

Python Develop More

Enos Chou

Before Python

Why Python?

TIOBE Index

<https://www.tiobe.com/> 點擊 TIOBE-index

PYPL Index

<http://pypl.github.io/>

TIOBE vs PYPL Index

Python Territory 應用領域

資料科學
人工智慧

後端

前端

AI & Data Science

AIoT

Crawler

IoT & Edge

API

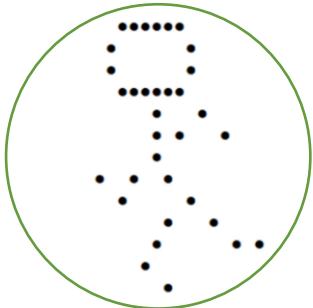
Windows APP

Desktop GUI APP

Web

LINE Bot

Python Advance Demo



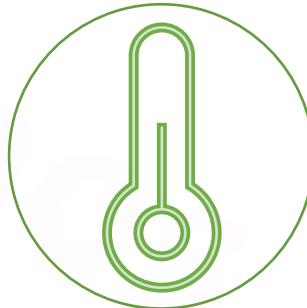
LINE Bot



Analysis



AI



Edge



IoT



AIoT

Python 的世界

Python Preparation

Python Basics

Why Python?

Python Setup

Jupyter Notebook

Python Reference

Variable, Constant, Operator

Python Built-in Data Types

Python Operators

Python Casting

Python Built-in Functions

Python Coding Style

Python Flow Control

Programming Logic

Extreme Search

Sort Algorithms

Recursive Logic

Fault Tolerance Design

Robust Design

Python More

Python Class

Python Exception

Python Application

Python Packages Management

requests

JSON

Beautiful Soup

Jieba

Python Engineering

wordcloud

Matplotlib

NumPy

pandas

Flask

Python Crawler

Python Desktop

Python Executable

Python Monitor

Python Web

Python API

Python Delivery

本課程涵蓋範圍

Python Preparation

Python Basics

Why Python?

Variable, Constant, Operator

Python Setup

Python Built-in Data Types

Jupyter Notebook

Python Operators

Python Reference

Python Casting

Python Built-in Functions

Python Coding Style

Python Flow Control

Programming Logic

Extreme Search

Sort Algorithms

Recursive Logic

Fault Tolerance Design

Robust Design

Python Function

Python Variable Scope

Python More

Python Class

Python Exception

Python Application

Python Packages Management

requests

JSON

Beautiful Soup

Jieba

Python Engineering

wordcloud

Matplotlib

NumPy

pandas

Flask

Python Crawler

Python Desktop

Python Executable

Python Monitor

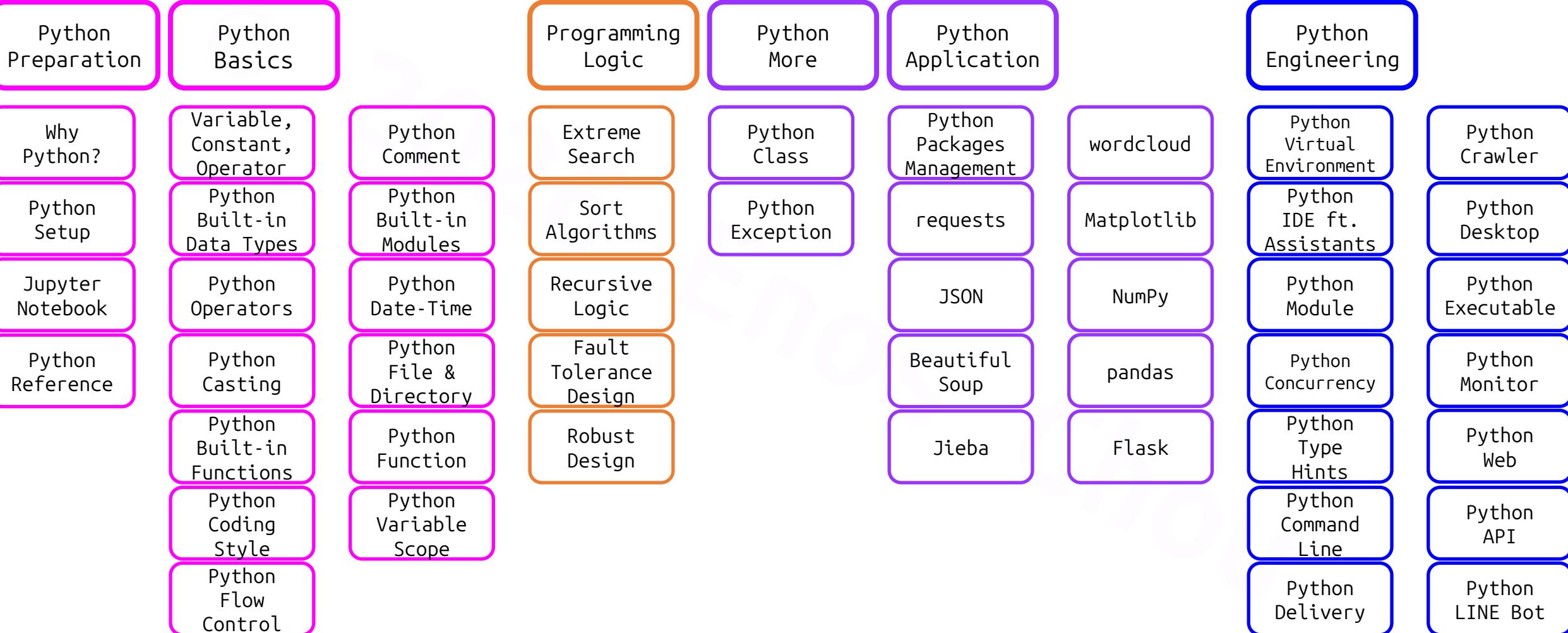
Python Type Hints

Python Command Line

Python Delivery

Python LINE Bot

工程師



數據分析師/ AI 應用

Python Preparation

Python Basics

Why Python?

Variable, Constant, Operator

Python Setup

Python Built-in Data Types

Jupyter Notebook

Python Operators

Python Reference

Python Casting

Python Built-in Functions

Python Coding Style

Python Flow Control

Programming Logic

Extreme Search

Sort Algorithms

Recursive Logic

Fault Tolerance Design

Robust Design

Python More

Python Class

Python Exception

Python Application

Python Packages Management

requests

JSON

Beautiful Soup

Jieba

Python Engineering

Python Virtual Environment

Python IDE ft. Assistants

Python Module

Python Concurrency

Python Type Hints

Python Command Line

Python API

Python LINE Bot

Python Preparation

Python Basics

Why Python?

Variable, Constant, Operator

Python Setup

Python Built-in Data Types

Jupyter Notebook

Python Operators

Python Reference

Python Casting

Python Built-in Functions

Python Coding Style

Python Flow Control

Programming Logic

Extreme Search

Sort Algorithms

Recursive Logic

Fault Tolerance Design

Robust Design

Python More

Python Class

Python Exception

Python Application

Python Packages Management

requests

JSON

Beautiful Soup

Jieba

Python Engineering

wordcloud

Matplotlib

NumPy

pandas

Flask

Python Virtual Environment

Python IDE ft. Assistants

Python Module

Python Concurrency

Python Type Hints

Python Command Line

Python Delivery

Python Crawler

Python Desktop

Python Executable

Python Monitor

Python Web

Python API

Python LINE Bot

學習關鍵

課前預習

授課時有 **理解** 的餘裕

課後練習

以每週 6 小時以上的練習量
持續累積程式修為

質疑每行程式碼

建立 **自我成長** 的模式

Python Reference

Python 教學網站

<https://www.programiz.com/> 點擊 Python Programming

Python Documentation

<https://docs.python.org/> 選擇 3.12

Python Coding Style

<https://peps.python.org/> 選擇 PEP 8

演算法程式設計 in Python

<https://leetcode.com/>

大學程式設計先修檢測 APCS 考古題

<https://apcs.csie.ntnu.edu.tw/> 點擊 試題資訊 > 題目範例 > 初級題本範例

Pre-Python Environment

Why Python, Python Setup, Jupyter Notebook, Python Reference

Python Development Concept 開發觀念

Development 開發 vs Deployment 部署

開發 OS : Windows 10, 11
部署 OS : any (Linux-based,
 Windows, ...)
部署 HW : \$

IDE 整合開發環境

Python IDLE
Jupyter Notebook
Visual Studio Code
PyCharm
Sublime Text
Spyder
Thonny
Google Colab
*Anaconda (Miniconda)

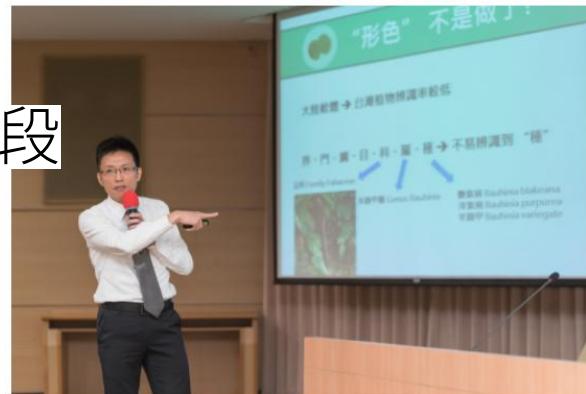
Python IDLE vs Jupyter Notebook

Jupyter Notebook

File Edit View Insert Cell Kernel Widgets Help

Why Jupyter Notebook

1. Interactive 交談式互動
2. 可任意於任何位置重新執行程式碼片段
3. 易於包裝華麗註解



Python Practice

Python IDLE

Python 3.6.8 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, Type "help", "copyright", "credits" or "license()")  
>>> a = 100  
>>> b = 250  
>>> c = a + b  
>>> print(c)  
350  
>>> |
```

Pre-Python

In [1]: 1 a = 100

In [2]: 1 b = 250

In [3]: 1 c = a + b

In [4]: 1 print(c)

350

Python Environment Setup

1. 安裝 Miniconda
2. 建立 Python 環境並安裝 Jupyter Notebook

Note

1. 不適用於 Windows 7 及以下
2. 作業系統之用戶帳號不得為中文
3. 用戶帳號須擁有系統管理員權限

Python Environment Setup

1. 安裝 Miniconda

a. 下載 Miniconda

- ① 進入 Anaconda 官網 anaconda.com
- ② 點擊  [Free Download](#)
- ③ 滾到底，點擊 [Download Miniconda Installer >](#)
- ④ 依作業系統 (Windows/ Mac/ Linux) 下載 Miniconda Installer
- ⑤ 取得下載檔案

Python Environment Setup

1. 安裝 Miniconda

b. 安裝 Miniconda on Windows

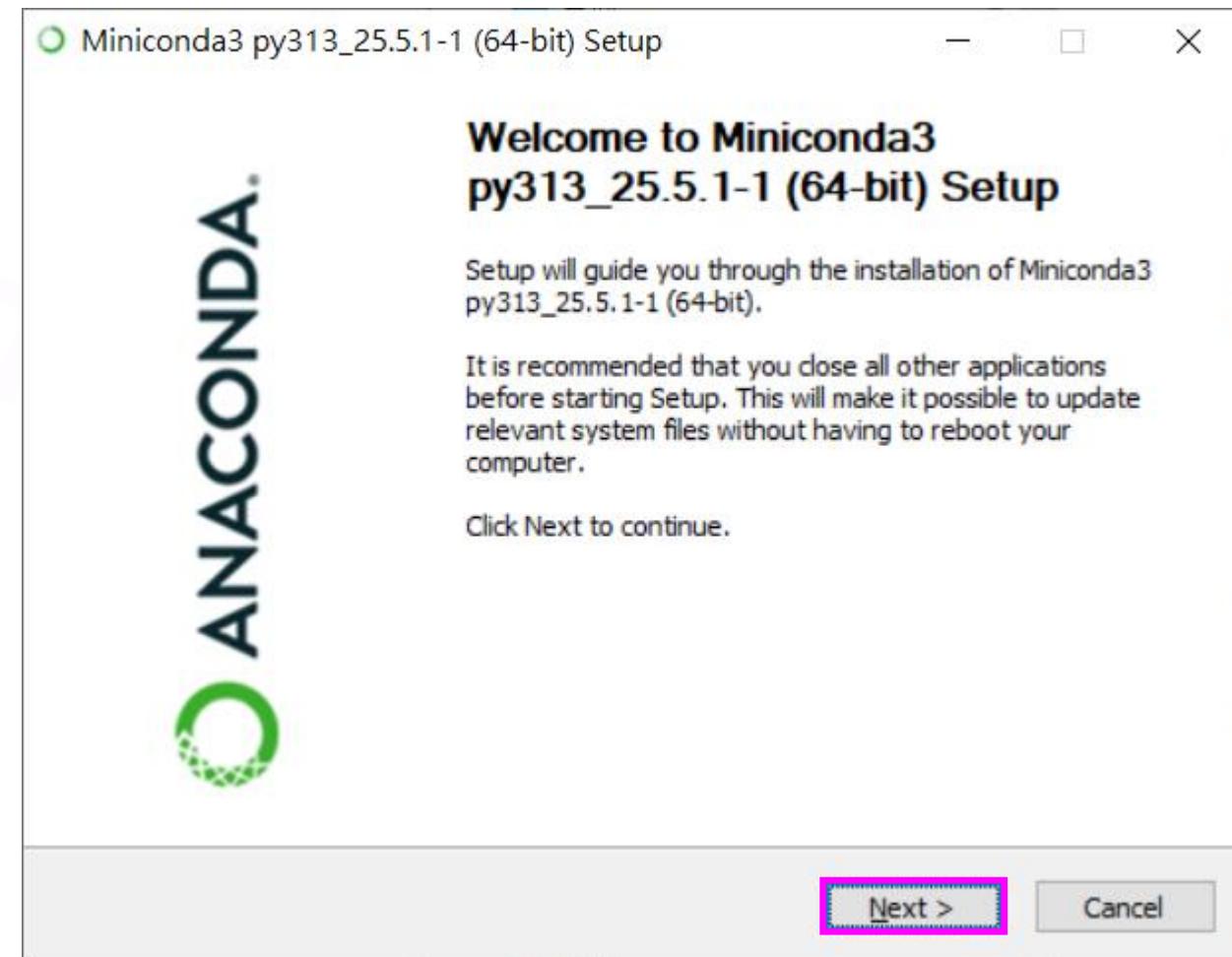
- ① 點擊下載檔案

Python Environment Setup

1. 安裝 Miniconda

b. 安裝 Miniconda on Windows

- ① 點擊下載檔案
- ② Next >

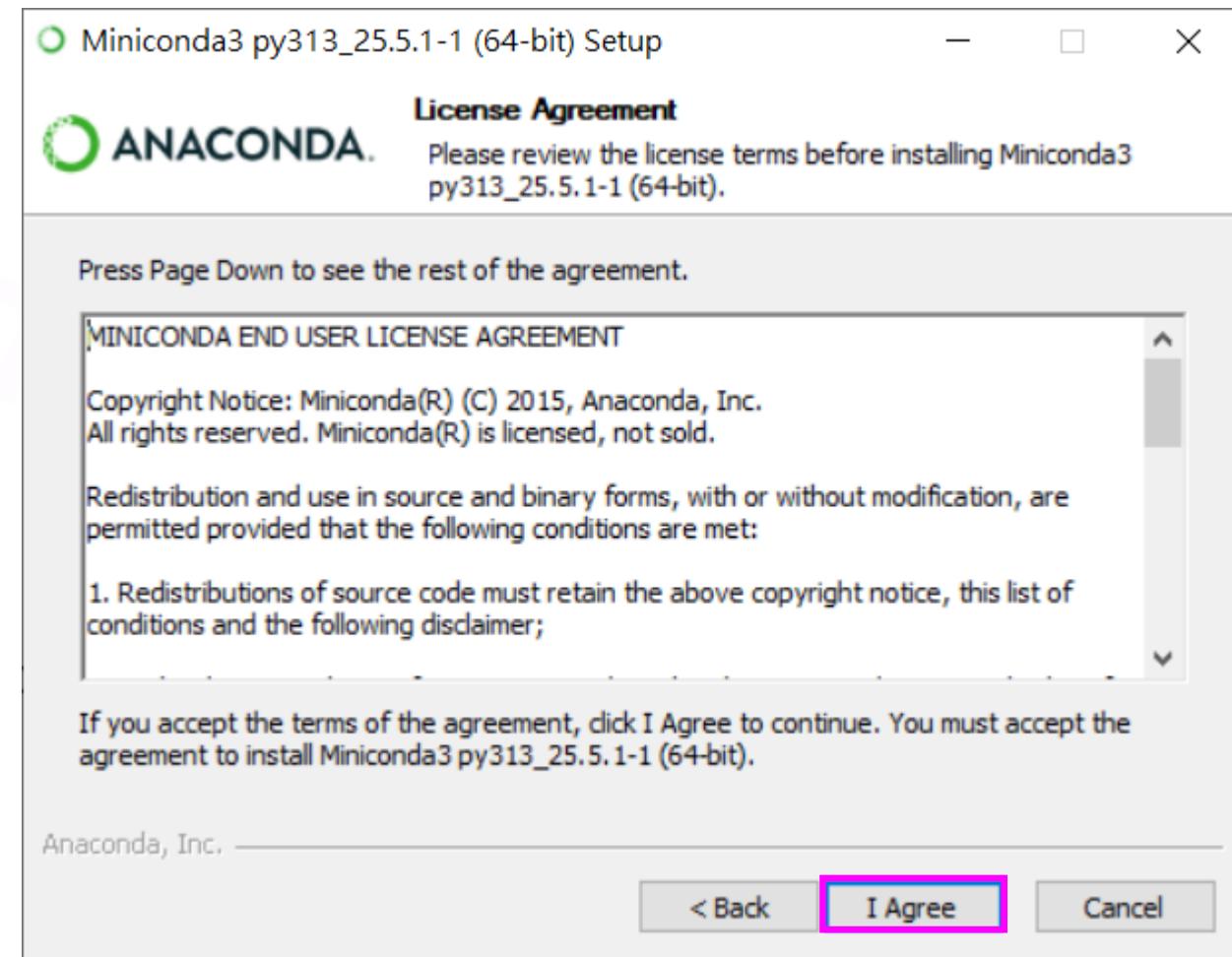


Python Environment Setup

1. 安裝 Miniconda

b. 安裝 Miniconda on Windows

- ① 點擊下載檔案
- ② Next >
- ③ I Agree

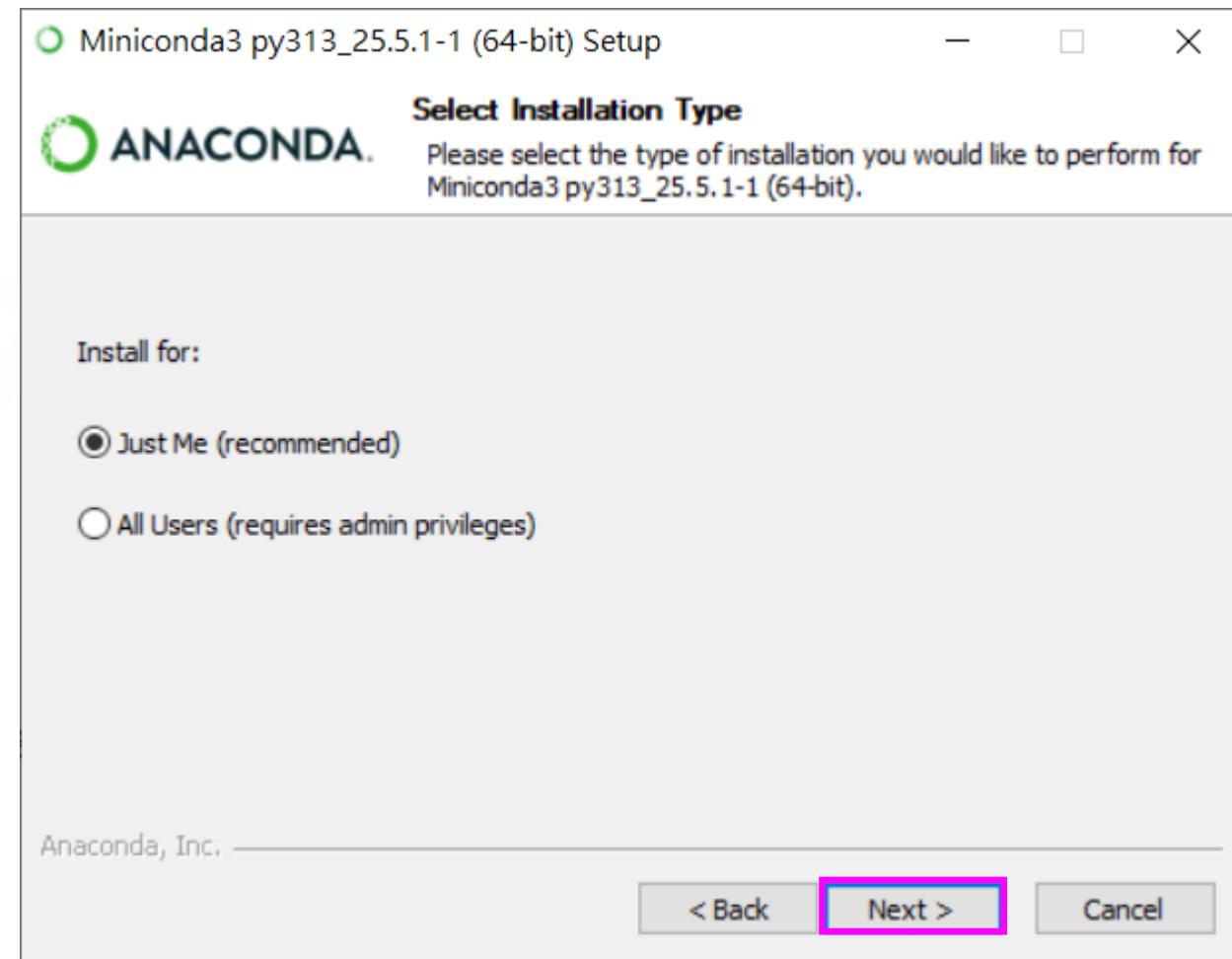


Python Environment Setup

1. 安裝 Miniconda

b. 安裝 Miniconda on Windows

- ① 點擊下載檔案
- ② Next >
- ③ I Agree
- ④ Next >



Python Environment Setup

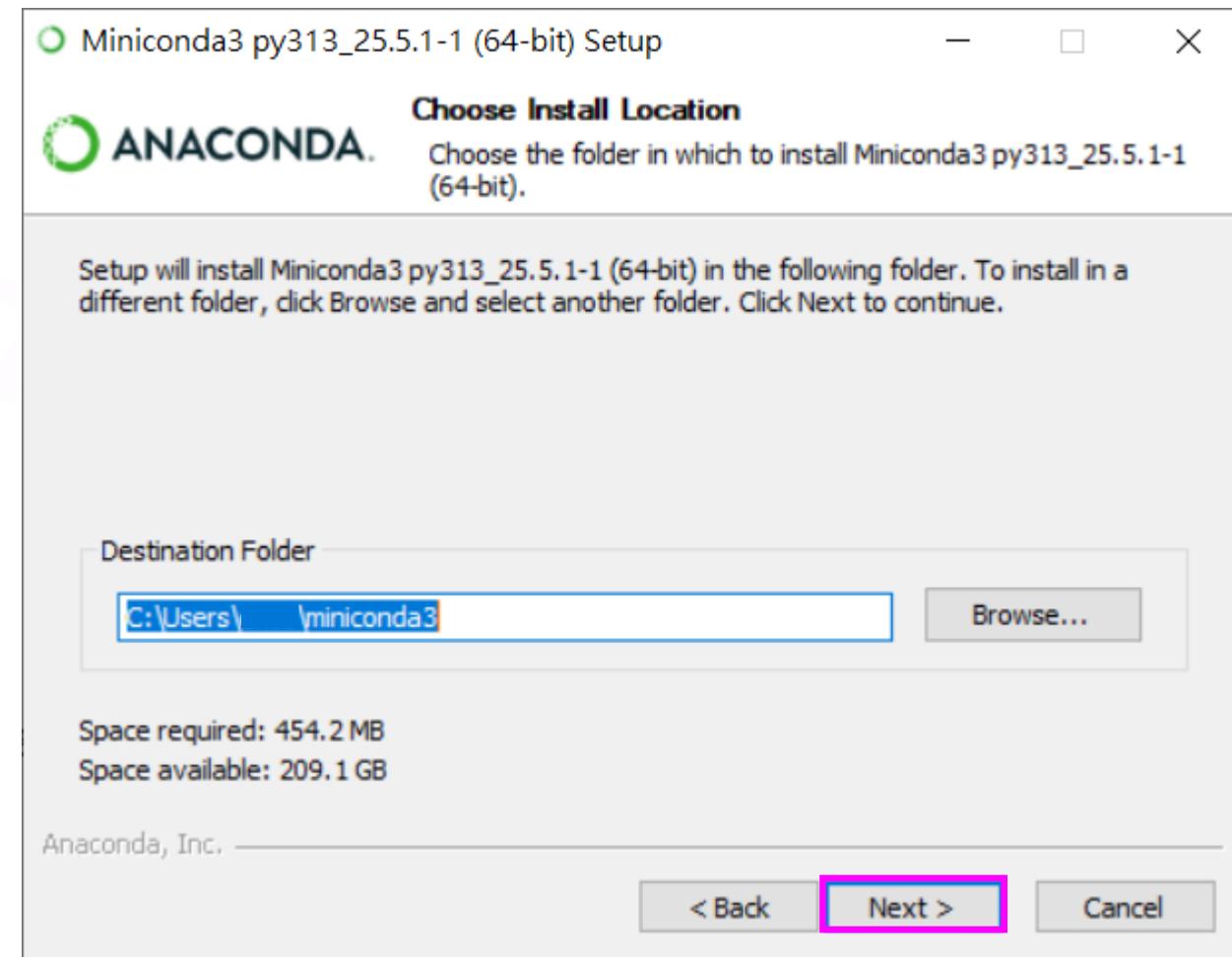
1. 安裝 Miniconda

b. 安裝 Miniconda on Windows

- ① 點擊下載檔案
- ② Next >
- ③ I Agree
- ④ Next >
- ⑤ Next >

Note

不建議調整安裝路徑

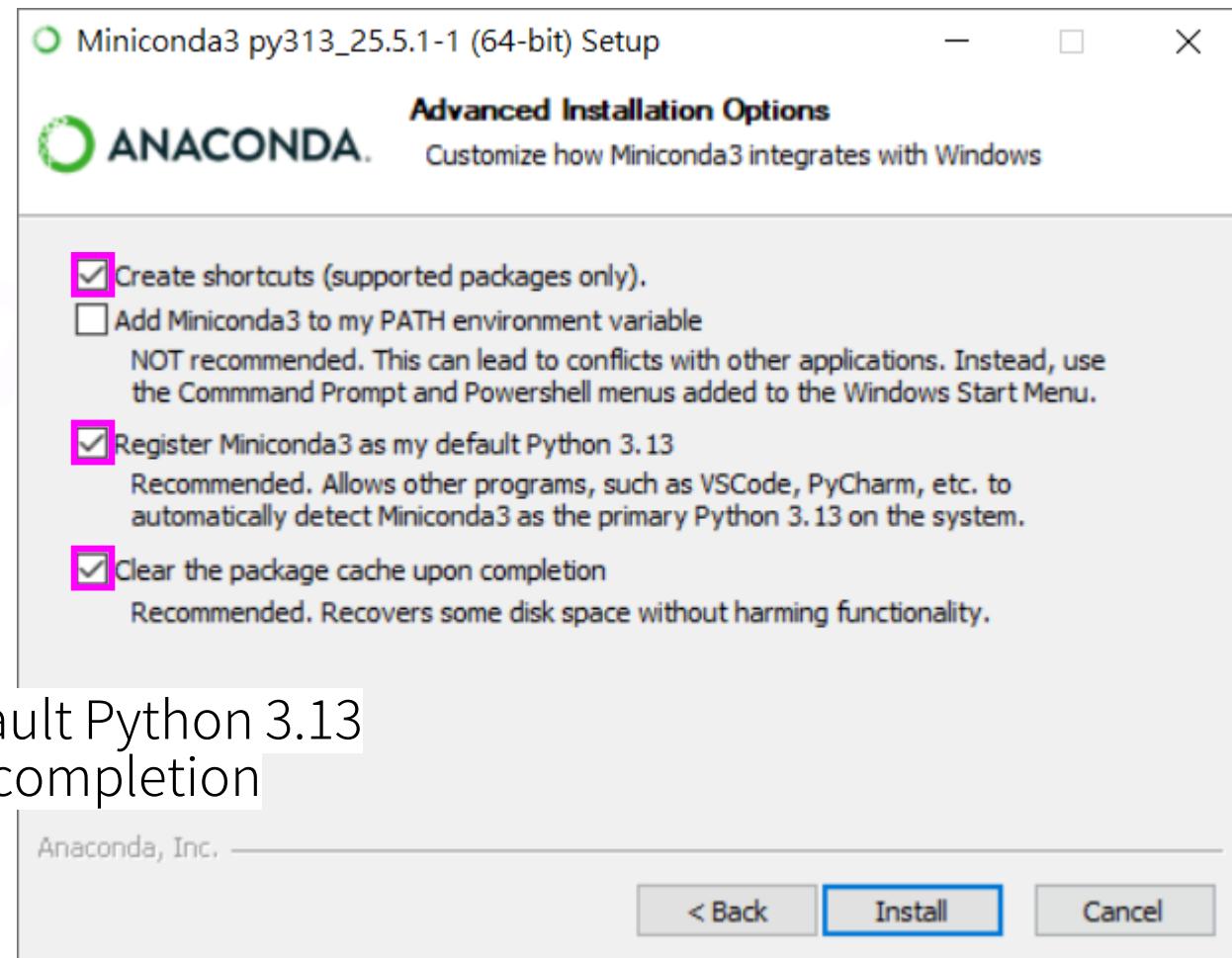


Python Environment Setup

1. 安裝 Miniconda

b. 安裝 Miniconda on Windows

- ① 點擊下載檔案
- ② Next >
- ③ I Agree
- ④ Next >
- ⑤ Next >
- ⑥ Register Miniconda3 as my default Python 3.13
 Clear the package cache upon completion

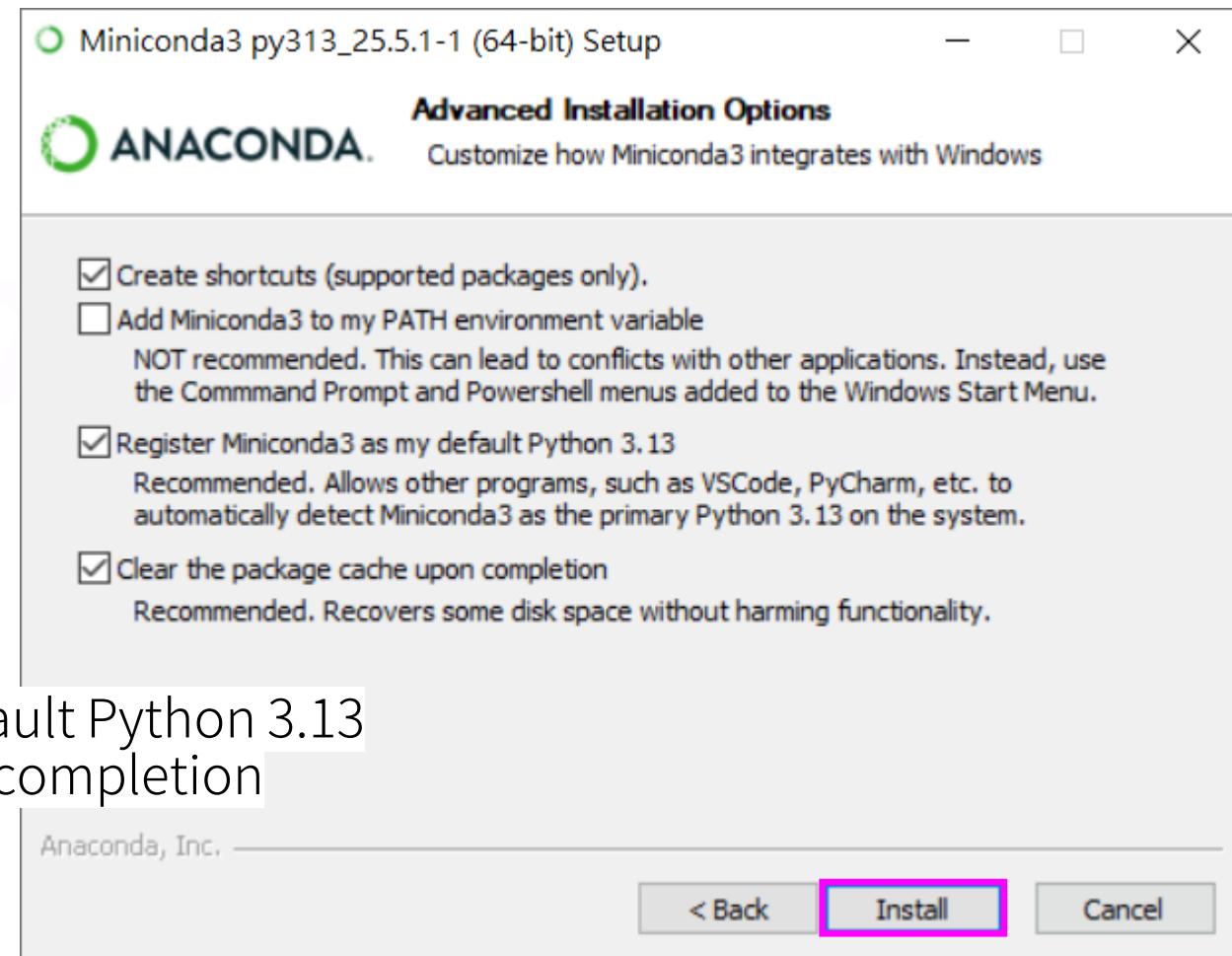


Python Environment Setup

1. 安裝 Miniconda

b. 安裝 Miniconda on Windows

- ① 點擊下載檔案
- ② Next >
- ③ I Agree
- ④ Next >
- ⑤ Next >
- ⑥ Register Miniconda3 as my default Python 3.13
 Clear the package cache upon completion
- ⑦ Install

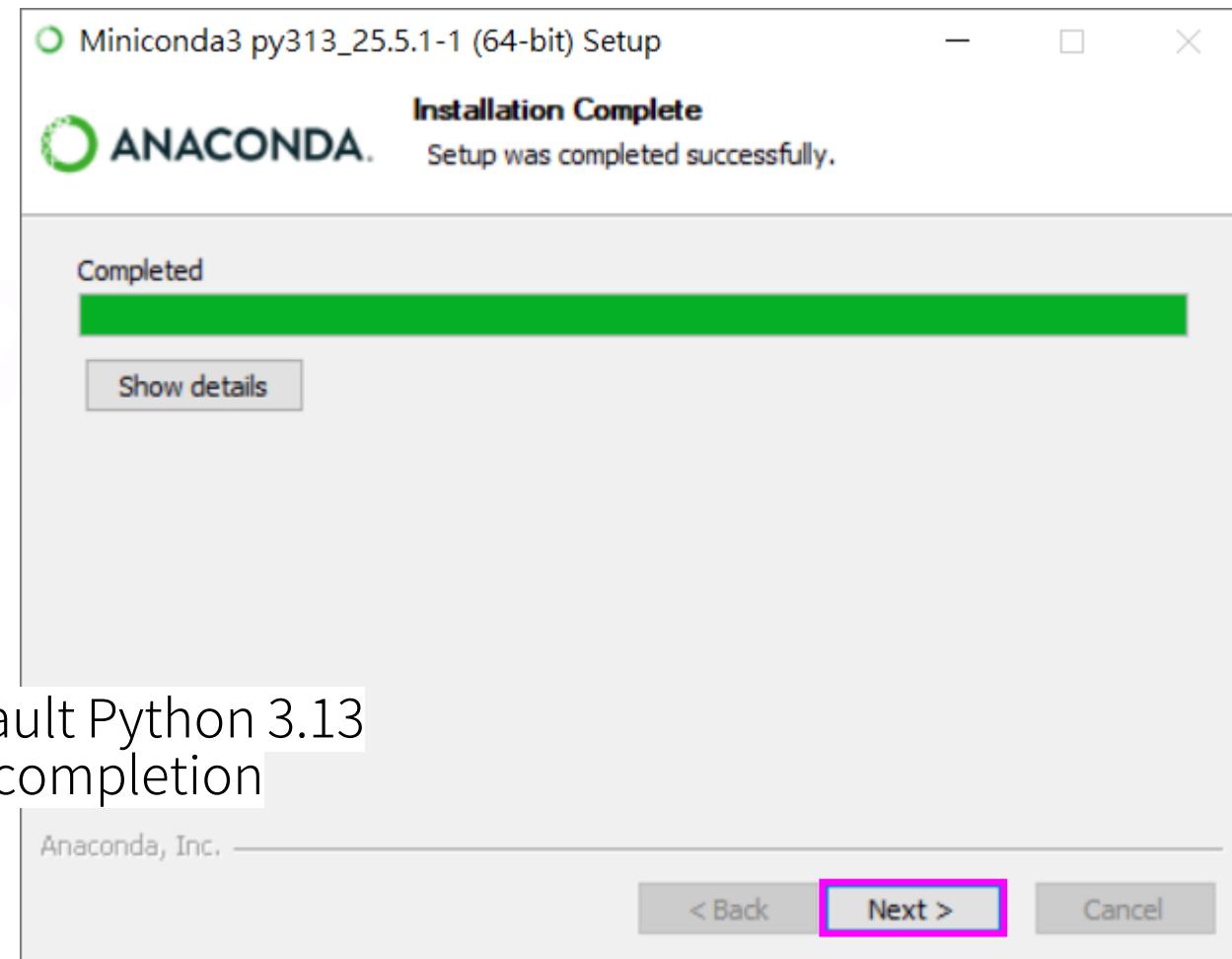


Python Environment Setup

1. 安裝 Miniconda

b. 安裝 Miniconda on Windows

- ① 點擊下載檔案
- ② Next >
- ③ I Agree
- ④ Next >
- ⑤ Next >
- ⑥ Register Miniconda3 as my default Python 3.13
 Clear the package cache upon completion
- ⑦ Install
- ⑧ Next >

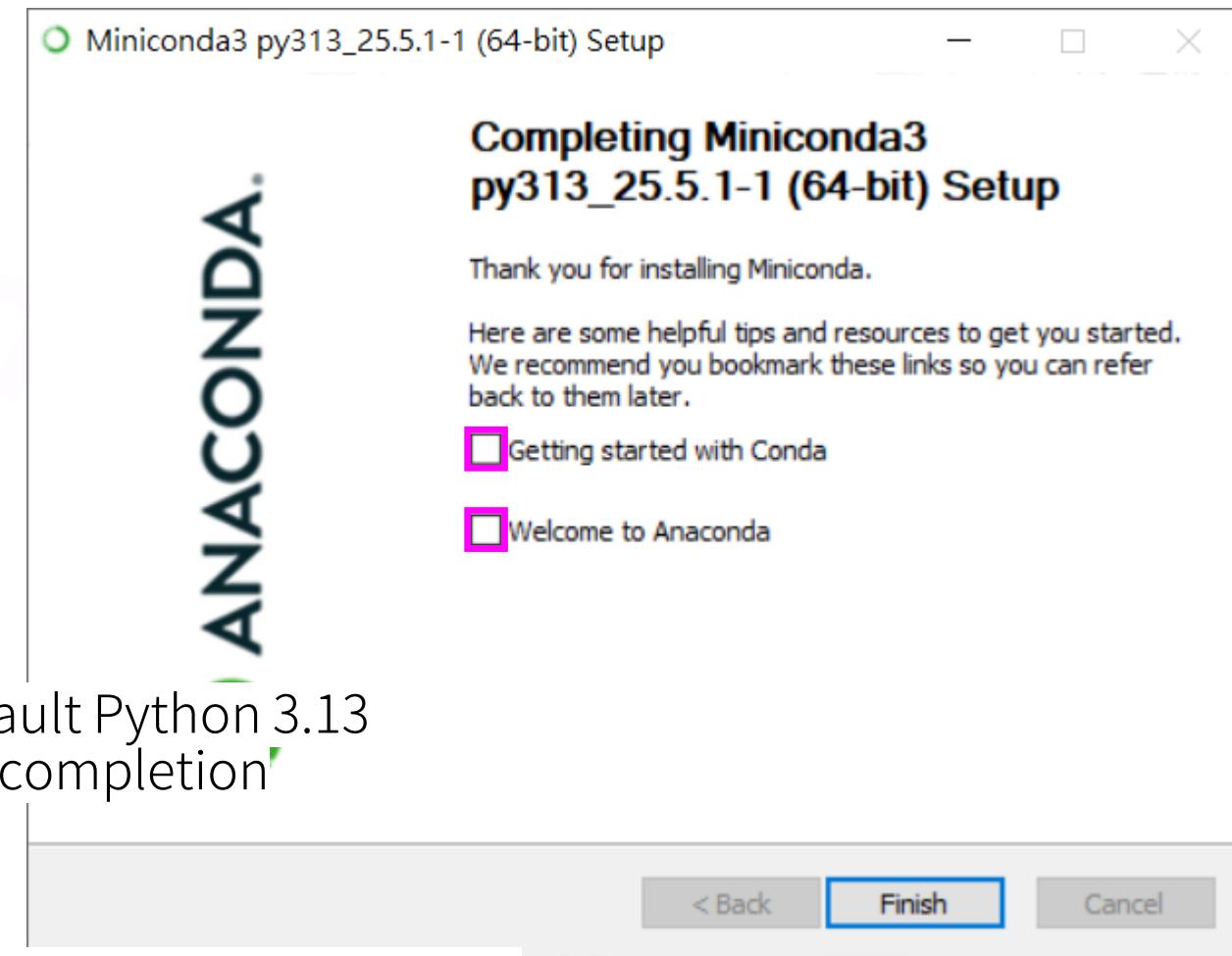


Python Environment Setup

1. 安裝 Miniconda

b. 安裝 Miniconda on Windows

- ① 點擊下載檔案
- ② Next >
- ③ I Agree
- ④ Next >
- ⑤ Next >
- ⑥ Register Miniconda3 as my default Python 3.13
 Clear the package cache upon completion
- ⑦ Install
- ⑧ Next >
- ⑨ Getting started with Conda Welcome to Anaconda

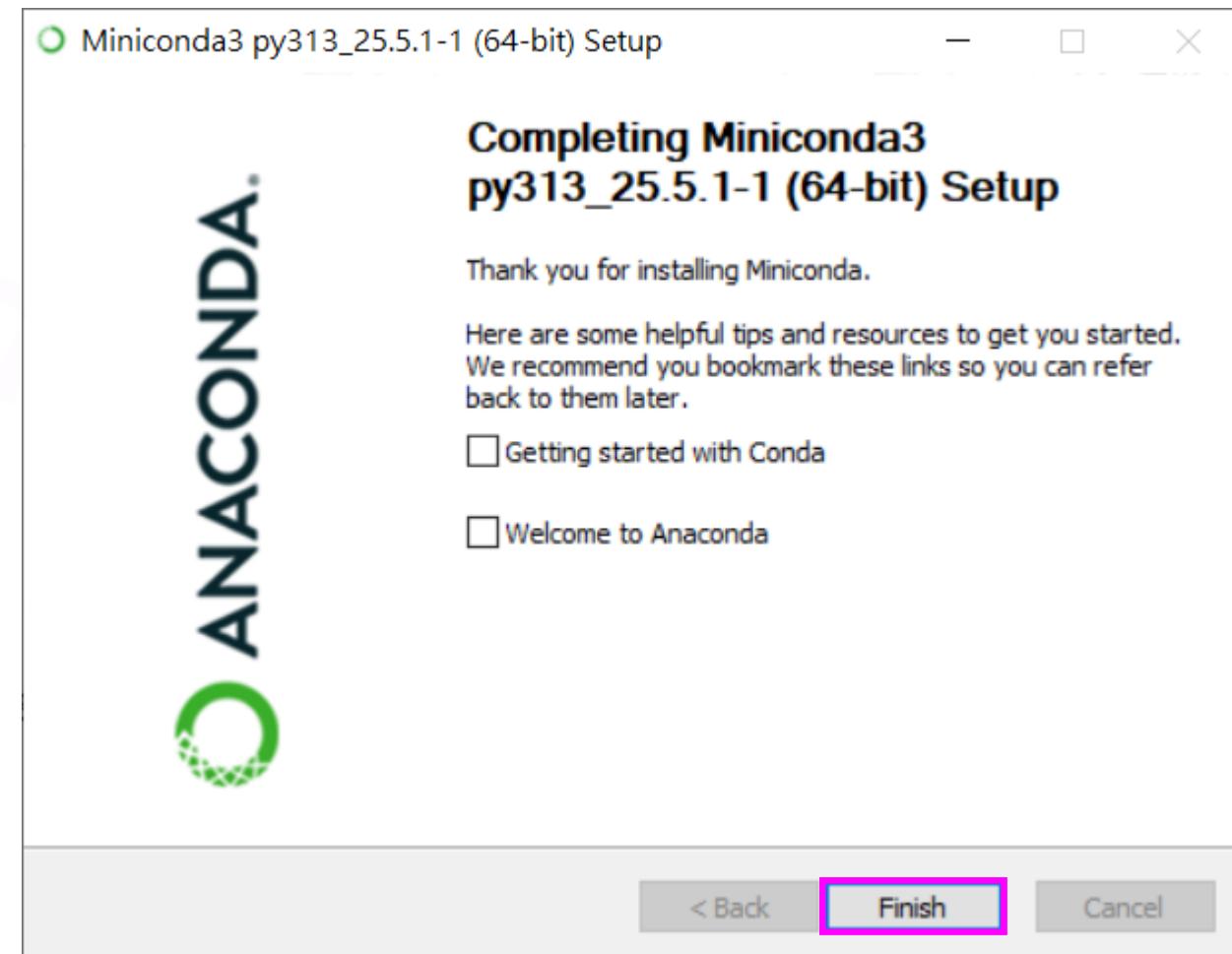


Python Environment Setup

1. 安裝 Miniconda

b. 安裝 Miniconda on Windows

⑩ Finish

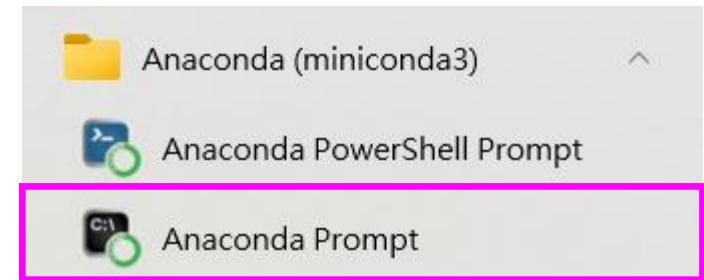


Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

a. 啟動 Miniconda on Windows

Anaconda Prompt (miniconda3)

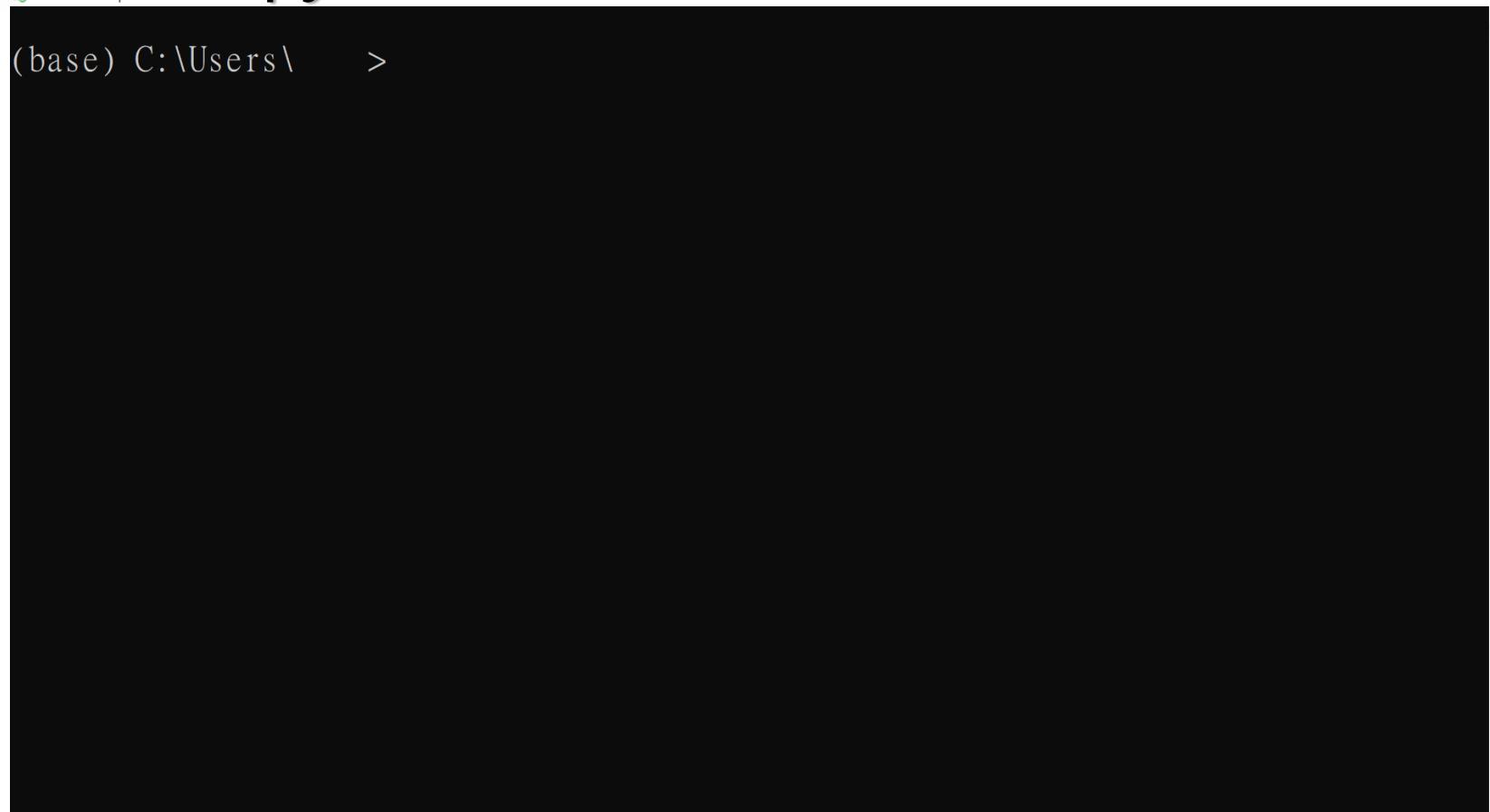


Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

a. 啟動 Miniconda

Anaconda Prompt



The image shows a screenshot of the Anaconda Prompt application window. The title bar reads "Anaconda Prompt". The main area is a black terminal window with white text. At the top left, it says "(base) C:\Users\<username> >". The rest of the window is completely blank, indicating no input or output has been entered.

Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

b. 建立 Python 環境並安裝 Jupyter Notebook

```
conda create -n your_env python=python_ver notebook  
conda create -n pydm python=3.12 notebook
```

Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

b. 建立 Python 環境並安裝 Jupyter Notebook

```
conda create -n your_env python=python_ver notebook
```

```
conda create -n pydm python=3.12 notebook
```

Anaconda Prompt - conda create -n pydm python=3.12 notebook

Package	Version	Hash
tinycc2	1.4.0	py312haa95532_0
tk	8.6.14	h0416ee5_0
tornado	6.4.2	py312h827c3e9_0
traitlets	5.14.3	py312haa95532_0
typing-extensions	4.12.2	py312haa95532_0
typing_extensions	4.12.2	py312haa95532_0
vc	14.42	h04d1e81_0
vs2015_runtime	14.42.34433	he0abc0d_4
wincertstore	0.2.5	pyhd3eb1b0_0
wcwidth	0.2.5	pyhd3eb1b0_0
webencodings	0.5.1	py312haa95532_2
websocket-client	1.8.0	py312haa95532_0
wheel	0.45.1	py312haa95532_0
win_inet_pton	1.1.0	py312haa95532_0
winpty	0.4.3-4	py312haa95532_0
xz	5.6.4	h4754444_1
yaml	0.2.5	he774522_0
zeromq	4.3.5	hd77b12b_0
zlib	1.2.13	h8cc25b3_1

Proceed ([y]/n)?

Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

b. 建立 Python 環境

```
Anaconda Prompt
done
#
# To activate this environment, use
#     $ conda activate pydm
#
conda create -n yd
# To deactivate an active environment, use
#
conda create -n p
#     $ conda deactivate

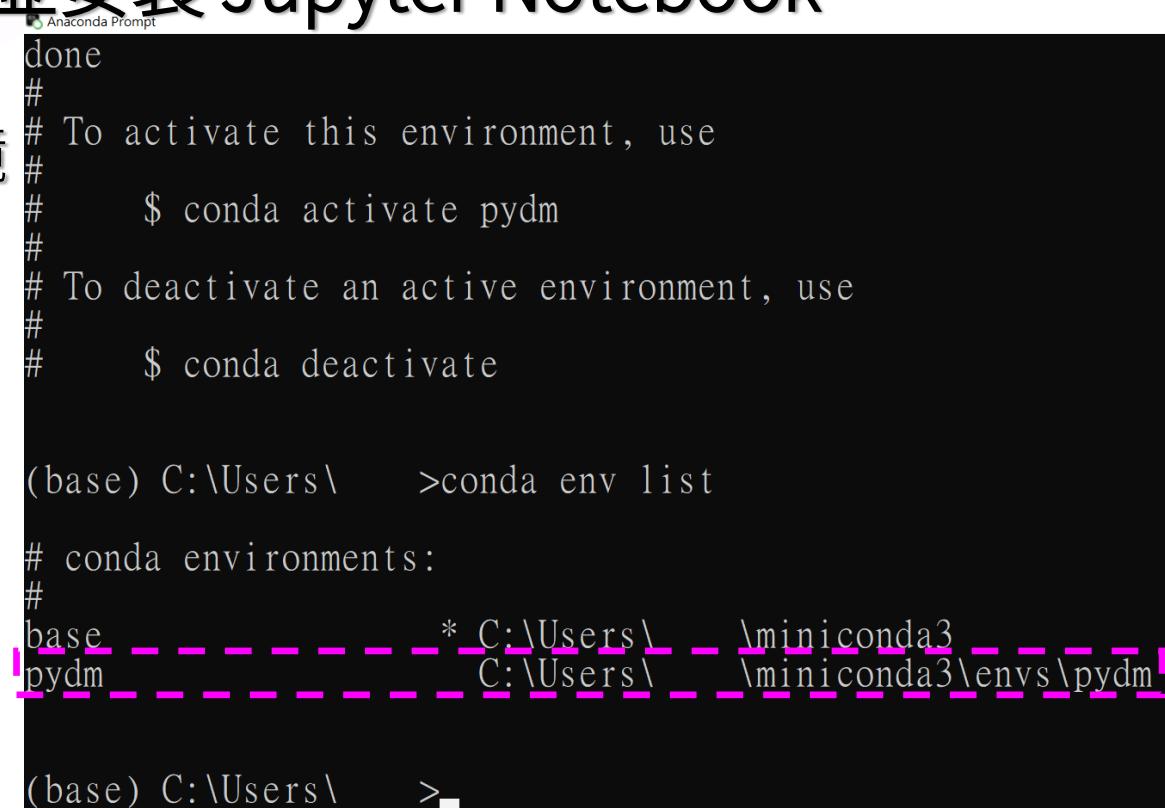
(base) C:\Users\    >_
```

Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

c. 條列 Python 環境

`conda env list`



```
done
#
# To activate this environment, use
#
#     $ conda activate pydm
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:\Users\    >conda env list

# conda environments:
#
base          * C:\Users\_\_\_miniconda3
pydm          C:\Users\_\_\_miniconda3\envs\pydm

(base) C:\Users\    >
```

Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

d. 啟動 Python 環境

```
Anaconda Prompt
done
#
# To activate this environment, use
#
#     $ conda activate pydm
#
conda activate your_env          # activate an active environment, use
conda deactivate                  # conda deactivate

(base) C:\Users\      >conda env list

# conda environments:
#
base                      * C:\Users\      \miniconda3
pydm                      C:\Users\      \miniconda3\envs\pydm

(base) C:\Users\      >conda activate pydm
[pydm] C:\Users\      >
```

Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

e. (關閉 Python 環境)

```
conda deactivate
```

```
conda deactivate
```

Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

f. (刪除 Python 環境)

```
conda remove -n your_env --all  
conda remove -n pydm --all
```

Note

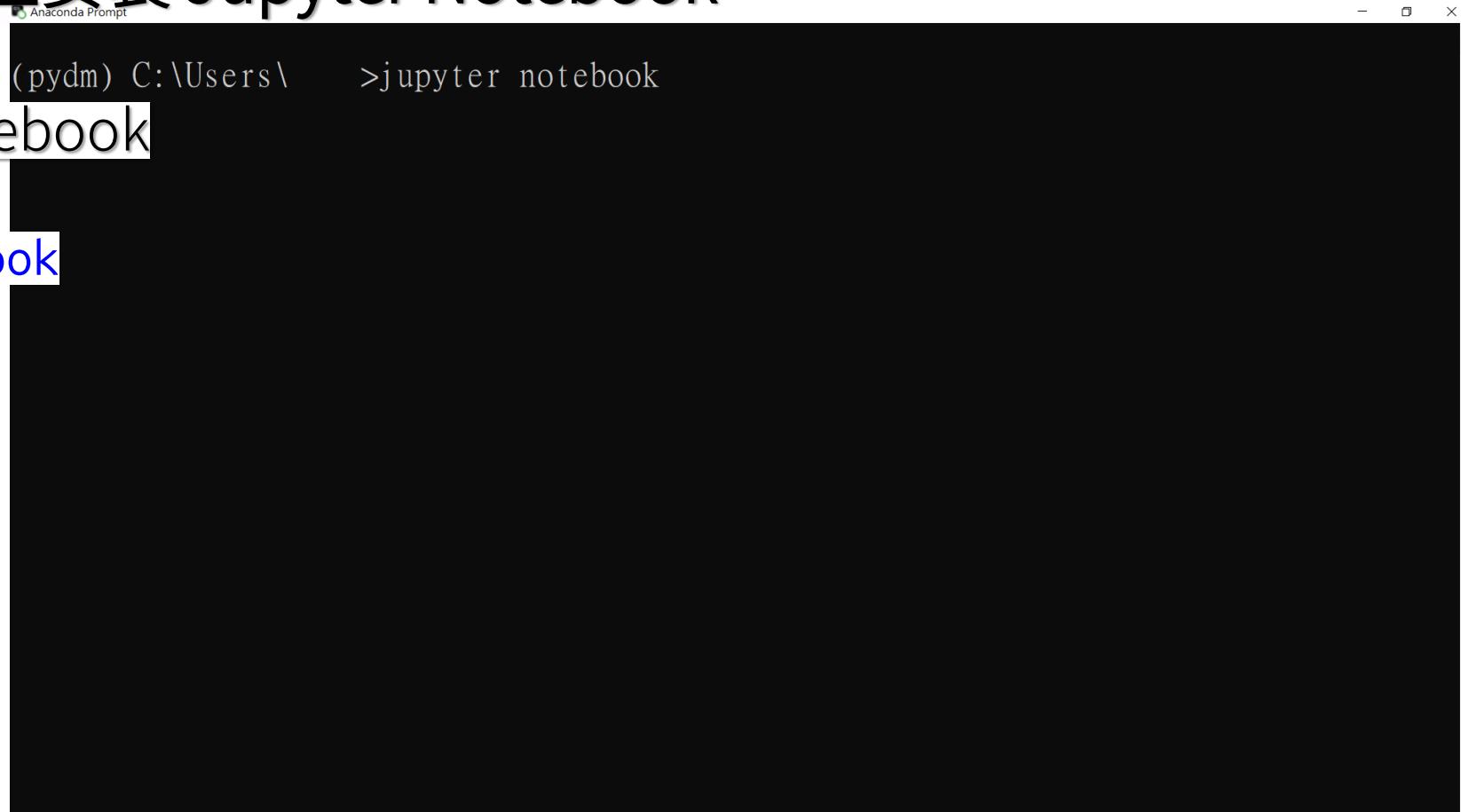
會遺留以環境名稱為名的空資料夾，可手動移除

Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

g. 啟動 Jupyter Notebook

① jupyter notebook



Anaconda Prompt
(pydm) C:\Users\>jupyter notebook

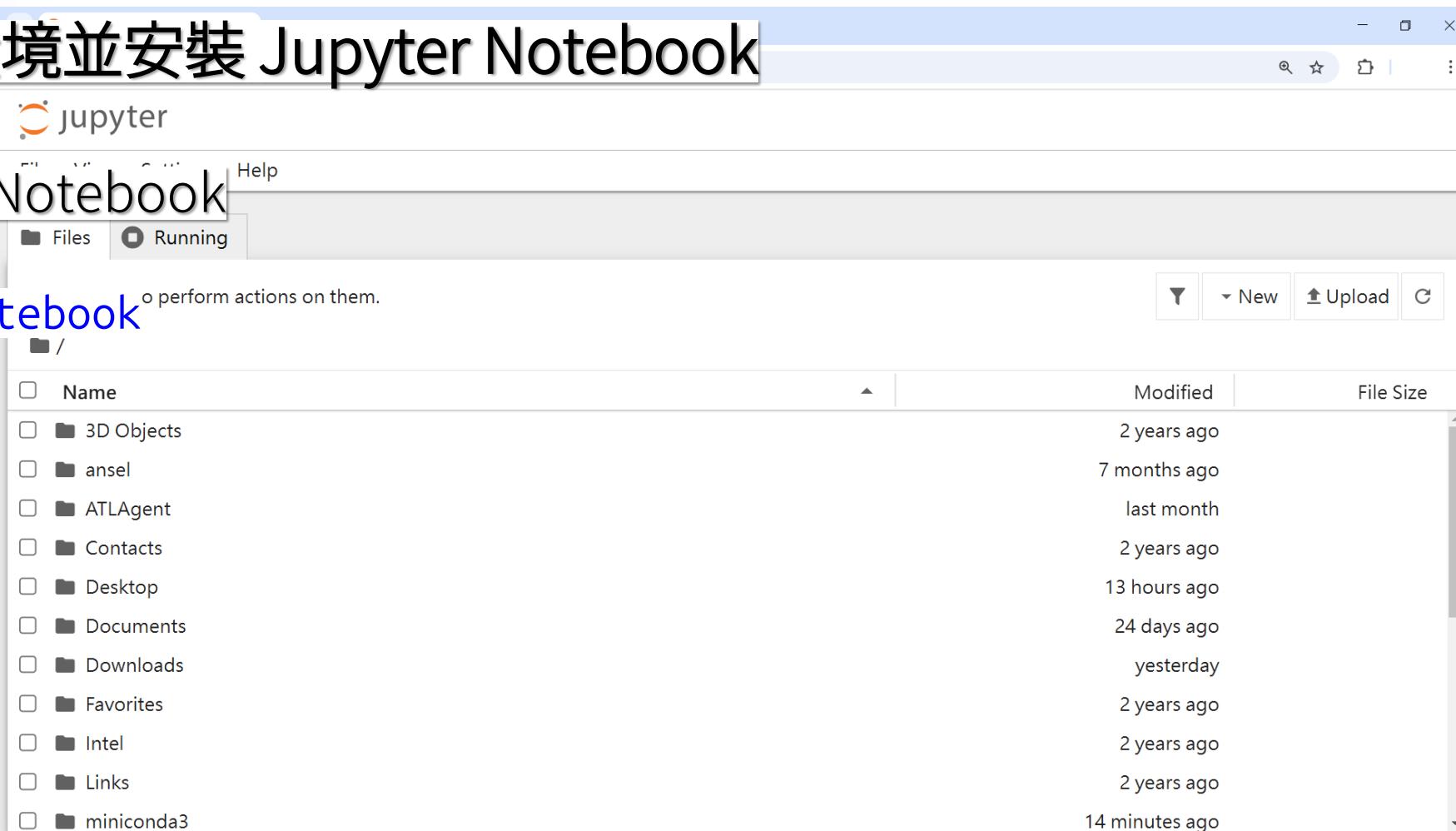
A screenshot of the Anaconda Prompt window. The title bar says "Anaconda Prompt". The command "(pydm) C:\Users\>jupyter notebook" is typed in the prompt area. The rest of the window is blacked out.

Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

g. 啟動 Jupyter Notebook

① jupyter notebook



Python Environment Setup

2. 建立 Python 環境並安裝 Jupyter Notebook

g. 啟動 Jupyter Notebook

```
[I 2025-02-14 23:47:39.624 ServerApp] http://127.0.0.1:8888/tree?token=2c554046e83316f1a005c14f32aae28f3c1368bad0154e7f
[I 2025-02-14 23:47:39.625 ServerApp] Use Control-C to stop this server and shut down kernels (twice to skip confirmation).
[C 2025-02-14 23:47:39.700 ServerApp]
```

- ① [jupyter notebook](#) To access the server, open this file in a browser:
file:///C:/Users/ /AppData/Roaming/jupyter/runtime/jpserver-10764-open.
- ② 亦可將下列網址貼在瀏覽器啟動 Jupyter Notebook:

```
Or copy and paste one of these URLs:
http://localhost:8888/tree?token=2c554046e83316f1a005c14f32aae28f3c1368bad0154e7f
http://127.0.0.1:8888/tree?token=2c554046e83316f1a005c14f32aae28f3c1368bad0154e7f
[W 2025-02-14 23:47:39.750 ServerApp] Could not determine npm prefix: [WinError 2]
系統找不到指定的檔案。
```

```
[I 2025-02-14 23:47:39.965 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-language-server-bin, vscode-json-languageserver-bin, yaml-language-server
```

Python Environment Setup

3. (移除 Miniconda)

a. 移除 Monicoda

預設安裝路徑 `C:\Users\your_account\miniconda3`

- ① 設定 > 應用程式 > 應用程式與功能 > 解除安裝 `Miniconda3 py313_...` > Next > Uninstall > 是 > Next > Finish

Python Environment Setup

3. (移除 Miniconda)

a. 移除 Monicoda

預設安裝路徑 `C:\Users\your_account\miniconda3`

② 手動移除資料夾

`C:\Users\your_account\miniconda3`

`C:\Users\your_account\.conda`

`C:\Users\your_account\.jupyter`

`C:\Users\your_account\.ipython`

`C:\Users\your_account\AppData\Local\conda`

`C:\Users\your_account\AppData\Local\conda-anaconda-tos`

`C:\Users\your_account\AppData\Roaming\.anaconda`

`C:\Users\your_account\AppData\Roaming\jupyter`

Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

2. Hotkey 熱鍵

3. 環境微調

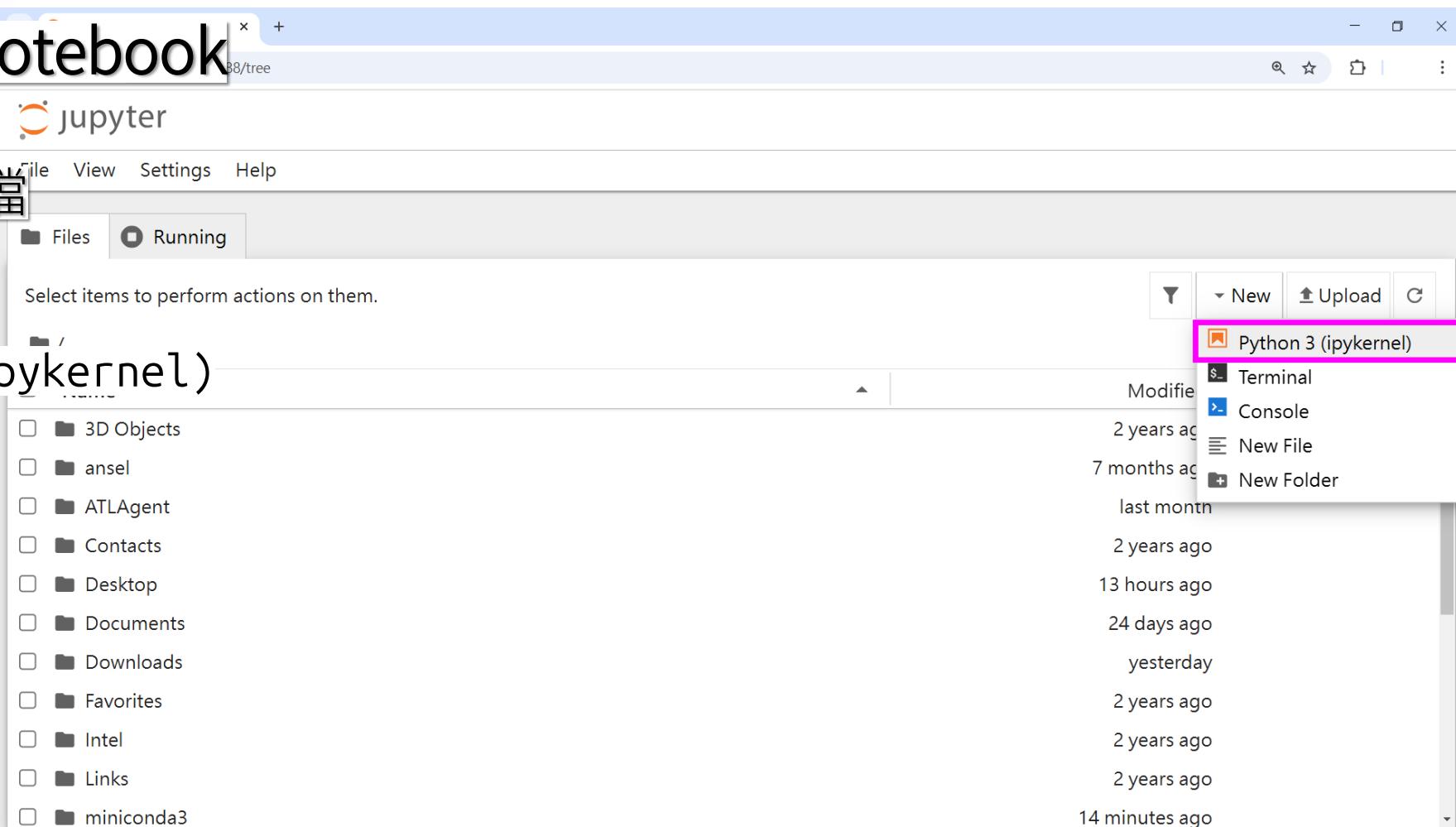
Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

a. 新建 Python 檔

① New

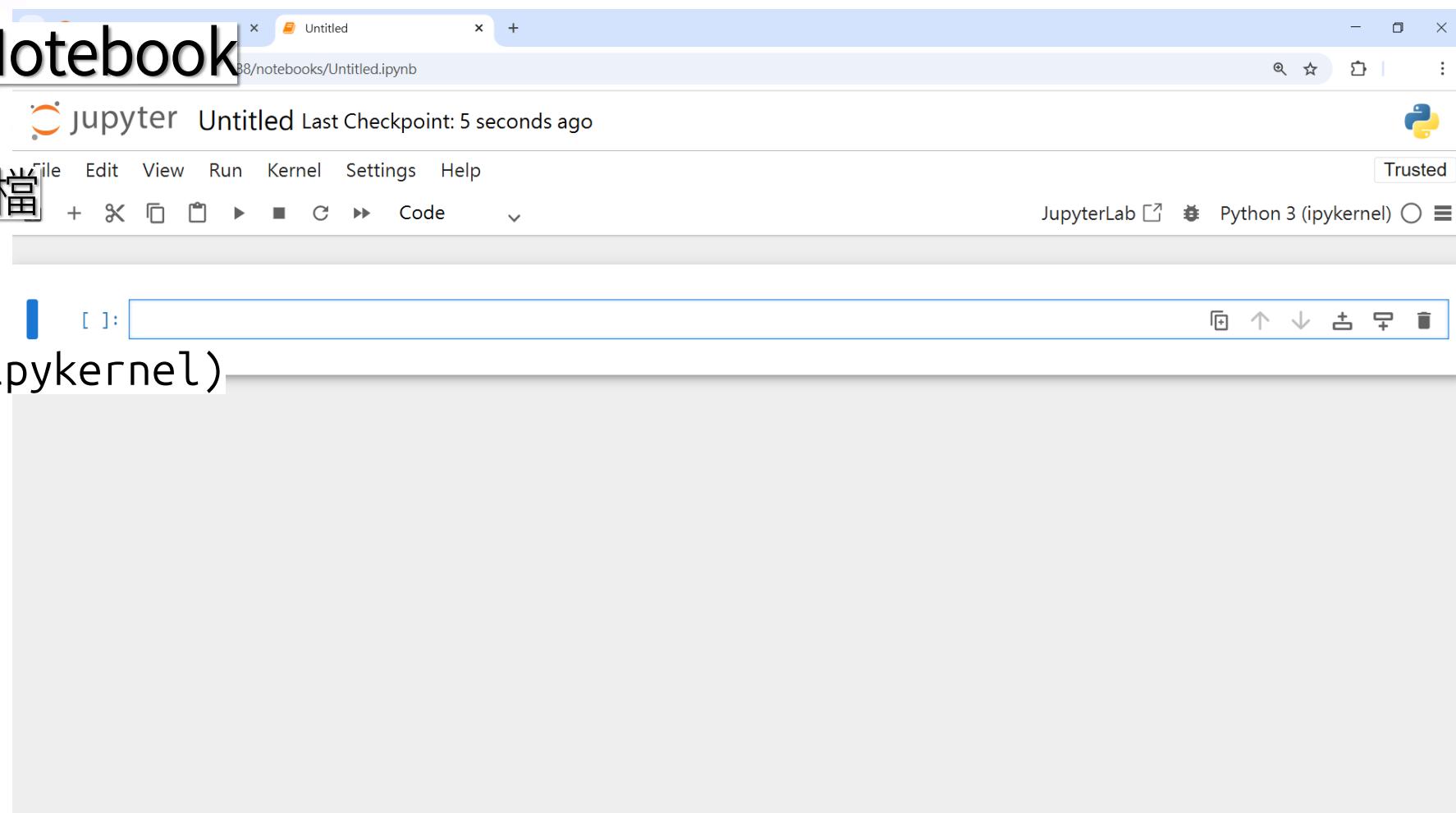
② Python3 (ipykernel)



Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

a. 新建 Python 檔



① New

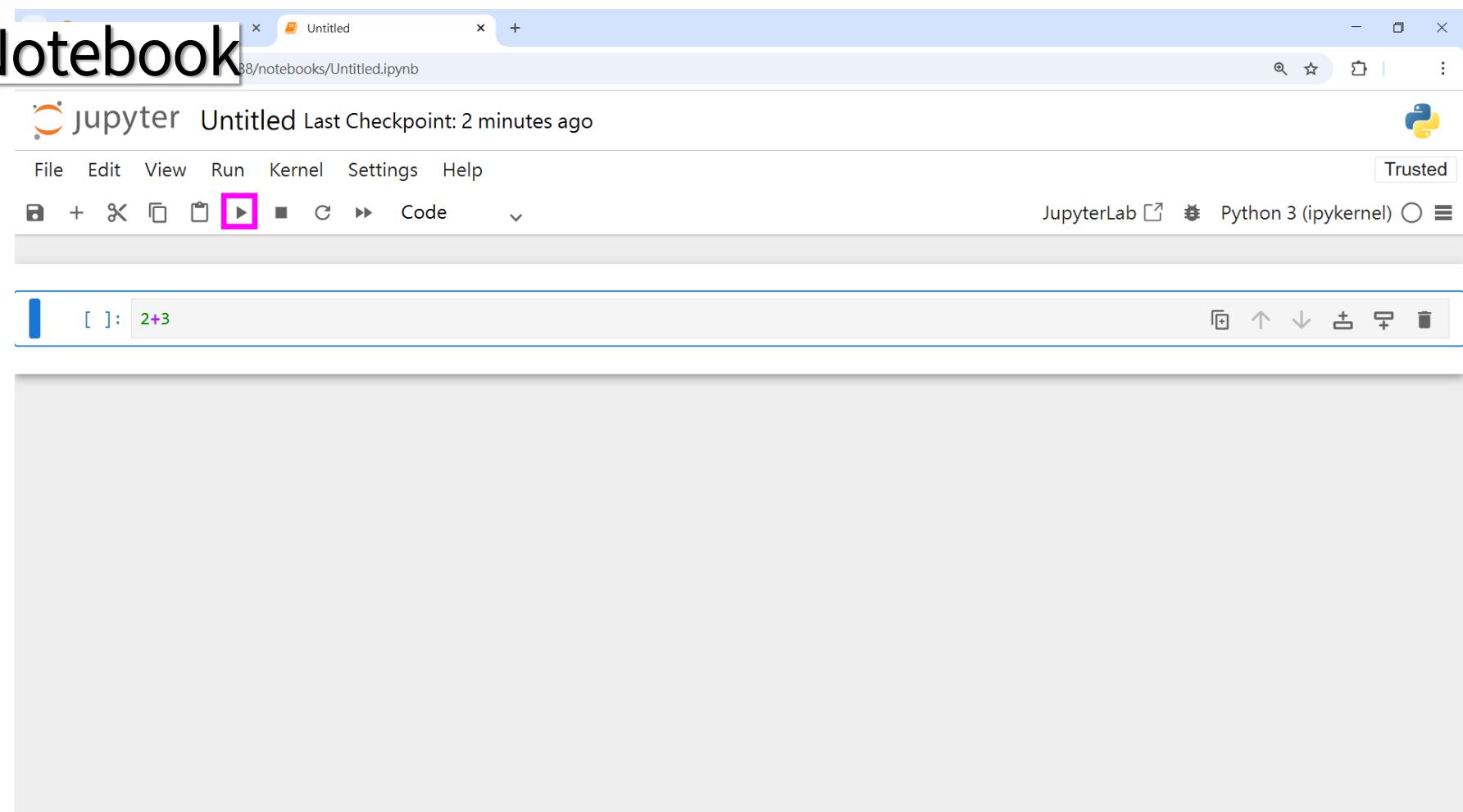
② Python3 (ipykernel)

Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

b. 運算

① $2+3$



A screenshot of a Jupyter Notebook interface. The title bar shows "Untitled" and the URL "88/notebooks/Untitled.ipynb". The main window displays a code cell with the expression $2+3$. The cell has a blue header bar with the text "[]: 2+3". The Python logo icon is highlighted with a pink box. The menu bar includes File, Edit, View, Run, Kernel, Settings, and Help. The toolbar below the menu includes icons for file operations like new, open, save, and run, along with a "Code" dropdown. On the right, there are buttons for "JupyterLab" and "Python 3 (ipykernel)". The status bar at the bottom right shows "Trusted".

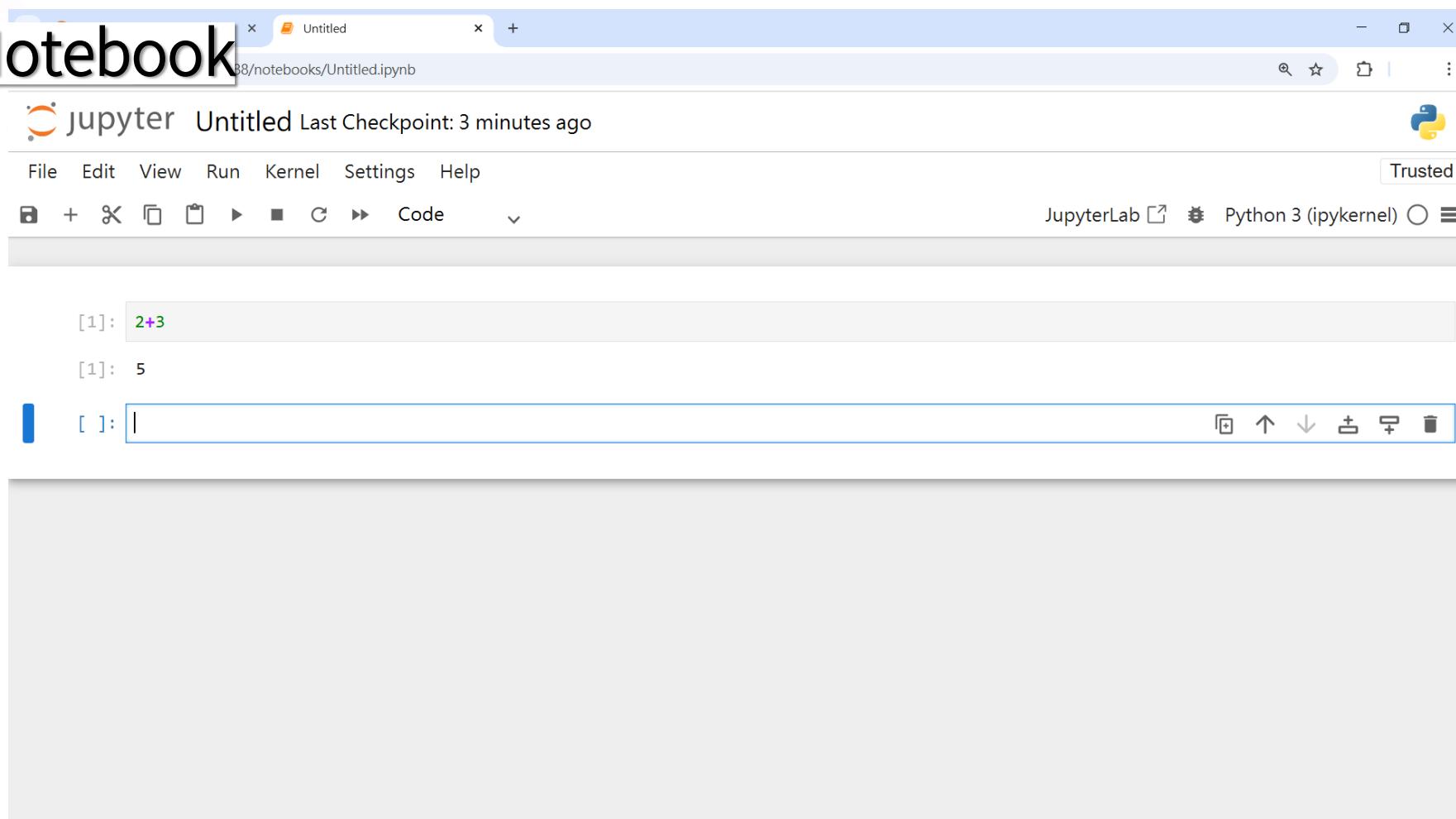
Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

b. 運算

① 2+3

② 

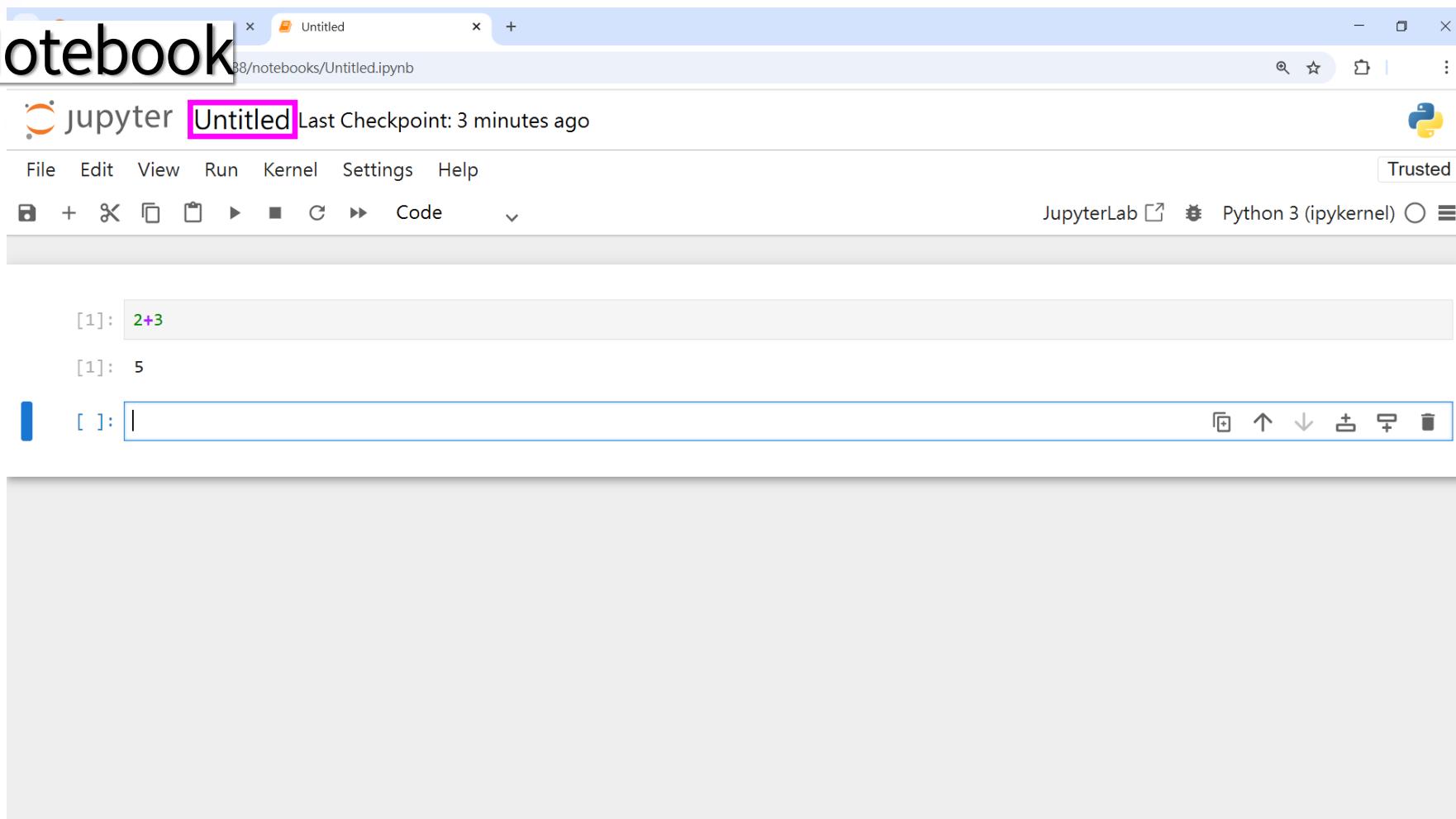


Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

c. 儲存

① 修改檔名



The screenshot shows a Jupyter Notebook interface. The title bar indicates the file is titled "Untitled". The main area displays three code cells:

- [1]: `2+3` (evaluated result: 5)
- [2]: `5`
- [3]: An empty cell with an input placeholder.

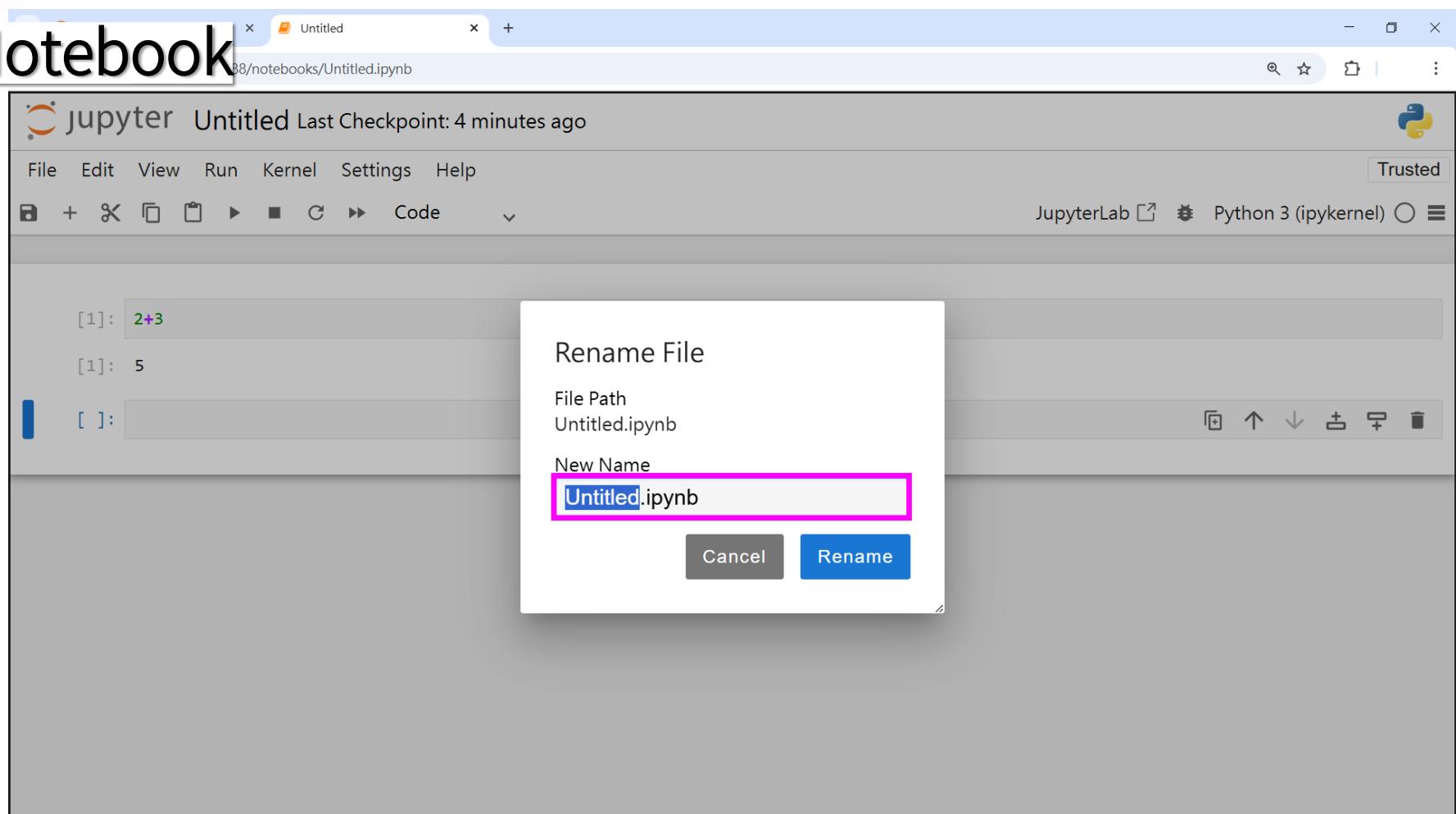
The toolbar at the top includes icons for file operations like new, open, save, and run, as well as kernel selection and help. A status bar at the bottom right shows "Trusted" and "Python 3 (ipykernel)".

Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

c. 儲存

① 修改檔名

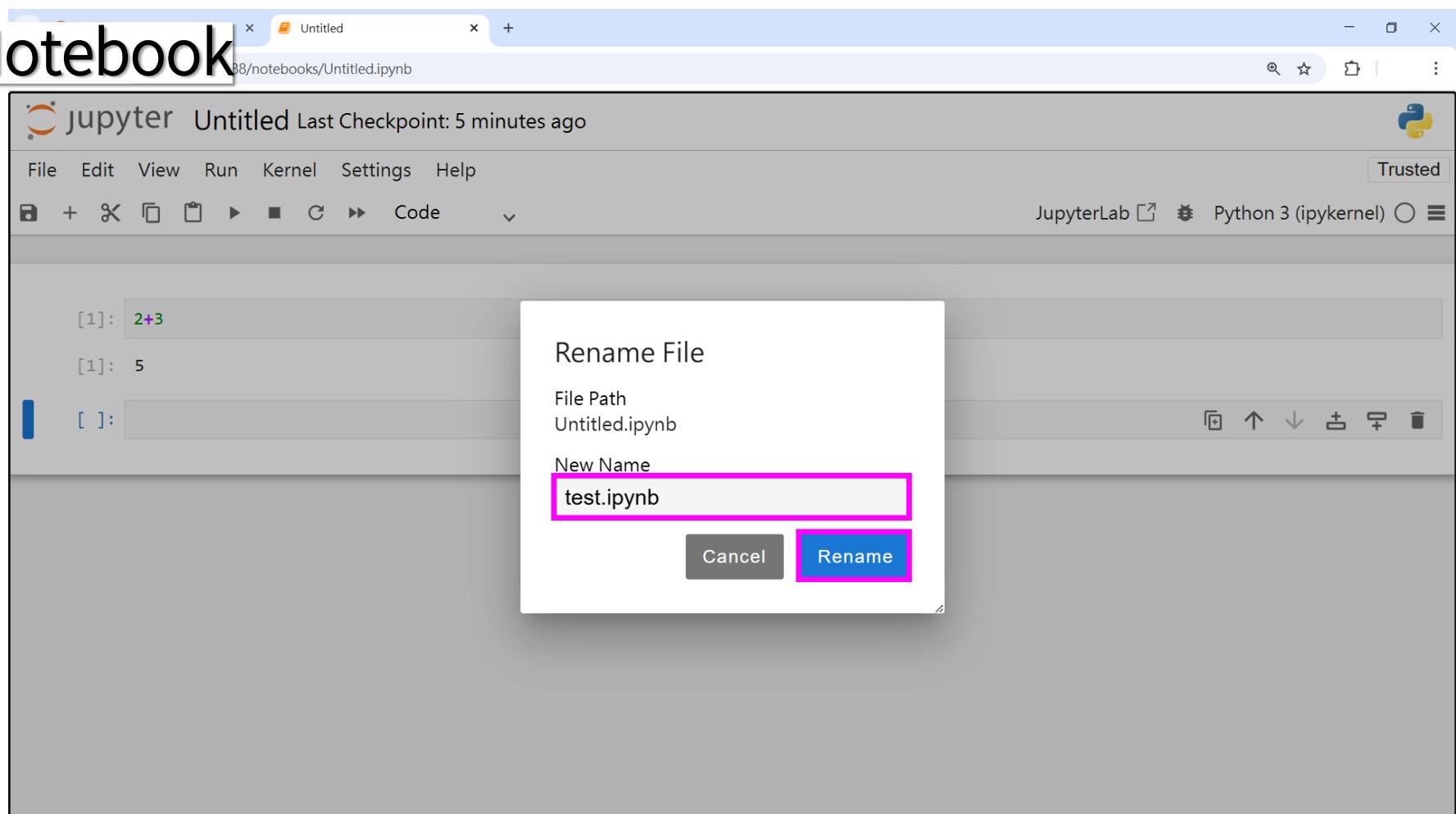


Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

c. 儲存

① 修改檔名

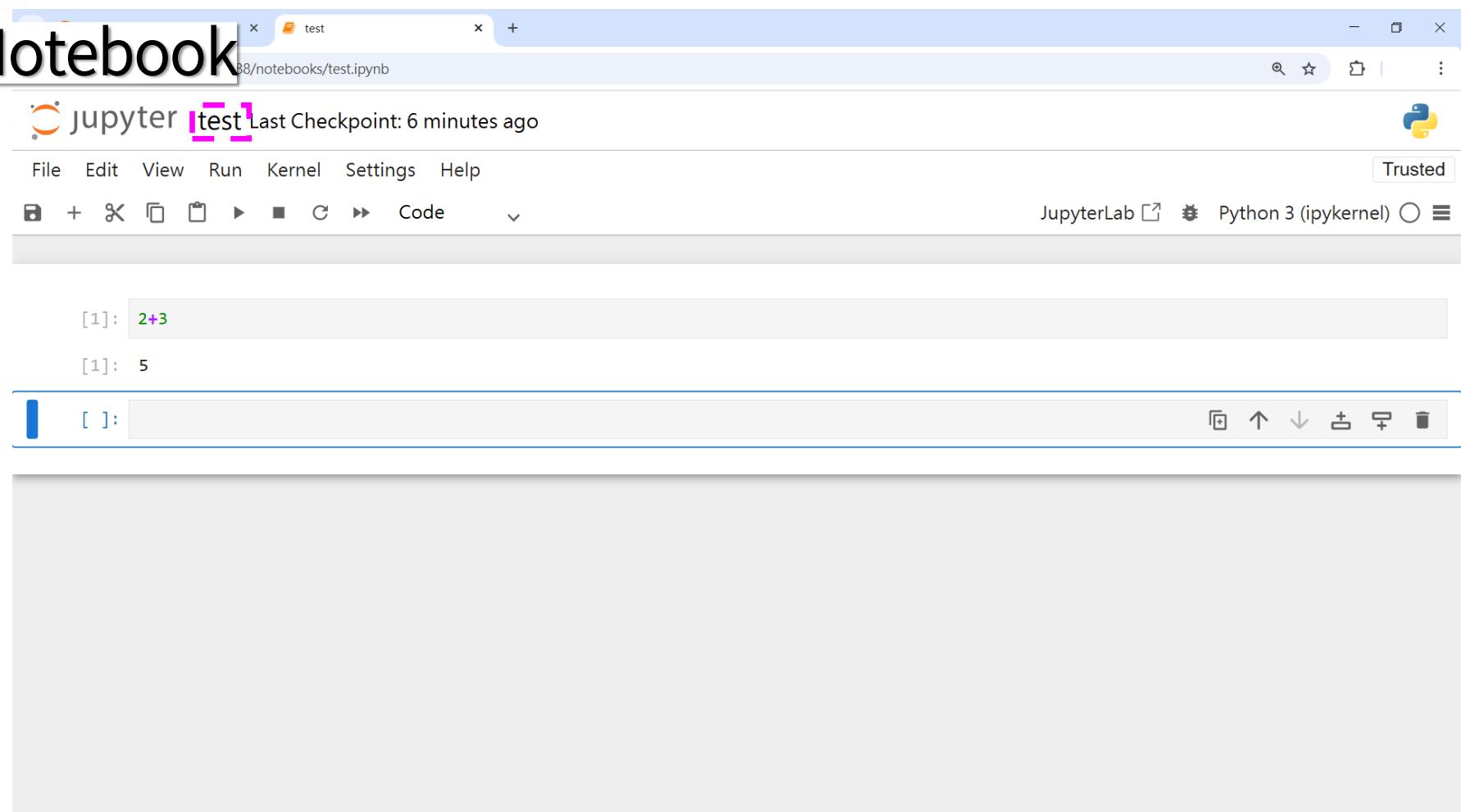


Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

C. 儲存

① 修改檔名

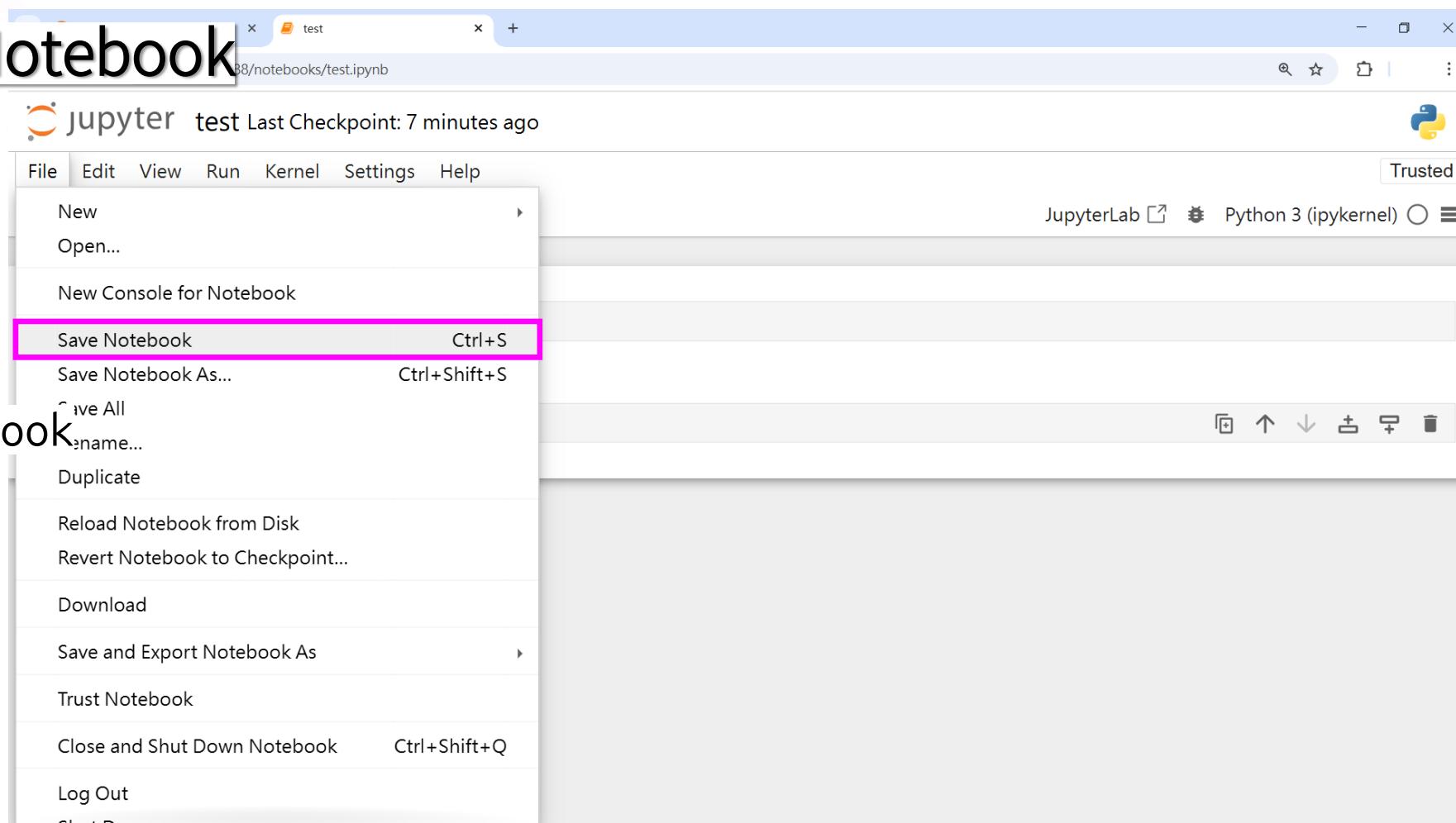


Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

C. 儲存

- ① 修改檔名
- ② File
- ③ Save Notebook

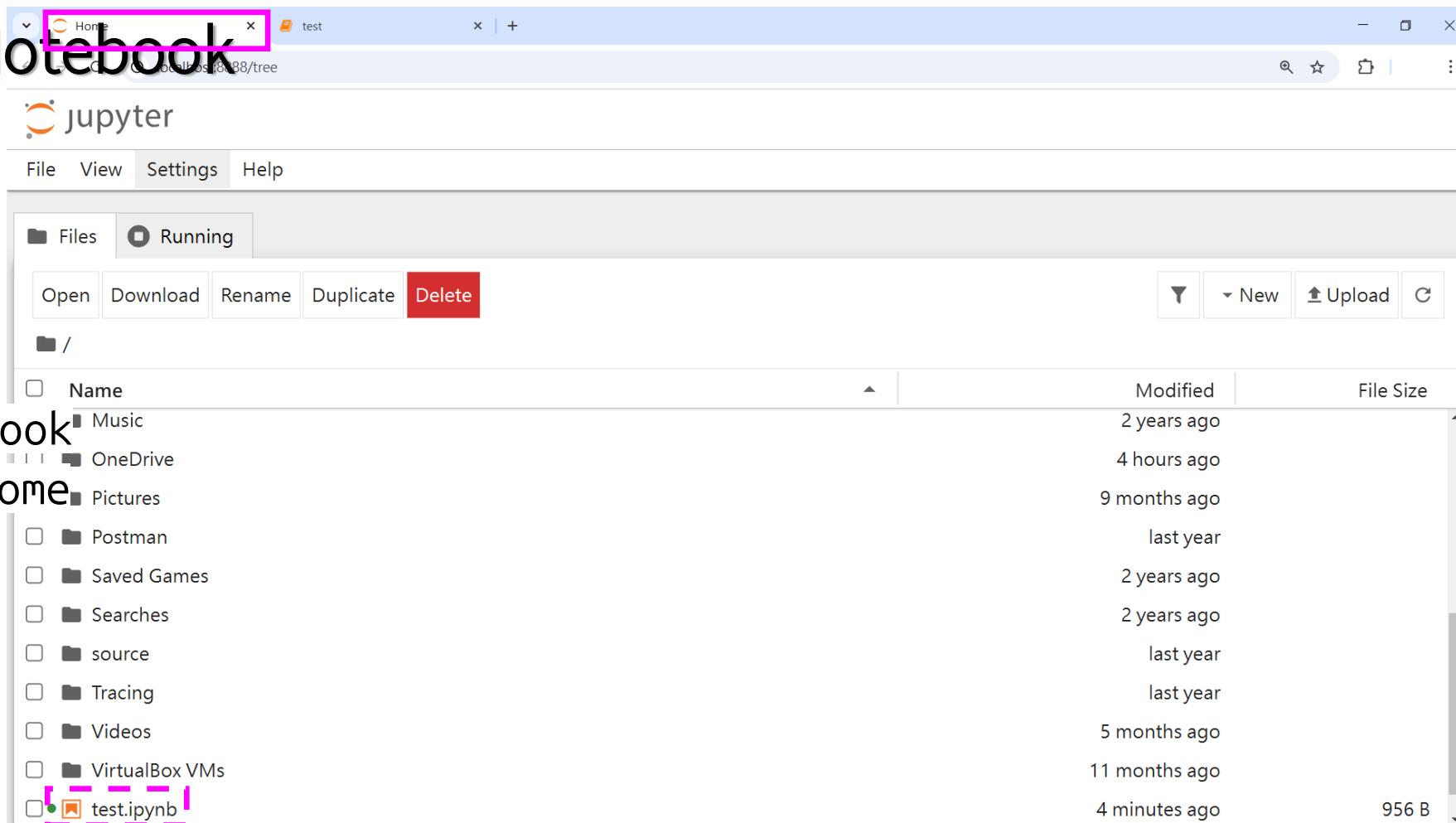


Python ft. Jupyter Notebook

1. 試用 Jupyter Notebook

C. 儲存

- ① 修改檔名
- ② File
- ③ Save Notebook
- ④ 確認儲存，Home



Python ft. Jupyter Notebook

2. Hotkey 熱鍵

常用熱鍵

熱鍵	意義
CTRL + S	儲存
SHIFT + ENTER	執行 (Run) 該 cell
CTRL + /	多列同時註解 / 取消註解
ESC + NUMBER	Markdown 註解
TAB	Autocompletion 自動完成

Your Turn

熟悉 Jupyter Notebook (5 mins)

- a. 尋找 Jupyter Notebook 儲存的檔案位置
- b. 從指定目錄開啟 Jupyter Notebook
- c. 重啟 Jupyter Notebook 關閉的網頁
- d. 關閉 Jupyter Notebook

Python ft. Jupyter Notebook

3. (微調環境)
 - a. 啟動路徑
 - b. 資料夾右鍵啟動 Miniconda
 - c. 資料夾右鍵啟動 Jupyter Notebook (Windows 11 : 顯示其他選項)
 - d. (關閉 Chrome 本機安全設定)

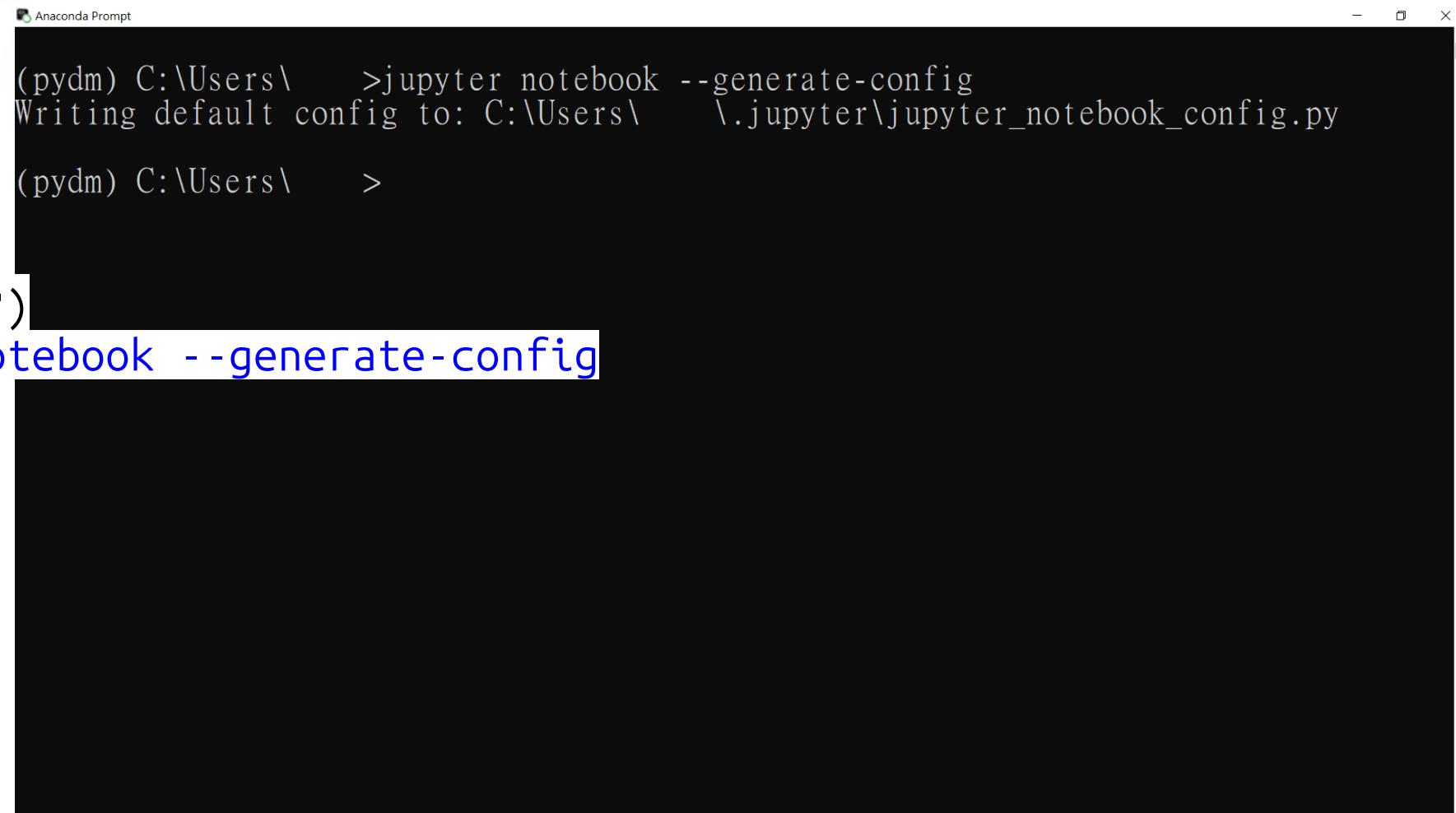
Python ft. Jupyter Notebook

3. (微調環境)

a. 啟動路徑

① (建立組態檔)

```
jupyter notebook --generate-config
```



The screenshot shows a terminal window titled "Anaconda Prompt". The command `jupyter notebook --generate-config` is being run. The output indicates that a default configuration file is being written to `C:\Users\...\\.jupyter\jupyter_notebook_config.py`. The terminal prompt then changes to `(pydm) C:\Users\...>`.

```
Anaconda Prompt
(pydm) C:\Users\... >jupyter notebook --generate-config
Writing default config to: C:\Users\...\.jupyter\jupyter_notebook_config.py
(pydm) C:\Users\... >
```

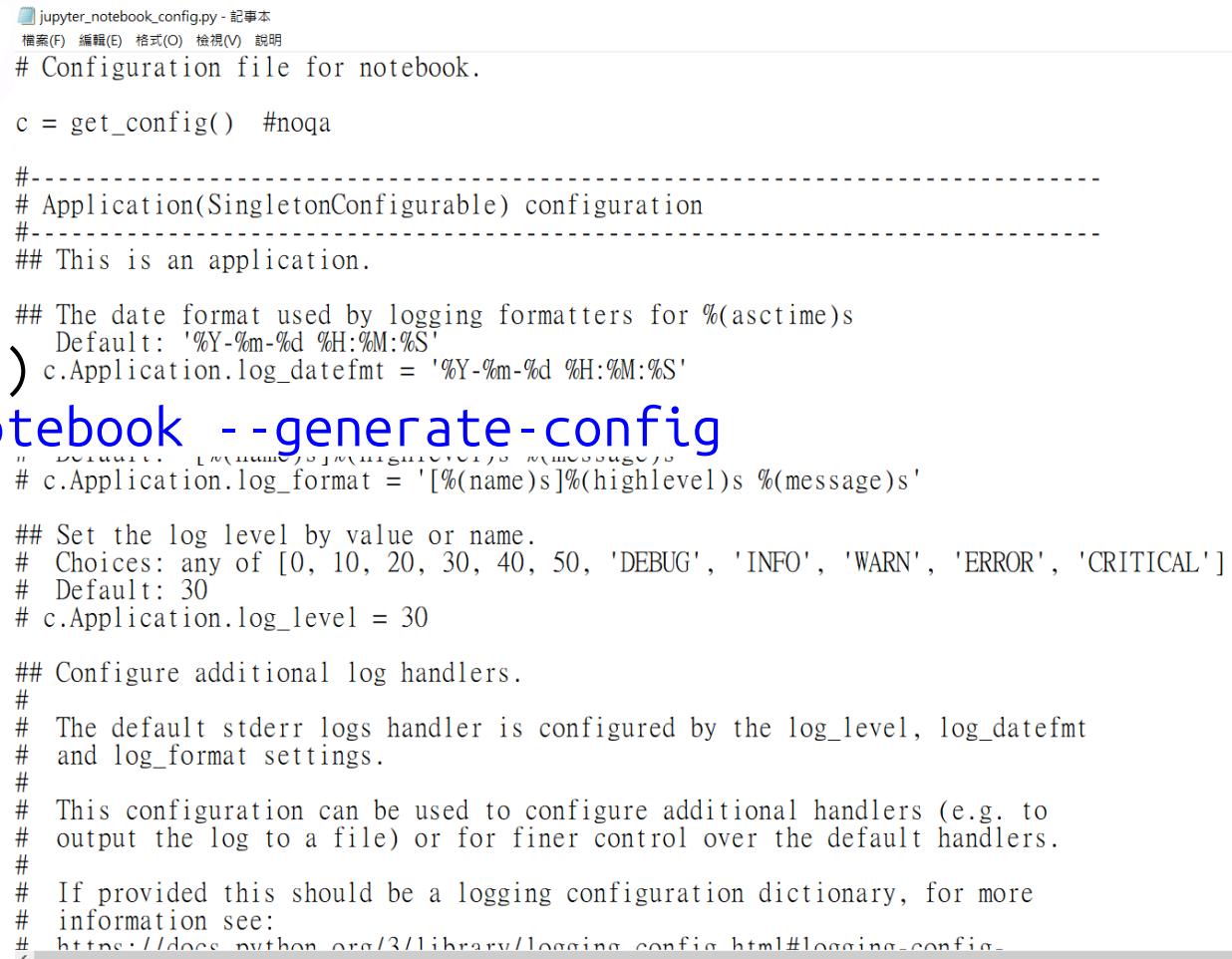
Python ft. Jupyter Notebook

3. (微調環境)

a. 啟動路徑

- ① (建立組態檔) `jupyter notebook --generate-config`

- ② 開啟組態檔



```
jupyter_notebook_config.py - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明
# Configuration file for notebook.

c = get_config() #noqa

#-----
# Application(SingletonConfigurable) configuration
#-----
## This is an application.

## The date format used by logging formatters for %(asctime)s
# Default: '%Y-%m-%d %H:%M:%S'
c.Application.log_datefmt = '%Y-%m-%d %H:%M:%S'

## Set the log level by value or name.
# Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICAL']
# Default: 30
c.Application.log_level = 30

## Configure additional log handlers.
#
# The default stderr logs handler is configured by the log_level, log_datefmt
# and log_format settings.
#
# This configuration can be used to configure additional handlers (e.g. to
# output the log to a file) or for finer control over the default handlers.
#
# If provided this should be a logging configuration dictionary, for more
# information see:
# https://docs.python.org/3/library/logging_config.html#logging-config-
```

Python ft. Jupyter Notebook

3. (微調環境)

a. 啟動路徑

- ① (建立組態檔)

```
jupyter notebook --generate-config
```

- ② 開啟組態檔

```
## Shut down the server after N seconds with no kernels running and no activity.  
# This can be used together with culling idle kernels
```

- ③ 修改 `c.ServerApp.root_dir` 為啟動路徑

```
# Instead of starting the Application, dump configuration to stdout (as JSON)  
# See also: Application.show_config  
c.ServerApp.show_config = False
```

```
## The UNIX socket the Jupyter server will listen on.  
# Default: ''  
# c.ServerApp.sock = ''
```

```
## The permissions mode for UNIX socket creation (default: 0600).  
# Default: '0600'  
# c.ServerApp.sock_mode = '0600'
```

```
## Supply SSL options for the tornado HTTPServer
```



```
jupyter_notebook_config.py - 記事本  
檔案(E) 編輯(E) 格式(O) 檢視(V) 說明  
## The directory to use for notebooks and kernels.  
# Default: ''  
c.ServerApp.root_dir = r'C:\Users\...\Desktop\pydm'  
  
## The session manager class to use.  
# Default: 'builtins.object'  
# c.ServerApp.session_manager_class = 'builtins.object'  
  
## Instead of starting the Application, dump configuration to stdout  
# See also: Application.show_config  
c.ServerApp.show_config = False
```

Python ft. Jupyter Notebook

3. (微調環境)

b. 資料夾右鍵啟動 Miniconda

- ① Windows Key + S 開啟搜尋視窗

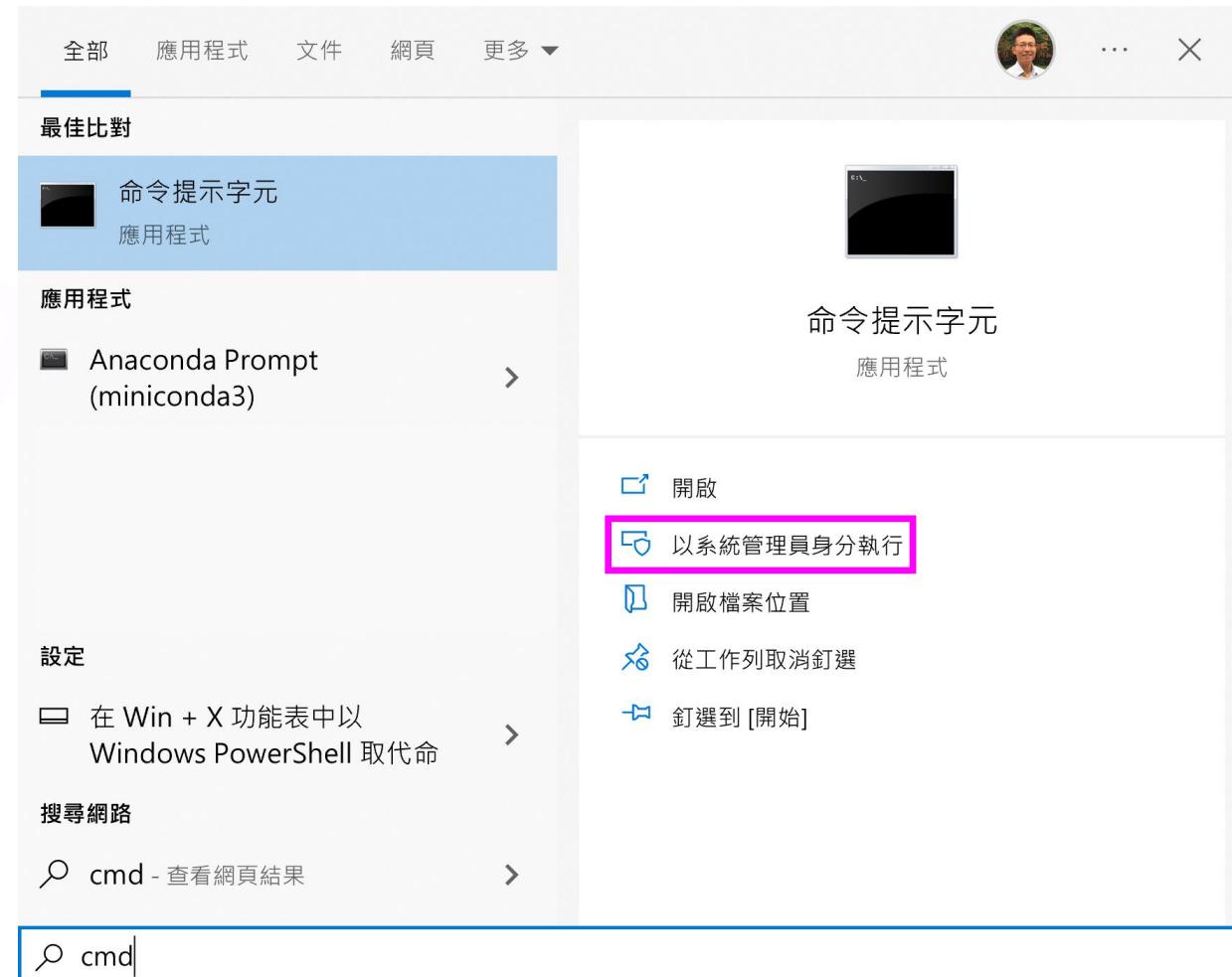
 在這裡輸入文字來搜尋

Python ft. Jupyter Notebook

3. (微調環境)

b. 資料夾右鍵啟動 Miniconda

- ① Windows Key + S 開啟搜尋視窗
- ② 搜尋 命令提示字元 (cmd)
- ③ 以系統管理員身分執行

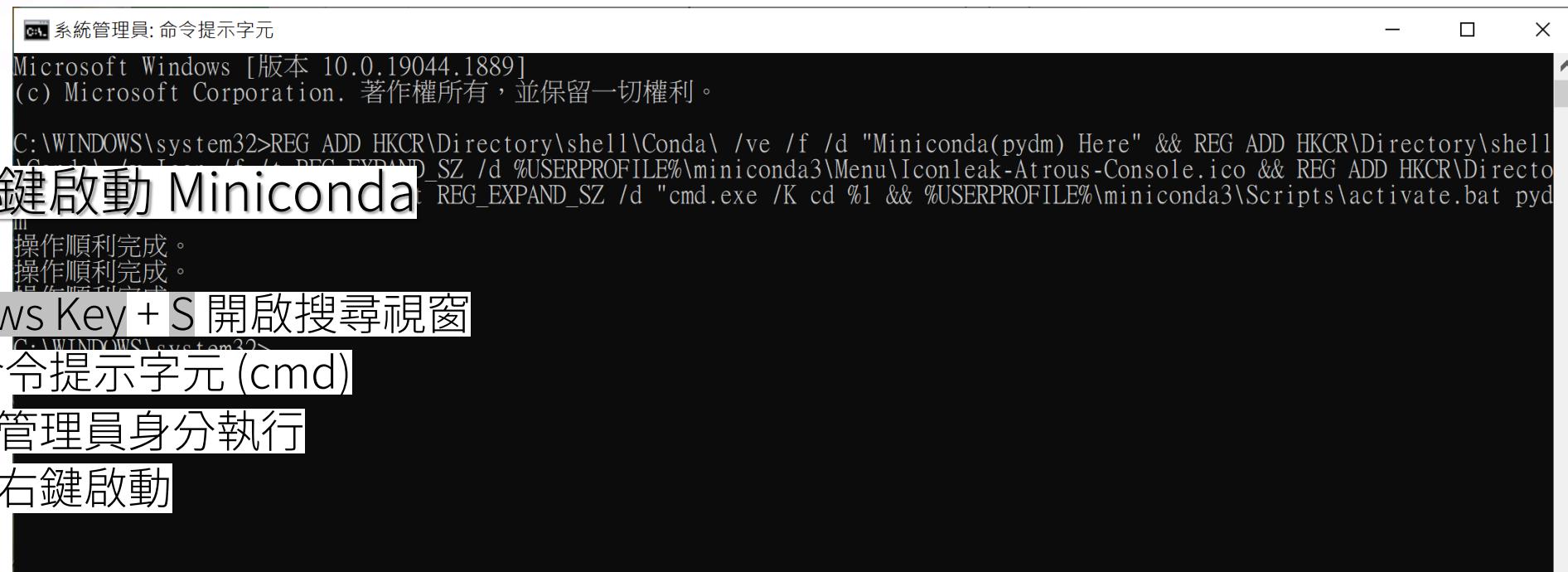


Python ft. Jupyter Notebook

3. (微調環境)

b. 資料夾右鍵啟動 Miniconda

- ① Windows Key + S 開啟搜尋視窗
- ② 搜尋 命令提示字元 (cmd)
- ③ 以系統管理員身分執行
- ④ 資料夾右鍵啟動



```
REG ADD HKCR\Directory\shell\Conda\ /ve /f /d "Miniconda(pydm) Here" &&
REG ADD HKCR\Directory\shell\Conda\ /v Icon /f /t REG_EXPAND_SZ /d %USERPROFILE%\miniconda3\Menu\Iconleak-Atrous-Console.ico && REG ADD HKCR\Directory\shell\Conda\command /f /ve /t REG_EXPAND_SZ /d "cmd.exe /K cd %1 && %USERPROFILE%\miniconda3\Scripts\activate.bat pydm" && title Anaconda Prompt (miniconda3)"
```

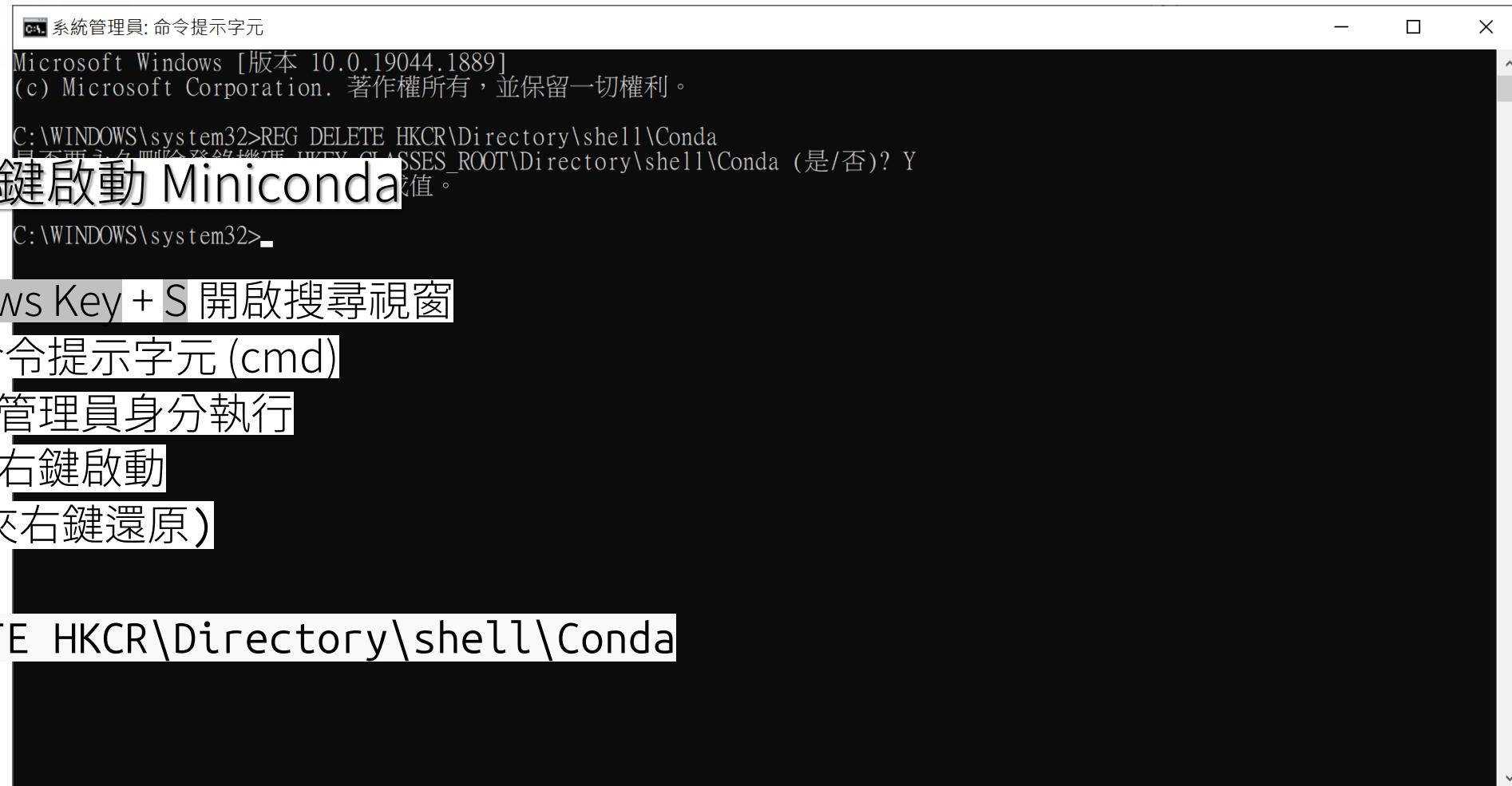
Python ft. Jupyter Notebook

3. (微調環境)

b. 資料夾右鍵啟動 Miniconda

- ① Windows Key + S 開啟搜尋視窗
- ② 搜尋 命令提示字元 (cmd)
- ③ 以系統管理員身分執行
- ④ 資料夾右鍵啟動
- ⑤ (資料夾右鍵還原)

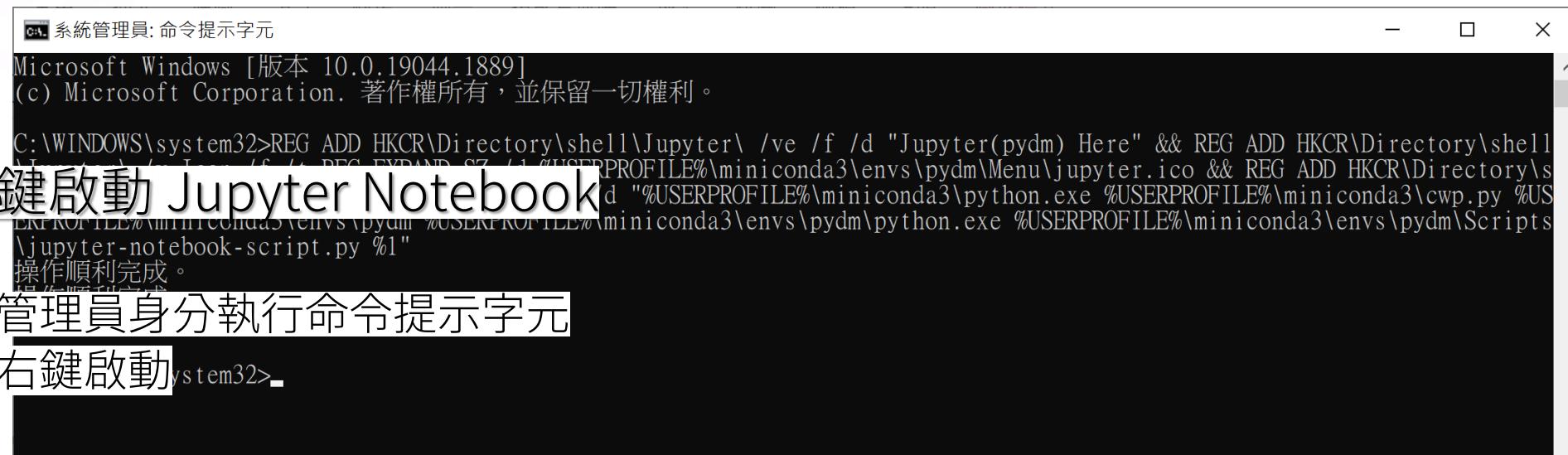
```
REG DELETE HKCR\Directory\shell\Conda
```



Python ft. Jupyter Notebook

3. (微調環境)

c. 資料夾右鍵啟動 Jupyter Notebook



- ① 以系統管理員身分執行命令提示字元
- ② 資料夾右鍵啟動

```
REG ADD HKCR\Directory\shell\Jupyter\ /ve /f /d "Jupyter(pydm) Here" &&
REG ADD HKCR\Directory\shell\Jupyter\ /v Icon /f /t REG_EXPAND_SZ /d
%USERPROFILE%\miniconda3\envs\pydm\Menu\jupyter.ico" && REG ADD
HKCR\Directory\shell\Jupyter\command /f /ve /t REG_EXPAND_SZ /d
"%USERPROFILE%\miniconda3\python.exe" %USERPROFILE%\miniconda3\cwp.py
%USERPROFILE%\miniconda3\envs\pydm\python.exe
%USERPROFILE%\miniconda3\envs\pydm\Scripts\jupyter-notebook-script.py %1"
```

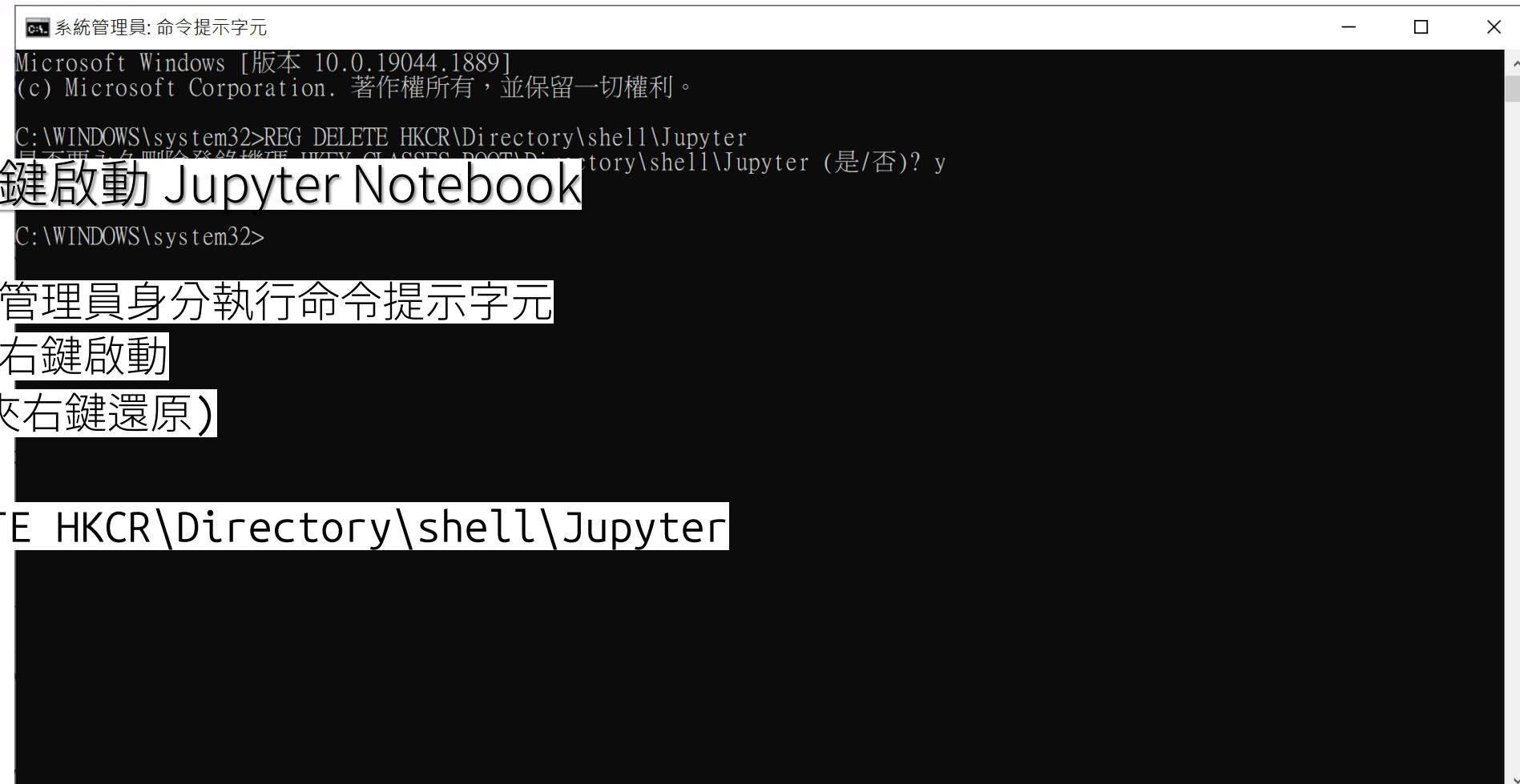
Python ft. Jupyter Notebook

3. (微調環境)

c. 資料夾右鍵啟動 Jupyter Notebook

- ① 以系統管理員身分執行命令提示字元
- ② 資料夾右鍵啟動
- ③ (資料夾右鍵還原)

```
REG DELETE HKCR\Directory\shell\Jupyter
```



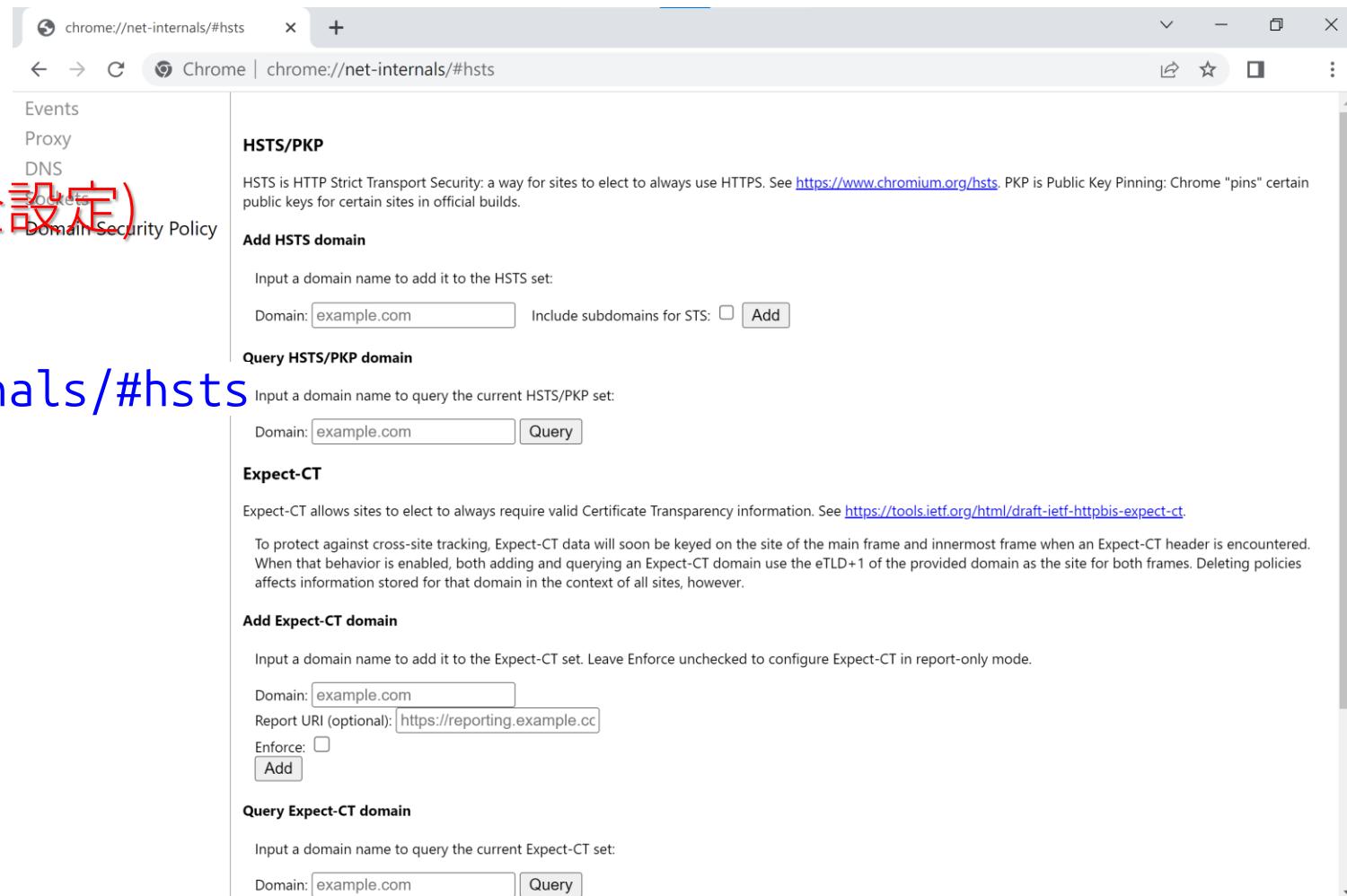
Python ft. Jupyter Notebook

3. (微調環境)

d. (移除 Chrome 本機安全設定)

① 開啟 Chrome 設定網頁

`chrome://net-internals/#hsts`



Python ft. Jupyter Notebook

3. (微調環境)

d. (移除 Chrome 本機安全設定)

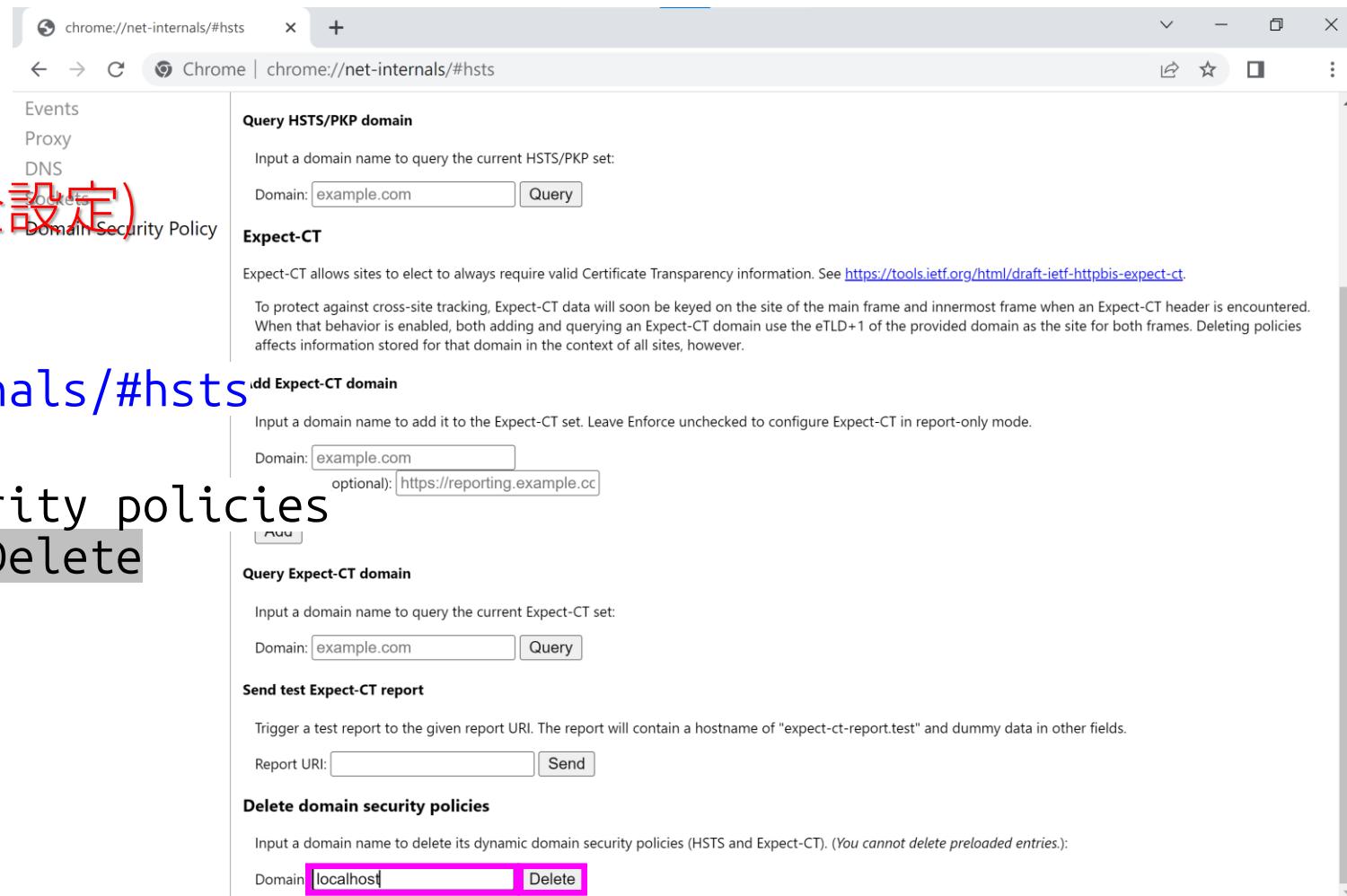
① 開啟 Chrome 設定網頁

<chrome://net-internals/#hsts>

② 移除本機安全設定

Delete domain security policies

Domain: localhost **Delete**



Python 1 Data Types

Variable-Constant-Operator, Python Built-in Data Types

Variable, Constant, Operator

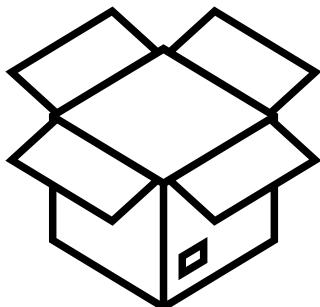
假設 Abc_123 代表 123，計算 Abc_123 + 2

Python 的語法：

```
Abc_123 = 123  
Abc_123 + 2
```

Variable, Constant, Operator

為了程式運算或資料保存，需要裝資料的容器，稱為 variable (變數)

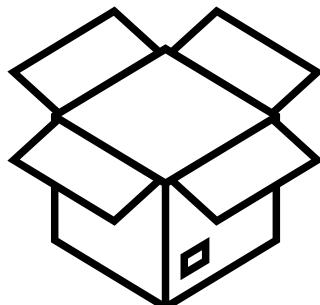


```
Abc_123 = 123  
Abc_123 + 2
```

Variable, Constant, Operator

怕忘，需要幫這個容器隨意取個名字，稱為變數名稱

- 變數名稱由英文字母、數字、_ 組成，此處命名為 Abc_123



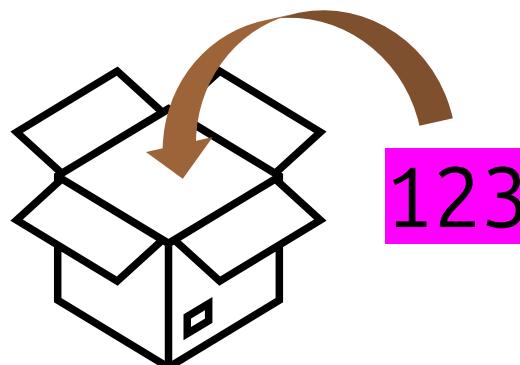
Abc_123

Abc_123 = 123
Abc_123 + 2

Variable, Constant, Operator

將資料裝進變數

- 將 123 裝入 Abc_123

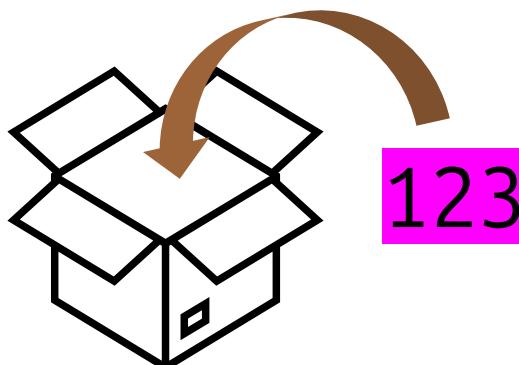


Abc_123 = 123
Abc_123 + 2

Variable, Constant, Operator

將資料裝進變數

- 裝東西這個動作稱為 assign (指派)，是一種 operation (運算)，符號為 =

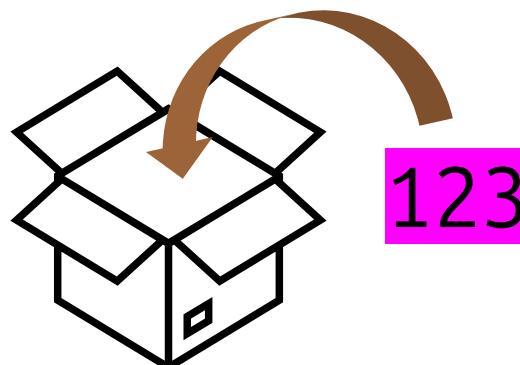


Abc_123 = 123
Abc_123 + 2

Variable, Constant, Operator

將資料裝進變數

- 被裝進去的東西稱為 constant (常數)

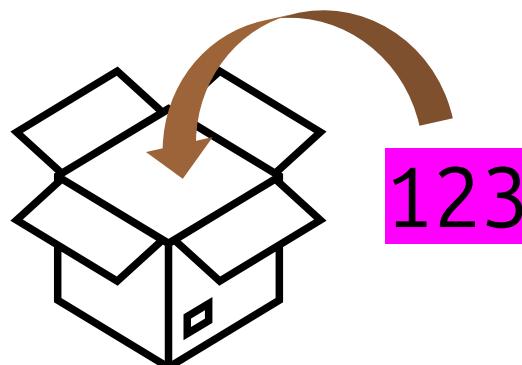


Abc_123 = 123
Abc_123 + 2

Variable, Constant, Operator

將資料裝進變數

- Python 世界裡的變數啥都能裝，而且它會記錄每種被裝進去常數的 data type (資料型態)；而 123 的 data type 為 int (整數)

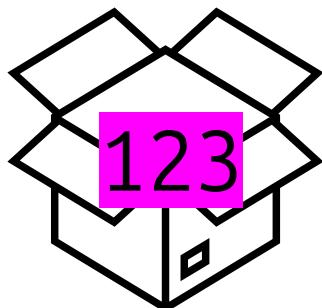


```
Abc_123 = 123  
Abc_123 + 2
```

Variable, Constant, Operator

指派完畢，變數即代表資料

- 當指派完成，`Abc_123` 這個變數即裝載 123 這個整數



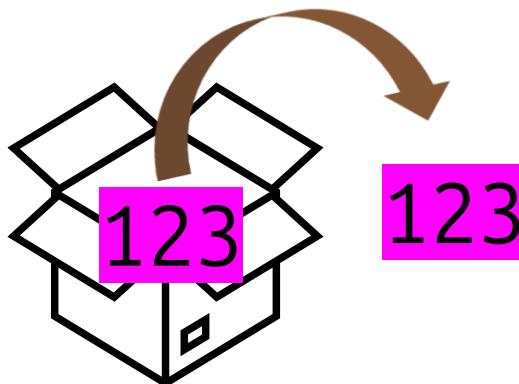
`Abc_123`

`Abc_123 = 123`
`Abc_123 + 2`

Variable, Constant, Operator

將被計算的資料提取至暫存器

- 我們只需要理解任何計算都只能在暫存器(在 CPU 中)裡進行即可，亦即，任何資料在變數中無法被計算



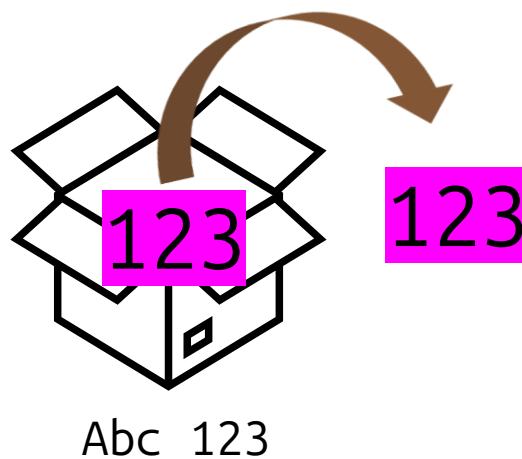
Abc_123

Abc_123 = 123
Abc_123 + 2

Variable, Constant, Operator

將被計算的資料提取至暫存器

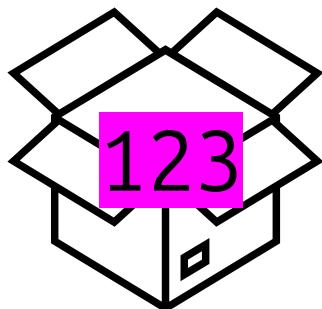
- 先自 Abc_123 取出 123 至暫存器 → 注意此時 123 仍儲存於 Abc_123 中


$$\begin{aligned} \text{Abc_123} &= 123 \\ \text{Abc_123} &+ 2 \end{aligned}$$

Variable, Constant, Operator

進行計算

- 與 2 相加得到 125 → 此時完成 `Abc_123 + 2`，得到 125，而 125 存在暫存器中你碰不到，若未來會使用通常先另行裝入變數



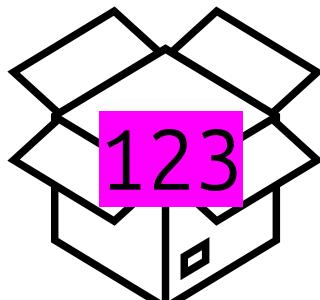
`Abc_123`

$$123 + 2 = 125$$

`Abc_123 = 123`
`Abc_123 + 2`

Variable, Constant, Operator

為了以後能夠應用，稍微複雜一點的作法



Abc_123

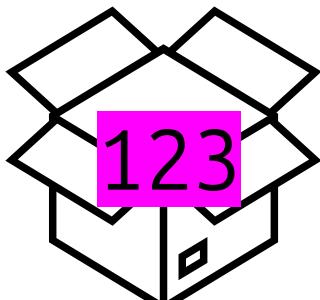
$$123 + 2 = 125$$

```
Abc_123 = 123  
Abc_123 + 2  
xyz = Abc_123 + 2  
Abc_123 = Abc_123 + 2
```

Variable, Constant, Operator

計算後將結果存回新的變數

- $xyz = \text{Abc_123} + 2$



`Abc_123`

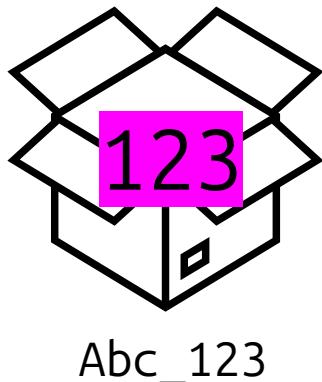
$$123 + 2 = 125$$

`Abc_123 = 123`
`Abc_123 + 2`
`xyz = Abc_123 + 2`
`Abc_123 = Abc_123 + 2`

Variable, Constant, Operator

計算後將結果存回新的變數

- $xyz = \text{Abc_123} + 2$



$$123 + 2 = 125$$

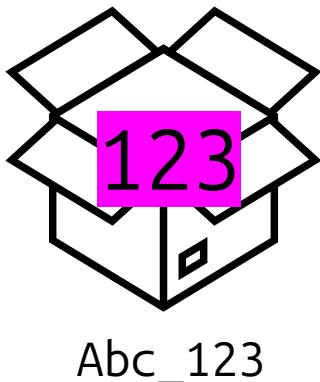
A diagram illustrating the calculation $123 + 2 = 125$. The numbers '123' and '2' are in pink boxes, and the result '125' is in a green box. A brown arrow points from the green box '125' to an open cardboard box labeled 'xyz'.

$\text{Abc_123} = 123$
 $\text{Abc_123} + 2$
 $xyz = \text{Abc_123} + 2$
 $\text{Abc_123} = \text{Abc_123} + 2$

Variable, Constant, Operator

計算後將結果存回新的變數

- Abc_123 與 xyz 兩個變數並存

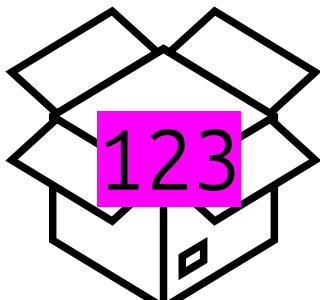


Abc_123 = 123
Abc_123 + 2
xyz = Abc_123 + 2
Abc_123 = Abc_123 + 2

Variable, Constant, Operator

計算後將結果存回原來的變數 Abc_123

- $\text{Abc_123} = \text{Abc_123} + 2$



Abc_123

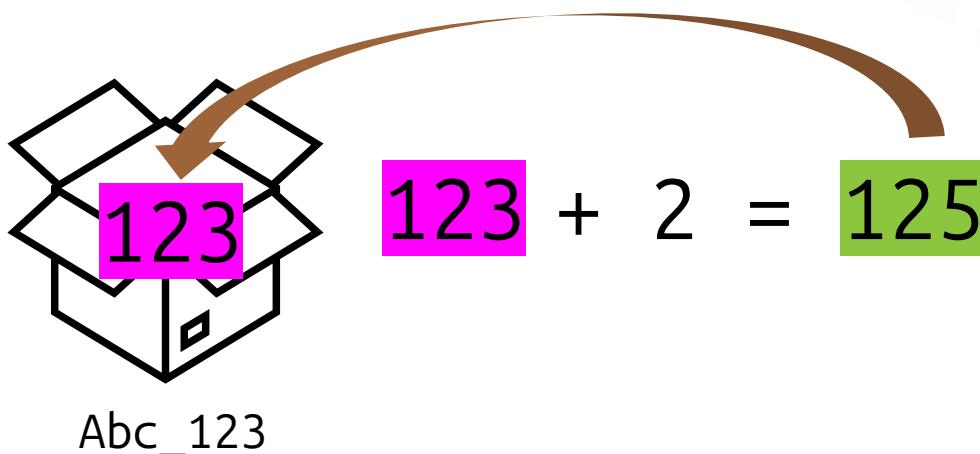
$$123 + 2 = 125$$

Abc_123 = 123
Abc_123 + 2
xyz = Abc_123 + 2
Abc_123 = Abc_123 + 2

Variable, Constant, Operator

計算後將結果存回原來的變數 Abc_123

- $\text{Abc_123} = \text{Abc_123} + 2$

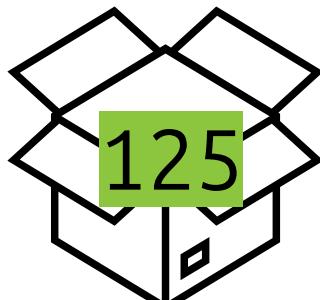


$\text{Abc_123} = 123$
 $\text{Abc_123} + 2$
 $\text{xyz} = \text{Abc_123} + 2$
 $\text{Abc_123} = \text{Abc_123} + 2$

Variable, Constant, Operator

計算後將結果存回原來的變數 Abc_123

- 此時 Abc_123 的值更新為 125



Abc_123

```
Abc_123 = 123  
Abc_123 + 2  
xyz = Abc_123 + 2  
Abc_123 = Abc_123 + 2
```

Your Turn 1-1

1. 假設 Abc_123 為 123，嘗試計算 (5 mins)

Abc_123 + 2

Abc_123 + 2 的結果儲存至 xyz

Abc_123 + 2 的結果儲存回 Abc_123

2. 熟悉 Jupyter Notebook Cell

Common Python Built-in Data Types

最常見的資料型態

Data	Data Type	Note
256	int	Integer 整數
256.	float	Floating point number 浮點數
256.1	float	Floating point number 浮點數
True	bool	Boolean 布林
'256.1'	str	String 字串
'Enos'	str	String 字串
[1, 2, 3, 4]	list	List 串列
(1, 2, 3, 4)	tuple	Tuple 元組
{'a':1, 'b':2, 'c':3}	dict	Dictionary 字典
{1, 3, 6}	set	Set 集合

Common Python Built-in Data Types

最常見的資料型態

```
1 print(type(256))  
<class 'int'>
```

```
1 print(type(256.))  
<class 'float'>
```

```
1 print(type(256.1))  
<class 'float'>
```

```
1 print(type(True))  
<class 'bool'>
```

```
1 print(type('256.1'))  
<class 'str'>
```

```
1 print(type('Enos'))  
<class 'str'>
```

```
1 print(type([1,2,3,4]))  
<class 'list'>
```

```
1 print(type((1,2,3,4)))  
<class 'tuple'>
```

```
1 print(type({'a':1, 'b':2, 'c':3}))  
<class 'dict'>
```

```
1 print(type({1, 3, 6}))  
<class 'set'>
```

Common Python Built-in Data Types

觀察資料型態

Function	Results or Outputs	Note
<code>print(23.5)</code>	23.5	<code>print(v)</code> → 印出 v 的值
<code>a = 256</code>		
<code>print(a)</code>	256	
<code>type(a)</code>		<code>type(a)</code> → 傳回 a 的 data type
<code>print(type(a))</code>	<code><class 'int'></code>	<code>print(type(a))</code> → 印出 a 的 data type
<code>b = type(a)</code> <code>print(b)</code>	<code><class 'int'></code>	
<code>print(type(a))</code>	<code><class 'int'></code>	

function 函數：搭配參數，執行定義好的程式碼；多數函數執行完畢後會回傳結果

Your Turn 1-2

運用 print() 與 type() 檢視 a, b, c, d 的資料型態 (5 mins)

```
a = {2, 3, 5}
```

```
b = [23.5, {1:2, 3:5}, (2, 3, 8)]
```

```
c = "This is Python"
```

```
d = False
```

Common Python Built-in Data Types

最常見的複雜資料型態

list

[], [1, 3, 5]

sequence 依序儲存多筆
資料 (elements)

elements 可以為不同
data type

elements 可以為任何
data type

mutable 可修改

tuple

(), (1, 3, 5)

sequence

elements 可以為不同
data type

elements 可以為任何
data type

immutable 不可修改

dict

{}, {'a': 'hi', 'b': 5}

mapping, 依 key-value
儲存多筆資料

key-value 可以為不同
data type

key 須為 hashable

value 可以為任何
data type

mutable

str

'', "", '''''

'hi', "hi",
'''hi''', """"hi""""

sequence

element 為 unicode
character

immutable

list

list features 特性

Data	Result or Output	Note
[]		空 list
[1, 2, 3, 4]		整數 list
[1, 2, 'fcc', 4]		混合 list
['f', 'c', 'c', 'fcc']		字串 list
[1, 2, 'fcc', [3, 4, 5]]		巢狀 list

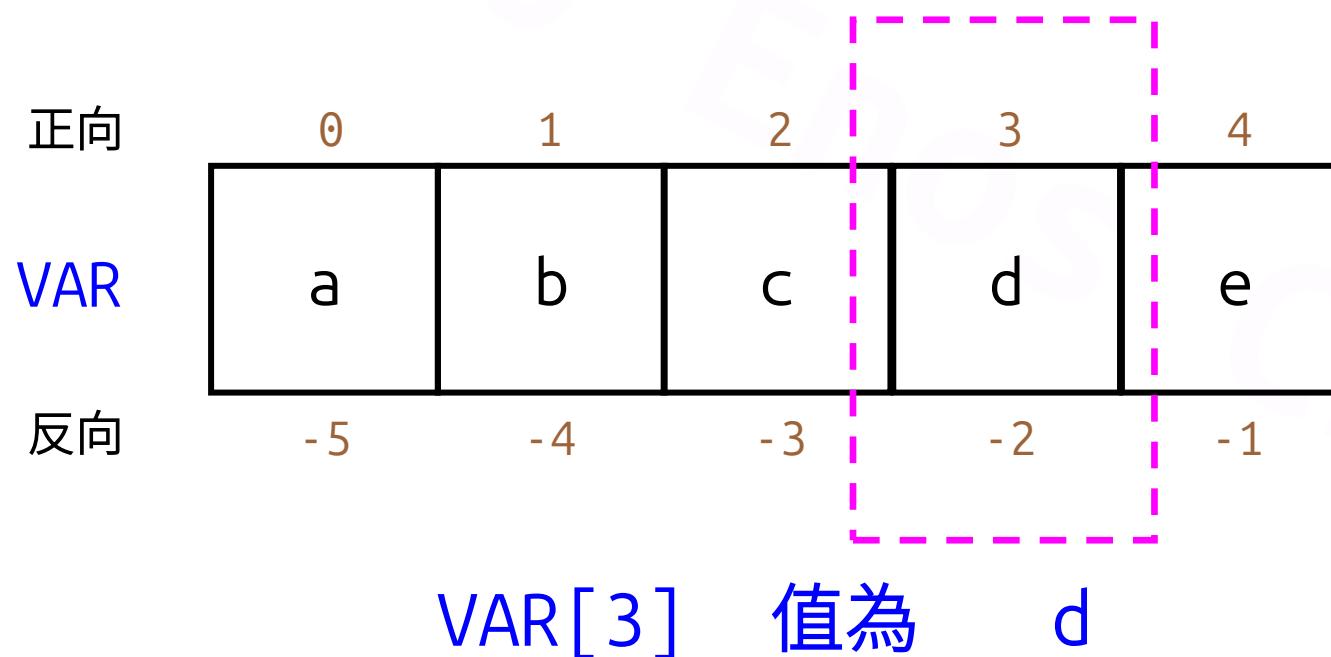
list

list features 特性

Data	Result or Output	Note
x = [1, 2, 'fcc', [3, 4, 5]]		
x[1] ↪ subscript	2	讀取 list element 元素
x[-2]	fcc	反序讀取 list element
x[3][2]	5	讀取巢狀 list element
x[:2]	[1, 2]	list slicing 切片
x[1:-2]	[2]	list slicing
x[:]	[1, 2, 'fcc', [3, 4, 5]]	list slicing

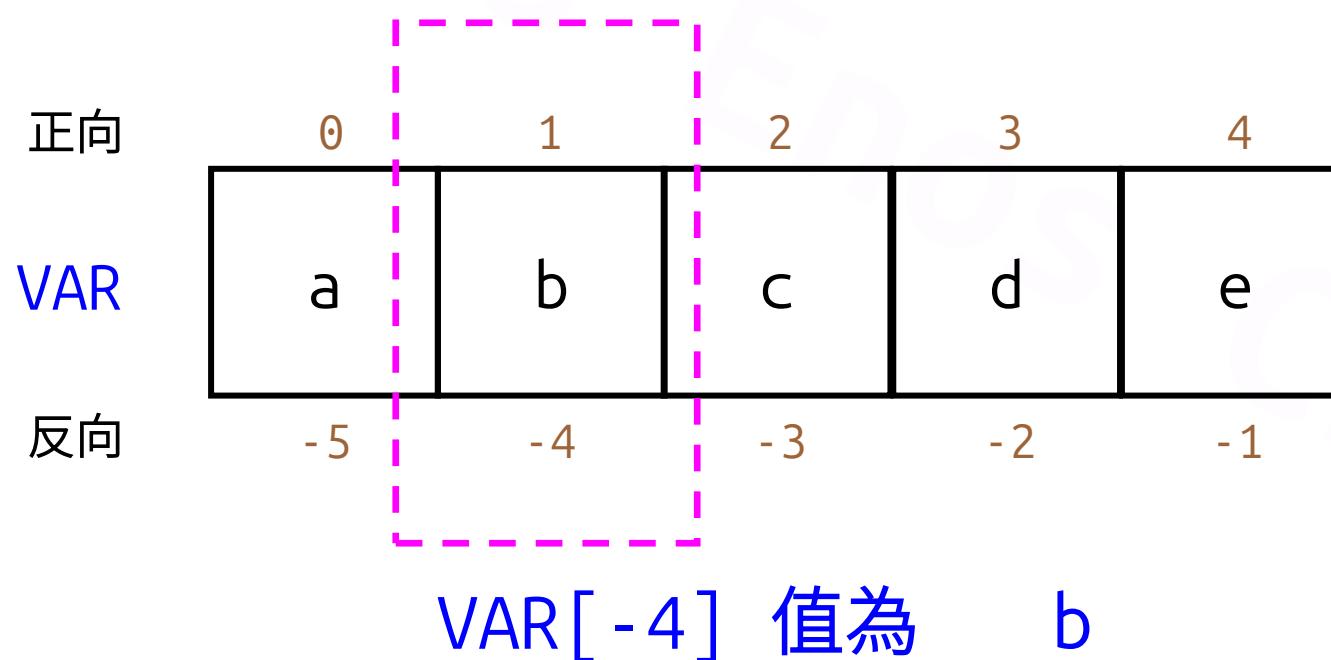
list, tuple, str Index & Slicing Concept

list, tuple, str 變數取指定元素 → VAR[i]



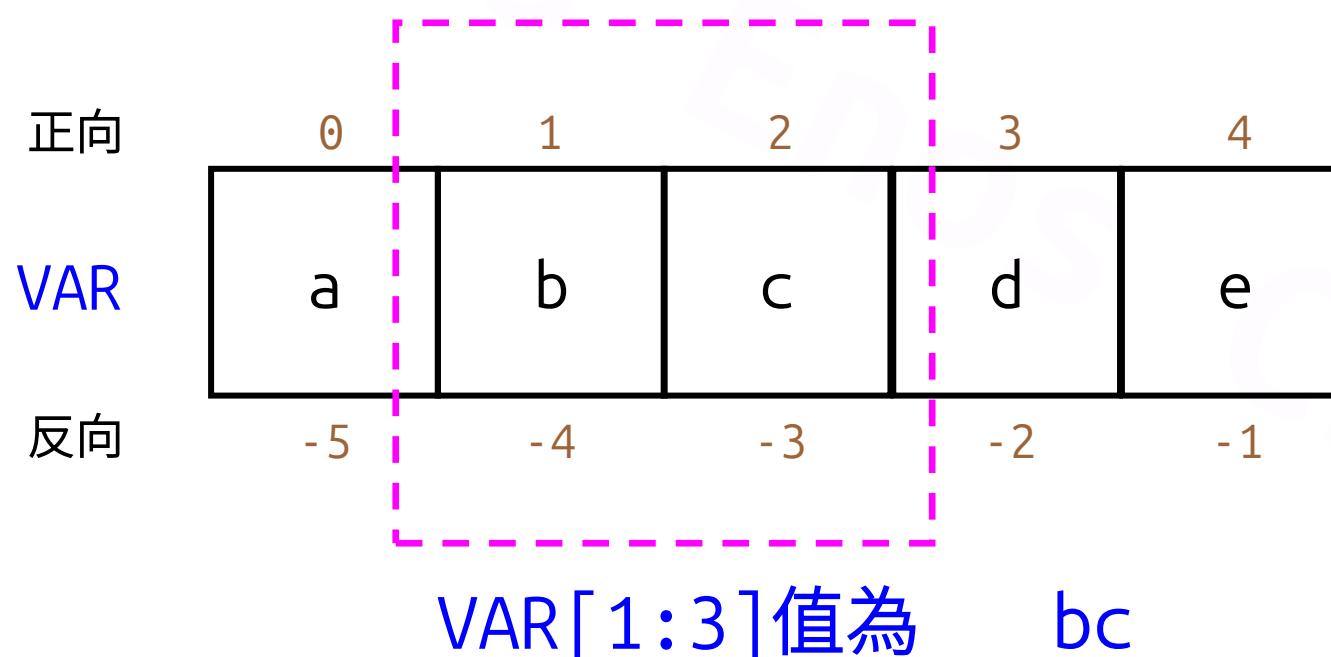
list, tuple, str Index & Slicing Concept

list, tuple, str 變數取指定元素 → VAR[i]



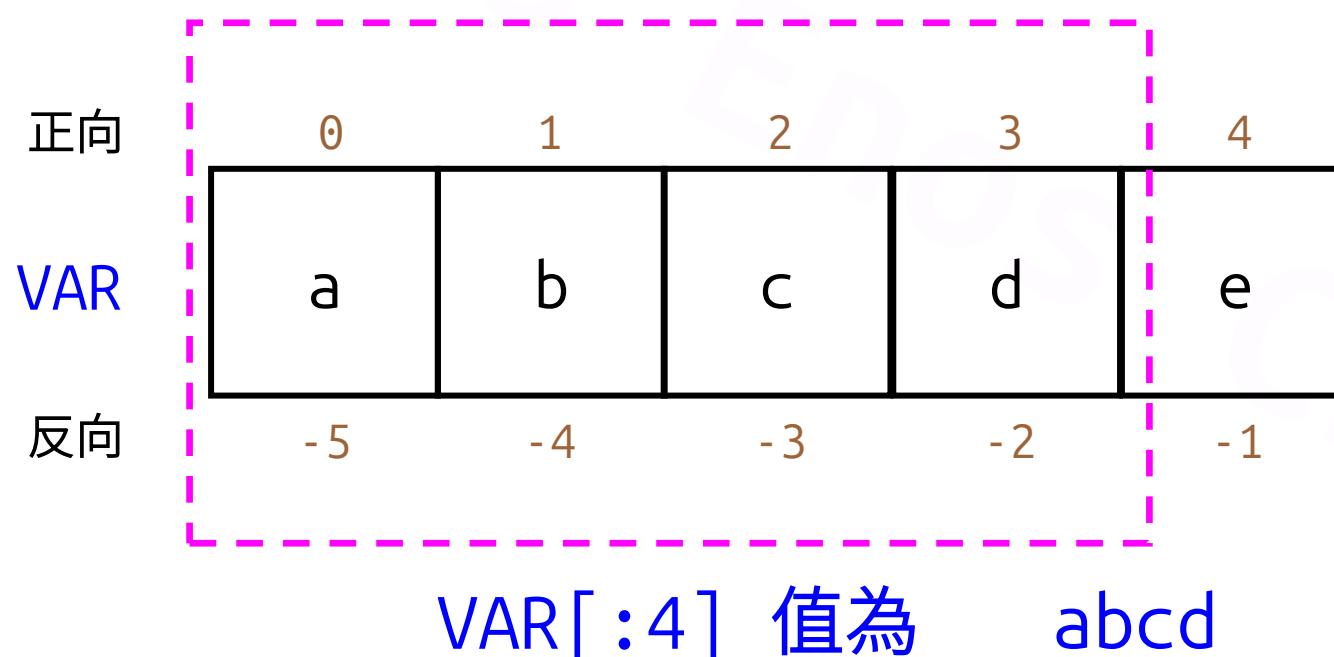
list, tuple, str Index & Slicing Concept

list, tuple, str 變數取指定元素 → VAR[i:j]



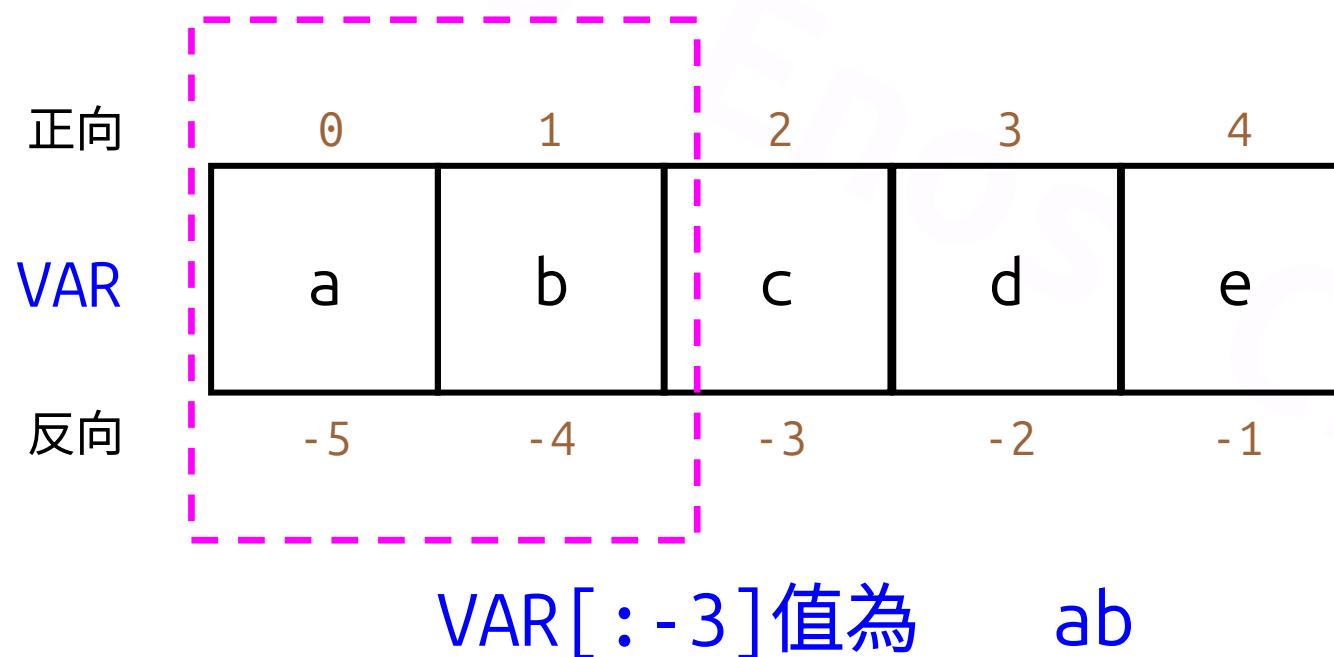
list, tuple, str Index & Slicing Concept

list, tuple, str 變數取指定元素 → VAR[i:j]



list, tuple, str Index & Slicing Concept

list, tuple, str 變數取指定元素 → VAR[i:j]



list

list methods 常用方法

Method	Result or Output	Note
<code>x.append(5.2)</code>	<code>[1, 2, 'fcc', [3, 4, 5], 5.2]</code>	<code>x.append(v) → list</code> x 尾端加入 v
<code>x.insert(0, 'enos')</code>	<code>['enos', 1, 2, 'fcc', [3, 4, 5], 5.2]</code>	<code>x.insert(p, v) →</code> 於 list x 位置 p 插入 v
<code>y = [9, 8]</code>		
<code>x.extend(y)</code>	<code>['enos', 1, 2, 'fcc', [3, 4, 5], 5.2, 9, 8]</code>	<code>x.extend(v) → 將 v</code> 中所有 element 逐一 加入 list x 尾端
<code>x.insert(0, y)</code>	<code>[[9, 8], 'enos', 1, 2, 'fcc', [3, 4, 5], 5.2, 9, 8]</code>	

list

list methods 常用方法

Method	Result or Output	Note
<code>x.remove(9)</code>	<code>[[9, 8], 'enos', 1, 2, 'fcc', [3, 4, 5], 5.2, 8]</code>	<code>x.remove(v) → 移除 list x 中第一個符合的 element v</code>
<code>x[0] = 100</code>	<code>[100, 'enos', 1, 2, 'fcc', [3, 4, 5], 5.2, 8]</code>	更新 list 中指定 element 值
<code>x[2:4] = ['x', 'y']</code>	<code>[100, 'enos', 'x', 'y', 'fcc', [3, 4, 5], 5.2, 8]</code>	更新 list 中指定 element 值
<code>list('enos')</code>	<code>['e', 'n', 'o', 's']</code>	<code>list(v) → 將 v 轉為 list</code>

Your Turn 1-3

1. $x = [[1, 2, 3], [4, [51, 52, 53], 6], [[71, 72, 73], [81, 82, 83], 9]]$
(3 mins)

- 如何用 x 表示 82 ?



Your Turn 1-3

2. $y = [2, 4, 6, 8, 7, 8, 9, 4, 3, 0]$ (15 mins)

- a. 用 list 的 method (function) 計算 4 的個數
(hint: 參考 Python 教學網站)

- b. 排序 y 中的數字
(hint: 參考 Python 教學網站)

```
2 | print(y)  
[0, 2, 3, 4, 4, 6, 7, 8, 8, 9]
```

- c. 合併 y 中末 4 個數字到 x
(note: y 為排序前或排序後皆可)

```
2 | print(x)  
[[1, 2, 3], [4, [51, 52, 53], 6], [[71, 72, 73], [81, 82, 83], 9], 7, 8, 8, 9]
```

tuple

tuple features

Data	Result or Output	Note
()		空 tuple
(1, 2, 3, 4)		整數 tuple
(1, 2, 'fcc', 4)		混合 tuple
('f', 'c', 'c', 'fcc')		字串 tuple
(1, 2, 'fcc', (3, 4, 5))		巢狀 tuple

tuple

tuple features

Data	Result or Output	Note
y = (1, 2, 'fcc', (3, 4, 5)) y[1]	2	讀取 tuple element
y[-2]	fcc	反序讀取 tuple element
y[3][2]	5	讀取巢狀 tuple element
y[:2]	(1, 2)	tuple slicing
y[1:-2]	(2,)	tuple slicing
y[:]	(1, 2, 'fcc', (3, 4, 5))	tuple slicing

Your Turn 1-4

1. 只含一個 element 的 list 與 tuple 如何表示？(5 mins)
- a. 設計一個 list 變數 a，使其中只有一個元素 3
 - b. 設計一個 tuple 變數 b，使其中只有一個元素 3

Your Turn 1-4

2. $a = (1, 2, 3)$ ，修改 $a[1]$ 為 5 (2 mins)

3. 查閱參考資料，比較 list 與 tuple 支援的函數 (5 mins)

dict

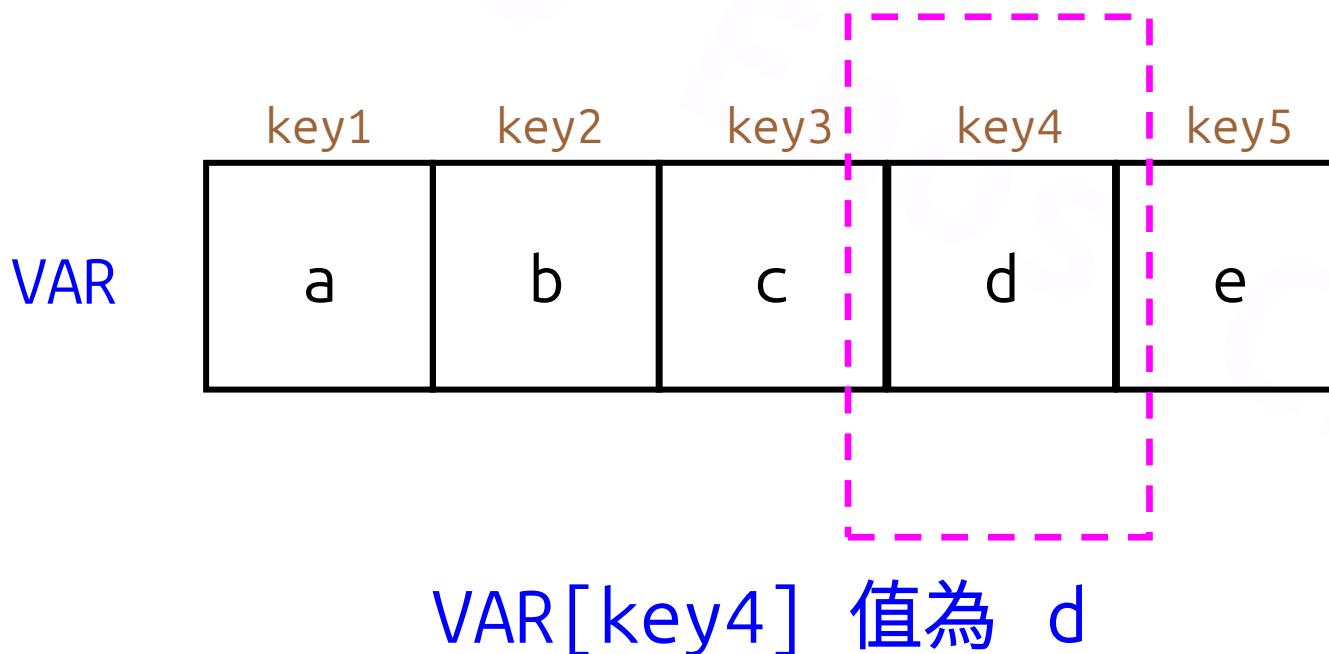
dict features

Data	Result or Output	Note	
{}		空 dict	
{'a':22, 6:'xzy'}		混和 dict	
{'a':22, 6:'xzy', (1,2):[2,3]}		混合 dict	
d = {'a':22, 6:'xzy', (1,2):[2,3]}			
d['a']	22	讀取 dict item value	
d[(1,2)]	{'a':22, 6:'xzy'}	[2, 3]	讀取 dict item value
d[6] = 'QQQ'	'a':22 → item 'a' → key	{'a':22, 6:'QQQ', (1, 2):[2, 3]}	更新 key 對應之 item value

22 → value

dict Index Concept

dict 變數取指定元素 → VAR[i]



dict

dict methods

Method	Result or Output	Note
<code>del d[6]</code>	{'a': 22, (1, 2): [2, 3]}	移除 dict d 中 key 對應之 item
<code>d[10] = 'Go'</code>	{'a': 22, (1, 2): [2, 3], 10: 'Go'}	新增 dict d 之 key-value item
<code>d.keys()</code>	<code>dict_keys(['a', (1, 2), 10])</code>	<code>d.keys()</code> → 列出 dict d 所有 key
<code>d.values()</code>	<code>dict_values([22, [2, 3], 'Go'])</code>	<code>d.values()</code> → 列出 dict d 所有 value

dict

dict methods

Method	Result or Output	Note
<code>d.get('c', 'nono')</code>	nono	<code>d.get(v1, v2)</code> → 讀取 dict d 中 key v1 對應 value；若無則回傳預設值 v2
<code>d.items()</code>	<code>dict_items([('a', 22), ((1, 2), [2, 3]), (10, 'Go')])</code>	<code>d.items()</code> → 讀取 dict d 之所有 key-value 對應
<code>dict([('a', 1), ('b', 2)])</code>	<code>{'a': 1, 'b': 2}</code>	<code>dict(v)</code> → 將 v 轉為 dict

Your Turn 1-5

1. $d = \{5:100, 26:300, 4:500\}$

- 用一個 method 取得 d 中 key 為 26 的值同時將該 item 移除 (10 mins)
(hint: 參考 Python 教學網站)

```
1 | d = {5:100, 26:300, 4:500}
```

```
value of key 26:  
300  
d =  
{5: 100, 4: 500}
```

2. $d = \{5:100, 26:300, 4:500\}$

- slice d 中 key 為 26 至 4 的 items (5 mins)

```
1 | d = {5:100, 26:300, 4:500}
```

str

str features

Data	Result or Output	Note
''		空 str
'abcde'	abcde	單引號 str
"abcde"	abcde	雙引號 str
'''abc de'''	abc de	三引號換行 str
'abc\nde'	abc de	換行 str
s = 'abcde' s[-1]	e	讀取 str element
s[:2]	ab	str slicing

str

str methods

Method	Result or Output	Note
<code>s = '12345'</code>		
<code>s.isdigit()</code>	True	<code>s.isdigit() → str s</code> 是否全為數字
<code>s = 'count 1294 pics for 0.2354 s'</code>		
<code>s.upper()</code>	COUNT 1294 PICs FOR 0.2354 S	<code>s.upper() → str s</code> 內容轉大寫
<code>ss = s.upper() ss.lower()</code>	count 1294 pics for 0.2354 s	<code>s.lower() → str s</code> 內容轉小寫

str

str methods

Method	Result or Output	Note
<code>s = 'count 1294 pics for 0.2354 s'</code>		
<code>s.split()</code>	<code>['count', '1294', 'pics', 'for', '0.2354', 's']</code>	<code>s.split()</code> → 以空白字元拆分 str s
<code>s.split('2')</code>	<code>['count 1', '94 pics for 0.', '354 s']</code>	<code>s.split(v)</code> → 以 str v 拆分 str s
<code>w = ['d:', 'pic', 't.jpg']</code>		
<code>'/'.join(w)</code>	<code>d:/pic/t.jpg</code>	<code>s.join(v)</code> → 以 str s 連結 v 中所有字串
<code>s.find('12')</code>	6	<code>s.find(v)</code> → 找出 str s 中 str v 位置

str

str methods

Method	Result or Output	Note
<code>s.replace('12', 'XXX')</code>	count XXX94 pics for 0.2354 s	<code>s.replace(u, v) →</code> 找出 str s 中之 str u , 並以 str v 取代
<code>s = ' A B C '</code>		
<code>s.strip()</code>	A B C	<code>s.strip() →</code> 清除 str s 前後的空白字元
<code>s = 'ax A aB C x'</code>		
<code>s.strip('xa')</code>	A aB C	<code>s.strip(cs) →</code> 清除 str s 前後出現在 cs 中的字元
<code>str([1, 3, 6])</code>	[1, 3, 6]	<code>str(v) →</code> 將 v 轉為 str

Your Turn 1-6

清除字串 a, b 頭尾雜訊 (10 mins)

a = ' This is a book '

b = '__This is a book__'

str

str formatting 字串格式化

Command	Result or Output	Note
a = 1294 t = 0.2354		
print('count 1294 pics for 0.2354s')	count 1294 pics for 0.2354s	
print('count', a, 'pics for', t, 's')	count 1294 pics for 0.2354 s	
print('count {:.} pics for {:.3f}s'.format(a, t))	count 1,294 pics for 0.235s	str.format()
print('count %d pics for %.3fs' % (a, t))	count 1294 pics for 0.235s	% formatting

str

str formatting 字串格式化

Command	Result or Output	Note
a = 1294 t = 0.2354		
print(f'count {a} pics for {t}s')	count 1294 pics for 0.2354s	f-string
print(f'count {a:,.} pics for {t:.3f}s')	count 1,294 pics for 0.235s	f-string

str

Format Specification Mini-Language

- <https://docs.python.org/> 搜尋 Format Specification Mini-Language

Your Turn 1-7

- 印出以下內容：(15 mins)

祖沖之在公元 0480 年計算圓周率為 3.1415926，算至小數點後 5 位為 3.14159

- 限制條件
 - 程式碼不含 '5'
 - 使用變數如下

```
1 name = '祖沖之'  
2 pi = 3.1415926  
3 year = 480  
4 digit = 5
```

祖沖之在公元 0480 年計算圓周率為 3.1415926，算至小數點後 5 位為 3.14159

Your Turn 1-7

2. 從鐵達尼號生存紀錄 CSV 檔 (titanic.csv) 擷取下列標題與內容；試將標題與內容分別拆解成 list，並逐項印出標題與對應的內容 (10 mins)

標題列字串：

index|name|sex|age|price|
level|port|surviving

內容列字串：

002|Mary|Female|34|432.23|
3|C|1

```
1 title = 'index|name|sex|age|price|level|port|surviving'  
2 text = '002|Mary|Female|34|432.23|3|C|1'
```

index : 002
name : Mary
sex : Female
age : 34
price : 432.23
level : 3
port : C
surviving: 1

Your Turn 1-7

3. 印出 I like 'Python' very much! (3 mins)

I like 'Python' very much!

str

escape character 逃脫字元 \

Character	Effect	Example	Output
\\	Backslash 反斜線		\
\'	Single quote 單引號	print('\'test\'')	'test'
\"	Double quote 雙引號	print('\"test\"')	"test"
\a	Bell		
\b	Backspace 倒退	print('xyz\b')	xy
\f	Formfeed		
\n	Linefeed 換行	print('test1\ntest2')	test1 test2

str

escape character 逃脫字元 \

Character	Effect	Example	Output
\r	Carriage return		
\t	水平 tab	print('test1\ttest2')	test1 test2
\v	垂直 tab		
\xHH	Hex code 十六進位字元	print('\xBD')	%

Python 2 Operation

Python Operators, Python Casting, Python Built-in Functions,
Python Coding Style, Python Flow Control, Python Comment

Python Operators 運算子

想像中的運算子

- 加、減、乘、除
- 大於、小於、等於
- 且、或

+ - * / ()
> < =
and or

Python 的運算子分為以下類別

- Arithmetic
- Bitwise
- Assignment
- Comparison
- Logical
- Identity
- Membership

Python Operators

Python Arithmetic Operators 算數運算子

Operation or Command	Result or Output	Note
a = 15		=
a + 3	18	+
a - 3	12	-
a * 2	30	*
a / 2	7.5	/
a // 5	3	// 商
a % 5	0	% 餘數
a ** 2	225	** 次方

Your Turn 2-1

1. 將算數運算子應用在各種 data types (3 mins)

a. $a = [1, 2, 3]$
 $b = ['a', 'b', 'c']$

```
1 a = [1, 2, 3]
2 b = ['a', 'b', 'c']
3 print(a + b)
```

$a + b$?
 $a * 2$?
 $a - b$?

```
1 print(a * 2)
```

```
1 print(a - b)
```

Your Turn 2-1

1. 將算數運算子應用在各種 data types (2 mins)

b. c = '123'
d = 'abc'

```
1 | c = '123'  
2 | d = 'abc'  
3 | print(c + d)
```

c + d ?

```
1 | print(c * 2)
```

c * 2 ?

```
1 | print(c - d)
```

c - d ?

Your Turn 2-1

2. 不同 data types 相加 (3 mins)

a. $2.5 + 3$

```
1 | print(2.5 + 3)
```

c. $\text{True} + 3$

```
1 | print(True + 3)
```

b. $'s' + 3$

```
1 | print('s' + 3)
```

Python Casting

Cast 型別轉換

Function or Command	Result or Output	Note
a = 23.1 b = 135 c = True d = '10' print(a, b, c, d)	23.1 135 True 10	
a + b	158.1	
a + c	24.1	
a + d	TypeError: unsupported operand type(s) for +: 'float' and 'str'	
a + int(d)	33.1	int(d) → 轉換 d 為 int

Python Casting

Cast 型別轉換

Function or Command	Result or Output	Note
<code>str(a)</code>	'23.1'	<code>str(a)</code> → 轉換 a 為 str
<code>float(b)</code>	135.0	<code>float(b)</code> → 轉換 b 為浮點數
<code>list((1,2,3))</code>	[1, 2, 3]	<code>list(v)</code> → 轉換 v 為 list
<code>tuple([1,2,3])</code>	(1, 2, 3)	<code>tuple(v)</code> → 轉換 v 為 tuple

Python Casting

Explicit Casting vs Implicit Casting 顯式型別轉換 vs 隱式型別轉換

```
1 a = 1.5
2 b = 3
3 c = a + b
4 print(a, type(a), b, type(b), c, type(c))
```

1.5 <class 'float'> 3 <class 'int'> 4.5 <class 'float'>

```
1 a = 1.5
2 b = True
3 c = a + b
4 print(a, type(a), b, type(b), c, type(c))
```

1.5 <class 'float'> True <class 'bool'> 2.5 <class 'float'>

implicit casting

```
1 a = 1.5
2 b = 'True'
3 c = a + b
4 print(a, type(a), b, type(b), c, type(c))
```

```
-----  
TypeError: unsupported operand type(s) for +: 'float' and 'str'  
Traceback (most recent call last)  
<ipython-input-140-8e2d797874ef> in <module>  
      1 a = 1.5  
      2 b = 'True'  
----> 3 c = a + b  
      4 print(a, type(a), b, type(b), c, type(c))
```

ambiguous

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

```
1 a = 1.5
2 b = 'True'
3 c = str(a) + b
4 print(a, type(a), b, type(b), c, type(c))
```

1.5 <class 'float'> True <class 'str'> 1.5True <class 'str'>

```
1 a = 1.5
2 b = 'True'
3 c = a + bool(b)
4 print(a, type(a), b, type(b), c, type(c))
```

1.5 <class 'float'> True <class 'str'> 2.5 <class 'float'>

bug

Your Turn 2-2

```
a = 100  
print('There are ' + a + ' people playing outside.')
```

以上程式碼報錯，請修正 (3 mins)

```
1 a = 100
2 print('There are ' + a + ' people playing outside.')
-----
TypeError Traceback (most recent call last)
<ipython-input-1-158b17548eae> in <module>
      1 a = 100
----> 2 print('There are ' + a + ' people playing outside.')
TypeError: must be str, not int
```

Python Operators

Python Assignment Operators 指派運算子

Command	Result or Output	Note
a = 2	a == 2	=
a += 3	a == 5	+= 相當於 a = a + 3
a -= 1	a == 4	-= 相當於 a = a - 1
a *= 5	a == 20	*= 相當於 a = a * 5
a /= 4	a == 5.0	/= 相當於 a = a / 4
a //= 2	a == 2.0	//= 相當於 a = a // 2
a %= 3	a == 2.0	%= 相當於 a = a % 3
a **= 4	a == 16.0	**= 相當於 a = a ** 4

Your Turn 2-3

1. 比較 `list.extend()` 與 `list +=` (3 mins)

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

```
b = [9, 9, 9]
```

```
a.extend(b)
```

```
a += b
```

Your Turn 2-3

2. 各種 assignment 實驗 (3 mins)

a = b = 1024

```
1 a = b = 1024
2 print(a, b)
```

a, b = 1024, 2048

```
1 a, b = 1024, 2048
2 print(a, b)
```

a = 1024, b = 2048

```
1 a = 1024, b = 2048
2 print(a, b)
```

Python Operators

Python Comparison Operators 比較運算子

Command	Result or Output	Note
a, b, c = 1, 2, 1		
a == b	False	== 相等
a <= b	True	<= 小於或等於
a < c	False	< 小於
a > b	False	> 大於
a >= c	True	>= 大於或等於
a != b	True	!= 不等於

Your Turn 2-4

1. 連續比較 3 個值 (3 mins)

a, b, c = 1, 2, 3

a < b < c

a < c > b

```
1 a, b, c = 1, 2, 3
```

```
1 print(a < b < c)
```

```
1 print(a < c > b)
```

2. 不等於 (3 mins)

a != b

a <> b

```
1 print(a != b)
```

```
1 print(a <> b)
```

Python Operators

a	not a
True	False
False	True

Python Logical Operators 邏輯運算子

Command	Result or Output	Note	a	b	a and b
a = 100		and	True	True	True
b = 150			True	False	False
c = 200			False	True	False
a > 100 and b > 100	False	or	False	False	False
a > 100 or b > 100	True		a	b	a or b
not a > 100	True	not	True	True	True
a > 100 and b > 50 or c > 120	True	and, or	True	False	True
a > 100 and b > 150	False	and	False	True	True
not a > 100 and b > 150	False	not, and	False	False	False
not (a > 100 and b > 150)	True	not, and			

Your Turn 2-5

1. 實作 xor (5 mins)

x = 100

y = 200

計算 $x > 100$ 與 $y < 200$
兩個條件 xor 的結果

a	b	a xor b
True	True	False
True	False	True
False	True	True
False	False	False

Your Turn 2-5

2. 先乘除後加減 (2 mins)

$$a = 100$$

$$b = 150$$

$$c = 200$$

心算 $a > 50$ or $b > 100$ and $c > 300$

Python Operators

Python Operator Precedence 優先順序

優先順序	()
	**
	+x -x ~x
	* / // %
	+
	-
	<< >>
	&
	^
	== != > >= < <= is is not in not in
	not
	and
	or
	assignment

Your Turn 2-6

2 ** 3 ** 2 (1 min)

1. 心算
2. Python 算

Python Operators

Python Operator Associativity 結合律

優先順序	()	結合律	右至左
	**		
	+x -x ~x		
	* / // %		
	+		
	<< >>		
	&		
	^		
	== != > >= < <= is is not in not in		
	not		
	and		
	or		
	assignment		

不要背 → 多用 ()

Your Turn 2-7

用 () 確保程式如你所願 (5 mins)

1. 如何讓 $a > 50 \text{ or } b > 100 \text{ and } c > 300$ 結果為 False ?

$a = 100$
 $b = 150$
 $c = 200$

```
1 a = 100
2 b = 150
3 c = 200
4
5 print(a > 50 or b > 100 and c > 300)
```

True

2. 如何讓 $2^{**} 3^{**} 2$ 為 64 ?

```
1 print(2 ** 3 ** 2)
```

512

Python Operators

Python Bitwise Operators 位元運算子

Command	Result or Output	Note
4 & 2	0	& bitwise and
4 2	6	bitwise or
~4	-5	~ bitwise not
4 ^ 2	6	^ bitwise xor
4 << 2	16	<< bitwise shift-left
4 >> 2	1	>> bitwise shift-right

Python Operators

Python Bitwise Operators

7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0

4

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

2

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

4 & 2 == 0

a	b	a & b
1	1	1
1	0	0
0	1	0
0	0	0

a	$\sim a$
1	0
0	1

a	b	a b
1	1	1
1	0	1
0	1	1
0	0	0

a	b	a ^ b
1	1	0
1	0	1
0	1	1
0	0	0

Python Operators

Python Bitwise Operators

7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0

4

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

2

0	0	0	0	0	1	1	0
---	---	---	---	---	---	---	---

4

| 2 == 6

a	b	a & b
1	1	1
1	0	0
0	1	0
0	0	0

a	$\sim a$
1	0
0	1

a	b	a b
1	1	1
1	0	1
0	1	1
0	0	0

a	b	a ^ b
1	1	0
1	0	1
0	1	1
0	0	0

Python Operators

Python Bitwise Operators

7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0

4

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

 $\sim 4 == -5$

a	b	a & b
1	1	1
1	0	0
0	1	0
0	0	0

a	$\sim a$
1	0
0	1

a	b	a b
1	1	1
1	0	1
0	1	1
0	0	0

a	b	a ^ b
1	1	0
1	0	1
0	1	1
0	0	0

Python Operators

Python Bitwise Operators

7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0

4

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

2

0	0	0	0	0	1	1	0
---	---	---	---	---	---	---	---

4 ^ 2 == 6

a	b	a & b
1	1	1
1	0	0
0	1	0
0	0	0

a	$\sim a$
1	0
0	1

a	b	a b
1	1	1
1	0	1
0	1	1
0	0	0

a	b	a ^ b
1	1	0
1	0	1
0	1	1
0	0	0

Python Operators

Python Bitwise Operators

7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0

4

0	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---

$4 \ll 2 == 16$

a	b	$a \& b$
1	1	1
1	0	0
0	1	0
0	0	0

a	$\sim a$
1	0
0	1

a	b	$a b$
1	1	1
1	0	1
0	1	1
0	0	0

a	b	$a ^ b$
1	1	0
1	0	1
0	1	1
0	0	0

Python Operators

Python Bitwise Operators

7 6 5 4 3 2 1 0
0 0 0 0 0 1 0 0 4

0 0 0 0 0 0 0 1 4 >> 2 == 1

a	b	a & b
1	1	1
1	0	0
0	1	0
0	0	0

a	$\sim a$
1	0
0	1

a	b	a b
1	1	1
1	0	1
0	1	1
0	0	0

a	b	a ^ b
1	1	0
1	0	1
0	1	1
0	0	0

Python Operators

Python Assignment Operators

Command	Result or Output	Note
a, b = 2, 2	a == 2, b == 2	=
b &= 2	b == 2	&= 相當於 b = b & 2
b = 1	b == 3	= 相當於 b = b 1
b ^= 0	b == 3	^= 相當於 b = b ^ 0
b <<= 3	b == 24	<<= 相當於 b = b << 3
b >>= 2	b == 6	>>= 相當於 b = b >> 2

Python Operators

Python Membership Operators

Command	Result or Output	Note
<code>x = [1, 3, 5]</code> <code>y = {'a':2, 'b':6}</code>		
<code>3 in x</code>	True	in
<code>2 in x</code>	False	in
<code>2 not in x</code>	True	not in
<code>'a' in y</code>	True	in
<code>2 in y</code>	False	in
<code>2 not in y</code>	True	not in

Your Turn 2-8

回答下列 in 相關問題 (5 mins)

```
a = [1, 3, 5]
b = 'This is Python'
c = {1:2, 7:9}
d = {2, 4, 6, 8}
```

1 in a ?	2 not in a ?
'i' in b ?	'x' in b ?
1 in c ?	2 in c ?
2 in d ?	3 in d ?

Python Operators

Python Identity

Command	Result or Output	Note
a, c = 1, 1		
a is None	False	is
a is c	True	is
a is not c	False	is not

Your Turn 2-9

is 實驗 (5 mins)

a. $a = 1$

$b = 1$

$a \text{ is } b ?$

$a \text{ is } 1 ?$

b. $a = 1024$

$b = 1024$

$a \text{ is } b ?$

$a \text{ is } 1024 ?$

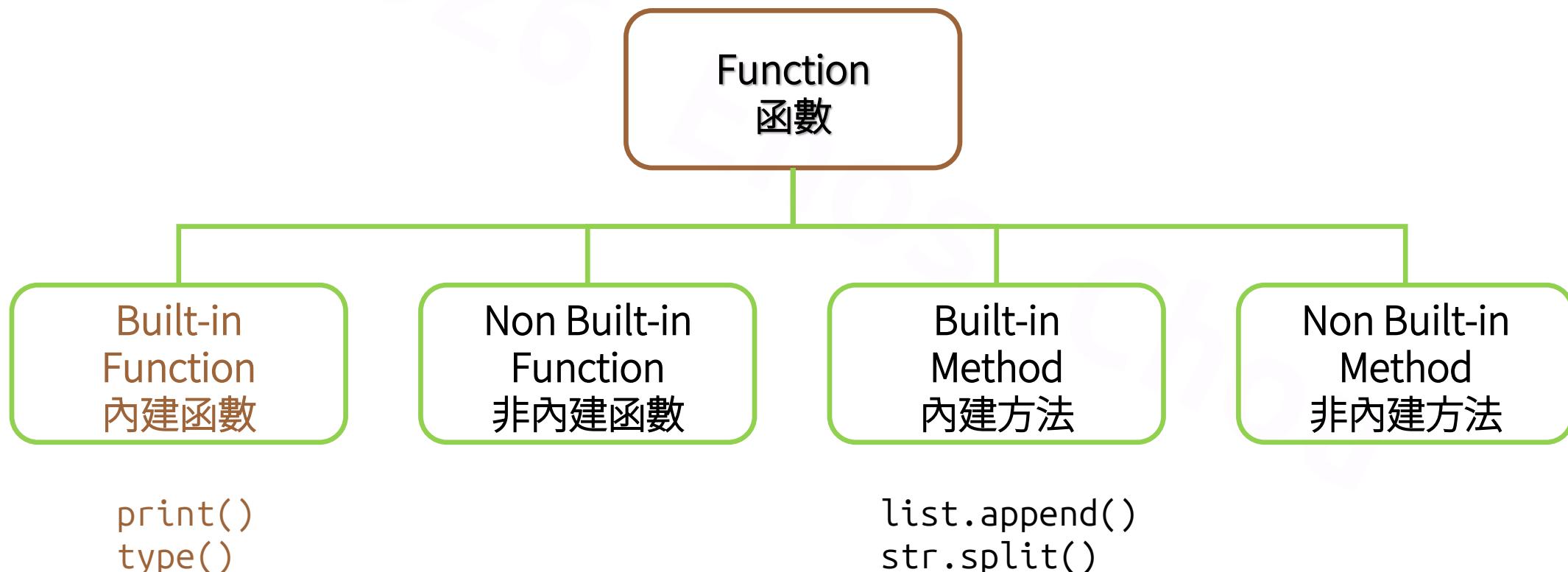
c. $a = \text{None}$

$b = \text{None}$

$a \text{ is } b ?$

$a \text{ is } \text{None} ?$

Python Built-in Functions



Python Built-in Functions

內建函數總覽

abs()	classmethod()	filter()	id()	max()	property()	str()
all()	compile()	float()	input()	memoryview()	range()	sum()
any()	complex()	format()	int()	min()	repr()	super()
ascii()	delattr()	frozenset()	isinstance()	next()	reversed()	tuple()
bin()	dict()	getattr()	issubclass()	object()	round()	type()
bool()	dir()	globals()	iter()	oct()	set()	vars()
bytearray()	divmod()	hasattr()	len()	open()	setattr()	zip()
bytes()	enumerate()	hash()	list()	ord()	slice()	__import__()
callable()	eval()	help()	locals()	pow()	sorted()	
chr()	exec()	hex()	map()	print()	staticmethod()	

Python Built-in Functions

最常用的內建函數

Function or Command	Result or Output	Note
<code>help(print)</code>	<code>print</code> function document	<code>help(v)</code> → 提供 <code>v</code> 的規格說明
<code>u = [1, 3, 5, 8]</code>		
<code>len(u)</code>	4	<code>len(u)</code> → 計算 <code>u</code> 中 <code>element</code> 個數
<code>max(u)</code>	8	<code>max(u)</code> → 計算 <code>u</code> 中最大值
<code>min(u)</code>	1	<code>min(u)</code> → 計算 <code>u</code> 中最小值
<code>sum(u)</code>	17	<code>sum(u)</code> → 加總 <code>u</code> 中所有 <code>element</code>
<code>input()</code>	inputted string	<code>input()</code> → 交談式字串輸入

Python Built-in Functions

最常用的內建函數

Function or Command	Result or Output	Note
<code>range(6)</code> <code>list(range(6))</code>	<code>range(0, 6)</code> <code>[0, 1, 2, 3, 4, 5]</code>	<code>range(v) →</code> 依 v 參數值建立範圍數列，常搭配 loop 運用
<code>v = ['a', 'b', 'c', 'd']</code>		
<code>enumerate(v)</code> <code>list(enumerate(v))</code>	<code><enumerate at 0x2839335eaf8></code> <code>[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd')]</code>	<code>enumerate(v) →</code> 以編號逐項列舉 v 之 element，常搭配 loop 運用
<code>zip(u, v)</code> <code>list(zip(u, v))</code>	<code><zip at 0x28393326d08></code> <code>[(1, 'a'), (3, 'b'), (5, 'c'), (8, 'd')]</code>	<code>zip(u, v) →</code> 將 u 與 v 中 element 逐項配對
<code>open('reading.txt')</code>		<code>open(v) →</code> 開啟檔案 v

Your Turn 2-10

1. 內建函數演練 (5 mins)

`x = {22, 7, 9, 13, 5, 20.5}`

印出 x 的元素個數、最大值、最小值、總和

x 元素個數: 6
x 最大值: 22
x 最小值: 5
x 總和: 76.5

Your Turn 2-10

2. 承 Your Turn 1-6 第 2 題 (10 mins)

標題列如下：index|name|sex|age|price|level|port|surviving
內容列如下：002|Mary|Female|34|432.23|3|C|1

請將上述資料製作成 dict 讓標題與內容正確對應，並調整 age、price、surviving 對應內容為正確資料型態，印出該 dict

```
1 title = 'index|name|sex|age|price|level|port|surviving'  
2 text = '002|Mary|Female|34|432.23|3|C|1'
```

```
{'index': '002', 'name': 'Mary', 'sex': 'Female', 'age': 34, 'price': 432.23, 'level': '3', 'port': 'C', 'surviving': 1}
```

Your Turn 2-10

3. 變數命名 (3 mins)

承上題，試將 title 重新命名為 print，觀察執行結果

承上題，試將 title 重新命名為 and，觀察執行結果

Python Keywords 保留字

保留字總覽

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

Python Coding Style

程式風格

- 參考規範

PEP 8 -- Style Guide for Python
Code

- 最重要的是縮排

- Python 與 C/ Java 等傳統程式語言不同，以縮排控制程式段 (Code Block) 而非大括號；誤用會報錯
- Python 程式段須採用相同縮排
- 縮排可採用 tab 或 space，一旦採用需保持一致；官方建議為 4 spaces

```
1 x_flag = True
2 y_flag = False
3 x = 256
4 y = 128
5 ans = None
6
7 if x_flag and (x-y) > 150:
8     ans = 1
9 elif y_flag or y > 100:
10     ans = 2
11 else:
12     ans = 3
13
14 print(ans)
```

Python Coding Style

縮排目的

- 定義程式段 Code Block 範圍
- 相同程式段內的程式碼歸屬於首行

```
1 def fn(val):
2     print('fn start')
3     print('this is fn')
4     print('fn end')
5     return 1 if val > 0 else 0
6
7 if fn(5) == 1:
8     print('if start')
9     print('this is if')
10    print('if end')
11 elif fn(5) == 0:
12     print('elif start')
13     print('this is elif')
14     print('elif end')
15
16 for i in range(4):
17     print(f'this is for, {i}')
18     print('for will go next')
```

fn start
this is fn
fn end
if start
this is if
if end
this is for, 0
for will go next
this is for, 1
for will go next
this is for, 2
for will go next
this is for, 3
for will go next

Python Flow Control

會寫一點點程式就想偷懶

程式不應該只能直直往下跑，能不能有時跑有時不跑？

`if-elif-else` 條件成立才執行

做同樣事情的程式碼能不能寫一次就好？不想一直重複寫？

`for loop` 連續性的重複執行

`while loop` 細膩控制連續性的重複執行

`break` 重複執行提早結束

`continue` 重複執行快轉

Your Turn 2-11

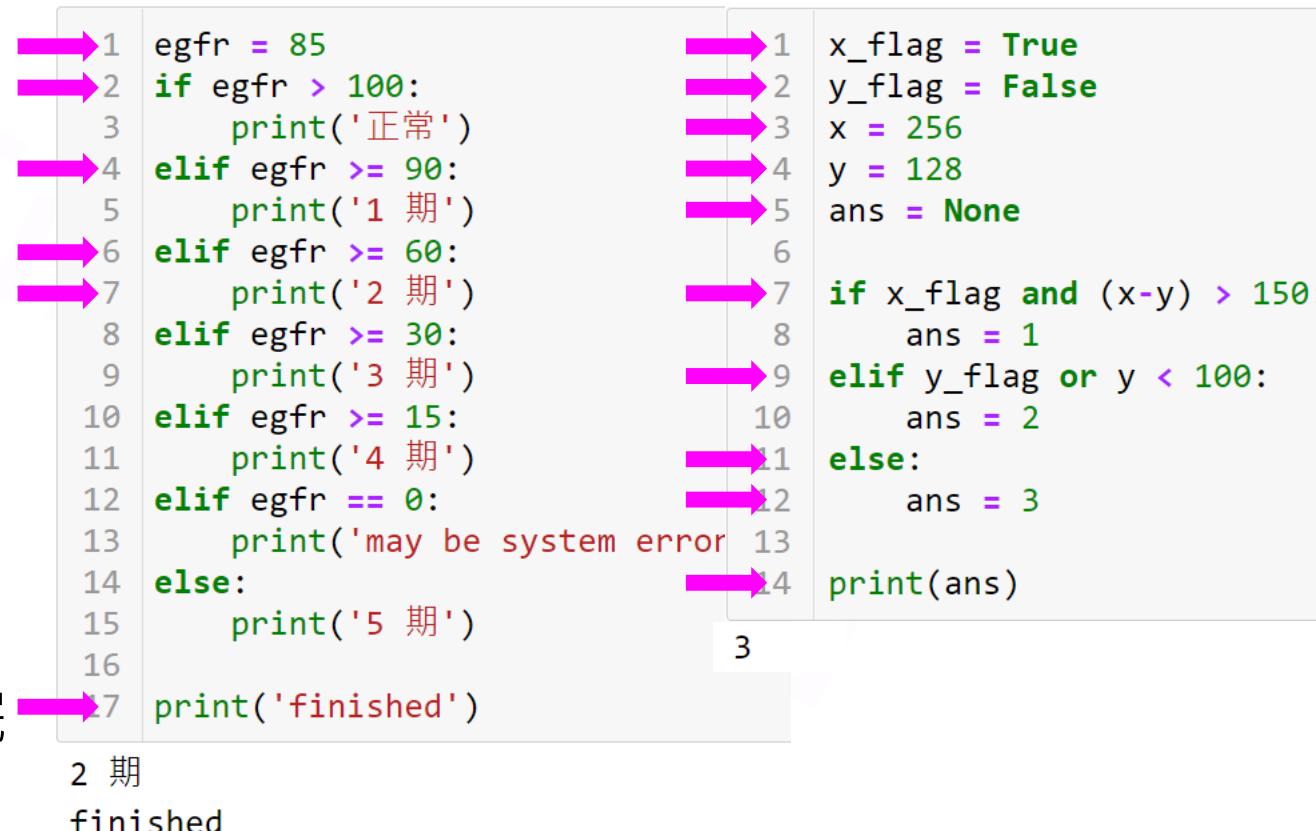
Without Flow Control，若欲依 eGFR 數值 (假設為 85) 告知病患腎臟病分級，直線式的程式如何實現？(7 mins)

eGFR > 100	正常
100 >= eGFR >= 90	1 期
90 > eGFR >= 60	2 期
60 > eGFR >= 30	3 期
30 > eGFR >= 15	4 期
eGFR < 15	5 期

Python Flow Control

if-elif-else

1. 檢查 if 條件是否成立，成立則執行 if 程式段，完畢即離開該 if-elif-else；否則進入下個 elif 或 else，或離開該 if-elif-else
2. 若進入 elif 則檢查條件是否成立，成立則執行 elif 程式段，完畢即離開該 if-elif-else；否則進入下個 elif 或 else，或離開該 if-elif-else
3. 若進入 else 則執行程式段，完畢即離開該 if-elif-else



Python Flow Control

if-elif-else

1. 檢查 if 條件是否成立，成立則執行 if 程式段，完畢即離開該 if-elif-else；否則進入下個 elif 或 else，或離開該 if-elif-else
2. 若進入 elif 則檢查條件是否成立，成立則執行 elif 程式段，完畢即離開該 if-elif-else；否則進入下個 elif 或 else，或離開該 if-elif-else
3. 若進入 else 則執行程式段，完畢即離開該 if-elif-else

限制

	出現次數
if	1
elif	0 ~ N
else	0 ~ 1

Your Turn 2-12

1. 呈上，若 egfr 值為 26，則腎臟病分期為第幾期？(2 mins)

Your Turn 2-12

2. 用 if-elif-else 語法完成下列猜拳演算法並印出 user vs computer 的猜拳結果 (10 mins)

```
from random import choice  
  
computer = choice(['剪刀', '石頭', '布']) ← 隨機取得參數串列其中一個值  
user = '石頭'  
print(f'user {user} vs computer {computer}: ', end='')
```

匯入非內建函數

user 石頭 vs computer 剪刀: user win!

Your Turn 2-12

3. 用 if-elif-else 語法完成完整版猜拳演算法 (user 任意出拳) 並印出 user vs computer 的猜拳結果 (15 mins)

```
from random import choice  
  
user = input('請輸入剪刀、石頭、布：')
```

請輸入剪刀、石頭、布： 剪刀

user 剪刀 vs computer 布: user win!

© 2026 Enos Chou

Python Flow Control

Without Flow Control

```
1 print('This is a book.')
2 print('This is a book.')
3 print('This is a book.')
4 print('This is a book.')
5 print('This is a book.')
6 print('This is a book.'
```

This is a book.
This is a book.

Python Flow Control

for loop 迴圈

- 進入 for 迴圈，檢查序列 (sequence) 是否仍有元素，若有則執行程式段，執行完畢後跳回序列取得下一個元素 (疊代，iteration)；否則離開 for 迴圈

```
1 for i in [0 1 2 3 4 5]:  
2     print('This is a book.')
```

iteration 疊代

Python Flow Control

Without Flow Control

```
400.00176 m  
407.038944 m  
414.704448 m  
422.369952 m  
430.03545600000007 m  
437.70096 m  
445.36646400000006 m  
453.031968 m
```

```
1 # 計算標準跑道各道單圈總距離；最內圈半徑 36.5m，直道 84.39m,  
2 # 跑道寬 1.22m, pi=3.1416, 第一道自內緣+30cm處起算距離,  
3 # 其他道自前一道外緣+20cm處起算距離  
4 pi = 3.1416  
5 r = 36.5  
6 w = 1.22  
7 straight = 84.39  
8  
9 lane1 = 2 * pi * (r + 0.3) + 2 * straight  
10 print(lane1, 'm')  
11 lane2 = 2 * pi * (r + 1 * w + 0.2) + 2 * straight  
12 print(lane2, 'm')  
13 lane3 = 2 * pi * (r + 2 * w + 0.2) + 2 * straight  
14 print(lane3, 'm')  
15 lane4 = 2 * pi * (r + 3 * w + 0.2) + 2 * straight  
16 print(lane4, 'm')  
17 lane5 = 2 * pi * (r + 4 * w + 0.2) + 2 * straight  
18 print(lane5, 'm')  
19 lane6 = 2 * pi * (r + 5 * w + 0.2) + 2 * straight  
20 print(lane6, 'm')  
21 lane7 = 2 * pi * (r + 6 * w + 0.2) + 2 * straight  
22 print(lane7, 'm')  
23 lane8 = 2 * pi * (r + 7 * w + 0.2) + 2 * straight  
24 print(lane8, 'm')
```

Python Flow Control

for loop 迴圈

- 進入 for 迴圈，檢查序列 (sequence) 是否仍有元素，若有則執行程式段，執行完畢後跳回序列取得下一個元素 (疊代，iteration)；否則離開 for 迴圈

```
1 # 計算標準跑道各道單圈總距離；最內圈半徑 36.5m, 直道 84.39m,
2 # 跑道寬 1.22m, pi=3.1416, 第一道自內緣+30cm處起算距離,
3 # 其他道自前一道外緣+20cm處起算距離
4 pi = 3.1416
5 r = 36.5
6 w = 1.22
7 d1, d2 = 0.3, 0.2
8 straight = 84.39
9
10 for i in [0 1 2 3, 4, 5, 6, 7]: ← iteration 疊代
11     d = d1 if i == 0 else d2
12     lane = 2 * pi * (r + i * w + d) + 2 * straight
13     print(lane, 'm')
```

Python Flow Control

for loop 迴圈

- 進入 for 迴圈，檢查序列 (sequence) 是否仍有元素，若有則執行程式段，執行完畢後跳回序列取得下一個元素 (疊代，iteration)；否則離開 for 迴圈

```
1 # 計算標準跑道各道單圈總距離；最內圈半徑 36.5m, 直道 84.39m,
2 # 跑道寬 1.22m, pi=3.1416, 第一道自內緣+30cm處起算距離,
3 # 其他道自前一道外緣+20cm處起算距離
4 pi = 3.1416
5 r = 36.5
6 w = 1.22
7 d1, d2 = 0.3, 0.2
8 straight = 84.39
9
10 for i in [0, 1, 2, 3, 4, 5, 6, 7]:
11     d = d1 if i == 0 else d2
12     lane = 2 * pi * (r + i * w + d) + 2 * straight
13     print(lane, 'm')
```

```
400.00176 m
407.038944 m
414.704448 m
422.369952 m
430.03545600000007 m
437.70096 m
445.36646400000006 m
453.031968 m
```

Your Turn 2-13

s = ['David', 'Joe', 'Jack', 'Lance']
以 for loop 試算字首為 'J' 的有幾人 (7 mins)

Hint

- 字首為字串位置 0 的字元

Python Flow Control

for loop

- 進入 for 迴圈，檢查序列 (sequence) 是否仍有元素，若有則執行程式段，執行完畢後跳回序列取得下一個元素 (疊代，iteration)；否則離開 for 迴圈

- 元素列常見用法

range

```
1 # 計算標準跑道各道單圈總距離；最內圈半徑 36.5m, 直道 84.39m,
2 # 跑道寬 1.22m, pi=3.1416, 第一道自內緣+30cm處起算距離,
3 # 其他道自前一道外緣+20cm處起算距離
4 pi = 3.1416
5 r = 36.5
6 w = 1.22
7 d1, d2 = 0.3, 0.2
8 straight = 84.39
9
10 for i in range(8):
11     d = d1 if i == 0 else d2
12     lane = 2 * pi * (r + i * w + d) + 2 * straight
13     print(lane, 'm')
```

```
400.00176 m
407.038944 m
414.704448 m
422.369952 m
430.03545600000007 m
437.70096 m
445.36646400000006 m
453.031968 m
```

Python Flow Control

for loop

- 進入 for 迴圈，檢查序列 (sequence) 是否仍有元素，若有則執行程式段，執行完畢後跳回序列取得下一個元素 (疊代，iteration)；否則離開 for 迴圈

- 元素列常見用法

`range`

```
1 for i in range(5):  
2     print(i)
```

0
1
2
3
4

```
1 print(range(5))
```

range(0, 5)

```
1 print(list(range(5)))
```

[0, 1, 2, 3, 4]

Your Turn 2-14

1. 以 for loop 建立 list，內容依序包含 1~15 計 15 個整數，
並印出該 list (5 mins)

```
1 x = [1, 2, 3, 4, 5, 6, 7 ,8, 9, 10, 11, 12, 13, 14, 15]  
2 print(x)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

Your Turn 2-14

2. 以 for loop 建立 list，內容依序包含 1~15 計 8 個奇數，
並印出該 list (5 mins)

Hint

奇數可檢查除以 2 的餘數判定

```
1 x = [1, 3, 5, 7, 9, 11, 13, 15]  
2 print(x)
```

[1, 3, 5, 7, 9, 11, 13, 15]

Python Flow Control

for loop

- 進入 for 迴圈，檢查序列 (sequence) 是否仍有元素，若有則執行程式段，執行完畢後跳回序列取得下一個元素 (疊代，iteration)；否則離開 for 迴圈

- 元素列常見用法
`dict`

```
1 s = {1:'a', 2:'b', 3:'c'}
2 for c in s:
3     print(c)
```

1
2
3

```
1 s = {1:'a', 2:'b', 3:'c'}
2 for c in s:
3     print(s[c])
```

a
b
c

Your Turn 2-15

四人跑步公里數如下：

```
s = {'David': 3.2, 'Ray': 6, 'Bill': 12, 'Sam': 9.5} ,
```

試以 for loop 計算四人總共跑了幾公里？誰跑最遠？(7 mins)

Hint

每次比較時紀錄目前為止跑最遠的人

```
s = {'David': 3.2, 'Ray': 6, 'Bill': 12, 'Sam': 9.5}
```

total 30.7 KM, and Bill run farest

Python Flow Control

for loop

- 進入 for 迴圈，檢查序列 (sequence) 是否仍有元素，若有則執行程式段，執行完畢後跳回序列取得下一個元素 (疊代，iteration)；否則離開 for 迴圈

- 元素列常見用法
 - str
 - len
 - enumerate

```
1 s = 'abcde'  
2 for c in s:  
3     print(c)
```

a
b
c
d
e

```
1 s = 'abcde'  
2 for i in range(len(s)):  
3     print(i, s[i])
```

0 a
1 b
2 c
3 d
4 e

```
1 s = 'abcde'  
2 for v in enumerate(s):  
3     print(v)
```

(0, 'a')
(1, 'b')
(2, 'c')
(3, 'd')
(4, 'e')

```
1 s = 'abcde'  
2 for v in enumerate(s):  
3     print(v[0], v[1])
```

0 a
1 b
2 c
3 d
4 e

```
1 s = 'abcde'  
2 for i, v in enumerate(s):  
3     print(i, v)
```

0 a
1 b
2 c
3 d
4 e

Your Turn 2-16

以 for loop 計算 'studying' 是 'How about studying Python' 中的第幾個 word ? **(5 mins)**

Hint

split() first

Python Flow Control

for loop

- 進入 for 迴圈，檢查序列 (sequence) 是否仍有元素，若有則執行程式段，執行完畢後跳回序列取得下一個元素 (疊代，iteration)；否則離開 for 迴圈

- 元素列常見用法
`zip`

```
1 s = [2, 4, 6]
2 for c in s:
3     print(c)
```

2
4
6

```
1 a = ['f', 'g', 'h']
2 for c in a:
3     print(c)
```

f
g
h

```
1 for c in zip(s, a):
2     print(c)
```

(2, 'f')
(4, 'g')
(6, 'h')

```
1 for c in zip(s, a):
2     print(c[0], c[1])
```

2 f
4 g
6 h

```
1 for c, d in zip(s, a):
2     print(c, d)
```

2 f
4 g
6 h

Your Turn 2-17

s 為四個團體名稱，f 為四個團體對應人數，試以 for loop 搭配 zip()
印出每個團體人數，並製作成 dict (5 mins)

```
1 s = ['T-ara', 'BLACKPINK', 'Wonder Girls', 'TWICE']
2 f = [7, 4, 5, 9]
```

```
T-ara 7
BLACKPINK 4
Wonder Girls 5
TWICE 9
{'T-ara': 7, 'BLACKPINK': 4, 'Wonder Girls': 5, 'TWICE': 9}
```

Python Flow Control

有限次數執行

```
1 from random import choice
2 user = '石頭'
3
4 for i in range(5):
5     computer = choice(['剪刀', '石頭', '布'])
6
7     if computer == '剪刀':
8         print('user 石頭 vs computer 剪刀, user win!')
9     elif computer == '石頭':
10        print('user 石頭 vs computer 石頭, tie.')
11    else:
12        print('user 石頭 vs computer 布, user lose.')
```

```
user 石頭 vs computer 布, user lose.
user 石頭 vs computer 剪刀, user win!
user 石頭 vs computer 剪刀, user win!
user 石頭 vs computer 石頭, tie.
user 石頭 vs computer 剪刀, user win!
```

無限次數執行？

無限次數執行，特定條件中止？

Python Flow Control

while loop

1. 進入 while 迴圈先檢查條件列，若滿足則進入程式段，否則離開迴圈
2. 執行程式段，視需要調整影響迴圈條件的變數
3. 程式段執行完畢回到步驟 1 檢查條件列

case 1：有限次數執行

```
1 from random import choice
2 user = '石頭'
3
4 i = 0
5 amount = 5
6 while i < amount: ← 檢查條件是否滿足
7     computer = choice(['剪刀', '石頭', '布'])
8     if computer == '剪刀':
9         print('user 石頭 vs computer 剪刀, user win!')
10    elif computer == '石頭':
11        print('user 石頭 vs computer 石頭, tie.')
12    else:
13        print('user 石頭 vs computer 布, user lose.')
14    i += 1 ← 調整影響迴圈條件的變數
```

```
user 石頭 vs computer 石頭, tie.
user 石頭 vs computer 剪刀, user win!
user 石頭 vs computer 剪刀, user win!
user 石頭 vs computer 剪刀, user win!
user 石頭 vs computer 布, user lose.
```

Python Flow Control

while loop

1. 進入 while 迴圈先檢查條件列，若滿足則進入程式段，否則離開迴圈
2. 執行程式段，視需要調整影響迴圈條件的變數
3. 程式段執行完畢回到步驟 1 檢查條件列

case 2：無限次數執行

```
1 from random import choice
2 user = '石頭'
3
4 while True: ← 檢查條件是否滿足
5     computer = choice(['剪刀', '石頭', '布'])
6     if computer == '剪刀':
7         print('user 石頭 vs computer 剪刀, user win!')
8     elif computer == '石頭':
9         print('user 石頭 vs computer 石頭, tie.')
10    else:
11        print('user 石頭 vs computer 布, user lose.')
```

```
user 石頭 vs computer 布, user lose.
user 石頭 vs computer 布, user lose.
user 石頭 vs computer 剪刀, user win!
user 石頭 vs computer 剪刀, user win!
user 石頭 vs computer 石頭, tie.
user 石頭 vs computer 石頭, tie.
user 石頭 vs computer 布, user lose.
user 石頭 vs computer 剪刀, user win!
user 石頭 vs computer 剪刀, user win!
....
```

Python Flow Control

while loop

1. 進入 while 迴圈先檢查條件列，若滿足則進入程式段，否則離開迴圈
2. 執行程式段，視需要調整影響迴圈條件的變數
3. 程式段執行完畢回到步驟 1 檢查條件列

case 3：無限次數執行，特定條件中止

```
1 from random import choice
2 user = '石頭'
3
4 target = 2
5 win = lose = 0
6 while win < target and lose < target:
7     computer = choice(['剪刀', '石頭', '布'])
8     if computer == '剪刀':
9         win += 1 ← 調整影響迴圈條件的變數
10    print('user 石頭 vs computer 剪刀, user win!')
11 elif computer == '石頭':
12    print('user 石頭 vs computer 石頭, tie.')
13 else:
14     lose += 1 ← 調整影響迴圈條件的變數
15     print('user 石頭 vs computer 布, user lose.')
16     print(f'user win {win} lose {lose}')
17
18 print(f'{user} {if win == 2 else "computer"} win')
```

↑ 檢查條件是否滿足

user 石頭 vs computer 石頭, tie.

user win 0 lose 0

user 石頭 vs computer 剪刀, user win!

user win 1 lose 0

Python Flow Control

while loop vs for loop

1. while loop 需要手動控制疊代；for loop 為全自動疊代
2. while loop 手動控制疊代的特性可以更細膩的控制迴圈，完全涵蓋 for loop 功能

Your Turn 2-18

承 Your Turn 2-17，s 為四個團體名稱，f 為四個團體對應人數，試以 while loop 找出人數為 5 的團 (5 mins)

```
1 | s = ['T-ara', 'BLACKPINK', 'Wonder Girls', 'TWICE']  
2 | f = [7, 4, 5, 9]
```

Python Flow Control

break

- 立刻結束所在 for 或 while loop

```
1 s = ['T-ara', 'BLACKPINK', 'Wonder Girls', 'TWICE']
2 f = [7, 4, 5, 9]
```

```
1 i = 0
2 while i < len(s):
3     print(f'run {i}')
4     if f[i] == 5:
5         print(f'{s[i]} is 5')
6         break
7     i += 1
```

```
run 0
run 1
run 2
Wonder Girls is 5
```

```
1 i = 0
2 for i in range(len(f)):
3     print(f'run {i}')
4     if f[i] == 5:
5         print(f'{s[i]} is 5')
6         break
```

```
run 0
run 1
run 2
Wonder Girls is 5
```

Python Flow Control

continue

- 立刻跳回 for loop 序列 或 while loop 條件列進行下一次檢查

```
1 s = ['T-ara', 'BLACKPINK', 'Wonder Girls', 'TWICE']
2 f = [7, 4, 5, 9]
```

```
1 i = 0
2 for i in range(len(f)):
3     print(f'run {i}')
4     if f[i] == 5:
5         print(f'{s[i]} is 5')
6         break
```

```
run 0
run 1
run 2
Wonder Girls is 5
```

```
1 i = 0
2 found = False
3 while not found and i < len(s):
4     print(f'run {i}')
5     if f[i] != 5:
6         i += 1
7         continue
8     print(f'{s[i]} is 5')
9     found = True
```

```
run 0
run 1
run 2
Wonder Girls is 5
```

Python Flow Control

pass

- 什麼都不做

```
1 s = ['T-ara', 'BLACKPINK', 'Wonder Girls', 'TWICE']
2 f = [7, 4, 5, 9]
```

```
1 i = 0
2 for i in range(len(f)):

File "<ipython-input-15-3072b12c43b7>", line 2
    for i in range(len(f)):
        ^
SyntaxError: unexpected EOF while parsing
```

```
1 i = 0
2 for i in range(len(f)):
    pass
```

Python Flow Control

... (Ellipsis)

- 近似 pass

```
s = ['T-ara', 'BLACKPINK', 'Wonder Girls', 'TWICE']  
f = [7, 4, 5, 9]
```

```
for i in range(len(f)):  
    pass
```

```
for i in range(len(f)):  
    ...
```

Python Comment 註解

Python 註解

Jupyter Notebook 註解

Python Comment

Python 註解

- 讓人理解，但不會被 Python 執行的區塊

- 種類

之後為標準註解

未被 assign 的字串也可以做為註解

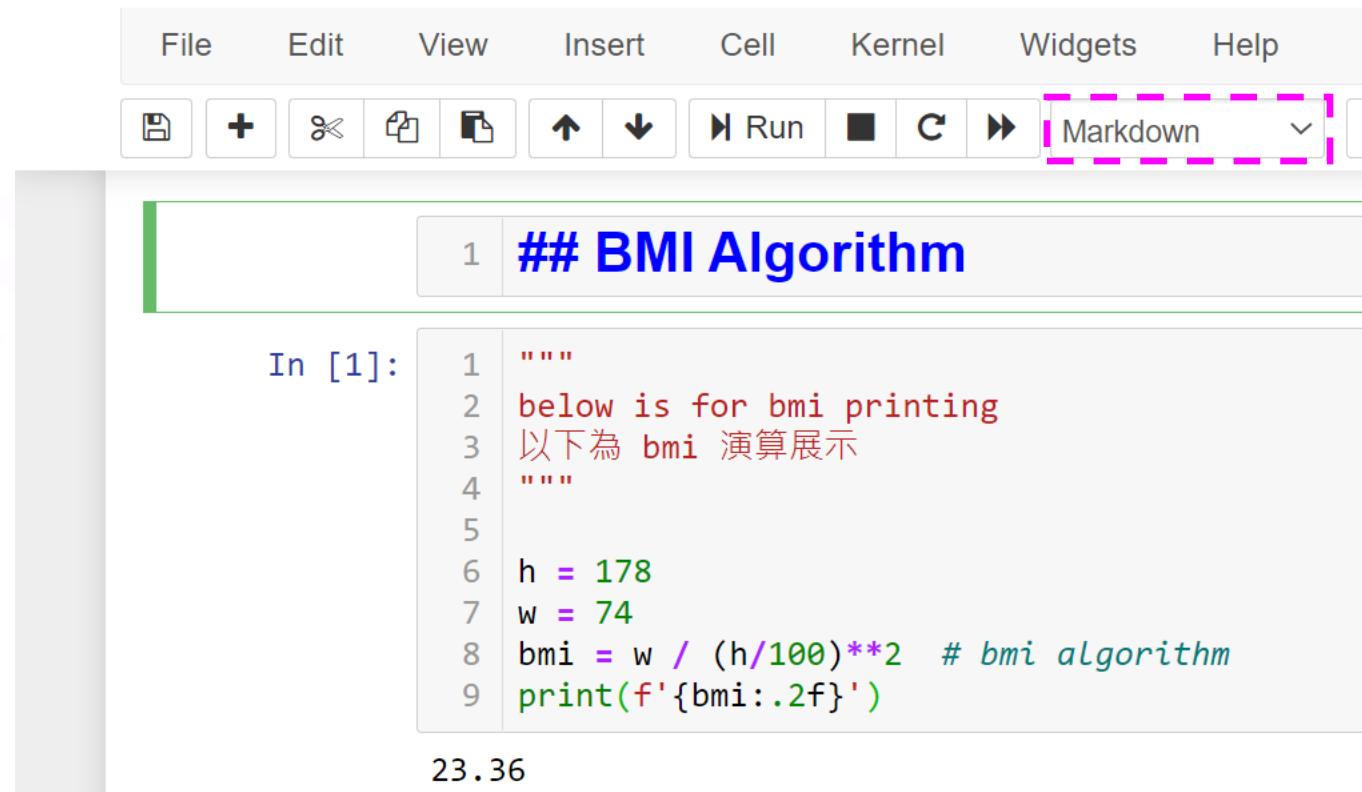
```
1 """
2 below is for bmi printing
3 以下為 bmi 演算展示
4 """
5
6 h = 178
7 w = 74
8 bmi = w / (h/100)**2 # bmi algorithm
9 print(f'{bmi:.2f}')
```

23.36

Python Comment

Jupyter Notebook 註解

- 讓人理解，但不會被 Jupyter Notebook 與 Python 執行的區塊
- 種類
 - ESC + NUMBER 簡單文字註解



The screenshot shows a Jupyter Notebook interface. The top menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, and Run. The main area shows a code cell with the following content:

```
In [1]: 1 ## BMI Algorithm  
         2 """  
         3 below is for bmi printing  
         4 以下為 bmi 演算展示  
         5 """  
         6 h = 178  
         7 w = 74  
         8 bmi = w / (h/100)**2 # bmi algorithm  
         9 print(f'{bmi:.2f}')
```

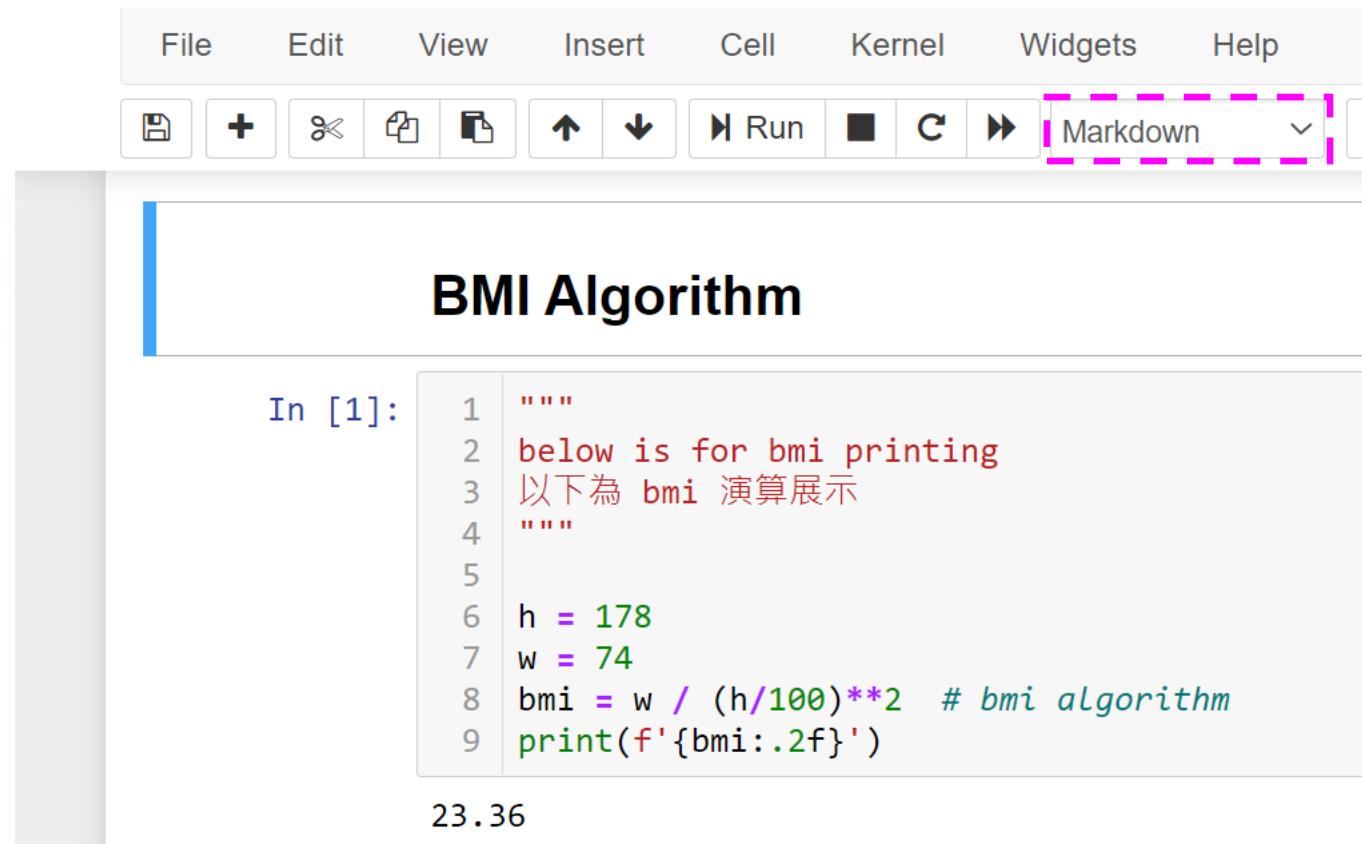
The output of the cell is 23.36.

23.36

Python Comment

Jupyter Notebook 註解

- 讓人理解，但不會被 Jupyter Notebook 與 Python 執行的區塊
- 種類
ESC + NUMBER 簡單文字註解



The screenshot shows a Jupyter Notebook interface. The top navigation bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, and New, followed by Run, Cell, and Kernel. A dropdown menu labeled 'Markdown' is highlighted with a pink dashed box. The main area contains a title 'BMI Algorithm' and a code cell labeled 'In [1]'. The code cell contains the following Python code:

```
1 """  
2 below is for bmi printing  
3 以下為 bmi 演算展示  
4 """  
5  
6 h = 178  
7 w = 74  
8 bmi = w / (h/100)**2 # bmi algorithm  
9 print(f'{bmi:.2f}')
```

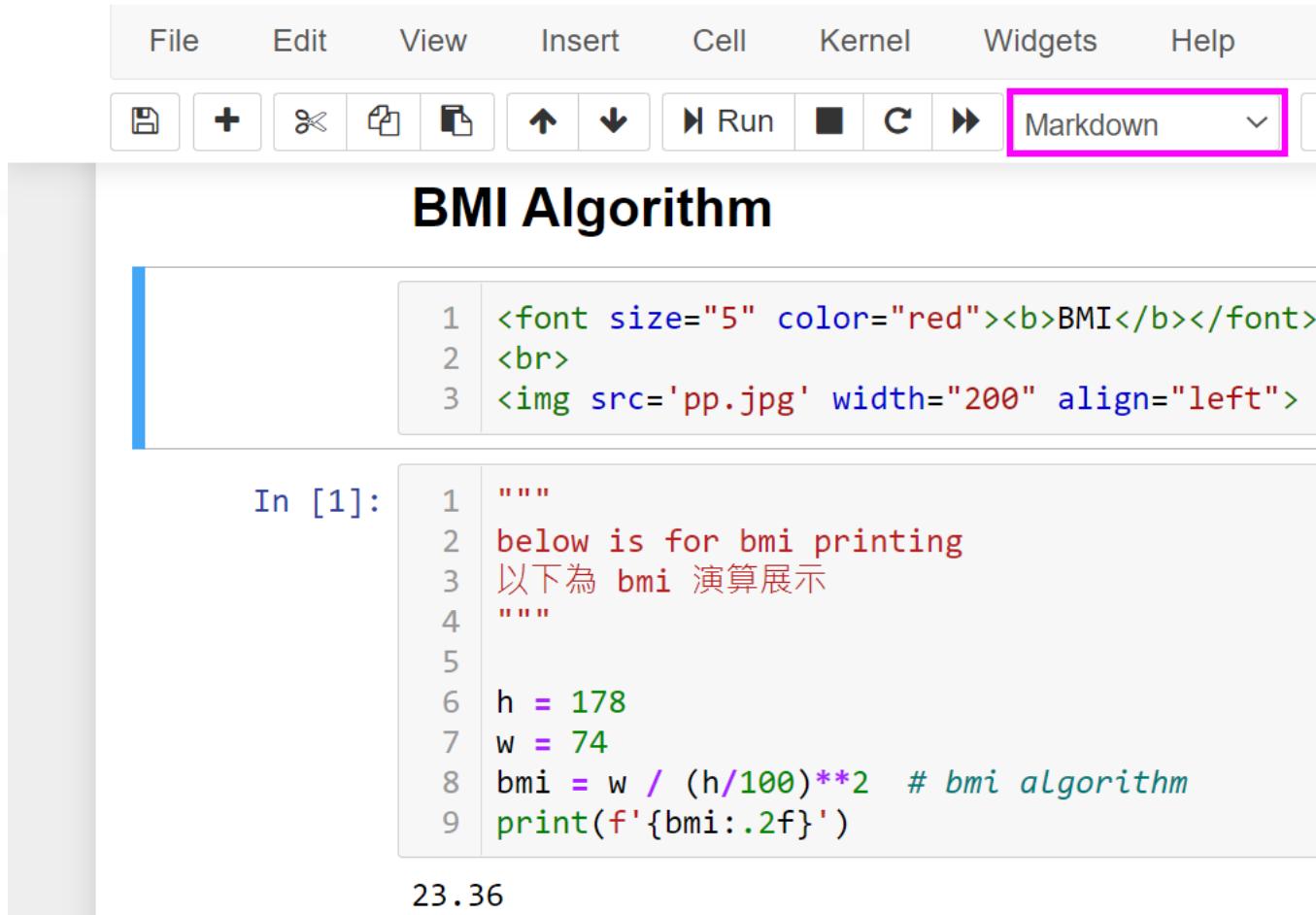
The output of the code cell is '23.36'.

Python Comment

Jupyter Notebook 註解

- 讓人理解，但不會被 Jupyter Notebook 與 Python 執行的區塊
- 種類

Markdown 圖文註解



The screenshot shows a Jupyter Notebook interface. At the top, there's a menu bar with File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with various icons. A pink box highlights the "Markdown" button in the toolbar. The main area has a title "BMI Algorithm". Below it, there's a code cell containing three lines of HTML-like code. Then, under "In [1]:" there's another code cell with nine lines of Python code. The Python code includes comments in red and some explanatory text in Chinese.

```
1 <font size="5" color="red"><b>BMI</b></font>
2 <br>
3 <img src='pp.jpg' width="200" align="left">
```

```
In [1]:
1 """
2 below is for bmi printing
3 以下為 bmi 演算展示
4 """
5
6 h = 178
7 w = 74
8 bmi = w / (h/100)**2 # bmi algorithm
9 print(f'{bmi:.2f}')
```

23.36

Python Comment

Jupyter Notebook 註解

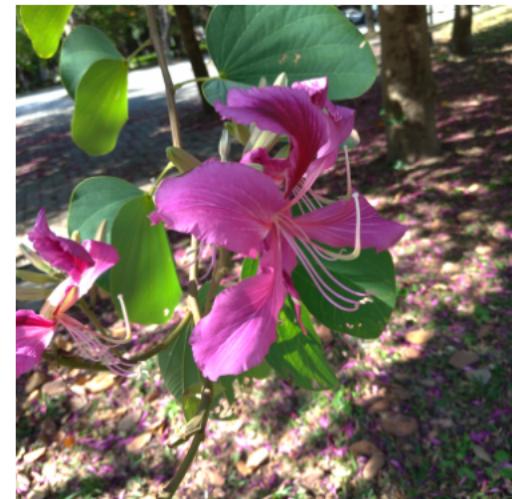
- 讓人理解，但不會被 Jupyter Notebook 與 Python 執行的區塊

- 種類

Markdown 圖文註解

BMI Algorithm

BMI



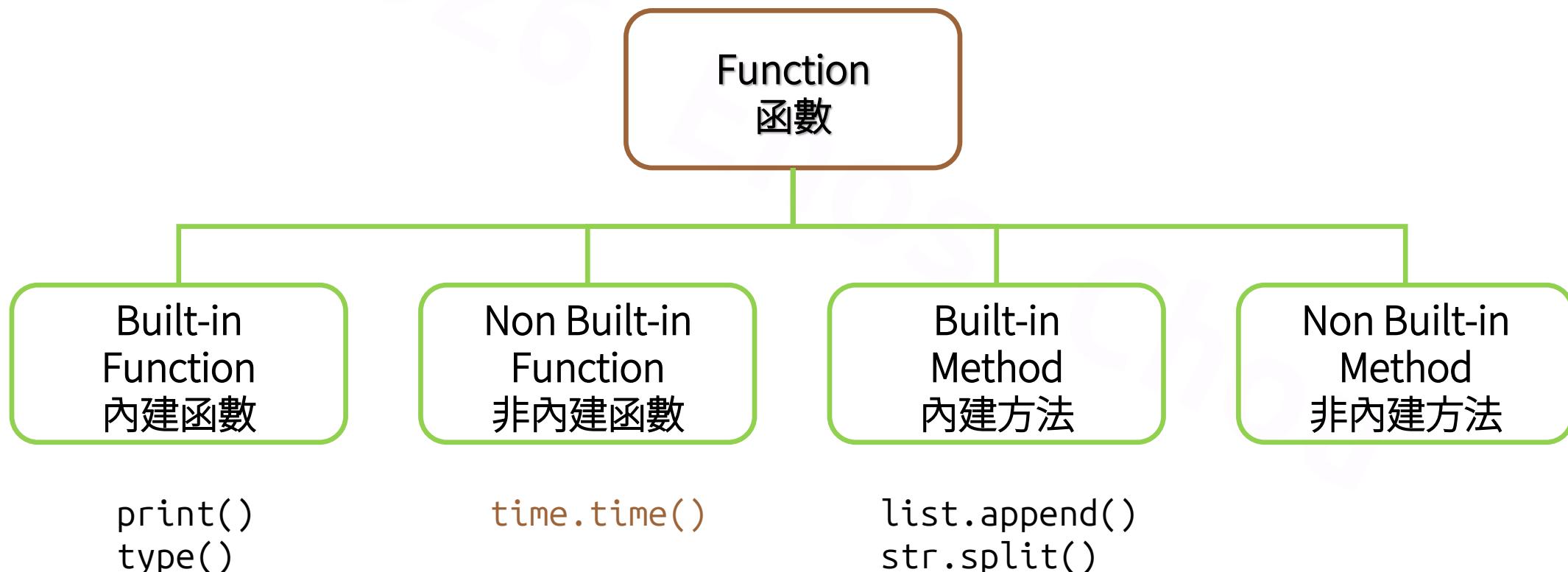
In [1]:

```
1 """
2 below is for bmi printing
3 以下為 bmi 演算展示
4 """
5
6 h = 178
7 w = 74
8 bmi = w / (h/100)**2 # bmi algorithm
9 print(f'{bmi:.2f}')
```

Python 3 Function

Python Import, Python Date-Time, Python File, pip install, Python Function,
Python Variable Scope

Python Non Built-in Functions



Python Import

import 汇入

- 若需要內建函數以外的支援，須以 import 方式將其他模組與函數匯入供後續使用

`import 模組`

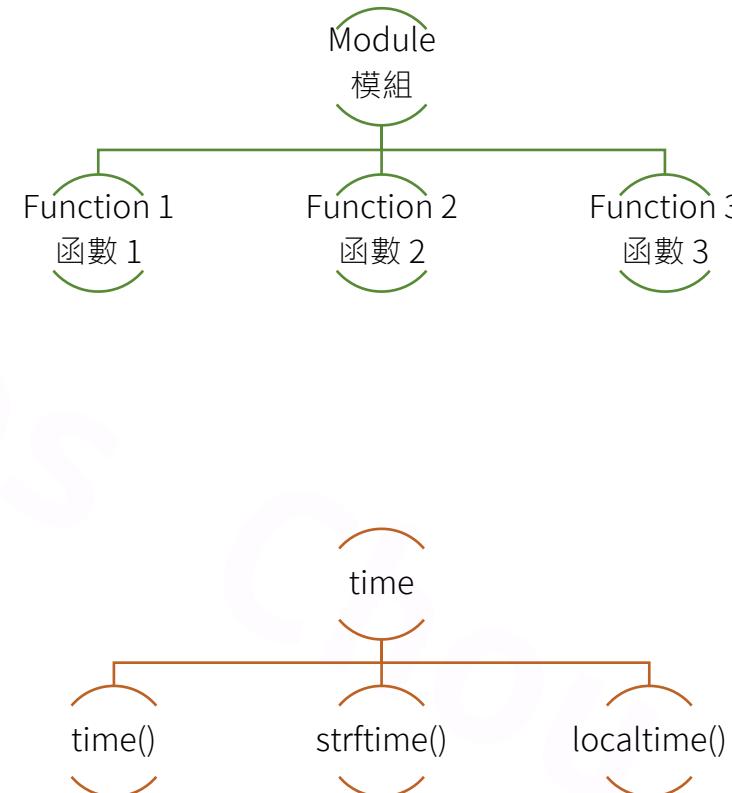
`import 模組 as 簡稱`

`from 模組 import 函數`

`from 模組 import *`

- 代表的意義

- 將系統原來沒有的函數載入供後續使用
- 載入一次就夠



Python Import

import 汇入

- 若需要內建函數以外的支援，須以 import 方式將其他模組與函數匯入供後續使用

import 模組

import 模組 as 簡稱

from 模組 import 函數

from 模組 import *

- 代表的意義

- 將系統原來沒有的函數載入供後續使用
- 載入一次就夠

```
import time

start = time.time()
c = 1
for i in range(1, 100000):
    c *= i
print(f'{time.time() - start} secs')

now = time.strftime('%Y/%m/%d-%H:%M:%S')
print(now)
```

2.316023588180542 secs
2024/08/11-15:59:35

```
import time as t

start = t.time()
c = 1
for i in range(1, 100000):
    c *= i
print(f'{t.time() - start} secs')

now = t.strftime('%Y/%m/%d-%H:%M:%S')
print(now)
```

2.348362922668457 secs
2024/08/11-15:59:42

Python Import

import 汇入

- 若需要內建函數以外的支援，須以 import 方式將其他模組與函數匯入供後續使用

import 模組

import 模組 as 簡稱

from 模組 import 函數

from 模組 import *

- 代表的意義

- 將系統原來沒有的函數載入供後續使用
- 載入一次就夠

```
from time import time, strftime

start = time()
c = 1
for i in range(1, 100000):
    c *= i
print(f'{time() - start} secs')

now = strftime('%Y/%m/%d-%H:%M:%S')
print(now)
```

2.348341941833496 secs
2024/08/11-15:59:51

```
from time import *

start = time()
c = 1
for i in range(1, 100000):
    c *= i
print(f'{time() - start} secs')

now = strftime('%Y/%m/%d-%H:%M:%S')
print(now)
```

2.3241806030273438 secs
2024/08/11-15:59:57

Your Turn 3-1

運用 random 與 time 模組，任選兩種方式建立含有 10,000,000 筆隨機且介於 0 到 1,000 的整數 list，比較哪種方式執行速度最快
(15 mins)

Hint

1. 計算耗用時間
 - 結束時間 – 開始時間
2. 如何得知要用什麼 module？
什麼 function？
 - 以 隨機整數為例，Google 搜尋
“python 隨機”

```
1 '''method 1'''
2
3 import random
4 import time
```

method 1:

- 10.447 secs

Your Turn 3-1

運用 random 與 time 模組，任選兩種方式建立含有 10,000,000 筆隨機且介於 0 到 1,000 的整數 list，比較哪種方式執行速度最快
(15 mins)

Hint

1. 計算耗用時間
 - 結束時間 – 開始時間
2. 如何得知要用什麼 module？
什麼 function？
 - 以 隨機整數為例，Google 搜尋
“python 隨機”

```
1 '''method 2'''
2
3 import random
4 import time
```

method 2:
- 3.123 secs

Python Date-Time

印出日期時間

```
1 import time  
2  
3 x = time.localtime() ← 取得目前日期時間  
4 print(x)
```

```
time.struct_time(tm_year=2020, tm_mon=12, tm_mday=21, tm_hour=14, tm_min=54, tm_sec=36, tm_wday=0, tm_yday=356, tm_isdst=0)
```

Python Date-Time

格式化印出日期時間 (refer to <https://strftime.org/>)

```
1 import time  
2  
3 x = time.localtime()  
4 print(x)
```

```
time.struct_time(tm_year=2020, tm_mon=12, tm_mday=21, tm_hour=14, tm_min=54, tm_sec=36, tm_wday=0, tm_yday=356, tm_isdst=0)
```

```
1 print(time.strftime('%Y%m%d')) ↪ 格式化日期時間
```

```
20201221
```

```
1 print(time.strftime('%Y%m%d %H%M%S', x))
```

```
20201221 145436
```

```
1 print(time.strftime('%Y/%m/%d-%H:%M:%S', x))
```

```
2020/12/21-14:54:36
```

Your Turn 3-2

運用 time.strftime() 印出目前時間，格式如下 (5 mins)

Hint

參閱 <https://strftime.org/>

```
import time
```

```
2024/08/11(0) PM 04:14:04 +0800
```

Python Date-Time

儲存日期時間

```
1 import time  
2  
3 s = '2019-10-12 9:10:10'  
4 print(s)  
5  
6 f = time.strptime(s, '%Y-%m-%d %H:%M:%S') ← 依格式儲存日期時間  
7 print(f)
```

```
2019-10-12 9:10:10  
time.struct_time(tm_year=2019, tm_mon=10, tm_mday=12, tm_hour=9, tm_min=10, tm_sec=10, tm_wday=5, tm_yday=285, tm_isdst=-1)
```

```
1 print(f.tm_year, f.tm_mday)
```

```
2019 12
```

Python File

檔案內容

File Operation

- 開檔、讀檔

開檔 檔名 讀取模式 UTF-8 編碼 file handler

```
1 text = []
2 with open('reading.txt', 'r', encoding='utf-8') as f:
3     x = f.read(3)      ← 讀 3 個字元
4     print(x)
5     text.append(x)
6     x = f.readline() ← 讀到換行符號為止
7     print(x)
8     text.append(x)
9     x = f.read()      ← 讀到檔尾
10    print(x)
11    text.append(x)
12 print(text)
```

```
1 When I was young,
2 I like reading but understand few.
3 I think this is wasting time so I exercise more.
4 程式語言設計。
5
```

Python File

File Operation

- 開檔、讀檔

檔案內容

```
1 When I was young,  
2 I like reading but understand few.  
3 I think this is wasting time so I exercise more.  
4 程式語言設計。  
5
```

```
1 text = []  
2 with open('reading.txt', 'r', encoding='utf-8') as f:  
3     x = f.read(3)  
4     print(x)  
5     text.append(x)  
6     x = f.readline() 輸出  
7     print(x)  
8     text.append(x)  
9     x = f.read()  
10    print(x)  
11    text.append(x)  
12    print(text)
```

Whe
n I was young,
I like reading but understand few.
I think this is wasting time so I exercise more.
程式語言設計。

```
['Whe', 'n I was young,\n', 'I like reading but understand few.\nI think this is wasting  
ercise more.\n程式語言設計。\\n']
```

Python File

File Operation

- 關檔

控制資源使用
用完回收

```
1 text = []
2 with open('reading.txt', 'r', encoding='utf-8') as f:
3     text.append(f.read(3))
4     text.append(f.readline())
5     text.append(f.read())
6 print(text)
7
8 text = []
9 f = open('reading.txt', 'r', encoding='utf-8')
10 text.append(f.read(3))
11 text.append(f.readline())
12 text.append(f.read())
13 f.close() ← 使用完畢後要關檔
14 print(text)
```

```
['Whe', '\n I was young,\n', 'I like reading but understand few.\nI think this is wasting time\n']
['Whe', '\n I was young,\n', 'I like reading but understand few.\nI think this is wasting time\n']
```

Python File

File Operation

- 開檔、寫檔

檔案內容

```
1 When I was young,  
2 I like reading but understand few.  
3 I think this is wasting time so I exercise more.  
4 程式語言設計。  
5 No
```

```
1 text.append('No')  
2 print(text)
```

```
['Whe', 'n I was young,\n', 'I like reading but understand few.\nI think this is wasting time so I ex  
ercise more.\n程式語言設計。', 'n', 'No']
```

寫入模式

```
1 with open('reading_2.txt', 'w', encoding='utf-8') as f:  
2     for line in text:  
3         f.write(line) ← str 寫入
```

Python File

File Operation

- 開檔、寫檔

檔案內容

```
1 When I was young,  
2 I like reading but understand few.  
3 I think this is wasting time so I exercise more.  
4 程式語言設計。  
5 No
```

```
1 text.append('No')  
2 print(text)
```

```
['Whe', 'n I was young,\n', 'I like reading but understand few.\nI think this is wasting time so I ex  
ercise more.\n程式語言設計。', 'No']
```

```
1 with open('reading_2.txt', 'w', encoding='utf-8') as f:  
2     for line in text:  
3         f.write(line)
```

```
→ 1 with open('reading_2.txt', 'w', encoding='utf-8') as f:  
→ 2     f.writelines(text) ← str of list 寫入
```

Python File

檔案內容

File Operation

- 開檔、附加

```
1 with open('reading_2.txt', 'a', encoding='utf-8') as f:  
2     f.write(' excuse!')
```

附加模式

```
1 When I was young,  
2 I like reading but understand few.  
3 I think this is wasting time so I exercise more.  
4 程式語言設計。  
5 No excuse!
```

Python File

以行為單位的資料檔案，常見讀檔形式

```
with open('your_file') as f:
```

a. 以 for 迴圈逐行讀取

```
for row in f: ...
```

b. 以字串的形式一次讀完，得到一個字串，後續以 split 搭配迴圈逐行處理

```
for row in f.read().split('\n'): ...
```

c. 以行為單位一次讀完，得到多行字串 的 list，後續以迴圈逐行處理

```
for row in f.readlines(): ...
```

d. 以 while 迴圈逐行讀取

```
row = f.readline()
```

```
# while row := f.readline(): ...
```

```
while row:
```

```
    ...
```

```
    row = f.readline()
```

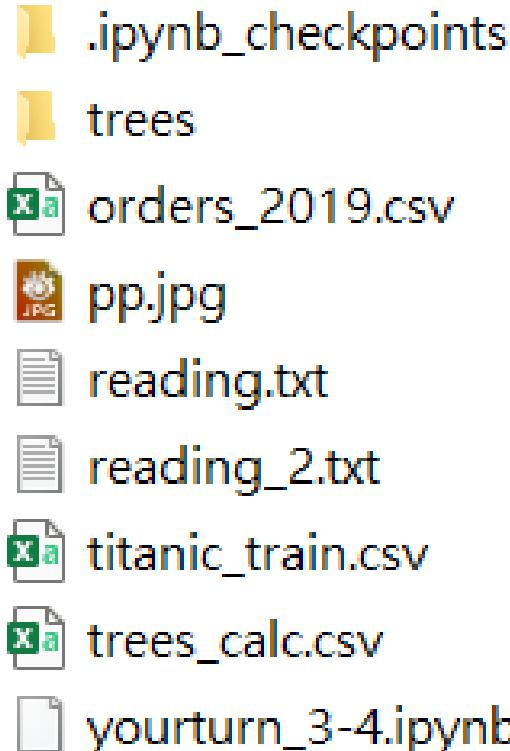
Your Turn 3-3

讀取 “orders.csv” 內容， 將數量 10,000 以上的訂單編號輸出至檔案
(20 mins)

Python File

Directory

`os.listdir(v)` 列出 v 之下所有目錄名稱或檔案名稱；v 預設為當前目錄



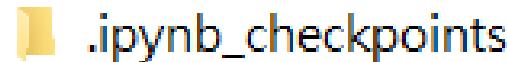
```
1 import os  
2 print(os.listdir())
```

```
['.ipynb_checkpoints', 'orders_2019.csv', 'pp.jpg', 'reading.txt', 'reading_2.txt', 'titanic_train.csv', 'trees', 'trees_calc.csv', 'yourturn_3-4.ipynb']
```

Python File

Directory

`os.path.isdir(v)` 判斷 v 是否為目錄



```
1 print(os.path.isdir('reading.txt'))
```

False



```
1 print(os.path.isdir('trees'))
```

True



Python File

Directory

`os.path.isfile(v)` 判斷 v 是否為檔案



```
1 print(os.path.isfile('reading.txt'))
```

True

```
1 print(os.path.isfile('.ipynb_checkpoints'))
```

False

```
1 print(os.path.isfile('abc.txt'))
```

False

Python File

Directory

`os.path.exists(v)` 判斷 v 是否存在



```
1 print(os.path.exists('reading.txt'))
```

True

```
1 print(os.path.exists('abc.txt'))
```

False

```
1 print(os.path.exists('.ipynb_checkpoints'))
```

True

Python File

Directory

注意：Windows & MAC 不分大小寫；Linux 區分大小寫



```
1 import platform  
2 print(platform.system())  
3 print(os.path.exists('reading.TXT'))
```

Windows

True

```
1 import platform  
2 print(platform.system())  
3 print(os.path.exists('reading.TXT'))
```

Linux

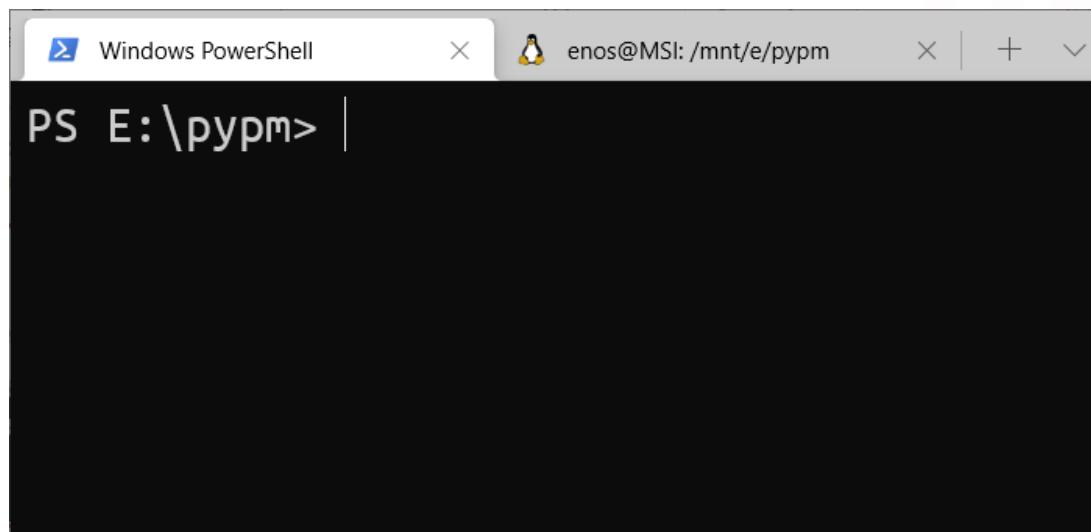
False

Python File

Directory

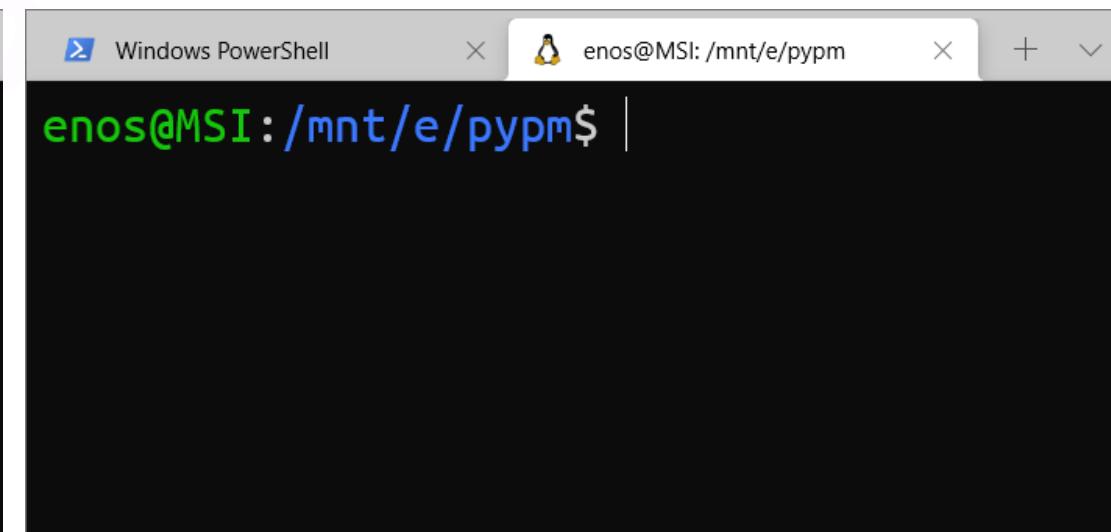
Windows 與 Linux 系統路徑分隔符號不同

Windows



A screenshot of a Windows PowerShell window. The title bar says "Windows PowerShell". The command prompt shows "PS E:\pypm> |". The background is black.

Linux



A screenshot of a Linux terminal window. The title bar says "Windows PowerShell". The command prompt shows "enos@MSI:/mnt/e/pypm\$ |". The background is black.

Python File

Directory

為提升程式碼的相容性 (i.e. Windows 開發的程式也能夠在 Linux 執行)，應該讓 Python 協助處理檔名與路徑的拆解與組合 (i.e. 不建議自行組裝)

```
1 import os
2 import platform
3
4 print(platform.system())
5 print(os.listdir())
6 print('./trees', os.path.isdir('./trees'))
7 print('.\trees', os.path.isdir('.\trees'))
8 print('.\\trees', os.path.isdir('.\\trees'))
9 print(os.path.join('.', 'trees'), os.path.isdir(os.path.join('.', 'trees'))))
```

Windows

```
['.ipynb_checkpoints', 'orders_2019.csv', 'pp.jpg', 'reading.txt', 'reading_2.txt', 'titanic_train.cs
v', 'trees', 'trees_calc.csv', 'yourturn_3-4.ipynb']
./trees True
.      trees False
.\trees True
.\trees True
```

Python File

Directory

為提升程式碼的相容性 (i.e. Windows 開發的程式也能夠在 Linux 執行)，應該讓 Python 協助處理檔名與路徑的拆解與組合 (i.e. 不建議自行組裝)

```
1 import os
2 import platform
3
4 print(platform.system())
5 print(os.listdir())
6 print('./trees', os.path.isdir('./trees'))
7 print('.\trees', os.path.isdir('.\trees'))
8 print('.\\trees', os.path.isdir('.\\trees'))
9 print(os.path.join('.', 'trees'), os.path.isdir(os.path.join('.', 'trees')))
```

```
Linux
['reading_2.txt', 'reading.txt', '.ipynb_checkpoints', 'trees_calc.csv', 'pp.jpg', 'orders_2019.csv',
'yourturn_3-4.ipynb', 'trees', 'titanic_train.csv']
./trees True
.      trees False
.\trees False
./trees True
```

Python File

Directory

`os.path.join(v1, v2, v3, ...)` 組成路徑字串，參數數目不限
`os.getcwd()` 取得目前絕對路徑

```
1 import os
2
3 p = os.path.join('.', 'trees', 'train') # p = os.path.join('trees', 'train')
4 print(p)
5 print(os.listdir(p))
6 p = os.path.join(os.getcwd(), 'trees', 'train')
7 print(p)
8 print(os.listdir(p))

.\trees\train
['AS', 'BJ', 'CC', 'DR', 'FM', 'KE', 'LF', 'MA', 'MI', 'MP', 'PC', 'RR', 'TC', 'TM']
C:\Users\    \Desktop\pydm\trees\train
['AS', 'BJ', 'CC', 'DR', 'FM', 'KE', 'LF', 'MA', 'MI', 'MP', 'PC', 'RR', 'TC', 'TM']
```

Your Turn 3-4

於 trees 目錄中，組出並印出指定圖檔之絕對路徑檔名，同時判斷該檔是否存在 (7 mins)

Hint

由 os.path 中尋找藉由檔名取得絕對路徑的函式

trees > train > FM

					
2019-10-03-10- 57-21_336_56.j	2019-10-03-11- 00-11_336_56.j	2019-10-03-11- 00-41_336_56.j	2019-10-03-11- 03-24_336_56.j	2019-10-03-11- 03-45_336_56.j	
pg	pg	pg	pg	pg	

Python File

Directory

`glob.glob(v)` 列出 v 字串指定路徑中符合指定檔名樣式的所有檔案

```
1 import glob
2 import os
3
4 p = './*.txt'
5 print(p)
6 print(glob.glob(p))
7 p = os.path.join('trees', 'train', '*')
8 print(p)
9 print(glob.glob(p))

./*.txt
['.\\reading.txt', '.\\reading_2.txt']
trees\train\
['trees\\train\\AS', 'trees\\train\\BJ', 'trees\\train\\CC', 'trees\\train\\DR', 'trees\\train\\FM',
'trees\\train\\KE', 'trees\\train\\LF', 'trees\\train\\MA', 'trees\\train\\MI', 'trees\\train\\MP',
'trees\\train\\PC', 'trees\\train\\RR', 'trees\\train\\TC', 'trees\\train\\TM']
```

Your Turn 3-5

計算並分別印出 trees > train 目錄中，14 個子目錄裡的 jpg 檔案個數 (15 mins)

Note

AS: 黑板樹

MA: 苦棟

BJ: 茄冬

MI: 白千層

CC: 樟樹

MP: 水黃皮

DR: 鳳凰木

PC: 阿勃勒

FM: 榕樹

RR: 大王椰子

KE: 台灣欒樹

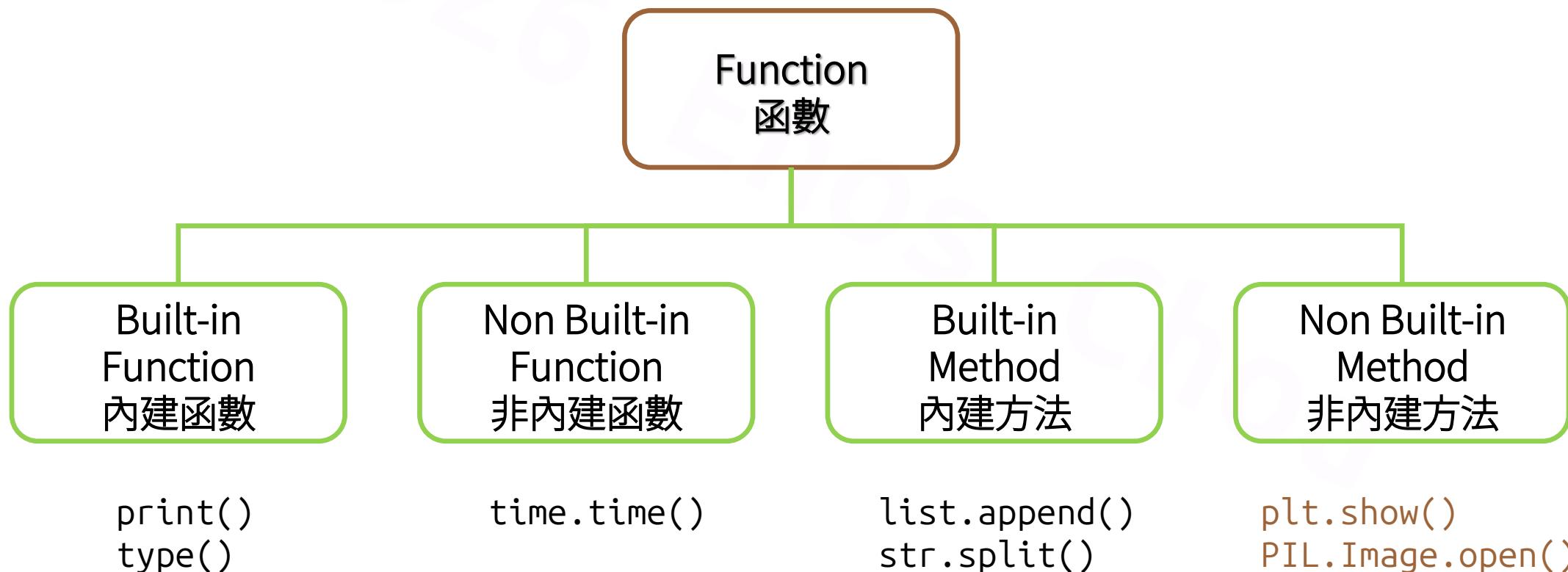
TC: 大葉欒仁

LF: 楓香

TM: 小葉欒仁

AS	96
BJ	164
CC	201
DR	105
FM	295
KE	203
LF	150
MA	56
MI	100
MP	68
PC	105
RR	68
TC	62
TM	170

Python Non Built-in Functions



pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

pip install package_name 安裝套件

pip install package_name --upgrade
更新套件

pip uninstall package_name 移除套件

pip list 現行所有套件資訊

pip install -r requirements.txt 依
需求檔安裝套件

pip show package_name 檢視套件相依性

```
1 from PIL import Image
2
3 img = Image.open('pp.jpg')
4 img.show()
```

```
ModuleNotFoundError: No module named 'PIL'          Traceback (most recent call last)
<ipython-input-1-bfe10e7bb047> in <module>
      1 from PIL import Image
      2
----> 3 img = Image.open('pp.jpg')
      4 img.show()

ModuleNotFoundError: No module named 'PIL'
```

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

`pip install package_name --upgrade`
更新套件

`pip uninstall package_name` 移除套件

`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依需求檔安裝套件

`pip show package_name` 檢視套件相依性

`pip install pillow`

```
1 !pip install pillow
```

Collecting pillow

 Downloading Pillow-9.3.0-1-cp37-cp37m-win_amd64.whl (2.5/2.5 M)

Installing collected packages:

 Successfully installed pillow-9.3.0

Note: you may need to restart the kernel or invoke %reload_ipython_extension to refresh:

```
1 from PIL import Image
2
3 img = Image.open('pp.jpg')
4 img.show()
```



pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

`pip install package_name --upgrade`
更新套件

`pip uninstall package_name` 移除套件

`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依需求檔安裝套件

`pip show package_name` 檢視套件相依性

`pip install pillow`

Anaconda Prompt (miniconda3)

```
(pydm) C:\Users\    \Desktop\pydm>pip install pillow
Collecting pillow
  Downloading Pillow-9.2.0-cp37-cp37m-win_amd64.whl (3.3 MB)
    3.3/3.3 MB 3.2 MB/s eta
Installing collected packages: pillow
Successfully installed pillow-9.2.0

(pydm) C:\Users\    \Desktop\pydm>
```

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

`pip install package_name --upgrade`
更新套件

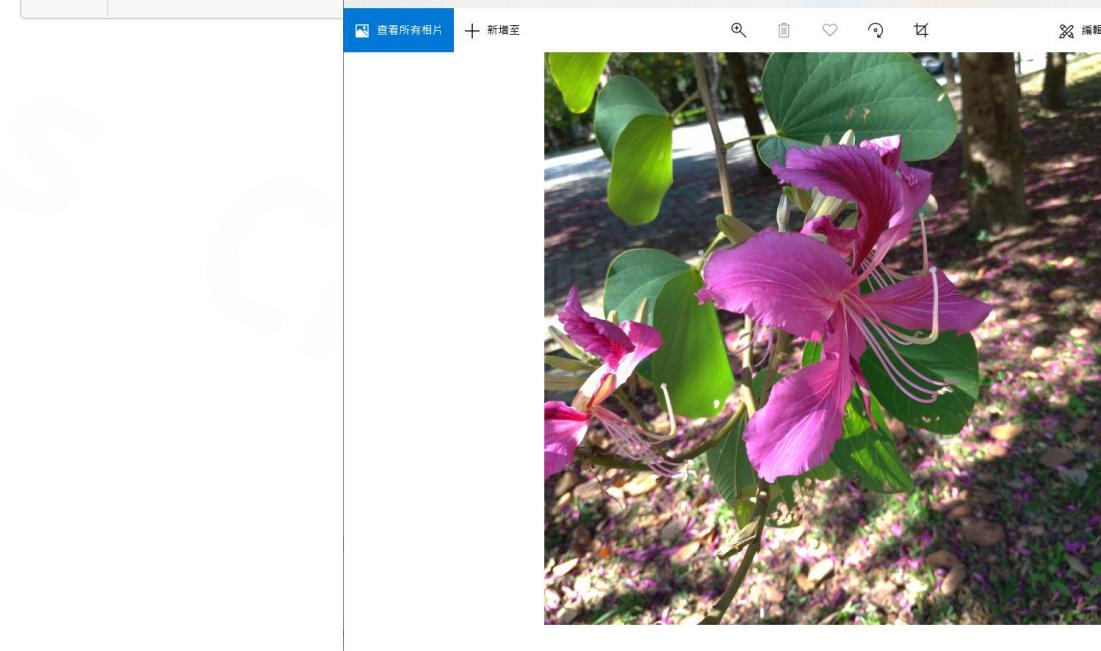
`pip uninstall package_name` 移除套件

`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依
需求檔安裝套件

`pip show package_name` 檢視套件相依性

```
1 from PIL import Image  
2  
3 img = Image.open('pp.jpg')  
4 img.show()
```



pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

`pip install package_name --upgrade`
更新套件

`pip uninstall package_name` 移除套件

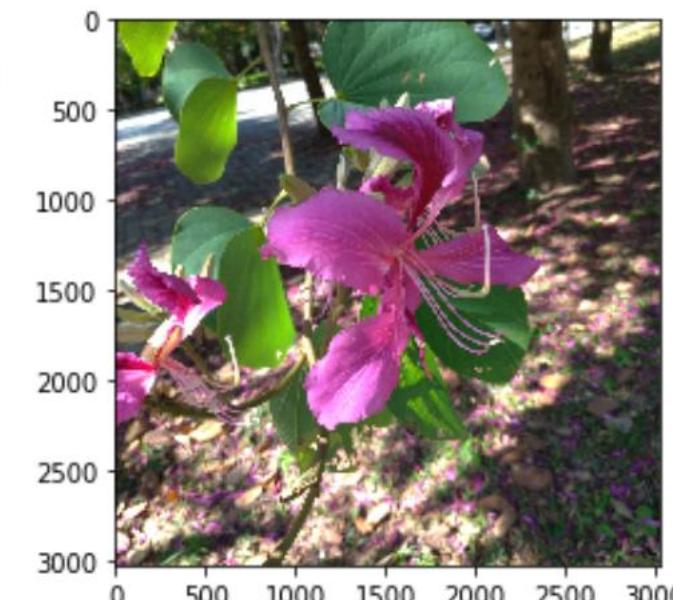
`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依需求檔安裝套件

`pip show package_name` 檢視套件相依性

`pip install matplotlib`

```
1 from PIL import Image  
2 import matplotlib.pyplot as plt  
3  
4 img = Image.open('pp.jpg')  
5 plt.imshow(img)  
6 plt.show()
```



Module vs Package

模組 Module vs 套件 Package

- 一個檔案(xxx.py) 為一個 module，xxx 為 module 名
- 一個 package 含多個 module
- package 可以含 `__init__.py`，用來預先匯入必要之相依套件

Module vs Package

查詢 Package/ Module 安裝位置

```
1 import sys  
2 import PIL  
3 print(sys.modules['PIL'])
```

```
<module 'PIL' from 'C:\\\\Users\\\\    \\\\miniconda3\\\\envs\\\\pydm\\\\lib\\\\site-pac  
ages\\\\PIL\\\\__init__.py'>
```

```
1 import os  
2 print(os.listdir(PIL.__path__[0]))
```

```
['BdfFontFile.py', 'BlpImagePlugin.py', 'BmpImagePlugin.py', 'BufrStubImage  
Plugin.py', 'concr140.dll', 'ContainerIO.py', 'CurImagePlugin.py', 'DcxIma  
gePlugin.py', 'DdsImagePlugin.py', 'EpsImagePlugin.py', 'ExifTags.py', 'fea  
tures.py', 'FitsImagePlugin.py', 'FitsStubImagePlugin.py', 'FliImagePlugin.  
py', 'FontFile.py', 'FpxImagePlugin.py', 'FtexImagePlugin.py', 'GbrImagePlu  
gin.py', 'GdImageFile.py', 'GifImagePlugin.py', 'GimpGradientFile.py', 'Gim  
pPaletteFile.py', 'GribStubImagePlugin.py', 'Hdf5StubImagePlugin.py', 'Icns  
ImagePlugin.py', 'IcoImagePlugin.py', 'Image.py', 'ImageChops.py', 'ImageCm  
s.py', 'ImageColor.py', 'ImageDraw.py', 'ImageDraw2.py', 'ImageEnhance.py',  
'ImageFile.py', 'ImageFilter.py', 'ImageFont.py', 'ImageGrab.py', 'ImageMat
```

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制
 - `pip install package_name` 安裝套件
 - `pip install package_name --upgrade` 更新套件
 - `pip uninstall package_name` 移除套件
 - `pip list` 現行所有套件資訊
 - `pip install -r requirements.txt` 依需求檔安裝套件
 - `pip show package_name` 檢視套件相依性

pip uninstall pillow

```
■ Anaconda Prompt (miniconda3)

(pydm) C:\Users\    \Desktop\pydm>pip uninstall pillow
Found existing installation: Pillow 9.2.0
Uninstalling Pillow-9.2.0:
Would remove:
  c:\users\    \miniconda3\envs\pydm\lib\site-packages\pillow-9.2.0
Proceed (Y/n)? Y
Successfully uninstalled Pillow-9.2.0

(pydm) C:\Users\    \Desktop\pydm>
```

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

`pip install package_name --upgrade`
更新套件

`pip uninstall package_name` 移除套件

`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依需求檔安裝套件

`pip show package_name` 檢視套件相依性

pip list

Package	Version
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
attrs	21.4.0
backcall	0.2.0
beautifulsoup4	4.11.1
bleach	4.1.0
certifi	2022.6.15
cffi	1.15.1
colorama	0.4.5
cycler	0.11.0
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
entrypoints	0.4
fastjsonschema	2.15.1
fonttools	4.35.0
importlib-metadata	4.11.3
importlib-resources	5.2.0
ipykernel	6.9.1
ipython	7.31.1
ipython-genutils	0.2.0
jedi	0.18.1
Jinja2	3.0.3

Your Turn 3-6

安裝 pandas，並以 DataFrame 方式讀取 orders.csv，印出前五筆資料內容 (10 mins)

Hint

```
pandas.read_csv()  
DataFrame.head()
```

pip install

PyPI

<https://pypi.org/>

Help Sponsors Log in Register

Find, install and publish Python packages with the Python Package Index

pillow

Or [browse projects](#)

394,971 projects 3,704,899 releases 6,561,201 files 615,644 users



The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. [Learn about installing packages ↗](#)

Package authors use PyPI to distribute their software. [Learn how to package your Python code for PyPI ↗](#)

pip install

PyPI

<https://pypi.org/>

The screenshot shows the PyPI search interface. The search bar at the top contains the text "pillow". Below the search bar, there is a sidebar titled "Filter by classifier" with several dropdown menu items: Framework, Topic, Development Status, License, Programming Language, Operating System, Environment, Intended Audience, Natural Language, and Typing. The "Framework" option is currently selected. To the right of the sidebar, the text "1,555 projects for 'pillow'" is displayed. A dropdown menu for "Order by" is set to "Relevance". The main content area lists seven project results:

Project Name	Published Date
Pillow 9.2.0	Jul 2, 2022
Pillow3f 0.0.7	Jul 27, 2018
pillow-lut 1.0	Apr 23, 2018
Pillow-PIL 0.1dev	May 20, 2013
pillow-heif 0.6.0	Aug 6, 2022
pillow-affine 0.1	Mar 24, 2020

Each project entry includes a small icon representing its nature (e.g., a cube for Pillow).

pip install

PyPI

<https://pypi.org/>

Screenshot of the PyPI website showing the Pillow project page.

The page title is **Pillow 9.2.0**. A prominent button at the top says **pip install Pillow**.

The **Latest version** is highlighted with a green button.

The release date is **Released: Jul 2, 2022**.

The project description section includes a large Python logo composed of flowers.

The navigation sidebar on the left has three items: **Project description**, **Release history** (which is highlighted with a pink border), and **Download files**.

The project links section includes **Homepage** and **Changelog**.

The footer contains the text **Django**.

pip install

PyPI

<https://pypi.org/>

Navigation

- Project description
- Release history**
- Download files

Project links

- Homepage
- Changelog
- Documentation
- Funding
- Release notes
- Source
- Twitter

Statistics

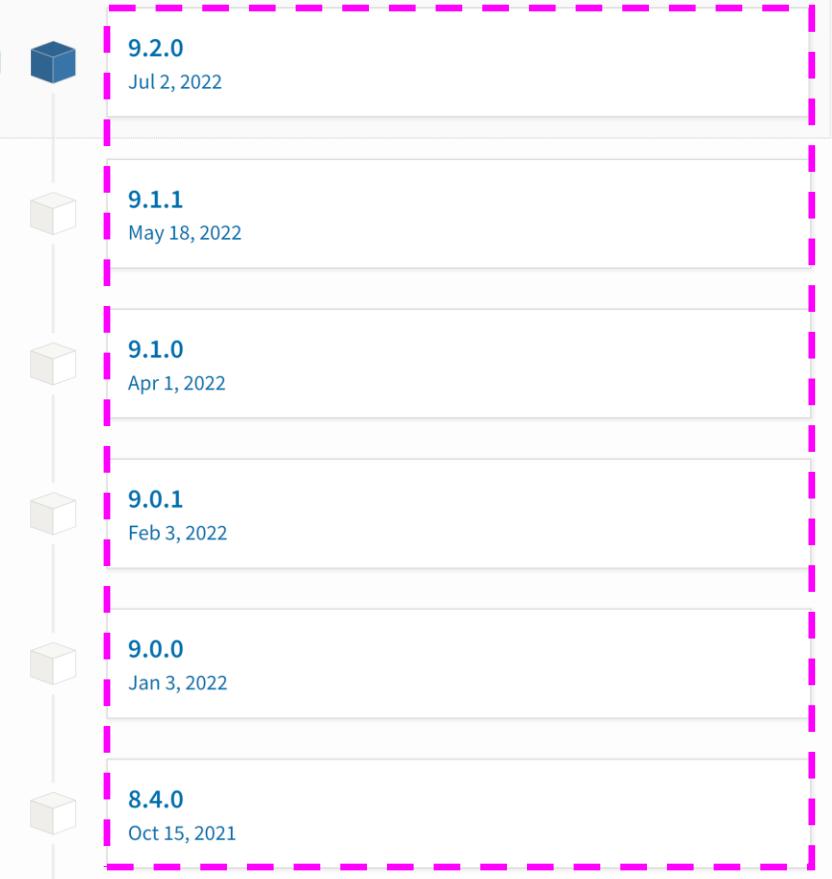
GitHub statistics:

- Stars: 10,052
- Forks: 1,892
- Open issues/PRs: 127

Release history

Release notifications | RSS feed 

THIS VERSION 



Version	Release Date
9.2.0	Jul 2, 2022
9.1.1	May 18, 2022
9.1.0	Apr 1, 2022
9.0.1	Feb 3, 2022
9.0.0	Jan 3, 2022
8.4.0	Oct 15, 2021

pip install

PyPI

<https://pypi.org/>

[Disclaimer \(MPND\)](#)

Programming Language

- o [Python :: 3](#)
- o [Python :: 3 :: Only](#)
- o [Python :: 3.10](#)
- o [Python :: 3.7](#)
- o [Python :: 3.8](#)
- o [Python :: 3.9](#)
- o [Python :: Implementation :: CPython](#)
- o [Python :: Implementation :: PyPy](#)

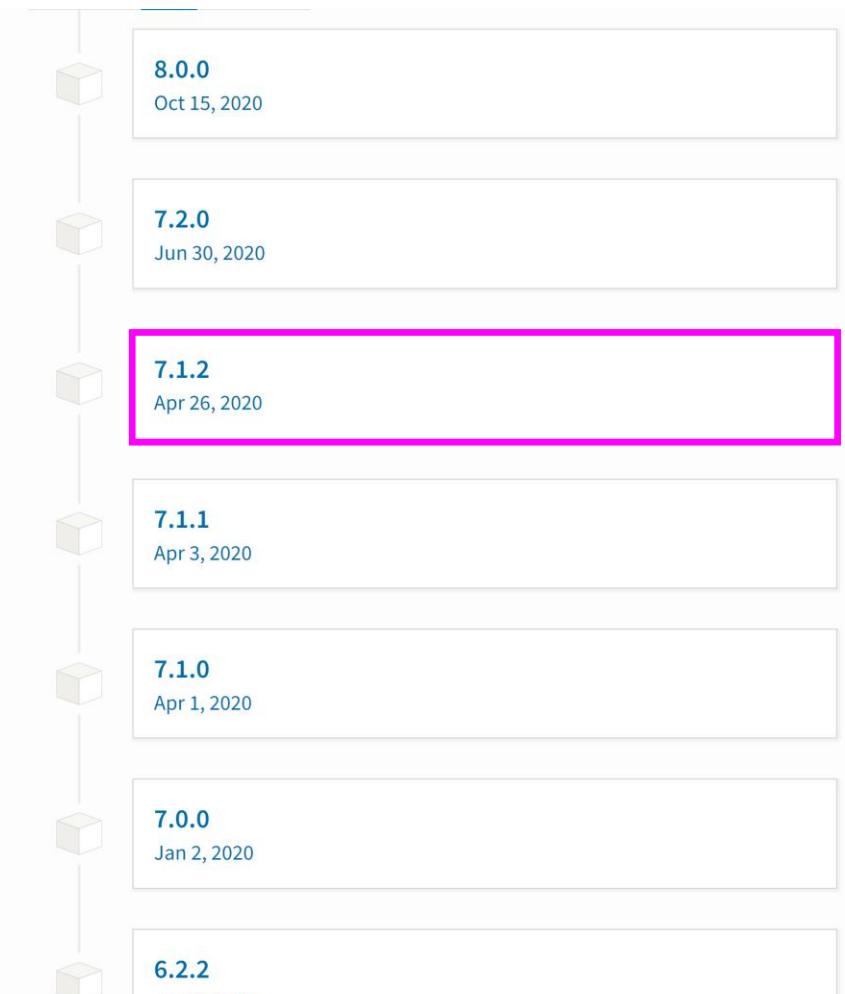
Topic

- o [Multimedia :: Graphics](#)
- o [Multimedia :: Graphics :: Capture :: Digital Camera](#)
- o [Multimedia :: Graphics :: Capture :: Screen Capture](#)
- o [Multimedia :: Graphics :: Graphics Conversion](#)
- o [Multimedia :: Graphics :: Viewers](#)

 **Meta** [↗](#)

Meta is a Visionary sponsor of the Python Software Foundation.

PSF Sponsor · Served ethically



The timeline shows the following releases:

- 8.0.0 - Oct 15, 2020
- 7.2.0 - Jun 30, 2020
- 7.1.2** - Apr 26, 2020 (highlighted with a pink box)
- 7.1.1 - Apr 3, 2020
- 7.1.0 - Apr 1, 2020
- 7.0.0 - Jan 2, 2020
- 6.2.2 - Jan 2, 2020

pip install

PyPI

<https://pypi.org/>

The screenshot shows the PyPI project page for Pillow 7.1.2. The page has a blue header with the Python logo and a search bar. The main title is "Pillow 7.1.2". Below it is a button with the command "pip install Pillow==7.1.2" and a download icon, which is highlighted with a pink box. To the right, there's a red box with a warning icon and the text "Newer version available (9.2.0)". On the far right, it says "Released: Apr 26, 2020". The main content area is titled "Python Imaging Library (Fork)".

Navigation

- [Project description](#) (highlighted)
- [Release history](#)
- [Download files](#)

Project links

- [Homepage](#)
- [Documentation](#)

Project description

Python Imaging Library (Fork)

Pillow is the friendly PIL fork by [Alex Clark and Contributors](#). PIL is the Python Imaging Library by Fredrik Lundh and Contributors. As of 2019, Pillow development is [supported by Tidelift](#).

docs	docs passing
tests	Linux build passing, macOS build error, Windows build passing, Lint passing, Test passing, Test Windows passing, Test Docker passing, codecov 91%
package	DOI 10.5281/zenodo.6788304, lifted!, pypi v9.2.0, downloads 44M/month

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

`pip install package_name --upgrade`
更新套件

`pip uninstall package_name` 移除套件

`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依
需求檔安裝套件

`pip show package_name` 檢視套件相依性

`pip install pillow==7.1.2`

```
Anaconda Prompt (miniconda3)

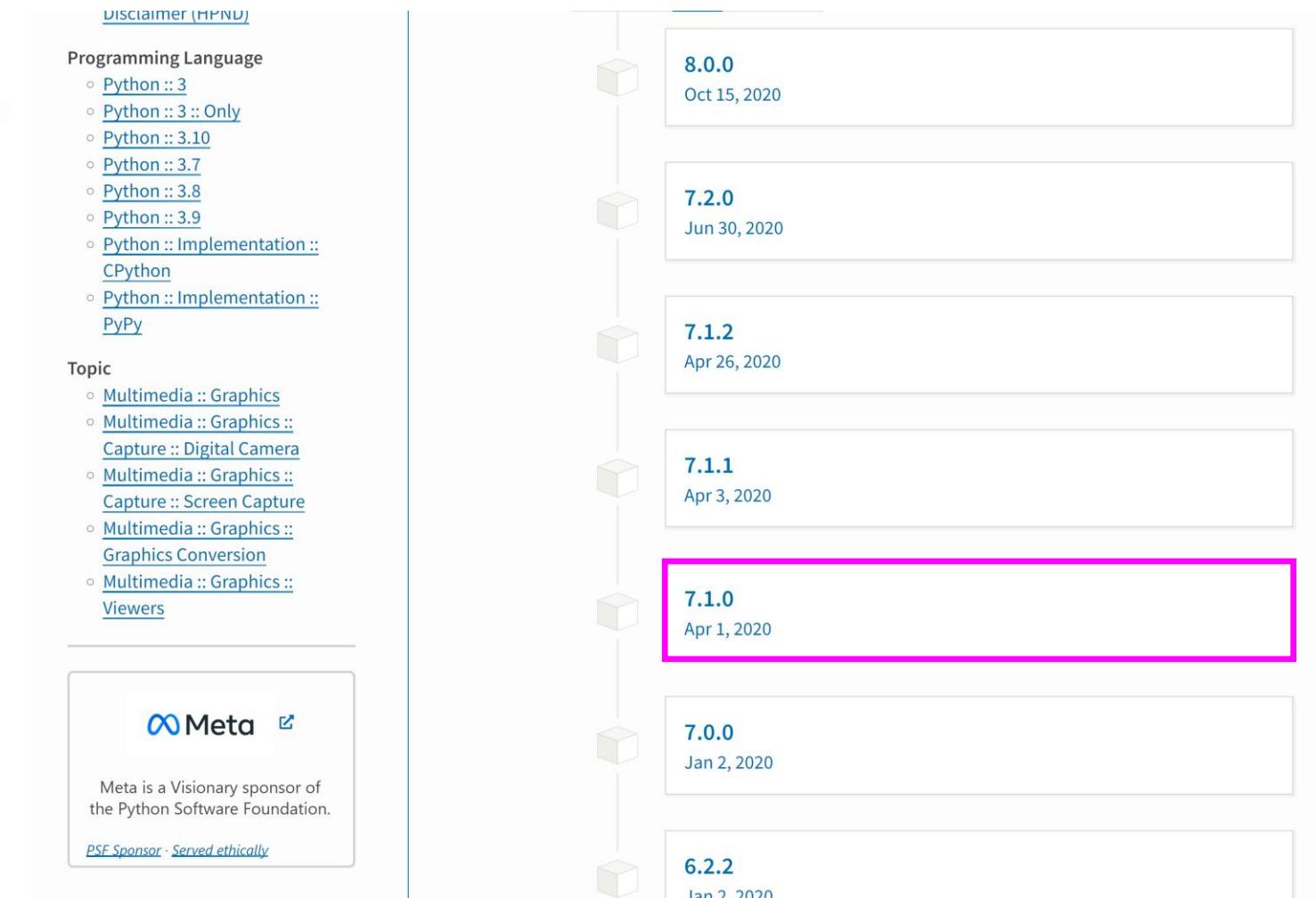
(pydm) C:\Users\...\Desktop\pydm>pip install pillow==7.1.2
Collecting pillow==7.1.2
  Downloading Pillow-7.1.2-cp37-cp37m-win_amd64.whl (2.0 MB)
    ...
    2.0/2.0 MB 377.5 kB/s et
Installing collected packages: pillow
Successfully installed pillow-7.1.2

(pydm) C:\Users\...\Desktop\pydm>
```

pip install

PyPI

<https://pypi.org/>



pip install

PyPI

<https://pypi.org/>

The screenshot shows the PyPI project page for Pillow 7.1.0. At the top, there's a search bar with the placeholder "Search projects" and a magnifying glass icon. To the right are links for "Help", "Sponsors", "Log in", and "Register". The main title is "Pillow 7.1.0" with a subtitle "Python Imaging Library (Fork)". Below the title is a button with the command "pip install Pillow==7.1.0" and a download icon. To the right of the title is a red box containing a warning icon and the text "Newer version available (9.2.0)". On the far right, it says "Released: Apr 1, 2020".

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage
- Documentation

Project description

Python Imaging Library (Fork)

Pillow is the friendly PIL fork by [Alex Clark and Contributors](#). PIL is the Python Imaging Library by Fredrik Lundh and Contributors. As of 2019, Pillow development is [supported by Tidelift](#).

docs	docs passing
tests	Linux build passing, macOS build error, Windows build passing, Lint passing, Test passing, Test Windows passing, Test Docker passing, codecov 91%
package	DOI 10.5281/zenodo.6788304, lifted!, pypi v9.2.0, downloads 44M/month

pip install

PyPI

<https://pypi.org/>

 [Open issues/PRs: 127](#)
View statistics for this project via
[Libraries.io](#), or by using [our](#)
[public dataset on Google](#)
[BigQuery](#)

Meta

License: Historical Permission
Notice and Disclaimer (HPND)
(HPND)

Author: [Alex Clark \(PIL Fork](#)
[Author\)](#)



Requires: Python >=3.5

Maintainers



[aclark](#)



[hugovk](#)



[radarhere](#)



[wiredfool](#)

- Upgraded Apr 1, 2020 | [View](#)
-  [Pillow-7.1.0-cp38-cp38-manylinux1_i686.whl](#) (2.0 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp38
 -  [Pillow-7.1.0-cp38-cp38-macosx_10_10_x86_64.whl](#) (2.2 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp38
 -  [Pillow-7.1.0-cp37-cp37m-win_amd64.whl](#) (2.0 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp37
 -  [Pillow-7.1.0-cp37-cp37m-win32.whl](#) (1.8 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp37
 -  [Pillow-7.1.0-cp37-cp37m-manylinux1_x86_64.whl](#) (2.1 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp37
 -  [Pillow-7.1.0-cp37-cp37m-manylinux1_i686.whl](#) (2.0 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp37
 -  [Pillow-7.1.0-cp37-cp37m-macosx_10_10_x86_64.whl](#) (2.2 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp37
 -  [Pillow-7.1.0-cp36-cp36m-win_amd64.whl](#) (2.0 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp36
 -  [Pillow-7.1.0-cp36-cp36m-win32.whl](#) (1.8 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp36
 -  [Pillow-7.1.0-cp36-cp36m-manylinux1_x86_64.whl](#) (2.1 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp36
 -  [Pillow-7.1.0-cp36-cp36m-manylinux1_i686.whl](#) (2.0 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp36
 -  [Pillow-7.1.0-cp36-cp36m-macosx_10_10_x86_64.whl](#) (2.2 MB [view hashes](#))
Uploaded Apr 1, 2020 | cp36

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

`pip install package_name --upgrade`
更新套件

`pip uninstall package_name` 移除套件

`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依
需求檔安裝套件

`pip show package_name` 檢視套件相依性

`pip install Pillow-7.1.0-xxx-win_amd64.whl`

```
Anaconda Prompt (miniconda3)

(pydm) C:\Users\...\Desktop\pydm>pip install Pillow-7.1.0-cp37-cp37m-win_amd64.whl
Processing c:\users\...\desktop\pydm\pillow-7.1.0-cp37-cp37m-win_amd64.whl
Installing collected packages: Pillow
  Attempting uninstall: Pillow
    Found existing installation: Pillow 7.1.2
    Uninstalling Pillow-7.1.2:
      Successfully uninstalled Pillow-7.1.2
Successfully installed Pillow-7.1.0

(pydm) C:\Users\...\Desktop\pydm>
```

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

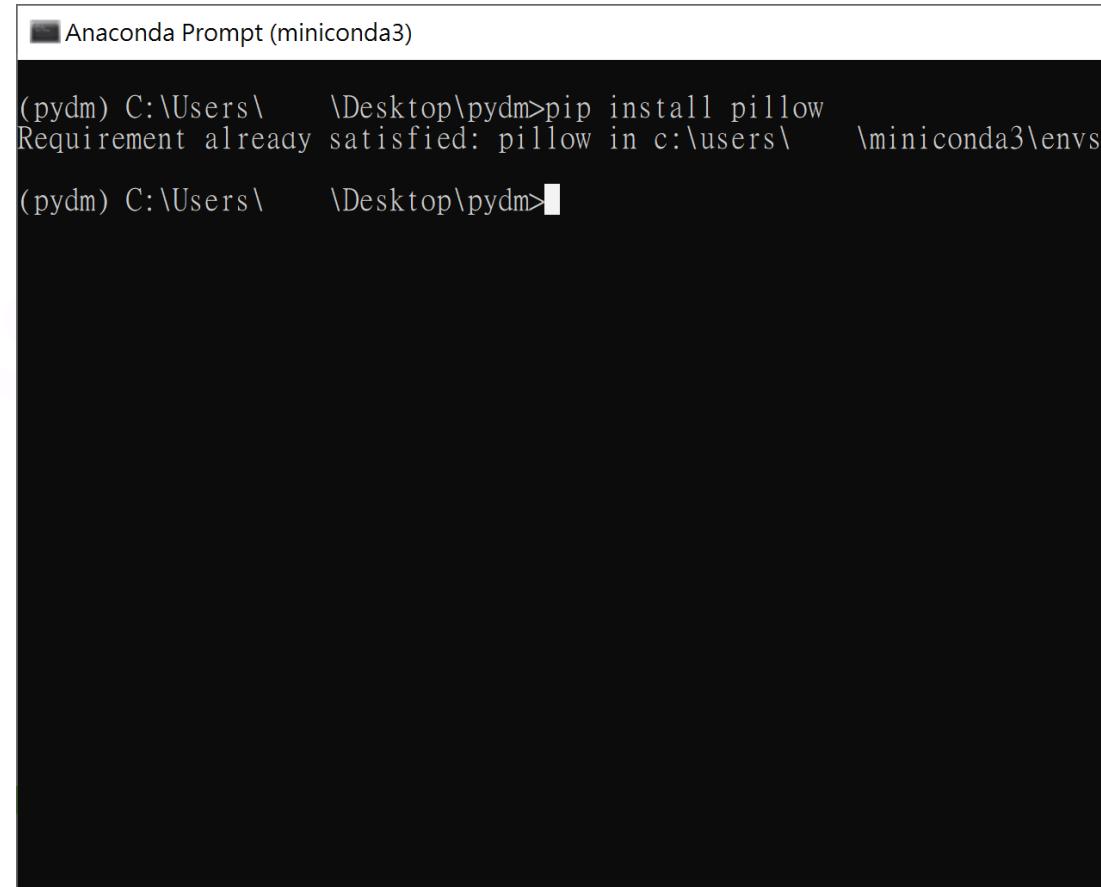
`pip install package_name --upgrade`
更新套件

`pip uninstall package_name` 移除套件

`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依
需求檔安裝套件

`pip show package_name` 檢視套件相依性



A screenshot of the Anaconda Prompt window titled "Anaconda Prompt (miniconda3)". The command entered is "pip install pillow". The output shows that the requirement is already satisfied, indicating that the pillow package is already installed in the miniconda3 environment.

```
Anaconda Prompt (miniconda3)
(pydm) C:\Users\    \Desktop\pydm>pip install pillow
Requirement already satisfied: pillow in c:\users\    \miniconda3\envs\pydm\lib\site-packages
(pydm) C:\Users\    \Desktop\pydm>
```

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

`pip install package_name --upgrade`
更新套件

`pip uninstall package_name` 移除套件

`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依需求檔安裝套件

`pip show package_name` 檢視套件相依性

`pip install pillow --upgrade`

```
Anaconda Prompt (miniconda3)

(pydm) C:\Users\...\Desktop\pydm>pip install pillow --upgrade
Requirement already satisfied: pillow in c:\users\...\miniconda3\envs
Collecting pillow
  Using cached Pillow-9.2.0-cp37-cp37m-win_amd64.whl (3.3 MB)
Installing collected packages: pillow
  Attempting uninstall: pillow
    Found existing installation: Pillow 7.1.0
    Uninstalling Pillow-7.1.0:
      Successfully uninstalled Pillow-7.1.0
  Successfully installed pillow-9.2.0

(pydm) C:\Users\...\Desktop\pydm>
```

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

`pip install package_name --upgrade`
更新套件

`pip uninstall package_name` 移除套件

`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依需求檔安裝套件

`pip show package_name` 檢視套件相依性

edit requirements.txt

```
1 | matplotlib==3.1.0
2 | opencv-python
3 |
```

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

`pip install package_name` 安裝套件

`pip install package_name --upgrade`
更新套件

`pip uninstall package_name` 移除套件

`pip list` 現行所有套件資訊

`pip install -r requirements.txt` 依需求檔安裝套件

`pip show package_name` 檢視套件相依性

`pip install -r requirements.txt`

Anaconda Prompt (miniconda3)

```
(pydm) C:\Users\    \Desktop\pydm>pip install -r requirements.txt
Collecting matplotlib==3.1.0
  Downloading matplotlib-3.1.0-cp37-cp37m-win_amd64.whl (9.1 MB)
    9.1/9.1 MB 253.0 kB/s et
Collecting opencv-python
  Downloading opencv_python-4.6.0.66-cp36-abi3-win_amd64.whl (35.6 MB)
    35.6/35.6 MB 218.9 kB/s
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\    \miniconda3\envs\pydm\lib\site-packages (from matplotlib>=3.1.0->-r requirements.txt (line 1)) (1.4.4)
Requirement already satisfied: numpy>=1.11 in c:\users\    \miniconda3\envs\pydm\lib\site-packages (from matplotlib>=3.1.0->-r requirements.txt (line 1)) (1.21.6)
Requirement already satisfied: cycler>=0.10 in c:\users\    \miniconda3\envs\pydm\lib\site-packages (from matplotlib>=3.1.0->-r requirements.txt (line 1)) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,>=2.1.2,>=2.1.6,>=2.2.0 in c:\users\    \miniconda3\envs\pydm\lib\site-packages (from matplotlib>=3.1.0->-r requirements.txt (line 1)) (3.0.4)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\    \miniconda3\envs\pydm\lib\site-packages (from matplotlib>=3.1.0->-r requirements.txt (line 1)) (2.8.2)
Requirement already satisfied: typing-extensions in c:\users\    \miniconda3\envs\pydm\lib\site-packages (from matplotlib>=3.1.0->-r requirements.txt (line 1)) (4.3.0)
Requirement already satisfied: six>=1.5 in c:\users\    \miniconda3\envs\pydm\lib\site-packages (from matplotlib>=3.1.0->-r requirements.txt (line 1)) (1.16.0)
Installing collected packages: opencv-python, matplotlib
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.5.3
    Uninstalling matplotlib-3.5.3:
      Successfully uninstalled matplotlib-3.5.3
```

pip install

pip

- 當需要外部套件支援 Python 程式開發，需先行安裝外部套件，採用 pip 進行
- pip 為 Python 套件管理機制

pip install package_name 安裝套件

pip install package_name --upgrade
更新套件

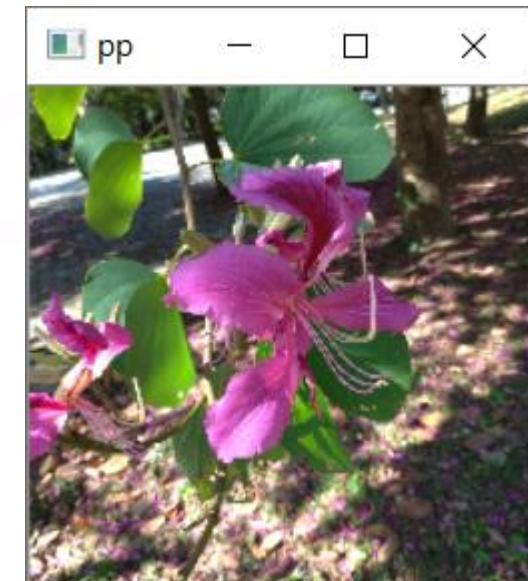
pip uninstall package_name 移除套件

pip list 現行所有套件資訊

pip install -r requirements.txt 依需求檔安裝套件

pip show package_name 檢視套件相依性

```
1 import cv2
2
3 img = cv2.imread('pp.jpg')
4 cv2.imshow('pp', cv2.resize(img, (200, 200)))
5 cv2.waitKey(0)
6 cv2.destroyAllWindows()
```



Your Turn 3-7

1. 建立 requirements.txt 檔，結合 pip 安裝 requests 2.20.1 版、Flask 1.1.1 版 (3 mins)
2. 將 requests 更新到最新版 (3 mins)

Python Function

自訂函式

程式結構化，易讀，易重複使用、
易維護

函式程式段

```
1 def print_something(something, times):
2     for i in range(times):
3         print(f'this is a {something}')
```

```
1 for i in range(3):
2     print('this is a book')
```

```
this is a book
this is a book
this is a book
```

```
1 for i in range(2):
2     print('this is a pencil')
```

```
this is a pencil
this is a pencil
```

```
1 print_something('book', 3)
```

```
this is a book
this is a book
this is a book
```

```
1 print_something('pencil', 2)
```

```
this is a pencil
this is a pencil
```

Python Function

```
1 # 計算以下配速 (分:秒/公里) 半馬耗時 1:53:10
2 k = 21.0975
3 t = '1:38:25'
4
5 t_split = t.split(':')
6 s = 0
7 for i in range(len(t_split)):
8     s += float(t_split[-i-1])*60**i
9
10 p = s / k
11 pace = divmod(p, 60)
12 print(f'{k}K 耗時 {t}, 配速 {int(pace[0])}:{pace[1]:.1f}')
```

21.0975K 耗時 1:38:25, 配速 4:39.9

```
1 # 計算以下配速 (分:秒/公里) 全馬耗時 3:48:31
2 k = 42.195
3 t = '3:48:31'
4
5 t_split = t.split(':')
6 s = 0
7 for i in range(len(t_split)):
8     s += float(t_split[-i-1])*60**i
9
10 p = s / k
11 pace = divmod(p, 60)
12 print(f'{k}K 耗時 {t}, 配速 {int(pace[0])}:{pace[1]:.1f}')
```

42.195K 耗時 3:48:31, 配速 5:24.9

```
1 # 計算以下配速 (分:秒/公里) 全馬耗時 3:48:31
2 k = 42.195
3 t = '3:48:31'
4
5 pace = calc_pace(k, t)
6 print(f'{k}K 耗時 {t}, 配速 {int(pace[0])}:{pace[1]:.1f}')
```

42.195K 耗時 3:48:31, 配速 5:24.9

Python Function

自訂函式

程式結構化，易讀，易重複使用、
易維護

```
1 def calc_pace(k, t='0:0:0'):
2     t_split = t.split(':')
3     s = 0
4     for i in range(len(t_split)):
5         s += float(t_split[-i-1])*60**i
6
7     p = s / k
8     pace = divmod(p, 60)
9
10    return pace
```

```
1 # 計算以下配速 (分:秒/公里) 全馬耗時 3:48:31
2 k = 42.195
3 t = '3:48:31'
4
5 t_split = t.split(':')
6 s = 0
7 for i in range(len(t_split)):
8     s += float(t_split[-i-1])*60**i
9
10 p = s / k
11 pace = divmod(p, 60)
12 print(f'{k}K 耗時 {t}, 配速 {int(pace[0])}:{pace[1]:.1f}')
```

42.195K 耗時 3:48:31, 配速 5:24.9

```
1 # 計算以下配速 (分:秒/公里) 全馬耗時 3:48:31
2 k = 42.195
3 t = '3:48:31'
4
5 pace = calc_pace(k, t)
6 print(f'{k}K 耗時 {t}, 配速 {int(pace[0])}:{pace[1]:.1f}')
```

42.195K 耗時 3:48:31, 配速 5:24.9

Python Function

自訂函式

程式結構化，易讀，易重複使用、
易維護

```
1 def calc_pace(k, t='0:0'):
2     t_split = t.split(':')
3     s = 0
4     for i in range(len(t_split)):
5         s += float(t_split[-i-1])*60**i
6
7     p = s / k
8     pace = divmod(p, 60)
9
10    return pace
```

```
1 # 計算以下配速 (分:秒/公里) 半馬耗時 1:53:10
2 k = 21.0975
3 t = '1:38:25'
4
5 pace = calc_pace(k, t)
6 print(f'{k}K 耗時 {t}, 配速 {int(pace[0])}:{pace[1]:.1f}')
7
8 # 計算以下配速 (分:秒/公里) 萬米耗時 48:11
9 k = 10
10 t = '48:11'
11
12 pace = calc_pace(k, t)
13 print(f'{k}K 耗時 {t}, 配速 {int(pace[0])}:{pace[1]:.1f}')
```

21.0975K 耗時 1:38:25, 配速 4:39.9

10K 耗時 48:11, 配速 4:49.1

```
1 # 計算以下配速 (分:秒/公里) 全馬耗時 3:48:31
2 k = 42.195
3 t = '3:48:31'
4
5 pace = calc_pace(k, t)
6 print(f'{k}K 耗時 {t}, 配速 {int(pace[0])}:{pace[1]:.1f}')
```

42.195K 耗時 3:48:31, 配速 5:24.9

Python Function

自訂函式

程式結構化，易讀，易重複使用、
易維護

函式程式段

函式架構

函式關鍵字 函式名 參數 參數帶預設值

```
1 def calc_pace(k, t='0:0'):
2     t_split = t.split(':')
3     s = 0
4     for i in range(len(t_split)):
5         s += float(t_split[-i-1])*60**i
6
7     p = s / k
8     pace = divmod(p, 60)
9
10    return pace
```

↑
結束函式並回傳結果

Python Function

自訂函式

程式結構化，易讀，易重複使用、
易維護

函式叫用

```
1 def calc_pace(k, t='0:0'):
2     t_split = t.split(':')
3     s = 0
4     for i in range(len(t_split)):
5         s += float(t_split[-i-1])*60**i
6
7     p = s / k
8     pace = divmod(p, 60)
9
10    return pace
```

得到回傳值

```
1 # 計算以下配速(分:秒/公里) 全馬耗時 3:48:31
2 k = 42.195
3 t = '3:48:31' 叫用函式語法
4
5 ➔ pace = calc_pace(k, t)
6 print(f'{k}K 耗時 {t}, 配速 {int(pace[0])}:{pace[1]:.1f}')
```

42.195K 耗時 3:48:31, 配速 5:24.9

Python Function

自訂函式

程式結構化，易讀，易重複使用、
易維護

問題

1. 多個回傳值怎麼處理？
 - ① 多對多個別承接
 - ② 一個變數承接

```
1 def calc_pace(k, t='0:0'):
2     t_split = t.split(':')
3     s = 0
4     for i in range(len(t_split)):
5         s += float(t_split[-i-1])*60**i
6
7     p = s / k
8     pace = divmod(p, 60)
9
10    return pace
11
12 # 計算以下配速 (分:秒/公里) 半馬耗時 1:53:10
13 k = 21.0975
14 t = '1:38:25'
15
16 pace = calc_pace(k, t)
17 print(f'{k}K 耗時 {t}, 配速 {int(pace[0]):{pace[1]:.1f}}')
18
19 # 計算以下配速 (分:秒/公里) 萬米耗時 48:11
20 k = 10
21 t = '48:11'
22
23 p1, p2 = calc_pace(k, t)
24 print(f'{k}K 耗時 {t}, 配速 {int(p1)}:{p2:.1f}')
```

21.0975K 耗時 1:38:25, 配速 4:39.9
10K 耗時 48:11, 配速 4:49.1

Python Function

自訂函式

程式結構化，易讀，易重複使用、
易維護

問題

2. 參數值與預設值的限制？

- ① 呼叫函數時若指定參數名稱，則參數位置可以隨意；
若不指定參數名稱則須依序
填入參數值

```
1 def calc_pace(k, t='0:0'):
2     t_split = t.split(':')
3     s = 0
4     for i in range(len(t_split)):
5         s += float(t_split[-i-1])*60**i
6
7     p = s / k
8     pace = divmod(p, 60)
9
10    return pace
11
12 # 計算以下配速 (分:秒/公里) 半馬耗時 1:53:10
13 k = 21.0975
14 t = '1:38:25'
15
16 pace = calc_pace(t=t, k=k)
17 print(f'{k}K 耗時 {t}, 配速 {int(pace[0])}:{pace[1]:.1f}')
18
19 # 計算以下配速 (分:秒/公里) 萬米耗時 48:11
20 k = 10
21 t = '48:11'
22
23 p1, p2 = calc_pace(k, t)
24 print(f'{k}K 耗時 {t}, 配速 {int(p1)}:{p2:.1f}')
```

21.0975K 耗時 1:38:25, 配速 4:39.9
10K 耗時 48:11, 配速 4:49.1

Python Function

自訂函式

程式結構化，易讀，易重複使用、
易維護

問題

2. 參數值與預設值的限制？
 - ① 呼叫函數時若指定參數名稱，則參數位置可以隨意；若不指定參數名稱則須依序填入參數值
 - ② 預設值只能設計在最後面

```
1 def calc_pace(k=0, t):
2     t_split = t.split(':')
3     s = 0
4     for i in range(len(t_split)):
5         s += float(t_split[-i-1])*60**i
6
7     p = s / k
8     pace = divmod(p, 60)
9
10    return pace
```

```
File "<ipython-input-108-a05ae001f3ec>", line 1
def calc_pace(k=0, t):
^
SyntaxError: non-default argument follows default argument
```

Python Function

自訂函式

程式結構化，易讀，易重複使用、
易維護

問題

3. Function 是否一定要 `return`？

① Function 最終可以不 `return`；若不 `return` 視為 `return None`

Python Function

自訂函式

程式結構化，易讀，易重複使用、
易維護

問題

4. Function 可否有多個 `return`？

Python Function

Basic Function Practice

1. 設計並實作 function，能夠計算指定跑道長度
2. 設計並實作 function，能夠查詢指定名字的身高、體重，與 BMI

Your Turn 3-8

開發一介面輸入 性別、年齡、血清肌酸酐 得知腎臟病分期 (20 mins)

腎絲球過濾率 eGFR 公式：

$$\text{男性} \rightarrow 186 \times \text{血清肌酸酐}^{-1.154} \times \text{年齡}^{-0.203}$$

$$\text{女性} \rightarrow 186 \times \text{血清肌酸酐}^{-1.154} \times \text{年齡}^{-0.203} \times 0.742$$

```
def egfr(creatinine, gender, age):  
    return egfr_value
```

腎臟病分期：

eGFR $\geq 100 \rightarrow 0$ 期

$100 > \text{eGFR} \geq 90 \rightarrow 1$ 期

$90 > \text{eGFR} \geq 60 \rightarrow 2$ 期

$60 > \text{eGFR} \geq 30 \rightarrow 3$ 期

$30 > \text{eGFR} \geq 15 \rightarrow 4$ 期

$15 > \text{eGFR} \rightarrow 5$ 期

```
def stage(egfr_value):  
    return stage_value
```

Your Turn 3-8

開發一介面輸入 性別、年齡、血清肌酸酐 得知腎臟病分期 (20 mins)

Input

性別 (f / m) : m

年齡 : 21

血清肌酸酐(mg/dL) : 1.02

Output

男性，21 歲，血清肌酸酐 1.02 mg/dL ==> eGFR 97.989 mL/min/1.73m²

腎臟病第 1 期

Hint

輸入可採用 `input()`

Python Function

More Function Practice

- 設計並實作 function，能夠發指定張數的撲克牌

Python Variable Scope 變數範圍

```
1 y = 1024
2
3 def fn():
4     h = 500
5     y = h + 100
6     return h
7
8 print(y)
9 yy = fn()
10 print(y)
```

1024
1024

Global y scope

Local y scope
Local h scope

Global y scope

```
1 y = 1024
2
3 def fn():
4     global y
5     h = 500
6     y = h + 100
7     return h
8
9 print(y)
10 yy = fn()
11 print(y)
```

1024
600

Global y scope

Local h scope

- 變數作用範圍：local (區域) 範圍優先
- 寫入 (assign) 狀態，除非變數名稱宣告為 global，否則只於 local 尋找該變數
- 讀取狀態，優先採用 local 同名變數，若不存在於 local 則採用 global 同名變數

Python Variable Scope

```
1 y = 1024
2
3 def fn():
4     h = 500
5     y = h + 100
6     return h
7
8 print(y)
9 yy = fn()
10 print(y)
```

```
1024
1024
```

Global y scope

Local y scope
Local h scope

Global y scope

```
1 y = 1024
2
3 def fn():
4     global y
5     h = 500
6     y = h + 100
7     return h
8
9 print(y)
10 yy = fn()
11 print(y)
12 print(h)
```

```
1024
600
```

NameError

<ipython-input-135-4cdd20a026b3> in <module>

```
10 yy = fn()
11 print(y)
---> 12 print(h)
```

Traceback (most recent call last):

NameError: name 'h' is not defined

Your Turn 3-9

完成下列半馬配速計算 function (5 mins)

Formula :

- 半馬距離 : 21.0975K
- 配速公式 : 分鐘數 / 距離

```
DISTANCE = 21.0975
```

```
def halfpace(minutes):
```

```
    print(f'{halfpace(96.5):.2f}分速')
```

4.57分速

Python 4 Run

Python Class, Python Exception, Run Python

Python Class

Procedure

```
1 import os
2 from glob import glob
3
4 path = 'trees'
5
6 w = os.path.join(path, 'train', 'DR', '*.jpg')
7 count = len(glob(w))
8 print(f"train DR: {count}")
9
10 w = os.path.join(path, 'test', 'AS', '*.jpg')
11 count = len(glob(w))
12 print(f"test AS: {count}")
```

train DR: 105

test AS: 35

Function

```
1 import os
2 from glob import glob
3
4 def count(path, dataset, sp):
5     w = os.path.join(path, dataset, sp, '*.jpg')
6     count = len(glob(w))
7
8     return count
9
10 path = 'trees'
11 print(f"train DR: {count(path, 'train', 'DR')}")
12 print(f"test AS: {count(path, 'test', 'AS')}")
```

train DR: 105

test AS: 35

隱藏風險

- 需在意 `count` 是否為保留字，或重複命名
- 可讀性不佳

Python Class

觀察 list 與 str 叫用 count

```
1 #a = [1, 2, 3, 1, 5, 1, 7]
2 a = list([1, 2, 3, 1, 5, 1, 7])
3 print(a.count(1))
```

3

```
1 #b = 'this is'
2 b = str('this is')
3 print(b.count('is'))
```

2

參考 list 叫用 count 的說明

```
1 help(a)
```

Help on list object:

```
class list(object)
| list() -> new empty list
| list(iterable) -> new list initialized from iterable's items
```

Methods defined here:

```
__add__(self, value, /)
    Return self+value.
```

```
__contains__(self, key, /)
    Return key in self.
```

```
1 help(a)
```

```
| count(...)
```

```
| L.count(value) -> integer -- return number of occurrence
```

Python Class

Class 類別

以物件的角度包裝函數

```
1 #a = [1, 2, 3, 1, 5, 1, 7]
2 a = list([1, 2, 3, 1, 5, 1, 7])
3 print(a.count(1))
```

3

```
1 #b = 'this is'
2 b = str('this is')
3 print(b.count('is'))
```

2

```
1 import os
2 from glob import glob
3
4 class Trees1:
5     _path = None
6
7     def __init__(self, path):
8         self._path = path
9
10    def count(self, dataset, sp):
11        w = os.path.join(self._path, dataset, sp, '*.jpg')
12        count = len(glob(w))
13
14        return count
15
16    t1 = Trees1('trees')
17    print(f"train DR: {t1.count('train', 'DR')}")
18    print(f"test AS: {t1.count('test', 'AS')}")
```

train DR: 105

test AS: 35

Python Class

Class 類別

以物件的角度包裝函數

Note

- object 物件 = instance 實例
- variable 變數 = attribute 屬性

創造 Trees1 類別之實例

```
1 import os
2 from glob import glob
3
4 class Trees1:
5     _path = None
6
7     def __init__(self, path):
8         self._path = path
9
10    def count(self, dataset, sp):
11        w = os.path.join(self._path, dataset, sp, '*.jpg')
12        count = len(glob(w))
13
14        return count
15
16 t1 = Trees1('trees')
17 print(f"train DR: {t1.count('train', 'DR')}")
18 print(f"test AS: {t1.count('test', 'AS')}")
```

train DR: 105
test AS: 35

叫用 t1 的 count 方法

Python Class

抽象概念

t1 →

記憶體位址	內容	變數型態
1002	Trees1	class
1003	_path	
1004	__init__()	
1005	count()	
...		
1200	Trees1	instance
1201	__init__()	
1202	_path	
1203	count()	
...		

Class

```
1 import os
2 from glob import glob
3
4 class Trees1:
5     _path = None
6
7     def __init__(self, path):
8         self._path = path
9
10    def count(self, dataset, sp):
11        w = os.path.join(self._path, dataset, sp, '*.jpg')
12        count = len(glob(w))
13
14        return count
15
16 t1 = Trees1('trees')
17 print(f"train DR: {t1.count('train', 'DR')}")
18 print(f"test AS: {t1.count('test', 'AS')}")
```

train DR: 105

test AS: 35

Python Class

Function

```
1 import os
2 from glob import glob
3
4 def count(path, dataset, sp):
5     w = os.path.join(path, dataset, sp, '*.jpg')
6     count = len(glob(w))
7
8     return count
9
10 path = 'trees'
11 print(f"train DR: {count(path, 'train', 'DR')}")
12 print(f"test AS: {count(path, 'test', 'AS')})")
```

```
train DR: 105
test AS: 35
```

Class

```
1 import os
2 from glob import glob
3
4 class Trees1:
5     _path = None
6
7     def __init__(self, path):
8         self._path = path
9
10    def count(self, dataset, sp):
11        w = os.path.join(self._path, dataset, sp, '*.jpg')
12        count = len(glob(w))
13
14        return count
15
16 t1 = Trees1('trees')
17 print(f"train DR: {t1.count('train', 'DR')}")
18 print(f"test AS: {t1.count('test', 'AS')})")
```

```
train DR: 105
test AS: 35
```

Python Class

Class

以物件的角度包裝函數

Classes in Python

```
1 a = 100
2 b = [1, 2, 3]
3 c = {'f':24, 'g':7}
4 d = 'test'
5 e = str('test')
6 f = list('abc')
7 g = int(100.3)
8 t1 = Trees1('test')
9
10 print(type(a), type(b), type(c), type(d), type(e), type(f), type(g), type(t1))

<class 'int'> <class 'list'> <class 'dict'> <class 'str'> <class 'str'> <class 'list'> <class 'int'>
<class '__main__.Trees1'>
```

Python Class

Class

以物件的角度包裝函數

Note

- _ 與 __ 為兩種 private
- 預設為 public

```
1 import os
2 from glob import glob
3
4 class Trees1:
5     _path = None ←不具強制性的 private
6
7     def __init__(self, path): ←系統 private
8         self._path = path
9
10    def count(self, dataset, sp):
11        w = os.path.join(self._path, dataset, sp, '*.jpg')
12        count = len(glob(w))
13
14    return count
15
16 t1 = Trees1('trees')
17 print(f"train DR: {t1.count('train', 'DR')}")
18 print(f"test AS: {t1.count('test', 'AS')}")
```

train DR: 105

test AS: 35

Python Class

Class

以物件的角度包裝函數

Note

- _ 與 __ 為兩種 private
- 預設為 public
- public vs private

```
1 import os
2 from glob import glob
3
4 class Trees1:
5     _path = None
6
7     def __init__(self, path):
8         if self._isValid(path):
9             self._path = path
10
11    def count(self, dataset, sp):
12        if not (self._isValid(dataset) and self._isValid(sp)):
13            return -1
14        w = os.path.join(self._path, dataset, sp, '*.jpg')
15        count = len(glob(w))
16
17        return count
18
19    def _isValid(self, s):
20        return bool(s) and 2 <= len(s) <= 5
21
22 t1 = Trees1('trees')
23 print(f"train DR: {t1.count('training', 'DR')}")
24 print(f"test AS: {t1.count('test', 'AS')}")
```

train DR: -1

test AS: 35

Python Class

Class Design Guidelines

`__init__()`

- 一次性

`method()`

- 多次使用

```
1 import random
2
3 class Dealer:
4
5     def deal(self):
6         flower = random.choice(['SPADE', 'HEART', 'DIAMOND', 'CLUB'])
7         number = random.choice(range(1, 14))
8         return (flower, number)
```

Python Class

Class Design Guidelines

`__init__()`

- 一次性

`method()`

- 多次使用

```
1 import random
2
3 class Dealer:
4
5     def __init__(self):
6         self._cards = []
7         for i in ['SPADE', 'HEART', 'DIAMOND', 'CLUB']:
8             for j in range(1, 14):
9                 self._cards.append((i, j))
10
11    def deal(self):
12        x = random.choice(self._cards)
13        self._cards.remove(x)
14        return x
```

Python Class

More Class Practice

```
bottle = Bottle(500)
bottle.get_volume()
bottle.fill(300)
bottle.drink(30)
bottle.clean()
```

Your Turn 4-1

承 Your Turn 3-8，開發一介面輸入 性別、年齡、血清肌酸酐 得知
腎臟病分期 (20 mins)

腎絲球過濾率 eGFR 公式：

$$\text{男性} \rightarrow 186 \times \text{血清肌酸酐}^{-1.154} \times \text{年齡}^{-0.203}$$

$$\text{女性} \rightarrow 186 \times \text{血清肌酸酐}^{-1.154} \times \text{年齡}^{-0.203} \times 0.742$$

} method egfr

腎臟病分期：

eGFR $\geq 100 \rightarrow 0$ 期

$100 > \text{eGFR} \geq 90 \rightarrow 1$ 期

$90 > \text{eGFR} \geq 60 \rightarrow 2$ 期

$60 > \text{eGFR} \geq 30 \rightarrow 3$ 期

$30 > \text{eGFR} \geq 15 \rightarrow 4$ 期

$15 > \text{eGFR} \rightarrow 5$ 期

} class CKDStager

} method stage

Your Turn 4-1

承 Your Turn 3-8，開發一介面輸入 性別、年齡、血清肌酸酐 得知
腎臟病分期 (20 mins)

Hint

Class 內以 `self.method_name()` 方式叫用其他 method

Python Class

Class

以物件的角度包裝函數

查詢物件所有屬性與方法

`dir(object)`

```
1 print(dir(t1))
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__len__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_path', 'count']
```

Your Turn 4-2

下列程式執行完畢後報錯，請以 dir 觀察有何問題？(5 mins)

```
1 class test:  
2  
3     def __init__(self):  
4         self.value = 10  
5         self._value = 20  
6         self.__value = 30  
7  
8     def inc(self):  
9         self.value += 1  
10        print(self.value)  
11  
12    def _inc(self):  
13        self._value += 1  
14        print(self._value)  
15  
16    def __inc(self):  
17        self.__value += 1  
18        print(self.__value)
```

```
1 t = test()  
2 t.inc()  
3 t._inc()  
4 t.__inc()
```

```
11  
21
```

```
-----  
AttributeError
```

```
Traceback (most recent call last)
```

```
<ipython-input-2-4a5c73709778> in <module>
```

```
    2 t.inc()  
    3 t._inc()  
----> 4 t.__inc()
```

```
AttributeError: 'test' object has no attribute '__inc'
```

Python Class

Class

以物件的角度包裝函數

預設系統 method
`__str__()`

```
1 print(t1)
```

```
<__main__.Trees1 object at 0x00000160136F4438>
```

```
1 import os
2 from glob import glob
3
4 class Trees1:
5     _path = None
6
7     def __init__(self, path):
8         self._path = path
9
10    def count(self, dataset, sp):
11        w = os.path.join(self._path, dataset, sp, '*.jpg')
12        count = len(glob(w))
13
14        return count
15
16    def __str__(self):
17        return f'Trees1 at {self._path}'
18
19 t1 = Trees1('trees')
20 print(t1)
```

Trees1 at trees

Python Class

Class

以物件的角度包裝函數

查詢物件所有屬性與方法

`dir(object)`

```
1 print(dir(list))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__',
 '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__',
 '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__',
 '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend',
 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

Python Class

Class

以物件的角度包裝函數

非預設系統 method

`__len__()`

```
1 print(len(t1))
```

```
-----  
TypeError  
<ipython-input-23-5cd7d63fc662> in <module>  
----> 1 print(len(t1))
```

```
TypeError: object of type 'Trees1' has no len()
```

```
1 import os  
2 from glob import glob  
3  
4 class Trees1:  
5     _path = None  
6  
7     def __init__(self, path):  
8         self._path = path  
9  
10    def count(self, dataset, sp):  
11        w = os.path.join(self._path, dataset, sp, '*.jpg')  
12        count = len(glob(w))  
13  
14        return count  
15  
16    def __str__(self):  
17        return f'Trees1 at {self._path}'  
18  
19    def __len__(self):  
20        return len(glob(os.path.join(self._path, '*', '*', '*.jpg')))  
21  
22 t1 = Trees1('trees')  
23 print(len(t1))
```

Python Class

Class

以物件的角度包裝函數

類別 vs 實例

類別屬性 vs 實例屬性

```
1 import os
2 from glob import glob
3
4 class Trees1:
5     _path = None
6
7     def __init__(self, path):
8         self._path = path
9
10    def count(self, dataset, sp):
11        w = os.path.join(self._path, dataset, sp, '*.jpg')
12        count = len(glob(w))
13
14        return count
15
16    def __str__(self):
17        return f'Trees1 at {self._path}'
18
19    def __len__(self):
20        return len(glob(os.path.join(self._path, '*', '*', '*.jpg')))
21
22 t1 = Trees1('trees')
23 print(t1._path)
```

trees

```
1 print(Trees1._path) # print(t1.__class__._path)
```

None

Python Class

抽象概念

t1 →

記憶體位址	內容	變數型態
1002	Trees1	class
1003	_path	
1004	__init__()	
1005	count()	
...		
1200	Trees1	instance
1201	__init__()	
1202	_path	
1203	count()	
...		

```
1 import os
2 from glob import glob
3
4 class Trees1:
5     _path = None
6
7     def __init__(self, path):
8         self._path = path
9
10    def count(self, dataset, sp):
11        w = os.path.join(self._path, dataset, sp, '*.jpg')
12        count = len(glob(w))
13
14        return count
15
16    def __str__(self):
17        return f'Trees1 at {self._path}'
18
19    def __len__(self):
20        return len(glob(os.path.join(self._path, '*', '*', '*.jpg)))
21
22 t1 = Trees1('trees')
23 print(t1._path)
```

trees

```
1 print(Trees1._path) # print(t1.__class__._path)
```

None

Python Class

Class

以物件的角度包裝函數

類別 vs 實例

類別屬性 vs 實例屬性

類別方法 vs 實例方法

Python Class

Class

以物件的角度包裝函數

方法 Method

(instance) method: 實例方法

static method: 靜態方法

class method: 類別方法

Python Class

Class

以物件的角度包裝函數

方法 Method

(instance) method: 實例方法

帶入 instance 指標，慣例命名為 self

static method: 靜態方法

class method: 類別方法

Python Class

Class

以物件的角度包裝函數

方法 Method

(instance) method: 實例方法

static method: 靜態方法

class method: 類別方法

```
1 import os
2 from glob import glob
3
4 class Trees1:
5     _path = None
6
7     def __init__(self, path):
8         self._path = path
9
10    def count(self, dataset, sp):
11        w = os.path.join(self._path, dataset, sp, '*.jpg')
12        count = len(glob(w))
13
14        return count
15
16 t1 = Trees1('trees')
17 print(f"train DR: {t1.count('train', 'DR')}")
18 print(f"test AS: {t1.count('test', 'AS')}")
```

train DR: 105
test AS: 35

Python Class

Class

以物件的角度包裝函數

方法 Method

(instance) method: 實例方法

static method: 靜態方法

`@staticmethod`

class method: 類別方法

不帶入 `instance` 與 `class` 指標

Python Class

Class

以物件的角度包裝函數

方法 Method

(instance) method: 實例方法

static method: 靜態方法

class method: 類別方法

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 print(f"train DR: {Trees2.count('trees', 'train', 'DR')}")
13 print(f"test AS: {Trees2.count('trees', 'test', 'AS')}")
```

train DR: 105

test AS: 35

Python Class

抽象概念

記憶體位址	內容	變數型態
1002	Trees2	class
1003	count()	

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 print(f"train DR: {Trees2.count('trees', 'train', 'DR')}")
13 print(f"test AS: {Trees2.count('trees', 'test', 'AS')}")
```

```
train DR: 105
test AS: 35
```

Python Class

Class

以物件的角度包裝函數

方法 Method

(instance) method: 實例方法

static method: 靜態方法

class method: 類別方法

`@classmethod`

帶入 class 指標，慣例命名為 `cls`

Python Class

Class

以物件的角度包裝函數

方法 Method

(instance) method: 實例方法

static method: 靜態方法

class method: 類別方法

```
1 import os
2 from glob import glob
3
4 class Trees3:
5     _path = None
6
7     @classmethod
8     def __init__(cls, path):
9         cls._path = path
10
11    @classmethod
12    def count(cls, dataset, sp):
13        w = os.path.join(cls._path, dataset, sp, '*.jpg')
14        count = len(glob(w))
15
16        return count
17
18 t3 = Trees3('trees')
19 print(f"train DR: {t3.count('train', 'DR')}")
20 print(f"test AS: {Trees3.count('test', 'AS')}")
```

train DR: 105

test AS: 35

```
1 t33 = Trees3('t')
2 print(f"train DR: {t33.count('train', 'DR')}")
3 print(f"test AS: {Trees3.count('test', 'AS')}")
```

train DR: 0

test AS: 0

```
1 print(t3._path, t33._path, Trees3._path)
```

t t t

Python Class

抽象概念

記憶體位址	內容	變數型態
1002	Trees3	class
1003	_path	
1004	__init__()	
1005	count()	
...		
1200	Trees3	instance
1201	_path	
1202	__init__()	
1203	count()	
...		
1900	Trees3	instance
1901	_path	

t3 →

t33 →

```
1 import os
2 from glob import glob
3
4 class Trees3:
5     _path = None
6
7     @classmethod
8     def __init__(cls, path):
9         cls._path = path
10
11    @classmethod
12    def count(cls, dataset, sp):
13        w = os.path.join(cls._path, dataset, sp, '*.jpg')
14        count = len(glob(w))
15
16        return count
17
18 t3 = Trees3('trees')
19 print(f"train DR: {t3.count('train', 'DR')}")
20 print(f"test AS: {Trees3.count('test', 'AS')})")
```

train DR: 105

test AS: 35

```
1 t33 = Trees3('t')
2 print(f"train DR: {t33.count('train', 'DR')}")
3 print(f"test AS: {Trees3.count('test', 'AS')})")
```

train DR: 0

test AS: 0

```
1 print(t3._path, t33._path, Trees3._path)
```

t t t

Your Turn 4-3

承 Your Turn 4-1，開發一介面輸入 性別、年齡、血清肌酸酐 得知
腎臟病分期 (5 mins)

腎絲球過濾率 eGFR 公式：

$$\text{男性} \rightarrow 186 \times \text{血清肌酸酐}^{-1.154} \times \text{年齡}^{-0.203}$$

$$\text{女性} \rightarrow 186 \times \text{血清肌酸酐}^{-1.154} \times \text{年齡}^{-0.203} \times 0.742$$

腎臟病分期：

eGFR $\geq 100 \rightarrow 0$ 期

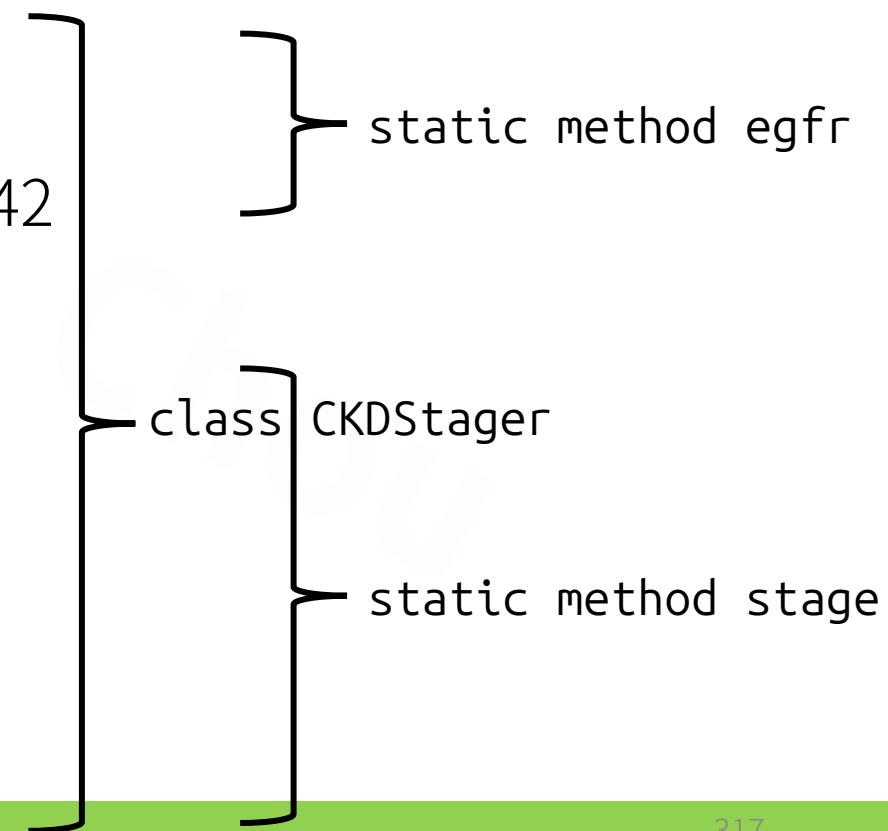
$100 > \text{eGFR} \geq 90 \rightarrow 1$ 期

$90 > \text{eGFR} \geq 60 \rightarrow 2$ 期

$60 > \text{eGFR} \geq 30 \rightarrow 3$ 期

$30 > \text{eGFR} \geq 15 \rightarrow 4$ 期

$15 > \text{eGFR} \rightarrow 5$ 期



Python Exception

例外處理

try-except

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 print(f"train DR: {Trees2.count('trees', 'train', 'DR')})
```

train DR: 105

Python Exception

例外處理

try-except

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 print(f"train DR: {Trees2.count('trees', None, 'DR')})")
```

```
-----  
TypeError                                     Traceback (most recent call last)
<ipython-input-2-016e7e234c7f> in <module>
      10     return count
      11
---> 12 print(f"train DR: {Trees2.count('trees', None, 'DR')})"

<ipython-input-2-016e7e234c7f> in count(path, dataset, sp)
      5     @staticmethod
      6     def count(path, dataset, sp):
---> 7         w = os.path.join(path, dataset, sp, '*.jpg')
```

Python Exception

例外處理

try-except

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 try:
13     print(f"tree DR: {Trees2.count('trees', None, 'DR')}")
14 except TypeError as e:
15     print(f'got exception: {e}')
```

got exception: join() argument must be str or bytes, not 'NoneType'

Python Exception

例外處理

try-except

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 try:
13     print(f"tree DR: {T.count('trees', 'train', 'DR')}")
14 except TypeError as e:
15     print(f'got exception: {e}')
16 except:
17     print(f'other except')
```

other except

Python Exception

例外處理

try-except-else

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 try:
13     print(f"tree DR: {Trees2.count('trees', 'train', 'DR')}")
14 except TypeError as e:
15     print(f'got exception: {e}')
16 except:
17     print(f'other except')
18 else:
19     print('we are fine')
```

```
tree DR: 105
we are fine
```

Python Exception

例外處理

try-except-else-finally

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 try:
13     print(f"tree DR: {T.count('trees', 'train', 'DR')}")
14 except TypeError as e:
15     print(f'got exception: {e}')
16 except:
17     print(f'other except')
18 else:
19     print('we are fine')
20 finally:
21     print('we have handled all exception!!')
```

other except

we have handled all exception!!

Your Turn 4-4

運用 try-except 機制，只用一個 except 擋截下列 3 個 print 指令產生的例外讓程式不中斷，並印出例外類別與內容 (10 mins)

Command

```
1 a = [1, 2, 3]
2
3 print(a[3])
4 print(a[0] / 0)
5 print(max(a[0]))
```

Output

```
IndexError: list index out of range
ZeroDivisionError: division by zero
TypeError: 'int' object is not iterable
```

Hint

一次擋截多項例外或擋截 Exception

Run Python

Python 程式應用方式

一次性的應用

執行方式

jupyter notebook

自動化、系統化、服務型式長期
執行的應用

執行方式

`python xxx.py`

on Windows 命令提示字元

`python3 xxx.py`

on MAC 終端機 or Linux Shell

Run Python

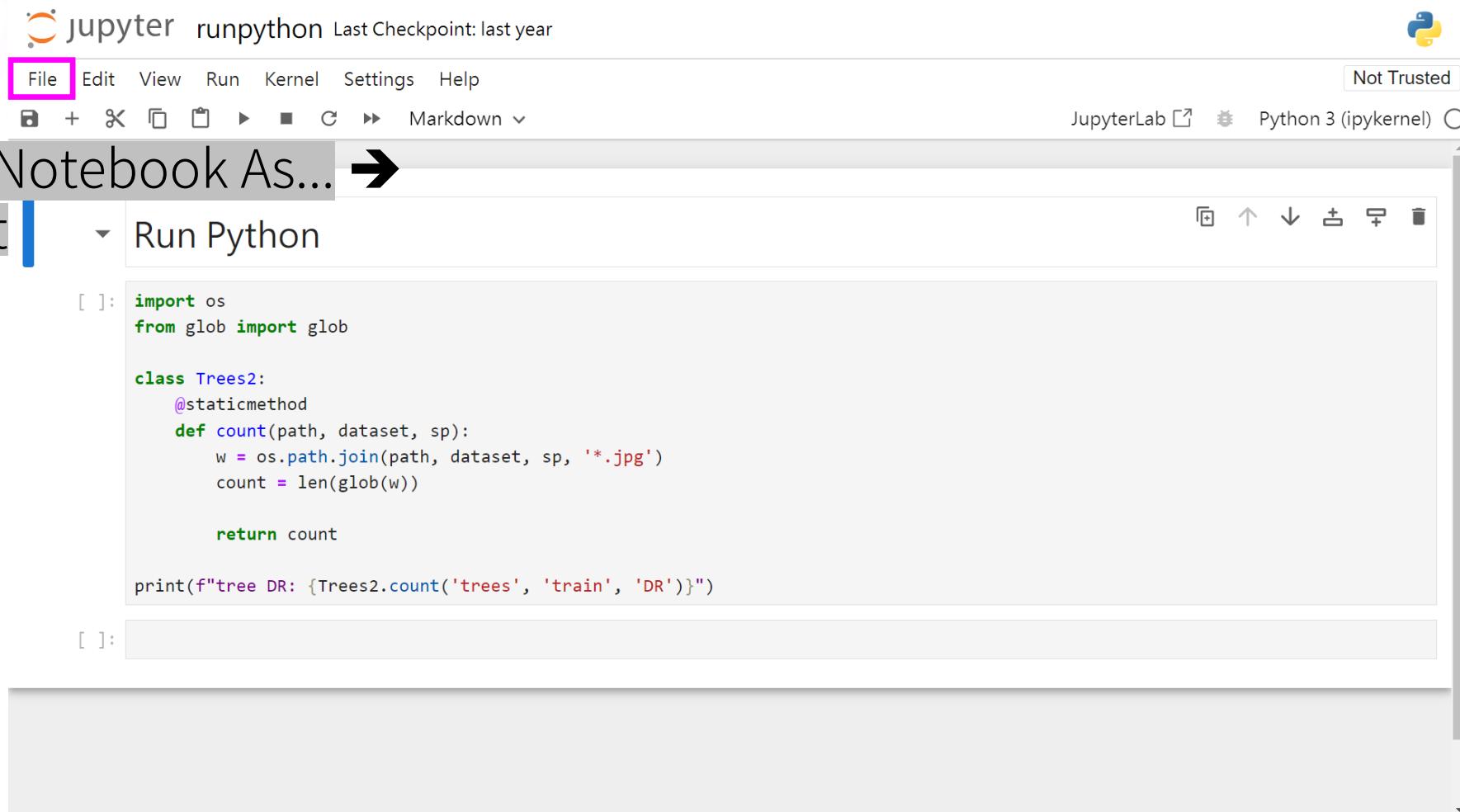
儲存為 .py

File →

Save and Export Notebook As... →

Executable Script

複製內容到文字
編輯器儲存為 .py



The screenshot shows the Jupyter Notebook interface. The title bar indicates the notebook is named "jupyter runpython" and was last checked on "last year". The top menu bar has "File" highlighted with a pink box, followed by "Edit", "View", "Run", "Kernel", "Settings", and "Help". Below the menu bar, there are icons for file operations like new, open, save, and run. To the right of the menu bar, it says "Not Trusted" and shows "JupyterLab" and "Python 3 (ipykernel)". The main area shows a code cell with the following Python script:

```
[ ]: import os
from glob import glob

class Trees2:
    @staticmethod
    def count(path, dataset, sp):
        w = os.path.join(path, dataset, sp, '*.jpg')
        count = len(glob(w))

    return count

print(f"tree DR: {Trees2.count('trees', 'train', 'DR')}")


[ ]:
```

Run Python

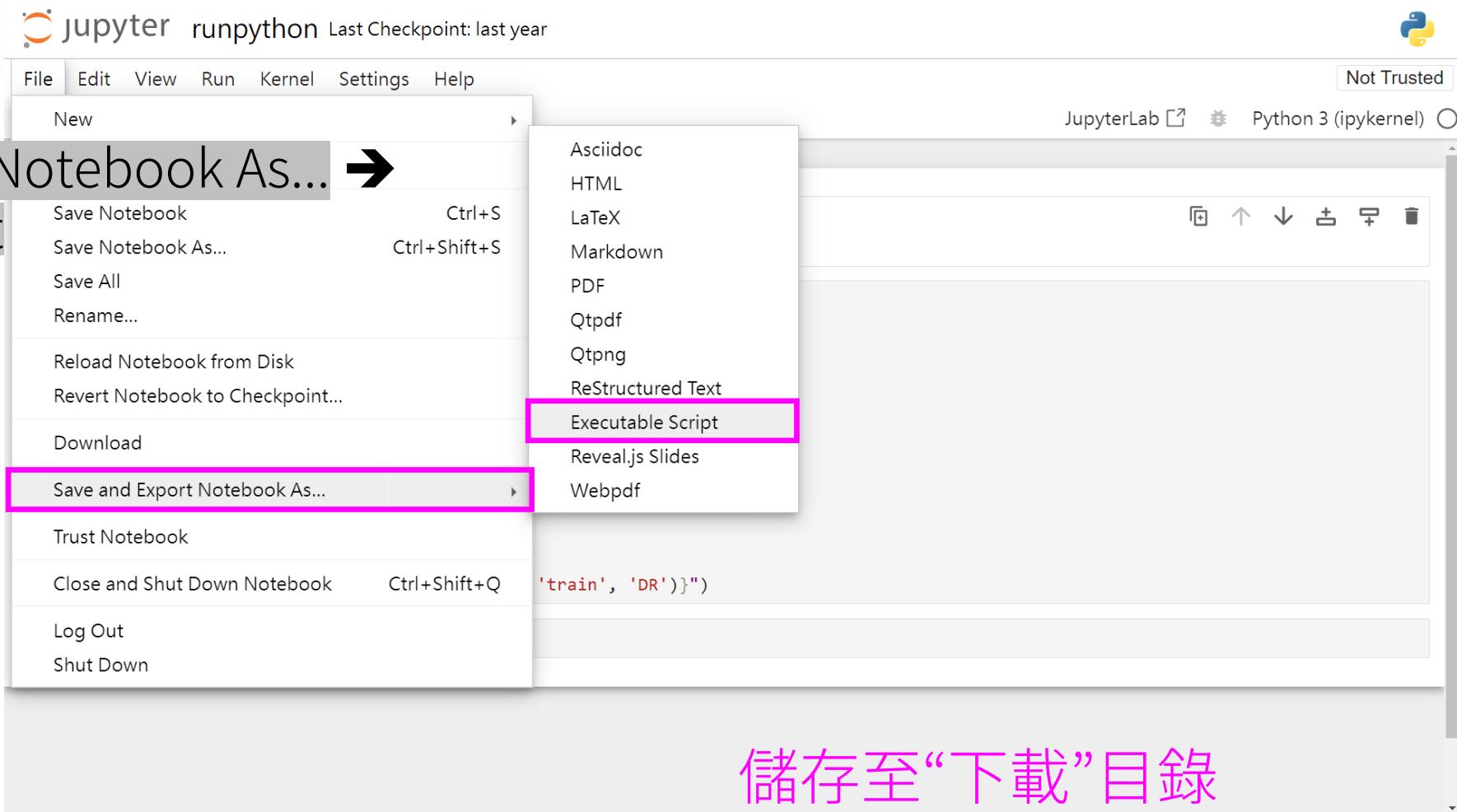
儲存為 .py

File →

Save and Export Notebook As... →

Executable Script

複製內容到文字
編輯器儲存為 .py



儲存至“下載”目錄

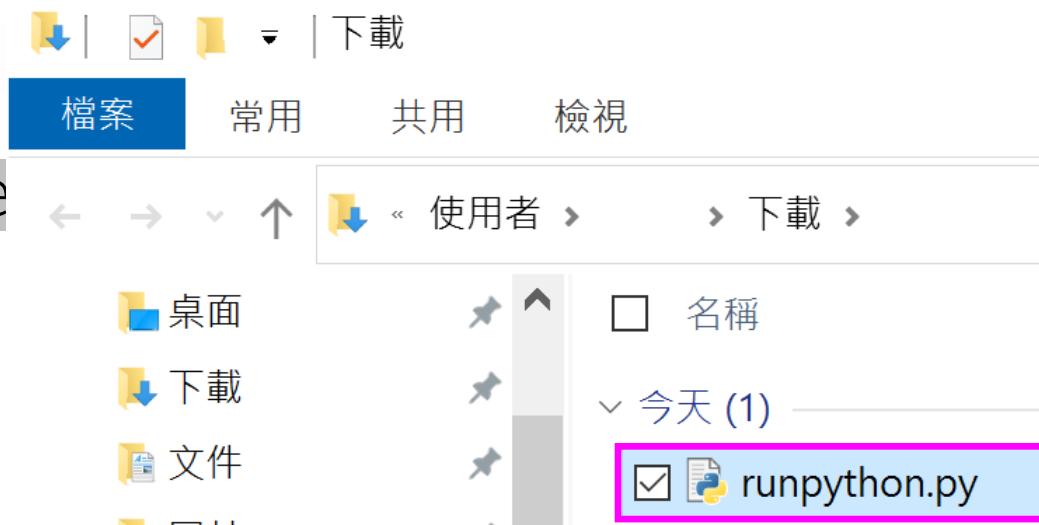
Run Python

儲存為 .py

File ➔

Save and Export Note
Executable Script

複製內容到文字
編輯器儲存為 .py



將檔案複製至課程目錄

Run Python

儲存為 .py

File →

Save and Export Notebook As... →

Executable Script

複製內容到文字
編輯器儲存為 .py

jupyter runpython.py ✓ 幾秒前

Logout

File Edit View Language Python

Save and Export Notebook As... →

```
3
4 # ## Run Python
5
6 # In[1]: 可以用 Jupyter Notebook 開啟及編輯 .py ,
7
8 import os
9 from glob import glob
10
11 class Trees2:
12     @staticmethod
13     def count(path, dataset, sp):
14         w = os.path.join(path, dataset, sp, '*.jpg')
15         count = len(glob(w))
16
17         return count
18
19
20 print(f"tree DR: {Trees2.count('trees', 'train', 'DR')}")
21
22
23 # In[ ]:
```

Run Python

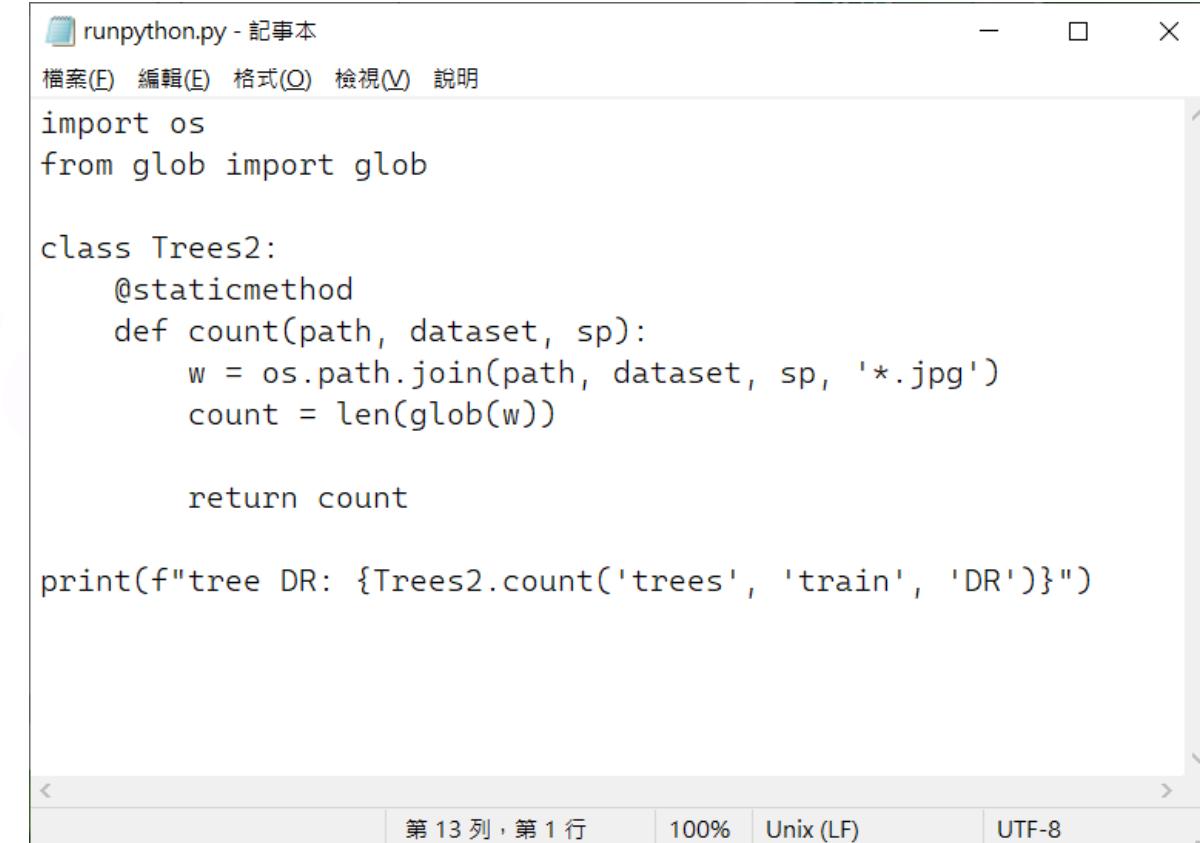
儲存為 .py

File →

Save and Export Notebook As... →

Executable Script

複製內容到文字
編輯器儲存為 .py



The screenshot shows a Windows Notepad window titled "runpython.py - 記事本". The window contains the following Python code:

```
import os
from glob import glob

class Trees2:
    @staticmethod
    def count(path, dataset, sp):
        w = os.path.join(path, dataset, sp, '*.jpg')
        count = len(glob(w))

        return count

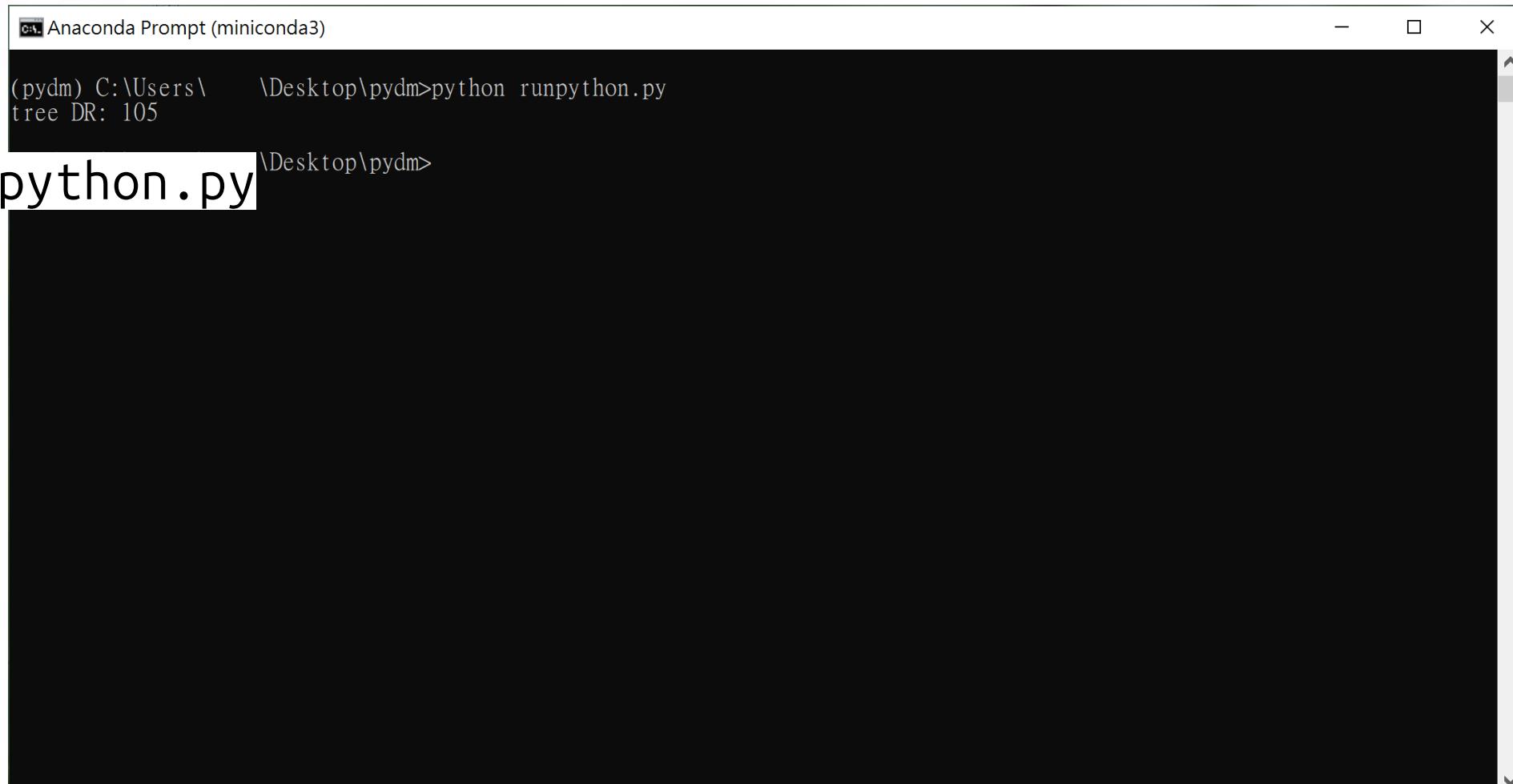
print(f"tree DR: {Trees2.count('trees', 'train', 'DR')}")
```

The status bar at the bottom of the window displays "第 13 列, 第 1 行" (Line 1, Column 13), "100%", "Unix (LF)", and "UTF-8".

Run Python

執行 .py

python runpython.py



Anaconda Prompt (miniconda3)

```
(pydm) C:\Users\    \Desktop\pydm>python runpython.py
tree DR: 105
```

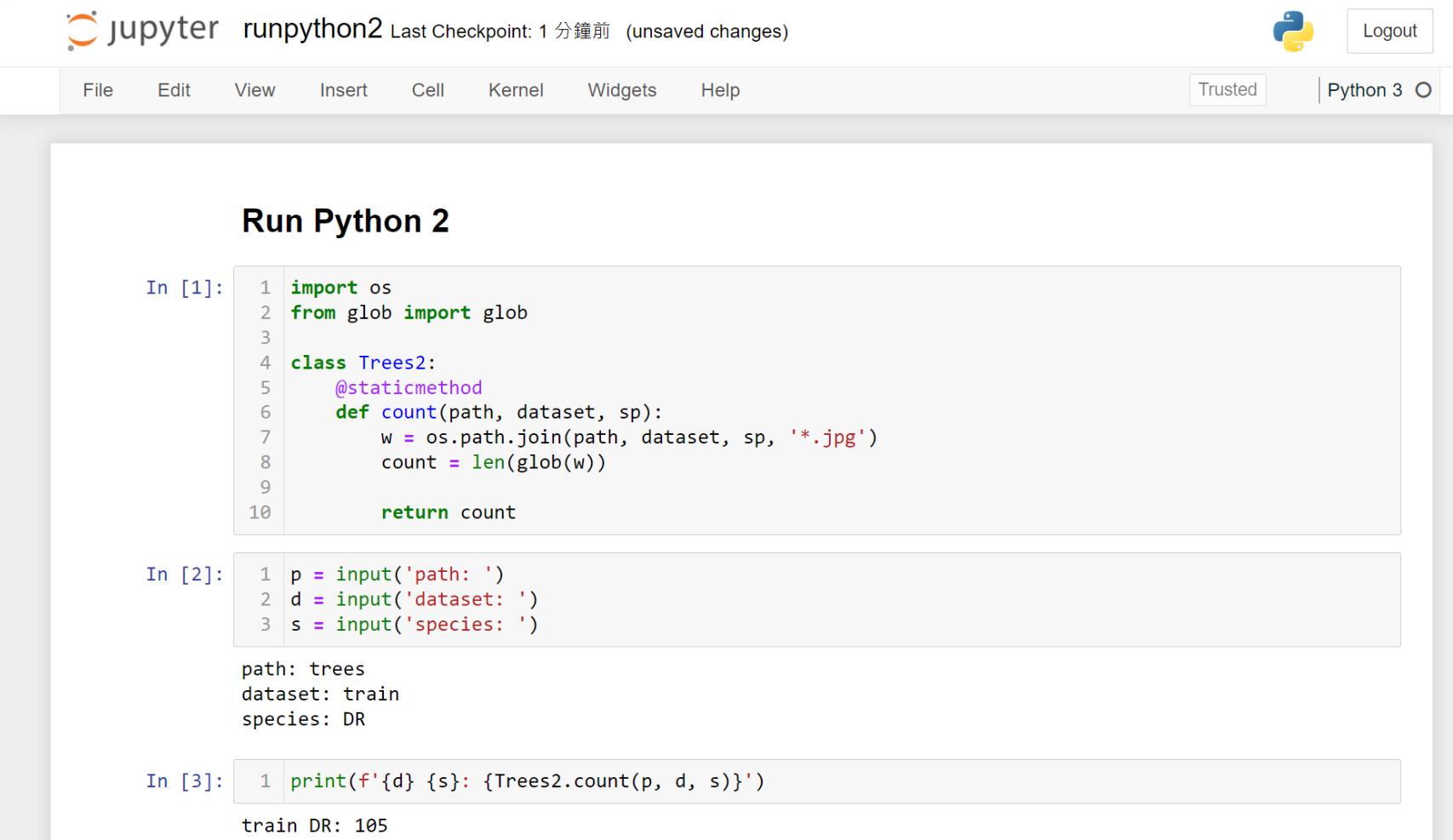
\Desktop\pydm>

A screenshot of a Windows terminal window titled "Anaconda Prompt (miniconda3)". The window shows a command-line interface with a black background and white text. The prompt "(pydm) C:\Users\ \Desktop\pydm>" is visible at the top. Below it, the command "python runpython.py" is entered, followed by the output "tree DR: 105". The window has standard window controls (minimize, maximize, close) in the top right corner.

Run Python

介面包裝
系統化執行

儲存為 .py



The screenshot shows a Jupyter Notebook interface with the title "Run Python 2". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A "Trusted" button and "Python 3" are also visible. The notebook contains three code cells:

- In [1]:**

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
```
- In [2]:**

```
1 p = input('path: ')
2 d = input('dataset: ')
3 s = input('species: ')
```

path: trees
dataset: train
species: DR
- In [3]:**

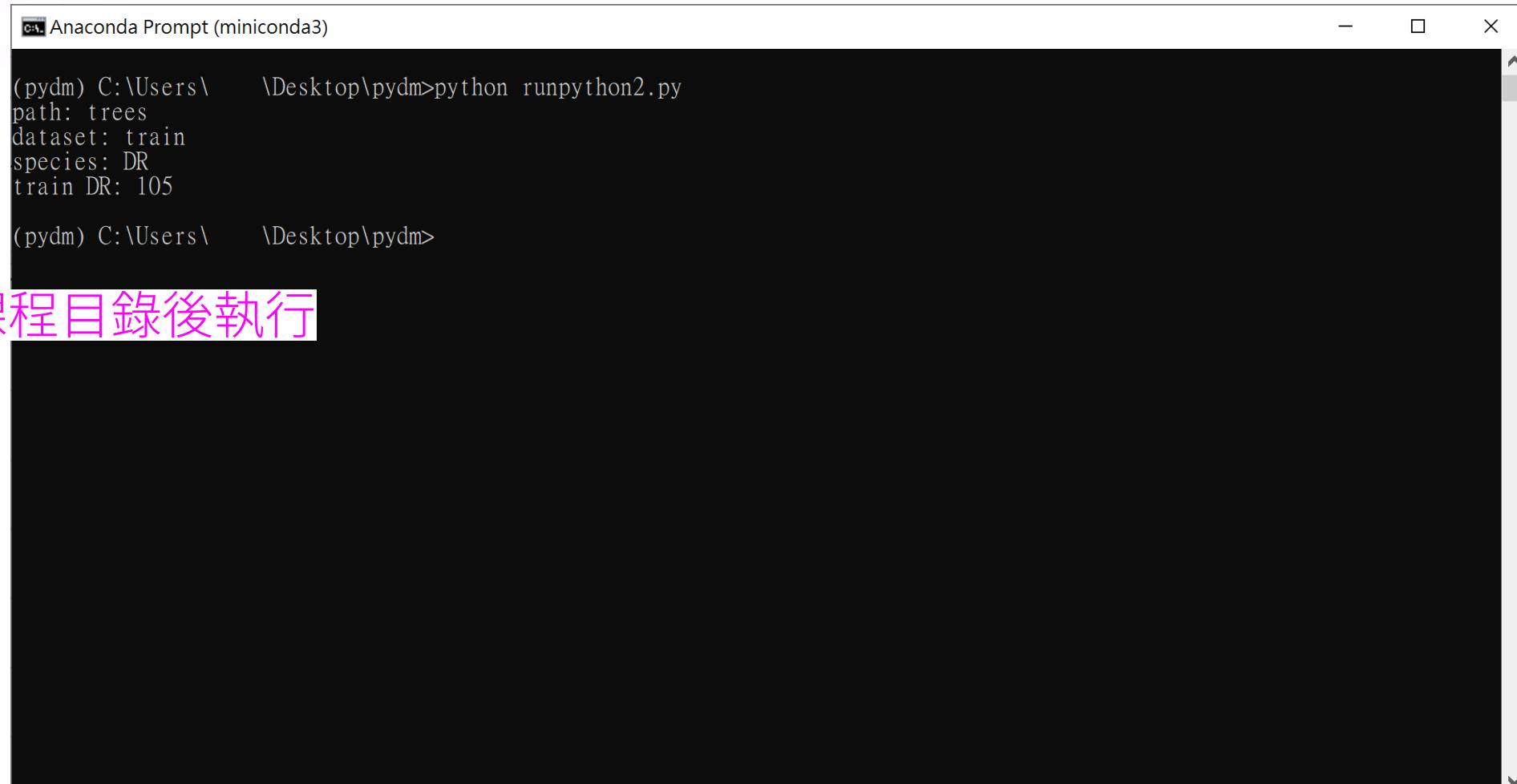
```
1 print(f'{d} {s}: {Trees2.count(p, d, s)}')
```

train DR: 105

Run Python

介面包裝
系統化執行

複製 .py 至課程目錄後執行



The screenshot shows a terminal window titled "Anaconda Prompt (miniconda3)". The command entered is "python runpython2.py". The output displays the following information:

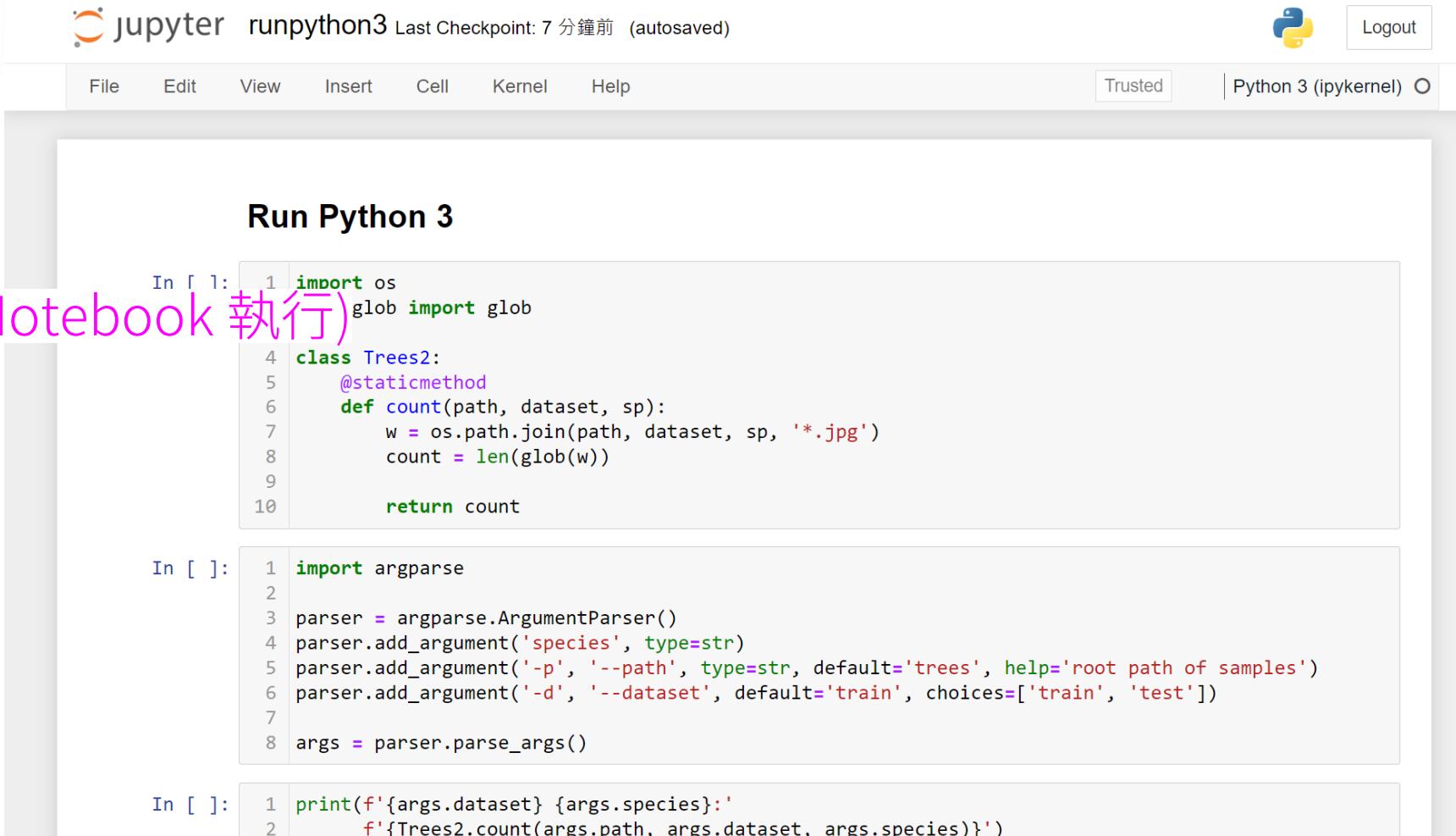
```
(pydm) C:\Users\    \Desktop\pydm>python runpython2.py
path: trees
dataset: train
species: DR
train DR: 105

(pydm) C:\Users\    \Desktop\pydm>
```

Run Python

介面包裝 系統化執行

儲存為 .py
(無法於 Jupyter Notebook 執行)



The screenshot shows a Jupyter Notebook interface with the title "Run Python 3". The notebook has three cells:

- In [1]:**

```
1 import os
2 glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
```
- In []:**

```
1 import argparse
2
3 parser = argparse.ArgumentParser()
4 parser.add_argument('species', type=str)
5 parser.add_argument('-p', '--path', type=str, default='trees', help='root path of samples')
6 parser.add_argument('-d', '--dataset', type=str, default='train', choices=['train', 'test'])
7
8 args = parser.parse_args()
```
- In []:**

```
1 print(f'{args.dataset} {args.species}:'
2       f'{Trees2.count(args.path, args.dataset, args.species)}')
```

Run Python

介面包裝 系統化執行

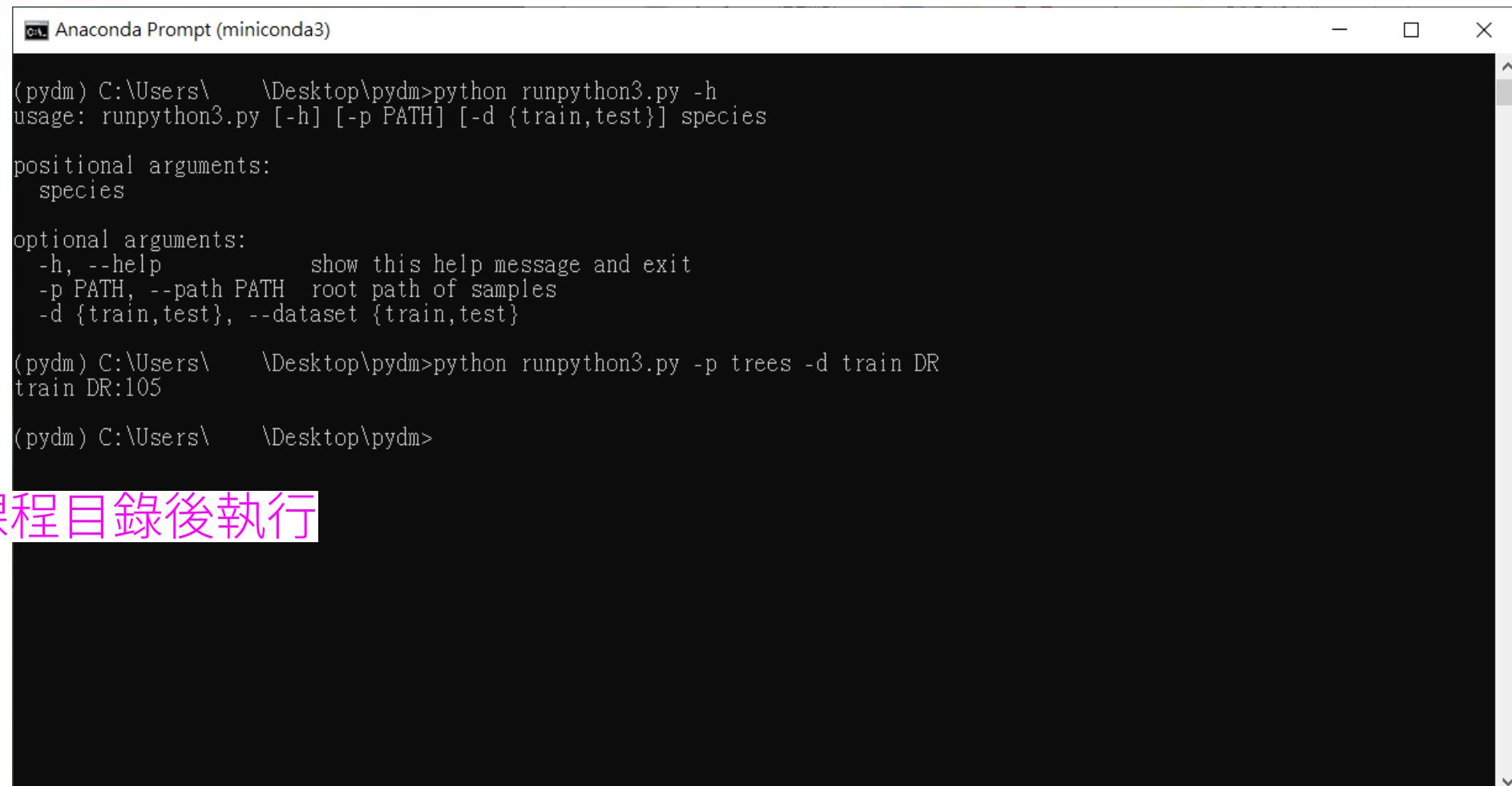
```
import argparse

parser = argparse.ArgumentParser()
    位置參數    參數型態
parser.add_argument('species', type=str)        預設值          參數說明
parser.add_argument('-p', '--path', type=str, default='trees', help='root path of samples')
parser.add_argument('-d', '--dataset', default='train', choices=['train', 'test'])
    參數簡稱    參數全名          參數提示

args = parser.parse_args()
args.species, args.path, args.dataset
```

Run Python

介面包裝
系統化執行



```
Anaconda Prompt (miniconda3)
(pydm) C:\Users\    \Desktop\pydm>python runpython3.py -h
usage: runpython3.py [-h] [-p PATH] [-d {train,test}] species

positional arguments:
  species

optional arguments:
  -h, --help            show this help message and exit
  -p PATH, --path PATH  root path of samples
  -d {train,test}, --dataset {train,test}

(pydm) C:\Users\    \Desktop\pydm>python runpython3.py -p trees -d train DR
train DR:105

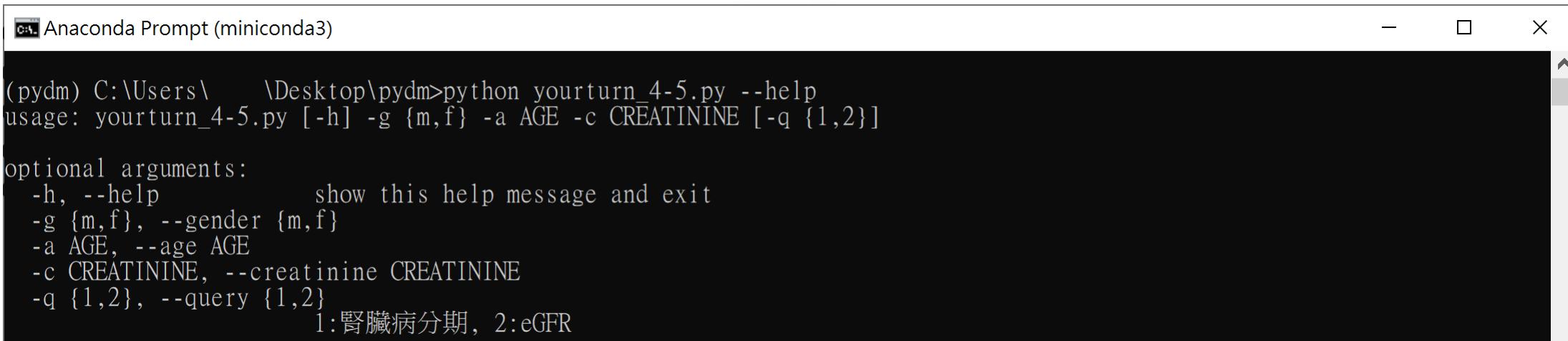
(pydm) C:\Users\    \Desktop\pydm>
```

複製 .py 至課程目錄後執行

Your Turn 4-5

承 Your Turn 4-1，將此程式製作成 .py 檔 (檔名自取)，能夠於 Windows 命令提示字元 執行 (15 mins)

SPEC



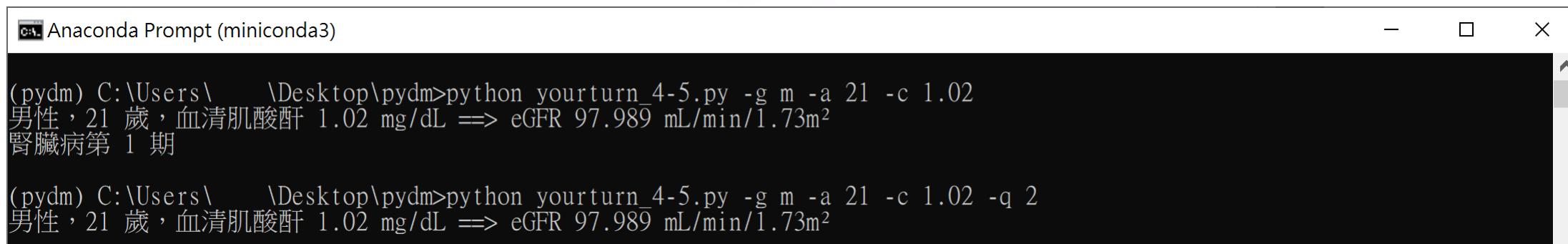
```
anaconda Prompt (miniconda3)
(pydm) C:\Users\    \Desktop\pydm>python yourturn_4-5.py --help
usage: yourturn_4-5.py [-h] -g {m,f} -a AGE -c CREATININE [-q {1,2}]

optional arguments:
  -h, --help            show this help message and exit
  -g {m,f}, --gender {m,f}
  -a AGE, --age AGE
  -c CREATININE, --creatinine CREATININE
  -q {1,2}, --query {1,2}
                                1:腎臟病分期, 2:eGFR
```

Your Turn 4-5

承 Your Turn 4-1，將此程式製作成 .py 檔 (檔名自取)，能夠於 Windows 命令提示字元 執行 (15 mins)

Output



```
anaconda Prompt (miniconda3)
(pydm) C:\Users\    \Desktop\pydm>python yourturn_4-5.py -g m -a 21 -c 1.02
男性，21 歲，血清肌酸酐 1.02 mg/dL ==> eGFR 97.989 mL/min/1.73m2
腎臟病第 1 期

(pydm) C:\Users\    \Desktop\pydm>python yourturn_4-5.py -g m -a 21 -c 1.02 -q 2
男性，21 歲，血清肌酸酐 1.02 mg/dL ==> eGFR 97.989 mL/min/1.73m2
```

Run Python

模組化 重複使用

jupyter runpython3.py ✓ 11 分鐘前

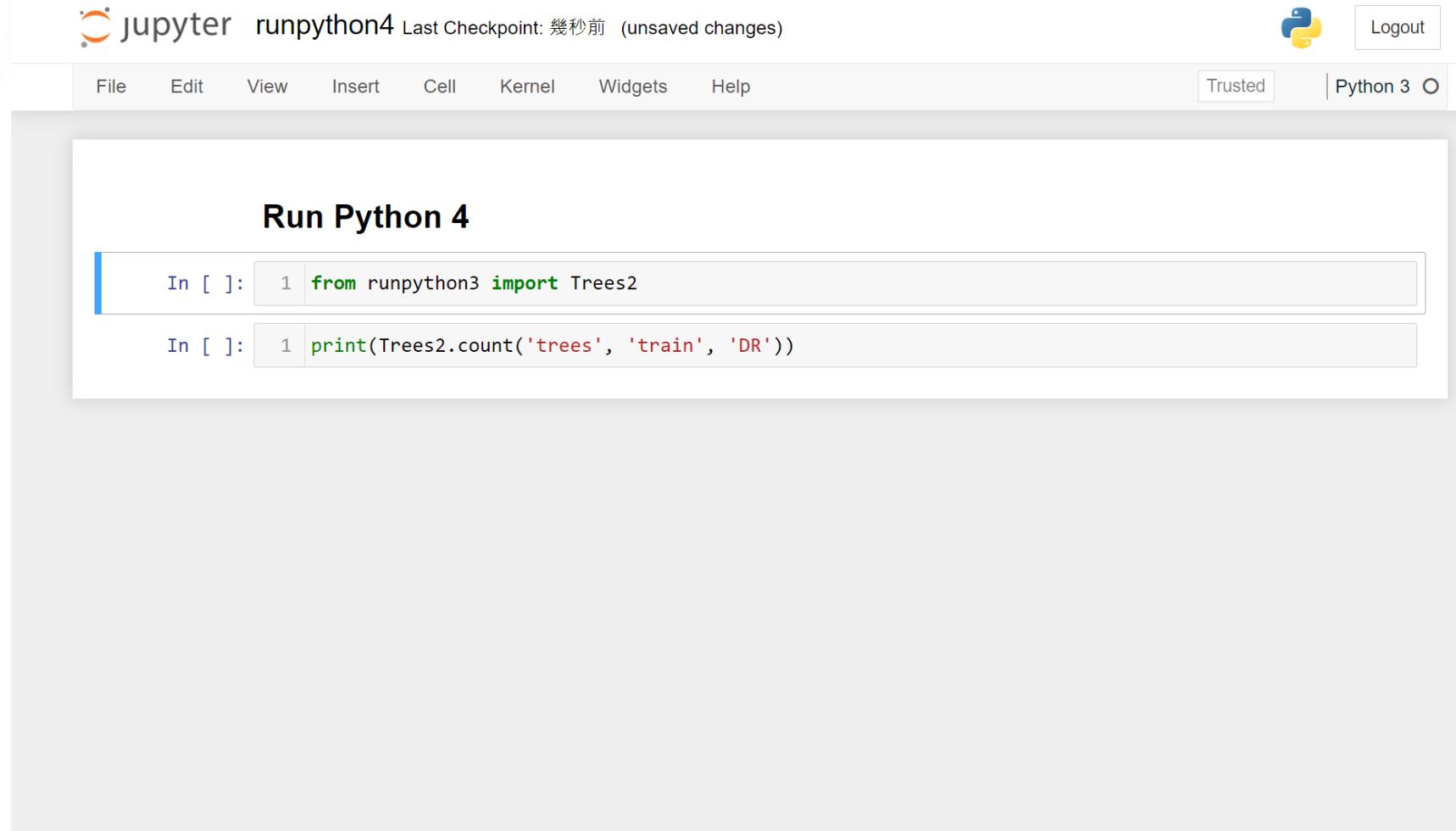
Logout Python

```
File Edit View Language
```

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 import argparse
13
14 parser = argparse.ArgumentParser()
15 parser.add_argument('species', type=str)
16 parser.add_argument('-p', '--path', type=str, default='trees', help='root path of samples')
17 parser.add_argument('-d', '--dataset', type=str, default='train', choices=['train', 'test'])
18
19 args = parser.parse_args()
20
21 print(f'{args.dataset} {args.species}:'
22       f'{Trees2.count(args.path, args.dataset, args.species)}')
```

Run Python

模組化
重複使用

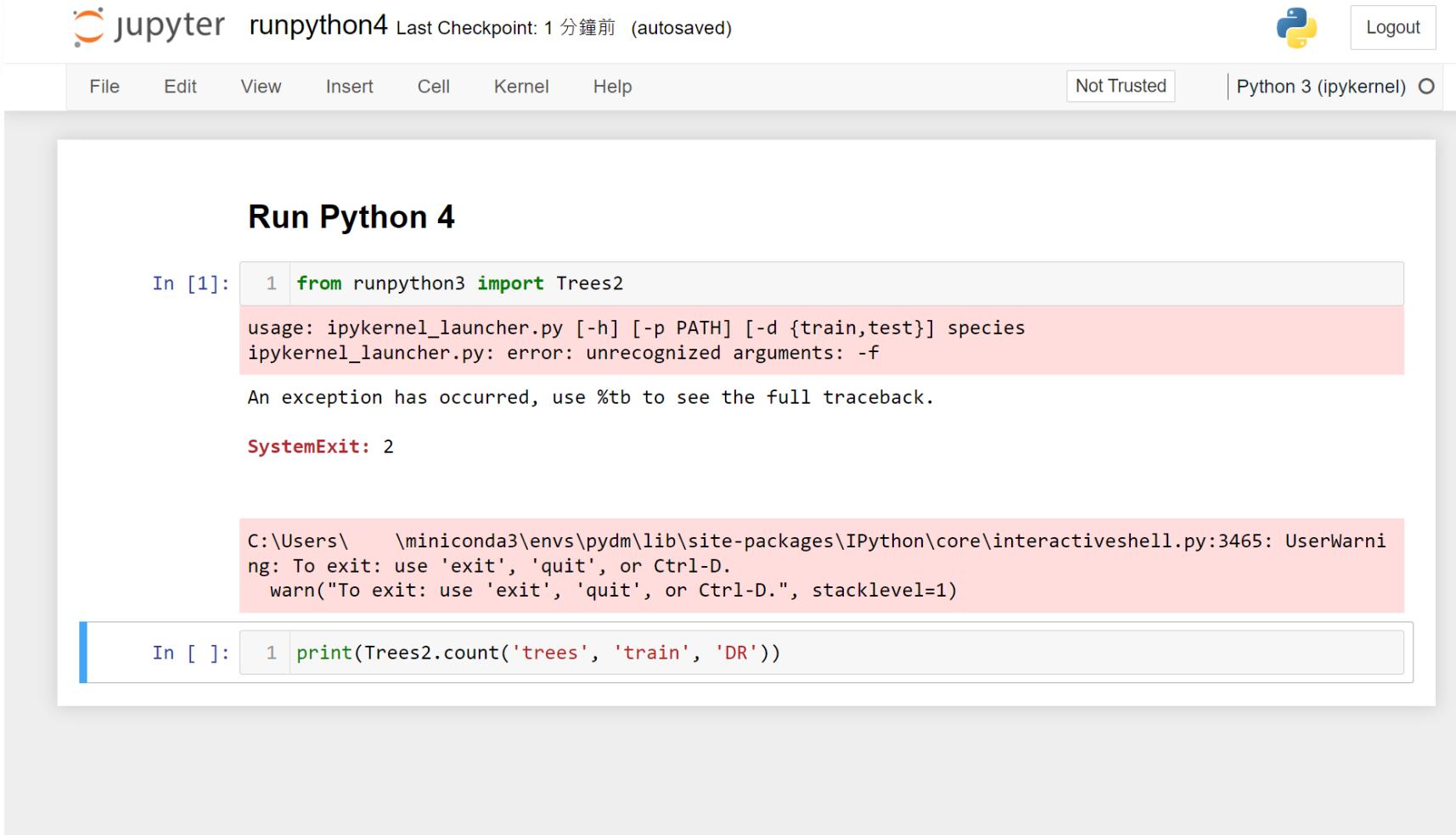


The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter runpython4 Last Checkpoint: 幾秒前 (unsaved changes)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help
- Status Bar:** Trusted | Python 3
- Code Cells:**
 - In []: 1 `from runpython3 import Trees2`
 - In []: 1 `print(Trees2.count('trees', 'train', 'DR'))`

Run Python

模組化
重複使用



The screenshot shows a Jupyter Notebook interface titled "jupyter runpython4 Last Checkpoint: 1 分鐘前 (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, and Help. The status bar indicates "Not Trusted" and "Python 3 (ipykernel)".

The notebook displays the following content:

```
In [1]: 1 from runpython3 import Trees2
usage: ipykernel_launcher.py [-h] [-p PATH] [-d {train,test}] species
ipykernel_launcher.py: error: unrecognized arguments: -f
An exception has occurred, use %tb to see the full traceback.

SystemExit: 2

C:\Users\    \miniconda3\envs\pydm\lib\site-packages\IPython\core\interactiveshell.py:3465: UserWarning
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

In []: 1 print(Trees2.count('trees', 'train', 'DR'))

Run Python

模組化 重複使用

jupyter runpython3.py ✓ 11 分鐘前

Logout

Python

```
File Edit View Language
```

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 import argparse
13
14 parser = argparse.ArgumentParser()
15 parser.add_argument('species', type=str)
16 parser.add_argument('-p', '--path', type=str, default='trees', help='root path of samples')
17 parser.add_argument('-d', '--dataset', type=str, default='train', choices=['train', 'test'])
18
19 args = parser.parse_args()
20
21 print(f'{args.dataset} {args.species}:'
22       f'{Trees2.count(args.path, args.dataset, args.species)}')
```

Run Python

模組化 重複使用

jupyter runpython5.py ✓ 12 分鐘前

Logout

Python

```
File Edit View Language
```

```
1 import os
2 from glob import glob
3
4 class Trees2:
5     @staticmethod
6     def count(path, dataset, sp):
7         w = os.path.join(path, dataset, sp, '*.jpg')
8         count = len(glob(w))
9
10    return count
11
12 if __name__ == '__main__':
13     import argparse
14
15     parser = argparse.ArgumentParser()
16     parser.add_argument('species', type=str)
17     parser.add_argument('-p', '--path', type=str, default='trees', help='root path of samples')
18     parser.add_argument('-d', '--dataset', type=str, default='train', choices=['train', 'test'])
19
20     args = parser.parse_args()
21
22     print(f'{args.dataset} {args.species}:'
23           f'{Trees2.count(args.path, args.dataset, args.species)}')
```

Run Python

模組化
重複使用

The screenshot shows a Jupyter Notebook interface titled "Run Python 4". The notebook has a "Not Trusted" status and is using a "Python 3 (ipykernel)" kernel.

In [1]:

```
1 from runpython3 import Trees2
```

usage: ipykernel_launcher.py [-h] [-p PATH] [-d {train,test}] species
ipykernel_launcher.py: error: unrecognized arguments: -f

An exception has occurred, use %tb to see the full traceback.

SystemExit: 2

Anaconda Prompt (miniconda3)

```
(pydm) C:\Users\    \Desktop\pydm>python runpython5.py -p trees -d train DR  
train DR:105
```

In []:

```
1 print(Trees2.count('trees', 'train', 'DR'))
```

In [2]:

```
1 from runpython5 import Trees2
```

In [3]:

```
1 print(Trees2.count('trees', 'train', 'DR'))
```

105

Run Python

模組化 重複使用

jupyter runpython6.py ✓ 幾秒前

Logout Python

```
File Edit View Language
```

```
1 import os
2 from glob import glob
3
4 print(f'runpython6.py\'s name is {__name__}')
5
6 class Trees2:
7     @staticmethod
8     def count(path, dataset, sp):
9         w = os.path.join(path, dataset, sp, '*.jpg')
10        count = len(glob(w))
11
12        return count
13
14 if __name__ == '__main__':
15     import argparse
16
17     parser = argparse.ArgumentParser()
18     parser.add_argument('species', type=str)
19     parser.add_argument('-p', '--path', type=str, default='trees', help='root path of samples')
20     parser.add_argument('-d', '--dataset', default='train', choices=['train', 'test'])
21
22     args = parser.parse_args()
23
24     print(f'{args.dataset} {args.species}:'
25           f'{Trees2.count(args.path, args.dataset, args.species)}')
```

Run Python

模組化 重複使用

The screenshot shows a Jupyter Notebook interface and an Anaconda Prompt window side-by-side.

Jupyter Notebook:

- Kernel: Python 3 (ipykernel)
- Cells:
 - In [1]: `print(Trees2.count('trees', 'train', 'DR'))`
 - In [2]: `from runpython5 import Trees2`
 - In [3]: `print(Trees2.count('trees', 'train', 'DR'))`
105
 - In [4]: `from runpython6 import Trees2`
runpython6.py's name is runpython6
 - In [5]: `print(Trees2.count('trees', 'train', 'DR'))`
105

Anaconda Prompt (miniconda3):

- (pydm) C:\Users\...\Desktop\pydm>python runpython6.py -p trees -d train DR
- runpython6.py's name is main
- train DR:105

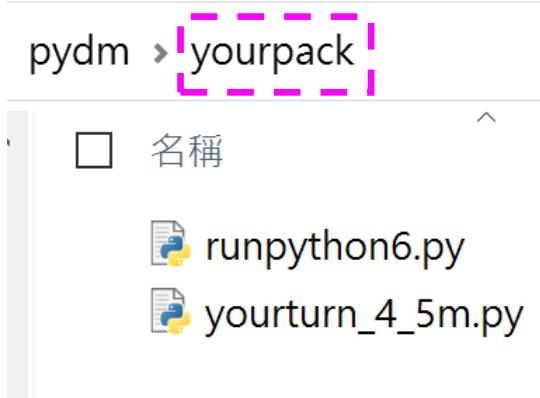
Your Turn 4-6

改動並叫用 module (20 mins)

1. 將 Your Turn 4-5 程式製作成 module (不修改 class 內容)
2. 開發新的 .ipynb 程式叫用上述 module 中的 class，將腎臟病分期改為：
 $eGFR \geq 100 \rightarrow 0$ 期
 $100 > eGFR \geq 60 \rightarrow 1$ 期
 $60 > eGFR \rightarrow 2$ 期
3. 依上述分期印出 42 歲男性、血清肌酸酐 1.02 的腎臟病分期

Run Python

套件化 包裝多個模組



```
1 from yourpack.runpython6 import *
2 from yourpack.yourturn_4_5m import *
```

runpython6.py's name is yourpack.runpython6

```
1 print(Trees2)
2 print(CKDStager)
```

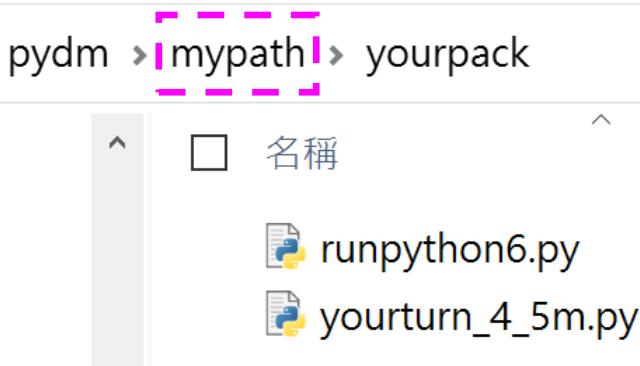
```
<class 'yourpack.runpython6.Trees2'>
<class 'yourpack.yourturn_4_5m.CKDStager'>
```

Run Python

套件化

包裝多個模組

套件與模組路徑



```
1 from yourpack.runpython6 import *
2 from yourpack.yourturn_4_5m import *
```

ModuleNotFoundError

Traceback (most recent call last)

```
~\AppData\Local\Temp\ipykernel_5276\3922422314.py in <module>
-----> 1 from yourpack.runpython6 import *
      2 from yourpack.yourturn_4_5m import *
```

ModuleNotFoundError: No module named 'yourpack'

```
1 import sys, os
2 sys.path.append(os.path.join('..', 'mypath'))
3 from yourpack.runpython6 import *
4 from yourpack.yourturn_4_5m import *
```

runpython6.py's name is yourpack.runpython6

```
1 print(Trees2)
2 print(CKDStager)
```

```
<class 'yourpack.runpython6.Trees2'>
<class 'yourpack.yourturn_4_5m.CKDStager'>
```

Python 5 Application

Python Application, requests, JSON, BeautifulSoup, Jieba, wordcloud, Matplotlib

Weather Grabber

擷取即時天氣資料並顯示

Tasks:

1. 資料
2. 界接
3. 規格
4. 開發
5. 測試

Weather 資料

氣象資料開放平台

<https://opendata.cwa.gov.tw/>

觀測

The screenshot shows the 'Observation' section of the Open Data Platform. On the left, there's a sidebar with categories: 預報 (Forecast), 觀測 (Observation, highlighted in yellow), 地震海嘯 (Earthquake Tsunami), 氣候 (Climate), 天氣警特報 (Weather Alerts), 數值預報 (Numerical Forecast), and 天文 (Astronomy). Below the sidebar is a link to '資料下載總排行' (Data Download Ranking). The main content area has a title '觀測' (Observation) with a search bar. A table lists various observation datasets:

資料名稱	資料編號
自動氣象站-氣象觀測資料	O-A0001-001
JSON XML API	
自動雨量站-雨量觀測資料	O-A0002-001
JSON XML API	
自動雨量站-過去1小時雨量觀測資料	O-A0002-002
ZIP	
局屬氣象站-現在天氣觀測報告	O-A0003-001
JSON XML API	
酸雨pH值-每日酸雨pH值	O-A0004-001
JSON XML API	

At the bottom, there are navigation links for pages 1, 2, 3, ..., >, >>.

Central footer: 中央氣象局 | 隱私權保護政策 | 資訊安全政策
地址：100006臺北市中正區公園路64號 總機：(02)2349-1000(代表號)

Weather 資料

氣象資料開放平台

<https://opendata.cwa.gov.tw/>

觀測



家 > 資料主題 > 觀測 > 局屬氣象站-現在天氣觀測報告

預報
觀測
地震海嘯
氣候
天氣警特報
數值預報
天文

資料下載總排行 >

局屬氣象站-現在天氣觀測報告

資料集

資料預覽

局屬氣象站-現在天氣觀測報告

檔案下載

請登入

資料集類型

rawData

資料集描述

局屬氣象站資料(現在天氣觀測報告)-本局局屬有人氣象站資料

主要欄位說明

STID、STNM、TIME、LAT、LON、ELEV、WDIR、WDSD、TEMP、HUMD、PRES、24R、H_FX、H_XD、H_FXT、H_F10、H_10D、H_F10T、H_UVI、D_TX、D_TXT、D_TS、VIS、Weather、CITY、CITY_SN、TOWN、TOWN_SN

資料集提供機關

中央氣象局

更新頻率

10分鐘

Weather 資料

氣象資料開放平台

<https://opendata.cwa.gov.tw/>

>
觀測

The screenshot shows the homepage of the 'Open Weather Data' platform. The header features a green background with a stylized illustration of weather icons like a sun, clouds, and a lightning bolt. The title 'OPEN WEATHER DATA' is prominently displayed in large blue letters. The top navigation bar includes links for '公告事項', '資料主題', '開發指南', '應用活化', '推廣發展', '常見問答', and '登入/註冊'. Below the header, there are two main login options: '會員登入' (highlighted in yellow) and '氣象會員登入' (highlighted in pink). Further down, there is a '其他方式登入' section with a 'Facebook會員登入' button. The footer contains links to '中央氣象局 | 隱私權保護政策 | 資訊安全政策' and the address '地址：100006臺北市中正區公園路64號 總機：(02)2349-1000(代表號)'.

Weather 資料

氣象資料開放平台

<https://opendata.cwa.gov.tw/>

觀測

會員登入 > 會員登入

會員登入

氣象會員登入

氣象會員登入

郵件帳號

密碼

我不是機器人 
reCAPTCHA
隱私權 · 條款

氣象會員登入

[加入會員](#) [忘記密碼](#)

其他方式登入

Facebook會員登入

中央氣象局 | [隱私權保護政策](#) | [資訊安全政策](#)
地址 : 100006臺北市中正區公園路64號 總機 : (02)2349-1000(代表號)

Weather 資料

氣象資料開放平台

[https://opendata.cwa.gov.tw/ >
觀測](https://opendata.cwa.gov.tw/)



氣象局會員服務

中央氣象局會員申請同意書

為保障您的權益，請於註冊成為本局會員並使用各項服務前，詳細閱讀本進駐同意書所有內容。當您在線上點選「我同意」鍵，表示您已同意使用本局所提供之任何服務，並同意遵守以下所有規範。

會員基本資料之註冊、更新及保管

- 1、於註冊時提供真實、正確、現行及完整之個人 email 資料。請勿使用「10分鐘信箱」等暫時信箱來註冊，若未來無法收取通知郵件可能影響會員的使用權利。
- 2、維持並更新個人 email 資料，使其保持真實、正確、現行及完整。
- 3、提供的資料若不真實、不正確、並非現行或不完整，或使用暫時性信箱，本局保留隨時終止會員資格及使用各項服務資格的權利。
- 4、為提供更有效率的服務，本系統將於每月1日凌晨0點，清除前1個月份「未完成會員申請確認」的暫時性資料，屆時敬請因故「未完成會員申請確認」的申請者重新「加入會員」，不便之處，尚請見諒，已成為會員者將不受任何影響。
- 5、申請忘記密碼時系統會發送一次性密碼連結至帳號信箱，需於信件寄發後60分鐘內修改完成。

會員資訊內容及行為規範

- 1、不得轉載任何未經本局授權的相關氣象資料。
- 2、不張貼或傳播任何廣告及其他形式之不實宣傳資料。
- 3、不傳輸任何會侵犯他人權利的資料。
- 4、不傳輸會侵犯現行法律、國際相關法律的資料。
- 5、不從事干擾或中斷此項服務或連接至此服務之伺服器或網路的行為。
- 6、遵守所有連接至此網路的要求、規則、程序及政策。
- 7、本局擁有會員在CWB網站所公佈內容及行為的最終決定權，並可不經通知刪除在CWB網站之不當的內容。
- 8、有違上述行為嚴重者，本局將保留隨時終止會員資格及使用各項服務資格的權利。

服務之停止與更改

於發生下列情形之一時，本局有權停止或中斷提供服務：

- 1、通訊設備進行必要之保養及施工時。
- 2、發生突發性之電子通信設備故障時。
- 3、由於天災等不可抗力之因素致使本局無法提供服務時。

同意 | 不同意

Weather 資料

氣象資料開放平台

<https://opendata.cwa.gov.tw/>

觀測

*為必填

電子郵件 * @gmail.com
*若您未收到註冊確認信，請您至垃圾郵件區檢查。

密碼 * ..

再次輸入密碼 * ..
*請輸入6(含)~12個字元，含有英文與數字，請勿使用空白字元及「+」「%」「&」符號。

性別 * 女 男 其他

居住地區 * 臺灣地區 大陸地區 國外地區

居住都市 * 市

出生年(西元) * 19

工作性質 *

教育程度 *

訂閱 每日氣象電子報 天氣警、特報 地震報告

325103 CWB

325103   

送出

Weather 資料

氣象資料開放平台

[https://opendata
.cwa.gov.tw/ >
觀測](https://opendata.cwa.gov.tw/)



Weather 資料

氣象資料開放平台

<https://opendata.cwa.gov.tw/>

觀測

The screenshot shows the homepage of the 'Open Weather Data' platform. The header features a logo with a stylized cloud and raindrop, followed by the text '氣象資料開放平臺 OPEN WEATHER DATA'. On the right side of the header are links for '登入/註冊' (Login/Register), '民意信箱' (Feedback Box), and '網站地圖' (Site Map). Below the header is a colorful illustration of a green landscape with a sun, clouds, trees, and people, with large 3D letters spelling 'OPEN WEATHER'. A red warning sign icon is also present. The main navigation menu below the illustration includes links for '公告事項', '資料主題', '開發指南', '應用活化', '推廣發展', '常見問答', and '登入/註冊'. The main content area has a yellow button labeled '會員登入' (Member Login) and a blue button labeled '氣象會員登入' (气象 Member Login), which is highlighted with a pink border. Below these are links for '其他方式登入' (Other Login Methods) and 'Facebook會員登入' (Facebook Member Login). At the bottom of the page is a dark blue footer bar containing links for '中央氣象局 | 隱私權保護政策 | 資訊安全政策' and the address '地址：100006臺北市中正區公園路64號 總機：(02)2349-1000(代表號)'.

Weather 資料

氣象資料開放平台

<https://opendata.cwa.gov.tw/>

觀測

會員登入 > 會員登入

會員登入

氣象會員登入

氣象會員登入

郵件帳號
@gmail.com

密碼

我不是機器人  reCAPTCHA
隱私權 - 僅數

氣象會員登入

加入會員 | 忘記密碼

其他方式登入

Facebook會員登入

中央氣象局 | 隱私權保護政策 | 資訊安全政策
地址：100006臺北市中正區公園路64號 總機：(02)2349-1000(代表號)

Weather 資料

氣象資料開放平台

<https://opendata.cwa.gov.tw/>

觀測



The screenshot shows the homepage of the Open Weather Data platform. The main header features the text "OPEN WEATHER DATA" above a large, stylized "OPEN WEATHER" graphic. Below the graphic are various icons representing weather and technology. A navigation bar at the top includes links for "公告事項", "資料主題", "開發指南", "應用活化", "推廣發展", "常見問答", and "會員資訊". The current page is "會員資訊 > API授權碼". On the left, there's a sidebar with "會員基本資料" and "API授權碼" options. The "API授權碼" option is highlighted with a yellow background. The main content area is titled "API授權碼" and contains text about obtaining authorization codes via URLs or RESTful APIs. It includes two buttons: "取得授權碼" (Obtain Authorization Code) and "更新授權碼" (Update Authorization Code). The footer contains links to "中央氣象局 | 隱私權保護政策 | 資訊安全政策" and "地址 : 100006臺北市中正區公園路64號 繼機 : (02)2349-1000(代表號)".

Weather 資料

氣象資料開放平台

<https://opendata.cwa.gov.tw/> >
觀測

The screenshot shows the homepage of the Open Weather Data platform. The main header features a large, stylized 'OPEN WEATHER' logo with various weather icons (clouds, sun, rain) floating around it. Below the header is a navigation bar with links: 公告事項, 資料主題, 開發指南, 應用活化, 推廣發展, 常見問答, and 會員資訊. The main content area has a breadcrumb navigation: 會員資訊 > API授權碼. A sidebar on the left has two tabs: 會員基本資料 (selected) and API授權碼 (highlighted in yellow). The main content area is titled 'API授權碼' and contains text explaining that authorization codes are required for data retrieval via URLs or RESTful APIs. It includes sections for obtaining and updating authorization codes. A large redacted authorization code 'CWB-B8D832AF-' is shown, with the last character '6' highlighted in pink. At the bottom, there is a footer with links to Central Meteorological Bureau policies and contact information.

OPEN WEATHER DATA

公告事項 資料主題 開發指南 應用活化 推廣發展 常見問答 會員資訊

會員資訊 > API授權碼

會員基本資料 API授權碼

API授權碼

本平臺提供透過URL下載檔案以及RESTful API 資料擷取方法取用資料，惟因本平臺採用會員服務機制，需帶入資料項目代碼以及有效會員之授權碼，方可取得各式開放資料。其中，資料項目代碼可至資料清單列表查詢。

一、取得授權碼
會員之授權碼可於下方按鈕取得

取得授權碼 CWB-B8D832AF-
6

二、更新授權碼
一旦更新授權碼後，舊的授權碼將永久失效，並且更新授權碼後七日內無法再進行更新。

更新授權碼

中央氣象局 | 隱私權保護政策 | 資訊安全政策
地址：100006臺北市中正區公園路64號 繼機：(02)2349-1000(代表號)

請複製並保存

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

氣候
天氣警特報
數值預報
天文
資料下載總排行 >

局屬氣象站-現在天氣觀測報告	
檔案下載	JSON XML
資料擷取API服務說明網址	API
資料集類型	rawData
資料集描述	局屬氣象站資料(現在天氣觀測報告)-本局局屬有人氣象站資料
主要欄位說明	STID、STNM、TIME、LAT、LON、ELEV、WDIR、WDSD、TEMP、HUMD、PRES、24R、H_FX、H_XD、H_FXT、H_F10、H_10D、H_F10T、H_UVI、D_TX、D_TXT、D_TS、VIS、Weather、CITY、CITY_SN、TOWN、TOWN_SN
資料集提供機關	中央氣象局
更新頻率	10分鐘
授權方式	政府資料開放授權條款-第1版
授權說明網址	http://data.gov.tw/license
計費方式	免費
編碼格式	UTF-8
資料集提供機關聯絡人	蔡先生
資料集提供機關聯絡人電話	02-23497970
備註(說明資料)	說明資料

回上頁

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

GET /v1/rest/datastore/0-A0003-001 局屬氣象站-現在天氣觀測報告

局屬氣象站資料(現在天氣觀測報告)-本局局屬有人氣象站資料

Parameters

Try it out

Name	Description
Authorization * required string (query)	氣象開放資料平台會員授權碼 Authorization - 氣象開放資料平台會員授權碼
limit number(\$int) (query)	限制最多回傳的資料，預設為回傳全部筆數 limit - 限制最多回傳的資料，預設為回傳全部
offset number(\$int) (query)	指定從第幾筆後開始回傳，預設為第 0 筆開始回傳 offset - 指定從第幾筆後開始回傳，預設為第 0 筆
format string (query)	回傳資料格式，預設為 json 格式 Available values : JSON, XML --
locationName array[string] (query)	測站代碼，請參考 https://e-service.cwb.gov.tw/wdps/obs/state.htm ，預設為全部回傳
elementName array[string] (query)	氣象因子，預設為全部回傳 Available values : TIME, ELEV, WDIR, WDSD, TEMP, HUMD, PRES, 24R, H_FX, H_XD, H_FXT, H_F10, H_10D, H_F10T, H_UVI, D_TX, D_TXT, D_TN, D_TNT, D_TS, VIS Weather

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

GET /v1/rest/datastore/0-A0003-001 局屬氣象站-現在天氣觀測報告

局屬氣象站資料(現在天氣觀測報告)-本局局屬有人氣象站資料

Cancel

Name	Description
Authorization * required	氣象開放資料平台會員授權碼 string (query) 38D832AF-8709-4117-A01C-A50B699380D6
limit	限制最多回傳的資料，預設為回傳全部筆數 number(\$int) (query) limit - 限制最多回傳的資料，預設為回傳全部
offset	指定從第幾筆後開始回傳，預設為第 0 筆開始回傳 number(\$int) (query) offset - 指定從第幾筆後開始回傳，預設為第 0 筆開始回傳
format	回傳資料格式，預設為 json 格式 string (query) --
locationName	測站代碼，請參考 https://e-service.cwb.gov.tw/wdps/obs/state.htm ，預設為全部回傳 array[string] (query) Add item
elementName	氣象因子，預設為全部回傳 array[string] (query) TIME

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

CWB-B8D832AF-8709-4117-A01C-A50B699

limit
number(\$int)
(query)

限制最多回傳的資料，預設為回傳全部筆數
limit - 限制最多回傳的資料，預設為回傳全部

offset
number(\$int)
(query)

指定從第幾筆後開始回傳，預設為第 0 筆開始回傳
offset - 指定從第幾筆後開始回傳，預設為第 0 筆

format
string
(query)

回傳資料格式，預設為 json 格式
--

locationName
array[string]
(query)

測站代碼，請參考<https://e-service.cwb.gov.tw/wdps/obs/state.htm>，預設為全部回傳
Add item

elementName
array[string]
(query)

氣象因子，預設為全部回傳
--
TIME
ELEV
WDIR
WDOP

parameterName
array[string]
(query)

參數，預設為全部回傳
--
CITY
CITY_SN
TOWN

Execute

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

Request URL
<https://opendata.cwb.gov.tw/api/v1/rest/datastore/O-A0003-001?Authorization=CwB-B8D832AF-8709-4117-A01C-A50>

Server response

Code	Details
200	<p>Response body</p> <pre>{ "success": "true", "result": { "resource_id": "O-A0003-001", "fields": [{ "id": "lat", "type": "Double" }, { "id": "lon", "type": "Double" }, { "id": "locationName", "type": "String" }, { "id": "stationId", "type": "String" }, { "id": "obsTime", "type": "Timestamp" }, { "id": "elementName", "type": "String" }] } }</pre> <p>Download</p>

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-encoding: gzip
content-type: application/json; charset=utf-8
date: Tue, 02 Feb 2021 13:52:30 GMT
transfer-encoding: chunked
```

Responses

Code	Description
200	OK

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

```
import requests

url = 'https://opendata.cwa.gov.tw/api/v1/rest/datastore/0-A0003-001?Authorization=CWB-F'
r = requests.get(url)

print(r)
print(r.status_code)

<Response [200]>
200
```

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜合觀氣象資料

```
print(r.text)
```

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

```
j = r.json()
print(j)

{'success': 'true', 'result': {'resource_id': '0-A0003-001', 'fields': [{"id': 'StationName', 'type': 'String'}, {"id': 'StationId', 'type': 'String'}, {"id': 'CoordinateName', 'type': 'String'}, {"id': 'CoordinateFormat', 'type': 'String'}, {"id': 'StationAltitude', 'type': 'String'}, {"id': 'IntScaleValue', 'type': 'Float'}, {"id': 'StationAltitude', 'type': 'CountyName'}, {"id': 'TownName', 'type': 'String'}, {"id': 'CountyCode', 'type': 'String'}, {"id': 'Weather', 'type': 'String'}, {"id': 'VisibilityDescription', 'type': 'String'}, {"id': 'SunshineDuration', 'type': 'Float'}, {"id': 'Precipitation', 'type': 'Float'}, {"id': 'WindDirection', 'type': 'String'}, {"id': 'WindSpeed', 'type': 'Float'}, {"id': 'AirTemperature', 'type': 'Float'}, {"id': 'RelativeHumidity', 'type': 'Float'}, {"id': 'AirPressure', 'type': 'Float'}, {"id': 'UVIndex', 'type': 'Float'}, {"id': 'PeakGustSpeed', 'type': 'Float'}, {"id': 'Occurred_at', 'type': 'DateTime'}]}, 'records': {'Station': [{"StationName": "基隆", "StationId": "46694", "Occurred_at": "2024-02-17T20:50:00+08:00", "GeoInfo": {"Coordinates": [{"CoordinateName": "TWD67", "CoordinateFormat": "WGS84", "StationLatitude": 25.13513, "StationLongitude": 121.73226}, {"CoordinateName": "WGS84", "CoordinateFormat": "WGS84", "StationLatitude": 25.133314, "StationLongitude": 121.74048}], "StationAltitude": "26.7", "County": "仁愛區", "CountyCode": "10017", "TownCode": "10017040", "WeatherElement": {"Weather": "多雲有雨", "CloudCover": "7-10", "SunshineDuration": 4.0, "Now": {"Precipitation": -990.0}, "WindDirection": 0.0, "WindSpeed": 16.6, "RelativeHumidity": 96, "AirPressure": 1014.5, "UVIndex": 0.0, "Max10MinAverage": {"WindDirection": 330.0, "DateTime": "2024-02-17T19:18:00+08:00"}, "GustInfo": {"PeakGustSpeed": 290.0, "DateTime": "2024-02-17T19:16:00+08:00"}, "DailyExtreme": {"DailyHigh": {"Temperature": 19.8, "Occurred_at": {"DateTime": "2024-02-17T11:03:00+08:00"}}, "DailyLow": {"Temperature": 19.8, "Occurred_at": {"DateTime": "2024-02-17T11:03:00+08:00"}}}}]}}
```

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

j

```
{'success': 'true',
'result': {'resource_id': '0-A0003-001',
'fields': [{"id': 'StationName', 'type': 'String'},
{'id': 'StationId', 'type': 'String'},
{'id': 'CoordinateName', 'type': 'String'},
{'id': 'CoordinateFormat', 'type': 'String'},
{'id': 'StationAltitude', 'type': 'String'},
{'id': 'IntScaleValue', 'type': 'Float'},
{'id': 'StationAltitude', 'type': 'Float'},
{'id': 'CountyName', 'type': 'String'},
{'id': 'TownName', 'type': 'String'},
{'id': 'CountyCode', 'type': 'String'},
{'id': 'TownCode', 'type': 'String'},
{'id': 'Weather', 'type': 'String'},
{'id': 'VisibilityDescription', 'type': 'String'},
{'id': 'SunshineDuration', 'type': 'Float'},
{'id': 'Precipitation', 'type': 'Float'},
{'id': 'WindDirection', 'type': 'Float'}]
```

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

```
locs = j['records']['Station']
locs
[{'StationName': '基隆',
 'StationId': '466940',
 'ObsTime': {'DateTime': '2024-02-17T20:50:00+08:00'},
 'GeoInfo': {'Coordinates': [{'CoordinateName': 'TWD67',
                                'CoordinateFormat': 'decimal degrees',
                                'StationLatitude': 25.13513,
                                'StationLongitude': 121.73226},
                                {'CoordinateName': 'WGS84',
                                'CoordinateFormat': 'decimal degrees',
                                'StationLatitude': 25.133314,
                                'StationLongitude': 121.74048}],
             'StationAltitude': '26.7',
             'CountyName': '基隆市',
             'TownName': '仁愛區',
             'CountyCode': '10017',
             'TownCode': '10017040'},
 'WeatherElement': {'Weather': '多雲有霧',
                    'VisibilityDescription': '7-10'}
```

Weather 界接

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

```
print(f'{len(locs)} sites')
227 sites

for loc in locs:
    if loc['StationName'] == '臺北':
        print(loc)

{'StationName': '臺北', 'StationId': '466920', 'ObsTime': {'DateTime': '2024-02-17T20:50:00+08:00',
[{'CoordinateName': 'TWD67', 'CoordinateFormat': 'decimal degrees', 'StationLatitude': 25.03941, 'StationLongitude': 121.56266}, {'CoordinateName': 'WGS84', 'CoordinateFormat': 'decimal degrees', 'StationLatitude': 25.037654854}], 'StationAltitude': '6.3', 'CountyName': '臺北市', 'TownName': '中正區', 'CountyCode': '6300', 'WeatherElement': {'Weather': '多雲', 'VisibilityDescription': '16-20', 'SunshineDuration': 7.0, 'CloudCover': 0}, 'WindDirection': 110.0, 'WindSpeed': 0.7, 'AirTemperature': 18.6, 'RelativeHumidity': 87, 'AirPressure': 1010.0, 'Max10MinAverage': {'WindSpeed': 1.5, 'Occurred_at': {'WindDirection': 90.0, 'DateTime': '2024-02-17T18:00:00+08:00'}}, 'WindGustInfo': {'PeakGustSpeed': 3.2, 'Occurred_at': {'WindDirection': 100.0, 'DateTime': '2024-02-17T18:00:00+08:00'}}, 'TemperatureInfo': {'DailyHigh': {'Temperature': 23.4, 'Occurred_at': {'DateTime': '2024-02-17T18:00:00+08:00'}}, 'DailyLow': {'Temperature': 16.6, 'Occurred_at': {'DateTime': '2024-02-17T00:00:00+08:00'}}]}
```

Your Turn 5-1 Weather 界接

界接即時天氣資訊擷取功能 (10 mins)

Task

1. 完成界接 氣象觀測站-全測站逐時氣象資料
2. 完成界接 雨量觀測站-雨量資料

Weather 規格

現在天氣觀測報告

<https://opendata.cwa.gov.tw/> >

觀測 >

氣象觀測站-10分鐘綜觀氣象資料

氣候
天氣警特報
數值預報
天文
資料下載總排行 >

局屬氣象站-現在天氣觀測報告	
檔案下載	JSON XML
資料擷取API服務說明網址	API
資料集類型	rawData
資料集描述	局屬氣象站資料(現在天氣觀測報告)-本局局屬有人氣象站資料
主要欄位說明	STID、STNM、TIME、LAT、LON、ELEV、WDIR、WDSD、TEMP、HUMD、PRES、24R、H_FX、H_XD、H_FXT、H_F10、H_10D、H_F10T、H_UVI、D_TX、D_TXT、D_TS、VIS、Weather、CITY、CITY_SN、TOWN、TOWN_SN
資料集提供機關	中央氣象局
更新頻率	10分鐘
授權方式	政府資料開放授權條款-第1版
授權說明網址	http://data.gov.tw/license
計費方式	免費
編碼格式	UTF-8
資料集提供機關聯絡人	蔡先生
資料集提供機關聯絡人電話	02-23497970
備註(說明資料)	說明資料

回上頁

Your Turn 5-2 Weather Grabber 開發測試

開發即時天氣資訊擷取程式 (45 mins)

SPEC

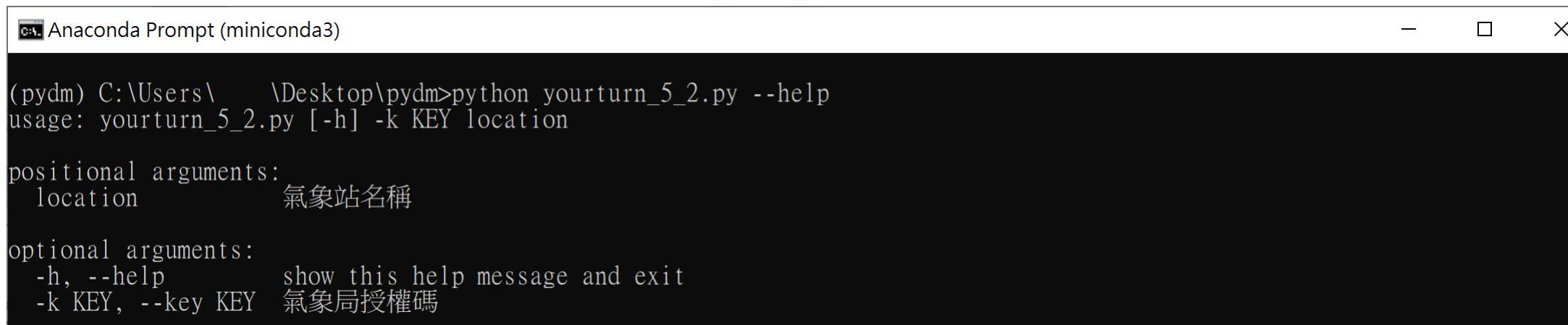
1. 設計為 class，並包裝為 module 以利重複使用
 - Class name 為 WeaG
 - public instance method name 為 grab，可依據輸入之站台名稱取得 觀測時間、溫度、濕度、24小時雨量等資訊
 - 同時支援 有人氣象站 以及 自動氣象站 資料

Your Turn 5-2 Weather Grabber 開發測試

開發即時天氣資訊擷取程式 (45 mins)

SPEC

2. 可於命令提示字元測試此 module



```
Anaconda Prompt (miniconda3)
(pydm) C:\Users\    \Desktop\pydm>python yourturn_5_2.py --help
usage: yourturn_5_2.py [-h] -k KEY location

positional arguments:
  location          氣象站名稱

optional arguments:
  -h, --help        show this help message and exit
  -k KEY, --key KEY 氣象局授權碼
```

Weather Grabber

擷取即時天氣資料並於網頁顯示

Tasks:

1. 網站架設
2. 測試

Weather 網站架設

網站架設

<https://flask.palletsprojects.com/>

> Quickstart - A Minimal Application

```
1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def wea():
7     content = '''
8         <form>
9             <input type="text" name="location"> <input type="submit" value="確定">
10        </form>
11        ...
12
13    return content
```

Note

Suggest to upgrade Flask to the latest version

Weather 網站架設

網站架設

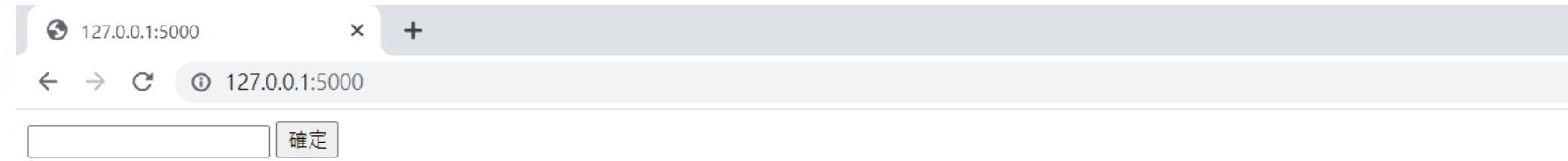
```
1 app.run()
```

```
* Serving Flask app "__main__" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
```

```
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [10/Sep/2021 08:36:12] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Sep/2021 08:36:15] "GET /?location=中和 HTTP/1.1" 200 -
```

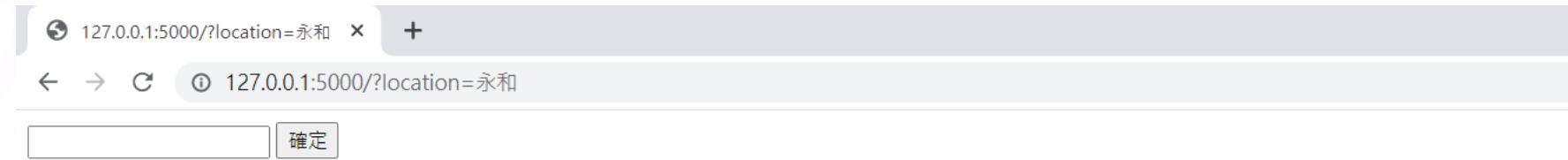
Weather 網站架設

網站架設



Weather 網站架設

網站架設



Weather 網站架設

網站架設

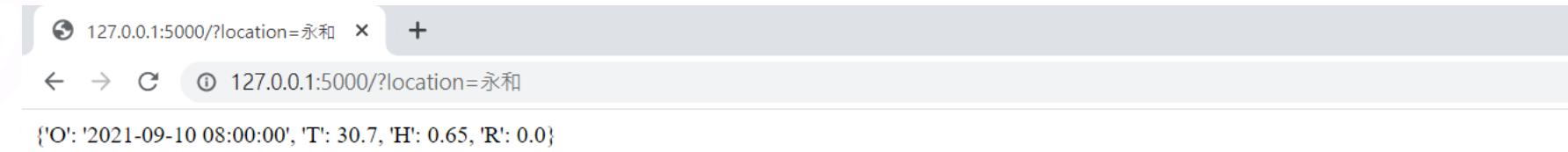
```
1 from flask import Flask, request
2
3 from yourturn_5_2 import WeaG
4
5 w = WeaG('CWB-F')
6 app = Flask(__name__)
7
8 @app.route('/')
9 def wea():
10     loc = request.args.get('location')
11
12     if loc:
13         content = f'{w.grab(loc)}'
14     else:
15         content = ''
16         <form>
17             <input type="text" name="location"> <input type="submit" value="確定">
18         </form>
19         ...
20
21     return content
```

```
1 app.run()
```

* Serving Flask app " main " (lazy loading)

Weather 網站架設

網站架設

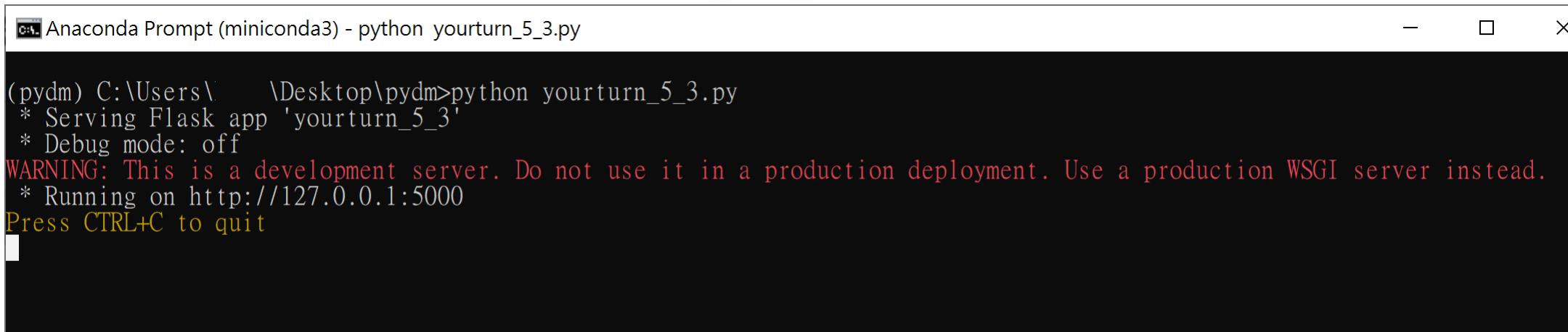


Your Turn 5-3 Weather Web 開發

開發即時天氣資訊網頁服務 (15 mins)

SPEC

- 服務須能夠由 命令提示字元 啟動



Anaconda Prompt (miniconda3) - python yourturn_5_3.py

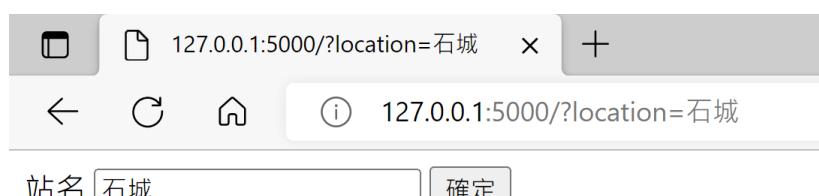
```
(pydm) C:\Users\    \Desktop\pydm>python yourturn_5_3.py
 * Serving Flask app 'yourturn_5_3'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Your Turn 5-3 Weather Web 開發

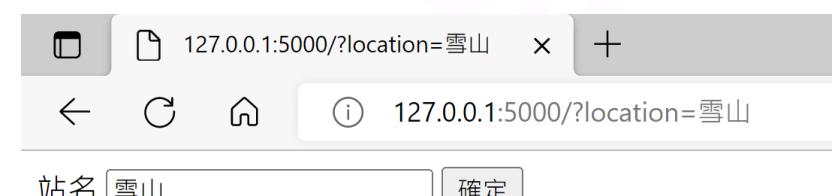
開發即時天氣資訊網頁服務 (15 mins)

SPEC

2. 網頁呈現如下



2022-08-20 21:00:00 溫度:28.2°C, 濕度:83%, 雨量:0.0mm



no such site!

Weather Crawler

爬取即時天氣資料並顯示

Tasks:

1. 資料
2. 界接
3. 規格
4. 開發
5. 測試

Weather 資料

氣象局官網

<https://www.cwa.gov.tw/> >

天氣 >

縣市測站列表 > 臺北

交通部中央氣象局
Central Weather Bureau

回首頁 | EN | 網站導覽 | 意見箱 | 常見問答 | 關於本局 | 小 | 中 | 大 | 🔍 | 🌐 | 📁 | 🌡 | 🔍

警特報 天氣 生活 地震 海象 氣候 資料 知識與天文 常用服務

！ 陸上強風特報 |

臺北測站觀測資料

過去24小時資料 過去24小時變化圖 ⚙️測站地圖位置

觀測時間	溫度 °C	天氣	風向	風力 (級)	陣風 (級)	能見度 (公里)	相對濕度 (%)	海平面氣壓 (百帕)	當日累積雨量(毫米)	日照時數
10/20 21:50	24.3		南	1	-	>30	83	1015.4	0.0	0.5
10/20 21:40	24.3		東南	1	-	>30	83	1015.3	0.0	0.5
10/20 21:30	24.3		東南東	1	-	>30	83	1015.3	0.0	0.5
10/20 21:20	24.3		東南東	1	-	>30	83	1015.4	0.0	0.5
10/20 21:10	24.4		東南	1	-	>30	82	1015.4	0.0	0.5
10/20 21:00	24.3		東	1	3	>30	82	1015.4	0.0	0.5
10/20 20:50	24.2		東	1	-	>30	83	1015.5	0.0	0.5
10/20 20:40	24.3		東	1	-	>30	82	1015.5	0.0	0.5
10/20 20:30	24.3		東南東	2	-	>30	82	1015.4	0.0	0.5
10/20 20:20	24.3		東南東	2	-	>30	82	1015.3	0.0	0.5
10/20 20:10	24.3		東	2	-	>30	82	1015.6	0.0	0.5

Weather 資料

氣象局官網

<https://www.cwa.gov.tw/> >

天氣 >
縣市測站列表 >
臺北

```
1 import requests
2
3 url = 'https://www.cwb.gov.tw/V8/C/W/OBS_Station.html?ID=46692'
4 r = requests.get(url)
5 print(r.text)

!DOCTYPE html>
!--[if IE 8]> <html lang="zh-Hant-TW" class="ie8"> <![endif]-->
!--[if IE 9]> <html lang="zh-Hant-TW" class="ie9"> <![endif]-->
!--[if !IE]><!--
html lang="zh-Hant-TW">
!--<![endif]-->
head>
<title>測站觀測資料 | 交通部中央氣象局</title>
<!-- Meta -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="">
<meta name="author" content="">
<!--Facebook_meta_begin-->
<!-- Facebook for W--><meta property="og:url" content="https://www.cwb.gov.tw/V8/C/W/OBS_Station.html" /><meta property="og:title" content="天氣 - 中央氣象局全球資訊網" /><meta property="og:image" content="https://www.cwb.gov.tw/Data/fcst_img/cloud_weather.png" /><meta property="og:description" content="氣象局觀測站資料" />
```

Weather 資料

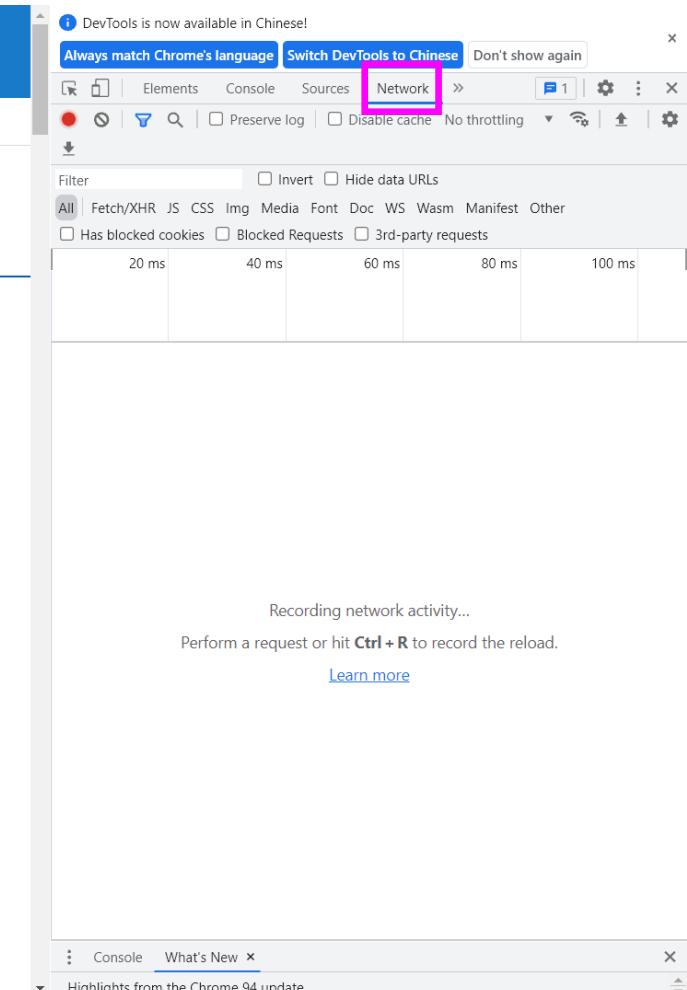
F12 或 Ctrl + Shift + I

氣象局官網

[https://www.cwa.gov.tw/ >](https://www.cwa.gov.tw/)

天氣 >
縣市測站列表 >
臺北

觀測時間	溫度 °C	天氣
10/20 22:00	24.3	多雲
10/20 21:50	24.3	多雲
10/20 21:40	24.3	多雲
10/20 21:30	24.3	多雲
10/20 21:20	24.3	多雲
10/20 21:10	24.4	多雲
10/20 21:00	24.3	多雲
10/20 20:50	24.2	多雲
10/20 20:40	24.3	多雲
10/20 20:30	24.3	多雲
10/20 20:20	24.3	多雲



Weather 資料

氣象局官網

[https://www.cwa.gov.tw/ >](https://www.cwa.gov.tw/)

天氣 >
縣市測站列表 >
臺北

The screenshot shows the Central Weather Bureau's website for Taipei. At the top, there is a banner with a red exclamation mark and the text "陸上強風特報". Below the banner, the page title is "臺北測站觀測資料". There are three tabs at the bottom of this section: "過去24小時資料" (selected), "過去24小時變化圖", and "測站地圖位置". The main content area displays a table of weather data for the past 24 hours. The table has three columns: "觀測時間" (Observation Time), "溫度 °C" (Temperature in °C), and "天氣" (Weather). The data shows temperatures ranging from 24.2°C to 24.4°C and mostly cloudy conditions.

觀測時間	溫度 °C	天氣
10/20 22:00	24.3	☁️
10/20 21:50	24.3	☁️
10/20 21:40	24.3	☁️
10/20 21:30	24.3	☁️
10/20 21:20	24.3	☁️
10/20 21:10	24.4	☁️
10/20 21:00	24.3	☁️
10/20 20:50	24.2	☁️
10/20 20:40	24.3	☁️
10/20 20:30	24.3	☁️
10/20 20:20	24.3	☁️

F5 重新載入

The screenshot shows the Network tab in the Chrome DevTools developer console. It lists 46 requests made during the page load. The requests are categorized by type (script, CSS, JS, etc.) and initiator (OBS_Stat...). The total transferred data is 29.1 kB, and the total resources are 2.2 MB. The page finished loading in 5.31 seconds, and the DOM content was loaded in 196 ms. The status bar at the bottom indicates "Highlights from the Chrome 94 update".

Weather 資料

尋找適用檔案

氣象局官網

[https://www.cwa.gov.tw/ >](https://www.cwa.gov.tw/)

天氣 >
縣市測站列表 >
臺北

The screenshot shows a table of weather data for Taipei. The columns are '觀測時間' (Observation Time), '溫度 °C' (Temperature °C), and '天氣' (Weather). The data is as follows:

觀測時間	溫度 °C	天氣
10/20 22:00	24.3	雲
10/20 21:50	24.3	雲
10/20 21:40	24.3	雲
10/20 21:30	24.3	雲
10/20 21:20	24.3	雲
10/20 21:10	24.4	雲
10/20 21:00	24.3	雲
10/20 20:50	24.2	雲
10/20 20:40	24.3	雲
10/20 20:30	24.3	雲
10/20 20:20	24.3	雲

The screenshot shows the Network tab in the Chrome DevTools. A request for '46692.html?T=590' is highlighted with a pink box. The table below lists other network requests:

Name	Time	Size
fontawesome-webfont.woff2?v=	10/20 22:00 24.376	隱 南 0.81 2.32 >3083
icomoon.woff2??z2agr	10/15.4 0.0 0.5 10/20	
o-0IpQlx3QUIC5A4PNr5TRF.ttf	21:50 24.376	隱 南 0.61 -->3083
_header.html?v=20211020	10/15.4 0.0 0.5 10/20	
_footer.html?v=20200505		
STMap.json		
46692.html?T=590	10/20 22:00 24.376	隱 東南 0.71 -->3083
util.js	10/15.3 0.0 0.5 10/20	
pagejs.js	21:30 24.376	隱 東南東 1.01 -->3083
AAccessibility.jpg	10/15.3 0.0 0.5 10/20	
cwb-logoBlue.svg	21:20 24.376	隱 東南東 1.21 -->3083
js?id=G-K6HENPOXVS&l=data	10/15.4 0.0 0.5 10/20	
wmts?SERVICE=WMTS&REQUEST=GetCapabilities	21:10 24.476	隱 東南 0.91 -->3082
analytics.js	10/15.4 0.0 0.5 10/20	
collect?v=1&_v=j94&a=133472	21:00 24.376	隱 東 0.91 4.53 >3082
wmts?SERVICE=WMTS&REQUEST=GetFeatureInfo	10/15.4 0.0 0.5 10/20	
07.svg	20:50 24.276	隱 東 1.01 -->3083
07.svg	10/15.5 0.0 0.5 10/20	
04.svg	20:40 24.376	隱 東 1.21 -->3082
favicon.ico	10/15.5 0.0 0.5 10/20	
favicon.ico	20:30 24.376	隱 東南東 1.92 -->3082
collect?v=2&tid=G-K6HENPOXVS&l=	10/15.4 0.0 0.5 10/20	
07.svg	20:20 24.376	隱 東南東 2.42 -->3082

Weather 資料

右鍵複製連結

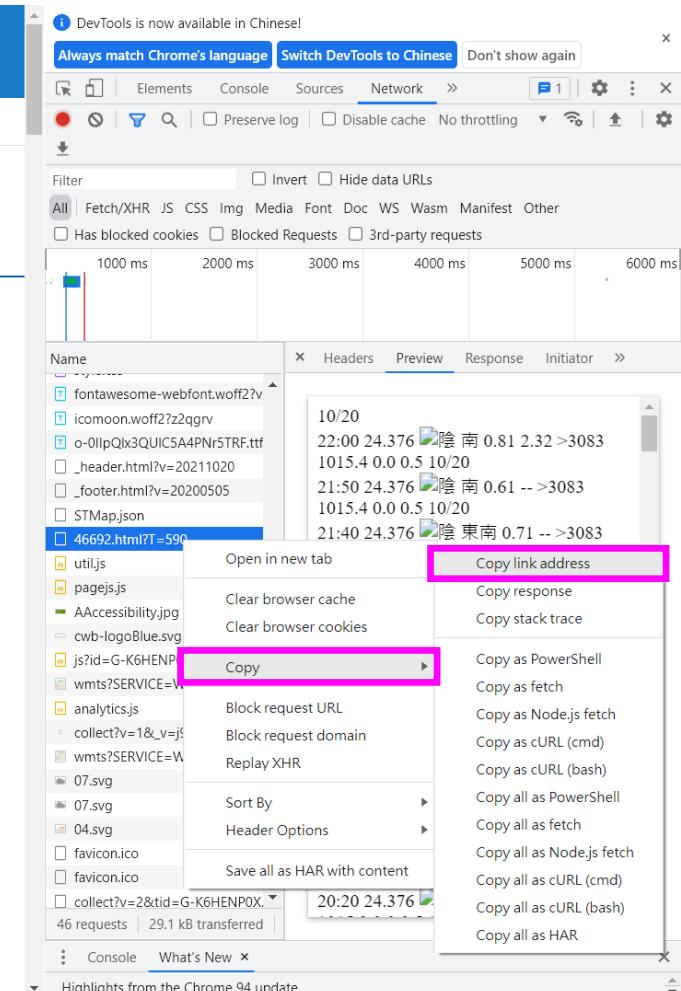
氣象局官網

[https://www.cwa.gov.tw/ >](https://www.cwa.gov.tw/)

天氣 >
縣市測站列表 >
臺北

The screenshot shows a table of weather data for Taipei. The columns are '觀測時間' (Observation Time), '溫度 °C' (Temperature °C), and '天氣' (Weather). The data is as follows:

觀測時間	溫度 °C	天氣
10/20 22:00	24.3	雲
10/20 21:50	24.3	雲
10/20 21:40	24.3	雲
10/20 21:30	24.3	雲
10/20 21:20	24.3	雲
10/20 21:10	24.4	雲
10/20 21:00	24.3	雲
10/20 20:50	24.2	雲
10/20 20:40	24.3	雲
10/20 20:30	24.3	雲
10/20 20:20	24.3	雲



Weather 資料

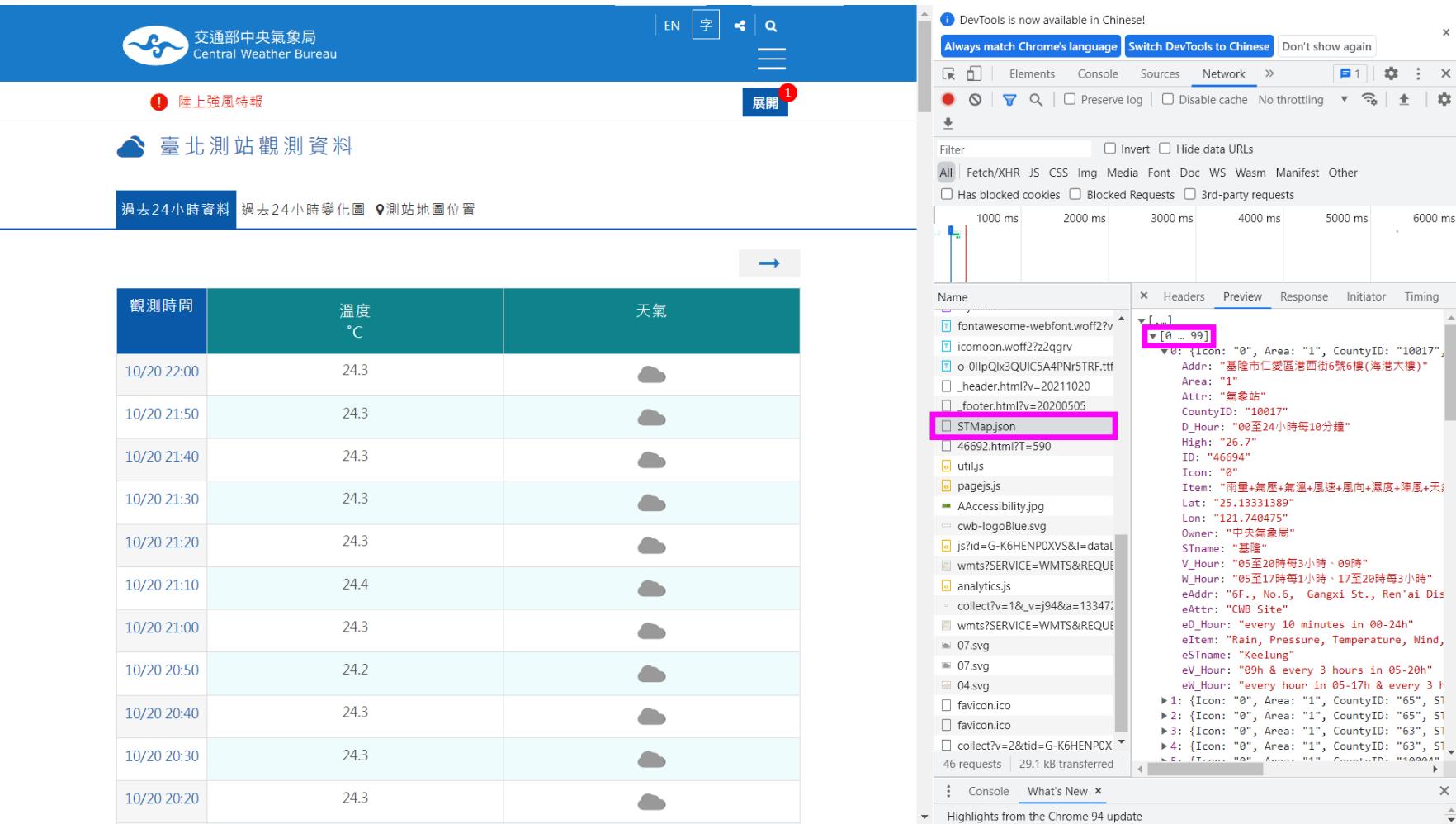
發現適用檔案

氣象局官網

<https://www.cwa.gov.tw/> >

天氣 >
縣市測站列表 >
臺北

觀測時間	溫度 °C	天氣
10/20 22:00	24.3	
10/20 21:50	24.3	
10/20 21:40	24.3	
10/20 21:30	24.3	
10/20 21:20	24.3	
10/20 21:10	24.4	
10/20 21:00	24.3	
10/20 20:50	24.2	
10/20 20:40	24.3	
10/20 20:30	24.3	
10/20 20:20	24.3	



Weather 界接

現在天氣觀測報告

<https://www.cwa.gov.tw/> >

天氣 >
縣市測站列表 >
臺北

```
1 import requests
2
3 url = 'https://www.cwb.gov.tw/V8/C/W/Observe/MOD/24hr/46692.html'
4 r = requests.get(url)
5 print(r.text)

<tr data-cstname="臺北" data-estname="Taipei" data-countyid="63">
    <th scope="row" headers="time" class="is_show">02/26<br c1
ble-md"> 13:30</th>
    <td headers="temp" class="is_show"><span class="tem-C is-a
4.3</span><span class="tem-F is-hidden">76</span></td>
    <td headers="weather" class="is_show"></td>
    <td headers="w-1"><span class="wind">東</span></td>
    <td headers="w-2"><span class="wind_2 is-active">1.0</span
ass="wind_1 is-hidden">1</span></td>
    <td headers="w-3"><span class="wind_2 is-active">-</span><
s="wind_1 is-hidden">-</span></td>
    <td headers="visible-1">11-15</td><td headers="hum">70</td
    <td headers="pre">1009.1</td>
    <td headers="rain">0.0</td>
    <td headers="sunlight">0.3</td>
</tr><tr>
    <td headers="time" class="is_show">02/26<br c1
ble-md"> 13:30</td>
    <td headers="temp" class="is_show"><span class="tem-C is-a
4.3</span><span class="tem-F is-hidden">76</span></td>
    <td headers="weather" class="is_show"></td>
    <td headers="w-1"><span class="wind">東</span></td>
    <td headers="w-2"><span class="wind_2 is-active">1.0</span
ass="wind_1 is-hidden">1</span></td>
    <td headers="w-3"><span class="wind_2 is-active">-</span><
s="wind_1 is-hidden">-</span></td>
    <td headers="visible-1">11-15</td><td headers="hum">70</td
    <td headers="pre">1009.1</td>
    <td headers="rain">0.0</td>
    <td headers="sunlight">0.3</td>
</tr>
```

Weather 界接

pip install beautifulsoup4

現在天氣觀測報告

<https://www.cwa.gov.tw/> >

天氣 >

縣市測站列表 >

臺北

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 url = 'https://www.cwb.gov.tw/V8/C/W/Observe/MOD/24hr/46692.html'
5 r = requests.get(url)
6 soup = BeautifulSoup(r.text, 'html.parser')
7 trs = soup.find_all('tr')
8 print(trs)

[<tr data-countyid="63" data-cstname="臺北" data-estname="Taipei">
<th class="is_show" headers="time" scope="row">02/26<br class="visible-md"/> 13:30</th>
<td class="is_show" headers="temp"><span class="tem-C is-active">24.3</span><span class="te
dden">76</span></td>
<td class="is_show" headers="weather"></td>
<td headers="w-1"><span class="wind">東</span></td>
<td headers="w-2"><span class="wind_2 is-active">1.0</span><span class="wind_1 is-hidden">1
td>
<td headers="w-3"><span class="wind_2 is-active">-</span><span class="wind_1 is-hidden">-<
>
<td headers="visible-1">11-15</td><td headers="hum">70</td>
<td headers="pre">1009.1</td>
<td headers="rain">0.0</td>
<td headers="sunlight">0 3</td>
```

Weather 界接

現在天氣觀測報告

<https://www.cwa.gov.tw/> >

天氣 >

縣市測站列表 >

臺北

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 url = 'https://www.cwb.gov.tw/V8/C/W/Observe/MOD/24hr/46692.html'
5 r = requests.get(url)
6 soup = BeautifulSoup(r.text, 'html.parser')
7 tr = soup.find('tr')
8 print(tr)

<tr data-countyid="63" data-cstname="臺北" data-estname="Taipei">
<th class="is_show" headers="time" scope="row">02/26<br class="visible-md"/> 13:50</th>
<td class="is_show" headers="temp"><span class="tem-C is-active">24.3</span><span class="te
den">76</span></td>
<td class="is_show" headers="weather"></td>
<td headers="w-1"><span class="wind">西北</span></td>
<td headers="w-2"><span class="wind_2 is-active">1.5</span><span class="wind_1 is-hidden">1
d>
<td headers="w-3"><span class="wind_2 is-active">-</span><span class="wind_1 is-hidden">-</
<td headers="visible-1">11-15</td><td headers="hum">70</td>
<td headers="pre">1008.9</td>
<td headers="rain">0.0</td>
<td headers="sunlight">0.3</td>
</tr>
```

Weather 界接

現在天氣觀測報告

<https://www.cwa.gov.tw/> >

天氣 >

縣市測站列表 >

臺北

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 url = 'https://www.cwb.gov.tw/V8/C/W/Observe/MOD/24hr/46692.html'
5 r = requests.get(url)
6 soup = BeautifulSoup(r.text, 'html.parser')
7 tr = soup.find('tr')
8 print(tr.find(headers="time"))
9 print(tr.find(headers="temp"))
10 print(tr.find(headers="hum"))
11 print(tr.find(headers="rain"))

<th class="is_show" headers="time" scope="row">02/26<br class="visible-md"/> 13:50</th>
<td class="is_show" headers="temp"><span class="tem-C is-active">24.3</span><span class="te
den">76</span></td>
<td headers="hum">70</td>
<td headers="rain">0.0</td>
```

Weather 界接

現在天氣觀測報告

<https://www.cwa.gov.tw/> >

天氣 >

縣市測站列表 >

臺北

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 url = 'https://www.cwb.gov.tw/V8/C/W/Observe/MOD/24hr/46692.html'
5 r = requests.get(url)
6 soup = BeautifulSoup(r.text, 'html.parser')
7 tr = soup.find('tr')
8 print(f'時間 : {tr.find(headers="time").text}')
9 print(f'溫度 : {tr.find(headers="temp").find("span").text}')
10 print(f'濕度 : {tr.find(headers="hum").text}')
11 print(f'雨量 : {tr.find(headers="rain").text}')
```

時間 : 02/26 14:00

溫度 : 24.0

濕度 : 72

雨量 : 0.0

Your Turn 5-4

運用爬網技術重製即時天氣資訊網頁服務 (30 mins)

1. 以爬網技術重製 Your Turn 5-2 規格之 WeaG
2. 套用至 Your Turn 5-3 之 Web

Hint

站名對應運用 SMap.json

Word Cloud

建立文字雲

1. 環境準備
2. 選擇文章
3. 文章斷詞
4. 計算詞頻
5. 詞頻繪圖

Word Cloud

建立文字雲

1. 環境準備

Python 套件

- ① `jieba`
- ② `wordcloud`
- ③ `matplotlib`

檔案

- ① `jieba` 字典檔 `dict.txt` (可自訂)
- ② 字型 `TaipeiSansTCBeta-Regular.ttf` (可自選)

Word Cloud

建立文字雲

2. 選擇文章

```
text = '''your content''' # 指定文章為字串
```

Word Cloud

建立文字雲

3. 文章斷詞

```
import jieba

jieba.set_dictionary('dict.txt') # 設定字典檔，可忽略移除
jieba.add_word('') # 持續增加字典
segs = jieba.lcut(text, cut_all=False) # 斷詞
```

Word Cloud

建立文字雲

4. 計算詞頻

```
from collections import Counter

dictionary = Counter(segs) # 計算詞頻
# 持續調整停用詞
stopwords = [',', '，', '；', '；', '\n', '、', '的', '是', '也', '有']
for s in stopwords:
    dictionary.pop(s, None) # 詞頻移除停用詞

dictionary.most_common() # 檢視高頻用詞
```

Word Cloud

建立文字雲

5. 詞頻繪圖

```
import wordcloud
import matplotlib.pyplot as plt

wc = wordcloud.WordCloud(font_path='TaipeiSansTCBeta-Regular.ttf',
                           max_words=20,
                           width=800, height=80,
                           background_color='white')
wc.generate_from_frequencies(dictionary) # 製作文字雲
plt.imshow(wc) # 顯示文字雲
wc.to_file('your_wordcloud.png') # 輸出文字雲
```

Python 6 Development

Python Virtual Environment, Python IDE, Python Executable

Python Virtual Environment

Python 虛擬環境目的

- 單系統並存不同的模組版本，確認或避免**不同模組版本**影響 Python 應用程式的運作
- 單系統並存不同的 Python 版本，確認或避免**不同 Python 版本**影響 Python 應用程式的運作

Python 虛擬環境常用工具

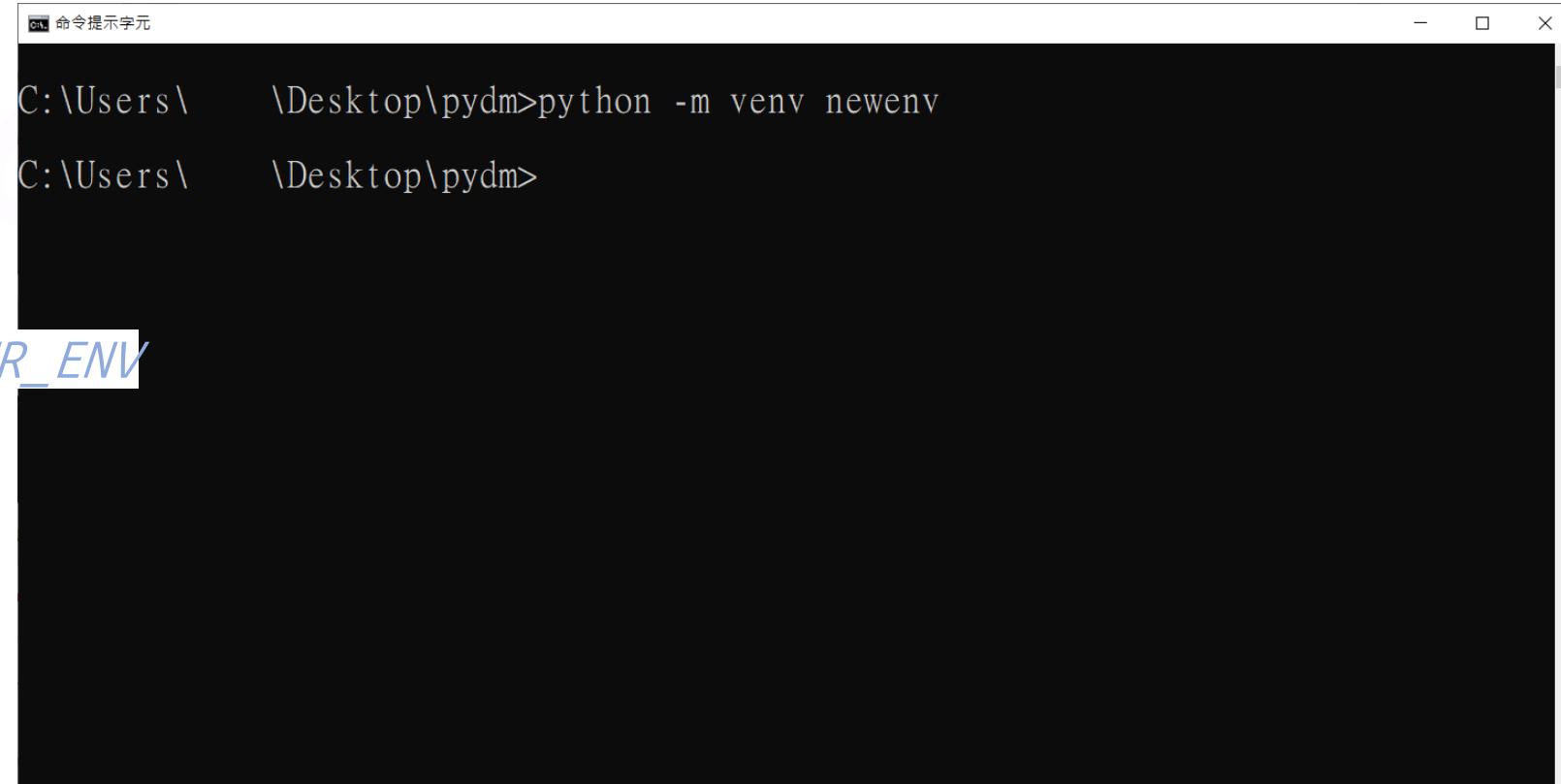
- Python venv → 原生、不同模組版本
- Anaconda, Miniconda → 非原生、不同模組版本、不同 Python 版本

Python Virtual Environment

Python venv

建立虛擬環境

`python -m venv YOUR_ENV`



A screenshot of a Windows Command Prompt window titled "命令提示字元". The window shows the command `python -m venv newenv` being run in the directory `C:\Users\...\Desktop\pydm`. After the command is executed, the prompt changes to `C:\Users\...\Desktop\pydm>`.

Python Virtual Environment

Python venv

建立虛擬環境

```
python -m venv YOUR_ENV
```

pydm

名稱

- .ipynb_checkpoints
- mypath
- newenv**
- trees

pydm > newenv >

名稱

- Include
- Lib
- Scripts
- pyvenv.cfg

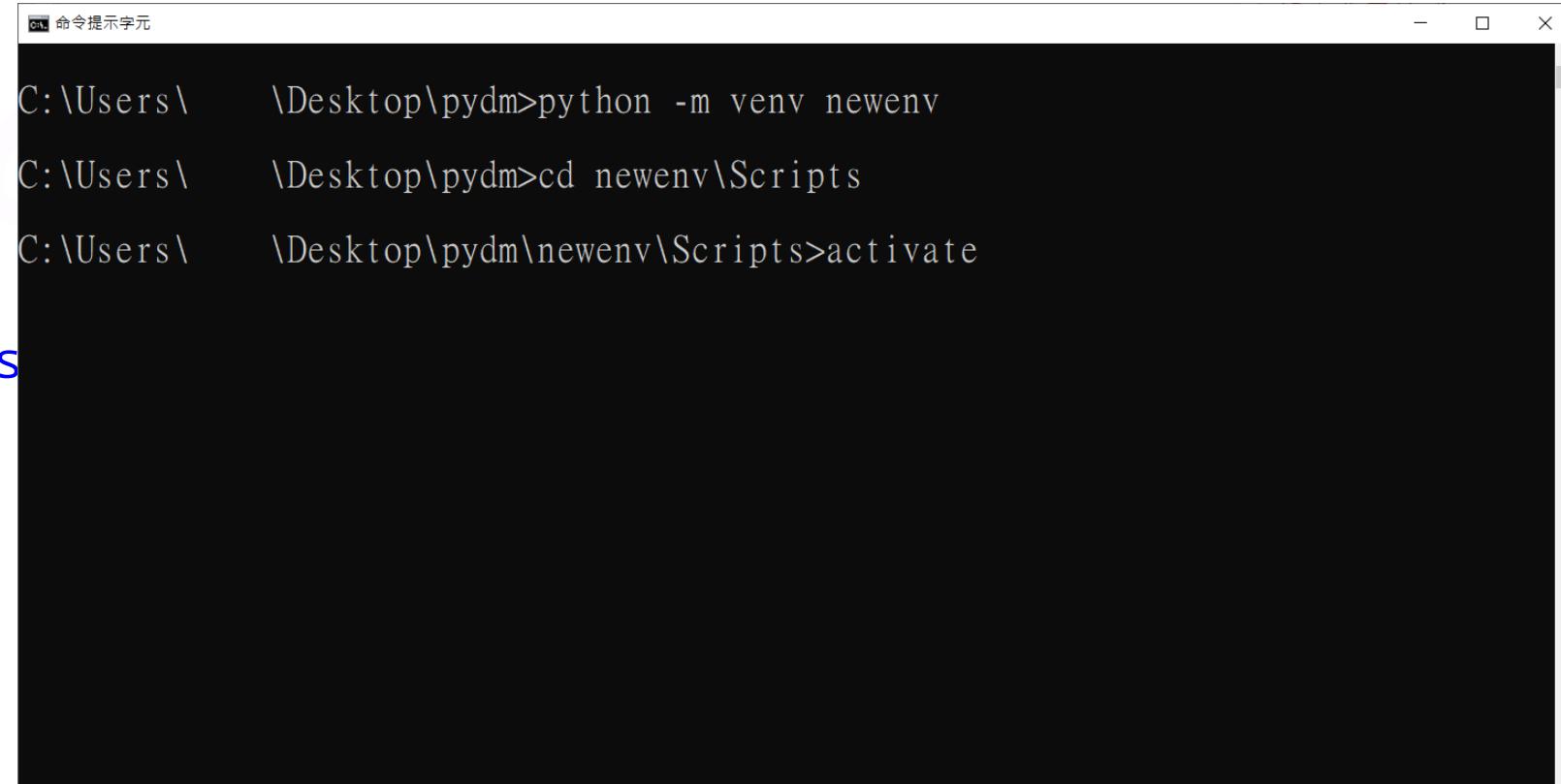
Python Virtual Environment

Python venv

啟動虛擬環境

cd *YOUR_ENV/Scripts*

activate



The screenshot shows a Windows Command Prompt window titled "命令提示字元". The command history is as follows:

```
C:\Users\    \Desktop\pydm>python -m venv newenv
C:\Users\    \Desktop\pydm>cd newenv\Scripts
C:\Users\    \Desktop\pydm\newenv\Scripts>activate
```

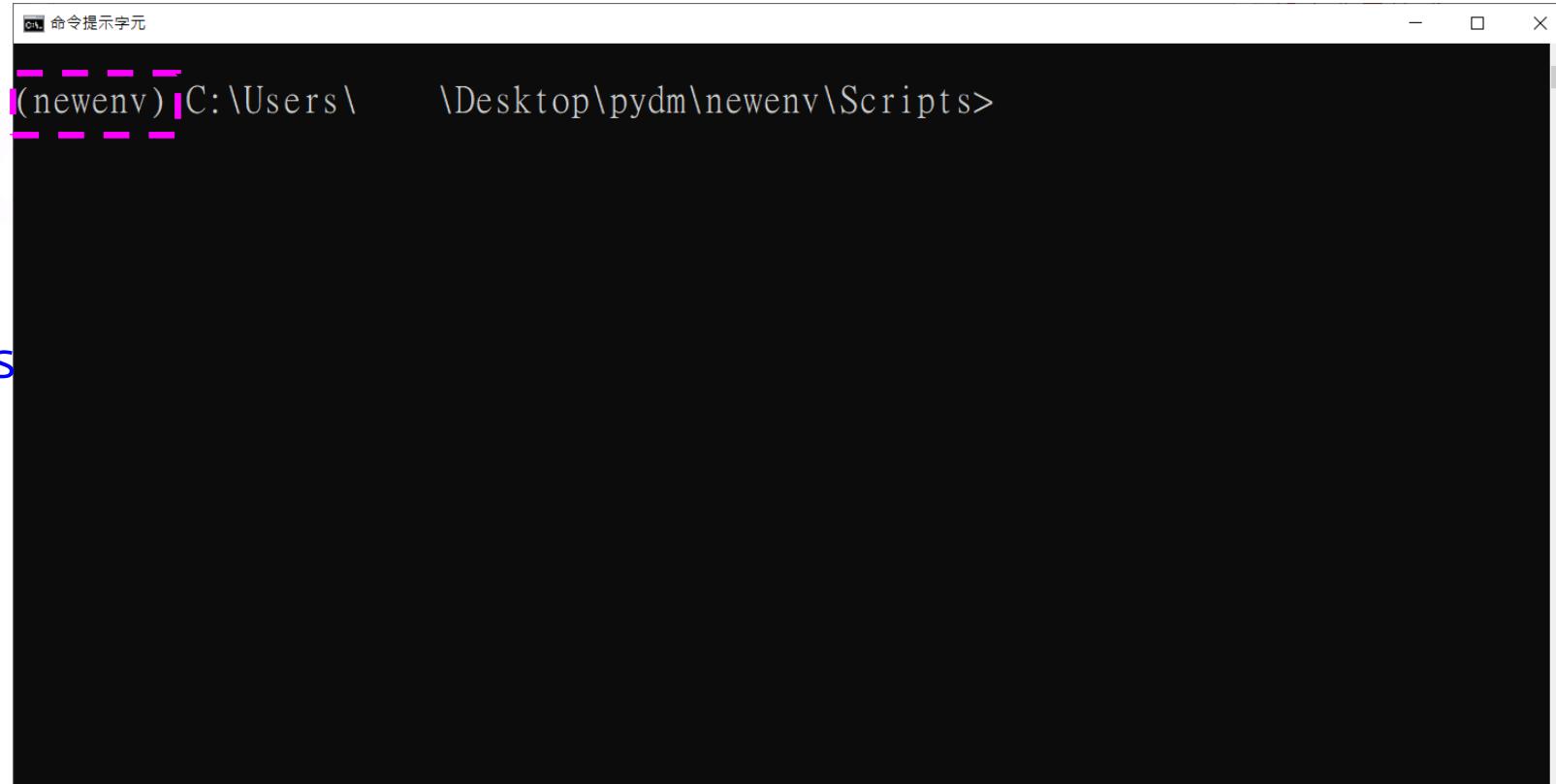
Python Virtual Environment

Python venv

啟動虛擬環境

```
cd YOUR_ENV/Scripts
```

```
activate
```



命令提示字元
(newenv) C:\Users\...\Desktop\pydm\newenv\Scripts>

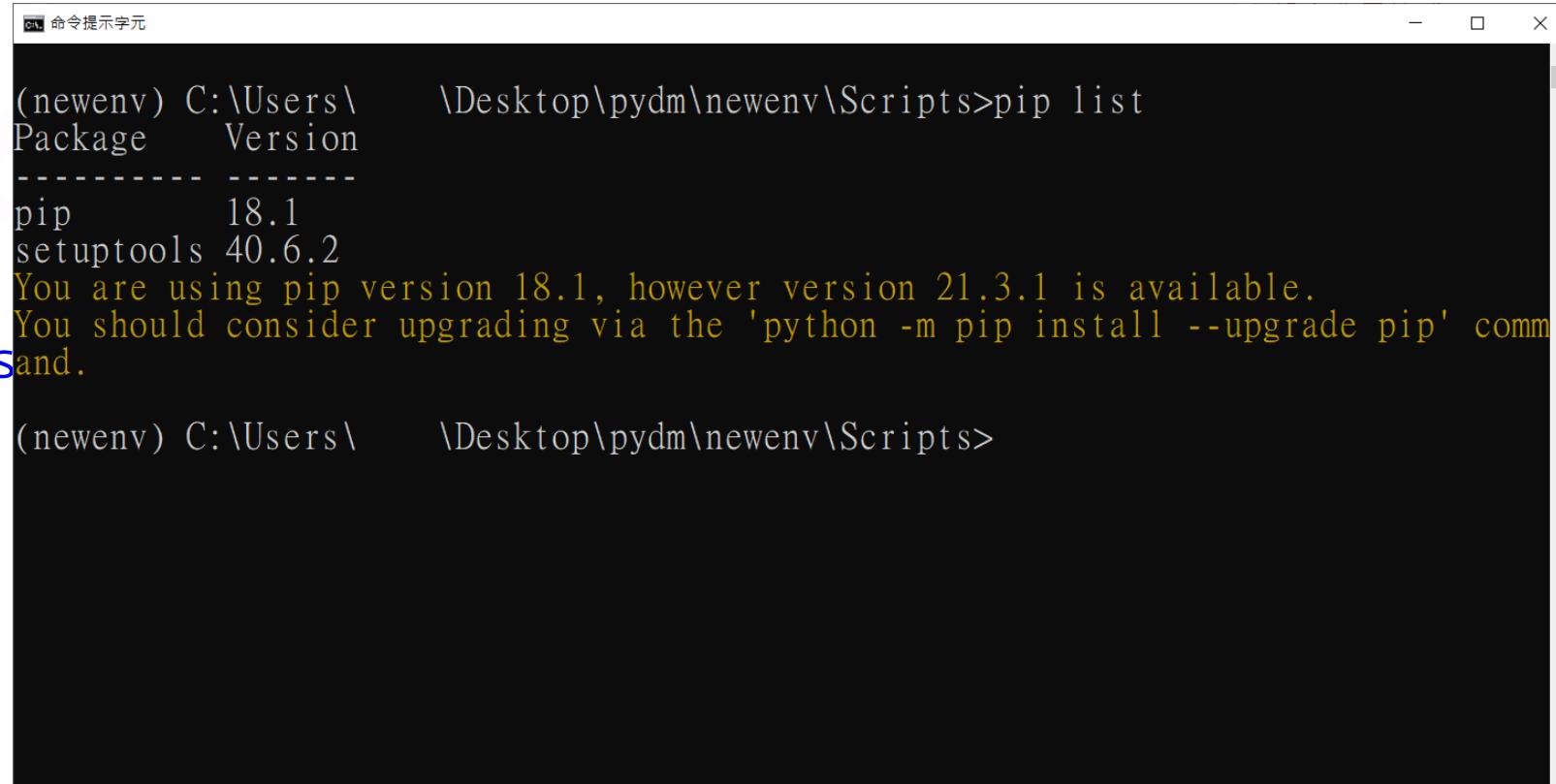
Python Virtual Environment

Python venv

啟動虛擬環境

cd *YOUR_ENV/Scripts*

activate



```
命令提示字元
(newenv) C:\Users\          \Desktop\pydm\newenv\Scripts>pip list
Package      Version
-----
pip          18.1
setuptools  40.6.2
You are using pip version 18.1, however version 21.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(newenv) C:\Users\          \Desktop\pydm\newenv\Scripts>
```

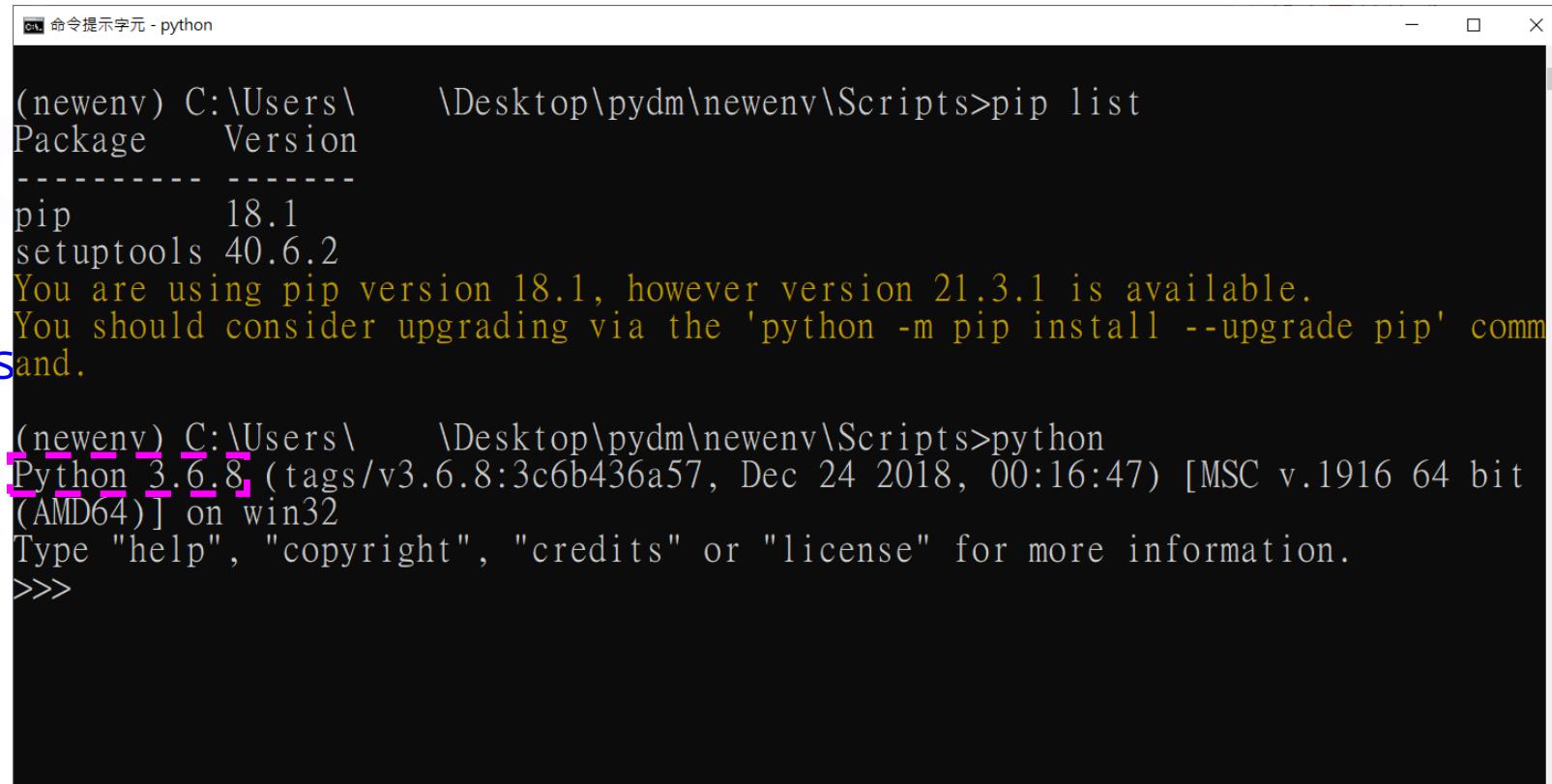
Python Virtual Environment

Python venv

啟動虛擬環境

cd *YOUR_ENV/Scripts*

activate



```
命令提示字元 - python
(newenv) C:\Users\          \Desktop\pydm\newenv\Scripts>pip list
Package      Version
-----
pip          18.1
setuptools  40.6.2
You are using pip version 18.1, however version 21.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(newenv) C:\Users\          \Desktop\pydm\newenv\Scripts>python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python Virtual Environment

Python venv

啟動虛擬環境

`cd YOUR_ENV/Scripts`

`activate`

```
命令提示字元 - python
(newenv) C:\Users\          \Desktop\pydm\newenv\Scripts>pip list
Package      Version
-----
pip          18.1
setuptools  40.6.2
You are using pip version 18.1, however version 21.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(newenv) C:\Users\          \Desktop\pydm\newenv\Scripts>python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import PIL
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'PIL'
>>>
```

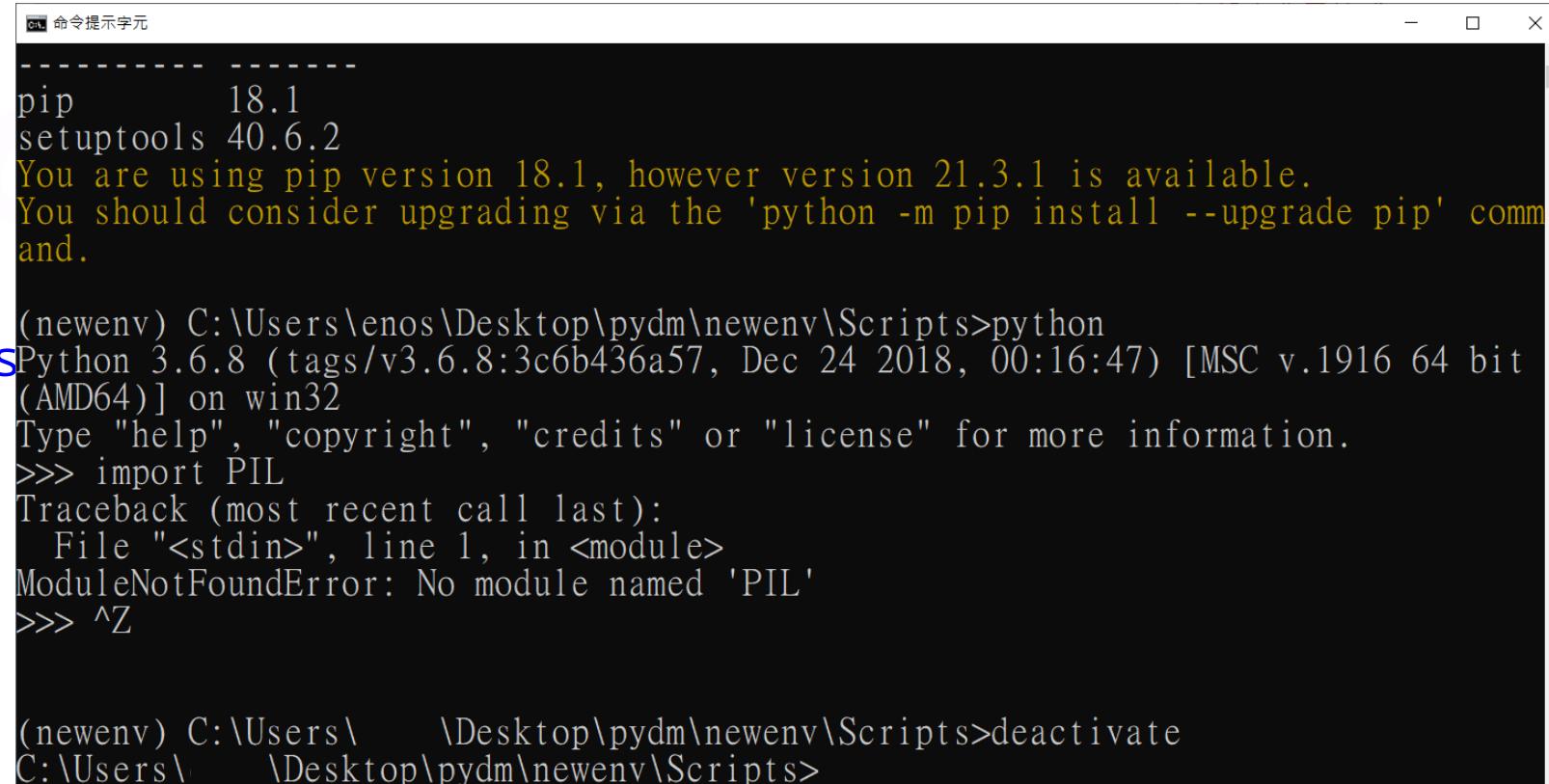
Python Virtual Environment

Python venv

關閉虛擬環境

cd *YOUR_ENV/Scripts*

deactivate



The screenshot shows a Windows Command Prompt window titled "命令提示字元". The command history and output are as follows:

```
pip      18.1
setuptools 40.6.2
You are using pip version 18.1, however version 21.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(newenv) C:\Users\enos\Desktop\pydm\newenv\Scripts>python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import PIL
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'PIL'
>>> ^Z

(newenv) C:\Users\enos\Desktop\pydm\newenv\Scripts>deactivate
C:\Users\enos\Desktop\pydm\newenv\Scripts>
```

Python Virtual Environment

Python venv
(Linux-based)

建立虛擬環境

```
python3 -m venv YOUR_ENV
```

```
eng@test:~$ python3 -m venv newenv  
eng@test:~$ █
```

Python Virtual Environment

Python venv (Linux-based)

啟動虛擬環境

`cd YOUR_ENV/bin`

`source activate`

```
eng@test:~$ python3 -m venv newenv
eng@test:~$ cd newenv/bin
eng@test:~/newenv/bin$ source activate
(newenv) eng@test:~/newenv/bin$ █
```

Python Virtual Environment

Python venv (Linux-based)

關閉虛擬環境

deactivate

```
eng@test:~$ python3 -m venv newenv
eng@test:~$ cd newenv/bin
eng@test:~/newenv/bin$ source activate
(newenv) eng@test:~/newenv/bin$ deactivate
eng@test:~/newenv/bin$ █
```

Python IDE

1. Visual Studio Code
2. PyCharm
3. Spyder
4. Sublime Text
5. Thonny

Python IDE

1. Visual Studio Code

特色

- 擴充套件

來源

<https://code.visualstudio.com/>

設定

IDE 右下角選擇直譯器

The screenshot shows the Visual Studio Code interface. On the left is a dark sidebar with icons for search, file, folder, and settings. The main area has a tab for 'idetest.py - Visual Studio Code'. The code editor contains the following Python script:

```
1  from random import sample
2
3  nums = sample(range(10), 5)
4  found = False
5  for i, n in enumerate(nums):
6      if n < 5:
7          found = True
8          print(f'found {n} at {i} < 5')
9          break
10 if not found:
11     print('not found')
```

Below the code editor is a terminal window with the title 'TERMINAL'. It shows the command line output:

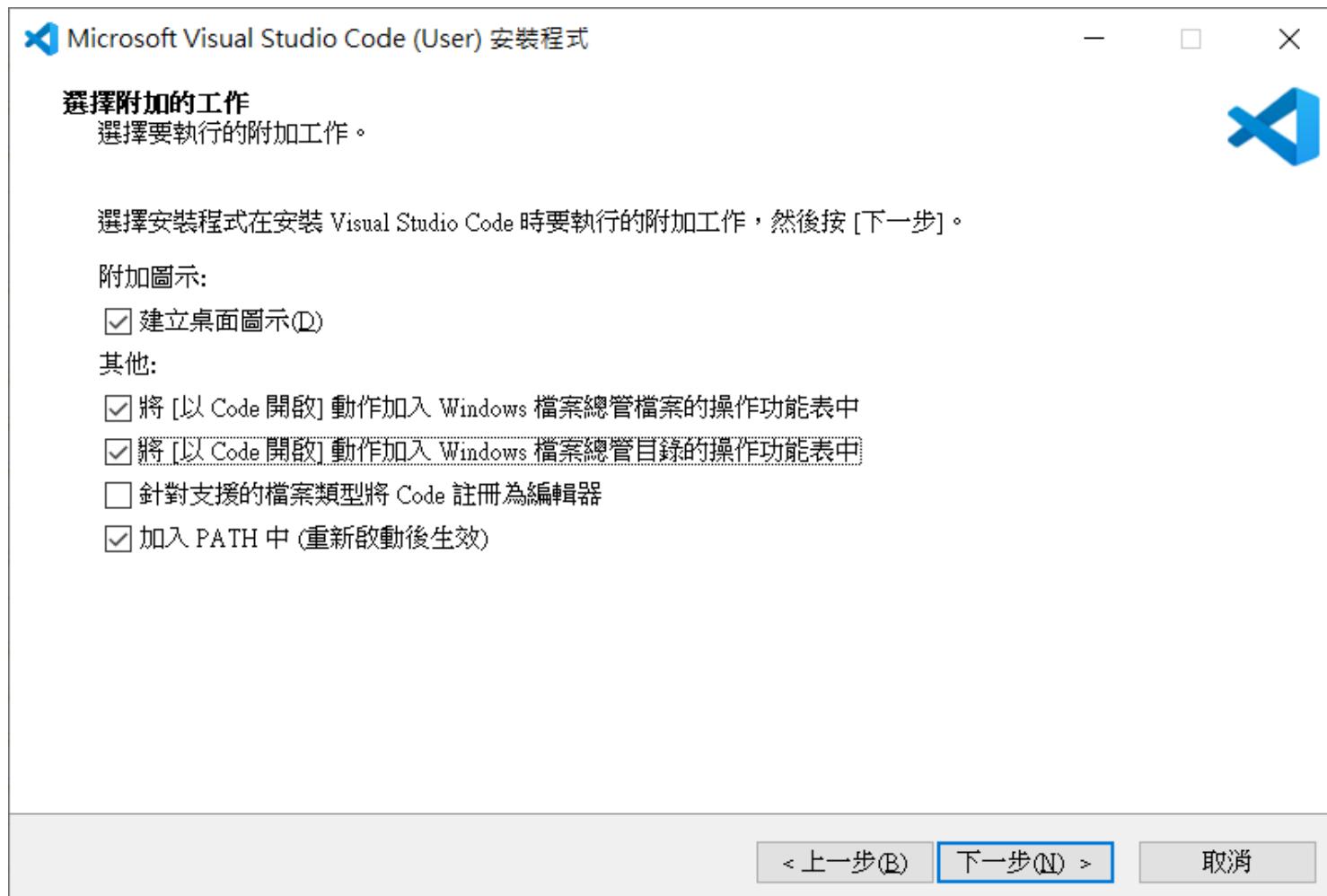
```
PS C:\Users\...\Desktop\pydm> c:; cd 'c:\Users\...\Desktop\pydm'; & 'C:\Users\...\AppData\Local\Programs\Python\Python36\python.exe' 'c:\Users\...\vscode\extensions\ms-python.python\2022.4.1\pythonFiles\lib\python\debugpy\launcher' '59922' '--' 'c:\Users\...\Desktop\pydm\idetest.py'
j 3 at 0 < 5
:Users\...\Desktop\pydm>
```

The status bar at the bottom indicates: Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python 3.6.8 64-bit.

Python IDE

1. Visual Studio Code

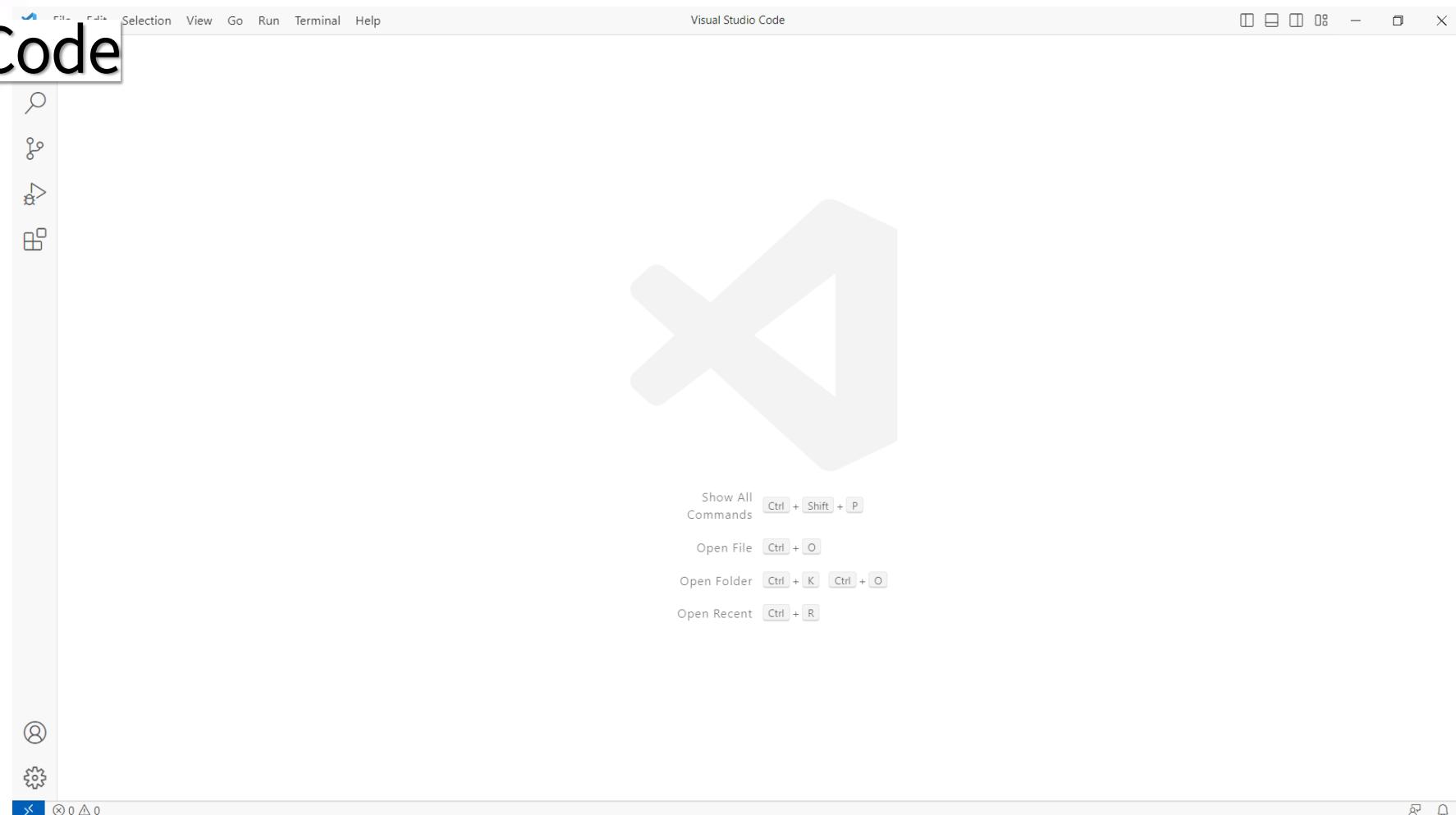
a. 安裝



Python IDE

1. Visual Studio Code

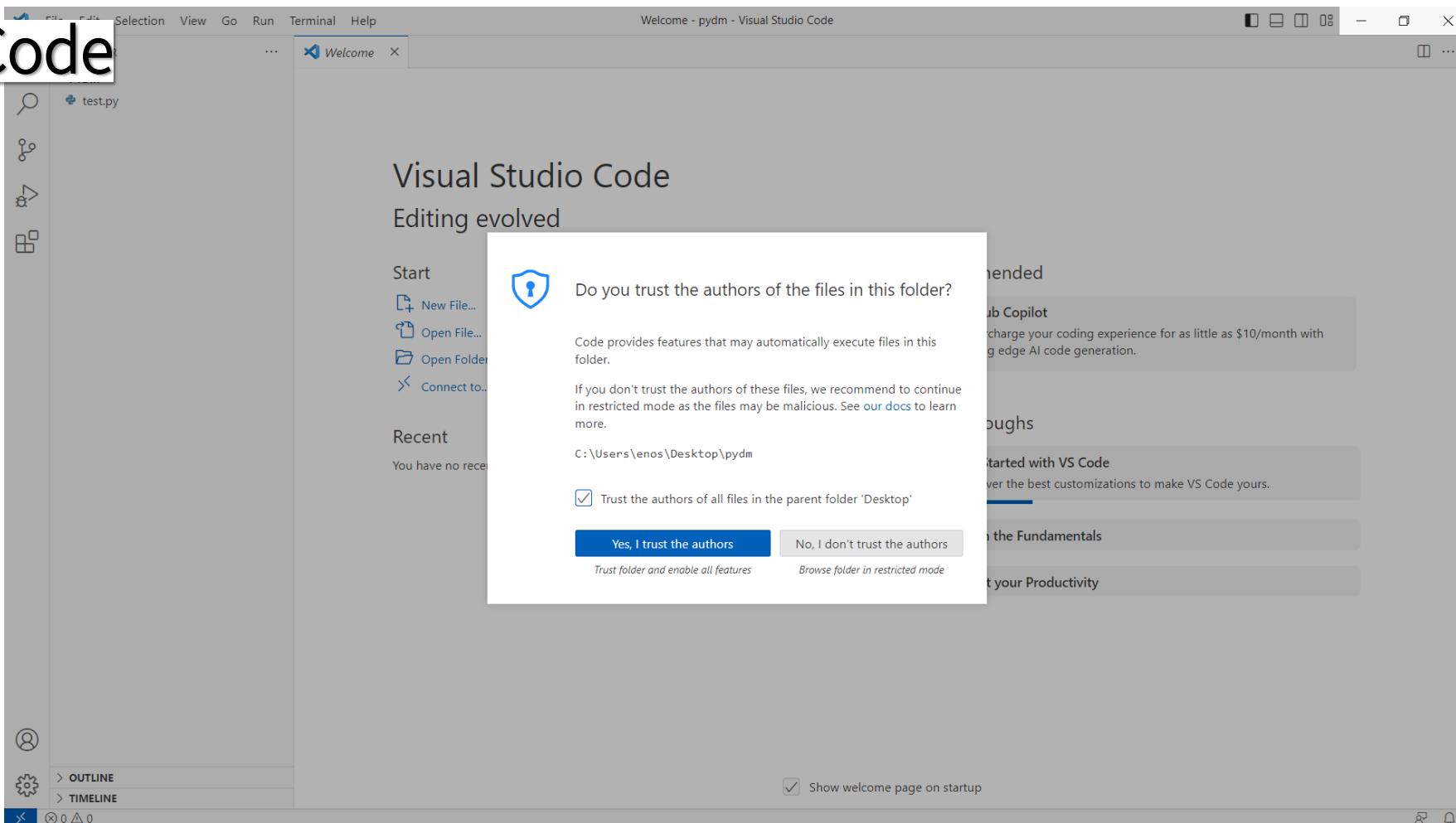
a. 安裝



Python IDE

1. Visual Studio Code

b. 開啟 Folder



Python IDE

1. Visual Studio Code

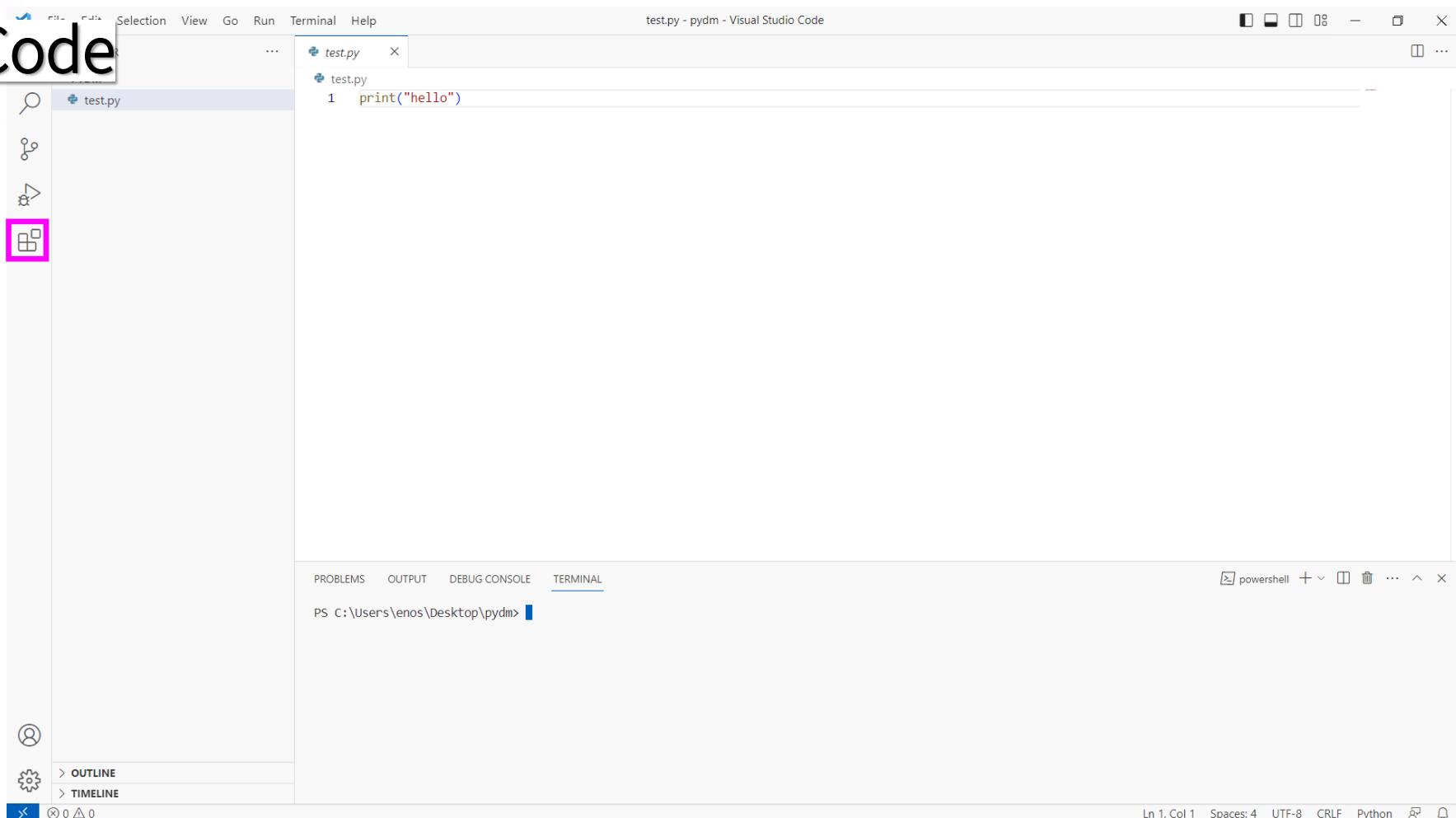
c. 編輯程式

A screenshot of the Visual Studio Code interface. The title bar reads "test.py - pydm - Visual Studio Code". The main editor area shows the Python code: "1 print("hello")". The left sidebar includes icons for search, file operations, and user profile, along with sections for "OUTLINE" and "TIMELINE". The bottom status bar displays "Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python" and a small icon.

Python IDE

1. Visual Studio Code

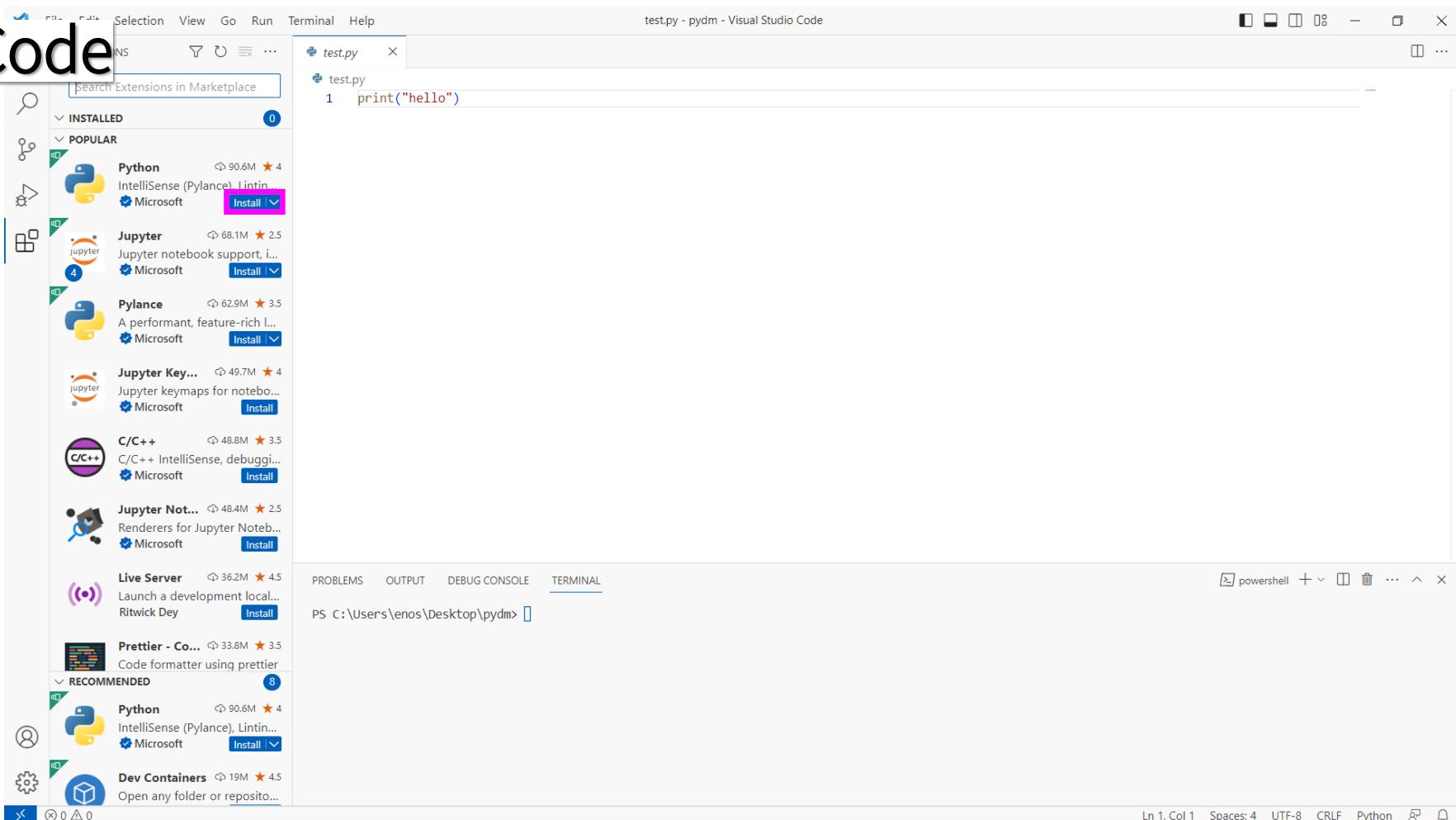
d. 安裝擴充套件



Python IDE

1. Visual Studio Code

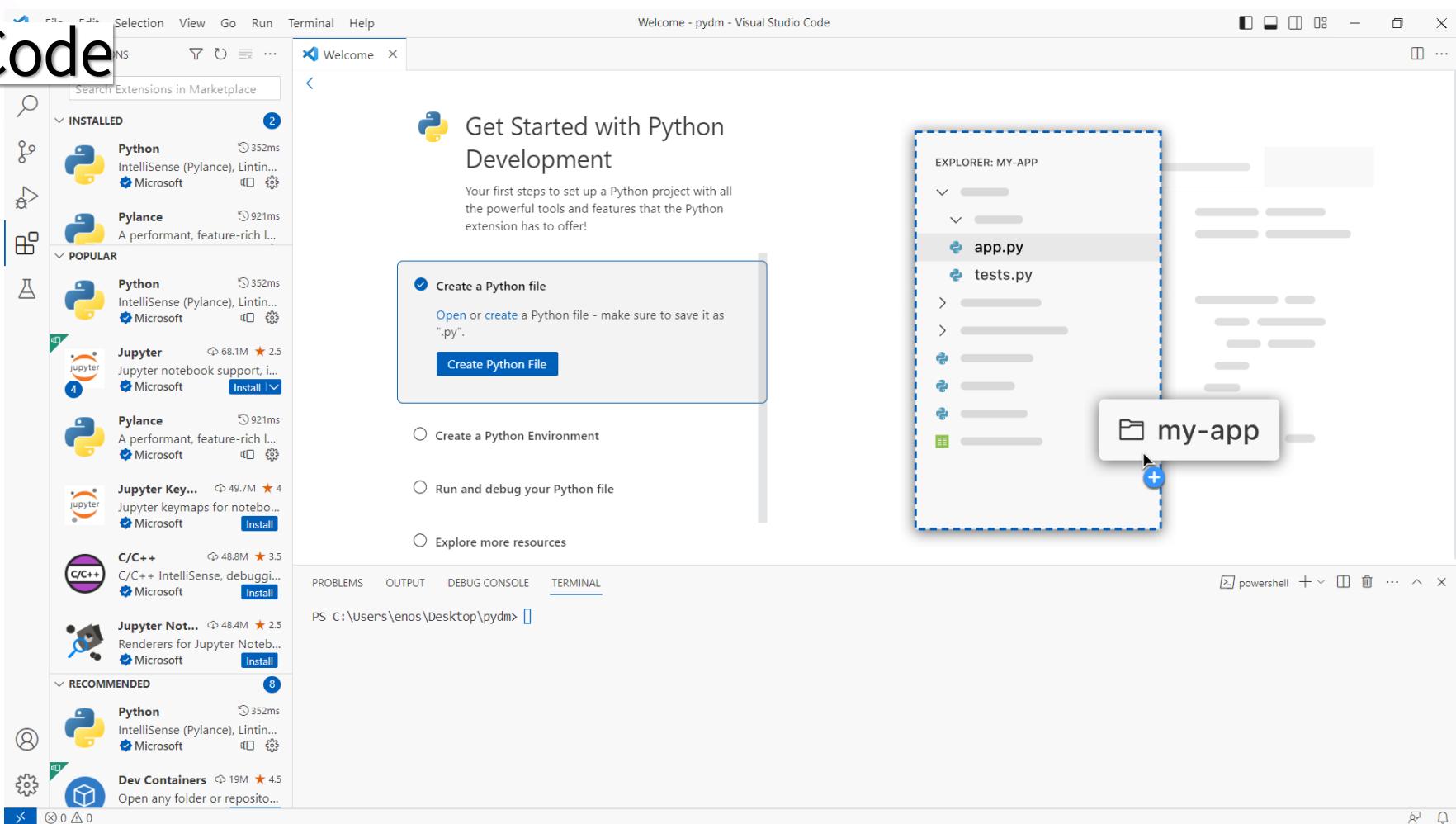
d. 安裝擴充套件



Python IDE

1. Visual Studio Code

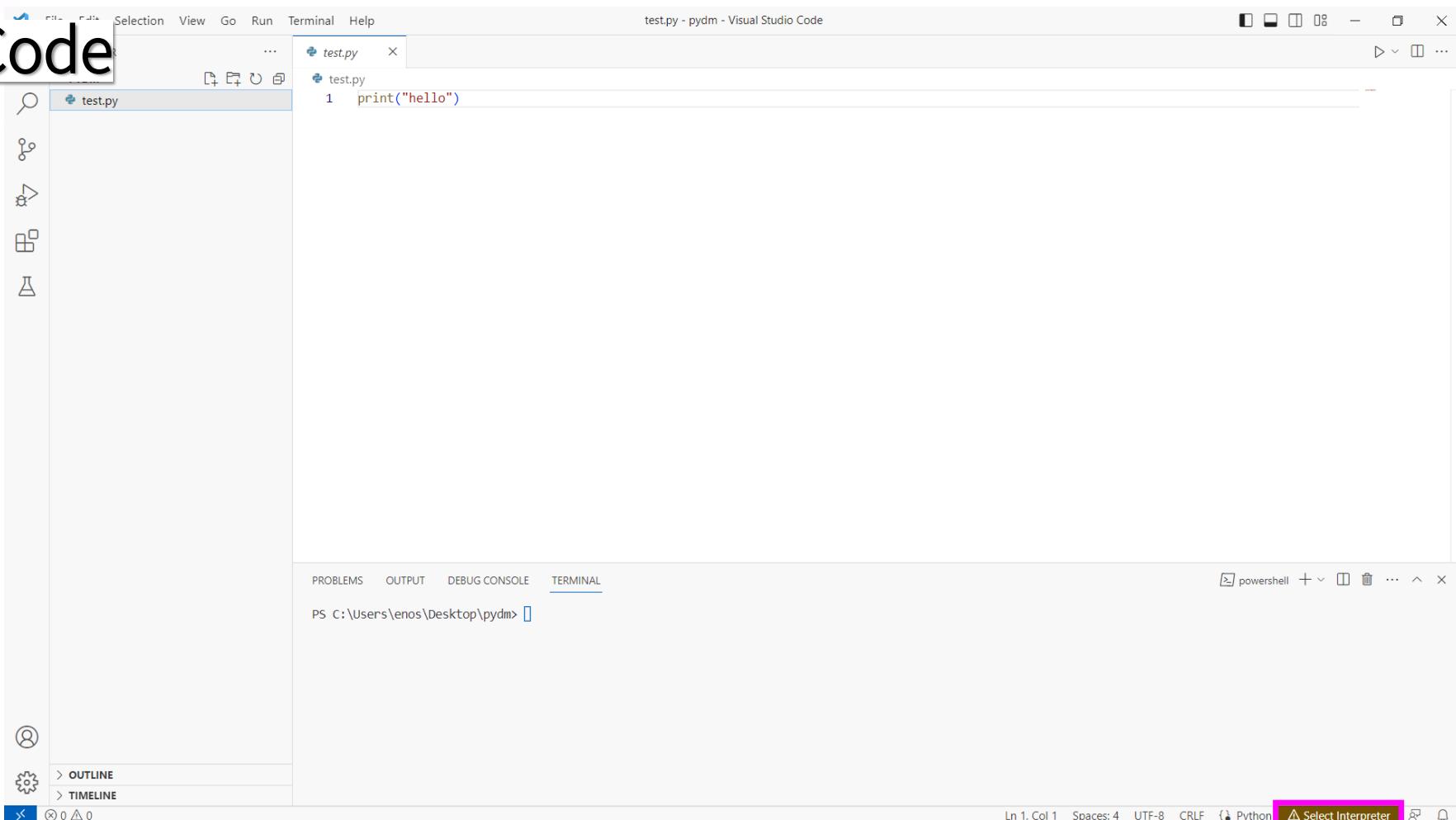
d. 安裝擴充套件



Python IDE

1. Visual Studio Code

e. 指定直譯器



Python IDE

1. Visual Studio Code

f. 執行程式

The screenshot shows the Visual Studio Code interface. On the left is the sidebar with icons for search, file explorer, and other tools. The main area has a tab for 'test.py' containing the Python code:

```
1 print("hello")
```

Below the code editor is a terminal window with the command PS C:\Users\enos\Desktop\pydm> followed by a blue terminal icon. The status bar at the bottom right shows the path C:\Users\enos\Desktop\pydm, and the status bar also includes information about the current file: Ln 1, Col 1, Spaces: 4, UTF-8, CRLF, Python 3.8.10 ('pydm': conda).

Python IDE

1. Visual Studio Code

f. 執行程式

The screenshot shows the Visual Studio Code interface. On the left is the sidebar with icons for search, file explorer, and other tools. The main area has a tab for 'test.py' containing the code: 'print("hello")'. Below the editor is a 'TERMINAL' tab showing command-line output:

```
PS C:\Users\enos\Desktop\pydm> C:/Users/enos/miniconda3/Scripts/activate
PS C:\Users\enos\Desktop\pydm> conda activate pydm
conda : 無法辨識 'conda' 詞彙是否為 Cmdlet、函數、指令檔或可執行程式的名稱。請檢查名稱拼字是否正確，如果包含路徑的話，請確認路徑是否正確，然後再試一次。
位於 線路:1 字元:1
+ conda activate pydm
+ ~~~~
+ CategoryInfo          : ObjectNotFound: (conda:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\enos\Desktop\pydm> & C:/Users/enos/miniconda3/envs/pydm/python.exe c:/Users/enos/Desktop/pydm/test.py
hello
PS C:\Users\enos\Desktop\pydm>
```

The status bar at the bottom indicates the code is in Python 3.8.10 ('pydm': conda) environment.

Python IDE

1. Visual Studio Code

f. 執行程式

The screenshot shows the Visual Studio Code interface. On the left is the sidebar with icons for search, file, folder, and settings. The main area has a tab bar with 'test.py' selected. The code editor contains the following Python script:

```
print("hello")
```

Below the code editor is a terminal window showing the following command and its output:

```
PS C:\Users\enos\Desktop\pydm> C:/Users/enos/miniconda3/Scripts/activate
PS C:\Users\enos\Desktop\pydm> conda activate pydm
conda : 無法辨識 'conda' 詞彙是否為 Cmdlet、函數、指令檔或可執行程式的名稱。請檢查名稱拼字是否正確，如果包含路徑的話，請確認路徑是否正確，然後再試一次。
位於 總路:1 字元:1
+ conda activate pydm
+ ~~~~
+ CategoryInfo          : ObjectNotFound: (conda:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

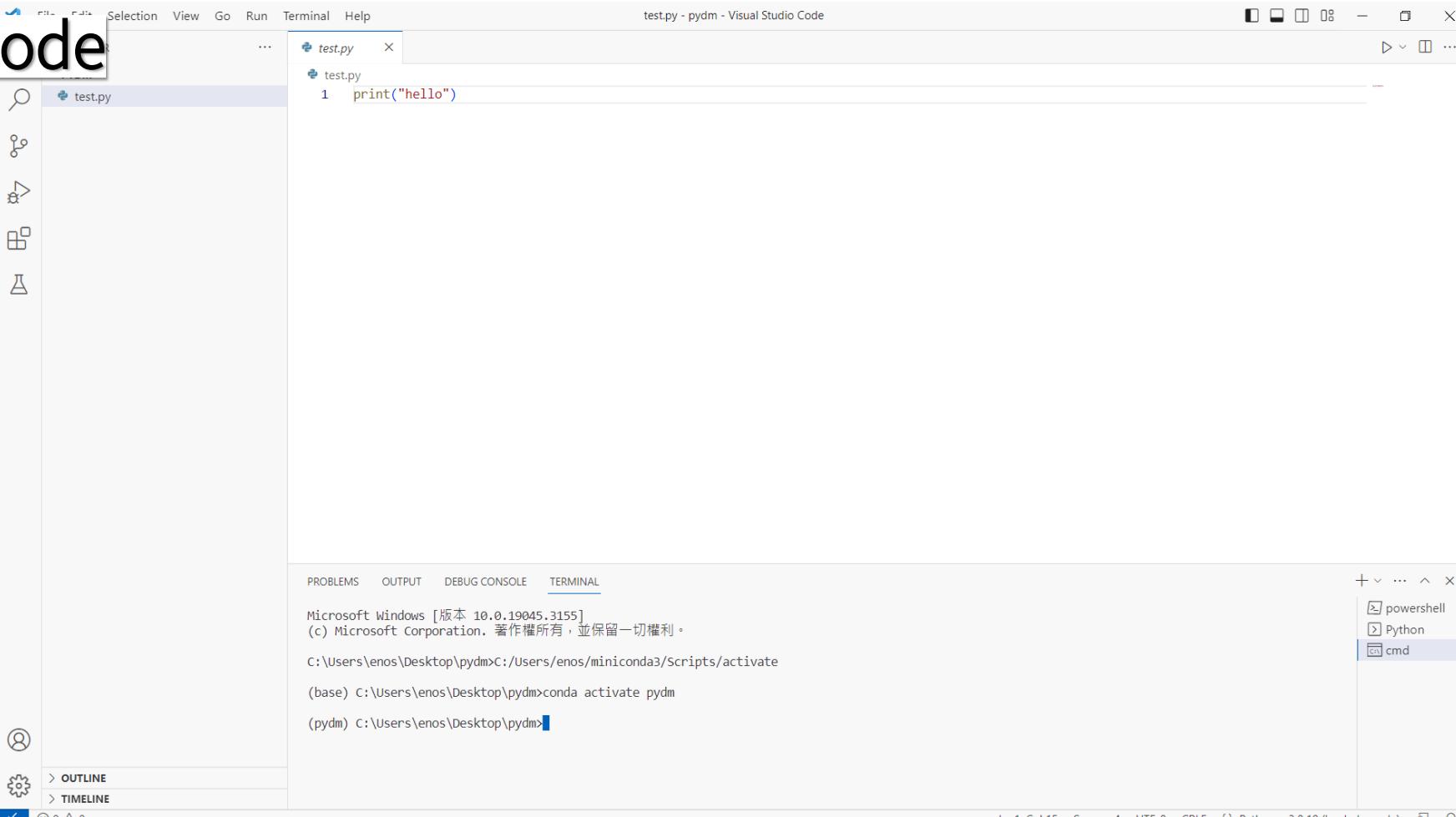
PS C:\Users\enos\Desktop\pydm> & C:/Users/enos/miniconda3/envs/pydm/python.exe c:/Users/enos/Desktop/pydm/test.py
hello
PS C:\Users\enos\Desktop\pydm>
```

A context menu is open over the terminal window, with the 'Command Prompt' option highlighted.

Python IDE

1. Visual Studio Code

f. 執行程式



A screenshot of the Visual Studio Code interface. The left sidebar shows icons for search, file explorer, and other tools. The main editor window displays a Python file named `test.py` with the single line of code `print("hello")`. Below the editor is a terminal window showing the output of running the script:

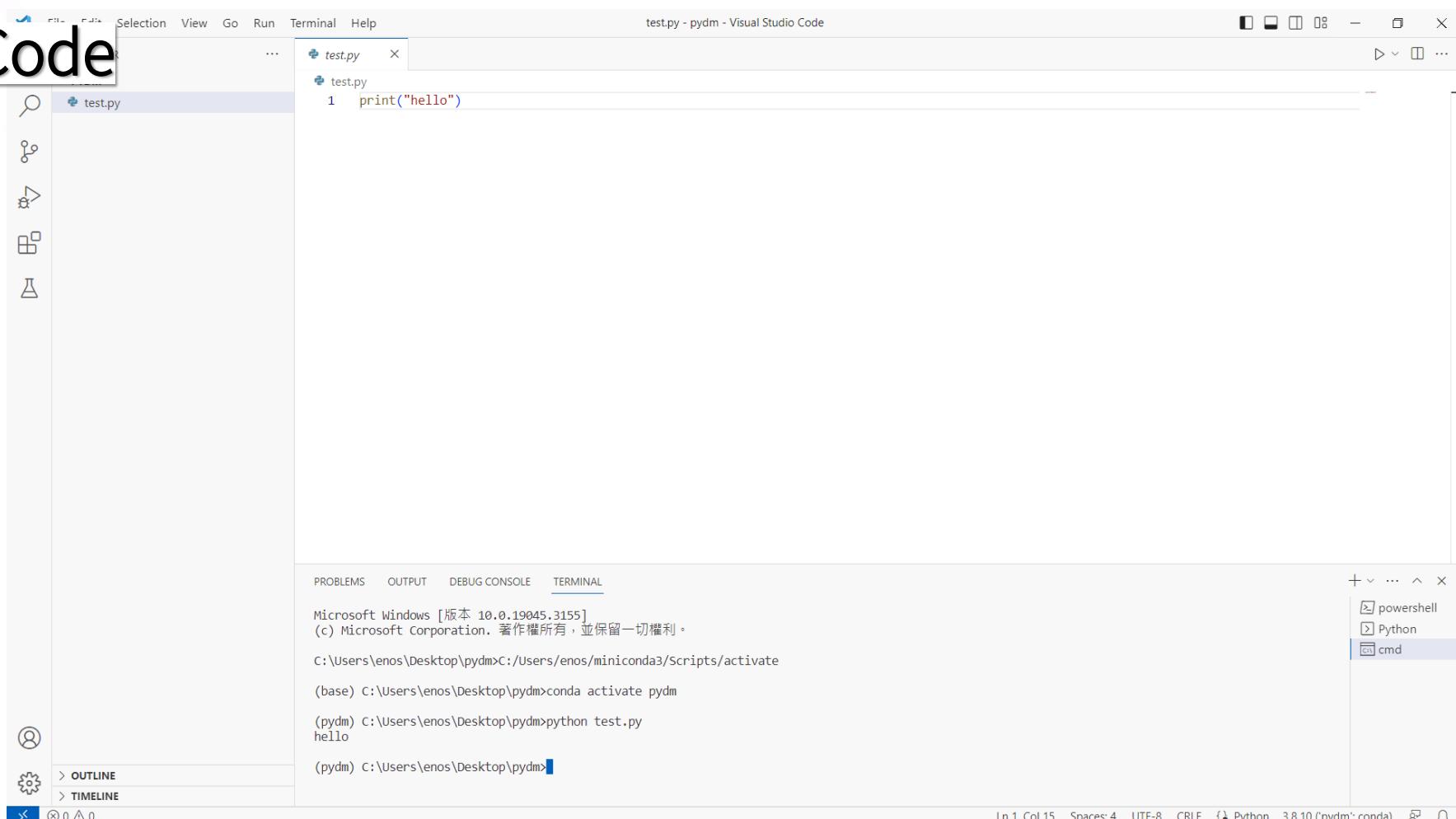
```
Microsoft Windows [版本 10.0.19045.3155]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\enos\Desktop\pydm>C:/Users/enos/miniconda3/Scripts/activate
(base) C:\Users\enos\Desktop\pydm>conda activate pydm
(pydm) C:\Users\enos\Desktop\pydm>
```

The status bar at the bottom indicates the code is in Python 3.8.10 ('pydm': conda) mode.

Python IDE

1. Visual Studio Code

f. 執行程式



A screenshot of the Visual Studio Code interface. On the left is the sidebar with icons for search, file explorer, and other development tools. The main workspace shows a single file named 'test.py' with the code 'print("hello")'. Below the workspace is the 'TERMINAL' tab, which displays the output of running the script: 'Microsoft Windows [版本 10.0.19045.3155] (c) Microsoft Corporation. 著作權所有，並保留一切權利。 C:\Users\enos\Desktop\pydm>C:/Users/enos/miniconda3/Scripts/activate (base) C:\Users\enos\Desktop\pydm>conda activate pydm (pydm) C:\Users\enos\Desktop\pydm>python test.py hello (pydm) C:\Users\enos\Desktop\pydm>'.

Python IDE

1. Visual Studio Code

g. (Terminal 異常排除)

① 無法執行 `conda`

編輯環境變數，於 PATH 新增 `C:\Users\your_account\miniconda3\Scripts`

② 無法自動啟動指定環境

於 Terminal 執行 `conda init`

Python IDE

1. Visual Studio Code

h. (移除)

預設安裝路徑 C:\Users\your_account\AppData\Local\Programs\Microsoft VS Code

- ① 設定 > 應用程式 > 應用程式與功能 > 解除安裝 Microsoft Visual Studio Code
- ② 手動移除資料夾

C:\Users\your_account\.vscode

C:\Users\your_account\AppData\Roaming\Code

Python IDE

2. PyCharm

特色

- 各專案專屬虛擬環境
- 終端機良好配對 virtualenv

來源

<https://www.jetbrains.com/pycharm/>

設定

建立專案目錄 >
IDE 右下角建立虛擬環境

The screenshot shows the PyCharm IDE interface. The code editor displays a Python script named 'idetest.py' with the following content:

```
from random import sample

nums = sample(range(10), 5)
found = False
for i, n in enumerate(nums):
    if n < 5:
        found = True
        print(f'found {n} at {i} < 5')
        break
```

The terminal window at the bottom shows the output of running the script in a virtual environment:

```
(venv) PS C:\Users\...\Desktop\pydm> python .\idetest.py
found 0 at 2 < 5
(venv) PS C:\Users\...\Desktop\pydm>
```

The status bar at the bottom right indicates: 13:1 CRLF UTF-8 4 spaces Python 3.6 (pydm)

Python IDE

3. Spyder

特色

- 顯示變數
- pip 安裝

來源

[pip install spyder](#)

<https://www.spyder-ide.org/>

The screenshot shows the Spyder Python IDE interface. The code editor window displays a script named 'idetest.py' with the following content:

```
1 from random import sample
2
3 nums = sample(range(10), 5)
4 found = False
5 for i, n in enumerate(nums):
6     if n < 5:
7         found = True
8         print(f'found {n} at {i} < 5')
9         break
10
11 if not found:
12     print('not found')
```

To the right of the code editor is the Variable Explorer window, which lists the following variables:

Name	Type	Size	Value
found	bool	1	True
i	int	1	0
n	int	1	2
nums	list	5	[2, 8, 3, 6, 4]

At the bottom of the interface is the IPython Console window, which shows the following session:

```
In [1]: runfile('C:/Users/enos/Desktop/pydm/idestest.py', wdir='C:/Users/enos/Desktop/pydm')
found 0 at 0 < 5

In [2]: runfile('C:/Users/enos/Desktop/pydm/idestest.py', wdir='C:/Users/enos/Desktop/pydm')
found 2 at 0 < 5

In [3]:
```

Python IDE

4. Sublime Text

來源

<https://www.sublimetext.com/>

設定

Tools > Build System > New Build System... >

貼上以下內容並儲存為 xxx.sublime-build

```
{  
    "cmd": ["python.exe", "-u", "$file"],  
    "file_regex": "^[ ]*File \"(...*)\"", line ([0-9]*),  
    "selector": "source.python"  
}
```

Tools > Build System > xxx > 編輯並檔案儲存為 .py > CTRL + B

The screenshot shows the Sublime Text interface with two tabs: 'idetest.py' and 'p36.sublime-build'. The 'idetest.py' tab contains the following Python code:

```
1 from random import sample  
2  
3 nums = sample(range(10), 5)  
4 found = False  
5 for i, n in enumerate(nums):  
6     if n < 5:  
7         found = True  
8         print(f'found {n} at {i} < 5')  
9         break  
12 print('not found')  
13
```

The 'p36.sublime-build' tab shows the build configuration:

```
{  
    "cmd": ["python.exe", "-u", "$file"],  
    "file_regex": "^[ ]*File \"(...*)\"", line ([0-9]*),  
    "selector": "source.python"
```

At the bottom of the screen, the status bar displays 'Line 13, Column 1; Build finished' and 'Spaces: 4 Python'.

Python IDE

5. Thonny

特色

- 適用 MicroPython
- 顯示變數
- 輕量

來源

<https://thonny.org>

設定

工具 >
選項... >
直譯器 >

Thonny 應該使用哪一個直譯器或設備來執行你的程式：其他的 Python 3 直譯器或虛擬環境/
Python 可執行檔：指定你的 Python 執行程式

The screenshot shows the Thonny IDE interface. In the top-left, there's a status bar with 'Thonny - C:\Users\...\Desktop\pydm\idetest.py @ 1:1' and a menu bar with Chinese characters. The main window has several panes: a code editor with Python code, a shell pane showing the output of running the script, a variables pane listing local variables, and a warnings pane.

Code in idetest.py:

```
1 from random import sample
2
3 nums = sample(range(10), 5)
4 found = False
5 for i, n in enumerate(nums):
6     if n < 5:
7         found = True
8         print(f'found n at {i} < 5')
9         break
10
11 if not found:
12     print('not found')
```

Shell output:

```
>>> %Run idetest.py
found n at 0 < 5
>>>
```

Variables pane:

名稱	值
found	True
i	0
n	0
nums	[0, 3, 2, 8, 1]
sample	<bound method Random.sample of <random.Random object at 0x0000000000000000>

Warnings pane:

- May be ignored if you are happy with your program.
- idetest.py
- Line 3: range built-in referenced when not iterating

Bottom right corner shows the path: C:\Users\...\AppData\Local\Programs\Python\Python36\python.exe

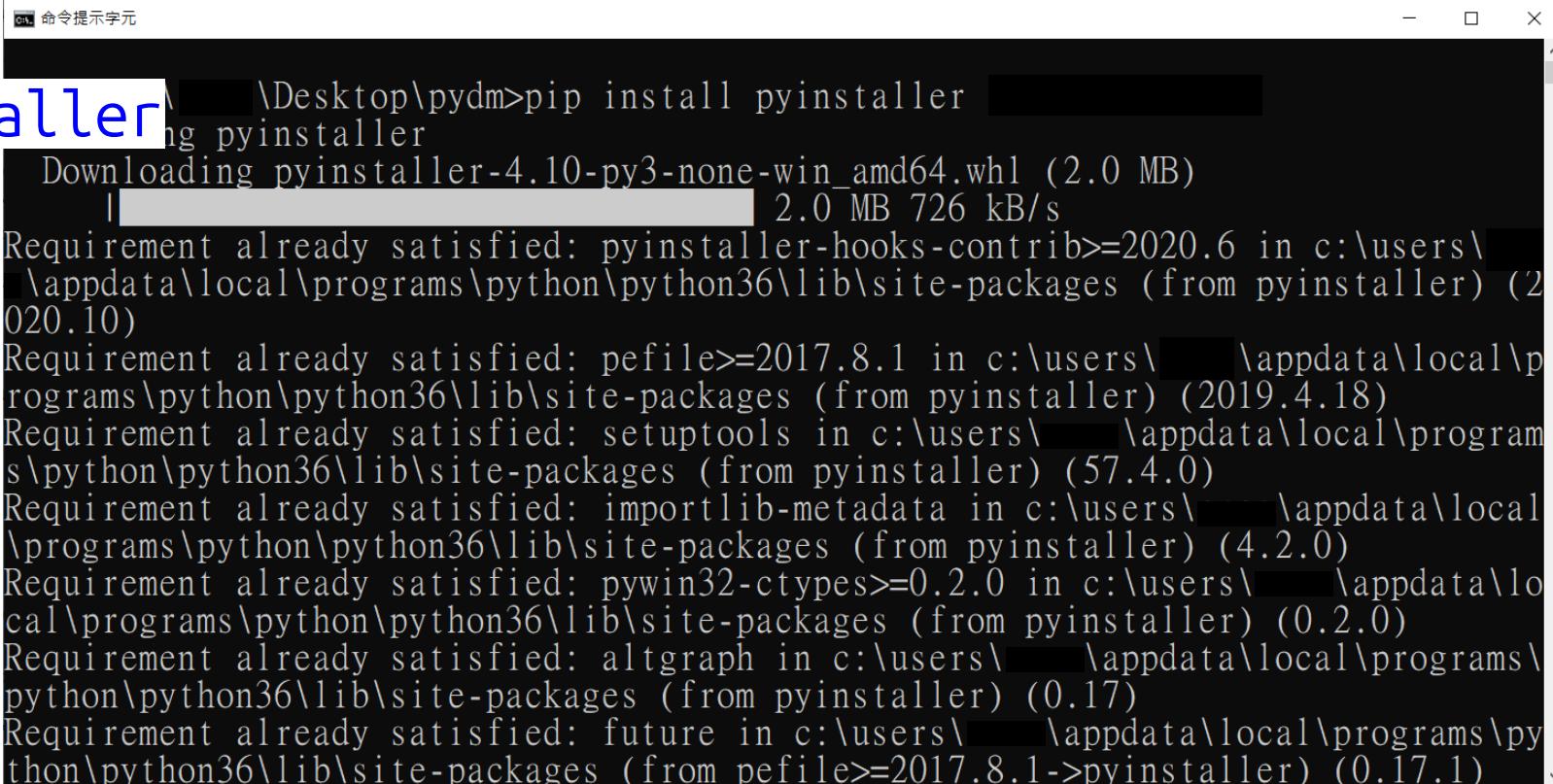
Python Executable

1. 安裝 PyInstaller
2. 轉換 .py 為 executable
3. 執行 executable

Python Executable

1. 安裝 PyInstaller

`pip install pyinstaller`



```
命令提示字元
C:\Desktop\pydm>pip install pyinstaller
Requirement already satisfied: pyinstaller-hooks-contrib>=2020.6 in c:\users\appdata\local\programs\python\python36\lib\site-packages (from pyinstaller) (2020.10)
Requirement already satisfied: pefile>=2017.8.1 in c:\users\appdata\local\programs\python\python36\lib\site-packages (from pyinstaller) (2019.4.18)
Requirement already satisfied: setuptools in c:\users\appdata\local\programs\python\python36\lib\site-packages (from pyinstaller) (57.4.0)
Requirement already satisfied: importlib-metadata in c:\users\appdata\local\programs\python\python36\lib\site-packages (from pyinstaller) (4.2.0)
Requirement already satisfied: pywin32-ctypes>=0.2.0 in c:\users\appdata\local\programs\python\python36\lib\site-packages (from pyinstaller) (0.2.0)
Requirement already satisfied: altgraph in c:\users\appdata\local\programs\python\python36\lib\site-packages (from pyinstaller) (0.17)
Requirement already satisfied: future in c:\users\appdata\local\programs\python\python36\lib\site-packages (from pefile>=2017.8.1->pyinstaller) (0.17.1)
```

Python Executable

2. 轉換 .py 為 executable

`pyinstaller -F your_module.py`

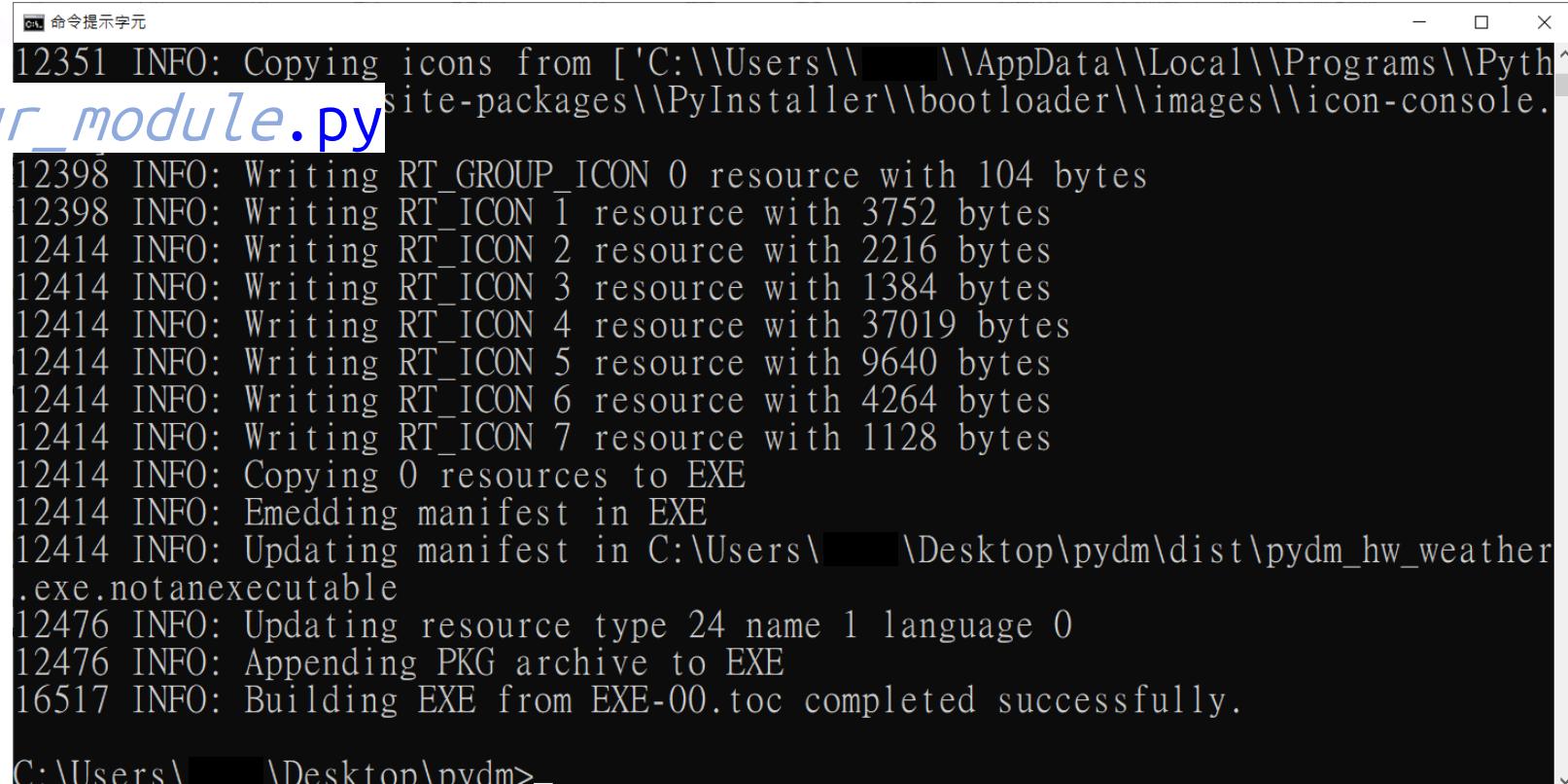
```
命令提示字元
s\python\python36\lib\site-packages (from pyinstaller) (57.4.0)
Requirement already satisfied: importlib-metadata in c:\users\[REDACTED]\appdata\local\programs\python\python36\lib\site-packages (from pyinstaller) (4.2.0)
Requirement already satisfied: pywin32-ctypes>=0.2.0 in c:\users\[REDACTED]\appdata\local\programs\python\python36\lib\site-packages (from pyinstaller) (0.2.0)
Requirement already satisfied: altgraph in c:\users\[REDACTED]\appdata\local\programs\python\python36\lib\site-packages (from pyinstaller) (0.17)
Requirement already satisfied: future in c:\users\[REDACTED]\appdata\local\programs\python\python36\lib\site-packages (from pefile>=2017.8.1->pyinstaller) (0.17.1)
Requirement already satisfied: zipp>=0.5 in c:\users\[REDACTED]\appdata\local\programs\python\python36\lib\site-packages (from importlib-metadata->pyinstaller) (3.4.1)
Requirement already satisfied: typing-extensions>=3.6.4 in c:\users\[REDACTED]\appdata\local\programs\python\python36\lib\site-packages (from importlib-metadata->pyinstaller) (3.10.0.0)
Installing collected packages: pyinstaller
Successfully installed pyinstaller-4.10

C:\Users\[REDACTED]\Desktop\pydm>pyinstaller -F pydm_hw_weather.py
```

Python Executable

2. 轉換 .py 為 executable

```
pyinstaller -F your_module.py
```



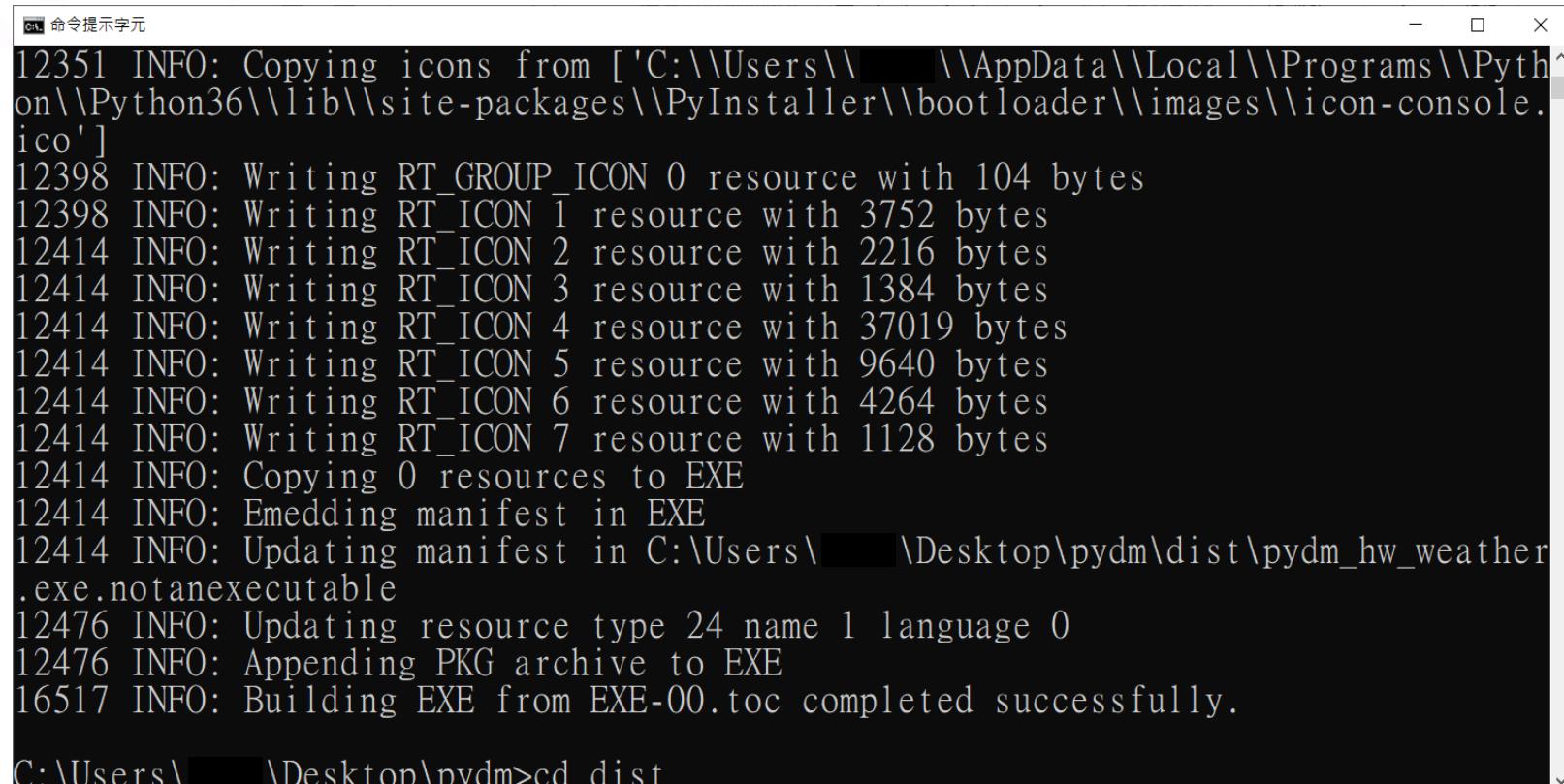
```
命令提示字元
12351 INFO: Copying icons from [ 'C:\\\\Users\\\\\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\3.8\\\\site-packages\\\\PyInstaller\\\\bootloader\\\\images\\\\icon-console.ico' ]
12398 INFO: Writing RT_GROUP_ICON 0 resource with 104 bytes
12398 INFO: Writing RT_ICON 1 resource with 3752 bytes
12414 INFO: Writing RT_ICON 2 resource with 2216 bytes
12414 INFO: Writing RT_ICON 3 resource with 1384 bytes
12414 INFO: Writing RT_ICON 4 resource with 37019 bytes
12414 INFO: Writing RT_ICON 5 resource with 9640 bytes
12414 INFO: Writing RT_ICON 6 resource with 4264 bytes
12414 INFO: Writing RT_ICON 7 resource with 1128 bytes
12414 INFO: Copying 0 resources to EXE
12414 INFO: Embedding manifest in EXE
12414 INFO: Updating manifest in C:\\Users\\\\\\\\Desktop\\\\pydm\\\\dist\\\\pydm_hw_weather.exe.notanexecutable
12476 INFO: Updating resource type 24 name 1 language 0
12476 INFO: Appending PKG archive to EXE
16517 INFO: Building EXE from EXE-00.toc completed successfully.

C:\\Users\\\\\\\\Desktop\\\\pydm>
```

Python Executable

3. 執行 executable

cd dist



The screenshot shows a Windows Command Prompt window titled "命令提示字元". The command "cd dist" has been entered at the prompt. The window displays a log of build steps:

```
12351 INFO: Copying icons from [ 'C:\\\\Users\\\\\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python36\\\\lib\\\\site-packages\\\\PyInstaller\\\\bootloader\\\\images\\\\icon-console.ico' ]
12398 INFO: Writing RT_GROUP_ICON 0 resource with 104 bytes
12398 INFO: Writing RT_ICON 1 resource with 3752 bytes
12414 INFO: Writing RT_ICON 2 resource with 2216 bytes
12414 INFO: Writing RT_ICON 3 resource with 1384 bytes
12414 INFO: Writing RT_ICON 4 resource with 37019 bytes
12414 INFO: Writing RT_ICON 5 resource with 9640 bytes
12414 INFO: Writing RT_ICON 6 resource with 4264 bytes
12414 INFO: Writing RT_ICON 7 resource with 1128 bytes
12414 INFO: Copying 0 resources to EXE
12414 INFO: Embedding manifest in EXE
12414 INFO: Updating manifest in C:\\Users\\\\Desktop\\pydm\\dist\\pydm_hw_weather.exe.notanexecutable
12476 INFO: Updating resource type 24 name 1 language 0
12476 INFO: Appending PKG archive to EXE
16517 INFO: Building EXE from EXE-00.toc completed successfully.
```

C:\\Users\\\\Desktop\\pydm>cd dist

Python Executable

3. 執行 executable

cd dist

執行 *your_module.exe*

```
命令提示字元
12414 INFO: Writing RT_ICON 3 resource with 1384 bytes
12414 INFO: Writing RT_ICON 4 resource with 37019 bytes
12414 INFO: Writing RT_ICON 5 resource with 9640 bytes
12414 INFO: Writing RT_ICON 6 resource with 4264 bytes
12414 INFO: Writing RT_ICON 7 resource with 1128 bytes
12414 INFO: Copying 0 resources to EXE
12414 INFO: Embedding manifest in EXE
12414 INFO: Updating manifest in C:\Users\    \Desktop\pydm\dist\pydm_hw_weather
.exe.notanexecutable
12476 INFO: Updating resource type 24 name 1 language 0
12476 INFO: Appending PKG archive to EXE
16517 INFO: Building EXE from EXE-00.toc completed successfully.

C:\Users\    \Desktop\pydm>cd dist

C:\Users\    \Desktop\pydm\dist>pydm_hw_weather.exe -k
臺北
臺北 觀測時間: 2022-04-29 22:50:00 溫度: 20.7°C, 濕度: 84%, 雨量: 0.0mm

C:\Users\    \Desktop\pydm\dist>
```

Python More

Python One-Line Statements, Python Docstring, Python Decorator,
More Python Argparse , Python Arbitrary Arguments, Python Lambda,
Python Threading

Python One-line Statements

One-line if

```
1 # 檢查 x 是否小於 y  
2 x = 60  
3 y = 100  
4 if x < y:  
    print('x is less than y')
```

x is less than y

```
1 # 檢查 x 是否小於 y  
2 x = 60  
3 y = 100  
4 if x < y: print('x is less than y')
```

x is less than y

Python One-line Statements

One-line if-else

```
1 # 計算 x 與 y 的差 (絕對值)
2 x = 60
3 y = 100
4 if x > y:
5     result = x - y
6 else:
7     result = y - x
8 print(result)
```

40

```
1 # 計算 x 與 y 的差 (絕對值)
2 x = 60
3 y = 100
4 result = x - y if x > y else y - x
5 print(result)
```

40

```
1 # 計算 x 與 y 的差 (絕對值)
2 x = 60
3 y = 100
4 result = x > y and x - y or y - x
5 print(result)
```

40

Python One-line Statements

One-line for loop (List Comprehension)

```
1 # 1 ~ 15 int list
2 y = []
3 for i in range(1, 16):
4     y.append(i)
5 print(y)
6
7 # 1 ~ 15 odd int list
8 q = []
9 for i in range(1, 16):
10    if i % 2 == 1:
11        q.append(i)
12 print(q)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
[1, 3, 5, 7, 9, 11, 13, 15]
```

```
1 # 1 ~ 15 int list
2 yy = [i for i in range(1, 16)]
3 print(yy)
4
5 # 1 ~ 15 odd int list
6 qq = [i for i in range(1, 16) if i % 2 == 1]
7 print(qq)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
[1, 3, 5, 7, 9, 11, 13, 15]
```

The Alternatives of if-else

original

```
if j > 1:  
    w = 2  
else:  
    w = 1
```

by one-line if-else

```
w = 2 if j > 1 else 1
```

by logic

```
w = j > 1 and 2 or 1
```

by trick

```
w = (j > 1) + 1
```

Python Docstring

Docstring of Function

```
1 def fn():
2     ...
3     This function will always return 100
4     ...
5
6     # function one
7     print('one')
8
9     # function two
10    print('two')
11
12    return 100
```

```
1 help(fn)

Help on function fn in module __main__:

fn()
    This function will always return 100
```

Python Docstring

Docstring of Class

```
1 class Test:  
2     '''This is Test class.'''  
3  
4     def say():  
5         ...  
6         This method always print good job.  
7         ...  
8  
9         print('goode job')
```

```
1 help(Test)  
  
Help on class Test in module __main__:  
  
class Test(builtins.object)  
|   This is Test class.  
|  
|   Methods defined here:  
  
|       say()  
|           This method always print good job.  
  
-----  
|  
|   Data descriptors defined here:  
  
|       __dict__  
|           dictionary for instance variables (if defined)  
  
|       __weakref__  
|           list of weak references to the object (if defined)
```

Python Docstring

Docstring of Module

```
1 import yourturn_5_2d  
2  
3 help(yourturn_5_2d)
```

Help on module yourturn_5_2d:

NAME

yourturn_5_2d

CLASSES

 builtins.object
 WeaG

```
class WeaG(builtins.object)  
    WeaG(key)
```

 Weather Grabber, to get recent weather information via open data provided by CWA.

 Methods defined here:

Python Docstring

Docstring of Module

```
1 help(yourturn_5_2d.WeaG)
```

Help on class WeaG in module yourturn_5_2d:

```
class WeaG(builtins.object)
    WeaG(key)
```

Weather Grabber, to get recent weather information via open data provided by CWA.

Methods defined here:

```
__init__(self, key)
```

Initialize the grabber with key.

key - 授權碼 CWA open data

```
grab(self, location)
```

Grab the temperature, humidity, and rainfall of specific location.

location - weather site name, ex: 臺北

Python Docstring

Docstring of Module

```
1 help(yourturn_5_2d.Weather.grab)
```

Help on function grab in module yourturn_5_2d:

grab(self, location)

 Grab the temperature, humidity, and rainfall of specific location.

location - weather site name, ex: 臺北

return - observation time, temperature, humidity, and rainfall of location

 None if location is invalid or other issues

Python Decorator

Decorator 修飾器

```
1 def func1():
2     for i in range(100000):
3         i *= i
```

```
1 def func1():
2     from time import time
3     start = time()
4     for i in range(100000):
5         i *= i
6     print(f'{time()-start:.3f} secs')
```

```
1 func1()
```

0.007 secs

```
1 def func2():
2     from time import time
3     start = time()
4     for i in range(10000):
5         i *= i**i
6     print(f'{time()-start:.3f} secs')
```

```
1 func2()
```

4.384 secs

Python Decorator

Decorator 修飾器

```
1 def speed(func):
2     def wrapper():
3         from time import time
4         start = time()
5         func()
6         print(f'{time()-start:.3f} secs')
7     return wrapper
```

```
1 def func3():
2     for x in range(1000000):
3         x *= x
```

```
1 speed(func3)()
```

0.046 secs

```
1 @speed
2 def func4():
3     for x in range(1000000):
4         x *= x**3
```

1 func4() ←相當於 speed(func4)()

0.341 secs

Python Decorator

Decorator 修飾器

完美修飾

```
1 from functools import wraps ← 還原 __name__  
2  
3 def speed(func):  
4     @wraps(func) ← 還原 __name__  
5     def wrapper(*args, **kargs): ← 支援參數  
6         from time import time  
7         start = time()  
8         func(*args, **kargs) ← 支援參數  
9         print(f'{time()-start:.3f} secs')  
10        return wrapper
```

```
1 @speed  
2 def func4():  
3     for x in range(1000000):  
4         x *= x**3
```

```
1 func4()
```

0.499 secs

```
1 print(func4.__name__)
```

func4

More Python Argparse

Argparse

更多 add_argument() 用法

```
import argparse
parser = argparse.ArgumentParser()

# nargs='+' support 1 or more arguments be assigned at the same time;
# args.values would be a list to keep the multiple inputs
parser.add_argument('values', nargs='+')

# args.v would be kept as a bool
parser.add_argument('-v', action='store_true', default=False)
```

Python Arbitrary (Keyword) Arguments

Packing Arbitrary (Keyword) Arguments

```
def func(a, b, *args, **kwargs)
```

*args: tuple 形式的參數

**kwargs: dict 形式的參數

```
1 def func(a, b, *args, **kwargs):
2     print(f'a is {a}')
3     print(f'b is {b}')
4     print(f'args is {args}')
5     print(f'kwargs is {kwargs}')
```

```
1 func(10, 20, 30, 40, 50)
```

```
a is 10
b is 20
args is (30, 40, 50)
kwargs is {}
```

```
1 func(10, 20, 30, x=40, y=50)
```

```
a is 10
b is 20
args is (30,)
kwargs is {'x': 40, 'y': 50}
```

Python Arbitrary (Keyword) Arguments

Packing Arbitrary Arguments

```
1 # 學生資料
2 d = {'David': {'h': 180, 'w': 85, 'g': 'm'},
3       'John': {'h': 172, 'w': 72, 'g': 'm'},
4       'Mary': {'h': 158, 'w': 51, 'g': 'f'},
5       'Lora': {'h': 160, 'w': 49, 'g': 'f'},
6       'Bill': {'h': 168, 'w': 70, 'g': 'm'},
7       'Elsa': {'h': 154, 'w': 55, 'g': 'f'},
8       'Golden': {'h': 176, 'w': 69, 'g': 'm'},
9       'Jason': {'h': 182, 'w': 77, 'g': 'm'},
10      'Larry': {'h': 160, 'w': 64, 'g': 'm'},
11      'Michelle': {'h': 155, 'w': 57, 'g': 'f'}}}
```

```
1 def select1(name):
2     if name in d:
3         print(f'{name}: {d[name]}')
```

```
1 select1('Jason')
2 select1('Lora')
3 select1('Elsa')
```

```
Jason: {'h': 182, 'w': 77, 'g': 'm'}
Lora: {'h': 155, 'w': 49, 'g': 'f'}
Elsa: {'h': 154, 'w': 55, 'g': 'f'}
```

Python Arbitrary (Keyword) Arguments

Packing Arbitrary Arguments

```
1 # 學生資料
2 d = {'David': {'h': 180, 'w': 85, 'g': 'm'},
3       'John': {'h': 172, 'w': 72, 'g': 'm'},
4       'Mary': {'h': 158, 'w': 51, 'g': 'f'},
5       'Lora': {'h': 160, 'w': 49, 'g': 'f'},
6       'Bill': {'h': 168, 'w': 70, 'g': 'm'},
7       'Elsa': {'h': 154, 'w': 55, 'g': 'f'},
8       'Golden': {'h': 176, 'w': 69, 'g': 'm'},
9       'Jason': {'h': 182, 'w': 77, 'g': 'm'},
10      'Larry': {'h': 160, 'w': 64, 'g': 'm'},
11      'Michelle': {'h': 155, 'w': 57, 'g': 'f'}}}
```

```
1 def select2(*names):
2     for name in names:
3         if name in d:
4             print(f'{name}: {d[name]}')
```

```
1 select2('Jason', 'Lora', 'Elsa')
```

```
Jason: {'h': 182, 'w': 77, 'g': 'm'}
Lora: {'h': 155, 'w': 49, 'g': 'f'}
Elsa: {'h': 154, 'w': 55, 'g': 'f'}
```

Python Arbitrary (Keyword) Arguments

Packing Arbitrary Keyword Arguments

```
1 # 學生資料
2 d = {'David': {'h': 180, 'w': 85, 'g': 'm'},
3       'John': {'h': 172, 'w': 72, 'g': 'm'},
4       'Mary': {'h': 158, 'w': 51, 'g': 'f'},
5       'Lora': {'h': 160, 'w': 49, 'g': 'f'},
6       'Bill': {'h': 168, 'w': 70, 'g': 'm'},
7       'Elsa': {'h': 154, 'w': 55, 'g': 'f'},
8       'Golden': {'h': 176, 'w': 69, 'g': 'm'},
9       'Jason': {'h': 182, 'w': 77, 'g': 'm'},
10      'Larry': {'h': 160, 'w': 64, 'g': 'm'},
11      'Michelle': {'h': 155, 'w': 57, 'g': 'f'}}
```

```
1 def update1(user, h=None, w=None, g=None):
2     if user in d:
3         if h != None:
4             d[user]['h'] = h
5         if w != None:
6             d[user]['w'] = w
7         if g != None:
8             d[user]['g'] = g
```

```
1 name = 'David'
2 print(f'{name} before: {d[name]}')
3 update1(user=name, h=175)
4 print(f'{name} after: {d[name]}')
5
6 name = 'Mary'
7 print(f'{name} before: {d[name]}')
8 update1(user='Mary', w=60)
9 print(f'{name} after: {d[name]}')
10
11 name = 'Bill'
12 print(f'{name} before: {d[name]}')
13 update1(user=name, h=173, w=77)
14 print(f'{name} after: {d[name]}')
```

David before: {'h': 180, 'w': 85, 'g': 'm'}
David after: {'h': 175, 'w': 85, 'g': 'm'}
Mary before: {'h': 158, 'w': 51, 'g': 'f'}
Mary after: {'h': 158, 'w': 60, 'g': 'f'}
Bill before: {'h': 168, 'w': 70, 'g': 'm'}
Bill after: {'h': 173, 'w': 77, 'g': 'm'}

Python Arbitrary (Keyword) Arguments

Packing Arbitrary Keyword Arguments

```
1 # 學生資料
2 d = {'David': {'h': 180, 'w': 85, 'g': 'm'},
3       'John': {'h': 172, 'w': 72, 'g': 'm'},
4       'Mary': {'h': 158, 'w': 51, 'g': 'f'},
5       'Lora': {'h': 160, 'w': 49, 'g': 'f'},
6       'Bill': {'h': 168, 'w': 70, 'g': 'm'},
7       'Elsa': {'h': 154, 'w': 55, 'g': 'f'},
8       'Golden': {'h': 176, 'w': 69, 'g': 'm'},
9       'Jason': {'h': 182, 'w': 77, 'g': 'm'},
10      'Larry': {'h': 160, 'w': 64, 'g': 'm'},
11      'Michelle': {'h': 155, 'w': 57, 'g': 'f'}}
```

```
1 def update2(user, **kwargs):
2     if user in d:
3         for a in kwargs:
4             if a in d[user]:
5                 d[user][a] = kwargs[a]
```

```
1 name = 'David'
2 print(f'{name} before: {d[name]}')
3 update2(user=name, h=175)
4 print(f'{name} after: {d[name]}')

5
6 name = 'Mary'
7 print(f'{name} before: {d[name]}')
8 update2(user='Mary', w=60)
9 print(f'{name} after: {d[name]}')

10
11 name = 'Bill'
12 print(f'{name} before: {d[name]}')
13 update2(user=name, h=173, w=77)
14 print(f'{name} after: {d[name]}')
```

```
David before: {'h': 180, 'w': 85, 'g': 'm'}
David after: {'h': 175, 'w': 85, 'g': 'm'}
Mary before: {'h': 158, 'w': 51, 'g': 'f'}
Mary after: {'h': 158, 'w': 60, 'g': 'f'}
Bill before: {'h': 168, 'w': 70, 'g': 'm'}
Bill after: {'h': 173, 'w': 77, 'g': 'm'}
```

Python Arbitrary (Keyword) Arguments

Unpacking Arguments

```
1 # 學生資料
2 d = {'David': {'h': 180, 'w': 85, 'g': 'm'},
3       'John': {'h': 172, 'w': 72, 'g': 'm'},
4       'Mary': {'h': 158, 'w': 51, 'g': 'f'},
5       'Lora': {'h': 160, 'w': 49, 'g': 'f'},
6       'Bill': {'h': 168, 'w': 70, 'g': 'm'},
7       'Elsa': {'h': 154, 'w': 55, 'g': 'f'},
8       'Golden': {'h': 176, 'w': 69, 'g': 'm'},
9       'Jason': {'h': 182, 'w': 77, 'g': 'm'},
10      'Larry': {'h': 160, 'w': 64, 'g': 'm'},
11      'Michelle': {'h': 155, 'w': 57, 'g': 'f'}}
```

```
1 def select2(*names):
2     for name in names:
3         if name in d:
4             print(f'{name}: {d[name]}')
```

```
1 select2('Jason', 'Lora', 'Elsa')
```

```
Jason: {'h': 182, 'w': 77, 'g': 'm'}
Lora: {'h': 155, 'w': 49, 'g': 'f'}
Elsa: {'h': 154, 'w': 55, 'g': 'f'}
```

```
1 names = ['Jason', 'Lora', 'Elsa']
2 select2(*names) # * to unpack list/ tuple arguments
```

```
Jason: {'h': 182, 'w': 77, 'g': 'm'}
Lora: {'h': 155, 'w': 49, 'g': 'f'}
Elsa: {'h': 154, 'w': 55, 'g': 'f'}
```

Python Arbitrary (Keyword) Arguments

Unpacking Keyword Arguments

```
1 # 學生資料
2 d = {'David': {'h': 180, 'w': 85, 'g': 'm'},
3       'John': {'h': 172, 'w': 72, 'g': 'm'},
4       'Mary': {'h': 158, 'w': 51, 'g': 'f'},
5       'Lora': {'h': 160, 'w': 49, 'g': 'f'},
6       'Bill': {'h': 168, 'w': 70, 'g': 'm'},
7       'Elsa': {'h': 154, 'w': 55, 'g': 'f'},
8       'Golden': {'h': 176, 'w': 69, 'g': 'm'},
9       'Jason': {'h': 182, 'w': 77, 'g': 'm'},
10      'Larry': {'h': 160, 'w': 64, 'g': 'm'},
11      'Michelle': {'h': 155, 'w': 57, 'g': 'f'}}
```

```
1 def update2(user, **kwargs):
2     if user in d:
3         for a in kwargs:
4             if a in d[user]:
5                 d[user][a] = kwargs[a]
```

```
1 name = 'David'
2 print(f'{name} before: {d[name]}')
3 update2(user=name, h=175)
4 print(f'{name} after: {d[name]}')

5
6 name = 'Mary'
7 print(f'{name} before: {d[name]}')
8 update2(user='Mary', w=60)
9 print(f'{name} after: {d[name]}')

10
11 name = 'Bill'
12 print(f'{name} before: {d[name]}')
13 update2(user=name, h=173, w=77)
14 print(f'{name} after: {d[name]}')
```

```
David before: {'h': 180, 'w': 85, 'g': 'm'}
David after: {'h': 175, 'w': 85, 'g': 'm'}
Mary before: {'h': 158, 'w': 51, 'g': 'f'}
Mary after: {'h': 158, 'w': 60, 'g': 'f'}
Bill before: {'h': 168, 'w': 70, 'g': 'm'}
Bill after: {'h': 173, 'w': 77, 'g': 'm'}
```

Python Arbitrary (Keyword) Arguments

Unpacking Keyword Arguments

```
1 # 學生資料
2 d = {'David': {'h': 180, 'w': 85, 'g': 'm'},
3       'John': {'h': 172, 'w': 72, 'g': 'm'},
4       'Mary': {'h': 158, 'w': 51, 'g': 'f'},
5       'Lora': {'h': 160, 'w': 49, 'g': 'f'},
6       'Bill': {'h': 168, 'w': 70, 'g': 'm'},
7       'Elsa': {'h': 154, 'w': 55, 'g': 'f'},
8       'Golden': {'h': 176, 'w': 69, 'g': 'm'},
9       'Jason': {'h': 182, 'w': 77, 'g': 'm'},
10      'Larry': {'h': 160, 'w': 64, 'g': 'm'},
11      'Michelle': {'h': 155, 'w': 57, 'g': 'f'}}
```

```
1 def update2(user, **kwargs):
2     if user in d:
3         for a in kwargs:
4             if a in d[user]:
5                 d[user][a] = kwargs[a]
```

```
1 name = 'Lora'
2 print(f'{name} before: {d[name]}')
3 x = {'h': 155, 'w': 49}
4 update2(user=name, **x) # ** to unpack dict arguments
5 print(f'{name} after: {d[name]}')
```

```
Lora before: {'h': 160, 'w': 49, 'g': 'f'}
Lora after:  {'h': 155, 'w': 49, 'g': 'f'}
```

Python Lambda

簡化一次性、短函式

```
1 def func(n, m):  
2     return n * m / 10
```

```
1 print(func(41, 42))
```

172.2

```
1 print((lambda n, m : n * m / 10)(41, 42))
```

172.2

Python Lambda

簡化一次性、短函式

```
1 # 學生資料
2 d = {'David': {'h': 180, 'w': 85, 'g': 'm'},
3       'John': {'h': 172, 'w': 72, 'g': 'm'},
4       'Mary': {'h': 158, 'w': 51, 'g': 'f'},
5       'Lora': {'h': 160, 'w': 49, 'g': 'f'},
6       'Bill': {'h': 168, 'w': 70, 'g': 'm'},
7       'Elsa': {'h': 154, 'w': 55, 'g': 'f'},
8       'Golden': {'h': 176, 'w': 69, 'g': 'm'},
9       'Jason': {'h': 182, 'w': 77, 'g': 'm'},
10      'Larry': {'h': 160, 'w': 64, 'g': 'm'},
11      'Michelle': {'h': 155, 'w': 57, 'g': 'f'}}}
```

```
1 a = list(d.items())
2 a
```

```
[('David', {'h': 180, 'w': 85, 'g': 'm'}),
 ('John', {'h': 172, 'w': 72, 'g': 'm'}),
 ('Mary', {'h': 158, 'w': 51, 'g': 'f'}),
 ('Lora', {'h': 160, 'w': 49, 'g': 'f'}),
 ('Bill', {'h': 168, 'w': 70, 'g': 'm'}),
 ('Elsa', {'h': 154, 'w': 55, 'g': 'f'}),
 ('Golden', {'h': 176, 'w': 69, 'g': 'm'}),
 ('Jason', {'h': 182, 'w': 77, 'g': 'm'}),
 ('Larry', {'h': 160, 'w': 64, 'g': 'm'}),
 ('Michelle', {'h': 155, 'w': 57, 'g': 'f'})]
```

```
1 sorted(a)
```

```
[('Bill', {'h': 168, 'w': 70, 'g': 'm'}),
 ('David', {'h': 180, 'w': 85, 'g': 'm'}),
 ('Elsa', {'h': 154, 'w': 55, 'g': 'f'}),
 ('Golden', {'h': 176, 'w': 69, 'g': 'm'}),
 ('Jason', {'h': 182, 'w': 77, 'g': 'm'}),
 ('John', {'h': 172, 'w': 72, 'g': 'm'}),
 ('Larry', {'h': 160, 'w': 64, 'g': 'm'}),
 ('Lora', {'h': 160, 'w': 49, 'g': 'f'}),
 ('Mary', {'h': 158, 'w': 51, 'g': 'f'}),
 ('Michelle', {'h': 155, 'w': 57, 'g': 'f'})]
```

Python Lambda

簡化一次性、短函式

```
1 help(sorted)
```

Help on built-in function sorted in module builtins:

sorted(iterable, /, *, key=None, reverse=False)

Return a new list containing all items from the iterable in ascending order.

A custom key function can be supplied to customize the sort order, and the reverse flag can be set to request the result in descending order.

Python Lambda

簡化一次性、短函式

```
1 def h(x):
2     return x[1]['h']
3
4 def w(x):
5     return x[1]['w']
```

```
1 sorted(a, key=h)
```

```
[('Elsa', {'h': 154, 'w': 55, 'g': 'f'}),
 ('Michelle', {'h': 155, 'w': 57, 'g': 'f'}),
 ('Mary', {'h': 158, 'w': 51, 'g': 'f'}),
 ('Lora', {'h': 160, 'w': 49, 'g': 'f'}),
 ('Larry', {'h': 160, 'w': 64, 'g': 'm'}),
 ('Bill', {'h': 168, 'w': 70, 'g': 'm'}),
 ('John', {'h': 172, 'w': 72, 'g': 'm'}),
 ('Golden', {'h': 176, 'w': 69, 'g': 'm'}),
 ('David', {'h': 180, 'w': 85, 'g': 'm'}),
 ('Jason', {'h': 182, 'w': 77, 'g': 'm'})]
```

```
1 sorted(a, key=lambda x: x[1]['h'])
```

```
[('Elsa', {'h': 154, 'w': 55, 'g': 'f'}),
 ('Michelle', {'h': 155, 'w': 57, 'g': 'f'}),
 ('Mary', {'h': 158, 'w': 51, 'g': 'f'}),
 ('Lora', {'h': 160, 'w': 49, 'g': 'f'}),
 ('Larry', {'h': 160, 'w': 64, 'g': 'm'}),
 ('Bill', {'h': 168, 'w': 70, 'g': 'm'}),
 ('John', {'h': 172, 'w': 72, 'g': 'm'}),
 ('Golden', {'h': 176, 'w': 69, 'g': 'm'}),
 ('David', {'h': 180, 'w': 85, 'g': 'm'}),
 ('Jason', {'h': 182, 'w': 77, 'g': 'm'})]
```

Python Threading

與主程序同時執行的程式

threading

```
t = threading.Thread(target=func) # 建立 thread  
t.daemon = True # daemon 模式時，thread 會隨主程式結束（需於 CLI 驗證）  
t.start() # 啟動 thread  
... # 主程序內容  
t.join() # 主程序等待 thread 結束  
... # 主程序內容
```

Python Threading

與主程序同時執行的程式

threading

```
t = threading.Thread(target=...  
t.daemon = True # daemon  
t.start() # 啟動 thread  
... # 主程序內容  
t.join() # 主程序等待 thread  
... # 主程序內容
```

```
1 import threading  
2 from time import sleep  
3  
4 finished = False  
5  
6 def show(secs):  
7     i = 0  
8     while not finished:  
9         print(i*secs)  
10        sleep(secs)  
11        i += 1  
12  
13 def timer(secs):  
14     global finished  
15  
16     print(f'countdown for {secs} ses.')  
17     finished = False  
18     sleep(secs)  
19     finished = True  
20     print('timeout')  
21  
22 t = threading.Thread(target=timer, args=(5,))  
23 t.daemon = True  
24 t.start()  
25 show(1.3)  
26 t.join()  
27 print('finished')
```

CLI 驗證)

countdown for 5 ses.	0.0
	1.3
	2.6
	3.9000000000000004
	timeout
	finished

Python Threading

與主程序同時執行的程式

threading

```
t = threading.Thread(target=...)  
t.daemon = True # daemon  
t.start() # 啟動 thread  
... # 主程序內容  
t.join() # 主程序等待 thread  
... # 主程序內容
```

```
1 import threading  
2 from time import sleep  
3  
4 finished = False  
5  
6 def show(secs):  
7     i = 0  
8     while not finished:  
9         print(i*secs)  
10        sleep(secs)  
11        i += 1  
12  
13 def inp():  
14     global finished  
15  
16     finished = False  
17     input('any key to stop ...')  
18     finished = True  
19     print('stopped')  
20  
21 t = threading.Thread(target=inp)  
22 t.daemon = True  
23 t.start()  
24 show(1.3)  
25 t.join()  
26 print('finished')
```

結束（需於 CLI 驗證）

Python Regular Expression

用正規表示法判斷字串是否符合特定格式

re

常用語法，判斷 string 是否符合 pattern

`re.match(pattern, string)`

若 string 符合 pattern =>
回應 Match 物件

若 string 不符合 pattern =>
回應 None

```
import re
pattern = r'[\s*\d\s*]{4}'

string = ' 95 43'
print(re.match(pattern, string))

<re.Match object; span=(0, 4), match=' 95 '>
```

```
string = 'a987'
print(re.match(pattern, string))
```

None

Python Regular Expression

用正規表示法判斷字串是否符合特定格式

re

參考連結

<https://docs.python.org/> 搜尋 re

Python Walrus Operator

運算後儲存

`:=`

運算後當滿足條件再
儲存結果至變數

```
1 a = [1, 2, 3]
2
3 if len(a) > 0:
4     print(len(a))
```

3

```
1 a = [1, 2, 3]
2
3 len_a = len(a)
4 if len_a > 0:
5     print(len_a)
```

3

```
1 a = [1, 2, 3]
2
3 if (len_a := len(a)) > 0:
4     print(len_a)
```

3

Python Walrus Operator

運算後儲存

`:=`

運算後當滿足條件再
儲存結果至變數

```
1 with open('test.txt') as f:  
2     a = f.readline()  
3     while a:  
4         print(a, end=' ')  
5         a = f.readline()
```

abc
xyz
kkk
mmm

```
1 with open('test.txt') as f:  
2     while a := f.readline():  
3         print(a, end=' ')
```

abc
xyz
kkk
mmm

Python Walrus Operator

運算後儲存

`:=`

參閱 PEP 572 Assignment Expressions

Python Iterator

自定義迭代器

重置迭代器，
重新使用時被叫用

取得每次迭代的元素

迭代器元素個數

依鍵值取得內容

```
class Iter:

    def __init__(self):
        pass

    def __iter__(self): # reset iterator
        return self # always return self

    def __next__(self): # get item
        raise StopIteration # return item or stop iteration

    def __len__(self): # get length of iterable
        pass # return length of iterable

    def __getitem__(self, key): # get item of specific subscription
        pass # return subscripted item
```

Python Iterators

自定義迭代器

Example

```
class Alphabetiter:

    def __init__(self):
        self.alphabets = 'abcdefghijklmnopqrstuvwxyz'
        self.i = -1

    def __iter__(self):
        self.i = -1
        return self

    def __next__(self):
        self.i += 1
        if self.i < len(self.alphabets): return self.alphabets[self.i]
        raise StopIteration

    def __getitem__(self, key):
        return self.alphabets[key]

    def __len__(self):
        return len(self.alphabets)
```

Python Iterator

自定義迭代器

Example

```
a = Alphabetiter()
for x in a:
    print(x, end=' ')
print()
for i in range(len(a)):
    print(a[i], end=' ')
```

```
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
```

Programming Logic

Programming Principle

1. Correct Logic

- a. Delicate Error Handling
- b. Reasonable Fault Tolerance

2. Better Performance

- a. Faster Speed
- b. Fewer Resources
- c. Shorter Code
- d. Higher Readability

範例問題

找出 a 中最大值

a = [100, 2, 207, 350, 40, 0.3, -2, -18]

350

三角形辨別

以三角形的三個角度判定三角形類別

50 40 30

30 40 50

Right

邏輯運算子

判定輸入的前兩個參數透過哪幾種邏輯運算 (AND/ OR/ XOR) 可以得到第 3 個參數的結果

0 0 0

AND

OR

XOR

遊戲選角

遊戲角色的能力值以兩個整數代表，比較時以兩個值的平方和為強弱依據 (值愈大則角色能力愈強)，試篩選出第二強的角色

5 8

1 3

6 6

9 2

4 5

9 2

成績指標

找出最低分 PASS 與最高分 FAIL 的成績

71 68 92

68 71 92

best case

68

Missing Number

找出遺失的數字

Input: [3, 0, 1, 4]

Output: 2

Input: [4, 0, 2, 1, 3]

Output: 5

Sukudo

判斷數獨的正確性

5	2	
	6	8
8	5	2
	3	4
7		2
6		5
	7	
4		2
	2	3

True

最大公因數 (Recursive)

計算任意兩數字的最大公因數，例：

24 33

3

84 54

6

The End