資料分析導向的

Python 入門

蔡炎龍 政治大學應用數學系

0

簡介

Python 簡介

- 1991 由 Guido van Rossum 創
- 簡單、全方位、社群強大的程式語言
- 幾乎成為資料分析 No. 1!

Python 簡介

兩個重大系列

Python 2

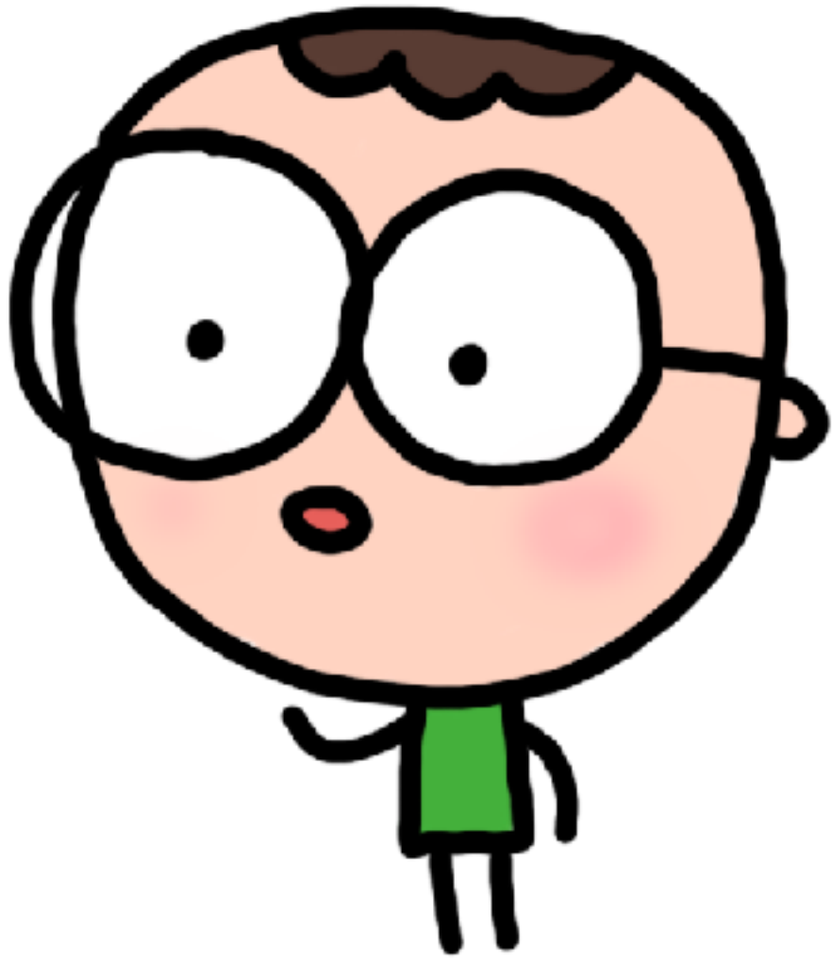
即將退休。

https://pythonclock.org/

Python 3

關於我

- UC Irvine 數學博士 (純數)
- 唸博士時知道 Python



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!

研習的內容

- Python 基礎
- Jupyter Notebook 環境
- Python 資料分析基本工具

Jupyter

Python

part 1

本投影片

NumPy

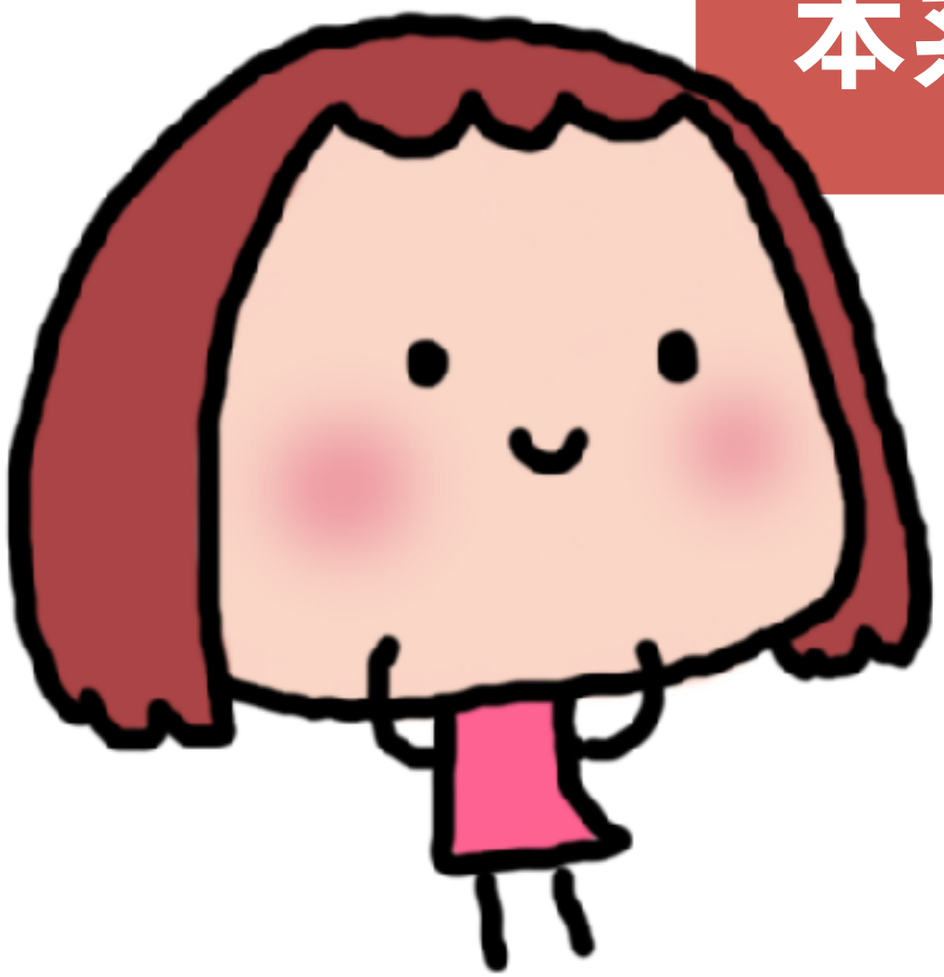
Matplotlib

Pandas

SciKit Learn

part 2 手把手打開 Python 資料分析大門

無痛走向下一步!



本系列

統計分析

機器學習

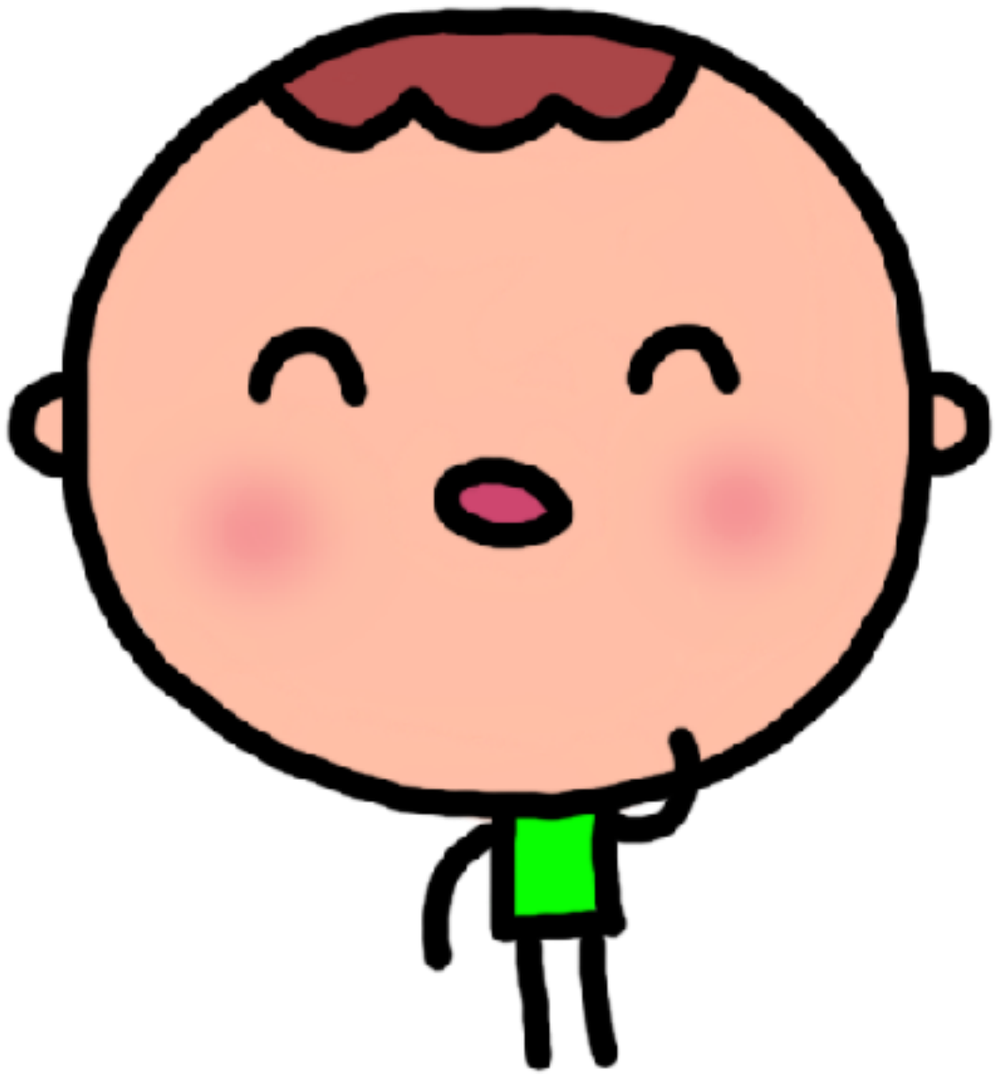
人工智慧

1

安裝

下載 Anaconda

https://www.anaconda.com/download/

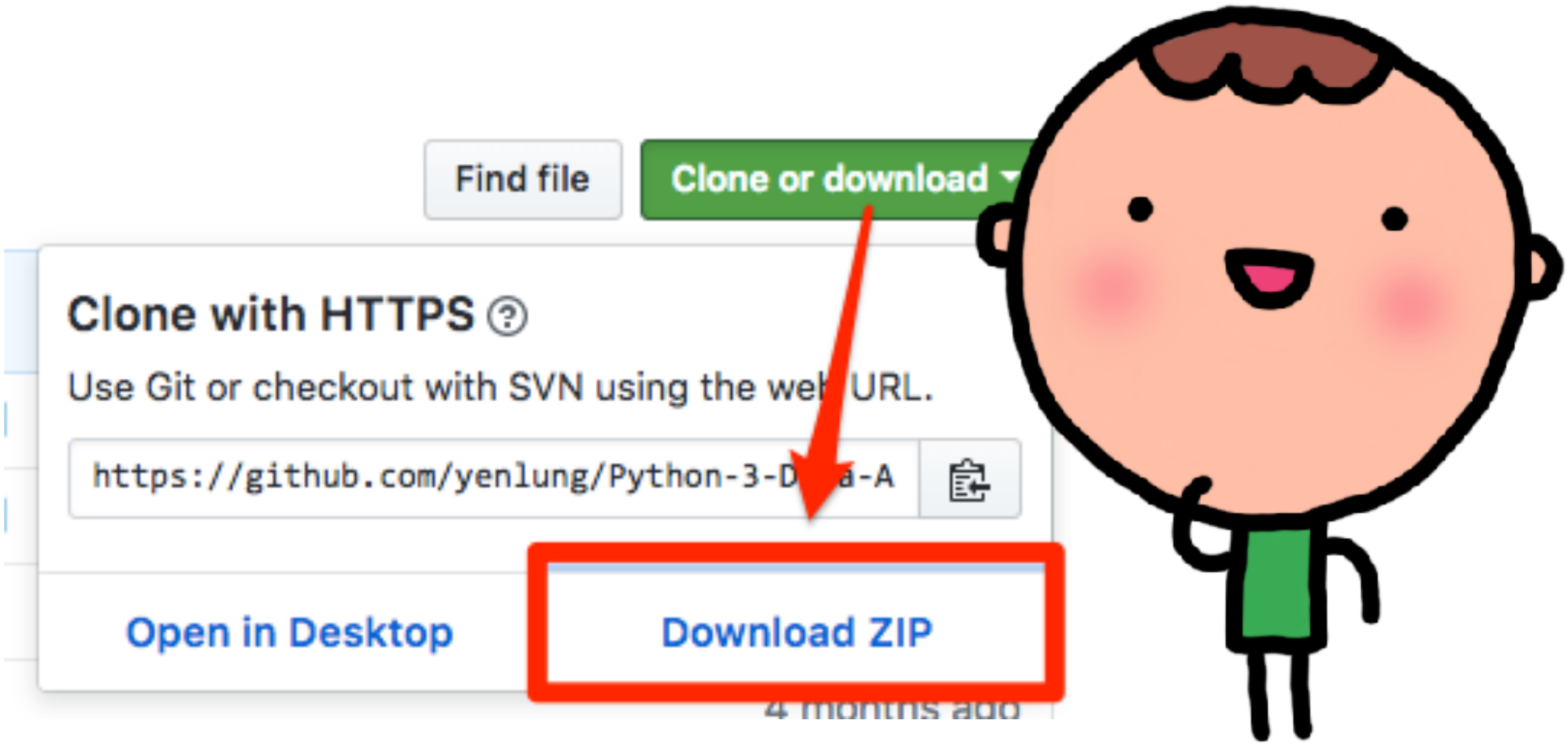


請裝 Python 3

的版本。

下載課程資料

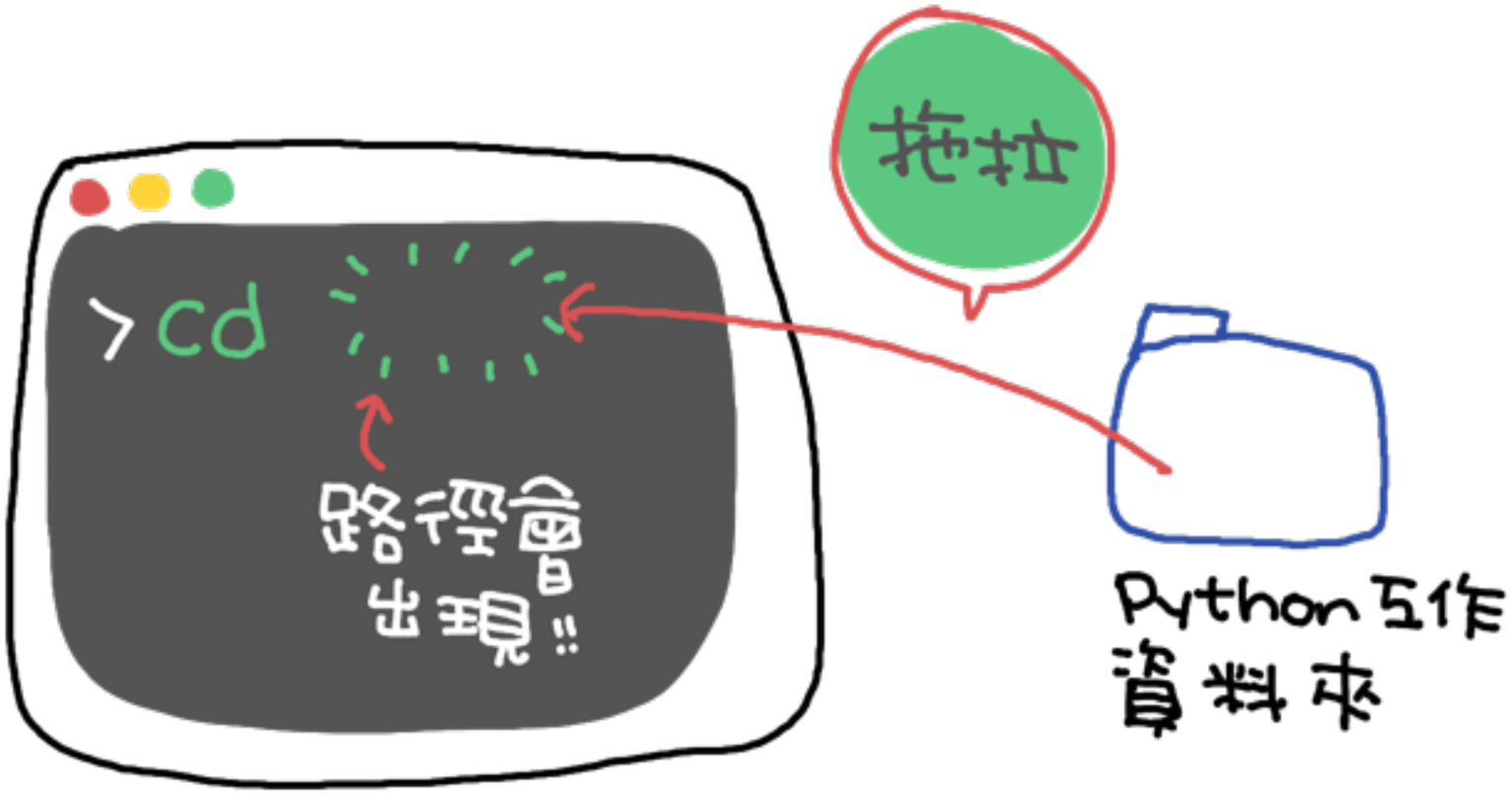
http://bit.ly/py3github



2

Jupyter Notebook

打開終端機 (Windows 下用 Anaconda Prompt)




```
> jupyter notebook
```

In[ ]:

記得在每個 cell 中是按

shift – enter

執行

Jupyter 有很多魔術指令, 都是以 % 開頭, 讓我們享有諸多方便功能。

我們的標準魔術指令

把 Jupyter Notebook 打造成一個方便的數學、計算、互動試驗場!

```
%pylab inline
```

「正確的」打法

我們暫時不要管太多, 但以下方式比較好。

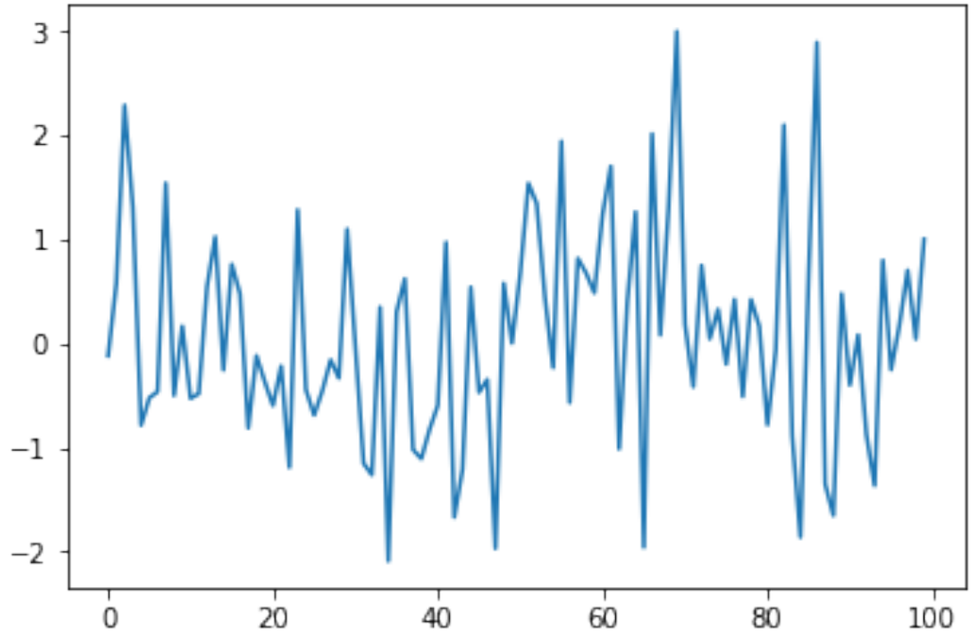
```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
```

使用 `numpy`, `matplotlib` 兩個套件

隨機取 100 點畫圖

從平均值為 0, 標準差為 1 的常態分布中, 隨機抽 100 個點, 畫出圖來。

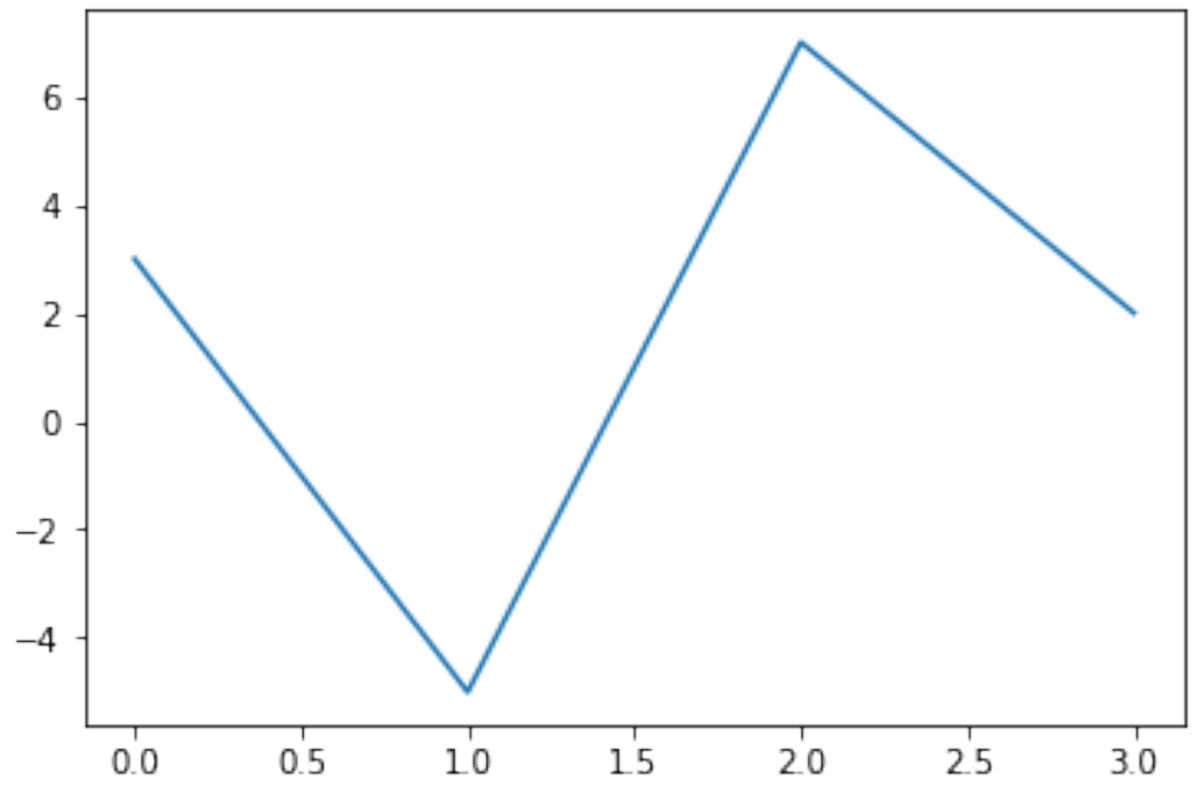
```
plot(randn(100))
```



畫出四個點

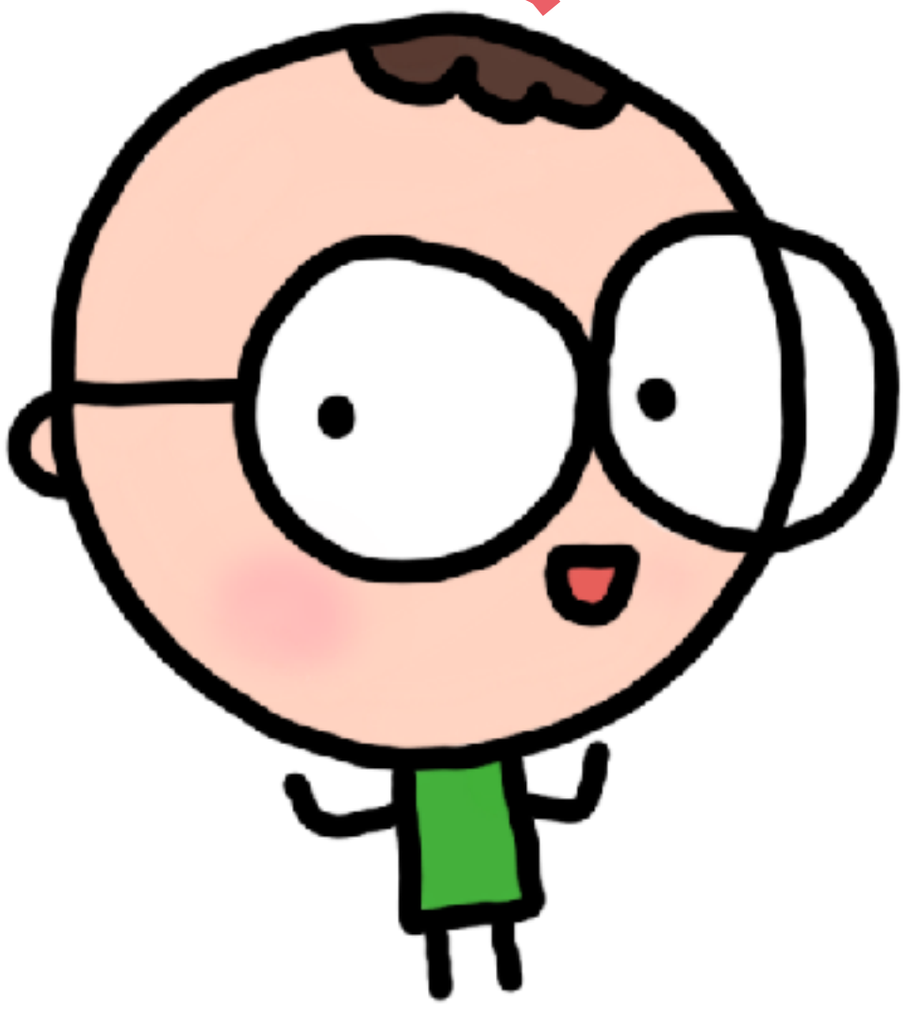
`plot` 用法是這樣子的:

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



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

畫圖語法



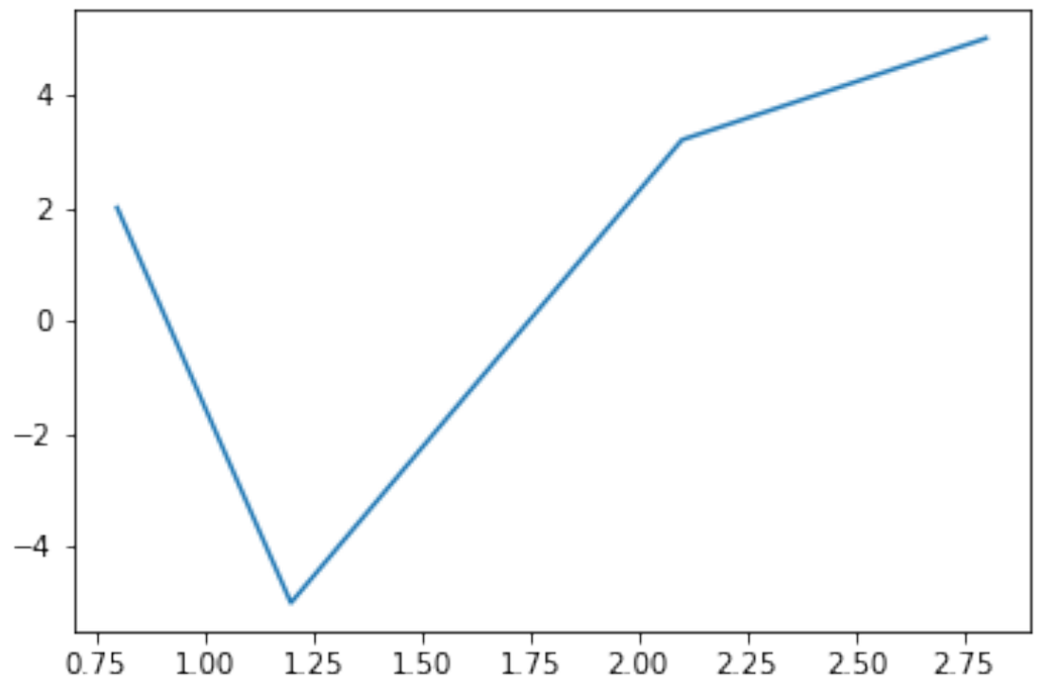
點的 x 座標 list (array)

```
plot(X,Y)
```

點的 y 座標 list (array)

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。




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

重點

亂數隨機抽

整數亂數的取法。

```
randint(a,b)
```

隨機抽取一個整數 k, 使得

$$a \leq k < b$$

重點

亂數隨機抽

標準亂數的取法。

```
rand(n)
```

隨機取 n 個亂數, 每個數字 k 範圍是:

$$0 \leq k < 1$$

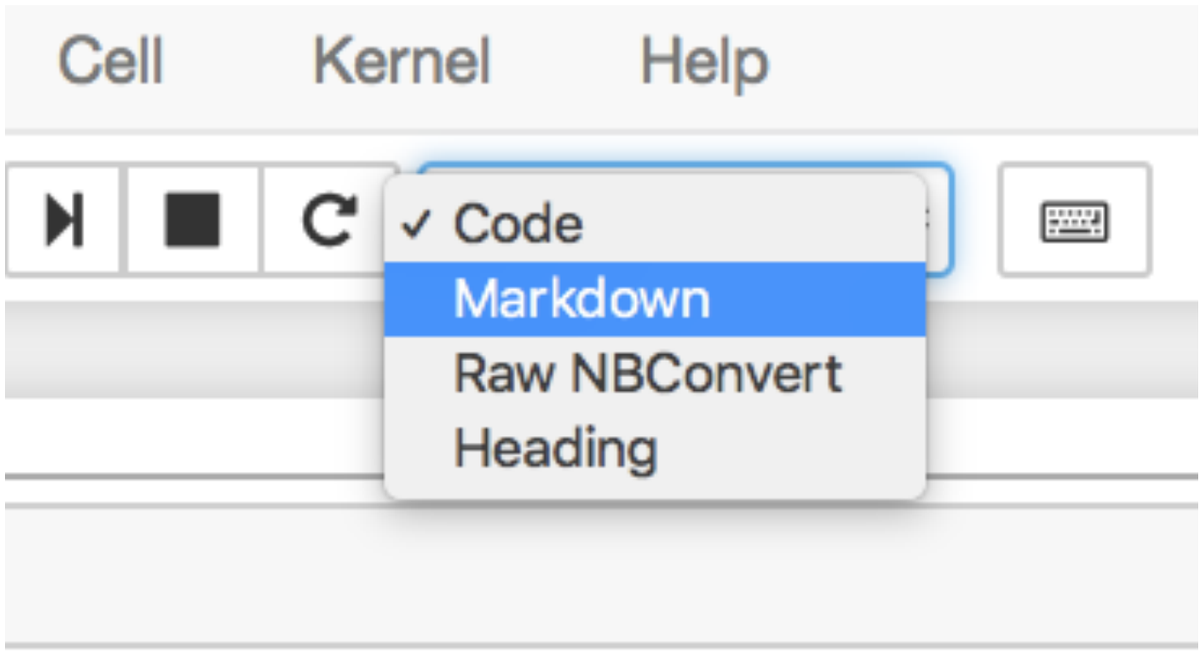
重點

亂數隨機抽

從標準差為 0, 變異數為 1 的常態分布取 n 個數字。

```
randn(n)
```

筆記神手 Markdown



用選單選 Markdown 格式,

就可以用 Markdown 語法

做筆記!

請搜尋 Markdown 語法, 而 Jupyter Notebook 也支援

用 LaTeX 打數學符號!

3

Python 計算機

基本運算

要算什麼就打什麼, 比計算機方便。

```
1 + 3*8
```

輸出：

25

次方

唯一要注意的是次方打法。

$$2^3$$

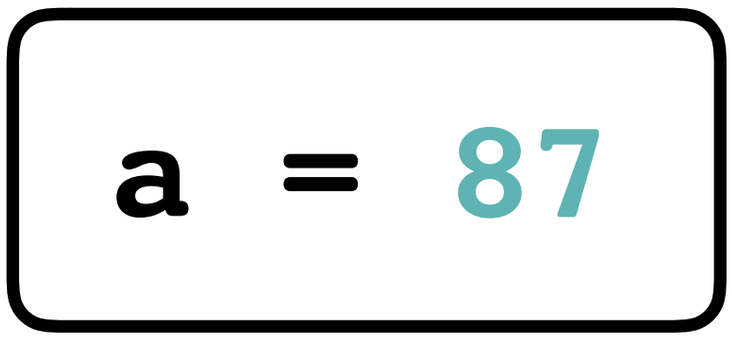
輸出：

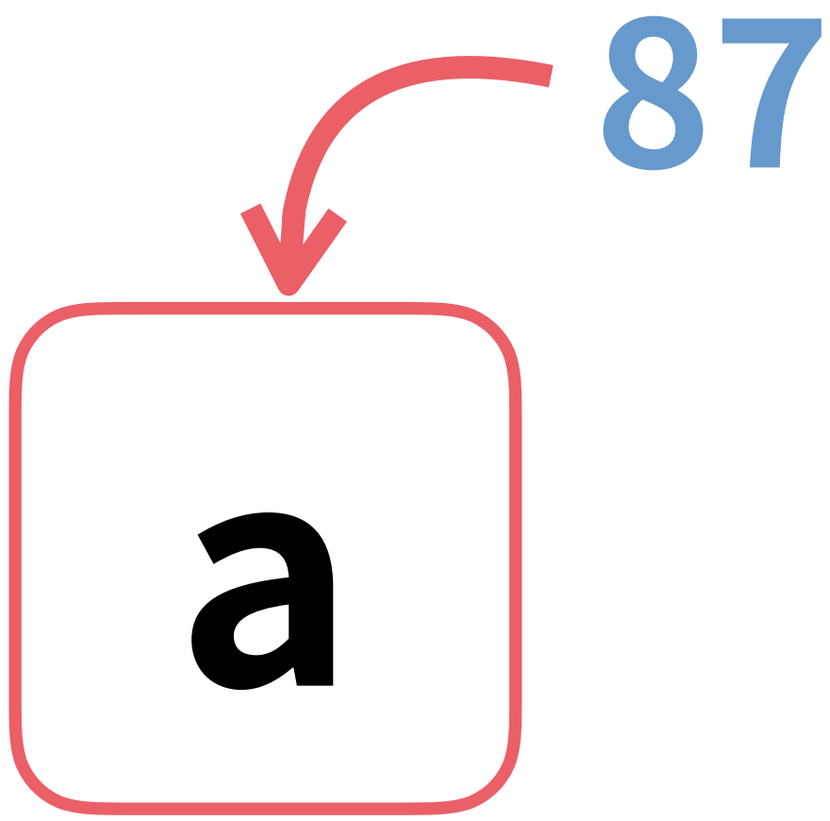
```
2**3
```

```
8
```

變數

程式的好處是可以用變數暫存。





例子

電池瓦時計算

為了飛行安全, 瓦時 (Wh) 太高的行動電池是有限制的。但平常的電池標示的是毫安時 (mAh) 和伏特電壓 (V), 我們怎麼換算呢?

$$Wh = \frac{mAh}{1000} \times V$$

例子

電池瓦時計算

假設 H 牌有個 10050 mAh, 12 V 的行動電源，換算為瓦時是多少呢?

```
10050/1000 * 12
```

例子

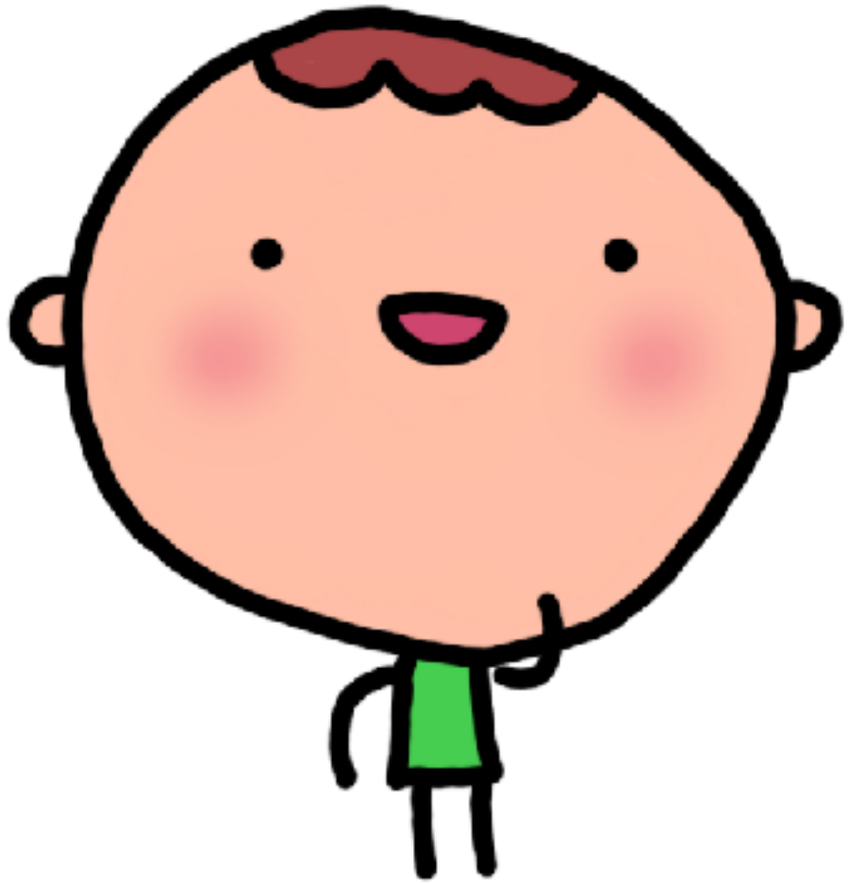
電池瓦時計算

同樣的問題令成變數可能更清楚。

```
mAh = 10050
V = 12
Wh = mAh/1000 * 12
```

更好的是寫成一個

函數



重點

函數的寫法

函數的寫法是這樣的。

```
def Wh(mAh, V):
    return mAh/1000 * V
```

重點

不同的資料型態

變數不只可以存數字 (事實上數字還可分整數、浮點數)。

`int`	`a = 3`	整數
`float`	`b = 3.2`	浮點數
`str`	`s = "hi"`	字串

重點

不同的資料型態

還有個很重要的串列 (list), 其實更符合我們計算需求的是陣列 (array), 我們先不做明確區分, 之後再說明。

list

```
L = [1,2,3,4]
```

串列

重點

不同的資料型態

最後有個叫字典 (dictionary) 的資料型態, 你就可以想成, 嗯, 一個字典。

字典

```
mydict = {'a':3, 'b':2, 'c':5}
```

重點

字典的查詢

```
mydict['a']
```

輸出: 3

重點

增加一筆資料

```
mydict['d'] = 7
```

結果:

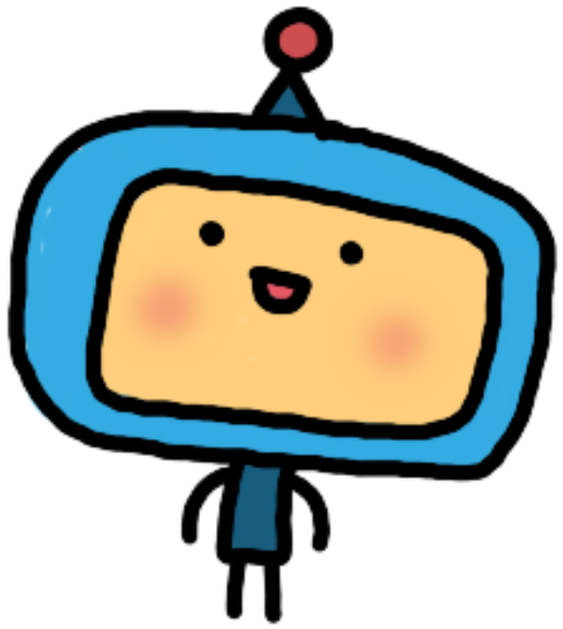
```
mydict = {'a': 3, 'b': 2, 'c': 5, 'd': 7}
```

串列、字串、陣列的

索引

+

切割



重點

索引分割

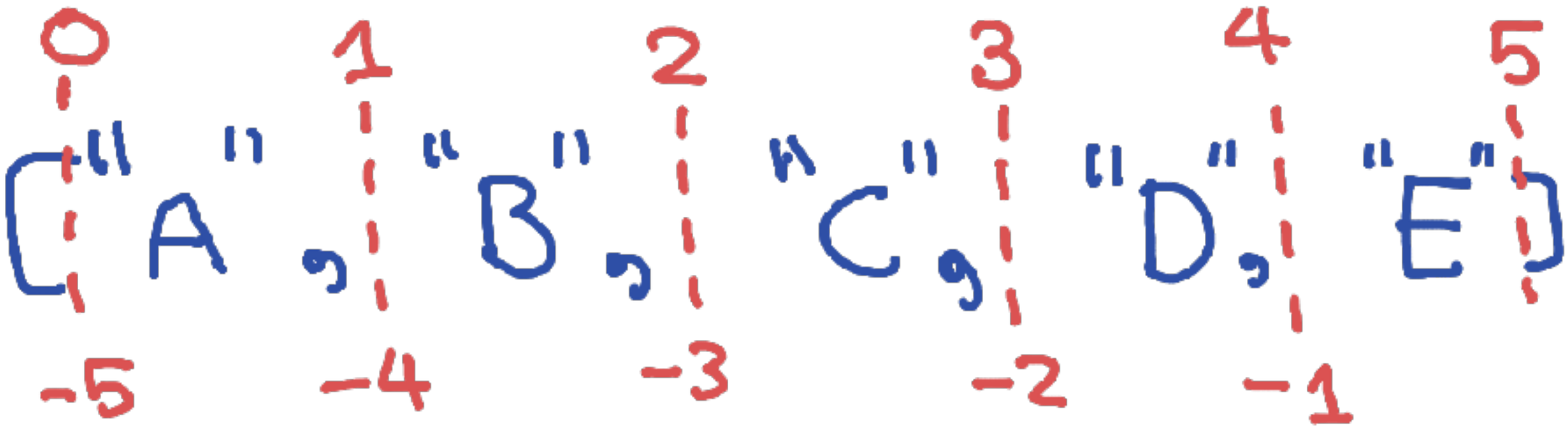
我們用個串列來說明, 但字串、陣列基本上是一樣的(除了陣列有更進階用法)。

```
L = ["A", "B", "C", "D", "E"]
```

重點

索引分割

Python 的切割點都在中間。

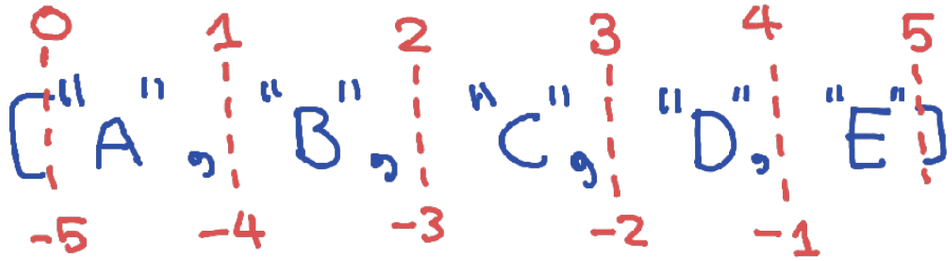


重點

索引分割

一些例子。

L




```
L[1]
```

→ 'B'

```
L[2:4]
```

→ ['C', 'D']

```
L[-1]
```

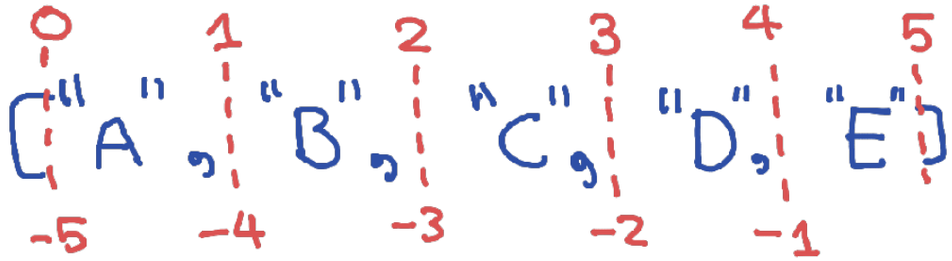
→ 'E'

重點

索引分割

一路到底。

L




```
L[:]
```

['A', 'B', 'C', 'D', 'E']

```
L[2:]
```

['C', 'D', 'E']

```
L[:-2]
```

['A', 'B', 'C']

重點

串列的擴充

```
L = [1, 3, 8, 9]
```

這個串列我們想「擴充」，可以用 append。

```
L.append(17)
```

L 會變成 [1, 3, 8, 9, 17]。

重點

串列的擴充

```
L = []
```

L 一開始設成空串列, 再慢慢擴充, 也是個常用的手法。

練習

複利計算

你因為急需一萬元現金,去找某錢莊借錢,老闆說「你看銀行信用卡利息動不動就 16%、18%。我們是『十天十分利』也就是 10%! 根本就慈善事業一樣了。」

複利公式

$$P = P_0(1 + r/k)^n$$

練習

複利計算

深怕你不會算, 老闆還很親切的教你「比如你借 1,000 元, 十天後就還 1,100 就好。」你覺得好像很合理, 於是借了 10,000 元, 複利計算, 每十天複利一次, 五年後你共欠多少?

寫個可以輸入本金、年利率、每年要複利幾次、借多少年的函數, 計算最後本利和是多少。

4

超炫的互動

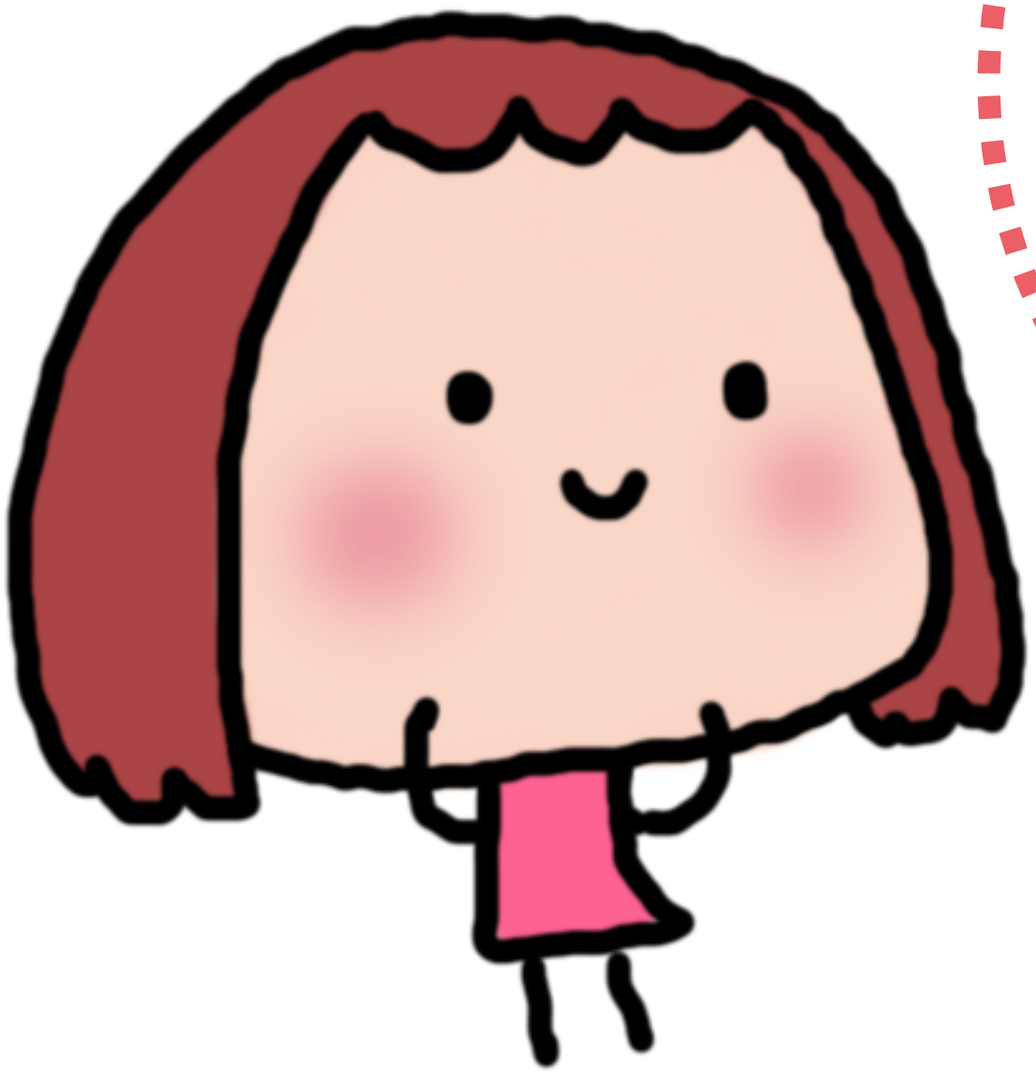
重點

互動準備

我們還沒學的套件指令讀入方式, 先用一下。

```
from ipywidgets import interact
```

這樣我們的 Python 就「多了」一個叫 `interact` 的指令函數。

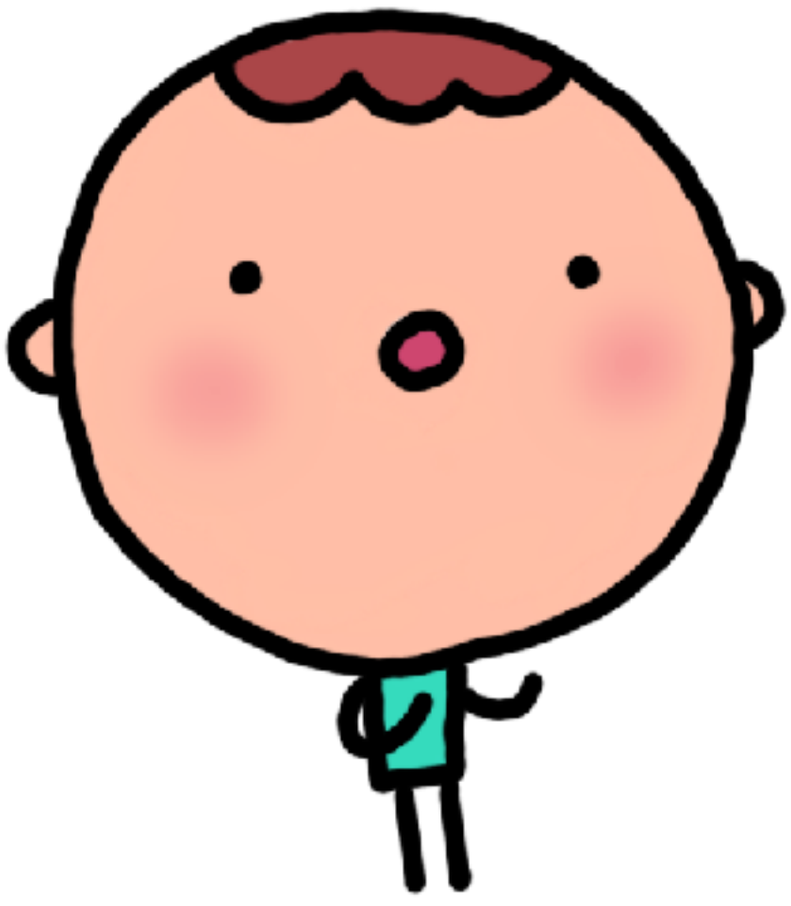


寫個函數

就能互動

例子

寫個簡單的函數



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

一定要用參數的, 這很難再更簡單。

例子

然後就互動!

```
interact(f, x=3)
```

這裡可以用任意的資料型態!

例子

有預設值的

來個「像樣點」的例子。

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

interact(move, n=(1,50))
```

重點

互動之二

我們介紹第二個互動指令。

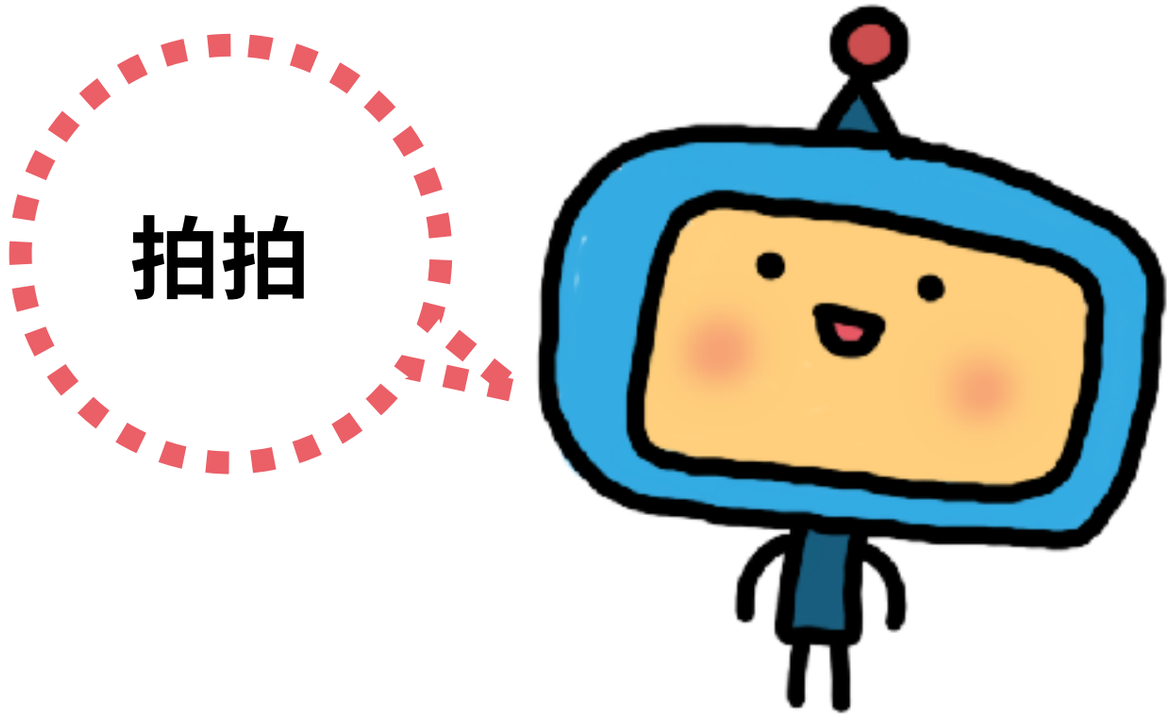
```
from ipywidgets import interact_manual
```

用法和 `interact` 基本上是一樣的, 試試有什麼不同?

練習

拍拍機器人

用 `interact_manual` 寫一個簡單的對話機器人, 不管使用者說什麼都回應「拍拍」。



5

畫圖

畫個

數學函數

重點

電腦畫函數就是描點法

假設我們要畫

$$y = \sin(x)$$

的函數圖形, x 範圍取 -10 到 10。

重點

電腦畫函數就是描點法

先來準備 x 座標的點, -10 到 10, 我們取, 比方說 100 個點。

```
x = linspace(-10, 10, 100)
```

這會產生一個陣列 (array)。

重點

電腦畫函數就是描點法

陣列很神的是, 100 個點算 *sin* 值可以一次算完!

```
y = sin(x)
```

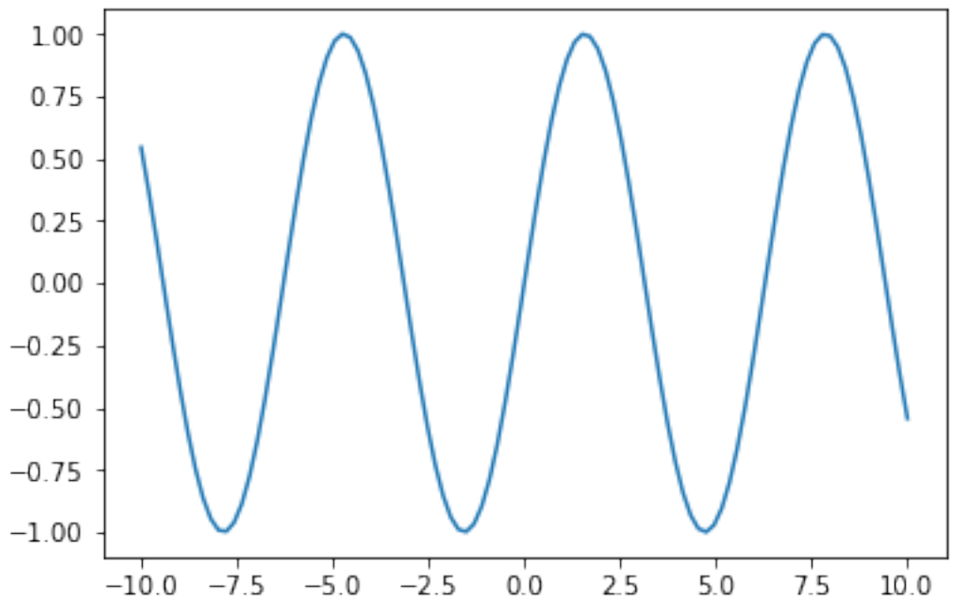
這樣 y 也是有 100 個點的陣列!

重點

電腦畫函數就是描點法

最後就是用 plot 畫出來!

```
plot(x,y)
```

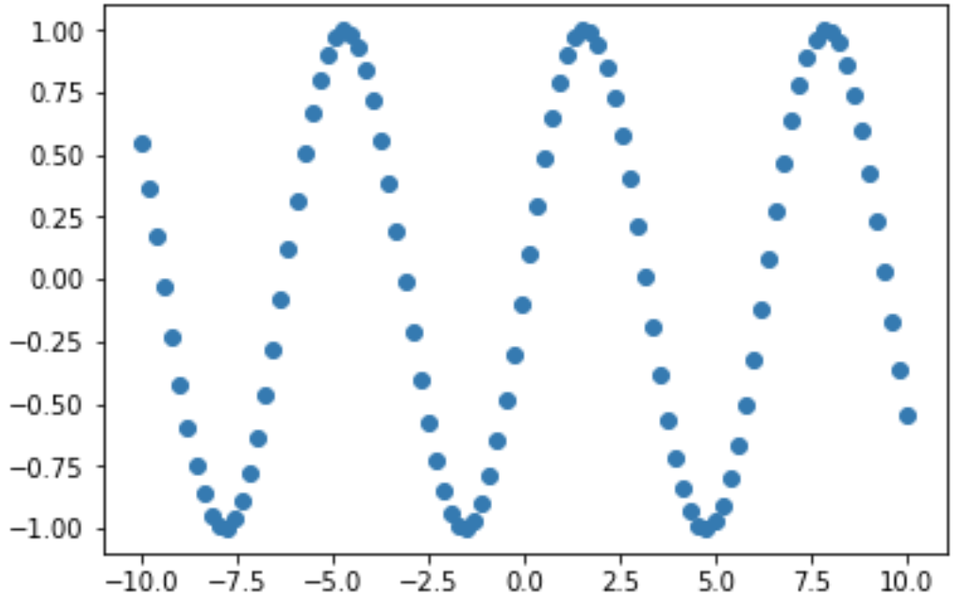


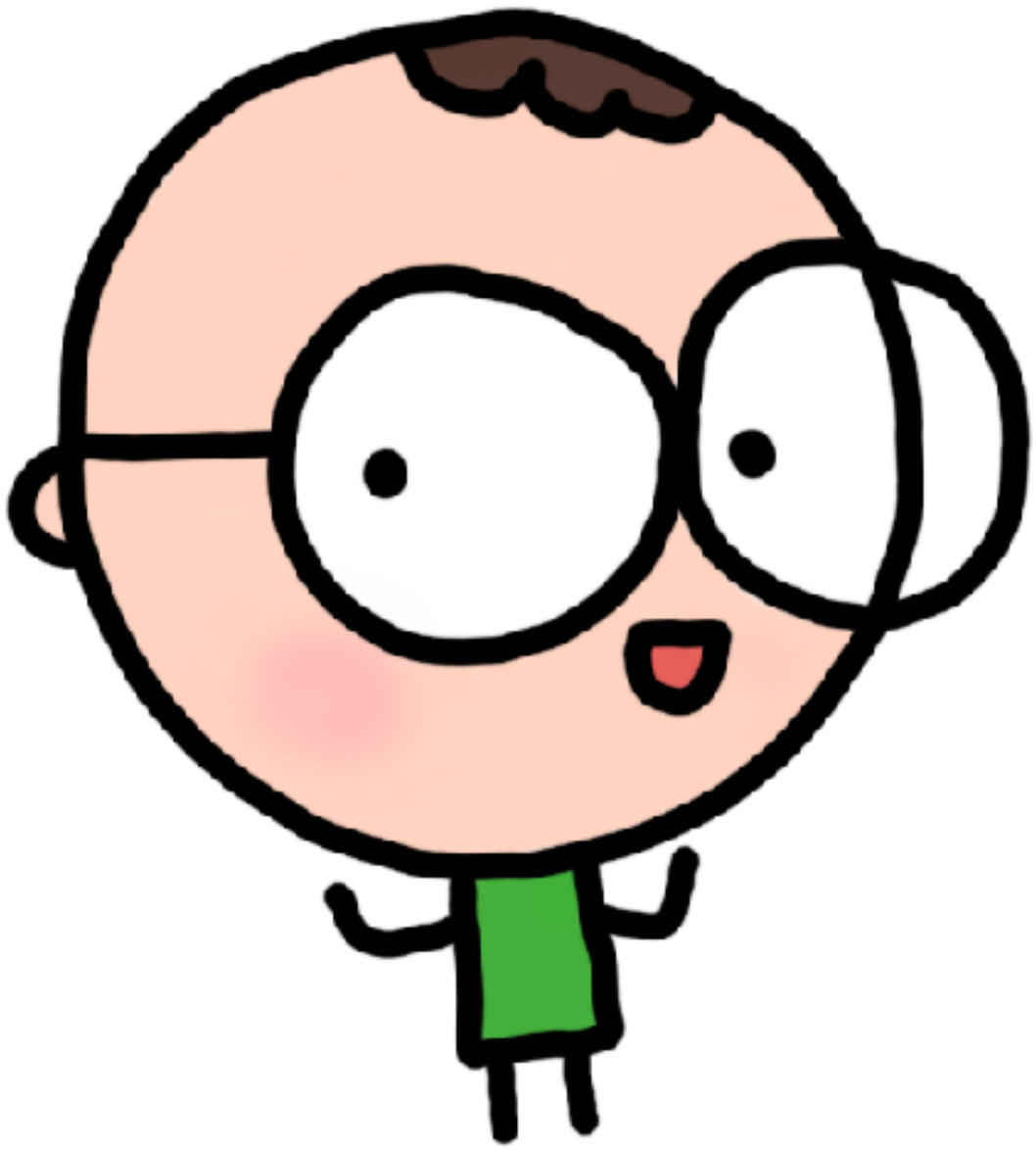
重點

電腦畫函數就是描點法

我們順便介紹另一種畫圖法 scatter。

```
scatter(x,y)
```





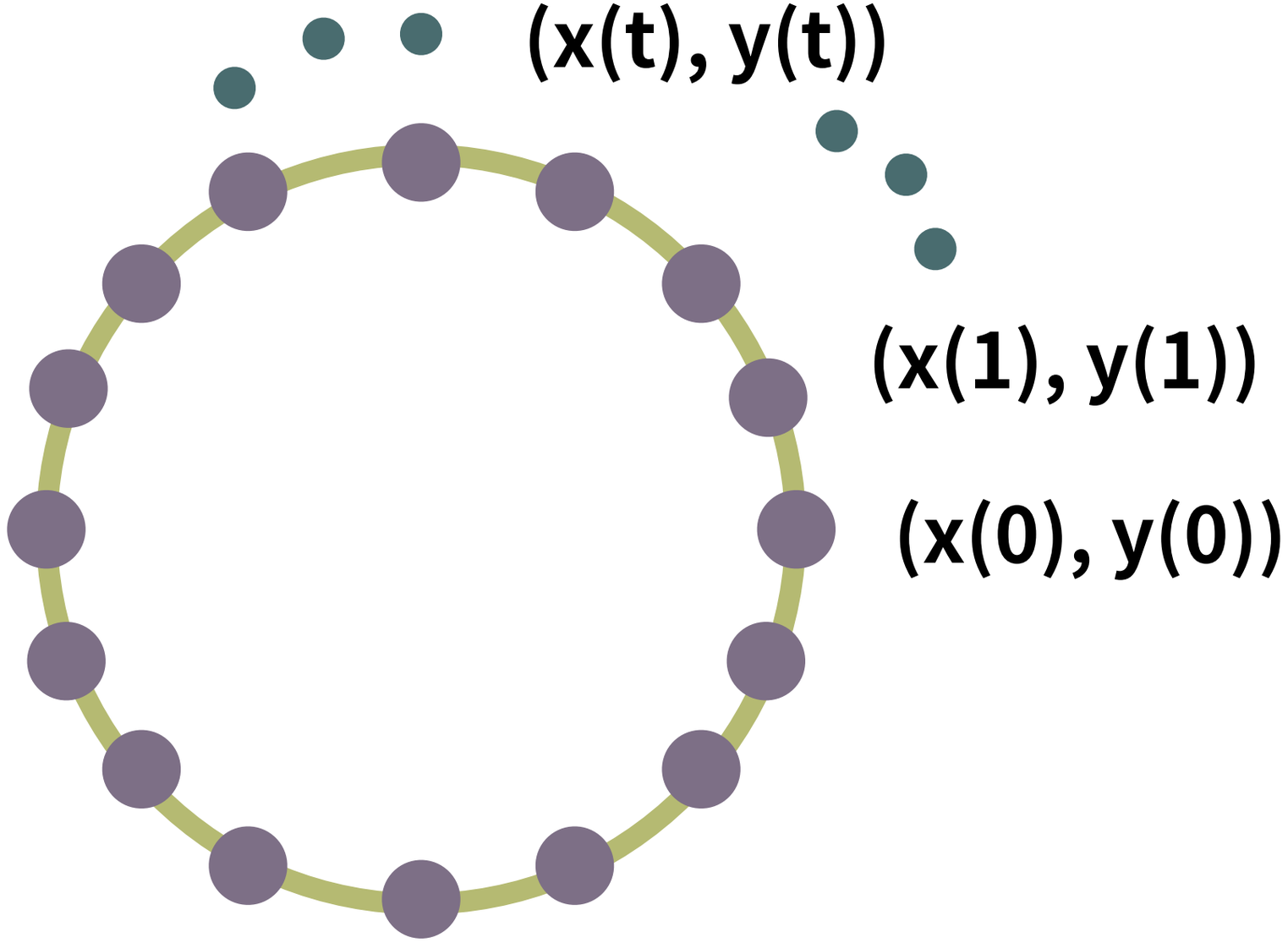
畫圖我們先學會兩大方式就好,

事實上這比我們想像還夠用!

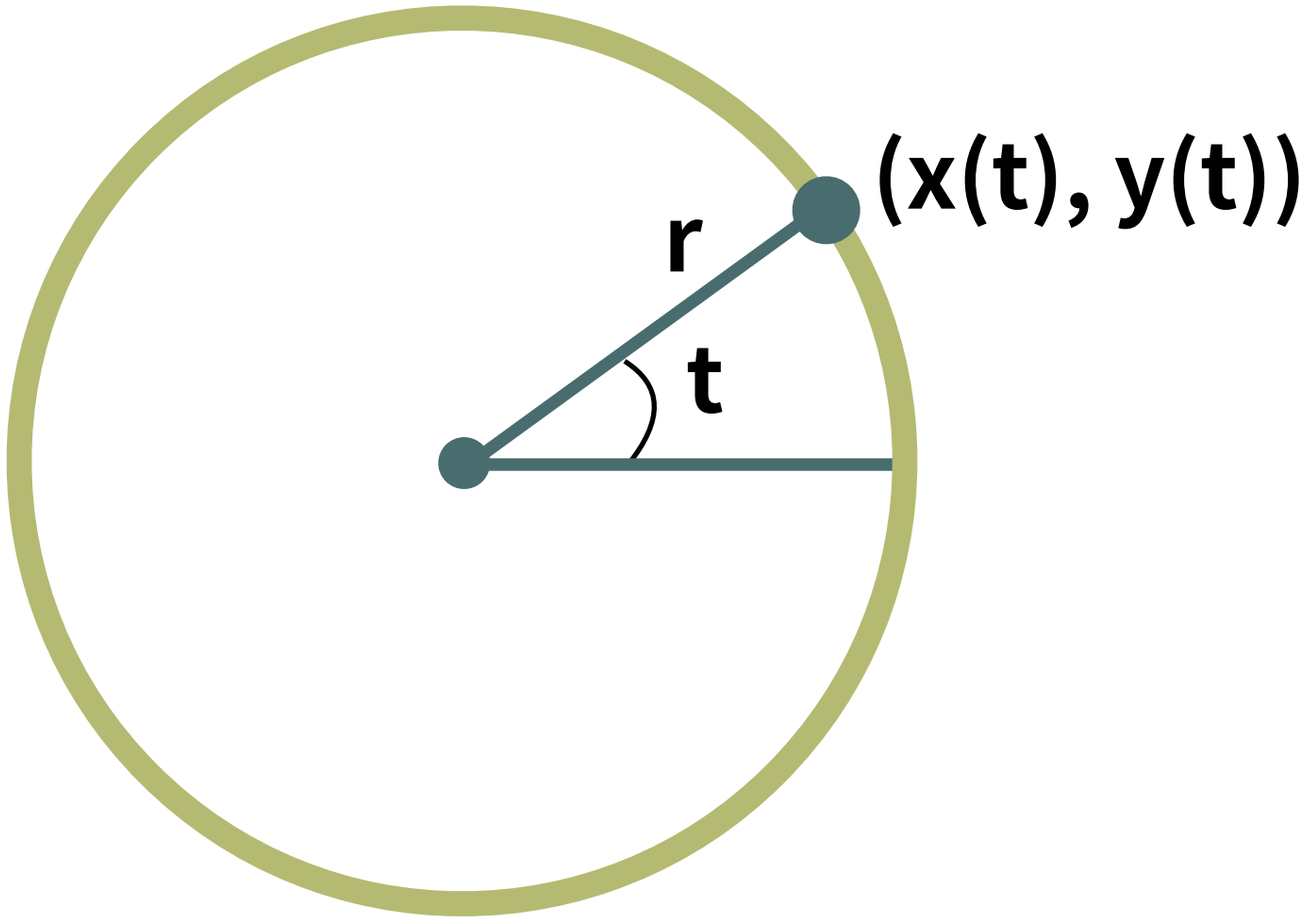
`plot`

`scatter`

matplotlib 畫個圓怎麼做?

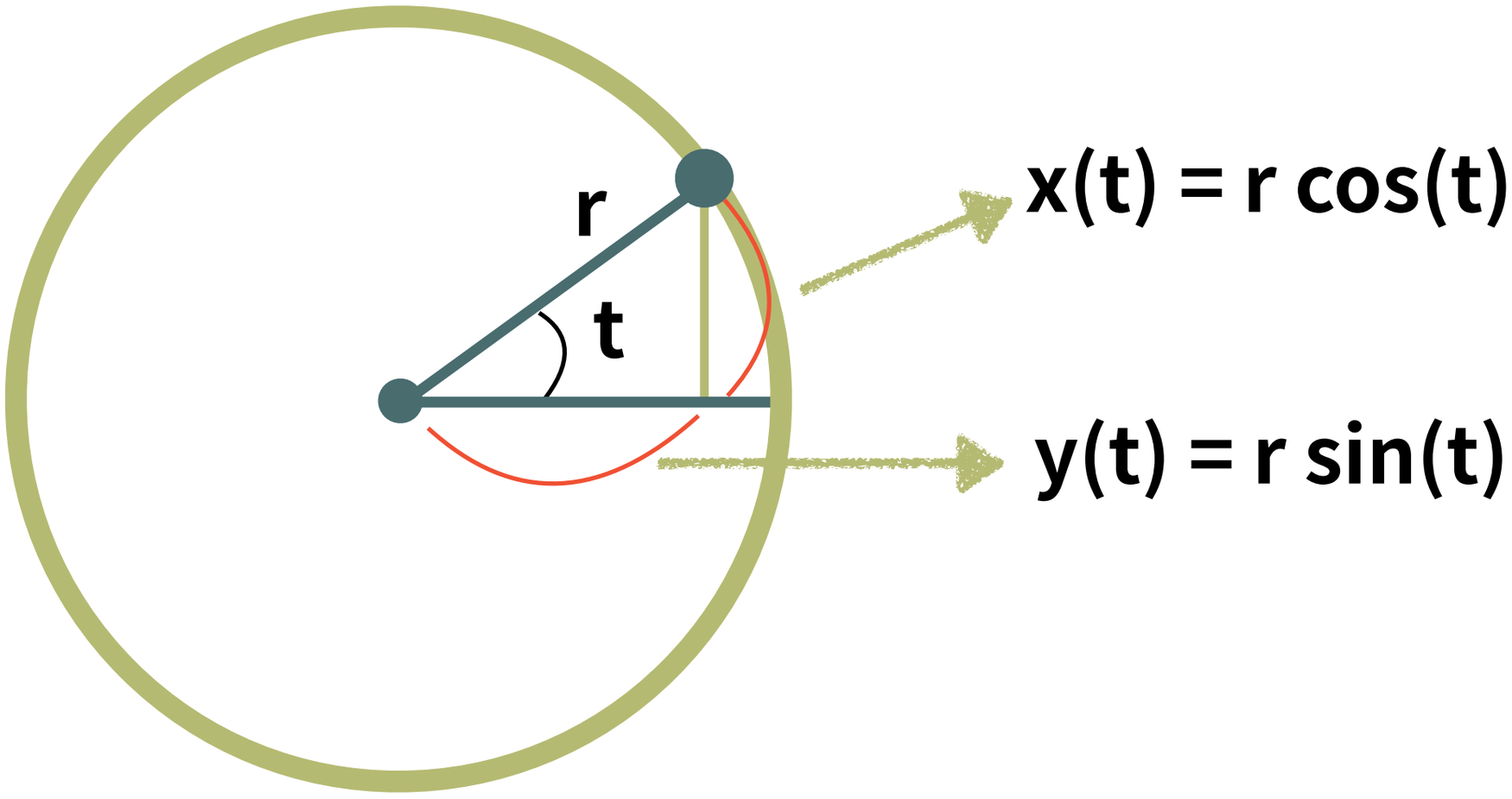


其實圓也就是一堆點連成的



半徑是 r, 時間 t 時我們的點剛好轉了 t 度。

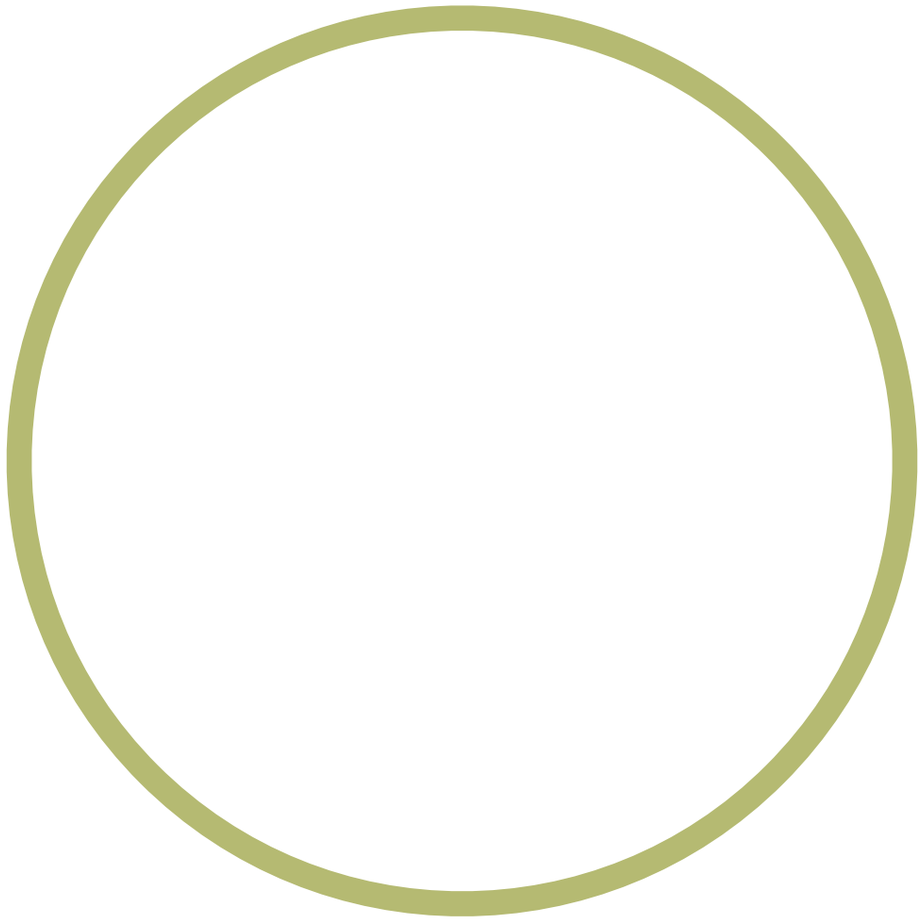
這時的 x(t), y(t) 座標是多少呢?



還記得極座標嗎?

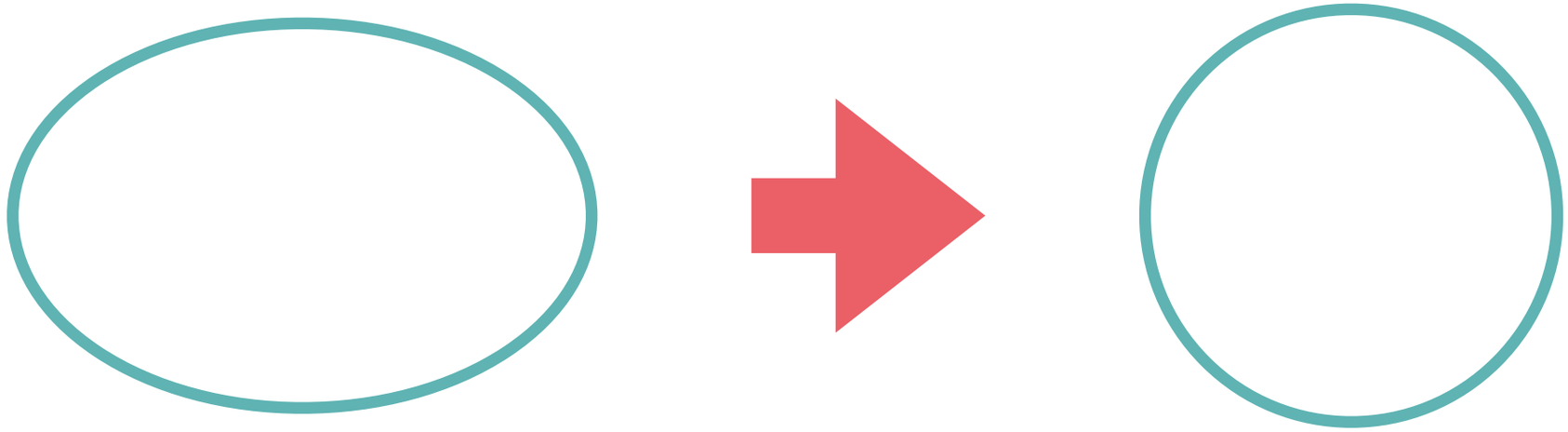
```
r = 3
t = linspace(-2*pi, 2*pi, 200)

x = r*cos(t)
y = r*sin(t)

plot(x, y, lw=3)
```

寫成程式是這樣



真的畫出一個圓!

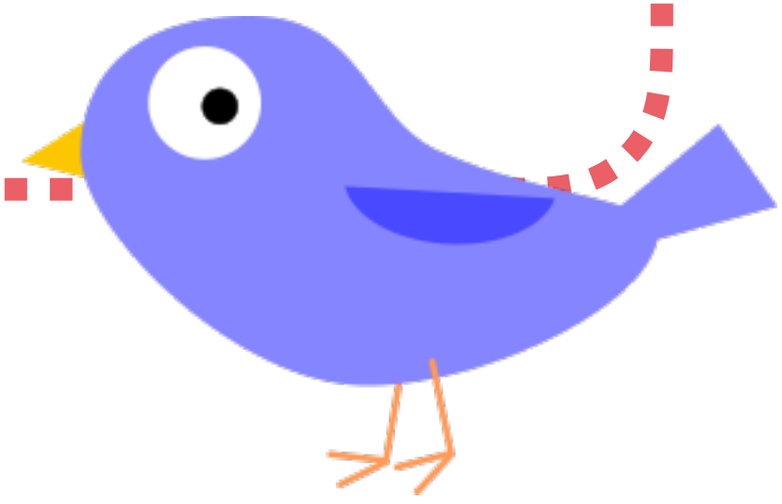
(其實我有點騙你)



```
ax = gca()
ax.set_aspect('equal')
```

我騙你的地方是，　畫出來怎麼看都是橢圓, 要修正就是在畫圖下 plt.plot 指令那需要修正。

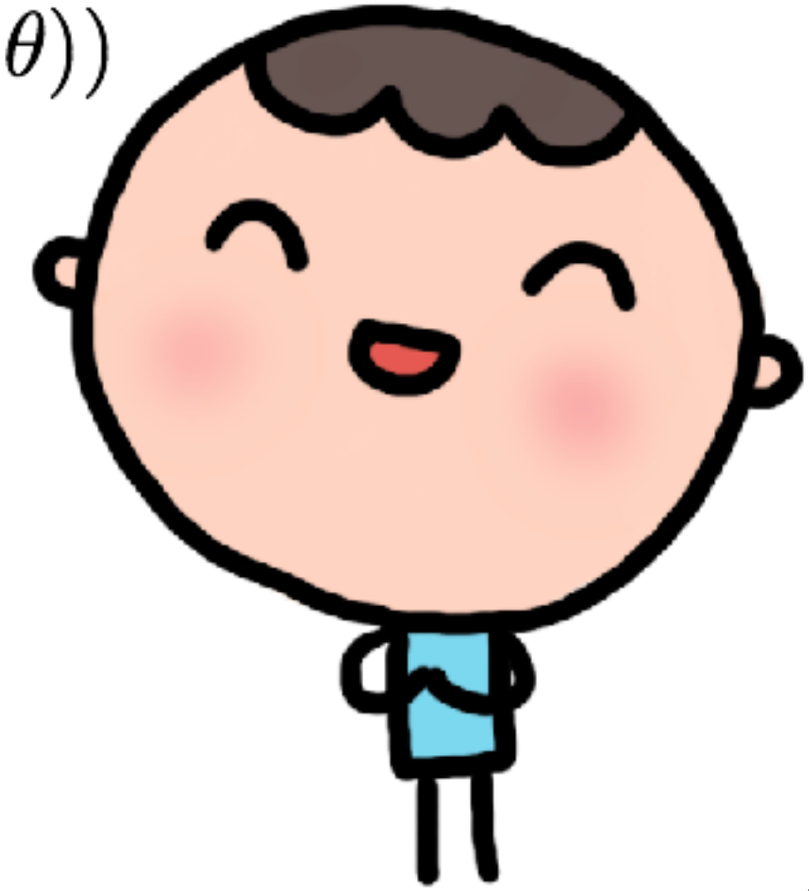
如果 r 不是故定的呢? 也就是它會變長變短, 是不是會畫出什麼特別的圖呢?



練習

畫出漂亮曲線

我們來試試, 會變化的 r 會畫出什麼樣的曲線? 這很容易, 比如你想要「只有中文才有的」感人笛卡兒愛情故事中出現的 $r = a(1 - \sin(\theta))$

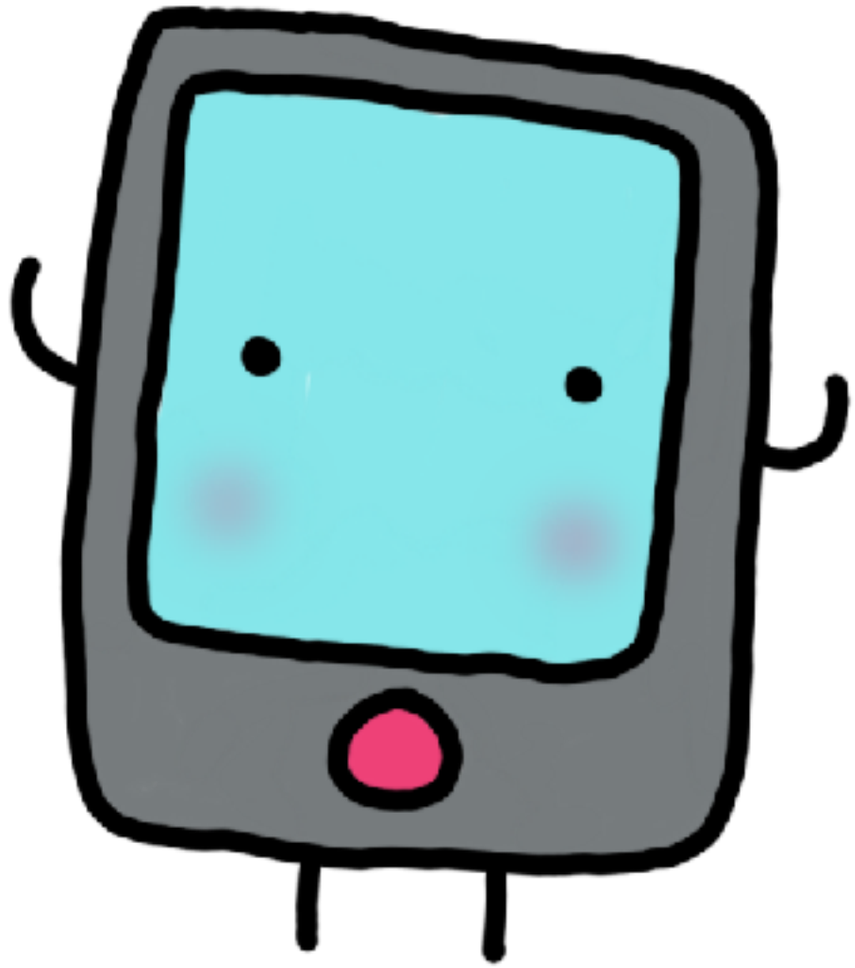


在前面的點設定只要改成...

```
r = 3*(1-sin(t))
x = r*cos(t)
y = r*sin(t)
```

6

條件判斷



重點

條件判斷

a = 3, b = 5

```
a>b
```
→ False

```
b>=5
```
→ True

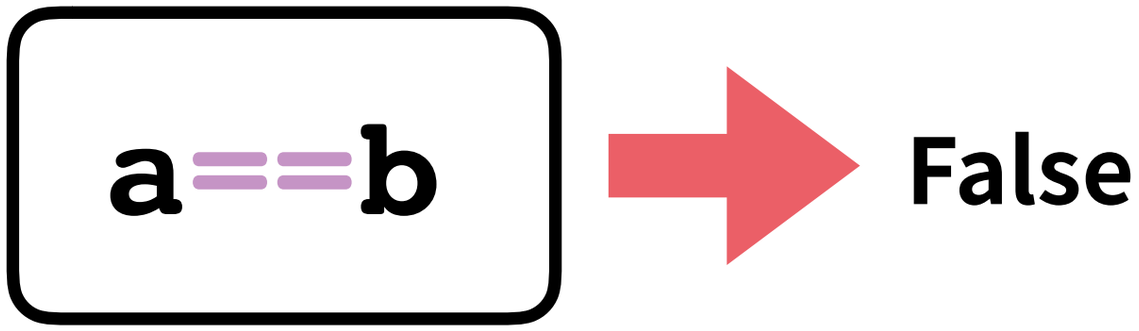
```
a!=b
```
→ True

重點

條件判斷

特別注意相等的判斷要兩個等號!

a = 3, b = 5



重點

陣列可以一次判斷!

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

```
L>0
```

輸出:

```
array([ True, False,  True,
True, False])
```

重點

陣列可以一次判斷!

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

```
L>0
```

輸出:

```
array([ True, False,  True,
True, False])
```

例子

遊戲的回應

假設你寫了個遊戲, 玩家會得到 0-5 的一個評等。你想依下表來回應...

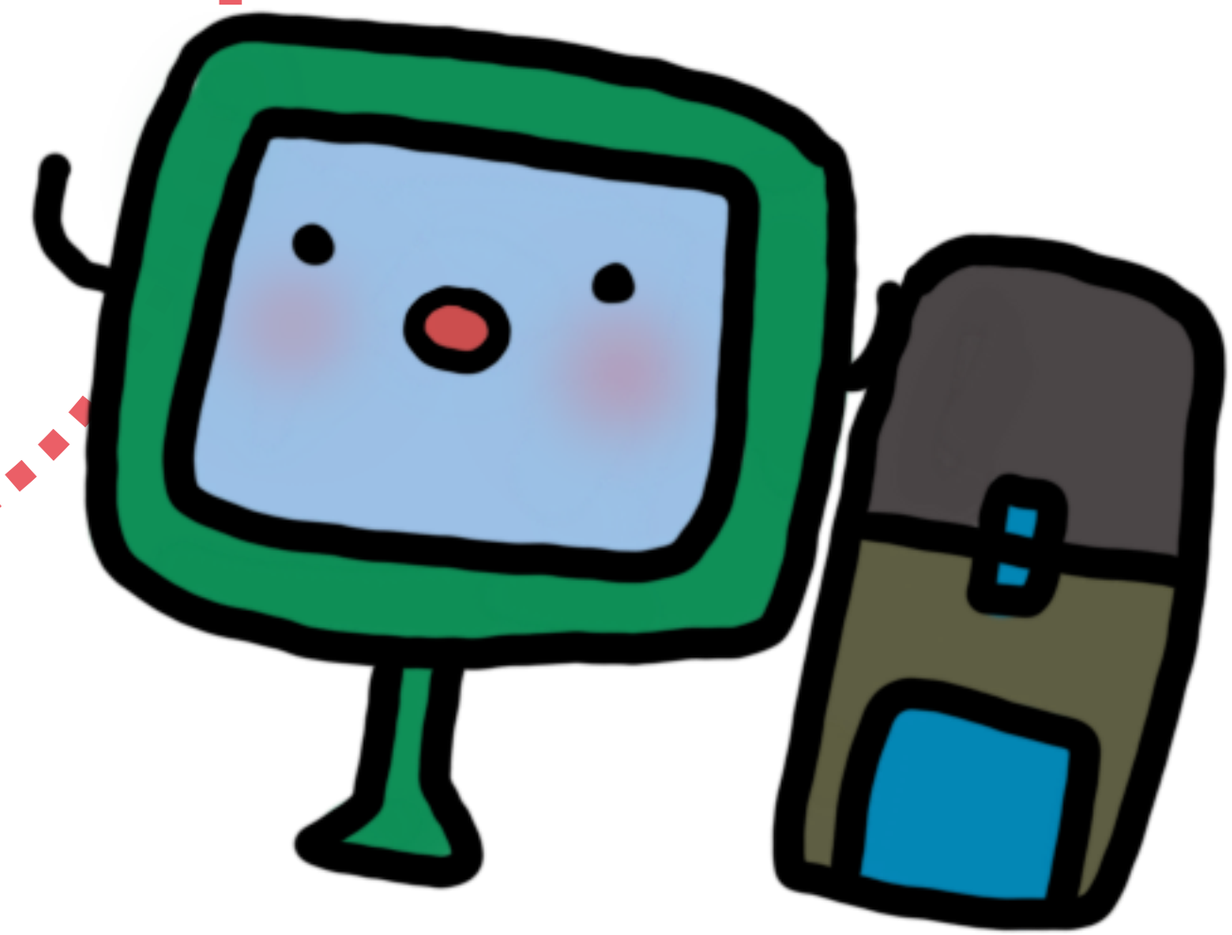
5	太棒了!
3-4	不錯!
0-2	加油好嗎?

例子

遊戲的回應

```
s = 4

if s==5:
    print("太棒了!")
elif s>=3:
    print("不錯!")
else:
    print("加油好嗎?")
```

input



和電腦互動

重點

input 用法

要等使用者輸入一些資料, 可以用 `input` 指令。

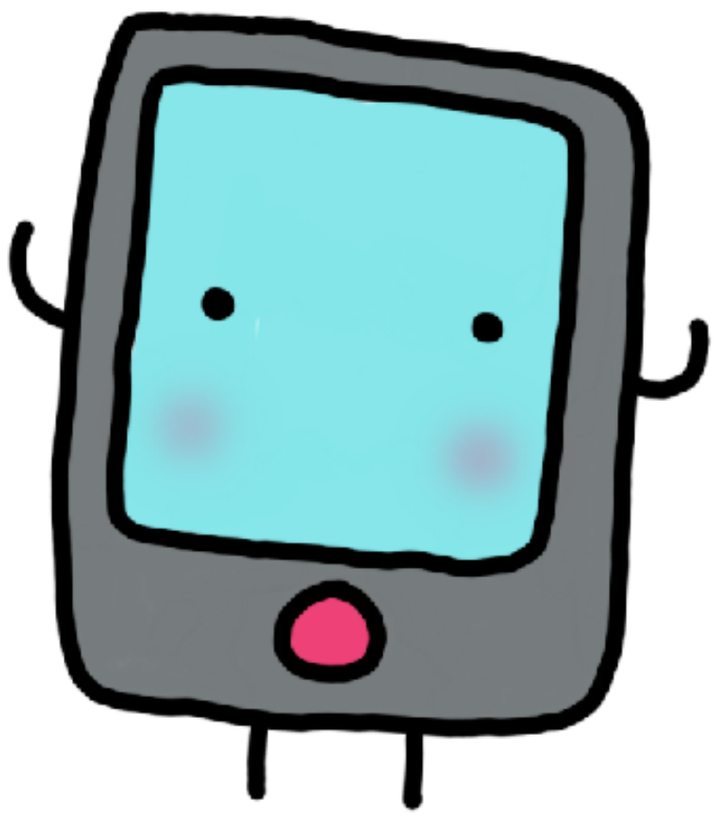
```
s = input("請輸入: ")
```

注意這時得到的一定是字串, 可用例如 `int(s)` 改成整數。

練習

匯率換算

用 `input` 讓使用者輸入多少美金, 換算成台幣顯示出來。



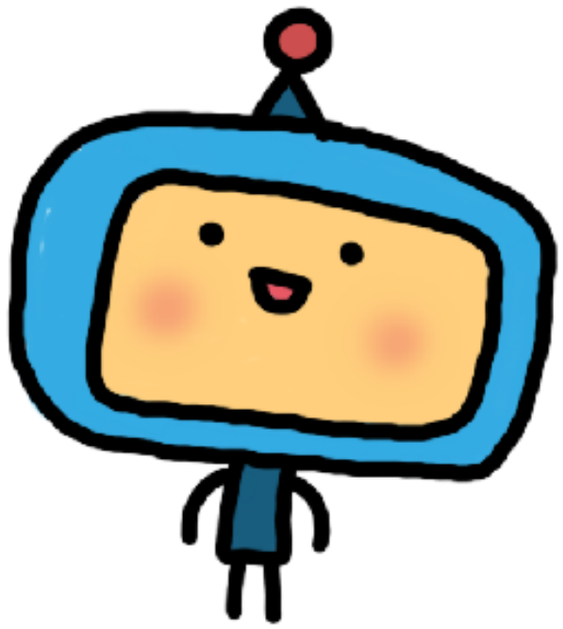
7

迴圈

迴圈就是讓電腦重覆做一些事的方法, 一般有 `for` 和 `while` 兩種方式。

`for`

`while`



for

就是從某個串列中, 一個一個拿出來。

```
L = [1, 2, 3]

for i in L:
    print(i)
```

輸出：

```
1
2
3
```

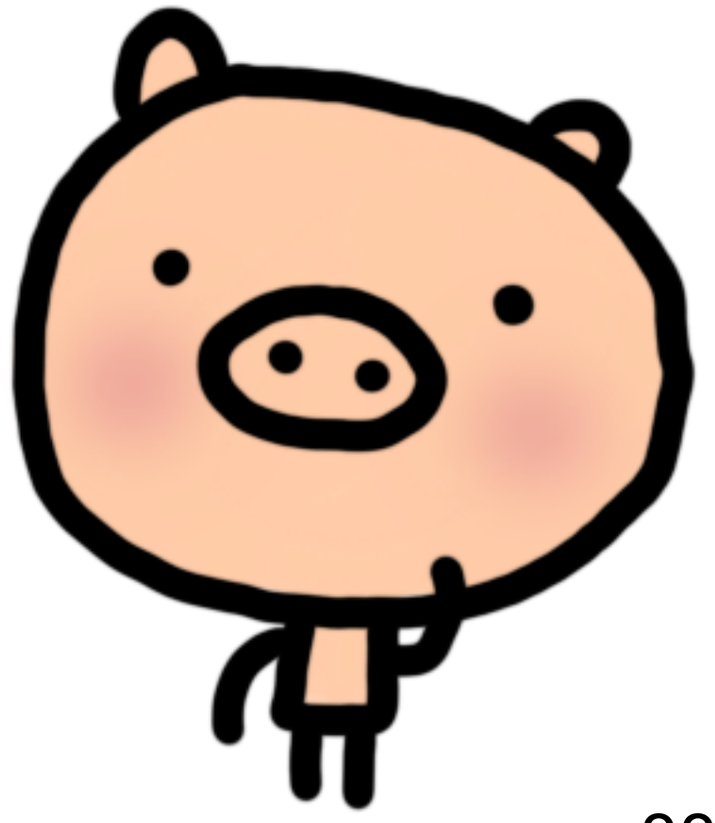
while

while 是做到條件不成立了才停止。

```
k = 0

while k<5:
    k = k + 1
    print(k)
```

輸出：

```
1
2
3
4
5
```



練習

冰雹數列

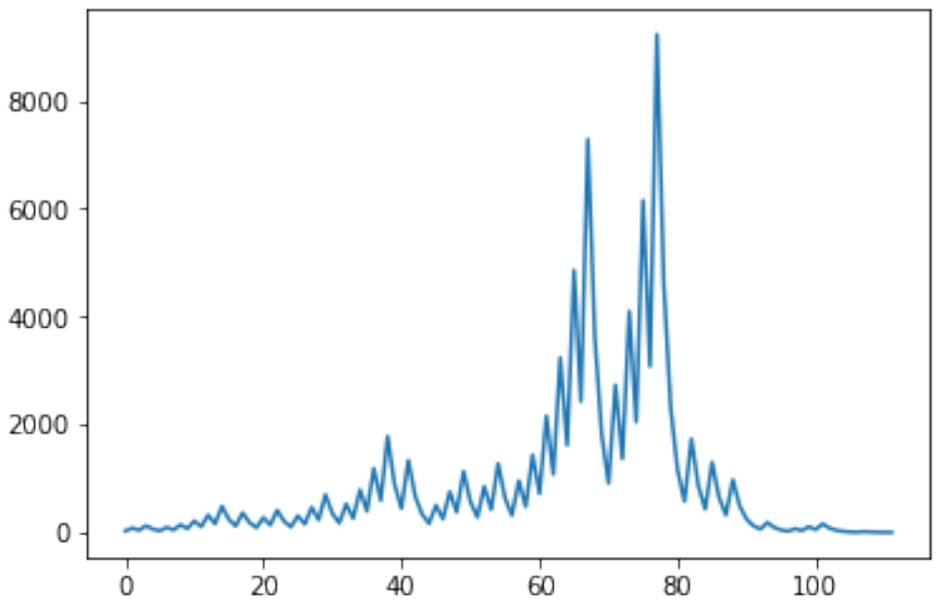
有一個數列 $\{a_n\}$, 是這樣決定的: 如果目前的數字是, $a_n = k$ 那下一數字是用下列方式決定的。

$$a_{n+1} = \begin{cases} \dfrac{k}{2}, & \text{if } k \text{ is even} \\ 3k+1, & \text{if } k \text{ is odd} \end{cases}$$

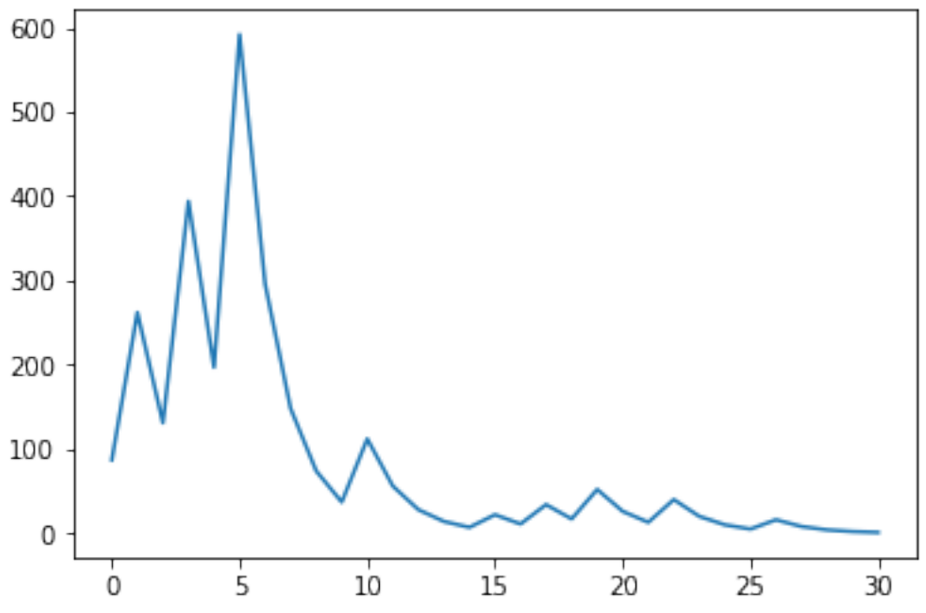
這種數列叫「冰雹數列」。給定任意起始值正整數 k, 最後都數變成 1。用 Python 寫個程式驗證看看!

練習

冰雹數列



k=27

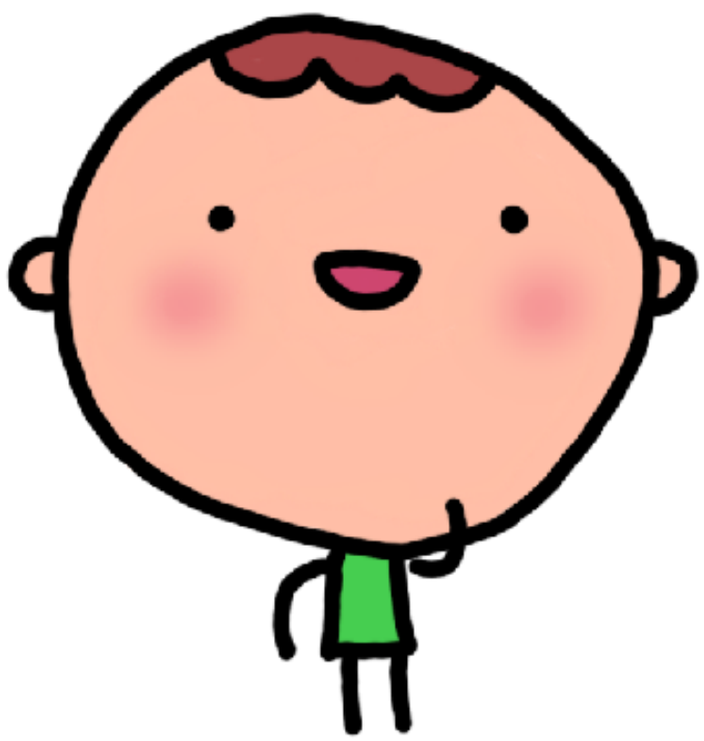


k=87

練習

猜數字遊戲

電腦隨機取一個 1-100 的數字讓使用者猜。使用者輸入答案後電腦要回應太大還是太小, 重覆到使用者猜對為止。



重點

梯度下降法

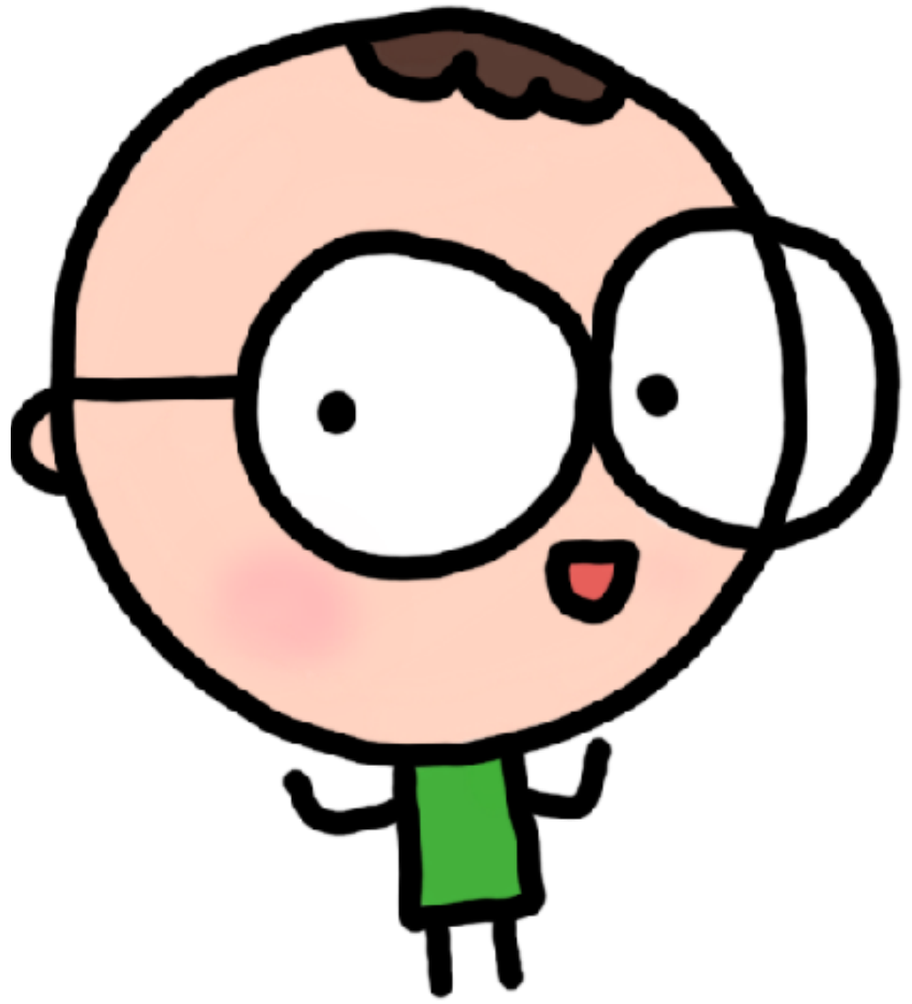
假設你做了一個神經網路, 對 w 這個參數的 loss function 如下:

$$L(w) = 0.12w^4 - 1.34w^3 + 4.86w^2 - 5.94w + 2.97$$

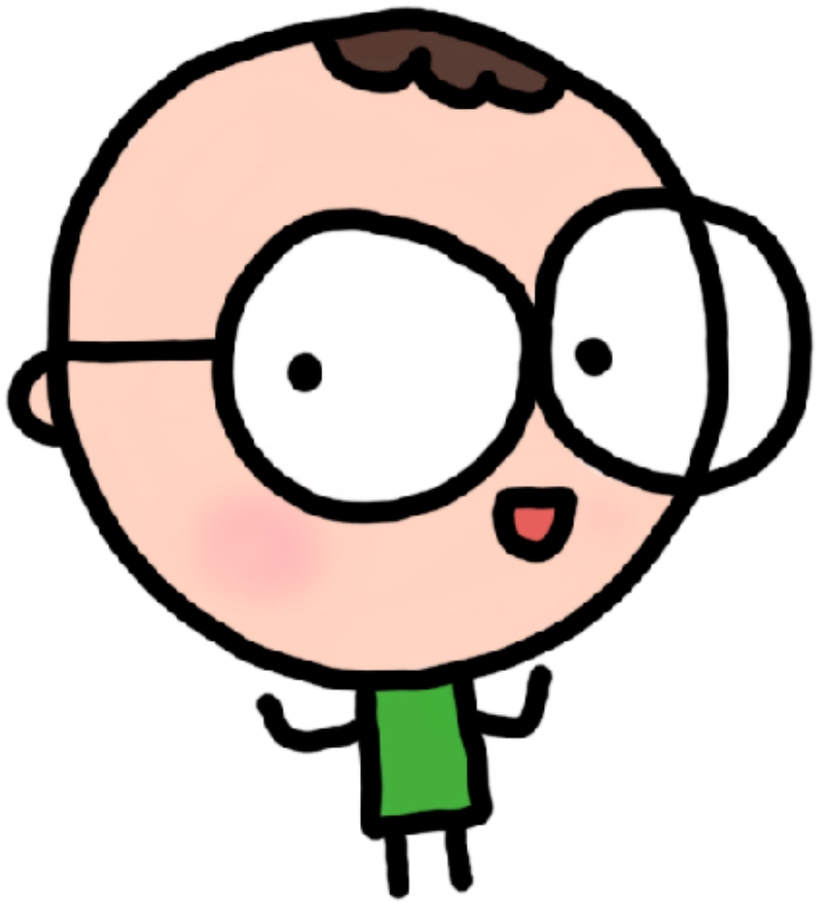
起始點 w 隨機從 0 到 5.5 選一個數, 用梯度下降法找出 (局部) 最小值發生的地方。

8

套件讀入



這裡介紹 Python 套件讀入的正規方式!



重點

import 法之 1

標準就是用 `import` 去讀入一個套件, 例如讀入 numpy 就是

```
import numpy
```

只是這樣以後要用到 numpy 相關指令都要像這樣...

```
y = numpy.sin(2*numpy.pi*t)
```

重點

import 法之 2

我們喜歡給個簡單的代號, 比方說:

```
import numpy as np
```

重點

import 法之 2

有趣的是很多套件有「標準」縮寫方式。

```
%matplotlib inline

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

這是我們推薦的資料分析「起手式」。

重點

import 法之 3

我們也可以單獨要某一個函數。

```
from numpy import sin
```

然後 Python 突然就會 sin 了!

重點

import 法之 4

也可以某個套件庫的函數全要! 但非常不推薦!

```
from numpy import *
```