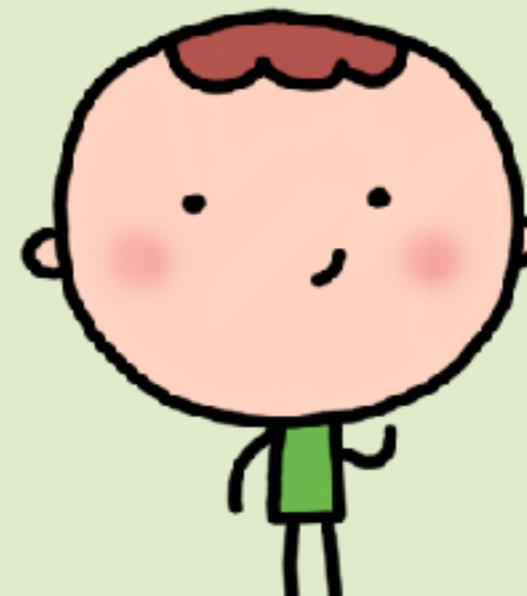
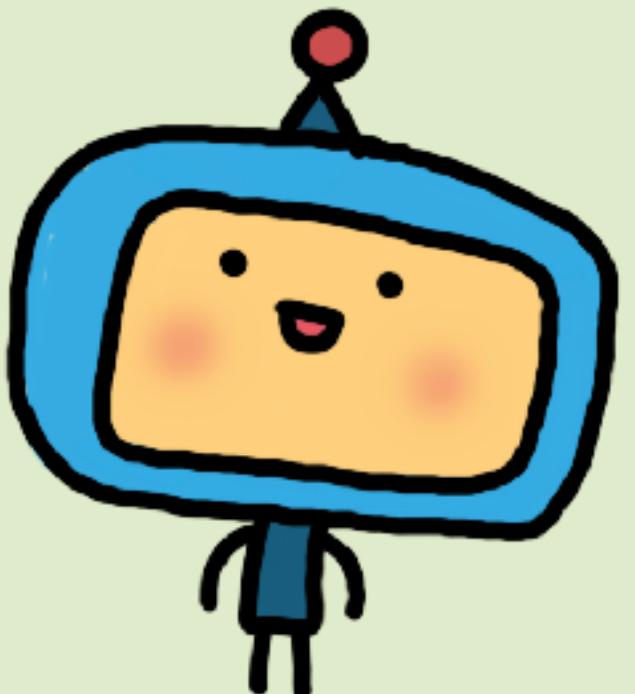
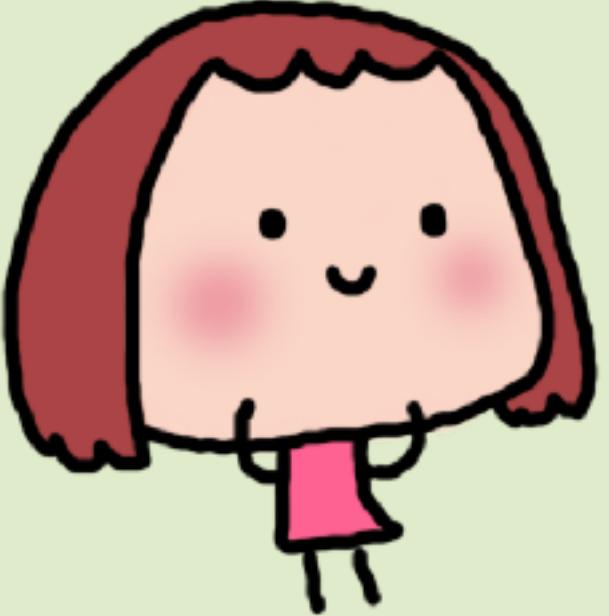


# 手把手打開 Python 資料分析大門

台灣資料科學年會系列活動

蔡炎龍

政治大學應用數學系



# 課程介紹

- 為什麼講師是這位？
- 為什麼要開這門課？
- 為什麼要教這些東西？



# 關於我

學歷

美國爾灣加州大學 (UC Irvine) 數學博士

現職

政治大學應用數學系副教授

## 關於我

寫程式出版



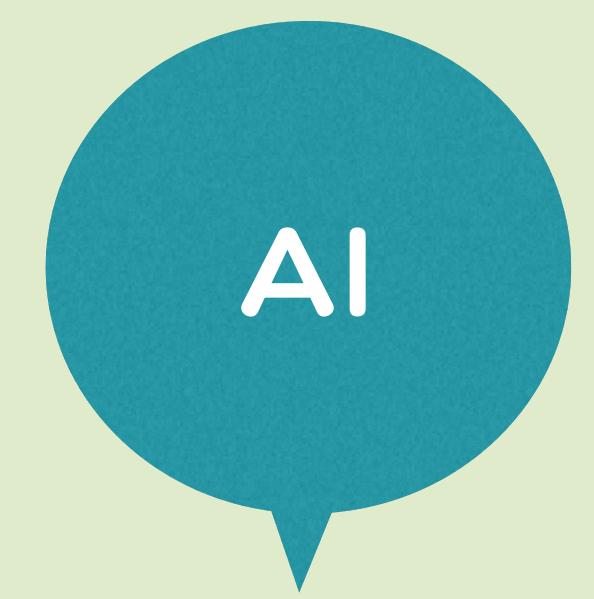
高中



大學

電腦社社長  
/程式助教

神經網路



碩士班

發現 Python



博士班

推廣 Python



現在

## 關於我

2000 年時, 加入 Python Software Activity,  
當時只有 271 人

- 212. [Mr. Paolo G. Cantore](#), Ludwigshafen, Germany
- 213. [Wilhelm G. Fitzpatrick](#), Seattle, USA
- 214. [Dr. Douglas K. Cooper](#), Tigard, Oregon
- 215. [Mr. Wolfgang H Feix](#), Hilzingen, Germany
- 216. [Mr. Stuart M Ford](#), Grafton, WI
- 217. [Mr. Sreeni R Nair](#), Parlin, New Jersey
- 218. [Dr. Michael R Haggerty](#), Cambridge, MA
- 219. [Mr. Thomas M. G. Bennett](#), Boone, North Carolina
- 220. [Mr. Joseph T Bore, Jr](#), Hoboken, NJ
- 221. [Mr. Yen-lung Tsai](#), Irvine, CA
- 222. [Mr. Conrad Schneiker](#), Austin, TX
- 223. [Mr. Ron West](#), ACT, Australia
- 224. [Dr. Luby Liao](#), San Diego, CA
- 225. [Jameson A Quinn](#), Seattle, wa
- 226. [Peter Kropf](#), Sunnyvale, CA
- 227. [Jonathan McLin](#), Tempe, AZ
- 228. [Nils Fischbeck](#), Stralsund, Germany
- 229. [Mike Howard](#), Cobleskill, NY
- 230. [Dr. Alex Martelli](#), BO, Italia
- 231. [Julio Carrera](#), Boston, MA
- 232. [Mr. David Walter Schere](#), Annapolis , MD

交了 50 美元的年費!

## 開課目的

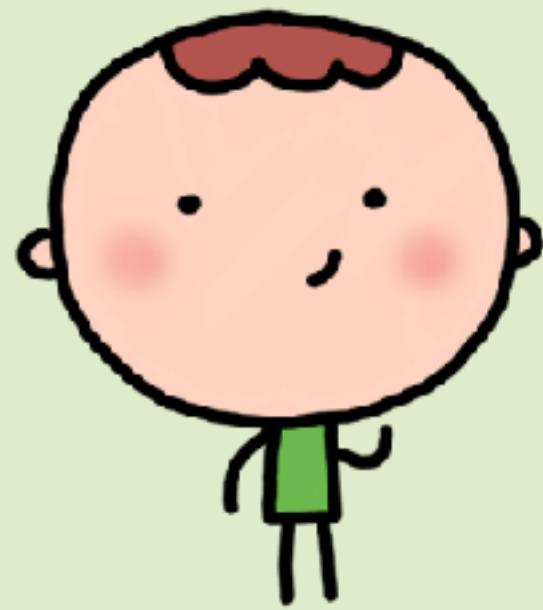


學了 Python 基本語  
法, 但要做資料分析時覺  
得卡卡的!

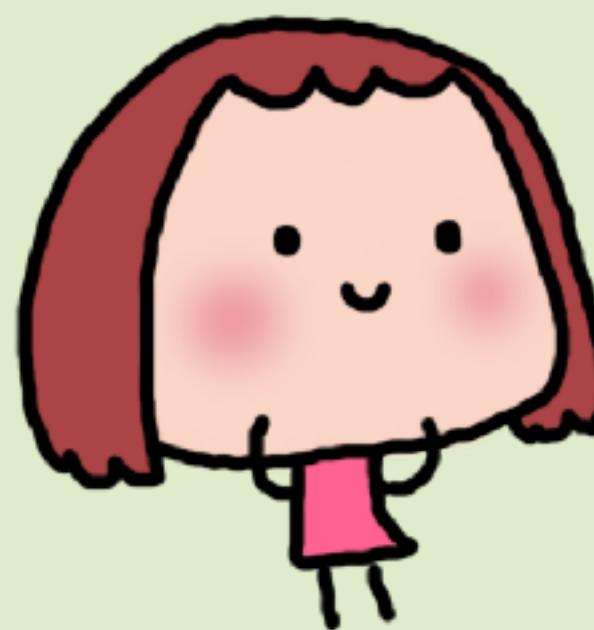
## 課程內容



Jupyter



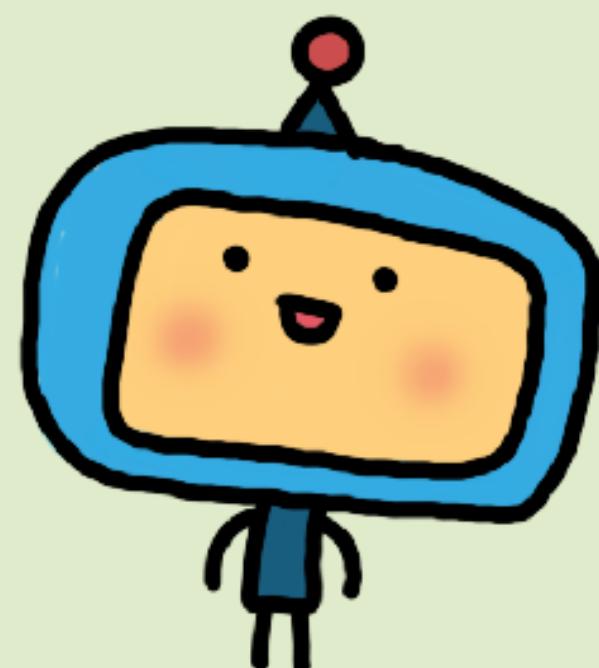
NumPy



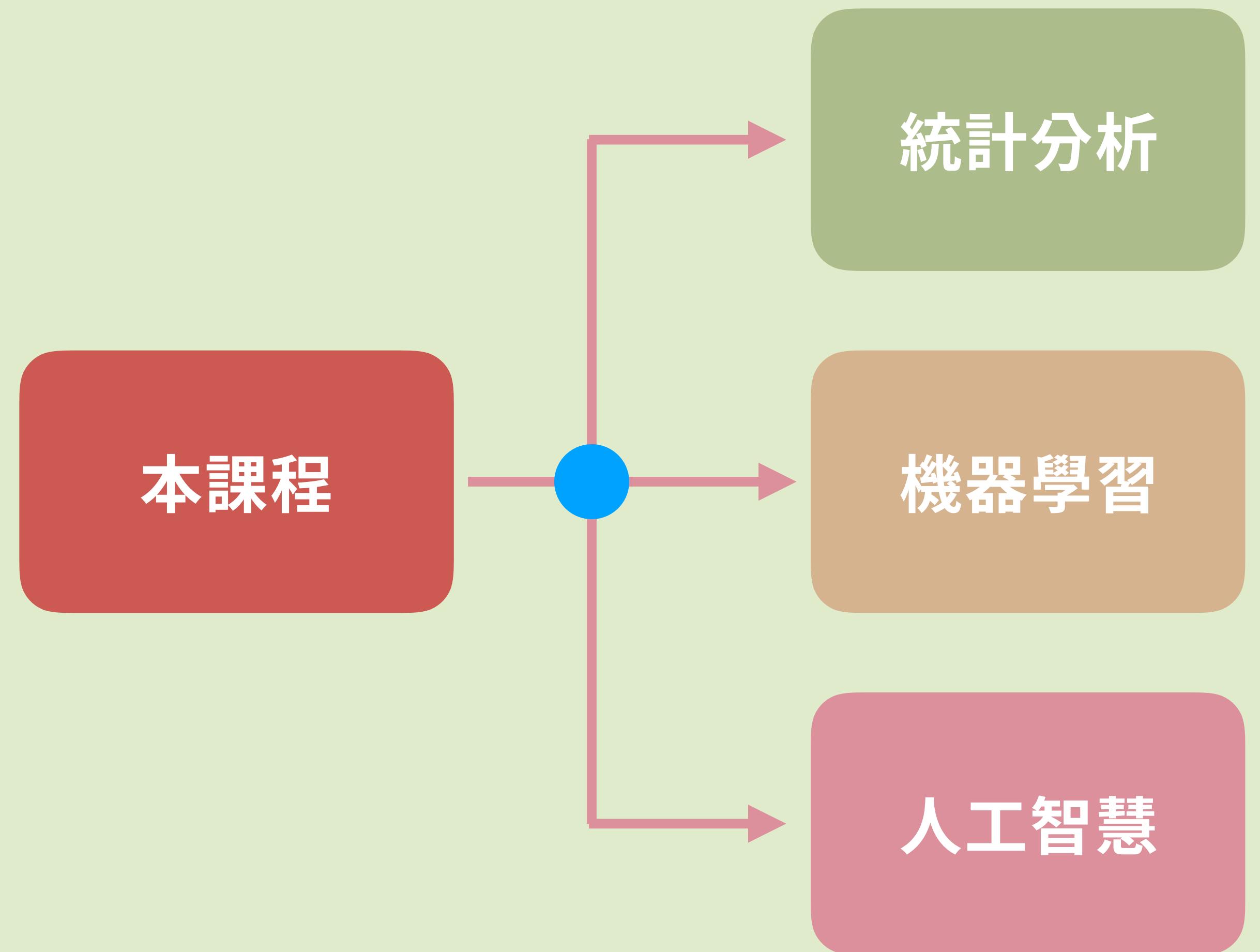
Matplotlib



Pandas



SciKit Learn



1

# Jupyter Notebook

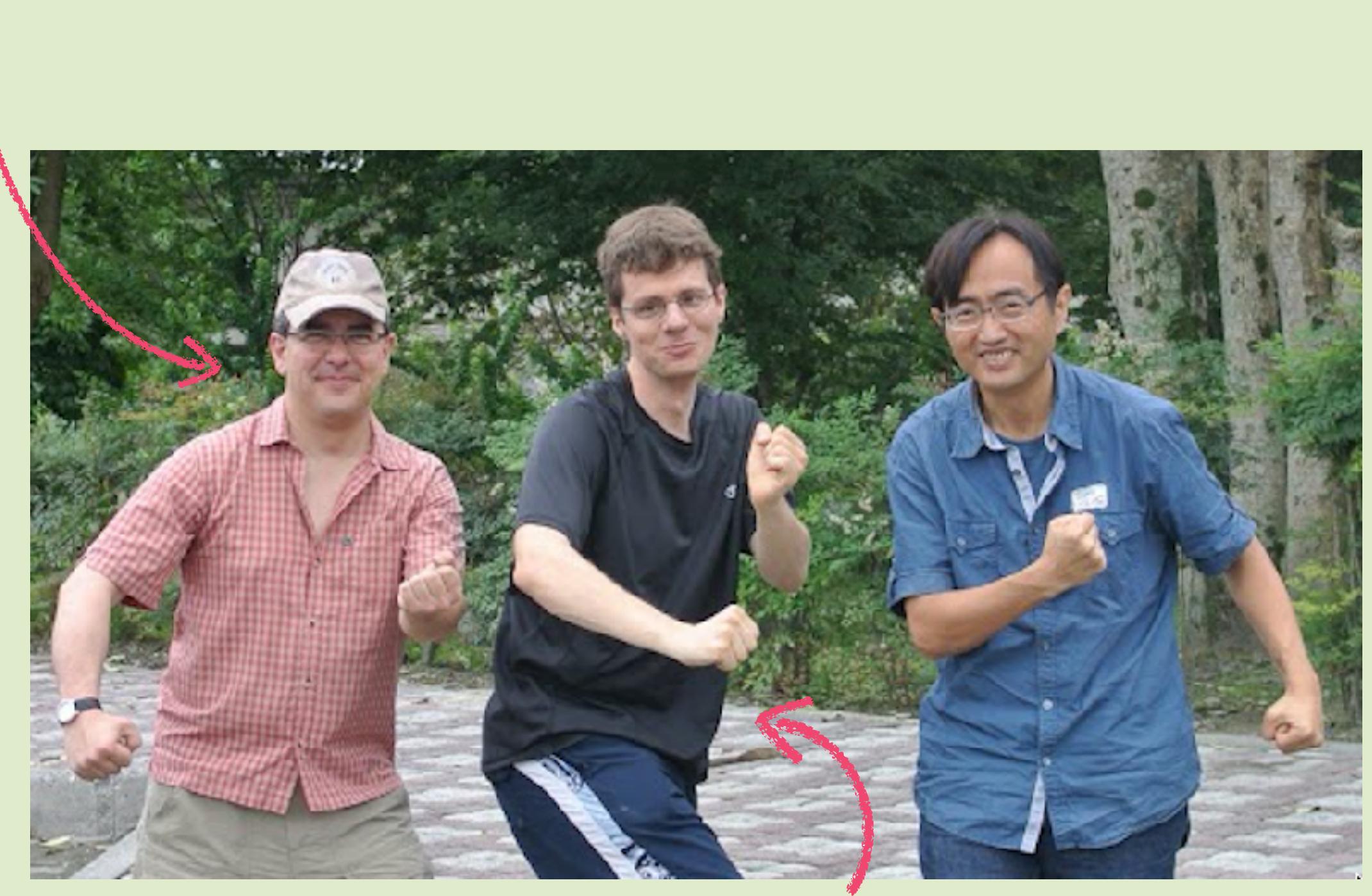
資料分析的主流平台



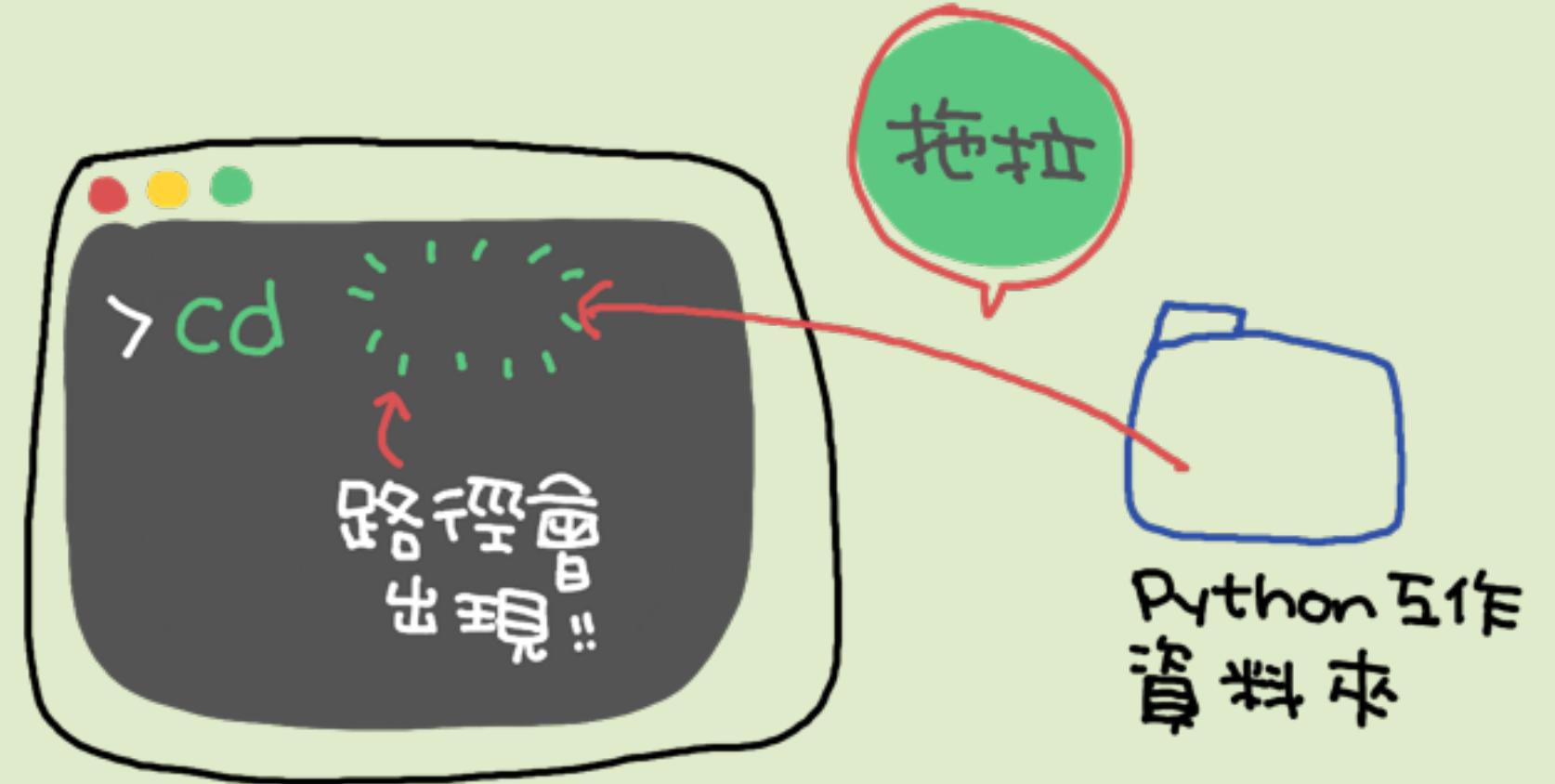
原作者

# Fernando Pérez

- 出生於哥倫比亞的美德林
- 物理學博士
- 柏克萊加州大學統計系教授
- 博士時代開發 IPython (Jupyter 前身)



Andreas Klöckner (PyCuda 作者)



打開終端機，進入你的工作資料夾。

接著執行：

> **jupyter notebook**

記得在每個 cell 中是按

In[ ]:

shift - enter

執行

# Jupyter 的奇幻世界之一

## 魔術指令

## 魔術指令

%

Jupyter 有很多魔術指令，都是以 % 開頭，讓我們享有諸多方便功能。

指令	說明
%cd	如系統的 cd, 即更改路徑。
%save	%save spam.py 1-3 7 9 將 1-3, 7, 9 輸入格存成 spam.py。
%run	%run spam.py 執行 spam.py。
%timeit	儲存格執行若干次，量時間。

我們的標準魔術指令。

```
%matplotlib inline
```

圖都直接顯示在網頁的頁面。

我們的標準開始，引入三個套件：

- numpy
- matplotlib
- pandas

```
%matplotlib inline  
  
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd
```

! **pwd**

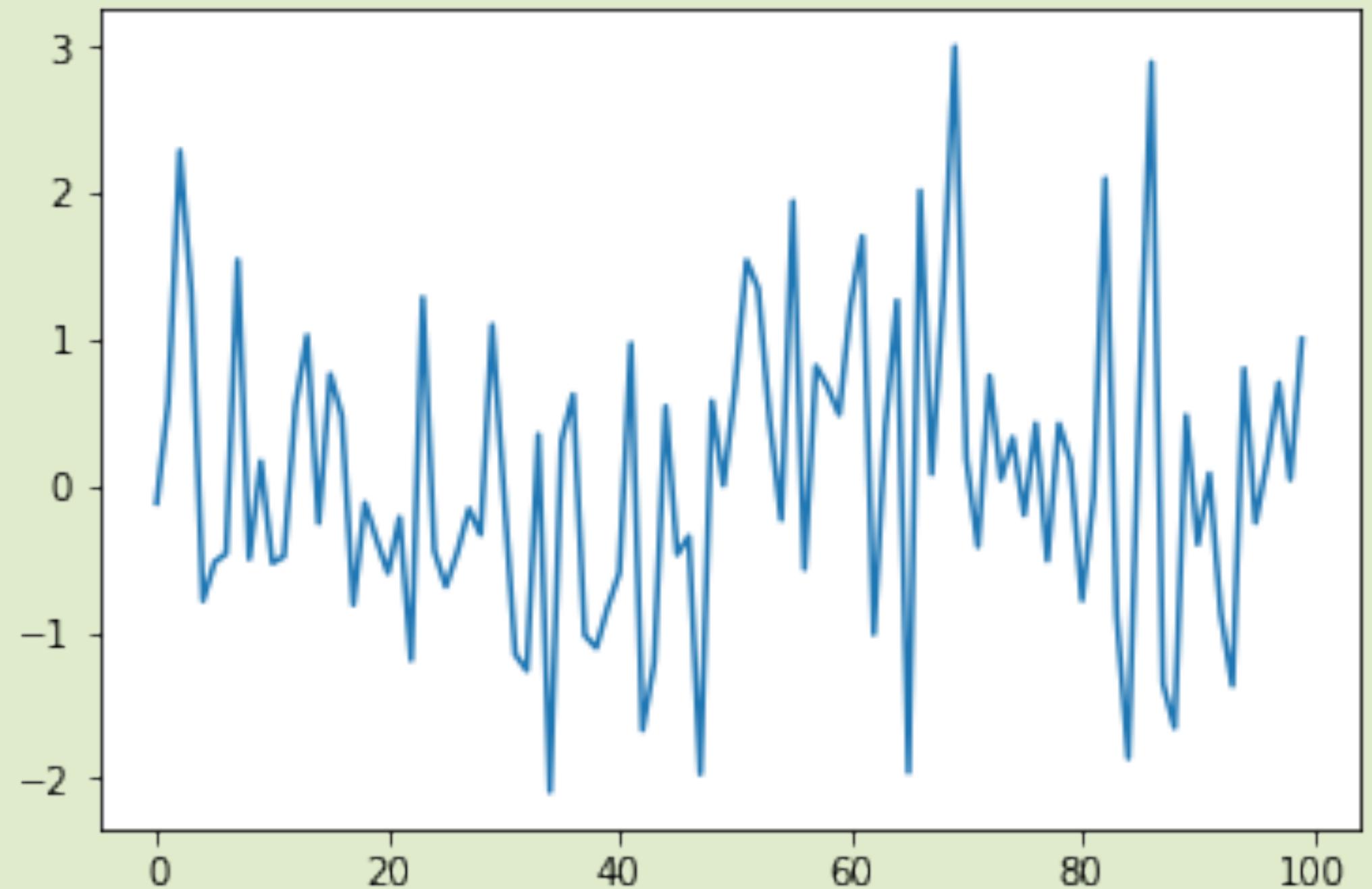
比魔術指令更炫的驚嘆號!  
直接用系統指令。

# Jupyter 的奇幻世界之二

## 自由方便的試驗場

由標準常態分布隨機  
取 100 個數。

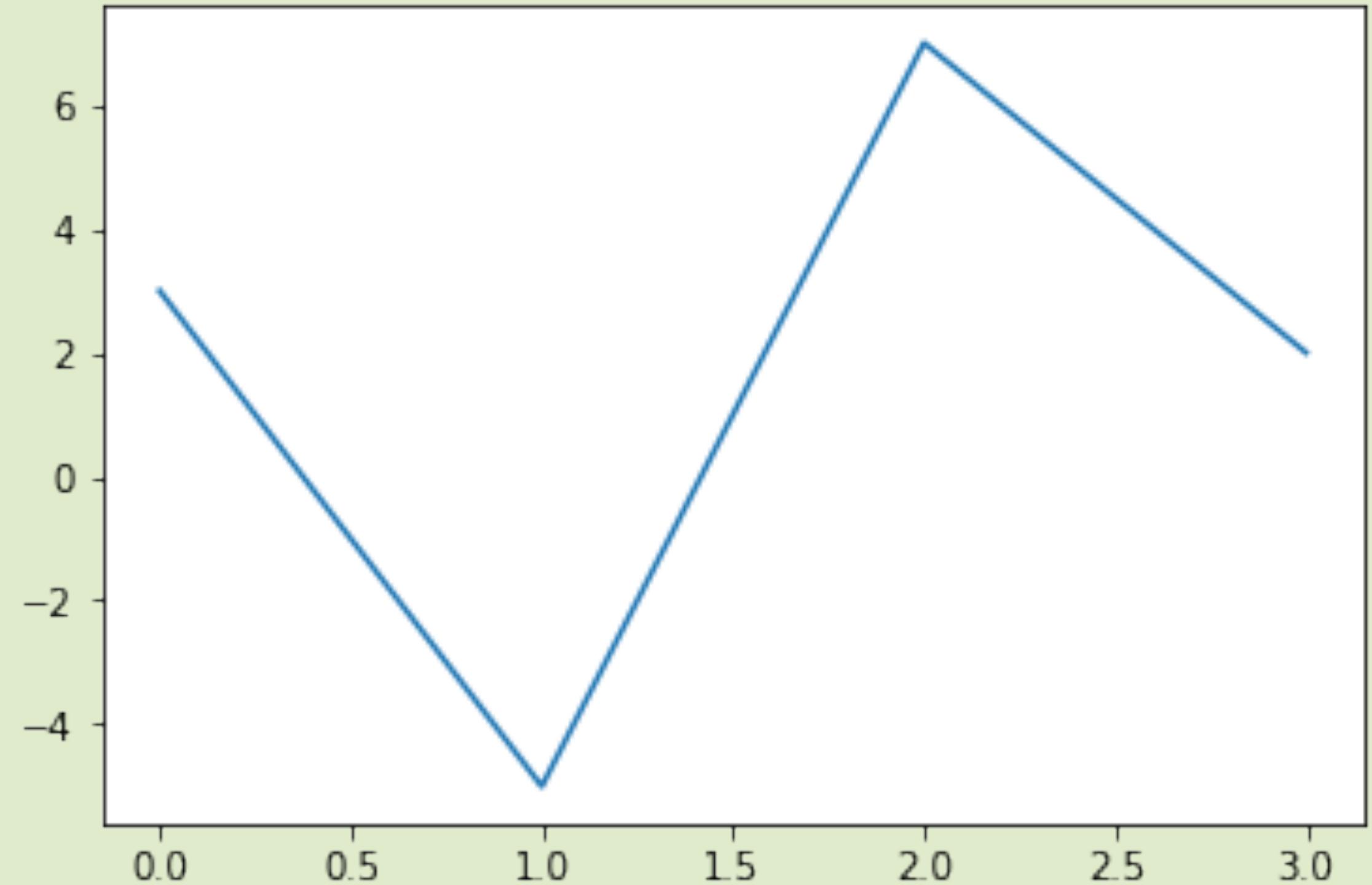
```
plt.plot(np.random.randn(100))
```



`plt.plot` 是用 `matplotlib` 畫圖。例如

```
plt.plot([3, -5, 7, 2])
```

是畫出  $(0, 3)$ ,  $(1, -5)$ ,  $(2, 7)$ ,  $(3, 2)$  幾個點再連起來。





`plt.plot(x, y)`

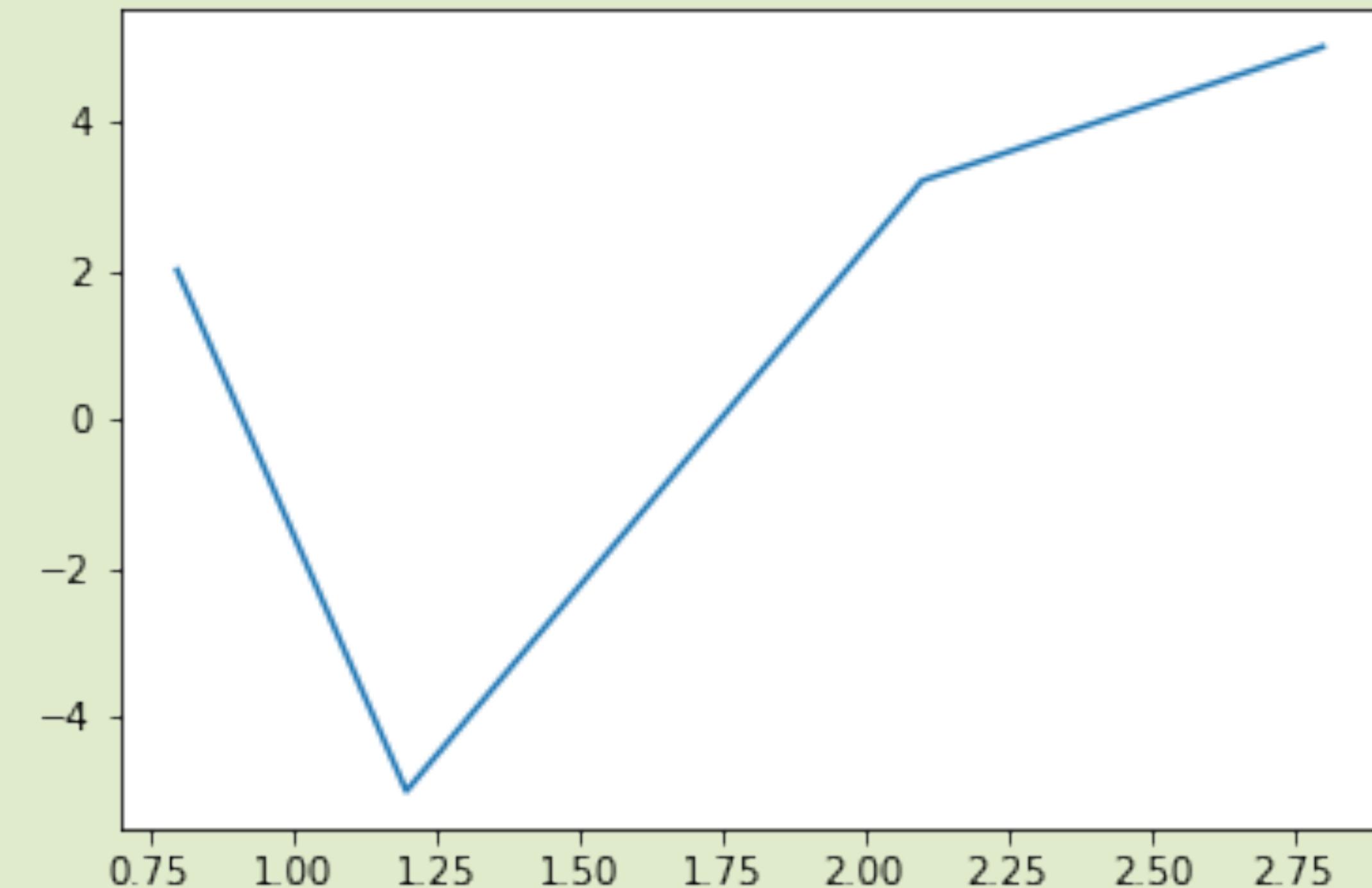
點的 x 座標 list (array)

點的 y 座標 list (array)

例子

## 標準折線圖

我們有點  $(0.8, 2)$ ,  $(1.2, -5)$ ,  
 $(2.1, 3.2)$ ,  $(2.8, 5)$ , 把它們連起來的圖畫出來。



```
plt.plot([0.8, 1.2, 2.1, 2.8], [2, -5, 3.2, 5])
```

## 問題

# 點的表示法

```
points = [(0.8, 2),  
          (1.2, -5), (2.1, 3.2),  
          (2.8, 5)]
```

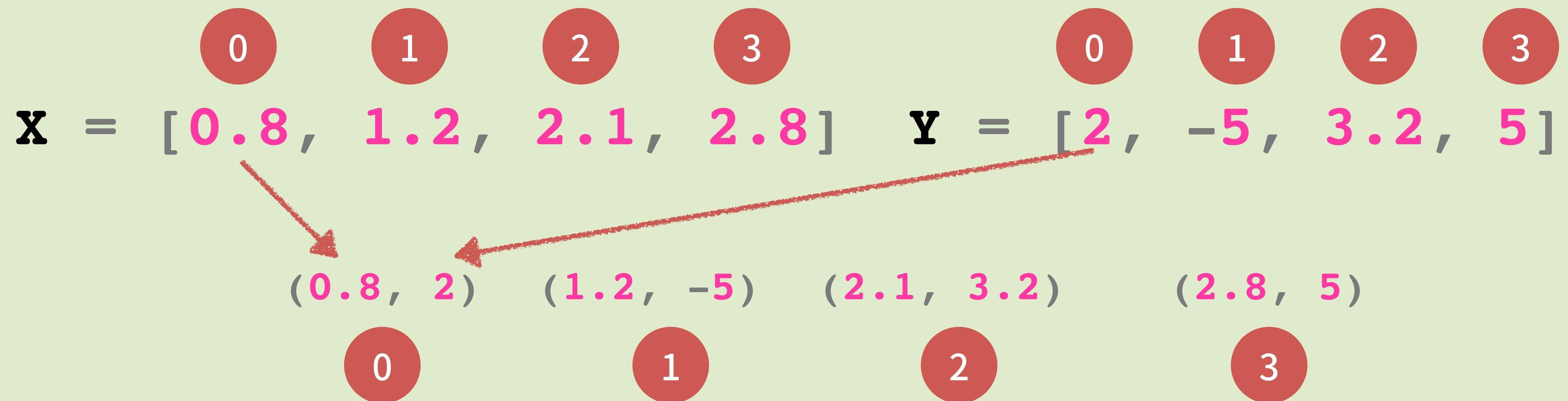
```
x = [0.8, 1.2, 2.1, 2.8]  
y = [2, -5, 3.2, 5]
```

有時我們是左邊表示點, 有時又要用右邊 x, y 座標分開。這要怎麼互換呢?

## 小技巧

# zip 和 unzip

**zip** 可以想成兩串或更多串資料，同編號放一起的動作。



## 小技巧

# zip 和 unzip

```
x = [0.8, 1.2, 2.1, 2.8]  
y = [2, -5, 3.2, 5]  
  
list(zip(x, y))
```

這合理



輸出: [(0.8, 2), (1.2, -5), (2.1, 3.2), (2.8, 5)]

## 小技巧

# zip 和 unzip

```
points = [(0.8, 2),  
(1.2, -5), (2.1, 3.2),  
(2.8, 5)]
```

```
x, y = zip(*points)
```

結果:

points 裡的點一個一個拿出來

```
x = (0.8, 1.2, 2.1, 2.8)  
y = (2, -5, 3.2, 5)
```

unzip 實際上就是 zip, 而且是一樣的邏輯! 想想為什麼。

這啥?



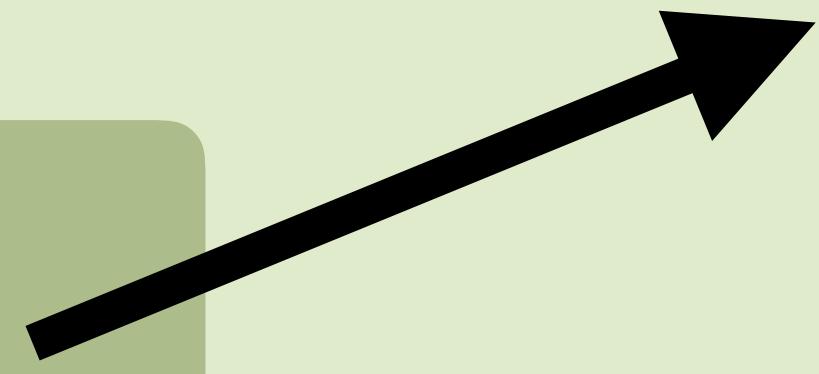


我需要幫助!

`print?`

Jupyter Notebook 當然可以按問號再 **shift-enter**, 但這很遜。

`print`



打到一半按

shift

tab

說明就出現了!

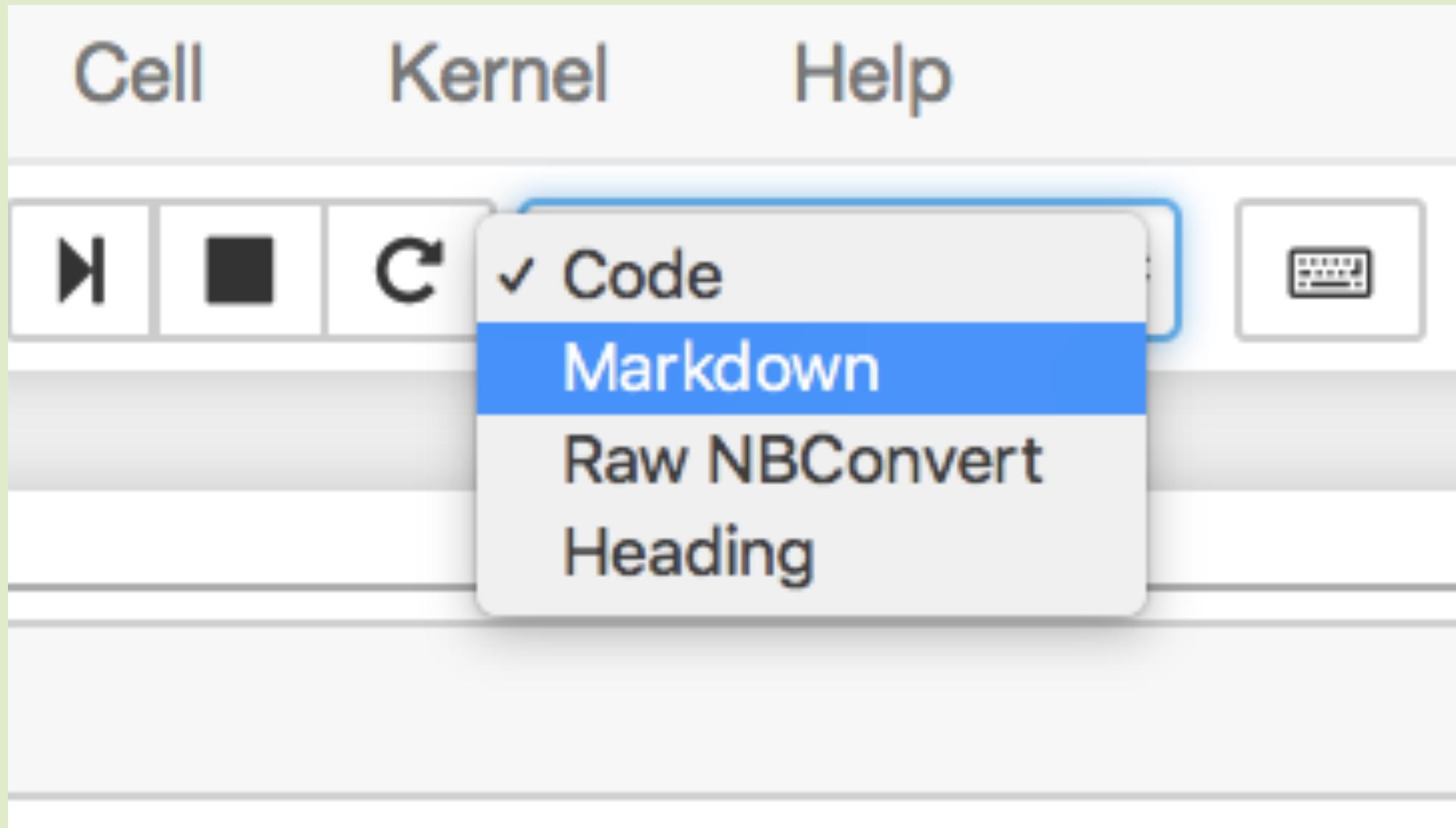
按兩次會出現說明全文。

tab

請愛用天下第一神鍵，補完指令。

# Jupyter 的奇幻世界之三

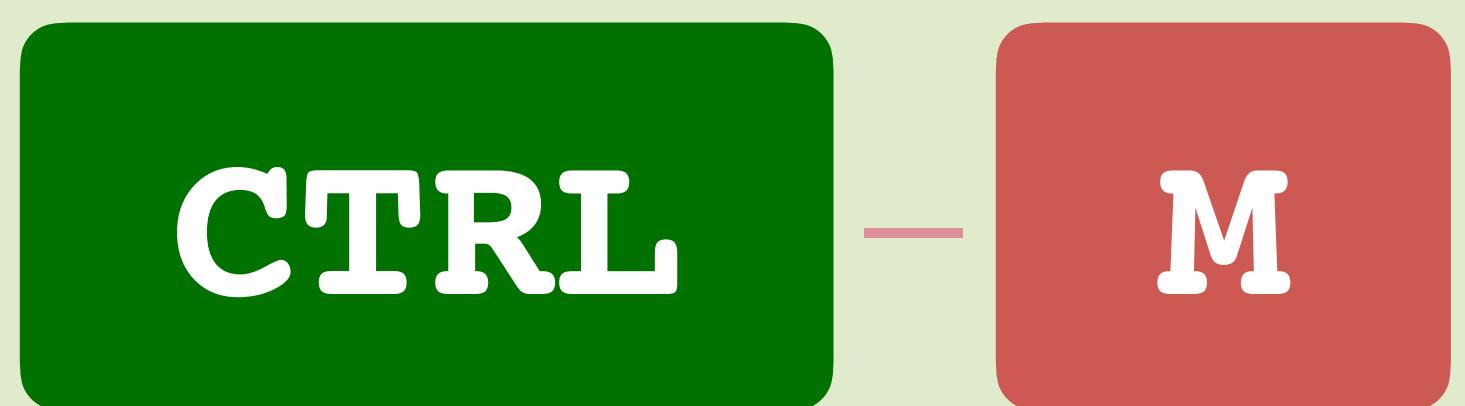
## Markdown 筆記



用選單選 **Markdown** 格式，就可以用  
Markdown 語法做筆記！

插撥

# 神秘的組合技



進入 meta 狀態

再按

按鍵	說明
M	切換為 Markdown 模式。
Y	切回 Code 模式。
L	顯示/隱藏 line number。
F	搜尋/置換。

## Markdown

### 標題

Markdown 基本上你可以打任意文字進去, 要加標題等等可以這樣做。

```
# Python 數據分析
```

```
## 簡介
```

我們可以用 `numpy` 等套件。

## Python 數據分析

### 簡介

我們可以用 numpy 等套件。

無序的條列方式。有序的條列打入

1., 2., 3. 等等即可。

### **# # 我們要學的套件**

- \* **Matplotlib**
- \* **Numpy**
- \* **Pandas**

### **我們要學的套件**

- **Matplotlib**
- **Numpy**
- **Pandas**

Markdown

## 超連結

網頁超連結用法。

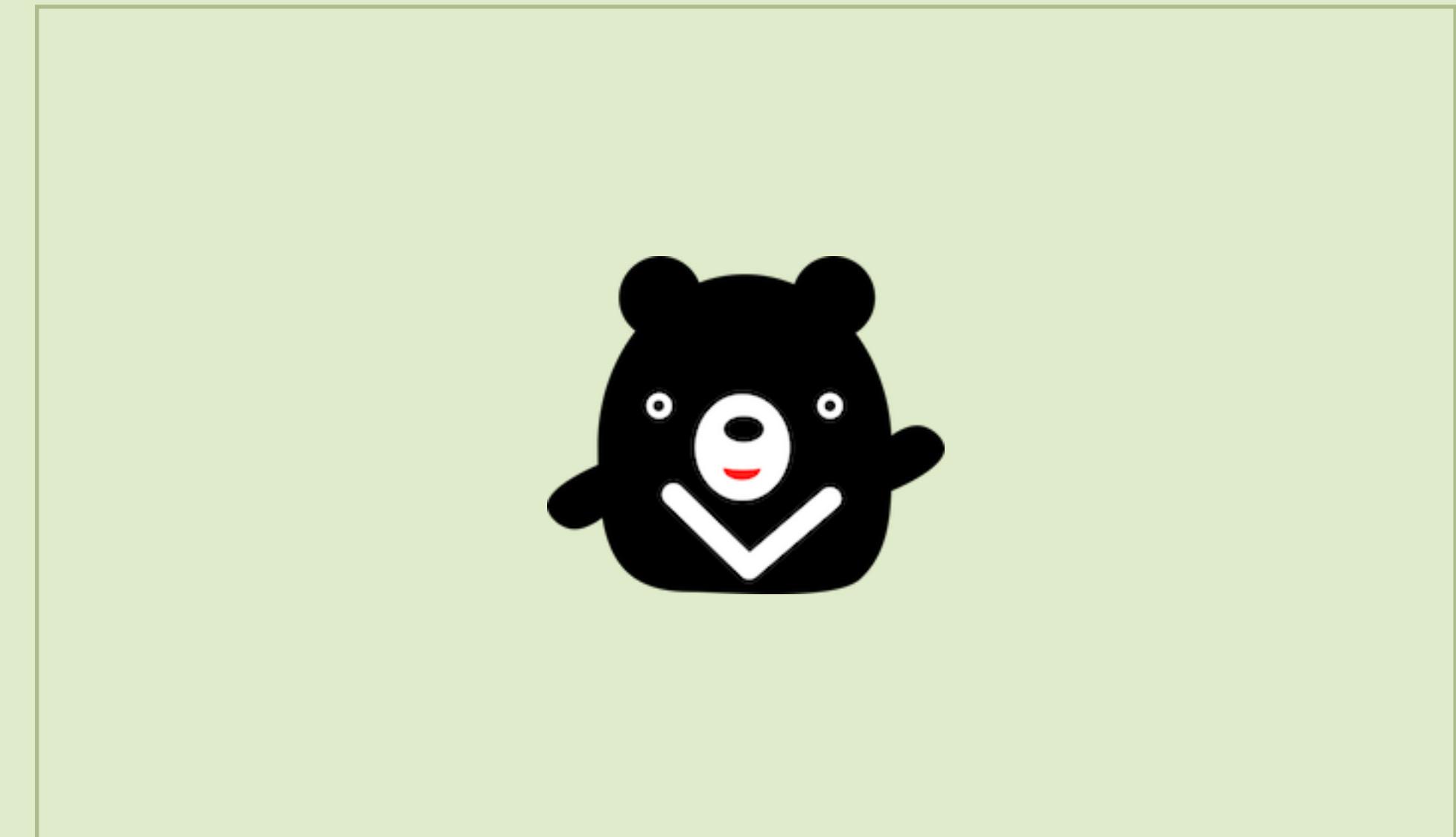
[台灣人工智慧學校]  
(<http://aiacademy.tw/>)

台灣人工智慧學校

## Markdown 圖片

假設有個檔名叫 **bear.png** 的圖片。

```
![台灣黑熊] (bear.png)
```



支援 LaTeX!

給個函數  $f(x) = x^2$ , 求它的積分

$\int_a^b f(x) dx$

給個函數  $f(x) = x^2$ , 求它的積分

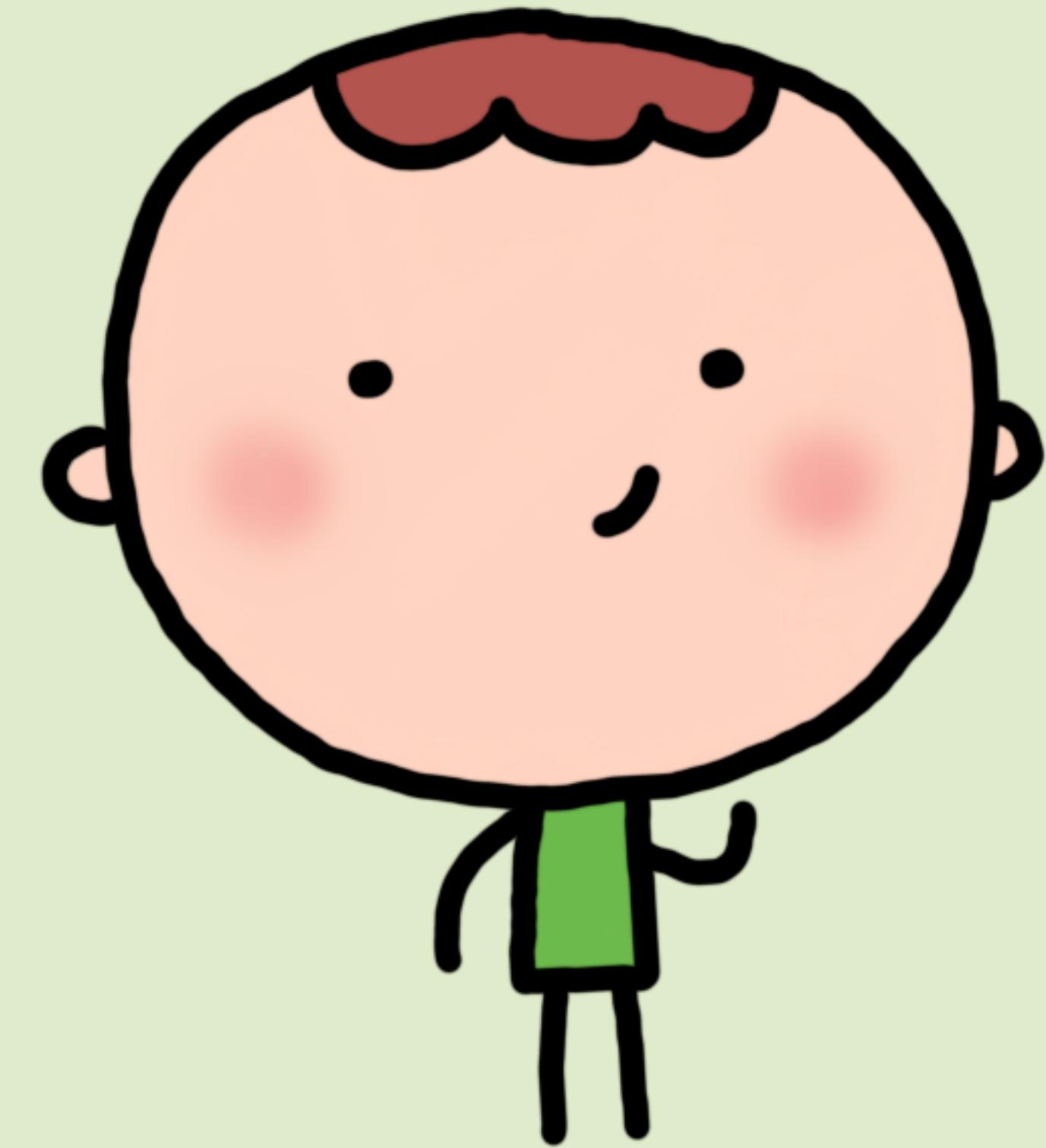
$$\int_a^b f(x) dx$$

# Jupyter 的奇幻世界之四

## 超炫的互動

## 準備進入互動模式

```
from ipywidgets import interact
```



寫個函數就可以互動!

```
def f(x):  
    print(x)
```

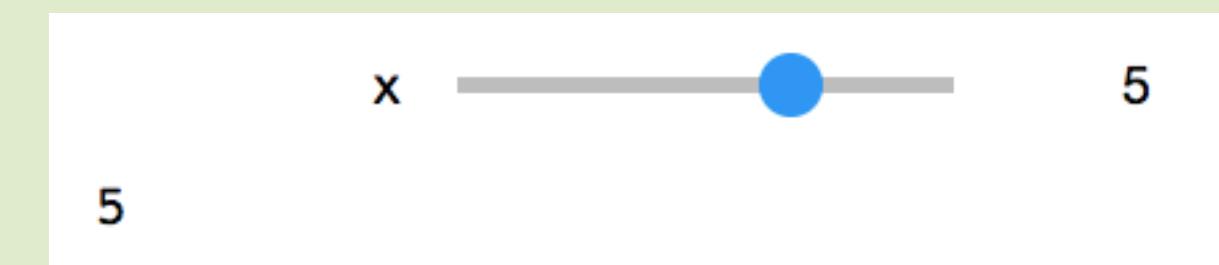
寫個任何帶參數的函數。



`interact(f, x=3)`

要互動的函數

參數設任一資料型態



插播

## 如果互動出不來

請在終端機打入：

```
jupyter nbextension enable --py widgetsnbextension
```



## 重點

一個資料型態就對應一種互動

`interact(f, x=3.)`

浮點數的數值滑桿

`interact(f, x=list("ABC"))`

下拉式選單



例子

## 給預設值

給個預設值

```
def move(n=1):  
    print(" "*n + "oooo")  
  
interact(move, n=(1,50))
```



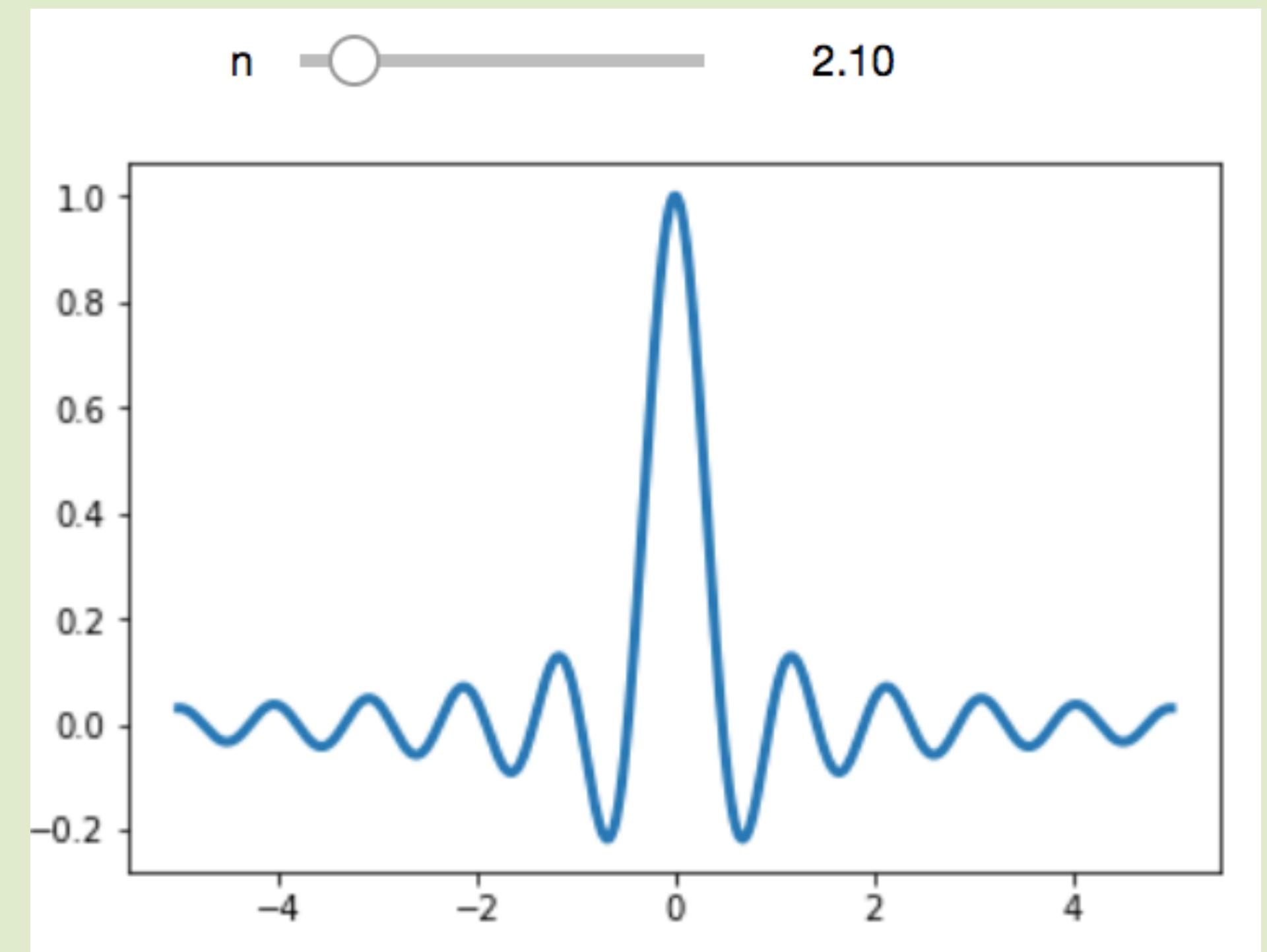
## 例子

# 視覺化的互動

```
x = np.linspace(-5, 5, 1000)

def draw(n=1):
    y = np.sinc(n*x)
    plt.plot(x, y, lw=3)
    plt.show()
```

注意一般在 Jupyter Notebook 畫圖不用這行。

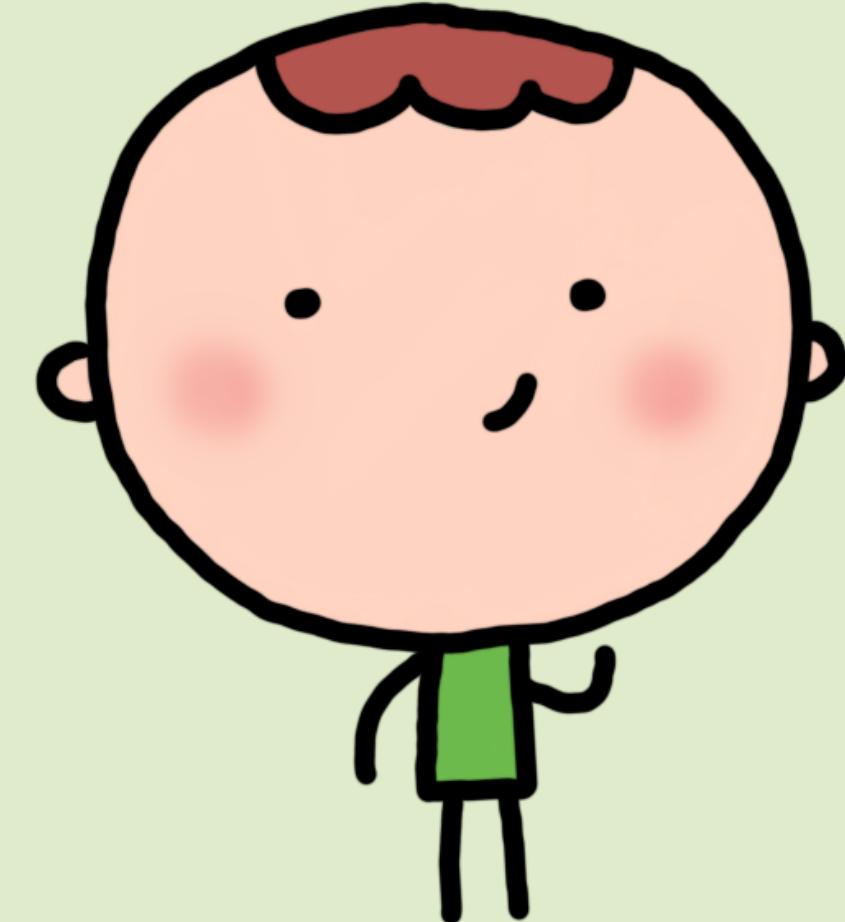


```
interact(draw, n=(1., 10.))
```

2

# NumPy

陣列導向的程式設計



原作者

# Travis Oliphant

- Anaconda 是他創立的 Continuum Analytics 產品
- 數學/電機碩士
- 有在 PyCon Taiwan 紿過 Keynote



Travis Oliphant 在 PyCon Taiwan 2012  
<https://youtu.be/vrPRwUOt-7k>

# array 和我們老朋友 list 很像...

```
mylist = [1, 2, 3, 4]  
  
myarr = np.array(mylist)
```

array([1, 2, 3, 4])



超炫的

array oriented



例子

## 每位同學加 5 分

考試成績存成一個叫  
**grades** 的 array, 老師說  
每位同學都加 5 分!

```
grades = np.array([87, 65, 77, 93])  
  
grades + 5
```

輸出: array([92, 70, 82, 98])

## 例子

### 整批匯率轉換

在美國查到號稱 Pentax 三公主的鏡頭 31mm, 43mm, 77mm 價格, 我們想知道合台幣多少。

```
prices = np.array([996.95, 596.95,  
796.95])  
  
prices * 29.99
```

輸出:

```
array([ 29898.5305, 17902.5305,  
23900.5305])
```

## 例子

# 成績計算

一位老師成績這樣算的：

- 平時成績 20%
- 期中考 35%
- 期末考 45%

```
grades = np.array([85, 70, 80])
weights = np.array([0.2, 0.35, 0.45])

weighted_grades = grades * weights
weighted_grades.sum()
```

某同學成績如下：

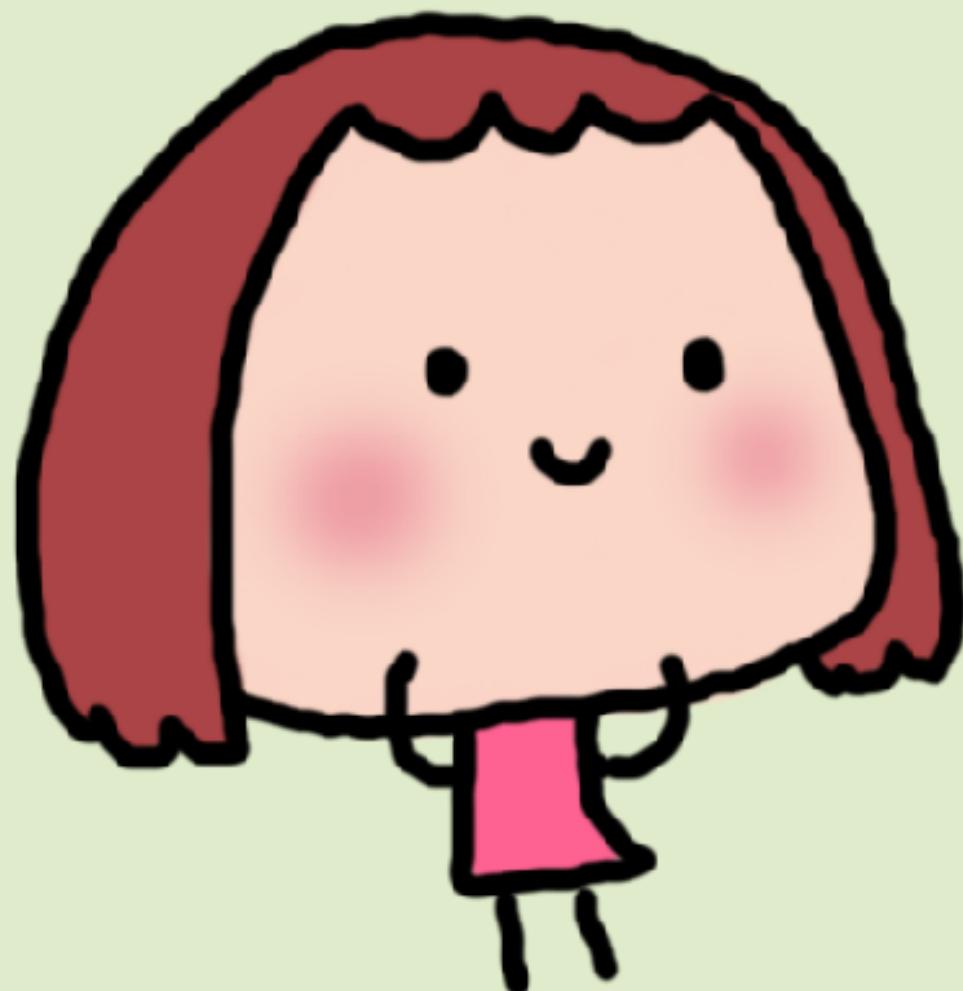
- 平時成績 85 分
- 期中 70 分
- 期末 80 分

輸出: 77.5

例子

## 成績計算

最後兩行其實換成內  
積就可以。



```
grades = np.array([85, 70, 80])  
weights = np.array([0.2, 0.35, 0.45])  
  
np.dot(grades, weights)
```

輸出: 77.5

# 快速生 array 的方法之一

從 0 到 10, 很均勻的找出 100 個點。

```
x = np.linspace(0, 10, 100)
```

輸出: array([0., 0.1010101, ..., 10.])

## 快速生 array 的方法之二

`np.arange` 和 Python 的 `range` 很像, 只是輸出的是 array, 而且範圍可以用浮點數。

```
A = np.arange(10)
```

輸出: `array([1, 2, ..., 10])`

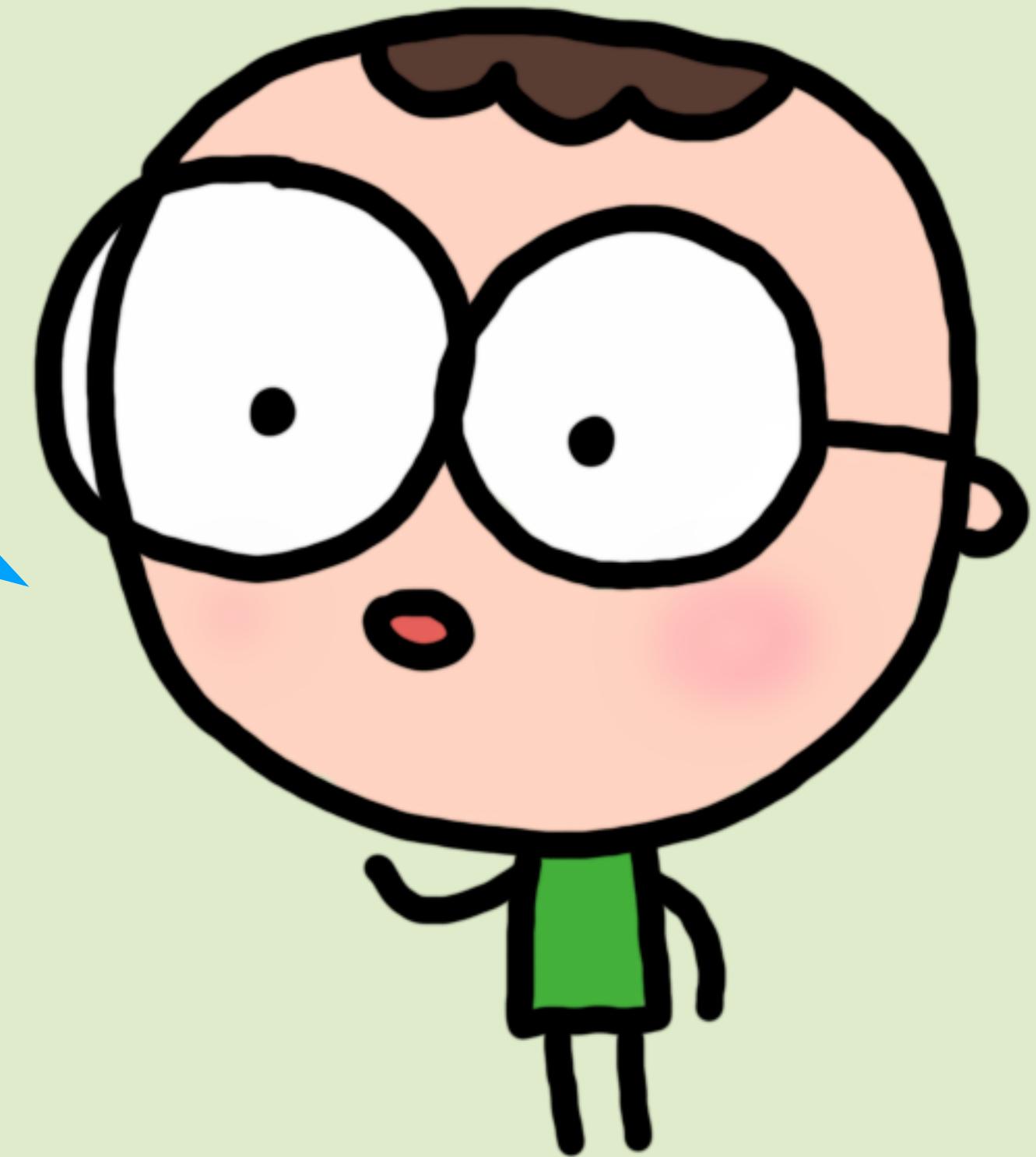


看看 A 這個 array  
的 shape。

A . shape

輸出: (10 , )

熟悉 array 的 shape  
怎麼變來變去是很重要  
的!



## 重點

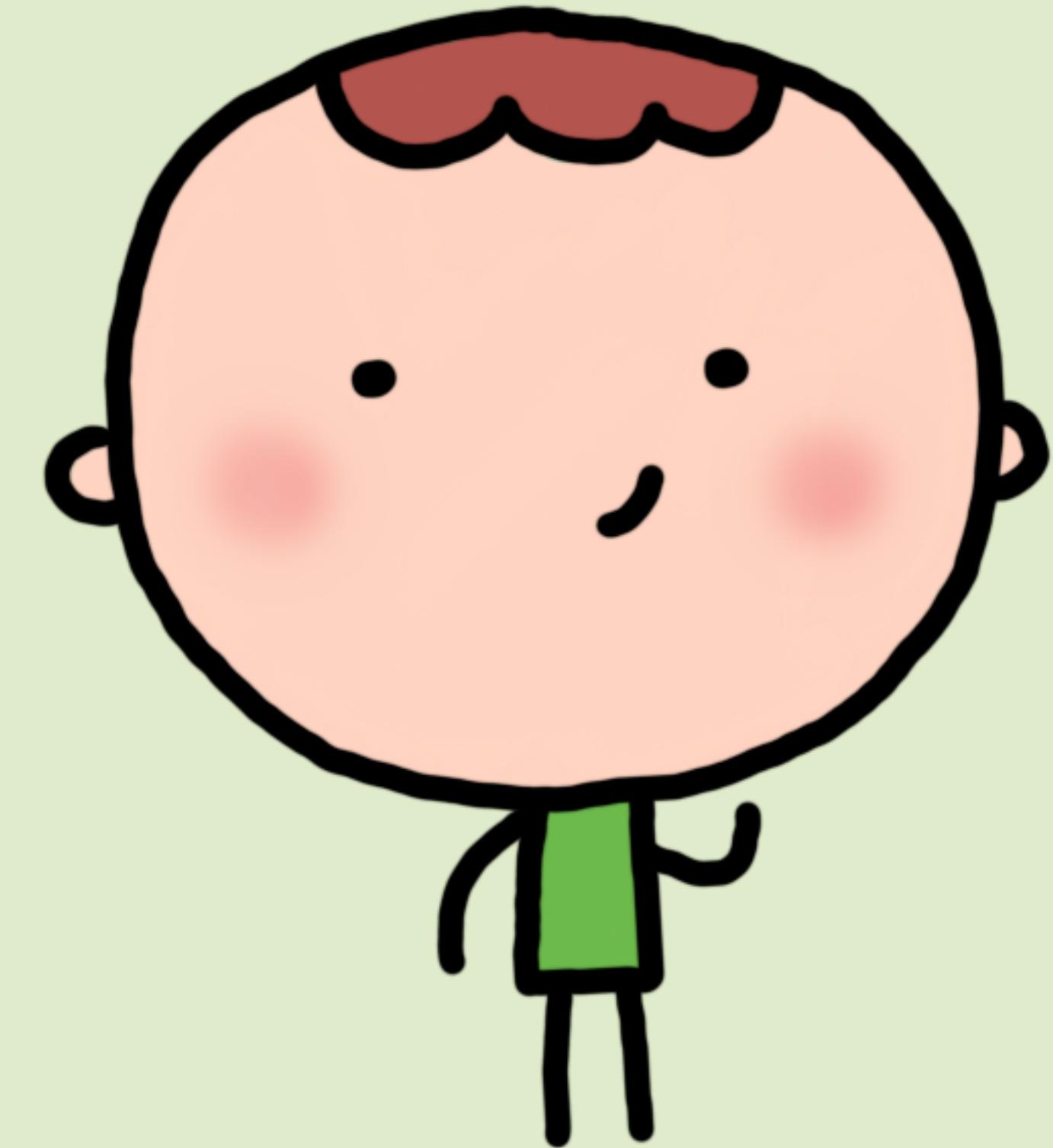
# Shape 更改法 (1)

```
A.shape = (2,5)
```

二維的陣列

結果：

```
A=array([[0, 1, 2, 3, 4],  
         [5, 6, 7, 8, 9]])
```



## 重點

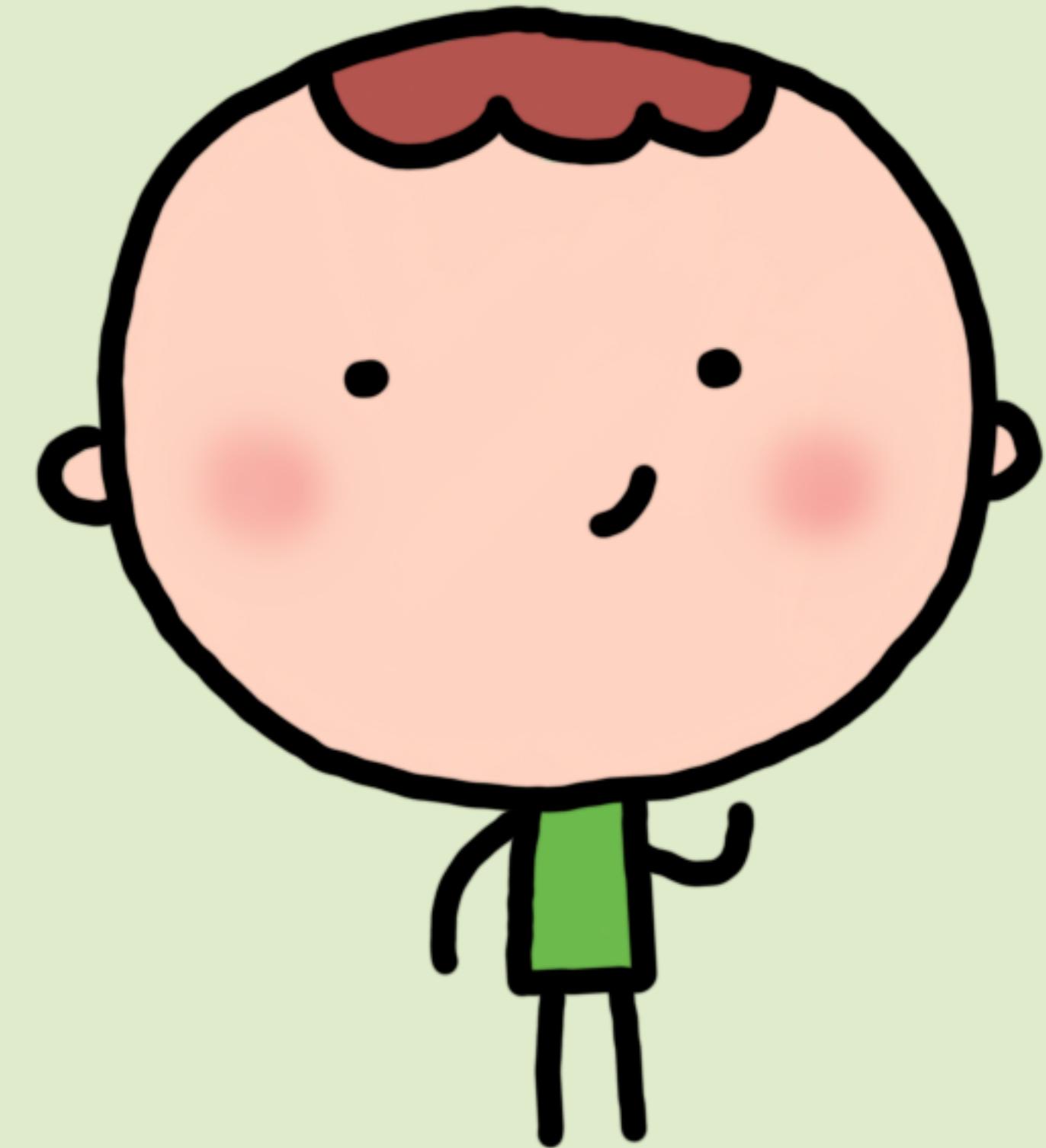
# Shape 更改法 (2)

A. **reshape(5, 2)**

輸出:

```
array([[0, 1],  
       [2, 3],  
       [4, 5],  
       [6, 7],  
       [8, 9]])
```

注意 A 本身  
不會改變



## 重點

# 超炫 reshape 密技

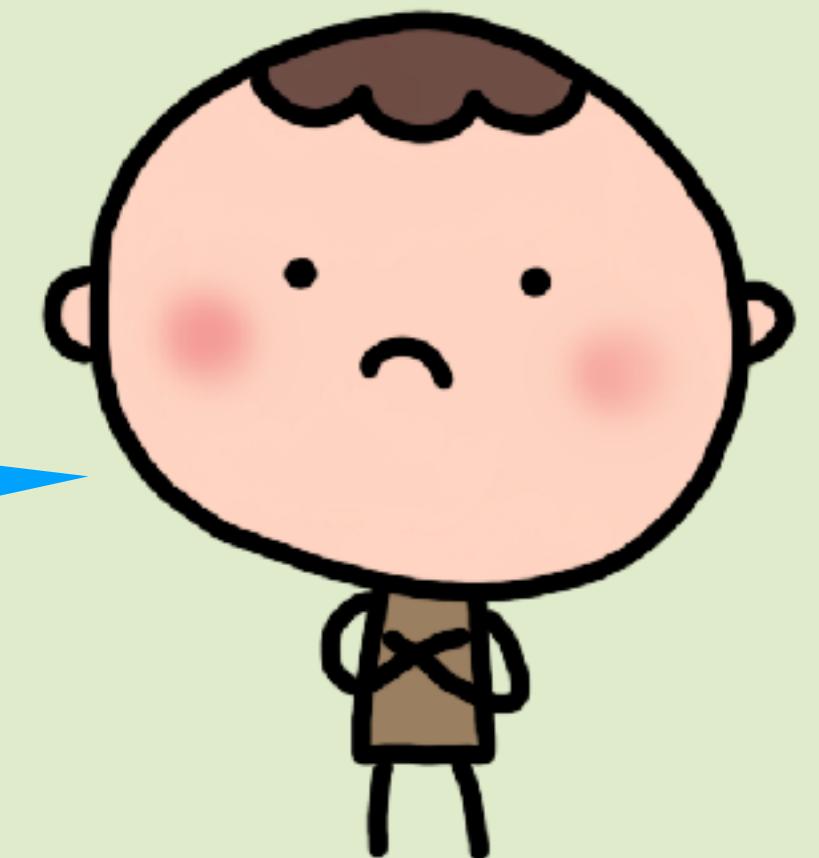
`reshape` 的數字不能亂放的, 比如說我們的例子中

```
A.reshape(a,b)
```

很明顯  $a \times b$  要等於 10。

你一定有個疑問...

這麼明顯 NumPy 是  
不會自己算哦?



結論是可以的! 你可以把不想算的部份放 -1, NumPy 就會自己幫你填! 例如:

```
A.reshape(-1,2)  
A.reshape(5,-1)  
A.reshape(-1,1)
```

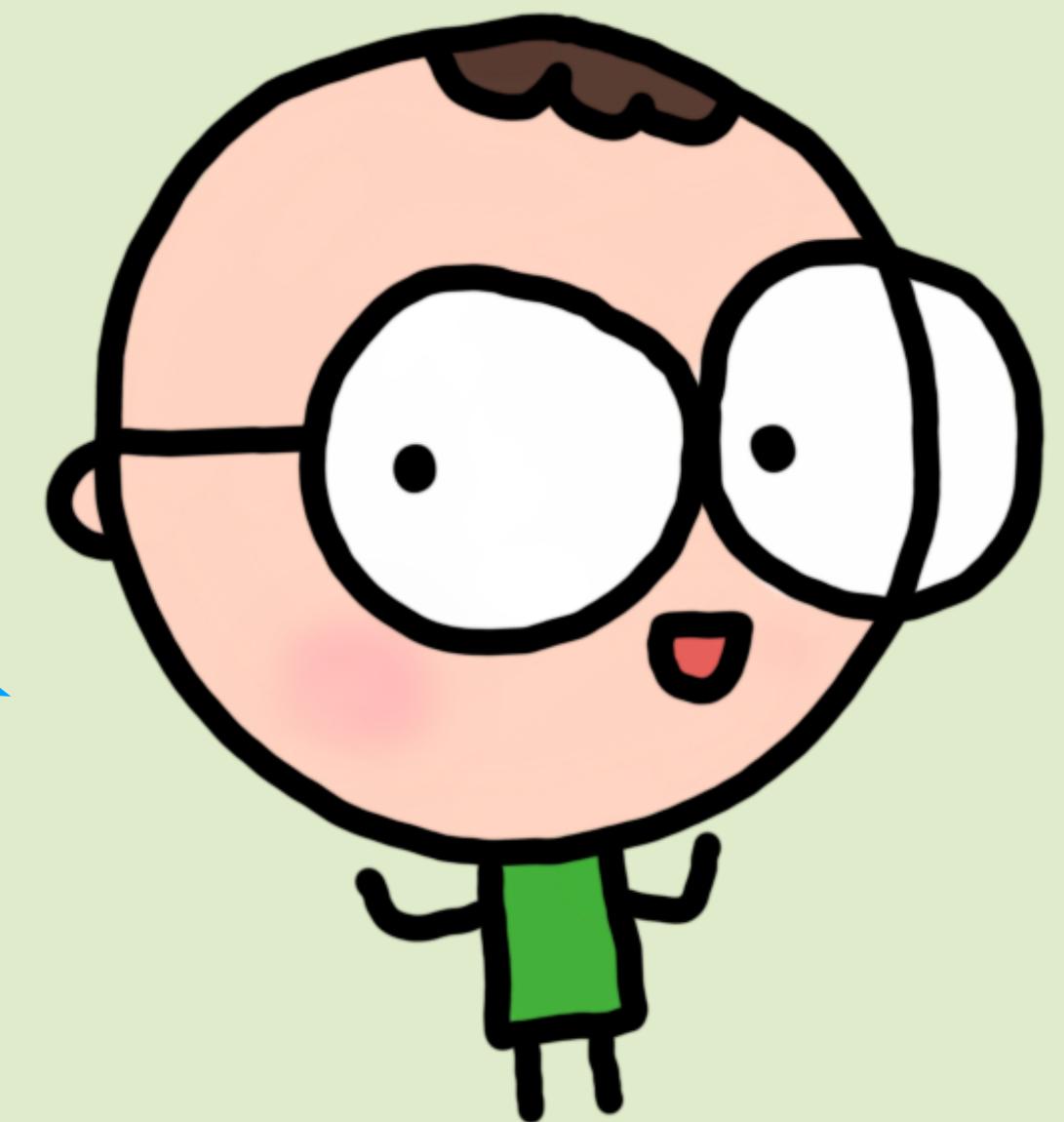
## 重點

# axis 的概念

`axis` 指運算的方向。在二維陣列有兩個方向：沿列 (`axis=0`)，或是沿行 (`axis=1`) 的方向。

`axis=0` ↓ `[[0, 1, 2, 3, 4],  
[5, 6, 7, 8, 9]]`

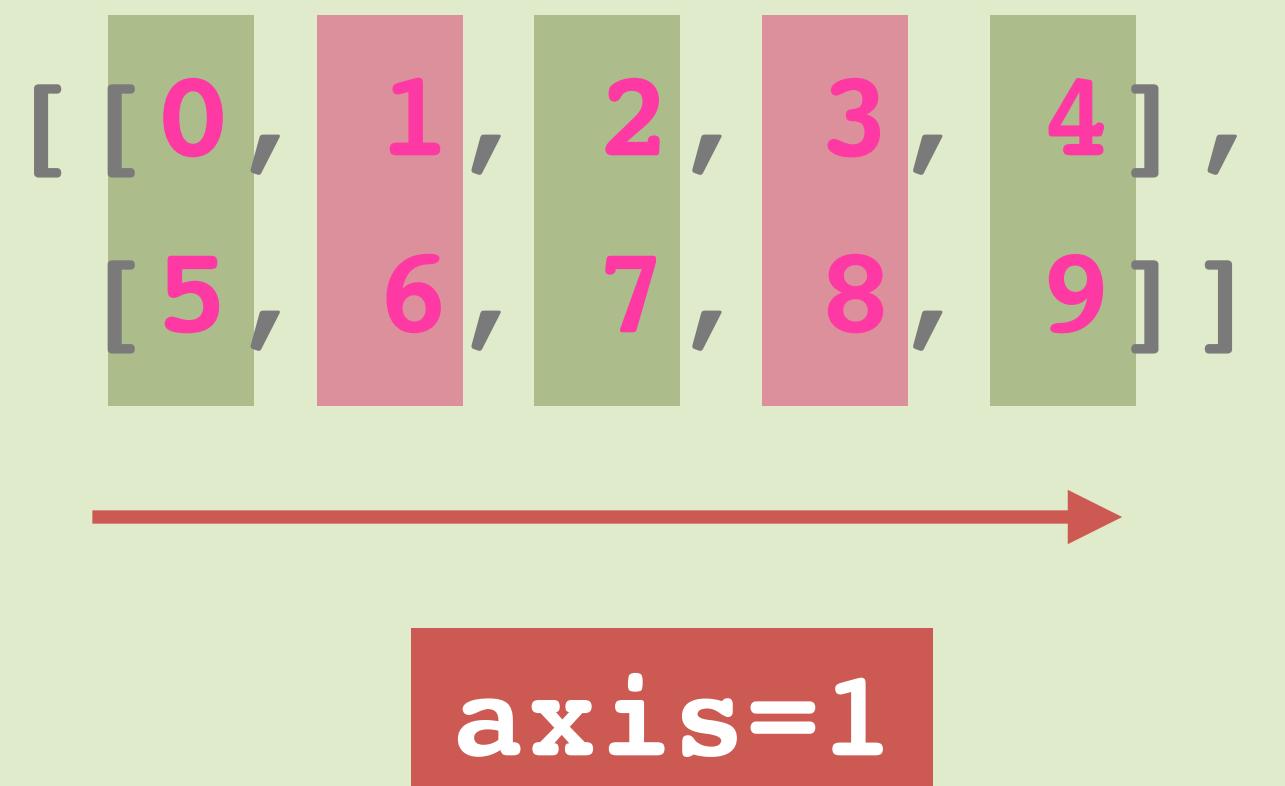
記得類似矩陣的東西  
都是「先列後行」。



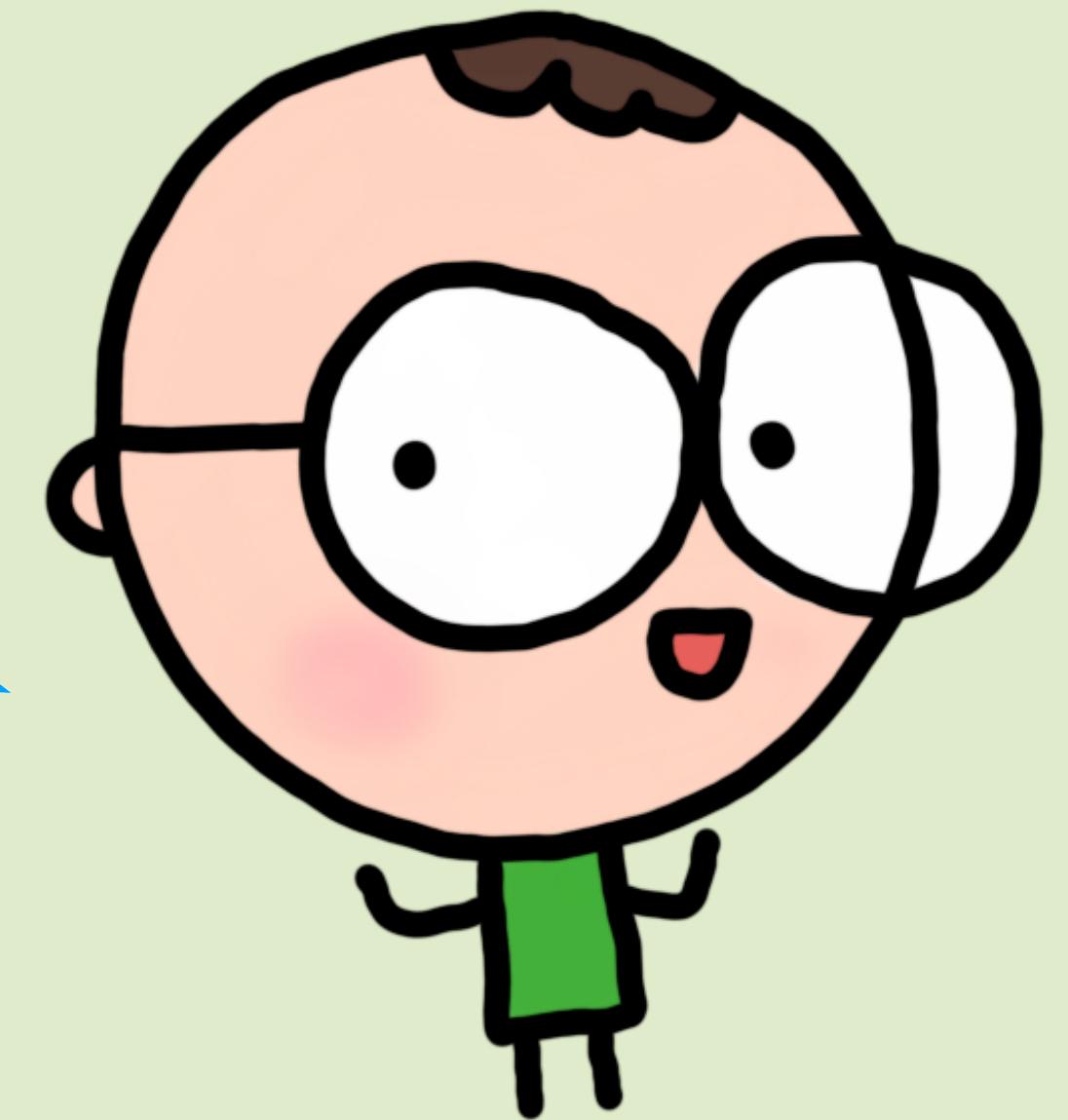
## 重點

# axis 的概念

`axis` 指運算的方向。在二維陣列有兩個方向：沿列 (`axis=0`)，或是沿行 (`axis=1`) 的方向。



記得類似矩陣的東西  
都是「先列後行」。



例子

## 二維陣列求某些元素的和

```
A = array([[0, 1, 2, 3, 4],  
          [5, 6, 7, 8, 9]])
```

```
A.sum()
```

輸出: 45

全加!



例子

## 二維陣列求某些元素的和

```
A = array([[0, 1, 2, 3, 4],  
          [5, 6, 7, 8, 9]])
```

```
A.sum(axis=0)
```

輸出: array([5, 7, 9, 11, 13])

一列一列  
加起來!



例子

## 二維陣列求某些元素的和

```
A = array([[0, 1, 2, 3, 4],  
          [5, 6, 7, 8, 9]])
```

```
A.sum(axis=1)
```

輸出: array([10, 35])

一行一行  
加起來!



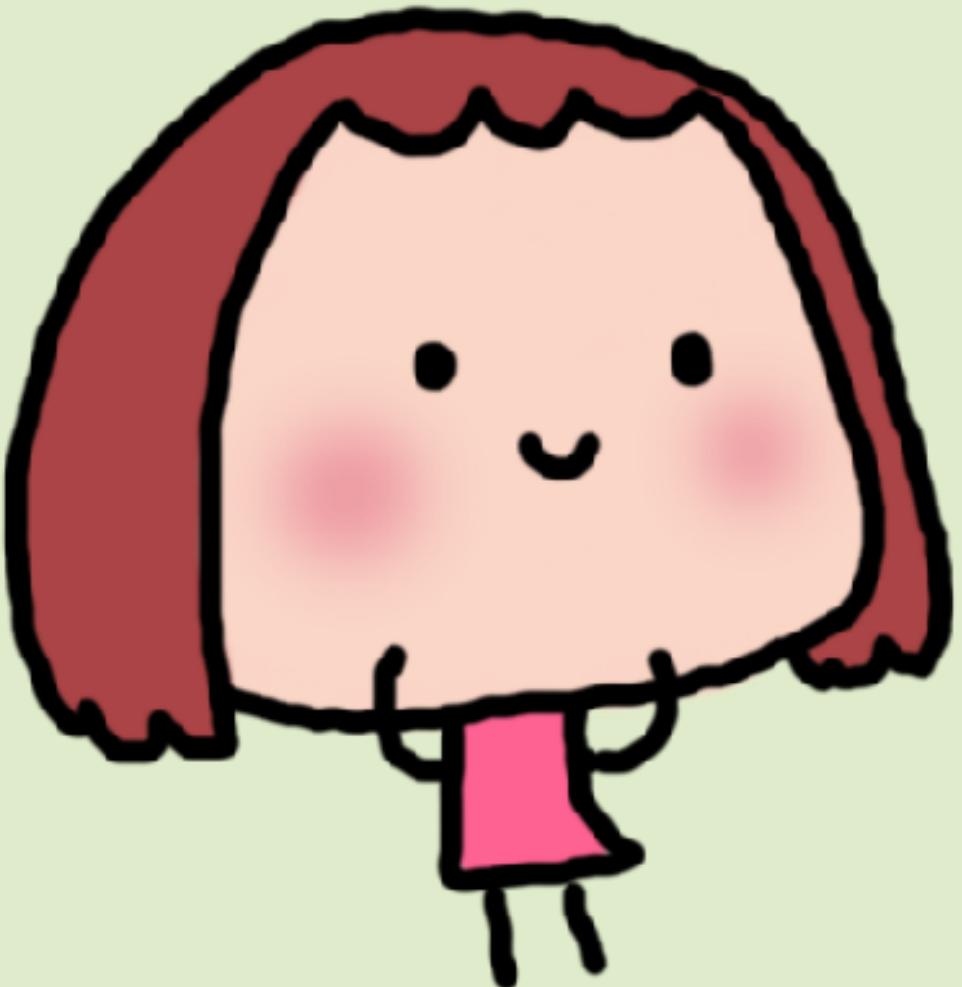
## 重點

# array 過濾器

假設我們有個 array

```
L = np.array([3, -2, -1, 5, 7, -3])
```

我們想取出大於 0 的那些數。



```
L = np.array([3, -2, -1, 5, 7, -3])
```

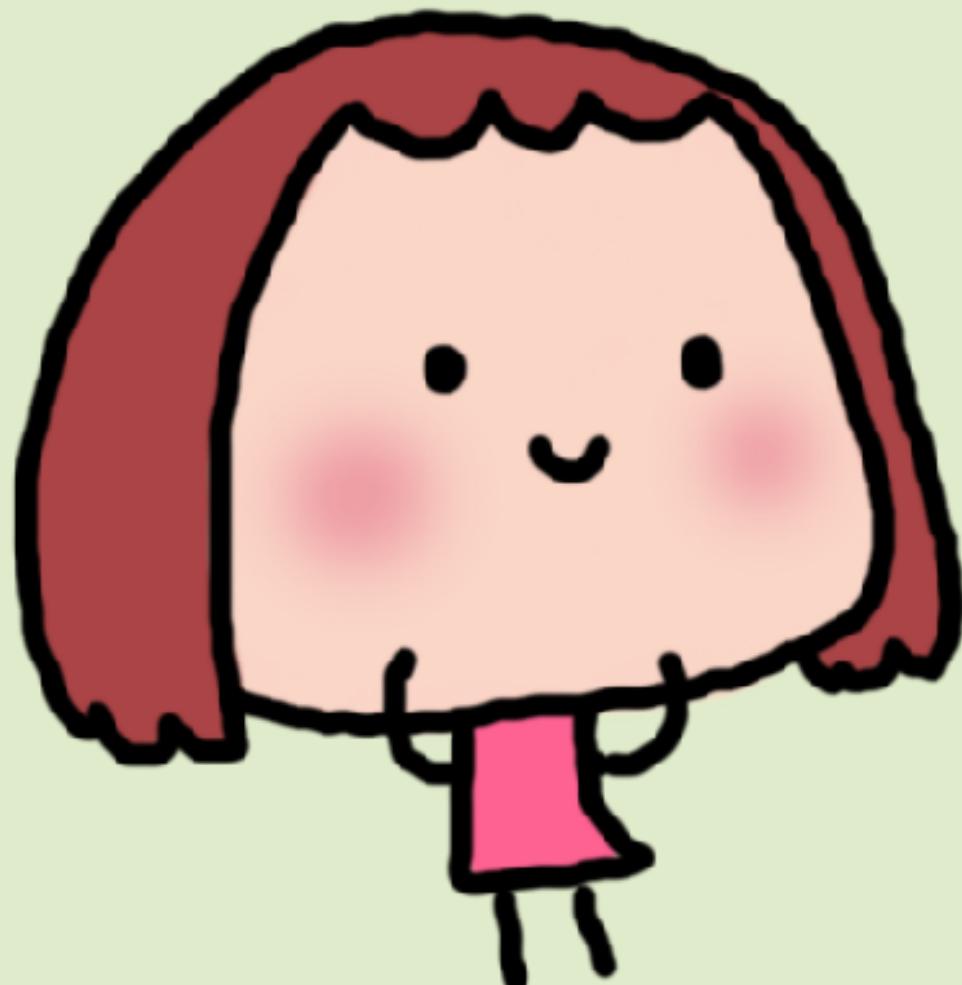
## 重點

# array 過濾器

我們可以用另一個 array, 把要的標 **True**, 不要的標 **False**。

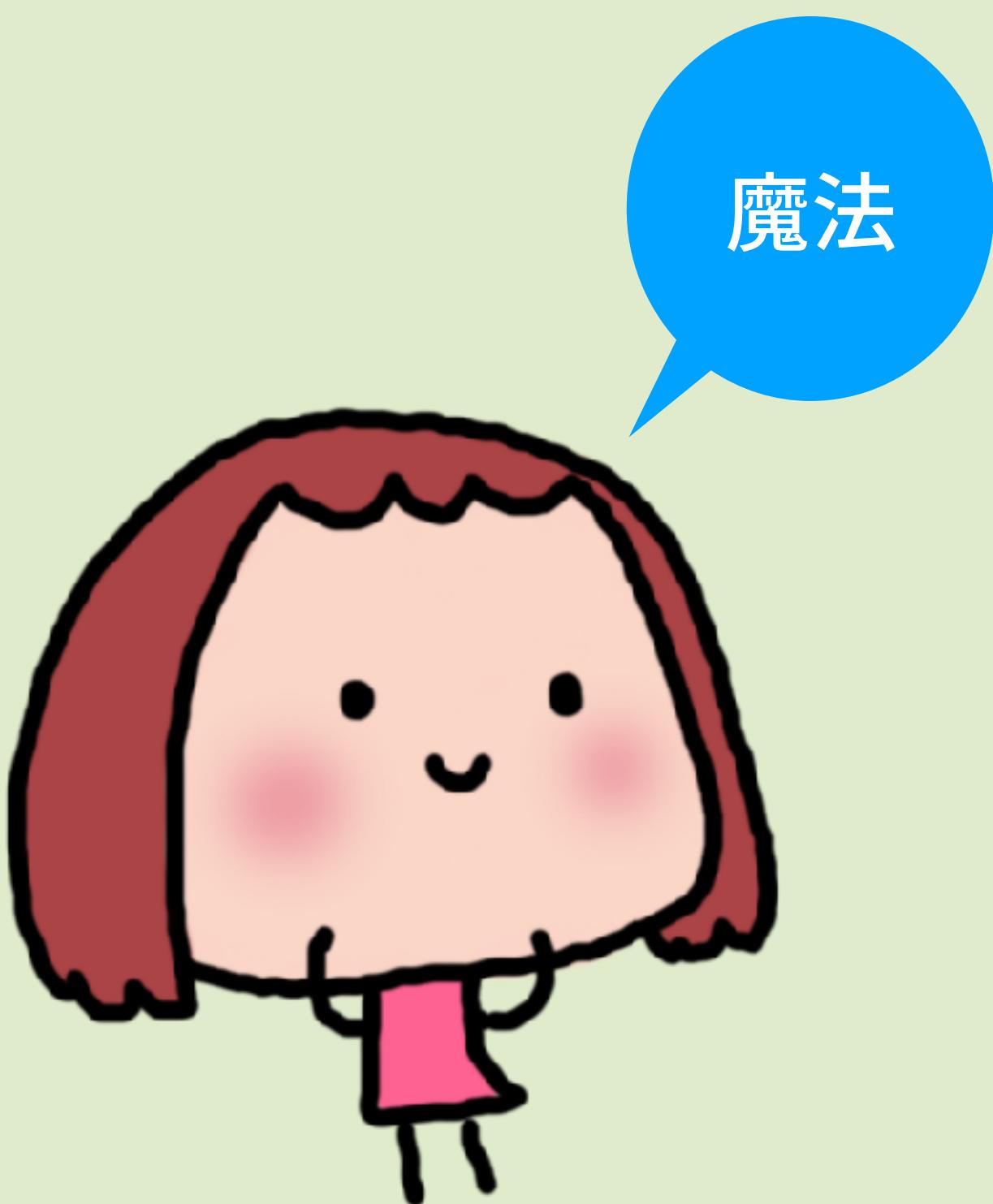
```
L = np.array([3, -2, -1, 5, 7, -3])
```

```
bool = np.array([True, False, False,  
True, True, False])
```



重點

## array 過濾器



然後魔法出現了!

```
L = np.array([3, -2, -1, 5, 7, -3])
```

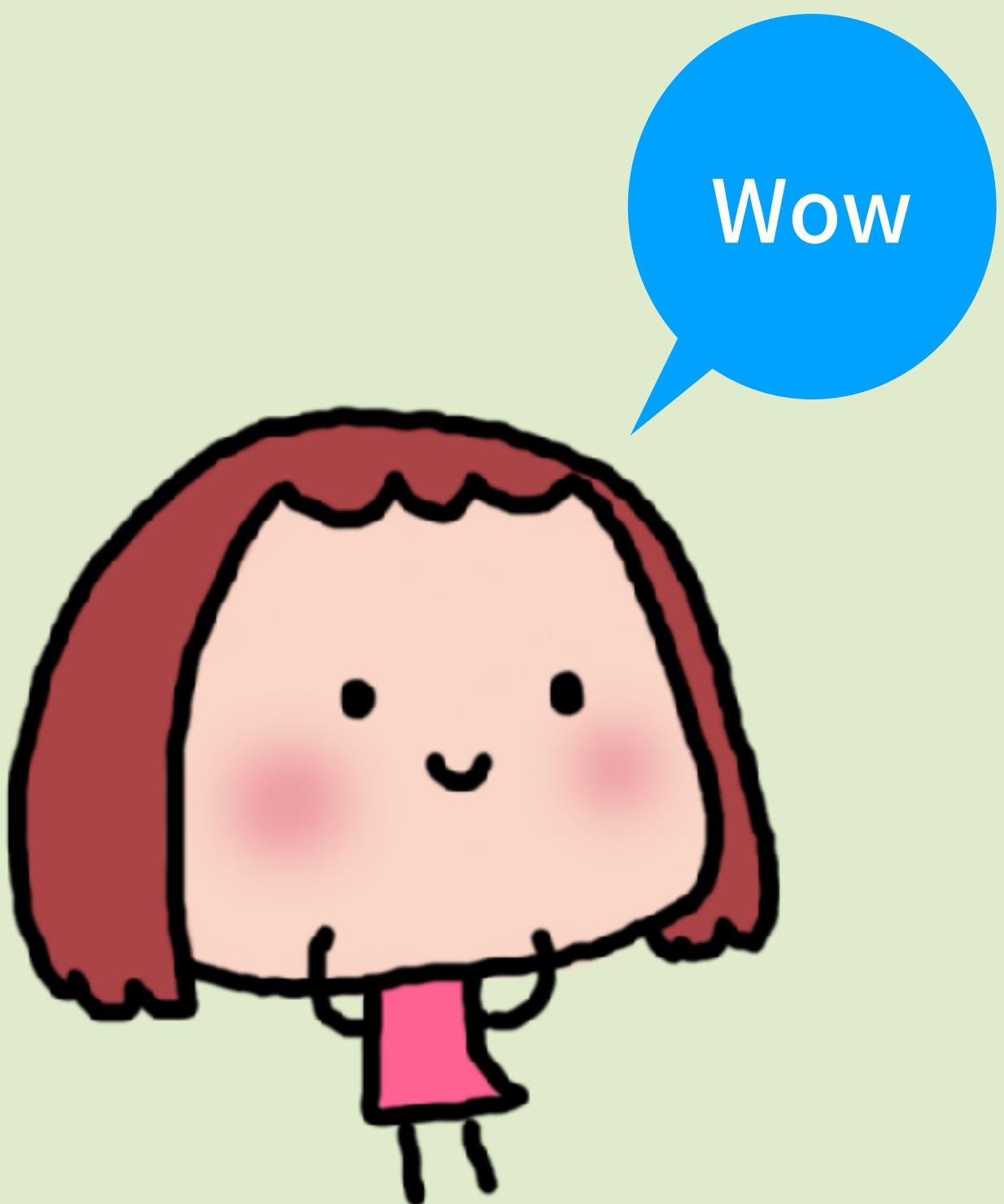
```
L[bool]
```

輸出: array([3, 5, 7])

但是 bool 那串是我們自己打的, 實在有點遜。

## 重點

# array 過濾器



其實我們可以這樣找出 `bool` 那個 `array`。

```
L = np.array([3, -2, -1, 5, 7, -3])
```

```
L>0
```

輸出: `np.array([True, False, False, True, True, False])`

重點

## array 過濾器



更過份的是這樣!

```
L = np.array([3, -2, -1, 5, 7, -3])
```

```
L[L>0]
```

輸出: array([3, 5, 7])

## 重點

# slicing

NumPy 的 slicing 和 list 的切法基本上是一樣的!

```
x = np.arange(10)  
x[2:5]
```

輸出: array([2, 3, 4])

x  
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])



## 重點

# slicing

2 維陣列就抓住先列, 後行的精神!

```
x.shape = (2,5)  
x[:,1:3]
```

輸出:

```
array([[1, 2],  
       [6, 7]])
```

x

```
array([[0, 1, 2, 3, 4],  
       [5, 6, 7, 8, 9]])
```

```
x[1, 1:3]
```

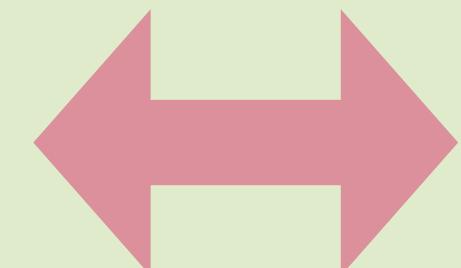
輸出: array([6, 7])

## 重點

# NumPy 的 zip 和 unzip

如同之前的 zip 和 unzip, 我們常要做這樣的格式互換...

```
x = np.array([1,2,3,4])  
y = np.array([5,6,7,8])
```



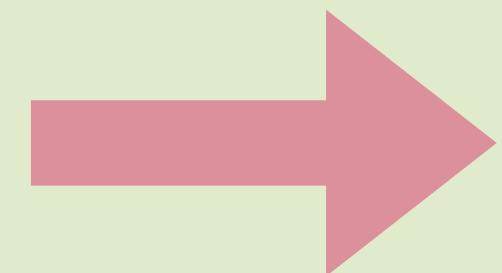
```
x = np.array([[1, 5],  
              [2, 6],  
              [3, 7],  
              [4, 8]])
```

## 重點

# NumPy 的 zip 和 unzip

```
x = np.array([1,2,3,4])  
y = np.array([5,6,7,8])
```

```
x = np.c_[x,y]
```



```
x = np.array([[1, 5],  
             [2, 6],  
             [3, 7],  
             [4, 8]])
```

神秘!



## 重點

# NumPy 的 zip 和 unzip

```
x = np.array([1,2,3,4])  
y = np.array([5,6,7,8])
```

```
x = x[:, 0]  
y = x[:, 1]
```

```
x = np.array([[1, 5],  
              [2, 6],  
              [3, 7],  
              [4, 8]])
```

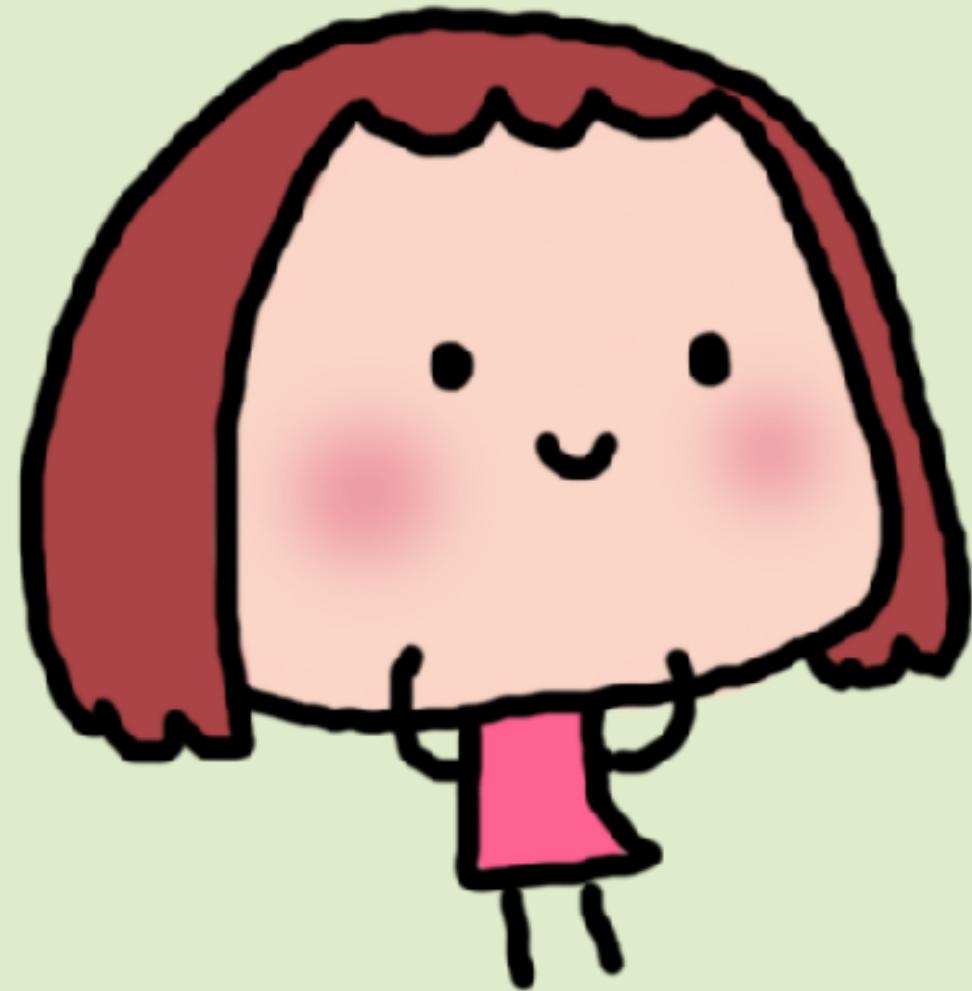
簡單!



3

# Matplotlib

視覺化的基本功



原作者

# John Hunter

- 芝加哥大學神經生物學博士
- 對於 2004 年 NASA 用 matplotlib 確保探測車安全登陸火星特別感到驕傲
- 2012 年在大腸癌治療過程中過逝, 年僅

44 歲



John Hunter 在 SciPy 2012

<https://youtu.be/e31Tby5RI54>

## 基本功

# 畫個函數的圖形

前面說過 plt.plot 的基本用法是

```
plt.plot(x, y)
```

其中 x, y 是分別是點 x 座標, y 座標形的  
的 list 或 array。

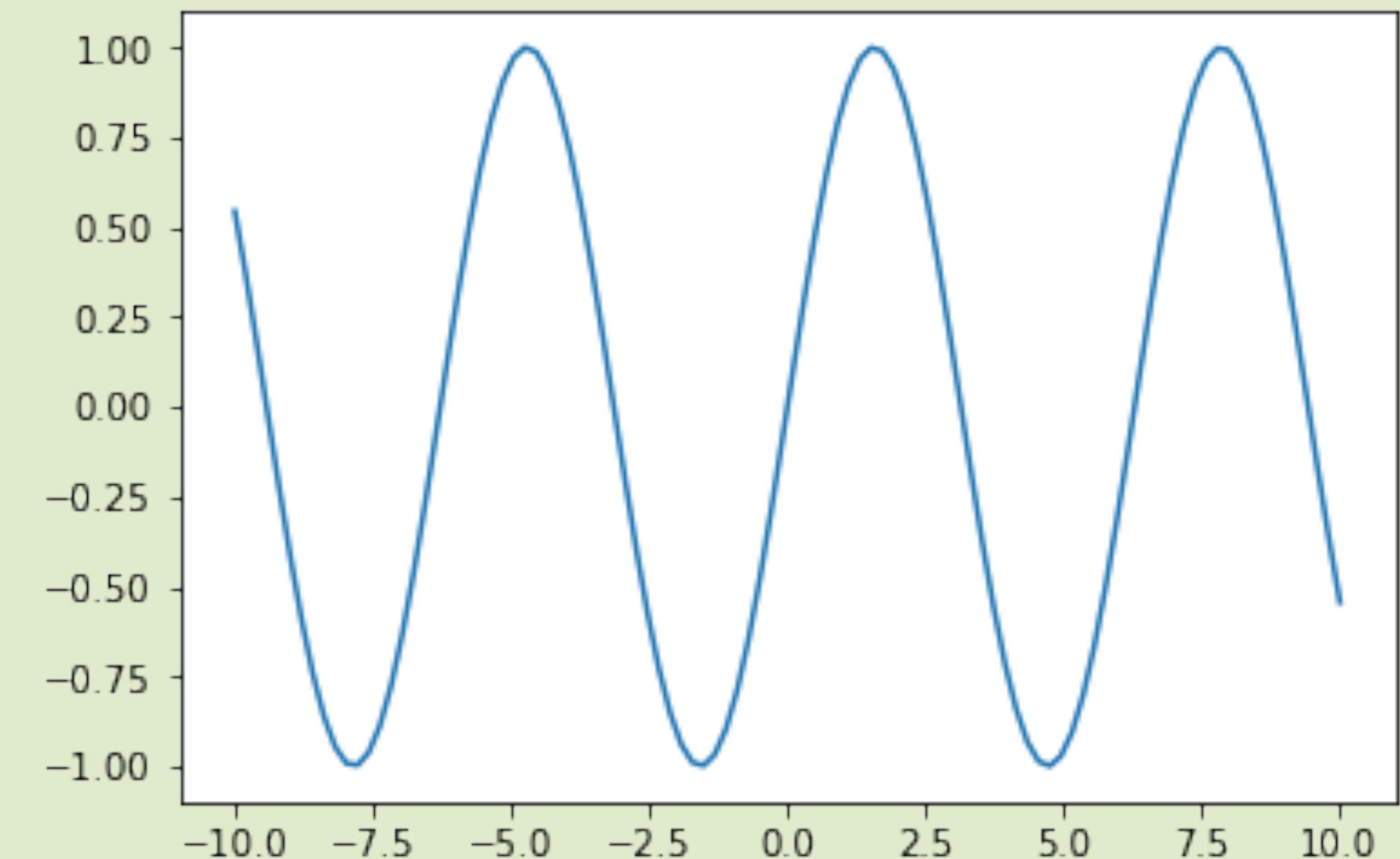


## 基本功

# 畫個函數的圖形

牛刀小試，來畫個  $y = \sin(x)$  的函數  
圖形。

```
x = np.linspace(-10, 10, 100)
y = np.sin(x)
plt.plot(x, y)
```



## 小重點

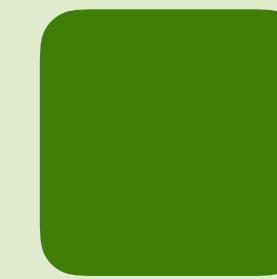
# 快速變色

紅色

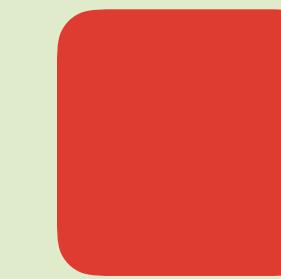
```
plt.plot(x, y, 'r')
```



'b'



'g'



'r'



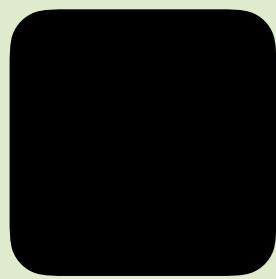
'c'



'm'



'y'



'k'

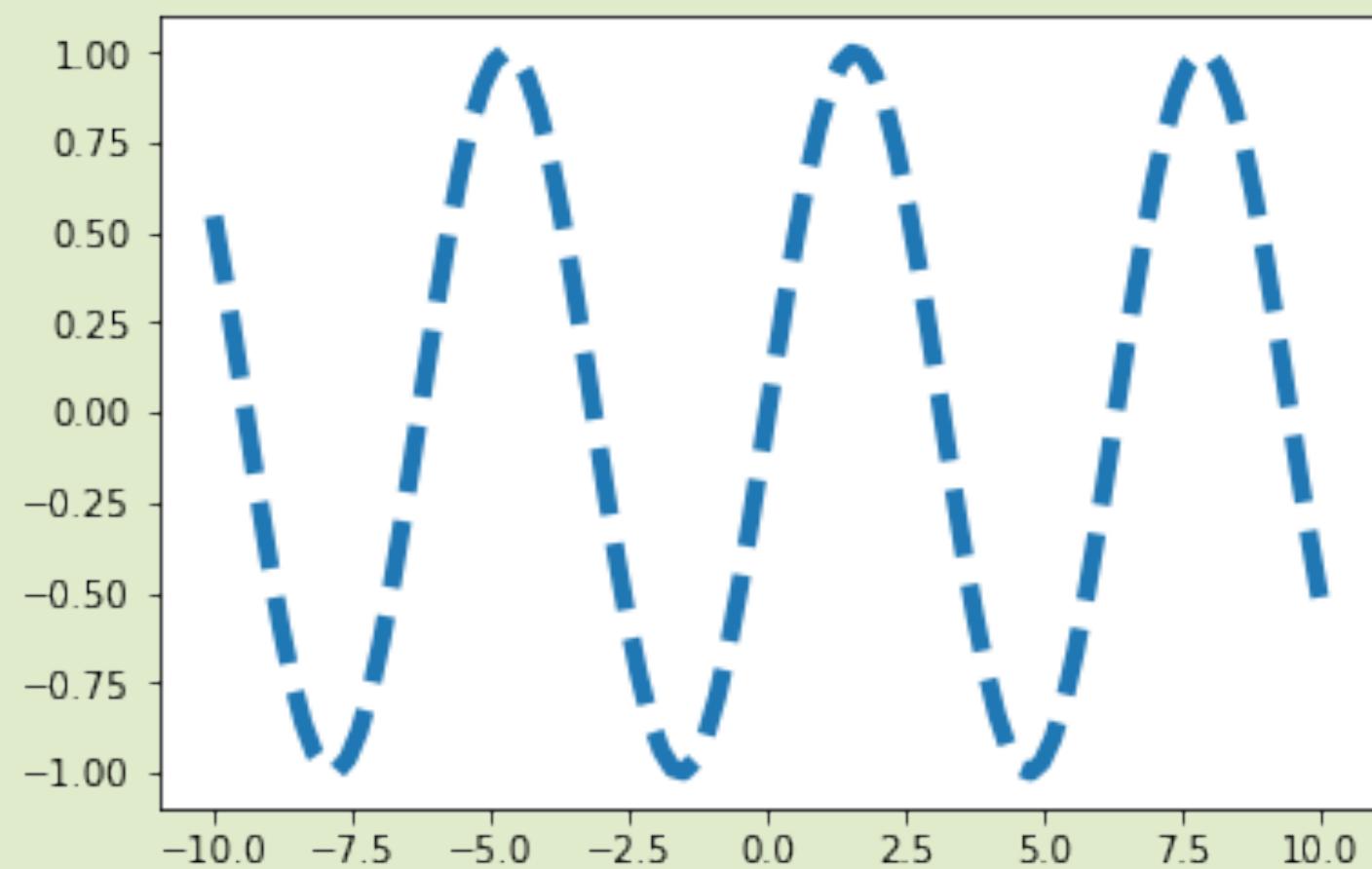


'w'

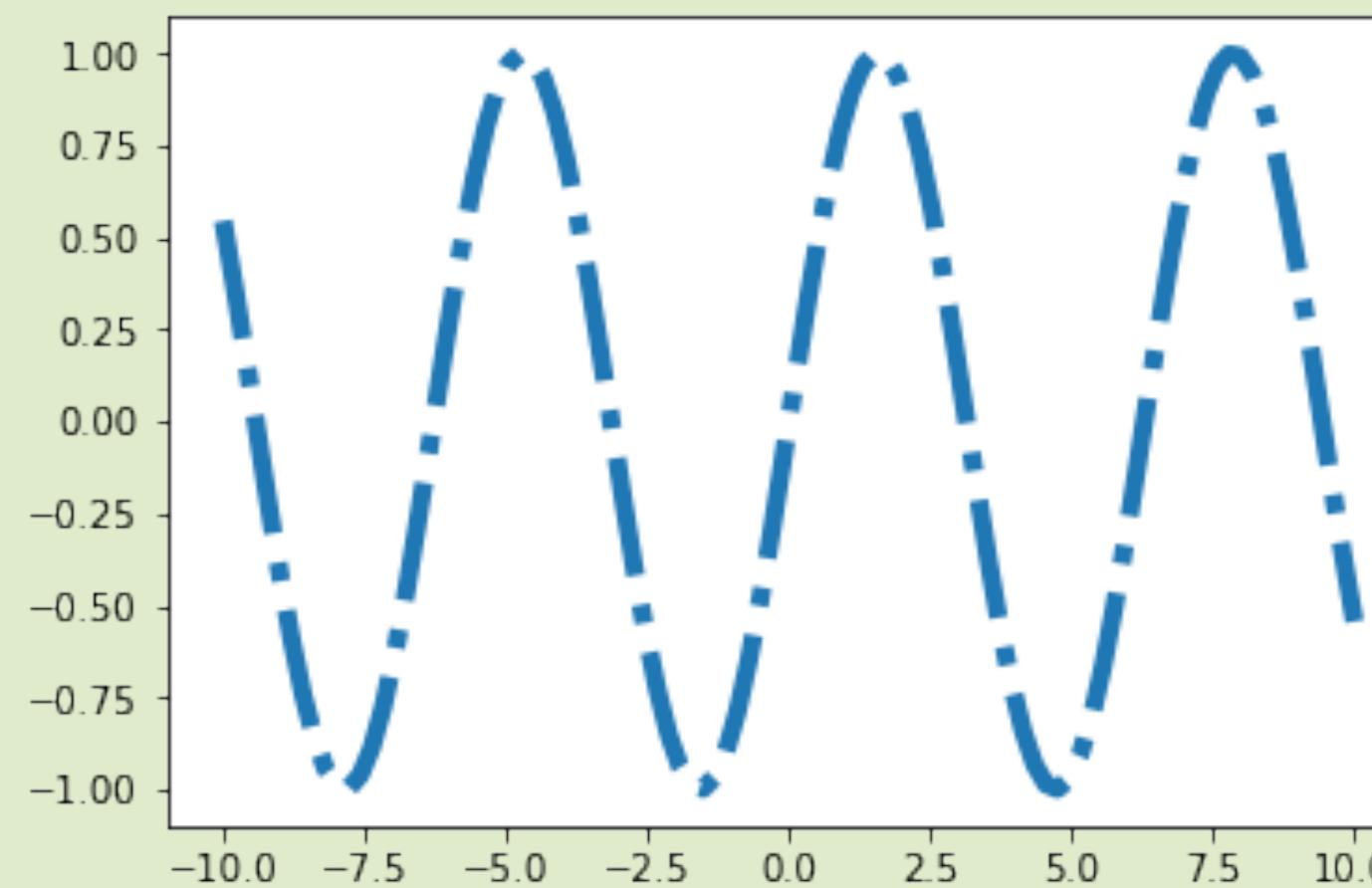
小重點

## 快速線條風格 `linestyle` (ls)

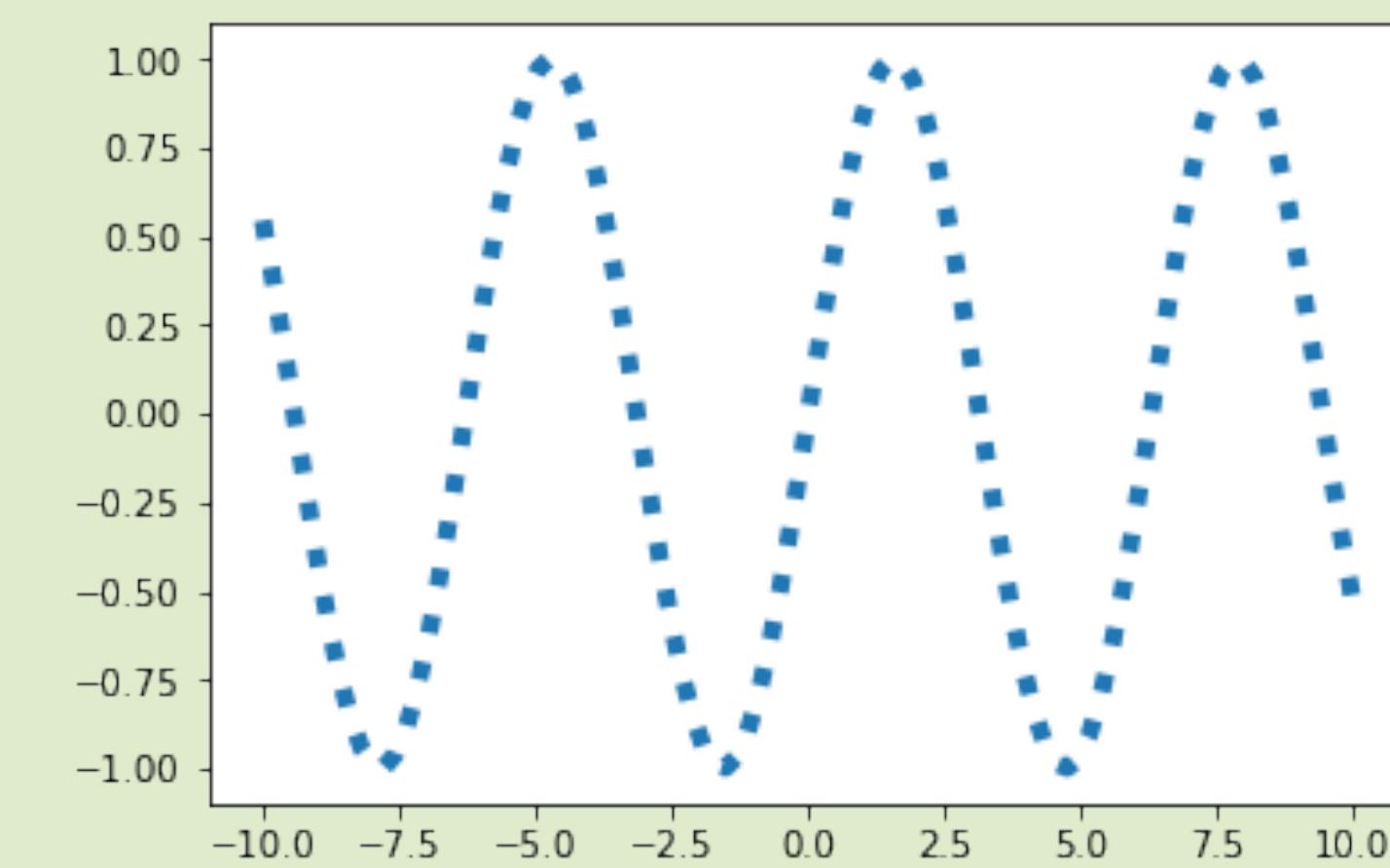
```
plt.plot(x, y, '--')
```



'--'



'-' . '

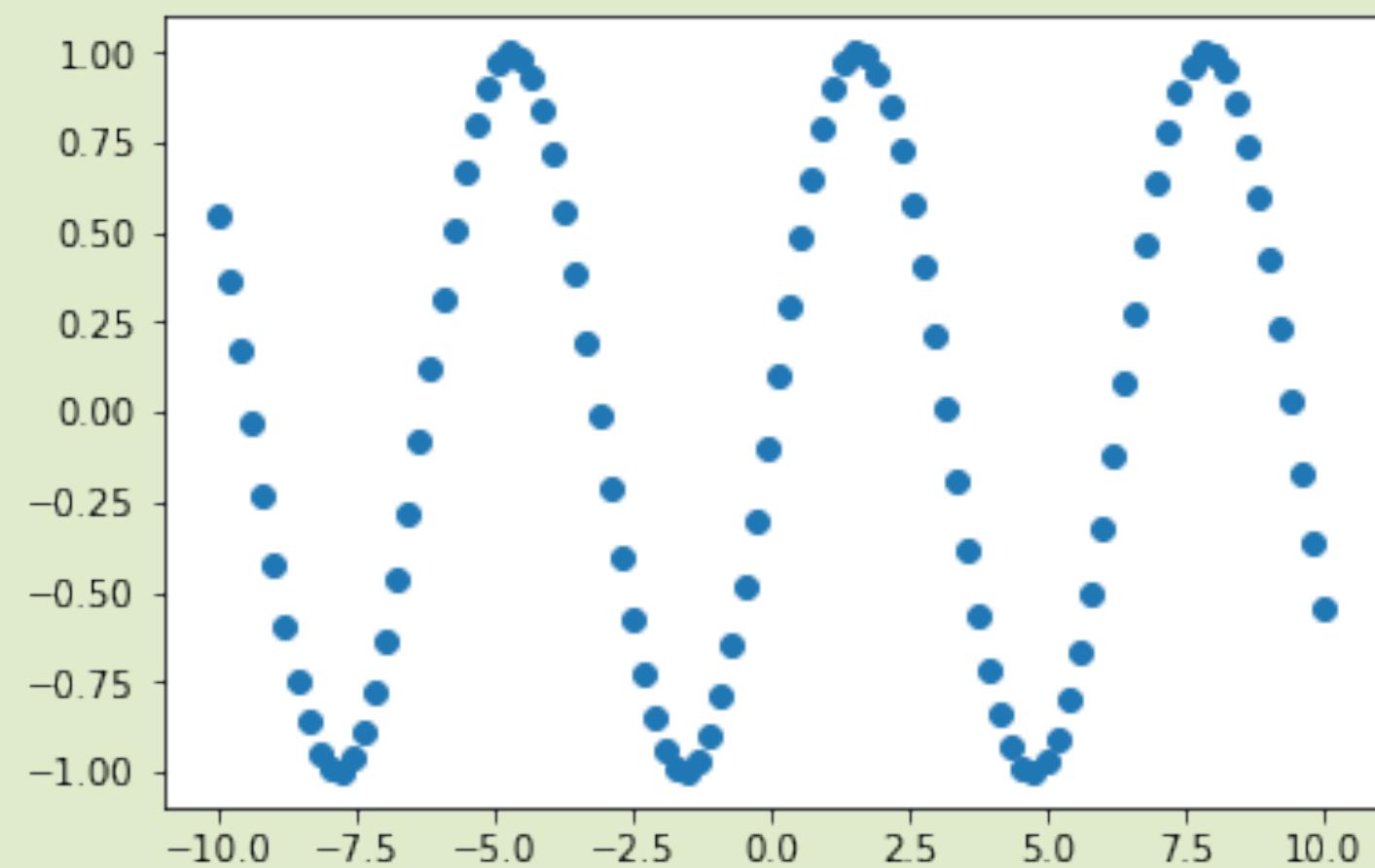


' : '

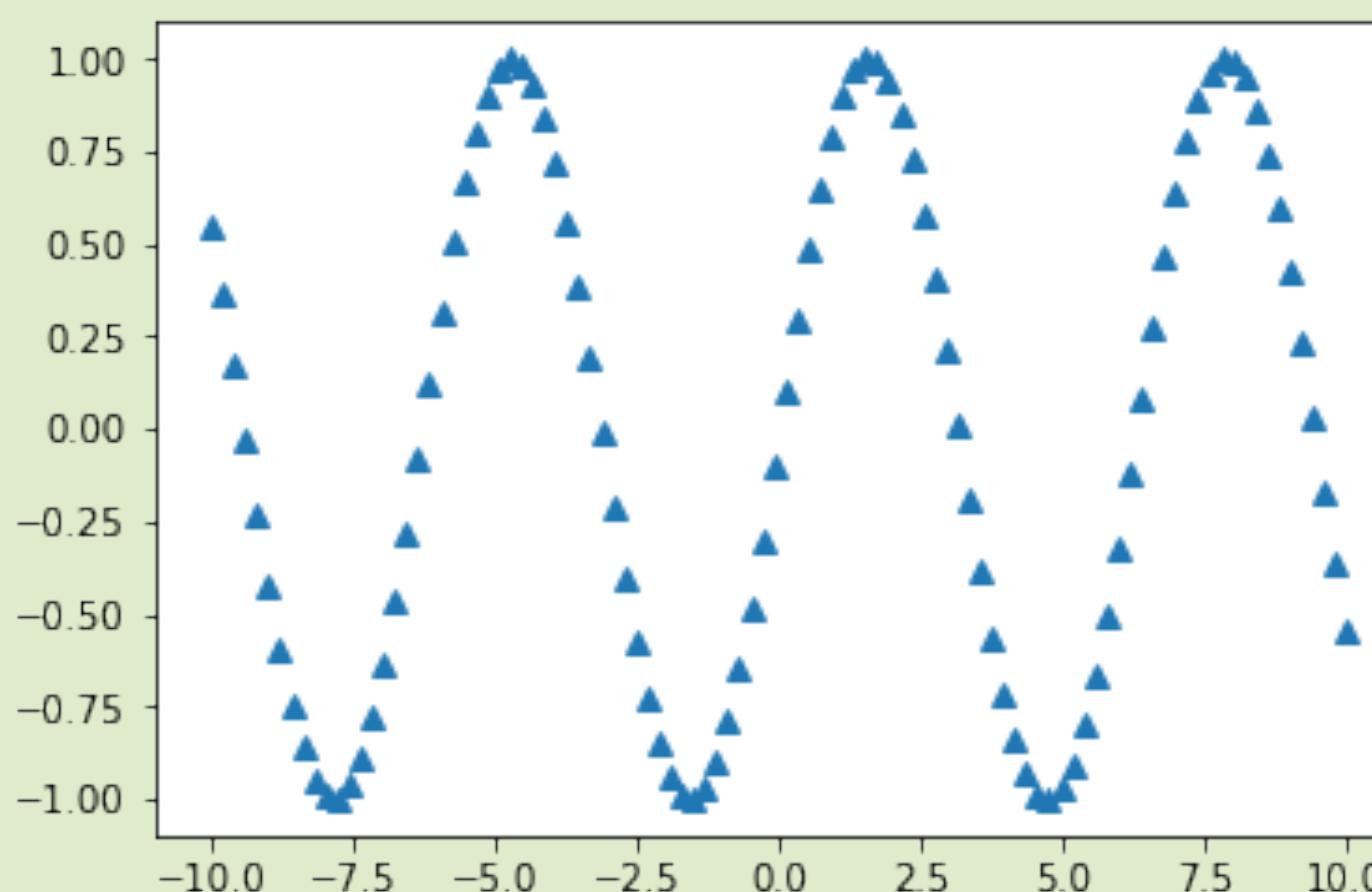
小重點

## 快速線條風格

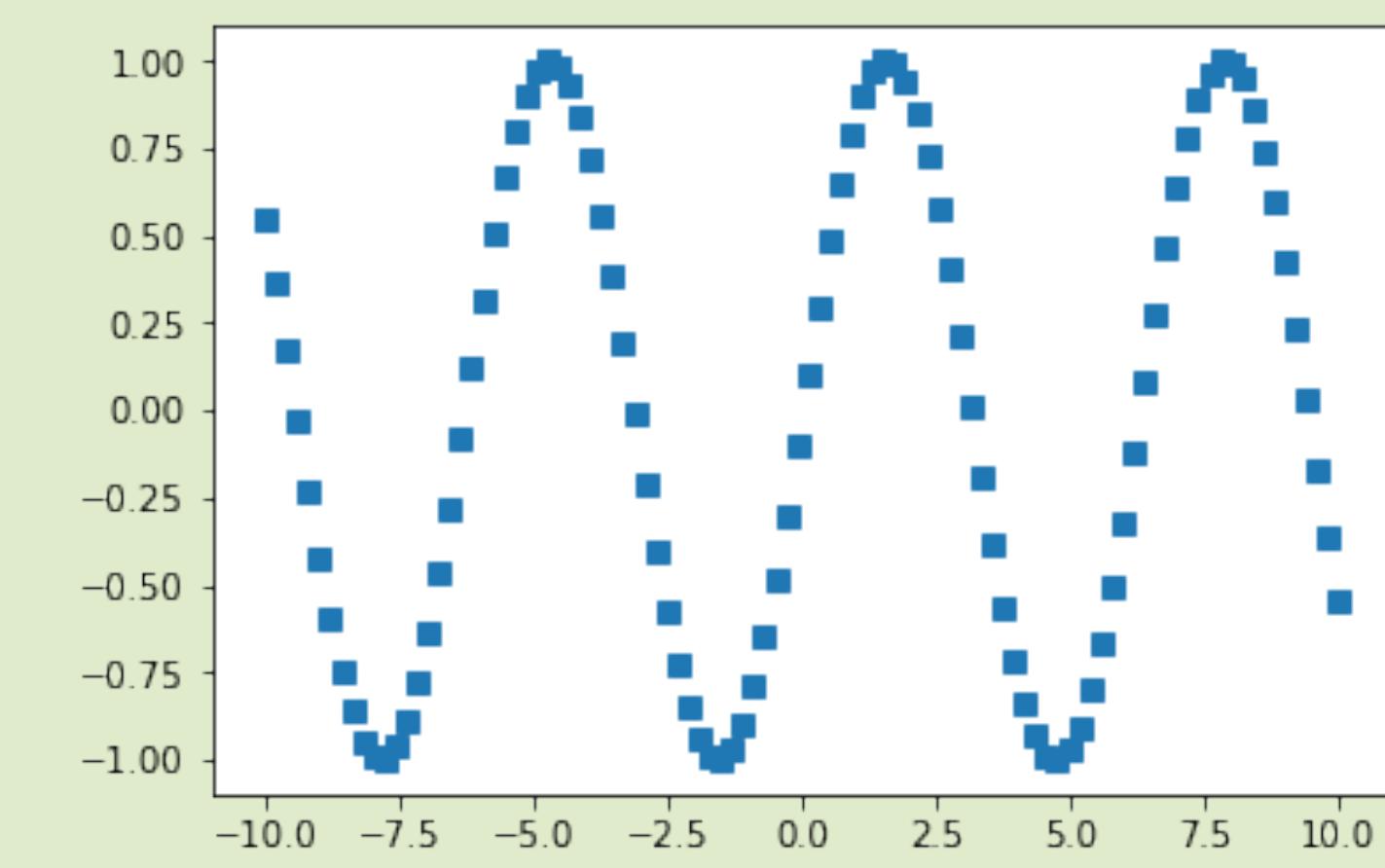
```
plt.plot(x, y, 'o')
```



'o'



'^'



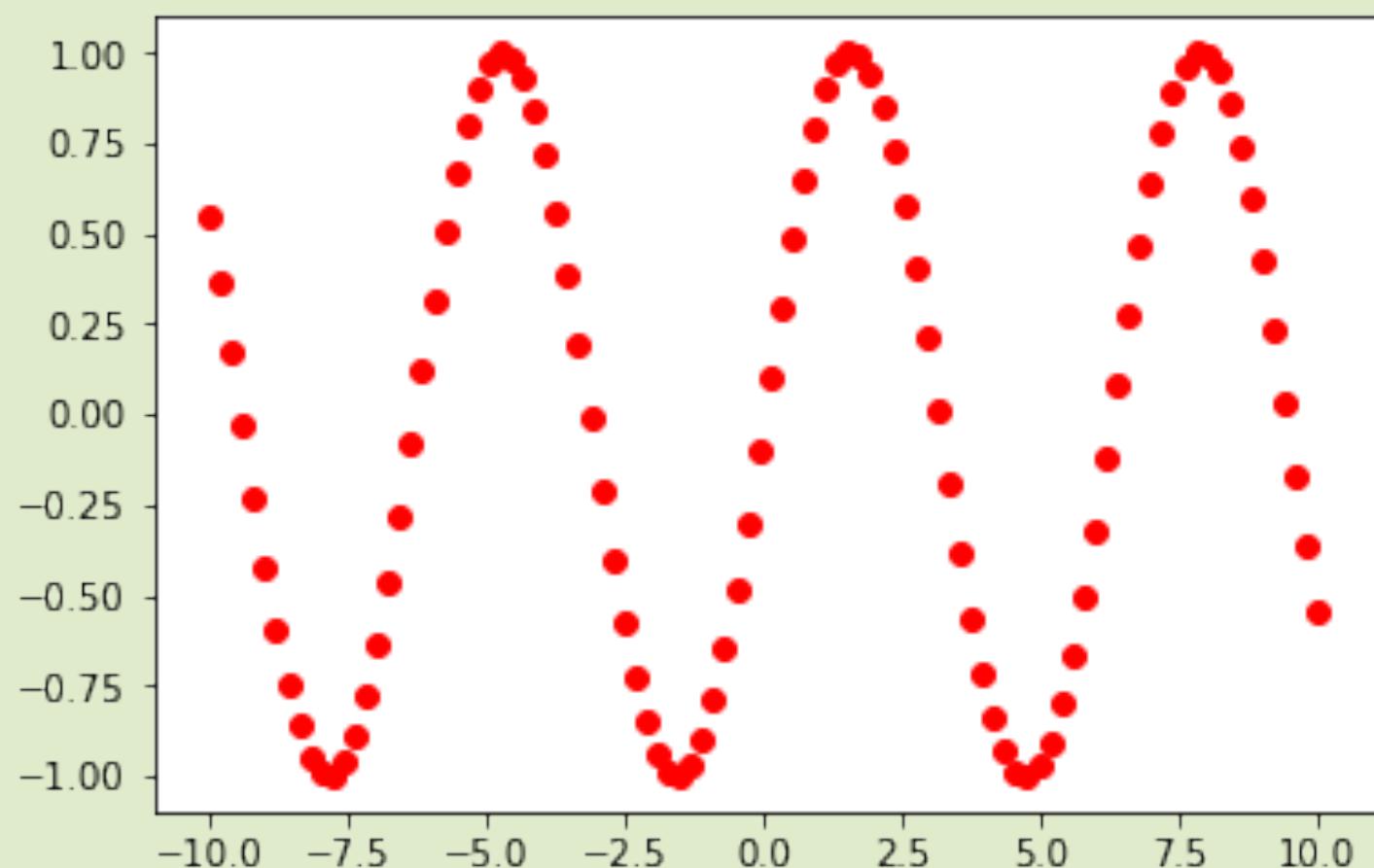
's'

小重點

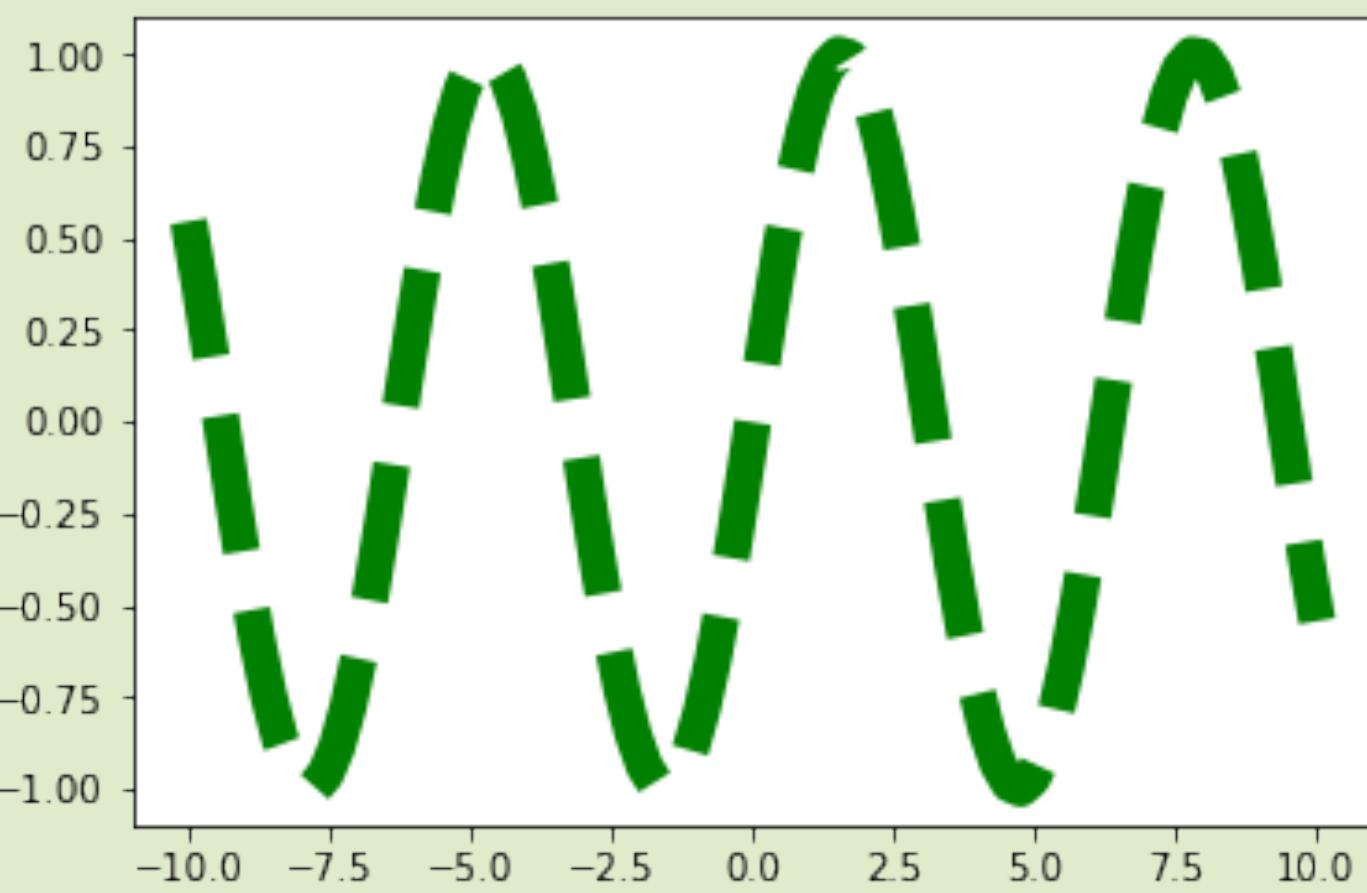
## 快速法可以一起用

這是最常用圖  
的修飾法。

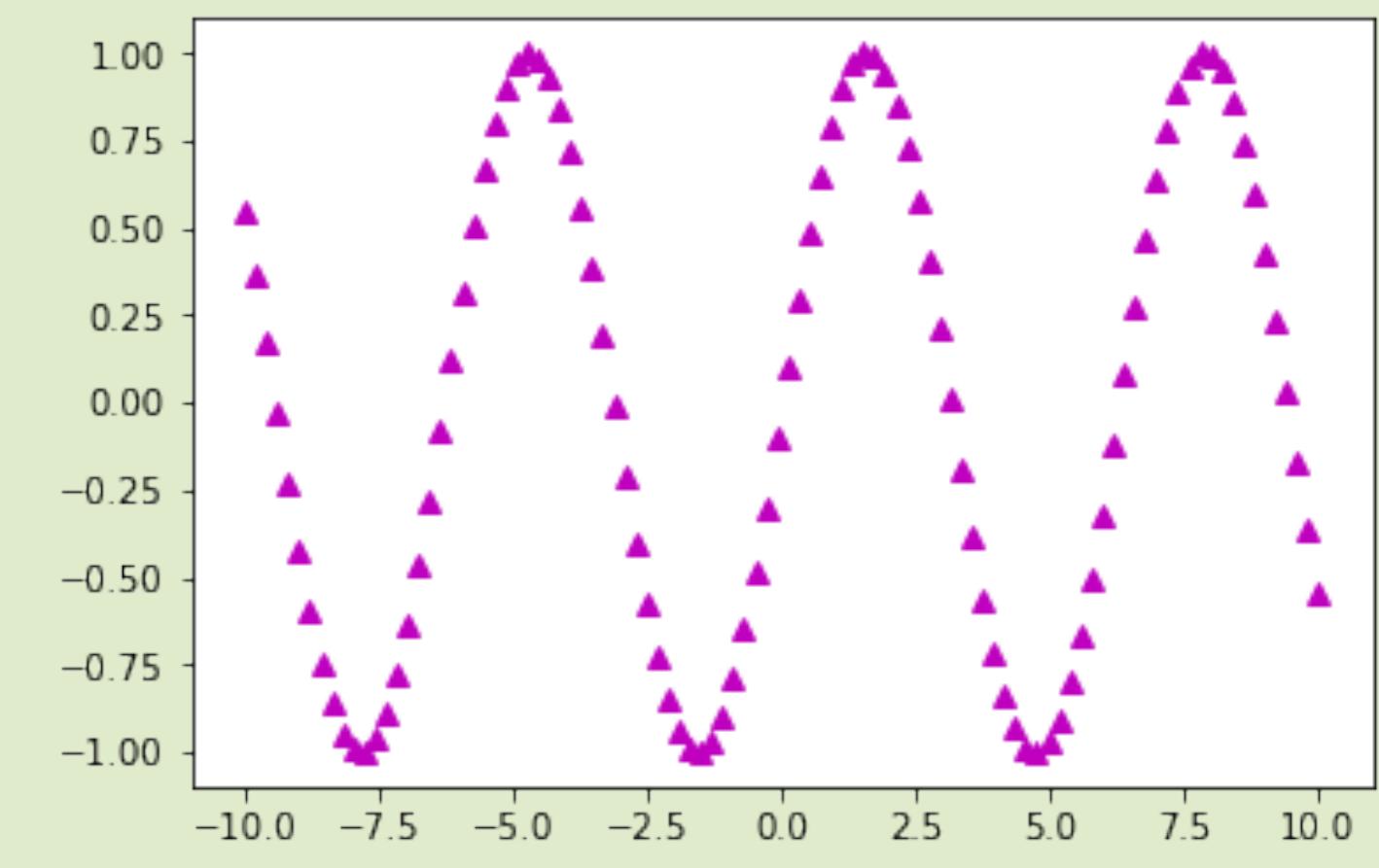
```
plt.plot(x, y, 'ro')
```



'ro'



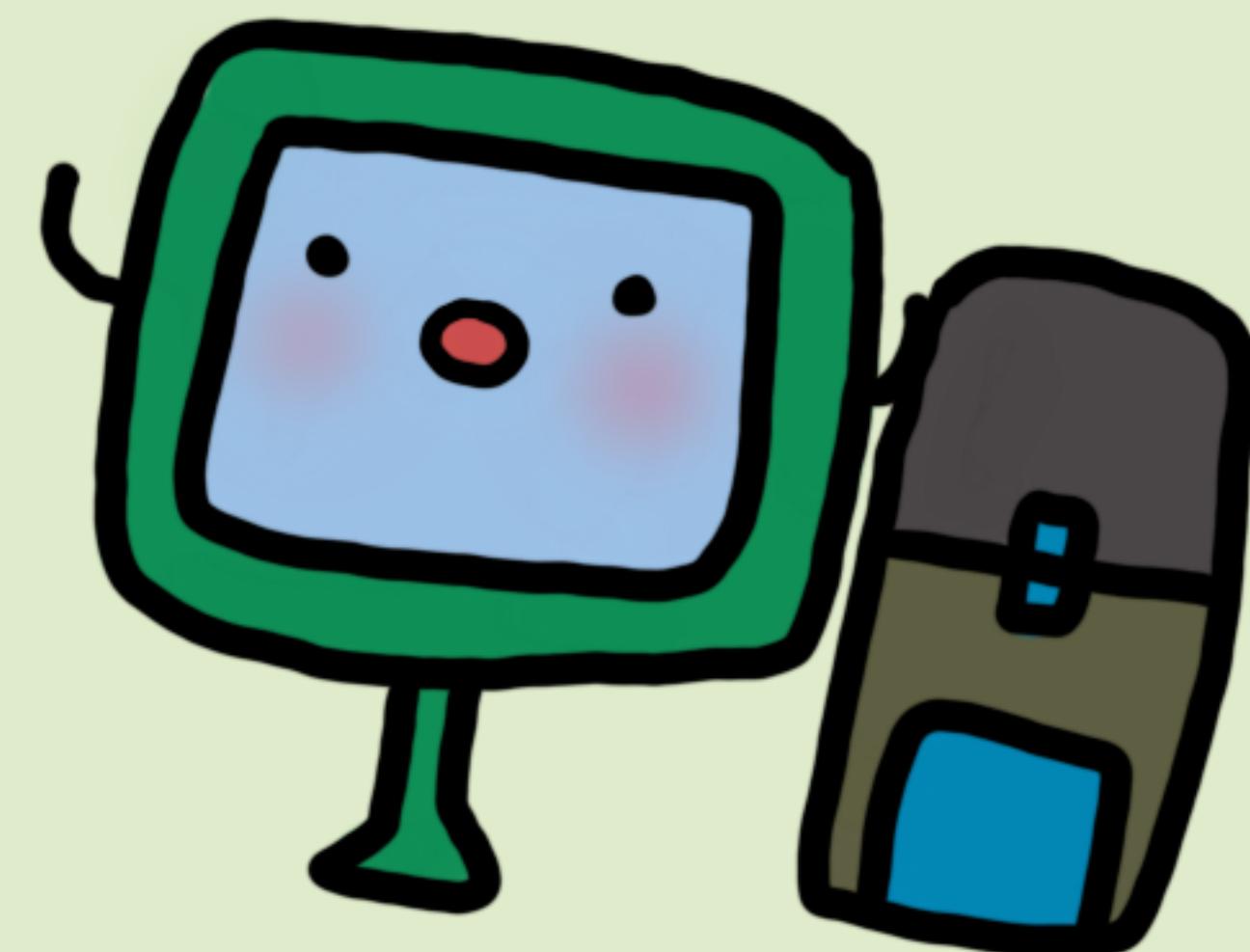
'g--'



'm^'

## 小重點

# Shape 更改法



開始就記這個。

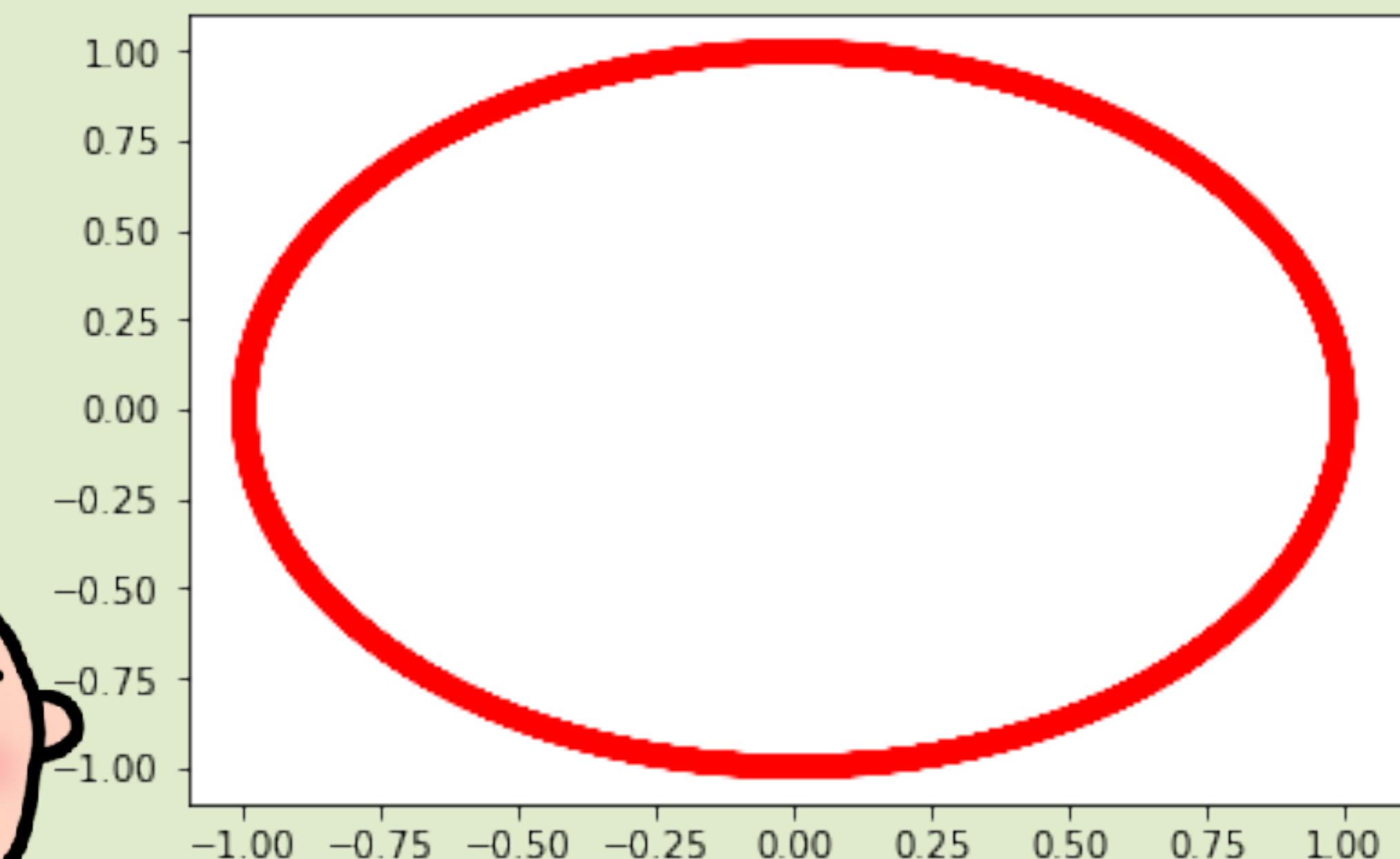
參數	說明
c	指定顏色
lw	指定線寬
ls	指定線的風格
marker , ms	標記點型式和大小

## 小重點

# plt.plot 其實可以畫參數式

```
t = np.linspace(0, 2*np.pi, 300)
x = np.cos(t)
y = np.sin(t)
plt.plot(x, y, 'r', lw=6)
```

這比例很不舒服!



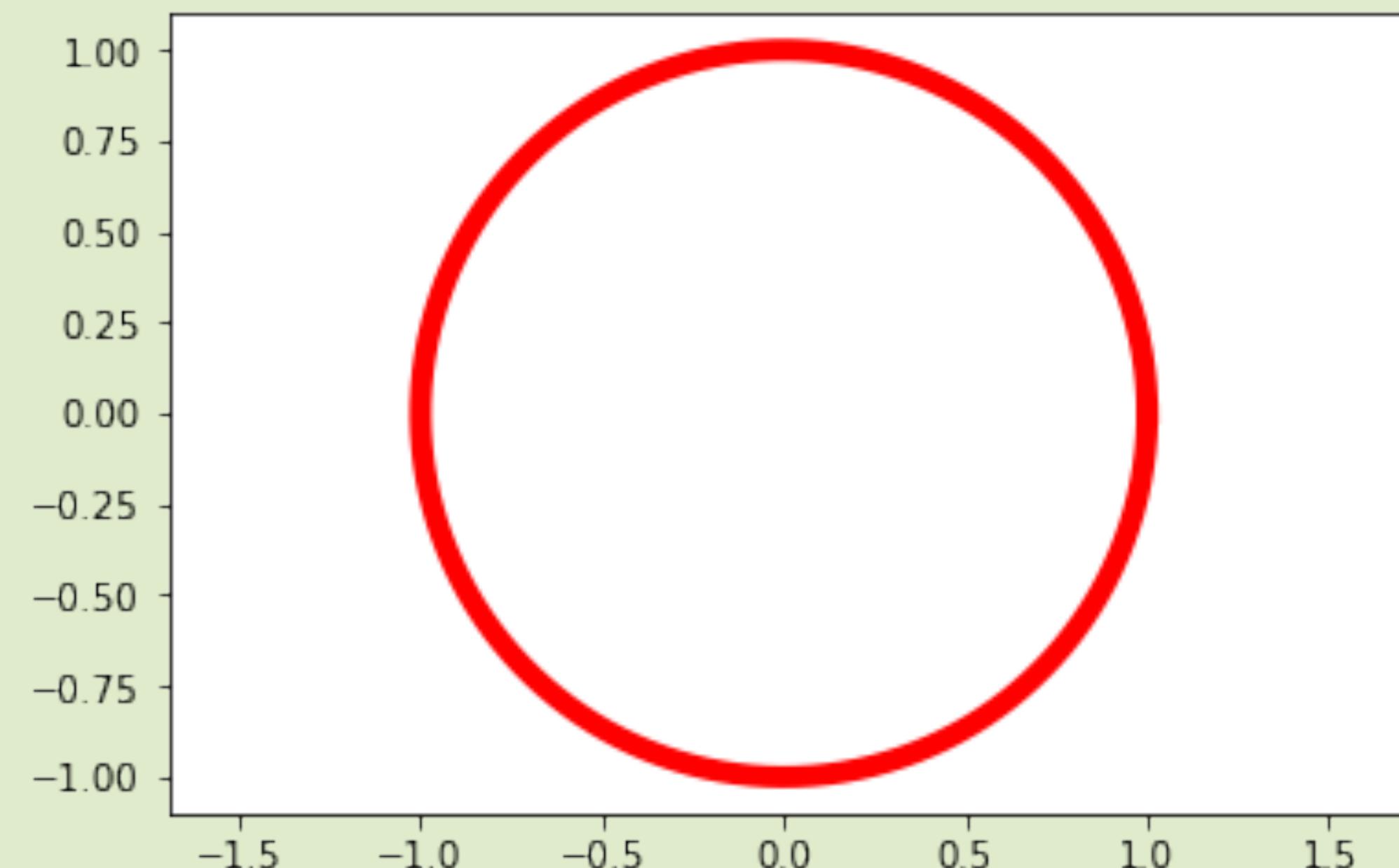
## 小重點

# plt.plot 其實可以畫參數式

只加一行!

```
plt.axis('equal')

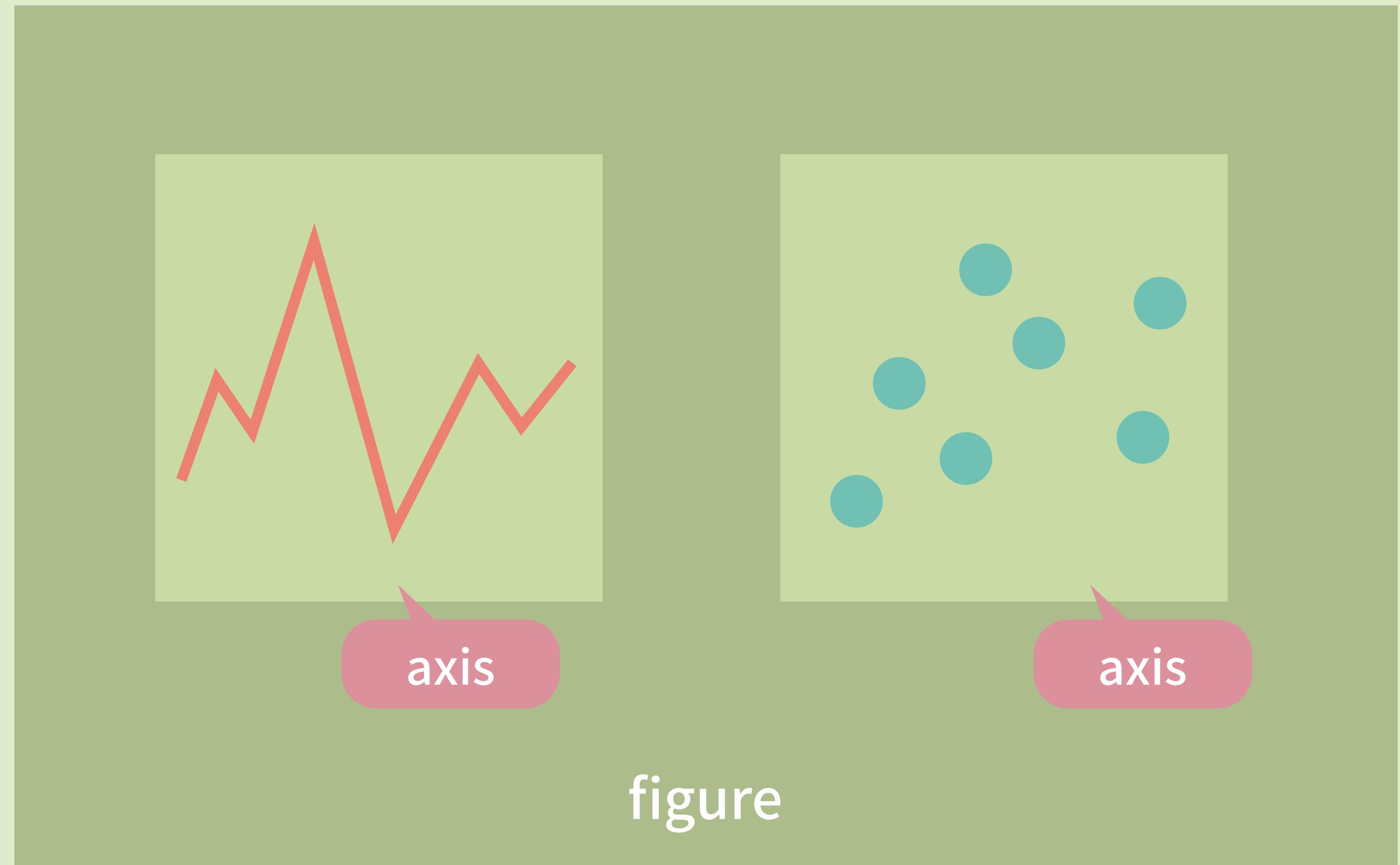
t = np.linspace(0,2*np.pi,300)
x = np.cos(t)
y = np.sin(t)
plt.plot(x, y, 'r', lw=6)
```



```
plt.axis('equal')
```

太神了!  
但這 `axis` 是什麼呢?





在 `matplotlib` 的畫圖中，  
每次都有一個大 `figure`，  
實際畫的圖是在其中的  
`axis` 上。

一個 `figure` 可以有很多  
`axes`。

## 小重點

# 目前所在的 figure 和 axis

```
fig = plt.gcf()  
ax = plt.gca()
```



# `plt.scatter`

和 `plt.plot` 是兩大畫圖指令

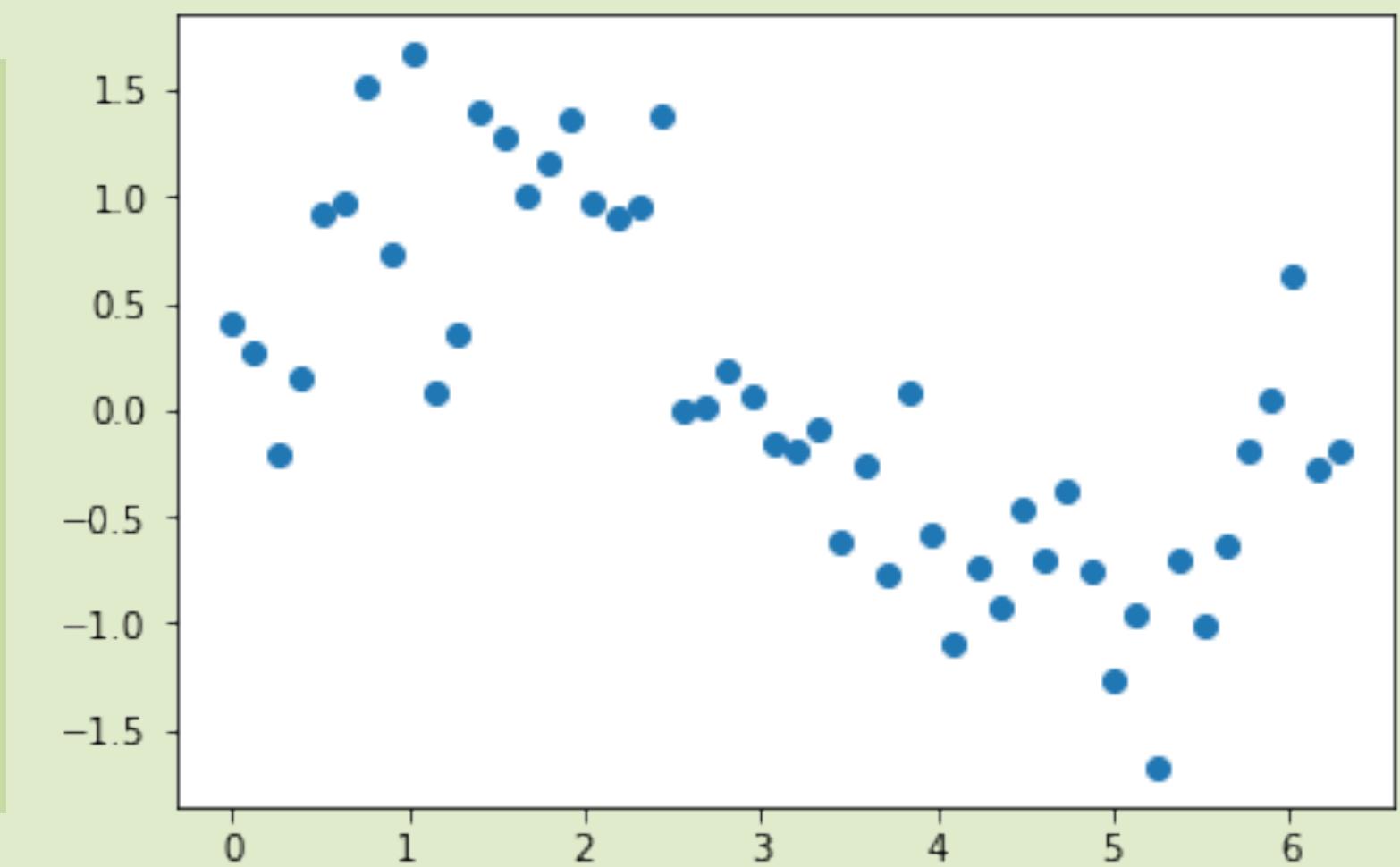
例子

## 其實就是畫點圖

```
x = np.linspace(0, 6.28, 50)  
y = np.sin(x) + 0.4*np.random.randn(50)  
  
plt.scatter(x, y)
```

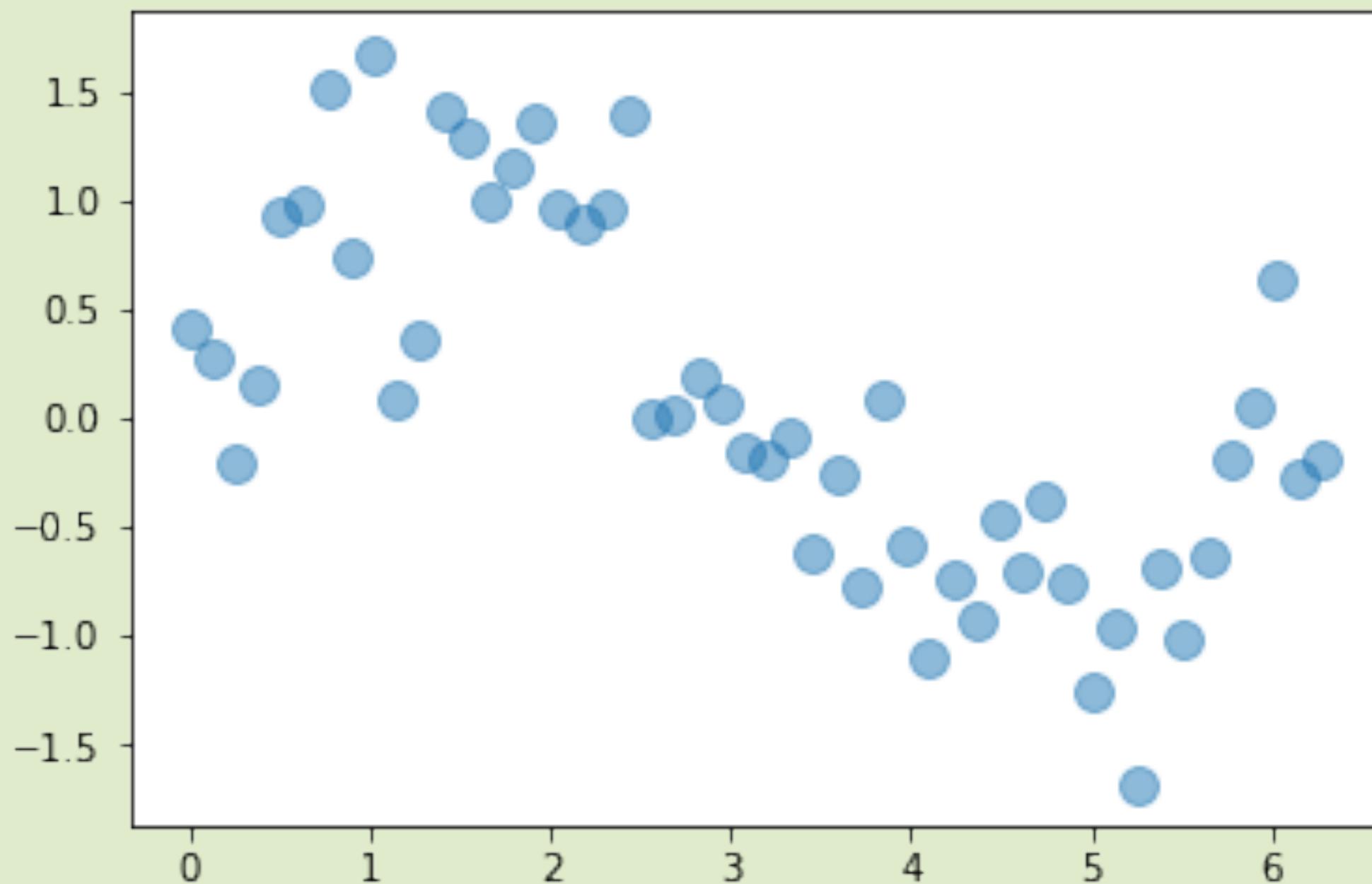


加上一點雜訊。



## 小重點

# 常用參數



```
plt.scatter(x, y, s=100, alpha=0.5)
```

參數	說明
c	指定顏色
s	點的大小 (面積)
alpha	不透明度
marker	點的型式

## 小重點

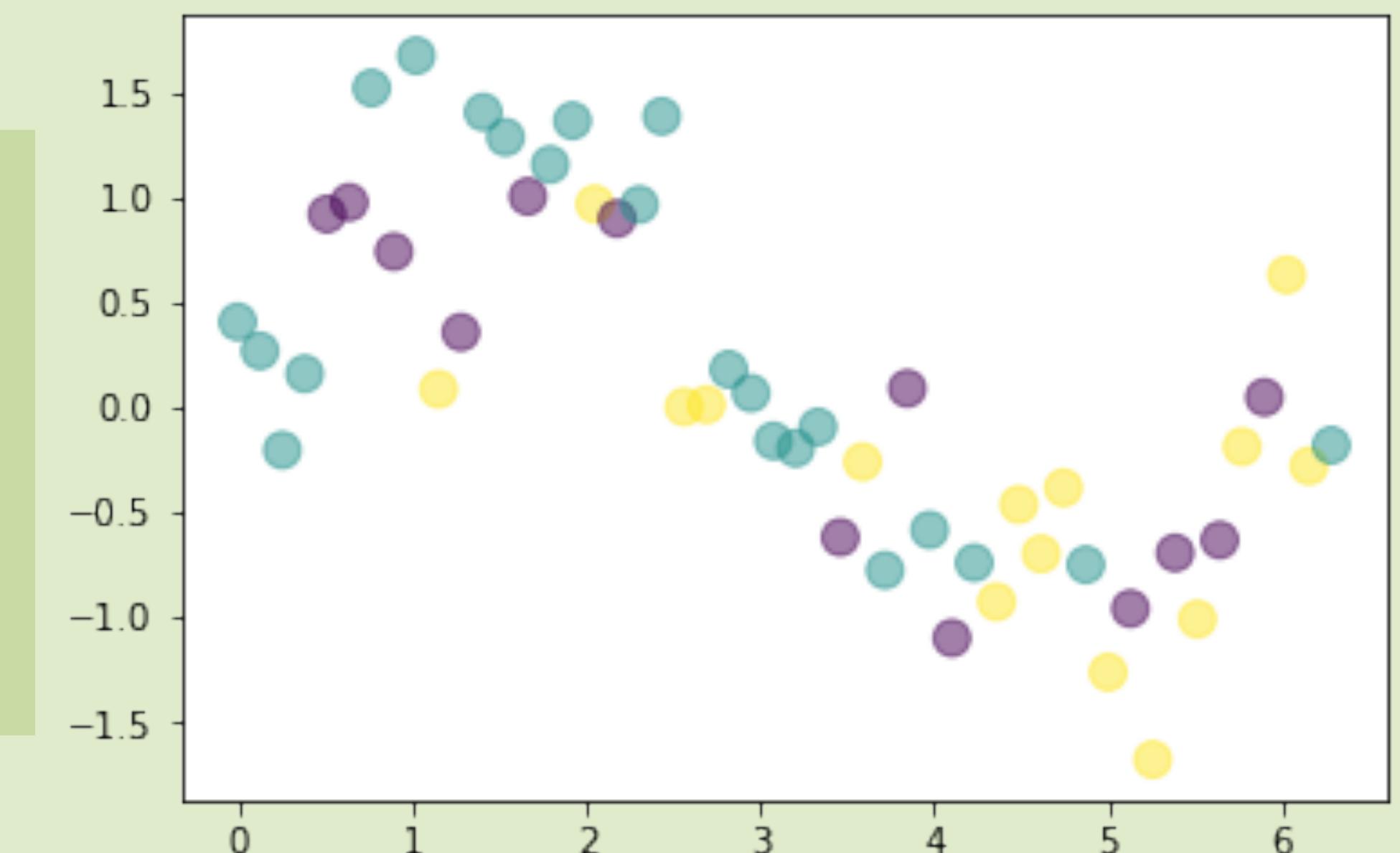
# 有趣的分類標色法

分三類由 `matplotlib` 自己決定用色!

在 0, 1, 2 抽 50 個亂數。

```
c = np.random.randint(0, 3, 50)  
plt.scatter(x, y, s=100, c=c, alpha=0.5)
```

py 人很愛這樣  
令變數。



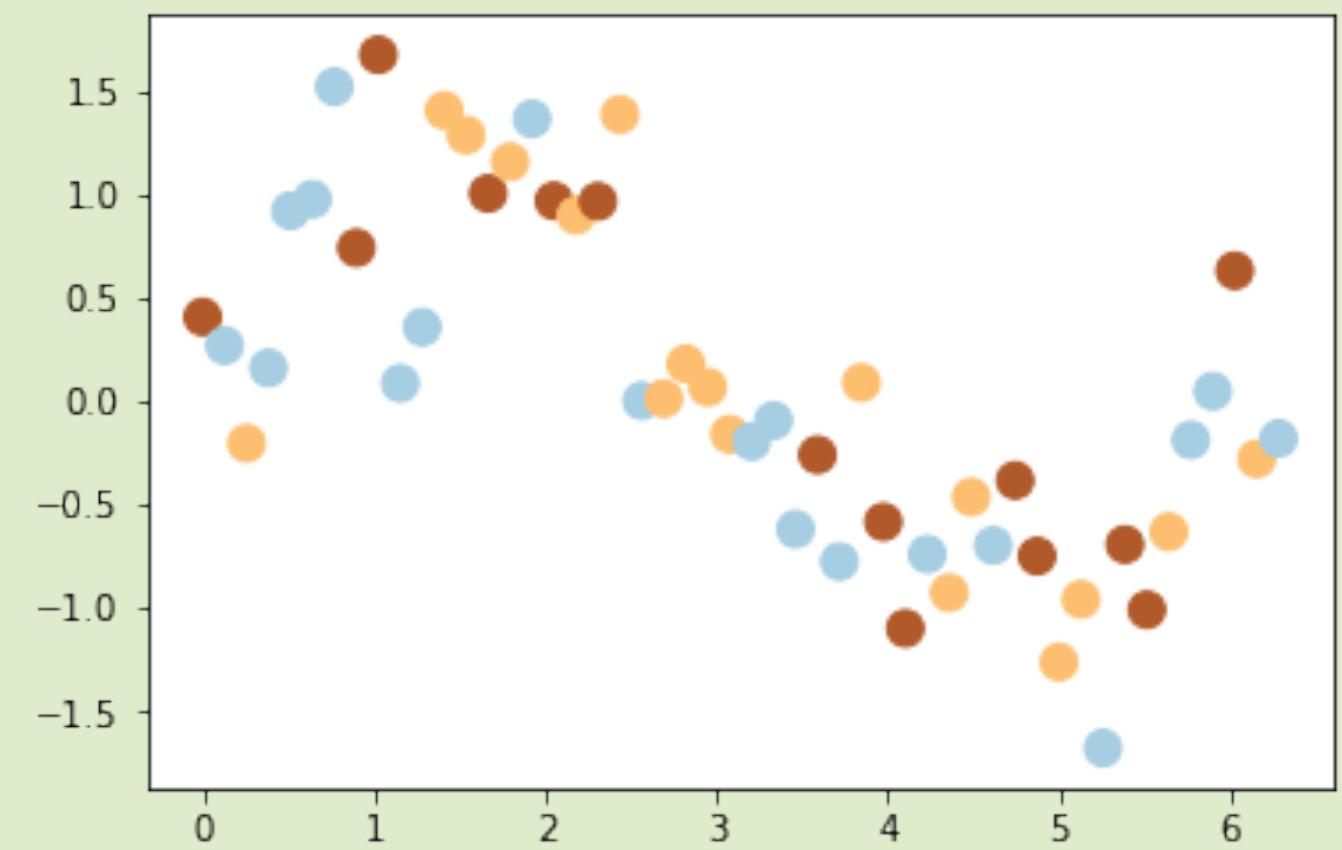
進階

## matplotlib 的顏色主題

剛剛我們看到 matplotlib 自己配色。

不同的配色就是用不同的 color map, 我們要改就是去設定 **cmap**。

```
plt.scatter(x, y, s=100, c=c, cmap="Paired")
```



詳情請參考：

<https://matplotlib.org/users/colormaps.html>

小重點

## 多圖同時呈現

1

2

3

4

5

6

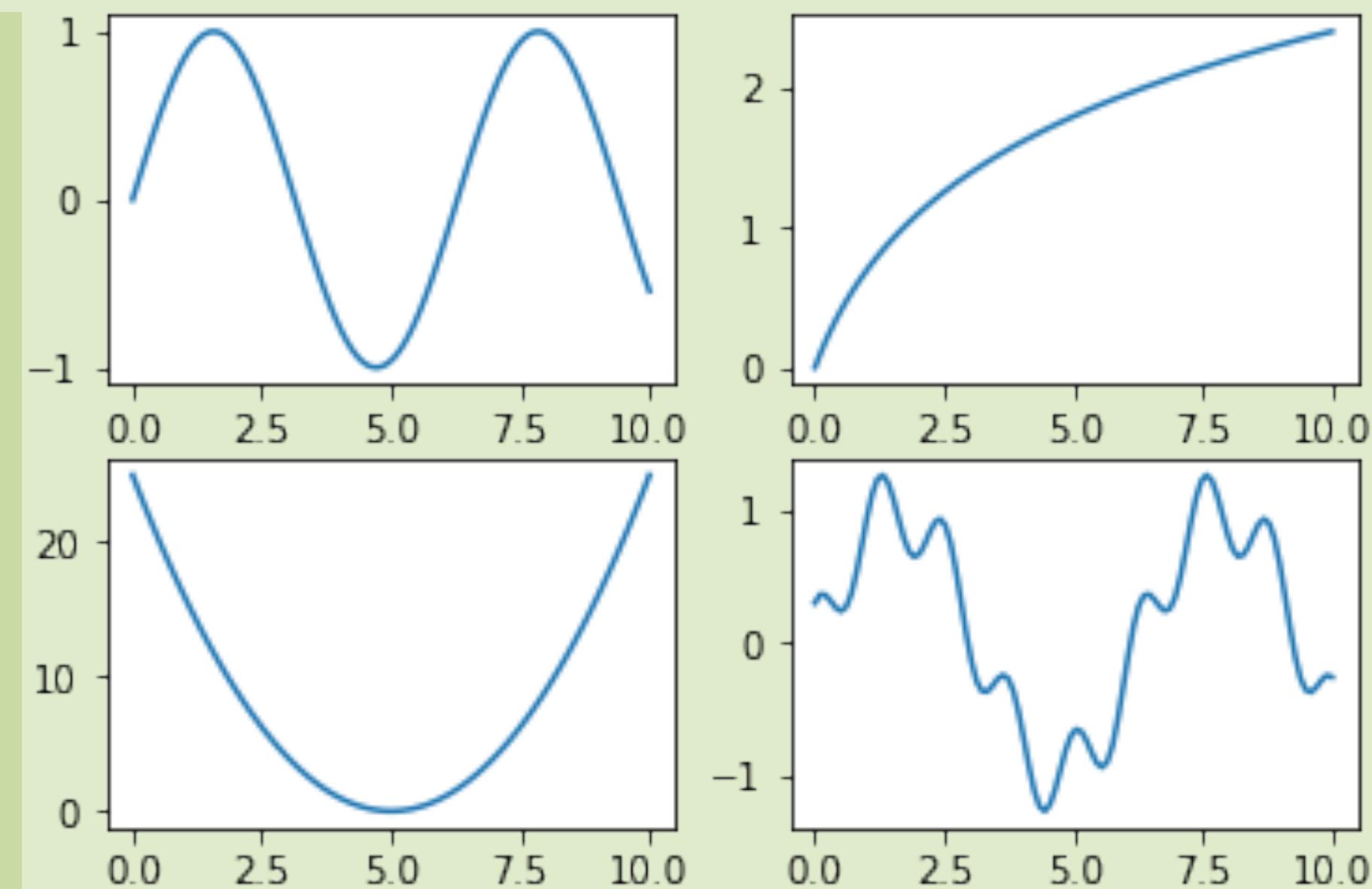
**plt.subplot**  
(列數, 行數, 圖形編號)

## 例子

# 多圖同時呈現

```
x = np.linspace(0,10,100)

plt.subplot(2,2,1)
plt.plot(x, np.sin(x))
plt.subplot(2,2,2)
plt.plot(x, np.log(x+1))
plt.subplot(2,2,3)
plt.plot(x, (x-5)**2)
plt.subplot(2,2,4)
plt.plot(x, np.sin(x)+0.3*np.cos(5*x))
```



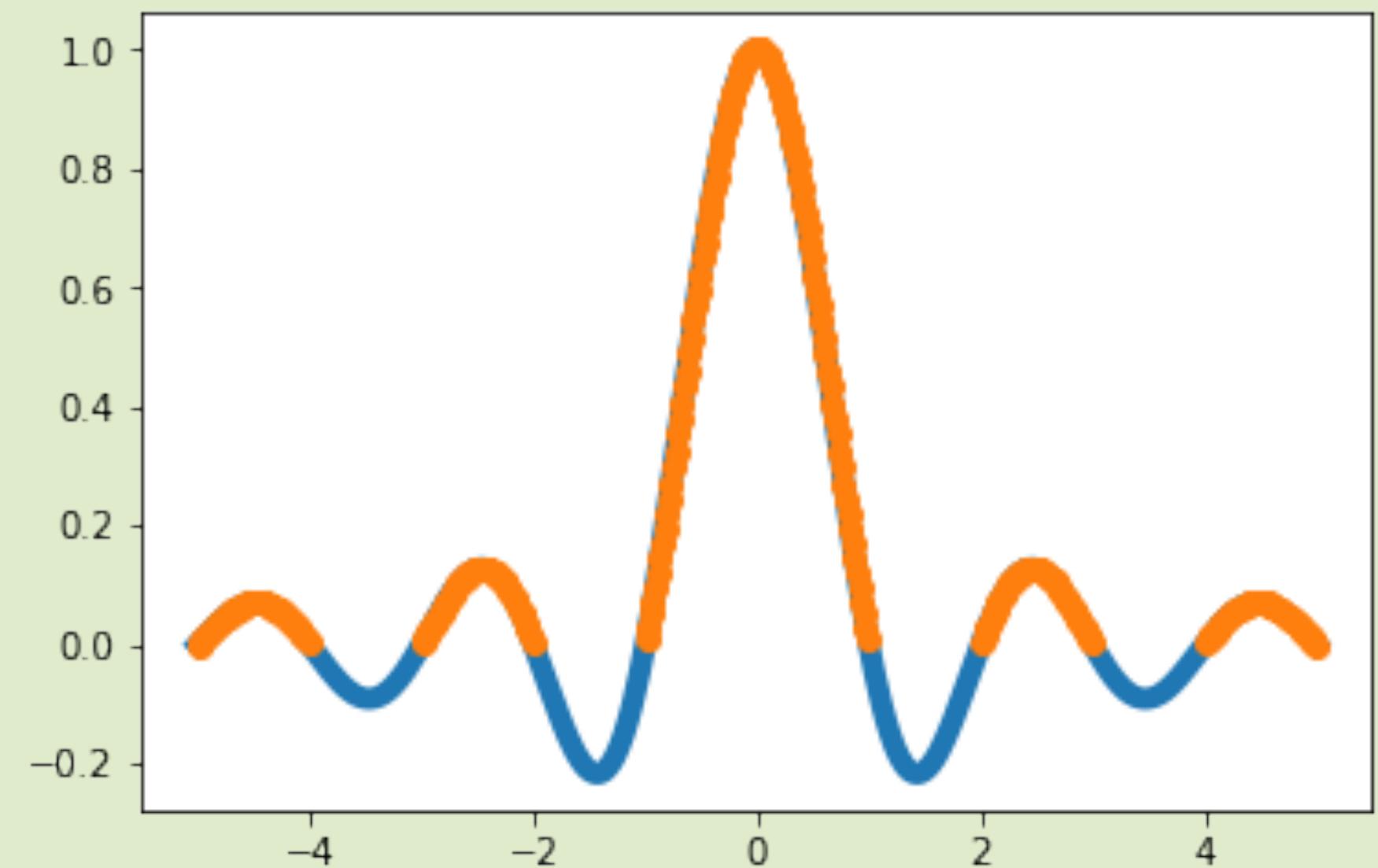
例子

## 活用 filter

畫個函數 (如  $\text{sinc}(x)$ ), 標出正的部份!

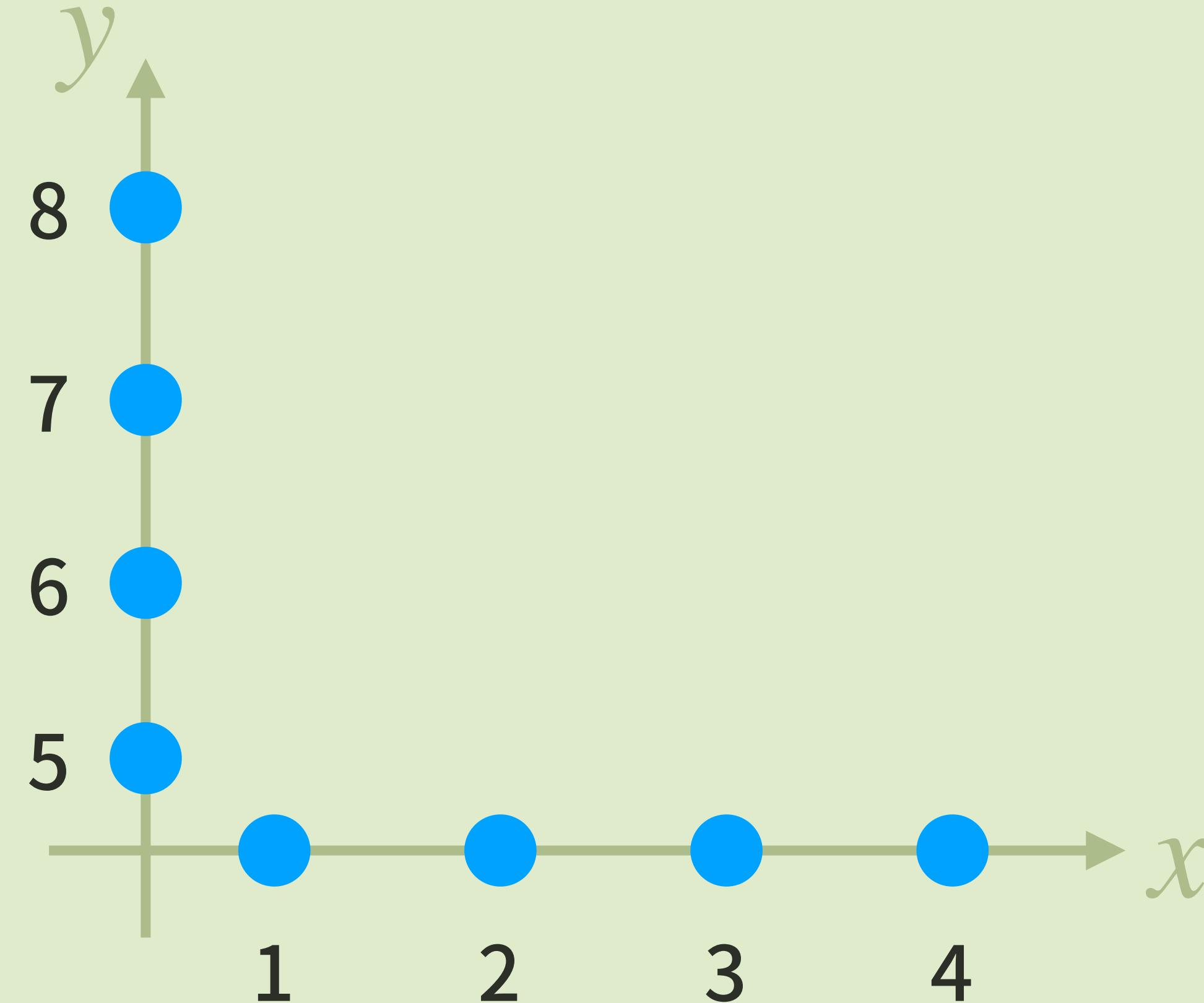
```
x = np.linspace(-5, 5, 500)
y = np.sinc(x)

plt.plot(x, y, lw=6)
plt.plot(x[y>0], y[y>0], 'o')
```



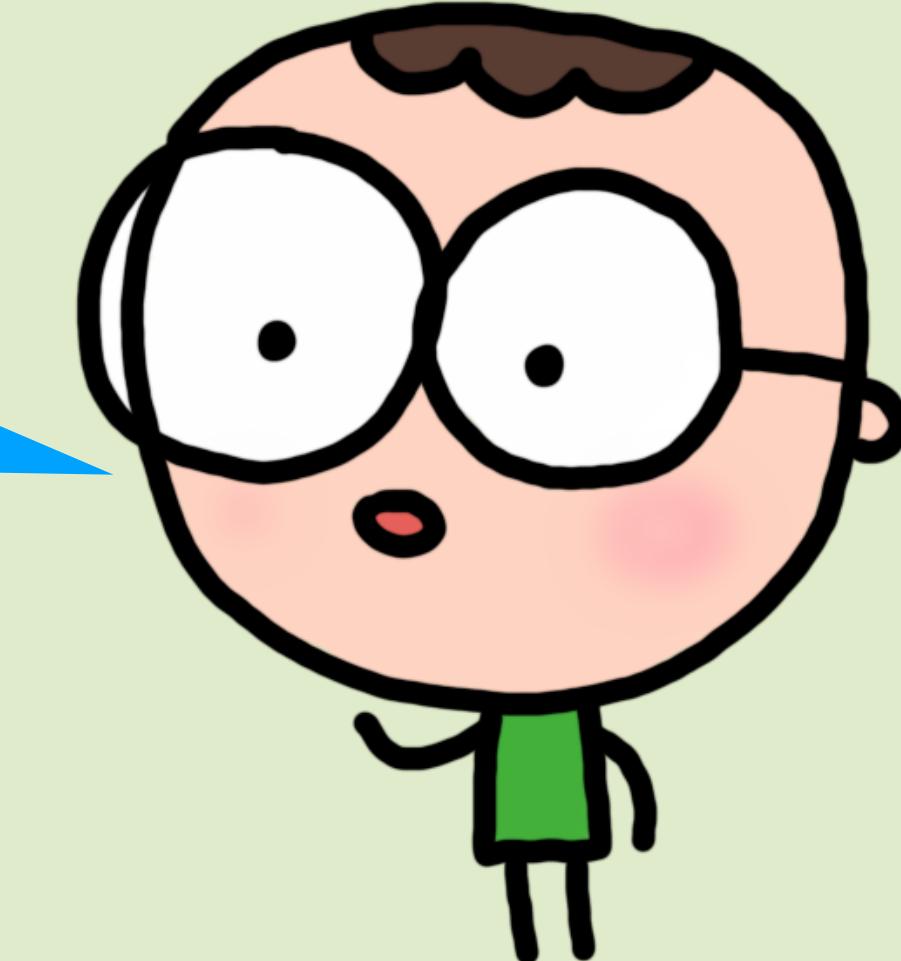
## 小重點

# meshgrid 概念



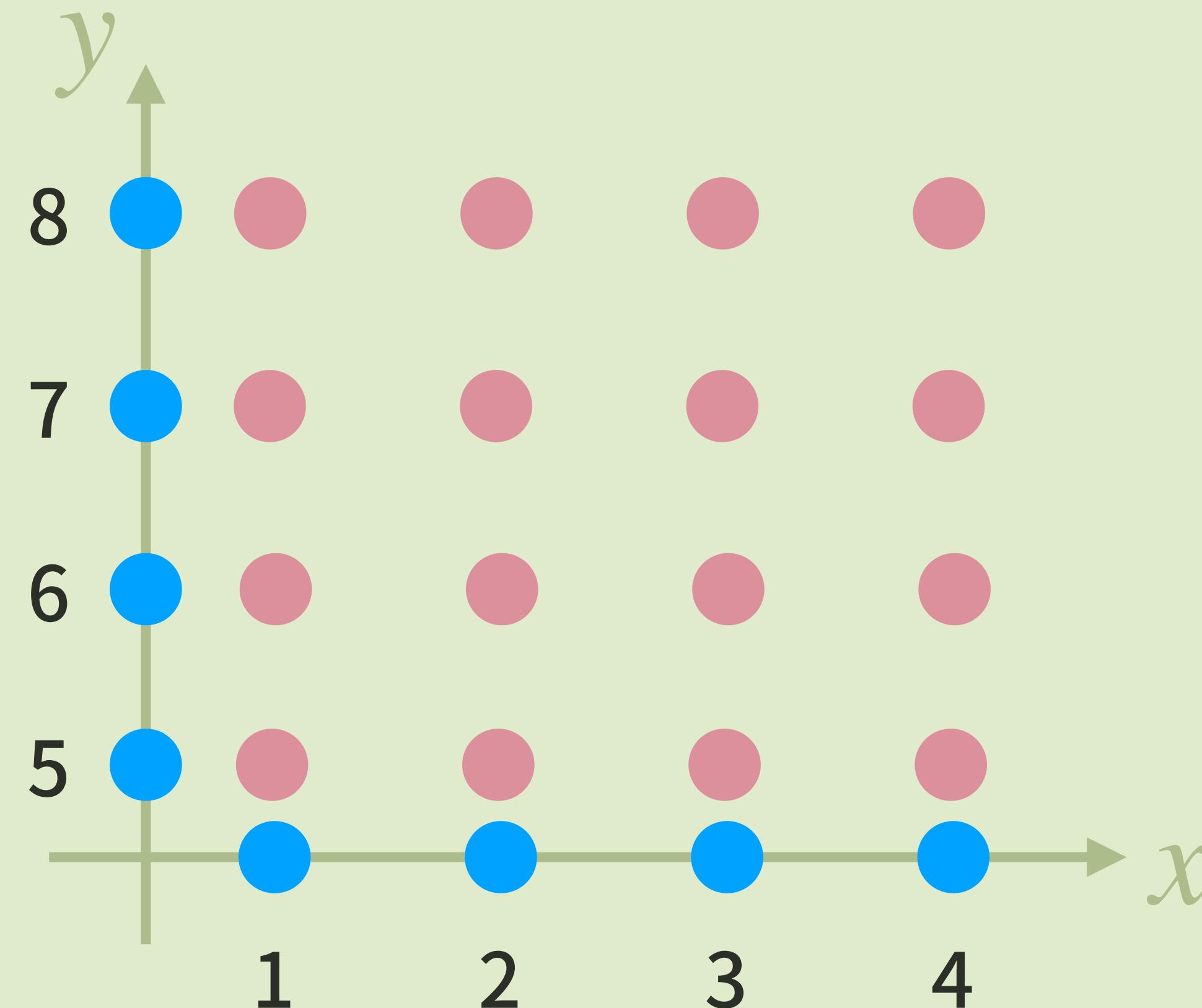
對初學者來說, **meshgrid** 是個有點讓人不懂的指令。它的目的很簡單, 其實就是產生格點。例如我們現在有的  $x, y\dots$

畫 3D, contour  
map 等要用。



## 小重點

# meshgrid 概念



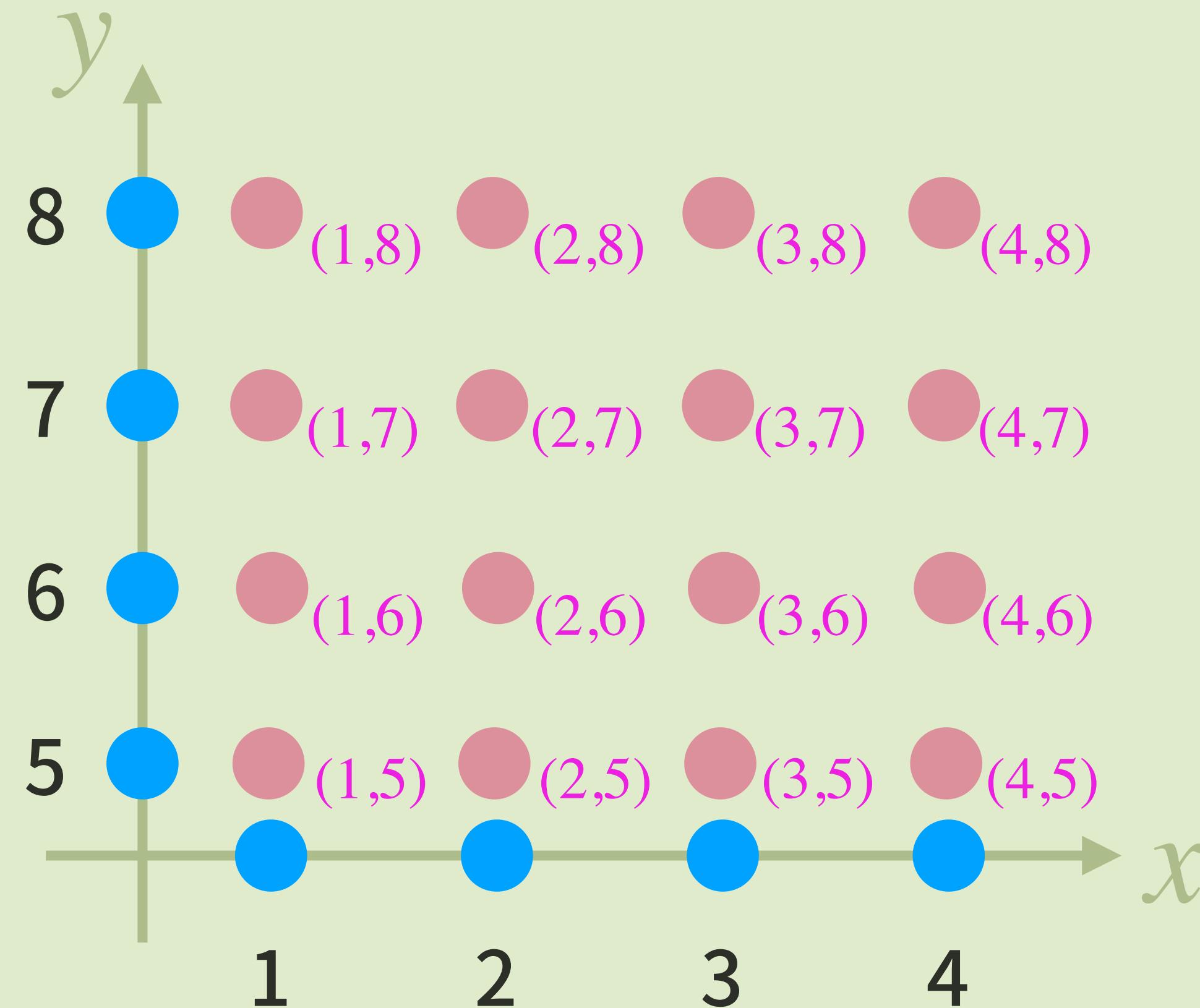
然後產生這些格點。

怎麼做?



## 小重點

# meshgrid 概念

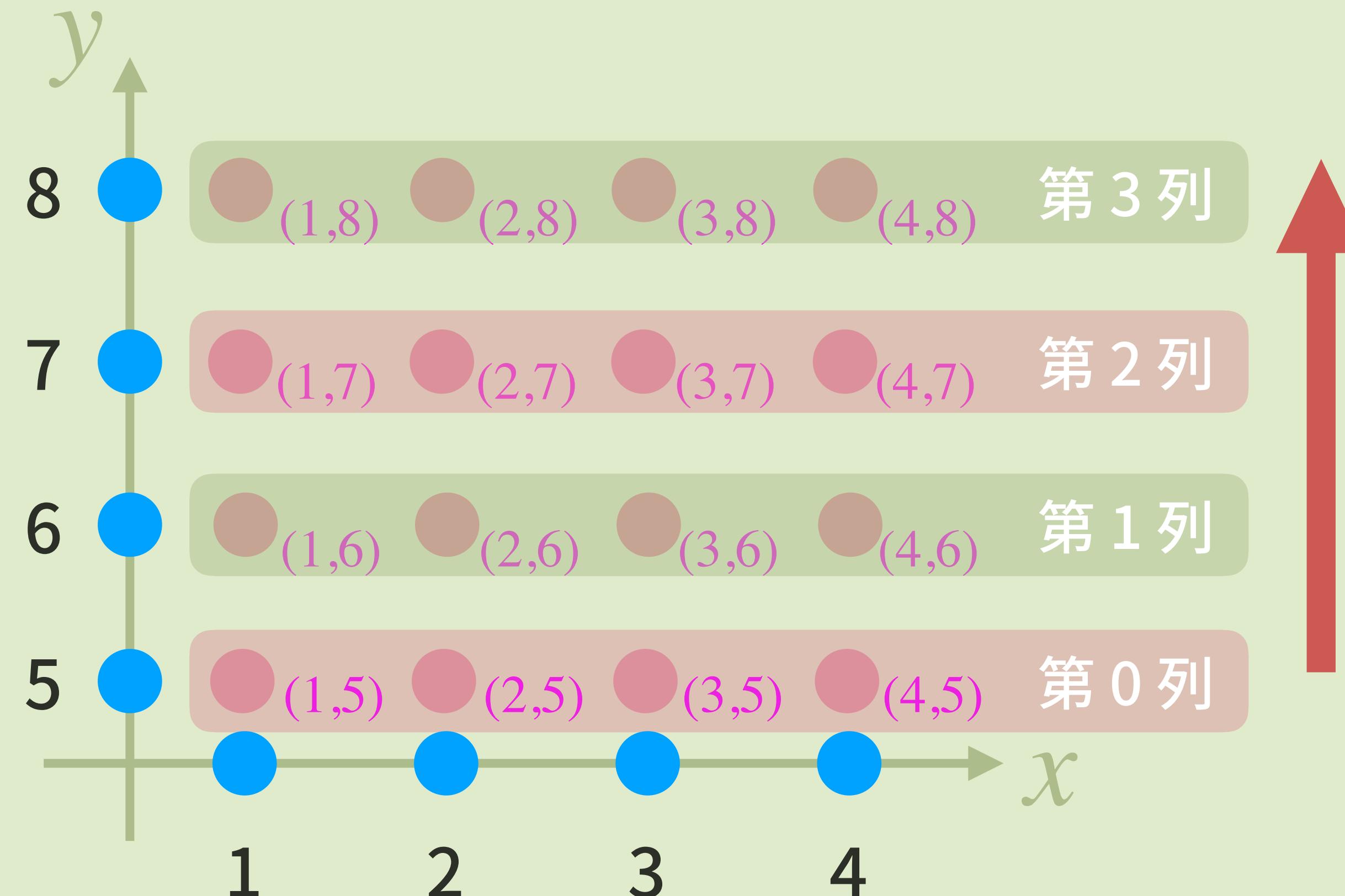


```
x = np.array([1, 2, 3, 4])
y = np.array([5, 6, 7, 8])
x, y = np.meshgrid(x, y)
```

這樣就完成了，但表示法很特別。

## 小重點

# meshgrid 概念



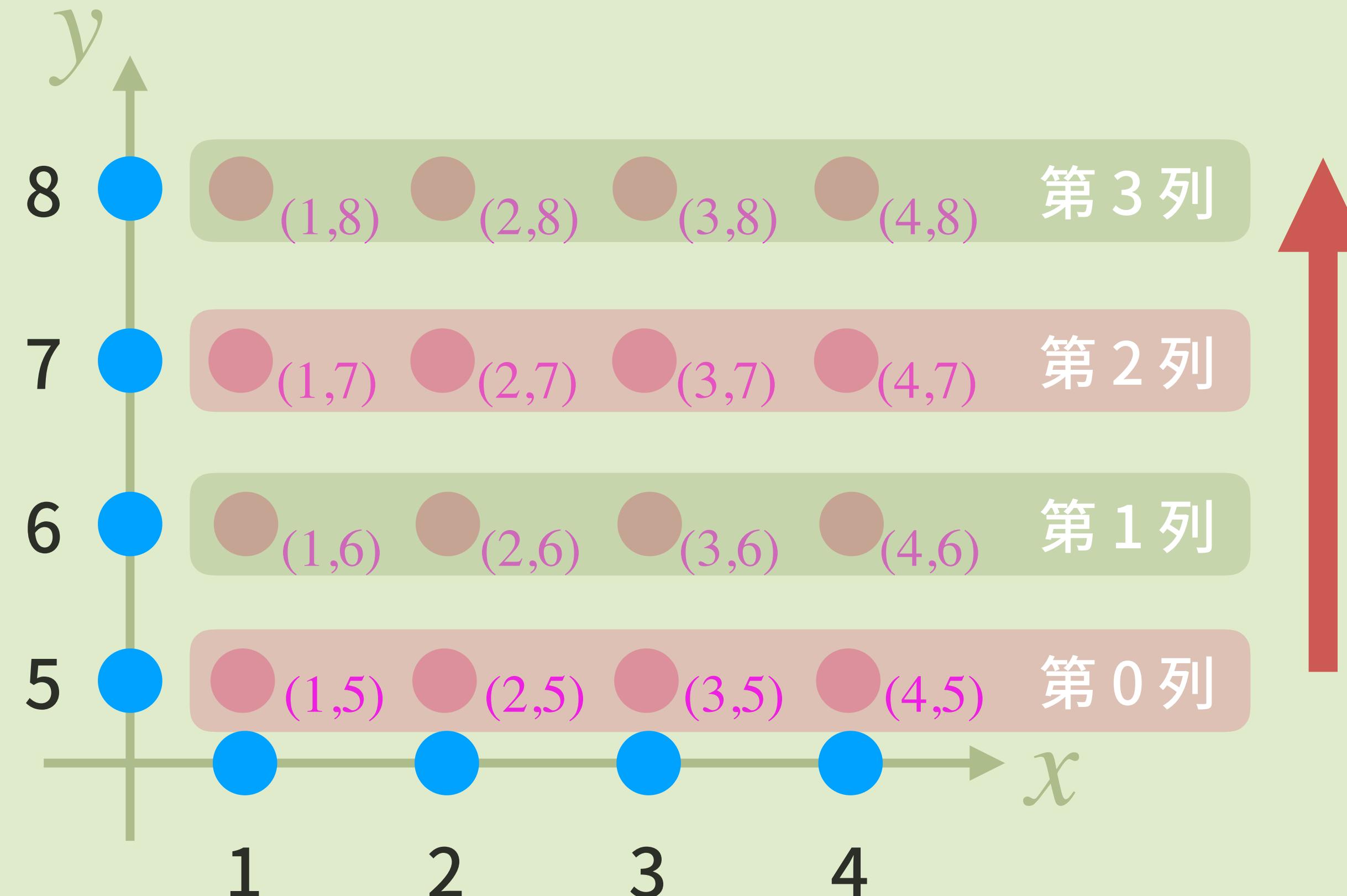
分 x, y 座標, 而且由下而上一列列  
排出。

X

```
array([[1, 2, 3, 4],  
       [1, 2, 3, 4],  
       [1, 2, 3, 4],  
       [1, 2, 3, 4]])
```

## 小重點

# meshgrid 概念



分  $x, y$  座標, 而且由下而上一列列  
排出。

Y

```
array([[5, 5, 5, 5],  
       [6, 6, 6, 6],  
       [7, 7, 7, 7],  
       [8, 8, 8, 8]])
```

應用

## 等高線 contour

```
z = np.random.randint(1, 3, (4, 4))
```

結果：

```
array([[1, 1, 2, 2],  
       [1, 1, 1, 1],  
       [2, 1, 2, 2],  
       [1, 2, 2, 1]])
```

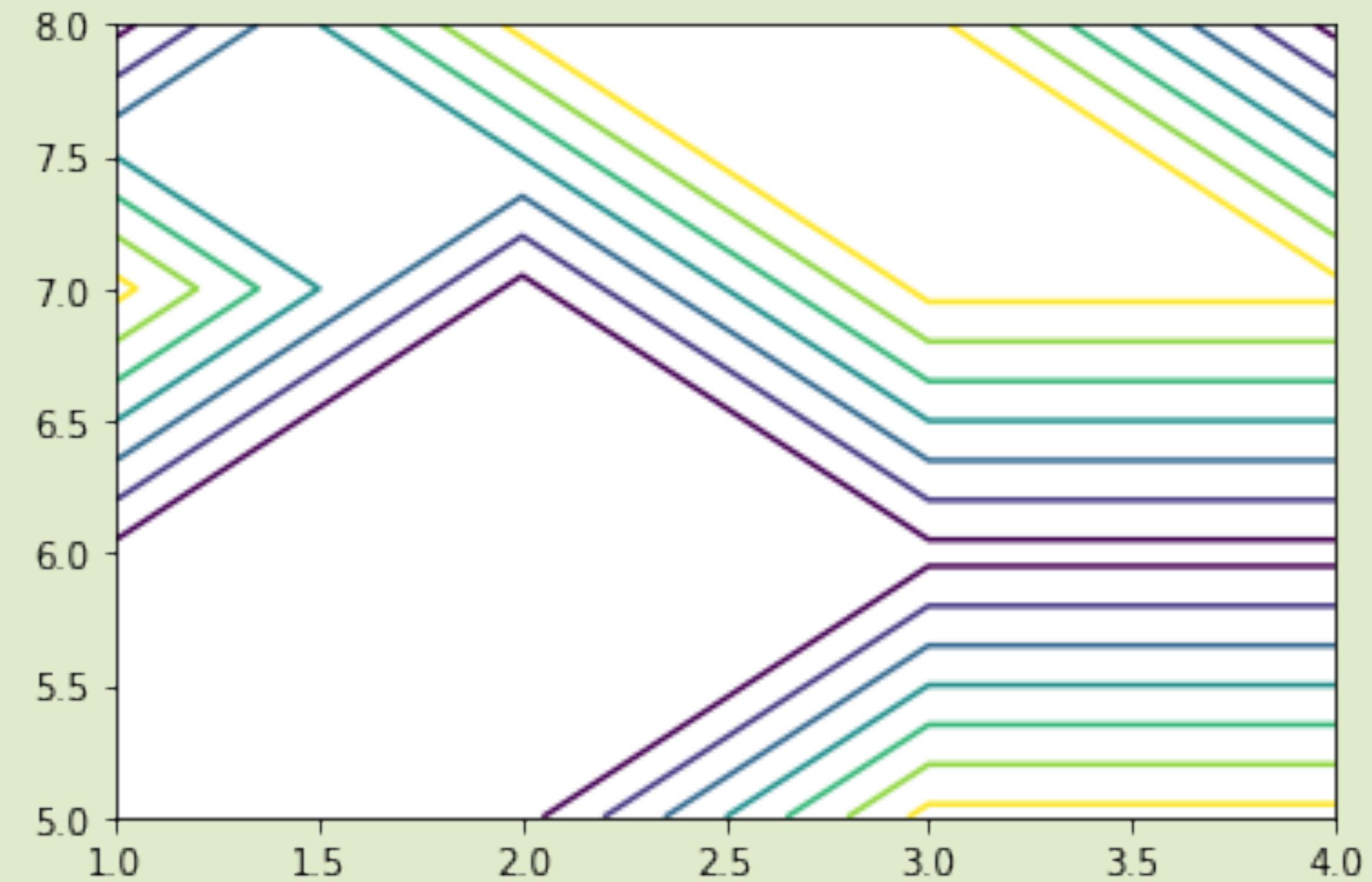
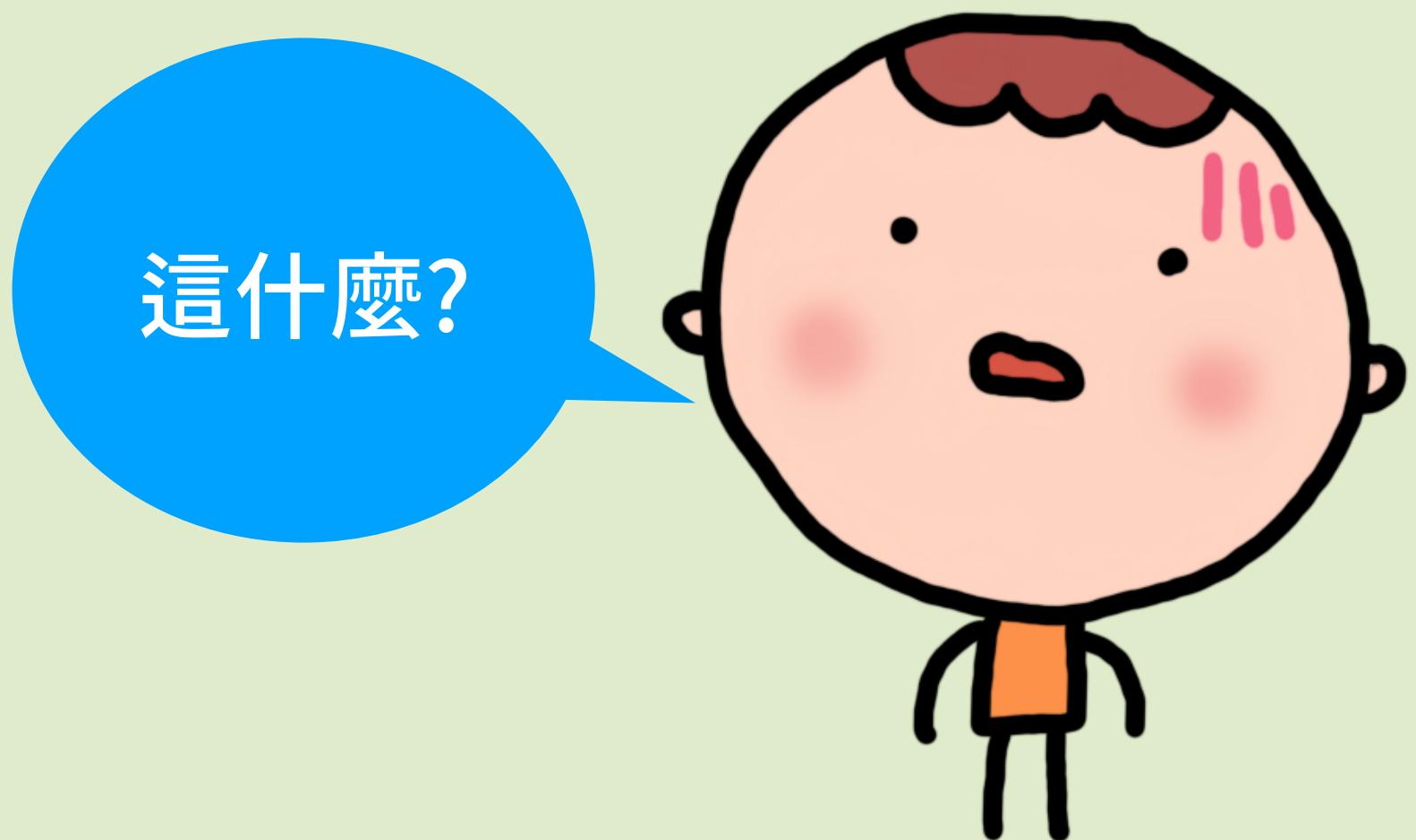
你的數字可  
能不一樣。



應用

## 等高線 contour

```
plt.contour(x, y, z)
```



小技巧

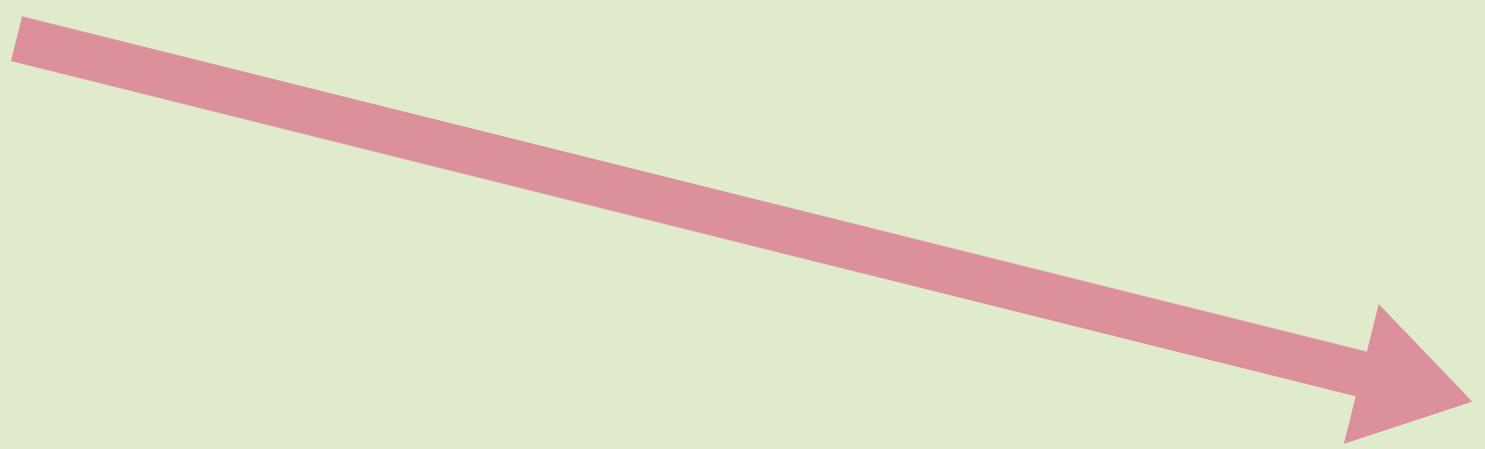
# 拉平一個 array

X

```
array([[1, 2, 3, 4],  
       [1, 2, 3, 4],  
       [1, 2, 3, 4],  
       [1, 2, 3, 4]])
```

輸出：

```
array([1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4])
```

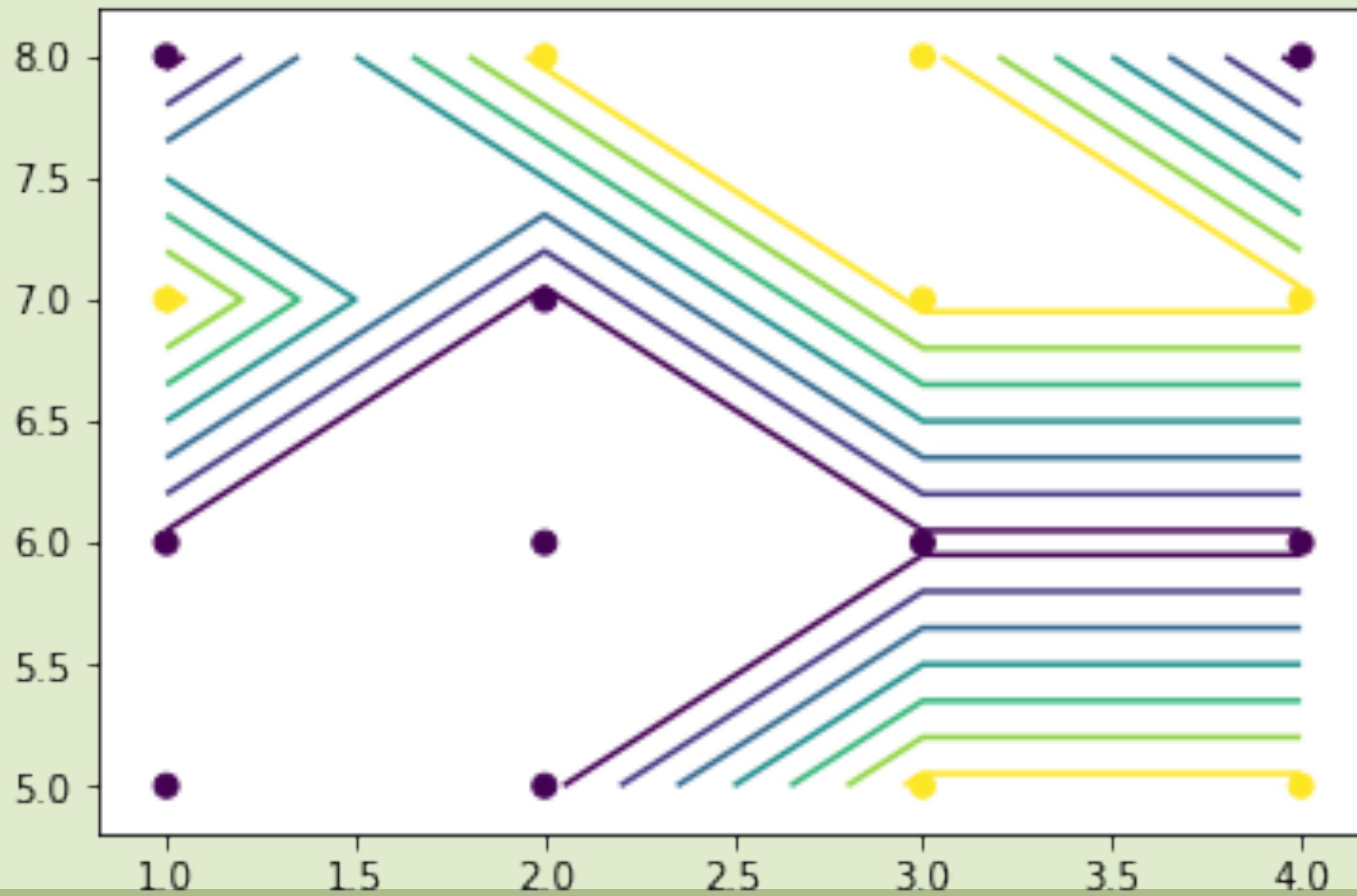


x.ravel()

應用

## 等高線 contour

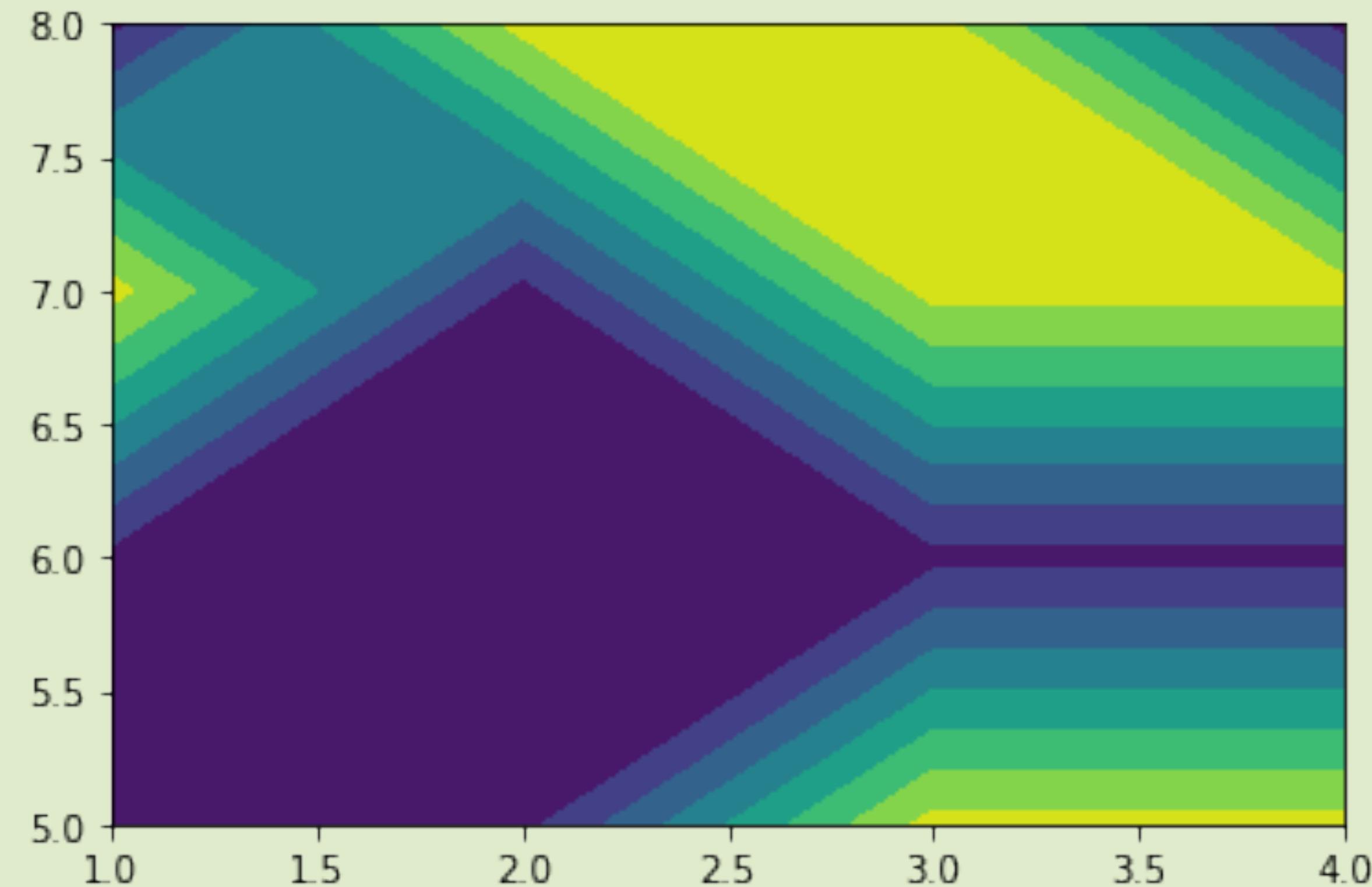
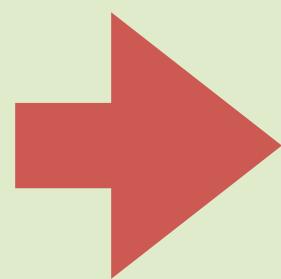
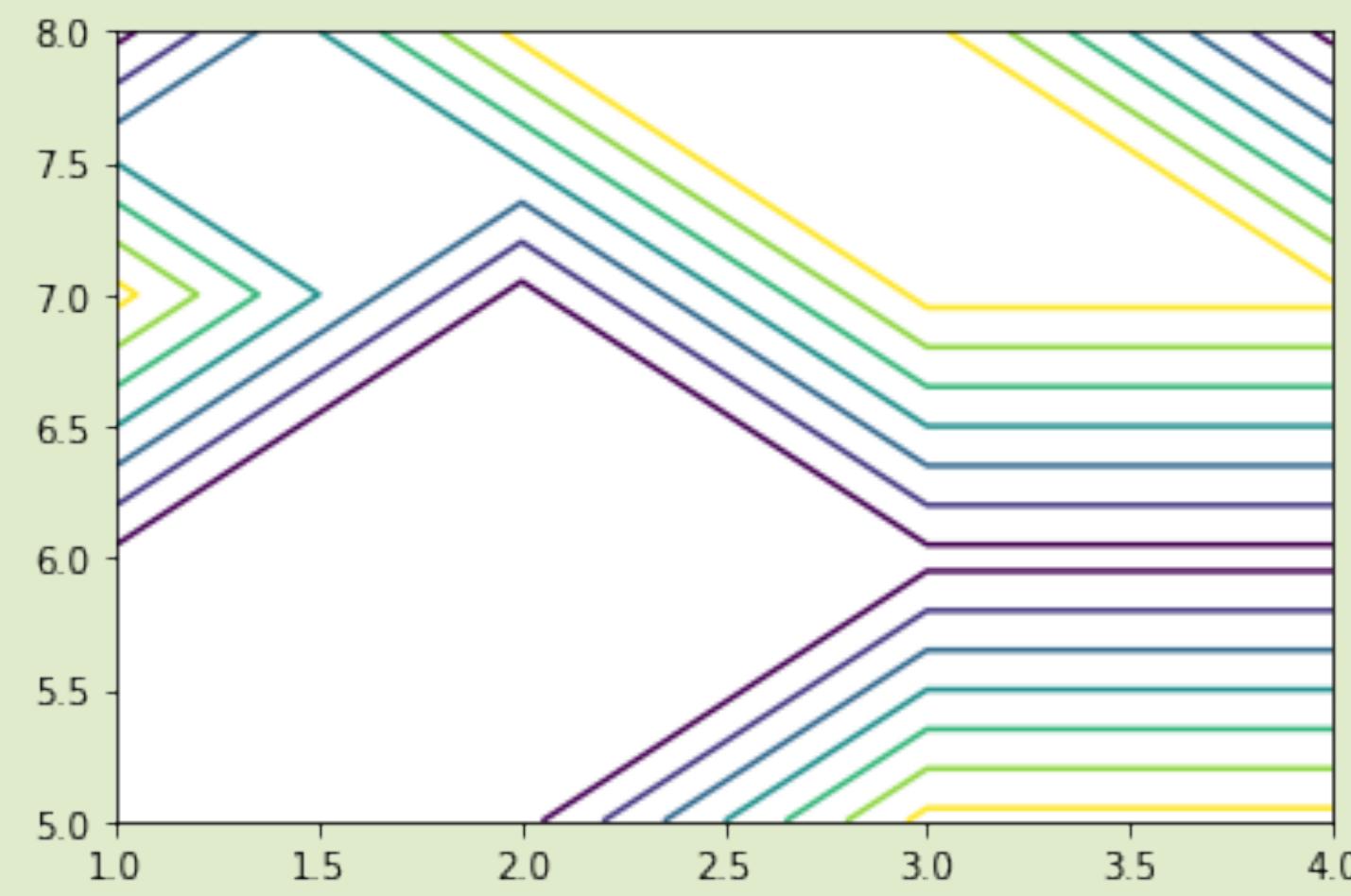
```
plt.contour(x,y,z)  
plt.scatter(x.ravel(),y.ravel(),c=z.ravel())
```



應用

## 填充型等高線 `contourf`

`plt.contourf(x, y, z)`



## 例子

# 要寶等高線

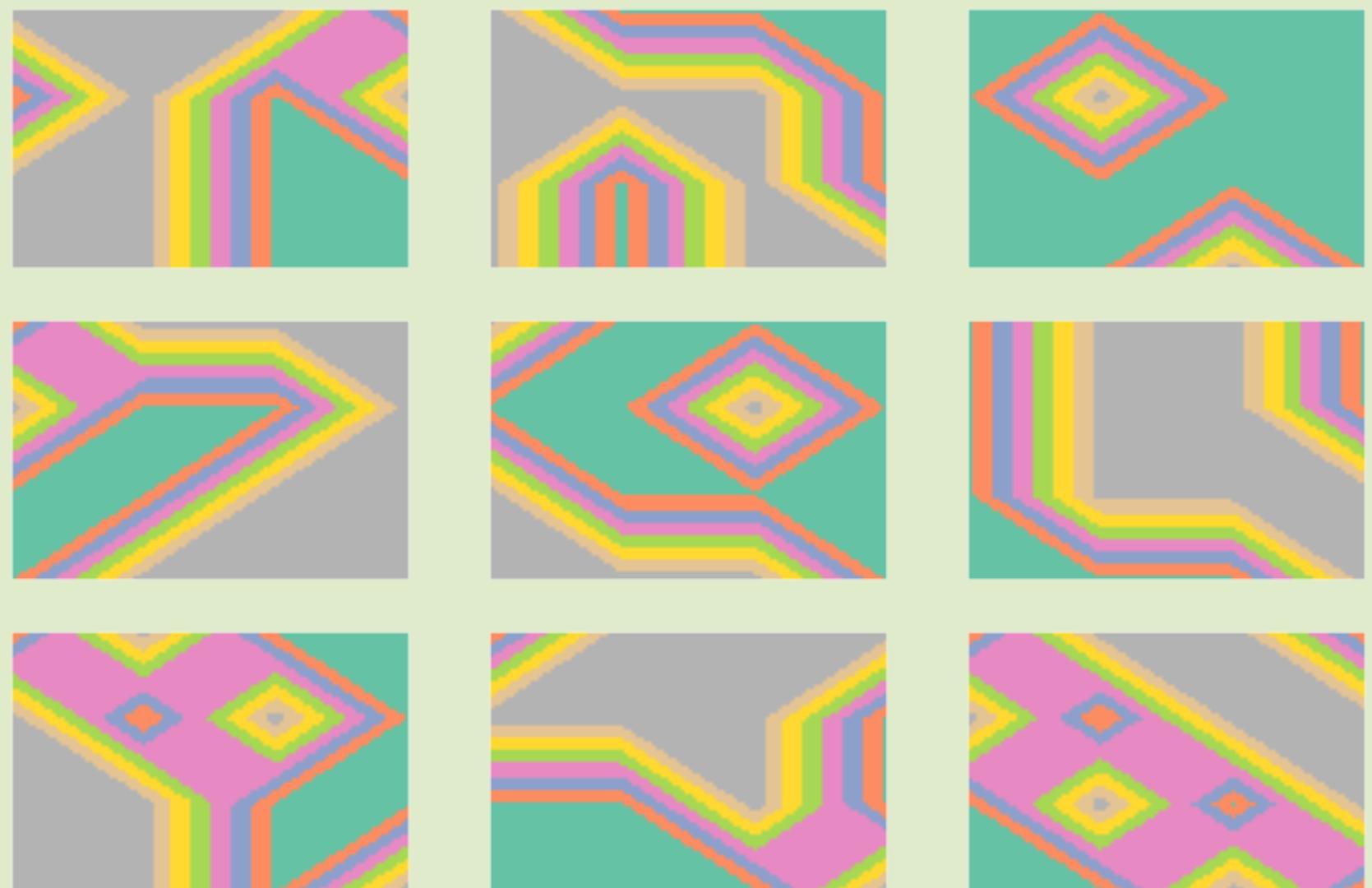
```
for i in range(9):
    plt.subplot(3,3,i+1)
    z = np.random.randint(1,3,(4,4))
    plt.axis('off')
    plt.contour(x,y,z,cmap="Set2")
```



例子

## 要寶等高線

```
for i in range(9):
    plt.subplot(3,3,i+1)
    z = np.random.randint(1,3,(4,4))
    plt.axis('off')
    plt.contourf(x,y,z,cmap="Set2")
```



4

# Pandas

Python 裡的 Excel



原作者

# Wes McKinney



Wes McKinney ✅ @wesmckinn · 2天

I'm believe this was the moment in 2011 where I resolved to quit grad school and work on pandas full time for at least a year or so

Wes McKinney ✅ @wesmckinn

The sleeping dragon has been awakened. I think a lot of code's going to get written the next few weeks  
#python

21

53

595



PyCon APAC 2014, Taiwan <https://youtu.be/i93jidckQ7A>

# Pandas 有兩種基本資料型態


**DataFrame**

如同 Excel 一張表格


**Series**

如同 list, array

## 重點

# 讀入一個 CSV 檔

讀入一個叫 `grades.csv` 的檔案。

```
df = pd.read_csv("grades.csv")
```

沒有下載練習檔案可直接從網路上獲得：

```
df = pd.read_csv("http://bit.ly/gradescsv")
```



重點

# 看一下開頭的內容

`df.head()`

最常用的 pandas  
指令!



	姓名	國文	英文	數學	自然	社會
0	劉俊安	9	10	15	10	13
1	胡玉華	10	10	10	8	9
2	黃淑婷	13	15	8	11	14
3	陳上紫	10	10	8	9	14
4	崔靜成	13	12	14	12	13

## 重點

# 列出一行的資料

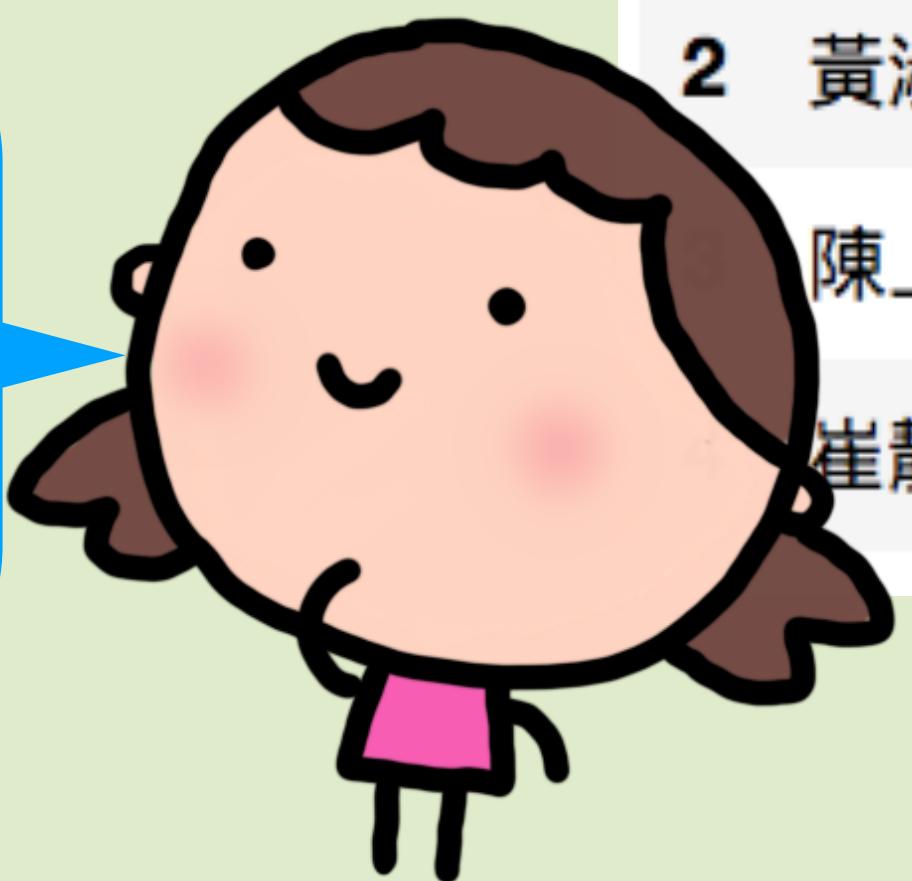
`df[ "國文" ]`

輸出:

```
0      9
1     10
2     13
3     10
4     13
5     13
6     11
7      8
:
:
:
```

看來很像 array, 事實上是 pandas 的 Series。

	姓名	國文	英文	數學	自然	社會
0	劉俊安	9	10	15	10	13
1	胡玉華	10	10	10	8	9
2	黃淑婷	13	15	8	11	14
	陳上紫	10	10	8	9	14
	崔靜成	13	12	14	12	13



## 重點

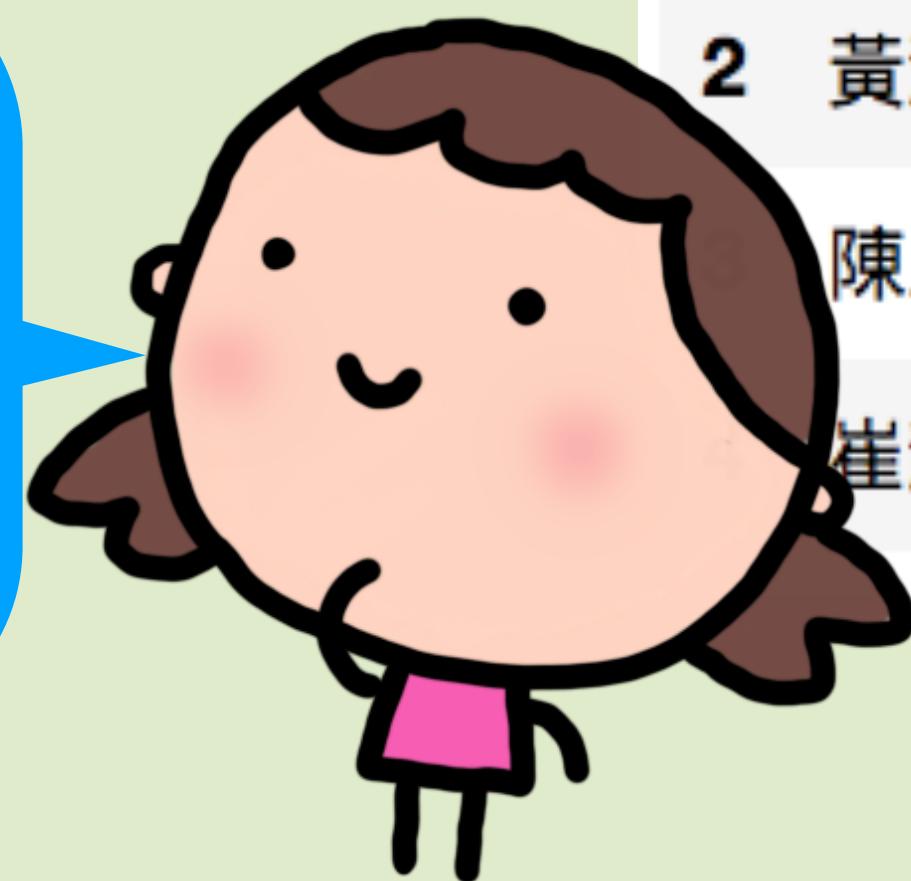
# 列出一行的資料

df. 國文

輸出:

```
0      9
1     10
2     13
3     10
4     13
5     13
6     11
7      8
:
:
:
```

完全一樣, 但  
更潮的作法!!



	姓名	國文	英文	數學	自然	社會
0	劉俊安	9	10	15	10	13
1	胡玉華	10	10	10	8	9
2	黃淑婷	13	15	8	11	14
	陳上紫	10	10	8	9	14
	崔靜成	13	12	14	12	13

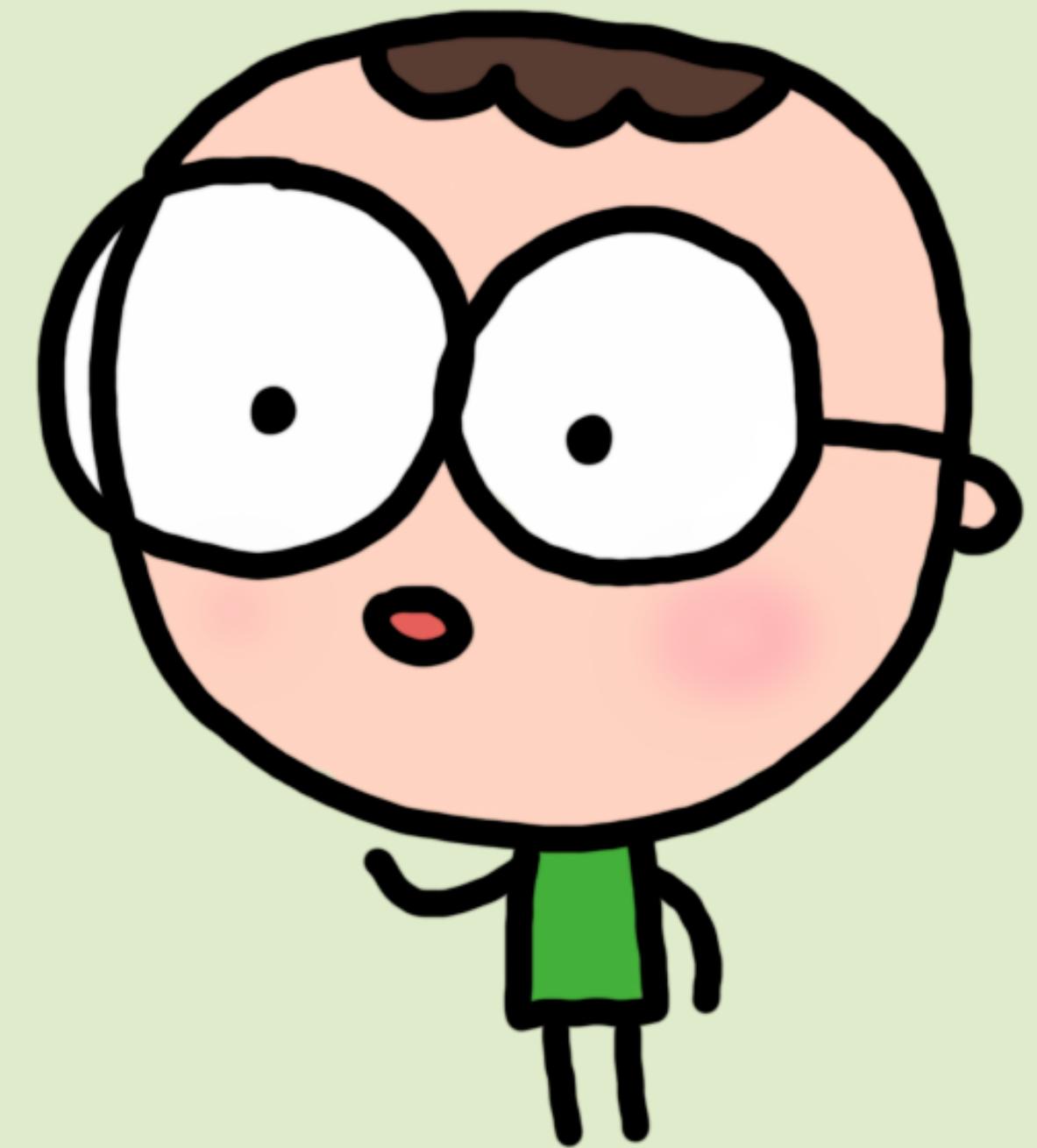
重點

## 資料轉成 numpy 的 array

```
cg = df.國文.values
```

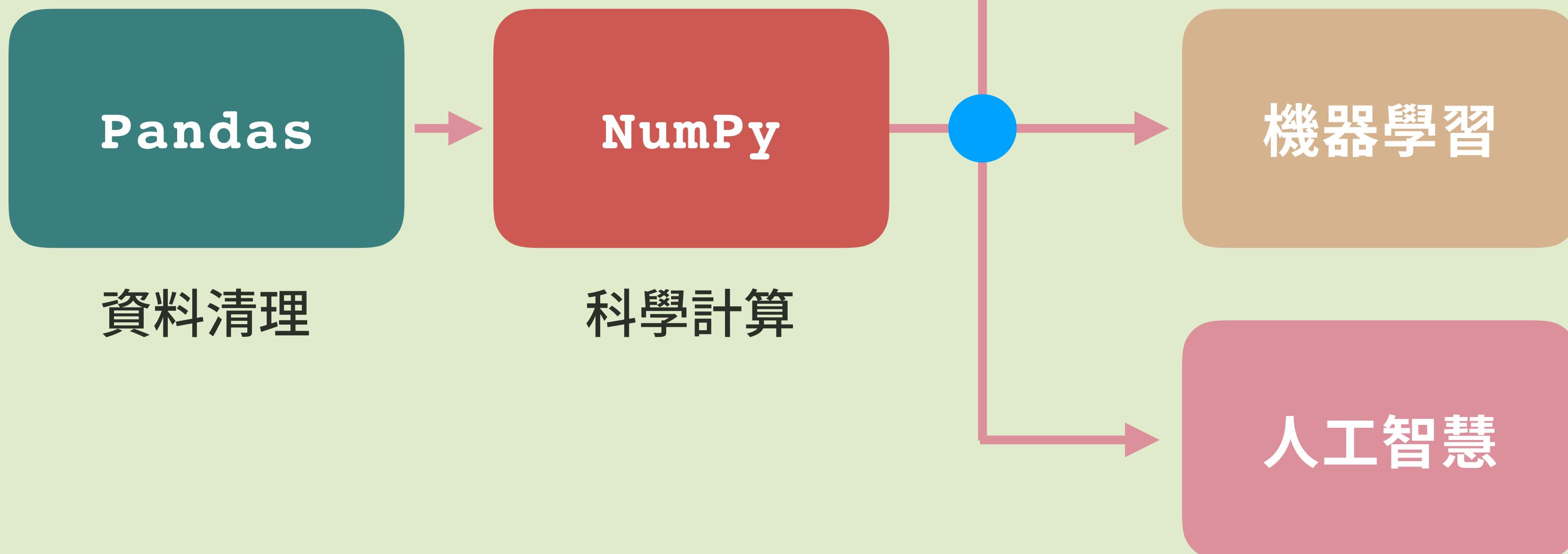
NumPy 的 array。

非常重要!





Python 資料分析流程。



**StatsModels**

**SciKit-Learn**

**Keras**

例子

## 轉成 NumPy 做計算

```
cg.mean()  
cg.std()
```

平均值

標準差

這當然只是舉例, 這麼  
衰敗的我也是會算的。

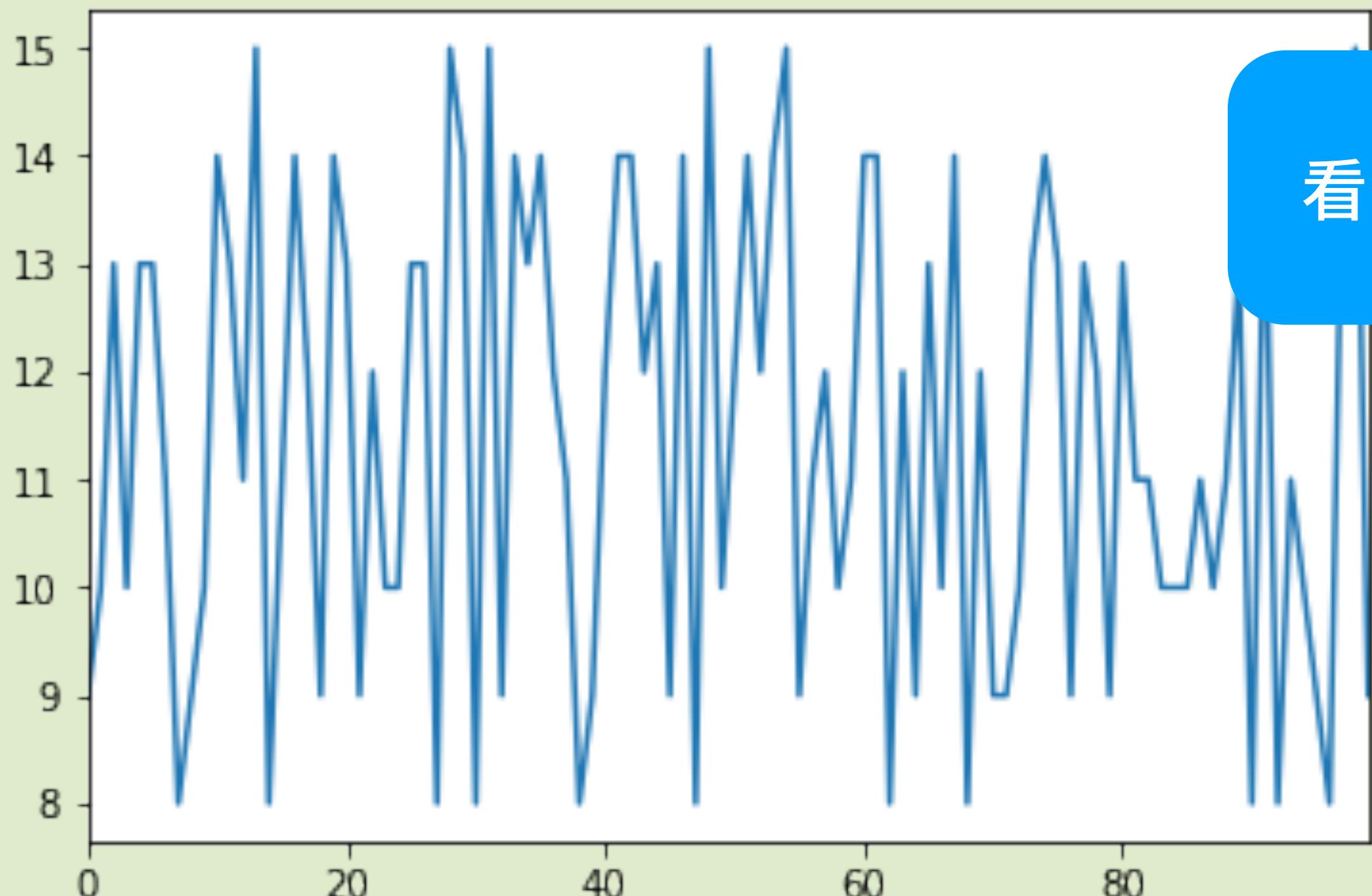


小重點

## 在 Pandas 畫圖

`df.國文.plot()`

`plt.plot` 的參數基本上都通用的!



看吧, 我畫圖都會!

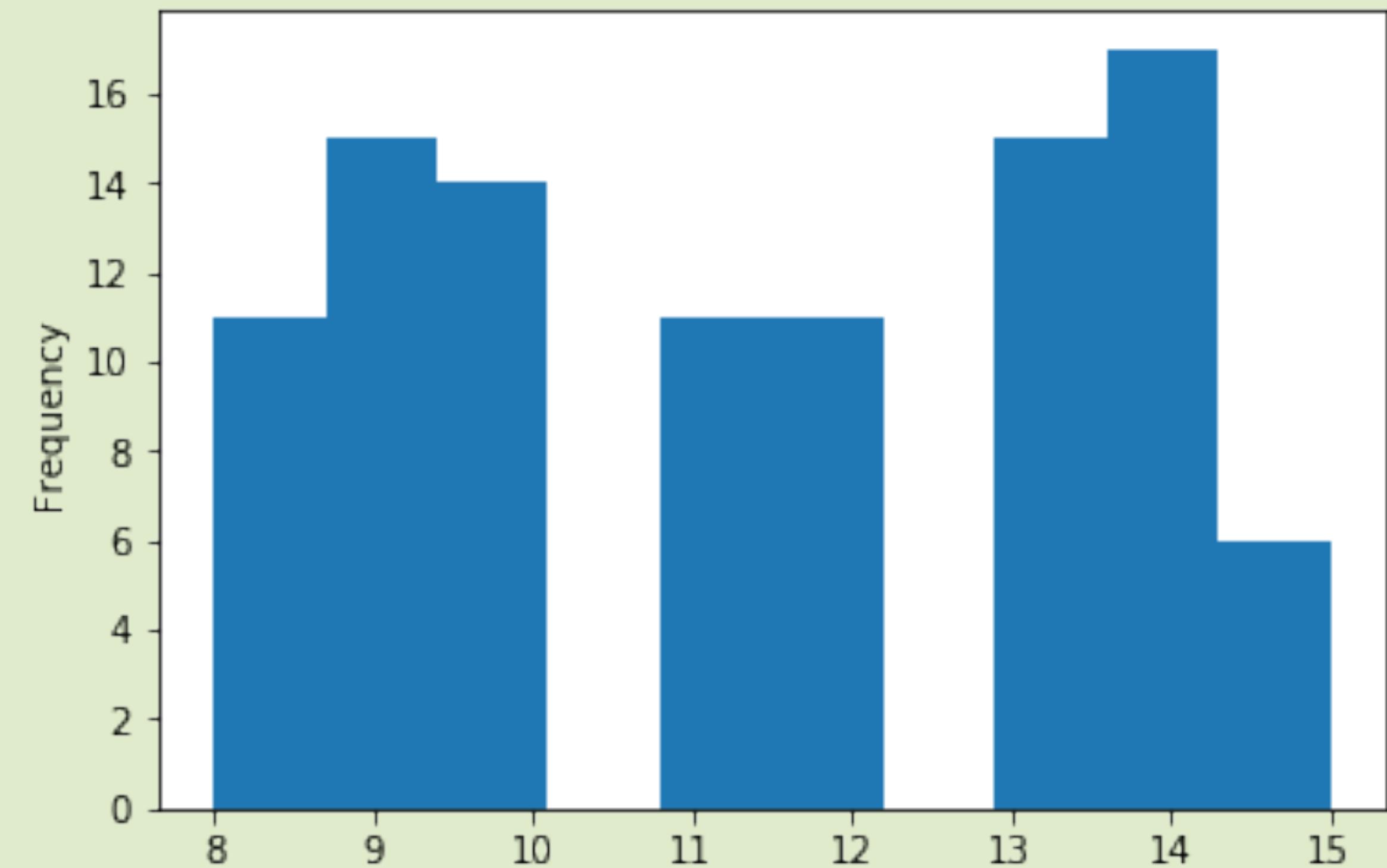


## 小重點

# 在 Pandas 畫圖

畫直方圖好像比較有道理。

```
df.國文.plot(kind='hist')
```

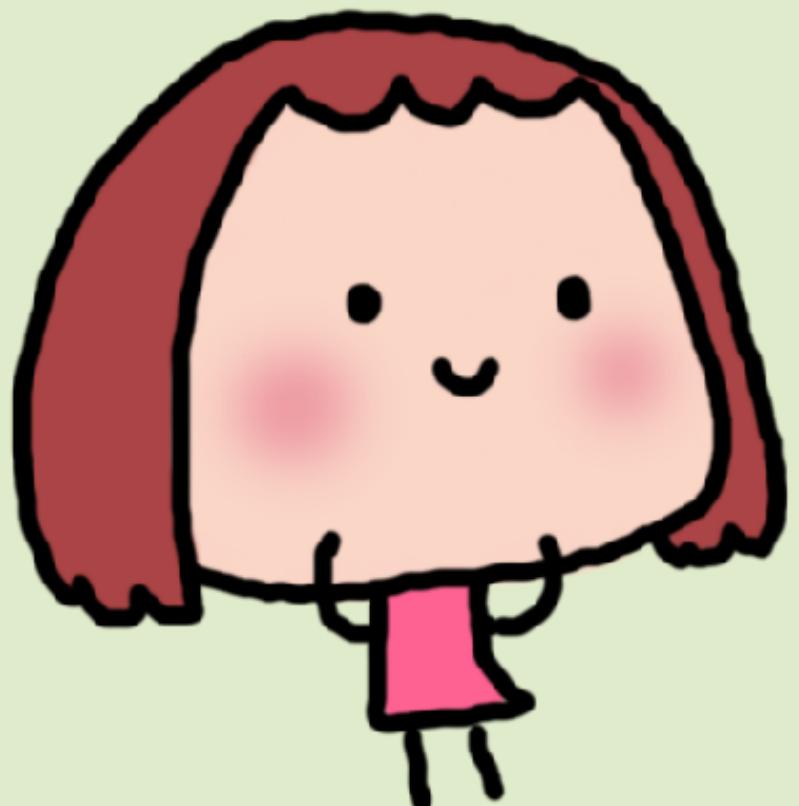


## 重點

# 增加一行

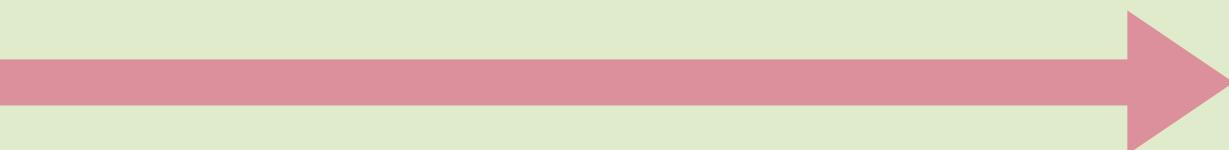
我們來算總級分。

```
df[ "總級分" ] = df.sum(axis=1)
```



注意 `axis` 方向。

	姓名	國文	英文	數學	自然	社會	總級分
0	劉俊安	9	10	15	10	13	57
1	胡玉華	10	10	10	8	9	47
2	黃淑婷	13	15	8	11	14	61
3	陳上紫	10	10	8	9	14	51
4	崔靜成	13	12	14	12	13	64



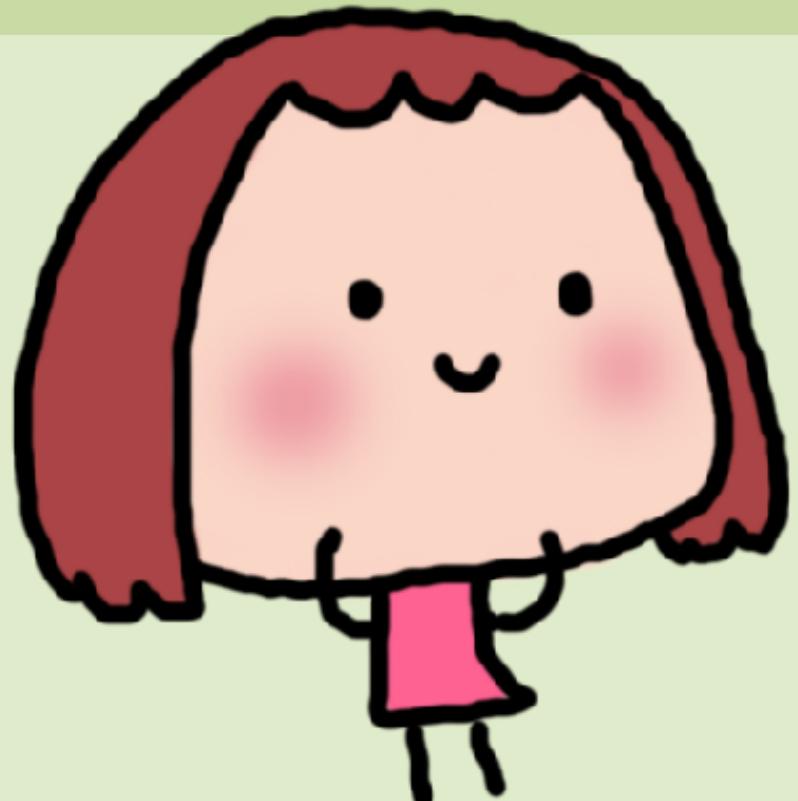
`axis=1`

## 重點

# 增加一行

假設某系特別想看國、英、數三科，而且  
數學要加權乘以 2。

`df[ "加權" ] = df . 國文 + df . 英文 + df . 數學 * 2`



這是 array 型計算！

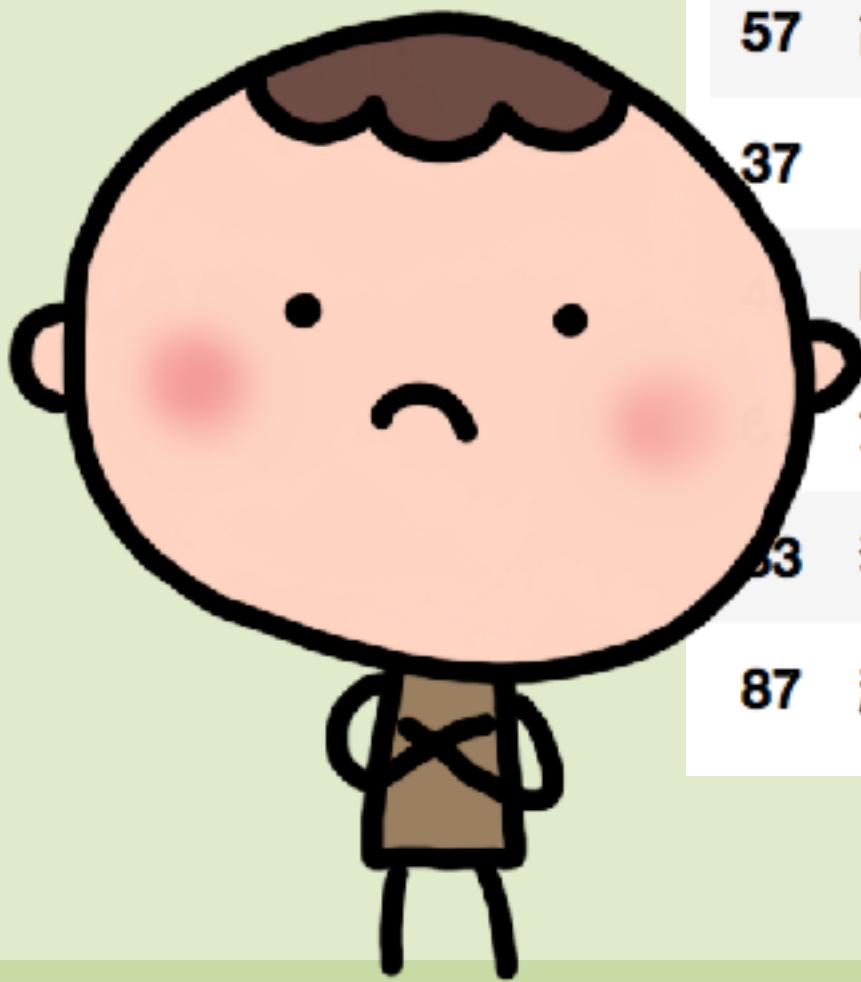
	姓名	國文	英文	數學	自然	社會	總級分	加權
0	劉俊安	9	10	15	10	13	57	49
1	胡玉華	10	10	10	8	9	47	40
2	黃淑婷	13	15	8	11	14	61	44
3	陳上紫	10	10	8	9	14	51	36
4	崔靜成	13	12	14	12	13	64	53

## 重點

# 排序

某系想看最高總級分前 10 名同學。

以後不可以。



	姓名	國文	英文	數學	自然	社會	總級分	加權
80	施雅鈴	13	15	12	13	13	66	52
12	李正偉	11	15	11	14	15	66	48
54	陳怡潔	15	15	9	15	11	65	48
25	蔡亦瑄	13	13	14	13	12	65	54
57	胡淳茜	12	15	14	13	11	65	55
37	曾怡君	11	12	15	13	14	65	53
	陳怡婷	15	14	12	9	15	65	53
	俞志峰	9	14	13	14	15	65	49
63	李士賢	10	14	15	13	13	65	54
87	趙偉希	10	13	14	13	15	65	51

```
df.sort_values(by="總級分", ascending=False).head(10)
```

## 重點

# 排序

想想其實是要看加權分最高，同分才看總級分。

我們把這段叫 df2

```
df.sort_values(by=["加權", "總級分"], ascending=False).head(10)
```

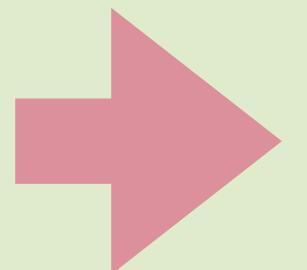
	姓名	國文	英文	數學	自然	社會	總級分	加權
73	吳志遠	13	15	15	8	8	59	58
57	胡淳茜	12	15	14	13	11	65	55
36	詹威德	12	13	15	10	14	64	55
25	蔡亦瑄	13	13	14	13	12	65	54
83	李士賢	10	14	15	13	13	65	54
44	童心怡	13	13	14	8	13	61	54
37	曾怡君	11	12	15	13	14	65	53
48	陳怡婷	15	14	12	9	15	65	53
4	崔靜成	13	12	14	12	13	64	53
67	林哲法	14	9	15	10	14	62	53

## 重點

# 重設 index

這 `index` 好醜，我們想重設一下，從小到大。記得我們的資料庫有 100 位同學。

	姓名	國文	英文	數學	自然	社會	總級分	加權
73	吳志遠	13	15	15	8	8	59	58
57	胡淳茜	12	15	14	13	11	65	55
36	詹威德	12	13	15	10	14	64	55
25	蔡亦瑄	13	13	14	13	12	65	54
83	李士賢	10	14	15	13	13	65	54
44	童心怡	13	13	14	8	13	61	54
37	曾怡君	11	12	15	13	14	65	53
48	陳怡婷	15	14	12	9	15	65	53
4	崔靜成	13	12	14	12	13	64	53
67	林哲法	14	9	15	10	14	62	53



	姓名	國文	英文	數學	自然	社會	總級分	加權
0	劉俊安	9	10	15	10	13	57	49
1	胡玉華	10	10	10	8	9	49	49
2	黃淑婷	13	15	8	11	14	57	49
3	陳上紫	10	10	8	9	9	45	42
4	崔靜成	13	12	14	12	12	57	49
5	張雅岳	13	12	12	12	8	45	42
6	梁俊翔	11	13	10	10	14	57	49
7	林金鳳	8	9	10	10	8	45	37
8	許協旺	9	9	12	10	10	50	42
9	郭雅惠	10	15	12	11	9	57	49

`df2.reindex(range(100))`

怎麼排序也變  
回去了？



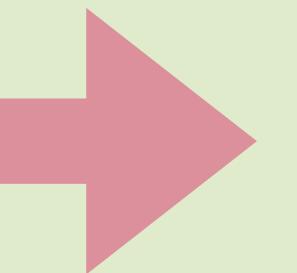
重點

# 重設 index

我們發現有個 **reset\_index!**

`df2.reset_index()`

	姓名	國文	英文	數學	自然	社會	總級分	加權
73	吳志遠	13	15	15	8	8	59	58
57	胡淳茜	12	15	14	13	11	65	55
36	詹威德	12	13	15	10	14	64	55
25	蔡亦瑄	13	13	14	13	12	65	54
83	李士賢	10	14	15	13	13	65	54
44	童心怡	13	13	14	8	13	61	54
37	曾怡君	11	12	15	13	14	65	53
48	陳怡婷	15	14	12	9	15	65	53
4	崔靜成	13	12	14	12	13	64	53
67	林哲法	14	9	15	10	14	62	53



	index	姓名	國文	英文	數學	自然	社會	總級分	加權
0	73	吳志遠	13	15	15	8	8	59	58
1	57	胡淳茜	12	15	14	13			
2	36	詹威德	12	13	15	14	13	64	55
3	25	蔡亦瑄	13	13	14	13	12	65	54
4	83	李士賢	10	14	15	13	12	65	54
5	44	童心怡	13	13	14	13	12	61	54
6	37	曾怡君	11	12	15	13	12	65	53
7	48	陳怡婷	15	14	12	9	15	65	53
8	4	崔靜成	13	12	14	12	13	64	53
9	67	林哲法	14	9	15	10	14	62	53

在我發火之前  
你最好快弄對!



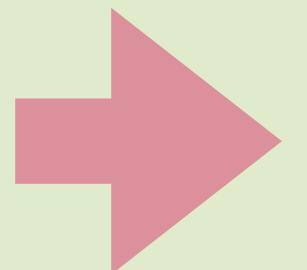
重點

# 重設 index

不會是這樣吧... 順便從 1 開始。

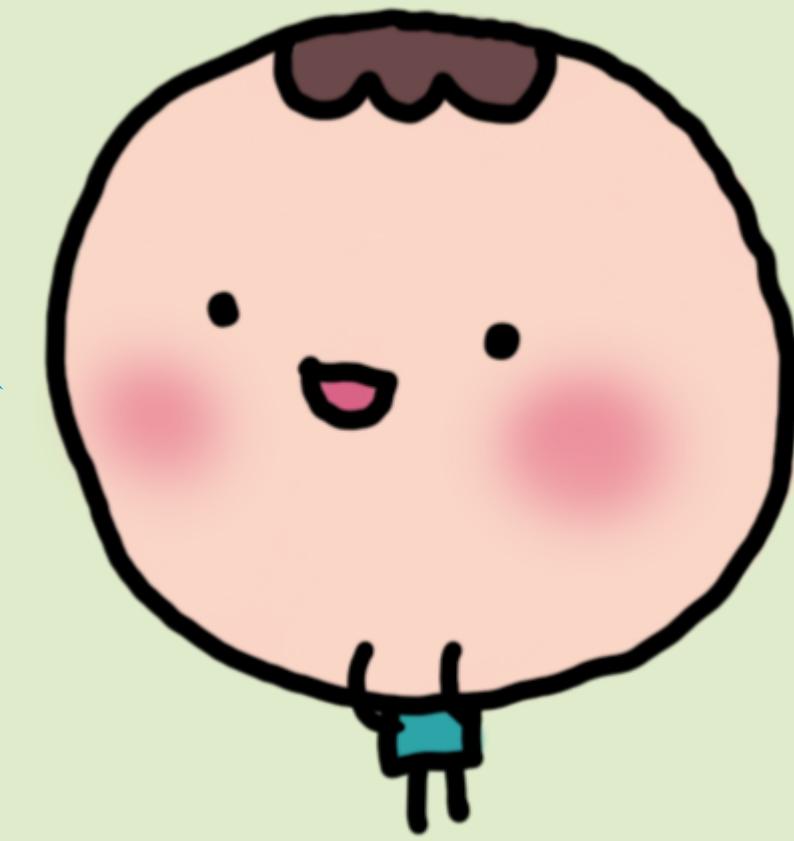
`df2.index = range(1, 101)`

	姓名	國文	英文	數學	自然	社會	總級分	加權
73	吳志遠	13	15	15	8	8	59	58
57	胡淳茜	12	15	14	13	11	65	55
36	詹威德	12	13	15	10	14	64	55
25	蔡亦瑄	13	13	14	13	12	65	54
83	李士賢	10	14	15	13	13	65	54
44	童心怡	13	13	14	8	13	61	54
37	曾怡君	11	12	15	13	14	65	53
48	陳怡婷	15	14	12	9	15	65	53
4	崔靜成	13	12	14	12	13	64	53
67	林哲法	14	9	15	10	14	62	53



	姓名	國文	英文	數學	自然	社會	總級分	加權
1	吳志遠	13	15	15	8	8	59	58
2	胡淳茜	12	15	14	13	11	65	55
3	詹威德	12	13	15	10	14	64	55
4	蔡亦瑄	13	13	14	13	12	65	54
5	李士賢	10	14	15	13	13	65	54
6	童心怡	13	13	14	8	13	61	54
7	曾怡君	11	12	15	13	14	65	53
8	陳怡婷	15	14	12	9	15	65	53
9	崔靜成	13	12	14	12	13	64	53
10	林哲法	14	9	15	10	14	62	53

得救了!

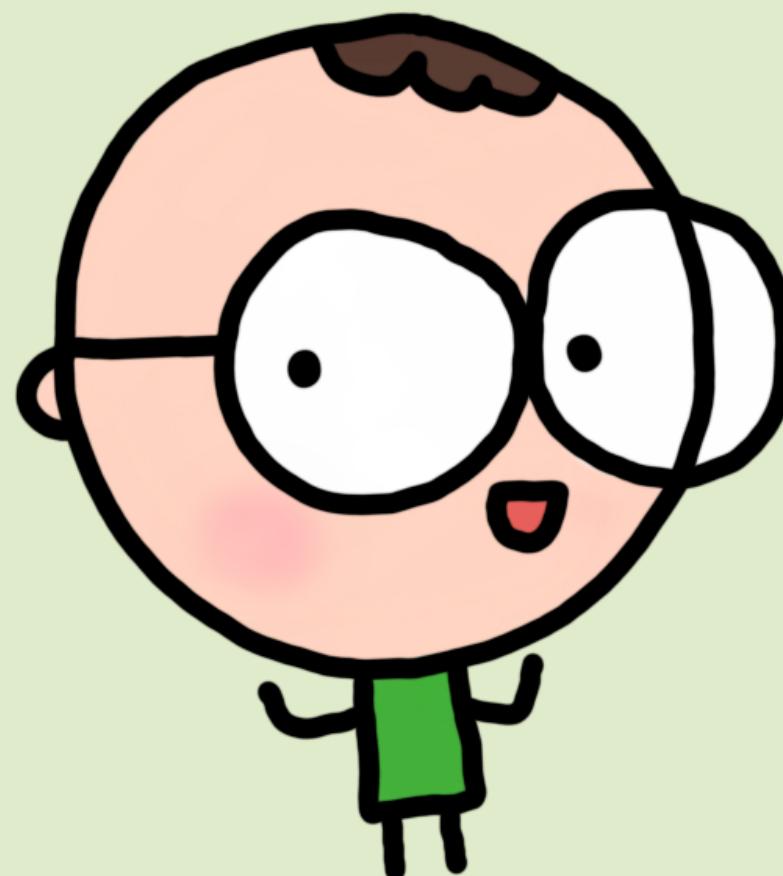


## 重點

# 篩出我們要的資料

假設我們要找數學滿級分的同學。

`df2[df2.數學==15]`



和 NumPy 基本上  
是一樣的!

	姓名	國文	英文	數學	自然	社會	總級分	加權
1	吳志遠	13	15	15	8	8	59	58
3	詹威德	12	13	15	10	14	64	55
5	李士賢	10	14	15	13	13	65	54
7	曾怡君	11	12	15	13	14	65	53
10	林哲法	14	9	15	10	14	62	53

## 重點

# 篩出我們要的資料

比較要注意的是如果數學、英文都要滿級分...

```
df2[ (df2.數學==15) & (df2.英文==15) ]
```

	姓名	國文	英文	數學	自然	社會	總級分	加權
73	吳志遠	13	15	15	8	8	59	58

注意 Pandas 繼  
輯運算符號。



運算元	說明
&	and
	or

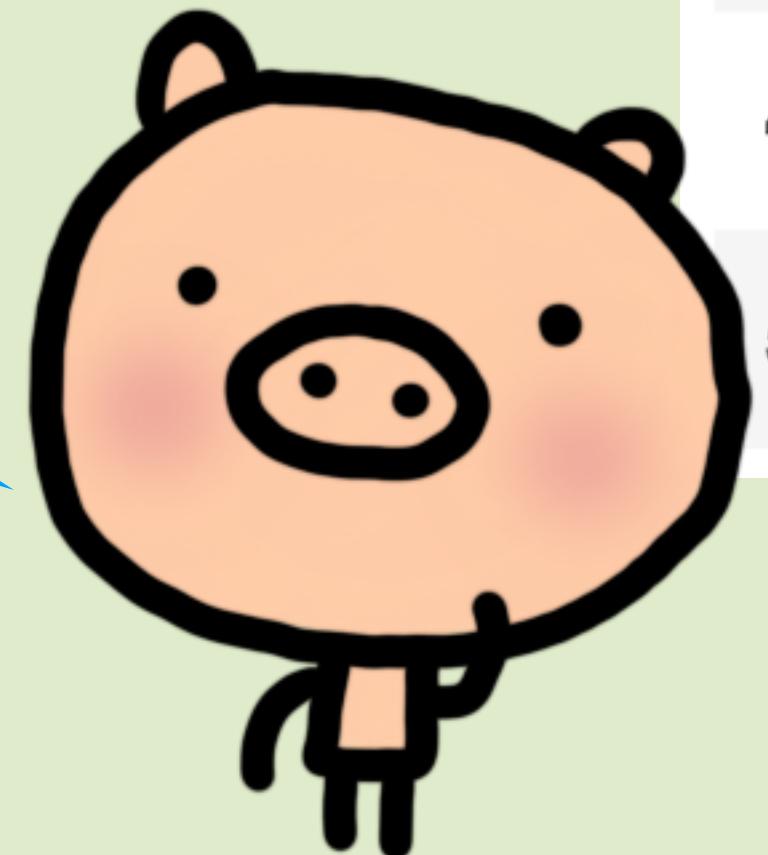
## 重點

# 刪除行

聽說不可以有總級分，那我們就刪了吧。

`df2.drop("總級分", axis=1)`

注意原本 df2  
並沒有被改變！



	姓名	國文	英文	數學	自然	社會	加權
1	吳志遠	13	15	15	8	8	58
2	胡淳茜	12	15	14	13	11	55
3	詹威德	12	13	15	10	14	55
4	蔡亦瑄	13	13	14	13	12	54
5	李士賢	10	14	15	13	13	54

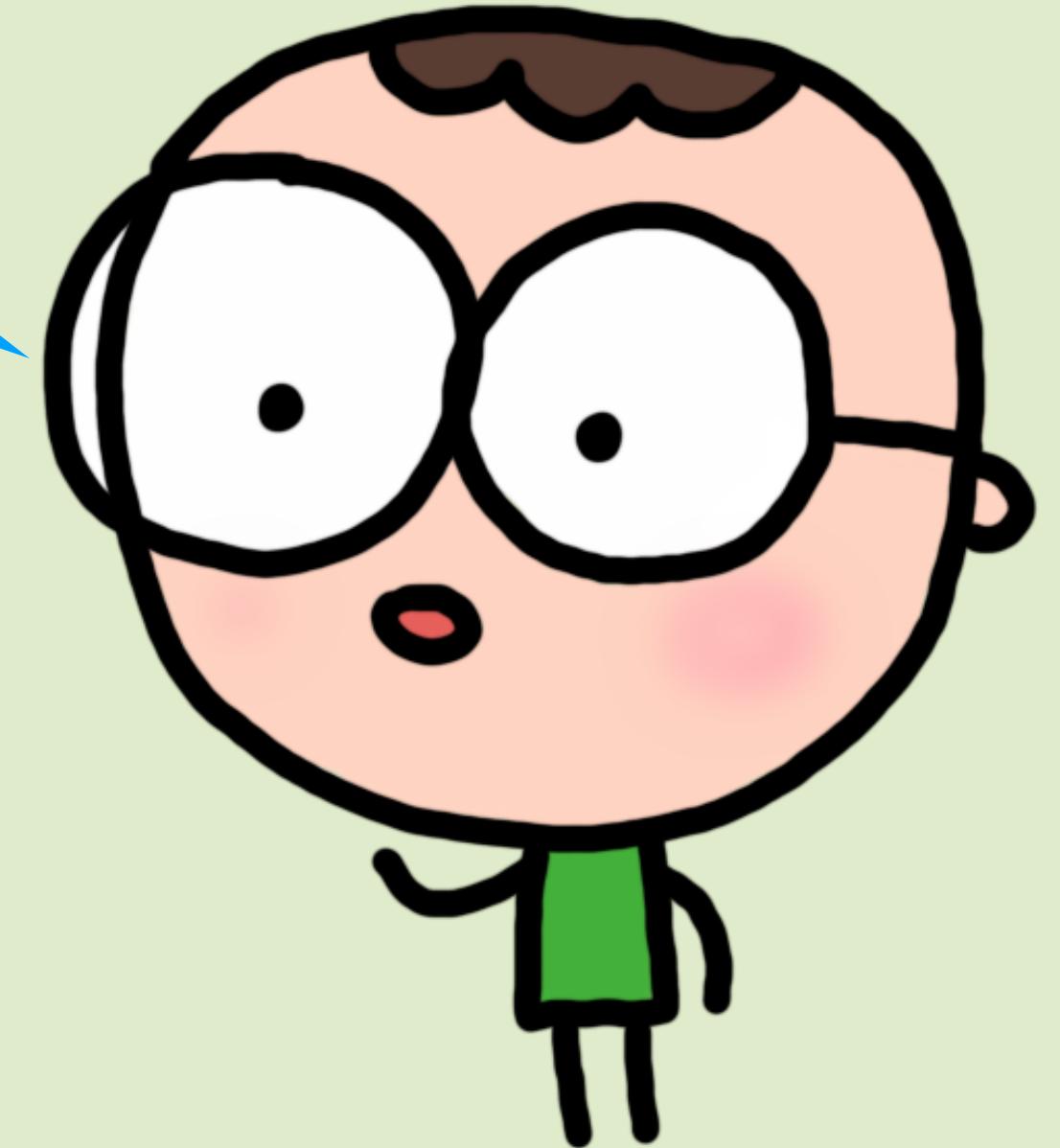
重點

## 刪除行

許多情況下我們都要加入這個, 原來的 DataFrame 才會被改變。

**inplace=True**

```
df2.drop("總級分", axis=1, inplace=True)
```



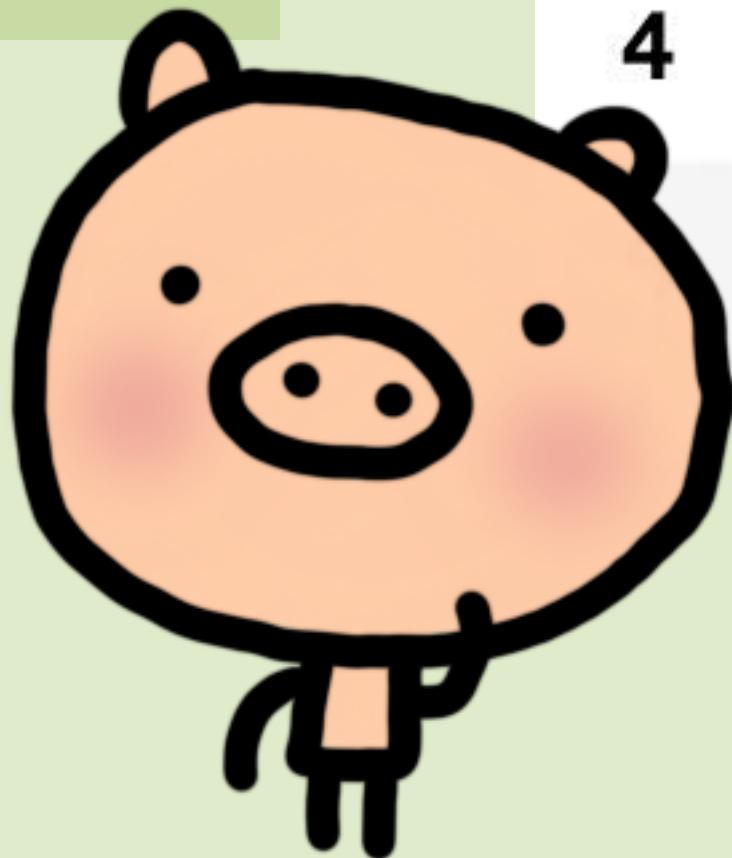
## 重點

# 刪除列

第 5 號同學聽說到別的學校去了，所以我們刪除他。

`df2.drop(5)`

但這有點容易刪錯人...



	姓名	國文	英文	數學	自然	社會	加權
1	吳志遠	13	15	15	8	8	58
2	胡淳茜	12	15	14	13	11	55
3	詹威德	12	13	15	10	14	55
4	蔡亦瑄	13	13	14	13	12	54
	童心怡	13	13	14	8	13	54

## 重點

# 刪除列

我們也可找出符合條件的，刪掉那一列。



找到了符合條件  
的，找出 `index`。

```
df2.drop(df2[df2.姓名=="李士賢"].index)
```

	姓名	國文	英文	數學	自然	社會	加權
1	吳志遠	13	15	15	8	8	58
2	胡淳茜	12	15	14	13	11	55
3	詹威德	12	13	15	10	14	55
4	蔡亦瑄	13	13	14	13	12	54
6	童心怡	13	13	14	8	13	54

## 重點

# 刪除列

相信你也發現,直接留下不合條件的就好...

```
df2[df2.姓名!="李士賢"]
```

	姓名	國文	英文	數學	自然	社會	加權
1	吳志遠	13	15	15	8	8	58
2	胡淳茜	12	15	14	13	11	55
3	詹威德	12	13	15	10	14	55
4	蔡亦瑄	13	13	14	13		
6	童心怡	13	13	14	8		



例子

# 用 Pandas 來玩股票

終端機

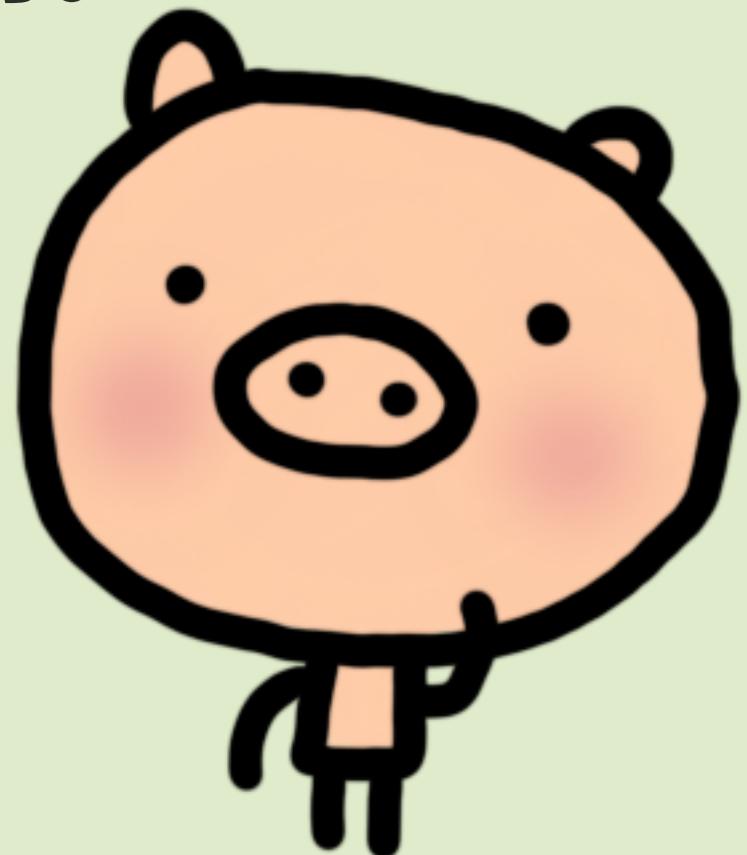
以前沒裝過。

```
conda install pandas-datareader
```

安裝過但很久沒更新。

```
conda update pandas-datareader
```

有個從 Pandas 獨立出來的套  
件叫 **pandas-datareader**,  
幾經波折, 不過至少現在看來  
Yahoo! 還可以使用。



例子

## 讀入 Apple 股價資料

```
import pandas_datareader as pdr  
  
df = pdr.get_data_yahoo('AAPL')
```

我們準備用這筆資料學習基本資料分析常見操作

這也太簡單了！



## 練習1

# 只要最後 300 期的資料

我們的資料太多了，現在我們想拿**最近 300 個交易日**的資料來分析。

把 `df` 重設為只有**最近 300 個交易日**的資料。

複習一下。



## 練習1

# 只要最後 300 期的資料

```
df = df[-300:]
```

最近 300 個交易日的資料

做出來了嗎？



## 重點

# 移動平均

雖然有時有特別好、特別壞的表現，但傳說很多東西都被「打回原型」。

比如說好學生可能會有一次考不好，普通的同學可能有一次有出人意料的佳績；爛球隊可能會不小心贏冠軍隊。但不管怎樣都會回到平均線。

好消息是平均線也是有可能變的！

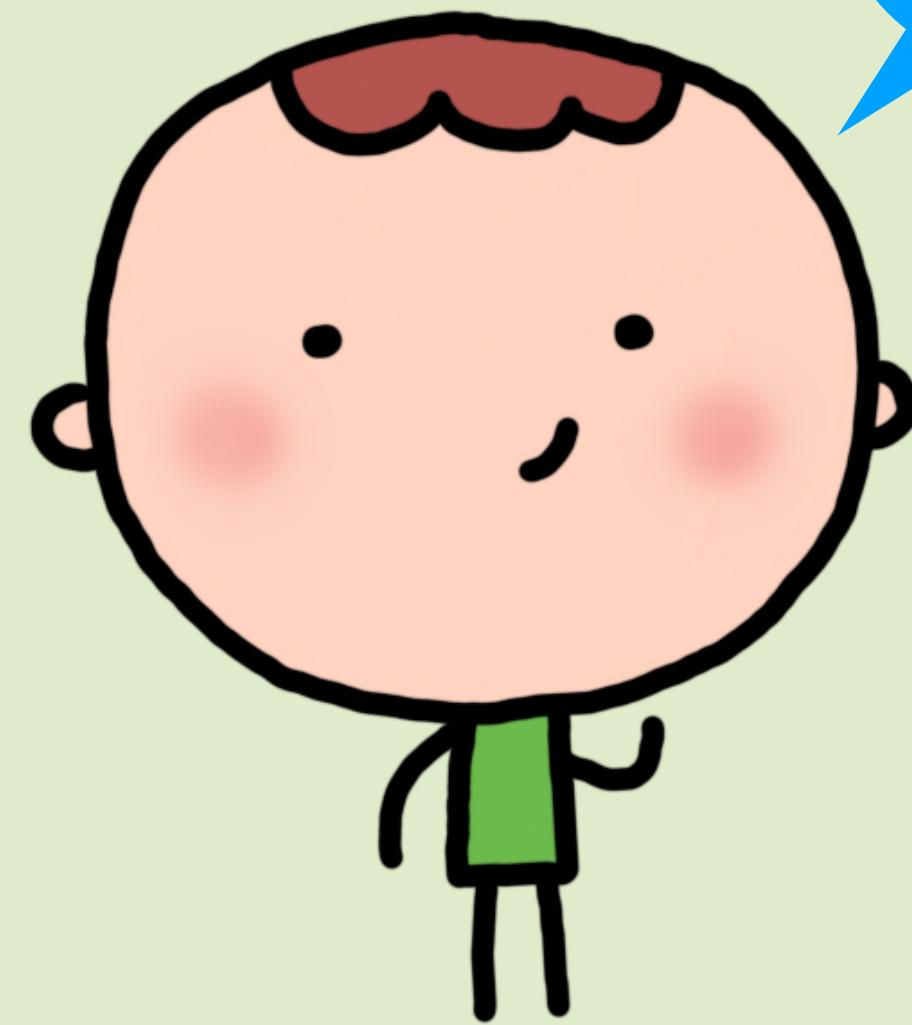


重點

## 移動平均

n 期的移動平均 (Moving Average)

$$\text{MA}_t = \frac{x_{t-n+1} + x_{t-8} + \cdots + x_t}{n}$$

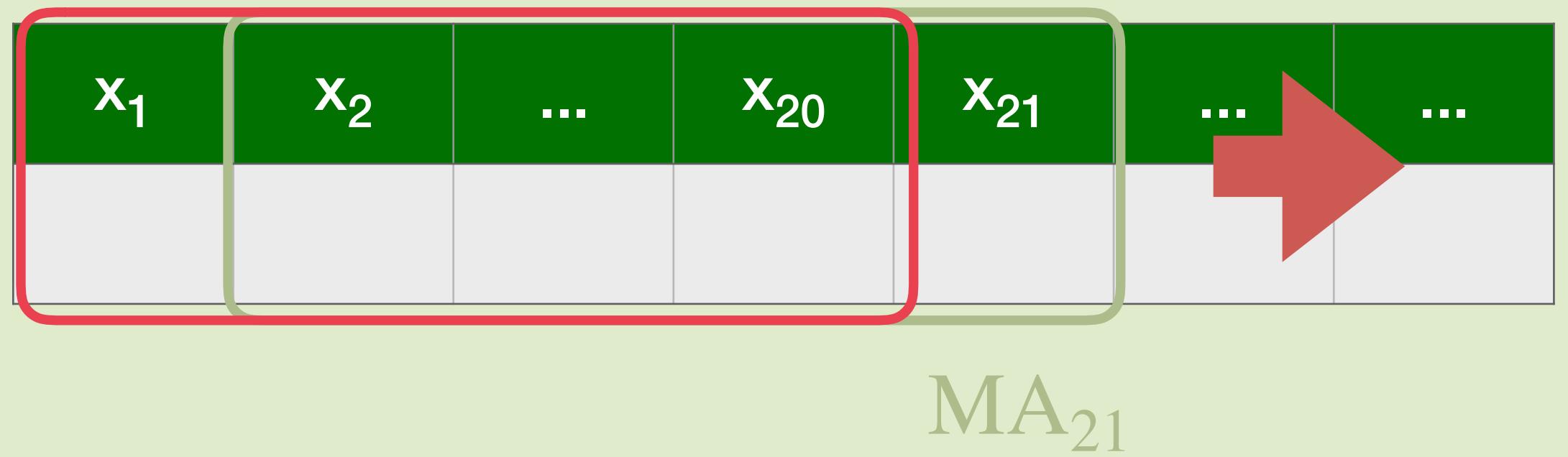


這事實上是簡單移動平均。

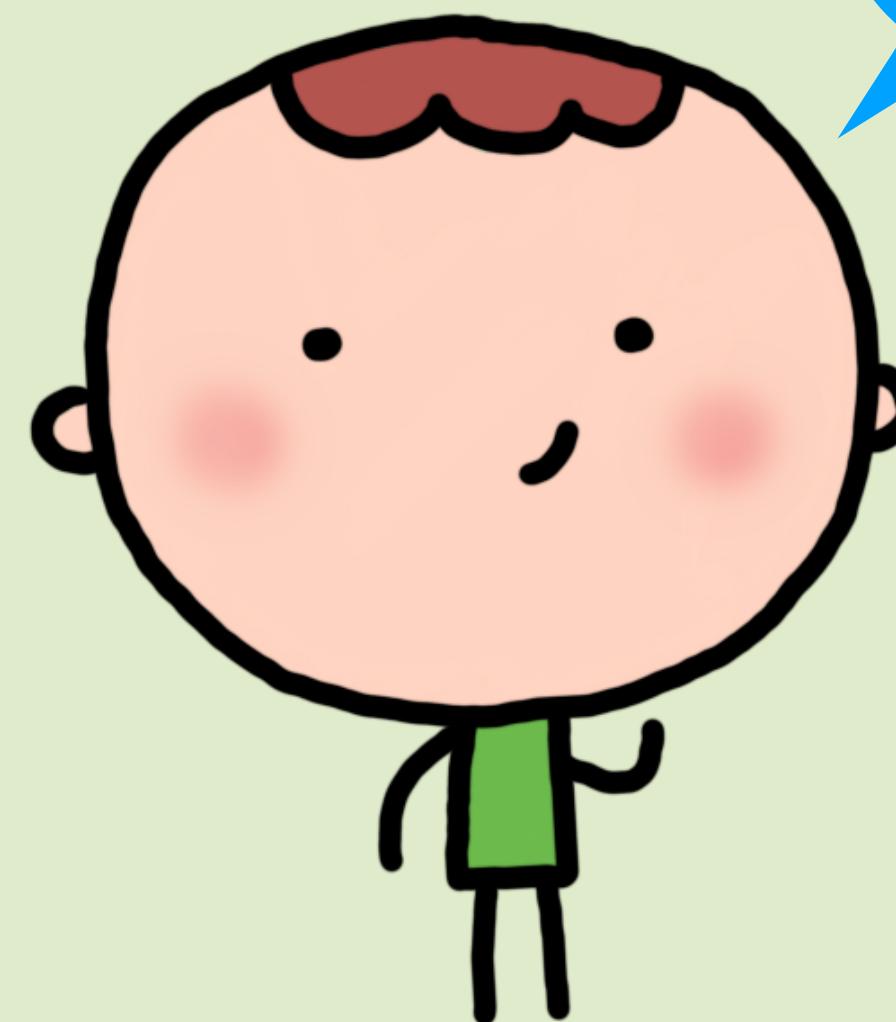
重點

# 設定 window 大小

$MA_{20}$



先設好一次要  
看多少範圍。



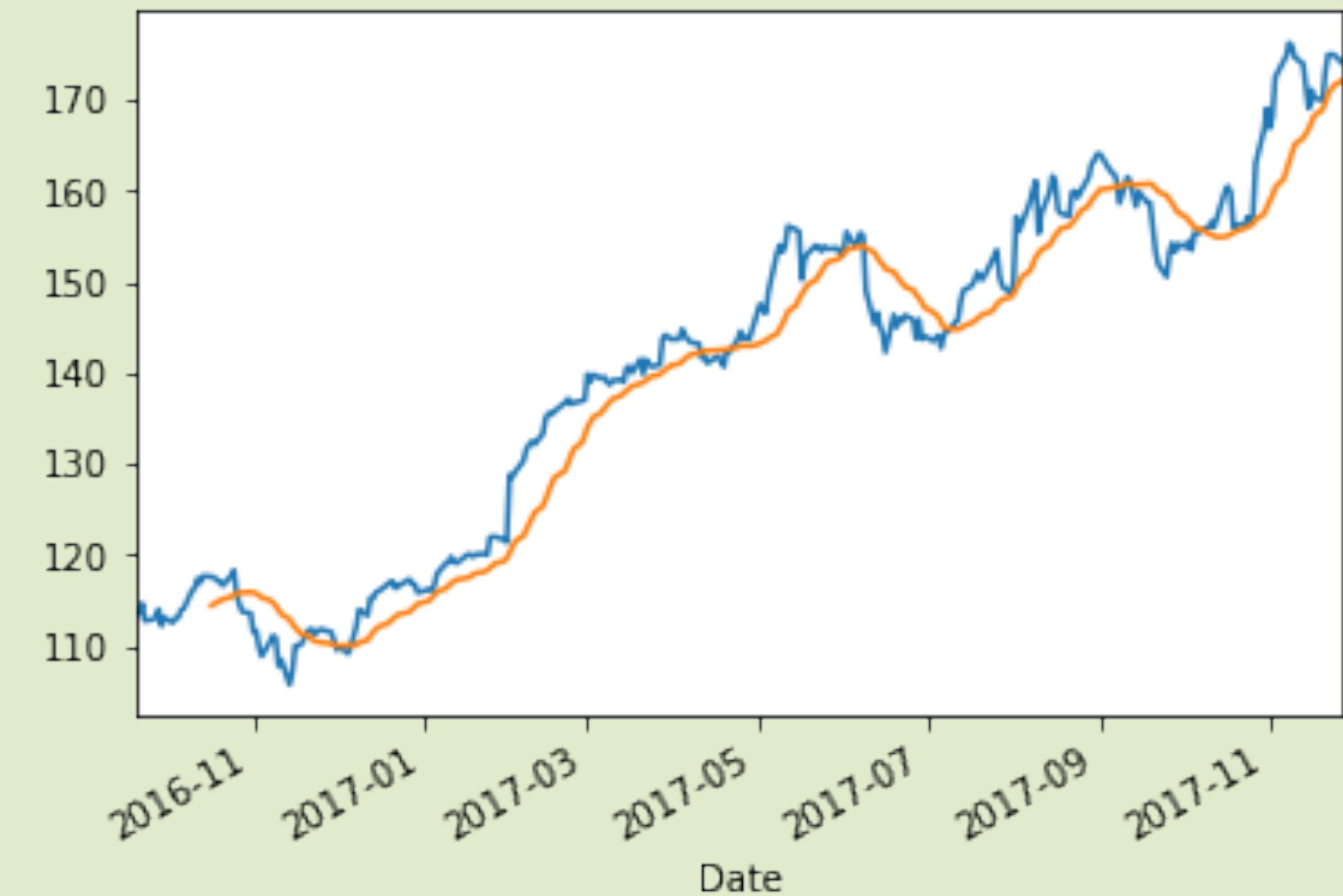
`df.Close.rolling(20)`

## 練習

# 收盤價格及其移動平均線

```
df.Close.plot()  
df.Close.rolling(20).mean().plot()
```

我們比較一下收盤價格和移動平均線的關係。



## 練習

# 收盤價格及其移動平均線

變本加厲，除了 20 期的移動平均，再加上 60 期的移動平均。

圖例要顯示

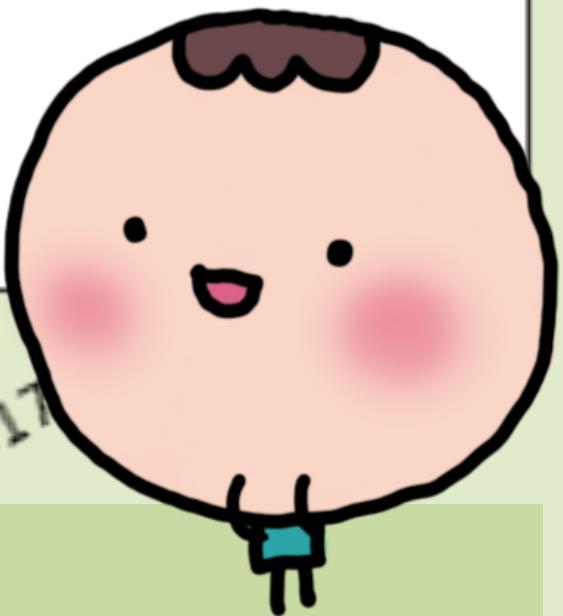
```
df.Close.plot(legend=True)
```

```
df.Close.rolling(20).mean().plot(label="$MA_{20}$", legend=True)
```

```
df.Close.rolling(60).mean().plot(label="$MA_{60}$", legend=True)
```

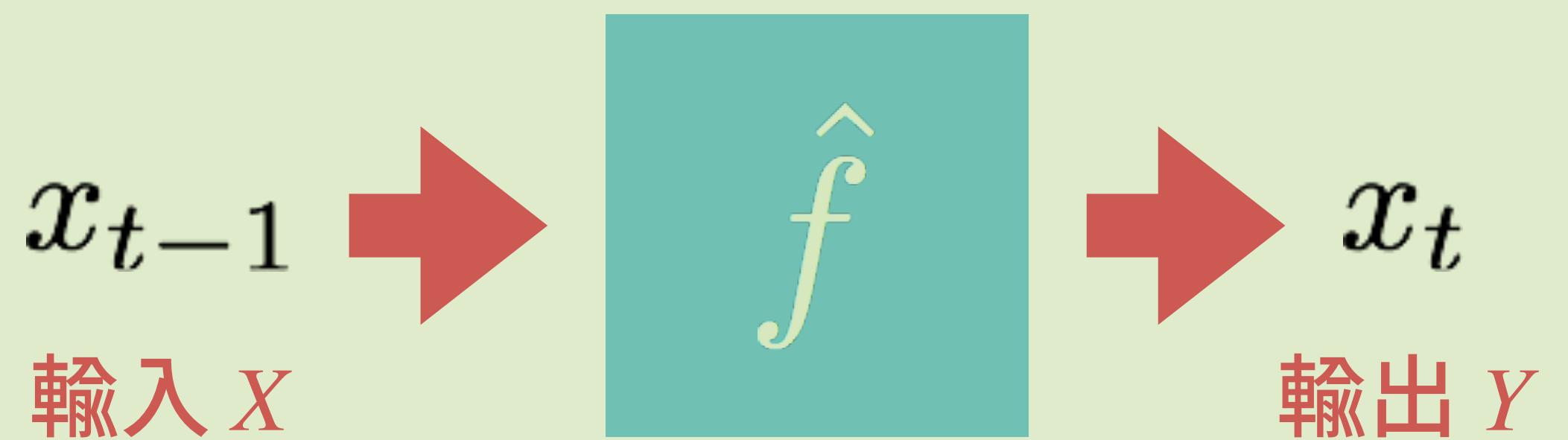


支援 LaTeX!



## 練習

# 準備做預測



我們準備做預測, 用一個非常天真的模型, 也就是輸入上一期的收盤價, 決定下一期收盤價!

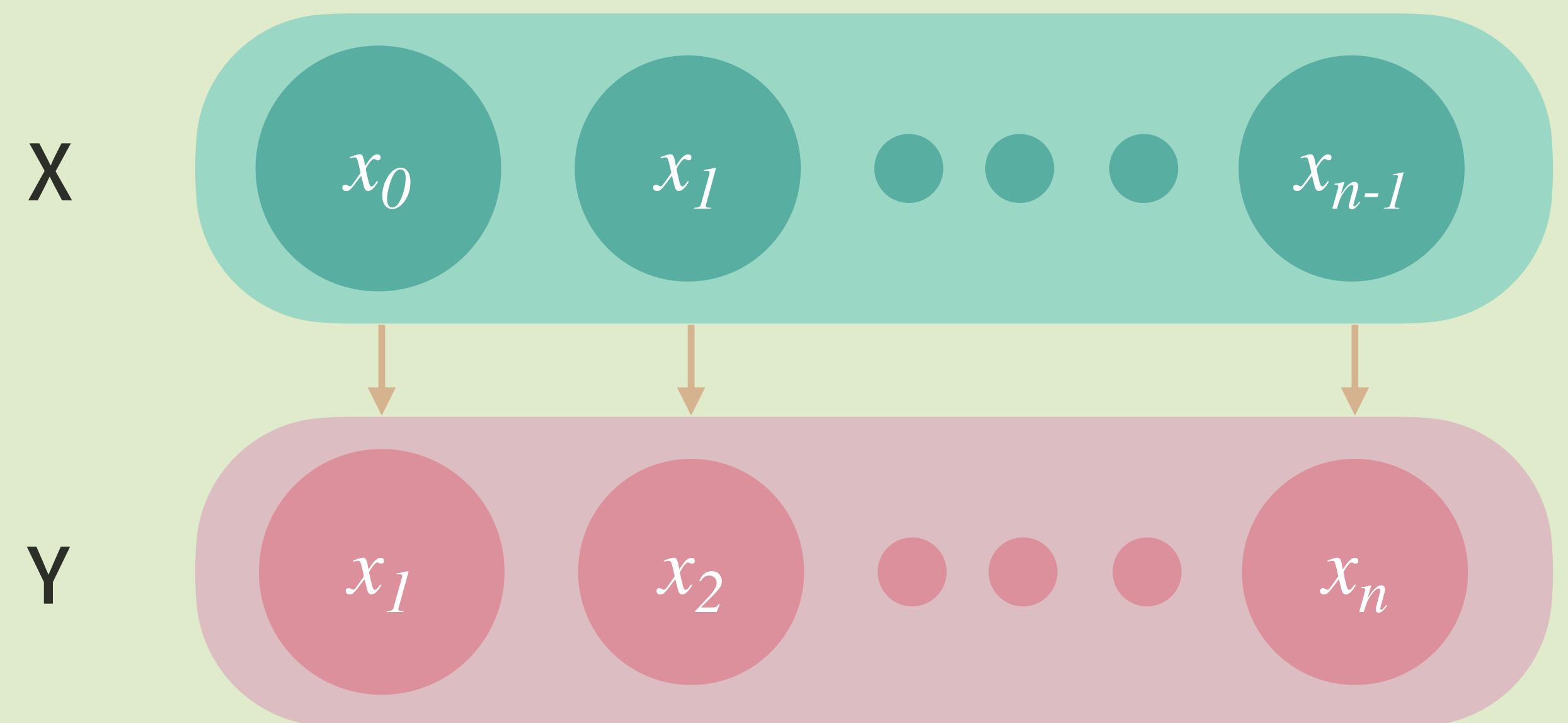
而且, 小豬說...

我覺得是線性的!



練習

# 準備「訓練資料」



練習

## 把要分析的重點轉 NumPy array

```
adj_close = df.Close.values
```

接下來，怎麼化成我們要的訓練資料呢？



## 練習

# 準備訓練資料



再來要看看是不是  
一副「線性樣」。

x = adj\_close[:-1]

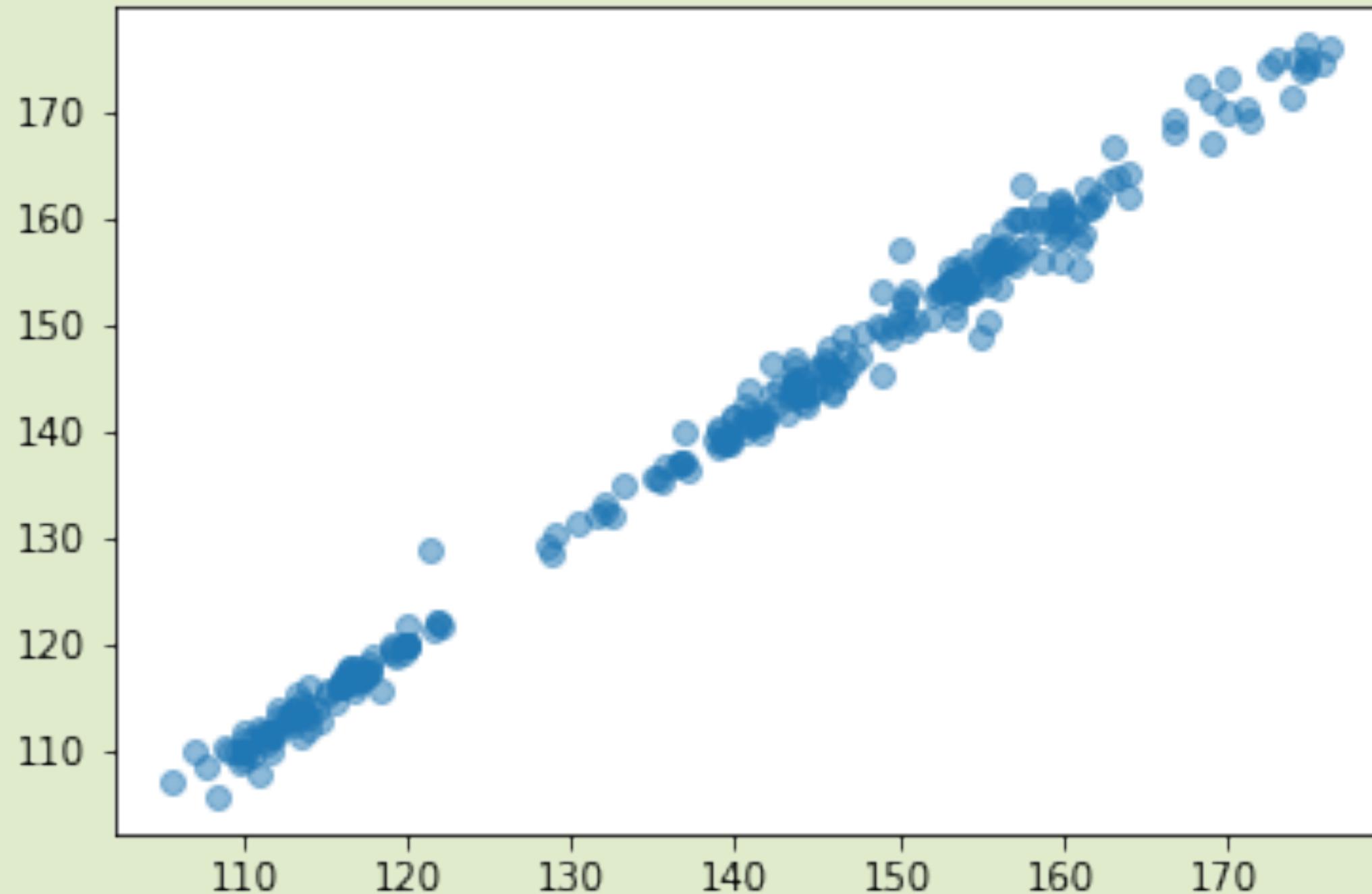
y = adj\_close[1:]

最後一筆不要!

第一筆不要!

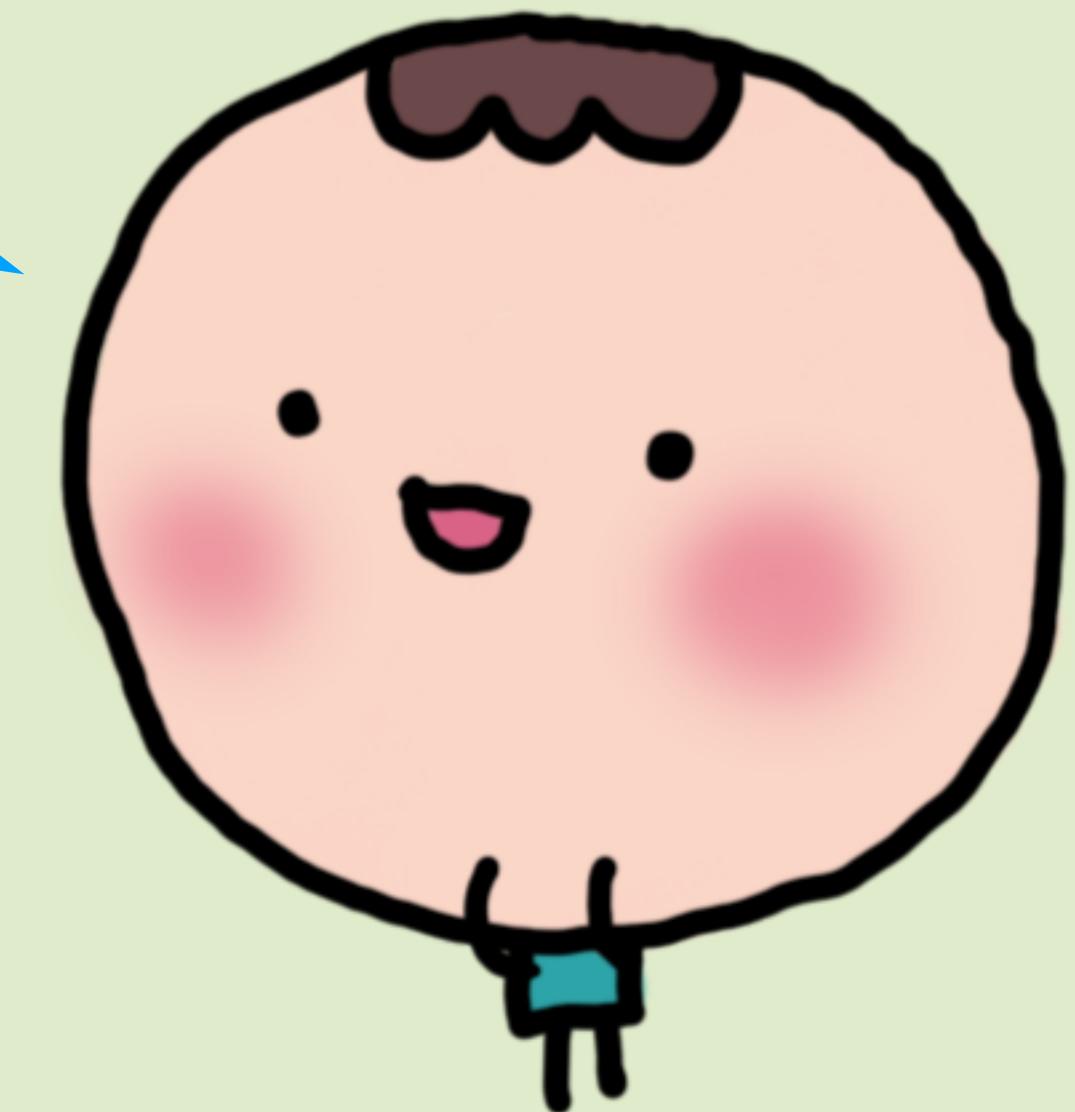
練習

畫圖



```
plt.scatter(x, y, alpha=0.5)
```

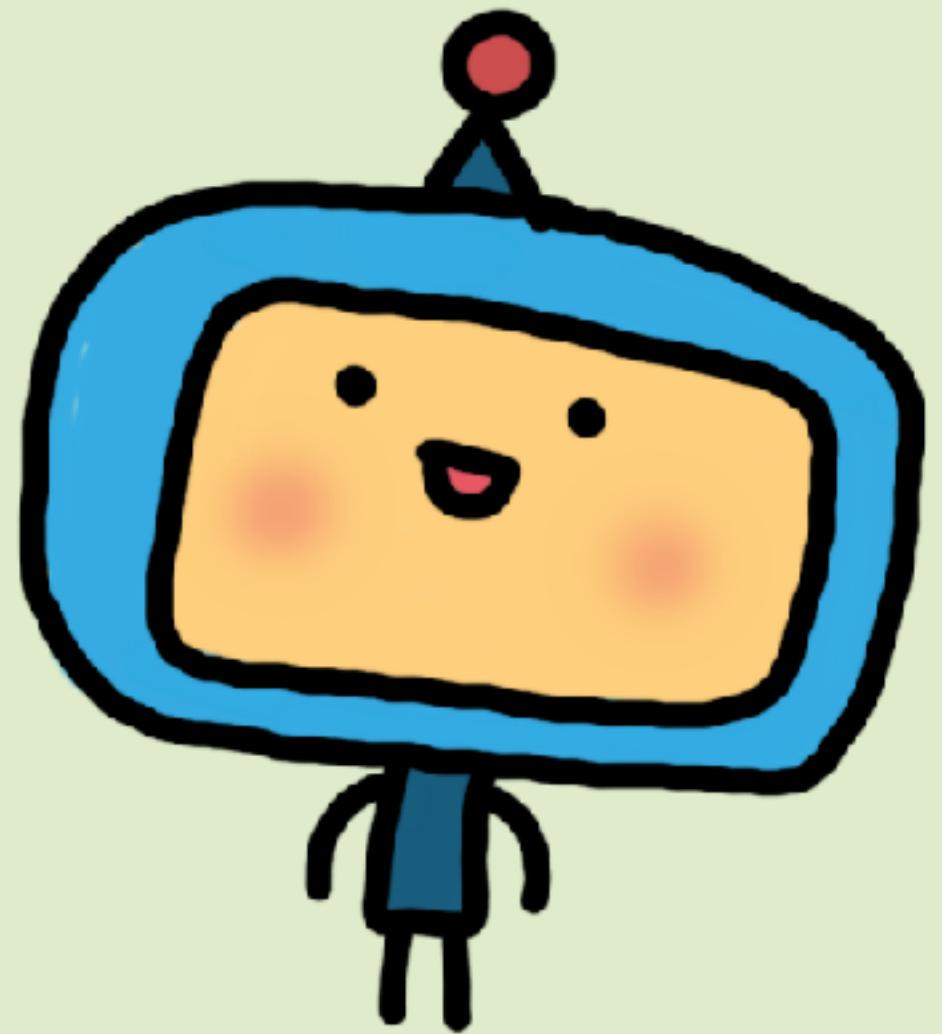
有像耶! 那我們可以來線性迴歸了!



5

# SciKit-Learn

輕輕鬆鬆做機器學習



- David Cournapeau 是京都大學資訊科學博士
- Scikit-Learn 最初是 2007 年的一個Google Summer of Code 專案
- 同年 Matthieu Brucher 繼續這個專案，為他論文的一部份
- 後由在 INRIA (French Institute for Research in Computer Science and Automation) Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort 及 Vincent Michel 等人領導，推出公眾版本

迴歸

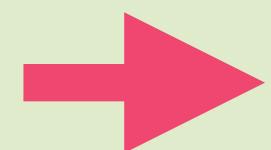
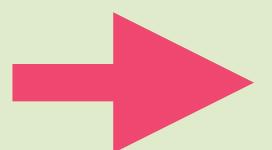
機器學習

其實就是用不同的方法學個函數

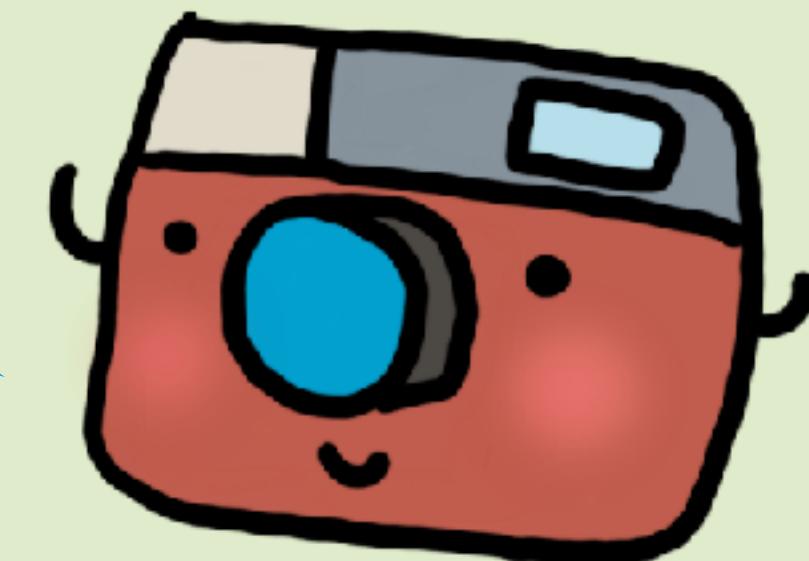
深度學習

例子

## 想問的問題化成函數

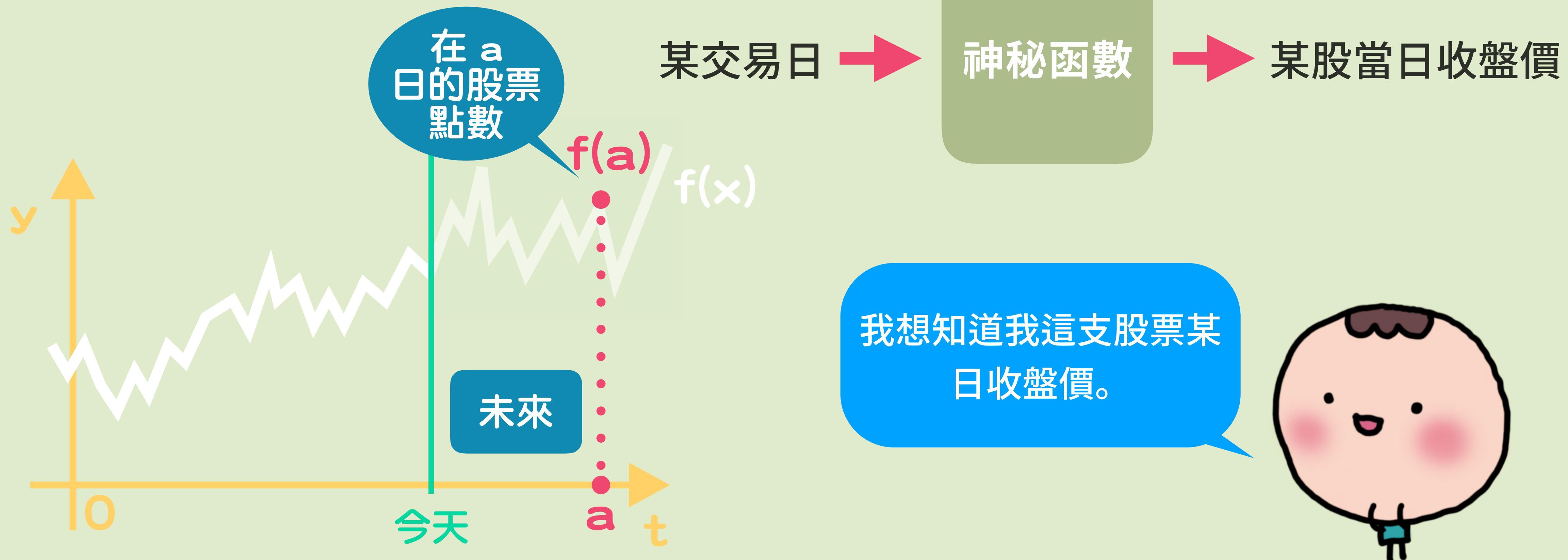


我想知道照片裡的動物  
是什麼？



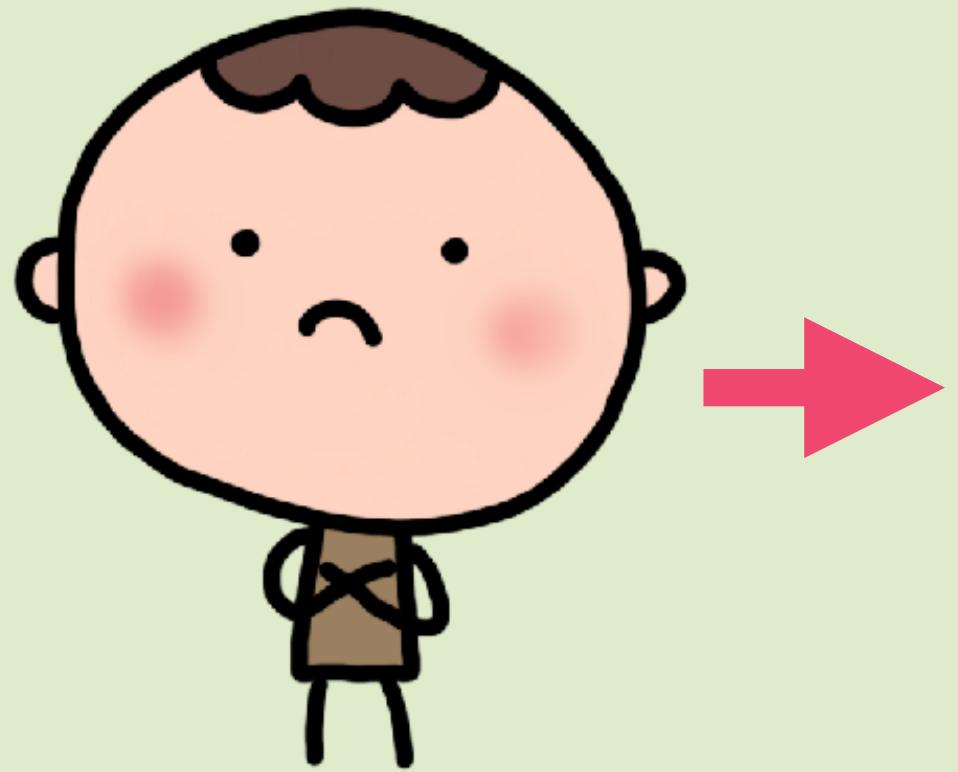
例子

## 想問的問題化成函數



例子

## 想問的問題化成函數

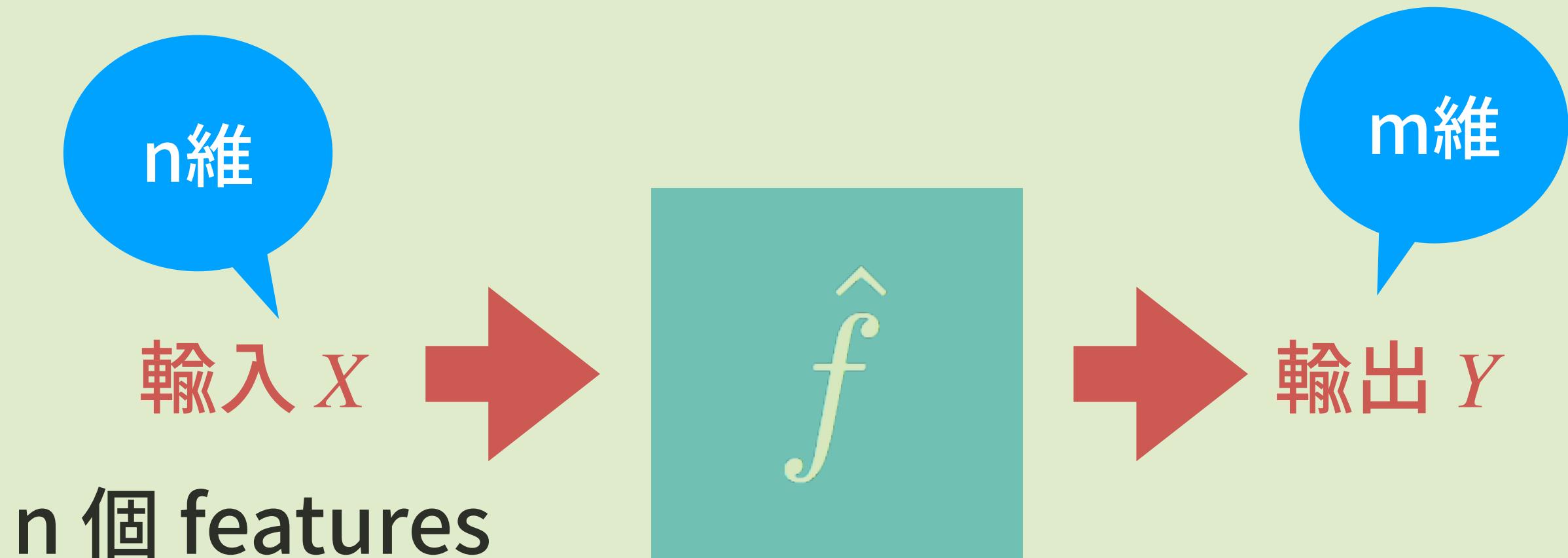


神秘函數

- A 有推有買
- B 有推可能買
- C 再推也不買

這顧客是哪一型?

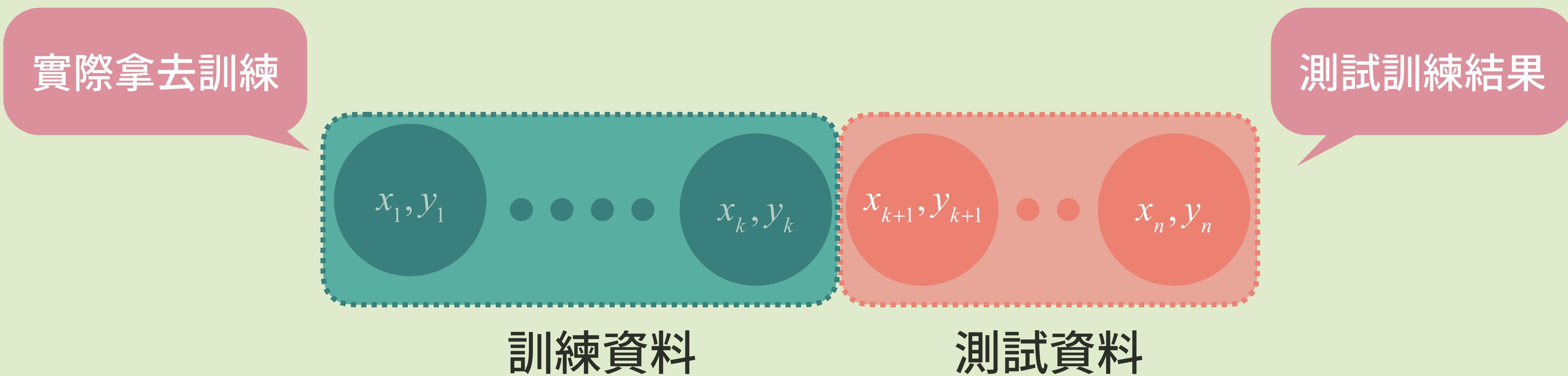


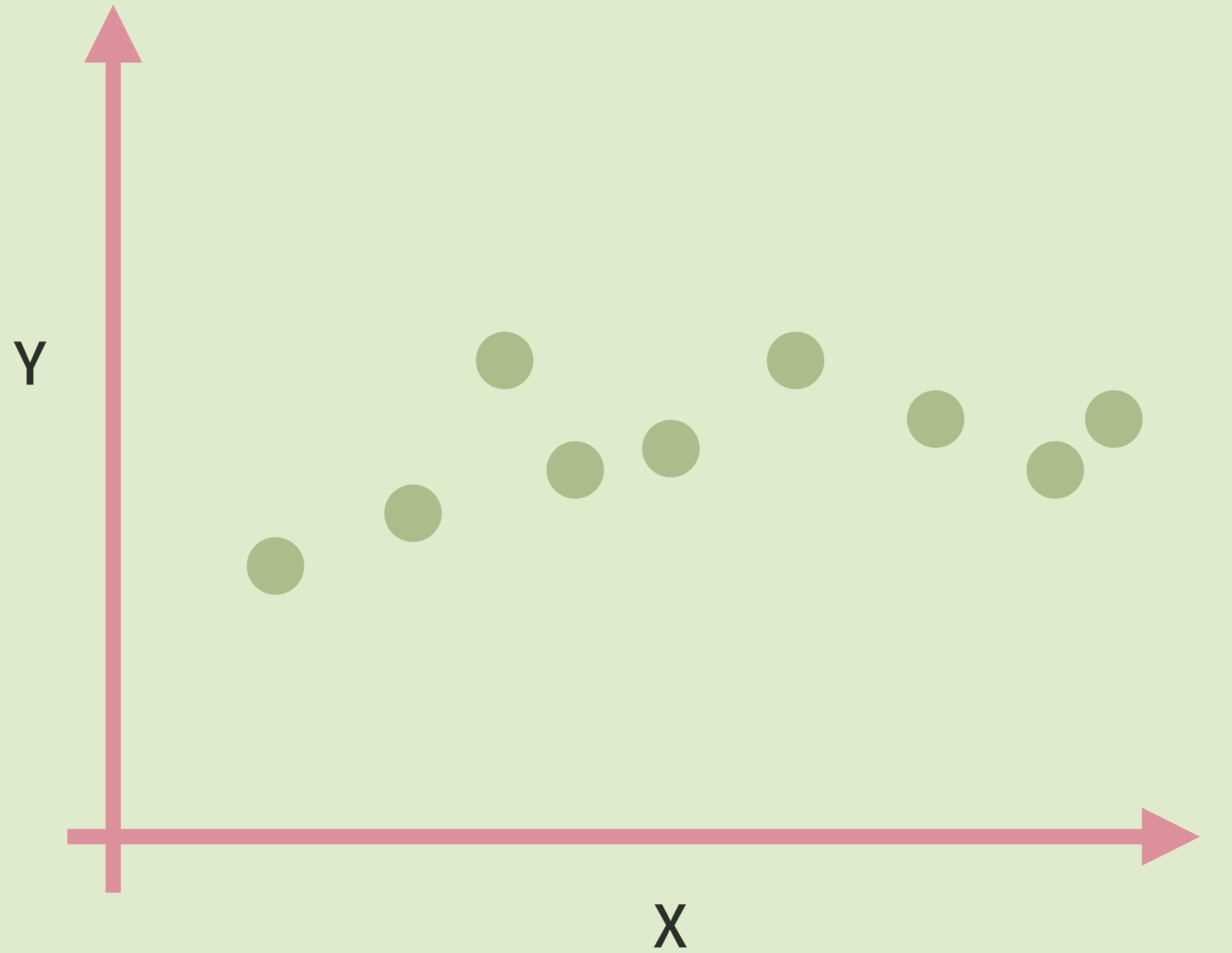


總之，把我的問題化成函數，然後  
再用各種手法把函數找出來！

# 資料區分為兩部份

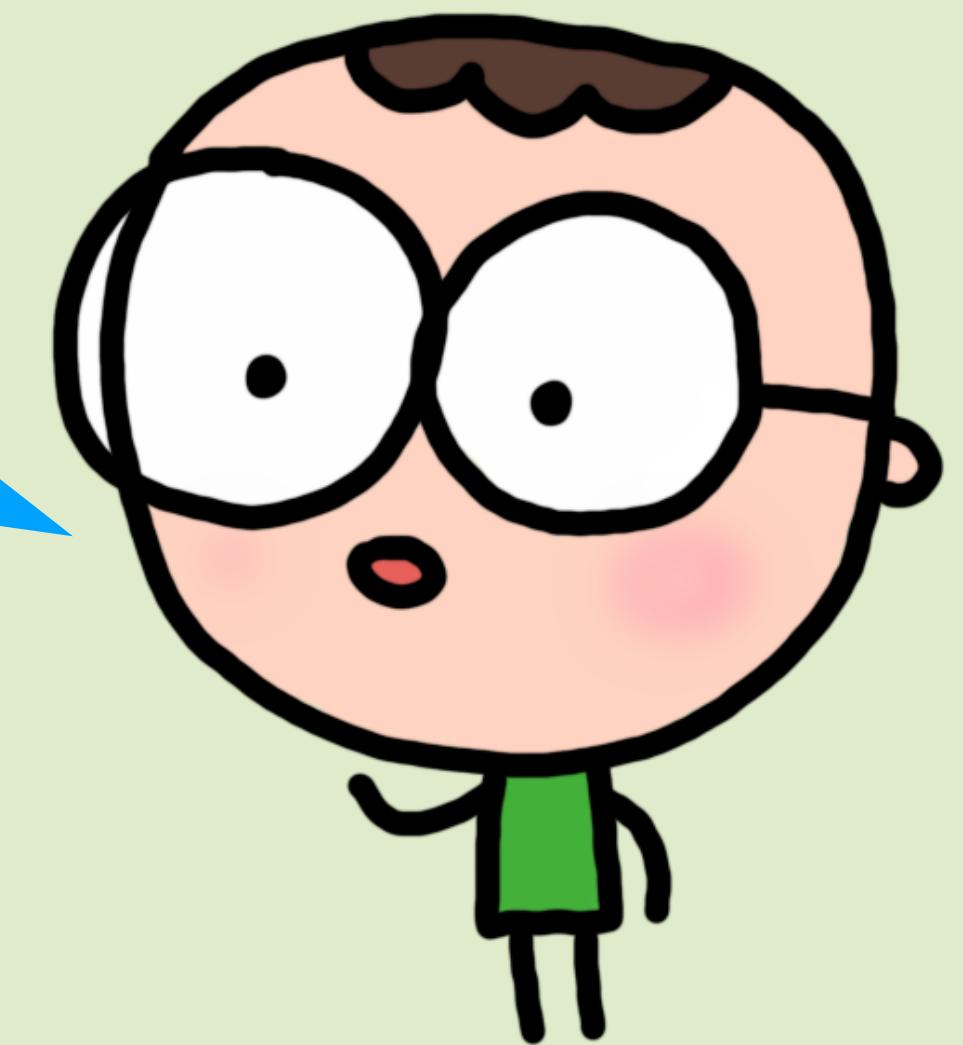
我們把所擁有的「歷史資料」，白話文說就是  
把我們知道正確答案的資料分成兩部分。





我們就是要去由歷史資料去學個函數，基本上「越像越好」。但是，「太像不一定好」...

要小心  
overfitting!



# 線性迴歸

## 簡單強大的函數學習法

例子

## 假的資料真迴歸

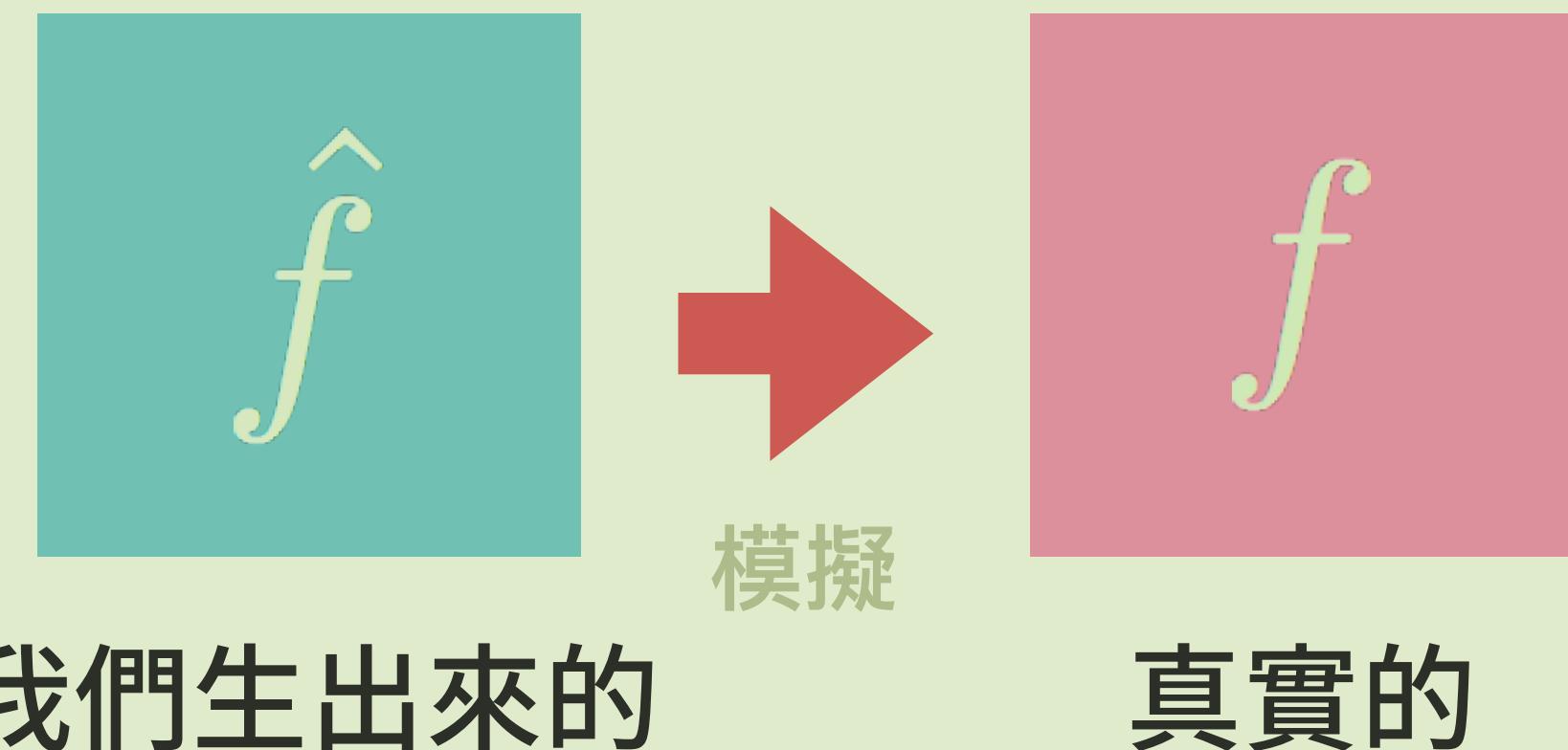
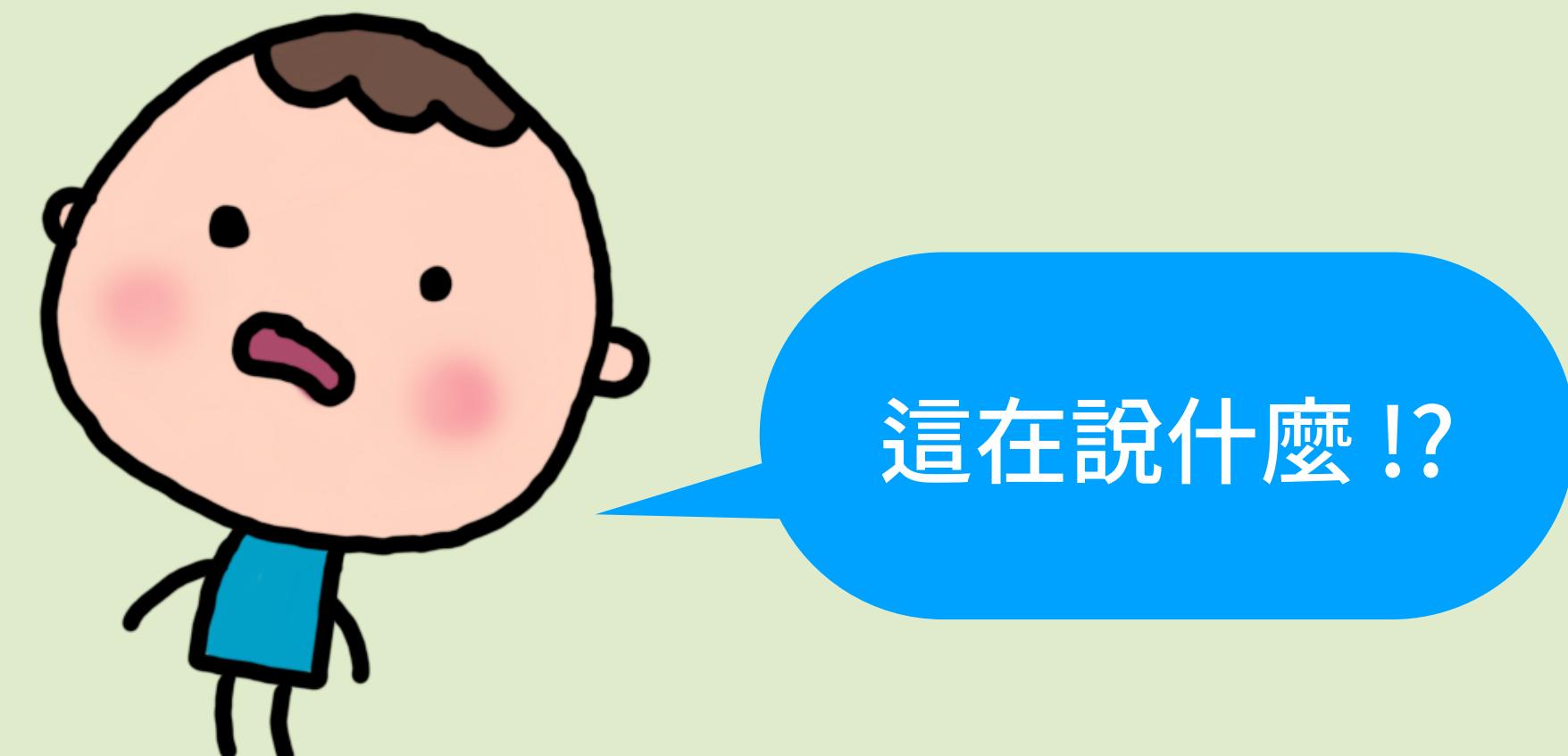
真實世界裡的函數  $f$ , 我們想生出個函數去模擬它。

我們有一堆歷史資料

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$$

有趣的是, 我們不一定有

$$f(x_i) = y_i$$



例子

## 假的資料真迴歸

$$\text{觀察值} = f(x) + \varepsilon$$

理想函數

雜訊

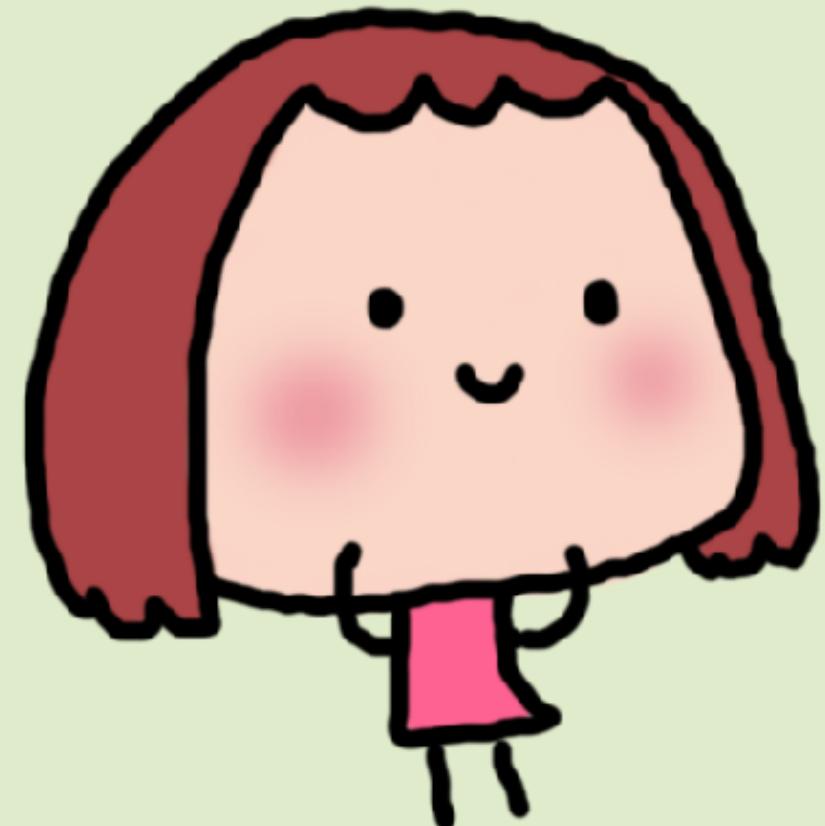
那是因為會有 noise

例子

## 假的資料真迴歸

假設理想函數長這樣

$$f(x) = 1.2x + 0.8$$



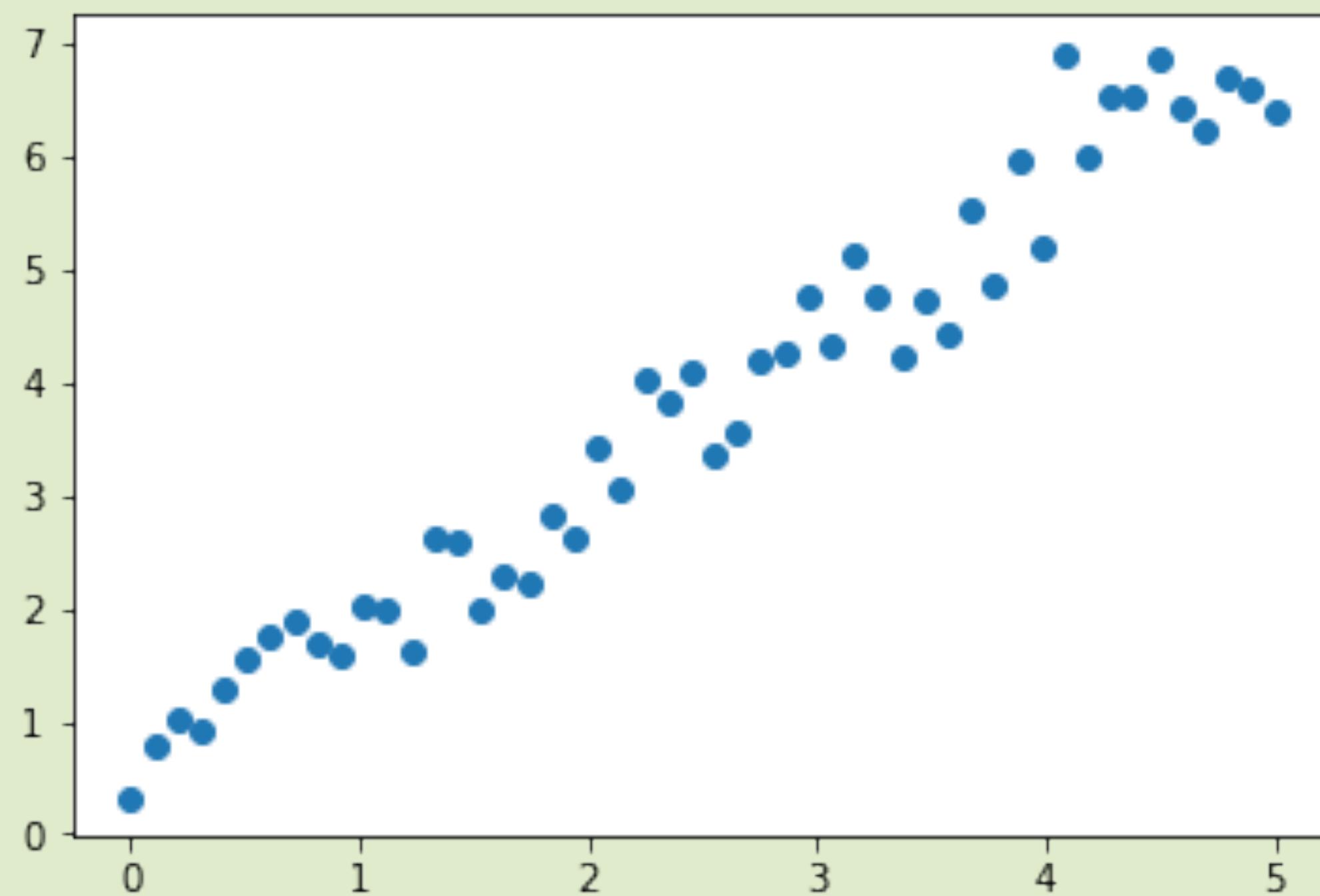
```
x = np.linspace(0, 5, 50)
y = 1.2*x + 0.8 + 0.4*np.random.randn(50)
```

雜訊

我們用程式模擬真實世界的樣子。

例子

## 假的資料真迴歸



`plt.scatter(x, y)`

真的一副線性樣！

(廢話)



# SciKit-Learn 登場!

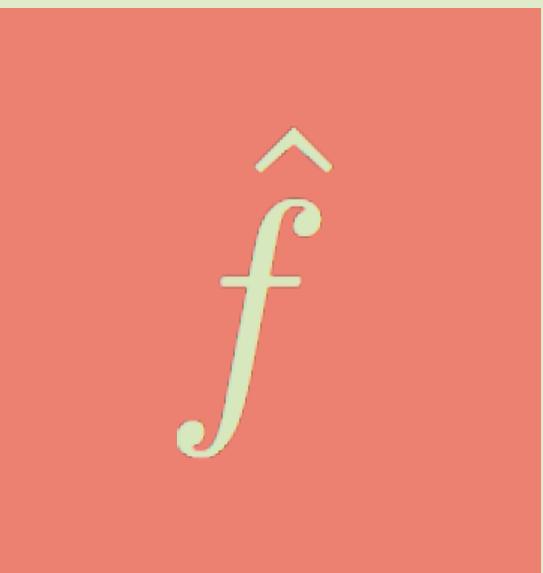
```
from sklearn.linear_model import LinearRegression
```

不管線性迴歸、機器學習，乃至未來  
神經網路，第一件事都是打開台「機  
器」。

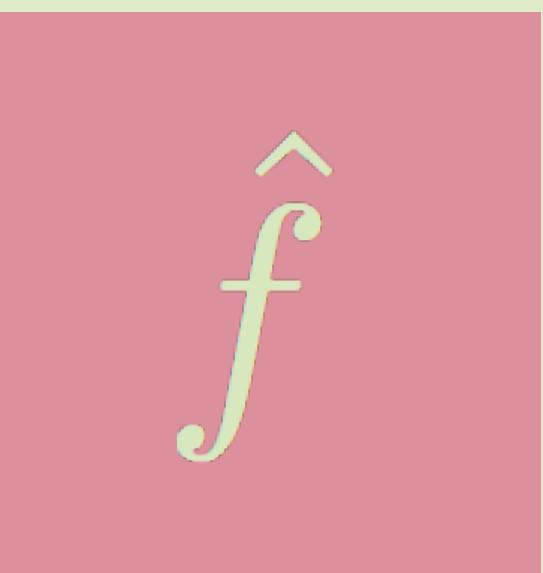
然後丟資訓練資料去訓練！



線性迴歸機



分類機



神經網路

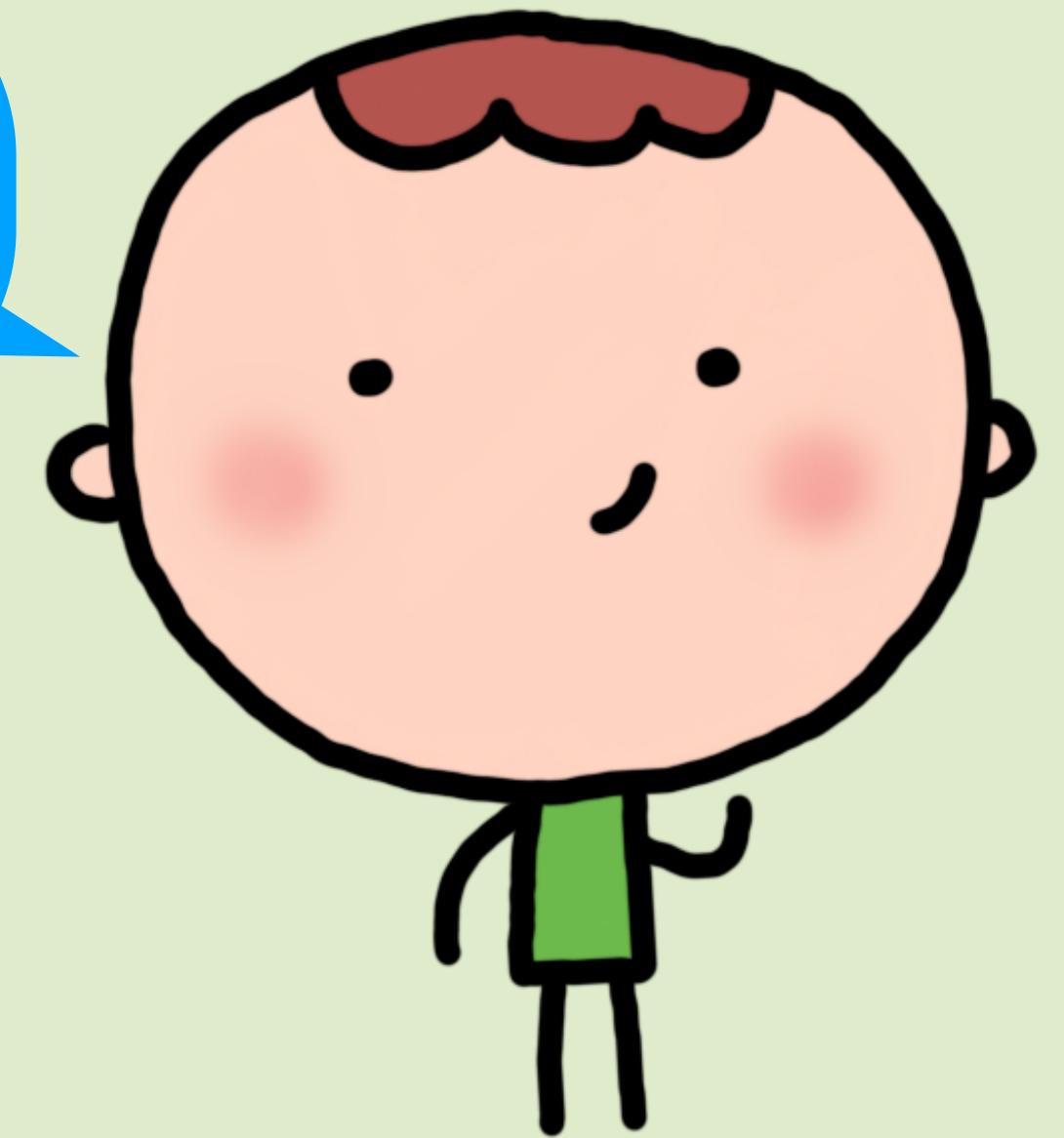
例子

## 假的資料真迴歸

打開線性迴歸機。

```
regr = LinearRegression()
```

可以開始訓練了！



例子

## 假的資料真迴歸

我們現在的  $x$  長這樣

$[x_1, x_2, \dots]$

但是我們可愛的學習機要求這樣

$[[x_1], [x_2], \dots]$

```
x = x.reshape(50, 1)
```

當然可以用我們教過的秘技  
`x.reshape(-1, 1)`

One More Thing



例子

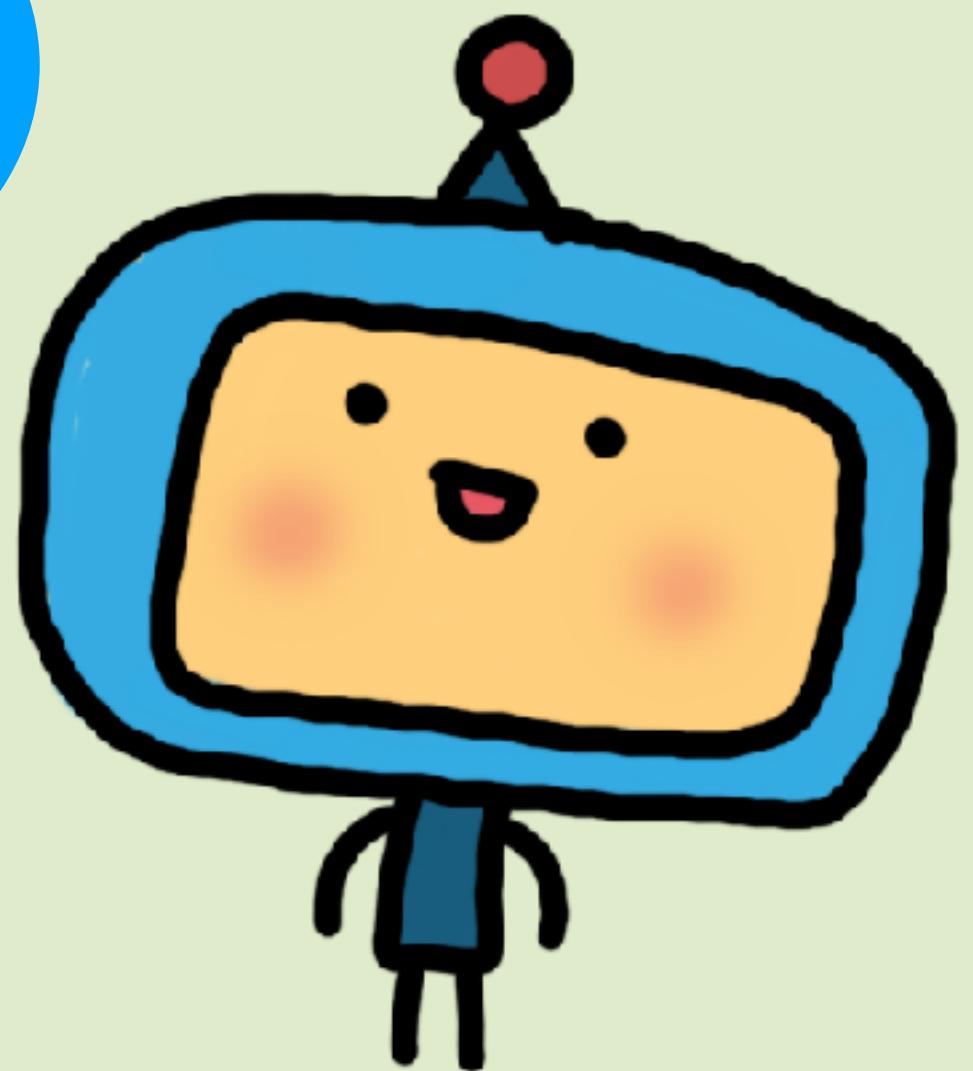
## 假的資料真迴歸

```
regr.fit(x, y)
```

輸入

正確答案

訓練!

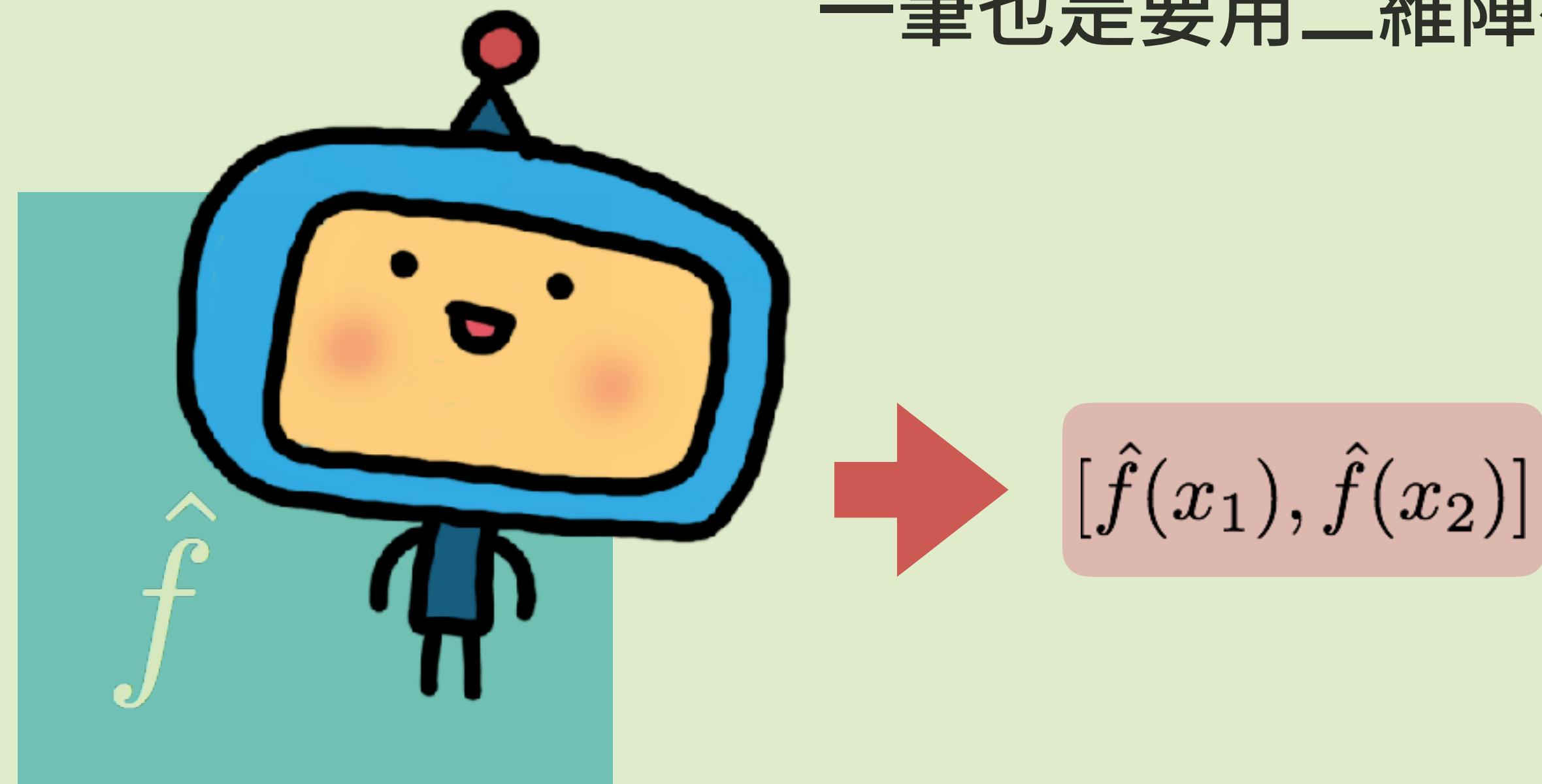


例子

## 假的資料真迴歸

開始送入我們學成的迴歸機預測! 注意輸入格式, 就算只輸入一筆也是要用二維陣列。

$[[x_1], [x_2]]$



`regr.predict`

例子

## 假的資料真迴歸

```
regr.predict([[2.8]])
```

輸出: array([ 4.14767509])

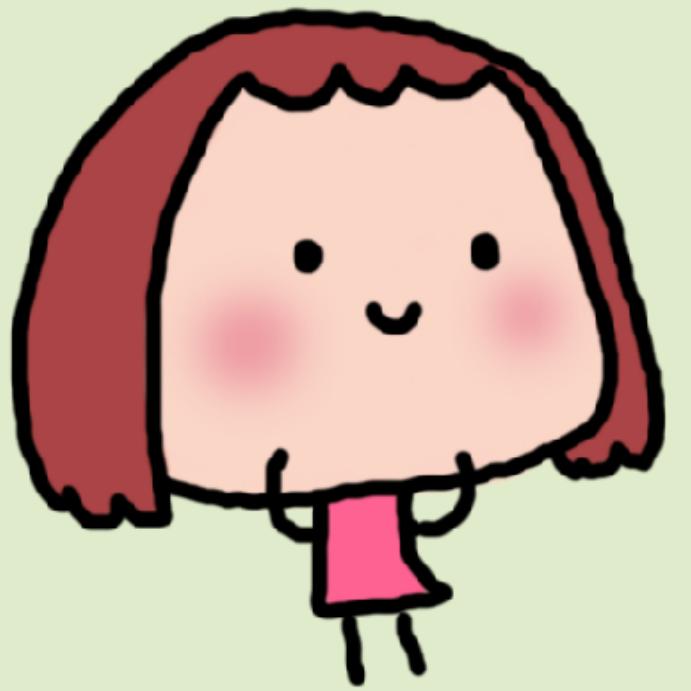
```
regr.predict([[1.3], [3.2]])
```

輸出: array([ 2.24486677, 4.65509065])



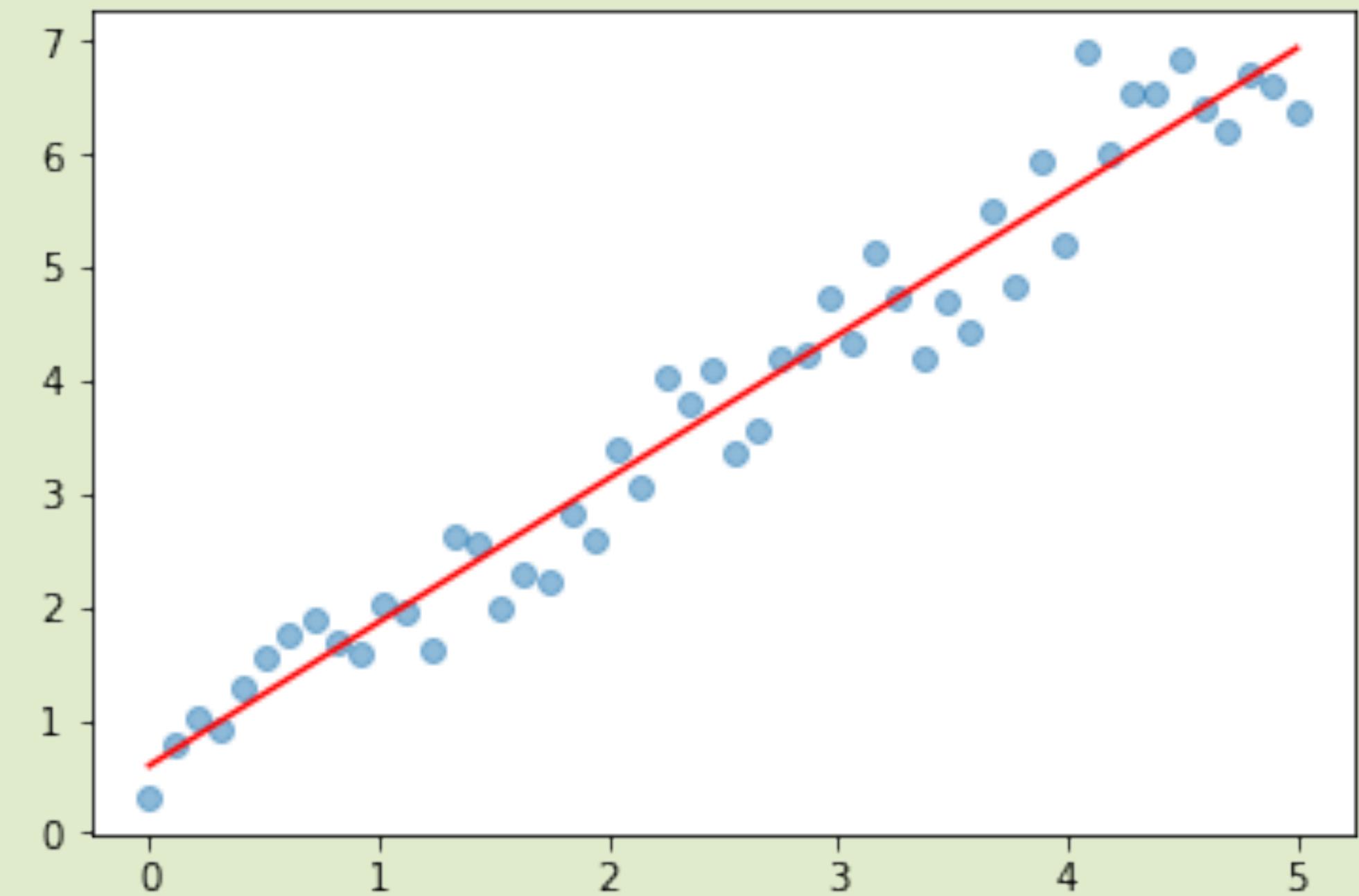
例子

## 假的資料真迴歸



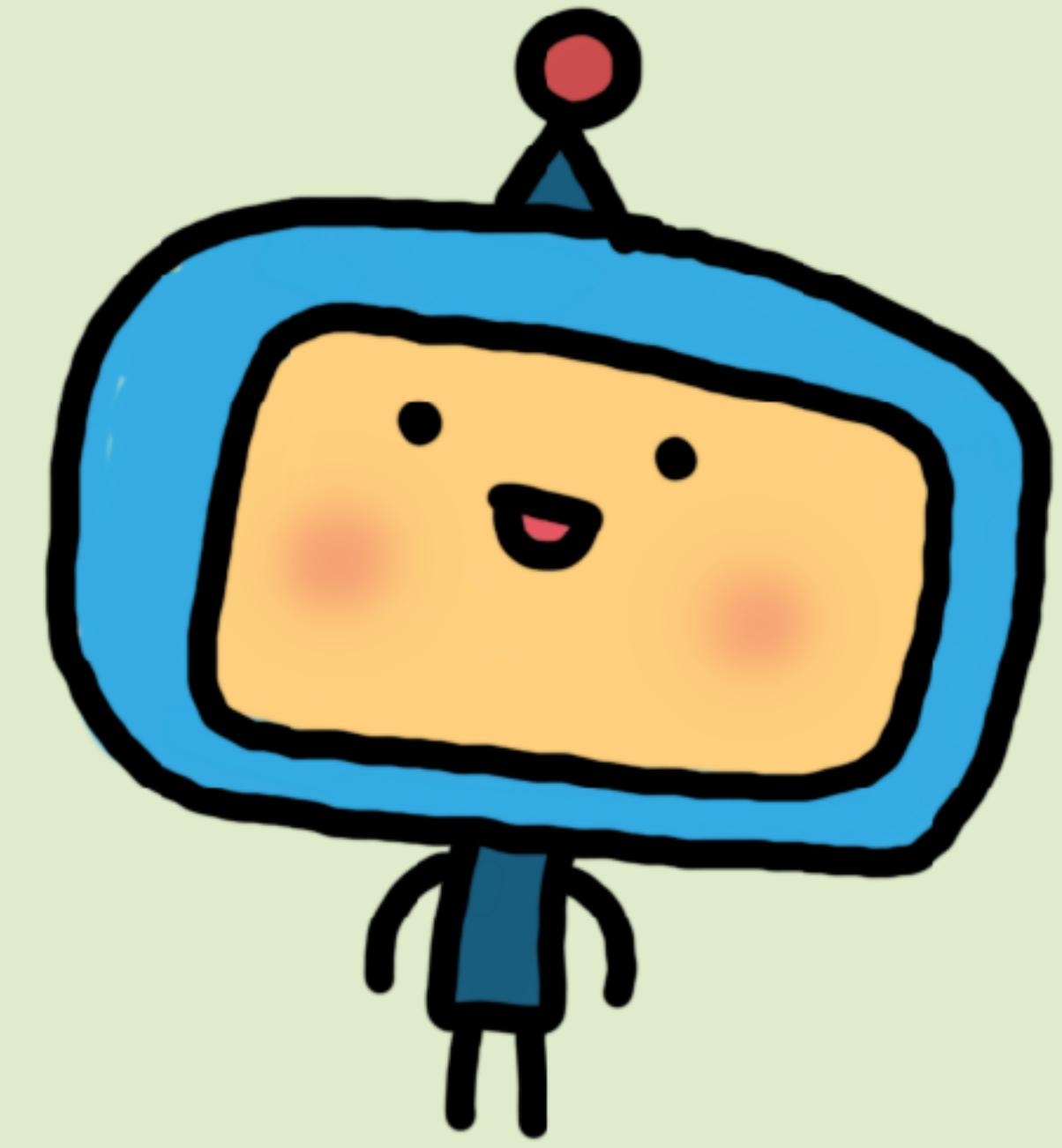
我們把原始資料和學出來的函數比較一下。

```
plt.scatter(x,y,alpha=0.5)  
plt.plot(x, regr.predict(x), 'r')
```



# 迴歸分析

股價預測

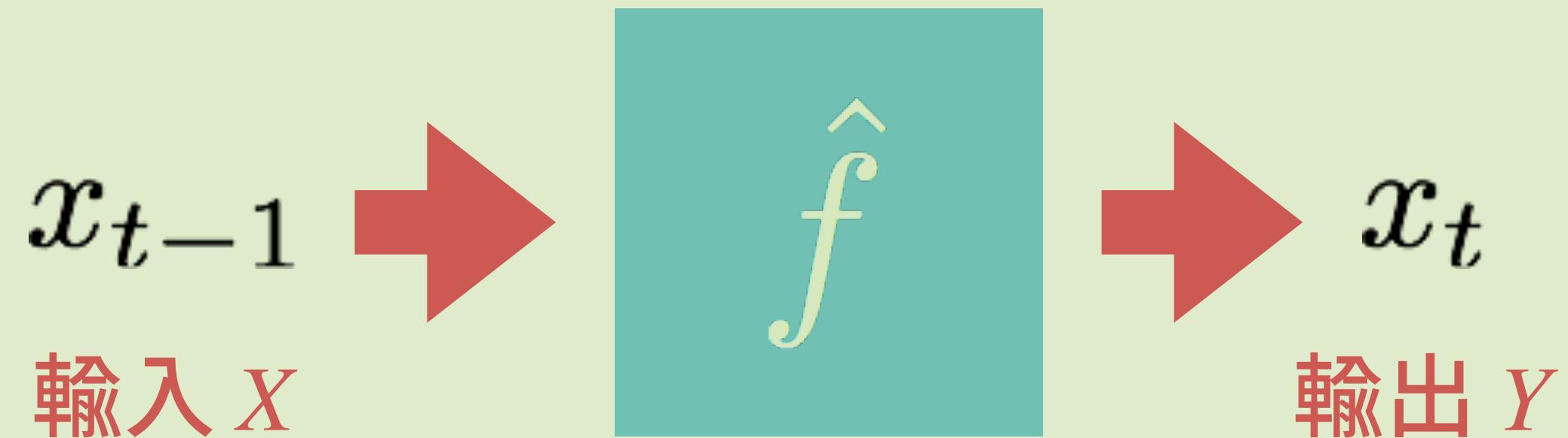


## 例子

# 股價預測

```
import pandas_datareader as pdr  
df = pdr.get_data_yahoo('AAPL')  
  
df = df[-300:]  
  
adj_close = df.Close.values  
  
X = adj_close[:-1].reshape(299,1)  
Y = adj_close[1:]
```

還記得之前我們讀入 Apple 股價，選最後 300 個交易量的資料，想學這個函數：



## 小重點

# 準備訓練和測試資料

```
x_train = x[:-40]  
x_test = x[-40:]  
y_train = y[:-40]  
y_test = y[-40:]
```

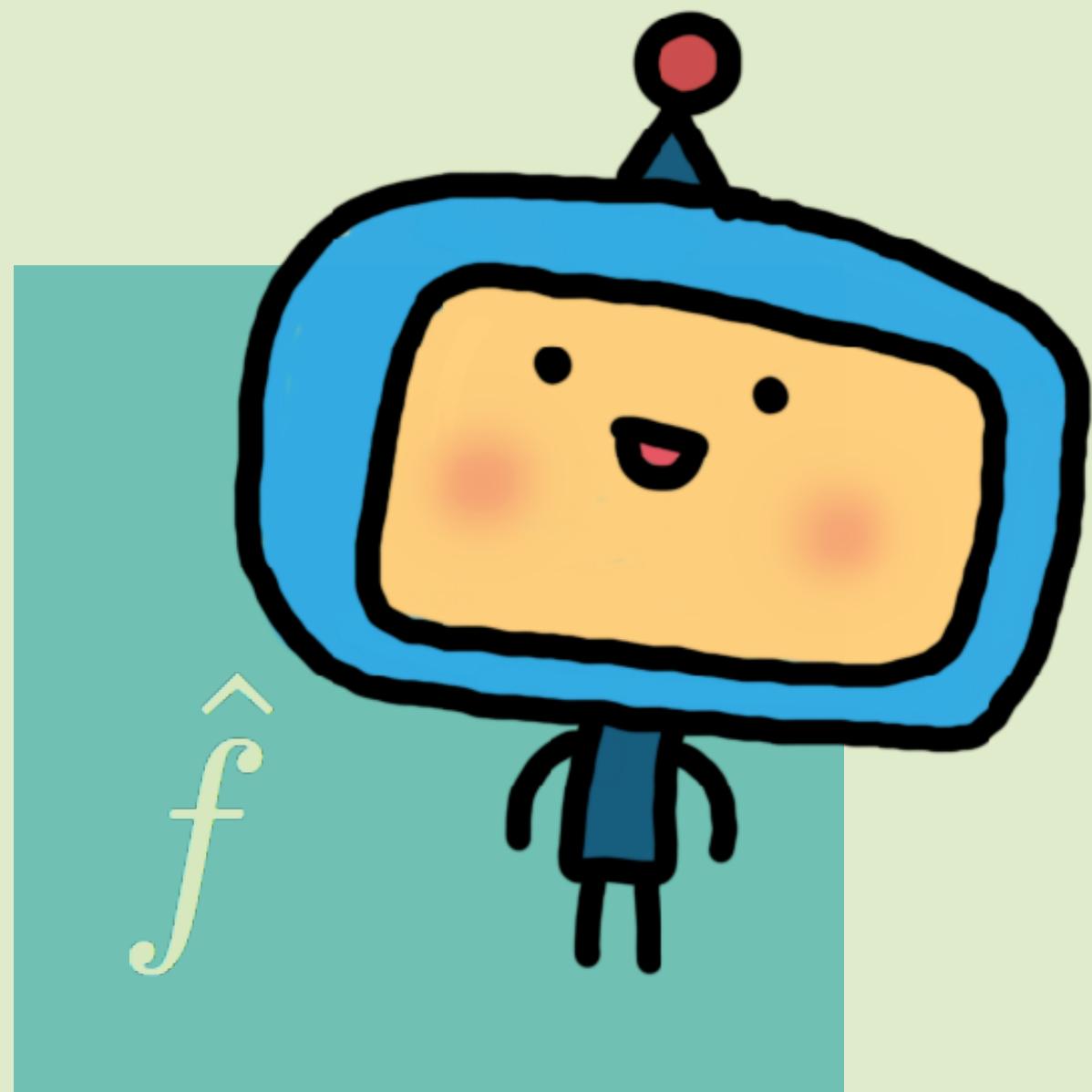


例子

## 股價預測

開個「迴歸機」，然後學習！

```
regr = LinearRegression()  
regr.fit(x_train, y_train)
```

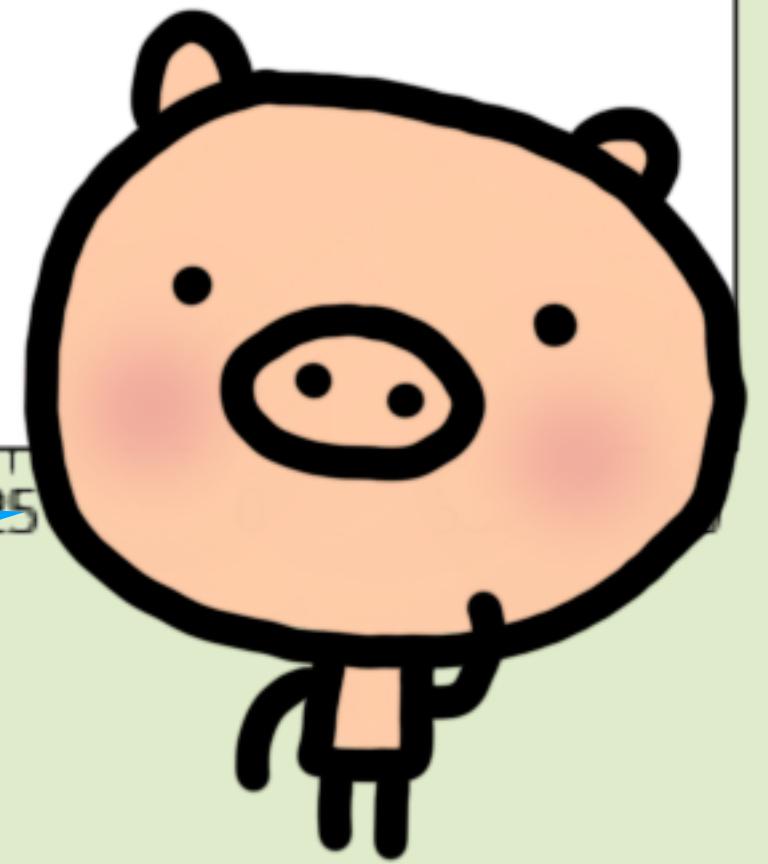
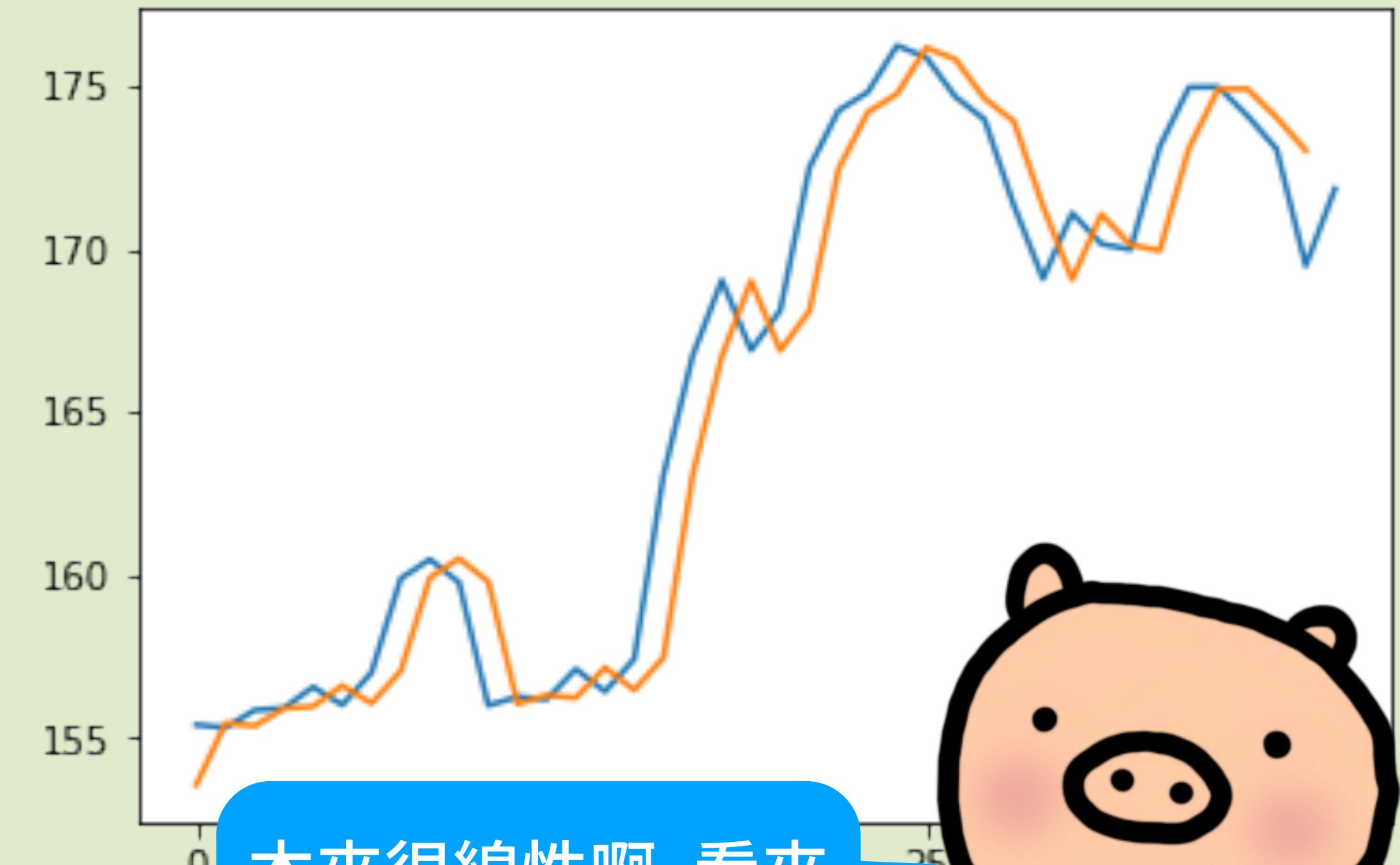


例子

## 股價預測

預測!

```
plt.plot(y_test)  
plt.plot(regr.predict(x_test))
```



## 例子

# 股價預測

直接預測未來 40 天...

```
x0 = x_test[0]
y_predict = np.array([])

for i in range(40):
    y0 = regr.predict([x0])
    y_predict =
    np.append(y_predict, y0)
    x0 = y0
```

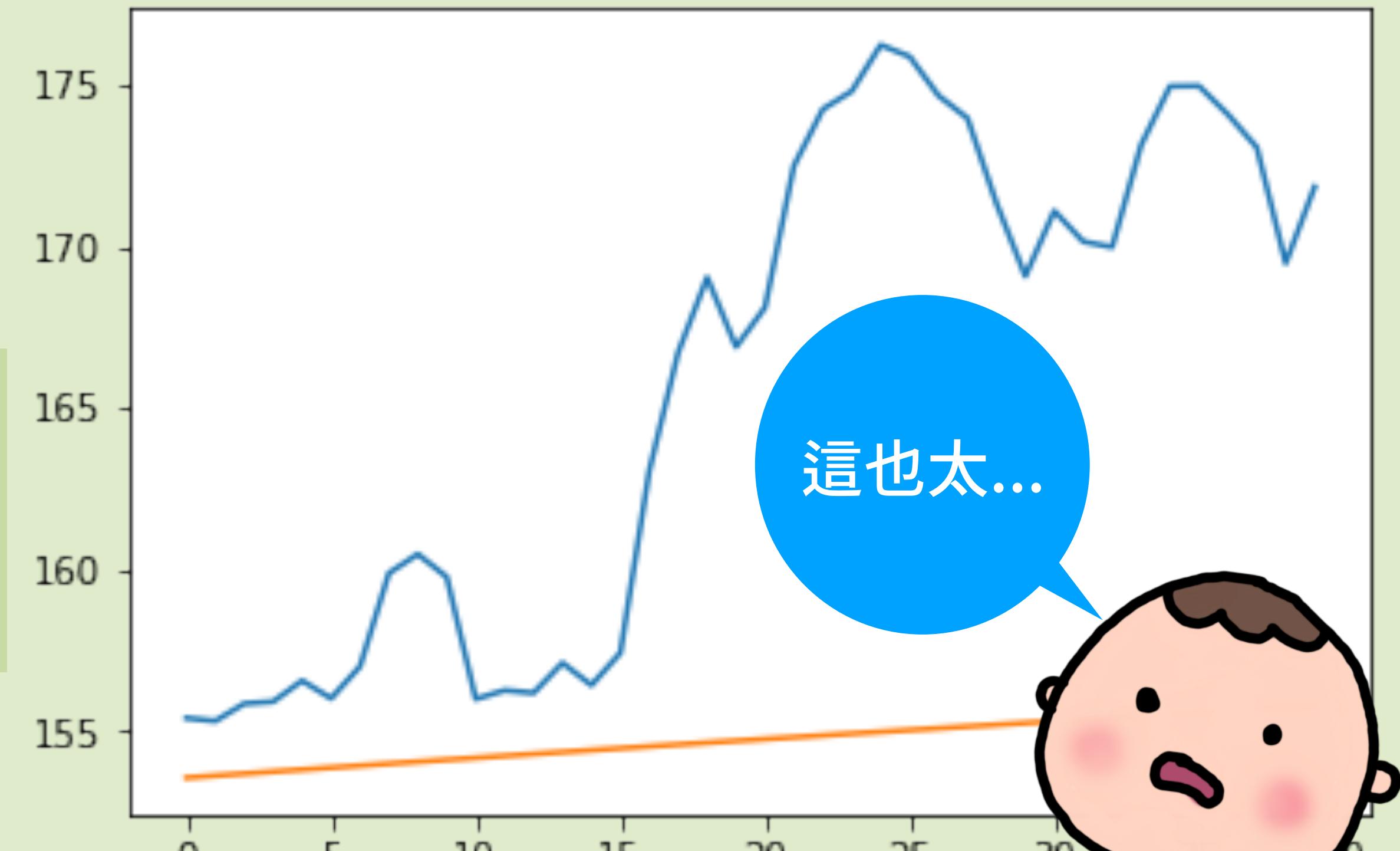


例子

## 股價預測

直接預測未來 40 天，畫出結果。

```
plt.plot(y_test)  
plt.plot(y_predict)
```



## 檢討

1

我們可能要換個方式問問題，例如 input  
多加入幾期：

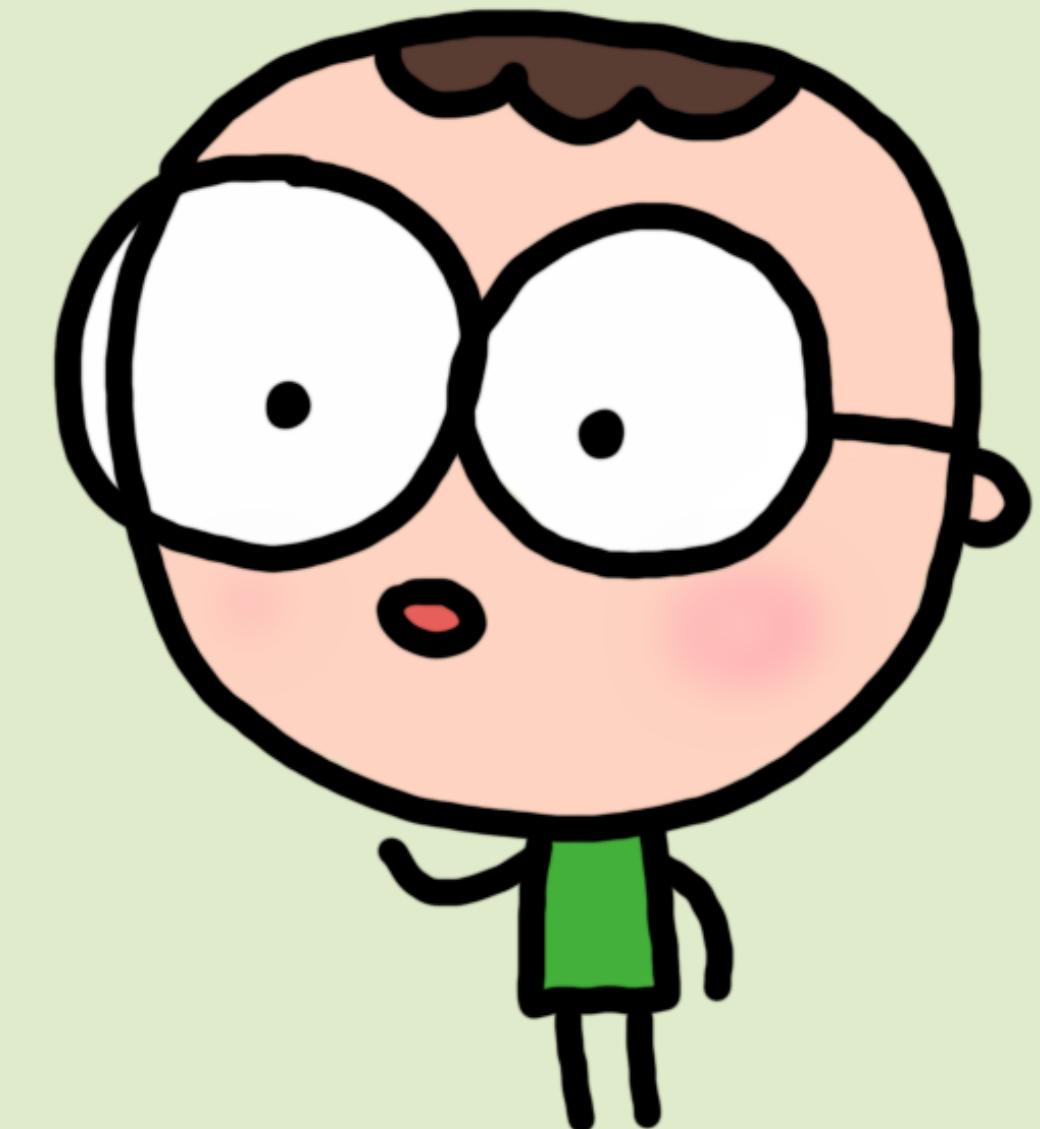
$$x_t = f(x_{t-1}, x_{t-2}, x_{t-3})$$

2

更換方法！

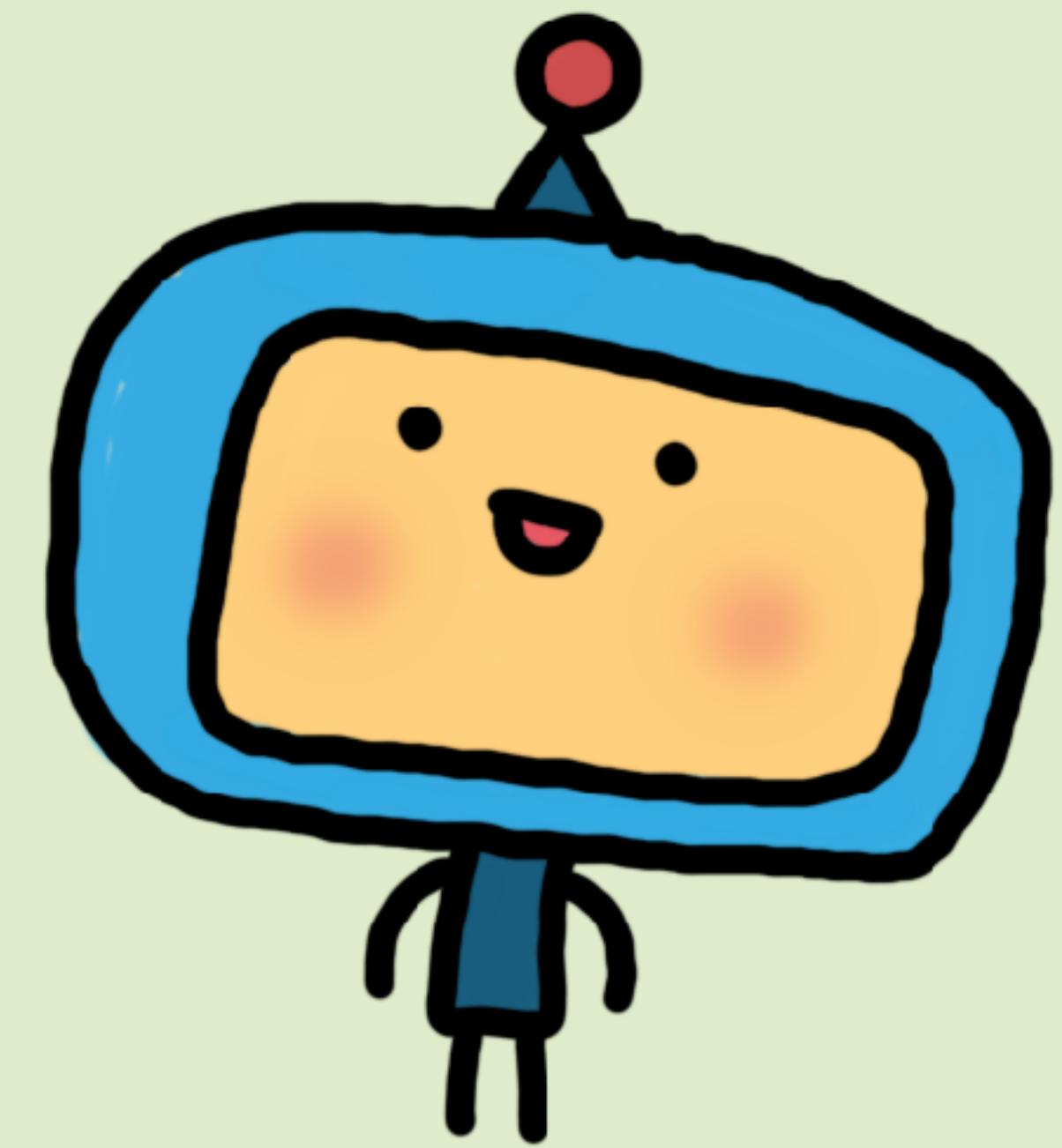
要輸出的也  
可重新考慮

也可以加入其他 features (如某個指標)。



# 迴歸分析

## 房價預測



準備

# 讀入波士頓房價數據庫

```
from sklearn.datasets import load_boston  
boston = load_boston()
```

小技巧

## 看數據庫的說明

SciKit-Learn 內建資料庫都有一些 features 等說明。

```
print(boston.DESCR)
```

也可看 features 有哪些。

```
boston.feature_names
```

準備

## inputs 和 outputs

數據庫的輸入和「正確答案」分別放在 `data` 和 `target` 中。

```
x = boston.data  
y = boston.target
```

# 訓練資料和測試資料

```
from sklearn.model_selection import train_test_split  
x_train, x_test, y_train, y_test = train_test_split(x, y,  
                                                    test_size=0.3,  
                                                    random_state=87)
```

重點

## 開個分類機、訓練

```
regr = LinearRegression()  
regr.fit(x_train, y_train)
```



小重點

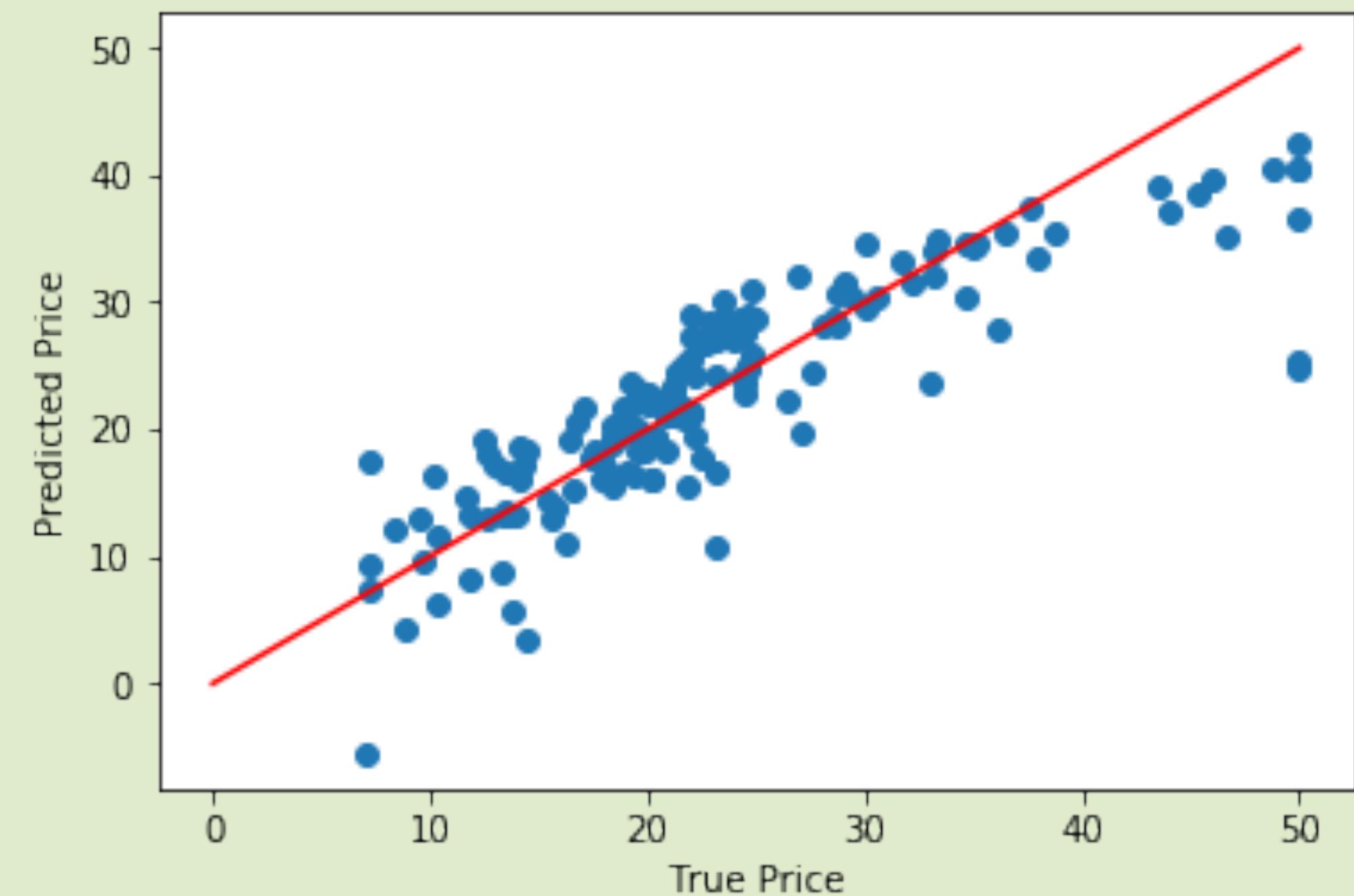
## 預測看看

```
y_predict = regr.predict(x_test)
```

小重點

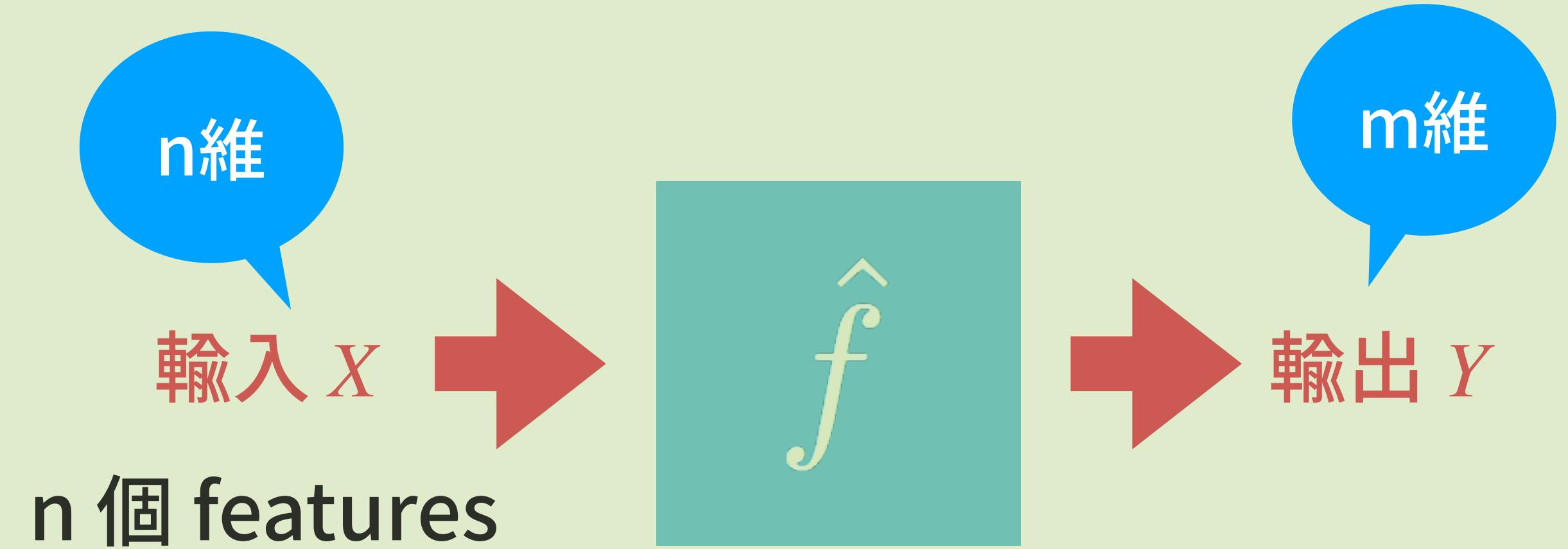
## 預測看看

```
plt.scatter(y_test, y_predict)  
plt.plot([0,50],[0,50], 'r')  
plt.xlabel('True Price')  
plt.ylabel('Predicted Price')
```



# 機器學習概要之一

## SVM



機器學習還是老話一句，就是要學個函數！

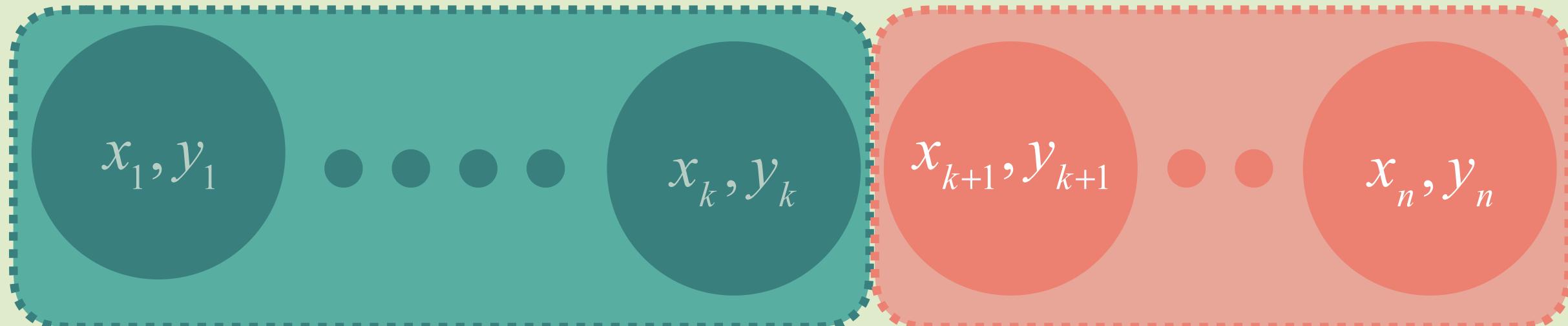


# 機器學習兩大方式

## 監督式學習

Supervised Learning

有一堆知道正確答案的資料。



## 非監督式學習

Unsupervised Learning

沒有提供正確答案, 電腦自理。

真的可以嗎?



# 機器學習兩大方式

監督式學習

Supervised Learning

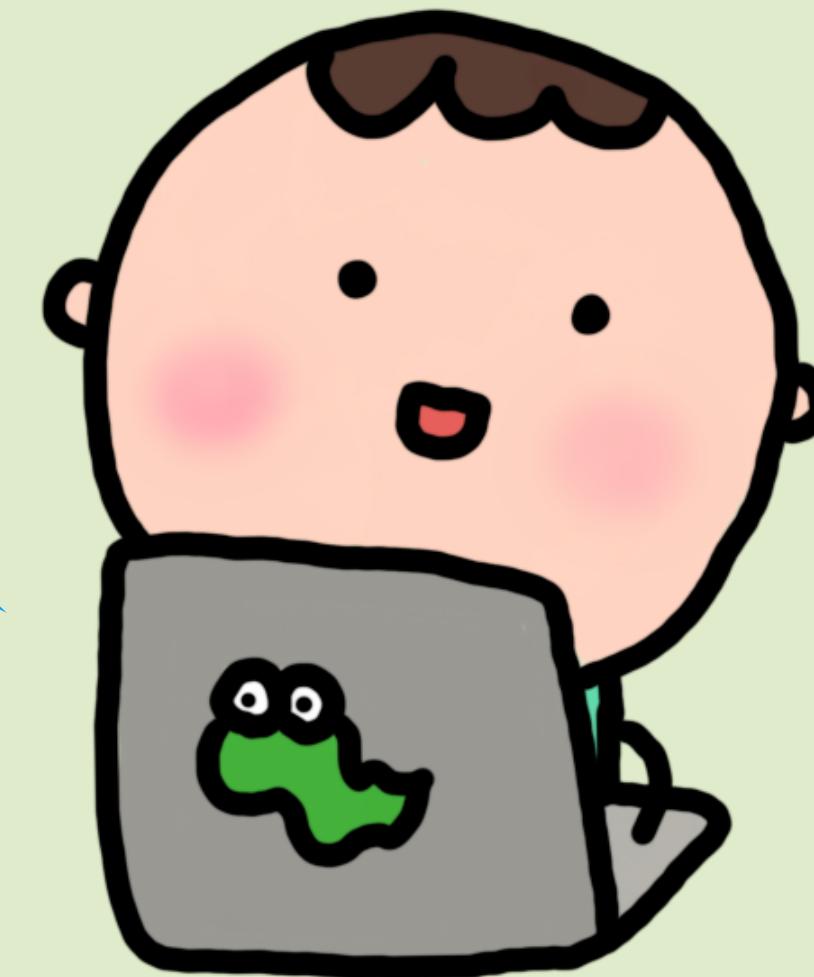
【例子】SVM

非監督式學習

Unsupervised Learning

【例子】K-Means

我們來做分類!

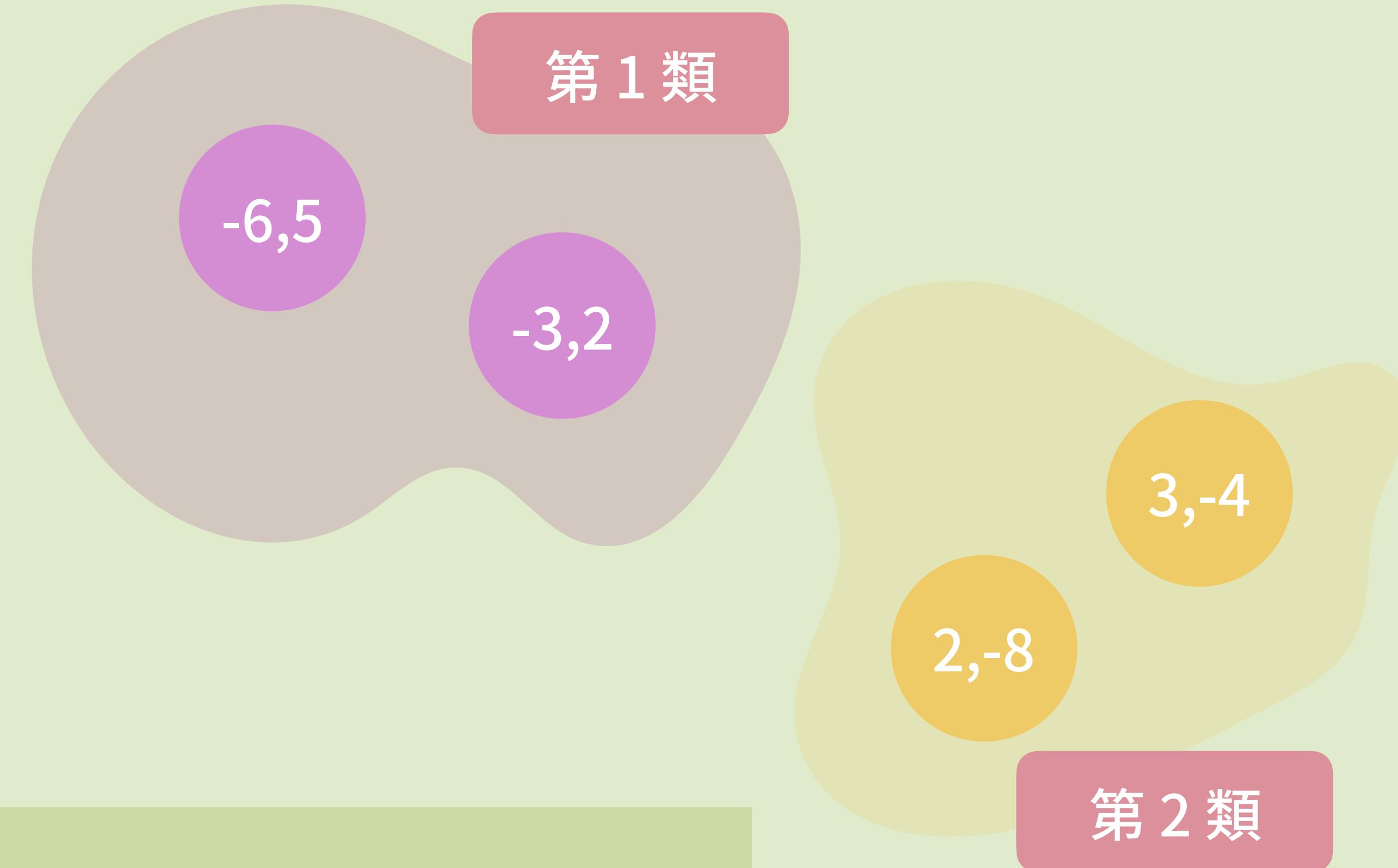


例子

# SVM 超容易

我們有兩類的資料，想用 SVM  
學習去分類。

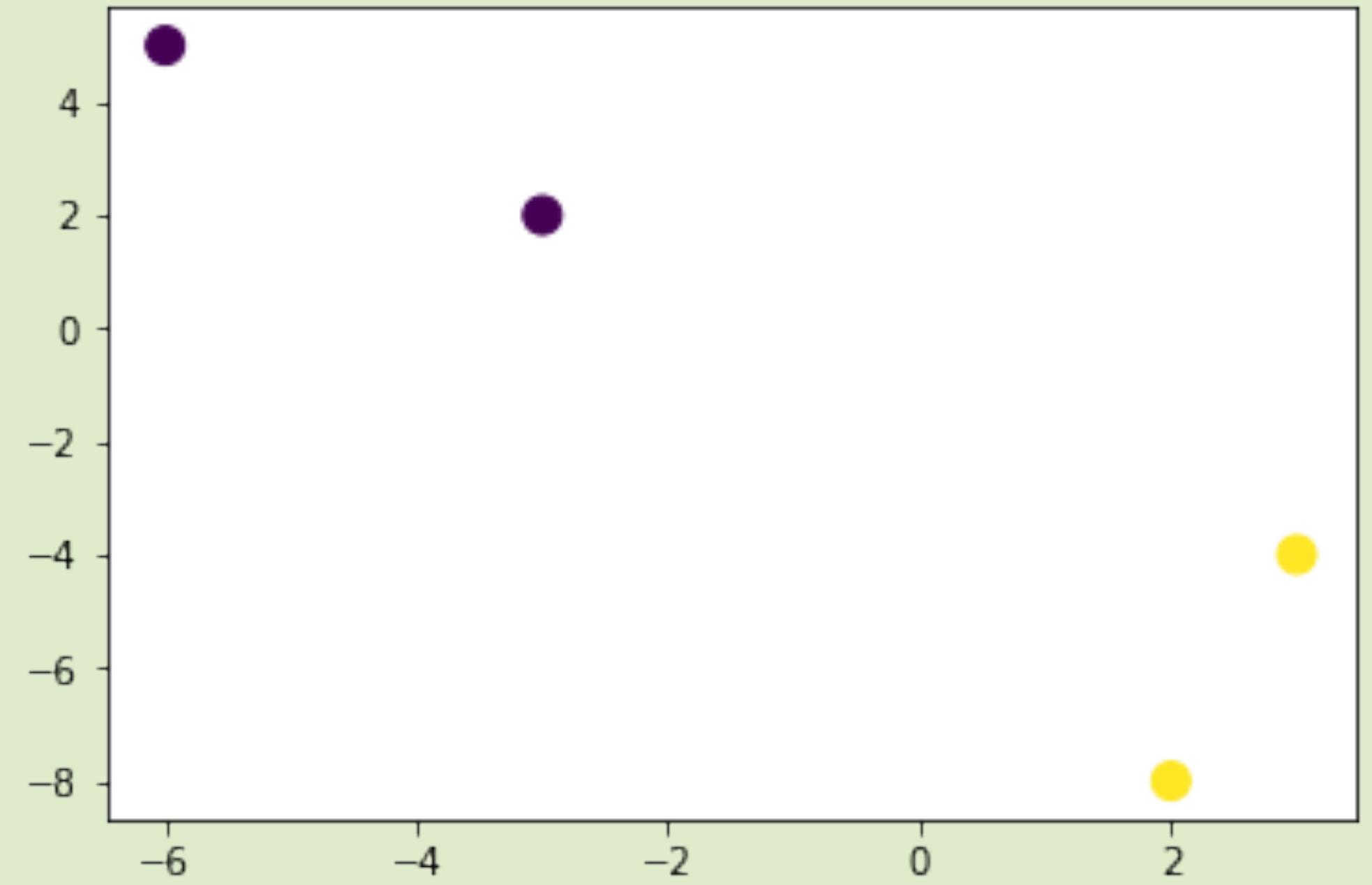
```
x = np.array([[-3, 2], [-6, 5], [3, -4], [2, -8]])  
y = np.array([1, 1, 2, 2])
```



例子

## SVM 超容易

畫出來看一看。



```
plt.scatter(x[:,0], x[:,1], c=y, s=100)
```

例子

## SVM 超容易

```
from sklearn.svm import SVC
```

我們要用 SVM 做分類，使用 SVC 分類器。



例子

## SVM 超容易

和迴歸一樣，我們打開一個分類機。

```
clf = SVC()
```



例子

## SVM 超容易

訓練就這樣，馬上完成！

```
clf.fit(x, y)
```



例子

## SVM 超容易

預測也和以前一樣!

```
clf.predict([[-4.2, 3.3]])
```

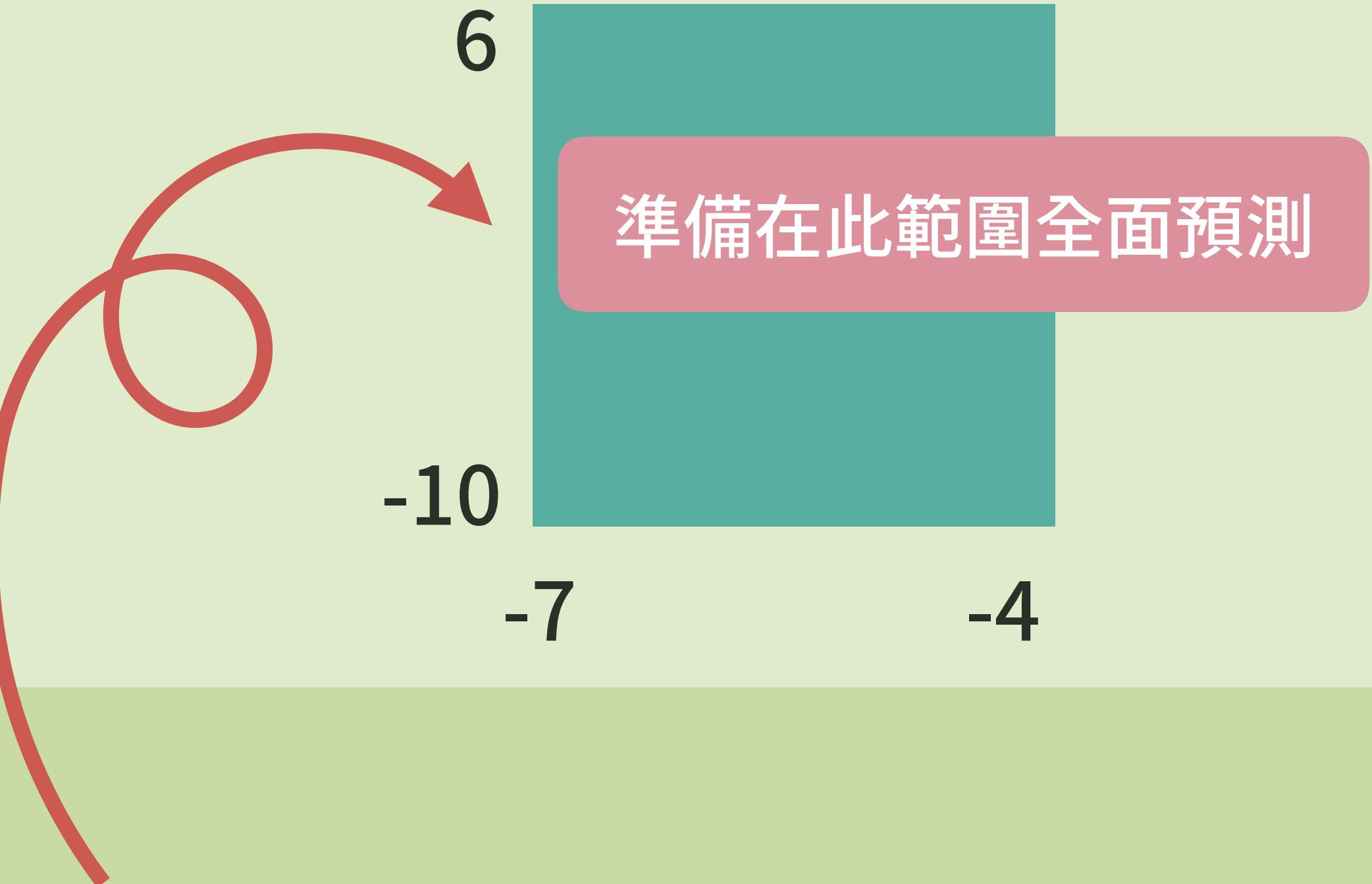
輸出: array([1])



例子

## SVM 超容易

準備看預測的結果。



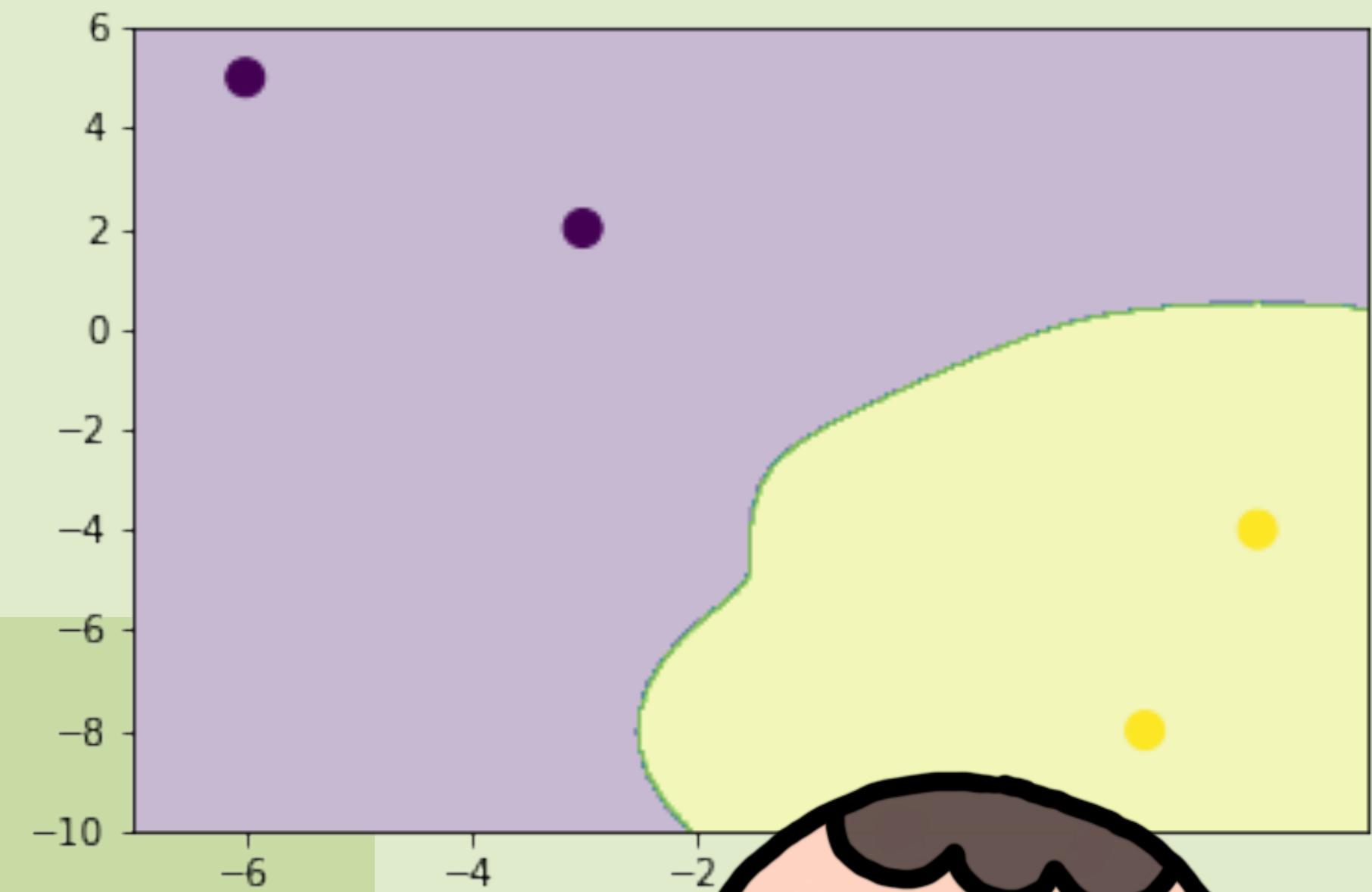
```
x1, x2 = np.meshgrid(np.arange(-7, 4, 0.02), np.arange(-10, 6, 0.02))  
z = clf.predict(np.c_[x1.ravel(), x2.ravel()])
```

例子

## SVM 超容易

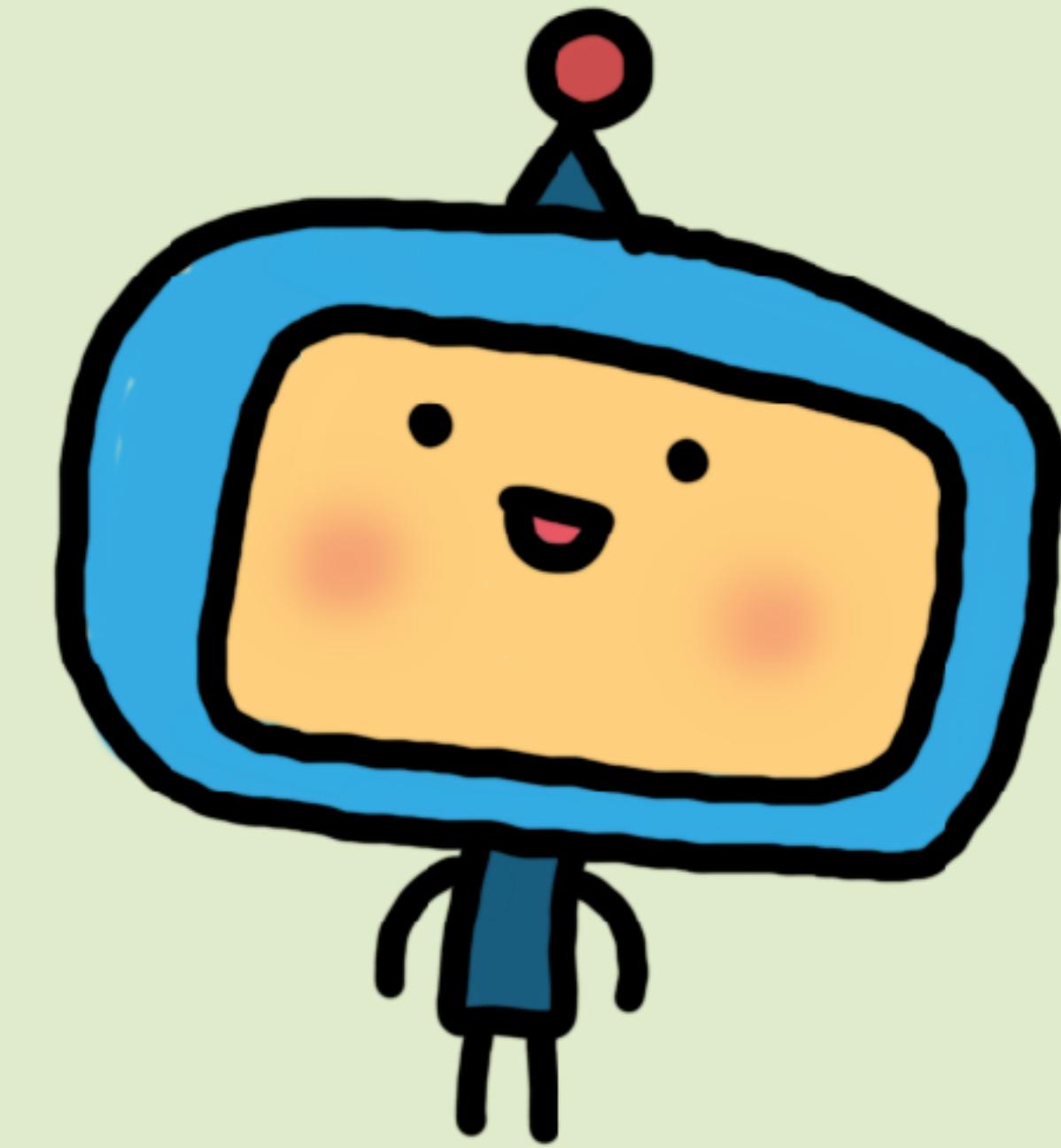
畫出來看看，發現原本的資料 100% 準確...

```
z = z.reshape(x1.shape)  
  
plt.contourf(x1, x2, z, alpha=0.3)  
plt.scatter(x[:,0], x[:,1], s=100, c=y)
```



# SVM

## 鳶尾花分類

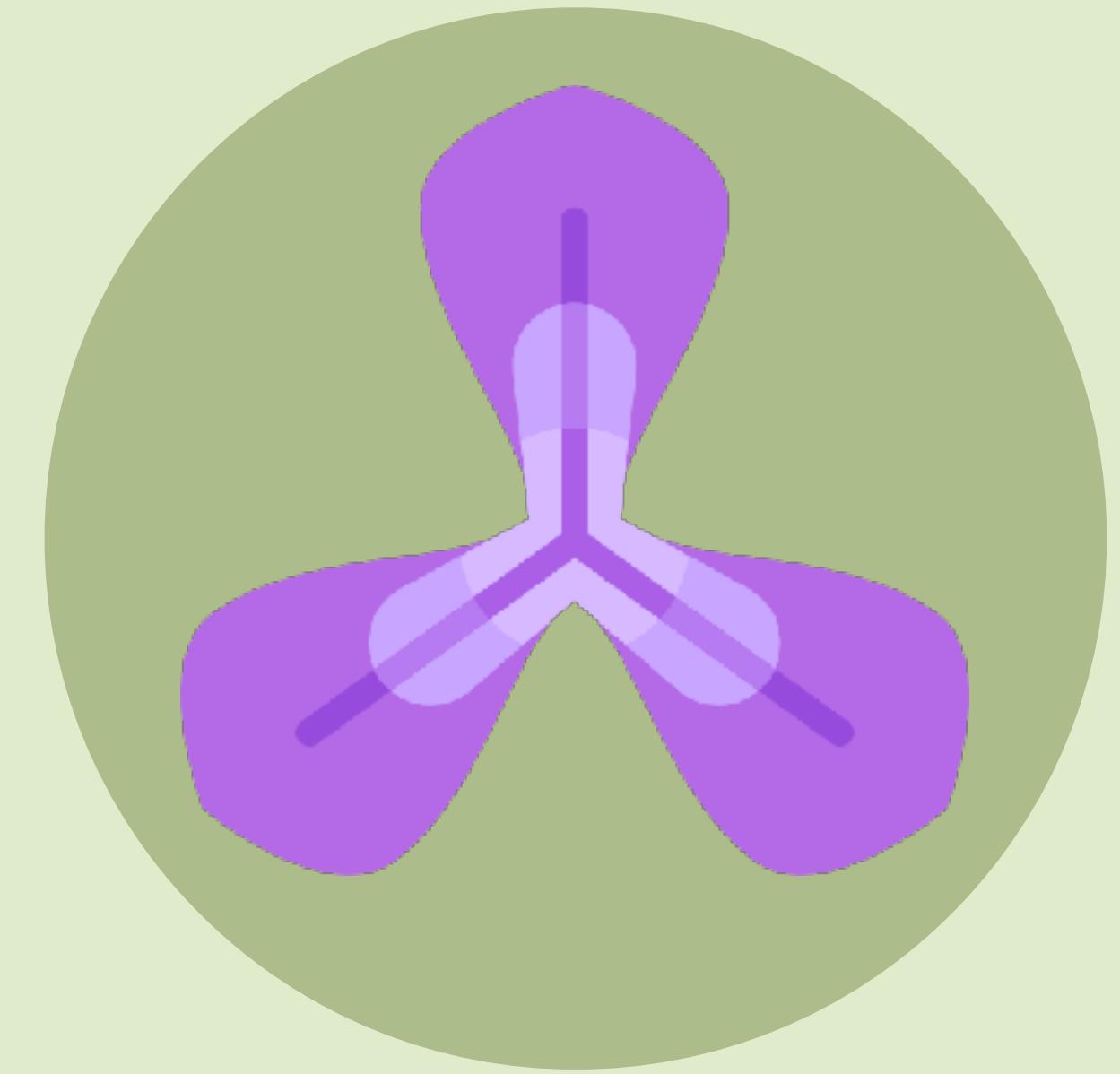


## 準備

# 讀入鳶尾花數據庫

鳶尾花 (Iris) 數據庫是很有名的資料, 就是試著以一朵鳶尾花花萼、花瓣的大小來分出是哪個的大小來分出是哪個亞種的鳶尾花。

```
from sklearn.datasets import load_iris  
  
iris = load_iris()
```



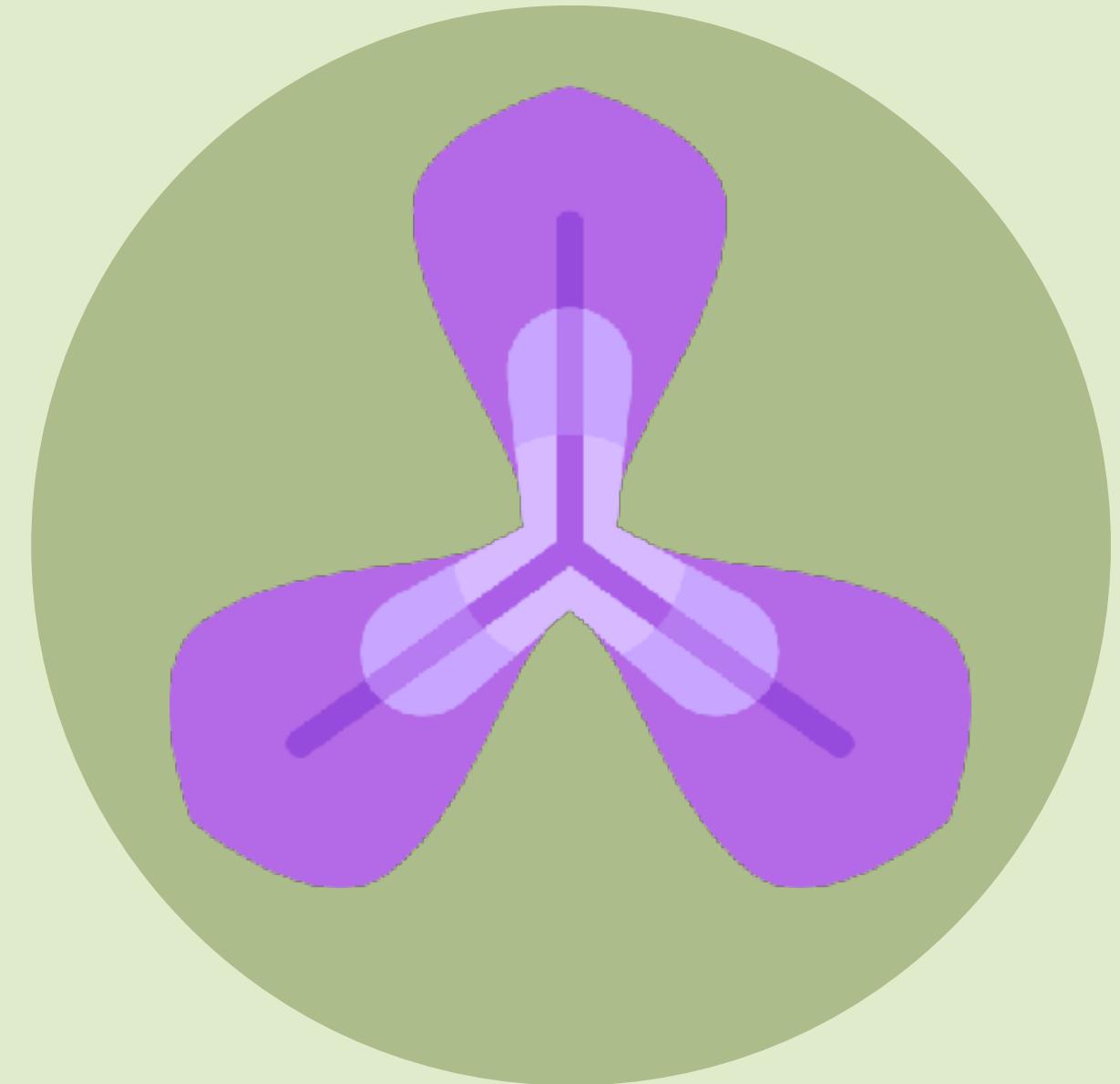
## 小技巧

# 看數據庫的說明

SciKit-Learn 內建資料庫都有一些 features 等說明。

```
print(iris.DESCR)
```

有四個 features: 花萼長度、寬度和花瓣長度、寬度

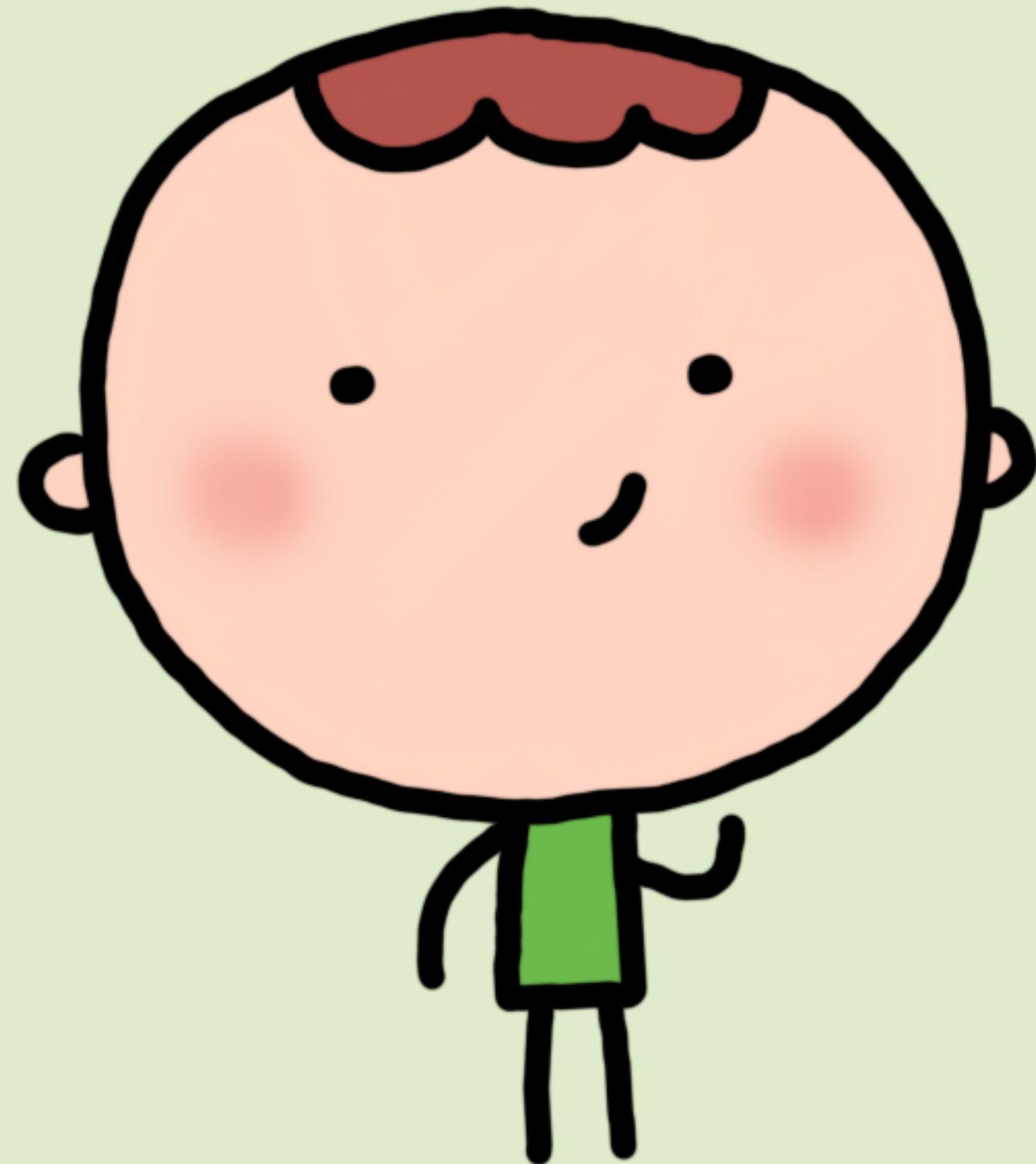


準備

## inputs 和 outputs

數據庫的輸入和「正確答案」分別放在 `data` 和 `target` 中。

```
x = iris.data  
y = iris.target
```



準備

## inputs 和 outputs

為了表示我們很神 (事實上只是好畫圖), 我們只用兩個 features (花萼長度、寬度)。

```
x = x[:, 2:]
```



# 訓練資料和測試資料

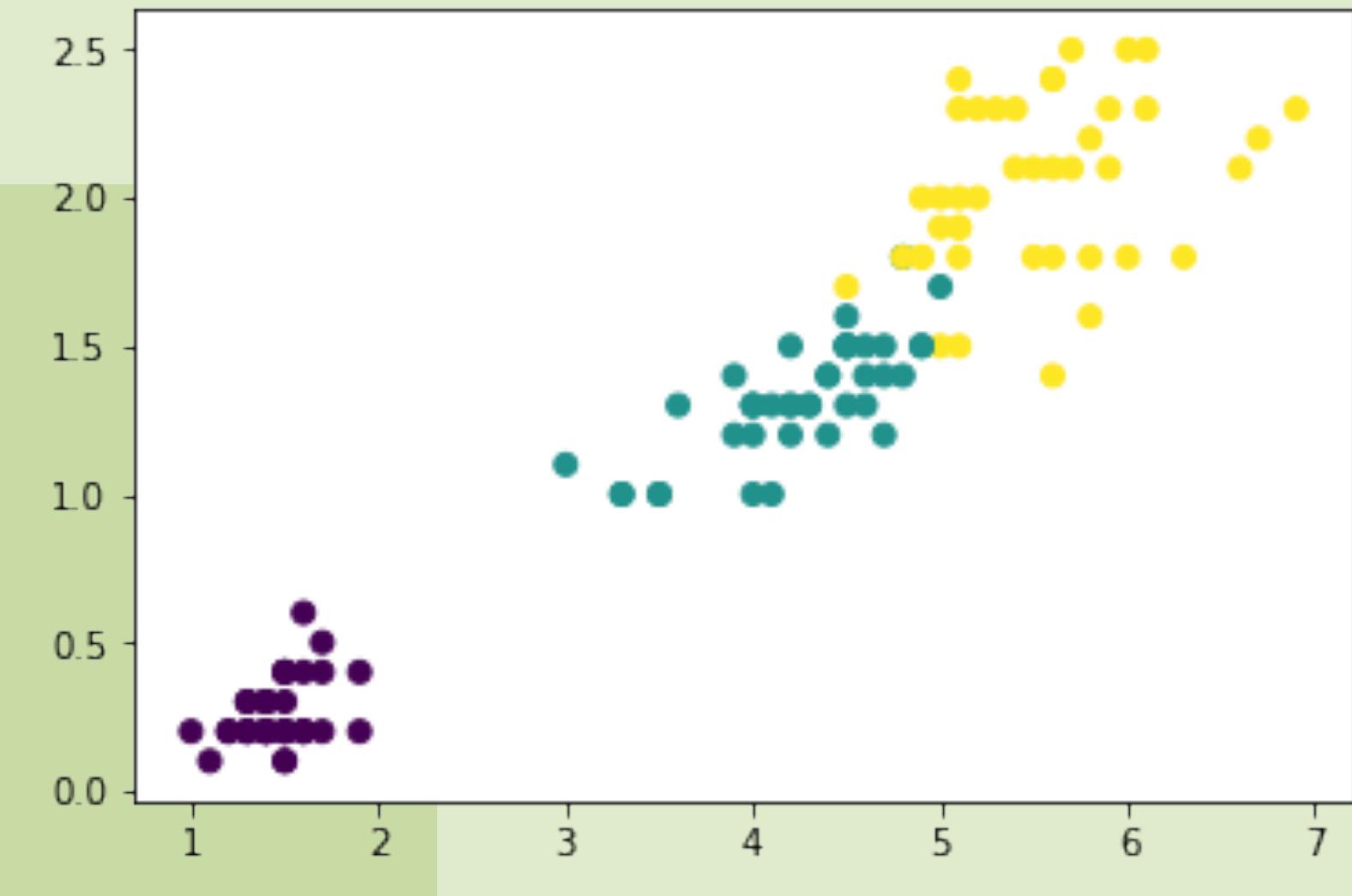
用 80% 當訓練資料, 留 20% 看我們做得如何?

```
from sklearn.model_selection import train_test_split  
  
x_train, x_test, y_train, y_test = train_test_split(x, y,  
                                                test_size=0.2,  
                                                random_state=87)
```

小重點

畫圖出來感受一下

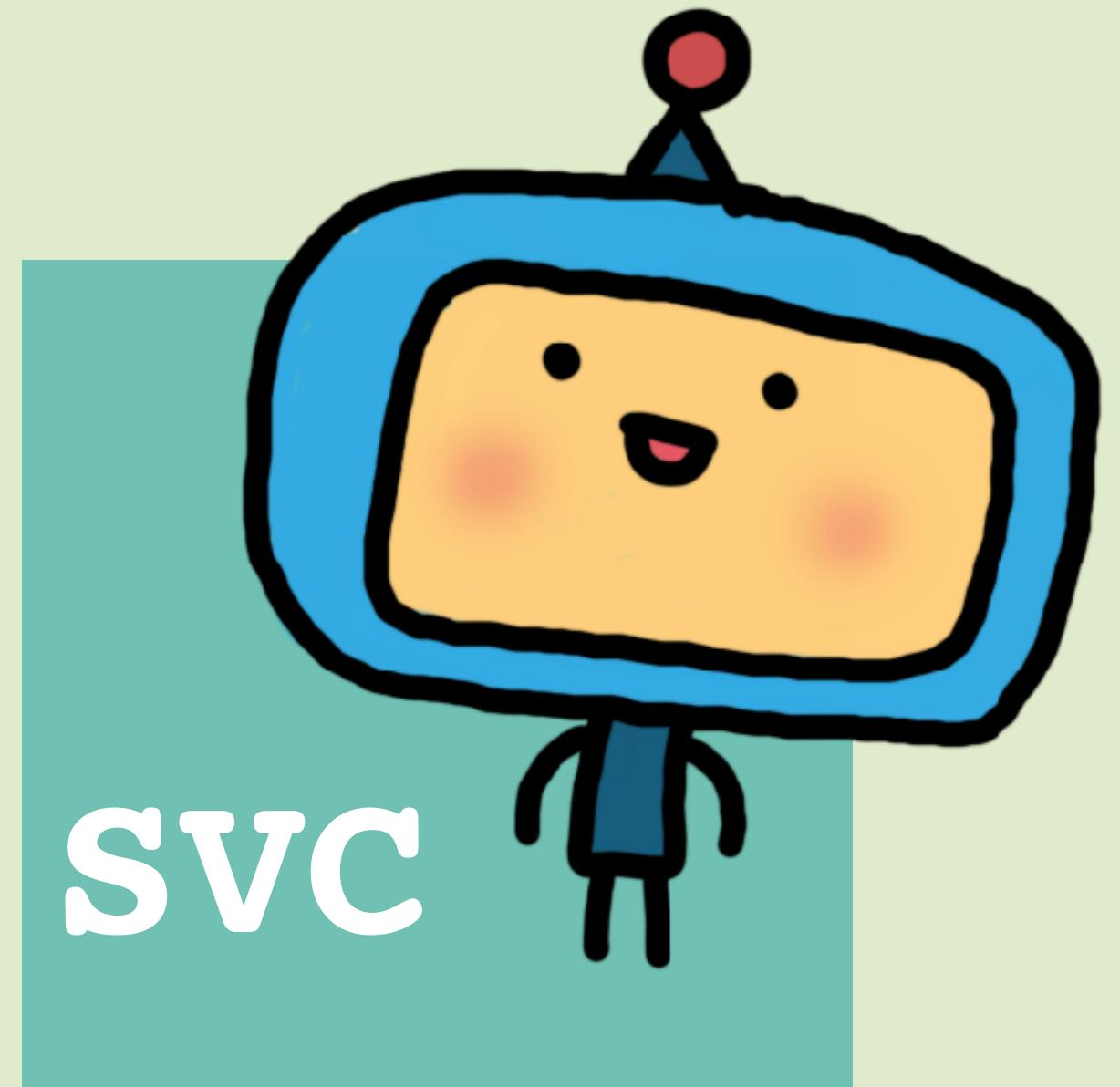
```
plt.scatter(x_train[:,0], x_train[:,1],  
c=y_train)
```



重點

## 開個分類機、訓練

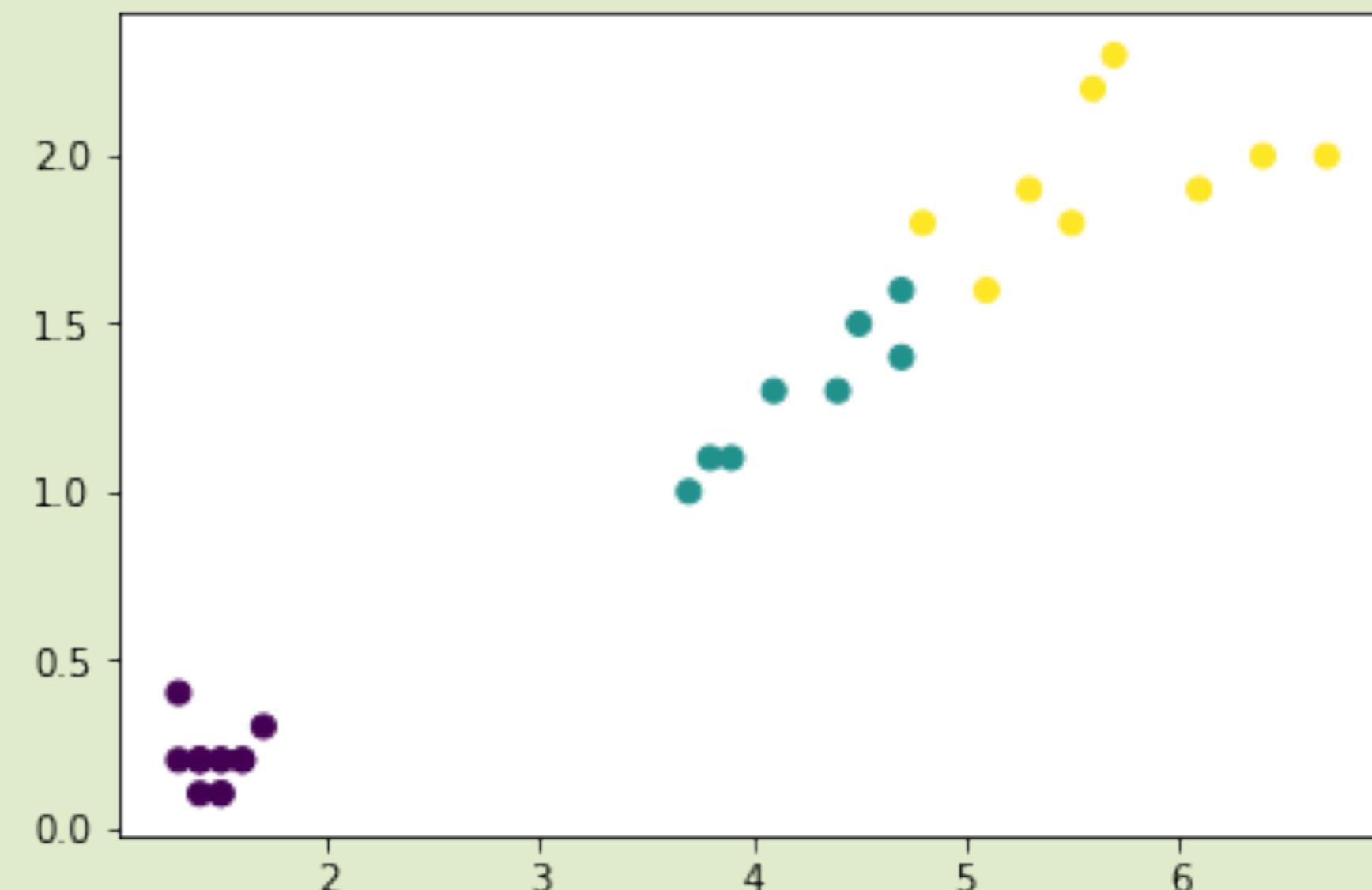
```
from sklearn.svm import SVC  
clf = SVC()  
clf.fit(x_train, y_train)
```



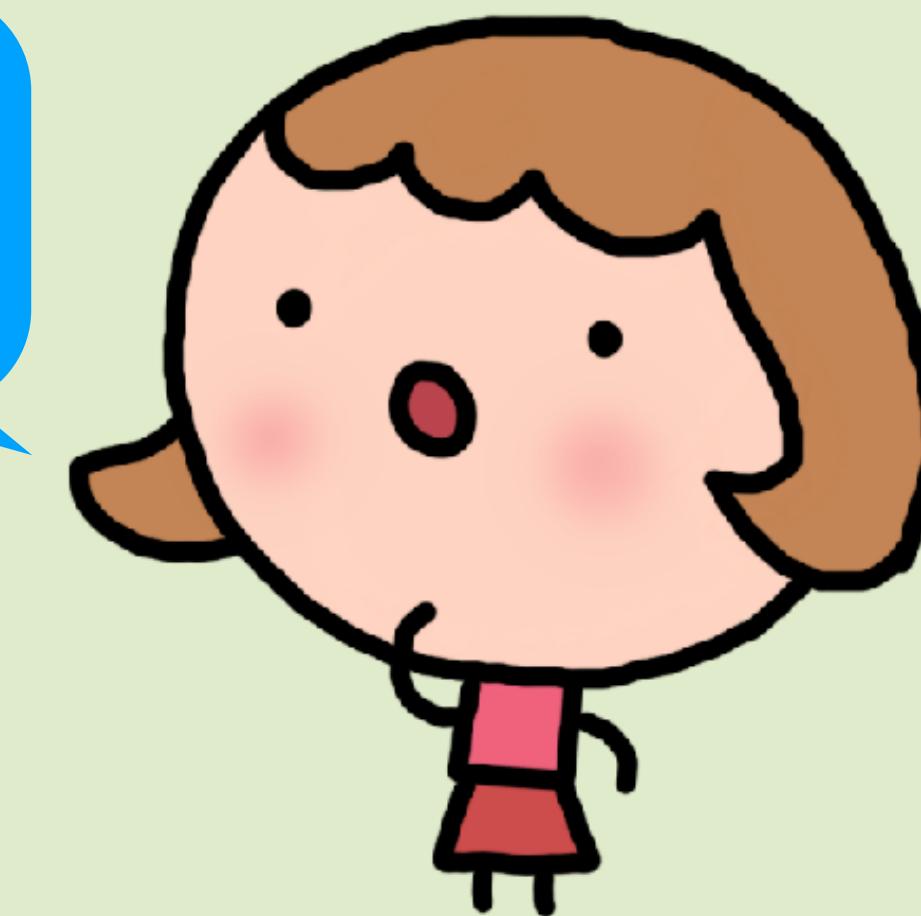
## 小重點

# 預測看看

```
y_predict = clf.predict(x_test)  
plt.scatter(x_test[:,0], x_test[:,1], c=y_predict)
```



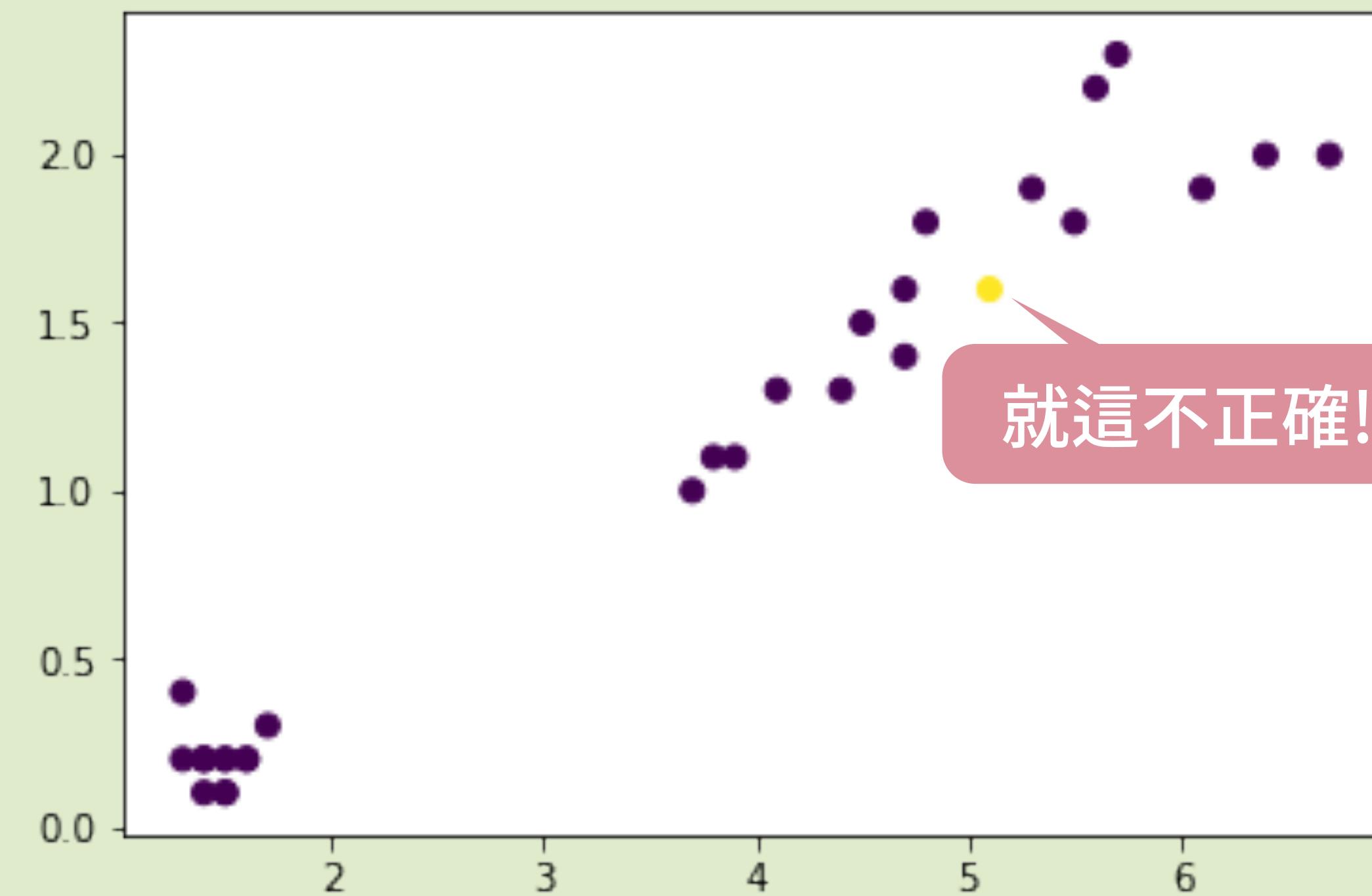
這是準還是不準？



## 小重點

# 預測看看

```
plt.scatter(x_test[:,0], x_test[:,1], c=y_predict - y_test)
```



看來還不錯!



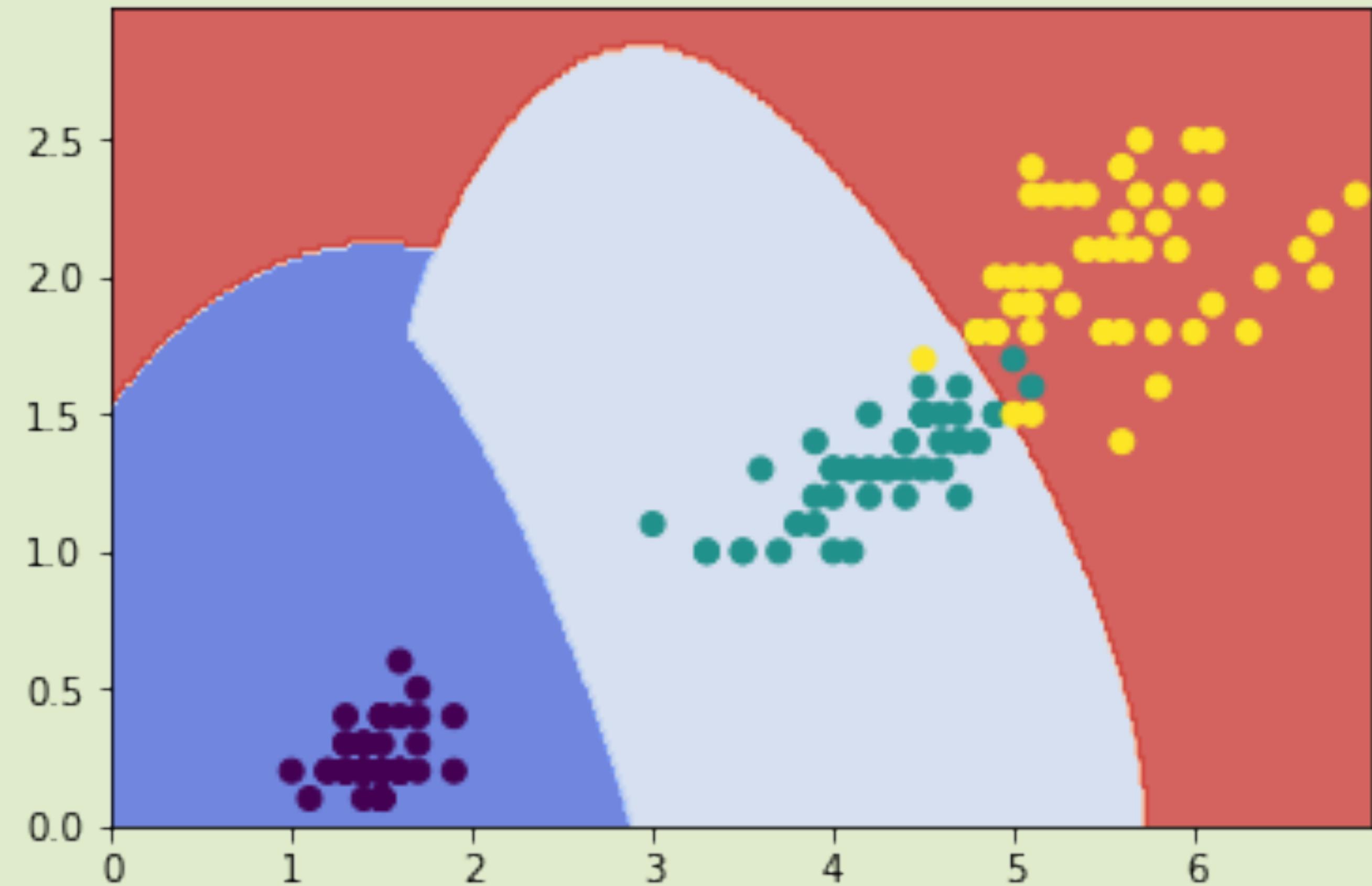
## 重點

# 高級畫圖

```
x1, x2 = np.meshgrid(np.arange(0,7,0.02), np.arange(0,3,0.02))
z = clf.predict(np.c_[x1.ravel(), x2.ravel()])
z = z.reshape(x1.shape)
plt.contourf(x1, x2, z, cmap=plt.cm.coolwarm, alpha=0.8)
plt.scatter(x[:,0], x[:,1], c=Y)
```

重點

## 高級畫圖



到現在我們也學了不少高級技巧啊!



# 機器學習概要之二

KMeans

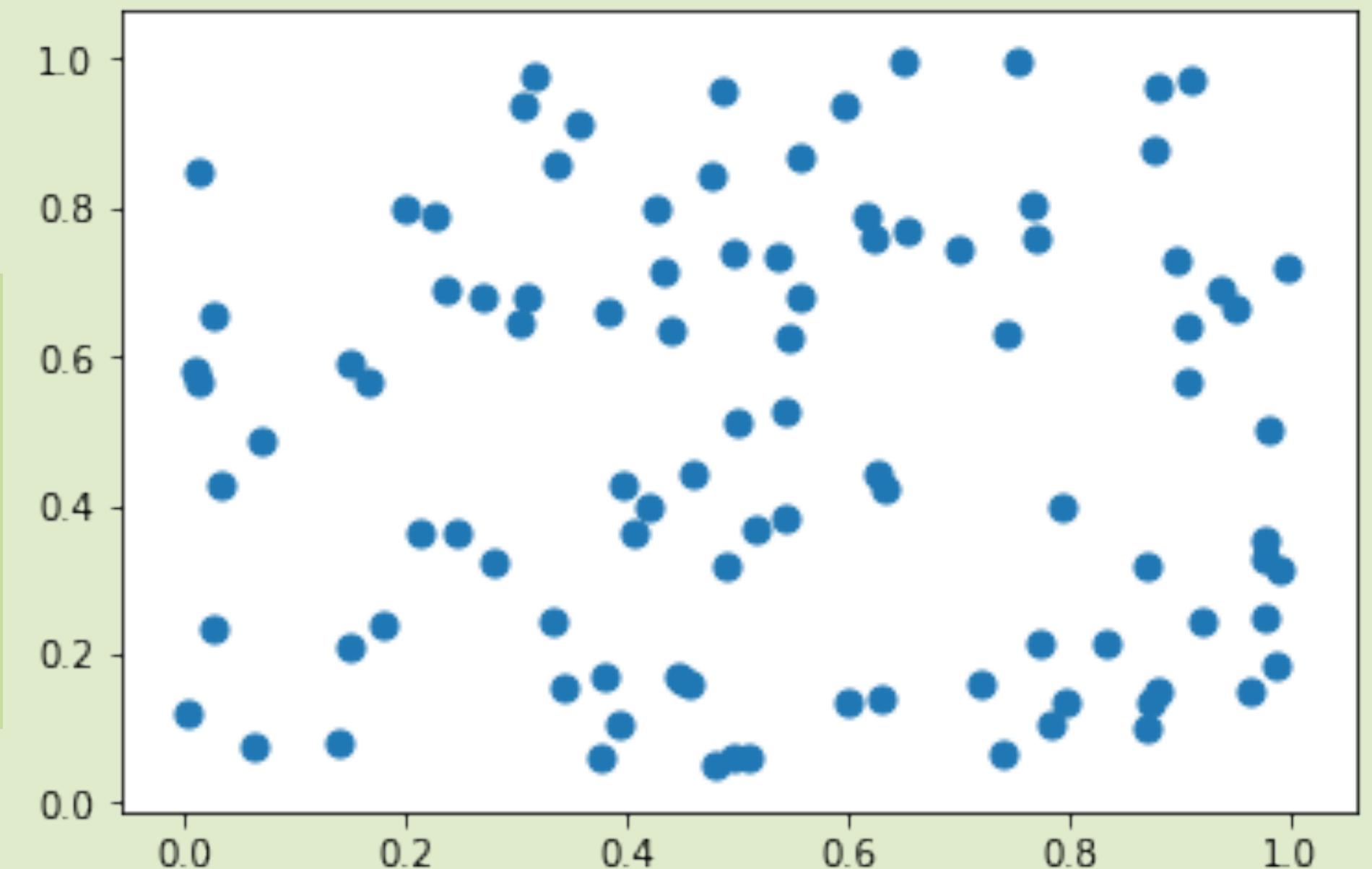
## 例子

# KMeans 基礎篇

生出 100 筆 2 維 (2 個 features) 的資料。

```
x = np.random.rand(100, 2)  
plt.scatter(x[:, 0], x[:, 1], s=50)
```

生出 100 筆 2 維 (2 個 features) 的資料。



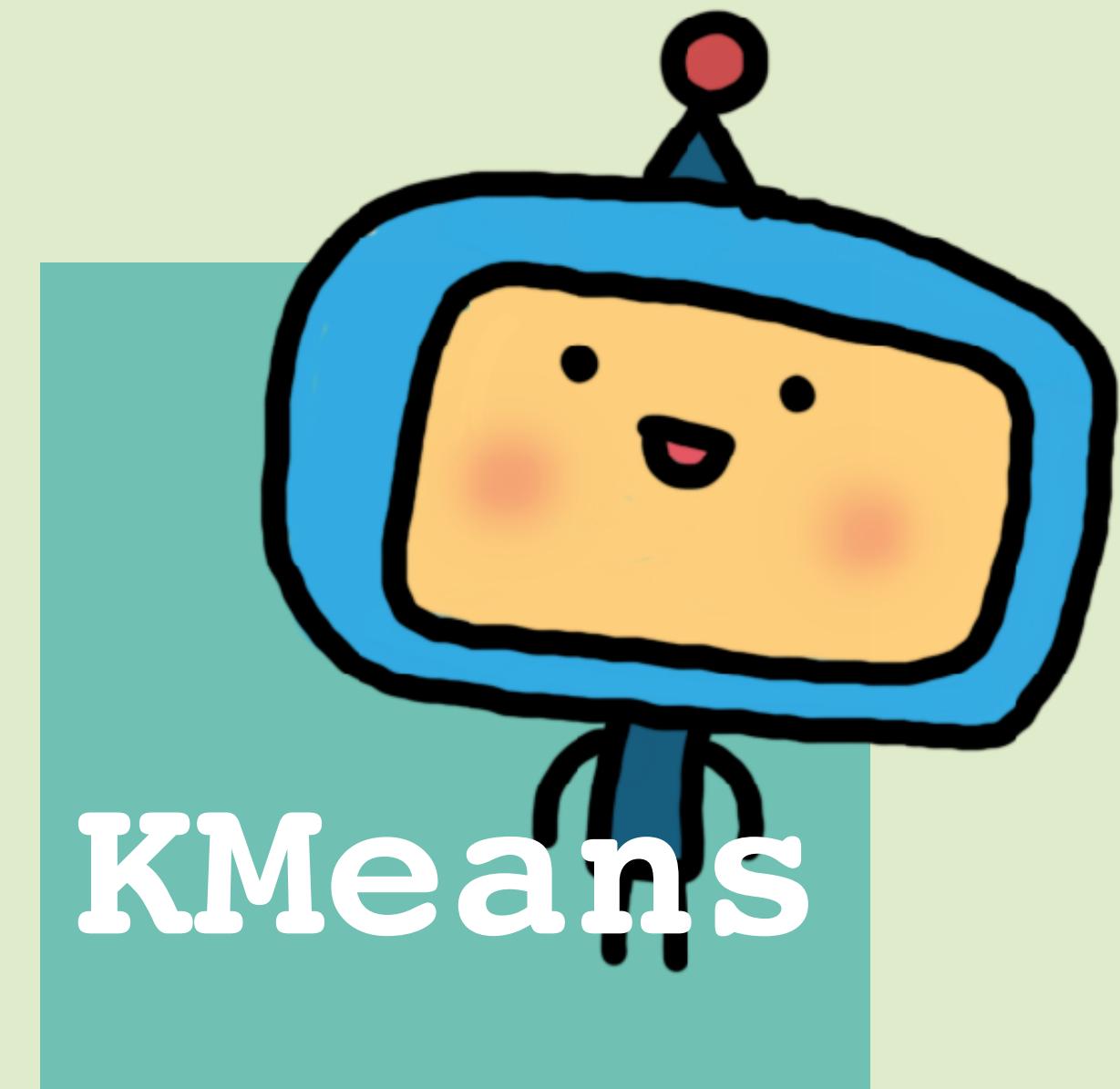
例子

## KMeans 基礎篇

開台分類機 (大家麻木了吧)。

```
from sklearn.cluster import KMeans  
clf = KMeans(n_clusters=3)
```

要表明分幾類

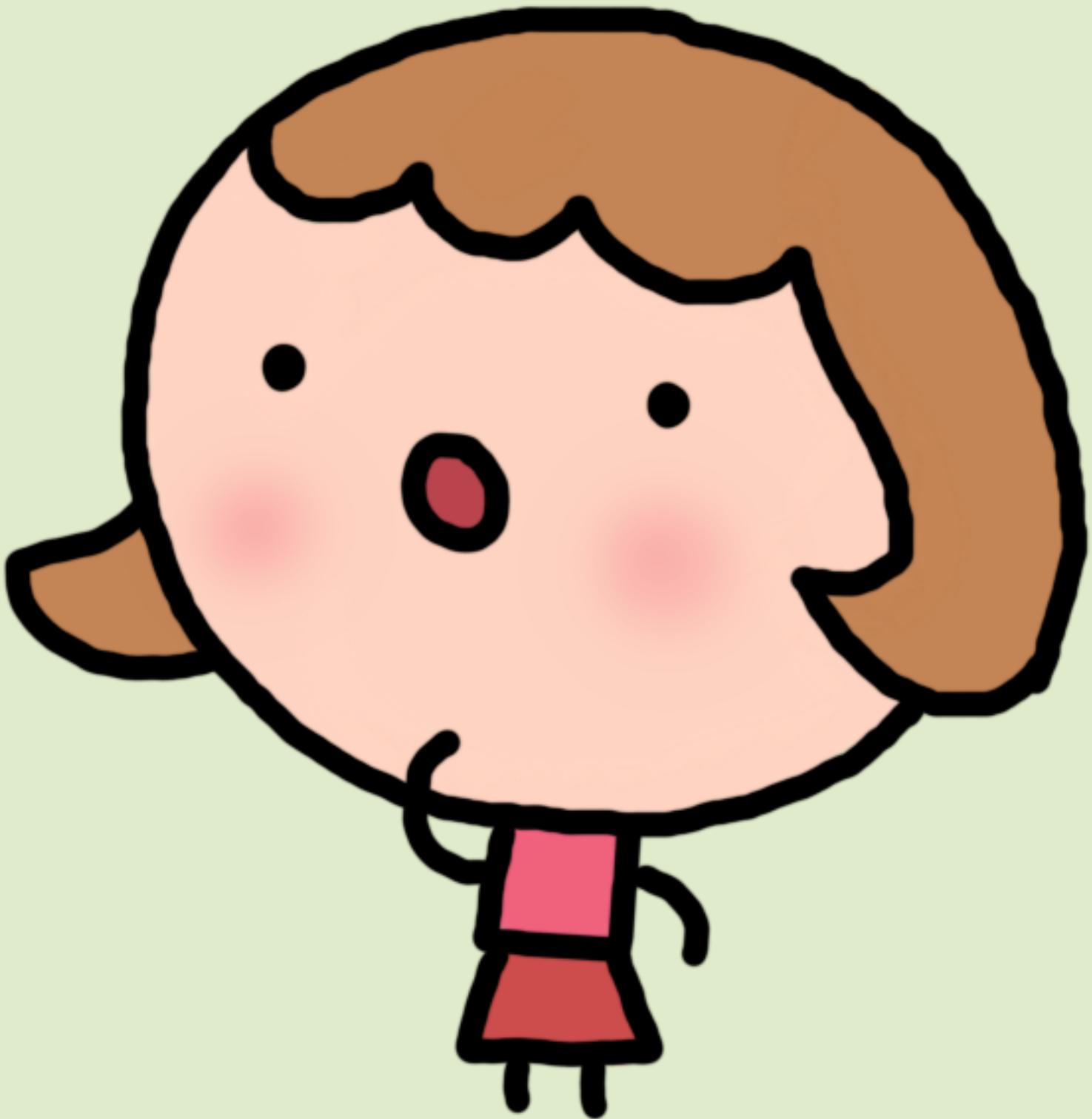


例子

## KMeans 基礎篇

訓練我們的分類機 (只有輸入資料哦)。

```
clf.fit(x)
```



例子

## KMeans 基礎篇

看看分類結果。

`clf.labels_`

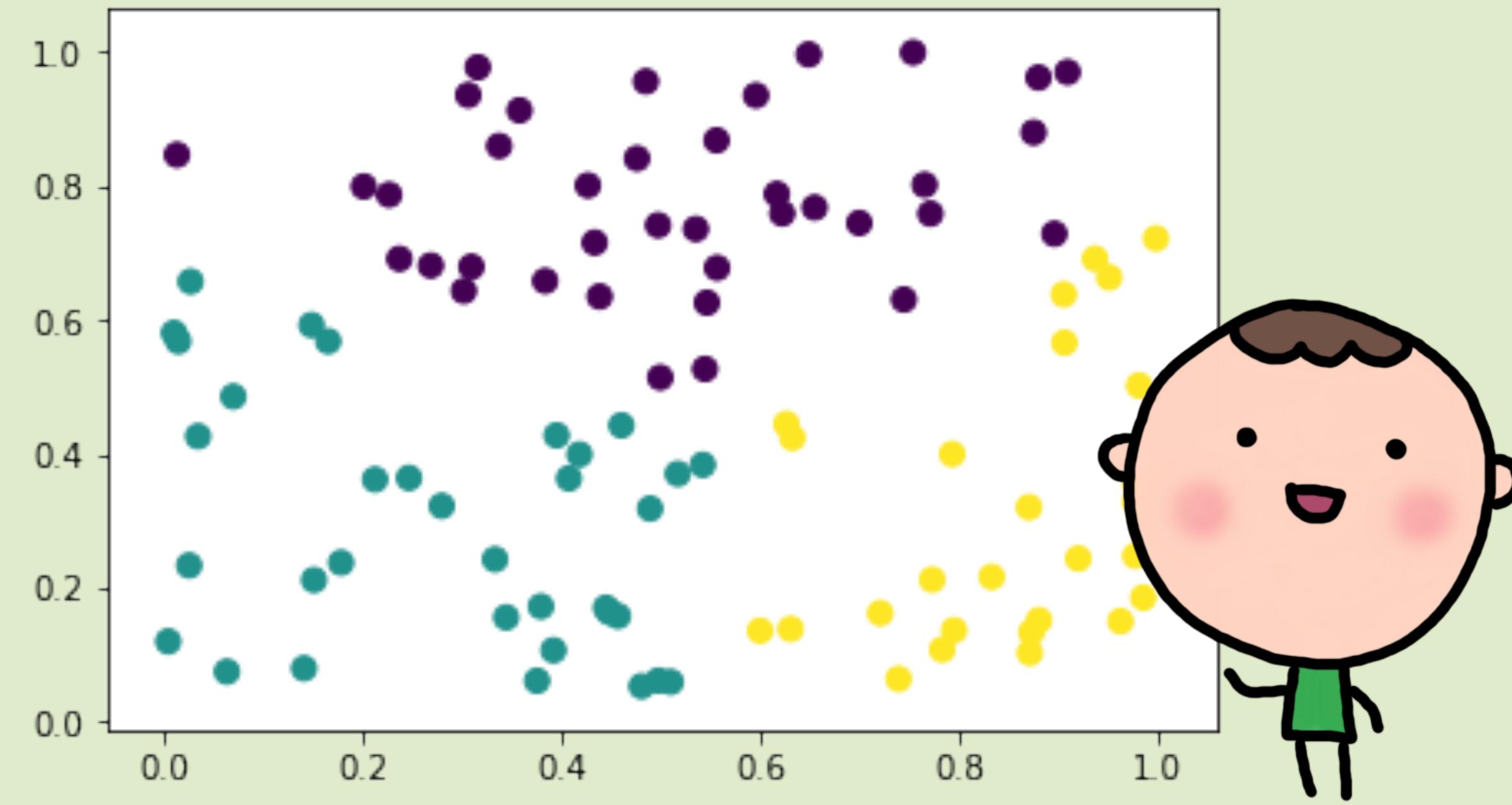
分類 0, 1, 2 放在  
`labels_` 裡



例子

# KMeans 基礎篇

畫出分類結果。

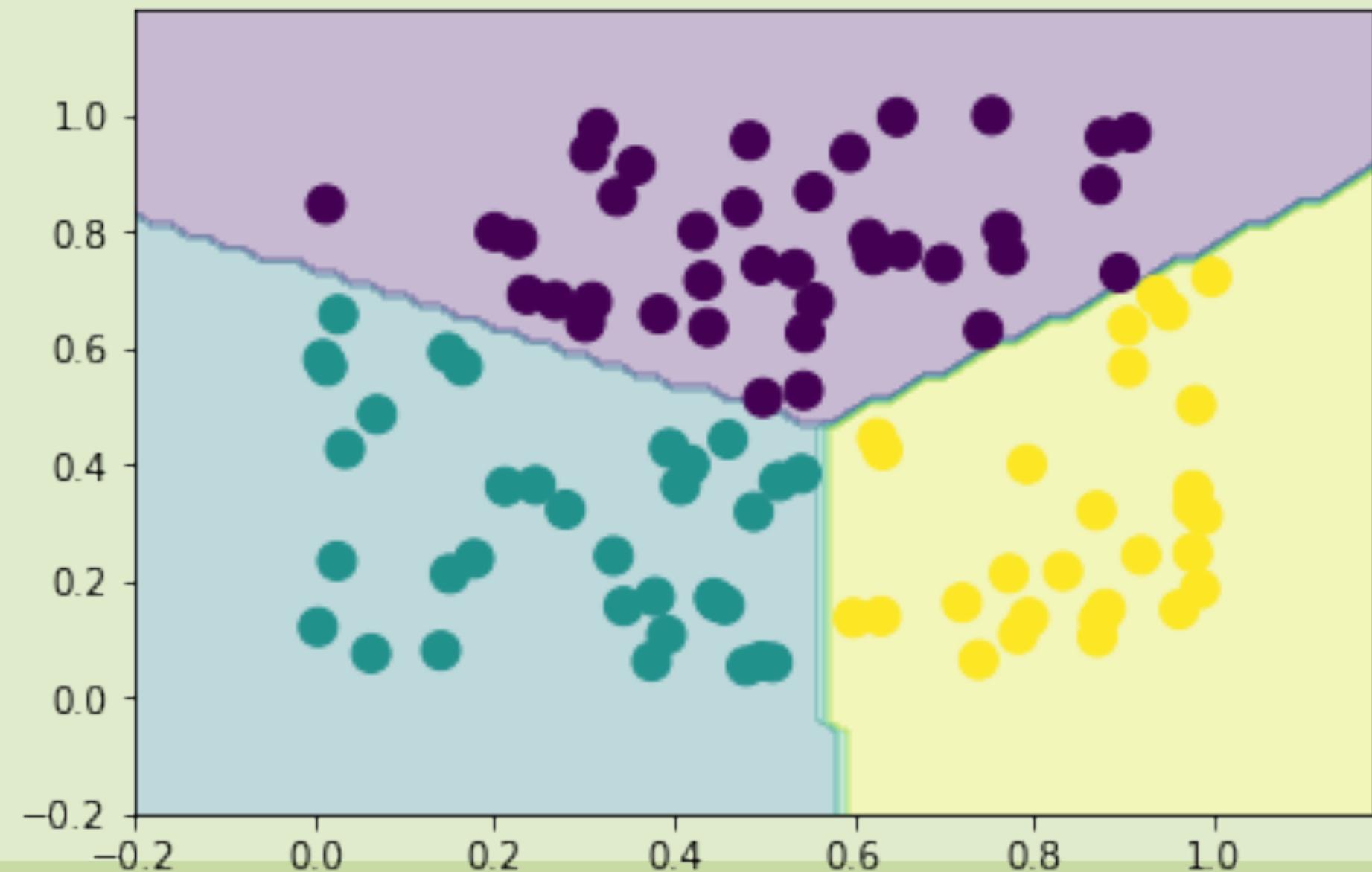


```
plt.scatter(x[:,0], x[:,1], c=clf.labels_, s=50)
```

例子

# KMeans 基礎篇

畫出分類結果。



```
x1, x2 = np.meshgrid(np.arange(-0.2, 1.2, 0.02), np.arange(-0.2, 1.2, 0.02))
z = clf.predict(np.c_[x1.ravel(), x2.ravel()])

z = z.reshape(x1.shape)

plt.contourf(x1, x2, z, alpha=0.3)
plt.scatter(x[:, 0], x[:, 1], s=100, c=clf.labels_)
```