# 程式設計 (二)

## Python CH2. 基本資料結構與資料分析實務

Ming-Hung Wang 王銘宏

tonymhwang@cs.ccu.edu.tw

Department of Computer Science and Information Engineering
National Chung Cheng University

Spring Semester, 2022

# 本章目錄

# Python 基本資料結構

- str (String, 字串)
- list (List, 列表)
- dict (Dictionary, 字典)
- tuple (Tuple, 元祖)
- set (Set, 集合)

本章會先討論 list、tuple

# 基本資料結構：list

# 基本資料結構：list

Python 的 list 可能會讓你聯想到其他語言的陣列，但事實上，它比陣列還要強大許多。

## 建立 list

使用中括號建立 list，list 中的元素以逗號分隔。

```
a = list() #建立空的list
b = []     #建立空的list
c = [1, 2, 3.5, 'abc', [-5, 'apple']]
```

## len 函式

可取得元素數量

```
1 a = [1, 2, 3]
2 b = ['a', 'bc', ['def', 'g']]
3 print(a, 'len:', len(a))
4 print(b, 'len:', len(b))
```

```
[1, 2, 3] len: 3
['a', 'bc', ['def', 'g']] len: 3
```

# 索引 (index)

可以從頭數也可以倒著數，從頭數由開頭從 0 開始遞增，倒著數由尾端從-1 開始遞減。

```
1 a = ['aa', 'bb', 'cc', 'dd']
2 print(a[0], a[1], a[2], a[3])
3 print(a[-1], a[-2], a[-3], a[-4])
```

```
aa bb cc dd
dd cc bb aa
```

```
1 b = ['a', 'bc', ['def', 'g']]
2 print(b[-1])
3 print(b[-1][0])
```

```
['def', 'g']
def
```

# 切片 (slicing) (1/3)

用 [index1:index2] 可提取 index1 到 index2 之間 (不包括 index2) 的元素。

```
1 a = ['aa', 'bb', 'cc', 'dd', 'ee']
2 b = a[1:3] #取index 1, 2
3 c = a[1:-1] #取index 1, 2, 3 (index -1 = index 4)
4 print(b)
5 print(c)
```

```
['bb', 'cc']
['bb', 'cc', 'dd']
```

# 切片 (slicing) (2/3)

可省略 [index1:index2] 中的 index1 代表從頭開始，或省略 index2 代表取到尾端。

```
1 a = ['aa', 'bb', 'cc', 'dd', 'ee']
2 b = a[:3] #取開頭3份
3 c = a[2:] #最頭2份不要
4 print(b)
5 print(c)
```

```
['aa', 'bb', 'cc']
['cc', 'dd', 'ee']
```

# 切片 (slicing) (3/3)

可同時省略 [index1:index2] 中的 index1 與 index2，代表複製一份 list。

```
1 a = ['aa', 'bb', 'cc', 'dd', 'ee']
2 b = a    # 如此a與b會指向同一個list
3 c = a[:] # 如此會複製一份新的list
4 b[0] = 'AAAAA' # 修改b，a也會一起更改，但c不受影響
5 print(a)
6 print(b)
7 print(c)
```

```
['AAAAA', 'bb', 'cc', 'dd', 'ee']
['AAAAA', 'bb', 'cc', 'dd', 'ee']
['aa', 'bb', 'cc', 'dd', 'ee']
```

# 指定間格切片

用 [i1:i2:n](不包括 index2)，
每間隔 n 提取元素。

```python
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(a[1:8:2])  # index 1 3 4 7
print(a[3::3])   # index 3 6
print(a[8:1:-2]) # index 8 6 4 2
print(a[::-1])   # reverse
```

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

# 更改 list 元素 (1/4)

更改指定 index 的元素值：

```
1 a = ['aa', 'bb', 'cc', 'dd', 'ee']
2 a[2] = '22222'
3 print(a)
```

```
['aa', 'bb', '22222', 'dd', 'ee']
```

替換指定的 slicing，
替換的 list 長度**不須**與指定的 slicing 長度相等：

```
1 a = ['aa', 'bb', 'cc', 'dd', 'ee']
2 a[2:4] = ['XXX', 'YYY', 'ZZZ'] #替換掉index 2, 3
3 print(a)
```

```
['aa', 'bb', 'XXX', 'YYY', 'ZZZ', 'ee']
```

# 更改 list 元素 (2/4)

利用替換指定的 slicing，可達成插入與刪除元素：

```
1 a = [1, 2, 3]
2 a[len(a):] = [4, 5, 6, 7] # 在後方插入多個值
3 print(a)
4 a[:0] = [-3, -2, -1, 0] # 在前方插入多個值
5 print(a)
6 a[2:3] = [] # 移除index 2
7 print(a)
8 a[1:-1] = [] # 移除頭尾以外的數值
9 print(a)
```

```
[1, 2, 3, 4, 5, 6, 7]
[-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7]
[-3, -2, 0, 1, 2, 3, 4, 5, 6, 7]
[-3, 7]
```

# 更改 list 元素 (3/4)

利用替換指定的指定間格切片，
間隔的方式更改元素值：

```
1 a = [0, 1, 2, 3, 4]
2 a[1::2] = [-1, -3]
3 print(a)

[0, -1, 2, -3, 4]
```

若使用指定間格切片，
替換的 list 長度**必須**與指定的 slicing 長度相等：

```
1 a = [0, 1, 2, 3, 4]
2 a[1::2] = [-1, -2, -3] # ValueError
3 print(a)
```

# 更改 list 元素 (4/4)

利用切片，其他容器的元素也可以存入 list：

```
1 a = [0, 1, 2, 3, 4]
2 a[2:3] = 'abc'
3 print(a)
```

```
[0, 1, 'a', 'b', 'c', 3, 4]
```

```
1 a = []
2 a[:] = 'abc'
3 print(a)
```

```
['a', 'b', 'c']
```

```
1 a = [0, 1, 2, 3, 4, 5, 6]
2 a[1::2] = 'abc'
3 print(a)
```

```
[0, 'a', 2, 'b', 4, 'c', 6]
```

## list 運算：+、*

可以使用 + 和 * 做 list 的連接或重複

```
1 a = [1, 2, 3]
2 b = [4, 5, 6]
3 c = a + b
4 print(c)
5 d = [0] * 10
6 print(d)
```

```
[1, 2, 3, 4, 5, 6]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

# in 運算子

in 運算子能尋找 list 中是否包含某元素

```
1 a = [1, 2, 3, 4, 5, 6]
2 n = -1
3 print(n in a)
4 if not n in a:
5     print(n, 'not in', a)
```

```
False
-1 not in [1, 2, 3, 4, 5, 6]
```

# list 常用的方法與函式

# 方法 (method) vs 函式 (function)

- 函式：由上帝主導
- 方法：由物件 (object) 主導
- 舉個例子
  - 函式：move(car, 5) 移動車子 5 公尺
  - 方法：car.go(5) 車子移動 5 公尺

可以理解成 method 是 object 內建的功能。關於 object 與 method 的觀念會在物件導向 (OOP) 課程中詳細說明。

# list 常用的方法與函式

## append 方法與 extend 方法

append 方法：插入一個元素到尾端

extend 方法：連接一個 list 到尾端

```
1 a = [1, 2, 3]
2 a.append(4)
3 print(a)
4 a.append([5, 6, 7])
5 print(a)
6 a.extend([8, 9, 10])
7 print(a)
```

```
[1, 2, 3, 4]
[1, 2, 3, 4, [5, 6, 7]]
[1, 2, 3, 4, [5, 6, 7], 8, 9, 10]
```

# list 常用的方法與函式

## insert 方法

插入元素到指定 index

```
1 a = [1, 2, 3]
2 a.insert(0, 'hi') #插入到開頭
3 print(a)
4 a.insert(2, 'hi') #插入到index 2
5 print(a)
6 a.insert(len(a), '喵哈囉') #插入到尾端
7 print(a)
```

```
['hi', 1, 2, 3]
['hi', 1, 'hi', 2, 3]
['hi', 1, 'hi', 2, 3, '喵哈囉']
```

# list 常用的方法與函式

## pop 方法

刪除最後一個元素，或指定 index 刪除，
會回傳被刪除的數值。

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2 n = a.pop() #刪除最後一個元素
3 print('pop:', n, ' list:', a)
4 n = a.pop(2) #刪除index 2的元素
5 print('pop:', n, ' list:', a)
```

```
pop: 7  list: [1, 2, 3, 4, 5, 6]
pop: 3  list: [1, 2, 4, 5, 6]
```

# del 敘述

## 刪除 index 或 slicing

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2 del a[2] #刪除index 2的元素
3 print(a)
4 del a[4:] #刪除index 4與之後的元素
5 print(a)

[1, 2, 4, 5, 6, 7]
[1, 2, 4, 5]
```

雖然 del a[n] 與 a[n:n+1] = [] 的效果相同，
而 del a[m:n] 與 a[m:n] = [] 的效果相同，
但 del 敘述的可讀性相對較佳。

## index 方法與 remove 方法

- index 方法可尋找某元素在 list 中第 1 次出現的 index，若沒找到會 error，建議先用 in 運算子尋找一次
- remove 方法可刪除在 list 中第 1 次出現與指定內容相符的元素，若沒找到會 error，建議先用 in 運算子尋找一次

# list 常用的方法與函式

## index 方法與 remove 方法範例：

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2 if 3 in a:
3     print(3 , 'is at index', a.index(3))
4     a.remove(3)
5     print('I remove it, haha')
6 else:
7     print(3 , 'is not dound')
8 print(a)
```

```
3 is at index 2
I remove it, haha
[1, 2, 4, 5, 6, 7]
```

# list 常用的方法與函式

## count 方法

搜尋指定內容出現的次數

```
1 a = [1, 2, 5, 9, 1, 2, 3, 4, 5, 2]
2 n = 2
3 print('I count', a.count(n), 'time(s)', n, 'in', a)
4 n = 7
5 print('I count', a.count(n), 'time(s)', n, 'in', a)

I count 3 time(s) 2 in [1, 2, 5, 9, 1, 2, 3, 4, 5, 2]
I count 0 time(s) 7 in [1, 2, 5, 9, 1, 2, 3, 4, 5, 2]
```

## reverse 方法

反轉 list 的元素

```
1 a = [1, 2, 3, 4, 5, 6, 7]
2 a.reverse()
3 print(a)
```

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

# sort 方法

排序 list 元素，list 中的元素必須可以互相比較

```python
a = [2, 7, 3, 1, 8, 6]
a.sort()
print(a)
a = ['cat', 'apple', 'dog', 'banana']
a.sort()
print(a)
a = [[3, 1], [2, 0], [1, 3], [1, 4], [2, 3]]
a.sort()
print(a)
```

```
[1, 2, 3, 6, 7, 8]
['apple', 'banana', 'cat', 'dog']
[[1, 3], [1, 4], [2, 0], [2, 3], [3, 1]]
```

```python
a = [1, 5, 'egg']
a.sort() #TypeError
print(a)
```

# sort 方法 vs sorted 函式

- sort 方法：會修改原本的 list
- sorted 函式：不會修改原本的 list，回傳一個新的

```
1 a = [2, 7, 3, 1, 8, 6]
2 b = sorted(a)
3 print(a)
4 print(b)
```

```
[2, 7, 3, 1, 8, 6]
[1, 2, 3, 6, 7, 8]
```

# list 常用的方法與函式

## min 函式與 max 函式

回傳 list 的最小或最大數值，
list 中的元素必須可以互相比較

```
1 a = [1, 2, 5, 9, 1, 2, 3, 4, 5, 2]
2 print('min:', min(a))
3 print('max:', max(a))
```

```
min: 1
max: 9
```

# 基本資料結構：tuple

# 基本資料結構：tuple

tuple 與 list 非常相似，但 tuple 只能
被建立而不能被修改。
可以將 tuple 視為無法更動的 list。

# 建立 tuple

使用小括號建立 tuple，tuple 中的元素以逗號分隔。

```
a = tuple() #建立空的tuple
b = ()      #建立空的tuple
c = (1, 2, 3.5, 'abc')
d = (1, 2, 3, [4, 5])
e = (0,)    #若tuple只有一個元素，請在元素後多加一個逗號
```

# 基本資料結構：tuple

tuple 與 list 被稱為序列 (sequence)，都使用中括號取值，也可以使用切片 (slicing)，也可以使用 +、*、in 運算子。

```
1 a = (2, 4, 6, 8)
2 b = (1, 2, 3, [4, 5])
3 c = (0,) * 10
4 d = a + b
5 print(c)
6 print(d)
7 print(a[2])
```

```
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
(2, 4, 6, 8, 1, 2, 3, [4, 5])
6
```

# 基本資料結構：tuple

以前，如果要判斷變數數值是否為多種可能的
其中一個，可能要這樣寫：

```
1 a = input()
2 if (a == 'orange' or a == 'banana' or
3     a == 'strawberry' or a == 'watermelon'):
4     print('I love it.')
```

有了 in 運算子就可以寫成這樣：

```
1 favorite = ('orange', 'banana',
2              'strawberry', 'watermelon')
3 a = input()
4 if a in favorite:
5     print('I love it.')
```

# 基本資料結構：tuple

tuple 無法更動元素。

```
1 a = (1, 2, 3, [4, 5])
2 a[1] = 10 #TypeError
3 print(a)
```

但 tuple 中的 list 依舊可以更動元素。

```
1 a = (1, 2, 3, [4, 5])
2 a[-1].append(10)
3 print(a)
```

```
(1, 2, 3, [4, 5, 10])
```

## tuple 與 list 轉型

使用 list() 函式將容器轉型成 list

使用 tuple() 函式將容器轉型成 tuple

```
1 a = (2, 4, 6, 8)
2 b = [1, 3, 5, 7]
3 c = list(a)
4 d = tuple(b)
5 print(c, type(c))
6 print(d, type(d))
```

```
[2, 4, 6, 8] <class 'list'>
(1, 3, 5, 7) <class 'tuple'>
```

# 元素是變數的 tuple 與 tuple 的自動解包

Python 允許元素是變數的 tuple 出現在等號左側。
利用 tuple 的自動解包 (unpacking) 特性，等號
右側的數值會一一對應到左側的變數。

```
1 (a, b, c, d) = (2, 4, 6, 8)
```

上面 1 行就可取代以下 4 行程式碼：

```
1 a = 2
2 b = 4
3 c = 6
4 d = 8
```

## tuple 的自動打包

上一頁的程式碼還能簡化成更簡單的方式：

```
1 a, b, c, d = 2, 4, 6, 8
```

因為 tuple 的自動打包 (packing) 特性，以逗號相隔的數值會被打包成 tuple，再利用上一頁的解包特性，完成數值的指派。

# 序列型別的自動解包

不只是 tuple，list 與 str 也都具有自動解包的特性。

```
1 a, b, c, d = [1, 2, 3, 4]
2 print(a, b, c, d)
```

```
1 2 3 4
```

```
1 ((a, b), c, d) = [(2, 3), 4, 5]
2 print(a, b, c, d)
```

```
2 3 4 5
```

```
1 a, b, c, d, e = 'Magic'
2 print(a, b, c, d, e, sep=', ')
```

```
M, a, g, i, c
```

## 交換數值

以前，如果要交換數值可能要這樣寫：

```
tmp = a
a = b
b = tmp
```

如今，我們已學會自動打包和自動解包，
可以簡化成 1 行：

```
a, b = b, a
```

# for 迴圈與生成式 (Comprehension)

# for 迴圈與生成式 (Comprehension)

## for in range

在上一章中提到，可以用 range 函式
生成物件給 for 迴圈做走訪。

```
1 for i in range(1, 10, 2):
2     print(i, end=' ')
3 print() #換行
```

```
1 3 5 7 9
```

## range to list

range 回傳的是 range 專屬的物件，
但我們可以將其轉型成 list。

```
1 r = range(2, 10, 2)
2 print(r, type(r))
3 l = list(r)
4 print(l, type(l))
```

```
range(2, 10, 2) <class 'range'>
[2, 4, 6, 8] <class 'list'>
```

# for 迴圈走訪容器

```
for 區域變數 in 容器:
```

for 的每一次迴圈，區域變數會照順序
被指派成容器元素的值。

```
1 fruit = ['apple', 'banana', 'orange']
2 for f in fruit:
3     print(f)
```

```
apple
banana
orange
```

# for 迴圈修改容器的值

區域變數並不是參照容器的元素。

```
1 fruit = ['apple', 'banana', 'orange']
2 for f in fruit:
3     f = 'papaya'
4 print(fruit)
```

```
['apple', 'banana', 'orange']
```

```
1 fruit = ['apple', 'banana', 'orange']
2 for i in range(len(fruit)):
3     fruit[i] = 'papaya'
4 print(fruit)
```

```
['papaya', 'papaya', 'papaya']
```

# for 迴圈使用 tuple 解包多重設定變數

若出現二維的容器，我們可以使用 tuple 解包的特性，讓 for 迴圈更簡潔易懂：

```
1 position = [(1, 5), (2, 4), (-3, 6), (1, -4)]
2 for x, y in position:
3     print(x, y)
```

```
1 5
2 4
-3 6
1 -4
```

此程式碼的 x,y 會自動封裝成 1 個 tuple

## enumerate 函式

enumerate 函式會將 list 或 tuple 的元素取出並加上 index 編號：

```
1 nums = [1, 5, 2, 4, -3, 6]
2 print(list(enumerate(nums)))
3 for i, n in enumerate(nums):
4     print('#%d:' % (i + 1), n)
```

```
[(0, 1), (1, 5), (2, 2), (3, 4), (4, -3), (5, 6)]
#1: 1
#2: 5
#3: 2
#4: 4
#5: -3
#6: 6
```

# zip 函式

zip 函式可將多個可走訪的物件結合起來：

```
1 fruit = ['apples', 'bananas', 'oranges', 'papayas']
2 nums = [5, 3, 7, 6]
3 print(list(zip(fruit, nums)))
4 for f, n in zip(fruit, nums):
5     print(n, f)
```

```
[('apples', 5),('bananas', 3),('oranges', 7),('papayas', 6)]
5 apples
3 bananas
7 oranges
6 papayas
```

# 生成式 (Comprehension)

我們常見用 for 迴圈產生新的 list 的方法通常如下：

```
square = []
for i in range(10):
    if i % 2 == 0:
        square.append(i ** 2)
print(square)
```

```
[0, 4, 16, 36, 64]
```

但 Python 有提供生成式語法可以更快解決這個問題。

# 生成式語法如下

```
new_lst = [運算式 for 區域變數 in 容器 if 條件]
```

生成式語法中，if 是非必需的，如：

```
new_lst = [運算式 for 區域變數 in 容器]
```

下方 2 段程式碼的效果是一樣的。

```
square = []
for i in range(10):
    if i % 2 == 0:
        square.append(i ** 2)
print(square)
```

```
square = [i ** 2 for i in range(10) if i % 2 == 0]
print(square)
```

# Python 套件與函式庫

## import

使用 import 可以將函式庫、套件或模組匯入，
類似 C 語言的 #include。

- 函式庫 (library)：Python 內建的模組，不須額外安裝，但使用時須先 import，詳見Python 標準函式庫
- 套件 (package)：Python 第三方的模組，須先安裝，使用時須先 import
- 模組 (module)：自行編寫的其他 Python 檔，也可以是 C/C++ 的 Object 檔

# import 語法

匯入模組並使用模組內所提供的函式、屬性 (變數)：

```
import 模組名稱

模組名稱.函式A()
模組名稱.屬性B
```

直接從模組中匯入函式、屬性：

```
from 模組名稱 import 函式A, 屬性B

函式A()
屬性B
```

如此便不需要在函式和屬性前面加上模組名稱

# 將 import 的模組取別稱

有時候，模組的名稱太長，或是函式和屬性的名稱有衝突，就可以使用 as 來取別稱

```
import 模組名稱 as 別稱

別稱.函式A()
別稱.屬性B
```

直接從模組中匯入函式、屬性：

```
from 模組名稱 import 函式A as 別稱A, 屬性B as 別稱B

別稱A()
別稱B
```

# 匯入子模組

如果想使用模組中的子模組中的函式、屬性，可以這樣寫：

```
import 模組名稱

模組名稱.子模組名稱.函式A()
模組名稱.子模組名稱.屬性B
```

但更好的方式是這樣寫：

```
import 模組名稱.子模組名稱

子模組名稱.函式A()
子模組名稱.屬性B
```

也可以幫子模組取別稱：

```
import 模組名稱.子模組名稱 as 別稱

別稱.函式A()
別稱.屬性B
```

直接從子模組中匯入函式、屬性：

```
from 模組名稱.子模組名稱 import 函式A, 屬性B

函式A()
屬性B
```

將函式、屬性取別稱：

```
from 模組名稱.子模組名稱 import 函式A as 別稱A, 屬性B as 別稱B

別稱A()
別稱B
```

## pip

pip 是 Python 的套件管理系統，
它可以安裝和管理 Python 套件。

pip 的套件會從 Python 套件索引
(PyPI, Python Package Index))
中取得。

# 使用 pip 安裝套件

假如今天要安裝 Scikit-learn 這個套件，只要在 Anaconda 的終端機上，下以下指令 ("$" 不用打)，如果在 Python 互動模式，請在開頭加上"!"。

```
$ pip install scikit-learn
```

若已經安裝，要升級版本時只須加上 −−upgrade

```
$ pip install --upgrade scikit-learn
```

若要安裝特定版本，加上 == 版本數)

```
$ pip install scikit-learn==0.22.2
```

# 其他 pip 指令

## 查看安裝過的套件

```
$ pip list
```

## 解除安裝套件

以下是解除安裝 Scikit-learn 的指令

```
$ pip uninstall scikit-learn
```


# Matplotlib - 資料的視覺化

# Matplotlib - 資料的視覺化

Python 的資料科學與機器學習套件包括
Nnumpy、Scipy、Pandas、Scikit-learn、
Statsmodels、Matplotlib、Scrapy、
Keras、TensorFlow 等。

Matplotlib 是 Python 的資料科學套件之一，
用於資料視覺化上。Matplotlib 常搭配其他套件，
如 Numpy 與 Pandas 做資料分析與視覺化。

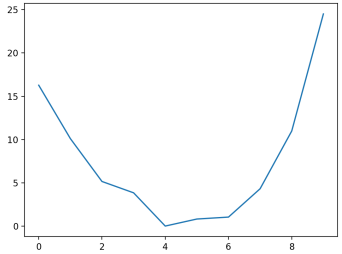
# Matplotlib - 資料的視覺化

安裝 (Anaconda 預設已安裝)：

```
$ pip install matplotlib
```

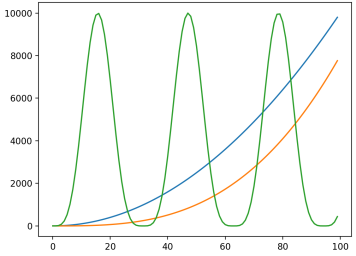
# 嘗試畫出折線圖

```python
from random import random #random()會回傳-1 ~ 1的數值
import matplotlib.pyplot as plt

x = list(range(10))
y = [(i - 5 + random()) ** 2 for i in range(10)]
plt.plot(x, y) #建立折線圖
plt.show() #顯示
```



# Matplotlib - 資料的視覺化

```python
import math
import matplotlib.pyplot as plt

x = list(range(100))
y1 = [i ** 2 for i in range(100)]
y2 = [(i / 5) ** 3 for i in range(100)]
y3 = [(math.sin(i / 10) * 10) ** 4 for i in range(100)]
y = list(zip(y1, y2, y3)) #使用zip可將多個list結合
plt.plot(x, y) #建立折線圖
plt.show() #顯示
```



# 讀取 csv 檔

網路上的公開下載數據許多都是 csv 檔，我們可以到 https://data.gov.tw/尋找有興趣的資料建立圖表。
下列程式會把 csv 檔輸入成 1 個二維的 list。
(with open() as 敘述會於第 3 章介紹)

```
1 import csv
2
3 with open('檔名.csv', encoding='utf-8') as csvfile:
4     datas = list(csv.reader(csvfile))
```

# 實作：使用政府開放資料繪製折線圖

開啟原住民族地方文物館參觀人次統計表 (108 年度)
下載 csv 檔

# 實作：使用政府開放資料繪製折線圖

將檔名重新命名為 data.csv，並將 data.csv 讀入

```
1 import csv
2
3 with open('data.csv', encoding='utf-8') as csvfile:
4     datas = list(csv.reader(csvfile))
5 for row in datas:
6     print(row)
```

```
['Seq', 'DateListed', ' 年度', ' 項次', ' 單位', ' 一月', ' 二月', ' 三月', ' 四月', ' 五月', ' 六月', ' 七月', ' 八月', ' 九月', ' 十月', ' 十一月', ' 十二月', ' 總計']
['1', '2020/09/16', '108', '1', ' 基隆市原住民文化會館', '4759', '3714', '5449', '5368', '6779', '5406', '6692', '8049', '5056', '5877', '6265', '5706', '69120']
['2', '2020/09/16', '108', '2', ' 臺北市政府原委員會凱達格蘭文化館', '23803', '8385', '5368', '0', '0', '0', '0', '29569', '34578', '46960', '53539', '36577', '238779']
['3', '2020/09/16', '108', '3', ' 新北市烏來泰雅民族博物館', '19213', '26585', '26994', '20437', '23390', '24746', '24993', '23493', '22755', '30045', '22273', '29811', '294735']
['4', '2020/09/16', '108', '4', ' 桃園市原住民文化會館', '4802', '4354', '9350', '12019', '11633', '9238', '11186', '16048', '14788', '10062', '18648', '13229', '135357']
...
```

# 實作：使用政府開放資料繪製折線圖

為了方便分析，我希望把從 csv 讀下來的資料建立 3 個 list，分別是 items(文物館名稱)、month(月份名稱) 與 people_per_month(每個月各文物館的人數)。

```
1 import csv
2
3 with open('data.csv', encoding='utf-8') as csvfile:
4     datas = list(csv.reader(csvfile))
5 for row in datas:
6     print(row)
```

```
[' 基隆市原住民文化會館', ' 臺北市政府原委員會凱達格蘭文化館', ' 新北市烏來泰雅民族博物館', ' 桃園市原住民文化會館', ' 新竹縣五峰鄉賽夏族矮人祭場文物館', ' 新竹縣尖石鄉原住民文化館', ' 苗栗縣賽夏族民俗文物館', ' 苗栗縣泰雅文物館', ' 臺中市原住民綜合服務中心', ' 彰化縣原住民生活館', ' 嘉義縣阿里山鄉族自然與文化中心', ' 臺南市原住民會館 (札哈木) ', ' 臺南市原住民文物館 (Palihanbasan 巴里和巴桑故事館) ', ' 高雄市那瑪夏區原住民文物館', ' 高雄市桃源區原住民文物館', ' 屏東縣原住民文化會館', ' 屏東縣三地門鄉原住民文化館', ' 屏東縣霧台鄉魯凱族文物館', ' 屏東縣來義鄉原住民文物館', ' 屏東縣獅子鄉文物陳列館', ' 臺東縣海端鄉布農族文化館', ' 臺東縣成功鎮原住民文物館', ' 花蓮縣瑞穗鄉奇美原住民文物館', ' 花蓮縣萬榮鄉原住民文物館', ' 花蓮縣壽豐鄉原住民文物館', ' 花蓮縣吉安鄉阿美族文物館', ' 台灣原住民族文化館 (花蓮縣) ', ' 宜蘭縣大同鄉泰雅生活館', ' 宜蘭縣南澳鄉泰雅文化館']
一月 [4759, 23803, 19213, 4802, 350, 5880, 1728, 3194, 1997, 1675, 2207, 2912, 914, 529, 577, 3212, 2972, 674, 1273, 1681, 427, 272, 480, 158, 344, 482, 114, 1446, 555]
二月 [3714, 8385, 26585, 4354, 325, 3710, 1892, 3346, 1257, 1651, 1588, 890, 629, 234, 522, 1712, 4701, 927, 6483, 530, 259, 250, 375, 92, 294, 371, 33, 1985, 567]
三月 [5449, 5368, 26994, 9350, 340, 4810, 2384, 3010, 1320, 1926, 1081, 2591, 870, 430, 1477, 1972, 2035, 783, 10656, 928, 372, 230, 450, 415, 575, 402, 480, 1889, 959]
四月 [5368, 0, 20437, 12019, 443, 3650, 2153, 3498, 1482, 1876, 1157, 2455, 828, 785, 395, 2537, 2176, 583, 829, 745, 2627, 202, 435, 423, 545, 882, 175, 1835, 622]
...
```

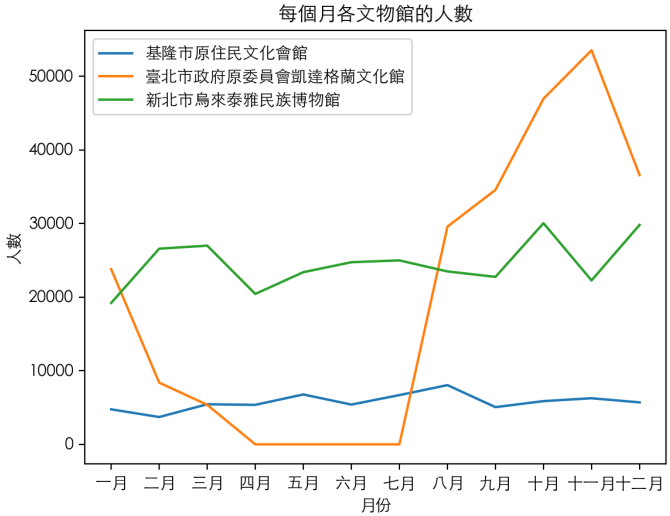
# 實作：使用政府開放資料繪製折線圖

把前 3 個文物館的每月人數畫成折線圖

```
14 import matplotlib.pyplot as plt
15
16 x = month[:12]
17 y = [r[:3] for r in people_per_month[:12]]
18 plots = plt.plot(x, y) # 建立折線圖
19 plt.legend(plots, items[:3]) # 設定標籤
20 plt.title('每個月各文物館的人數')
21 plt.xlabel('月份')
22 plt.ylabel('人數')
23 plt.rcParams['font.sans-serif'] = ['Microsoft JhengHei']
24 #plt.rcParams['font.sans-serif'] = ['Heiti TC'] #MacOS
25 plt.show()
```

備註：程式碼第 23 行是 Windows 的寫法；第 24 行是 MacOS 的寫法。

# 實作：使用政府開放資料繪製折線圖



# 實作：把成績分布繪製成直方圖

下載grade.csv，這是一個虛構的成績表。
我們想要將此成績表繪製成一個
每 10 分為 1 個單位的直方圖。

# 實作：把成績分布繪製成直方圖

將 grade.csv 放到程式檔的同個目錄中，使用程式讀入。

```
1 import csv
2 import matplotlib.pyplot as plt
3
4 with open('grade.csv', encoding='utf-8') as csvfile:
5     datas = list(csv.reader(csvfile))
6 for row in datas:
7     print(row)
```

```
['編號', '學號', '分數']
['1', 'D05102XX', '14']
['2', 'D05106XX', '88']
['3', 'D05114XX', '51']
['4', 'D05117XX', '65']
['5', 'D05122XX', '68']
['6', 'D05125XX', '27']
...
```

# 實作：把成績分布繪製成直方圖

將 datas 中的分數取出，並轉型成 int：

```
8 grades = [int(r[2]) for r in datas[1:]]
9 print(grades)
```

```
[14, 88, 51, 65, 68, 27, 33, 86, 51, 70, 68, 65, 65, 35, 50,
```

建立級距區間：

```
10 bins = list(range(0, 111, 10))
11 print(bins)
```

```
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
```

上面 list 代表區間 [0,10),[10,20),[20,30), ..., [80,90),[90,100),[100,110]

# 實作：把成績分布繪製成直方圖

給予分數、直方圖寬度 (width)、區間 (bins)，繪製直方圖。

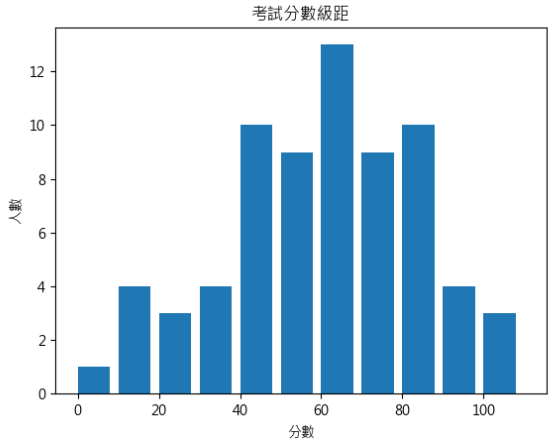
```
12 plt.hist(grades, width=8, bins=bins)
13 plt.title('考試分數級距')
14 plt.xlabel('分數')
15 plt.ylabel('人數')
16 plt.rcParams['font.sans-serif'] = ['Microsoft JhengHei']
17 #plt.rcParams['font.sans-serif'] = ['Heiti TC'] #MacOS
18 plt.show()
```

plt.hist 的 bins 參數若為 list 則代表區間界線，若為整數則代表將值域分為指定個等份數，若不指定 bins 則預設為 10。

備註：程式碼第 16 行是 Windows 的寫法；第 17 行是 MacOS 的寫法。

# 實作：把成績分布繪製成直方圖

完成：



# 實作：繪製臺南市人口數-月份散布圖，並繪製回歸曲線預測人口數

下載tainan_population.zip
(壓縮檔中的文件由https://data.gov.tw/網站免費提供)。
我們將讀取每個檔案中的人口總數欄位，並繪製成人口數-月份散布圖與回歸曲線。

# 實作：臺南市人口散布圖與回歸曲線

使用以下套件：

```
1 import csv
2 import matplotlib.pyplot as plt
3 import numpy as np
4
```

NumPy 是 Python 常用的數學套件，主要用來計算線性代數相關的數學問題，如維度、矩陣運算等。

今天我們要使用 NumPy 來計算回歸曲線函數。

NumPy 安裝 (Anaconda 預設已安裝)：

```
$ pip install numpy
```

# 實作：臺南市人口散布圖與回歸曲線

tainan_population.zip 解壓縮後，
將 tainan_population 資料夾放到程式檔的同個目錄。
將資料夾中的 csv 檔讀入，並紀錄人口總數：

```
 5 years = []
 6 population = []
 7 for y in range(107, 111): #107~110年
 8     for m in range(1, 13): #1~12月
 9         if y == 110 and m > 4: #110年只到4月
10             break
11         file = 'tainan_population/%d%d.csv' % (y, m)
12         with open(file, encoding='utf-8') as csvfile:
13             datas = list(csv.reader(csvfile))
14             #紀錄人口總數
15             population.append(int(datas[1][2]))
16             #為方便計算 將y年m月紀錄為y+(m-1)/12年
17             years.append(y + (m - 1) / 12)
```

# 實作：臺南市人口散布圖與回歸曲線

years 代表時間 (年)，1 個月計做 1/12 年，population 代表人口數

```
18 print(years)
19 print(population)
```

```
[107.0, 107.08333333333333, 107.16666666666667, 107.25, 107.33333333333333, 107.41666666666667, 107.5, 107.58333333333333, 107.66666666666667, 107.75, 107.83333333333333, 107.91666666666667, 108.0, ... , 110.25]
[1886251, 1886074, 1885882, 1885478, 1884935, 1884717, 1884596, 1884327, 1884036, 1883804, 1883760, 1883831, 1883723, ... , 1871224]
```

為了做回歸曲線預測，
我們將資料分割成前 80% 與後 20%

```
20 #數據分割
21 train_pop = population[:int(len(population) * 0.8)]
22 train_year = years[:int(len(years) * 0.8)]
23 test_pop = population[int(len(population) * 0.8):]
24 test_year = years[int(len(years) * 0.8):]
```

# 實作：臺南市人口散布圖與回歸曲線

使用 NumPy 套件建立回歸曲線。
曲線的最高次方 t 可自行調整。

```
25 #設定曲線範圍
26 xx = np.linspace(years[0], years[-1], 50)
27 #設定回歸曲線最高項次
28 t = 3
29 #建立回歸曲線函數
30 omega = np.polyfit(train_year, train_pop, t)
31 func_pop = np.poly1d(omega)
```

使用 plot 繪製折線圖，使用 scatter 繪製散布圖。

```
32 #畫出回歸曲線
33 plt.plot(xx, func_pop(xx), color='green')
34 #畫出訓練用座標（綠色）
35 plt.scatter(train_year, train_pop, marker='x', c='green')
36 #畫出測試用座標（紅色）
37 plt.scatter(test_year, test_pop, marker='x', c='red')
```

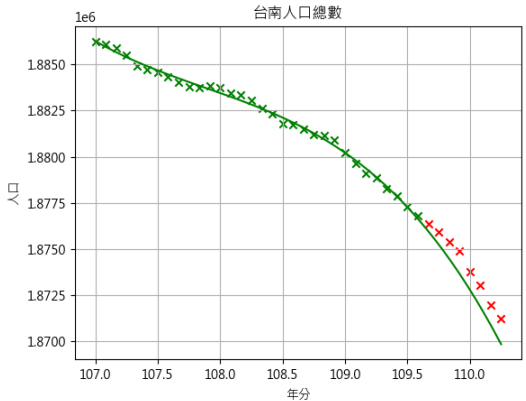
# 實作：臺南市人口散布圖與回歸曲線

## 設定外觀並繪製

```
38 plt.title('台南人口總數')
39 plt.xlabel('年分')
40 plt.ylabel('人口')
41 plt.grid() #顯示網格
42 plt.rcParams['font.sans-serif'] = ['Microsoft JhengHei']
43 #plt.rcParams['font.sans-serif'] = ['Heiti TC'] #MacOS
44 plt.show()
```

# 實作：臺南市人口散布圖與回歸曲線

綠色 X 是作為訓練的數據，曲線是訓練出的回歸曲線，用來預測紅色 X 的數據。



# 實作：臺南市人口散布圖與回歸曲線

事實上，「回歸」屬於人工智慧機器學習中的「監督式學習」。
回歸模型常用來做數值關係的預測，例如
未來人口變化、車禍頻率、股票漲跌、原物料價格等。
本次使用的回歸模型是多項式回歸。另外還有
線性回歸、邏輯回歸等許多不同的回歸模型。

END