Chapter 1：Python X 機器學習

賴秉樑 debugger
學院創辦人

課程網址　https://max543.com/debugger

max 極限翻轉學院
青少年與兒童程式教育

PERSONAL INFO

姓名 NAME

賴秉樑 debugger

學歷 EDUCATION

台科大資工、電子雙學位

中興資訊科學與工程碩士

經歷 EXPERIENCE

國立大學電子系、資工系講師

職業訓練、產投電腦講師

竹科半導體研發工程師

個人興趣 INTERST

玩數學、打電腦

我的座右銘 MOTTO

動手做、樂趣多

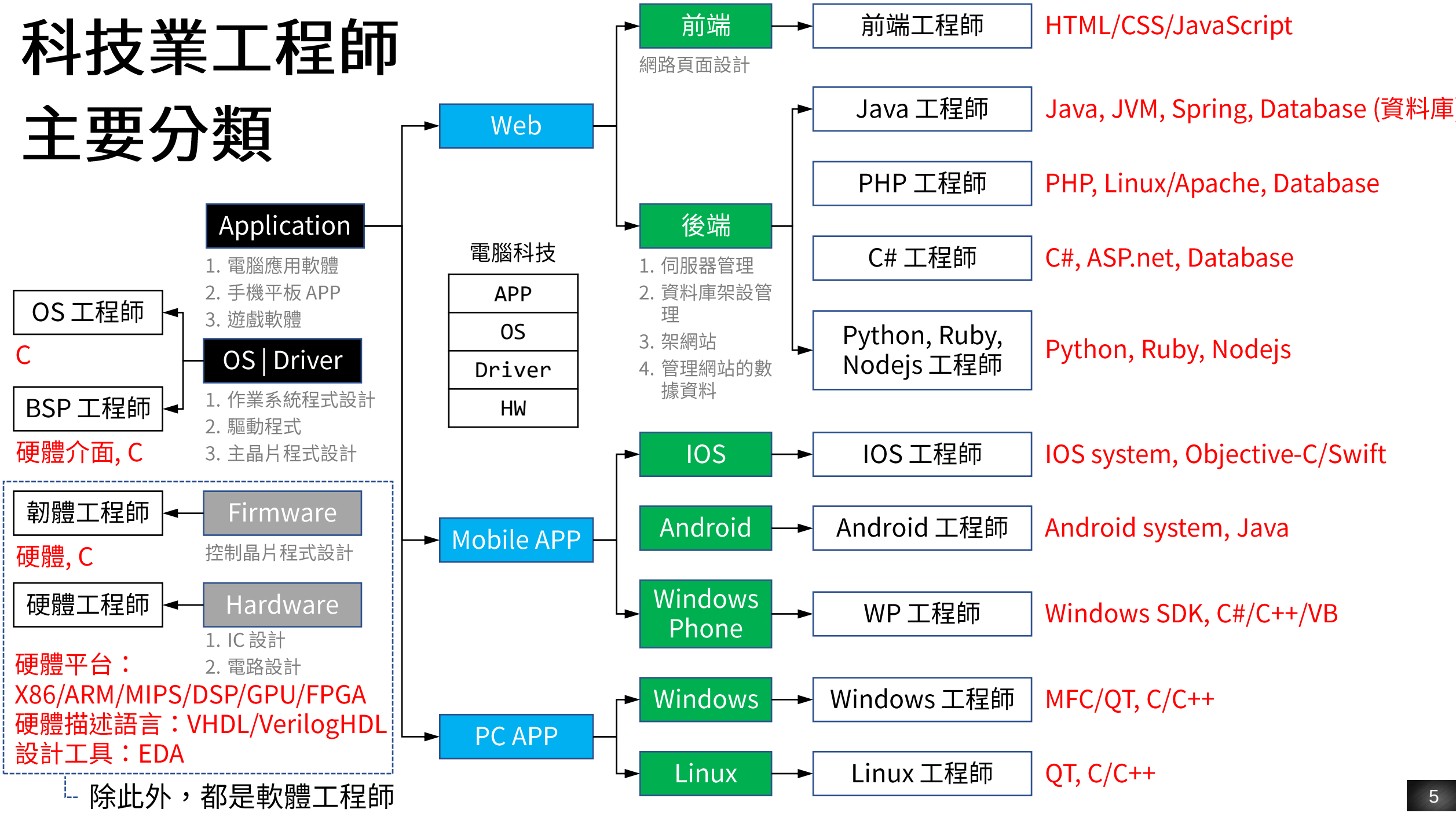
Outline

- 美好的程式世界
- 人工智慧 (Artificial Intelligence, AI)

Coder, Hacker, and Maker

- 程式設計師（coder, 或 programmer）
  - ✓ 主要透過編輯程式，簡稱編程（coding），它可以指在程式設計某個專業領域的專業人士，或是從事軟體撰寫，程式開發、維護的專業人員。
- 駭客（hacker）
  - ✓ 除了精通程式設計、作業系統的人可以被視作駭客，對硬體裝置做創新的工程師通常也被認為是駭客，精通網路入侵的人也被看作是駭客。
- 創客（maker）
  - ✓ 又稱自造者。是一群酷愛科技、熱衷實踐的人群，他們以分享技術、激發的創造力與交流思想為樂。

科技業工程師主要分類


Application
1. 電腦應用軟體
2. 手機平板 APP
3. 遊戲軟體
OS 工程師
C
OS | Driver
1. 作業系統程式設計
2. 驅動程式
3. 主晶片程式設計
BSP 工程師
硬體介面, C
韌體工程師
Firmware
控制晶片程式設計
硬體, C
硬體工程師
Hardware
1. IC 設計
2. 電路設計
硬體平台：
X86/ARM/MIPS/DSP/GPU/FPGA
硬體描述語言：VHDL/VerilogHDL
設計工具：EDA
除此外，都是軟體工程師
電腦科技
APP
OS
Driver
HW
Web
前端
網路頁面設計
前端工程師
HTML/CSS/JavaScript
後端
1. 伺服器管理
2. 資料庫架設管理
3. 架網站
4. 管理網站的數據資料
Java 工程師
Java, JVM, Spring, Database (資料庫)
PHP 工程師
PHP, Linux/Apache, Database
C# 工程師
C#, ASP.net, Database
Python, Ruby, Nodejs 工程師
Python, Ruby, Nodejs
Mobile APP
IOS
IOS 工程師
IOS system, Objective-C/Swift
Android
Android 工程師
Android system, Java
Windows Phone
WP 工程師
Windows SDK, C#/C++/VB
PC APP
Windows
Windows 工程師
MFC/QT, C/C++
Linux
Linux 工程師
QT, C/C++

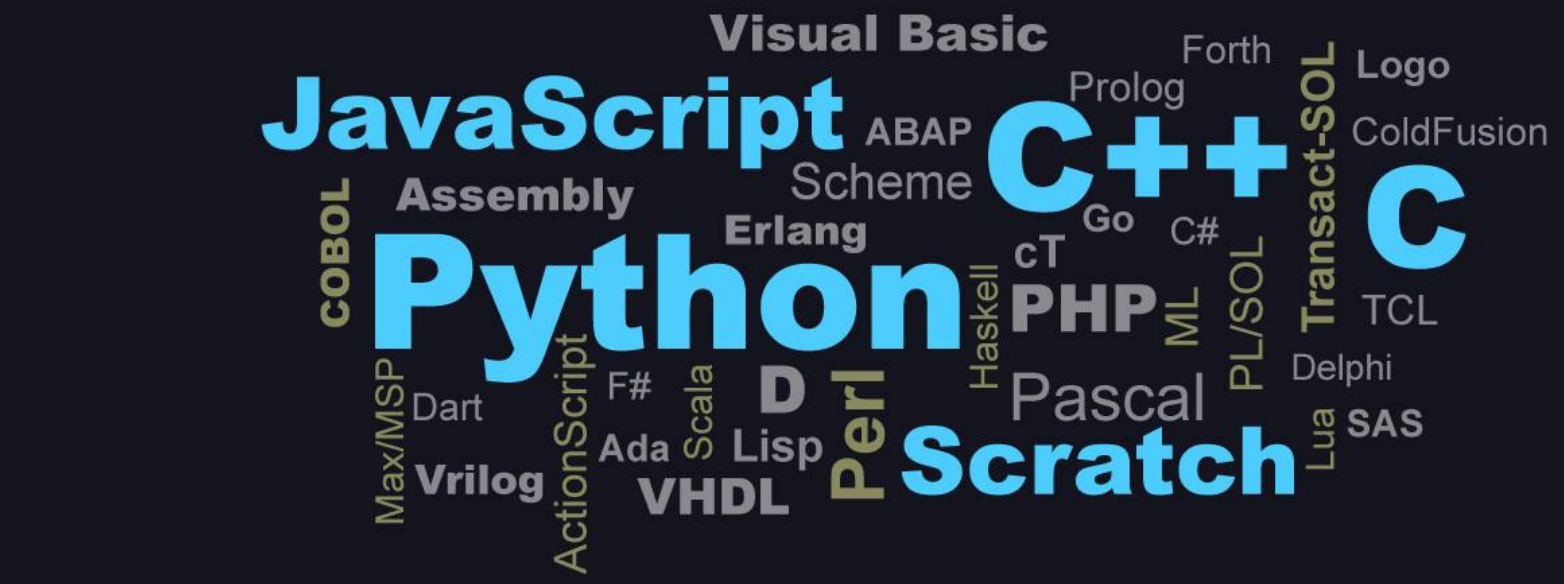
運算思維為何很重要？(1/2)

- 學會了運算思維，讓我們也能擁有電腦科學家面對問題時，所持有的科學方法。各種領域都需要運算思維，例如：
- 科學與工程領域
  - ✓ 利用運算模擬建築結構，以確認安全性。
  - ✓ 利用運算預測氣象，以增加準確性。
- 金融領域
  - ✓ 利用運算研究經濟大數據。
  - ✓ 利用運算完成自動交易。

運算思維為何很重要？(2/2)

- 人文與社會領域
  - ✓ 利用運算分析，並優化廣告投放策略。
  - ✓ 利用運算分析人口老化趨勢，與醫療資源分布。
- 藝術領域
  - ✓ 利用運算建構三維動畫。
  - ✓ 利用運算創作數位音樂。
- 工業設計
  - ✓ 利用運算實現工業 4.0。
  - ✓ 利用運算實現更多的加值應用，例如：自動化與智慧化。

因為有這些程式
生活更美、更好



科技業的下一個神話

電子商務 x 社群網路 x 串流媒體

物聯網 x 人工智慧 x 區塊鏈

Python Python Python

Hello, World! 最像英文的 **Python**

Python

```python
#  印出 Hello World! 字串物件
print("Hello World!")
```

C

```c
/*  印出 Hello World! 字串物件*/
 include <stdio.h>

int main()
{
    printf("Hello, World!\n");
    return 0;
}
```

C++

```cpp
//印出 Hello World! 字串物件
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello World" << endl;
   return 0;
```

Java

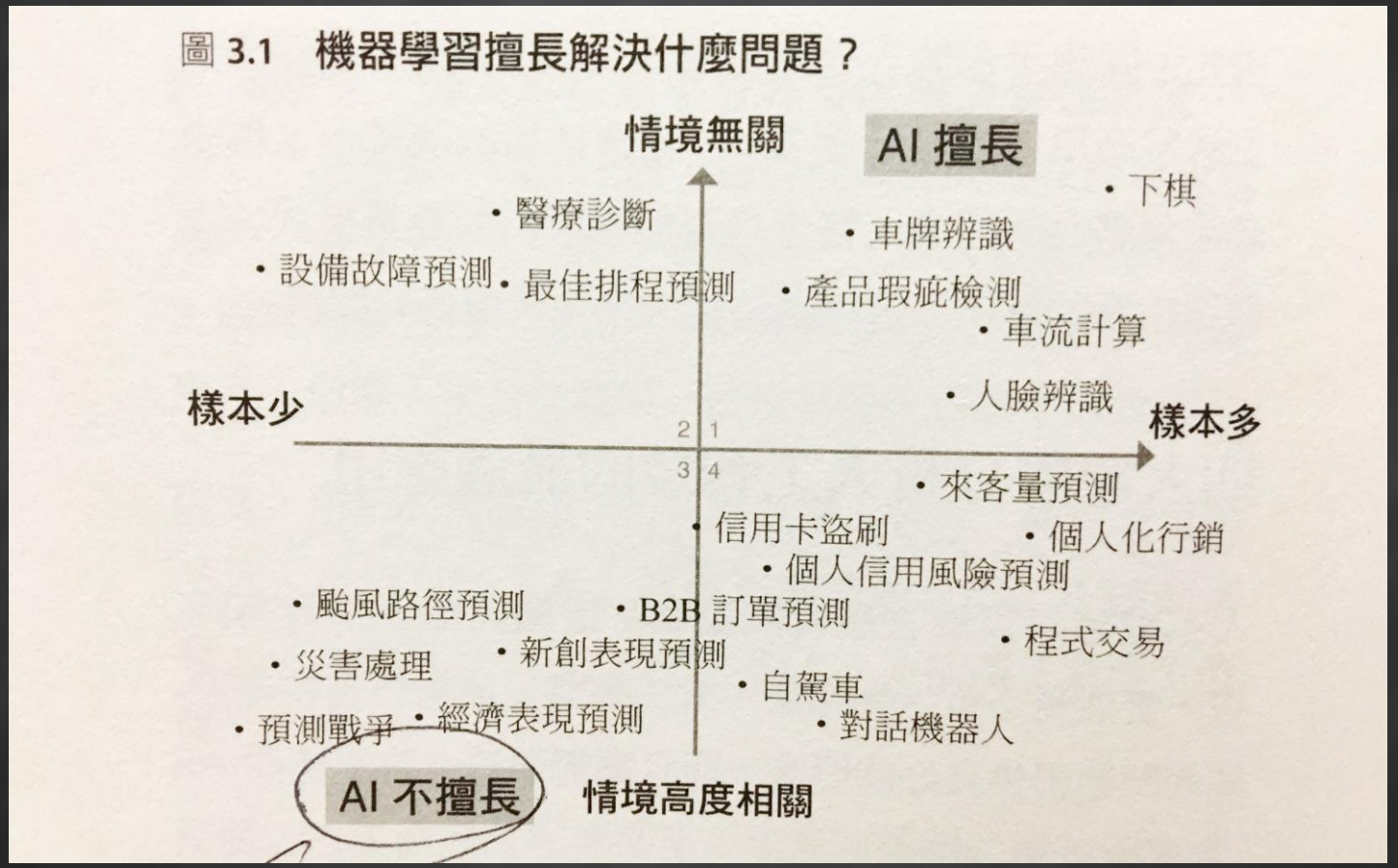
```java
//印出 Hello World! 字串物件
public class HelloWorld{

     public static void main(String []args){
        System.out.println("Hello World");
     }
}
```

Outline

- 美好的程式世界
- 人工智慧 (Artificial Intelligence, AI)

現在的人工智慧

- 現在的 AI 只能針對符合特定形式與條件的問題，提供良好的解答。
- 目前以機器學習為基礎的人工智慧，不可能擁有人類的思考及情緒。



人工智慧發展簡史

第一波
1950-1960

失敗

符號邏輯

把人的思考邏輯放進電腦

由領域專家寫下決策邏輯。

人類還沒辦法清楚理解自己的思考過程，如何告訴電腦？

第二波
1980-1990

失敗

專家系統

把人的所有知識放進電腦

由領域專家寫下經驗規則。

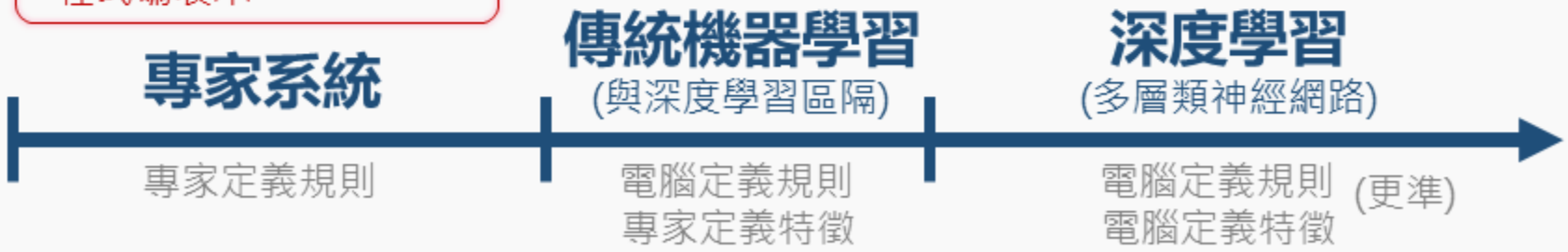
太多難題人類無法解答、無法寫成規則、無法以程式碼表示。

第三波
2010-Present

機器學習

把人的所有看見放進電腦

由領域專家提供歷史資料，讓電腦自己歸納規則。



資料來源：《人工智慧在台灣》，劉奕酉整理

機器學習技術很成熟

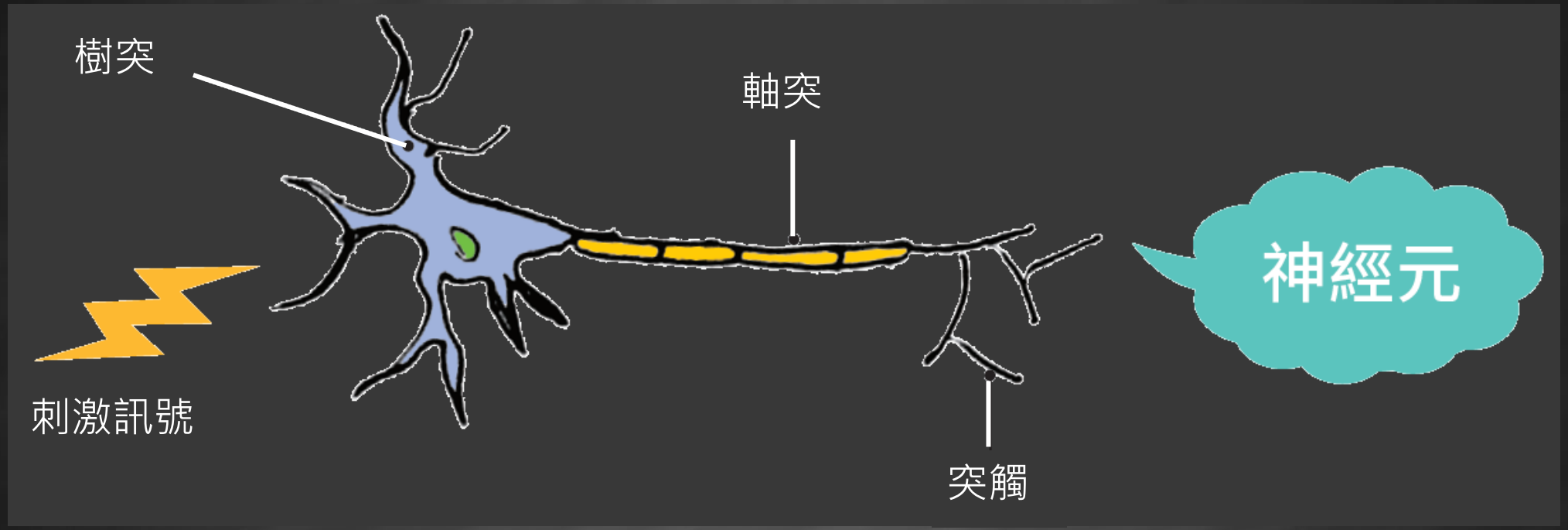
建神經模型，資料就丟進去，電腦自己找解答

8 小時的課程

- Chapter 1：Python X 機器學習
- Chapter 2：影像處理 X 深度學習
- Chapter 3：影像處理 X App Inventor (雲端自動的深度學習)

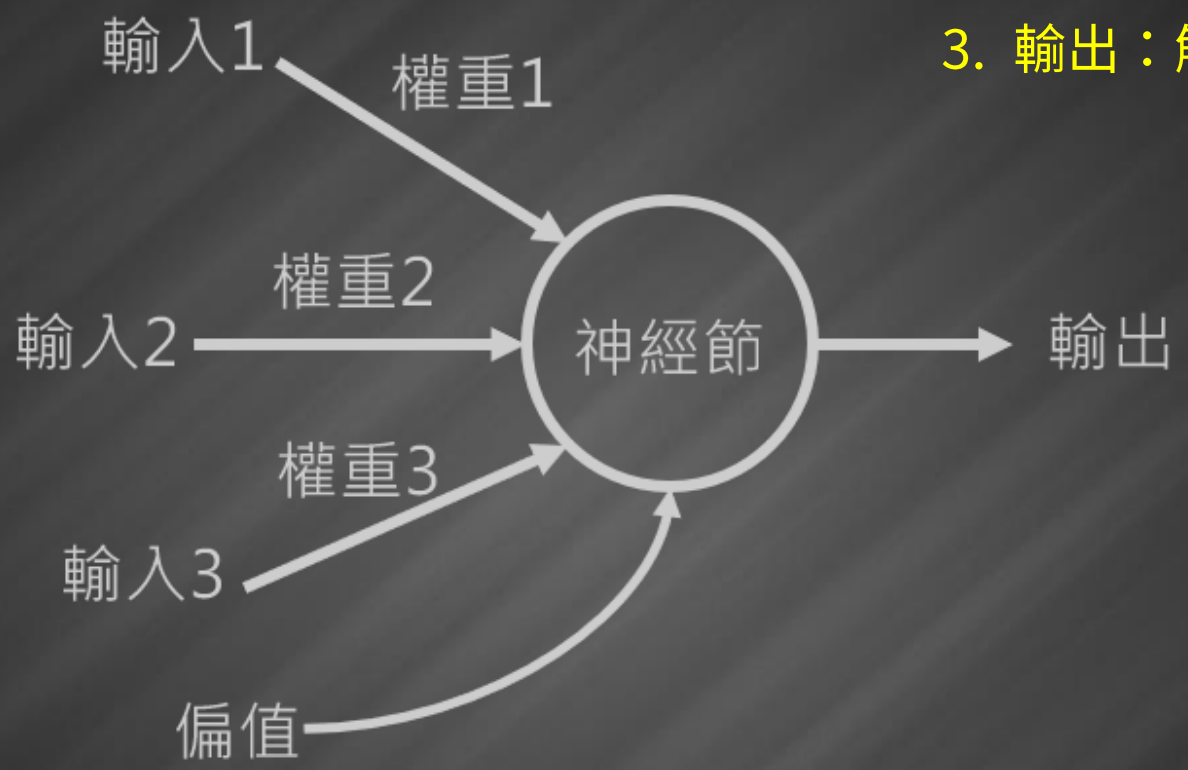
機器學習的簡單例子

初探 AI-神經網路（主流的機器學習技術）



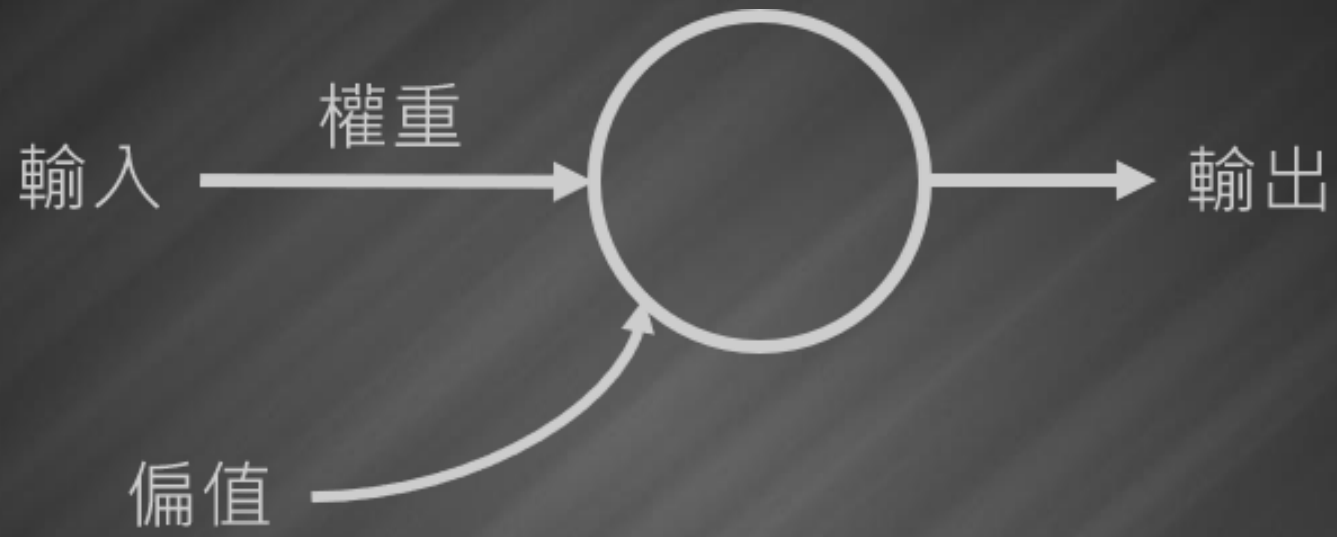
人工神經元

1. 輸入：指問題
2. 權重和偏值：自我學習的參數
3. 輸出：解答



輸出 = 輸入 1× 權重 1 + 輸入 2× 權重 2 + 輸入 3× 權重 3 + 偏值

神經元如何學習迴歸問題



輸出 = 輸入 × 權重 + 偏值

迴歸問題

輸出 = 輸入 × 權重 + 偏值

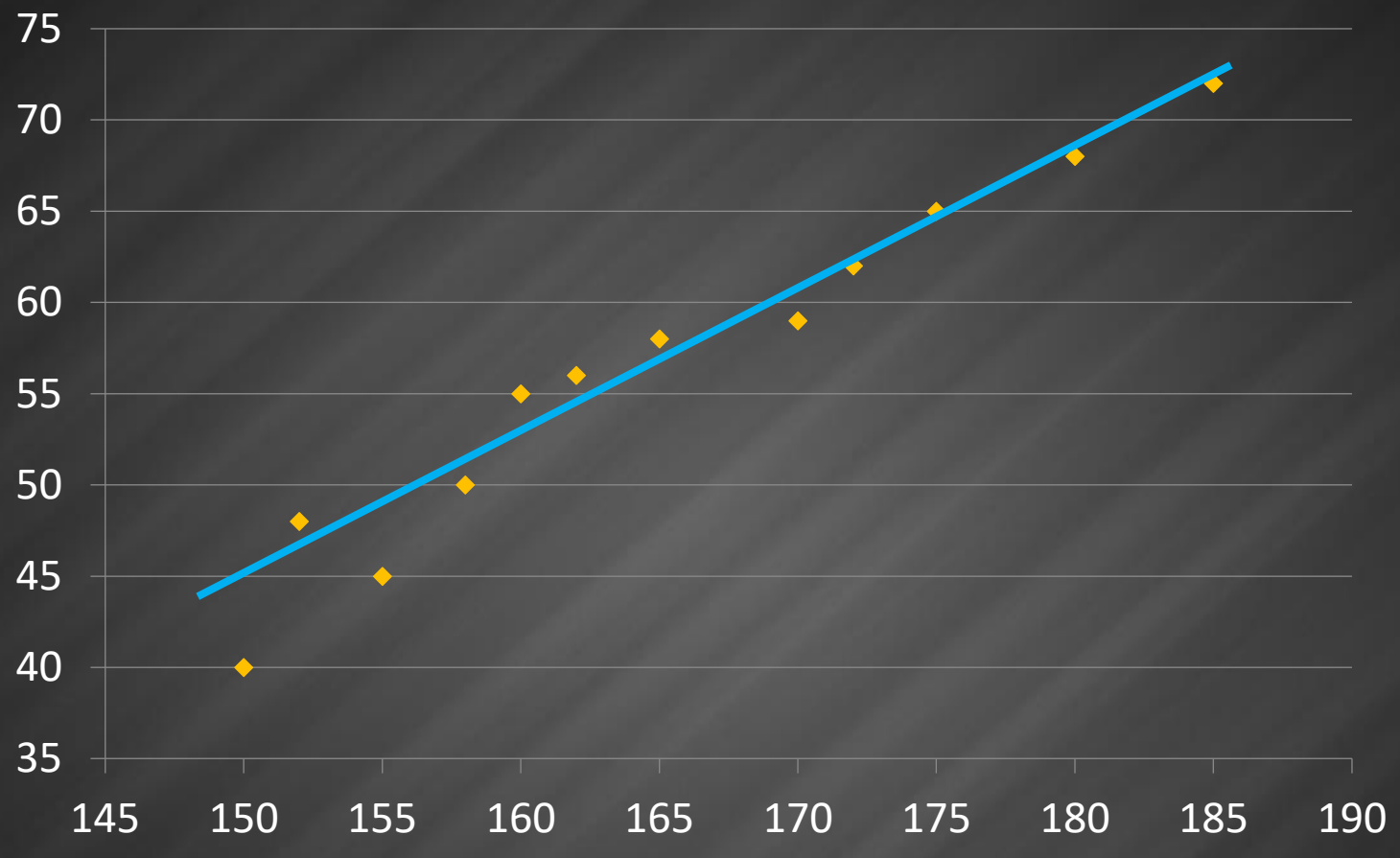
某一班學生的身高和體重是否相關?

能否用身高來推測某位學生的體重?

身高	體重
150	40
152	48
155	45
158	50
160	55
162	56
165	58
170	59
172	62
175	65
180	68
185	72

迴歸線（函數）

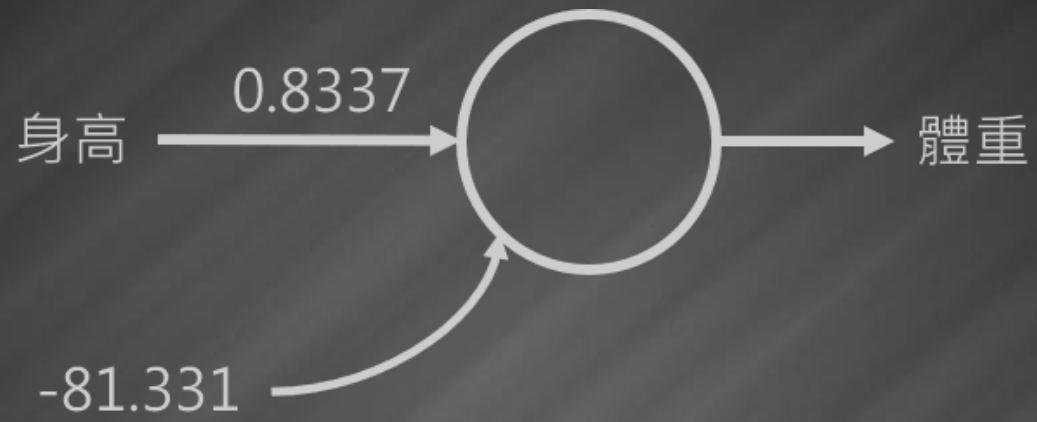
輸出 = 輸入 × 權重 + 偏值



y = 0.8337x - 81.331

迴歸線（函數）

輸出 = 輸入 × 權重 + 偏值



體重 = 身高×0.8337 - 81.331

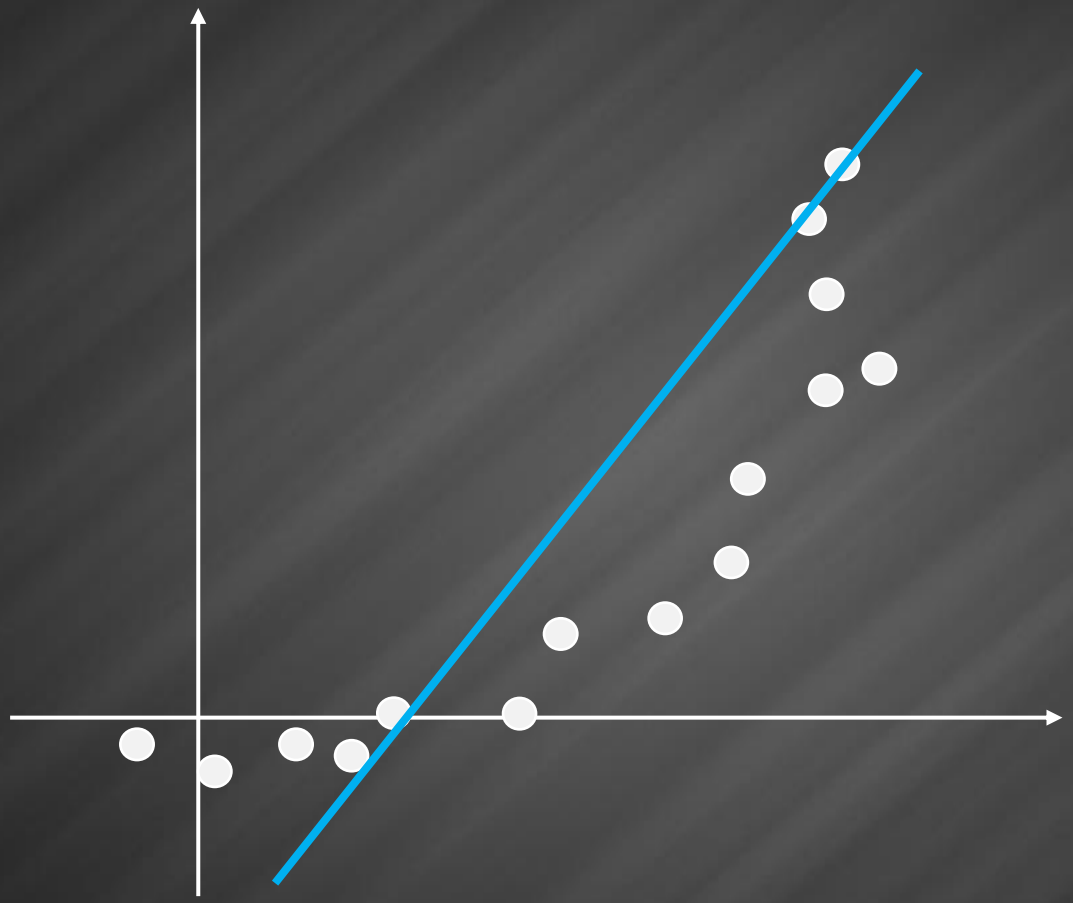
建立兩組資料間的對應函數

非線性問題

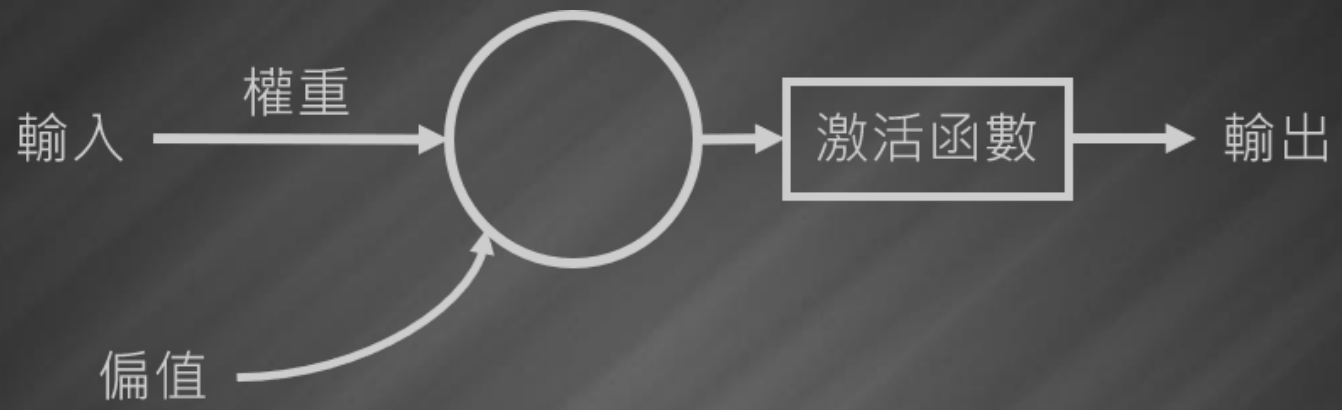
輸出 = 輸入 × 權重 + 偏值

線性函數（又稱一次函數）

$$y = f(x) = kx + b$$

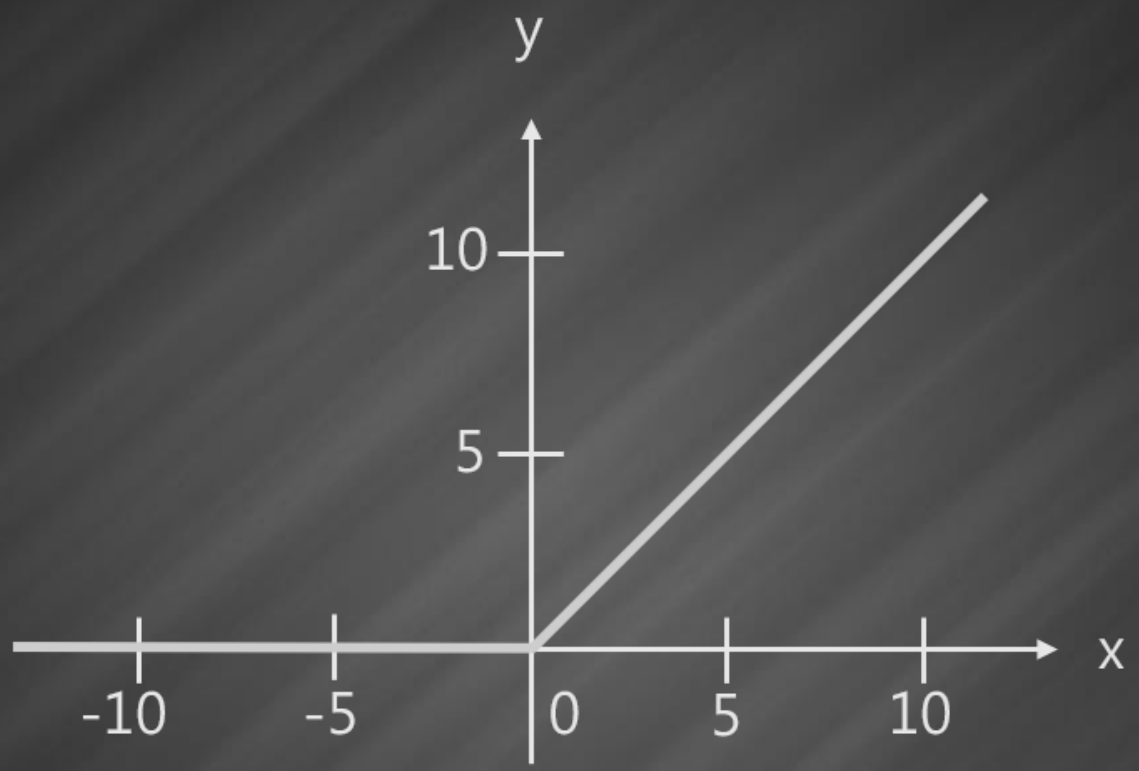


激活函數（activation function）



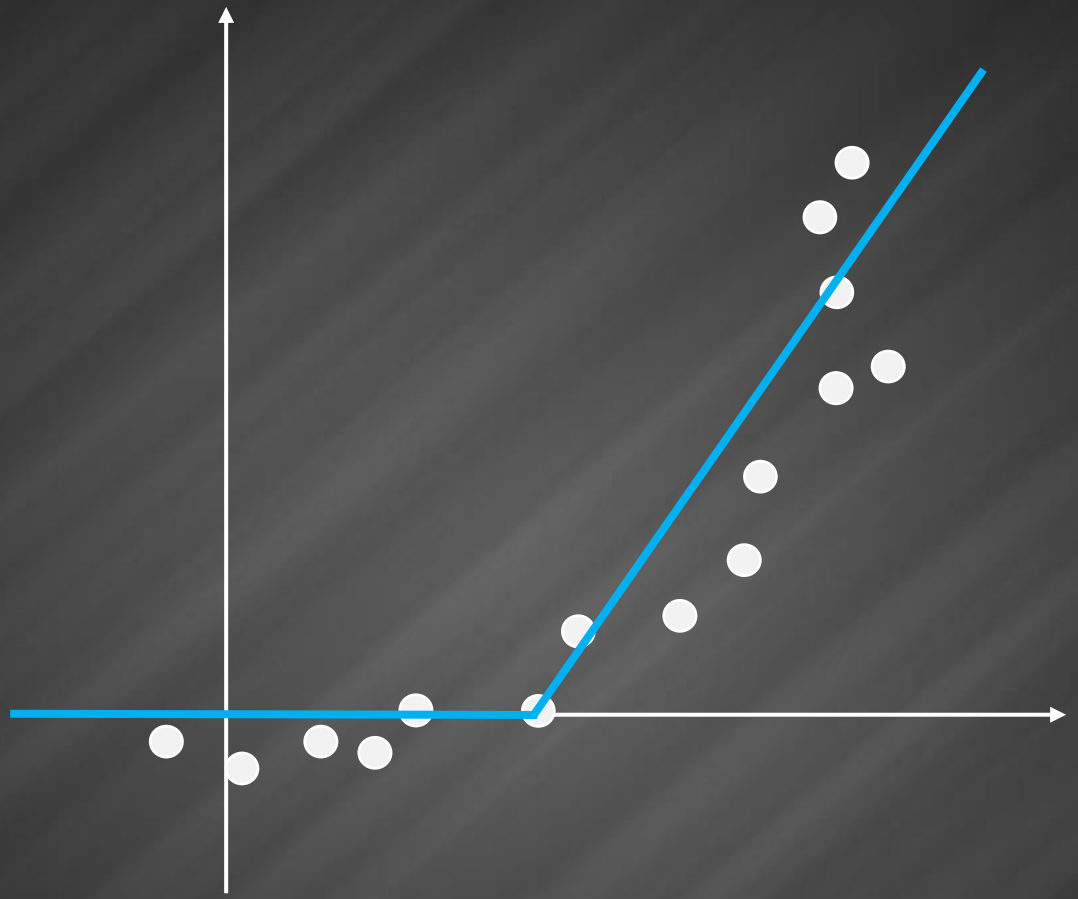
輸出 = 激活函數(輸入× 權重+ 偏值)

ReLU 函數（線性整流函數，增加非線性度）



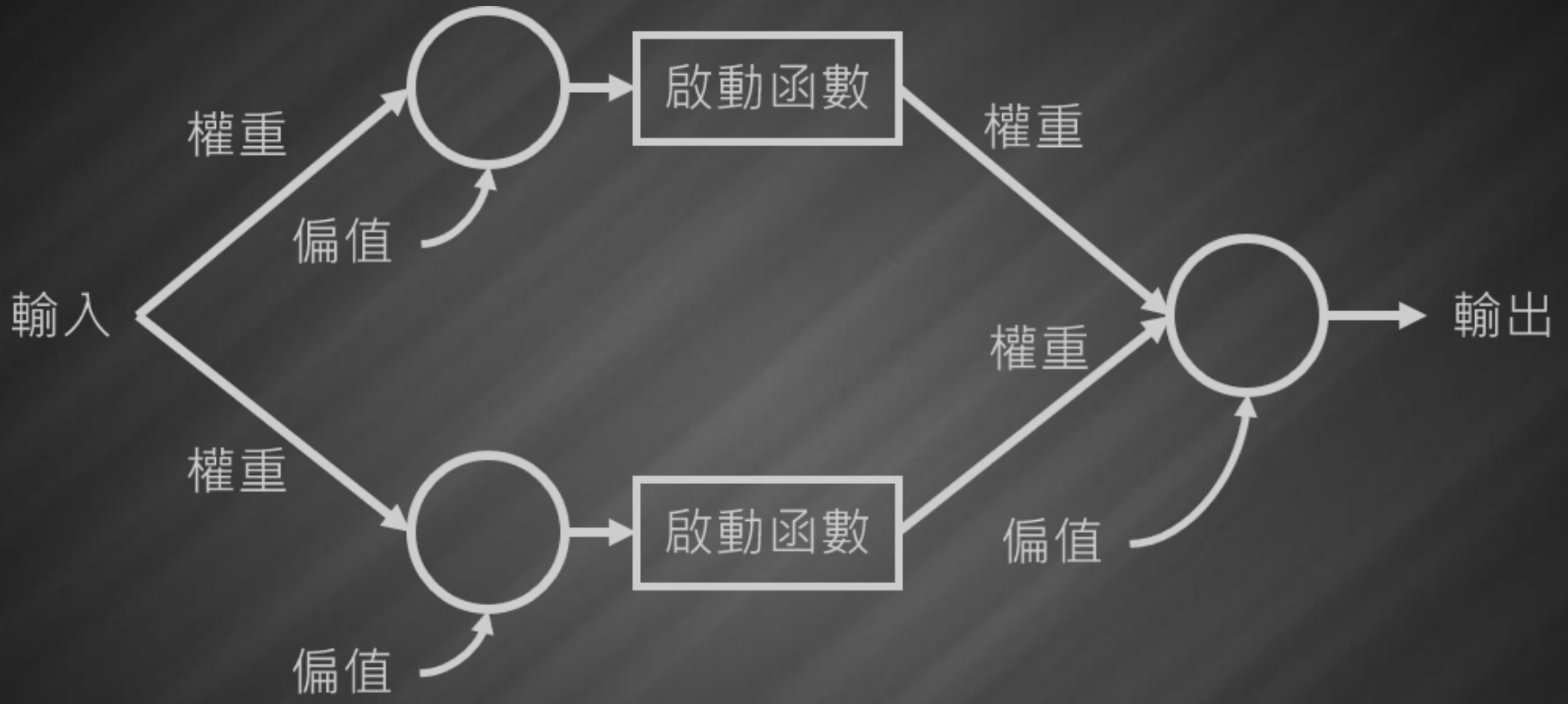
小於 0 就等於 0

ReLU 函數（線性整流函數，增加非線性度）

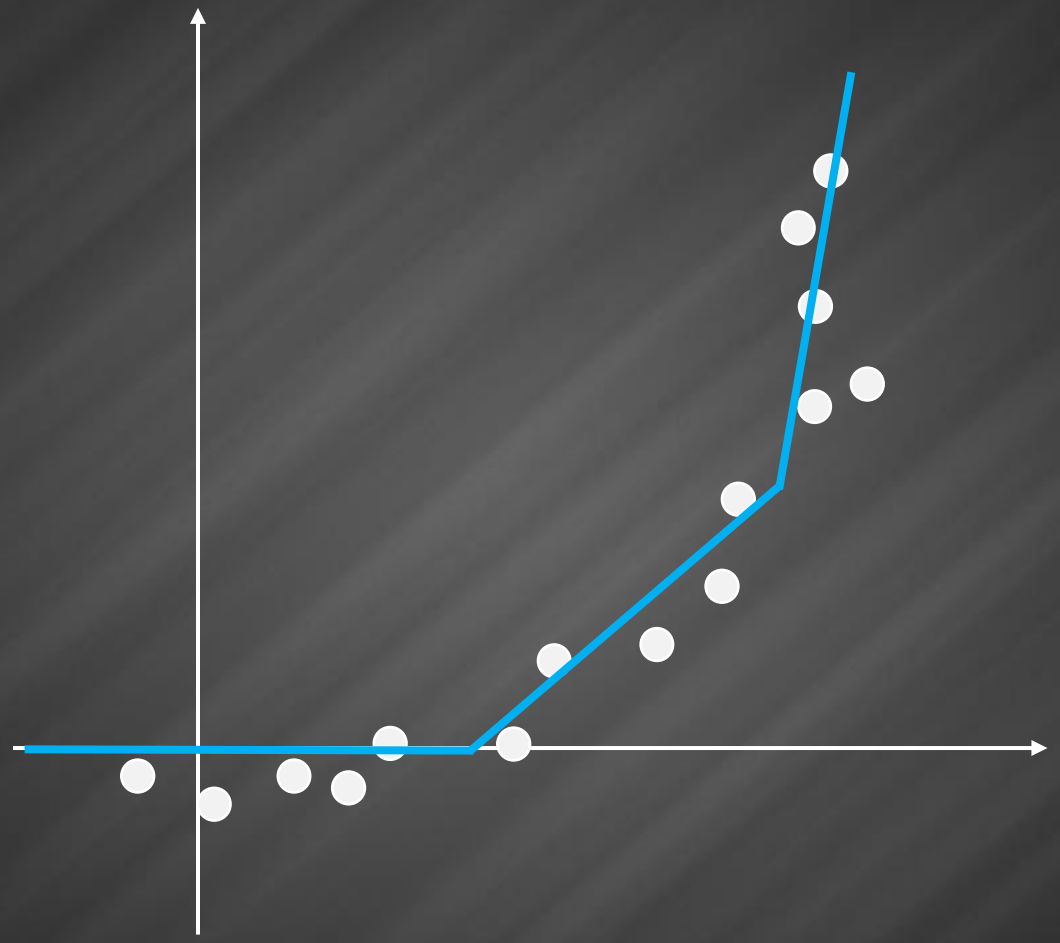


小於 0 就等於 0

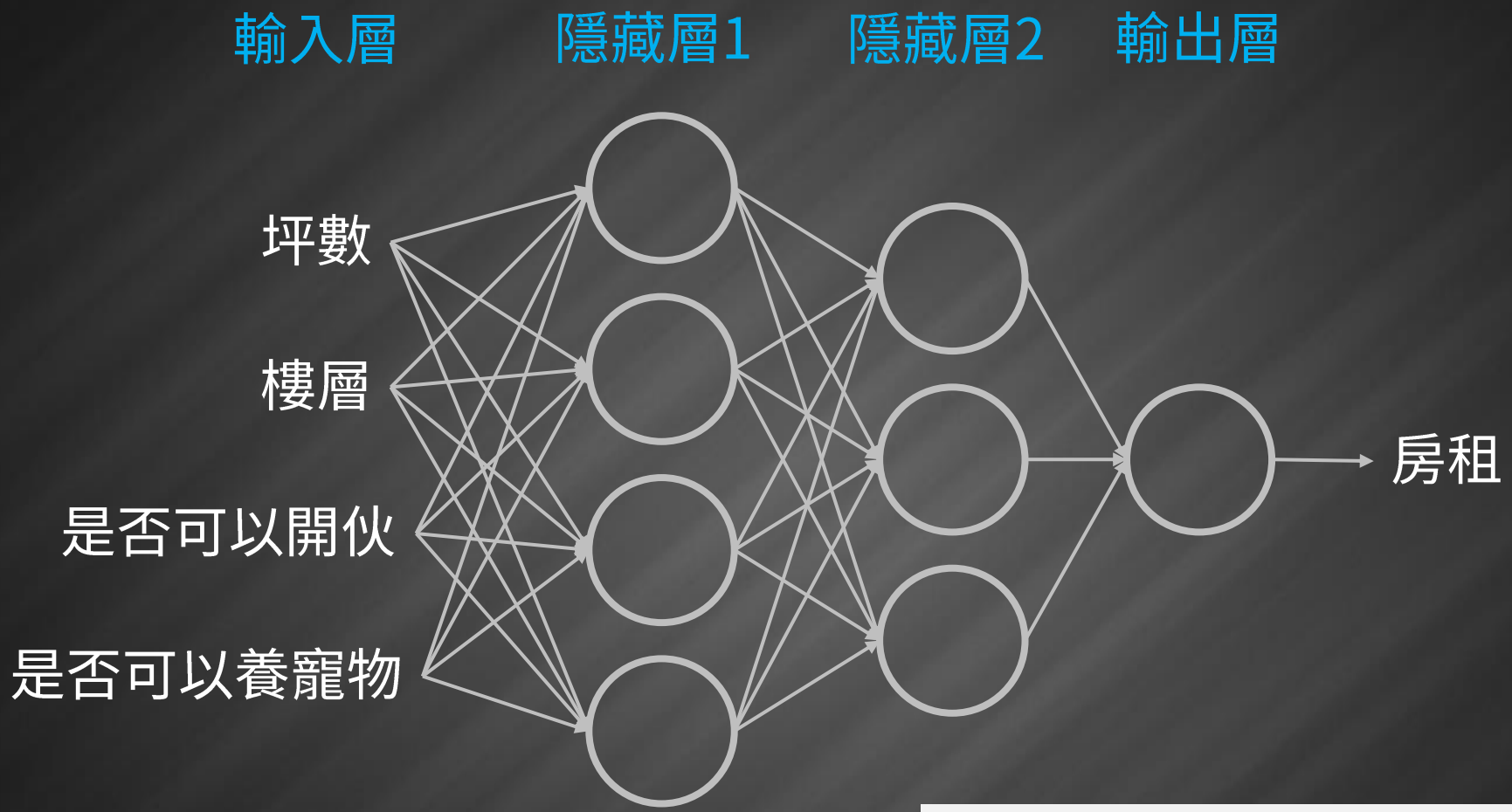
更貼近資料：多神經元串聯



更貼近資料：多神經元串聯

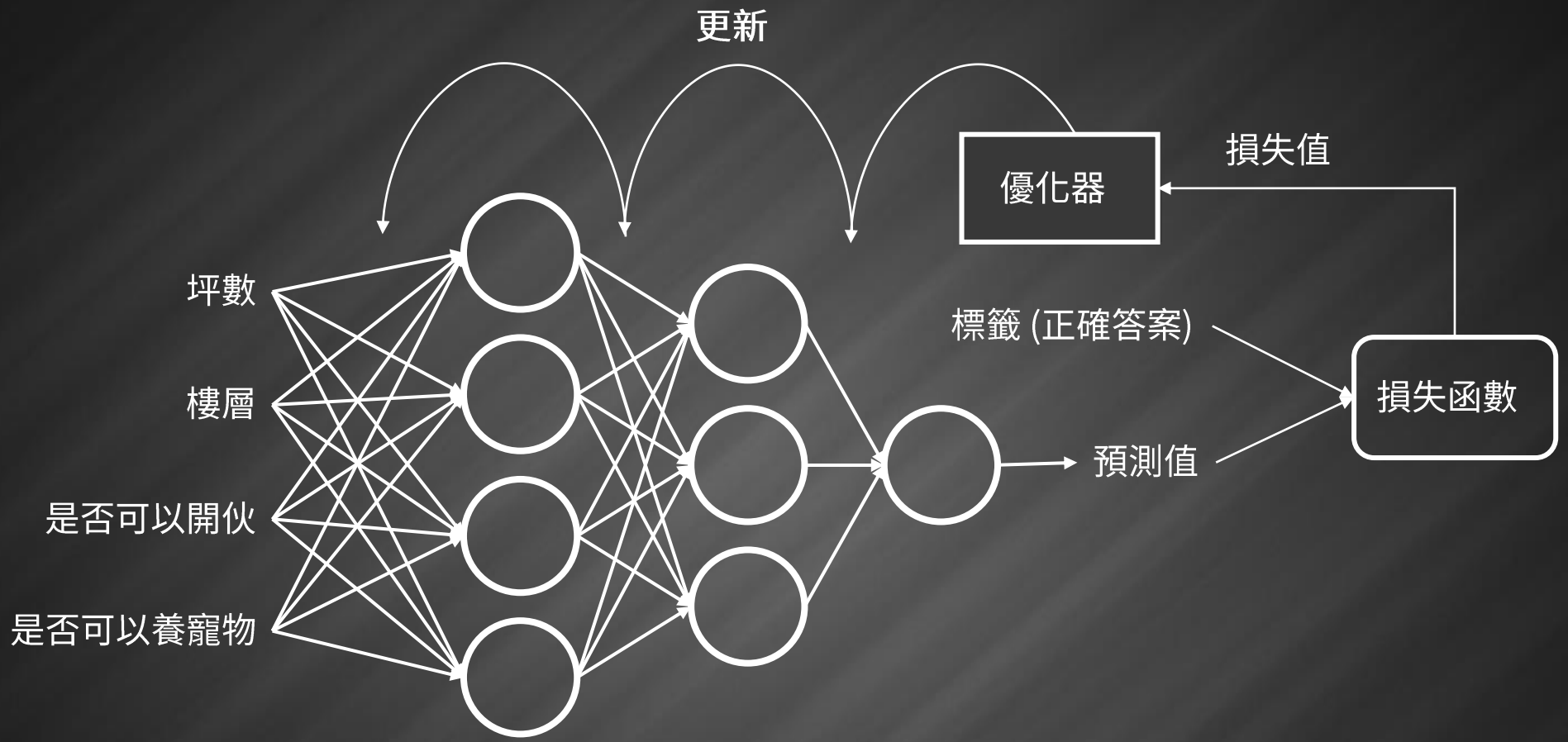


神經網路（又稱為模型）



忽略偏值與激活函數，增加閱讀性

神經網路的學習過程



反向傳播法 (Backpropagation, BP)

損失函數

均方誤差 (MSE)，是將每筆標籤減掉預測值（即誤差值）取平方，再取平均值。

標籤：

$$y_1 \text{、} y_2 \text{、} y_3 \text{、} y_4 \text{、} y_5 \cdots y_n$$

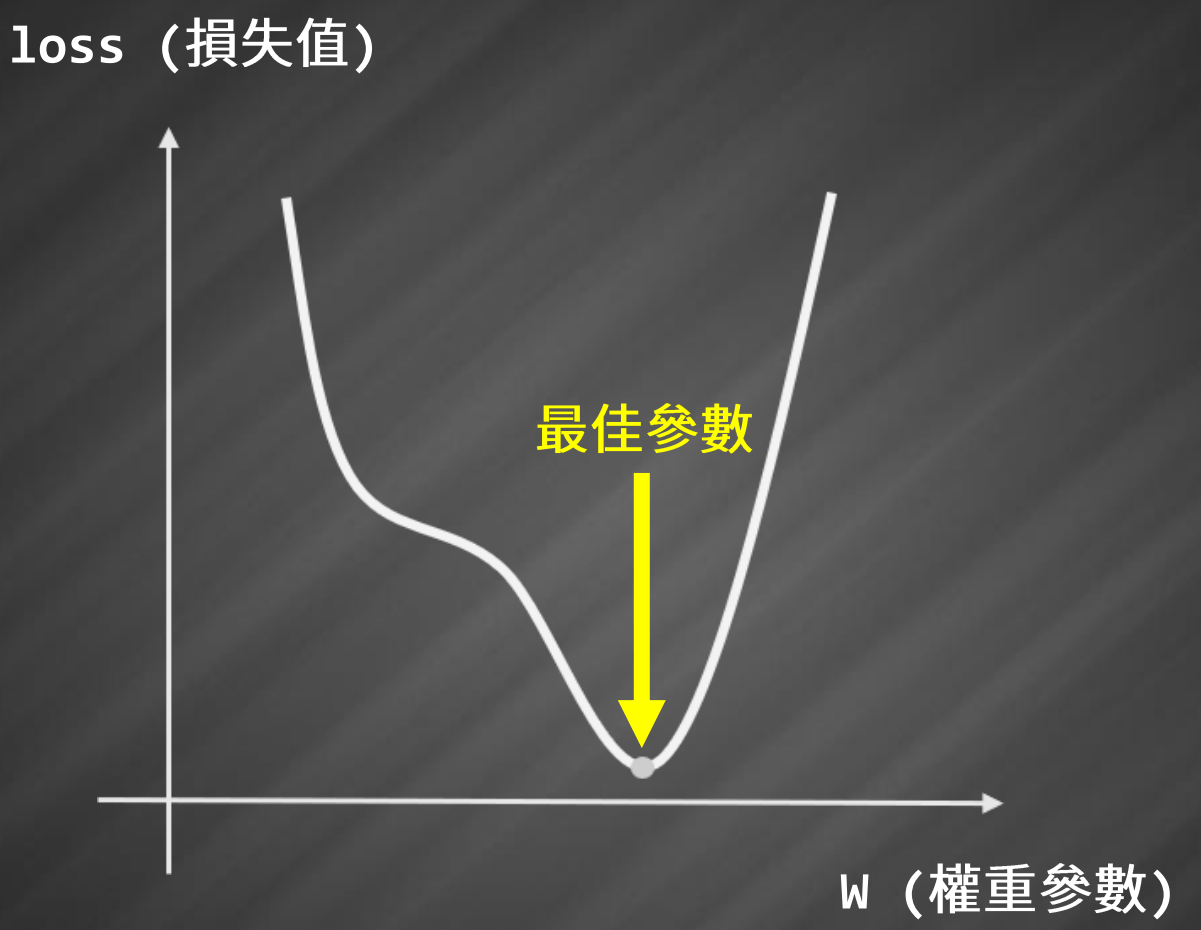
預測值：

$$\widehat{y_1} \text{、} \widehat{y_2} \text{、} \widehat{y_3} \text{、} \widehat{y_4} \text{、} \widehat{y_5} \cdots \widehat{y_n}$$

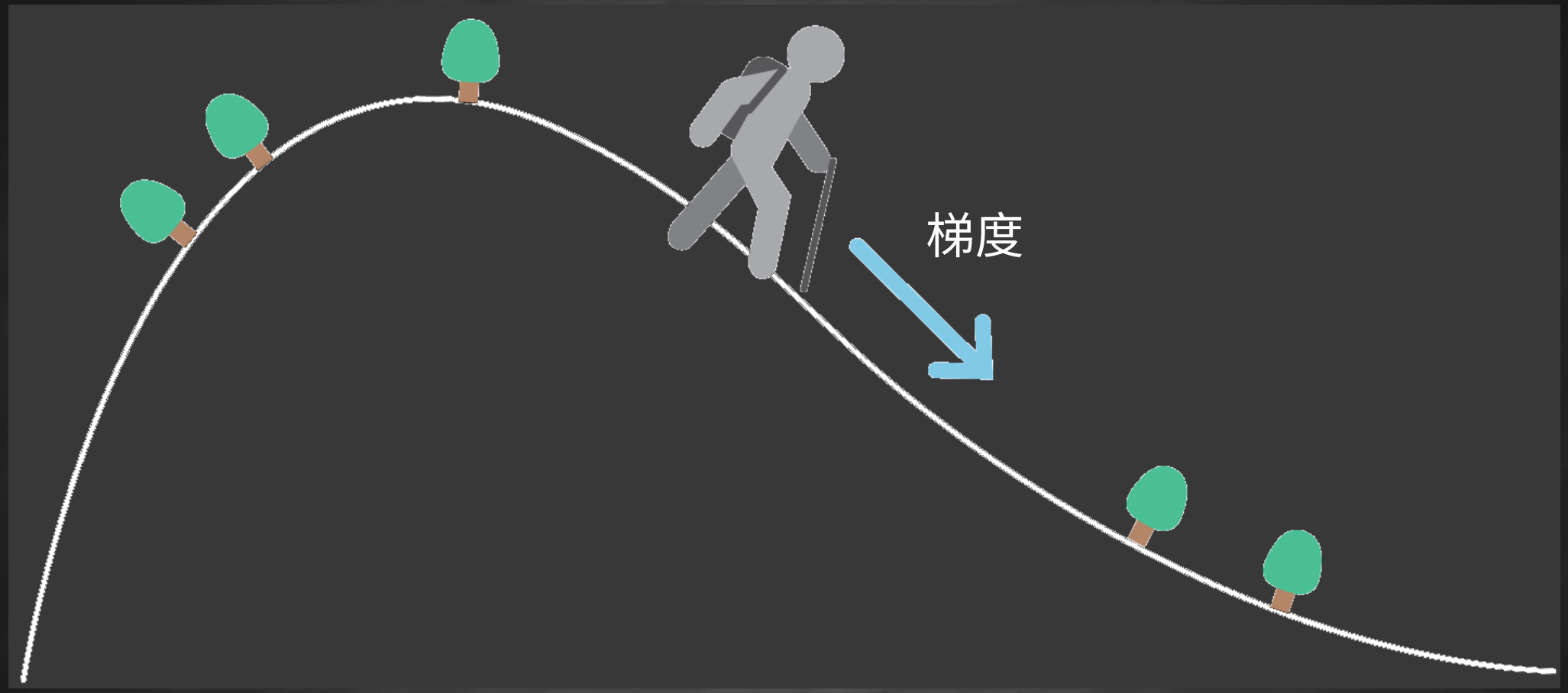
MSE：

$$\frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{n}$$

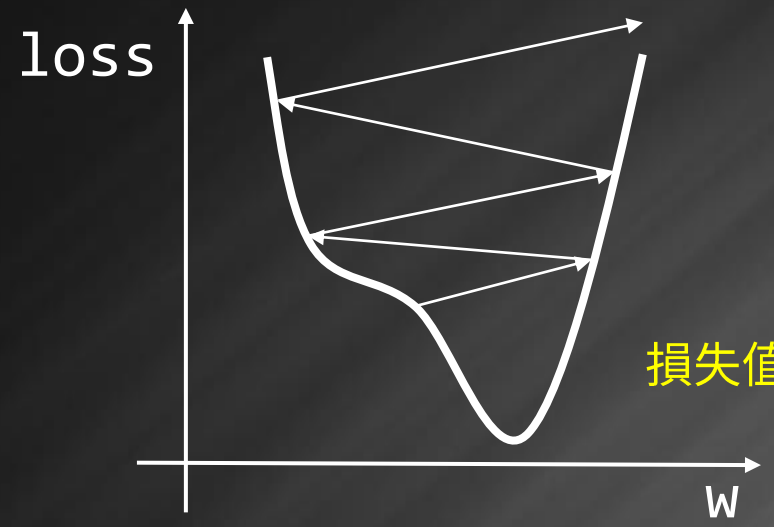
優化器：使用梯度下降（Gradient descent）



優化器：使用梯度下降（Gradient descent）

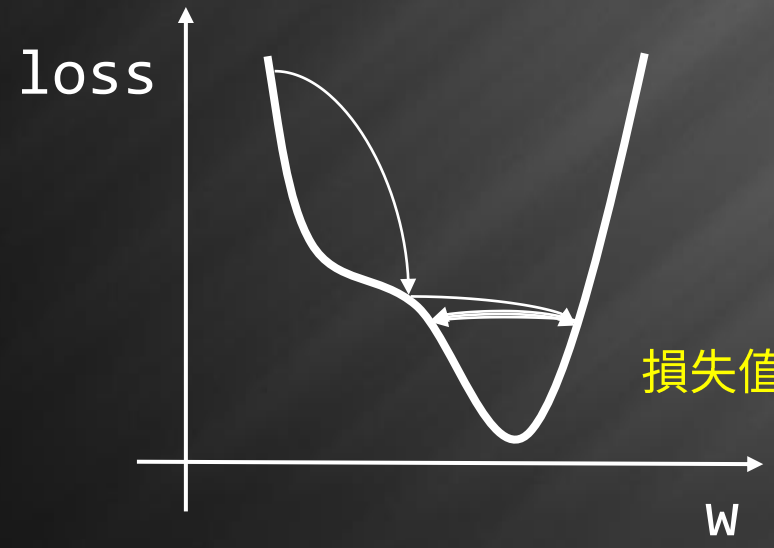


學習率：介於 0 ~ 1 (調整步伐大小)



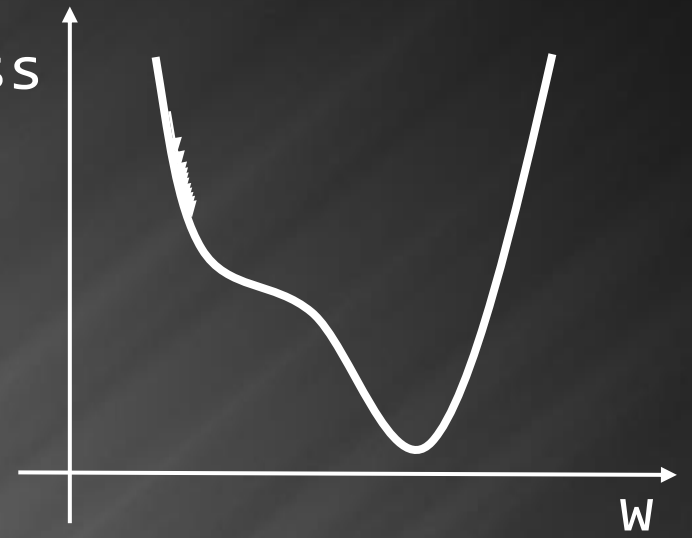
損失值爆炸

太大

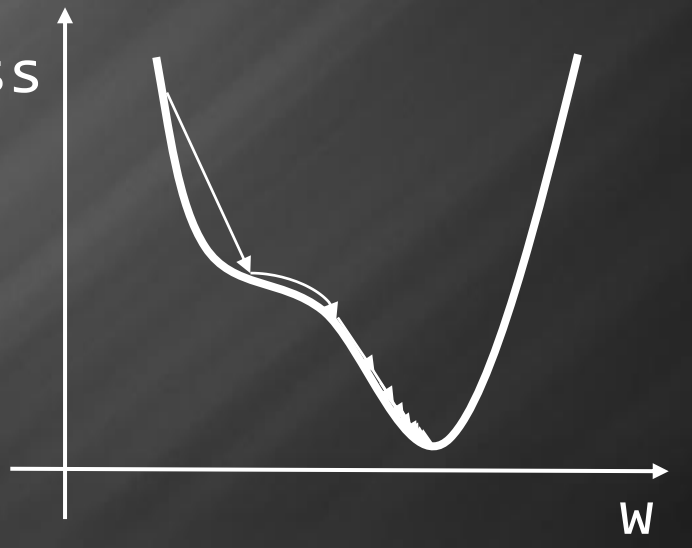


損失值震盪

太小

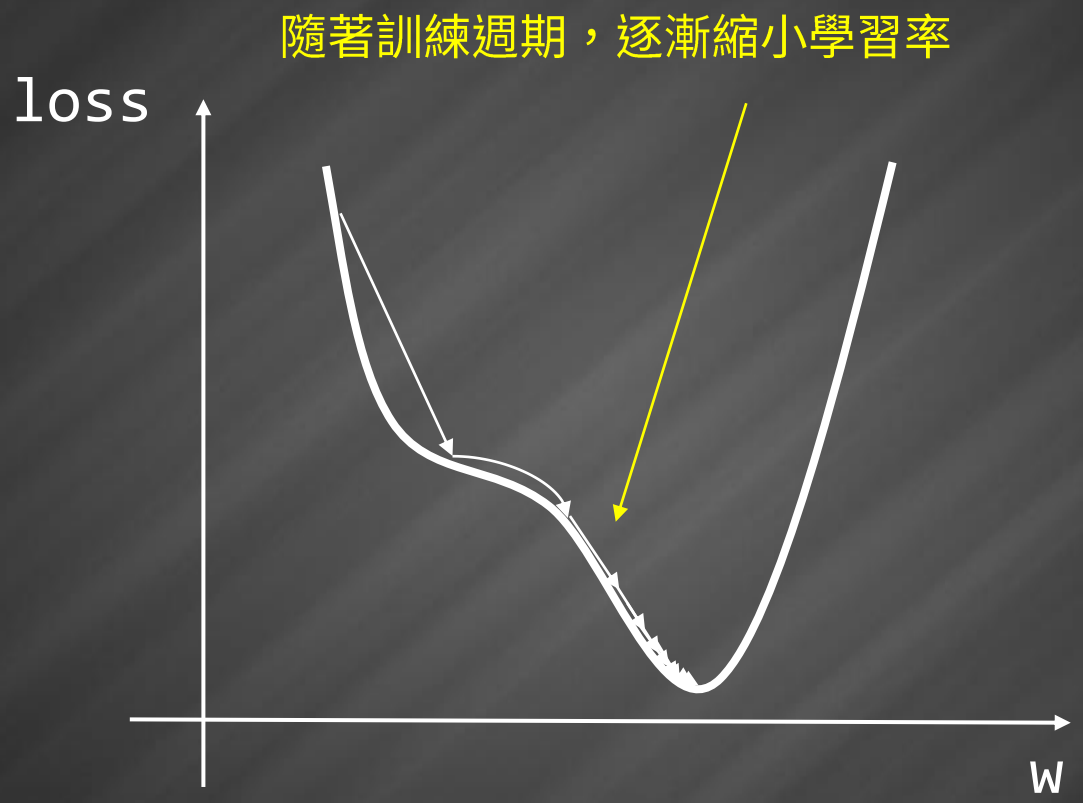


剛好



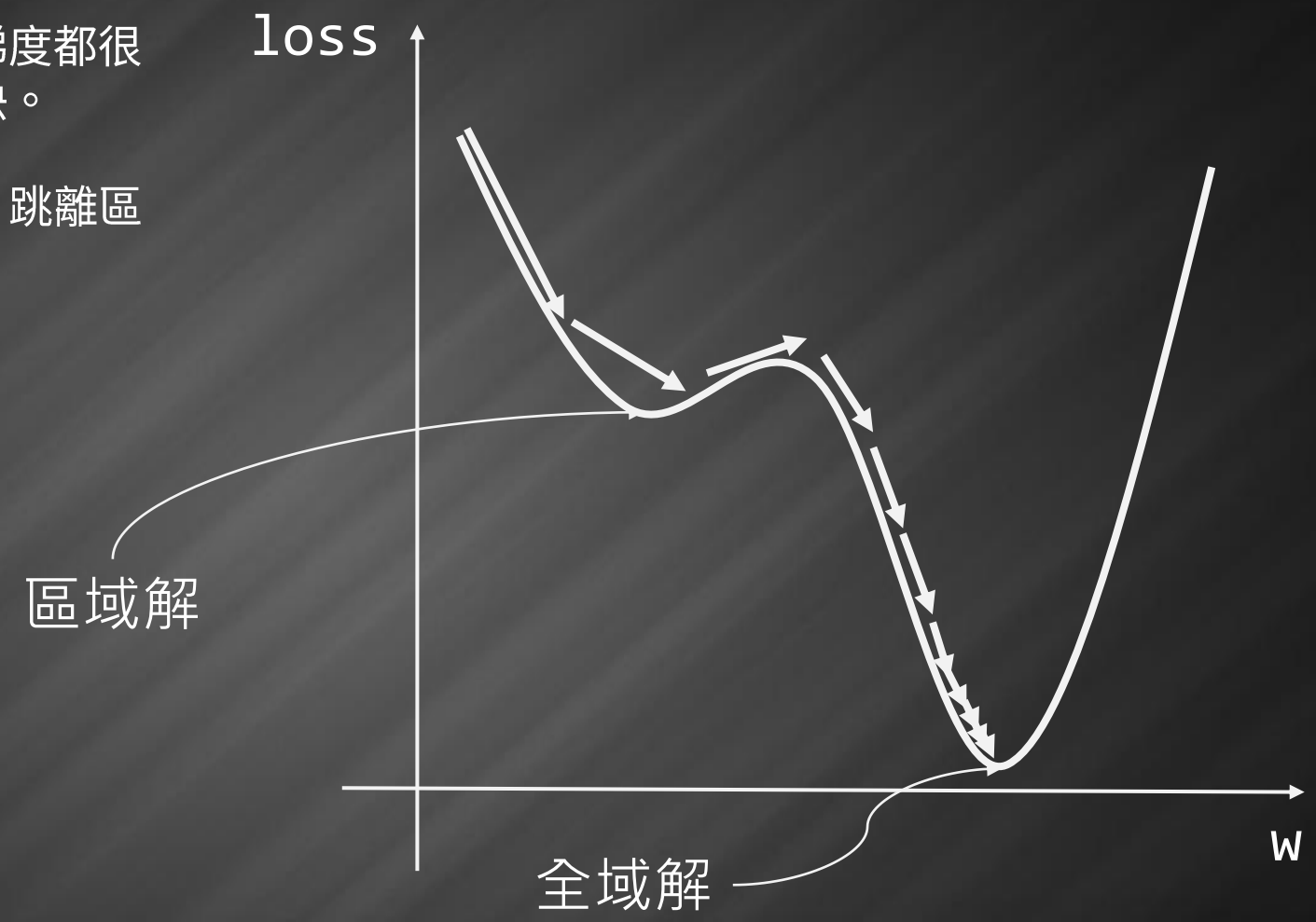
自適應（Adaptive ）

- 自適應：自動調整學習率



動量（Momentum）

- 動量：解決兩個問題
  1. 學習速度太慢：如果連續幾次梯度都很大，則動量可以讓移動速度加快。
  2. 停留在區域最低點：加上動量，跳離區域解。



學習 AI 時的重要工具 - Python

Python 的最基礎

- 物件
- 物件的資料型別
- 整數與浮點數物件
- 變數
- 內建函式
- 匯入模組

物件

- 英文一般寫句子時，會以名詞 + 動詞。Python 是以物件.方法來描述。

文章寫作	寫 Python 程式	
車子	car	car 物件
車子向前進	car.start()	car 物件的 start 方法

- 方法後面會加上括號()，有些方法需要加入額外的參數。
  - ✓ 例如：car.go(100)，車子加速到 100。
- 若方法有多個參數，以逗點分隔。
  - ✓ 例如：car.left(50, 30)，以 50 的速度，向左轉 30 度。

練習：字串物件

ex1-1

- 在互動模式中，輸入下列敘述：(>>> 指的是在互動模式中，執行單行敘述)

```
>>> "abc".upper()          ——— 使用字串物件 "abc" 的 upper() 方法，將字串轉成大寫
'abc'
>>> "abc".find('b')        ——— find() 方法尋找 'b' 的位置 (從 0 開始)
1
>>> "abc".replace('b', 'z') ——— replace() 方法將所有的 'b' 取代成 'z'
'azc'
```

- 不同的物件會有不同的方法。例如：字串物件與整數物件。

資料型別

ex1-2

- 除字串物件以雙引號或單引號來表示，寫程式常有整數與浮點數(小數) 物件，例如：111 與 11.1。

```
>>> 111 + 111 ———— 整數物件相加
222
>>> "111" + "111" ———— 字串物件串聯
'111111'
```

- 上述 + 的運算，因物件的資料不同而產生不同的結果。物件的種類，程式語言稱之為『物件型態』或『資料型態』(Data Type)。

練習：要分清楚資料型別

ex1-3

- 兩個資料型別若不同，可能會導致程式錯誤。

```
>>> 111 + "111" —— 不同型別的資料相加發生錯誤
Traceback (most recent call last):

  File "<ipython-input-6-4832c22160be>", line 1, in
<module>
    111 + "111"

TypeError: unsupported operand type(s) for +: 'int'
and 'str'
```

型別轉換

ex1-4

- 兩個資料型別若要運算，可以使用型別轉換。

```
>>> str(111) + "111"   ——— str() 可轉換物件為字串型別
'111111'
>>> 111 + int("111")   ——— int() 可轉換物件為整數型別
222
```

整數與浮點數

ex1-5

- 整數與浮點數的數學運算
  - ✓ +(加)、-(減)、*(乘)、/(取商)、//(取商，整數)、%(取餘數)、**(指數)。
  - ✓ Python 允許整數與浮點數直接運算，執行下列程式：

加法 (+)

```
>>> 10 + 5.5
15.5
```

減法 (-)

```
>>> 10 - 5.5
4.5
```

ex1-5

乘法 (*)

```
>>> 10 * 5.5
55.0
```

除法取商 (/)

```
>>> 10 / 5.5
1.8181818181818181
```

除法取整數商 (//)

```
>>> 10 // 5.5
1.0
```

ex1-5

取餘數 (%)

```
>>> 10 % 5.5
4.5
```

指數 (**)

```
>>> 4 ** 0.5, 8 ** (1/3)
(2.0, 2.0)
```

TIP

1. 整數與浮點數做運算，結果一定為浮點數。
2. 只有整數與整數做除法，結果為浮點數。

常用的變數運算

ex1-6

- 把整數加上特定的值：

```
>>> x = 1
>>> x = x + 1
>>> x
2
```

- 常用的簡式：

簡式	意義	簡式	意義
x += 2	x = x + 2	x /= 2	x = x / 2
x -= 2	x = x – 2	x //= 2	x = x // 2
x *= 2	x = x * 2	x %= 2	x = x % 2

變數

ex1-7

- 『變數』(variable) 就像是掛在物件的名牌，幫物件取名之後，讓我們方便識別物件與操作，其語法為：

```
變數名稱 = 物件
```

- 例如：

```
>>> n1 = 123456789 ——— 將整數物件 123456789 指派給變數 n1
>>> n2 = 987654321 ——— 將整數物件 987654321 指派給變數 n2
>>> n1 + n2 ——————— 實際上是 123456789 + 987654321
1111111110
```

寫出可讀性高的程式 (1/2)

ex1-8

- 使用有意義的變數 (variable) 的名稱。

```
a = 3.1
b = 2.2
c = a * b * b
```

```
pi = 3.1
radius = 2.2
# 使用公式計算圓面積
circle_area = pi * radius * radius
```

寫出可讀性高的程式 (2/2)

- 將程式加上註解。
  - ✓ 註解可以幫助其他人了解這個程式。
- 較佳的註解：

```
# 註解 1：使用公式計算圓面積
```

- 不好閱讀的註解：

```
# 註解 2：將圓周率乘半徑乘半徑
```

內建函式

ex1-9

- 『函式』(function) 是一段預先寫好的程式，方便重複使用。而程式先將經常使用到的功能以函式的形式先寫好，稱為『內建函式』。
- 例如：print() 是最常用的顯示函數：

```
>>> print("abc") ——— 顯示字串物件
abc
>>> print("abc".upper()) ——— 顯示字串物件.方法的執行結果
ABC
>>> print(111 + 111) ——— 顯示整數物件運算的結果
222
```

匯入模組

ex1-10

- 內建函式不就越多越好？若內建函式無限制增加，會導致啟動速度越來越慢，執行時佔用的記憶體越來越多。
- 『模組』(module)，就是將同一類的函式打包成模組，預設不會啟用。需要時，再用匯入 (import) 的方式啟用。預先寫好的稱為『內建模組』。

```
>>> import time ——— 匯入時間相關的 time 模組
>>> time.sleep(3)  ——— 執行 time 模組的 sleep() 函式，暫停 3 秒


>>> from time import sleep ——— 從 time 模組裡匯入 sleep() 函式
>>> sleep(5) ——— 執行 sleep() 函式，暫停 5 秒
```

練習：匯入模組

ex1-11

程式 暫停 3 秒後，印出 Hello World! 字串物件

```
#  暫停 3 秒後，印出 Hello World!
from time import sleep
sleep(3)
print("Hello World!")
```

指的是在程式編輯窗格中編輯與執行。

觀念整理

各種程式語言的語法邏輯都差不多，就像人類語言的文法。

1. 程式的每一個東西都是物件，有些物件有其特定的操作方法。
2. 基本物件有資料型別，型別不同，結果不同。甚至有時會錯誤。
3. 變數是物件的名牌而已，也方便程式設計師操作。
4. 使用有意義的變數名稱與註解。
5. 內建函式是經常用的函式，預先寫好的。
6. 適當的匯入模組，能精簡效能。

免費的雲計算

Google colab

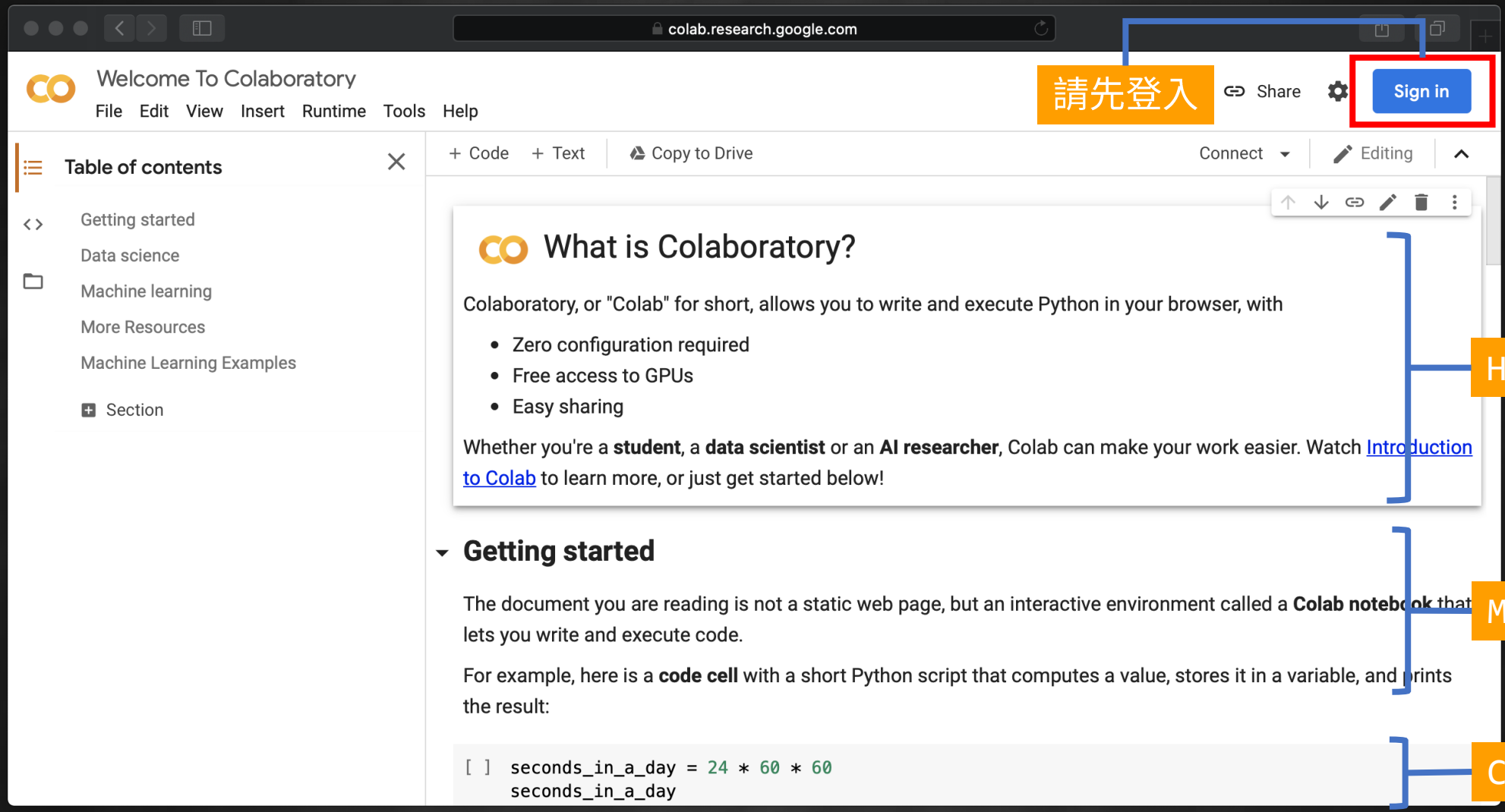
Welcome To Colaboratory

- 快速上手
- 雲端虛擬主機的管理與設定
- 目錄窗格與檔案管理
- 偏好設定

快速上手

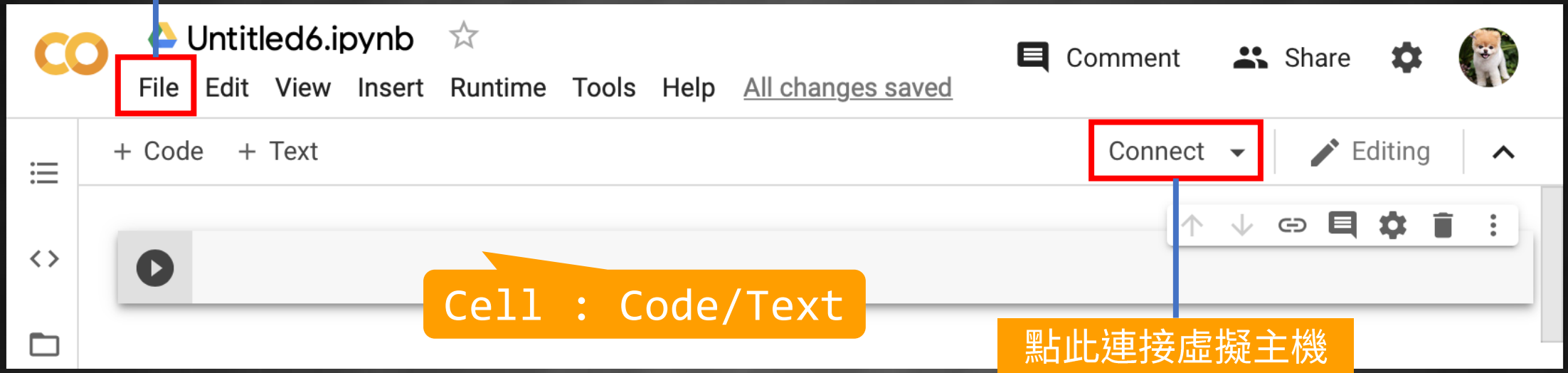
快速上手

ex1-12


colab.research.google.com
Welcome To Colaboratory
File Edit View Insert Runtime Tools Help
請先登入
Share
Sign in
Table of contents
Getting started
Data science
Machine learning
More Resources
Machine Learning Examples
Section
+ Code + Text
Copy to Drive
Connect
Editing
What is Colaboratory?
Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with
Zero configuration required
Free access to GPUs
Easy sharing
Whether you're a student, a data scientist or an AI researcher, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!
Getting started
The document you are reading is not a static web page, but an interactive environment called a Colab notebook that lets you write and execute code.
For example, here is a code cell with a short Python script that computes a value, stores it in a variable, and prints the result:
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
HTML
Markdown
Cell

快速上手 - 新建筆記本

File/New notebook



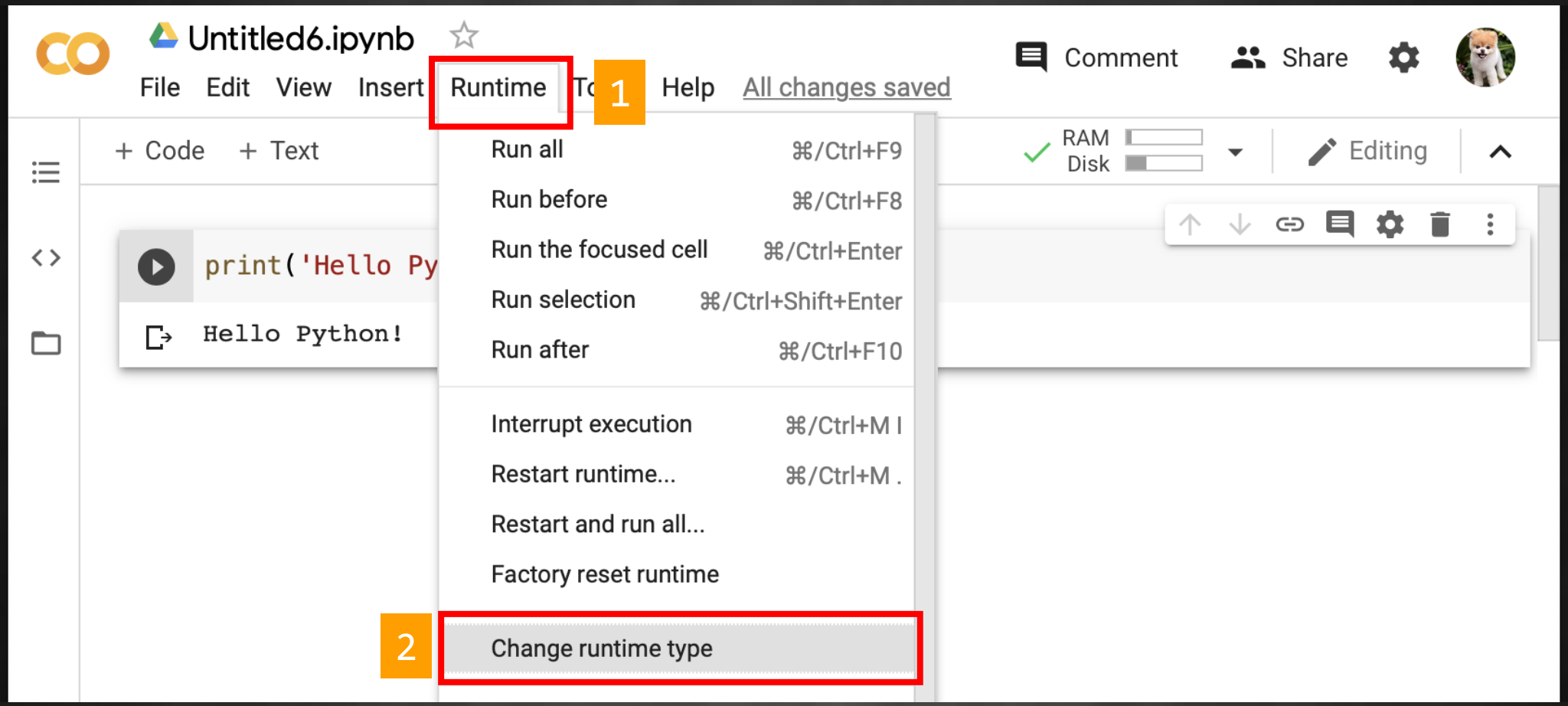
快速上手 – 執行程式碼

ex1-13

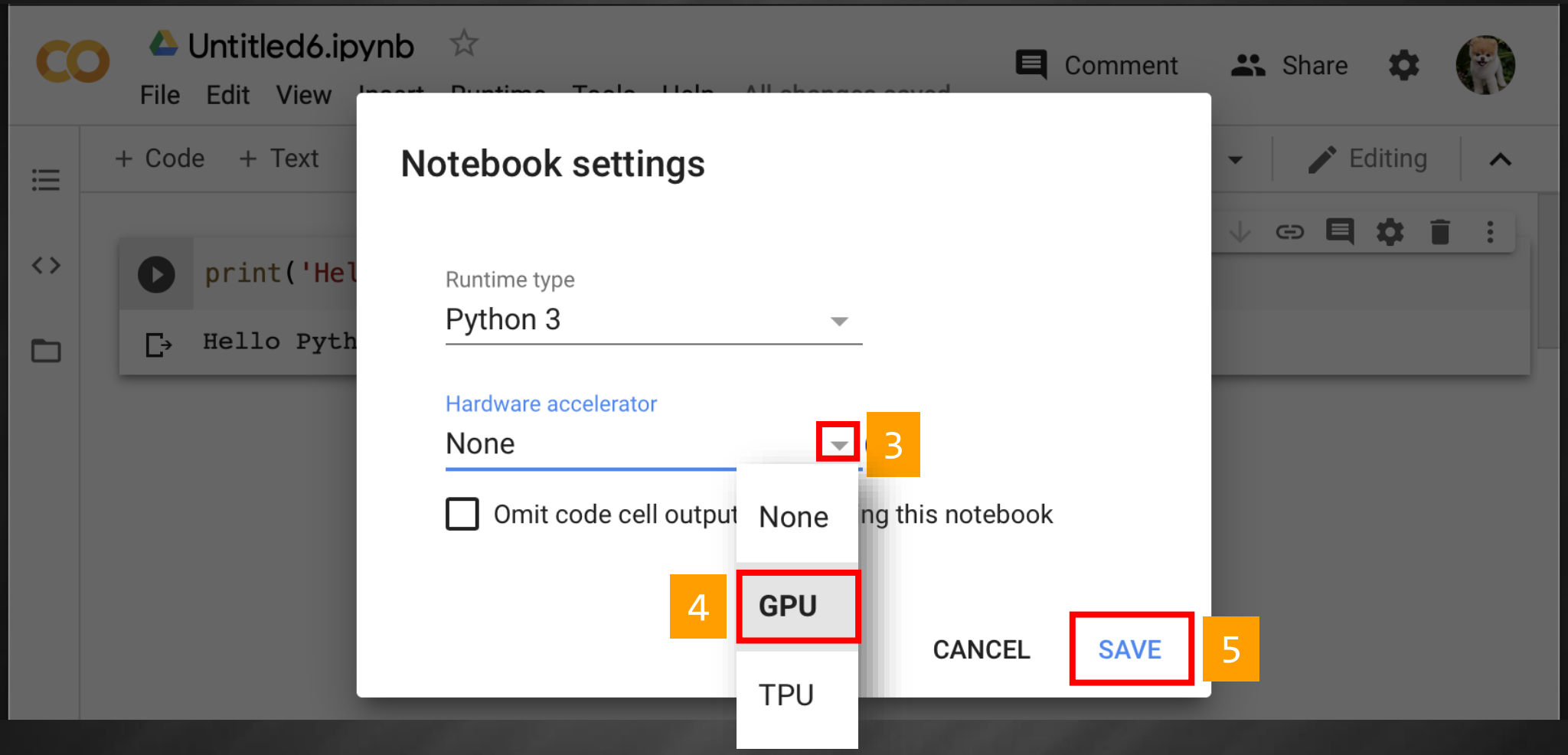
Welcome To Colaboratory

- 快速上手
- 雲端虛擬主機的管理與設定
- 目錄窗格與檔案管理
- 偏好設定

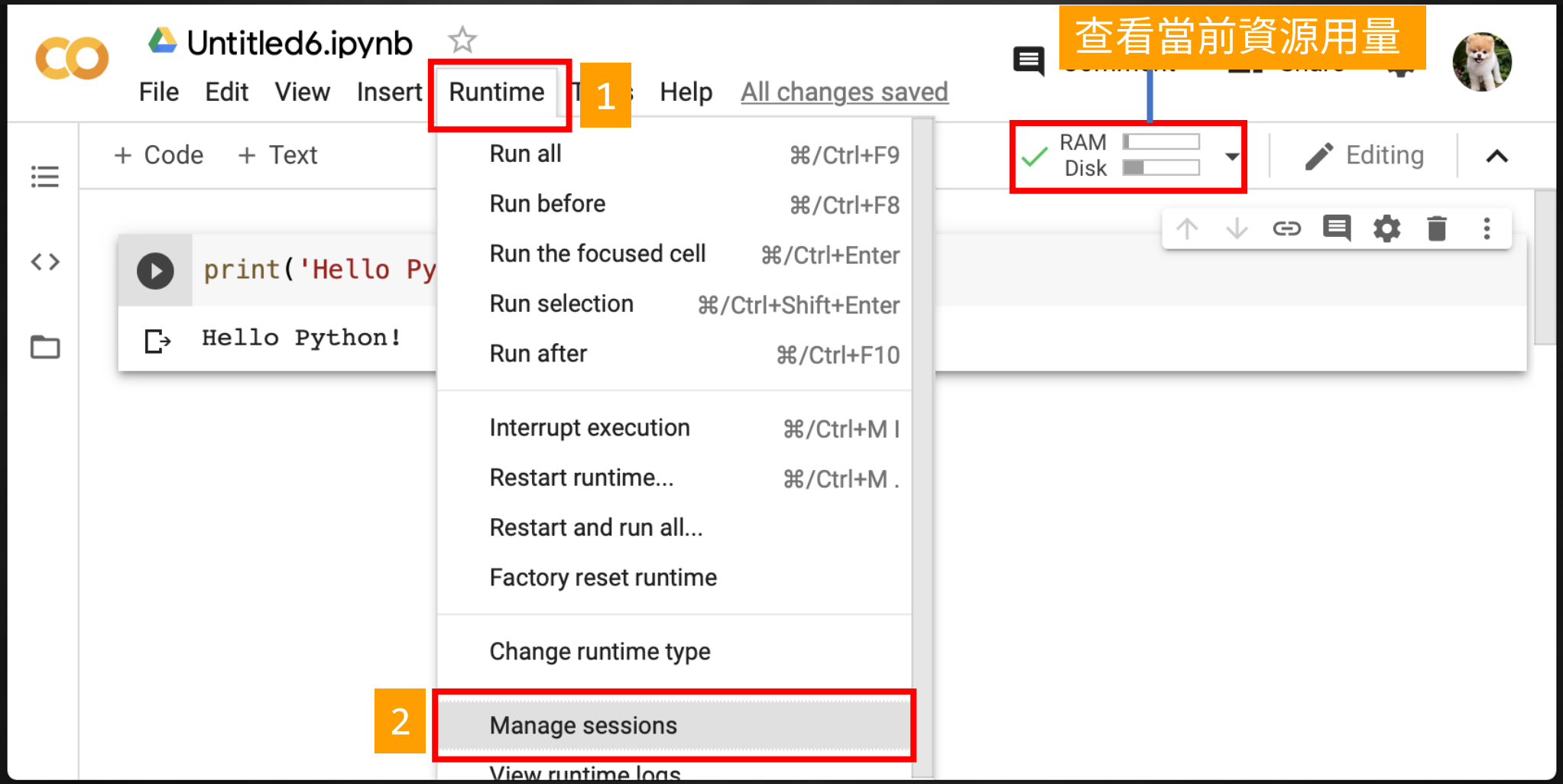
雲端主機的管理與設定 - GPU 加速



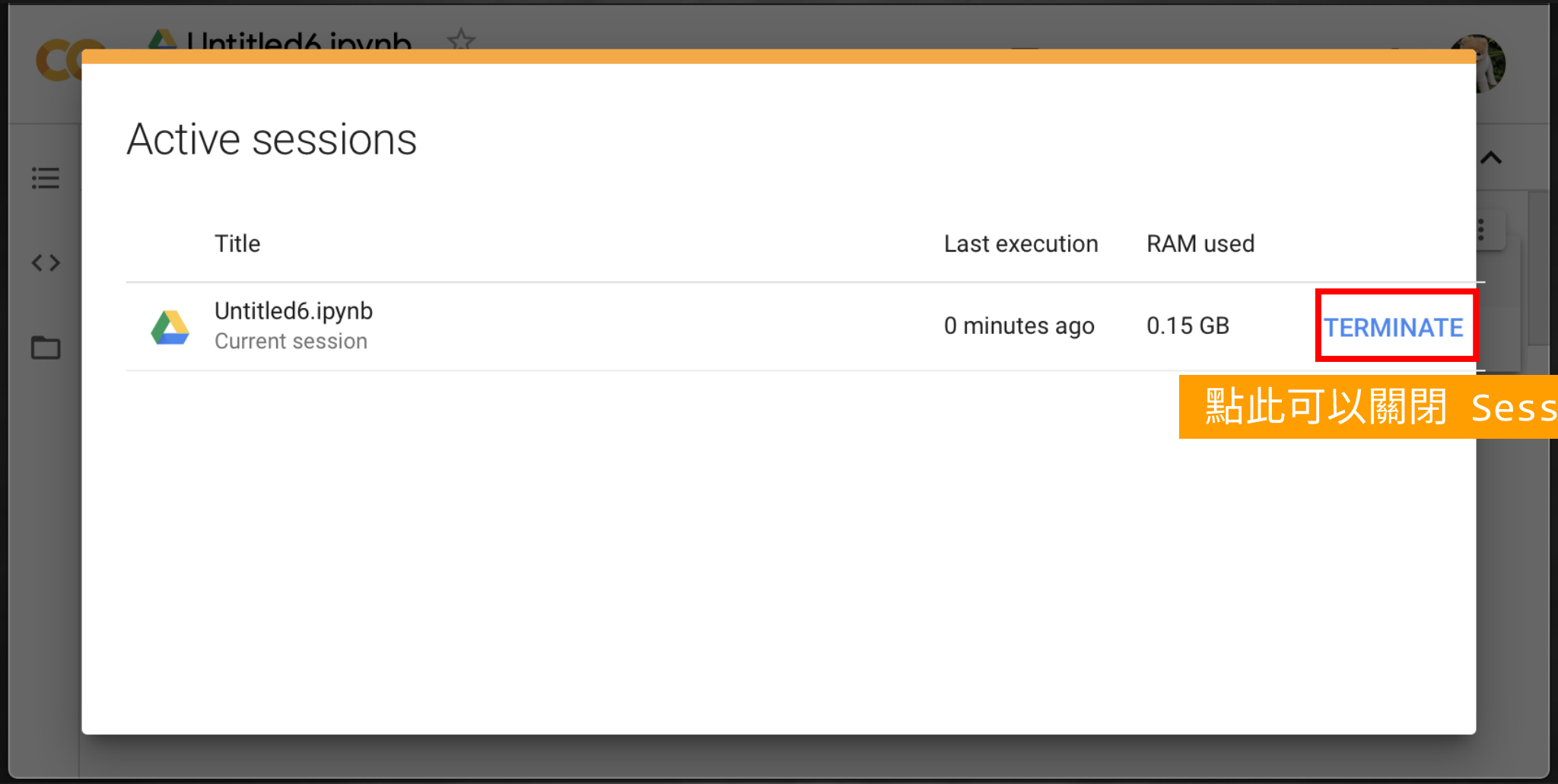
雲端主機的管理與設定 - GPU 加速



雲端主機的管理與設定 – Session 管理



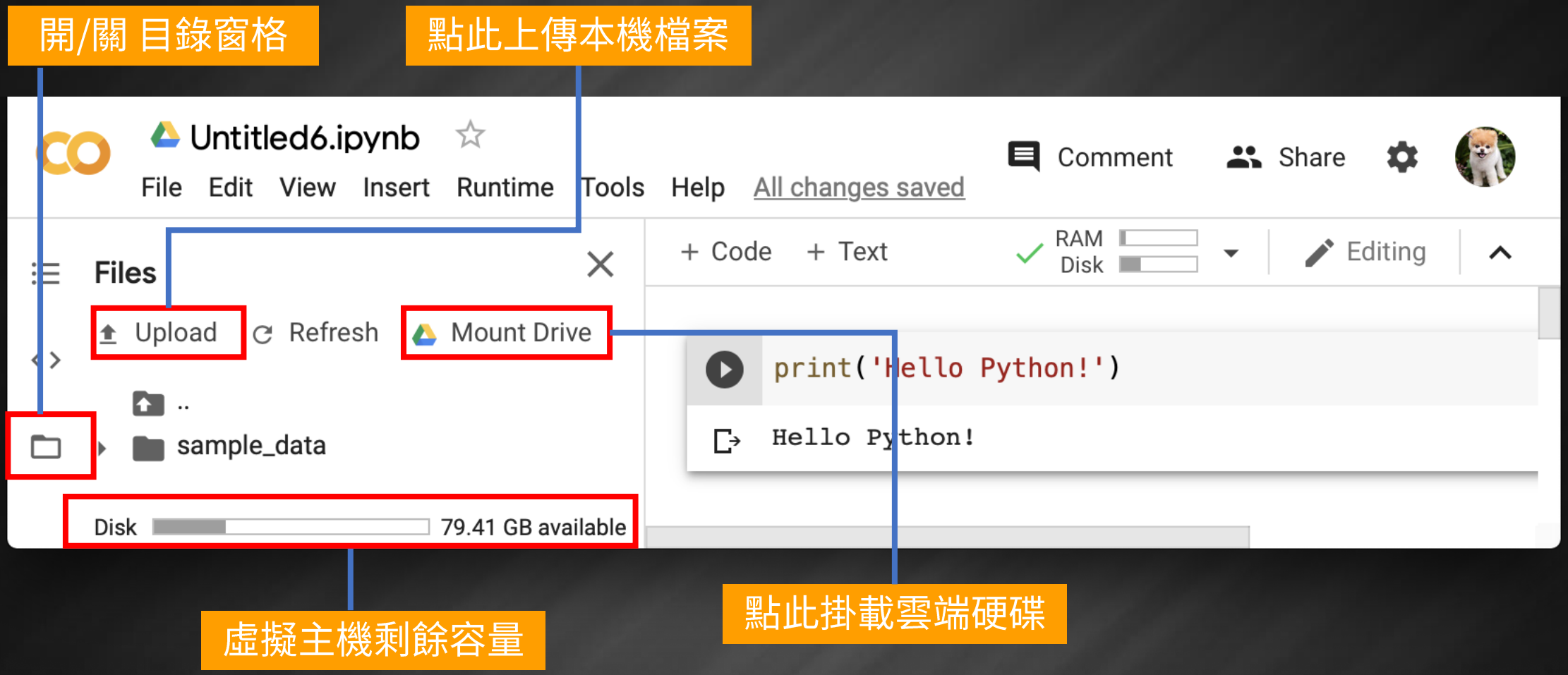
雲端主機的管理與設定 – Session 管理



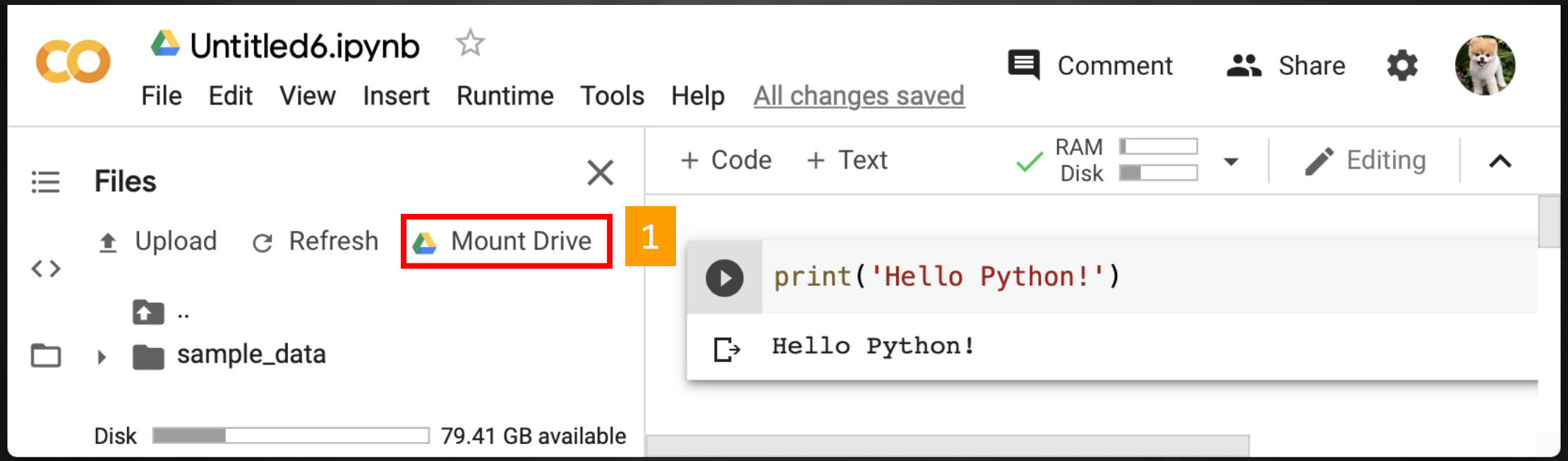
Welcome To Colaboratory

- 快速上手
- 雲端虛擬主機的管理與設定
- 目錄窗格與檔案管理
- 偏好設定

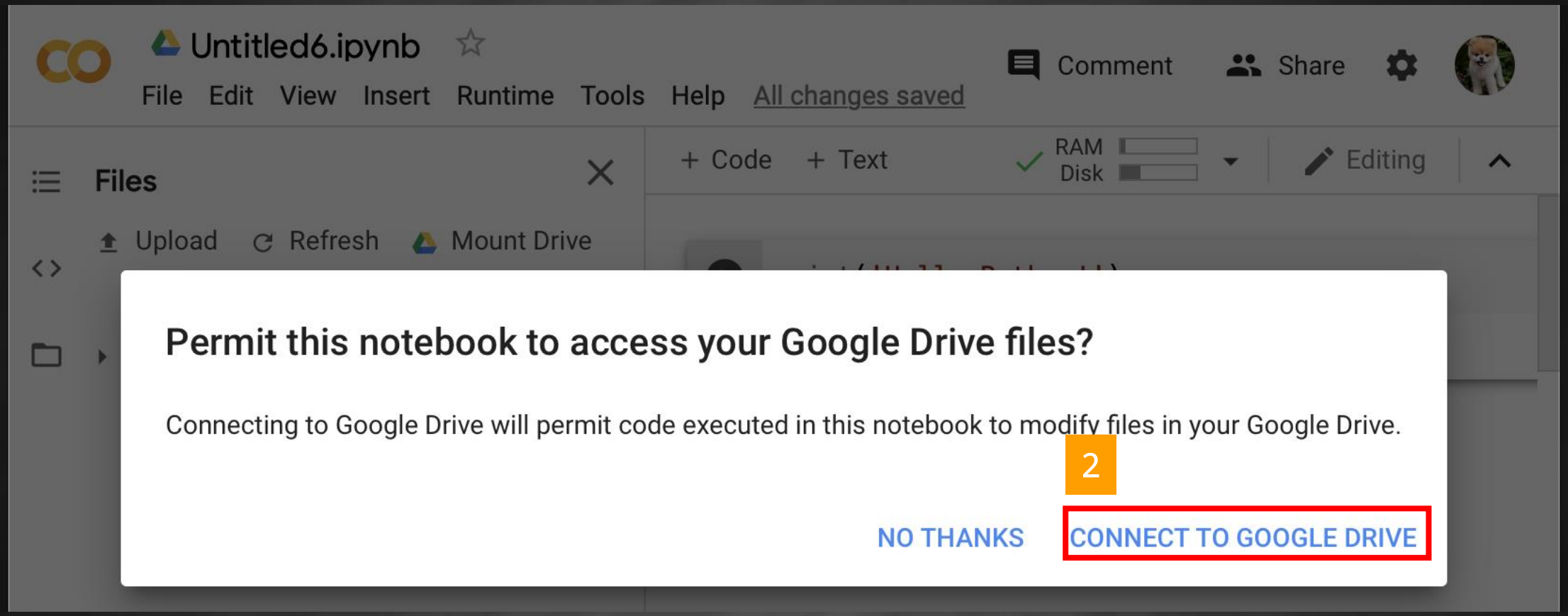
目錄窗格與檔案管理


開/關 目錄窗格
點此上傳本機檔案
Untitled6.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
Upload
Refresh
Mount Drive
+ Code
+ Text
RAM
Disk
Editing
print('Hello Python!')
Hello Python!
..
sample_data
Disk
79.41 GB available
虛擬主機剩餘容量
點此掛載雲端硬碟

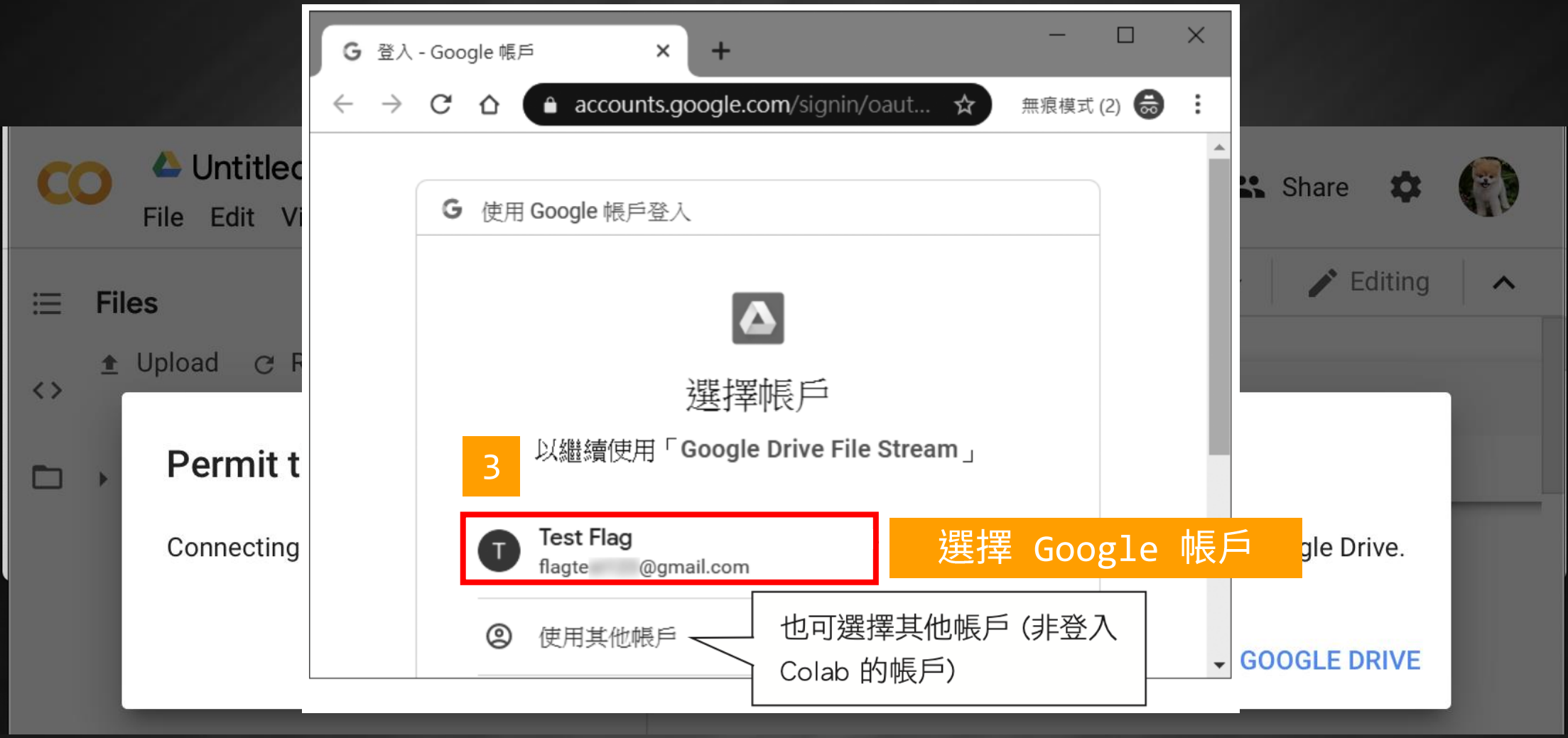
目錄窗格與檔案管理 – 掛載雲端



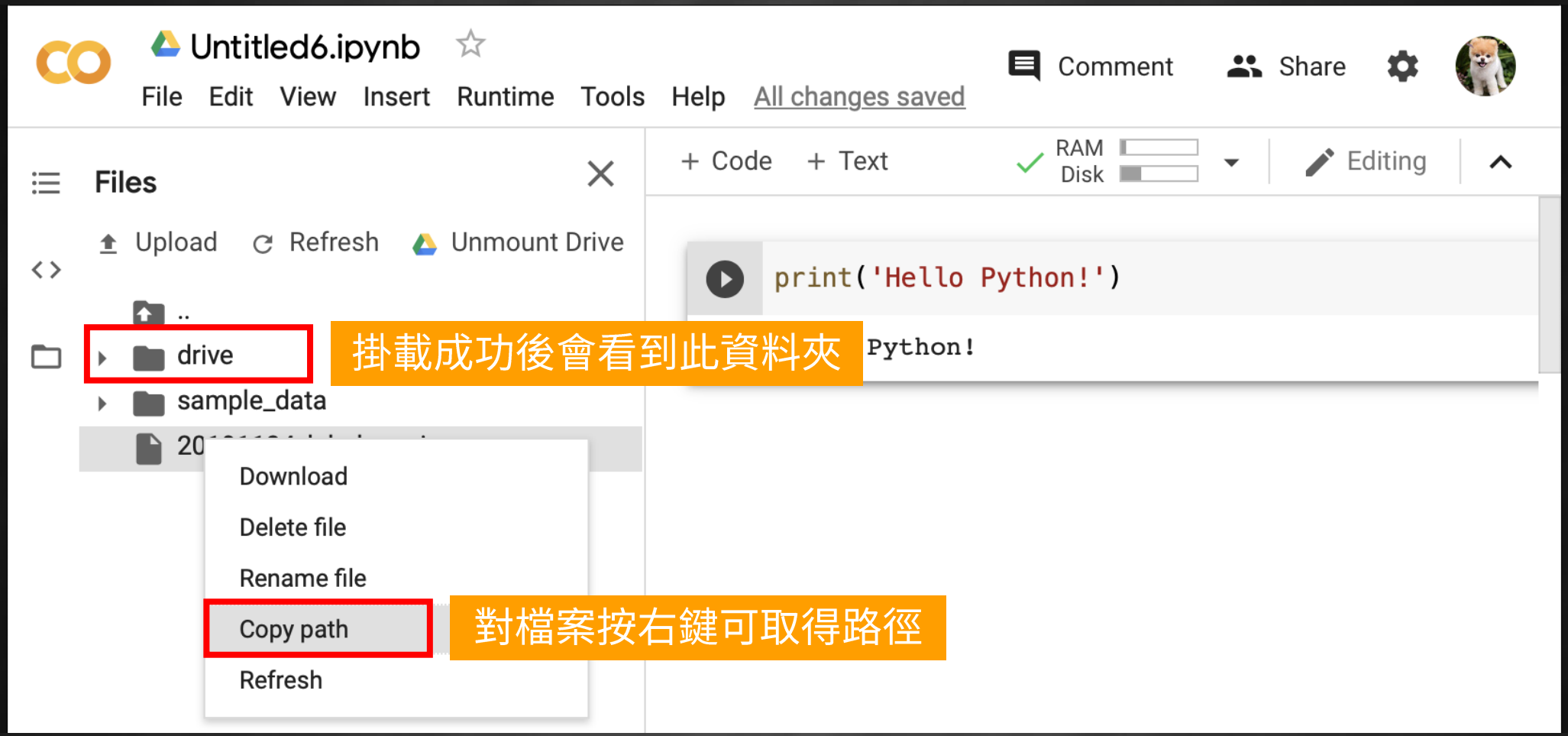
目錄窗格與檔案管理 – 掛載雲端



目錄窗格與檔案管理 – 掛載雲端

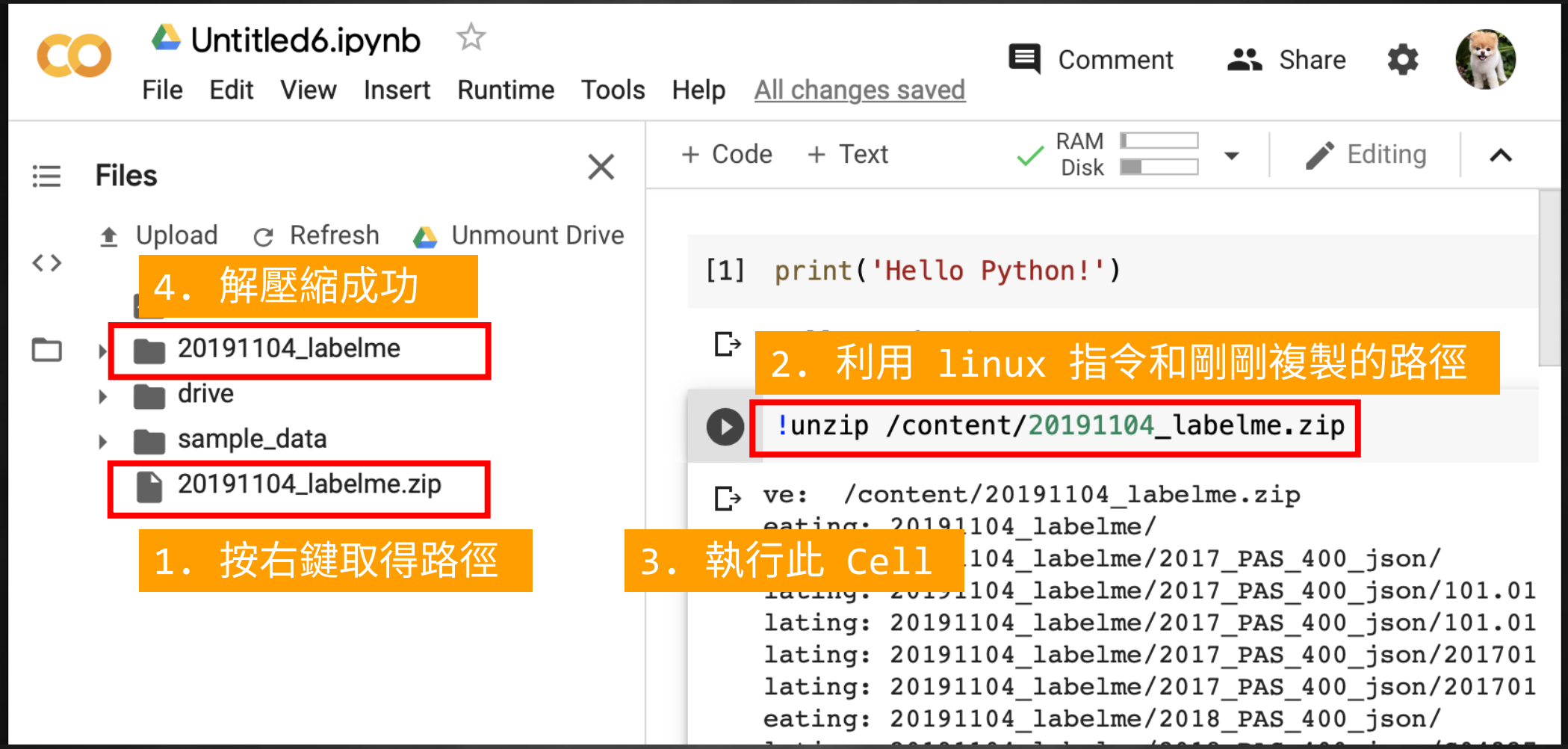


目錄窗格與檔案管理 – 掛載雲端



目錄窗格與檔案管理 – 線上解壓縮

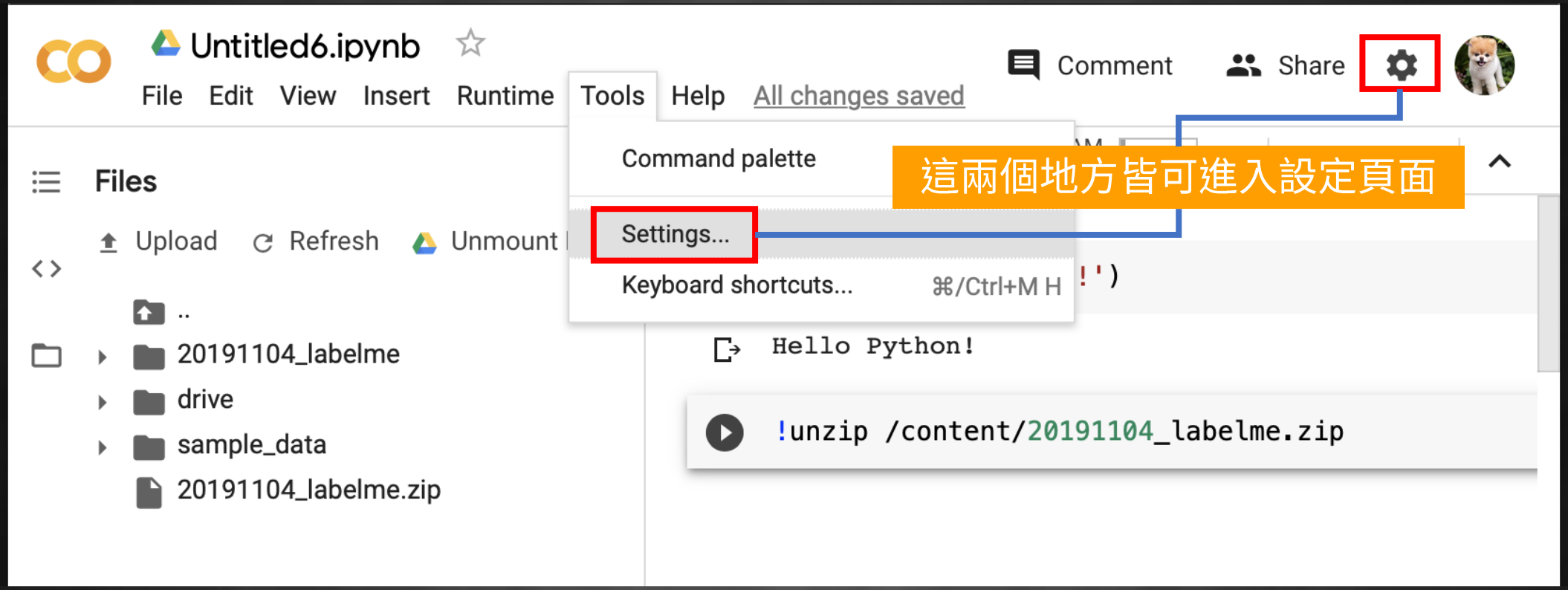
ex1-14



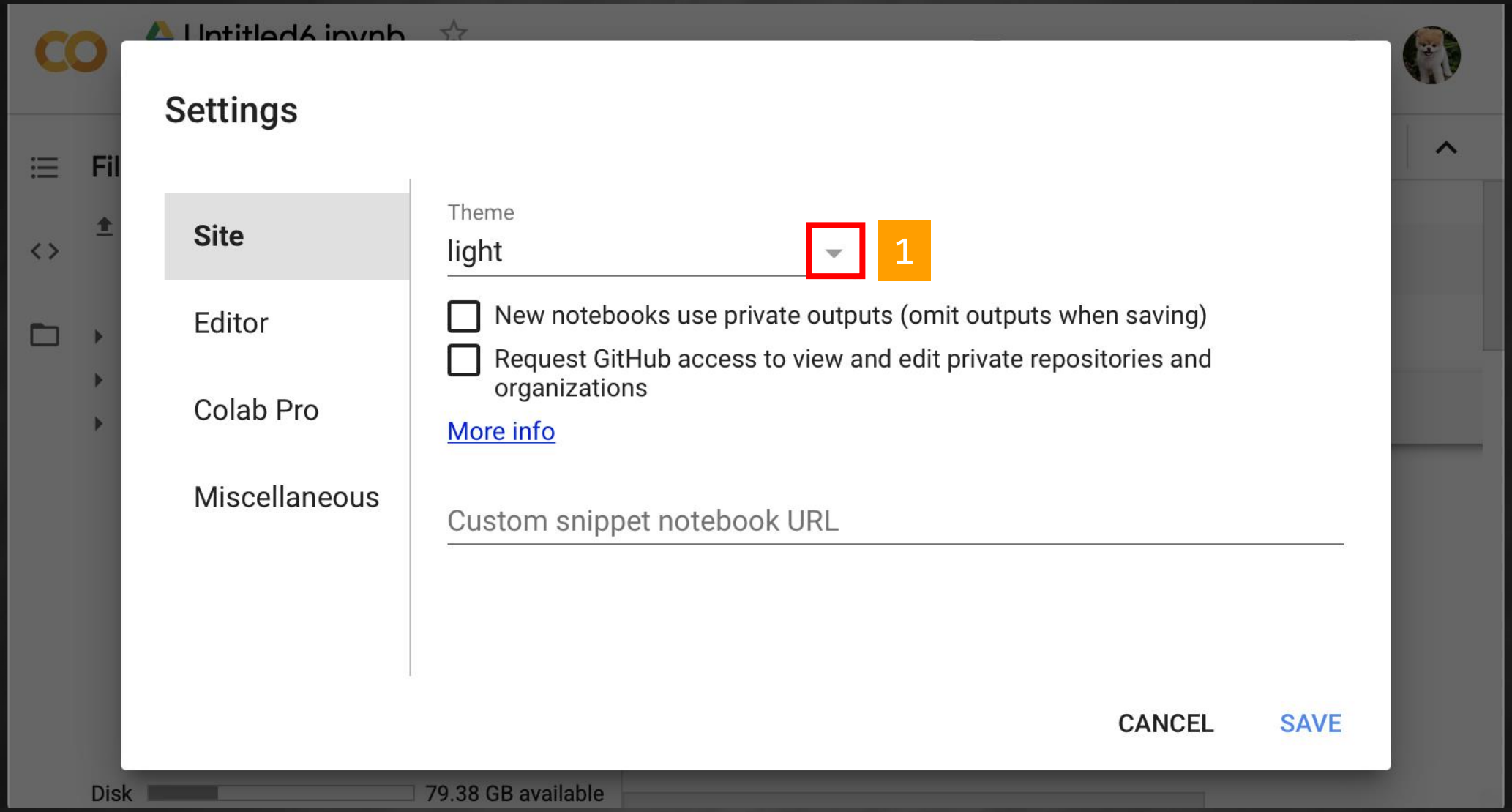
Welcome To Colaboratory

- 快速上手
- 雲端虛擬主機的管理與設定
- 目錄窗格與檔案管理
- 偏好設定

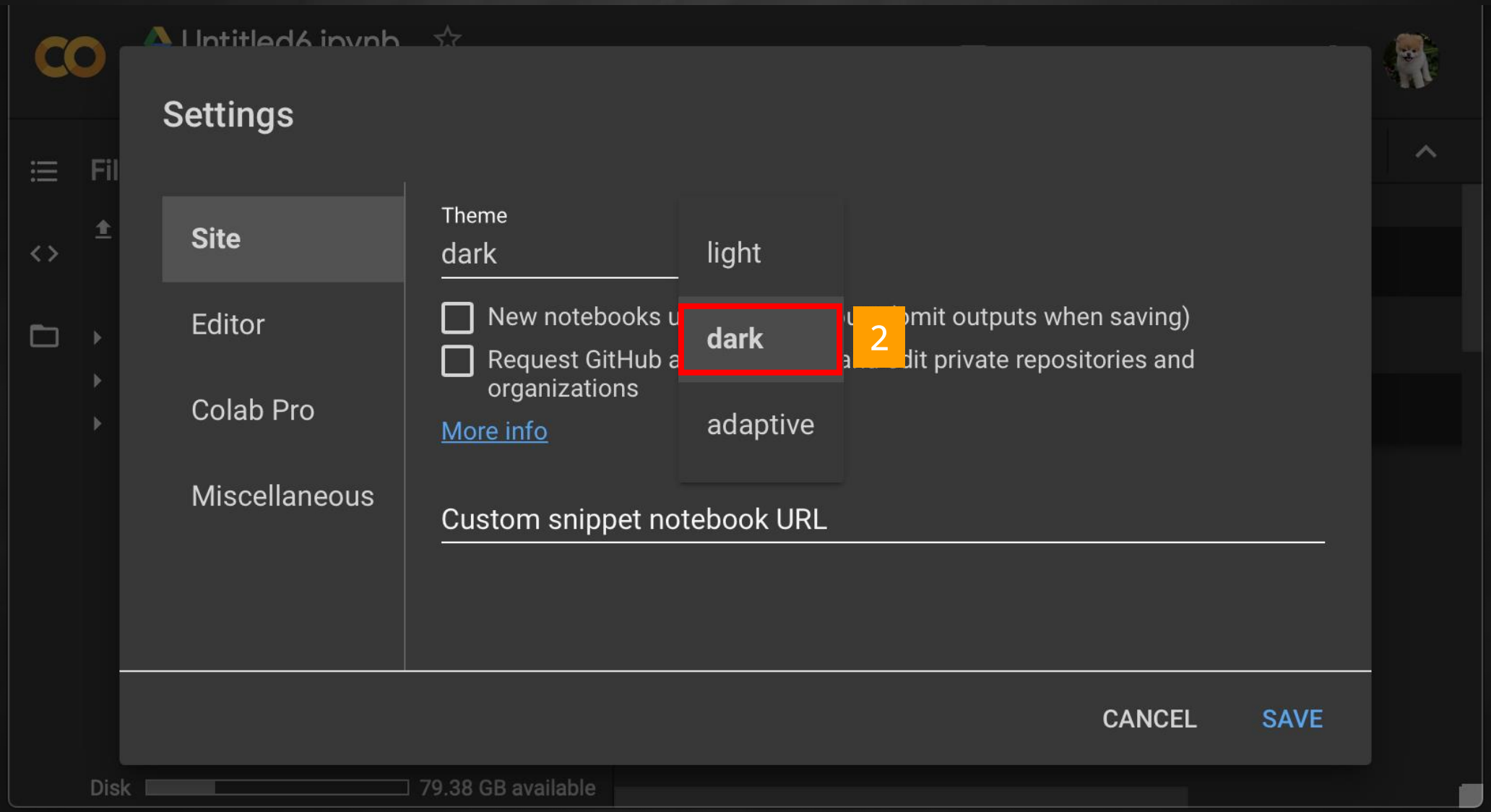
偏好設定


Untitled6.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Command palette
Settings...
Keyboard shortcuts... ⌘/Ctrl+M H
這兩個地方皆可進入設定頁面
Files
Upload
Refresh
Unmount
..
20191104_labelme
drive
sample_data
20191104_labelme.zip
Hello Python!
!unzip /content/20191104_labelme.zip

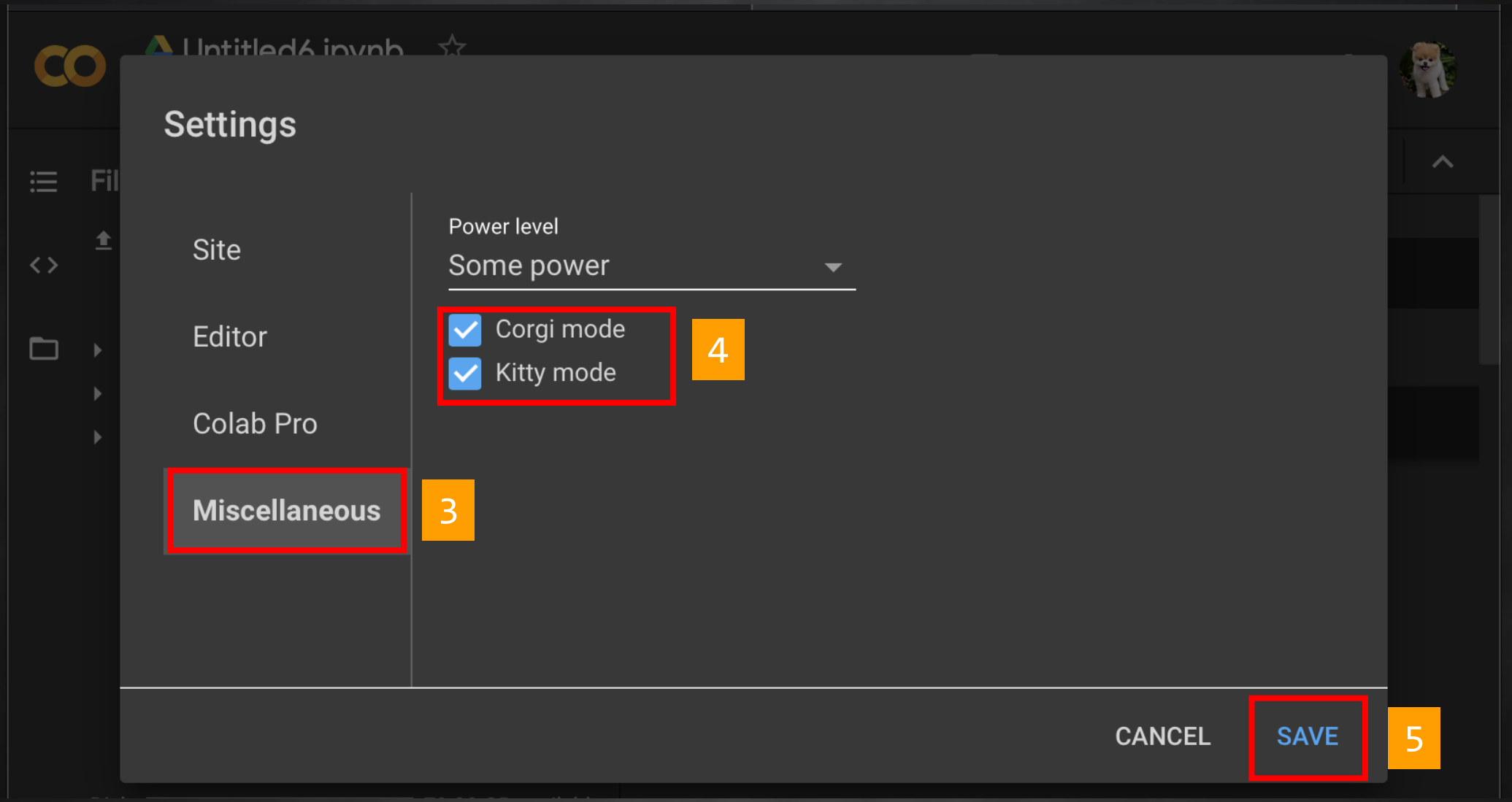
偏好設定 – 主題設定（暗黑）


Settings
Site
Editor
Colab Pro
Miscellaneous
Theme
light
1
New notebooks use private outputs (omit outputs when saving)
Request GitHub access to view and edit private repositories and organizations
More info
Custom snippet notebook URL
CANCEL
SAVE
Disk
79.38 GB available

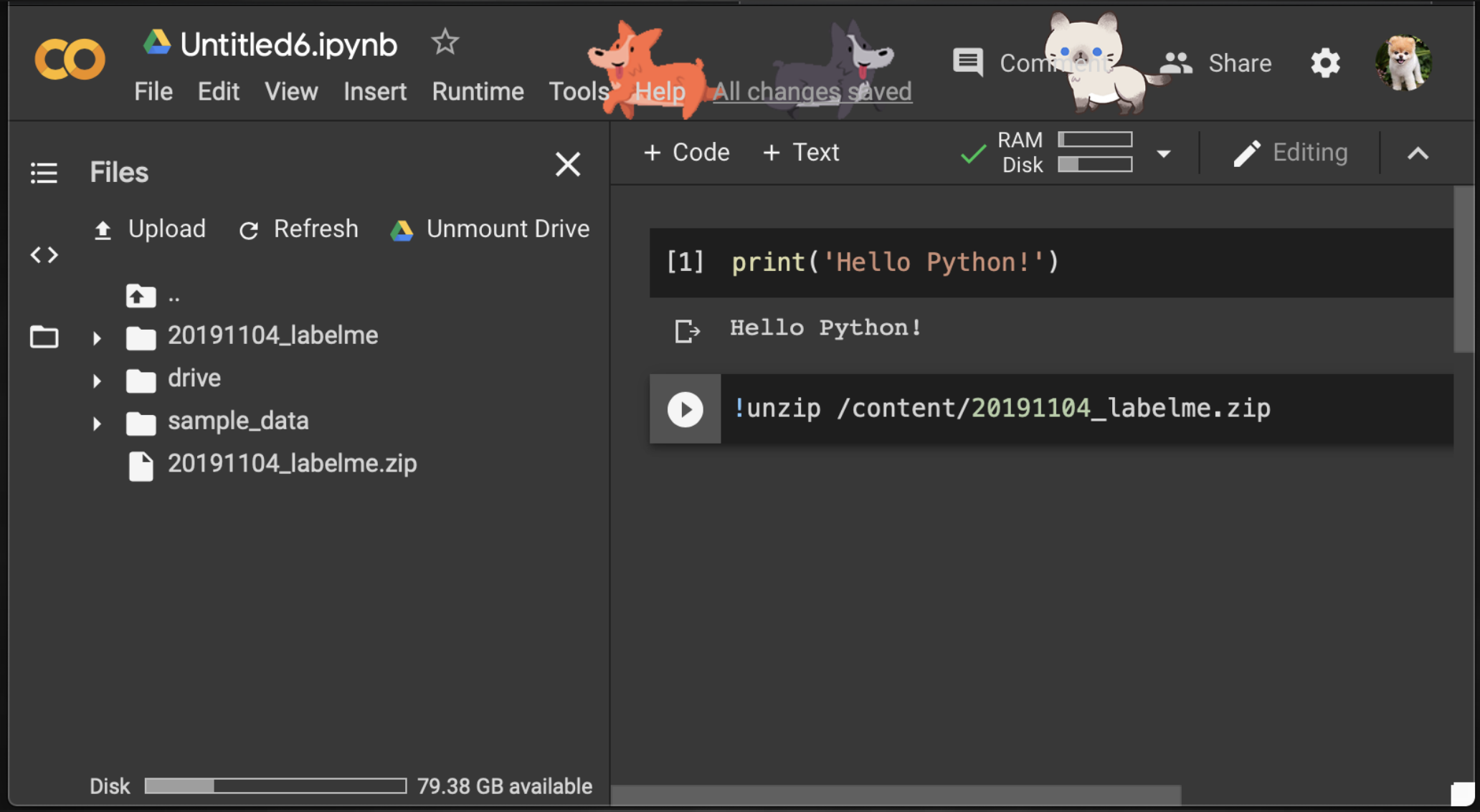
偏好設定 – 主題設定（暗黑）


Settings
Site
Editor
Colab Pro
Miscellaneous
Theme
dark
light
dark
adaptive
2
New notebooks u
mit outputs when saving)
Request GitHub a
dit private repositories and
organizations
More info
Custom snippet notebook URL
CANCEL
SAVE
Disk
79.38 GB available

偏好設定 – 柯基犬與小貓模式



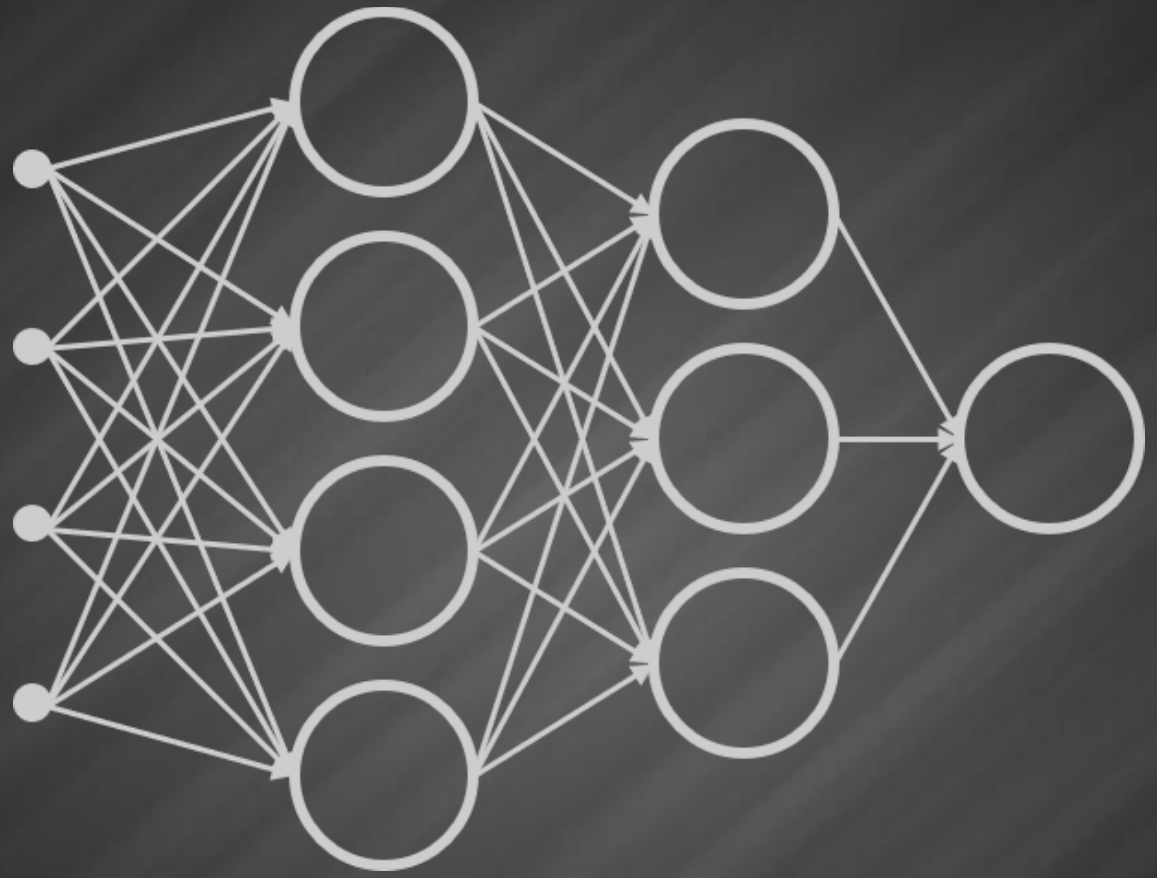
偏好設定 – 柯基犬與小貓模式



Keras 基本介紹

- 建立神經網路像是堆積木一樣，簡單明瞭。
- 支援處理影像辨識、文字…，等內容的神經網路 (ex：CNN、RNN)。
- 程式碼在更換硬體環境時 (CPU、GPU)，無須做任何更改。

用 Keras 建構神經網路



建立空的神經網路模型

ex1-15

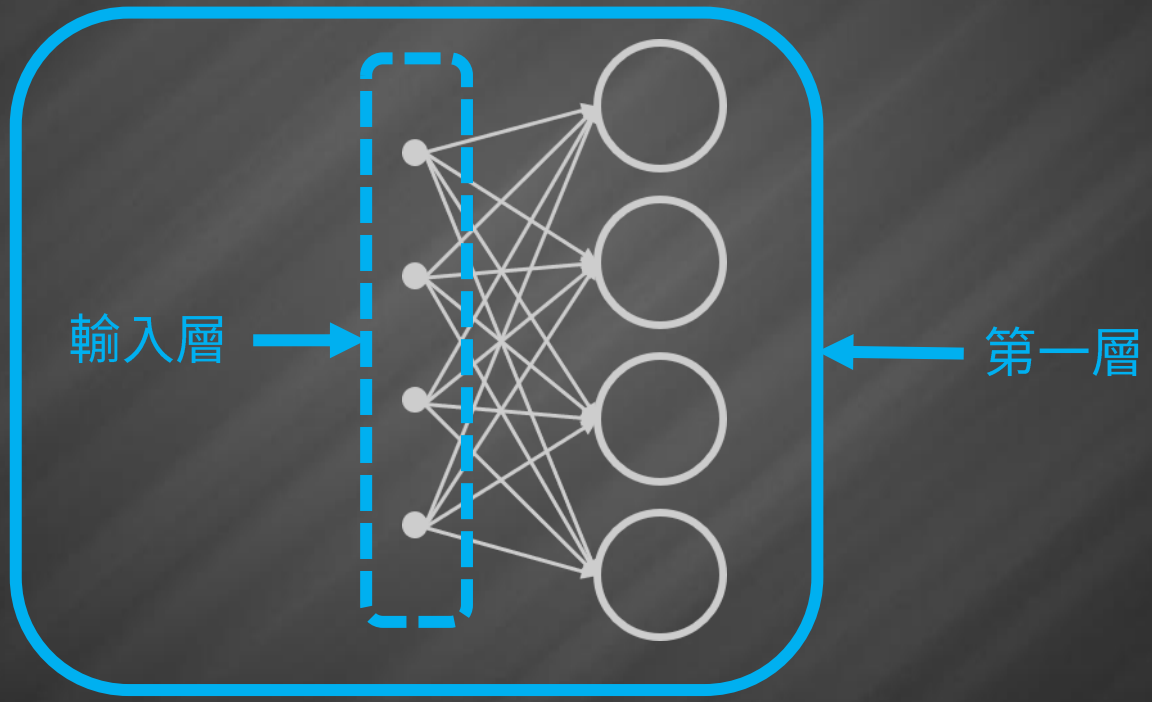
```
# 匯入 Keras 的序列式模型類別
from tensorflow.keras.models import Sequential

# 匯入 Keras 的密集層類別
from tensorflow.keras.layers import Dense

# 建立神經網路
model = Sequential() ← 建立序列模型物件，並指定給 model 變數，這時的
model 就是一個神經網路了，但內容是空的
```

加入第一層的神經層（包含輸入層功能）

ex1-15

```
model.add(Dense(4, activation='relu', input_shape= (4, )))  ← 輸入層形狀
```

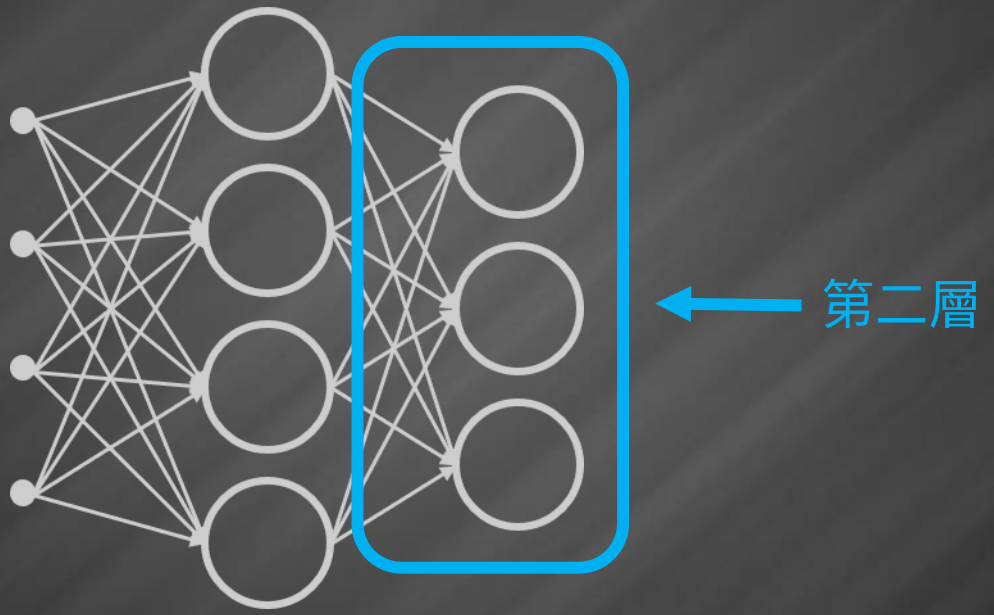
密集層 (Dense layer) 是最普通的神經層，它的每一個神經元都會與上一層的每個神經元連接，又稱為全連接層



加入第二層的神經層

ex1-15

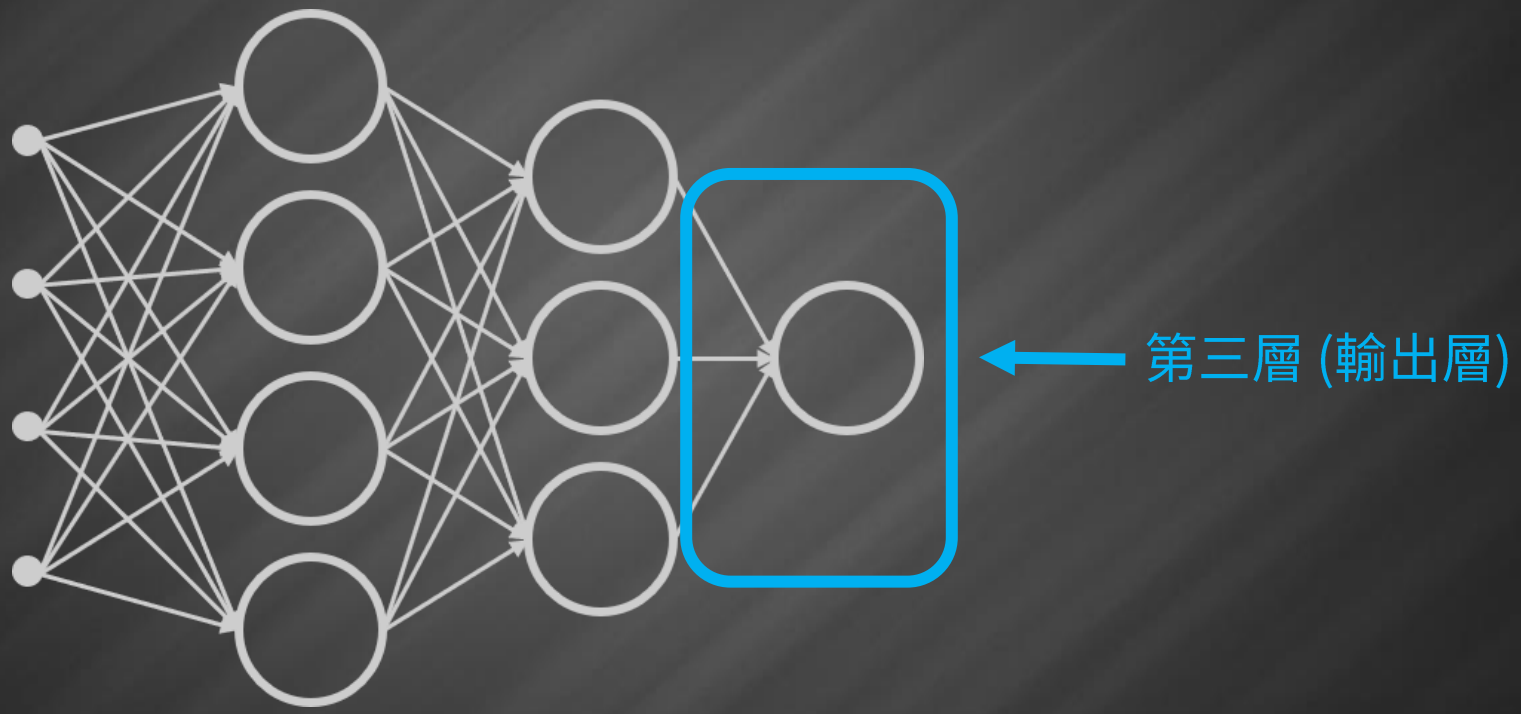
```
model.add(Dense(3, activation='relu')) ← 除第一層之外都不用指定
input_shape 參數
```



加入輸出層

ex1-15

```
model.add(Dense(1)) ← 再加入一個密集層，只有 1 個神經元，並且不使用激活函數
```



顯示當前模型架構

ex1-15

```
model.summary()  ← 顯示當前模型架構及參數
```

LAB01 預測台灣房租

實驗目的	利用神經網路的迴歸模型來預測『台北市中山區』房租。
材料	無

開發環境

Colab

資料介紹

- 此資料集蒐集自『591 房屋交易網』，地點為『台北市中山區』，每筆資料蒐集了『坪數』、『樓層』、『是否可以開伙』和『是否可以養寵物』共 4 種特徵，並有對應的『房租價格』。
  - ✓ 總資料數：689
  - ✓ 特徵資料數量：4
  - ✓ 標籤資料數量：1（房租價格）

上傳房屋資料、第三方模組

ex1-16

```
# 匯入「房屋txt檔」和 『第三方函式庫』到 Colab
from google.colab import files

uploaded = files.upload()    # 匯入房屋 .txt 檔
uploaded = files.upload()    # 匯入第三方函式庫 keras_lite_convertor
```

讀取檔案

ex1-17

```
# 讀取 house.txt 檔案, 並得出特徵和標籤
import keras_lite_convertor as kc

path_name = 'house.txt'  # 檔案路徑
Data_reader = kc.Data_reader(path_name, mode = 'regression')  # 指定讀檔模式 (regression 適用於迴歸預測)
data, label = Data_reader.read(random_seed = 12)  # 將檔案讀到的 5 種資料分為『特徵』和『標籤』，並設定亂數種子為 12 (data，label 為 numpy.ndarray 格式
```

物件一定有資料型態，可用 type(物件或變數名稱) 查詢。例如：

- type(data)：會回傳 numpy.ndarray。

等號左邊是指定變數，不會顯示物件內容。可用變數名稱或 print(變數名稱)，來檢查物件的內容。

- print(data)：會回傳 numpy 物件的內容。

Python 的資料結構（容器）

Python 的基本資料結構

ex1-18

- 字串容器：由字元組成。例如：string = "52python"。
- tuple 容器：由資料物件組成。例如：tuple = (1, (2,), 3)。
- 串列容器：由資料物件組成。例如：list = [1, [2], 3]。
- 集合容器：由資料物件組成。例如：set = {1, '2', 3}。
- 字典容器：由資料物件組成，以鍵：值表示。例如：dick = {'A':1, 'B':'2', 'C':3}。

Python 的第三方資料結構：numpy.array

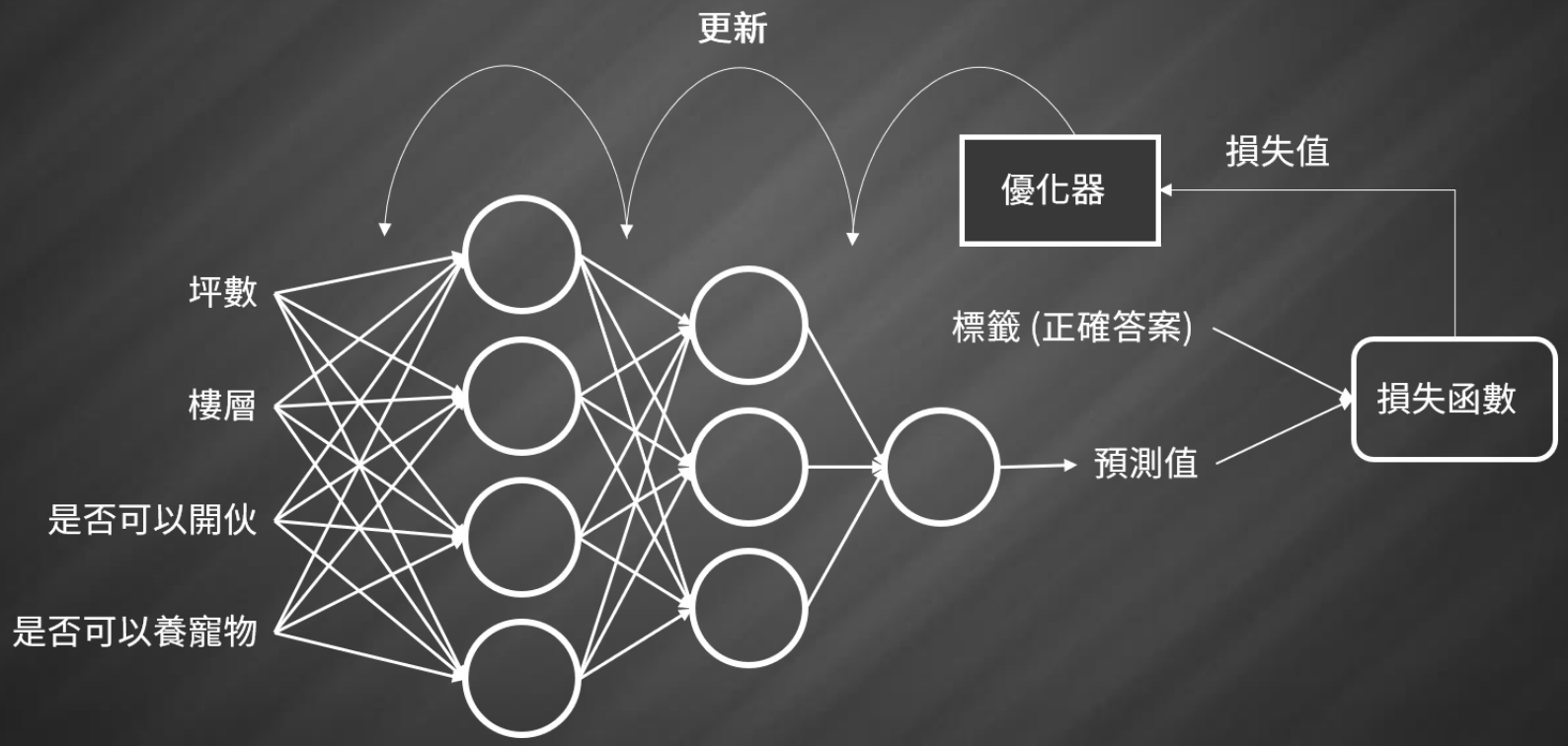
ex1-19

- Numpy：Python 的擴充模組，常用於資料處理。

```
import numpy as np                          # 匯入 numpy
a = np.array([10, 2, 45, 32, 24])           # 建立五個元素

>>> len(a)          # len() 會回傳容器內元素的數量

>>> a[2:4]          # 索引取值 (位置 2 ~ 3)

>>> a[:4]           # 索引取值 (位置 0 ~ 3)
```

訓練集、驗證集、測試集

- 訓練集：神經網路訓練時只會看到訓練集。(就像學生在學習時，寫的例題)
- 驗證集：訓練過程使用驗證集來模擬測試。(就像學生的習題)
- 測試集：訓練完畢，用測試集來考他。(就像學生的期末考試)



訓練集

ex1-20

```
# 資料預處理
# 取資料中的 90% 當作訓練集
split_num = int(len(data) * 0.9)
train_data = data[:split_num]
train_label = label[:split_num]
```

容器的元素個素，可用 len(容器的變數名稱) 查詢。例如：

- len(train_data)：會回傳 620。
- len(train_label)：會回傳 620。

資料正規化

- 每種特徵值的屬性和範圍都不太一樣，例如：
  - ✓ 坪數：範圍介於 4 ~ 26 坪。
  - ✓ 是否可以樣寵物：只有 0 與 1。
- 這會導致數值越大，對權重的影響也越大，解決方式：
  - ✓ 讓每種特徵使用相同的計量標準。
- 訓練集的特徵：先將資料減掉平均，再將其除以標準差。(以 0 作為基準，標準差作為單位)
- 訓練集的標籤：除以標籤的最大值。(落在 0 ~ 1 之間)

標準差的計算

- 例如：一群孩童年齡的數值為 {5，6，8，9}
  - ✓ 第一步：計算平均值

$\overline{x}$

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

n = 4 （因為集合里有 4 個數），分別設為：$x_1 = 5,\ x_2 = 6,\ x_3 = 8,\ x_4 = 9$

$$\overline{x} = \frac{1}{4}\sum_{i=1}^{4} x_i$$ 用 4 取代 N

$$\overline{x} = \frac{1}{4}(x_1 + x_2 + x_3 + x_4)$$

$$\overline{x} = \frac{1}{4}(5 + 6 + 8 + 9)$$

$\overline{x} = 7$此為平均值。

標準差的計算

- ✓ 第二步：計算標準差

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

$$\sigma = \sqrt{\frac{1}{4}\sum_{i=1}^{4}(x_i - \overline{x})^2}$$ 用 4 取代 N

$$\sigma = \sqrt{\frac{1}{4}\sum_{i=1}^{4}(x_i - 7)^2}$$用 7 取代$\overline{x}$

$$\sigma = \sqrt{\frac{1}{4}\left[(x_1 - 7)^2 + (x_2 - 7)^2 + (x_3 - 7)^2 + (x_4 - 7)^2\right]}$$

$$\sigma = \sqrt{\frac{1}{4}\left[(5 - 7)^2 + (6 - 7)^2 + (8 - 7)^2 + (9 - 7)^2\right]}$$

$$\sigma = \sqrt{\frac{1}{4}\left((-2)^2 + (-1)^2 + 1^2 + 2^2\right)}$$

$$\sigma = \sqrt{\frac{1}{4}(4 + 1 + 1 + 4)}$$

$$\sigma = \sqrt{\frac{10}{4}}$$

$$\sigma = 1.5811$$

標準差的計算

- {5, 6, 8, 9}
  - ✓ 平均值：7
  - ✓ 標準差：1.5811
  - ✓ 先將資料減掉平均，再將其除以標準差
    - 5 – 7 = -2, -2/1.5811 = -1.2649
    - 6 – 7 = -1, -1/1.5811 = -0.6324
    - 8 – 7 = 1, 1/1.5811 = 0.6324
    - 9 – 7 = 2, 2/1.5811 = 1.264
- {15, 16, 18, 19}
  - ✓ 平均值：17
  - ✓ 標準差：1.5811
  - ✓ 先將資料減掉平均，再將其除以標準差
    - 15 – 17 = -2, -2/1.5811 = -1.2649
    - 16 – 17 = -1, -1/1.5811 = -0.6324
    - 18 – 17 = 1, 1/1.5811 = 0.6324
    - 19 – 17 = 2, 2/1.5811 = 1.264

特徴正規化

ex1-20

```
# 正規化
mean = train_data.mean()    # 平均數
data -= mean
std = train_data.std()      # 標準差
data /= std
```

標籤正規化

ex1-20

```
# 將 label 範圍落在 0 ~ 1 (label 正規化)
New_label = label / max(label)
```

訓練集、驗證集、測試集的資料形狀

- house.txt 有 689 筆資料
  - ✓ 訓練集：佔 90% (620 筆)。
  - ✓ 驗證集：10% 減掉測試集，剩下的當驗證集 (39 筆)。
  - ✓ 測試集：10% 中的最後 30 筆。

訓練集、驗證集、測試集的資料形狀

ex1-21

```
# 訓練集、驗證集、測試集的資料形狀
# 訓練集
train_data = data[:split_num]                    # 訓練用資料 (620 筆)
print(train_data.shape)
train_label = new_label[:split_num]              # 訓練用標籤 (620 筆)
# 驗證集
validation_data = data[split_num:-30]            # 驗證用資料 (39 筆)
print(validation_data.shape)
validation_label = new_label[split_num:-30]      # 驗證用標籤 (39 筆)
# 測試集
test_data = data[-30:]                           # 測試用資料 (30 筆)
print(test_data.shape)
test_label = new_label[-30:]                     # 測試用標籤 (30 筆)
```

建立神經網路架構

ex1-22

- 建立幾層，以及每層多少神經元，只能透過經驗或不段測試。
  - ✓ 建立一個含輸入層共 3 層的神經網路，其中兩個隱藏層皆設定為 20 個神經元。

```
# 建立神經網路架構
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()  # 建構網路模型
# 增加一層神經層，使用 ReLU 激活函數，輸入層有 4 個輸入特徵
model.add(Dense(20, activation = 'relu', input_shape = (4,)))

# 增加一層神經層，使用 ReLU 激活函數
model.add(Dense(20, activation = 'relu'))
model.add(Dense(1))                          # 增加輸出為 1 的輸出層
```

編譯及訓練模型

ex1-23

- 回歸問題，使用均方誤差，且設定優化器為 "adam"。
  - ✓ Adam 具備了不錯的自適應與動量，來尋找最佳權重。

```
# 編譯及訓練模型

# 編譯模型
model.compile(optimizer = 'adam', loss = 'mse', metrics = ['mae'])
history = model.fit(train_data, train_label, validation_data = (validation_data,
validation_label), epochs = 200)
```

- optimizer = 'adam'：設定最優化方法為 adam。
- loss = 'mse'：設定損失函式為均方誤差 mse。
- metrics = ['mae']：設定評估模型方式 mae 準確率。
- epochs = 200：訓練次數為 200 次。

查看損失值

ex1-24

- 將損失值的曲線顯示出來，確認神經網路有往目標前進。

```
# 查看損失值
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], "r", label = 'loss')
plt.plot(history.history['val_loss'], "b", label = 'val loss')
plt.legend()                    # 顯示標籤
plt.show()                      # 顯示圖片
```

- loss：使用訓練資料，得到的損失函式誤差值（值越小準確率越高)。
- val_loss：使用驗證資料，得到的損失函式誤差值（值越小準確率越高)。 。

資料比較圖

ex1-25

```
# 資料比較圖
import numpy as np

plt.figure(figsize = (10, 8))           # 定義一個視窗(10,8 為視窗大小)
plt.subplots_adjust(hspace = 0.3)       # 調整兩張圖的間距
# 實際值-預測值(* max(label) 表示恢復原始值)
error = test_label.reshape(30, 1) * max(label) - model.predict(test_data) * max(label)
# 把誤差分成 15 等份, 求出每一等份的長度
step = (max(error) - min(error)) / 15
# 寫出每一等份的值
interval = [i for i in range(int(min(error)), int(max(error)) + int(step), int(step))]
# 實際預測比較圖
width = 0.3
plt.subplot(2, 1, 1)                    # 第一張圖位於視窗裡的位置 (2列1行的第二個位置 - 上)
plt.xlabel("test data")                 # x軸名稱
plt.ylabel("money")
plt.bar(np.linspace(1, 30, 30) - width / 2, (test_label * max(label)).reshape(30), width =
width, label = 'actual')
plt.bar(np.linspace(1, 30, 30) + width / 2, (model.predict(test_data) *
max(label)).reshape(30), width = width, label = 'predict')
plt.legend()
```

最後測試：資料預測

ex1-26

```
# 建立欲預測的資料
data = np.array([[8, 5, 0, 0],
           [15, 6, 0, 0],
           [12, 5, 1, 0],
           [17, 2, 1, 0]])

# 資料正規化與預測資料
data = data - mean              # data 減掉平均數
data = data/std                 # data 除以標準差
tem = model.predict(data)       # 得出預測值
tem = tem * max(label)          # 還原標籤資料
print(tem)                      # 顯示標籤資料
```

HomeWork

回歸分析：學生身高體重分析

(請省去建立資料比較圖，只查看損失值即可)

繳交 Word 檔：

1. 原始數據資料，與你的程式碼
2. loss 與 val_loss 損失值比較圖
3. 預測自訂資料的結果
4. Email to：phd9512@cs.nchu.edu.tw

迴歸問題

某一班學生的身高和體重是否相關?

能否用身高來推測某位學生的體重?

身高	體重
150	40
152	48
155	45
158	50
160	55
162	56
165	58
170	59
172	62
175	65
180	68
185	72