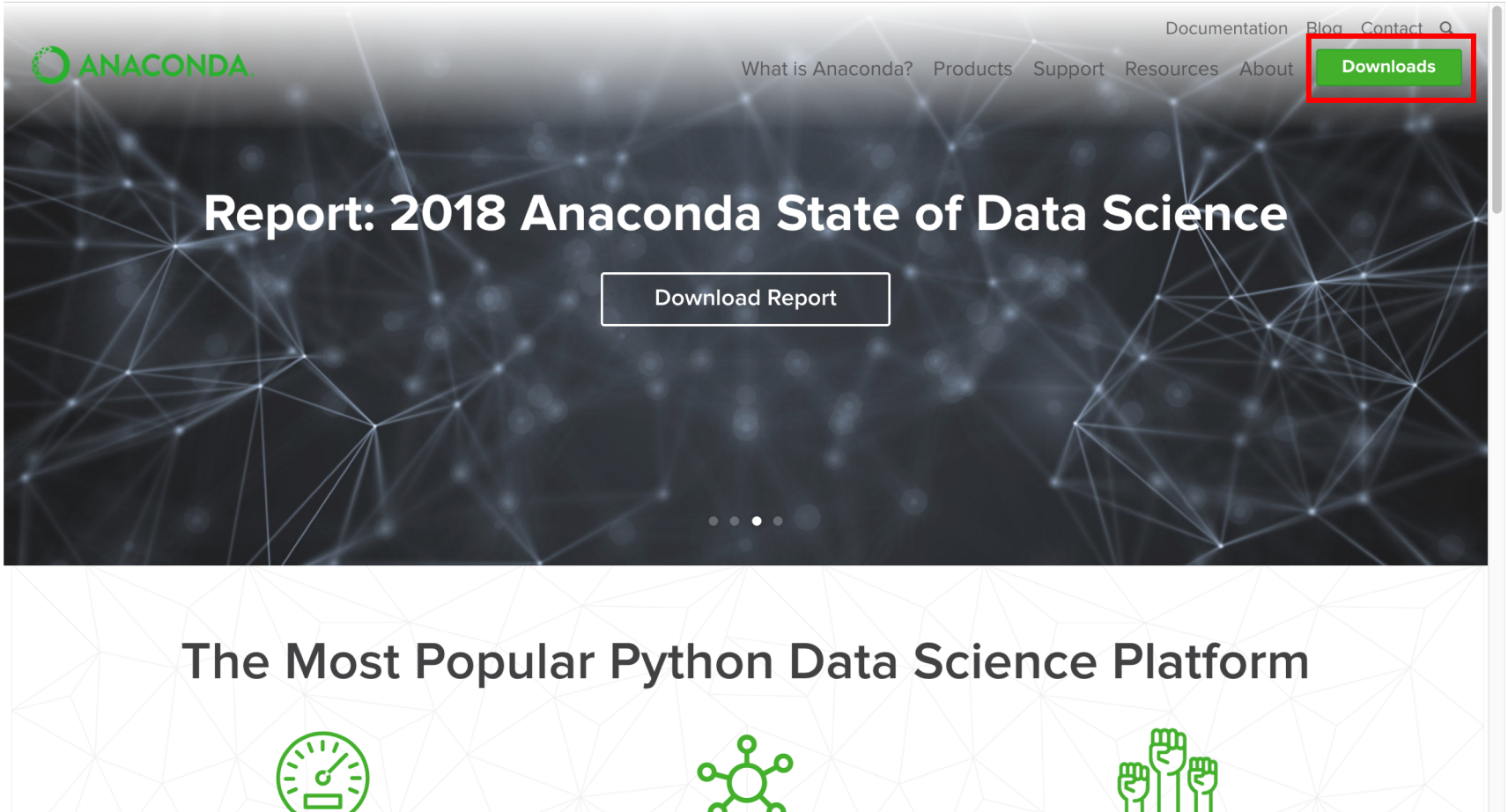
安装python

- Step1: https://www.anaconda.com/



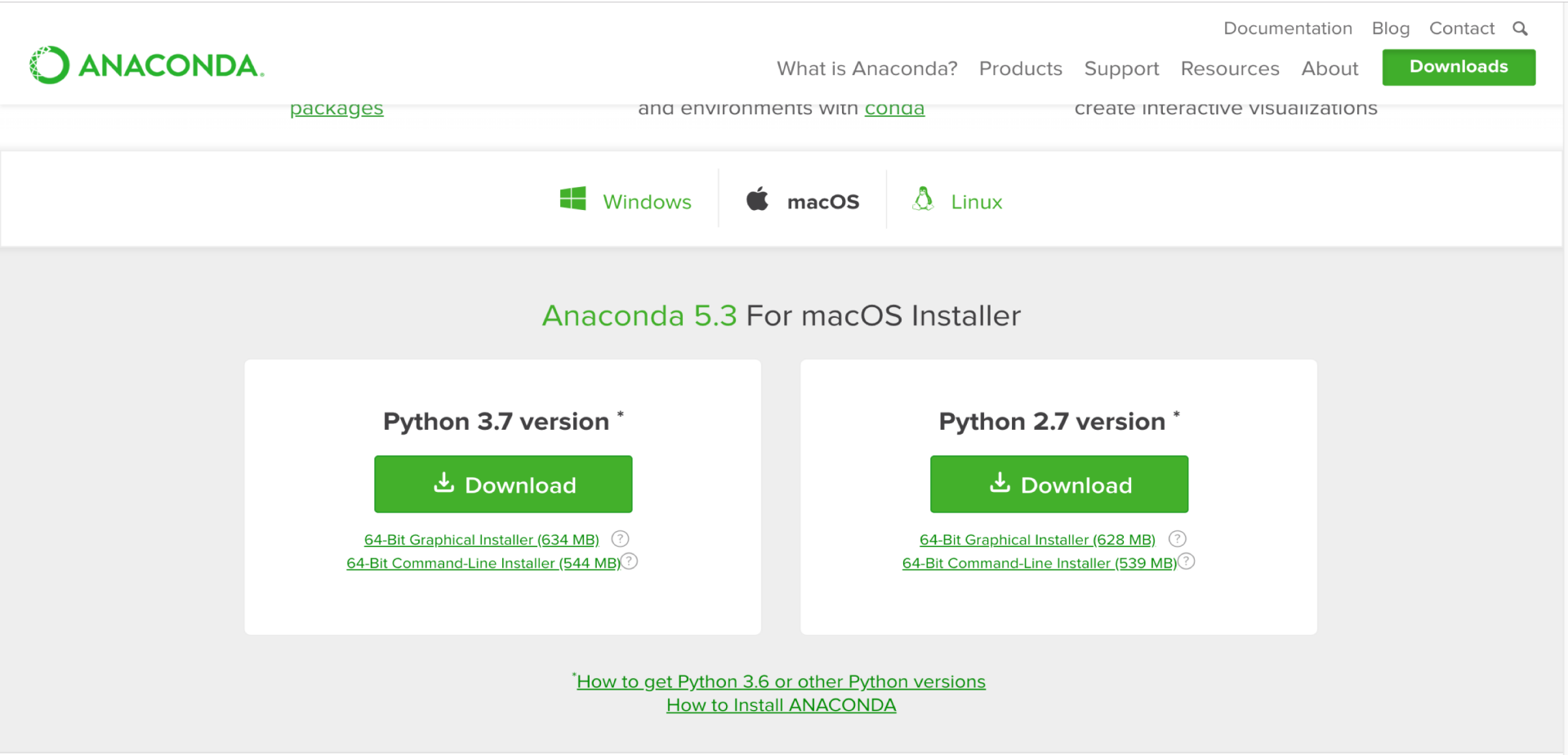
安裝python

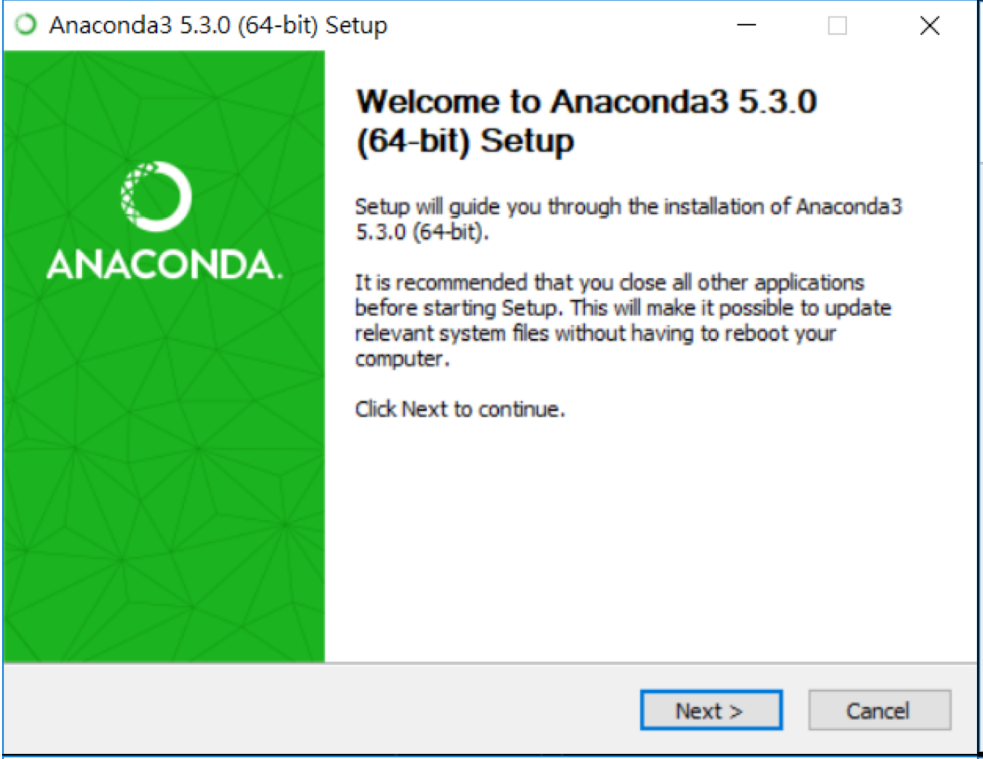
- Step2: 點選Downloads

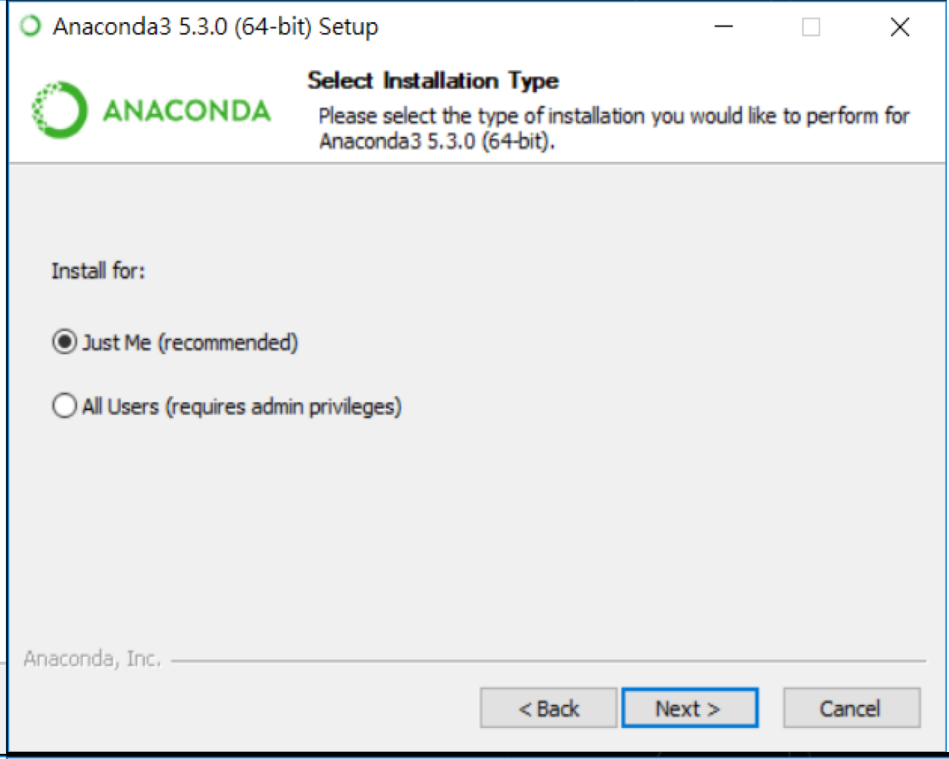


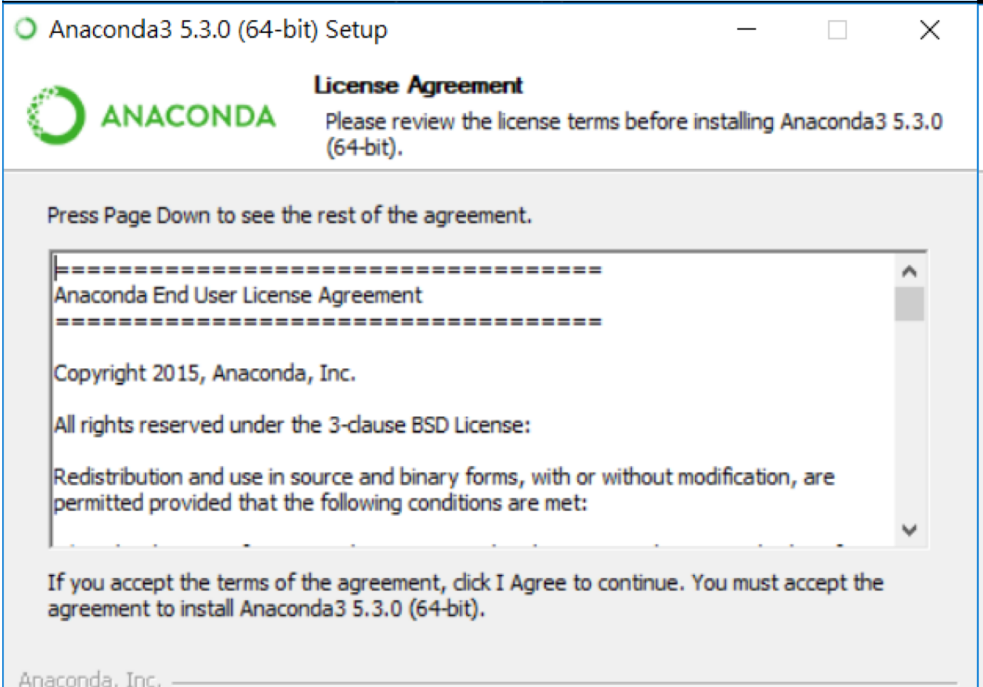
安裝python

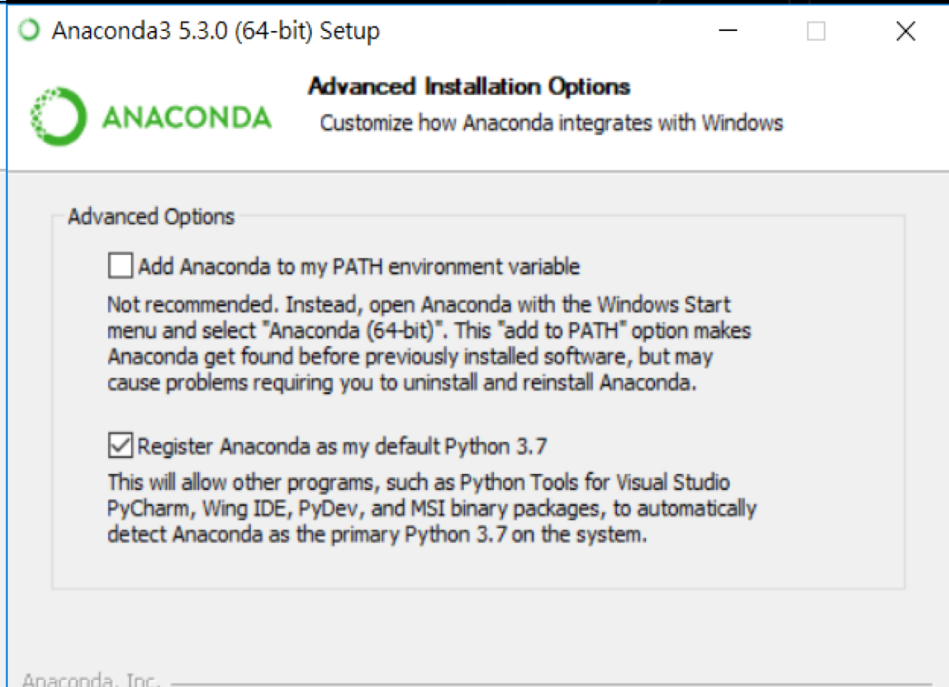
- Step3: 選擇python3.X的版本




Anaconda3 5.3.0 (64-bit) Setup
ANACONDA.
Welcome to Anaconda3 5.3.0 (64-bit) Setup
Setup will guide you through the installation of Anaconda3 5.3.0 (64-bit).
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel


Anaconda3 5.3.0 (64-bit) Setup
ANACONDA
Select Installation Type
Please select the type of installation you would like to perform for Anaconda3 5.3.0 (64-bit).
Install for:
Just Me (recommended)
All Users (requires admin privileges)
Anaconda, Inc.
< Back
Next >
Cancel


Anaconda3 5.3.0 (64-bit) Setup
ANACONDA
License Agreement
Please review the license terms before installing Anaconda3 5.3.0 (64-bit).
Press Page Down to see the rest of the agreement.
Anaconda End User License Agreement
Copyright 2015, Anaconda, Inc.
All rights reserved under the 3-clause BSD License:
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 5.3.0 (64-bit).
Anaconda, Inc.


Anaconda3 5.3.0 (64-bit) Setup
ANACONDA
Advanced Installation Options
Customize how Anaconda integrates with Windows
Advanced Options
Add Anaconda to my PATH environment variable
Not recommended. Instead, open Anaconda with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.
Register Anaconda as my default Python 3.7
This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.7 on the system.
Anaconda, Inc.

安裝python

- 執行anaconda navigator
- 點選 jupyter notebook
- Let‘s start our first code...

基本語法

資料型態

變數與運算

流程控制

基本語法

- Print (‘Hello World!’)

```
In [2]: print ('Hello world!')

        Hello world!
```

基本語法

- 若有任何不了解的函數或物件
- 輸入 help()

```
In [*]: help()
```

```
help>
```

```
Welcome to Python 3.6's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.6/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".
```

編碼

- 如果編碼錯誤: 「獢撣葉慴啣撣擃撣獢撣叢啣撣擀」
- 在windows系統裡，繁體中文編碼主要分成以下 兩種:BIG-5 以及UTF-8。
- 在Python 3.x 版以後的版本也將UTF-8 設為預設編碼，因此即便不手動執行編碼的動作也可以使用中文。

資料型態

- 在Python的世界裡面，資料會以各種不同的型別儲存在電腦裡，以供Python 程式進行
- 其中，資料型態主要可分成三種類型:
  - 數值型別
  - 字串型別
  - 容器型別

數值型別 (Numeric Type)

- int：integer的縮寫，也就是整數的意思，負責儲存沒有小數點的整數數值。
- float：浮點數，可以儲存具有小數點的數值。
- bool：boolean的縮寫，只有三種資料值， 包括True、False與None，分別代表真、假和空值三種意思。
- complex：代表數學中的虛數，以complex(x, y) 的形式出現。其中，x為實部的數字，而y為虛部的數字。除此之外，也可以在數字後面直接加j來代表虛數部，如:1+5j、6.7–4.2j 等。

字串型別(String Type)

- 各位可以用「‘ ’」、「“ ”」、「‘’‘ ’‘’」把文字包住，這些都代表字串的意思。
- 字串有可以被拆解使用，且具有前後順序的特性， 例如在字串變數x=“Hello, World” 裡面
- x 這個字串裡面有12 個字元

  依序是「H」、「e」、「l」、「l」、「o」、「,」、「 」、「W」、「o」、「r」、「l」、及「d」。

字串型別(String Type)

- 如果想要取出x裡面第8個字元，就得下x[7] 這個指令。
- Why?
- 因為從0開始算的，所以從0算到7剛好是8 個

內容	H	e	l	l	o	,		W	o	r	l	d
index	0	1	2	3	4	5	6	7	8	9	10	11

字串型別(String Type)

```
In [2]: x = 'Hello, World'
        print (x[7])
```

執行結果

```
W
```

變數

- 一般只會使用英文字母、底線、數字等國際通用的文字與符號來當作變數名稱。
- 盡可能的使用英文單字或縮寫為變數命名，對於變數所代表的意義就可以一目了然，但也不要太冗長。
- 變數也不能以數字開頭來命名。

關鍵字

- 在Python 裡面有特殊含意的字，這些字有其所代表的功能或特殊意義，因此在命名時要特別注意千萬不可以使用到這些字。
- 注意:字母有大小寫之分喔!
- 列出常用的33個關鍵字

False	assert	del	for	in	or	while
None	break	elif	from	is	pass	with
True	class	else	global	lambda	raise	yield
and	continue	except	if	nonlocal	return	
as	def	finally	import	not	try	

小練習

- x = ‘Hello, World’
- 請試著取出上面字串x中的「,」號

容器型別(Container Type)

- **tuple**: 序對，在( )裡面可以放置一個以上的資料值，有順序性，但是不能更改其內容。
- **list**: 串列，在[ ]裡面可以放置一個以上的資料值，是一種有順序且可以更改其內容的型態。
- **set**: 集合，在{ }裡面可以放置一個以上的資料值，類似數學裡面的集合概念，所以內容並無順序性。
- **dict**: 字典，是dictionary的縮寫，以Key–Value 對應的型態在{ }裡面放置一個以上的元素。字典是種配對型別，也可以放置一個以上的資料值，而 key 就是用來存取每個 value 的索引值。

容器型別(Container Type)

容器型別	tuple	list	set	dict
中文譯名	序對	串列	集合	字典
使用符號	()	[]	{}	{}
具順序性?	有	有	無	無
更改內容?	不可以	可以	可以	可以

資料型態

- 示範程式碼：

• x = 123	int 型態
• x = ‘123’	str 型態
• x = 123.0	float 型態
• x = 123+0j	complex 型態
• x = True	bool 型態
• x = (123, 456)	tuple 型態
• x = [123]	list 型態
• x = {123}	set 型態
• x = {1:123}	dict 型態

輸出

print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

- ... 代表可以用「,」隔開多個想輸出的值
- **sep** 是將多個印出的值隔開的方式，預設為一個半形空白
- **end** 是每一次print 輸出之後會在最後面加上的內容，預設為「\n」， 也就是換行符號
- **file** 為輸出串流裝置，預設為 sys.stdout，也就是標準輸出裝置(通 常是螢幕)
- **flush** 是3.3版所增加的新功能，意思是在這行print指令執行後是否 要強制將欲輸出之資訊輸出，而不是先放在緩衝區，等同一區塊的程式碼都執行完畢後才一次性輸出

輸出

- 示範程式碼：

```
print(‘Hello, World’)

print(‘Hello, World’, end=‘ ’)

print(‘Hello, ’, ‘World’, sep=‘@’)
```

輸出– ESCAPE (逸出/脫逸 字元)

- 像「‘」、「‘‘」等不會被輸出至螢幕上
- 如果需要輸出這些字元時，就需要加上「\」。
- 「\」後面接一些固定的字元時會有特別的用途所有跳脫 序列的字元在輸出時看不見，由反斜線 \ 開始，後接一 些特定字元，常見的有:

跳脫序列的字元	說明	跳脫序列的字元	說明
\a	響鈴	\v	垂直定位
\b	空白	\\	印出反斜線
\f	換頁	\t	tab鍵
\n	換行	\'	印出單引號
\r	歸位	\"	印出雙引號

輸出

- 示範程式碼

```
print(‘test’)
print(‘print with reserved words including「 \’」 「 \“」 「 \\」 「 \b」‘)
print(‘ 姓名\t性別’)
print(‘ 王小明\t男性’)
print(‘ 無敵破壞王\t女性’, end = ‘\n\n’)
print(‘ 姓名\t\t年齡’)
print(‘ 王小明\t\t16 歲’)
print(‘ 無敵破壞王\t35 歲’)
```

範例

- 輸出結果：

孟璇　　Python
http://www.google.com.tw

- 程式碼：

```
print("孟璇\t\tPython\nhttp://www.google.com.tw")
```

小練習

- 請用print輸出下列結果：

姓名	年齡	身高
王小明	20	176
陳小美	21	160

註解

- 語法：

單行註解

”””多行 註解”””

- 需要每行註解的情況，建議註解在每行的上面

範例：

單行註解是以「#」作為開頭，從每一行程式碼的第一個「#」後直到這一行結束都會被程式碼視為註解

註解範例

```
# print ('112233')
# print ('string')
'''
多行註解
敘述1
敘述2
'''
```

運算式、運算子及運算元

- 大部分的程式碼都是由判斷式及運算式組成的
- 1 + 1 = 2 這種就是運算式。
- 運算式是由運算子及運算元所組成
  - 運算子就像是這個運算的種類(如+, = 等)
  - 運算元則是要被用來運算的資料(如1, 2, integer 等)

運算式、運算子及運算元

- 運算子可以分成:
  - 算術運算子
  - 指定運算子
  - 比較運算子
  - 邏輯運算子(又稱布林運算子)
  - 其他運算子

算術運算子

- 一般數學式中常用的加減乘除

運算子	意義
+	加法運算子，執行兩運算元之加法。
–	減法運算子，執行兩運算元之減法。
*	乘法運算子，執行兩運算元之乘法。
**	指數運算子。
/	除法運算子，執行兩運算元之除法。
//	整數除法運算子，出來的結果會自動取整數
%	模數運算子，取餘數用

算術運算子

- 程式碼範例

```
1  print(1+1)          # 加法運算子
2  print(1–2)          # 減法運算子
3  print(3*2)          # 乘法運算子
4  print(2**3)         # 指數運算子
5  print(8/3)          # 除法運算子
6  print(8//3)         # 整數除法運算子
7  print(120%7)        # 模數運算子
```

輸出結果：

```
2
–1
6
8
2.6666666666666665
2
1
```

算術運算子

範例：

到Mall並拿出了剛中樂透的 50000元，「在這裡盡情的挑選採買一番吧! 」

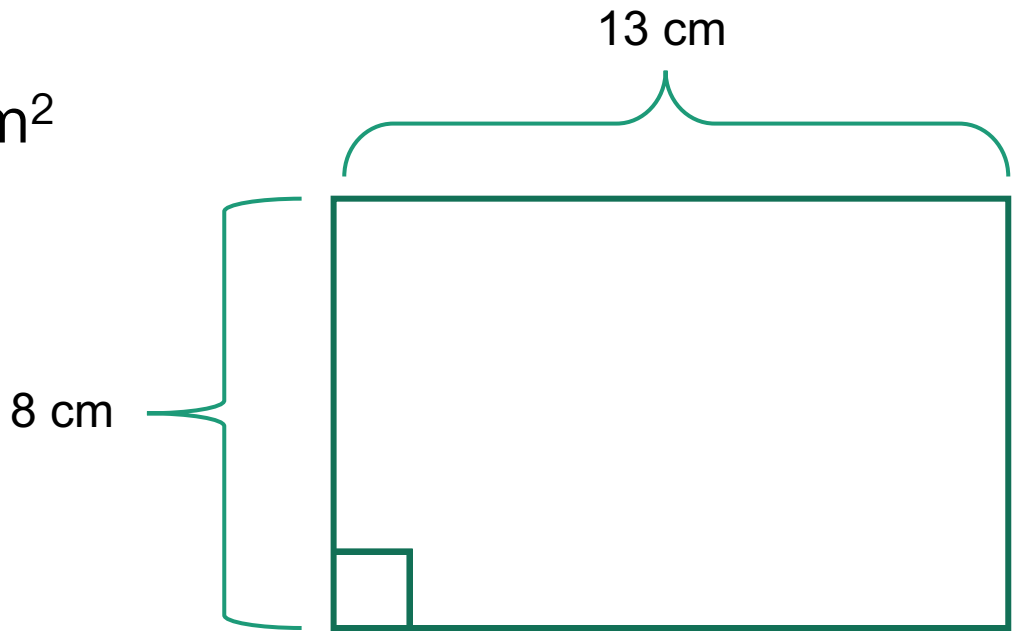
```
1  Coin = 50000                      # 一開始有50000元
2  Coin = Coin − (2000+3000)         # 買了價值2000的上衣和3000的鞋子
3  Coin = Coin − (2800*0.8)          # 買了打了8折的披風
4  print(“還剩下”, Coin,”元”)        # 印出剩餘金額
```

輸出結果：

還剩下 42760.0 元

小練習

- 請算出此矩形面積
- 輸出結果
  - 長：13 cm
  - 寬：8 cm
  - 面積：104 cm^2



關係運算子

- 程式碼範例：

```
1  print( 2 == 2 )          # 相等
2  print( 2 != 2 )          # 不相等
3  print( 6 > 4 )           # 大於
4  print( 4 < 6 )           # 小於
5  print( 6 >= 3)           # 大於等於
6  print( 6 <= 3)           # 小於等於
```

輸出結果：

```
True
False
True
True
True
False
```

關係運算子

- 比較運算子可以用在判斷變數的值是否在一個範圍之內
- 程式碼範例

```
1  integer = 8
2  print( 6 <  integer <  10 )
```

- 輸出結果

```
True
```

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
- 故兩個變數可以有完全相同的值，但是卻指向不同的物件

關係運算子

- 程式碼範例

```
1  list1 = [1, 'test']        # 賦予 list1 值
2  list2 = [1, 'test']        # 將一樣的值賦予到兩個不同的變數上
3  print(list1 == list2)      # 測試 list1 跟 list2 的值是否相等
4  print(list1 is list2)      # 測試 list1 跟 list2 的指向目標物件是否為同一個
```

- 輸出結果

```
True
False
```

關係運算子

- 程式碼範例

```
5   list1 = list2                # 將list1 指向list2 所指向的物件
6
7   print(list1 == list2)        # 測試list1 跟list2 的值是否相等
8   print(list1 is list2)        # 測試list1 跟list2 的指向目標物件是否為同一個
```

- 輸出結果

```
True
True
```

關係運算子

- 程式碼範例

```
9   str1 = ‘test’              # 賦予 str1 值
10  str2 = ‘test’              # 賦予到兩個不同的變數上
11  print(str1 == str2)        # 測試str1 跟str2 的值是否相等
12  print(str1 is str2)        # 測試str1 跟str2 的指向目標物件是否為同一個
```

- 輸出結果

```
True
True
```

關係運算子

- 在將 test 這個內容賦值給str1跟str2時，照理說str1跟str2只有值一樣，所指向的物件應該是不同的
- 但是Python為了節省記憶體空間，因此會將使用比較簡單的資料型態且其內容相同的不同變數，在短時間內指向同一個物件。
- 所以造成上面list 在測試時所指向物件雖然不同，string卻相同的情形。

關係運算子

- in 的功能則是判斷一個元素是否為一集合的元素，適用範圍包含字串、集合、清單、序對和字典。
- 除了字串外，其他幾個資料型態都屬於容器資料型態，類似集合的概念，所以可以使用in來判斷
- 字串也可以使用in，原因是因為字串可以視為由一連串長度為1 的字串所組成的長字串

關係運算子

- 程式碼範例：

```
1  tuple1 = (1, 2, '3', '4', [1, 2])      # 賦予 tuple1 值
2  list1 = [5, 6, '7', '8']               # 賦予 list1 值
3  dict1 = {9 : 'b', 0 : 'd'}             # 賦予 dict1 值
4  set1 = {'x', 'y', 'z'}                 # 賦予 set1 值
5  str1 = 'abcdefghijklmn'                # 賦予 str1 值
```

```
6  print(1 in tuple1, 3 in tuple1, 9 not in tuple1)     True   False   True
7  print(5 in list1, 7 in list1, 'a' not in list1)      True   False   True
8  print('b' in dict1, 9 in dict1)                      False  True
9  print('x' in set1, 10 in set1)                       True   False
10 print('z' in str1, 'c' in str1)                      False  True
```

關係運算子

- 範例

有一個巷口，兩側都是牛肉麵店。國士麵店的老闆說: 「我這邊的牛肉麵原價一碗500元，現在打55折給你。」而無雙麵店的老闆則說: 「我這邊的牛肉麵一碗400元，再打6折給你。」Buck認為國士麵店比較便宜，而Eric認為無雙麵比較便宜，到底誰比較便宜呢？馬上算算看摟

```
1  si = 500 * 0.55          #  國士麵店打折後的價錢
2  swang = 400 * 0.6        #  無雙麵店打折後的價錢
3  print ( si < swang)      # 判斷國士麵店打折後的價格是否低於無雙麵店打折後的價格
```

```
False
```

指派運算子

- 指派運算子是將運算過後的結果儲存在某個變數中
- 好處是可以讓運算式變簡單

運算子	例子	意義
=	a = b	將b的值賦予a
+=	a += b	a = a+ b
–=	a –= b	a = a–b
*=	a *= b	a = a* b
/=	a /= b	a = a/ b
%=	a %= b	a = a%b

指派運算子

- 使用復合指定運算子，把中樂透的範例加以簡化的方法吧

```
1  Coin = 50000                    # 一開始有50000元
2  Cost = 0                        # 假設起始花費是0元
3  Cost += 2000+3000               # 買了價值2000的上衣和3000的鞋子
4  Cost += 2800*0.8                # 買了打了8折的披風
5  Coin -= Cost
6  print("還剩下", Coin,"元")       # 印出剩餘金額
```

```
還剩下 42760.0 元
```

小練習

- 請列出一半徑為100公分的圓形之圓週率， 半徑，圓周長及圓面積
- Hint: import math
- math.pi

```
圓周率 3.141592653589793
半徑 100
圓周長 628.3185307179587
圓面積 31415.926535897932
```

其他運算子

- 還有一些比較少用，且無法歸類到上述四大類運算子之中的運算子

運算子名稱	運算子	意義
逗號運算子	,	分隔變數、資料集裡的元素等等
分號運算子	;	分隔運算式
點號運算子	.	存取類別、模組的方法或屬性
小括弧運算子	()	定義tuple、函式/方法呼叫
中括弧運算子	[]	定義list、序列形態的索引符號
大括弧運算子	{}	定義dict
冒號運算子	:	控制條件後的分隔符號或辭典元素之配對

其他運算子

```
1  import math                 #  import math 模組
2  a, b = (10, 20)             # 定義a 為10 而b 為20
3  x = 3; y = 4; z = 5         # 定義x 為3，y 為4，z 為5
4  print(math.pi)              # 呼叫math 的pi 這個屬性，也就是圓周率
5  tuple1 = (1, 2, 3, 4)       # 定義tuple1 為(1, 2, 3, 4)
6  list1 = [5, 6, 7, 8]        # 定義list1 為[5, 6, 7, 8]
7  dict1 = {'a' : 1, 'b' : 2}  # 定義dict1 中'a' 為1，'b' 為2
8  print(list1[:3])            # 印出list1 裡面第一個元素到第三個元素
```

```
3.141592653589793
[5, 6, 7]
```

小練習

- 用dict的方式叫出下列菜單的價格
- Hint: dict[key]

商品	價格
沙鍋魚頭	2000
羊肉爐	1200
豬肋排	1500
鴨胸	1600
烤全雞	688

流程控制

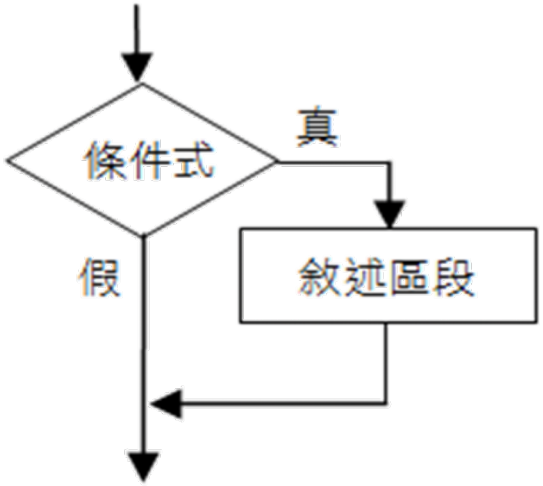
- 判斷
  - 單向判斷(if…)
  - 雙向判斷(if…else)
  - 多向判斷(if…elif…else)
  - 巢狀判斷
- 迴圈
  - for…in 迴圈
  - while 迴圈
  - 巢狀迴圈
  - 強制結束:break
  - 強制回頭:continue

判斷

- 從前面的練習中，Python 是依照什麼樣的方式在執行程式碼？
- 在沒有特殊情形之下，會從程式碼的第一行開始執行，依序向下

判斷式

- if 判斷式
- 當程式執行到if 判斷式時，判斷現在的情形是否符合撰寫所設定的條件，所以可能會因條件不符合，而發生跳過不執行某些區塊程式碼的情形

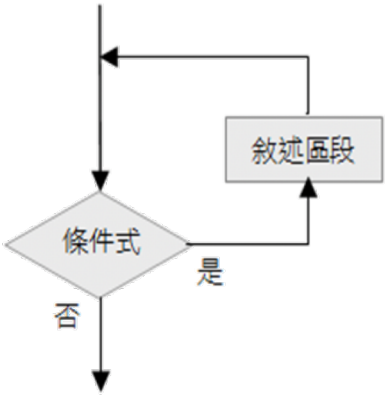


判斷

- 迴圈：

迴圈內的程式碼是可能要重複執行的程式碼， 有for 迴圈跟while迴圈兩種類型

- 當程式碼執行到迴圈時，會根據程式撰寫時 所設定的條件作判斷，若符合則重複執行迴圈裡面的程式碼



判斷

- 函數：

函數是為了完成特定功能而被撰寫的程式碼，有可以被重複呼叫的特性。

- 例外:

這裡所說的例外就是錯誤，例如縮排不一致、格式發生衝突等等。

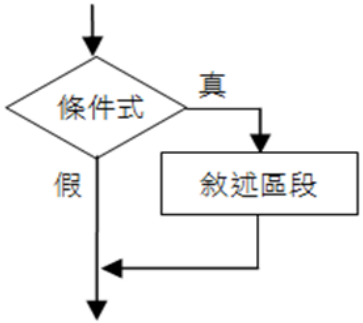
出現這種情形如果沒有妥善處理，程式的執行就會強制中斷。

單向判斷（IF...）

- if 判斷式是程式語言中最基本的判斷式
- 在這裡先以單層if 來做講解:

```
if 判斷式:
    執行的程式碼
```

```
if a > b:
    print ('a greater than b')
```



- 在if後面加上條件判斷式，後面加上「:」符號下一行開始一定要縮排，直到在此條件下欲執行的程式碼結束為止。

縮排

- Python在程式區塊中強制使用縮排
  - 令程式碼更簡潔，增加可讀性
  - 如果縮排沒有對齊，那程式碼是無法執行的 Python 會跟你說出現預期以外的縮排行為

```
1  print ('Hello')        # 預設的print方式
2   print ('Hello')       # 空一格後print
```

```
IndentationError: unexpected indent
```

縮排

- 在Python 裡面，Tab鍵也可以拿來縮排
- 注意：

就算看起來縮排一致了，但是空白鍵的縮排與Tab鍵的縮排計算方式是不一樣的， 所以縮排時儘量不要混用空白鍵和Tab 鍵!

單向判斷(if ...)

- 程式碼示範

```
1  age = 30
2
3  if  int(age) >= 20 :
4      print('市長大選近了，您具備投票資格囉')
```

```
市長大選近了，您具備投票資格囉
```

雙向判斷(if...else)

- 此種決策模式有兩種解決問題的方案，故稱為雙向判斷
- 這種時候就使用else來呈現沒有達成條件的情
- else後面也要加上「:」符號。

```
if 判斷條件:
    執行的程式碼1
else:
    執行的程式碼2
```

```
if a > b:
    print ('a is greater than b')
else:
    print ('a is lower than b')
```

雙向判斷(if...else)

- 程式碼示範

```
1  age = 16
2
3  if  int(age) >= 20 :
4      print('市長大選近了，您具備投票資格囉')
5  else:
6      print('您還不能投票喔')
```

```
您還不能投票喔
```

小練習

- 判斷成績是否及格

恭喜你及格了
沒及格也不要氣餒喔，再接再厲

if–else

- 邏輯判斷可以使用的比較/關係運算子

運算符號	意義
>	大於
<	小於
>=	大於等於
<=	小於等於
==	等於　　※ 和 「=」 (指定運算子)不同
!=	不等於

多向判斷(if...elif...else)

- 在實際使用上，常出現不只一種條件需要做判斷的情形使用elif來解決，是「else if」的縮寫，可以有許多個

```
if 判斷條件1:
    執行的程式碼1
elif 判斷條件2:
    執行的程式碼2
elif 判斷條件3:
    執行的程式碼3
…
else:
    執行的程式碼n
```

```
if (a > b) and (a > c):
    print ('a is the biggest')
elif (b > a) and (b > c):
    print ('b is the biggest')
else:
    print ('c is the biggest')
```

多向判斷(if...elif...else)

- 程式碼示範

```
1  age = 16
2
3  if  int(age) >= 20 :
4      print('市長大選近了，您具備投票資格囉')
5  elif (int(age) < 20) and (int(age) >= 18):
6      print('您的年齡還不能投票，但是刑法成年了噢')
7  else:
8      print('你的年齡在刑法跟民法都還未成年呢！')
```

```
你的年齡在刑法跟民法都還未成年呢！
```

小練習

- 若你的分數達95分以上 –> 得獎金 2000 元
- 若是90分以上到94分 –> 得獎金 1000 元
- 若是80分以上到89分 –> 得獎金 500 元
- 若不到80分則無獎金 –> 得獎金 0 元
- HINT:

  放入參數（分數），可用 and 連接 或 範圍關系運算子如:

  5 < integer <= 10

成績為 90
獎金 1000 元

重複結構簡介

- Python允許將需要連續重複執行的敘述區段改用「迴圈敘述」，不但可縮短程式的長度，而且程式易維護及增加程式可讀性
- 此種程式架構稱為「重複結構」或「迴圈」(Loop)
- Python提供二種迴圈敘述:for、while敘述

for...in 迴圈

- for迴圈裡面的expression
  - 變數(如常用的i)
  - 變數序列(iterable迭代)如list、tuple與range()函數

```
for expression in iterable:
    執行的程式碼1
```

```
for i in range(10):
    print (i)
```

range()

- 一個在for迴圈中常被使用的函數—range()
- 顧名思義，range就是範圍的意思

函數	描述
range([start], stop[, step])	建立整數序列

- 如上表所示，range() 所必須要給予的傳入值是
  - **stop**，也就是停止條件
  - **start**是起始值，預設為0
  - **step**是迭代數字，預設為1(可為負數)
- start跟step都是選擇性選項屬性
- 三者所需要的資料型態都是int
- 通常會將range()轉型成其他容器資料型態如list、tuple等

range()

- 程式碼示範

```
1  print(range(10))                 # 直接印出range()物件
2  print(list(range(10)))           # 將range物件轉型為list
3  print(list(range(3, 15, 4)))     # 設定起始值為3, 終止值為5, 迭代數字4
4  print(list(range(10, -14, -3)))  # 設定起始值為10, 終止值為-14, 迭代數字-3
5  print(tuple(range(-7, 24, 5)))   # 將range物件轉型為tuple
```

※ range() 終止值的結果並不會被輸出，所以range的範圍要+1

```
range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[3, 7, 11]
[10, 7, 4, 1, -2, -5, -8, -11]
(-7, -2, 3, 8, 13, 18, 23)
```

巢狀迴圈

- 迴圈內的敘述區段還有迴圈 → 構成「巢狀迴圈」
- 試印出一列長度為8的星號
- ********
- 印出重覆性的圖案 (5*8)

****
****
****
****
****

小練習

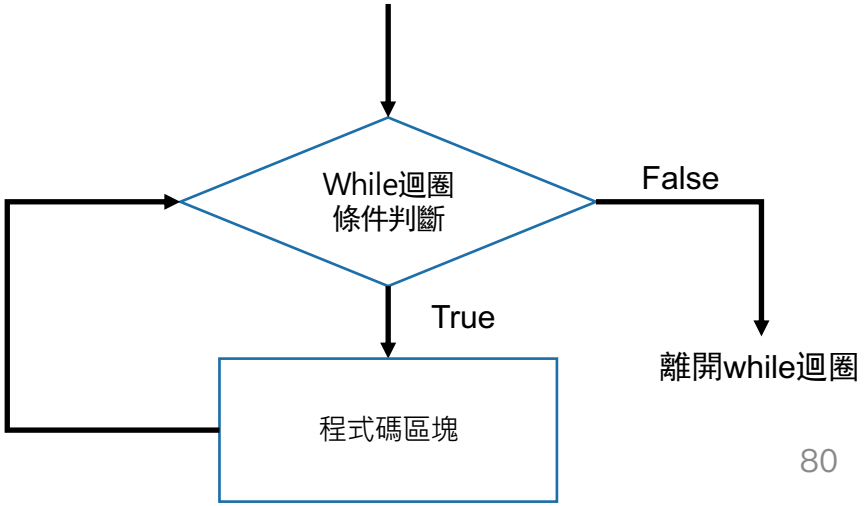
- 試寫出九九乘法表

```
...
1*1=1   2*1=2   3*1=3   4*1=4   5*1=5   6*1=6   7*1=7   8*1=8   9*1=9
1*2=2   2*2=4   3*2=6   4*2=8   5*2=10  6*2=12  7*2=14  8*2=16  9*2=18
1*3=3   2*3=6   3*3=9   4*3=12  5*3=15  6*3=18  7*3=21  8*3=24  9*3=27
1*4=4   2*4=8   3*4=12  4*4=16  5*4=20  6*4=24  7*4=28  8*4=32  9*4=36
1*5=5   2*5=10  3*5=15  4*5=20  5*5=25  6*5=30  7*5=35  8*5=40  9*5=45
1*6=6   2*6=12  3*6=18  4*6=24  5*6=30  6*6=36  7*6=42  8*6=48  9*6=54
1*7=7   2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49  8*7=56  9*7=63
1*8=8   2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64  9*8=72
1*9=9   2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

While 迴圈

- while迴圈就比較適合拿來處理沒有次序性，或者不知道總共需要執行幾次的問題 (但有次序性的也可，只是較適合)
- while迴圈的結構如下:
  - boolean_expression (布林) 是while 迴圈執行的條件
  - while 迴圈只有終止條件的敘述，因此控制變數的初始值跟調整方式就要在其他地方設定。
- **while** 迴圈正常中止，則**else:** 後面的程式碼就會被執行

```
while boolean_expression:
    欲執行的程式碼1…
else:
    欲執行的程式碼2…
```



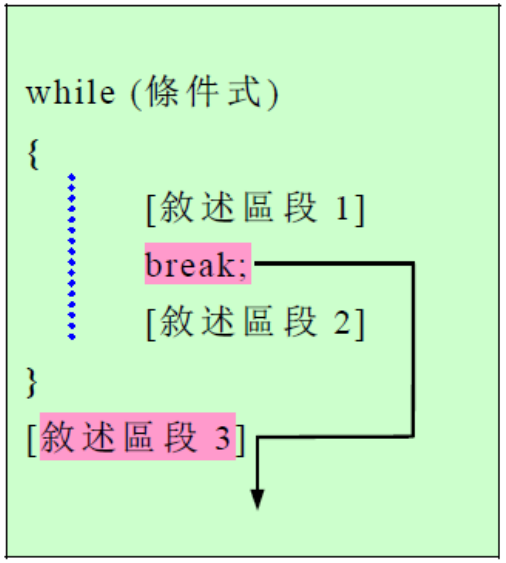
範例

```
1  sum = 0                    # 設定sum 為0
2  count = 0                  # 設定count 也就是起始條件為0
3  while count < 100:         # 當count<100 時執行以下的程式碼
4      count += 1             # 每次執行時count+1
5      sum += count           # 將count 的值加總到sum
6  else:                      # 當while 迴圈結束時
7    print (sum)              # 印出加總結果
```

```
5050
```

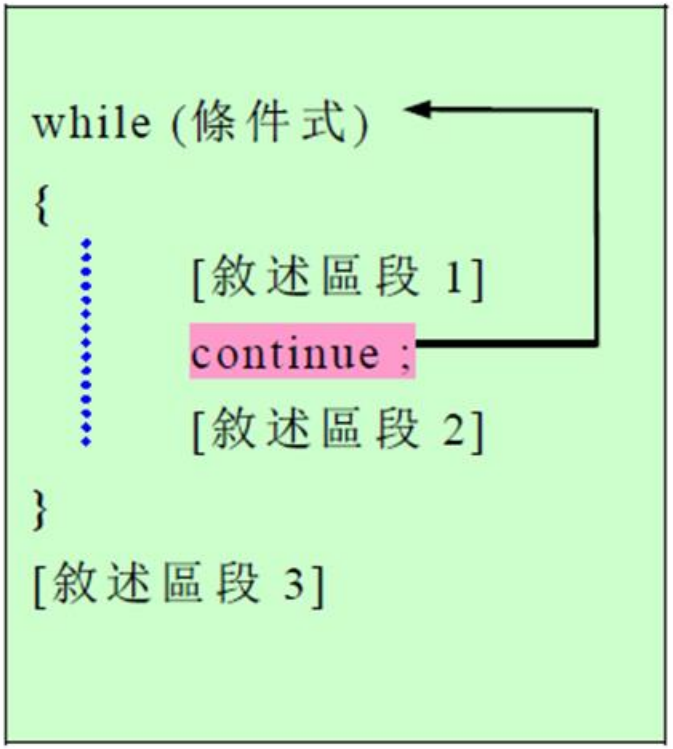
強制結束：break

- 如果在編寫有迴圈的程式碼時，有需要在終止條件達成前就強制終止迴圈使用break 指令
- 用break 跳出迴圈else: 的程式碼就不會被執行



強制回頭：continue

- continue:直接跳過後面的程式到迴圈開頭處繼續下一次執行



函數

- 當程式碼太多且會重覆出現時，可以將部份程式碼抽離主程式，寫成一段函式，有需要用到時再去呼叫它

函數

```
def 函數名稱(參數):
    函數內容
```

小練習

- 判斷1~100中是15的倍數，若是15的倍數就印出數字，不是的話印出此數字不是15的倍數，並將之寫成函數（變為可判斷任意數）

資料分析與科學計算視覺化

資料分析流程

目標定義

- 確認任務

數據採集

- 歸納
- 質量把控

數據整理

- 數據探索
- 數據清理
- 數據轉換

構建模型

- 建立模型
- 驗證模型

模型發布

- 模型視覺化
- 模型部署

常用函式庫

函式庫	簡介
numpy	支援高階大量的維度陣列與矩陣運算的函式庫
scipy	多用於矩陣相關的數值計算
matplotlib	將資料視覺化使用的函式庫
pandas	用於數據分析和探索的工具

numpy範例

```
1  import numpy as np                          # 套用函式庫
2  a = np.array([2, 0, 1, 5])                 # 創建數組
3  print (a)                                  # 輸出 a
4  print (a[:3])                              # 輸出數組內的前三個數字
5  print (a.min(), a.max())                   # 輸出a中最小和最大者
6  b = np.array([[1, 2, 3] , [4, 5, 6]])      # 創建二維數組
7  print (b * b)                              # 輸出相乘兩次的結果
8  print (b[1][1:2])                          # 輸出數組內的第2個數字
```

※ 維度不相同時，無法相乘。矩陣相乘最好使用 np.dot

```
[2 0 1 5]
[2 0 1]
0 5
[[ 1 4 9] [16 25 36]]
[5]
```

更多參考：
http://www.numpy.org/

小練習

- 取出 $\begin{bmatrix} 1 & 8 & 10 \\ 25 & 3 & 4 \end{bmatrix}$ 中 25 的數字

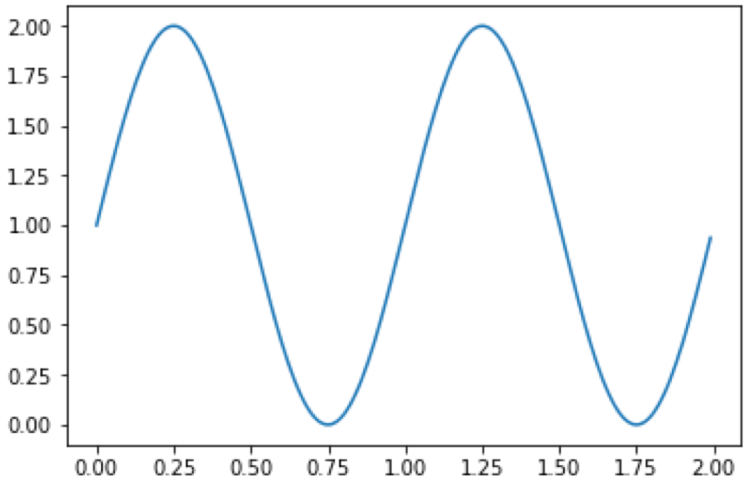
```
[25]
25
```

scipy

- scipy 依賴於 numpy 所以安裝此函式庫前需有 numpy函式庫
- scipy 包含的功能有最優化、線性代數、積分、常微分方程求解...

matplotlib範例

```
1  import numpy as np                                  # 套用函式庫
2  import matplotlib.pyplot as plt
3  t = np.arange(0.0, 2.0, 0.01)                      # 宣告 t
4  s = 1 + np.sin(2 * np.pi * t)                      # 宣告 s
5  plt.plot(t, s)                                     # 畫出以t為x軸，s為y軸的圖
6  plt.show()
```



更多參考：
https://matplotlib.org/index.html

pandas

- pandas.DataFrame
- pandas.read_csv
- Head
- Describe
- loc
- Isna

pandas.DataFrame()

- pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)
  - data：支援 numpy ndarray (structured or homogeneous), dict, or DataFrame
  - index： 用於索引
  - columns：欄位名稱
  - dtype：數據類型
  - copy：要不要指定一個新的物件

pandas.DataFrame()

- 程式碼範例

```
1  import pandas as pd                                  # 套用函式庫
2
3  d = {'col1': [1, 2], 'col2': [3, 4]}                 # 創建數據
4  df = pd.DataFrame(data=d)                            # 放入DataFrame中
5  print (df)                                           # print 資料
```

```
   col1 col2
0     3     1
1     2     4
```

pandas.read_csv()

- pandas.read_csv(filepath, … , sep=‘,’, names=None)
  - filepath：讀入資料的路徑及檔案名
  - sep：分開資料的符號
  - names：假設資料沒有欄位，給予欄位名稱

pandas.read_csv()

- 程式碼範例

```
1  import pandas as pd                          # 套用函式庫
2
3  df = pd.read_csv('menu.csv')                 # 讀入數據
4  print (df)                                   # print 資料
```

	菜單	價格
0	烤乳豬	3500
1	烤鴨	3200
2	牛排	4000
3	炸雞	500
4	薯條	100
5	啤酒	120
6	拼盤	1200

head

- 程式碼範例

```
1  import pandas as pd                          # 套用函式庫
2
3  df = pd.read_csv('menu.csv')                 # 讀入數據
4  # print (df)                                 # print 資料
5  print (df.head())                            # print 前五筆資料
```

	菜單	價格
0	烤乳豬	3500
1	烤鴨	3200
2	牛排	4000
3	炸雞	500
4	薯條	100

describe

- 程式碼範例

```
1  import pandas as pd                          # 套用函式庫
2
3  df = pd.read_csv('menu.csv')                 # 讀入數據
4  # print (df)                                 # print 資料
5  print (df.describe())                        # print 描述性資料
```

	價格
count	7.000000
mean	1802.857143
std	1705.498112
min	100.000000
25%	310.000000
50%	1200.000000
75%	3350.000000
max	4000.000000

loc

- 程式碼範例

```
1  import pandas as pd                          # 套用函式庫
2
3  df = pd.read_csv('menu.csv')                 # 讀入數據
4  # print (df)                                 # print 資料
5  print (df.loc[3])                            # print 第 ? 列資料
```

菜單	炸雞
價格	500

loc

- 程式碼範例

```
1  import pandas as pd                          # 套用函式庫
2
3  df = pd.read_csv('menu.csv')                 # 讀入數據
4  # print (df)                                 # print 資料
5  # print (df.loc[3])                          # print 第4列資料
6  print (df.loc[3,'菜單'])                     # print 第四列欄位是菜單的值
```

```
炸雞
```

loc

- 程式碼範例

```
1  import pandas as pd                              # 套用函式庫
2
3  df = pd.read_csv('menu.csv')                     # 讀入數據
4  # print (df)                                     # print 資料
5  # print (df.loc[3])                              # print 第4列資料
6  # print (df.loc[3,'菜單'])                       # print 第四列欄位是菜單的值
7  print ( df.loc[df['價格'] > 1000] )              # 找出價格大於1000的所有資料
```

	菜單	價格
0	烤乳豬	3500
1	烤鴨	3200
2	牛排	4000
6	拼盤	1200

Isna()

- 程式碼範例

```
1  import pandas as pd                                  # 套用函式庫
2
3  df = pd.read_csv(‘menu.csv’)                         # 讀入數據
4  # print (df)                                         # print 資料
5  # print ( df.loc[df['價格'] > 1000] )                # 找出價格大於1000的所有資料
6
7  print (df.loc[df['價格'].isna()])                    # 找出沒有資料的列
```

	菜單	價格
1	烤鴨	NaN

Pandas 實作

背景：
本次實作內容為空氣汙染與健康關聯模型中的一部份，空氣汙染與健康關聯模型希望探討各種空氣濃度資訊與過敏性疾病的關聯性。

實作目標：
藉由將空氣資訊及氣象資訊整合，熟悉python及pandas基本操作

材料：
空氣資訊檔案
氣象資訊檔案
測站資訊檔案(包含空氣測站、氣象測站)

讀入空氣資料

```
import numpy as np
import pandas as pd

#讀入空氣測站資訊
AirInfo = pd.read_csv("./practice/All201001_201805_v1.csv", encoding = "cp950")
AirInfo.columns = ['City', 'Date', 'Station', 'SO2', 'CO', 'CO2', 'O3', 'O3_Max_8_HR', 'PM10',
                   'PM25', 'NOx', 'NO', 'NO2', 'THC', 'NMHC', 'CH4', 'WIND_SPEED', 'WS_HR',
                   'AMB_TEMP', 'RAIN_INT', 'PH_RAIN', 'RH', 'RAIN_COND', 'PSI']
```

	City	Date	Station	SO2	CO	CO2	O3	O3_Max_8_HR	PM10	PM25	\
0	臺北市	2010/1/31	松山	6.3	1.32	446.69	15.9		37	118	65
1	臺北市	2010/1/31	古亭	4.6	1.11	-	20.6		50	89	58
2	臺北市	2010/1/31	中山	6.7	1.44	-	13.9		32	116	68
3	臺北市	2010/1/31	萬華	4.7	1.36	-	16.1		39	88	60
4	臺北市	2010/1/31	陽明	3.3	0.33	404.25	54		62	36	28
5	臺北市	2010/1/31	士林	5.6	1.02	-	20.8		52	98	54
6	臺北市	2010/1/30	松山	6.3	0.87	419.48	24.5		43	121	64
7	臺北市	2010/1/30	古亭	4.7	0.95	-	26.5		49	113	67
8	臺北市	2010/1/30	中山	6.3	1.05	-	23.9		38	127	71
9	臺北市	2010/1/30	萬華	-	-	-	-		17	-	-
10	臺北市	2010/1/30	陽明	3	0.43	400.88	50.5		59	38	28
11	臺北市	2010/1/30	士林	5.2	0.78	-	34.2		52	109	57
12	臺北市	2010/1/29	松山	3.6	0.81	408.9	28.9		36	93	57
13	臺北市	2010/1/29	古亭	3.4	0.82	-	31.5		37	89	58
14	臺北市	2010/1/29	中山	4.6	0.92	-	28.4		34	103	58

將缺值轉換為NaN

```
for col in AirInfo.columns:
    if col not in ['City', 'Date', 'Station']:
        AirInfo[col] = AirInfo[col].str.replace('-', 'NaN')
```

	City	Date	Station	SO2	CO	CO2	O3	O3_Max_8_HR	PM10	PM25	\
0	臺北市	2010/1/31	松山	6.3	1.32	446.69	15.9		37	118	65
1	臺北市	2010/1/31	古亭	4.6	1.11	NaN	20.6		50	89	58
2	臺北市	2010/1/31	中山	6.7	1.44	NaN	13.9		32	116	68
3	臺北市	2010/1/31	萬華	4.7	1.36	NaN	16.1		39	88	60
4	臺北市	2010/1/31	陽明	3.3	0.33	404.25	54		62	36	28
5	臺北市	2010/1/31	士林	5.6	1.02	NaN	20.8		52	98	54
6	臺北市	2010/1/30	松山	6.3	0.87	419.48	24.5		43	121	64
7	臺北市	2010/1/30	古亭	4.7	0.95	NaN	26.5		49	113	67
8	臺北市	2010/1/30	中山	6.3	1.05	NaN	23.9		38	127	71
9	臺北市	2010/1/30	萬華	NaN	NaN	NaN	NaN		17	NaN	NaN
10	臺北市	2010/1/30	陽明	3	0.43	400.88	50.5		59	38	28
11	臺北市	2010/1/30	士林	5.2	0.78	NaN	34.2		52	109	57
12	臺北市	2010/1/29	松山	3.6	0.81	408.9	28.9		36	93	57
13	臺北市	2010/1/29	古亭	3.4	0.82	NaN	31.5		37	89	58
14	臺北市	2010/1/29	中山	4.6	0.92	NaN	28.4		34	103	58
15	臺北市	2010/1/29	萬華	NaN	NaN	NaN	NaN		36	NaN	NaN
16	臺北市	2010/1/29	陽明	3	0.5	403.63	52.2		54	32	22

嘗試轉換為數字(原本為字串)

```
print(type(AirInfo['PM25'][0]))
```

```
<class 'str'>
```

```
for col in AirInfo.columns:
    if col not in ['City', 'Date', 'Station']:
        AirInfo[col] = AirInfo[col].str.replace('-', 'NaN')
        AirInfo[col] = AirInfo[col].astype(float)
```

```
ValueError: could not convert string to float: '1,000'
```

將「,」轉換為「 」

```
for col in AirInfo.columns:
    if col not in ['City', 'Date', 'Station']:
        AirInfo[col] = AirInfo[col].str.replace('-', 'NaN')
        AirInfo[col] = AirInfo[col].str.replace(',', '')
        AirInfo[col] = AirInfo[col].astype(float)
```

```
print(type(AirInfo['PM25'][0]))
```

```
<class 'numpy.float64'>
```

將欄位合併

```
AirInfo['Station'] = AirInfo['City'] + AirInfo['Station']
del AirInfo['City']
```

	City	Date	Station	SO2	CO	CO2	O3	O3_Max_8_HR	PM10	\
0	臺北市	2010/1/31	松山	6.3	1.32	446.69	15.9	37.0	118.0	
1	臺北市	2010/1/31	古亭	4.6	1.11	NaN	20.6	50.0	89.0	
2	臺北市	2010/1/31	中山	6.7	1.44	NaN	13.9	32.0	116.0	
3	臺北市	2010/1/31	萬華	4.7	1.36	NaN	16.1	39.0	88.0	
4	臺北市	2010/1/31	陽明	3.3	0.33	404.25	54.0	62.0	36.0	
5	臺北市	2010/1/31	士林	5.6	1.02	NaN	20.8	52.0	98.0	
6	臺北市	2010/1/30	松山	6.3	0.87	419.48	24.5	43.0	121.0	
7	臺北市	2010/1/30	古亭	4.7	0.95	NaN	26.5	49.0	113.0	
8	臺北市	2010/1/30	中山	6.3	1.05	NaN	23.9	38.0	127.0	
9	臺北市	2010/1/30	萬華	NaN	NaN	NaN	NaN	17.0	NaN	
10	臺北市	2010/1/30	陽明	3.0	0.43	400.88	50.5	59.0	38.0	
11	臺北市	2010/1/30	士林	5.2	0.78	NaN	34.2	52.0	109.0	
12	臺北市	2010/1/29	松山	3.6	0.81	408.90	28.9	36.0	93.0	
13	臺北市	2010/1/29	古亭	3.4	0.82	NaN	31.5	37.0	89.0	
14	臺北市	2010/1/29	中山	4.6	0.92	NaN	28.4	34.0	103.0	
15	臺北市	2010/1/29	萬華	NaN	NaN	NaN	NaN	36.0	NaN	
16	臺北市	2010/1/29	陽明	3.0	0.50	403.63	52.2	54.0	32.0	
17	臺北市	2010/1/29	士林	3.7	0.63	NaN	47.7	51.0	84.0	

	Date	Station	SO2	CO	CO2	O3	O3_Max_8_HR	PM10	PM25	\
0	2010/1/31	臺北市松山	6.3	1.32	446.69	15.9	37.0	118.0	65.0	
1	2010/1/31	臺北市古亭	4.6	1.11	NaN	20.6	50.0	89.0	58.0	
2	2010/1/31	臺北市中山	6.7	1.44	NaN	13.9	32.0	116.0	68.0	
3	2010/1/31	臺北市萬華	4.7	1.36	NaN	16.1	39.0	88.0	60.0	
4	2010/1/31	臺北市陽明	3.3	0.33	404.25	54.0	62.0	36.0	28.0	
5	2010/1/31	臺北市士林	5.6	1.02	NaN	20.8	52.0	98.0	54.0	
6	2010/1/30	臺北市松山	6.3	0.87	419.48	24.5	43.0	121.0	64.0	
7	2010/1/30	臺北市古亭	4.7	0.95	NaN	26.5	49.0	113.0	67.0	
8	2010/1/30	臺北市中山	6.3	1.05	NaN	23.9	38.0	127.0	71.0	
9	2010/1/30	臺北市萬華	NaN	NaN	NaN	NaN	17.0	NaN	NaN	
10	2010/1/30	臺北市陽明	3.0	0.43	400.88	50.5	59.0	38.0	28.0	
11	2010/1/30	臺北市士林	5.2	0.78	NaN	34.2	52.0	109.0	57.0	
12	2010/1/29	臺北市松山	3.6	0.81	408.90	28.9	36.0	93.0	57.0	
13	2010/1/29	臺北市古亭	3.4	0.82	NaN	31.5	37.0	89.0	58.0	
14	2010/1/29	臺北市中山	4.6	0.92	NaN	28.4	34.0	103.0	58.0	
15	2010/1/29	臺北市萬華	NaN	NaN	NaN	NaN	36.0	NaN	NaN	
16	2010/1/29	臺北市陽明	3.0	0.50	403.63	52.2	54.0	32.0	22.0	
17	2010/1/29	臺北市士林	3.7	0.63	NaN	47.7	51.0	84.0	50.0	

排序、重新編號

```
AirInfo["Date"] = pd.to_datetime(AirInfo["Date"])
AirInfo = AirInfo.sort_values(by=['Station', 'Date'])
AirInfo = AirInfo.reset_index(drop=True)
```

	Date	Station	SO2	CO	CO2	O3	O3_Max_8_HR	PM10	PM25	\
0	2010-01-01	南投縣南投	3.7	0.70	NaN	17.6	40.0	96.0	49.0	
1	2010-01-02	南投縣南投	3.3	0.64	NaN	15.6	28.0	87.0	44.0	
2	2010-01-03	南投縣南投	2.7	0.53	NaN	21.3	34.0	54.0	26.0	
3	2010-01-04	南投縣南投	3.6	0.67	NaN	20.8	46.0	112.0	55.0	
4	2010-01-05	南投縣南投	NaN	NaN	NaN	NaN	33.0	82.0	40.0	
5	2010-01-06	南投縣南投	3.7	0.72	NaN	4.9	8.0	48.0	24.0	
6	2010-01-07	南投縣南投	3.3	0.62	NaN	10.9	21.0	45.0	21.0	
7	2010-01-08	南投縣南投	3.3	0.74	NaN	6.8	9.0	45.0	22.0	
8	2010-01-09	南投縣南投	3.1	0.51	NaN	20.5	41.0	64.0	28.0	
9	2010-01-10	南投縣南投	2.4	0.49	NaN	21.3	42.0	71.0	33.0	
10	2010-01-11	南投縣南投	3.3	0.65	NaN	13.7	37.0	63.0	30.0	
11	2010-01-12	南投縣南投	4.9	0.78	NaN	8.4	14.0	56.0	30.0	
12	2010-01-13	南投縣南投	4.5	0.65	NaN	17.6	39.0	75.0	38.0	
13	2010-01-14	南投縣南投	4.8	0.73	NaN	18.1	46.0	89.0	46.0	
14	2010-01-15	南投縣南投	3.3	0.61	NaN	25.2	63.0	101.0	58.0	
15	2010-01-16	南投縣南投	3.1	0.59	NaN	28.3	70.0	96.0	56.0	
16	2010-01-17	南投縣南投	3.0	0.60	NaN	27.2	66.0	90.0	52.0	
17	2010-01-18	南投縣南投	2.8	0.56	NaN	22.0	49.0	90.0	43.0	

第一部分的資料整理

```
import numpy as np
import pandas as pd

#讀入空氣測站資訊
AirInfo = pd.read_csv("./practice/All201001_201805_v1.csv", encoding = "cp950")
AirInfo.columns = ['City', 'Date', 'Station', 'SO2', 'CO', 'CO2', 'O3', 'O3_Max_8_HR', 'PM10',
                   'PM25', 'NOx', 'NO', 'NO2', 'THC', 'NMHC', 'CH4', 'WIND_SPEED', 'WS_HR',
                   'AMB_TEMP', 'RAIN_INT', 'PH_RAIN', 'RH', 'RAIN_COND', 'PSI']

for col in AirInfo.columns:
    if col not in ['City', 'Date', 'Station']:
        AirInfo[col] = AirInfo[col].str.replace('-', 'NaN')
        AirInfo[col] = AirInfo[col].str.replace(',', '')
        AirInfo[col] = AirInfo[col].astype(float)

AirInfo['Station'] = AirInfo['City'] + AirInfo['Station']
del AirInfo['City']


AirInfo["Date"] = pd.to_datetime(AirInfo["Date"])
AirInfo = AirInfo.sort_values(by=['Station', 'Date'])
AirInfo = AirInfo.reset_index(drop=True)

print(AirInfo)
```

```
[228793 rows x 23 columns]
```

日期問題？

```
print('2010年1月1日到2018年5月7日共有幾天：', len(pd.date_range(start='1/1/2010', end='5/7/2018')))
print('==========================================================')

for Station in list(set(AirInfo['Station'])):
    print(Station + '：', len(AirInfo.loc[AirInfo['Station']==Station]))
```

```
2010年1月1日到2018年5月7日共有幾天： 3049
==========================================================
臺北市陽明： 3025
雲林縣斗六： 3031
高雄市美濃： 3016
金門縣金門： 3030
桃園市龍潭： 3021
高雄市大寮： 2989
臺北市士林： 3031
雲林縣崙背： 3028
新北市板橋： 3025
嘉義縣新港： 3030
高雄市林園： 2924
花蓮縣花蓮： 3025
連江縣馬祖： 3031
臺北市松山： 3020
高雄市復興： 3031
桃園市中壢： 3026
臺北市古亭： 3029
```

缺了那些天？

```
Station_臺北市陽明 = AirInfo.loc[AirInfo['Station']=='臺北市陽明']
print(set(pd.date_range(start='1/1/2010', end='5/7/2018'))-set(Station_臺北市陽明['Date']))
```

```
{Timestamp('2015-08-11 00:00:00', freq='D'), Timestamp('2017-11-24 00:00:00', freq='D'), Timestamp('2016-12-04 00:00:00', freq
='D'), Timestamp('2015-08-10 00:00:00', freq='D'), Timestamp('2016-11-28 00:00:00', freq='D'), Timestamp('2016-12-01 00:00:00',
freq='D'), Timestamp('2013-07-13 00:00:00', freq='D'), Timestamp('2015-08-08 00:00:00', freq='D'), Timestamp('2016-01-19 00:00:
00', freq='D'), Timestamp('2016-08-27 00:00:00', freq='D'), Timestamp('2016-11-30 00:00:00', freq='D'), Timestamp('2016-02-18 0
0:00:00', freq='D'), Timestamp('2016-04-30 00:00:00', freq='D'), Timestamp('2016-07-29 00:00:00', freq='D'), Timestamp('2017-05
-07 00:00:00', freq='D'), Timestamp('2017-07-02 00:00:00', freq='D'), Timestamp('2015-08-09 00:00:00', freq='D'), Timestamp('20
17-08-16 00:00:00', freq='D'), Timestamp('2016-10-20 00:00:00', freq='D'), Timestamp('2017-08-22 00:00:00', freq='D'), Timestam
p('2016-12-03 00:00:00', freq='D'), Timestamp('2016-07-28 00:00:00', freq='D'), Timestamp('2017-07-03 00:00:00', freq='D'), Tim
estamp('2017-07-18 00:00:00', freq='D')}
```

```
Station_臺北市陽明 = AirInfo.loc[AirInfo['Station']=='臺北市陽明']
print(set(pd.date_range(start='1/1/2010', end='5/7/2018').astype(str))-set(Station_臺北市陽明['Date'].astype(str)))
```

```
{'2017-08-22', '2017-07-02', '2016-07-29', '2016-08-27', '2016-01-19', '2015-08-08', '2016-07-28', '2016-11-30', '2015-08-11',
'2013-07-13', '2016-11-28', '2016-12-04', '2015-08-09', '2016-12-01', '2017-08-16', '2017-05-07', '2016-10-20', '2017-07-18',
'2017-07-03', '2017-11-24', '2015-08-10', '2016-12-03', '2016-02-18', '2016-04-30'}
```

填補日期

```
Station_臺北市陽明 = AirInfo.loc[AirInfo['Station']=='臺北市陽明'].reset_index(drop=True)
Station_臺北市陽明 = Station_臺北市陽明.set_index('Date')
Station_臺北市陽明 = Station_臺北市陽明.reindex(pd.date_range(start='1/1/2010', end='5/7/2018'))
Station_臺北市陽明 = Station_臺北市陽明.reset_index()
Station_臺北市陽明.rename(columns={'index': 'Date'}, inplace=True)
Station_臺北市陽明 = Station_臺北市陽明.sort_values(by=['Date']).reset_index(drop = True)
Station_臺北市陽明['Station'] = '臺北市陽明'

print(Station_臺北市陽明)
```

	Date	Station	SO2	CO	CO2	O3	O3_Max_8_HR	PM10	PM25	\
0	2010-01-01	臺北市陽明	2.6	0.27	397.66	36.8		41.0	25.0	17.0
1	2010-01-02	臺北市陽明	1.2	0.32	399.48	41.0		42.0	14.0	11.0
2	2010-01-03	臺北市陽明	3.4	0.45	404.45	43.8		54.0	40.0	28.0
3	2010-01-04	臺北市陽明	2.5	0.40	402.30	42.8		50.0	38.0	27.0
4	2010-01-05	臺北市陽明	2.6	0.40	398.30	36.8		38.0	9.0	6.0
5	2010-01-06	臺北市陽明	0.7	0.23	383.19	39.2		40.0	5.0	2.0
6	2010-01-07	臺北市陽明	1.2	0.25	383.43	42.2		43.0	5.0	4.0
7	2010-01-08	臺北市陽明	1.4	0.31	385.07	41.6		45.0	5.0	4.0
8	2010-01-09	臺北市陽明	1.4	0.20	380.27	39.6		41.0	7.0	5.0
9	2010-01-10	臺北市陽明	1.7	0.28	385.08	36.9		45.0	11.0	8.0
10	2010-01-11	臺北市陽明	1.3	0.38	387.92	34.2		41.0	6.0	4.0
11	2010-01-12	臺北市陽明	4.0	0.48	394.47	29.9		43.0	17.0	12.0
12	2010-01-13	臺北市陽明	4.5	0.32	388.95	48.8		55.0	48.0	27.0
13	2010-01-14	臺北市陽明	5.4	0.27	387.28	44.0		47.0	42.0	21.0
14	2010-01-15	臺北市陽明	3.5	0.22	383.36	44.5		47.0	27.0	16.0
15	2010-01-16	臺北市陽明	2.9	0.25	385.12	49.3		55.0	17.0	10.0
16	2010-01-17	臺北市陽明	1.6	0.26	383.68	48.8		50.0	13.0	8.0
17	2010-01-18	臺北市陽明	1.9	0.25	384.35	46.2		48.0	17.0	12.0

[3049 rows x 23 columns]

各測站填補日期

```
AirInfo_AddLossDate = pd.DataFrame(columns = AirInfo.columns)
for Station in list(set(AirInfo['Station'])):
    globals()['Station_' + Station] = AirInfo.loc[AirInfo['Station']==Station].reset_index(drop=True)
    globals()['Station_' + Station] = globals()['Station_' + Station].set_index('Date')
    globals()['Station_' + Station] = globals()['Station_' + Station].reindex(pd.date_range(start='1/1/2010', end='5/7/2018'))
    globals()['Station_' + Station] = globals()['Station_' + Station].reset_index()
    globals()['Station_' + Station].rename(columns={'index': 'Date'}, inplace=True)
    globals()['Station_' + Station] = globals()['Station_' + Station].sort_values(by=['Date']).reset_index(drop = True)
    globals()['Station_' + Station]['Station'] = Station
    AirInfo_AddLossDate = AirInfo_AddLossDate.append(globals()['Station_' + Station])
```

	Date	Station	SO2	CO	CO2	O3	O3_Max_8_HR	PM10	PM25	\
0	2010-01-01	臺北市陽明	2.6	0.27	397.66	36.8	41.0	25.0	17.0	
1	2010-01-02	臺北市陽明	1.2	0.32	399.48	41.0	42.0	14.0	11.0	
2	2010-01-03	臺北市陽明	3.4	0.45	404.45	43.8	54.0	40.0	28.0	
3	2010-01-04	臺北市陽明	2.5	0.40	402.30	42.8	50.0	38.0	27.0	
4	2010-01-05	臺北市陽明	2.6	0.40	398.30	36.8	38.0	9.0	6.0	
5	2010-01-06	臺北市陽明	0.7	0.23	383.19	39.2	40.0	5.0	2.0	
6	2010-01-07	臺北市陽明	1.2	0.25	383.43	42.2	43.0	5.0	4.0	
7	2010-01-08	臺北市陽明	1.4	0.31	385.07	41.6	45.0	5.0	4.0	
8	2010-01-09	臺北市陽明	1.4	0.20	380.27	39.6	41.0	7.0	5.0	
9	2010-01-10	臺北市陽明	1.7	0.28	385.08	36.9	45.0	11.0	8.0	
10	2010-01-11	臺北市陽明	1.3	0.38	387.92	34.2	41.0	6.0	4.0	
11	2010-01-12	臺北市陽明	4.0	0.48	394.47	29.9	43.0	17.0	12.0	
12	2010-01-13	臺北市陽明	4.5	0.32	388.95	48.8	55.0	48.0	27.0	
13	2010-01-14	臺北市陽明	5.4	0.27	387.28	44.0	47.0	42.0	21.0	
14	2010-01-15	臺北市陽明	3.5	0.22	383.36	44.5	47.0	27.0	16.0	
15	2010-01-16	臺北市陽明	2.9	0.25	385.12	49.3	55.0	17.0	10.0	
16	2010-01-17	臺北市陽明	1.6	0.26	383.68	48.8	50.0	13.0	8.0	
17	2010-01-18	臺北市陽明	1.9	0.25	384.35	46.2	48.0	17.0	12.0	

[234773 rows x 23 columns]

讀入氣象資料

```
#讀入氣象站資訊
WeatherInfo = pd.read_csv("./practice/Weather_20180711.csv", sep = ",")

print(WeatherInfo.columns)
print(WeatherInfo)
```

```
C:\Users\User\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2698: DtypeWarning: Columns (0,1,2,3,4,5,6,7,8,9,
0,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,30) have mixed types. Specify dtype option on import or set low_memo
y=False.
  interactivity=interactivity, compiler=compiler, result=result)

Index(['StnPres', 'SeaPres', 'StnPresMax', 'StnPresMaxTime', 'StnPresMin',
       'StnPresMinTime', 'Temperature', 'TMax', 'TMaxTime', 'TMin', 'TMinTime',
       'Tddewpoint', 'RH', 'RHMin', 'RHMinTime', 'WS', 'WD', 'WSGust',
       'WDGust', 'WGustTime', 'Precp', 'PrecpHour', 'PrecpMax10',
       'PrecpMax10Time', 'PrecpHrMax', 'PrecpHrMaxTime', 'SunShine',
       'SunShineRate', 'GloblRad', 'VisbMean', 'EvapA', 'Stname', 'Date_time'],
      dtype='object')
   StnPres SeaPres StnPresMax    StnPresMaxTime StnPresMin  \
0   1017.6  1018.9     1021.1  2010-01-01 00:04     1015.1
1   1016.0  1017.3     1018.7  2010-01-02 23:08     1013.4
2   1016.2  1017.5     1018.6  2010-01-03 08:47     1014.8
3   1014.8  1016.1     1016.6  2010-01-04 21:42     1012.4
4   1018.2  1019.5     1021.0  2010-01-05 09:51     1016.2
5   1019.6  1020.9     1021.7  2010-01-06 20:18     1016.8
```

檢查欄位狀況

```
for col in WeatherInfo.columns:
    if col not in ['StnPresMaxTime', 'TMaxTime', 'StnPresMinTime', 'TMinTime', 'RHMinTime', 'WGustTime', 'PrecpMax10Time',
                   'PrecpHrMaxTime', 'Stname', 'Date_time']:
        WeatherInfo[col] = WeatherInfo[col].astype(float)
```

```
ValueError: could not convert string to float: 'X'
```

```
for col in WeatherInfo.columns:
    if col not in ['StnPresMaxTime', 'TMaxTime', 'StnPresMinTime', 'TMinTime', 'RHMinTime', 'WGustTime', 'PrecpMax10Time',
                   'PrecpHrMaxTime', 'Stname', 'Date_time']:
        WeatherInfo[col] = WeatherInfo[col].replace('X', 'NaN')
        WeatherInfo[col] = WeatherInfo[col].astype(float)
```

```
ValueError: could not convert string to float: 'T'
```

```
#讀入氣象站資訊
WeatherInfo = pd.read_csv("Weather_20180711.csv", sep = ",", low_memory=False)

for col in WeatherInfo.columns:
    if col not in ['StnPresMaxTime', 'TMaxTime', 'StnPresMinTime', 'TMinTime', 'RHMinTime', 'WGustTime', 'PrecpMax10Time',
                   'PrecpHrMaxTime', 'Stname', 'Date_time']:
        WeatherInfo[col] = WeatherInfo[col].replace('X', 'NaN')
        WeatherInfo[col] = WeatherInfo[col].replace('T', '0')
        WeatherInfo[col] = WeatherInfo[col].astype(float)

print(WeatherInfo)
```

讀入測站距離資料

```
#讀入測站間的距離檔
StationInfo = pd.read_csv("./practice/dis_address.csv",sep = ",")
StationInfo['Station'] = StationInfo['city'] + StationInfo['st_name']
del StationInfo['city']
del StationInfo['st_name']

print(StationInfo)
```

	lng	lat	IsAir	address	Station
0	121.442000	24.997600	0	板橋區大觀路二段265巷62號	新北市板橋
1	121.448900	25.164900	0	淡水區中正東路42巷6號	新北市淡水
2	121.529700	25.182600	0	北投區陽明山竹子湖路111號	臺北市鞍部
3	121.514900	25.037700	0	中正區公園路64號	臺北市臺北
4	121.544500	25.162100	0	北投區陽明山竹子湖路2號	臺北市竹子湖
5	121.740500	25.133300	0	仁愛區港西街6號6樓(海港大樓6樓)	基隆市基隆
6	122.079700	25.628000	0	中正區彭佳嶼	基隆市彭佳嶼
7	121.613300	23.975100	0	花蓮市花崗街24號	花蓮縣花蓮
8	121.047500	25.006700	0	新屋區東興路二段946號	桃園市新屋
9	121.857400	24.596700	0	蘇澳鎮港區路1號6樓(蘇澳港行政大樓6樓)	宜蘭縣蘇澳
10	121.756500	24.764000	0	宜蘭市力行路150號	宜蘭縣宜蘭
11	118.289300	24.407300	0	金城鎮金水里西海路一段250號	金門縣金門
12	119.667500	23.257000	0	望安鄉東吉村156號	澎湖縣東吉島
13	119.563100	23.565500	0	馬公市新興路2號	澎湖縣澎湖
14	120.204800	22.993200	0	中西區公園路21號	臺南市臺南
15	120.236700	23.038400	0	永康區鹽行里正南五街520巷88號	臺南市永康
16	120.315700	22.566000	0	前鎮區明孝里26鄰漁港南二路2號	高雄市高雄
17	120.432900	23.495900	0	西區北新里海口寮路56號	嘉義市嘉義

製作距離表

```
#製作測站間距離表
StationInfo_Air = StationInfo[StationInfo.IsAir==1].reset_index(drop = True)
StationInfo_Weather = StationInfo[StationInfo.IsAir==0].reset_index(drop = True)

St_Dis = []
for len_air in range(len(StationInfo_Air)):
    for len_rain in range(len(StationInfo_Weather)):
        St_Dis.append(np.linalg.norm(StationInfo_Air.iloc[len_air,0:2]-StationInfo_Weather.iloc[len_rain,0:2]))

Dis = pd.DataFrame(data={'St_Air':StationInfo_Air['Station'].repeat(len(StationInfo_Weather)),
                         'St_Weather':np.tile(StationInfo_Weather['Station'],len(StationInfo_Air)),
                         'Dis':St_Dis}).reset_index(drop = True)
Dis_Reduced = Dis[Dis.Dis<0.07].reset_index(drop = True)
```

	St_Air	St_Weather	Dis
0	臺北市陽明	臺北市鞍部	0.000169
1	臺北市陽明	臺北市竹子湖	0.025452
2	臺北市陽明	臺北市石牌	0.068271
3	臺北市陽明	臺北市天母	0.065665
4	臺北市陽明	臺北市大屯山	0.010045
5	臺北市萬華	臺北市臺北	0.011202
6	臺北市萬華	臺北市大直	0.047032
7	臺北市萬華	臺北市士林	0.044078
8	臺北市萬華	新北市三重	0.021980
9	臺北市萬華	臺北市信義	0.057293
10	臺北市萬華	新北市新莊	0.061376
11	臺北市萬華	新北市蘆洲	0.053668
12	臺北市萬華	新北市中和	0.056695
13	臺北市萬華	新北市永和	0.035203
14	臺北市萬華	臺北市松山	0.042485
15	臺北市萬華	臺北市公館	0.044996
16	臺北市大同	臺北市臺北	0.025549
17	臺北市大同	臺北市社子	0.063605

將氣象測站與空氣測站資訊結合

```
#將氣象資訊Map回空氣測站資訊
WeatherByAirSt = pd.DataFrame(columns=['St_Air', 'StnPres', 'SeaPres', 'StnPresMax', 'StnPresMin', 'Temperature', 'TMax', 'TMin',
       'Tddewpoint', 'RH', 'RHMin', 'WS', 'WD', 'WSGust', 'WDGust', 'Precp', 'PrecpHour', 'PrecpMax10',
       'PrecpHrMax', 'SunShine', 'SunShineRate', 'GloblRad', 'VisbMean', 'EvapA'])
for date in list(set(AirInfo_AddLossDate["Date"].astype(str))):
    Map_Day = Dis_Reduced[['St_Air','St_Weather']].copy()
    Map_Day['St_Air'] = Map_Day['St_Air'] + date
    Weather_Day = WeatherInfo[WeatherInfo.Date_time==date].reset_index(drop = True)
    Weather_Day.index = Weather_Day['Stname']
    Weather_Day = Weather_Day[['StnPres', 'SeaPres', 'StnPresMax', 'StnPresMin', 'Temperature', 'TMax', 'TMin',
           'Tddewpoint', 'RH', 'RHMin', 'WS', 'WD', 'WSGust', 'WDGust', 'Precp', 'PrecpHour', 'PrecpMax10',
           'PrecpHrMax', 'SunShine', 'SunShineRate', 'GloblRad', 'VisbMean', 'EvapA']]
    Map_Day = Map_Day.join(Weather_Day,on = 'St_Weather')
    del Map_Day['St_Weather']
    Map_Day = Map_Day.groupby(['St_Air'], as_index=False).mean()
    WeatherByAirSt = WeatherByAirSt.append(Map_Day)

WeatherByAirSt.rename(columns={'St_Air': 'Mapping'}, inplace=True)
```

	Mapping	StnPres	SeaPres	StnPresMax	StnPresMin	\
0	南投縣南投2010-05-07	NaN	NaN	NaN	NaN	
1	南投縣埔里2010-05-07	NaN	NaN	NaN	NaN	
2	南投縣竹山2010-05-07	NaN	NaN	NaN	NaN	
3	嘉義市嘉義2010-05-07	1005.200000	1008.50	1006.600000	1003.300000	
4	嘉義縣新港2010-05-07	NaN	NaN	NaN	NaN	
5	嘉義縣朴子2010-05-07	NaN	NaN	NaN	NaN	
6	基隆市基隆2010-05-07	1001.900000	1008.10	1004.100000	1000.550000	
7	宜蘭縣冬山2010-05-07	NaN	NaN	NaN	NaN	
8	宜蘭縣宜蘭2010-05-07	1006.200000	1007.10	1008.800000	1004.500000	
9	屏東縣屏東2010-05-07	NaN	NaN	NaN	NaN	
10	屏東縣恆春2010-05-07	1004.300000	1007.80	1006.075000	1002.500000	
11	屏東縣潮州2010-05-07	NaN	NaN	NaN	NaN	
12	彰化縣二林2010-05-07	NaN	NaN	NaN	NaN	
13	彰化縣彰化2010-05-07	NaN	NaN	NaN	NaN	

將氣象測站與空氣測站資訊結合

```
#Mapping完成
AirInfo_AddLossDate['Mapping'] = AirInfo_AddLossDate['Station'] + AirInfo_AddLossDate['Date'].astype(str)
AirAddWeather = pd.merge(AirInfo_AddLossDate,WeatherByAirSt,on="Mapping")
```

	Date	Station	SO2	CO	CO2	O3	O3_Max_8_HR	PM10	PM25	\
0	2010-01-01	金門縣金門	12.4	0.54	NaN	27.4	34.0	151.0	61.0	
1	2010-01-02	金門縣金門	22.1	0.80	NaN	7.4	13.0	205.0	89.0	
2	2010-01-03	金門縣金門	7.1	0.75	NaN	33.8	50.0	102.0	69.0	
3	2010-01-04	金門縣金門	10.6	0.96	NaN	28.3	52.0	181.0	104.0	
4	2010-01-05	金門縣金門	7.8	0.69	NaN	23.6	29.0	162.0	92.0	
5	2010-01-06	金門縣金門	7.5	0.61	NaN	17.4	20.0	59.0	44.0	
6	2010-01-07	金門縣金門	6.0	0.51	NaN	22.7	27.0	57.0	37.0	
7	2010-01-08	金門縣金門	8.8	0.63	NaN	22.5	29.0	94.0	56.0	
8	2010-01-09	金門縣金門	14.0	0.87	NaN	16.1	31.0	136.0	71.0	
9	2010-01-10	金門縣金門	10.6	0.78	NaN	18.3	33.0	136.0	66.0	
10	2010-01-11	金門縣金門	4.5	0.50	NaN	24.5	30.0	46.0	25.0	
11	2010-01-12	金門縣金門	7.4	0.94	NaN	16.6	28.0	55.0	31.0	
12	2010-01-13	金門縣金門	9.5	0.62	NaN	37.2	52.0	94.0	51.0	
13	2010-01-14	金門縣金門	11.6	0.62	NaN	43.8	55.0	139.0	62.0	
14	2010-01-15	金門縣金門	11.5	0.74	NaN	26.3	49.0	148.0	73.0	
15	2010-01-16	金門縣金門	12.6	0.66	NaN	28.1	46.0	176.0	72.0	
16	2010-01-17	金門縣金門	8.2	0.49	NaN	38.6	49.0	120.0	54.0	
17	2010-01-18	金門縣金門	7.8	0.48	NaN	35.4	48.0	91.0	41.0	

補充 – Regular Expression

- 用途：執行模式的比對與搜尋

```
1  # 簡單的匹配
2  pattern1 = "cat"
3  pattern2 = "bird"
4  string = "dog runs to cat"
5
6  print (pattern1 in string)
```

```
True
```

補充 – Regular Expression

```
1  import re                                    # 套用函式庫
2
3  pattern1 = "cat"
4  pattern2 = "bird"
5  string = "dog runs to cat"
6
7  print (re.search(pattern1, string))          # 輸出匹配結果
```

```
<_sre.SRE_Match object; span=(12, 15), match='cat'>
```

補充 – Regular Expression

```
1  import re                                                    # 套用函式庫
2
3  print (re.search(r"r[A-Z]n", "dog runs to cat"))              #匹配任何大寫的英文字母
4  print (re.search(r"r[a-z]n", "dog runs to cat"))              #匹配任何小寫的英文字母
5  print (re.search(r"r[0-9]n", "dog r2ns to cat"))              #匹配任何數字
6  print (re.search(r"r[0-9a-z]n", "dog runs to cat"))           #匹配大小寫的英文字母
```

```
None
<_sre.SRE_Match object; span=(4, 7), match='run'>
<_sre.SRE_Match object; span=(4, 7), match='r2n'>
<_sre.SRE_Match object; span=(4, 7), match='run'>
```

補充 – Regular Expression

字元	使用說明
\d	0–9之間的整數字元
\D	除了0–9之間的整數字元以外的其它字元
\s	任何 white space, 如 [\t\n\r\f\v]
\S	不是 white space
\w	任何大小寫字母, 数字和 “” [a–zA–Z0–9]
\W	不是 \w
\b	空白字符 (只在某个字的開頭或结尾)
\B	空白字符 (不在某个字的開頭或结尾)
\	匹配 \
.	匹配任何字符 (除了 \n)
^	匹配開頭
$	匹配结尾
?	前面的字符可有可無

補充 – Regular Expression

```
1  import re                                                  # 套用函式庫
2
3  print (re.search(r"r\dn", "run r4n"))                      #匹配數字
4  print (re.search(r"r\sn", "r\nn r4n"))                     #匹配空白
5  print (re.search(r"r\wn", "r\nn r4n"))                     所有字母數字和"_"，等價於[a-zA-Z0-9]
6  print (re.search(r"\bruns\b", "dog runs to cat"))          # 匹配頭尾空白的單字
```

```
<_sre.SRE_Match object; span=(4, 7), match='r4n'>
<_sre.SRE_Match object; span=(0, 3), match='r\nn'>
<_sre.SRE_Match object; span=(4, 7), match='r4n'>
<_sre.SRE_Match object; span=(4, 8), match='runs'>
```

問題與討論