

AI機器學習與深度學習應用 – 使用Python與R語言 2022(Python)

大數據分析

- R/Python/Julia/SQL程式設計與應用
(R/Python/Julia/SQL Programming and Application)
- 資料視覺化 (Data Visualization)
- 機器學習 (Machine Learning)
- 統計品管 (Statistical Quality Control)
- 最佳化 (Optimization)



李明昌博士

alan9956@gmail.com

<http://rwepa.blogspot.com/>

個人簡介 <http://rwepa.blogspot.com/>

- 姓名：李明昌 (ALAN LEE)
- 現職：中華R軟體學會 常務理事
臺灣資料科學與商業應用協會 常務理事
- 學歷：中原大學 工業與系統工程所 博士
- 經歷：
 - 育達科技大學 資訊管理系(所) 專任助理教授
 - 佛光大學 兼任教師
 - 國立台北商業大學 兼任教師
 - 東吳大學 兼任教師
 - 崇友實業 行銷企劃專員
 - 國航船務代理股份有限公司 海運市場運籌管理員
- 大專院校、資策會、工業技術研究院、國家發展委員會、中央氣象局、公平交易委員會、各縣市政府與日本名古屋產業大學等公民營單位演講達300餘場，2600小時以上。
- 連絡資訊：alan9956@gmail.com



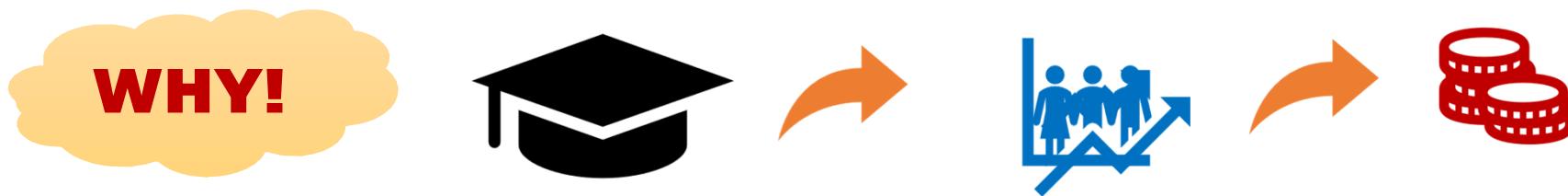
- iPAS 巨量資料分析師 證照推廣
- iPAS 營運智慧分析師 證照推廣

大綱

- 01.Python與Anaconda簡介,資料型別與運算子
- 02.Python資料物件,使用NumPy模組與reshape應用
- 03.字串與正規表示式,判斷式與函數應用
- 04.日期時間資料,檔案匯入pandas
- 05.MySQL與SQL語法,Python連結MySQL應用

如何學習 Python & R?

- 熟悉教材內容
- 將教材內容的資料集改為工作資料集
- 遇到問題時, 想辦法尋找答案
- 掌握 APC方法
- 掌握 摘要, 繪圖, 建模
- 參考網路應用文章 (進階) & 學術論文



R 入門資料分析與視覺化應用(7小時28分鐘)

- <https://mastertalks.tw/products/r?ref=MCLEE>

課程提供教學範例的原始程式檔案與資料集



- **主題**
 1. R, RStudio簡介與套件使用
 2. 認識資料物件
 3. 資料處理與分析
 4. 資料視覺化應用
- **特色**
 1. 資料分析的**關鍵八步**
 2. 提供必備**ggplot2**套件的應用知識與使用情境
 3. 提供日期時間**zoo, xts**套件的整合應用操作
 4. 提供**人力資源**資料與**銷售資料**，強化**實務資料**操作能力

R 商業預測應用(8小時53分鐘)

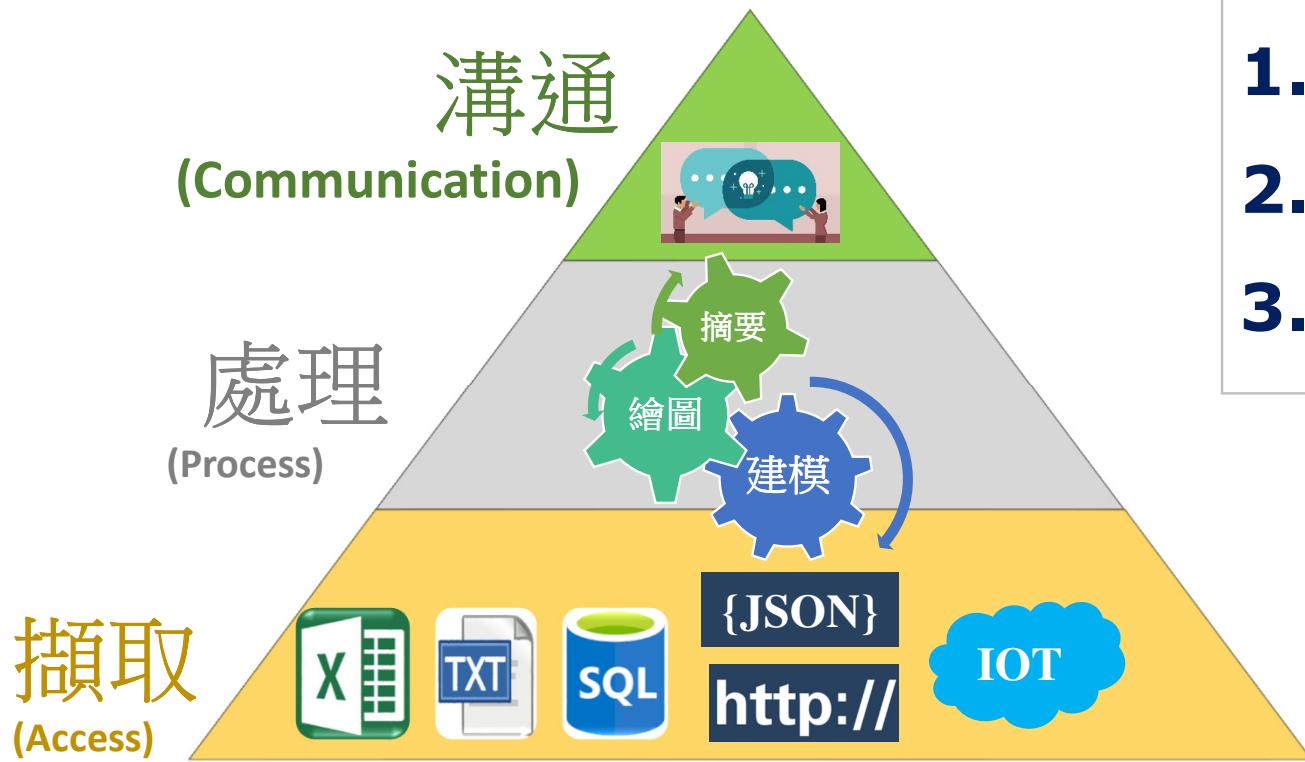
- <https://mastertalks.tw/products/r-2?ref=MCLEE>



課程提供教學範例的原始程式檔案與資料集

- **主題**
 1. R , RStudio工具操作
 2. 非監督式學習商業預測
 3. 監督式學習商業預測
 4. 財金資料預測應用
- **特色**
 1. 採用**最有效率**方式學習大數據R語言，並應用於**職場資料分析**與**商業預測應用**
 2. 提供**多元線性迴歸**的必備知識
 3. 提供**財金資料商業預測應用**的基礎與進階必學技能
 4. 提供學員人力資源資料與**台指期tick資料**預測演練

★★★資料分析架構→APC方法

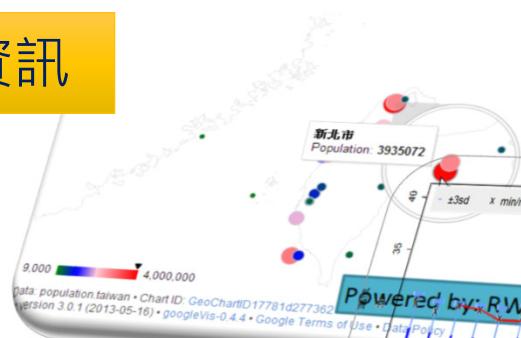


- 1.
- 2.
- 3.

資料分析/視覺化應用

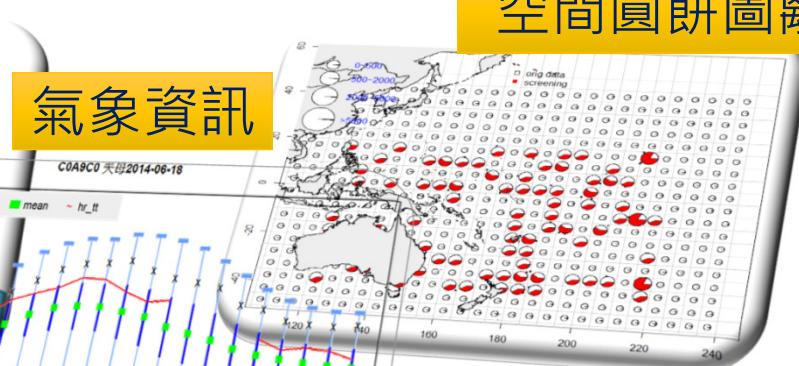
R + shiny → 互動式網頁

地理資訊



空間圓餅圖離群值分析

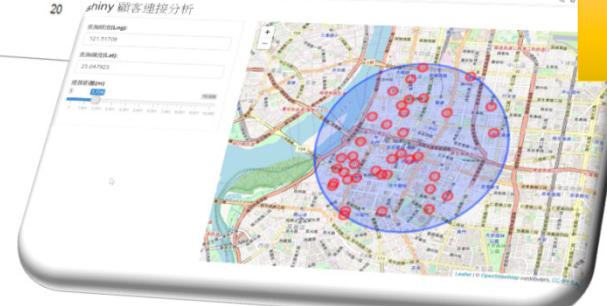
氣象資訊



保險預測



顧客連結資訊



中央氣象局 1,600萬筆資料

網頁呈現



客製化選單

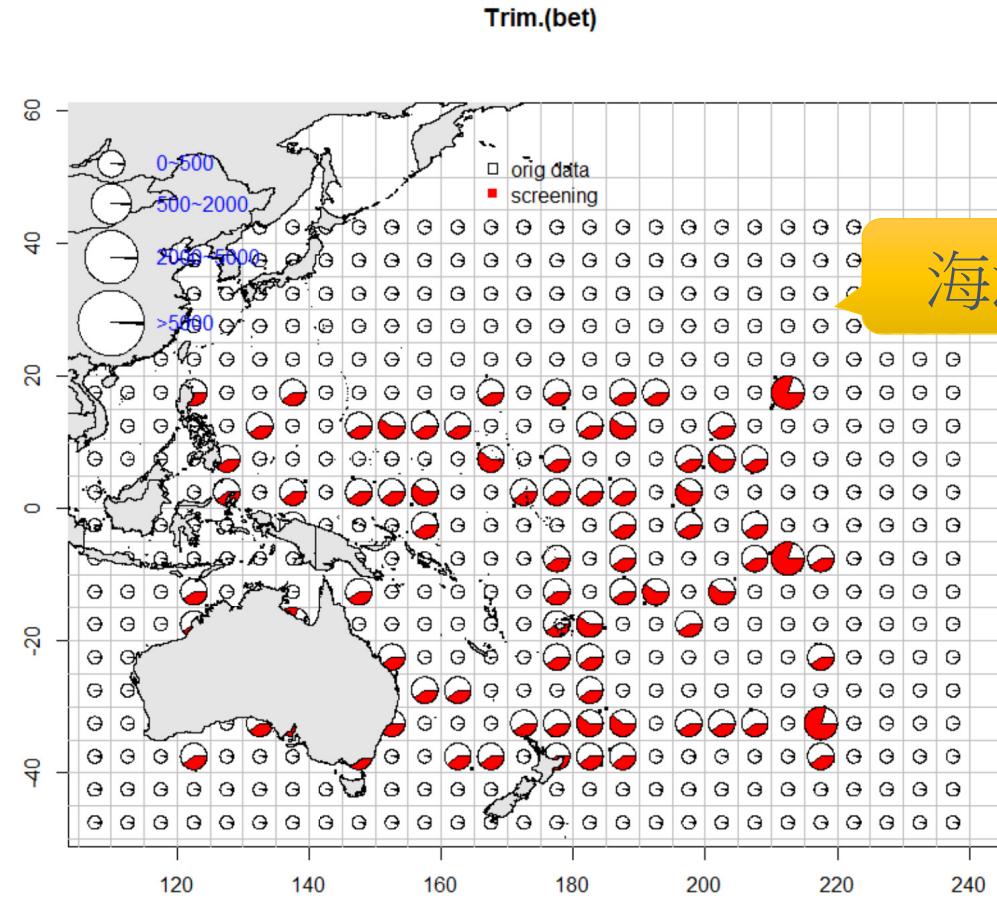
R統計運算

保險預測模型

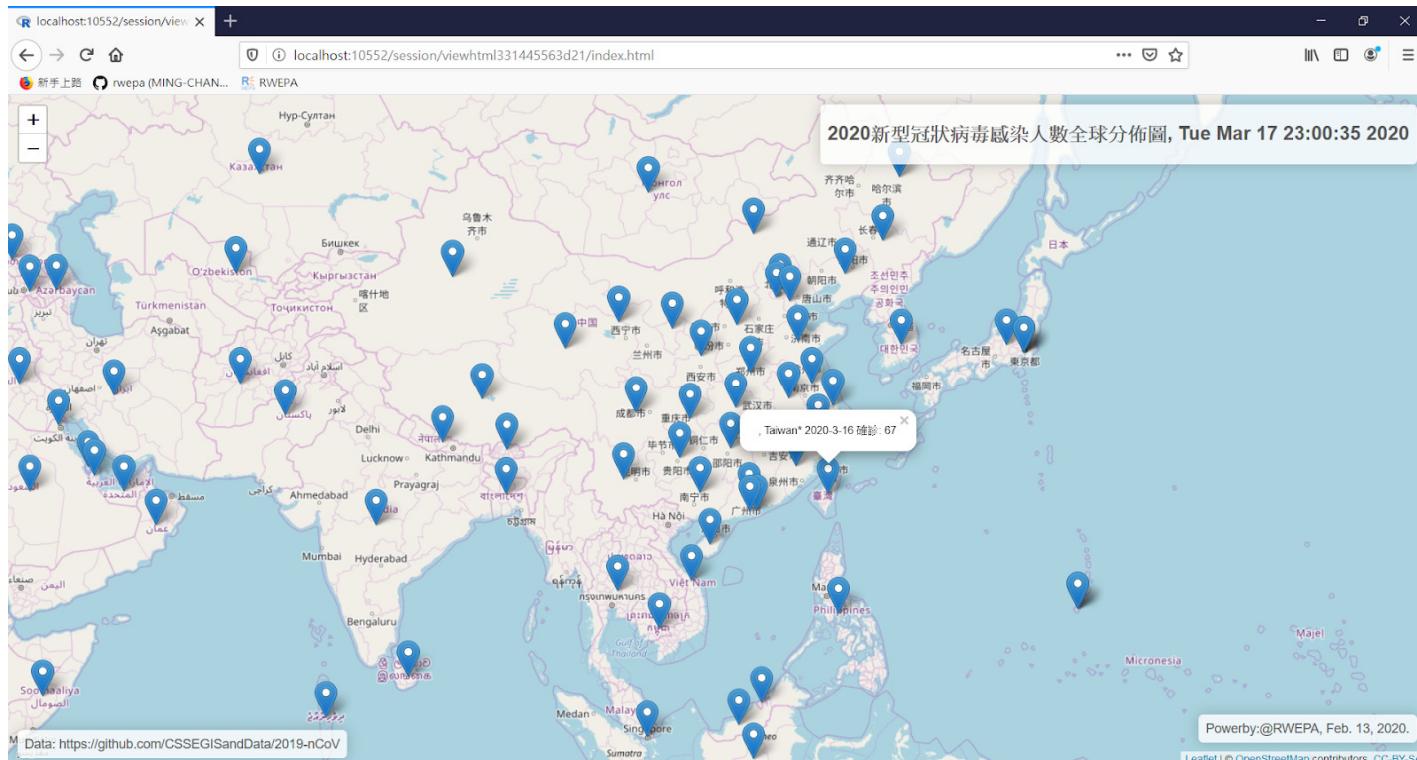
The screenshot shows a web-based data analysis interface for insurance claims prediction. The top navigation bar includes links for file upload, data processing, statistical charts, model evaluation, and prediction models. A red box highlights the 'Prediction Model' dropdown menu. Below it, a yellow callout bubble labeled '機率模型閾值調整' (Probability Model Threshold Adjustment) points to a slider control for setting the 'Probability Model Threshold' (機率模型閾值), currently set at 0.1. Another red box highlights the 'Review Results' (檢視結果) button in the 'Prediction Data Upload' (預測資料上傳) panel. A large yellow callout bubble labeled '預測結果' (Prediction Result) points to the main data table. The table has columns for gender (性別), female (女性), vehicle type (車輛種類), private car (私家車), exposure risk (曝露風險), exposure risk count (曝露風險對數), no claim discount (無索償折扣), insured person age (被保險人年齡), private car 0 (私家車 0), private car 1 (私家車 1), private car 2 (私家車 2), private car 0_1_2 combination (私家車 0_1_2 組合), car age (車齡), car age 0_1_2 combination (車齡 0_1_2 組合), prediction probability (預測機率), and claim (理賠). The table displays 10 entries out of 12, with the last two entries being '無' (No). The bottom of the page shows navigation links for previous, next, and first pages.

M	0	A	1	0.9144422	-0.08944106	50	4	1	0	0	1	0	2	0.1069	有
M	0	A	1	0.8158795	-0.20348856	20	4	0	0	1	1	2	2	0.1441	有
3	M	0	A	1	0.8377823	-0.17699695	50	3	0	0	1	1	2	0.1866	有
4	M	0	A	1	0.4325804	-0.83798702	50	6	0	1	0	1	1	0.0944	無
5	M	0	A	1	0.7173169	-0.33223755	50	4	0	0	1	1	2	0.1218	有
6	M	0	A	1	0.8377823	-0.17699695	50	4	0	0	1	1	2	0.1495	有
7	M	0	A	1	0.8487337	-0.16400975	50	5	0	0	1	1	2	0.1422	有
8	F	1	A	1	0.8268309	-0.19015503	10	3	0	0	1	1	2	0.1733	有
9	M	0	A	1	0.7145791	-0.33606164	0	5	1	0	0	1	0	0.0694	無
10	M	0	A	1	0.3340178	-1.09656101	0	3	0	0	1	1	2	0.0783	無

空間圓餅圖離群值分析



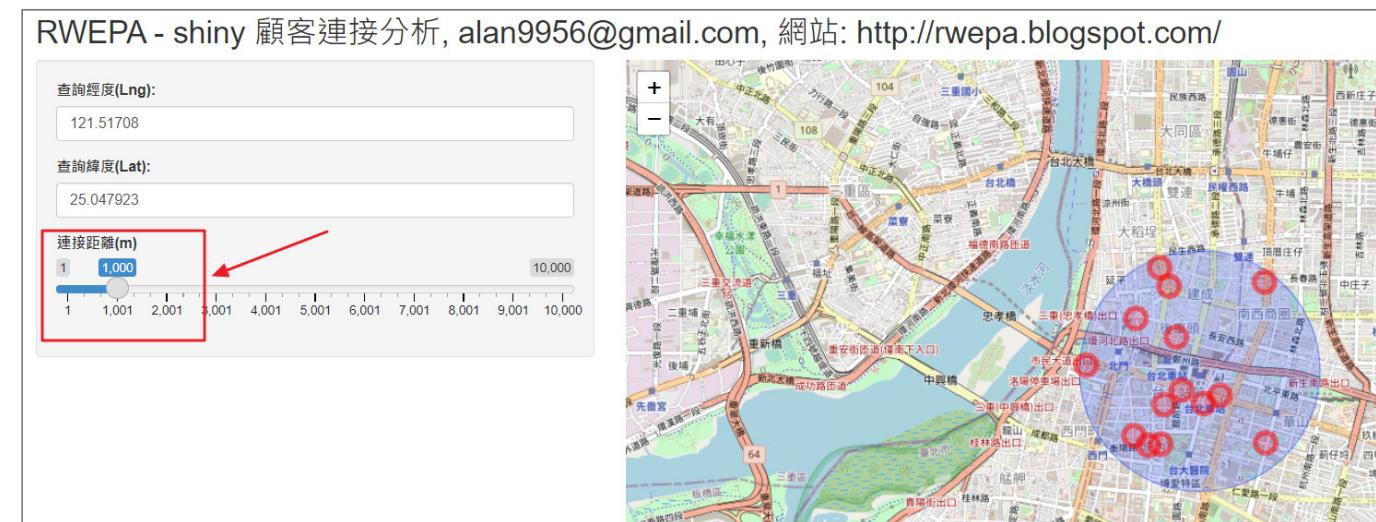
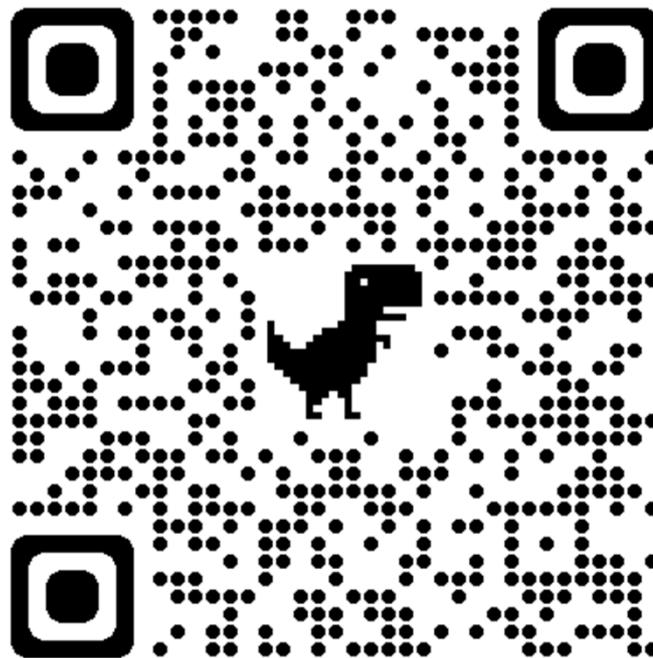
2020新型冠狀病毒視覺化



<http://rwepa.blogspot.com/2020/02/2019nCoV.html>

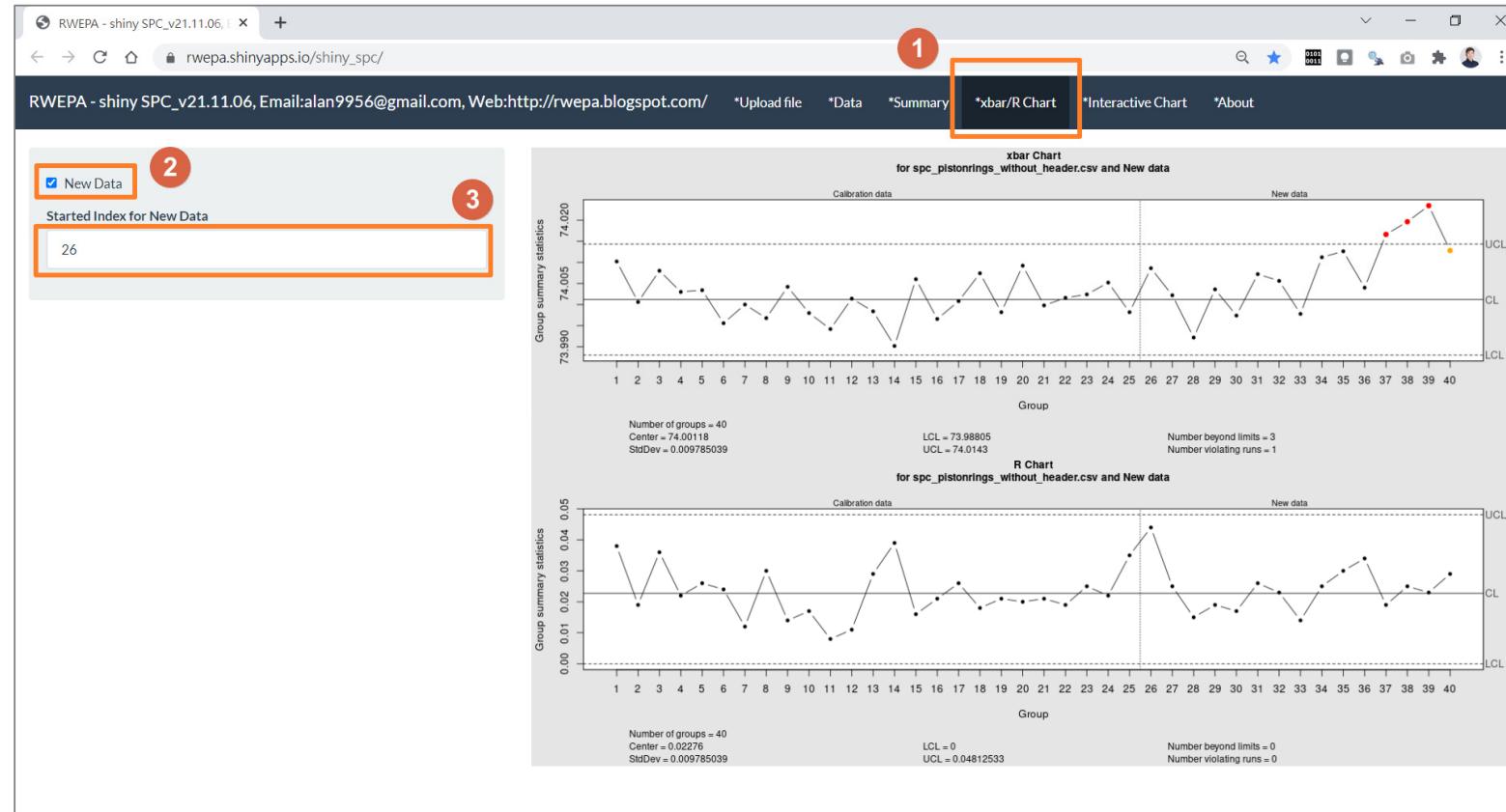
shiny 顧客連接分析

- <https://rwepa.shinyapps.io/shinyCustomerConnect/>



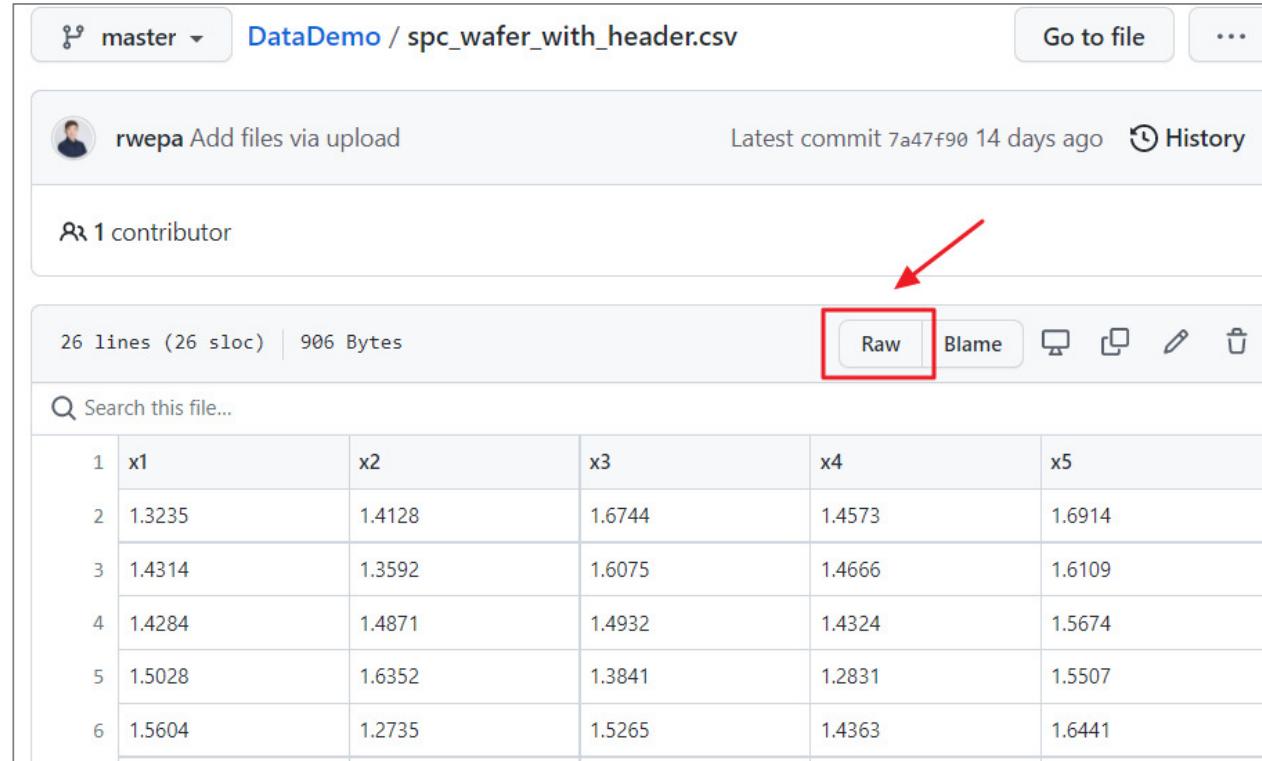
品質管制圖(quality control chart)應用

- <http://rwepa.blogspot.com/2021/10/r-shiny-quality-control-chart.html>



品質管制圖應用 (續)

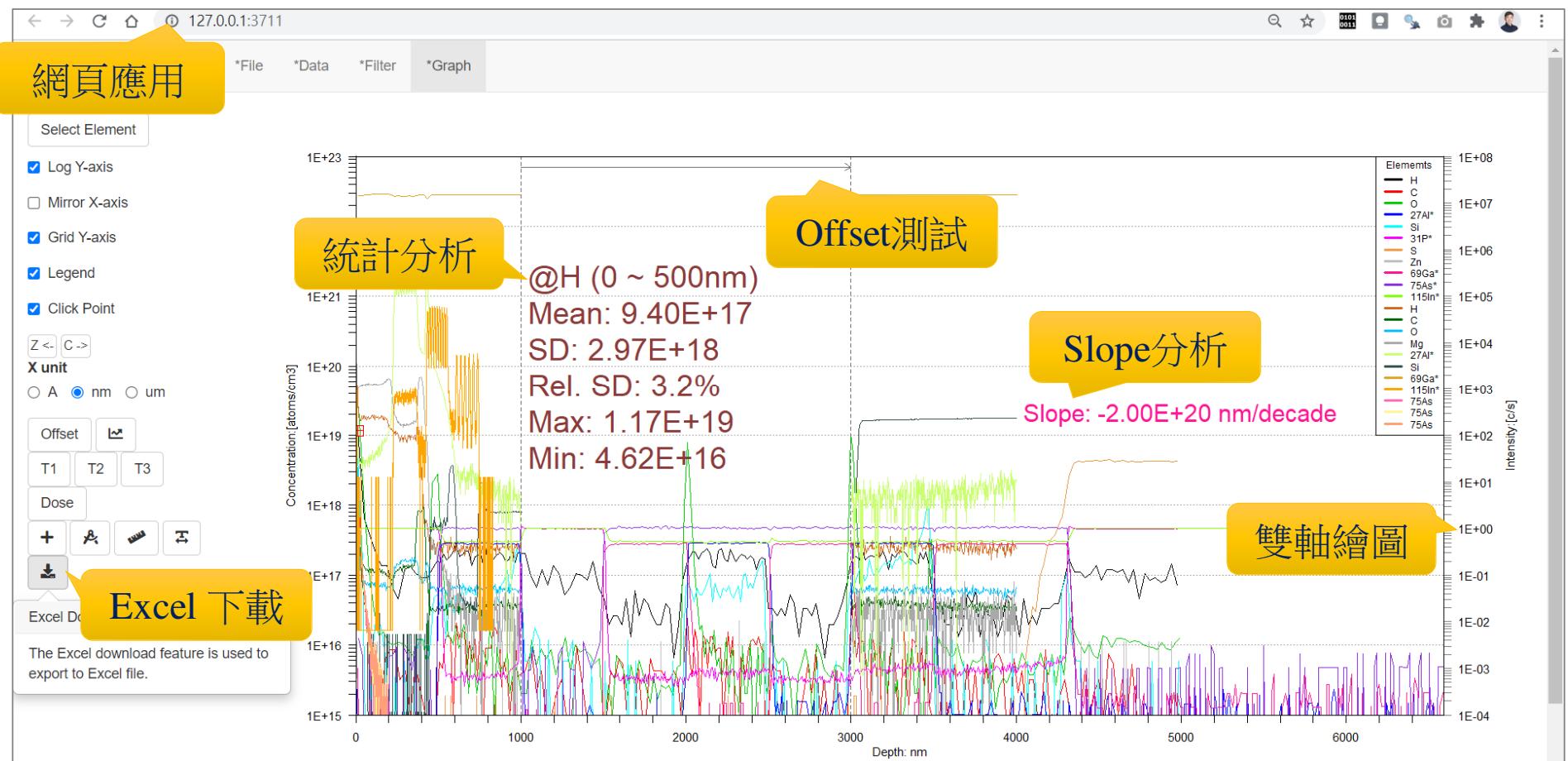
- https://github.com/rwepa/DataDemo/blob/master/spc_wafer_with_header.csv
- https://github.com/rwepa/DataDemo/blob/master/spc_pistonrings_without_header.csv



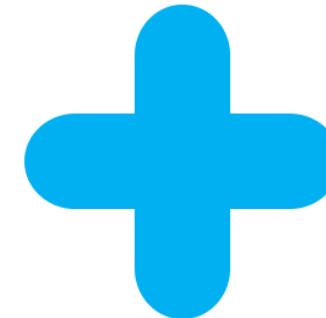
The screenshot shows a GitHub file viewer for `spc_wafer_with_header.csv`. The file was uploaded by `rwepa` 14 days ago. It has 1 contributor. The file contains 26 lines (26 sloc) and 906 Bytes. The `Raw` button in the toolbar is highlighted with a red box and an arrow pointing to it.

1	x1	x2	x3	x4	x5
2	1.3235	1.4128	1.6744	1.4573	1.6914
3	1.4314	1.3592	1.6075	1.4666	1.6109
4	1.4284	1.4871	1.4932	1.4324	1.5674
5	1.5028	1.6352	1.3841	1.2831	1.5507
6	1.5604	1.2735	1.5265	1.4363	1.6441

離子資料分析與視覺化應用



學習目標



01.Python與Anaconda簡介,資料型別與運算子



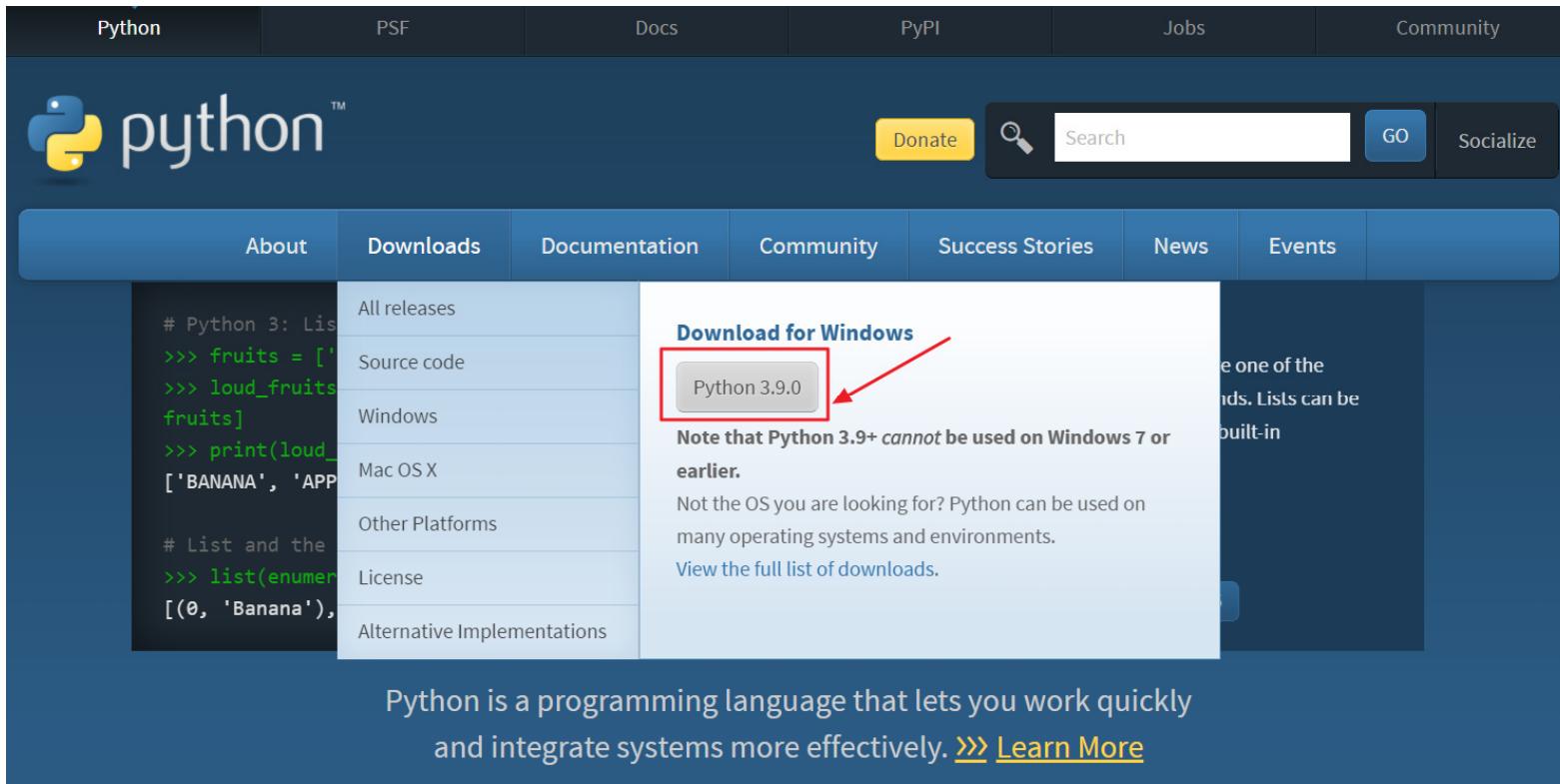
Python 簡介與安裝

Python 簡介

- 吉多·范羅蘇姆 (Guido van Rossum) 在1989年的聖誕節期間研發 Python 語言。
 - https://en.wikipedia.org/wiki/Guido_van_Rossum
 - Python 3.9.6 - 2021年6月28日
- 特性：
 - 跨平台
 - 開放性
 - 易讀性
 - 動態語言
 - 直譯語言
 - 豐富套件(模組)
 - 其他語言結合, 例: Cython 編譯成執行檔(.exe)

Python 下載

- <https://www.python.org/>

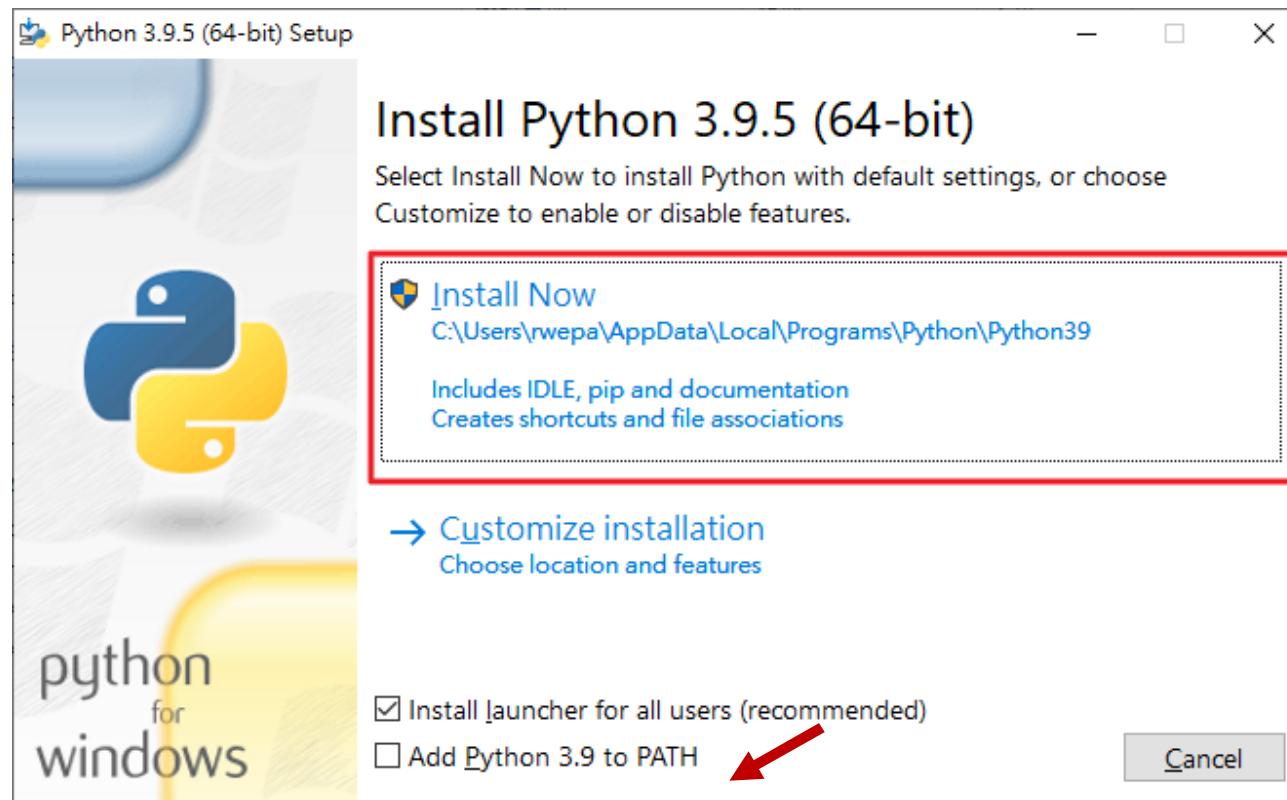


The screenshot shows the Python.org homepage with a focus on the 'Downloads' section. On the left, there's a code snippet demonstrating Python syntax. The 'Downloads' menu item is highlighted. In the center, there's a large 'Download for Windows' button with 'Python 3.9.0' written on it, which is also highlighted with a red box and an arrow pointing to it. Below this button, a note states: 'Note that Python 3.9+ cannot be used on Windows 7 or earlier.' To the right, there's a sidebar with some text and a 'View the full list of downloads.' link.

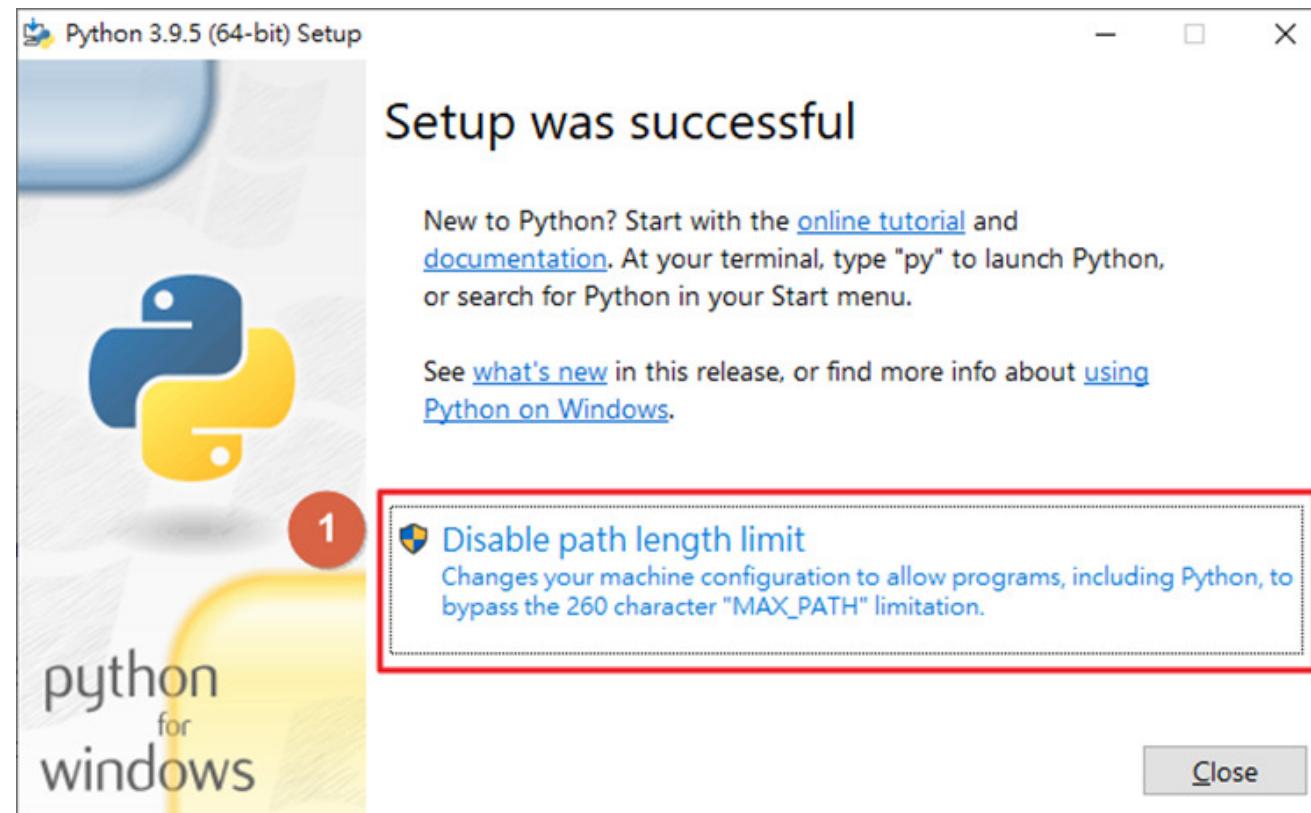
Python is a programming language that lets you work quickly
and integrate systems more effectively. [» Learn More](#)

Python 安裝

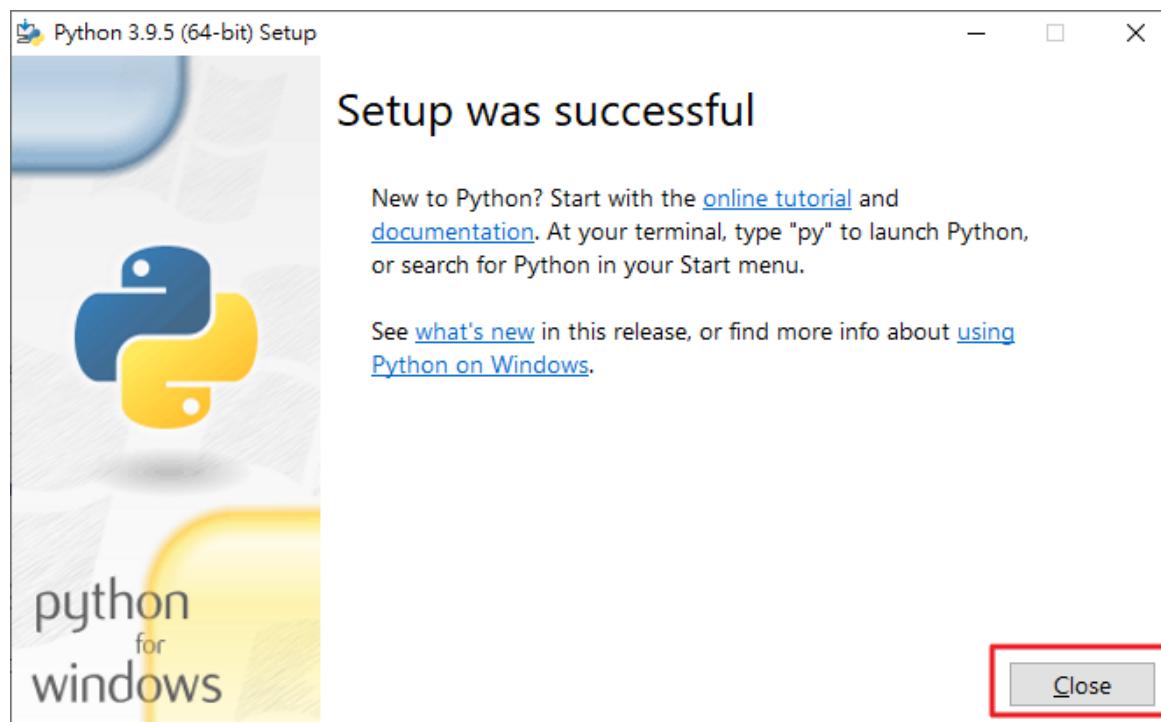
- python-3.9.5-amd64.exe



Python 安裝 – 取消字元長度限制



Python 安裝完成



Python 執行 <方法1> Python 3.9(64-bit)

```
Python 3.9 (64-bit)
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> help('print')
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
        file: a file-like object (stream); defaults to the current sys.stdout.
        sep: string inserted between values, default a space.
        end: string appended after the last value, default a newline.
        flush: whether to forcibly flush the stream.

>>> print('Hello World - RWEPA')
Hello World - RWEPA
>>> quit()
```

Windows 環境變數 Path: 加入
;C:\Windows\System32

- `help(print)`
- `print('Hello world')`
- `quit()`

Python 執行 <方法2> 命令提示字元



```
命令提示字元
Microsoft Windows [版本 10.0.19042.928]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\rwepa>python
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

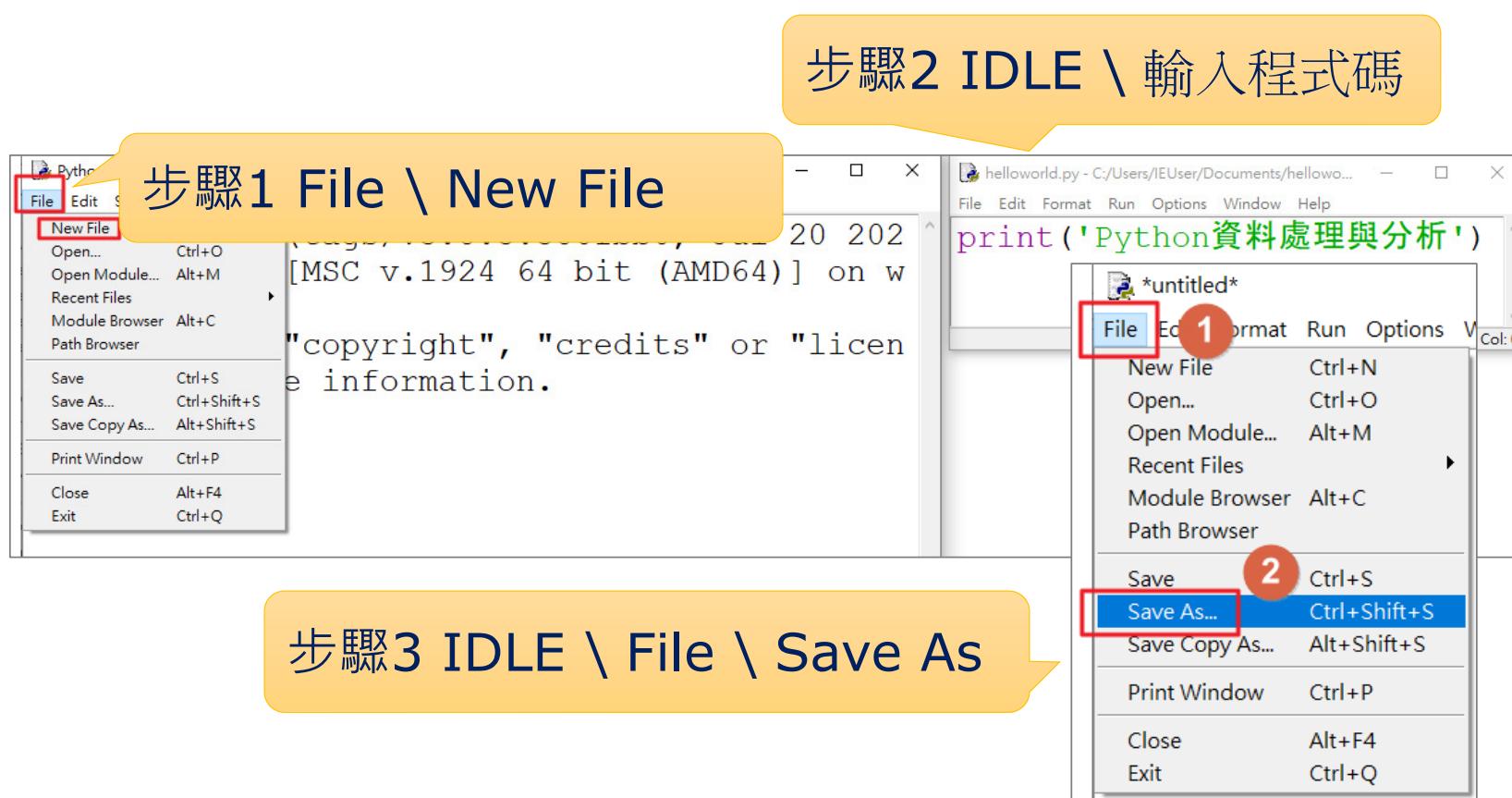
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()

C:\Users\rwepa>
```

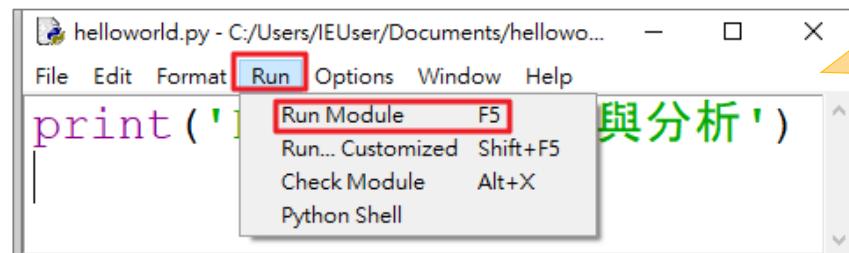
- python
- 1+2
- quit()

Python 執行 <方法3> IDLE模式

- IDLE 可先將檔案編輯與儲存



Python 執行 <方法3> IDLE模式 (續)



步驟4 Run \ Run Module

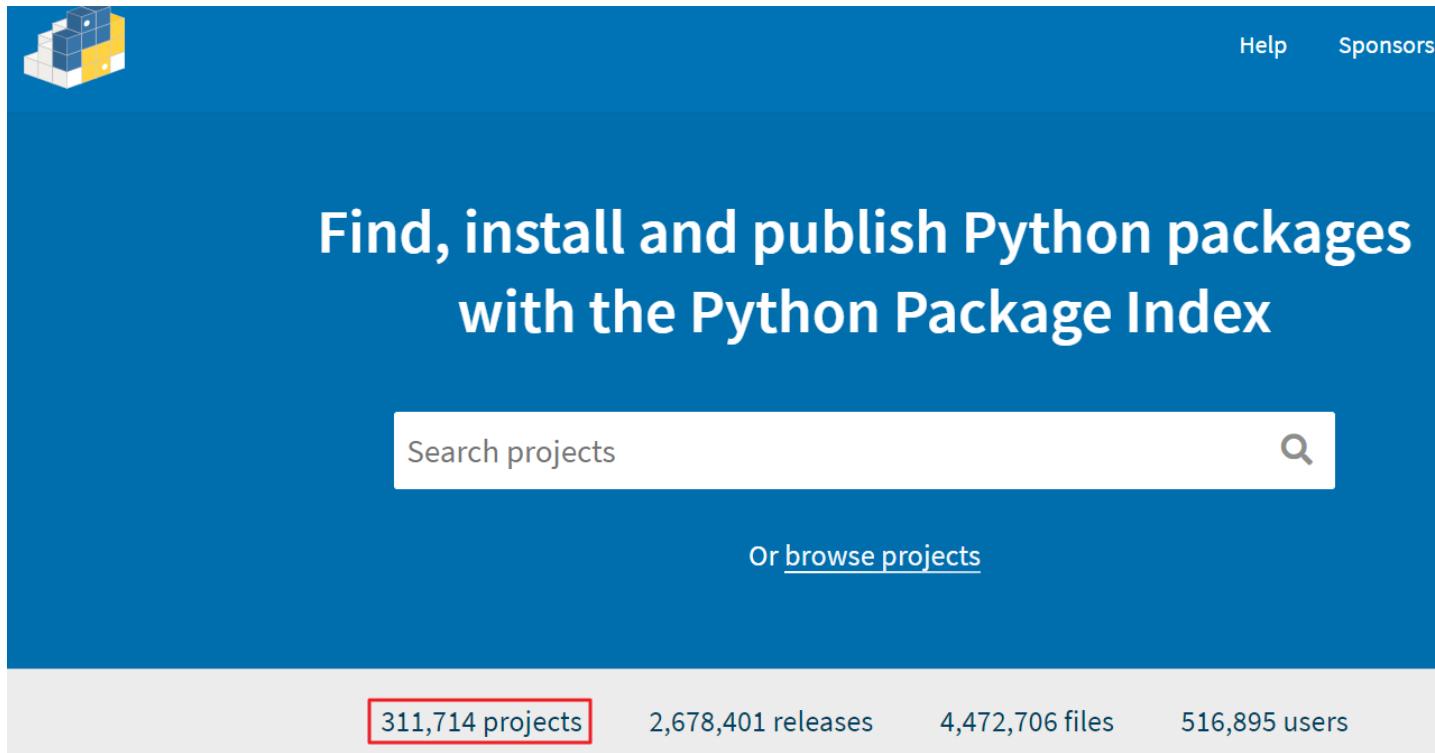
步驟5 執行結果

```
>>>
= RESTART: C:/Users/IEUser/Documents/helloworld.py
Python資料處理與分析
>>>
>>>
```

The screenshot shows the Python IDLE shell window. It displays the Python startup message and the command-line interface. The output of the script execution is highlighted with a red box, showing the text "Python資料處理與分析". The status bar at the bottom right indicates "Ln: 7 Col: 4".

PyPI (Python Package Index)

- <https://pypi.org/>
- 31多萬專案總表



已安裝模組(pip)

```
命令提示字元
Microsoft Windows [版本 10.0.19043.1055]
(c) Microsoft Corporation. 著作權所有，並保有所有權利。
C:\Users\88697>pip list
Package           Version
-----
absl-py          0.12.0
appdirs          1.4.4
astunparse       1.6.3
cachetools      4.2.1
certifi          -
chardet          -
defusedxml       -
distlib          -
engineering-n   Help on module os:
```

- **pip list**
- 模組說明
python -c help('模組')
- 按 Q 關閉說明

```
命令提示字元 - python -c help('os')
C:\Users\88697>python -c help('os')
Help on module os:

NAME
    os - OS routines for NT or Posix depending on what system we're on.

MODULE REFERENCE
    https://docs.python.org/3.8/library/os

    The following documentation is automatically generated from the Python
    source files. It may be incomplete, incorrect or include features that
    are considered implementation detail and may vary between -- More --
```

Python IDE

- Anaconda, 包括 Spyder:
 - <https://www.anaconda.com/>
- PyCharm:
 - <https://www.jetbrains.com/pycharm/>
- WinPython:
 - <http://winpython.github.io/>
- RStudio - Terminal 視窗
 - <https://www.rstudio.com/products/rstudio/>
- Visual Studio Code
 - <https://code.visualstudio.com/docs/python/python-tutorial>
- Google:
 -  Google Colaboratory

示範軟體 (Spyder)

IDE 整合開發環境-
Integrated
Development
Environment

RStudio – 執行 Python

The screenshot shows the RStudio interface with a Python script file open in the code editor. The script contains several lines of Python code, including imports for os and matplotlib, and operations like changing the current working directory. A yellow callout bubble points to the terminal window where the script is being run.

步驟2 執行 Ctrl + Alt + Enter

```

5 author: Ming-Chang Lee
6 email : alan9956@gmail.com
7 RWEPA : http://rwepa.blogspot.tw/
8
9
10 # 檔案資料夾 c:/pythondata
11 # 01.python.讀我.py      , 檔案說明
12 # 02.type.operations.py , 資料操作
13 # 03.matplotlib.demo.py , matplotlib繪圖
14 # 04.understandingdata.py , 資料理解 pandas
15 # 05.regression.py      , 迴歸分析,線性模型
16 # 06.logistic.regression.py , logistic regression-分類
17 # 07.decision.tree.py   , 決策樹 decision tree & 隨機森林法 random forest
18 # 08.decision.tree-codes.py , 決策樹 Python 程式碼
19 # 09.associationrule.py , 關聯規則
20
21 # 切換工作目錄
22 import os # 載入 os 套件
23 os.getcwd() # 讀取工作目錄
24 os.chdir("C:/pythondata") # 變更工作目錄
25 os.getcwd()
26 os.listdir(os.getcwd()) # 顯示檔案清單
27
28 # 顯示模組提供之函數
2633

```

步驟1 先在 Terminal 視窗輸入 python

步驟3 顯示結果

The terminal window shows the Python interpreter running, with the command 'python' entered and the resulting output of the script's execution. A yellow callout bubble points to the results displayed in the terminal.

```

C:\rdata>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Anaconda custom (64-bit) on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import os # 載入 os 套件
>>> os.getcwd() # 讀取工作目錄
'C:\rdata'
>>> os.chdir("C:/pythondata") # 變更工作目錄
>>> os.getcwd()
'C:\pythondata'
>>> os.listdir(os.getcwd()) # 顯示檔案清單
['01.python.讀我.py', '02.type.operations.py', '03.matplotlib.demo.py', '04.understandingdata.py', '05.regression.py', '06.logistic.regression.py', '07.decision.tree.py', '08.decision.tree-codes.py', '09.associationrule.py', '2017年10月每小時網路流量.png', 'data', 'random.plot.pdf', 'random.plot.png', 'titanic-RWEPA.png']
>>>

```

Anaconda 簡介與安裝



Anaconda 特性

- Anaconda是一個免費、易於安裝/管理並支援Python語言
- 支援7500個以上的資料科學模組 (package)
- 支援模組的下載, 安裝, 更新
- 支援 Spyder (支援 Python IDE)
- 支援 jupyter notebook
- 支援 Windows、Mac OS X和Linux

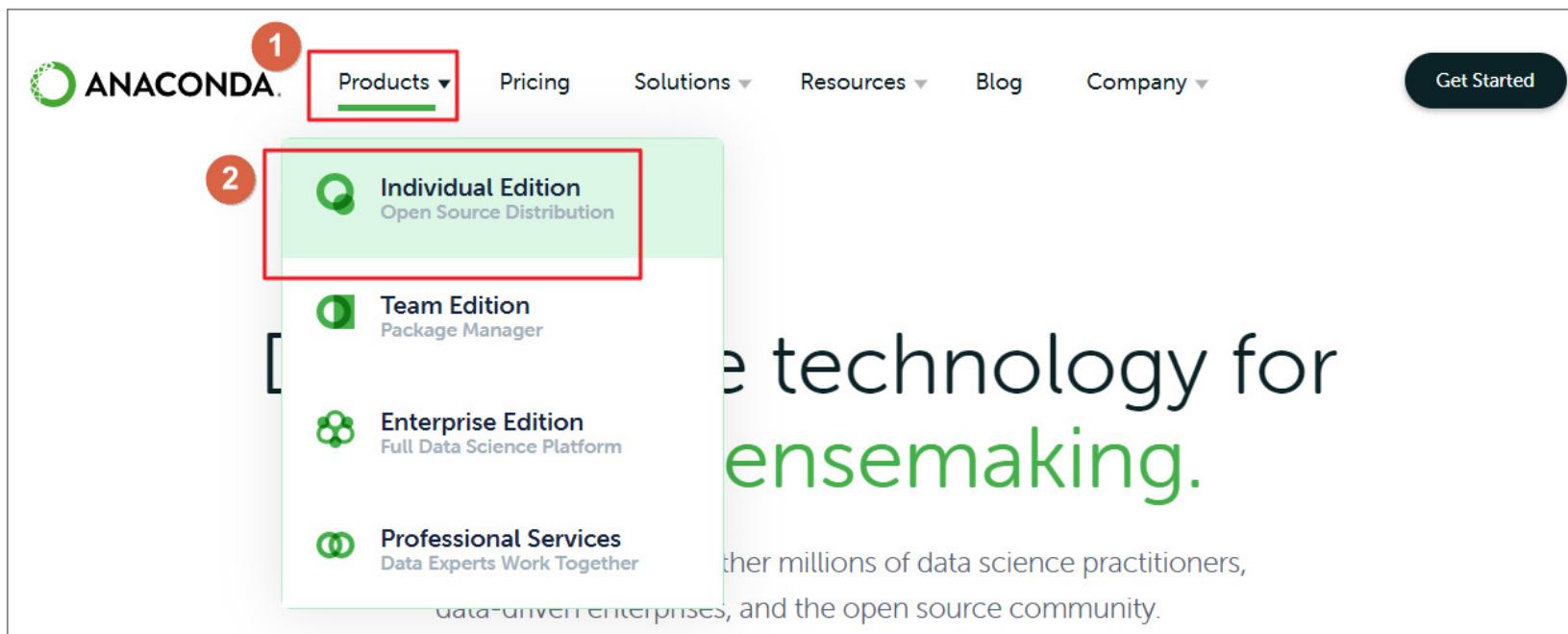
Anaconda 特性 (續)



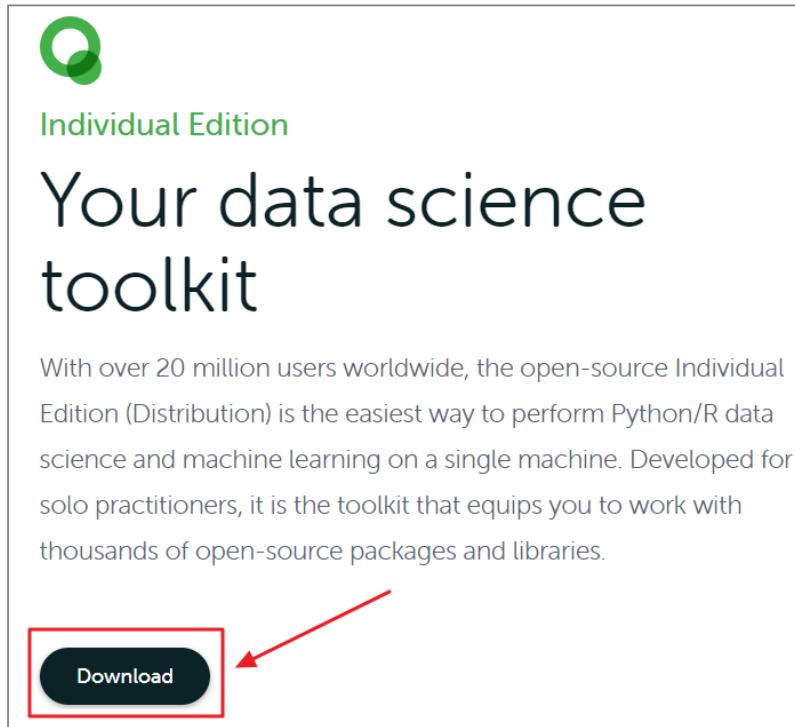
source: <https://www.anaconda.com/distribution/>

Anaconda 下載

- <https://www.anaconda.com/>



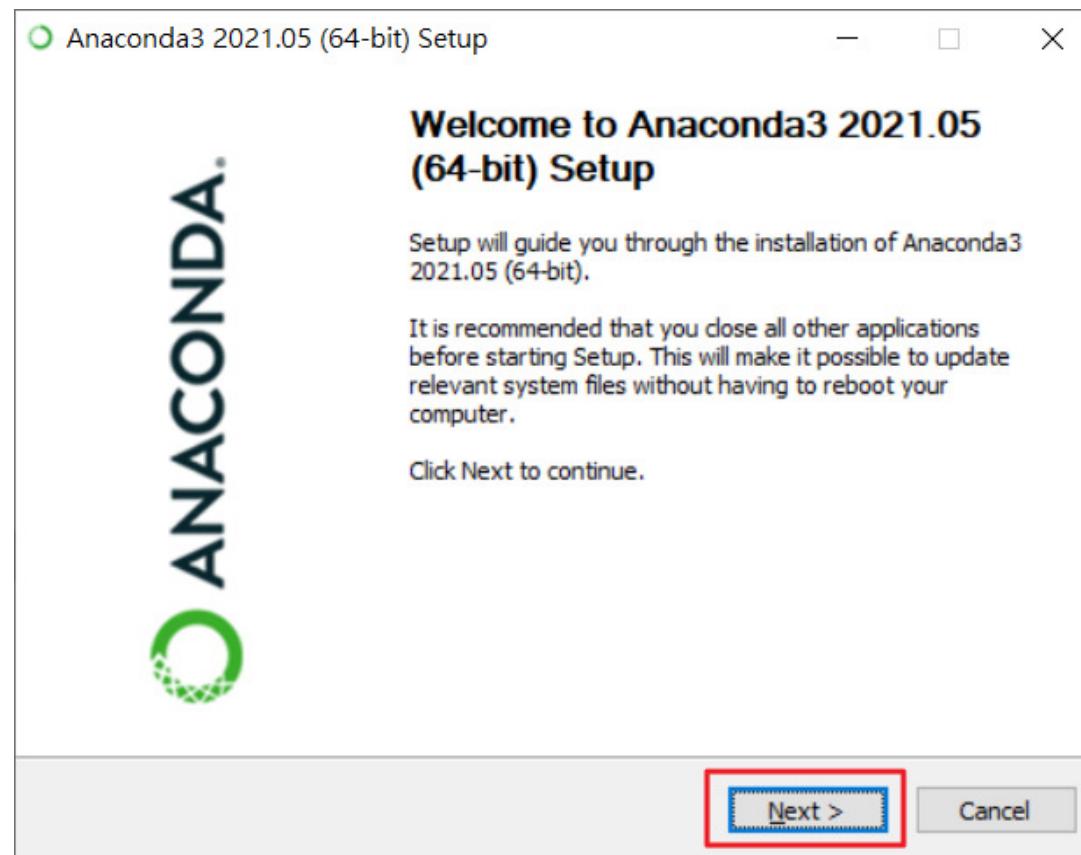
Anaconda 下載 (續)



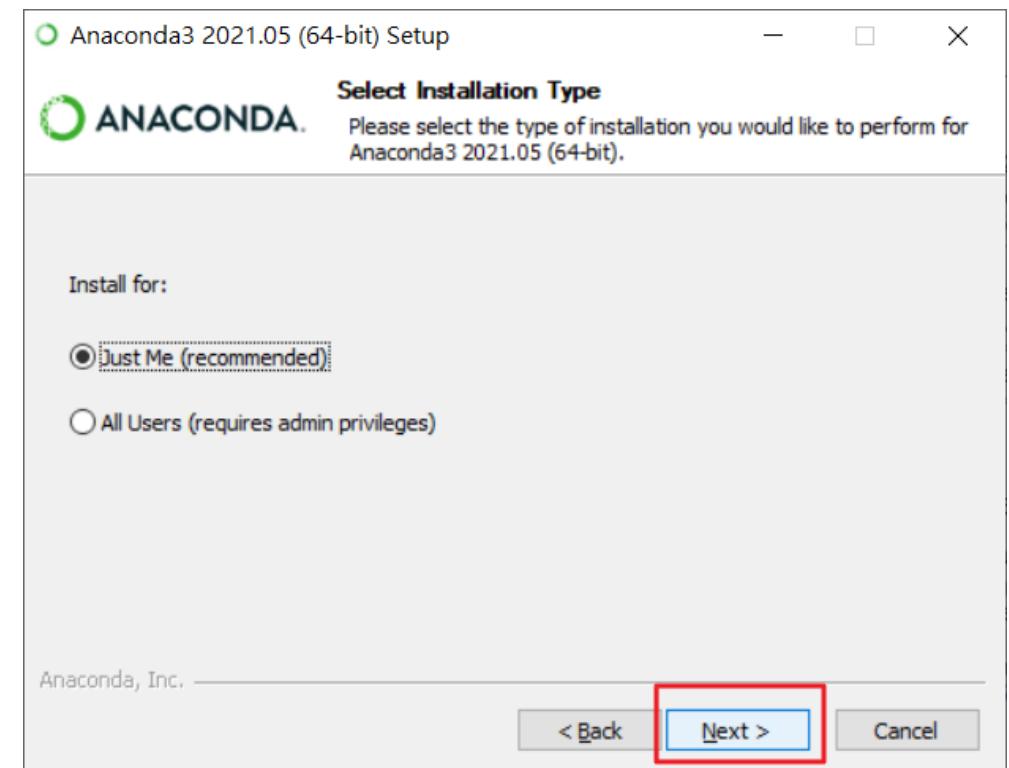
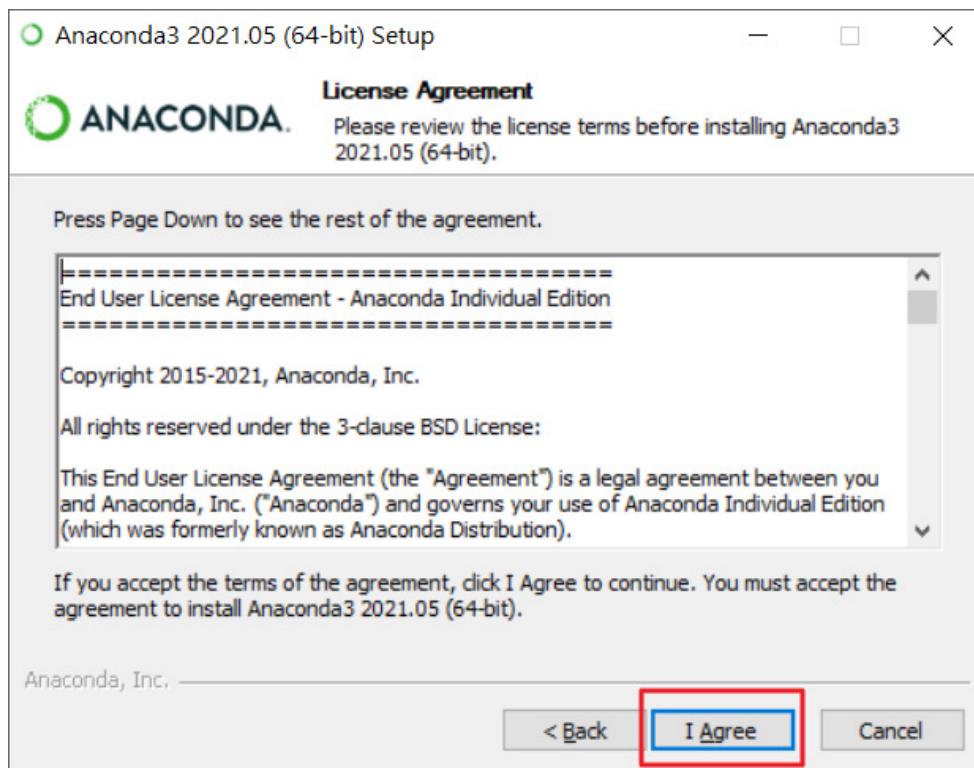
Anaconda3-2021.05-Windows-x86_64.exe

下載 64位元,
477 MB

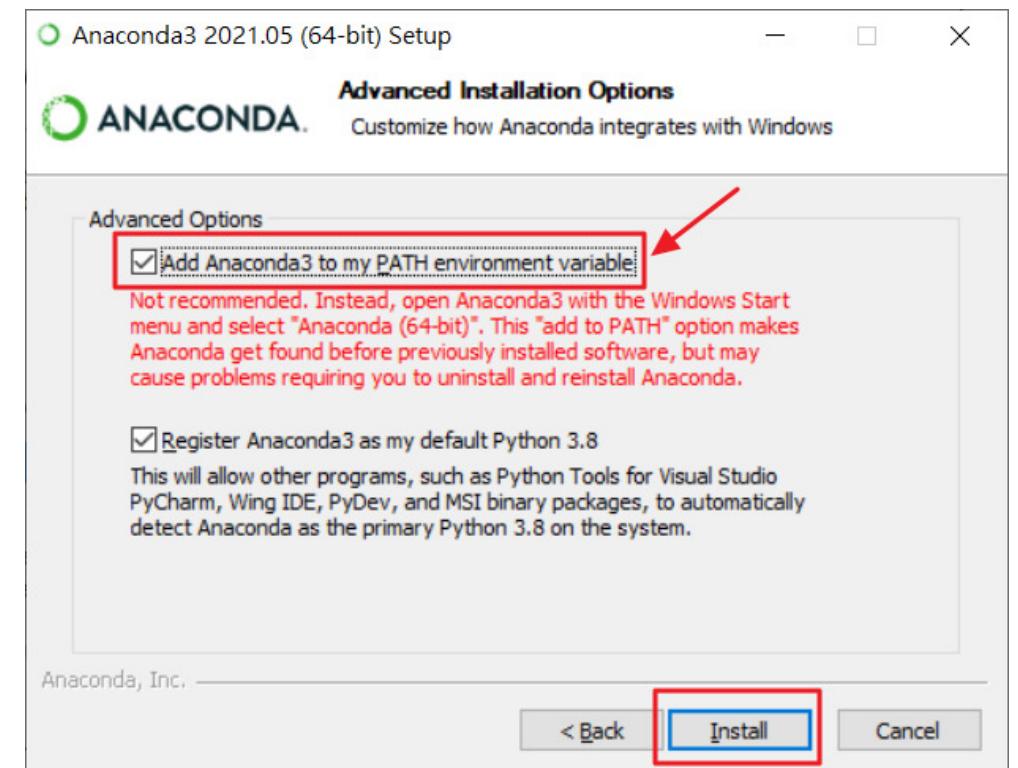
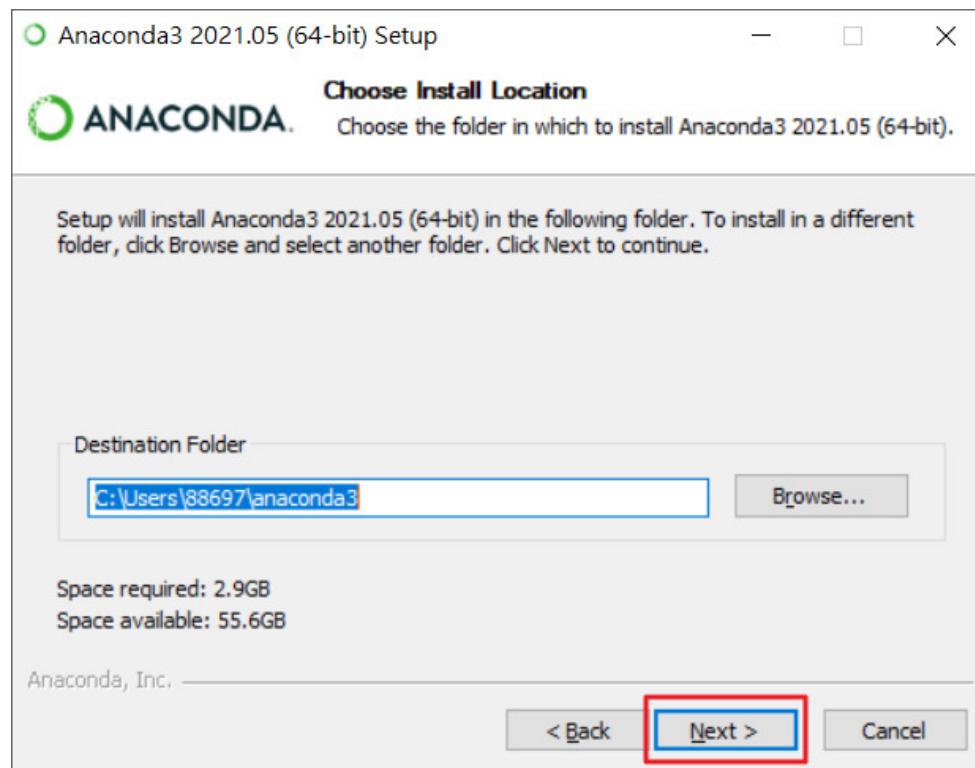
安裝 Anaconda



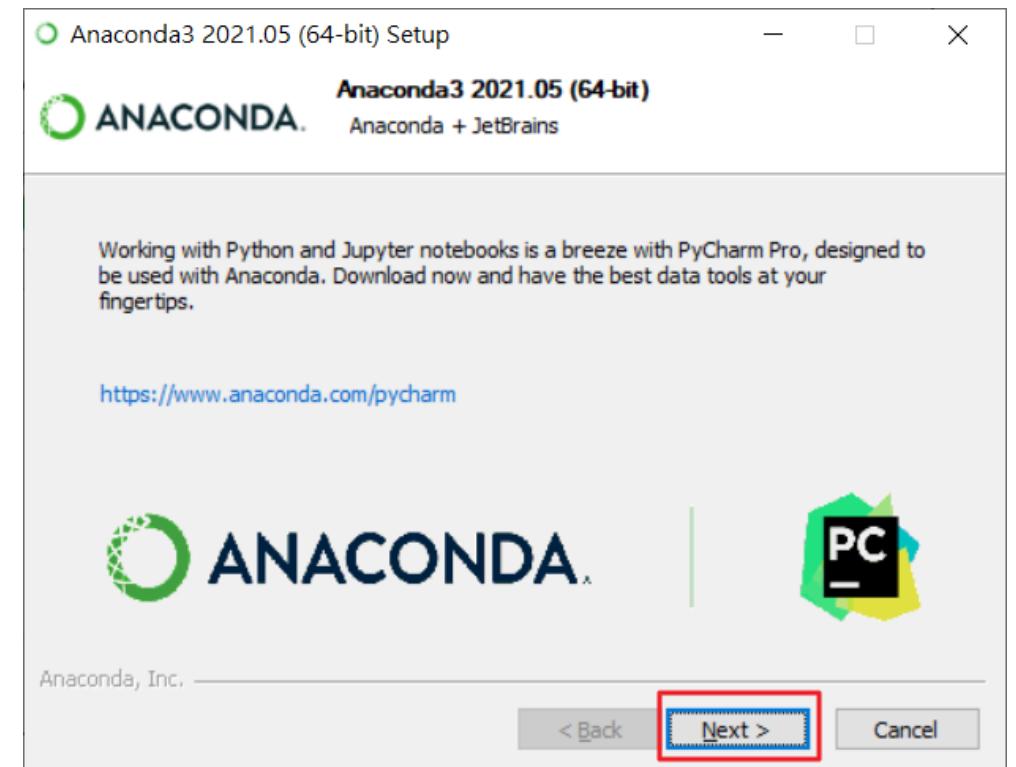
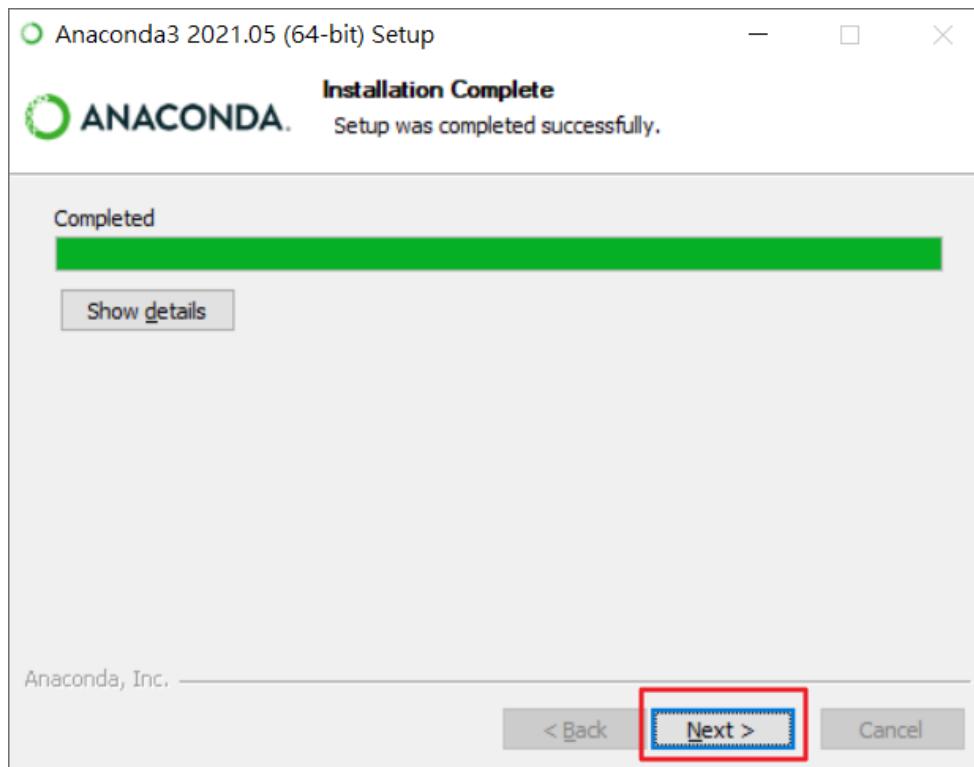
安裝畫面



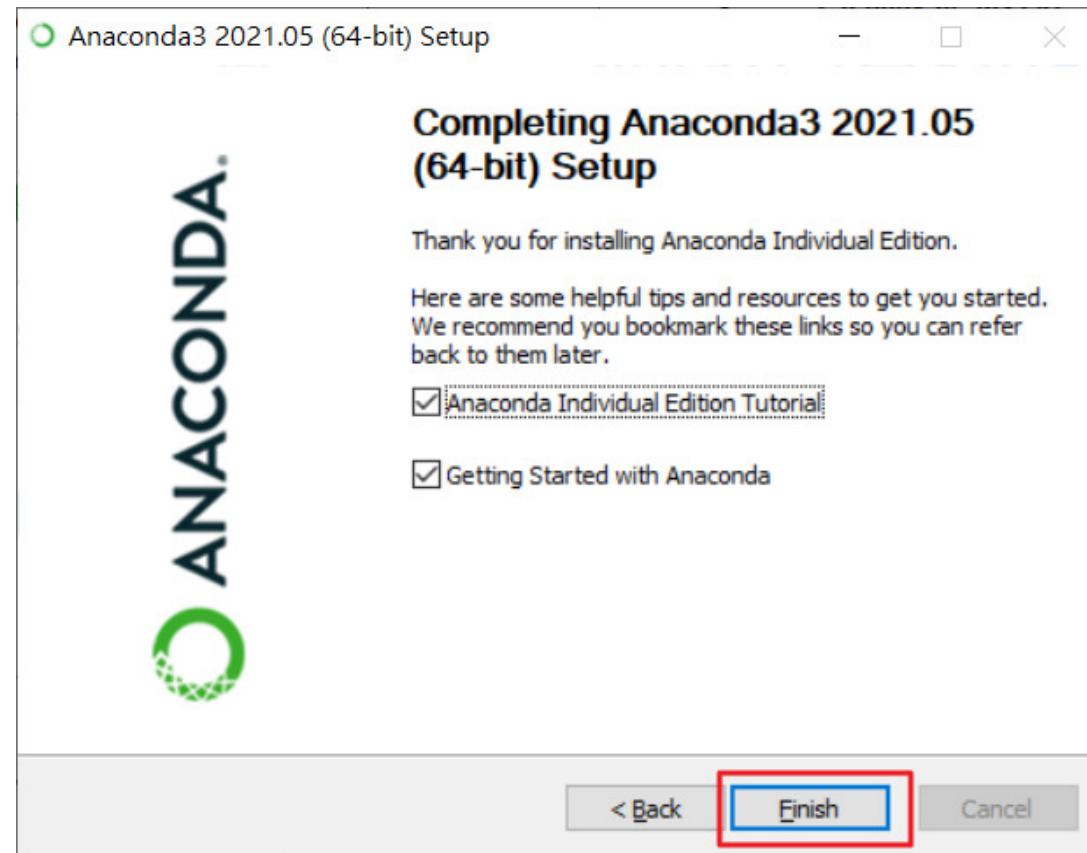
二個選項都打勾



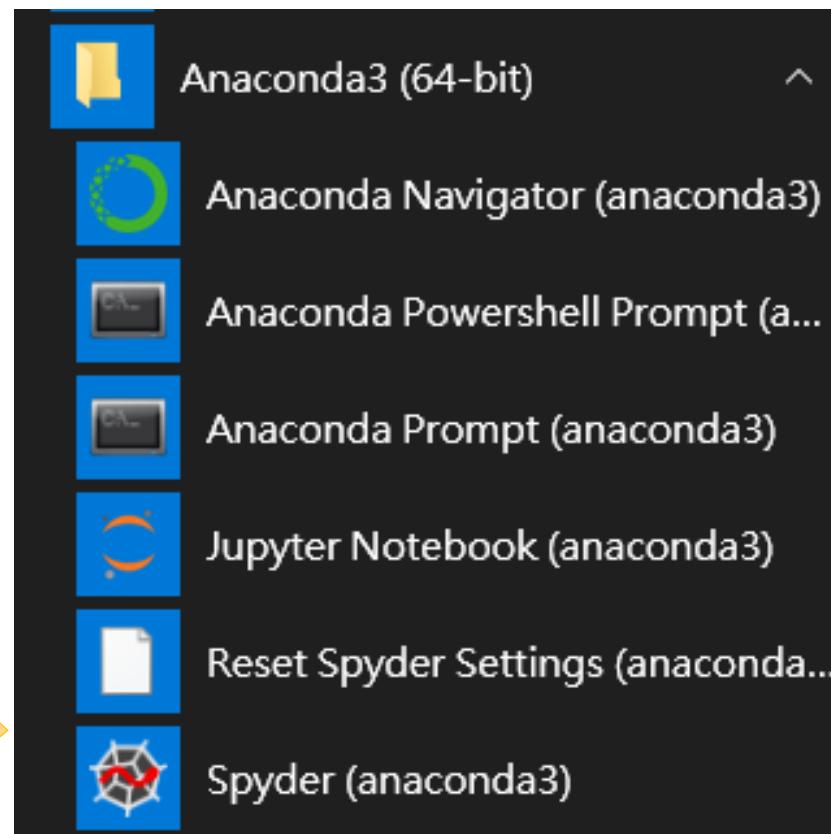
安裝畫面



Anaconda 安裝完成



Anaconda3 (64-bit)



已安裝
6個元件

Anaconda Navigator

The screenshot shows the Anaconda Navigator interface. On the left is a sidebar with links to Home, Environments, Learning, and Community. A central grid displays various applications:

- CMD.exe Prompt (0.1.1): Run a cmd.exe terminal with your current environment from Navigator activated. [Launch](#)
- Datalore (0.1.0): Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team. [Launch](#)
- IBM Watson Studio Cloud (0.1.0): IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling. [Launch](#)
- JupyterLab (3.0.14): An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. [Launch](#)
- Jupyter Notebook (6.3.0): Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. [Launch](#)
- Powershell Prompt (0.0.1): Run a Powershell terminal with your current environment from Navigator activated. [Launch](#)
- IPyConsole (5.0.3): PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. [Launch](#)
- Spyder (4.2.5): Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features. [Launch](#)
- VS Code (1.52.1): Streamlined code editor with support for development operations like debugging, task running and version control. [Launch](#)
- Glueviz (1.0.0): Multidimensional data visualization across files. Explore relationships within and among related datasets. [Install](#)
- Orange 3 (3.26.0): Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox. [Install](#)
- PyCharm Professional (2020.3.3): A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL. [Install](#)
- RStudio (1.1.456): A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks. [Launch](#)

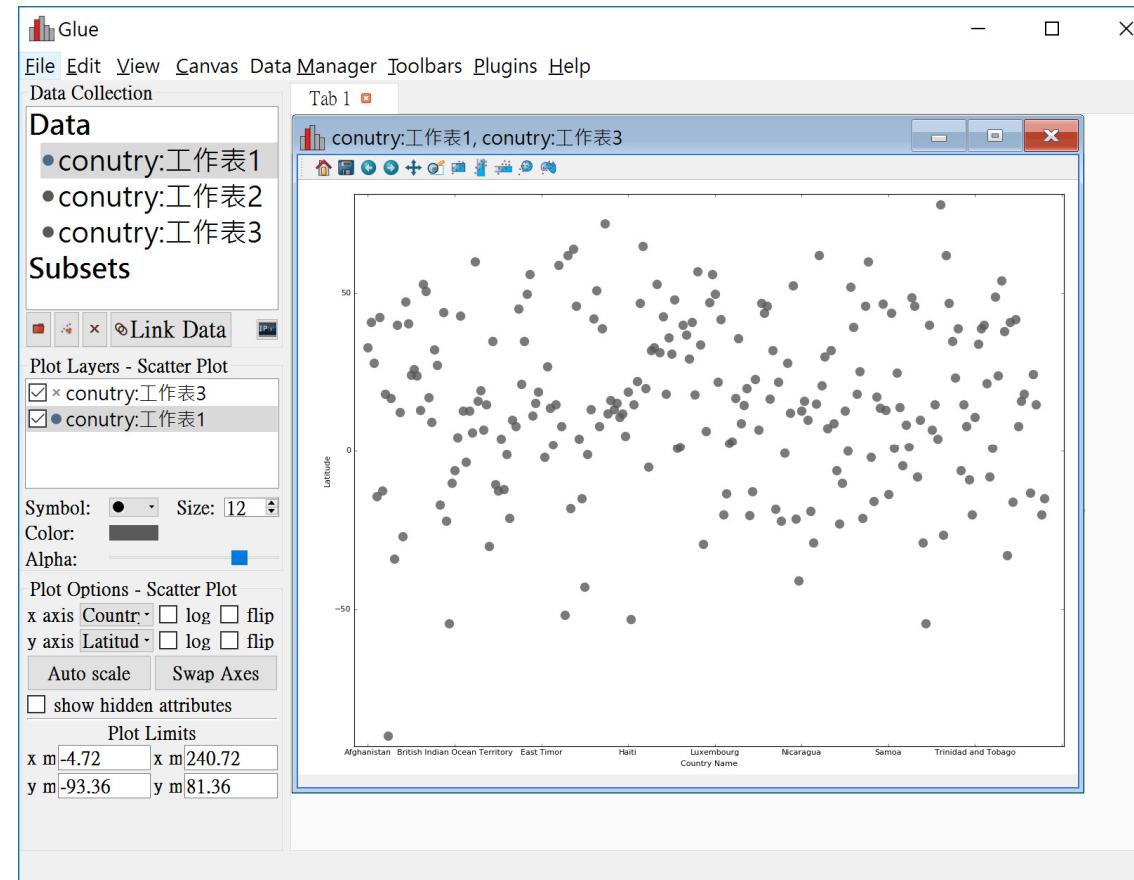
A yellow callout box contains the text: 不同電腦, 安裝模組可能有差異 (Different computers, module installation may differ).

A yellow callout box at the bottom contains the text: 注意: 不要在此安裝 RStudio, 可能會有問題? (Attention: Do not install RStudio here, there may be problems?).

Two bullet points in another yellow callout box are:

- Launch 可直接啟動
- Install 須安裝

glueviz 視覺化



jupyter notebook

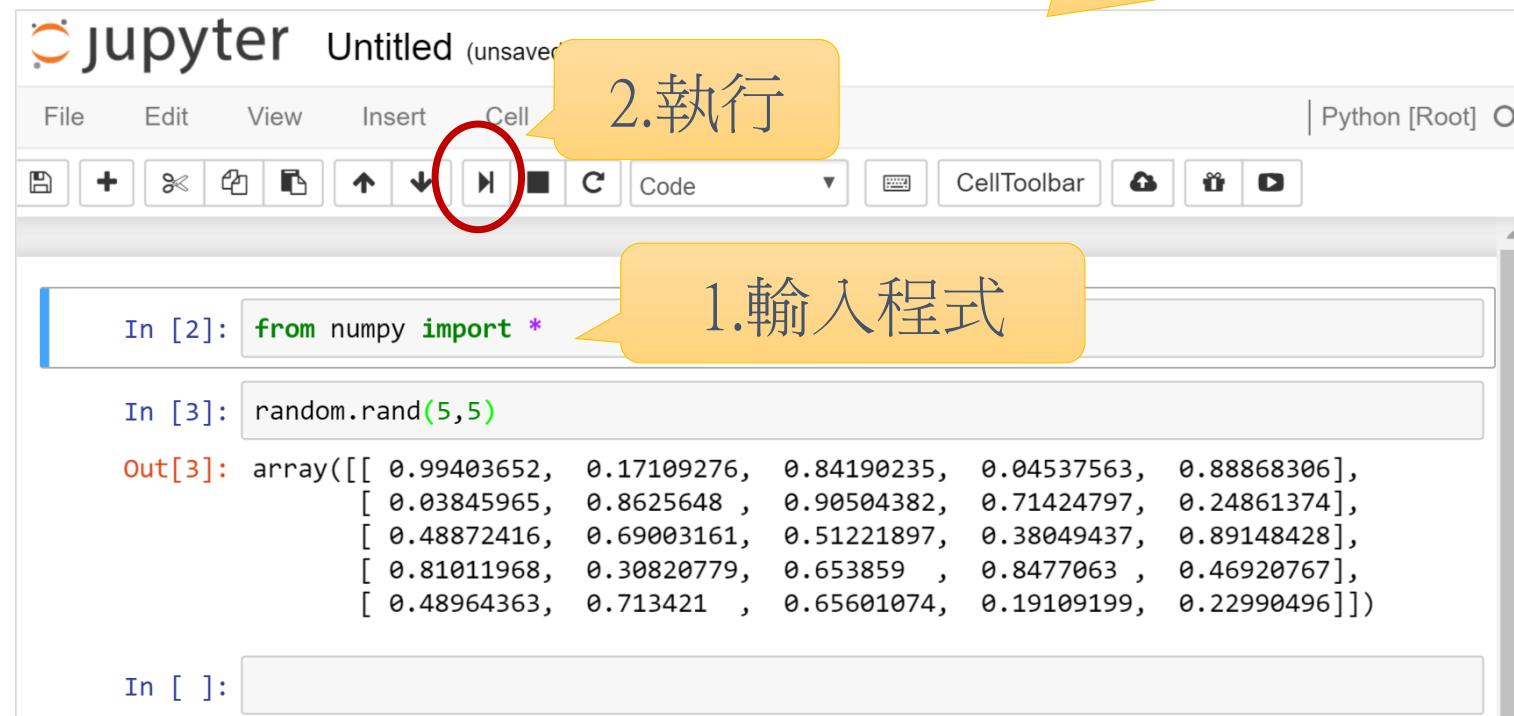




jupyter notebook (續)

理解重新命名方式

實作練習1



The screenshot shows the Jupyter Notebook interface with the following steps highlighted:

1. 輸入程式 (Input Program): In [2]: `from numpy import *`
2. 執行 (Execute): A yellow speech bubble points to the execute button in the toolbar.

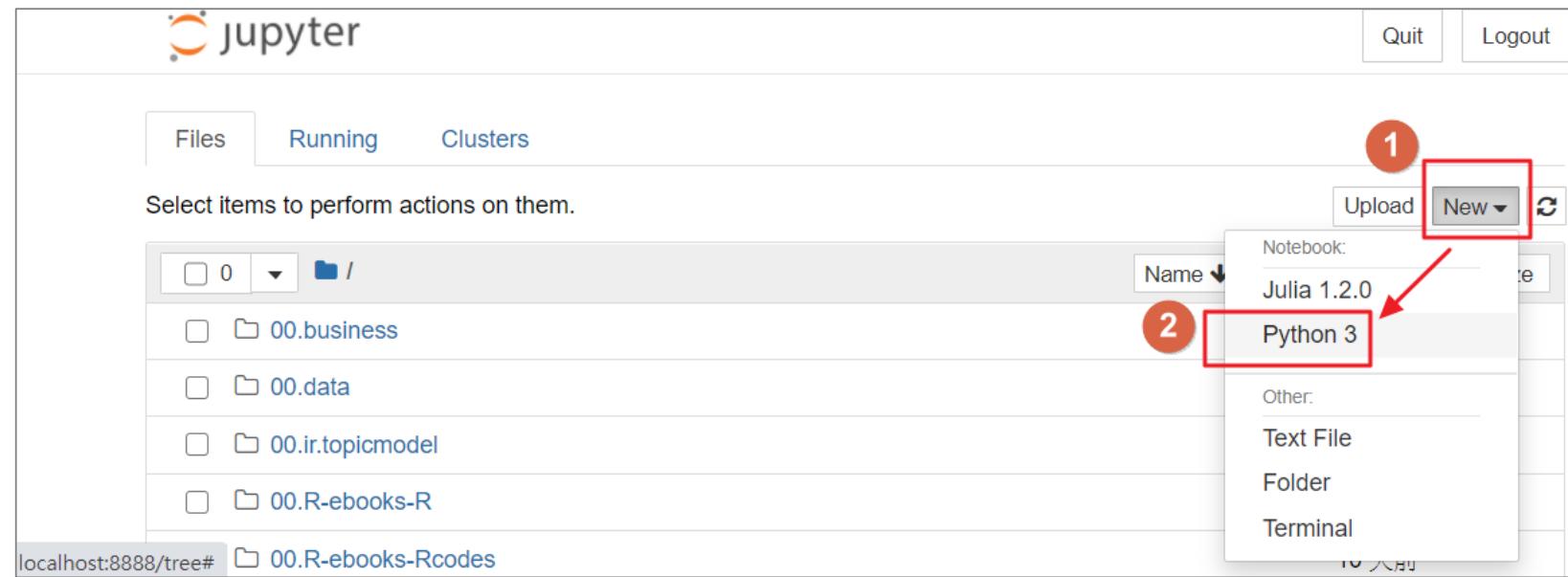
In [3]: `random.rand(5,5)`

Out[3]: `array([[0.99403652, 0.17109276, 0.84190235, 0.04537563, 0.88868306],
 [0.03845965, 0.8625648 , 0.90504382, 0.71424797, 0.24861374],
 [0.48872416, 0.69003161, 0.51221897, 0.38049437, 0.89148428],
 [0.81011968, 0.30820779, 0.653859 , 0.8477063 , 0.46920767],
 [0.48964363, 0.713421 , 0.65601074, 0.19109199, 0.22990496]])`

In []:

jupyter notebook – 更改預設目錄

- cd C:\
- jupyter-notebook



Jupyter Notebook 快速鍵

- 按 [Esc] cell旁邊為藍色：
 - 按 **x**：刪除當前選擇的cell
 - 按 **a**：在當前選擇的上方新增一個cell
 - 按 **b**：在當前選擇的下方新增一個cell
 - 按 **Shift + Enter**：執行當前的cell並且選到下一個cell
 - 按 **Ctrl + Enter**：執行當前cell
 - 按 **M**：轉成markerdown模式，可以看到紅色框框內容從code變成markerdown



開啟 → Python程式設計-李明昌.ipynb, 瀏覽加入數學式主題

- <http://rwepa.blogspot.com/2020/02/pythonprogramminglee.html>

實作練習2

主題: Python 程式設計-李明昌 - ipynb

檔名: Python_Programming_Lee_ipynb.zip

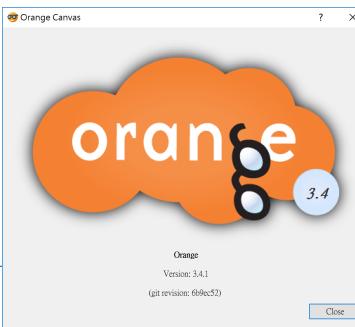
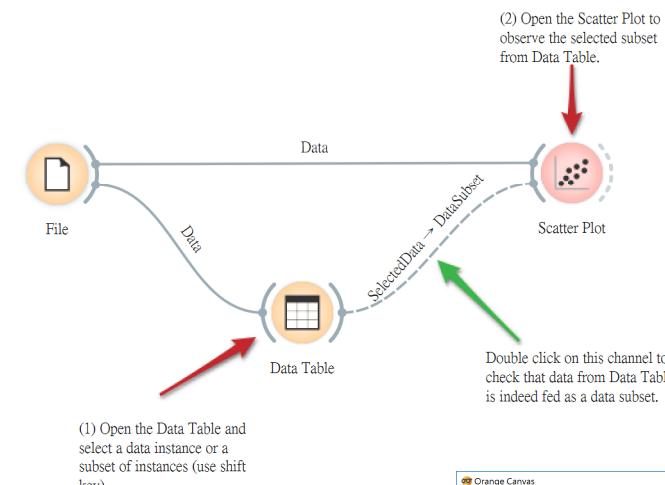
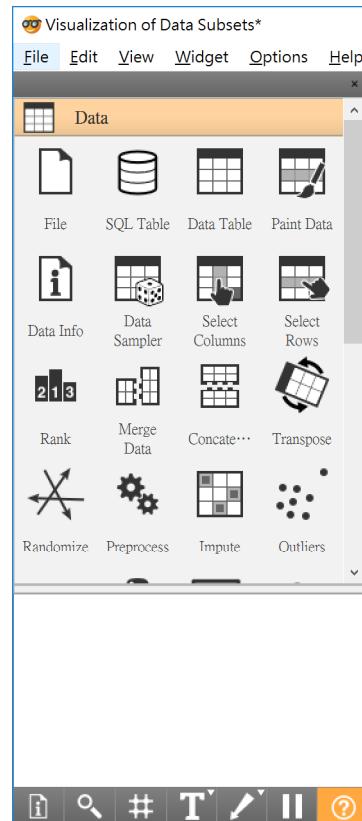
包括 Python 程式設計-李明昌電子書的原始 ipynb 檔案, 圖檔, 部分資料集

下載: https://github.com/rwepa/DataDemo/blob/master/Python_Programming_Lee_ipynb.zip



Python_Programming_Lee_ipynb.zip > python.book.lee >	
名稱	類型
.ipynb_checkpoints	檔案資料夾
data	檔案資料夾
img	檔案資料夾
Python程式設計-李明昌.ipynb	IPYNB 檔案

Orange



安裝 Orange

conda install -c conda-forge orange3

使用命令提示列 開啟 Orange

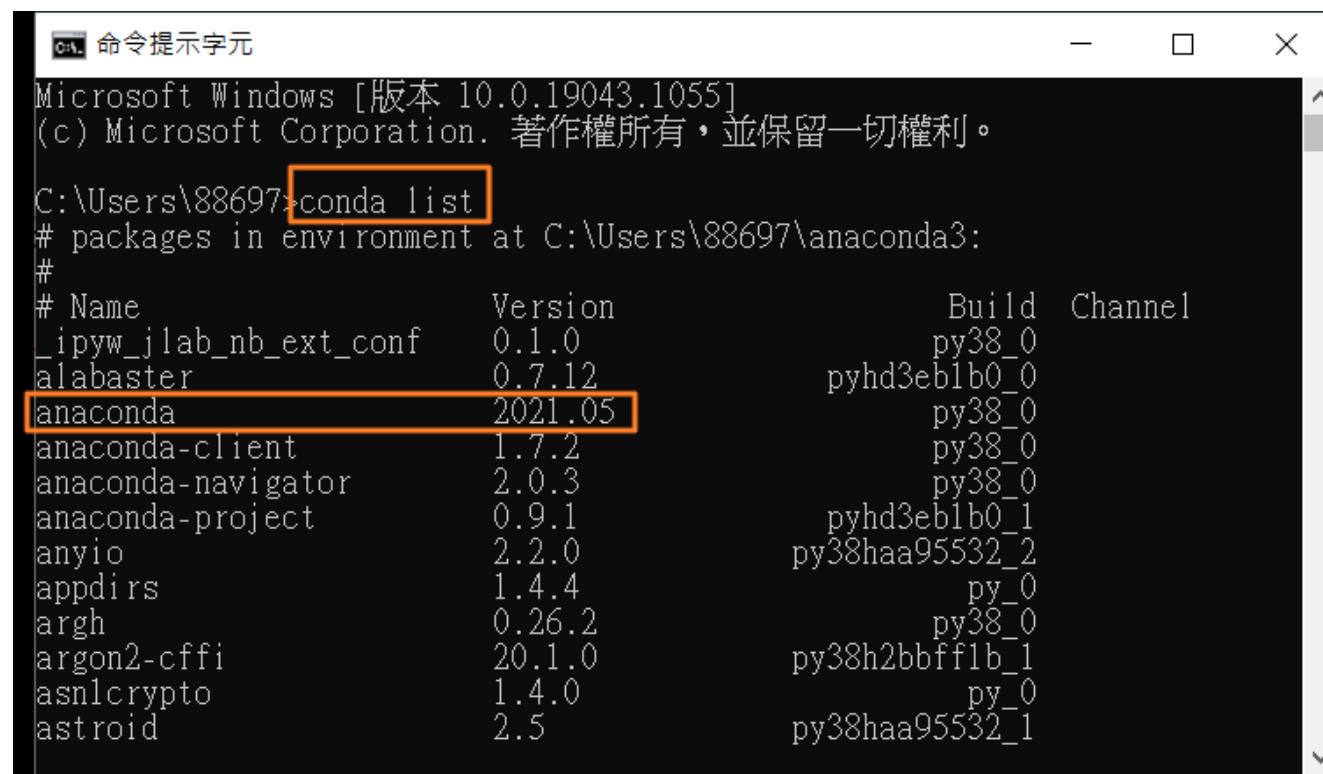
python -m Orange.canvas

Orange - Scatter Plot



Anaconda 模組管理

- 顯示已安裝模組
conda list



```
命令提示字元
Microsoft Windows [版本 10.0.19043.1055]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。

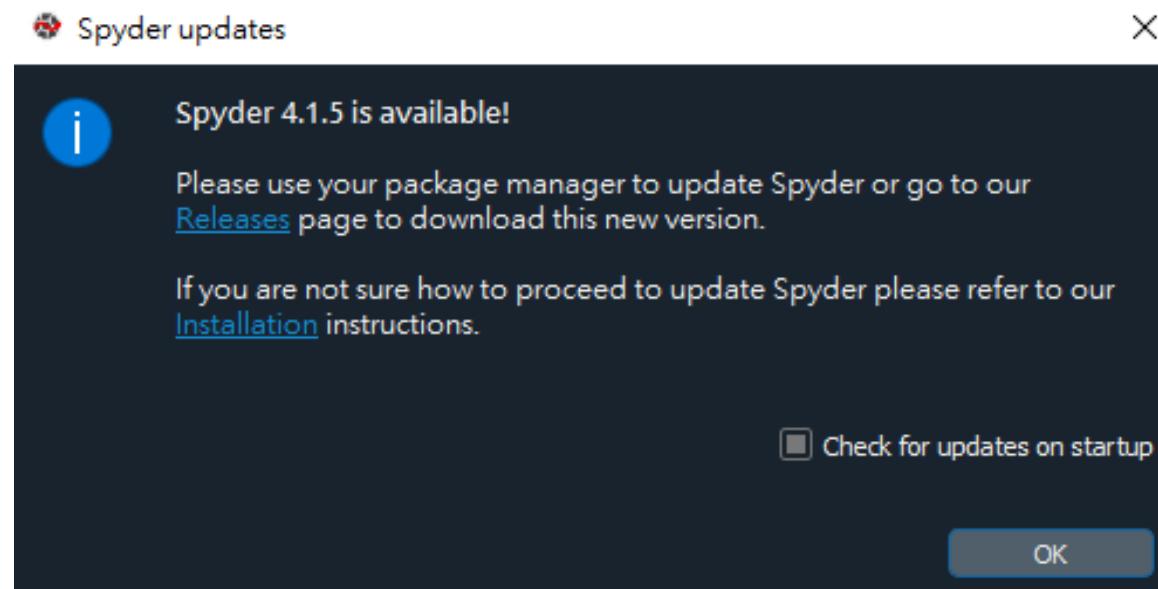
C:\Users\88697>conda list
# packages in environment at C:\Users\88697\anaconda3:
#
# Name           Version      Build Channel
_ipyw_jlab_nb_ext_conf    0.1.0       py38_0
alabaster          0.7.12      pyhd3eb1b0_0
anaconda           2021.05      py38_0
anaconda-client      1.7.2       py38_0
anaconda-navigator   2.0.3       py38_0
anaconda-project     0.9.1      pyhd3eb1b0_1
anyio                 2.2.0      py38haa95532_2
appdirs                1.4.4       py_0
argh                  0.26.2      py38_0
argon2-cffi         20.1.0      py38h2bbff1b_1
asn1crypto            1.4.0       py_0
astroid                 2.5      py38haa95532_1
```

Anaconda 模組管理(續)

- 尋找官網套件
`conda search matplotlib`
- 安裝模組
`conda install 模組名稱`
- 更新模組
`conda update 模組名稱`

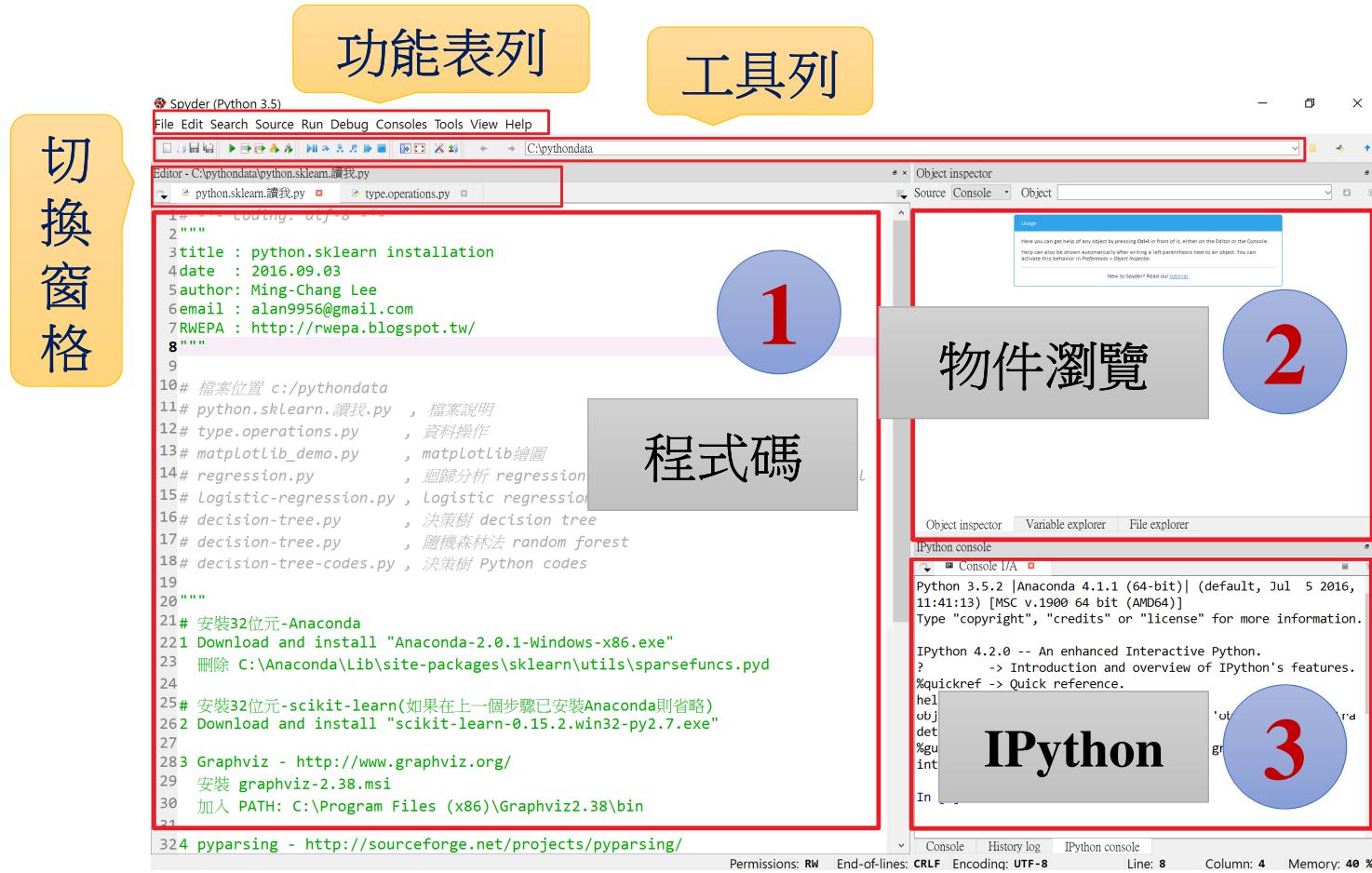
Spyder 軟體簡介

Spyder 更新

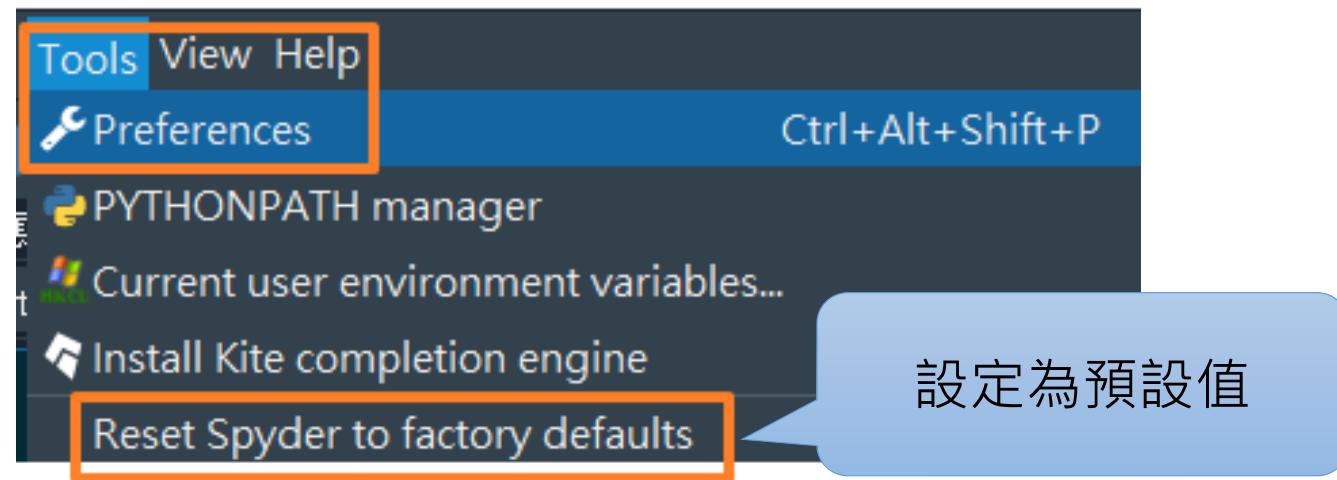


- Anaconda 整體更新
 - Spyder 更新
- `conda update anaconda`
 `conda update spyder`

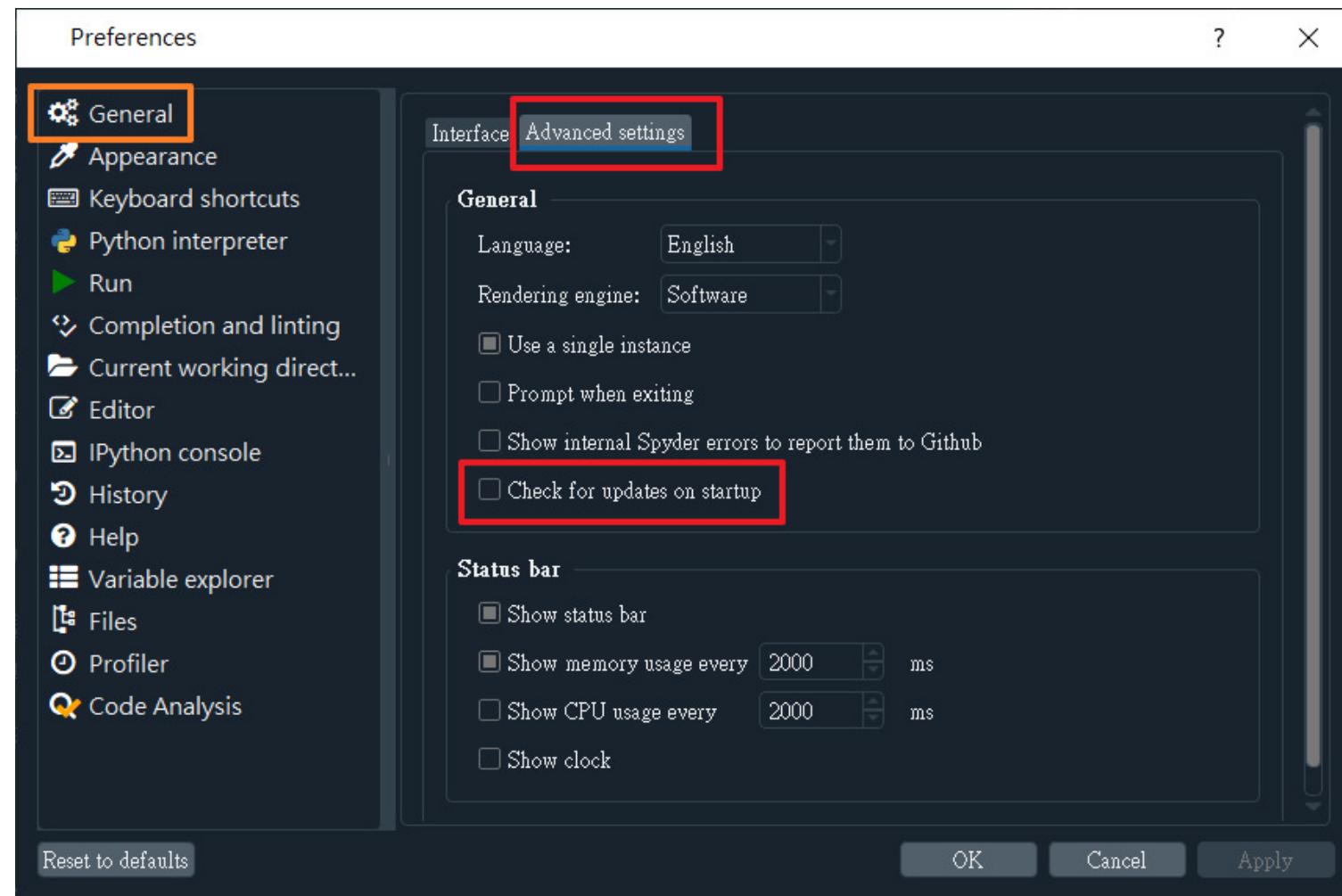
Spyder 畫面



喜好設定 Tools\Preferences



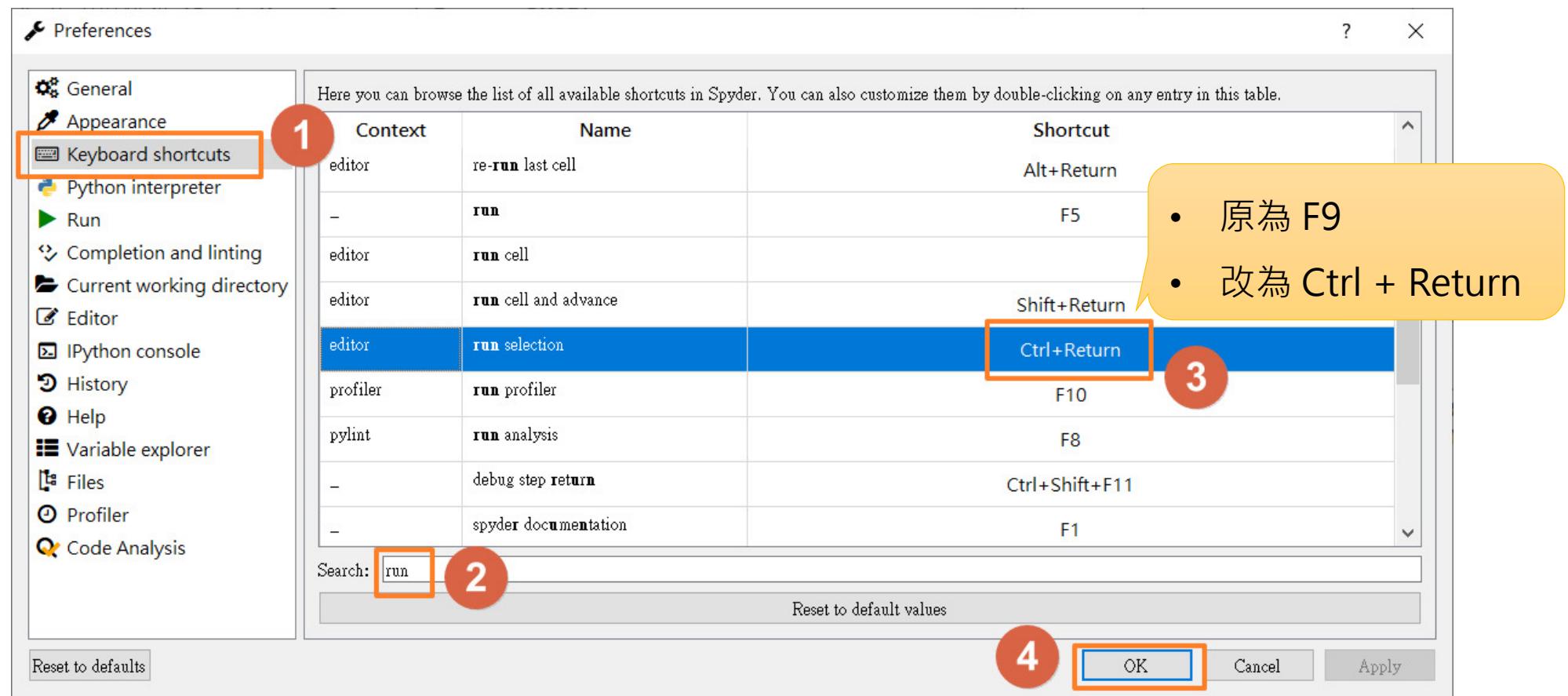
Preferences – General – Advanced setting



Preferences – Appearance



Preferences – Keyboard shortcuts



Spyder 好用的快速鍵

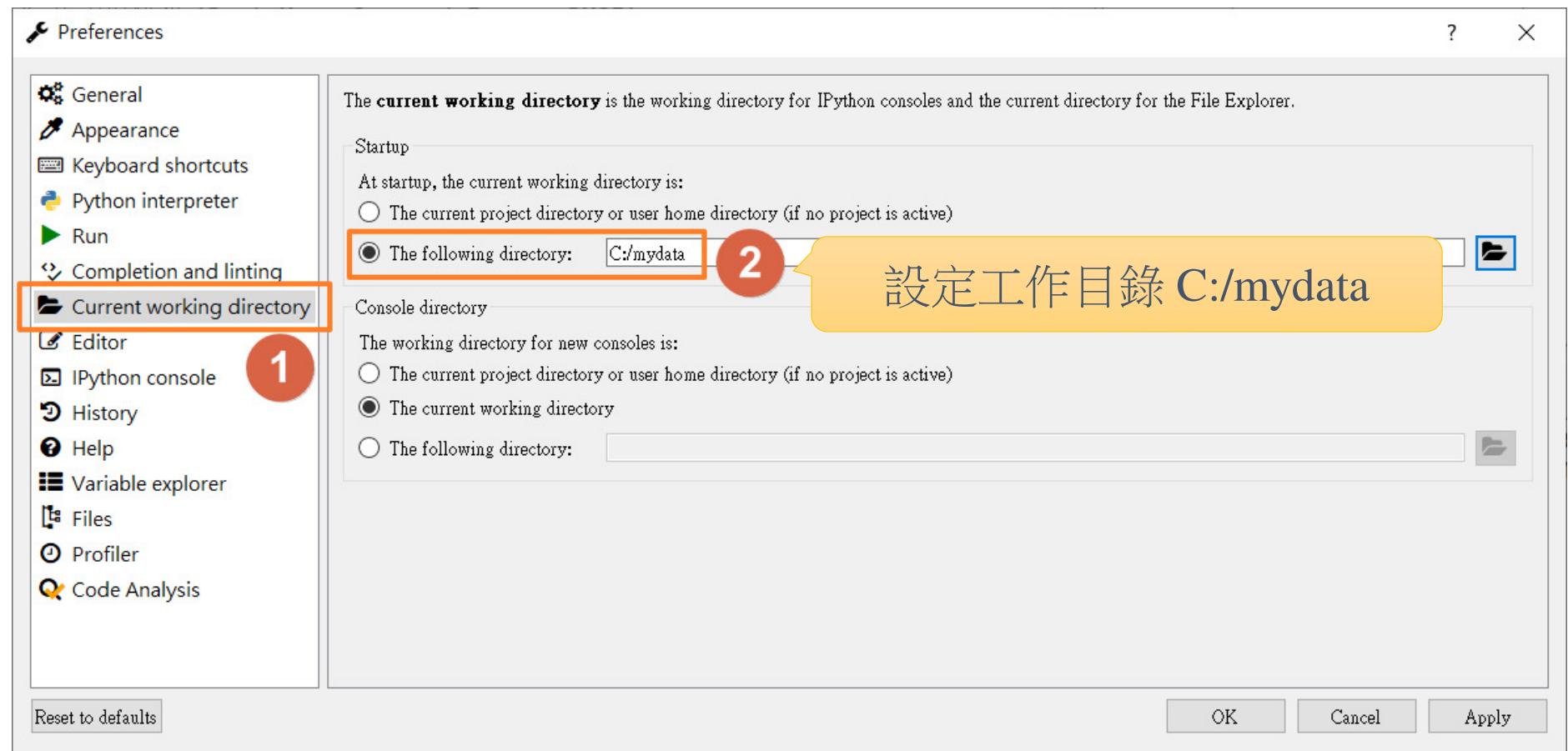
- 執行選取程式碼 **Ctrl + Enter (原為 F9)**
- 註解切換 **Ctrl + 1**
- 尋找 **Ctrl + F**
- 取代 **Ctrl + R**
- 切換至編輯視窗 **Ctrl + Shift + E**
- 切換至 Ipython Console **Ctrl + Shift + I**
- 檔案切換 **Ctrl + P**
- 重新啟動 Ipython Console **Ctrl + .**
- 檢視放大 **Ctrl + "+"**
- 檢視縮小 **Ctrl + "-"**
- 重新啟動 Spyder **Alt + Shift + R**

Ipython Console 視窗

- 清空: **Ctrl + L**
- 清空: **%clear**

 Split vertically	(上下垂直分割)	Ctrl+{
 Split horizontally	(左右水平分割)	Ctrl+_
 Close this panel		Alt+Shift+W

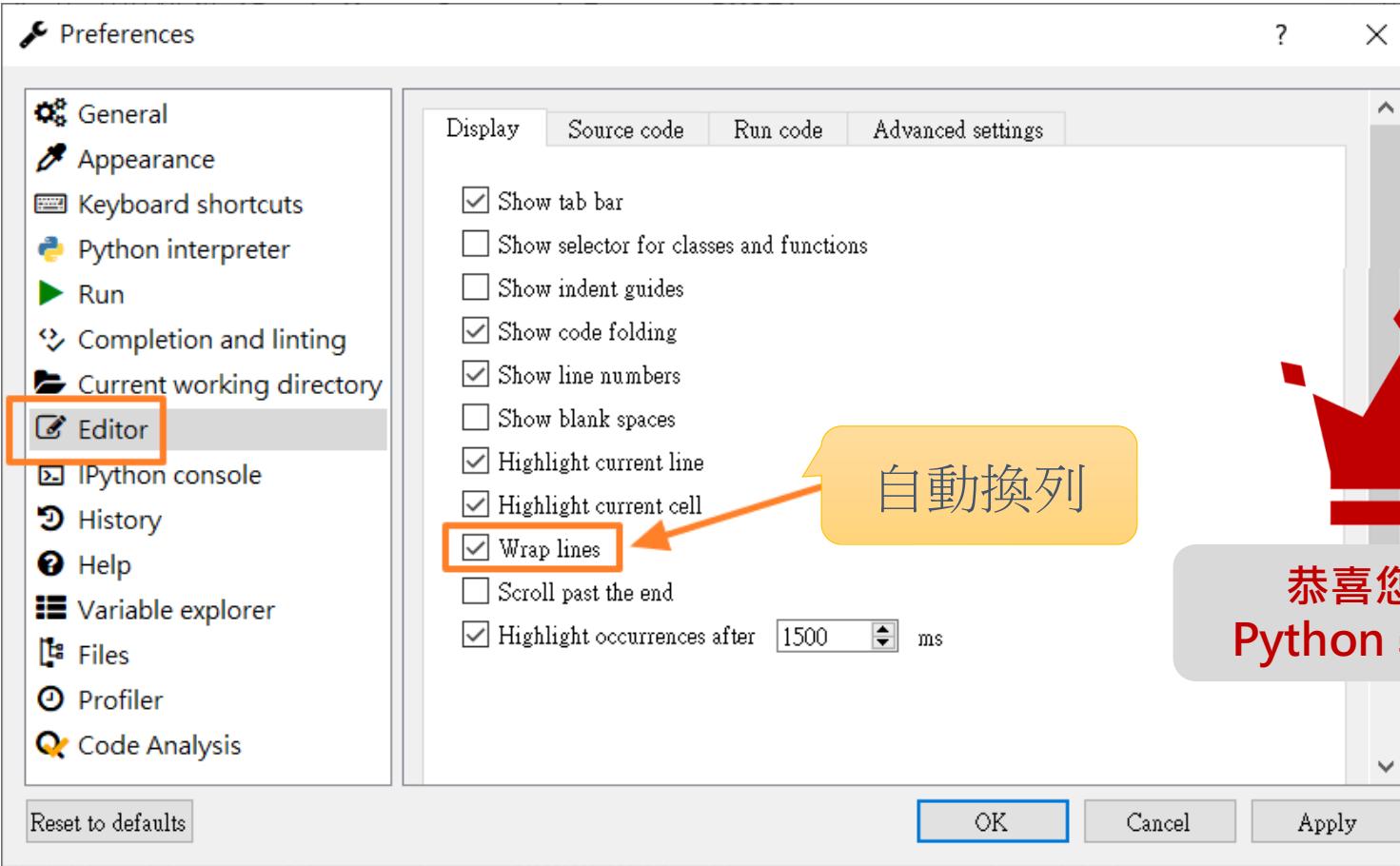
Preferences – Current working directory



Preferences – Editor – Wrap lines



實作練習3



自動換列

恭喜您, 開啟人生
Python 學習之旅 ^_^

Preferences

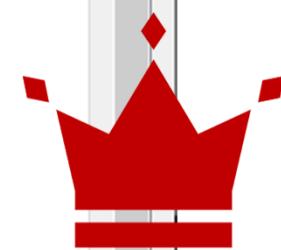
General Appearance Keyboard shortcuts Python interpreter Run Completion and linting Current working directory Editor IPython console History Help Variable explorer Files Profiler Code Analysis

Display Source code Run code Advanced settings

Show tab bar
Show selector for classes and functions
Show indent guides
Show code folding
Show line numbers
Show blank spaces
Highlight current line
Highlight current cell
Wrap lines
Scroll past the end
Highlight occurrences after 1500 ms

OK Cancel Apply

Reset to defaults



Python 執行-命令提示列

- 建立 C:\mydata\helloworld.py

```
C:\mydata\helloworld.py
helloworld.py x
1 # -*- coding: utf-8 -*-
"""
2
3 Created on Tue Jun 22 23:06:49 2021
4
5 @author: rwepa
"""
6
7
8 print("大數據Python應用")
```

```
命令提示字元
C:\mydata>python --version
Python 3.8.8

C:\mydata>dir
磁碟區 C 中的磁碟是 WIN10
磁碟區序號: E428-7C96

C:\mydata 的目錄

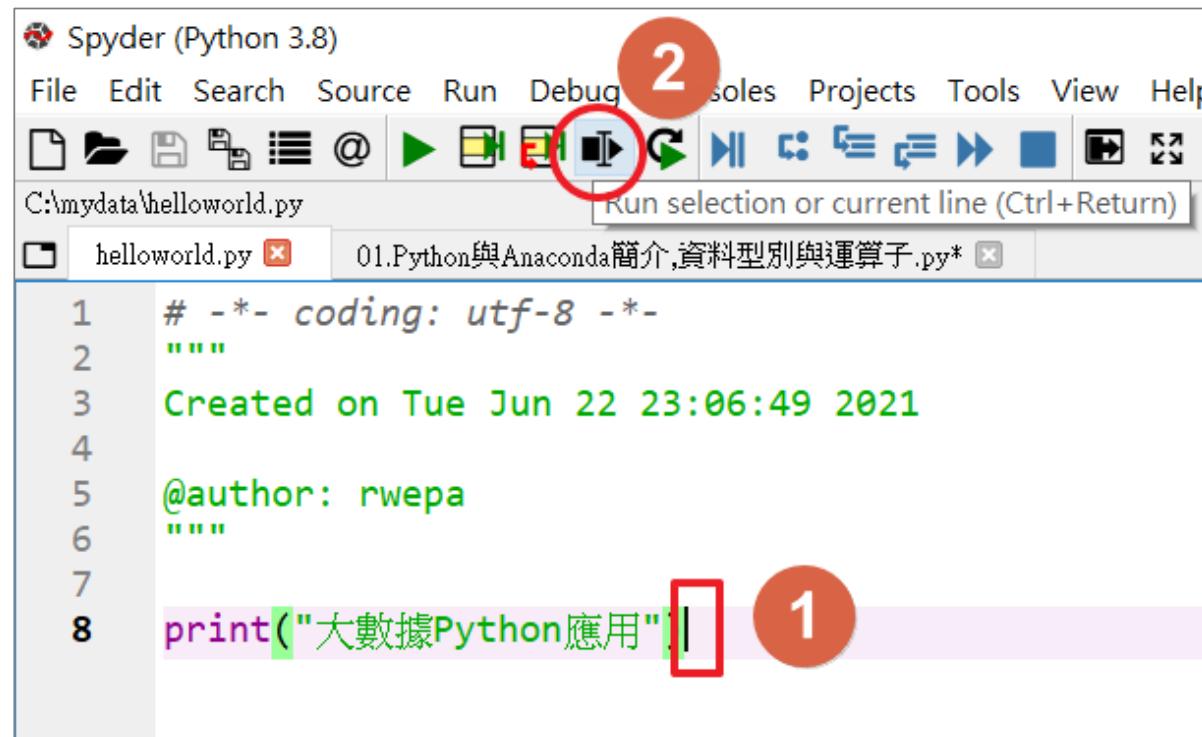
2021/06/22 下午 11:10    <DIR> .
2021/06/22 下午 11:10    <DIR> ..
2021/06/22 下午 11:08          122 helloworld.py
                                1 個檔案          122 位元組
                                2 個目錄   62,354,751,488 位元組可用

C:\mydata>python helloworld.py
大數據Python應用

C:\mydata>
```

Python 執行-使用 Spyder

- 選取資料列
- 按 [Run selection or current line]



變數

合法的變數名稱

- 必須以字母或下底線字符開頭
- 不能以數字開頭
- 只能包含字母、數字和下底線（A-z、0-9 和 _）
- 區分大小寫
 - 例: customer、Customer 和 CUSTOMER 是不同的變數
- 雙下底線開頭並結尾的名稱已經由Python保留, 例: __init__

合法變數 vs. 不合法變數

合法變數

```
大數據 = 1 # 中文亦可, 建議不要使用  
CustomerSaleReport = 1  
_CustomerSaleReport = 1  
Customer_Sale_Report = 1  
customer_sale_report = 1
```

- 命名為中文, OK?
- 命名為Python保留字, OK?

不合法變數

```
$CustomerSaleReport = 1 # SyntaxError: invalid syntax  
2020_sale = 100 # SyntaxError: invalid decimal literal  
break = 123 # SyntaxError: invalid syntax
```

內建保留字

```
dir(__builtins__)  
len(dir(__builtins__)) # 159
```

指派多個變數

In [1]:

```
...: x, y, z = "台北", "台中", "高雄"
```

```
...: print(x,y,z)
```

台北 台中 高雄

In [2]: type(x) # str

Out[2]: str

In [3]: address = ["台北", "台中", "高雄"]

```
...: x, y, z = address
```

In [4]: print(x)

台北

In [5]: print(y)

台中

In [6]: print(z)

高雄

Python Style Rules

- <https://google.github.io/styleguide/pyguide.html>



styleguide

Google Python Style Guide

▼ Table of Contents

- 1 Background
- 2 Python Language Rules
 - 2.1 Lint
 - 2.2 Imports
 - 2.3 Packages
 - 2.4 Exceptions
 - 2.5 Global variables
 - 2.6 Nested/Local/Inner Classes and Functions
 - 2.7 Comprehensions & Generator Expressions
 - 2.8 Default Iterators and Operators
 - 2.9 Generators
 - 2.10 Lambda Functions
 - 2.11 Conditional Expressions
 - 2.12 Default Argument Values
 - 2.13 Properties
 - 2.14 True/False Evaluations
 - 2.16 Lexical Scoping
 - 2.17 Function and Method Decorators
 - 2.18 Threading
 - 2.19 Power Features
 - 2.20 Modern Python: Python 3 and from `_future_` imports
 - 2.21 Type Annotated Code
- 3 Python Style Rules
 - 3.1 Semicolons

Python Style Rules (續)

3 Python Style Rules

3.1 Semicolons

Do not terminate your lines with semicolons, and do not use semicolons to put two statements on the same line.

3.2 Line length

Maximum line length is *80 characters*.

Explicit exceptions to the 80 character limit:

Python 註解

- 使用一個 # → 用於1行註解
- 使用二個 """ → 用於超過1行註解或函數之說明文件

```
1      """
2  file    : 01.Python與Anaconda簡介,資料型別與運算子.py
3  author   : Ming-Chang Lee
4  email    : alan9956@gmail.com
5  RWEPA    : http://rwepa.blogspot.tw/
6  GitHub    : https://github.com/rwepa
7  Encoding : UTF-8
8      """
```

內縮4個空白鍵之語法

```
with open('sampleinput1.txt', 'r') as f:  
    doc = [x.strip() for x in f.readlines()] # 使用 strip 刪除換行符號  
    docname = []  
    corpus = []  
    for index in range(len(doc)):  
        if index == 0:
```

注意：with 結尾加冒號

內縮4個空白鍵

資料型別

資料型別

- 數值型別

- 整數 int
- 長整數 long
- 浮點數 float
- 複數 complex

- 布林值 bool

- True
- False

- 空值 None → 類似 NULL

- 字串 (String) → 參考<03.字串與正規表示式,判斷式與函數應用>

參考: <https://docs.python.org/3/library/stdtypes.html>

廣義 Data Types

- Text Type: `str`
- Numeric Types: `int, float, complex`
- Boolean Type: `bool`
- Binary Types: `bytes, bytearray, memoryview`

- Sequence Types: `list, tuple, range`
- Set Types: `set, frozenset`
- Mapping Type: `dict`

資料型別 `type(x)`

參考: <https://www.w3schools.com/python/>

資料型別 – 範例

```
# 整數 int
```

```
x1 = 1
```

```
type(x1)
```

```
# 浮點數 float
```

```
x2 = 1.234
```

```
type(x2)
```

```
# 複數 complex
```

```
x3 = 1+2j
```

```
type(x3)
```

```
# 布林值 (Boolean)
```

```
x4 = True
```

```
type(x4)
```

```
x4 + 10
```

- 轉換為整數 int
- 轉換為浮點數 float
- 轉換為複數 complex

None值

```
In [1]: import numpy as np  
...: None == False  
Out[1]: False
```

```
In [2]: None == 0  
Out[2]: False
```

```
In [3]: False == 0  
Out[3]: True
```

```
In [4]: True == 1  
Out[4]: True
```

```
In [5]: None == np.nan  
Out[5]: False
```

```
In [6]: None == None  
Out[6]: True
```

整數亂數

```
In [1]: import random  
....: random.seed(168)  
....: myrandom = random.randrange(1, 100)  
....: myrandom  
Out[1]: 96
```

亂數種子 random.seed()

```
In [2]: random.seed(168)  
....: myrandom = random.randrange(1, 100)  
....: myrandom  
Out[2]: 96
```

運算子

運算子

運算子	功能
**	次方
*	乘法
/	除法
//	整除
%	餘數
+	加法
-	減法
	或運算子 OR
^	互斥運算子 XOR
&	且運算子 AND
<<	左移運算子
>>	右移運算子

- Excel: `=2^3`
- R: `2^3`
- Python: `2**3`
- SQL: `SELECT POWER(2, 3)`

運算子 – 範例

```
In [1]:  
....: 3 + 5  
Out[1]: 8
```

```
In [2]: 3 + (5 * 4)  
Out[2]: 23
```

```
In [3]: 3 ** 2  
Out[3]: 9
```

```
In [4]: "Hello" + "World"  
Out[4]: 'HelloWorld'
```

```
In [5]: 1 + 1.234  
Out[5]: 2.234
```

```
In [6]: 7 / 2  
Out[6]: 3.5
```

```
In [7]: 7 // 2  
Out[7]: 3
```

```
In [8]: 7 % 2  
Out[8]: 1
```

```
In [9]: 2 ** 10  
Out[9]: 1024
```

```
In [10]: 1.234e3 - 1000  
Out[10]: 234.0
```

```
In [11]: x5 = 1 == 2
```

```
In [12]: x5  
Out[12]: False
```

```
In [13]: x5 + 10  
Out[13]: 10
```

位移運算子

```
# 位移運算子: << 向左位移  
# 位移運算子: >> 向右位移  
a = 4 << 3 # 0100 --> 0100000, 32 16 8 4 2 1  
print(a)  
  
b = a * 4.5  
print(b)  
  
c = (a+b)/2.5
```

指派運算子

指派運算子	範例	結果
=	x = 9	x = 9
+=	x += 2	x = x + 2
-=	x -= 3	x = x - 3
*=	x *= 4	x = x * 4
/=	x /= 5	x = x / 5
%=	x %= 6	x = x % 6
//=	x //= 7	x = x // 7
**=	x **= 8	x = x ** 8
&=	x &= 3	x = x & 3
=	x = 3	x = x 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

```
In [1]:
....:
....: x = 9
....: x+=2
....: print(x)
```

11

02.Python資料物件,使用NumPy模組與 reshape應用

資料物件

資料物件

- 容器型別 (Container type)

- tuple → () 表示, 資料不可修改 (序列/元組)
- list → [] 表示, 資料可以修改 (串列/清單)
- set → { } 表示, 執行集合運算 (集合)
- dict → {'key1': value1, ... } 表示, 可藉由key查value (字典)

Tuple

Tuple

- 建立序列

```
f = (2,3,4,5)          # A tuple of integers  
g = ()                # An empty tuple  
h = (2, [3,4], (10,11,12)) # A tuple containing mixed objects
```

- Tuples操作

```
x = f[1]              # Element access. x = 3  
y = f[1:3]             # Slices (切片) y = (3,4)  
z = h[1][1]             # Nesting. z = 4
```

取出 y的第1個索引到 4-1=3

- 特色

- 與list類似，最大的不同tuple是一種唯讀且不可變更的資料結構
- 不可取代tuple中的任意一個元素，因為它是唯讀不可變更的
- Tuple 是具有 ordered 特性
- Python 的**索引(指標)從0開始**

基本型態 – Tuple

```
In [1]: xy = (2, 3)
...: xy
Out[1]: (2, 3)
```

```
In [2]: personal = ('Hannah', 14, 5*12+6)
...: personal
Out[2]: ('Hannah', 14, 66)
```

```
In [3]: singleton = ("hello",)
...: singleton
Out[3]: ('hello',)
```

```
In [4]: type(singleton) # tuple
Out[4]: tuple
```

```
In [5]: singleton1 = ("hello")
...: singleton1
Out[5]: 'hello'
```

```
In [6]: type(singleton1) # str
Out[6]: str
```

- 有加上「,」？
- 不加上「,」之資料型別為何？

基本型態 – Tuple (續)

```
In [1]: f = (2,3,4,5)
```

```
In [2]: f[0]  
Out[2]: 2
```

```
In [3]: f[-1]  
Out[3]: 5
```

```
In [4]: f[-2]  
Out[4]: 4
```

```
In [5]: f[len(f)-1]  
Out[5]: 5
```

```
In [6]: t=((1,2), (2,"Hi"), (3,"RWEPA"), 2+3j, 6E23)
```

```
In [7]: t[2]  
Out[7]: (3, 'RWEPA')
```

```
In [8]: t[:3]  
Out[8]: ((1, 2), (2, 'Hi'), (3, 'RWEPA'))
```

```
In [9]: t[3:]  
Out[9]: ((2+3j), 6e+23)
```

```
In [10]: t[-1]  
Out[10]: 6e+23
```

```
In [11]: t[-3:]  
Out[11]: ((3, 'RWEPA'), (2+3j), 6e+23)
```

索引 [-1] 表示倒數第1個元素

tuple 長度
len(t) # 5

Tuple 建構子

```
type(employeeGender) # tuple
```

```
# tuple 建構子，使用 tuple(( ... )) 或 tuple([ ... ])
employeeGender = tuple(("男", "女", "女"))
employeeGender
```

```
# tuple unpacking - 將元素指派至變數
fruits = ("apple", "banana", "cherry")
(green, yellow, red) = fruits
print(green)
print(yellow)
print(red)
```

```
type(green) # str
```

Tuple unpacking - 使用萬用字元*

開箱

In [1]:

```
....: fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
....: (green, yellow, *red) = fruits
....: print(green)
....: print(yellow)
....: print(red)
```

```
apple
banana
['cherry', 'strawberry', 'raspberry']
```

Tuple - loop 處理

```
# tuple - Loop 處理
fruits = ("apple", "banana", "cherry")
```

```
# 方法1. tuple - 取出元素, 使用for
for x in fruits:
    print(x)
```

```
# 方法2. tuple - 取出元素, 使用while
i = 0
while i < len(fruits):
    print(fruits[i])
    i = i + 1
```

```
# 方法3. tuple - 取出元素, 使用指標 range, Len
for i in range(len(fruits)):
    print(fruits[i])
```

- 三個方法結果皆相同,何者較快?

In [3]:
 for i in range(len(fruits)):
 print(fruits[i])

apple
banana
cherry

長度 len



Tuple - join 結合, 重複

In [1]: # tuple - join 結合

```
In [2]: tuple1 = ("台北", "台中", "高雄")
...: tuple2 = ("男", "女", "女")
...: tuple3 = tuple1 + tuple2      使用 + 號
...: print(tuple3)
('台北', '台中', '高雄', '男', '女', '女')
```

In [3]: # tuple - 重複

```
In [4]: tuple1*3      使用 * 號
Out[4]: ('台北', '台中', '高雄', '台北', '台中', '高雄', '台北', '台中', '高雄')
```

In [5]: 3*tuple1

```
Out[5]: ('台北', '台中', '高雄', '台北', '台中', '高雄', '台北', '台中', '高雄')
```

Tuple - count 次數統計

```
In [6]: # count 次數統計
```

```
In [7]: tuple = ("男", "女", "女", "男", "女")
```

```
In [8]: tuple.count("男") # 2
```

```
Out[8]: 2
```

```
In [9]: tuple.count("女") # 3
```

```
Out[9]: 3
```

TRY:
tuple.count("A")

序列 – 方法

方法	功能
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

List

基本型態 – 串列(List)

- 建立串列

```
In [1]: a = [2, 3, 4]          # 整數串列  
...: b = [2, 7, 3.5, "Hello"] # 混合資料串列  
...: c = []                   # 空串列  
...: d = [2, [a, b]]          # 巢狀串列
```

```
In [2]: a  
Out[2]: [2, 3, 4]
```

```
In [3]: b  
Out[3]: [2, 7, 3.5, 'Hello']
```

```
In [4]: c  
Out[4]: []
```

```
In [5]: d  
Out[5]: [2, [[2, 3, 4], [2, 7, 3.5, 'Hello']]]
```

1

2

串列操作

```
In [7]: a  
Out[7]: [2, 3, 4]
```

```
In [8]: a[1]      # 取得第2個元素  
Out[8]: 3
```

```
In [9]: a[-1]     # 取得最後一個元素  
Out[9]: 4
```

```
In [10]: b[1:3]    # 串列篩選  
Out[10]: [7, 3.5]
```

```
In [11]: d[1][0][2] # 巢狀串列操作  
Out[11]: 4
```

```
In [12]: b[0]      # 2  
Out[12]: 2
```

```
In [13]: b[0] = 42  # 修改元素值
```

```
In [14]: b[0]      # 42  
Out[14]: 42
```

```
a  
[2, 3, 4]
```

```
b  
[2, 7, 3.5, 'Hello']
```

```
d  
[2, [[2, 3, 4], [2, 7, 3.5, 'Hello']]]
```

b[1:3] 相當於 b[1], b[2]

串列 slice format

```
In [1]: t=[1, 2, (3,"Hi"), [4,"RWEPA"], 2+3j, 6E7]
```

```
In [2]: t
```

```
Out[2]: [1, 2, (3, 'Hi'), [4, 'RWEPA'], (2+3j), 60000000.0]  
        ① ② ③ ④ ⑤
```

```
In [3]: t[2]
```

```
Out[3]: (3, 'Hi')
```

```
In [4]: t[:3]
```

```
Out[4]: [1, 2, (3, 'Hi')]
```

[3:] 開始~指標3-1=2

```
In [5]: t[3:]
```

```
Out[5]: [[4, 'RWEPA'], (2+3j), 60000000.0]
```

[3:] 指標3 ~ 最後

```
In [6]: t[-1]
```

```
Out[6]: 60000000.0
```

```
In [7]: t[-3:]
```

```
Out[7]: [[4, 'RWEPA'], (2+3j), 60000000.0]
```

```
In [8]: # 串列長度
```

```
In [9]: len(t)
```

```
Out[9]: 6
```

串列建構子, 使用 `list((...))` 或 `list([...])`

```
In [1]: mylist1 = list(( "男", "女", "女" ))
```

```
In [2]: mylist1  
Out[2]: ['男', '女', '女']
```

```
In [3]: mylist2 = list(( "男", "女", "女" ))
```

```
In [4]: mylist2  
Out[4]: ['男', '女', '女']
```

```
In [5]: mylist1 == mylist2  
Out[5]: True
```

串列 unpacking - 將元素指派至變數

In [1]:

```
....: fruits = ["apple", "banana", "cherry"]
....: green, yellow, red = fruits
```

In [2]: print(green)

apple

In [3]: print(yellow)

banana

In [4]: print(red)

cherry

In [5]: type(green) # str

Out[5]: str

串列 unpacking - 使用萬用字元*

```
In [1]: fruits = ["apple", "banana", "cherry", "strawberry", "raspberry"]
```

```
In [2]: green, yellow, *red = fruits
```

```
In [3]: print(green)
```

```
apple
```

```
In [4]: print(yellow)
```

```
banana
```

```
In [5]: print(red)
```

```
['cherry', 'strawberry', 'raspberry']
```

```
In [6]: type(green) # str
```

```
Out[6]: str
```

串列 - loop 處理

```
# List - Loop 處理
mylist = [1, 2, 3, [4, 5], ["A", "B", "C"]]
# 練習 Loop 方法

# 方法1. List - 取出元素, 使用for
for x in mylist:
    print(x)

# 方法2. List - 取出元素, 使用while
i = 0
while i < len(mylist):
    print(mylist[i])
    i = i + 1

# 方法3. List - 取出元素, 使用指標 range, Len
for i in range(len(mylist)):
    print(mylist[i])

# 方法4. List - 取出元素, 使用串列包含法 (List Comprehension)
[print(x) for x in mylist]
```

```
In [2]: for x in mylist:
....:     print(x)
1
2
3
[4, 5]
['A', 'B', 'C']
```

串列包含法應用

```
In [1]: # for 資料篩選-包括字母 a
```

```
In [2]: codes = ["Python", "R", "SQL", "Julia", ".NET", "Java", "JavaScript"]
...: newlist = []
...: for x in codes:
...:     if "a" in x:
...:         newlist.append(x)
...: print(newlist)
['Julia', 'Java', 'JavaScript']
```

串列包含法應用1

- 串列包含法亦可用於序列, 集合, 字典等可反覆運算物件
(可迭代物件, iterable object)

```
In [1]: # 串列包含法應用1
```

```
In [2]: # 亦可用於序列, 集合, 字典等可反覆運算物件(可迭代物件, iterable object)
```

```
In [3]: codes = ["Python", "R", "SQL", "Julia", ".NET", "Java", "JavaScript"]
....: newlist = [x for x in codes if "a" in x]
....: print(newlist)
['Julia', 'Java', 'JavaScript']
```

串列包含法應用2

```
In [4]: # 串列包含法應用2
```

```
In [5]: newlist = [x.upper() for x in codes]
...: print(newlist)
['PYTHON', 'R', 'SQL', 'JULIA', '.NET', 'JAVA', 'JAVASCRIPT']
```

```
In [6]: codes.upper() # AttributeError: 'List' object has no attribute 'upper'
Traceback (most recent call last):
```

```
File "<ipython-input-6-19ae796b0f51>", line 1, in <module>
  codes.upper() # AttributeError: 'list' object has no attribute 'upper'
```

```
AttributeError: 'list' object has no attribute 'upper'
```

串列包含法應用3

In [7]: # 串列包含法應用3

```
In [8]: newlist = ['RWEPA' for x in codes]
...: print(newlist)
['RWEPA', 'RWEPA', 'RWEPA', 'RWEPA', 'RWEPA', 'RWEPA', 'RWEPA']
```

串列-結合, 重複

In [14]: # 串列 join 結合

In [15]: e = a + b # Join two lists

串列-結合: +

In [16]: e

Out[16]: [2, 3, 4, 2, 7, 3.5, 'Hello']

In [17]: # 串列 repeat 重複

In [18]: f1 = a*3 # repeat lists

串列-重複: *

In [19]: f1

Out[19]: [2, 3, 4, 2, 3, 4, 2, 3, 4]

In [20]: f2 = 3*a

In [21]: f2

Out[21]: [2, 3, 4, 2, 3, 4, 2, 3, 4]

串列 – 排序 sort

In [1]: # 串列排序-預設為遞增排序, 英文字母先大寫, 再小寫

```
In [2]: codes = ["python", "R", "SQL", "Julia", ".NET", "java", "JavaScript"]
...: codes.sort()
...: print(codes)
['.NET', 'JavaScript', 'Julia', 'R', 'SQL', 'java', 'python']
```

In [3]: # 串列排序-先全部小寫, 再排序

```
In [4]: codes = ["python", "R", "SQL", "Julia", ".NET", "java", "JavaScript"]
In [5]: codes.sort(key = str.lower)
In [6]: print(codes)
['.NET', 'java', 'JavaScript', 'Julia', 'python', 'R', 'SQL']
```

串列-遞減排序

```
In [1]: # 串列排序-遞減排序
```

```
In [2]: codes = ["python", "R", "SQL", "Julia", ".NET", "java", "JavaScript"]
```

```
In [3]: codes.sort(reverse =True)
```

```
In [4]: print(codes)  
['python', 'java', 'SQL', 'R', 'Julia', 'JavaScript', '.NET']
```

```
In [5]: # 串列反序
```

```
In [6]: codes = ["python", "R", "SQL", "Julia", ".NET", "java", "JavaScript"]
```

```
In [7]: codes.reverse()
```

```
In [8]: print(codes)  
['JavaScript', 'java', '.NET', 'Julia', 'SQL', 'R', 'python']
```

串列複製 – 使用等號

In [1]: # 串列複製，等號會建立參考物件

In [2]: a = [1, 2, 3]

In [3]: a

Out[3]: [1, 2, 3]

In [4]: b = a

In [5]: b[0] = 999 # 修改b，亦會修改a

In [6]: b

Out[6]: [999, 2, 3]

In [7]: a # a已經更新

Out[7]: [999, 2, 3]

串列複製 – 使用 copy

```
In [8]: # 串列複製- 使用 copy
```

```
In [9]: a = [1, 2, 3]
```

```
In [10]: b = a.copy()
```

```
In [11]: b  
Out[11]: [1, 2, 3]
```

```
In [12]: b[0] = 999
```

```
In [13]: b  
Out[13]: [999, 2, 3]
```

```
In [14]: a # a保持不變  
Out[14]: [1, 2, 3]
```

串列複製-使用 list

```
In [1]: # 串列複製-使用 list
```

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

```
In [3]: c = list(a)
```

```
In [4]: c
```

```
Out[4]: [1, 2, 3]
```

```
In [5]: c[0] = 123
```

```
In [6]: c
```

```
Out[6]: [123, 2, 3]
```

```
In [7]: a # a保持不變
```

```
Out[7]: [1, 2, 3]
```

串列附加 append, 延伸 extend

```
In [1]: # 附加元素 append
```

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

```
In [3]: a.append(['BigData', 'SQL']) # 新增1個元素
```

```
In [4]: a
```

```
Out[4]: [1, 2, 3, ['BigData', 'SQL']]
```

附加為1個值

```
In [5]: # 延伸元素 extend
```

```
In [6]: a.extend(['Python', 'R', "Julia"]) # 新增一個串列
```

```
In [7]: a
```

```
Out[7]: [1, 2, 3, ['BigData', 'SQL'], 'Python', 'R', "Julia"]
```

延伸為3個值

串列延伸 – tuple, list, set, dict

```
In [8]: # 延伸元素 extend - 加入tuple, list, set, dict
```

```
In [9]: a = [1, 2, 3]
```

```
In [10]: a.extend(['4', '5', 'RWEPA']) # 延伸一個序列
```

```
In [11]: a
```

```
Out[11]: [1, 2, 3, '4', '5', 'RWEPA']
```

```
In [12]: a.extend({'8', '8', '10'}) # 延伸一個集合
```

```
In [13]: a
```

```
Out[13]: [1, 2, 3, '4', '5', 'RWEPA', '8', '10']
```

```
In [14]: a.extend({'a': 'R', 'b': 'Python'}) # 延伸一個字典-ONLY KEY, NO VALUE
```

```
In [15]: a
```

```
Out[15]: [1, 2, 3, '4', '5', 'RWEPA', '8', '10', 'a', 'b']
```

串列 – insert

```
In [1]: # 插入元素
```

```
In [2]: a = list(range(5))
```

```
In [3]: a
```

```
Out[3]: [0, 1, 2, 3, 4]
```

```
In [4]: a.insert(2, 999) # 在指標為2的位置, 插入新元素
```

1 2

```
In [5]: a
```

```
Out[5]: [0, 1, 999, 2, 3, 4]
```

串列 – remove, pop, del

```
In [6]: # 刪除指定元素
```

```
In [7]: a.remove(999)
```

```
In [8]: a
```

```
Out[8]: [0, 1, 2, 3, 4]
```

```
In [9]: # 刪除指定指標元素
```

```
In [10]: a.pop(1)
```

```
Out[10]: 1
```

```
In [11]: a
```

```
Out[11]: [0, 2, 3, 4]
```

```
In [12]: # 刪除指定指標元素
```

```
In [13]: del a[1]
```

```
In [14]: a
```

```
Out[14]: [0, 3, 4]
```

串列 – pop

```
In [14]: a  
Out[14]: [0, 3, 4]
```

```
In [15]: # 刪除第一個元素
```

```
In [16]: a.pop(0)  
Out[16]: 0
```

```
In [17]: a  
Out[17]: [3, 4]
```

```
In [18]: # 刪除最後一個元素
```

```
In [19]: a.pop()  
Out[19]: 4
```

```
In [20]: a  
Out[20]: [3]
```

串列 – clear, del

```
# 清空物件元素，物件仍存在記憶體
a.clear()
a

# 刪除物件，物件不存在記憶體
del a
print(a) # NameError: name 'a' is not defined
```

串列 - zip 應用

```
In [1]: # zip 應用
```

```
In [2]: a = ("x1", "x2", "x3")
```

```
In [3]: b = ("y1", "y2", "y3")
```

```
In [4]: c = (1, 2, 3)
```

```
In [5]: x = zip(a, b, c)
```

```
In [6]: x
```

```
Out[6]: <zip at 0x19620369740>
```

```
In [7]: list(x)
```

```
Out[7]: [('x1', 'y1', 1), ('x2', 'y2', 2), ('x3', 'y3', 3)]
```



串列 – 方法

實作練習4

如何顯示list不以 __ 開始串列方法的總個數 11?

In [1]: # 顯示方法

Out[1]: 11

```
In [2]: 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']
```

串列 – 方法

方法	功能
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Set

基本型態 – 集合(Set)

- 集合與字典相似, 但集合沒有key,只有值
- 集合內容不可以修改
- 集合是 **unordered**
- 集合是 **unindexed**
- 集合會忽略重複的值

集合 - set

```
In [1]: a = set() # 空集合
```

```
In [2]: type(a)  
Out[2]: set
```

```
In [3]: b = {"台北市", "新北市", "桃園市", "台中市", "台北市", "新北市", "高雄市"}
```

```
In [4]: b # {'台中市', '台北市', '新北市', '桃園市', '高雄市'}  
Out[4]: {'台中市', '台北市', '新北市', '桃園市', '高雄市'}
```

```
In [5]: # b[0] = 1 # TypeError: 'set' object does not support item assignment
```

```
In [6]: # b[0]      # TypeError: 'set' object is not subscriptable
```

```
In [7]: len(b)  
Out[7]: 5
```

集合 - set

```
myset = {"台北市", "新北市", "桃園市", "台中市", "高雄市"}
```

使用 myset 練習集合 -
loop 處理

Python demo

集合-新增 add, update

In [1]: # 集合新增元素 add, 因為集合是unordered, 不一定新增在最後一個

In [2]: myset = {"台北市", "新北市", "桃園市", "台中市", "高雄市"}

In [3]: myset.add("臺南市")

In [4]: myset

Out[4]: {'台中市', '台北市', '台南市', '新北市', '桃園市', '高雄市'}

In [5]: # 集合新增集合

In [6]: myset.update({"澎湖", "金門"})

In [7]: myset

Out[7]: {'台中市', '台北市', '台南市', '新北市', '桃園市', '澎湖', '金門', '高雄市'}

集合 – 刪除 remove, clear, del

```
In [8]: # 刪除指定元素
```

```
In [9]: myset.remove("澎湖")
```

```
In [10]: myset
```

```
Out[10]: {'台中市', '台北市', '台南市', '新北市', '桃園市', '金門', '高雄市'}
```

```
In [11]: # 清空物件元素, 物件仍存在記憶體
```

```
In [12]: myset.clear()
```

```
In [13]: myset
```

```
Out[13]: set()
```

```
In [14]: # 刪除物件, 物件不存在記憶體
```

```
In [15]: del myset
```

```
...: myset # NameError: name 'myset' is not defined
Traceback (most recent call last):
```

```
File "<ipython-input-15-01ac0b906dce>", line 2, in <module>
    myset # NameError: name 'myset' is not defined
```

```
NameError: name 'myset' is not defined
```

集合運算

- $x \& y$
- intersection
- $x | y$
- union
- $x ^ y$
- $x - y$
- difference

```
In [1]: # 集合運算
```

```
In [2]: x = {1,2,3,4,5}  
...: y = {1,3,5,7}
```

```
In [3]: x & y # {1, 3, 5} # 交集  
Out[3]: {1, 3, 5}
```

```
In [4]: x.intersection(y) # 交集  
Out[4]: {1, 3, 5}
```

```
In [5]: x | y # {1, 2, 3, 4, 5, 7} # 聯集  
Out[5]: {1, 2, 3, 4, 5, 7}
```

```
In [6]: x.union(y) # 聯集  
Out[6]: {1, 2, 3, 4, 5, 7}
```

```
In [7]: x ^ y # {2, 4, 7} # XOR 互斥  
Out[7]: {2, 4, 7}
```

```
In [8]: x - y # 差集  
Out[8]: {2, 4}
```

```
In [9]: x.difference(y) # 差集  
Out[9]: {2, 4}
```

集合 - 方法

方法	功能
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item, 如果找不到元素, 不會有ERROR
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element, 如果找不到, 會有ERROR
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Dict

基本型態 – 字典(Dict)

- 字典與集合相似, 但字典有key, 有值
- 字典內容可以修改
- 字典是 ordered (Python 3.6/早期版本 字典是*unordered*)
- 字典是 indexed
- 字典不可以有重複的key

字典 - 宣告

```
In [1]: mydict = {  
...:     "language": "Python",  
...:     "designer": "Guido van Rossum",  
...:     "year": 1991  
...: }  
  
In [2]: print(mydict)  
{'language': 'Python', 'designer': 'Guido van Rossum', 'year': 1991}  
  
In [3]: type(mydict) # dict  
Out[3]: dict
```

字典 - 重複 key, 只保留1個

```
In [4]: mydict1 = {  
...:     "language": "Python",  
...:     "designer": "Guido van Rossum",  
...:     "year": 1991,  
...:     "year": 2021  
...: }  
...:  
...: print(mydict1)  
{'language': 'Python', 'designer': 'Guido van Rossum', 'year': 2021}
```

保留重複新 key - value

字典存取元素 – keys, values

In [1]:

```
....: b = {  
....:     "uid": 168,  
....:     "login": "marvelous",  
....:     "name" : 'Alan Lee'  
....: }  
....: b
```

Out[1]: {'uid': 168, 'login': 'marvelous', 'name': 'Alan Lee'}

In [2]: # dict 取得所有 keys

```
In [3]: mykeys = b.keys()  
....: print(mykeys)  
dict_keys(['uid', 'login', 'name'])
```

In [4]: # dict 取得所有 values

```
In [5]: myvalues = b.values()  
....: print(myvalues)  
dict_values([168, 'marvelous', 'Alan Lee'])
```

字典取得key的值

```
In [6]: u = b["uid"] # 168  
....: print(u)  
168
```

```
In [7]: # dict 更新值
```

```
In [8]: b.update({"uid": 123})  
....: print(b)  
{'uid': 123, 'login': 'marvelous', 'name': 'Alan Lee'}
```

```
In [9]: # dict 新增元素
```

```
In [10]: b["shell"] = "/bin/sh"  
....: print(b)  
{'uid': 123, 'login': 'marvelous', 'name': 'Alan Lee', 'shell': '/bin/sh'}
```

字典- 刪除

```
# dict 刪除元素 - pop  
b.pop("shell")  
print(b)
```

```
# dict 刪除元素 - del  
del b["login"]  
print(b)
```

```
# dict 清空整個物件 - clear  
b.clear()  
b
```

```
# dict 刪除整個物件 - del  
del b  
b
```

字典 - items 物件

- items 物件: 回傳 [(key1,value1), (key2,value2,...)]
- 回傳 list[(序列1), (序列2), ...]

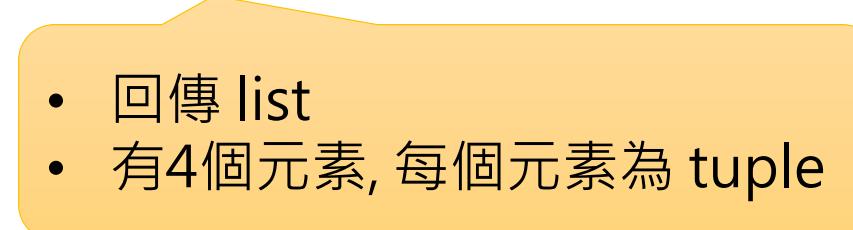
```
In [11]: b
```

```
Out[11]: {'uid': 123, 'login': 'marvelous', 'name': 'Alan Lee', 'shell': '/bin/sh'}
```

```
In [12]: x = b.items()
```

```
In [13]: print(x)
```

```
dict_items([('uid', 123), ('login', 'marvelous'), ('name', 'Alan Lee'), ('shell', '/bin/sh')])
```

- 
- 回傳 list
 - 有4個元素, 每個元素為 tuple

字典 – 檢查 key 是否存在

- 早期版本使用 has_key()

```
In [4]: if b.has_key("uid"):  
...:     d = b["uid"]  
...: else:  
...:     d = None  
Traceback (most recent call last):  
  
File "<ipython-input-4-aff2343519a9>", line 1, in <module>  
if b.has_key("uid"):  
  
AttributeError: 'dict' object has no attribute 'has_key'
```

字典 – 檢查 key: in, get

In [5]:

```
....: if "uid" in b:    # v3.x 直接使用 in
....:     d = b["uid"]
....: else:
....:     d = None
....: print(d)
```

168

In [6]: d = b.get("uid", None) # 敲簡潔

```
....: print(d) ① ②
```

168



字典 – loop 處理

```
# dict - Loop 處理
mydict = {
    "uid": 168,
    "login": "marvelous",
    "name" : 'Alan Lee'
}
mydict

# for - 回傳 keys
for x in mydict:
    print(x)

# for - 使用 keys
for x in mydict.keys():
    print(x)

# for - 回傳 values
for x in mydict:
    print(mydict[x])

# for - 使用 values()
for x in mydict.values():
    print(x)

# for - 回傳 (key, value) 使用 items()
for x,y in mydict.items():
    print(x, y)
```

字典複製 – copy, dict

```
# 字典複製-使用 copy
mydict = {
    "uid": 168,
    "login": "marvelous",
    "name" : 'Alan Lee'
}
mydict

mydict2 = mydict.copy()
print(mydict2)

# 字典複製-使用 dict
mydict3 = dict(mydict)
print(mydict3)

mydict2 == mydict3 # True
```

巢狀字典 (Nested Dictionaries)

```
# 方法1 一次建立一個巢狀字典
mycodes = {
    "code1" : {
        "name" : "Fortran77",
        "year" : 1977
    },
    "code2" : {
        "name" : "Python",
        "year" : 1991
    },
    "code3" : {
        "name" : "R",
        "year" : 2000
    }
}

mycodes
```

```
# 方法2 建立三個字典，再合併為一項字典
mycode1 = {
    "name" : "Fortran77",
    "year" : 1977
}

mycode2 = {
    "name" : "Python",
    "year" : 1991
}

mycode3 = {
    "name" : "R",
    "year" : 2000
}

mycodes2 = {
    "程式1" : mycode1,
    "程式2" : mycode2,
    "程式3" : mycode3
}

mycodes2
```



實作練習5

將 list 轉換為 dictionary

- 輸入: `lst = ['a', 1, 'b', 2, 'c', 3]`
- 結果: `{'a': 1, 'b': 2, 'c': 3}`
- 技巧: 使用 for loop

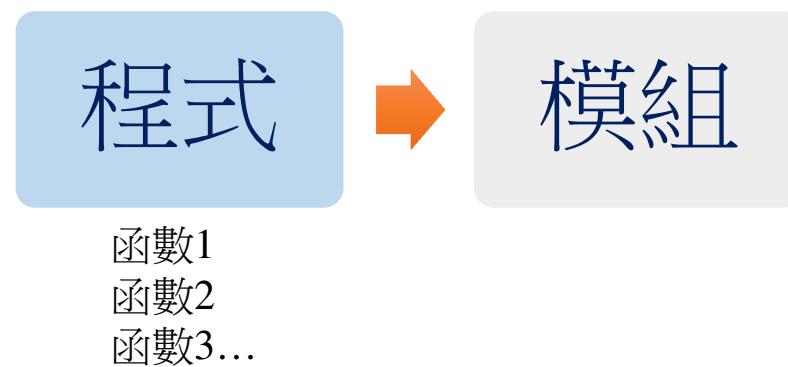
字典 – 方法

方法	功能
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

模組 (Modules)

模組

- Python 可以將一個主程式，分割成多個檔案，此分割的檔案可以形成模組 (module)。
- 主檔名表示模組的名稱，即檔案名稱 = 「**模組名稱 .py**」。
- 模組中包含 Python 函數和語法的檔案，可由其他專案使用。
- 模組中的函數可以被 import 到其他模組中，或是被 import 至Python程式。
- 在模組中，模組的名稱 (字串) 是全域變數 `_name_` 的值。
- 第1次執行時，原始程式會編譯成「**.pyc 檔案**」之**位元碼(bytecode)**，以增進執行效率。
- 模組下載：
 - **pip install moduleName**
 - **conda install moduleName**



使用模組

- 汇入模組的所有函數
 - import 模組名稱
 - import 模組名稱 as 別名
- 汇入特定函數
 - from 模組名稱 import 函數名稱1,...
 - from 模組名稱 import *
- 使用模組內的特定函數
 - 模組名稱.函數()

```
In [1]: import math
```

```
In [2]: math.sqrt(9)
```

```
Out[2]: 3.0
```

```
In [3]: from math import sqrt
```

```
In [4]: sqrt(9)
```

```
Out[4]: 3.0
```

切換工作目錄

- os.getcwd()
- os.chdir("C:/")

```
# 切換工作目錄
import os
os.getcwd() # 讀取工作目錄
os.chdir("C:/") # 變更工作目錄
os.getcwd()
os.listdir(os.getcwd()) # 顯示檔案清單
```

模組的搜尋路徑

```
In [1]: import sys
```

```
In [2]: sys.path
```

Out[2]:

```
['C:\\\\Users\\\\88697\\\\anaconda3\\\\python38.zip',
 'C:\\\\Users\\\\88697\\\\anaconda3\\\\DLLs',
 'C:\\\\Users\\\\88697\\\\anaconda3\\\\lib',
 'C:\\\\Users\\\\88697\\\\anaconda3',
 '',
 'C:\\\\Users\\\\88697\\\\anaconda3\\\\lib\\\\site-packages',
 'C:\\\\Users\\\\88697\\\\anaconda3\\\\lib\\\\site-packages\\\\locket-0.2.1-py3.8.egg',
 'C:\\\\Users\\\\88697\\\\anaconda3\\\\lib\\\\site-packages\\\\win32',
 'C:\\\\Users\\\\88697\\\\anaconda3\\\\lib\\\\site-packages\\\\win32\\\\lib',
 'C:\\\\Users\\\\88697\\\\anaconda3\\\\lib\\\\site-packages\\\\Pythonwin',
 'C:\\\\Users\\\\88697\\\\anaconda3\\\\lib\\\\site-packages\\\\IPython\\\\extensions',
 'C:\\\\Users\\\\88697\\\\.ipython']
```



自訂模組

- 自訂模組計算商數與餘數
檔案名稱為 **numberscompute.py**

```
# numberscompute.py
def divide(a,b):
    q = a/b          # q 商數
    r = a - q*b      # r 餘數
    return q,r
```

q = a/b 須要修改

- 練習檔案名稱為 **mycompute.py**

```
import numberscompute
x,y = numberscompute.divide(42,5)
```

模組名稱.函數()

類別

物件導向

- 物件用來描述一個物、事、人
- 物件內含：
 - 資訊, 即屬性
 - 可執行的操作, 即方法
- 以物件的概念來解決問題，稱為「物件導向程式設計」(Object-oriented programming, OOP)
- 支援物件的程式語言稱為「物件導向程式語言」(Object-oriented programming language, OOPL)，Python 屬此類語言

類別 (Class) → 實例 (Instance)

- 每個物件都是某個類別 (Class) 所產生的一個實例 (Instance)
 - 例: 工廠裡模具可以產生許多相同規格的產品一般：模具就是類別，產品就是實例
- 類別裡可包含屬性 (Attribute) 及方法 (Method)
 - 屬性：儲存物件的資訊，也就是變數
 - 方法：操作物件的資訊，也就是函式(函數)
- Python 使用 **class** 定義類別
- <https://docs.python.org/3/tutorial/classes.html>

範例1 - class

In [1]:

```
...:  
...: class MyClass:  
...:     x = 5
```

In [2]:

```
...: p1 = MyClass()  
...: print(p1.x)
```

5

範例2 - __init__

In [1]:

```
...:  
...: class Person:  
...:     def __init__(self, name, age):  
...:         self.name = name  
...:         self.age = age  
...:  
...: p1 = Person("John", 36)
```

- `__init__()` 函數表示類別初使化時,該函數會自動執行
- 左右為二個底線

In [2]: p1

Out[2]: <`__main__.Person` at 0x2b5a653b1f0>

In [3]: `print(p1.name)`

John

In [4]: `print(p1.age)`

36

範例3

In [1]:

```
...: class Person:  
...:     def __init__(self, fname, lname):  
...:         self.firstname = fname  
...:         self.lastname = lname  
...:  
...:     def printname(self):  
...:         print(self.firstname, self.lastname)
```

In [2]: x = Person("ALAN", "LEE")

In [3]: x.printname()

ALAN LEE

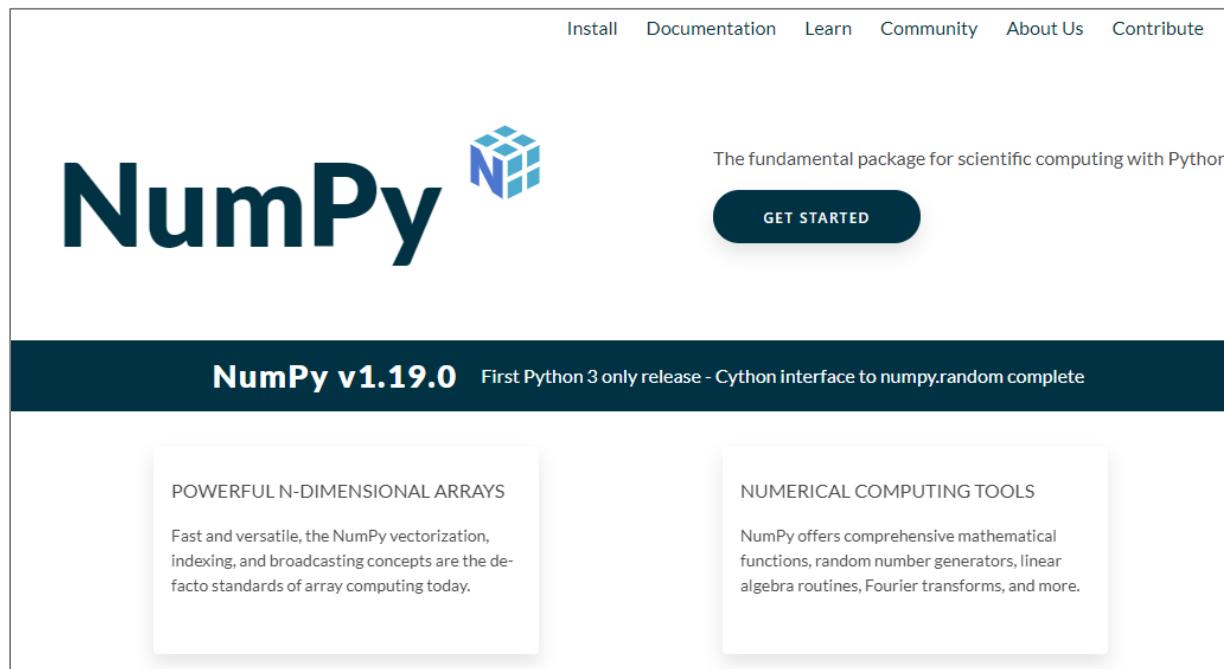
- 父類別(Parent class)是給其他繼承的類別, 稱為基本類別.
- 子類別(Child class)是從另一個父類別繼承的, 也稱為衍生類別(derived class).

Python demo

NumPy 模組

NumPy 模組

- 全名 Numeric Python
- 提供向量(1維)、矩陣(2維)運算與高效能陣列(多維)處理。
- <https://numpy.org/>



向量

- 向量 (Vector) : 向量是方向和大小值。 Deep Learning
- 可用來表示速度、加速度、動力與張量 (tensor) 等。
- 向量是一序列數值，有多種表示方法，在NumPy是使用一維陣列來表示。
- 向量範例
 - $[1, 2, 3]$
 - $\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$
 - $1i + 2j + 3k + 4l + 5m$

矩陣

- 矩陣 (Matrix) : 矩陣類似向量，只是形狀是二維表格的列 (Rows) 和欄 (Columns) 。
- 可以使用列和欄來取得指定元素值，在NumPy是使用二維陣列方式來表示。
- 矩陣範例：
 - $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

陣列

- 陣列 (Arrays) 類似 Python 清單 (Lists) ，但是陣列元素的資料型態必須是相同，不同於清單可以不同。
- 陣列的每一個元素值都是相同的資料型態。
- 陣列是程式語言的一種基本資料結構，屬於循序性的資料結構。
- 陣列是一序列的整數 int 或浮點數 float 值。
- 可以使用 tuple 或 list 來建立一維、二維或多維的陣列。

一維陣列

```
In [1]: import numpy as np  
....:  
....: #####  
....: # 一維陣列  
....: #####  
....:  
....: # 使用 tuple 或 list 建立一維陣列  
....: a = np.array([1, 2, 3, 4, 5])  
....: b = np.array((1, 2, 3, 4, 5), dtype=float)  
....:  
....: print(a)  
[1 2 3 4 5]
```

```
In [2]: print(b)  
[1. 2. 3. 4. 5.]
```

```
In [3]: print(type(a))  
<class 'numpy.ndarray'>
```

```
In [4]: print(type(b))  
<class 'numpy.ndarray'>
```

一維陣列 (續)

```
In [5]: print(a[0], a[1], a[2], a[3])  
1 2 3 4
```

```
In [6]: b[0] = 5
```

```
In [7]: print(b)  
[5. 2. 3. 4. 5.]
```

```
In [8]: b[4] = 0
```

```
In [9]: print(b)  
[5. 2. 3. 4. 0.]
```

二維陣列

```
In [2]: # 使用巢狀清單建立二維陣列
```

```
In [3]: # axis 0:列, axis 1:行
```

```
In [4]: a = np.array([[1,2,3],[4,5,6]])
```

```
In [5]: a
```

```
Out[5]:
```

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

```
In [6]: print(type(a))  
<class 'numpy.ndarray'>
```

```
In [7]: print(a[0, 0], a[0, 1], a[0, 2])  
1 2 3
```

```
In [8]: print(a[1, 0], a[1, 1], a[1, 2])  
4 5 6
```

```
In [9]: a[0, 0] = 6
```

```
In [10]: a[1, 2] = 1
```

```
In [11]: print(a)
```

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

巢狀清單 [[...], [...], ...]



np.arange

```
In [13]: a = np.arange(5) # [0 1 2 3 4]
```

```
In [14]: print(a)  
[0 1 2 3 4]
```

```
In [15]: b = np.arange(1, 11, 2) # 1<= x < 11
```

```
In [16]: print(b) # [1 3 5 7 9]  
[1 3 5 7 9]
```

np.zeros, np.ones, np.full

```
# np.zeros
np.zeros(5) # array([0., 0., 0., 0., 0.])

np.zeros(5, dtype=int) # array([0, 0, 0, 0, 0])

np.zeros((3, 2)) # 建立3列,2行皆為零的陣列
# array([[0., 0.],
#        [0., 0.],
#        [0., 0.]])  
  
# np.ones
np.ones(3) # array([1., 1., 1.])  
  
# np.full
np.full(shape = (3, 4), fill_value = 99)
# array([[99, 99, 99, 99],
#        [99, 99, 99, 99],
#        [99, 99, 99, 99]])
```

np.zeros_like, np.ones_like

```
# zeros_like
a = np.array([[1,2,3], [4,5,6]])
a
# array([[1, 2, 3],
#         [4, 5, 6]])

np.zeros_like(a)
# [[0 0 0]
#  [0 0 0]]

# ones_like
np.ones_like(a)
# [[1 1 1]
#  [1 1 1]]
```

陣列儲存與載入



實作練習7

```
# 使用 save 將 Numpy 陣列儲存成外部檔案
outfile = 'myarray.npy'
with open(outfile, 'wb') as fp:
    np.save(fp, a)

# 使用 Load 將外部檔案匯入至Numpy 陣列
outfile = "myarray.npy"
with open(outfile, 'rb') as fp:
    mydata = np.load(fp)
print(mydata)
```

常數 Constants

```

np.Inf # 無限大 inf

np.NAN # nan

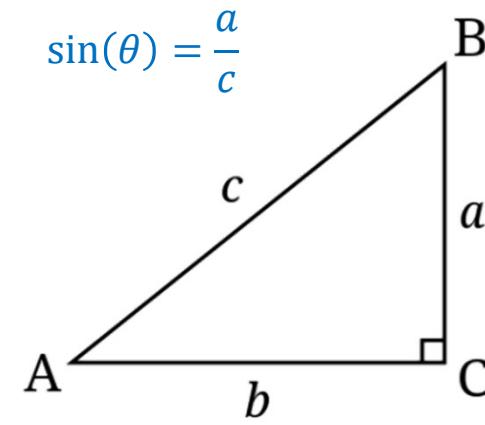
# 新版本使用 nan
np.nan

np.pi # 3.141592653589793

# Euler's constant, base of natural Logarithms
# Napier's constant(蘇格蘭數學家約翰·納皮爾)
np.e # 2.718281828459045

# 三角函數
# sin(30度) = sin(pi/6) = 0.5
# sin(45度) = sqrt(2)/2 = 0.707
# sin(60度) = sqrt(3)/2 = 0.866
# sin(90度) = 1
a = np.array([30, 45, 60, 90])
np.sin(a*np.pi/180)

```



亂數

- NumPy 的 random 子模組提供多種函數來產生亂數
- 函數 np.random.random() # 建立0~1之亂數
- 函數 np.random.random(樣本數)
- 函數 np.random.rand() # 建立0~1之亂數
- 函數 np.random.rand(樣本數)
- 函數 np.random.rand(列, 行)
- 函數 np.random.randint(最大值, size = 樣本數) # 建立整數亂數
- 函數 np.random.randint(最小值, 最大值, size = 樣本數)

random.random, random.rand

```
In [1]: import numpy as np  
  
In [2]: np.random.seed(123) # 設定亂數種子，須輸入 >= 1 的整數  
  
In [3]: x1 = np.random.random()  
  
In [4]: print(x1)  
0.6964691855978616  
  
In [5]: x2 = np.random.random(3)  
  
In [6]: print(x2)  
[0.28613933 0.22685145 0.55131477]  
  
In [7]: x3 = np.random.rand()  
  
In [8]: print(x3)  
0.7194689697855631  
  
In [9]: x4 = np.random.rand(3)  
  
In [10]: print(x4)  
[0.42310646 0.9807642 0.68482974]  
  
In [11]: x5 = np.random.rand(3, 2) # 3列, 2行  
  
In [12]: print(x5)  
[[0.4809319 0.39211752]  
 [0.34317802 0.72904971]  
 [0.43857224 0.0596779 ]]
```

random.randint

```
In [13]: x6 = np.random.randint(5, 10)
```

```
In [14]: print(x6)  
5
```

```
In [15]: x7 = np.random.randint(1, 11, size=10)
```

```
In [16]: print(x7)  
[1 5 2 8 4 3 5 8 3 5]
```

```
In [17]: x8 = np.random.randint(1, 11, size=(4, 5))
```

```
In [18]: print(x8)  
[[ 9  1  8 10  4]  
 [ 5  7  2  6  7]  
 [ 3  2  9  4  6]  
 [ 1  3  7  3  5]]
```

標準常態分配隨機樣本

```
In [1]: from numpy import random
```

```
In [2]: # 舊版用法
```

舊版方法

```
In [3]: vals = random.standard_normal(3)
....: print(vals)
[-2.51130748 -1.73201266 -0.83724647]
```

```
In [4]: more_vals = random.standard_normal(3)
....: print(more_vals)
[ 1.26288956 -1.29828688  0.14763598]
```

default_rng.standard_normal

In [5]: # 新版用法

In [6]: from numpy.random import default_rng

In [7]: rng = default_rng()
....: vals = rng.standard_normal(3)
....: print(vals)

[0.55039289 -0.96002248 -0.94853849]

新版方法,採用此法

In [8]: more_vals = rng.standard_normal(3)
....: print(more_vals)
[-0.9074675 1.14108665 -1.0669296]

參考: <https://numpy.org/doc/stable/reference/random/generator.html>

陣列屬性

- NumPy陣列提供相關屬性顯示陣列資訊。

屬性	說明
dtype	陣列元素的資料型態，整數int32/64或浮點數float32/64等
size	陣列的元素總數
ndim	陣列維度，一維是1；二維是2
shape	陣列的形狀 (Shape) ，(列,) ; (列, 行)
itemsize	一個元素佔用的位元組數
nbytes	整個陣列佔用的位元組數

陣列屬性 – 範例

```
import numpy as np
a = np.array([0,1,2,3,4,5])
a
a.dtype    # dtype('int32')
a.size     # 6
a.ndim     # 1
a.shape    # (6,)
a.itemsize # 4 bytes
a.nbytes   # 24
```

In [8]: b = np.array([[1,2,3,4], [4,5,6,7], [7,8,9,10.]])

In [9]: b
Out[9]:

```
array([[ 1.,  2.,  3.,  4.],
       [ 4.,  5.,  6.,  7.],
       [ 7.,  8.,  9., 10.]])
```

```
b.dtype      # float64
b.size       # 12
b.ndim       # 2
b.shape      # (3, 4)
b.itemsize   # 8
b nbytes    # 12*8=96
```

```
# 資料型別轉換
b.astype('int32')
b = b.astype('int32')
b.dtype    # int32
```



建立3維陣列

實作練習8

- 建立3維陣列 myzero 。
- 5個元素, 每個元素為 3列, 4行的零矩陣。
- myzero.shape 結果為 (5, 3, 4) 。
- 參考右圖結果。

```
array([[[0., 0., 0., 0.],  
       [0., 0., 0., 0.],  
       [0., 0., 0., 0.]],  
  
      [[0., 0., 0., 0.],  
       [0., 0., 0., 0.],  
       [0., 0., 0., 0.]],  
  
      [[0., 0., 0., 0.],  
       [0., 0., 0., 0.],  
       [0., 0., 0., 0.]],  
  
      [[0., 0., 0., 0.],  
       [0., 0., 0., 0.],  
       [0., 0., 0., 0.]],  
  
      [[0., 0., 0., 0.],  
       [0., 0., 0., 0.],  
       [0., 0., 0., 0.]])
```

array 一維陣列 - loop 處理

```
#####
# array 一維陣列 - Loop 處理
#####

a = np.array([1,2,3,4])
a

# 方法1. array - 取出元素, 使用for
for x in a:
    print(x)

# 方法2. array - 取出元素, 使用while
i = 0
while i < len(a):
    print(a[i])
    i = i + 1

# 方法3. array - 取出元素, 使用指標 range, Len
for i in range(len(a)):
    print(a[i])

# 方法4. array - 取出元素, 使用陣列包含法
[print(x) for x in a]
```

array 二維陣列 - loop 處理

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

```
In [3]: a
```

```
Out[3]:
```

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

```
In [4]:
```

```
...: for x in a:  
...:     print(x)
```

```
[1 2 3 4]
```

```
[5 6 7 8]
```

```
In [5]: for x in a:
```

```
...:     for item in x:
```

```
...:         print(str(item) + " ", end = " ")
```

```
1 2 3 4 5 6 7 8
```

陣列運算

```
In [2]: a = np.array([1,2,3])
```

```
In [3]: b = np.array([4,5,6])
```

```
In [4]: a+b # 加
```

```
Out[4]: array([5, 7, 9])
```

```
In [5]: a-b # 減
```

```
Out[5]: array([-3, -3, -3])
```

```
In [6]: a*b # 乘
```

```
Out[6]: array([ 4, 10, 18])
```

不是矩陣相乘

```
In [7]: a/b # 除
```

```
Out[7]: array([0.25, 0.4 , 0.5 ])
```

矩陣相乘, 轉置

```
# 矩陣相乘(dot)
a = np.array([[1,2],[3,4],[5,6]])
a
b = np.array([[1,2],[3,4]])
b
a.shape
b.shape
c = a.dot(b) # 矩陣相乘(dot)
c
```

$$\begin{aligned} a \cdot b &= \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \\ &= \begin{bmatrix} 7 & 10 \\ 15 & 22 \\ 23 & 34 \end{bmatrix} = c \end{aligned}$$

```
np.transpose(c) # 矩陣轉置
c.T # 矩陣轉置
```

```
array([[ 7, 15, 23],
       [10, 22, 34]])
```

反矩陣

```
# inv() : 反矩陣, 逆矩陣 (inverse matrix)
from numpy.linalg import inv

x = np.array([[1, 2], [3, 4]])

inv(x)
# array([[-2. ,  1. ],
#        [ 1.5, -0.5]])

# 單位矩陣 (Identity matrix)
x.dot(inv(x))
# array([[1.00000000e+00,  1.11022302e-16],
#        [0.00000000e+00,  1.00000000e+00]])

x.dot(inv(x)).round(1)
# array([[1.,  0.],
#        [0.,  1.]])
```

numpy.linalg.inv

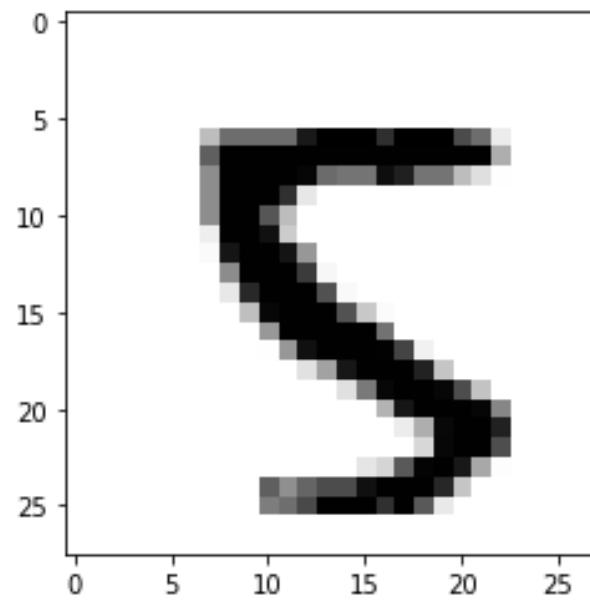
行列式值, 特徵值, 特徵向量

```
# 計算矩陣行列式值 (determinant)
np.linalg.det(x)
# -2.000000000000004

# 計算方形矩陣的特徵值 (eigenvalue) 與特徵向量 (eigenvector)
np.linalg.eig(x)
# (array([-0.37228132,  5.37228132]),
#  array([[-0.82456484, -0.41597356],
#         [ 0.56576746, -0.90937671]]))
```

陣列應用 - 高維度影像 MNIST 手寫數字辨識資料集

高維度影像



(28列*28行 = 784)

MNIST 手寫數字辨識資料集

```
# MNIST 手寫數字辨識資料集  
# http://yann.Lecun.com/exdb/mnist/
```

載入資料

```
from sklearn.datasets import fetch_openml  
from sklearn.model_selection import train_test_split  
import matplotlib.pyplot as plt
```

區分訓練集，
測試集

```
# 方法1 回傳 Bunch 資料物件  
# 原圖為28*28, 2維展開為1維 28*28=784
```

```
mnist_data = fetch_openml("mnist_784")  
xdata = mnist_data["data"] # 70000*784  
ydata = mnist_data["target"] # 70000
```

網路下載，
須一點時間

繪製數字影像

```
# 方法2 直接回傳 X, y

# Load data from https://www.openml.org/d/554
X, y = fetch_openml('mnist_784', return_X_y=True)
# X : 70000*784
# y : 70000

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    random_state=123,
                                                    test_size=10000)

type(X_train) # DataFrame (早期版本為 numpy.ndarray)

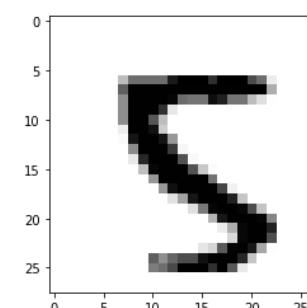
# 將 DataFrame 轉換成 array 物件
X_train = X_train.to_numpy()
y_train = y_train.to_numpy()

type(X_train) # numpy.ndarray
X_train.ndim # 2
X_train.shape # (60000, 784)
X_train.dtype # dtype('float64')

type(y_train) # numpy.ndarray
y_train.ndim # 1
y_train.shape # (60000,)
y_train.dtype # dtype('O')

# 繪製數字影像
plt.imshow(X_train[0].reshape(28,28), cmap='binary')

# 實際值
y_train[0] # '5'
```



- 'b' boolean
- 'i' (signed) integer
- 'u' unsigned integer
- 'f' floating-point
- 'c' complex-floating point
- 'O' (Python) objects, 字串
- 'S', 'a' (byte-)string
- 'U' Unicode
- 'V' raw data (void)

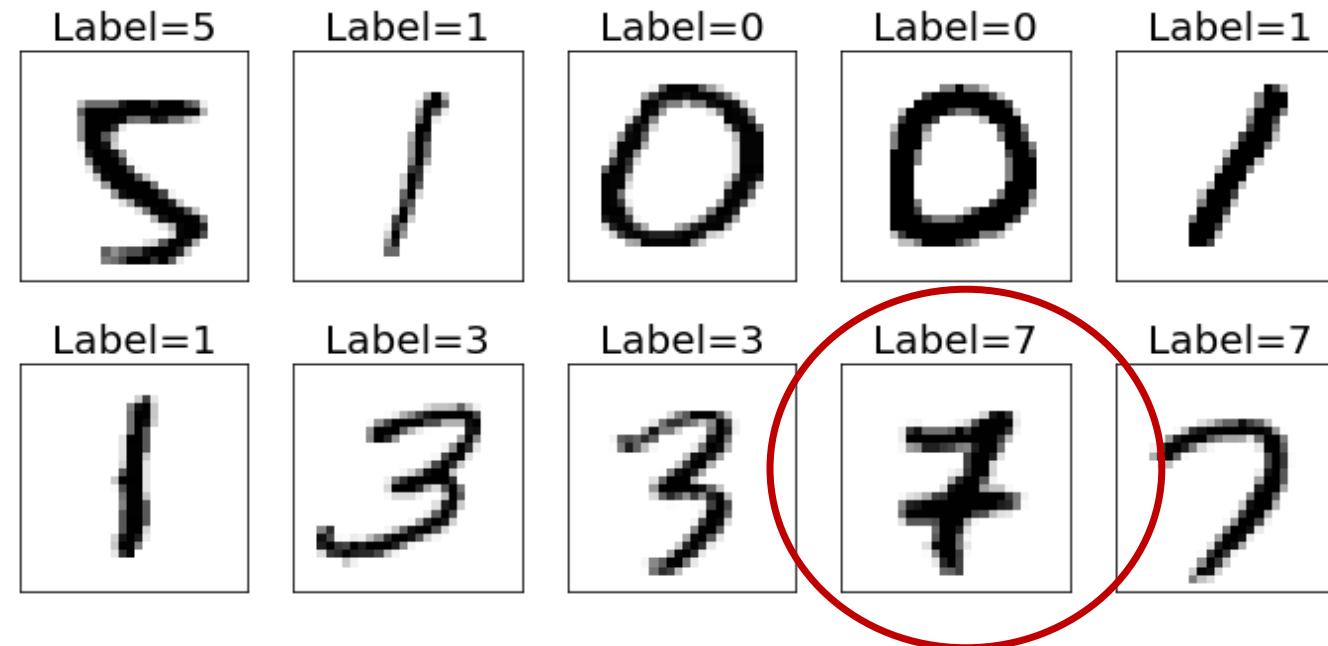
繪製多個數字影像

```
# 繪製多個數字影像，最多一次顯示25個
def plot_images_labels(images, labels, idx, num=10):
    fig = plt.gcf() # 取得目前的 figure
    fig.set_size_inches(12, 14) # 設定圖形大小
    if num > 25: num=25
    for i in range(0, num):
        ax=plt.subplot(5, 5, 1+i)
        ax.imshow(images[idx].reshape(28,28), cmap='binary')
        title= "Label=" + str(labels[idx])
        ax.set_title(title, fontsize=20)
        ax.set_xticks([])
        ax.set_yticks([])
        idx+=1
    plt.show()

plot_images_labels(X_train, y_train, 0, 10)
```

- def
- if
- for

多個數字繪圖





reshape 應用

```
In [1]: import numpy as np  
  
In [2]: z = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])  
...: z  
Out[2]:  
array([[ 1,  2,  3,  4],  
       [ 5,  6,  7,  8],  
       [ 9, 10, 11, 12]])  
  
In [3]: z.reshape(-1) # -1: unknown dimension  
Out[3]: array([ 1,  2,  3, ..., 10, 11, 12])  
  
In [4]: z.reshape(-1,1) # row -1: unknown , column:1  
Out[4]:  
array([[ 1],  
       [ 2],  
       [ 3],  
       ...,  
       [10],  
       [11],  
       [12]])  
  
In [5]: z.reshape(-1, 2) # row -1: unknown , column:2  
Out[5]:  
array([[ 1,  2],  
       [ 3,  4],  
       [ 5,  6],  
       [ 7,  8],  
       [ 9, 10],  
       [11, 12]])
```

Python demo

建立副本

```
# 建立副本-使用等號
a = np.array([0,1,1,2,3,5])
b = a.reshape((3,2)) # b之修改會影響a
b

b.ndim    # 2
b.shape   # (3,2)

b[1][0] = 168
b
a # a物件已經更改, array([ 0,  1, 168,  3,  4,  5])

# 建立副本-使用 copy
c = a.reshape((3,2)).copy()
c
c[0][0] = -999
c
a # a物件沒有更改
```

向量化處理

```
# 向量化處理  
a = np.array([0,1,1,2,3,5])  
  
a**2  
  
a**3 # 次方運算  
  
# indexing  
a[np.array([1,3,5])]
```

離群值調整

```
In [12]: a = a**3
```

```
In [13]: a
```

```
Out[13]: array([ 0, 1, 1, 8, 27, 125], dtype=int32)
```

```
In [14]: a > 10
```

```
Out[14]: array([False, False, False, False, True, True], dtype=bool)
```

```
In [15]: a[a > 10] = 10
```

```
In [16]: a
```

```
Out[16]: array([ 0, 1, 1, 8, 10, 10], dtype=int32)
```

```
In [17]: a.clip(0, 3)
```

```
Out[17]: array([0, 1, 1, 3, 3, 3], dtype=int32)
```

使用 clip

nan 處理

```
In [3]: x = np.array([1, 2, 3, np.NAN, 4])
...: x
...: np.isnan(x)
Out[3]: array([False, False, False,  True, False], dtype=bool)
```

```
In [4]: x[~np.isnan(x)]
Out[4]: array([ 1.,  2.,  3.,  4.])
```

```
In [5]: np.mean(x[~np.isnan(x)])
Out[5]: 2.5
```

```
In [6]: np.mean(x) # nan
Out[6]: nan
```

計算時間

```
In [1]: import timeit  
...: import numpy as np  
...:  
...: normal_py_sec = timeit.timeit('sum(x*x for x in range(1000))', number=10000)  
...: naive_np_sec = timeit.timeit('sum(na*na)', setup="import numpy as np; na=np.arange(1000)", number=10000)  
...: good_np_sec = timeit.timeit('na.dot(na)', setup="import numpy as np; na=np.arange(1000)", number=10000)  
  
In [2]: print("Normal Python: %f sec"%normal_py_sec)  
Normal Python: 0.944890 sec  
  
In [3]: print("Naive NumPy: %f sec"%naive_np_sec)  
Naive NumPy: 0.687164 sec  
  
In [4]: print("Good NumPy: %f sec"%good_np_sec)  
Good NumPy: 0.015138 sec
```

最快/最慢：62倍
$$\frac{0.944890}{0.015138} \approx 62$$

03.字串與正規表示式,判斷式與函數應用



str 字串

Text Sequence Type — str

Textual data in Python is handled with `str` objects, or *strings*. Strings are immutable *sequences* of Unicode code points. String literals are written in a variety of ways:

- Single quotes: `'allows embedded "double" quotes'`
- Double quotes: `"allows embedded 'single' quotes".`
- Triple quoted: `'''Three single quotes''', """Three double quotes"""`

- 字串輸入
- 字串處理
- 字串搜尋
- 字串格式化
- 參考: <https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str>

字串輸入

字串輸入 input

```
In [1]: name = input("keyin your name? ")
```

```
keyin your name? RWEPA
```

```
In [2]: print(name)  
RWEPA
```

```
In [3]: type(name) # str  
Out[3]: str
```

```
In [4]: mynumbers = input("輸入2個數值，中間使用逗號區隔，計算2個數值相加結果？")
```

輸入2個數值，中間使用逗號區隔，計算2個數值相加結果？3,5

```
In [5]: print(mynumbers)
```

3,5

思考-如何計算

x, y = eval(input())

字串處理

字串處理

- 字串 (String)由一個(以上)字元所組成
- 字串左右二側須使用單引號或是雙引號
- 字串屬於序列 (Sequence),可以使用指標存取

```
In [1]: city = '台北市'  
  
In [2]: district = '信義區'  
  
In [3]: road = '信義路五段7號'  
  
In [4]: # 使用 + 字串串接  
  
In [5]: city + district  
Out[5]: '台北市信義區'  
  
In [6]: address = city + district + road  
  
In [7]: print(address)  
台北市信義區信義路五段7號
```

字串物件: <https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str>

字串方法: <https://docs.python.org/3/library/string.html>

數值與字串轉換

```
In [1]: # 數值轉換為字串
```

```
In [2]: myinteger = 123
```

```
In [3]: mystr = str(myinteger)
```

```
In [4]: mystr
```

```
Out[4]: '123'
```

```
In [5]: type(mystr)
```

```
Out[5]: str
```

```
In [6]: # 字串轉換為數值
```

```
In [7]: mystr = "123.456"
```

```
In [8]: myvalue = float(mystr)
```

```
In [9]: myvalue
```

```
Out[9]: 123.456
```

```
In [10]: type(myvalue)
```

```
Out[10]: float
```

8進位,16進位,Unicode字串表示

```
# 8進位,16進位,Unicode字串表示
```

```
'\156' # 8進位
```

```
\aaa
```

```
'\x6E' # 16進位-英文大寫
```

```
\xFFFF
```

```
'\x6e' # 16進位-英文小寫
```

```
\xffff
```

```
mystr1 = '\N{GREEK SMALL LETTER ALPHA}' # \N Unicode名稱: 'α'
```

```
mystr1
```

```
mystr2 = '\N{LATIN SMALL LETTER U WITH DIAERESIS}' # 'ü'
```

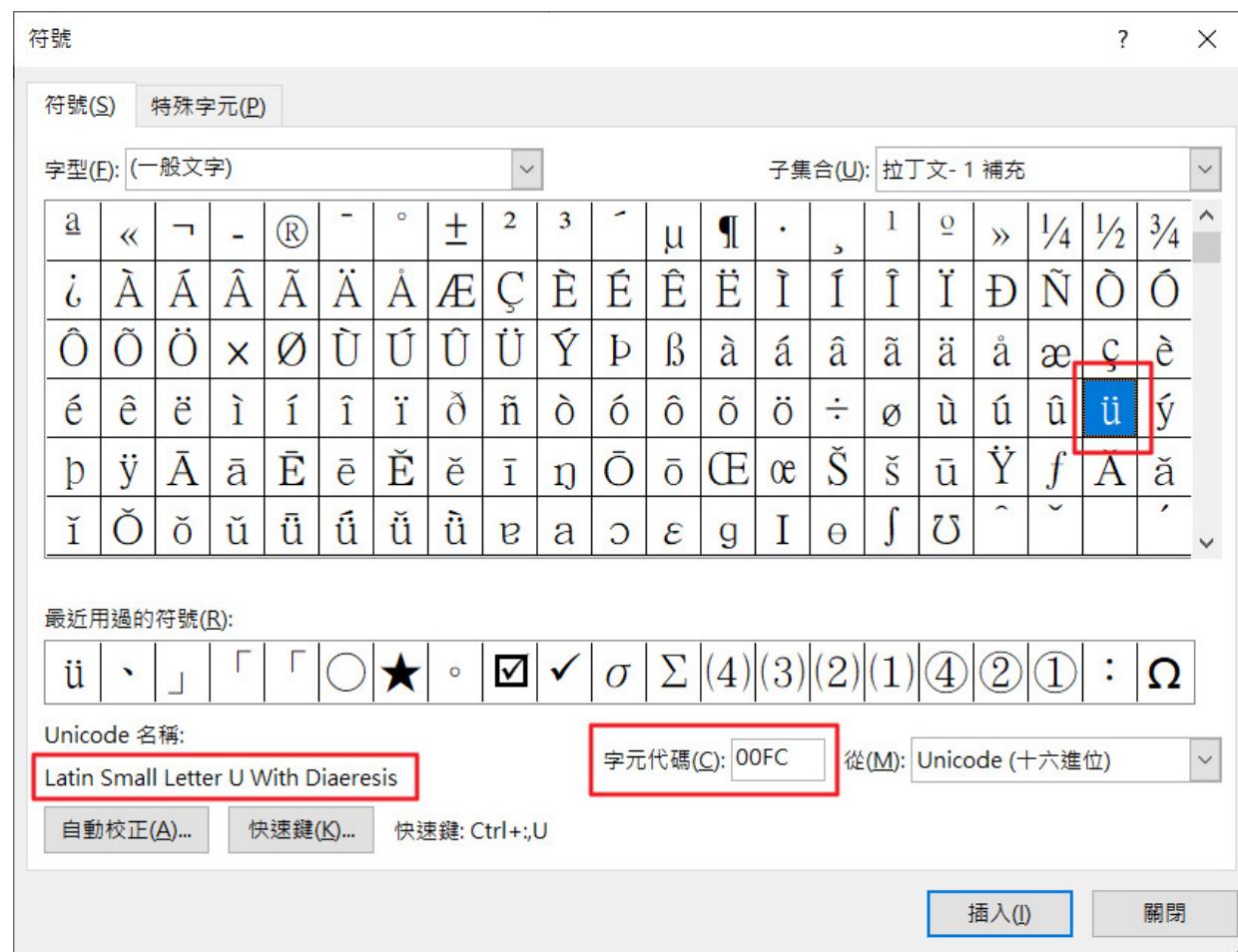
```
mystr2
```

```
mystr3 = '\u00FC' # \u 4碼16進位表示Unicode字元, 'ü'
```

```
mystr3
```

16進位名稱?

Word \ 插入 \ 符號





字串指標存取

```
# 字串指標存取
address      # '台北市,信義區,信義路五段7號'
address[0]    # '台'
address[-1]   # '號'
address[8:13] # '信義路五段'

# 使用 * 建立重複字串
city      # '台北市'
5*city
city*5

# Len 長度
len(address) # 15
```

字串 – split

- str.split(sep=None, maxsplit=-1)
- 預設分割 sep 為空白字元, maxsplit=-1 表示沒有分割次數限制

```
In [2]: address = city + ',' + district + ',' + road
```

```
In [3]: address
```

```
Out[3]: '台北市,信義區,信義路五段7號'
```

```
In [4]: address.split(',') # 預設分割字元為空白字元
```

```
Out[4]: ['台北市', '信義區', '信義路五段7號']
```

```
In [5]: address.split(',', maxsplit = 1) # 預設分割字元為空白字元
```

```
Out[5]: ['台北市', '信義區,信義路五段7號']
```

字串 - join

- join 方法將所有字串結合成單一字串

```
In [1]: city = "台北市"
       .... district = "信義區"
       .... road = "信義路五段7號"
       .... str1 = city + district + road
       .... str1
Out[1]: '台北市信義區信義路五段7號'
```

```
In [2]: ''.join([city, district, road])
Out[2]: '台北市信義區信義路五段7號'
```

```
In [3]: '-'.join([city, district, road])
Out[3]: '台北市-信義區-信義路五段7號'
```

```
In [4]: mytuple = ("台北市", "信義區", "信義路五段7號")
```

```
In [5]: ''.join(mytuple)
Out[5]: '台北市信義區信義路五段7號'
```

字串 if in (if not in)

```
In [1]: txt = "One of the world's strictest lockdowns is lifting, but many  
are scared to go back to normal life."
```

```
In [2]:
```

```
.... if "scared" in txt:  
....     print("Find 'scared'") # 完全比對
```

```
Find 'scared'
```

```
In [3]: if "scare" in txt:
```

```
....     print("Find 'scare'") # 部分字串相同
```

```
Find 'scare'
```

```
In [4]:
```

```
.... if "Scare" in txt:
```

```
....     print("Find 'Scare'") # 區分大小寫
```

刪除空白字元 strip, lstrip, rstrip

- 刪除左側或右側空白字元 strip (中間的字元不會刪除)
- 刪除左側空白字元 lstrip
- 刪除右側空白字元 rstrip

```
In [1]: x = " RWEPA - Python 大數據分析\n "
```

```
In [2]: x.strip() # \n 視為空白字元而加以刪除  
Out[2]: 'RWEPA - Python 大數據分析'
```

```
In [3]: x.lstrip()  
Out[3]: 'RWEPA - Python 大數據分析\n '
```

```
In [4]: x.rstrip()  
Out[4]: ' RWEPA - Python 大數據分析'
```

```
In [5]: import string
```

```
In [6]: string.whitespace # '\t\n\r\x0b\x0c' \x0b:印表機垂直定位符號, \x0c:換頁符號  
Out[6]: '\t\n\r\x0b\x0c'
```

空白字元包括6種

進階 strip, lstrip, rstrip

```
In [7]: x.strip("Python") # 中間字元會保留  
Out[7]: ' RWEPA - Python 大數據分析\n '
```

```
In [8]: "Python大數據分析Python".strip("python") # 有區分大小寫  
Out[8]: 'Python大數據分析P'
```

```
In [9]: "Python大數據分析Python".strip("Python") # 有區分大小寫  
Out[9]: '大數據分析'
```

```
In [10]: "Python大數據分析Python".lstrip("python")  
Out[10]: 'Python大數據分析Python'
```

```
In [11]: "Python大數據分析Python".rstrip("python")  
Out[11]: 'Python大數據分析P'
```

字串判斷 isdigit, isalpha

```
In [1]: "123".isdigit()          # 數字digits包括 '0123456789'  
Out[1]: True  
  
In [2]: "123".isalpha()          # 數字不是英文字母  
Out[2]: False  
  
In [3]: "alan9956".isalpha()     # 英文加數字不是英文字母  
Out[3]: False  
  
In [4]: "alan9956".isdigit()     # 英文加數字不是數字  
Out[4]: False
```

字串判斷 isascii, isupper, islower

```
In [5]: "!@#$%^&*()[]{}\\|".isascii() # 半型符號是ASCII  
Out[5]: True
```

```
In [6]: "おはようございます".isascii() # 日本字不是ASCII  
Out[6]: False
```

```
In [7]: "大數據分析".isascii() # 中文字不是ASCII  
Out[7]: False
```

```
In [8]: "RWEPA".isupper() # 大寫英文字母  
Out[8]: True
```

```
In [9]: "alan9956@gmail.com".islower() # 小寫英文字母  
Out[9]: True
```

字串處理練習



實作練習9

```
# 實作練習
# 計算 mystr 字串中，不區分大小寫，英文字母出前次數最多的前6名為何？
mystr = 'Scikit-learn is an Open source Machine Learning learning
library that supports supervised and unsupervised learning.'
```

- 使用 collections.Counter 方法
- 答案為 n, e, i, r, s, a

```
[('n', 12),
 ('e', 11),
 ('i', 10),
 ('r', 10),
 ('s', 9),
 ('a', 9),
 ('l', 5),
 ('p', 5),
 ('u', 5),
 ('t', 4),
 ('c', 3),
 ('o', 3),
 ('g', 3),
 ('d', 3),
 ('h', 2),
 ('v', 2),
 ('k', 1),
 ('m', 1),
 ('b', 1),
 ('y', 1)]
```



```
mystrsort[0:6]
[('n', 12), ('e', 11), ('i', 10), ('r', 10), ('s', 9), ('a', 9)]
```

字串的搜尋

字串 – find, index, rfind, replace

```
# 字串的搜尋
# find 回傳字串的索引，找不到回傳-1
mystr.find('learning') # 39
mystr.find('bigdata') # -1

mystr.index('learning') # 39
mystr.index('bigdata') # ValueError: substring not found

for x in mystr:
    print(x)

# rfind 從最後面查詢
mystr.rfind('learning') # 98

# 字串取代 - replace
address = '台北市,信義區,信義路五段7號'
address.replace(',', '*') # '台北市*信義區*信義路五段7號'
```



字串格式化 format

字串格式化 - 4種方式

1. 早期使用 %, 與 C語言 的 printf 相似
2. str.format 函數
3. Formatted String Literal - 將運算式置於字串之中 → 多採用
4. Template String 樣板字串 - 使用 string.Template → 多採用

字串格式化

```
# 1. 早期使用 %, 與 C語言 的 printf 相似
text = 'world'
print('Hello %s' % text) # Hello world
```

```
# 2. str.format 函數
```

```
# 1個參數
year = 2021
txt = "我的名字是 Alan, 現在是 {} 年"
print(txt.format(year)) # 我的名字是 Alan, 現在是 2021 年
```

字串格式化 (續)

```
# 多個參數
quantity = 1
item = "美國Blue Yeti 雪怪 USB麥克風"
price = 5590

myorder = "感謝您訂購-->產品名稱: {}, 數量: {}, 價格: {} 元"
print(myorder.format(item, quantity, price))
# 感謝您訂購-->產品名稱: 美國Blue Yeti 雪怪 USB麥克風, 數量: 1, 價格: 5590 元

# 多個參數-使用指標
quantity = 1
item = "美國Blue Yeti 雪怪 USB麥克風"
price = 5590

myorder = "感謝您訂購-->數量: {1}, 產品名稱: {0}, 價格: {2} 元"
print(myorder.format(item, quantity, price))
# 感謝您訂購-->數量: 1, 產品名稱: 美國Blue Yeti 雪怪 USB麥克風, 價格: 5590 元
```

字串格式化 (續)

3. Formatted String Literal - 將運算式置於字串之中

```
text = 'world'  
print(f'Hello, {text}') # Hello, world
```

```
x = 10  
y = 27  
print(f'x + y = {x + y}') # x + y = 37
```

4. Template String 樣板字串 - 使用 *string.Template*

```
from string import Template  
  
mytext = 'world'  
t = Template('Hello $text') # 使用1個變數 $text  
type(t) # string.Template  
t.substitute(text=mytext) # 將 text置換為 mytext, 'Hello world'
```

正規表示式

正規表示式 (Regular Expressions)

- 正規運算式(正則表示式)是一個字串撰寫規則，可以用來進行字串比對、分割與修改。
- 在正規表示式的範本字串中，中介字元(metacharacter)會有其特殊意義。
- 正規運算式直譯器(引擎)能夠將定義的正規運算式的範本字串和想要分析的字串變數進行比較，引擎可以傳回布林值，True表示字串符合範本字串的定義；False表示不符合。
- 使用 re 模組進行字串比對。
- 參考: <https://docs.python.org/3/library/re.html>

中介字元 (Metacharacter)

- 中介字元 . ^ \$ * + ? { } [] \ | ()

中介字元	功能
.	除了換行符號外的任何字元，例如 '!' 配對除了 '\n' 之外的任何字元。
^	字串開頭的子字串或排除指定字元或群組，例如 'a[^b]c' 配對除了 'abc' 之外的任何 a 開頭 c 結尾的三字元組合。
\$	字串結尾的子字串，例如 'abc\$' 配對以 'abc' 結尾的字串。
*	單一字元或群組出現任意次數，例如 'ab*' 配對 'a' 、 'ab' 或 'abb' 等等。
+	單一字元或群組出現至少一次，例如 'ab+' 配對 'ab' 或 'abb' 等等。
?	單一字元或群組 0 或 1 次，例如 'ab?' 配對 'a' 或 'ab' 。
{m,n}	單一字元或群組的 m 到 n 倍數，例如 'a{6}' 為連續六個 'a' ， 'a{3,6}' 為三到六個 'a' 。
[]	對中括弧內的字元形成集合，例如 '[a-z]' 為所有小寫英文字母。'[A-Z]' 為所有大寫英文字母。
\	特別序列的起始字元。
	單一字元或群組的或，例如 'alb' 為 'a' 或 'b' 。
()	對小括弧內的字元形成群組。

特別序列 (Special sequences)

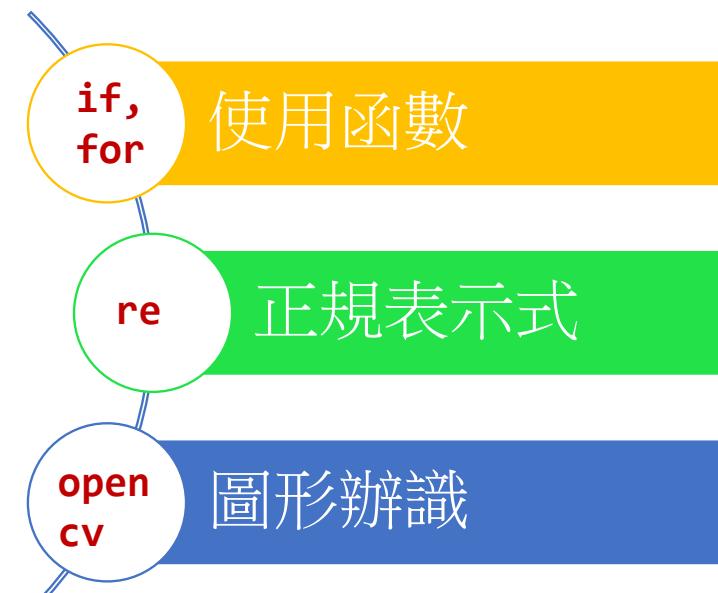
特別序列	功能
\number	群組的序數，群組以()表示。
\A	字串的開頭字元，單行模式與^是相同的。
\b	單字的界線字元(最左與最右)，例如 r'\bfoo\b' 配對 'foo' 或 'bar foo baz' 。
\B	單字的非界線字元。
\d	數字 0~9，與 [0-9] 相同。
\D	非數字，與 [^0-9] 相同。
\s	空白符號，包括新行符號 \n ，與 [\t\n\r\f\v] 相同。 例：[\s,] 符合空白符號或,或.
\S	非空白符號，與 [^ \t\n\r\f\v] 相同。
\w	任意文字字元，包括數字，與 [a-zA-Z0-9_] 相同。
\W	非文字字元，包括空白符號，與 [^a-zA-Z0-9_]
\Z	字串的結尾字元。

re 方法

- `re.compile` 建立正規表示式
- `re.match` 從開始位置檢查是否匹配，如果有，則回傳匹配結果，如果沒有，則回傳 `None`
- `re.fullmatch` 從開始或結束位置檢查是否匹配
- `re.search` 從任何位置開始模式匹配，回傳第 1 筆
- `re.findall` 回傳與模式匹配的所有字串
- `re.split` 分割
- `re.sub` 取代

re範例-電話號碼

- 02-8101-8800 (區域碼)2碼數字-4碼數字-4碼數字
- 如何判斷是否符合電話號碼



方法1-使用 if, for (17行程式)

02-8101-8800

```
def isPhoneNumber(text):
    if len(text) != 12:
        return False # not phone number-sized
    for i in range(0, 2):
        if not text[i].isdecimal():
            return False # not an area code
    if text[2] != '-':
        return False # does not have first hyphen
    for i in range(3, 7):
        if not text[i].isdecimal():
            return False # does not have first 3 digits
    if text[7] != '-':
        return False # does not have second hyphen
    for i in range(8, 12):
        if not text[i].isdecimal():
            return False # does not have last 4 digits
    return True # "text" is a phone number!
```

↑ ↑ ↑
2 7 11

使用 if, for 結果

```
In [2]: print('02-8101-8800is a phone number: ')
02-8101-8800is a phone number:
```

```
In [3]: print(isPhoneNumber('02-8101-8800'))
True
```

```
In [4]: print('02-1234-ALAN is a phone number: ')
02-1234-ALAN is a phone number:
```

```
In [5]: print(isPhoneNumber('02-1234-ALAN'))
False
```

方法2-使用 re 正規表示式

```
In [1]: import re  
....  
....: phoneRegex = re.compile(r'\d\d-\d\d\d\d\d\d-\d\d\d\d') # 建立正規表示式物件  
....:  
....: mystr = '02-8101-8800'  
....: myresult1 = phoneRegex.match(mystr)  
....: myresult1  
Out[1]: <re.Match object; span=(0, 12), match='02-8101-8800'>  
  
In [2]: mystr = '02-8101-ALAN'  
....: myresult2 = phoneRegex.match(mystr)  
....: myresult2
```

先建立re.compile() 物件

re.match 從開始位置檢查是否匹配

```
# re範例-字符串  
mystr = 'Please call ALAN at 02-2822-5252. 02-1234-5678 is his office  
number. Try 02-87654321'  
  
# match 方法  
phoneRegex = re.compile(r'\d\d-\d\d\d\d\d-\d\d\d\d') # 建立正規表示式物件  
  
# 方法1  
myresult = phoneRegex.match(mystr) # 使用match, 從最左側搜尋.  
print(myresult) # None 表示沒有找到符合字符串  
  
# 方法2  
myresult1 = re.match(phoneRegex, mystr)  
print(myresult1) # None, 與方法1結果相同.
```

物件.方法()

re.match()

fullmatch, search, findall

```
# fullmatch 方法，與 match類似，除了最左側另包括最右側搜尋
print(re.fullmatch(phoneRegex, mystr)) # None

# search 方法
phoneRegex = re.compile(r'\d\d-\d\d\d\d\d-\d\d\d\d')
phoneRegex.search(mystr)
# 有找到1筆資料
# <re.Match object; span=(20, 32), match='02-2822-5252'>

phoneRegex = re.compile(r'\d{2}-\d{4}-\d{4}')
phoneRegex.search(mystr) # 結果與前面相同

# findall 方法
phoneRegex.findall(mystr)
# ['02-2822-5252', '02-1234-5678']
```



字串練習-match, fullmatch, search, findall

實作練習10

```
# 實作練習-字串
# 以 mystr 字串練習 Learning 或 Learning
mystr = 'Learning is the obtaining of new knowledge. Scikit-learn is an
Open source Machine learning library that supports supervised and
unsupervised learning'
```

- 使用 match, fullmatch, search, findall



使用 `findall` 找出所有股票代碼

實作練習11

```
# 實作練習
# 以 mystr 字串練習，使用 findall 找出所有股票代碼
# 結果為 ['2481-TW', '4952-TW', '3264-TW', '6531-TW']
mystr = '隨著歐美多國解封，燃油車、電動車市況明顯回溫，帶動車用電子強勁需求，相關零組件廠出貨也陸續報捷，車電族群今（13）日再度發動猛攻，包括強茂（2481-TW）、凌通（4952-TW）、欣銓（3264-TW）、愛普（6531-TW）等多檔，盤中均亮燈漲停。'
```

group 方法, 使用左右括號 ()

```
# group 方法, 使用左右括號 ()
mystr = '101大樓客服電話是 02-8101-8800'
phoneRegex = re.compile(r'(\d{2})-(\d{4}-\d{4})')
myresult = re.search(phoneRegex, mystr)
print(myresult) # match='02-8101-8800'
```

第1組

第2組

```
myresult.group(0) # 取得全部號碼 '02-8101-8800'
myresult.group(1) # 取得區域號碼 '02'
myresult.group(2) # 取得電話號碼 '8101-8800'
```

```
# groups 方法
areaCode, mainCode = myresult.groups()
areaCode # '02'
mainCode # '8101-8800'
```

group 方法+跳脫字元

```
# group 方法+跳脫字元
mystr = '101大樓客服電話是 (02)-8101-8800'
phoneRegex = re.compile(r'(\d{2})-(\d{4}-\d{4})')
myresult = re.search(phoneRegex, mystr)
print(myresult) # None-->找不到???

phoneRegex = re.compile(r'(\(\d{2}\))- (\d{4}-\d{4})')
myresult = re.search(phoneRegex, mystr)
print(myresult)

myresult.group(0) # 取得全部號碼 '02-8101-8800'
myresult.group(1) # 取得區域號碼 '(02)'
myresult.group(2) # 取得電話號碼 '8101-8800'
```

Python demo

- 保留日期
- 保留字串

判斷式 if elif else

if 三種用法 if, elif, else

{ # case 1
 if 布林值:
 若布林值為 True，執行命令

{ # case 2
 if 布林值:
 若布林值為 True，執行命令
 else:
 若布林值為 False，執行命令

{ # case 3
 if 布林值一:
 若布林值一為 True，執行命令
 elif 布林值二:
 若布林值二為 True，執行命令
 ...
 else:
 若布林值一和二...都是 False，執行命令

elif

- Python 沒有 switch 的語法

```
# elif敘述
a = '+'

if a == '+':
    op = 'PLUS'
elif a == '-':
    op = 'MINUS'
else:
    op = 'UNKNOWN'

op
```

記得加冒號：

布林(Boolean)表示式 – and, or, not

```
# 沒有像C語言一樣，有switch的語法
# 布林表示式 - and, or, not

a = 1
b = 6
c = 9

if b >= a and b <= c:
    print('b is between a and c')

if not (b < a or c > c):
    print('b is still between a and c')
```

if 範例-測試所有輸入情形

```
# if 範例-測試所有輸入情形
mynameage = input('輸入姓名與年齡: ')

name = mynameage.split(',')[0]
age = mynameage.split(',')[1]

if name == 'Alan':
    print('Hi, Alan.')
elif age < 20:
    print('You are not Alan.') 
```



輸入姓名與年齡: abc, 30

TypeError: '<' not supported between instances of 'str' and 'int'

邏輯錯誤 (Logical Errors)

```
# if 範例 - age > 200 不會執行
name = 'RWEPA'
age = 300
if name == 'Alan':
    print('Hi, Alan.')
elif age < 20:
    print('You are not Alan.')
elif age > 100:
    print('You are not Alan. 大大')
elif age > 200:
    print('年齡異常')
# You are not Alan. 大大
```

age > 200
不會執行

迴圈 (Loops)

while 迴圈

In [1]:

```
....: name = ''  
....: while name != 'Alan Lee':  
....:     print('Please type your name.')  
....:     name = input()  
....: print('Thank you!')
```

Please type your name.

ABC

Please type your name.

ALAN

Please type your name.

Alan Lee

Thank you!

while 判斷式：

break, continue, pass

- break : 結束整個迴圈
- continue : 放棄本次迴圈，立即執行下一個迴圈
- pass : 不做任何事情，所有的程式都將繼續執行，可暫時備註。

while + break

```
# while + break
while True:
    print('Please type your name.')
    name = input()
    if name == 'Alan Lee':
        break
print('Thank you!')
```

while + break + continue

```
# while + break + continue
while True:
    print('Who are you?')
    name = input()
    if name != 'Alan':
        continue
    print('Hello, Alan. What is the password? (It is a fish.)')
    password = input()
    if password == 'swordfish':
        break
print('Access granted.')
```

for + range

範例1

```
In [1]: # 顯示list元素
```

```
In [2]: for i in [3, 4, 10, 25]:  
...:     print(i)
```

```
3
```

```
4
```

```
10
```

```
25
```

範例2

```
In [3]: # 顯示一個字元
```

```
In [4]: for c in "Hello":  
...:     print(c)
```

```
H
```

```
e
```

```
l
```

```
l
```

```
o
```

範例3

```
In [5]: # 顯示range 元素
```

```
In [6]: for i in range(1, 4):  
...:     print(i)
```

```
1
```

```
2
```

```
3
```

範例4

```
In [7]: for i in range(4, -2, -1):  
...:     print(i)
```

```
4
```

```
3
```

```
2
```

```
1
```

```
0
```

```
-1
```

零數值與非零數值判斷

```
# 零數值判斷，以下結果皆為 True
```

```
0 == False
```

```
0.0 == False
```

```
0.000 == False
```

```
'' == False
```

```
# 非零數值判斷
```

```
1 == True      # True
```

```
1.23 == True   # False
```

```
1.23 == False  # False
```

for + continue 迴圈練習



實作練習12

```
# 實作練習  
# 篩選出 List 的字串資料  
# 提示: 使用 if, for, append, continue 順序非固定  
# 結果 ['python', '123.45', 'RWEPA', 'R']  
mylist = [1, 3, "python", '123.45', "RWEPA", 100, "R"]
```

def 函數應用

函數 (Functions)

1. def
2. 函數名稱
3. (引數1, ...)
4. :
5. return 回傳值

```
# 回傳 a/b 之餘數
def remainder(a,b):
    1 q = a/b 2
    r = a - q*b
    3
    4
    return r
    5

# 如何修改回傳餘數2
a = remainder(42,5) # a = 2
print(a)

# 回傳 a/b 之商數與餘數
# 參考上例練習
def divide(a,b):
    q = a/b
    r = a - q*b
    6
    7
    return q,r

x,y = divide(42,5) # x = 8, y = 2
print(x)
print(y)
```





實作練習13

實作練習 def 函數

- 使用 def 建立計算2點之間的直線距離函數
- 函數名稱 distance
- 引數名稱 x_1, y_1, x_2, y_2
- 不可使用 math.sqrt 函數
- 輸入第1,2個點的座標值分別為 $(a, b), (c, d)$
- 輸出結果如下所示

```
In [3]: distance(a, b, c, d)  
Out[3]: 2.8284271247461903
```

Python: lambda 函數

- Python 提供了一個簡易的 function define : lambda , 用完即回收。可以實作出很簡單的 function (只處理一個運算式)。
 - `lambda param1, param2, ... : expression`
 - 其中的 `expression` 不能放 `assignment`，也就是這一行指令不能使用等號。
- 相當於使用 `def`
 - `def fun(param1, param2, ...) : return expression`

Python: lambda 函數 – 範例

```
In [1]:  
...: def funsum(x, y, z):  
...:     return x + y + z  
...: funsum(1, 2, 3)  
Out[1]: 6
```

```
In [2]:  
...: func2 = lambda x,y,z : x+y+z  
...: func2(1, 2, 3)  
Out[2]: 6
```

Python - 函數之參數

```
In [1]:  
...: def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):  
...:     print("-- This parrot wouldn't", action, end=' ')  
...:     print("if you put", voltage, "volts through it.")  
...:     print("-- Lovely plumage, the", type)  
...:     print("-- It's", state, "!")  
  
In [2]: parrot(1000)                                     # 1 positional argument  
-- This parrot wouldn't voom if you put 1000 volts through it.  
-- Lovely plumage, the Norwegian Blue  
-- It's a stiff !
```

參考: <https://docs.python.org/3/tutorial/controlflow.html#define-functions>

Python - 函數之參數 (續)

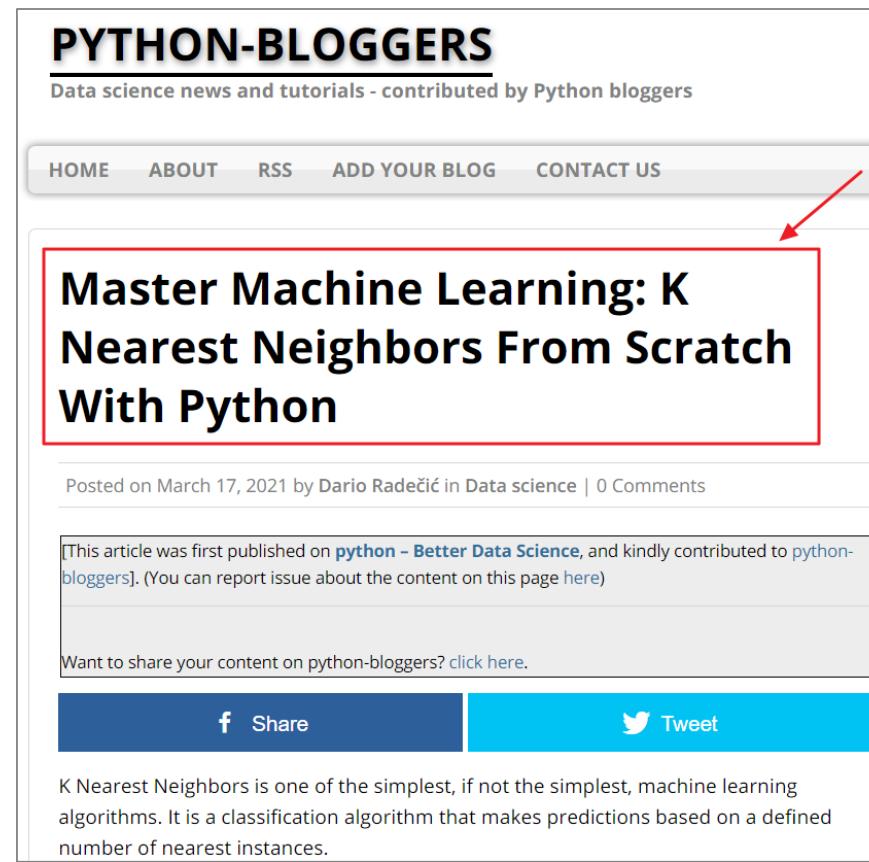
```
# 以下函數呼叫皆OK  
parrot(1000)                                     # 1 positional argument  
parrot(voltage=1000)                               # 1 keyword argument  
parrot(voltage=1000000, action='V00000M')          # 2 keyword arguments  
parrot(action='V00000M', voltage=1000000)           # 2 keyword arguments  
parrot('a million', 'bereft of life', 'jump')       # 3 positional arguments  
parrot('a thousand', state='pushing up the daisies') # 1 positional, 1 keyword  
  
parrot(100, 200)  
parrot(100, 200, 300)  
parrot(100, 200, 300, 400)
```

Python - 函數之參數 (續)

```
# 以下函數呼叫皆ERROR
parrot()                      # required argument missing
parrot(voltage=5.0, 'dead')    # non-keyword argument after a keyword argument
parrot(110, voltage=220)       # duplicate value for the same argument
parrot(actor='John Cleese')    # unknown keyword argument
parrot(100, 200, 300, 400, 500) # takes from 1 to 4 positional arguments but 5 were given
```

函數撰寫範例研究

- <https://python-bloggers.com/2021/03/master-machine-learning-k-nearest-neighbors-from-scratch-with-python/>



The screenshot shows a blog post on the Python-Bloggers website. The header reads "PYTHON-BLOGGERS" and "Data science news and tutorials - contributed by Python bloggers". The navigation bar includes links for HOME, ABOUT, RSS, ADD YOUR BLOG, and CONTACT US. A red arrow points to the title of the post, which is "Master Machine Learning: K Nearest Neighbors From Scratch With Python". The post was posted on March 17, 2021, by Dario Radečić in Data science with 0 comments. A note at the bottom of the post states: "[This article was first published on [python – Better Data Science](#), and kindly contributed to python-bloggers]. (You can report issue about the content on this page [here](#))". Below the post, there is a link to share the content on the Python-Bloggers platform. The main content of the post describes K Nearest Neighbors as one of the simplest machine learning algorithms, based on predictions from a defined number of nearest instances.

PYTHON-BLOGGERS
Data science news and tutorials - contributed by Python bloggers

HOME ABOUT RSS ADD YOUR BLOG CONTACT US

**Master Machine Learning: K
Nearest Neighbors From Scratch
With Python**

Posted on March 17, 2021 by Dario Radečić in Data science | 0 Comments

[This article was first published on [python – Better Data Science](#), and kindly contributed to python-bloggers]. (You can report issue about the content on this page [here](#))

Want to share your content on python-bloggers? [click here](#).

 Share  Tweet

K Nearest Neighbors is one of the simplest, if not the simplest, machine learning algorithms. It is a classification algorithm that makes predictions based on a defined number of nearest instances.

檔案處理

-
- open 開啟
 - read 讀取
 - write 寫入
 - close 關閉

檔案處理 – open 開啟

- myfile = open(filename, mode)

mode	feature
r	只能讀取, 例: df = open('data.csv', mode = 'r'), r 表示 read, mode預設為r
w	新建檔案寫入資料(檔案可以不存在, 若存在則清空)
a	將資料附加到舊檔案最後面位置
r+	讀取與寫入(檔案需存在且游標指在開頭)
w+	清空檔案內容, 寫入資料並可讀出(檔案如果不存在, 會自行新增)
a+	資料附加到原檔案後面, 可讀取資料

參考: <https://docs.python.org/3/library/functions.html#open>

檔案處理函數

函數	功能
open	開啟
write	寫入
read	讀取檔案的所有內容, 結果為一個長字串
readline	每次讀取1列, 結果為一個字串
readlines	讀取檔案的所有內容, 結果為一個串列
read(<i>n</i>)	從檔案讀取 <i>n</i> 個字元

os模組-建立與切換工作目錄

- getcwd
- mkdir
- chdir
- listdir

```
# os模組-建立與切換工作目錄
import os
os.getcwd()                      # 讀取工作目錄

dir = os.path.join("C:/", "mydata")
if not os.path.exists(dir):
    os.mkdir(dir)                 # 建立目錄

os.chdir(dir)                     # 變更工作目錄
os.listdir(os.getcwd())           # 顯示檔案名稱
```

檔案處理 方法1. 檔案的開啟/寫入/關閉

```
# 方法1. 檔案的開啟/寫入/關閉
f = open("coding.dat", "w") # Open a file for writing
f.write("Hello World\n")
f.write("Python\n")
f.write("R\n")
f.write("SQL\n")
f.write("Excel VBA\n")
f.close()

g = open("coding.dat", "a") # Open a file for appending
g.write(".NET")
g.close()
```

- open
- write
- close

coding.dat	
1	Hello World
2	Python
3	R
4	SQL
5	Excel VBA
6	.NET

檔案處理 方法2. 使用 with 區塊

```
# 方法2. 使用 with 區塊  
  
# with open("coding.dat", "r") as infile:  
  
# with 區塊特性  
# 檔案會自動關閉，可以不用撰寫 .close()  
# 即使出現以下狀況，檔案仍會自動關閉：  
# (1) 發生例外 (Exception)  
# (2) 執行 return, continue, break 等而跳出 with 區塊
```

with open() as xxx:

read 讀取全部資料

In [1]:

```
....: with open("coding.dat", "r") as infile:  
....:     mydata = infile.read()  
....:     print(type(mydata)) # str  
....:     print(mydata)
```

<class 'str'>

Hello World

Python

R

SQL

Excel VBA

.NET

readline 一次讀一列資料, while 迴圈-預設加入分隔列

```
In [1]: with open("coding.dat", "r") as infile:  
....:     while True:  
....:         line = infile.readline() # 一次讀一列資料  
....:         if not line:           # 所有資料讀取完畢  
....:             break  
....:         print(line)          # 預設加入分隔列
```

Hello World

Python

R

SQL

Excel VBA

.NET



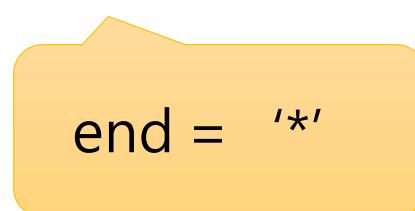
readline 一次讀一列資料, while 迴圈自訂分隔符號

In [1]:

```
.... with open("coding.dat", "r") as infile:  
....     while True:  
....         line = infile.readline()      # 一次讀一列資料  
....         if not line:                # 所有資料讀取完畢  
....             break  
....         print(line, end='*')        # end='*' 自訂分隔符號
```

Hello World

*Python
*R
*SQL
*Excel VBA
.NET

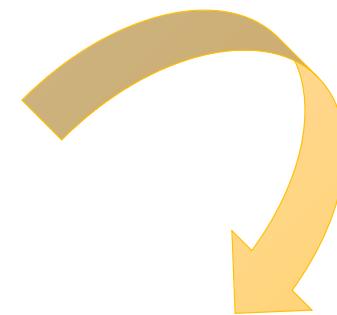
end = '*'


readlines 一次讀取所有資料

In [1]:

```
.... with open("coding.dat", "r") as infile:  
....     for line in infile.readlines(): # 一次讀取所有資料，再逐列處理  
....         print(line, end='')
```

Hello World
Python
R
SQL
Excel VBA
.NET



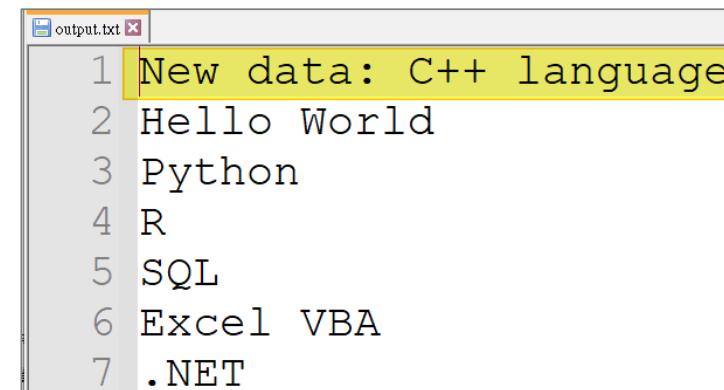
```
# readlines 簡化版本  
with open("coding.dat", "r") as infile:  
    for line in infile:  
        print(line, end='')
```

最終版

檔案處理範例

```
# 讀入 coding.dat, 加上額外一列, 寫入另一檔案 output.txt
with open("coding.dat", "r") as infile:
    data = infile.read()
data

with open("output.txt", "w") as outfile:
    outfile.write('New data: C++ language\n')
    outfile.write(data)
```



例外處理機制→確保程式執行

try:

敘述主體

except 例外名稱1:

處理方法

except 例外名稱2:

處理方法

...

except:

不是上述例外，即其他例外之執行敘述

else:

程式沒有例外之執行敘述

finally:

程式最後一定會執行敘述

try:

敘述主體

except 例外名稱 as 變數:

處理方法

例外處理機制

```

def exceptTest():
    try:
        x, y = eval(input("輸入2個數字, 中間以, 區隔: "))
        answer = x/y
        myanswer = " {} / {} = {} "
        print(myanswer.format(x, y, answer))
    except ZeroDivisionError:
        print('被除數不可為零 Division by zero!')
    except SyntaxError:
        print("二個數值中間要加上 , ")
    except:
        print("輸入有錯誤")
    else:
        print("<<<運算沒錯誤>>>")
    finally:
        print("程式已執行完成")

```

In [3]: exceptTest()

輸入2個數字, 中間以, 區隔: 12,5

$12/5 = 2.4$

<<<運算沒錯誤>>>

程式已執行完成

In [4]: exceptTest()

輸入2個數字, 中間以, 區隔: 12,0

被除數不可為零 Division by zero!

程式已執行完成

In [5]: exceptTest()

輸入2個數字, 中間以, 區隔: 12 5

二個數值中間要加上 ,

程式已執行完成

In [6]: exceptTest()

輸入2個數字, 中間以, 區隔: 12 rwepa

二個數值中間要加上 ,

程式已執行完成

例外階層架構

- 8. Errors and Exceptions
- <https://docs.python.org/3/tutorial/errors.html>
- Built-in Exceptions and Exception hierarchy
- <https://docs.python.org/3/library/exceptions.html>

The class hierarchy for built-in exceptions is:

```
BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
    +-- StopIteration
    +-- StopAsyncIteration
    +-- ArithmeticError
        +-- FloatingPointError
        +-- OverflowError
        +-- ZeroDivisionError
```

04. 日期時間資料, 檔案匯入pandas



日期時間資料

日期時間模組

- calendar - 輸出日曆並使用理想化的格里曆 (idealized Gregorian calendar)操作功能。
- time - 提供只需要時間操作功能。
- **datetime** - 提供用於操作日期和時間操作功能(重要模組)，提供 3 大類別。
 - datetime.date 類別：考慮是一個理想化的日期類別，儲存 year, month, day 屬性。
 - datetime.time 類別：考慮是一個理想化的時間，假設每天有 86,400 秒，沒有閏秒。儲存時、分、秒、微秒和 tzinfo(時區)。
 - datetime.datetime 類別：考慮是一個理想化的日期與時間類別。
 - 格里曆: https://en.wikipedia.org/wiki/Gregorian_calendar
 - 參考: <https://realpython.com/python-datetime/>

使用 datetime 模組

```
In [1]: # 使用 datetime 模組
```

```
In [2]: from datetime import date, time, datetime
```

```
In [3]: date(year=2021, month=8, day=10) # datetime.date(2021, 8, 10)
Out[3]: datetime.date(2021, 8, 10)
```

```
In [4]: time(hour=13, minute=30, second=31) # datetime.time(13, 30, 31)
Out[4]: datetime.time(13, 30, 31)
```

```
In [5]: datetime(year=2021, month=8, day=10, hour=13, minute=30, second=31)
Out[5]: datetime.datetime(2021, 8, 10, 13, 30, 31)
```

現在日期時間

```
In [6]: today = date.today()
```

```
In [7]: today  
Out[7]: datetime.date(2021, 7, 21)
```

```
In [8]: now = datetime.now()
```

```
In [9]: now  
Out[9]: datetime.datetime(2021, 7, 21, 20, 39, 22, 821125)
```

```
In [10]: current_time = time(now.hour, now.minute, now.second)
```

```
In [11]: current_time  
Out[11]: datetime.time(20, 39, 22)
```

```
In [12]: datetime.combine(today, current_time)  
Out[12]: datetime.datetime(2021, 7, 21, 20, 39, 22)
```

字串轉換為日期-fromisoformat

```
In [13]: # 字串轉換為日期-fromisoformat
```

```
In [14]: mystr = "2021-07-21"
```

```
In [15]: mydate = date.fromisoformat(mystr)
```

```
In [16]: mydate
```

```
Out[16]: datetime.date(2021, 7, 21)
```

```
In [17]: print(mydate)
```

```
2021-07-21
```

字串轉換為日期-strptime

- <https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior>

```
In [18]: # 字串轉換為日期-strptime
```

```
In [19]: date_string = "06-30-2021 12:34:56"
```

```
In [20]: format_string = "%m-%d-%Y %H:%M:%S"
```

```
In [21]: datetime.strptime(date_string, format_string)
```

```
Out[21]: datetime.datetime(2021, 6, 30, 12, 34, 56)
```

# Year	%Y (4位數值年)
# Month	%m (2位數值月)
# Date	%d (2位數字日)
# Hour	%H (2位數字24小時的時)
# Minute	%M (2位數字分)
# Second	%S (2位數字秒)

範例-日期計算

```
In [22]: # 範例-日期計算
```

```
In [23]: PYCON_DATE = datetime(year=2022, month=4, day=27, hour=8)
```

```
In [24]: countdown = PYCON_DATE - datetime.now()
```

```
In [25]: type(countdown) # datetime.timedelta
```

```
Out[25]: datetime.timedelta
```

```
In [26]: countdown
```

```
Out[26]: datetime.timedelta(days=279, seconds=40321, microseconds=35595)
```

```
In [27]: countdownDay = countdown.days
```

```
In [28]: txt = "距離 2022年4月27日 USA PyCon 還有 {} 天"
```

```
In [29]: print(txt.format(countdownDay))
```

```
距離 2022年4月27日 USA PyCon 還有 279 天
```

Time Zones 時區 - 使用 dateutil 模組

```
In [30]: # Time Zones 時區  
....: # 使用 dateutil 模組
```

```
In [31]: from dateutil import tz
```

```
In [32]: from datetime import datetime
```

```
In [33]: now = datetime.now(tz=tz.tzlocal())
```

```
In [34]: now
```

```
Out[34]: datetime.datetime(2021, 7, 21, 20, 51, 19, 326654, tzinfo=tzlocal())
```

Time Zones 時區

```
In [35]: now.tzname() # '台北標準時間'
```

```
Out[35]: '台北標準時間'
```

```
In [36]: my_tz = tz.gettz('Asia/Taipei')
```

```
In [37]: my_tz # tzfile('ROC')
```

```
Out[37]: tzfile('ROC')
```

```
In [38]: London_tz = tz.gettz("Europe/London")
```

```
In [39]: now = datetime.now(tz=London_tz)
```

```
In [40]: now
```

```
Out[40]: datetime.datetime(2021, 7, 21, 13, 51, 26, 305358, tzinfo=tzfile('GB-Eire'))
```

```
In [41]: now.tzname()
```

```
Out[41]: 'BST'
```

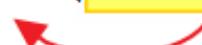
範例-轉換至美國紐約時區

```
In [42]: NYC_tz = tz.gettz('America/New_York')
```

```
In [43]: NYC_tz
```

```
Out[43]: tzfile('US/Eastern')
```

```
In [44]: now_utc = datetime.now(tz.tzutc())
```



```
In [45]: now_utc
```

```
Out[45]: datetime.datetime(2021, 7, 21, 12, 55, 28, 812304, tzinfo=tzutc())
```

```
In [46]: now_utc.astimezone(NYC_tz)
```

```
Out[46]: datetime.datetime(2021, 7, 21, 8, 55, 28, 812304, tzinfo=tzfile('US/Eastern'))
```

範例-計算程式執行時間

```
In [47]: from datetime import datetime
....: from numpy.random import default_rng
....:
....: # 開始計算時間
....: starttime = datetime.now()
....: starttime
Out[47]: datetime.datetime(2021, 7, 21, 21, 4, 12, 97529)
```

```
In [48]:
....: rng = default_rng()
....: vals = []
....: x = abs(rng.standard_normal(100000000))
....: x[0:3]
....: vals = x**0.5
....: vals[0:3]
....:
....: # 結束時間
....: endtime = datetime.now()
....: endtime
Out[48]: datetime.datetime(2021, 7, 21, 21, 4, 18, 575790)
```

```
In [49]: # 程式執行時間
```

```
In [50]: print(endtime - starttime)
0:00:06.478261
```

範例-時間運算

```
In [51]: from datetime import datetime, timedelta
```

```
In [52]: now = datetime.now()
```

```
In [53]: now
```

```
Out[53]: datetime.datetime(2021, 7, 21, 21, 6, 3, 976197)
```

```
In [54]: tomorrow = timedelta(days=+1) # 本地時間加1天
```

```
In [55]: now + tomorrow
```

```
Out[55]: datetime.datetime(2021, 7, 22, 21, 6, 3, 976197)
```

```
In [56]: delta = timedelta(days=+3, hours=-4) # # 本地時間加3天, 減4小時
```

```
In [57]: now + delta
```

```
Out[57]: datetime.datetime(2021, 7, 24, 17, 6, 3, 976197)
```



實作練習 – 檔案日期時間處理

- <https://www.kaggle.com/shawon10/web-log-dataset>

實作練習14

檔案名稱: weblog.csv

- 欄位個數:4
- 資料筆數:16007
- 練習使用 open, read, datetime, re 等處理技術(不可使用 pandas),計算下列3個時段的資料筆數:

06:00-14:00, 14:00-22:00, 22:00-06:00



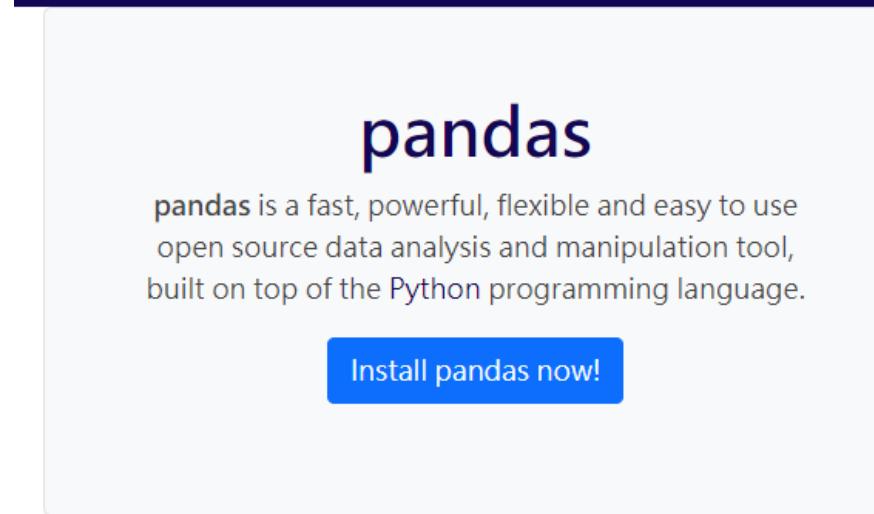
	A	B	C	D
1	IP	Time	URL	Status
2	10.128.2.1	[29/Nov/2017:06:58:55	GET /login.php HTTP/1.1	200
3	10.128.2.1	[29/Nov/2017:06:59:02	POST /process.php HTTP/1.1	302
4	10.128.2.1	[29/Nov/2017:06:59:03	GET /home.php HTTP/1.1	200
5	10.131.2.1	[29/Nov/2017:06:59:04	GET /js/vendor/moment.min.js HTTP/1.1	200
6	10.130.2.1	[29/Nov/2017:06:59:06	GET /bootstrap-3.3.7/js/bootstrap.js HTTP/1.1	200

檔案匯入 pandas

pandas 模組

- pandas 取名自 pan(el)-da(ta)-s
- pandas 提供資料讀取，資料整理，統計分析，繪圖等功能。
- pandas是一套使用Python語言開發的Python套件，完整包含 NumPy、Scipy和Matplotlib套件的功能.

<https://pandas.pydata.org/>



The main content area features the word "pandas" in large, bold, dark blue letters. Below it is a brief description: "pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language." A blue button labeled "Install pandas now!" is centered below the text.

Latest
version: 1.3.0

- What's new in 1.3.0
- Release date: Jul 02, 2021
- Documentation (web)
- Documentation (pdf)
- Download source code

Getting started

- Install pandas
- Getting started

Documentation

- User guide
- API reference
- Contributing to pandas
- Release notes

Community

- About pandas
- Ask a question
- Ecosystem

10 minutes to pandas

- https://pandas.pydata.org/docs/user_guide/10min.html

Tutorials

For a quick overview of pandas functionality, see [10 Minutes to pandas.](#)

Search the docs ...

10 minutes to pandas

Intro to data structures

Essential basic functionality

IO tools (text, CSV, HDF5, ...)

Indexing and selecting data

MultIndex / advanced indexing

Merge, join, concatenate and compare

Reshaping and pivot tables

pandas 資料結構

- **Series 物件：**

- 類似一維陣列的物件，擁有索引(index)與值的一維陣列.
- 可以將Series視為是2個陣列的組合，一個是類似索引的標籤，另一個是實際資料值
- `s = pd.Series(data, index=index)`
`data = array-like, Iterable, dict, scalar value`

- **DataFrame 物件：**

- 類似Excel 試算表的表格資料
- 具有索引的二維陣列
- 一個可以任易更改結構的表格，每一欄允許儲存不同資料型態的資料.

Series 物件

- `import numpy as np # Python Scientific Computing Library`
- `import pandas as pd # Python Data Analysis Library`

(1).Series - 使用 ndarray

```
# s = pd.Series(data, index=index)
# data 包括使用 array, Iterable, dict, scalar value
# 序列包括指標(Index) 與值(Value), 指標採用預設整數型態指標 0,1,2, ...
# (1).Series - 使用 ndarray
s = pd.Series(data = np.random.randn(5), index=["a", "b", "c", "d", "e"])
s
# a    -0.492604
# b    -0.073386
# c    -0.063632
# d     0.197128
# e     0.178333
# dtype: float64
type(s) # pandas.core.series.Series
```

結果類似 Excel 的一行資料

(2).Series - 使用 Iterable - 序列(tuple)

In [2]: # (2).Series - 使用 Iterable - 序列(tuple)

In [3]: s1 = pd.Series((1,3,5,np.nan,6,8))
....: s1

Out[3]:

```
0    1.0  
1    3.0  
2    5.0  
3    NaN  
4    6.0  
5    8.0  
dtype: float64
```

(3).Series - 使用 Iterable - 串列(List)

In [4]: # (3).Series - 使用 Iterable - 串列(List)

In [5]: s2 = pd.Series([1,3,5,np.nan,6,8])
...: s2

Out[5]:

0	1.0
1	3.0
2	5.0
3	NaN
4	6.0
5	8.0
dtype: float64	

相等(==) vs. 相同(is)

In [6]: s1 == s2 # equality 相等，比較每個元素是否相同，大部分使用此功能。

Out[6]:

```
0    True  
1    True  
2    True  
3   False  
4    True  
5    True  
dtype: bool
```



== 運算結果是 False

In [7]: s1 is s2 # identity 相同，比較二物件是否指向同一個記憶體

Out[7]: False

In [8]: id(s1)

Out[8]: 2824957766672

In [9]: id(s2) # 與id(s1) 不相等

Out[9]: 2824958104144

equality vs. identity

- identity - 使用 id 函數, 查看說明 help(id). 相同程式 id 結果,每次不一定相同.
- <https://realpython.com/python-is-identity-vs-equality/>

```
In [1]: a = 'Hello world'
```

```
In [2]: b = 'Hello world'
```

```
In [3]: a == b  
Out[3]: True
```

```
In [4]: a is b  
Out[4]: False
```

```
In [5]: id(a)  
Out[5]: 2166585399024
```

```
In [6]: id(b)  
Out[6]: 2166589303664
```

In [1]: # 整數 [-5 ~ 256] 會使用相同記憶體位址功能

```
In [2]: a = 256
...: b = 256
...: a == b    # True
...: a is b    # True
...: id(a)
```

Out[2]: 140734466311952

In [3]: id(b)

Out[3]: 140734466311952

```
In [4]: a = 1000
...: b = 1000
...: a == b    # True
...: a is b    # False
...: id(a)
```

Out[4]: 2085585143472

In [5]: id(b)

Out[5]: 2085585143312



In [2]: x1 = np.nan

In [3]: x2 = np.nan

In [4]: id(x1)

Out[4]: 2355752046768

In [5]: id(x2) # 與上面結果相同

Out[5]: 2355752046768

In [6]: x1 == x2 # False

Out[6]: False

In [7]: x1 is x2 # True

Out[7]: True

(4).Series - 使用 Iterable - 字典(Dict)

In [2]: # (4).Series - 使用 Iterable - 字典(Dict)

In [3]: # 在 pandas 模組之中, *NaN* 表示為 "not a number"

In [4]: x = {"x1": 1, "x2": 2, "a": np.nan, "b": 3, "c": 4}
....: c = pd.Series(x)
....: c

Out[4]:

```
x1    1.0
x2    2.0
a     NaN
b    3.0
c    4.0
dtype: float64
```

(5).Series - 使用 scalar value

In [5]: # (5).Series - 使用 scalar value

In [6]: pd.Series(999.0, index=["a", "b", "c", "d", "e"])

Out[6]:

```
a    999.0  
b    999.0  
c    999.0  
d    999.0  
e    999.0  
dtype: float64
```

可快速建立預設值

Series 使用 ndarray-like 操作

```
c  
# x1    1.0  
# x2    2.0  
# a     NaN  
# b     3.0  
# c     4.0  
# dtype: float64  
  
c[0]          # 1.0  
c[1]          # 2.0  
c[-1]         # 4.0  
c[:3]  
# x1    1.0  
# x2    2.0  
# a     NaN  
# dtype: float64
```

```
In [15]: c[c > c.median()]  
Out[15]:  
b    3.0  
c    4.0  
dtype: float64  
  
In [16]: c[[1, 3, 2]]  
Out[16]:  
x2    2.0  
b    3.0  
a    NaN  
dtype: float64  
  
In [17]: np.exp(c)  
Out[17]:  
x1    2.718282  
x2    7.389056  
a      NaN  
b    20.085537  
c    54.598150  
dtype: float64
```

Series.array vs. ExtensionArray

```
In [20]: c.array      # 將 series 轉換為 PandasArray
Out[20]:
<PandasArray>
[1.0, 2.0, nan, 3.0, 4.0]
Length: 5, dtype: float64

In [21]: c1 = c.to_numpy() # 將 series 轉換為 NumPy ndarray

In [22]: c1
Out[22]: array([ 1.,  2., nan,  3.,  4.])

In [23]: c2 = c.to_numpy

In [24]: c2
Out[24]:
<bound method IndexOpsMixin.to_numpy of x1    1.0
x2    2.0
a     NaN
b    3.0
c    4.0
dtype: float64>

In [25]: c1 == c2
Out[25]: array([False, False, False, False, False])

In [26]: c1 is c2
Out[26]: False
```

- Series.array 是 pandasExtensionArray.
- ExtensionArray 是包括一個或多個 numpy.ndarray 的 thin wrapper 類別

Series 使用 dict-like 操作

```
In [28]: c  
Out[28]:  
x1    1.0  
x2    2.0  
a      NaN  
b      3.0  
c      4.0  
dtype: float64
```

```
In [29]: c['x1']  
Out[29]: 1.0  
  
In [30]: c['a'] = np.pi
```

```
In [31]: 'x1' in c  
Out[31]: True
```

```
In [32]: c.get("a")  
Out[32]: 3.141592653589793
```

```
In [33]: c.get("e") # None
```

pandas 資料框(DataFrame)物件

方法1.建立指標與值,再合併為資料框

```
In [2]: # 方法1. 建立指標與值, 再合併為資料框
```

```
In [3]: # 步驟1-建立 DatetimeIndex 物件
```

```
In [4]: dates = pd.date_range('20210801', periods=6) # 日期指標
```

```
In [5]: dates
```

```
Out[5]:
```

```
DatetimeIndex(['2021-08-01', '2021-08-02', '2021-08-03', '2021-08-04',
                 '2021-08-05', '2021-08-06'],
                dtype='datetime64[ns]', freq='D')
```

```
In [6]: type(dates)
```

```
Out[6]: pandas.core.indexes.datetimes.DatetimeIndex
```

pd.DataFrame()

In [7]: # 步驟2-建立 DataFrame

In [8]: # 欄位名稱: A, B, C, D

In [9]: df1 = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
....: df1

Out[9]:

	A	B	C	D
2021-08-01	0.338935	1.610452	-1.534346	-1.008859
2021-08-02	-0.378642	-0.832767	-1.289297	1.327141
2021-08-03	0.706616	-1.376127	0.695725	0.446339
2021-08-04	0.267140	1.105310	0.072173	0.942306
2021-08-05	0.376187	0.220437	0.511613	0.828549
2021-08-06	0.414686	1.277009	0.614274	-0.324311

In [10]: type(df1)

Out[10]: pandas.core.frame.DataFrame

1

2

3

方法2. 使用字典建立資料框

```
In [11]: df2 = pd.DataFrame({ 'A' : 1.,
...:     'B' : pd.Timestamp('20210801'),
...:     'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
...:     'D' : np.array([3] * 4,dtype='int32'),
...:     'E' : pd.Categorical(["test","train","test","train"]),
...:     'F' : 'foo' })
...: df2
```

Out[11]:

	A	B	C	D	E	F
0	1.0	2021-08-01	1.0	3	test	foo
1	1.0	2021-08-01	1.0	3	train	foo
2	1.0	2021-08-01	1.0	3	test	foo
3	1.0	2021-08-01	1.0	3	train	foo

In [12]:

```
...: df2.dtypes
```

Out[12]:

A	float64
B	datetime64[ns]
C	float32
D	int32
E	category
F	object
	dtype: object

方法3. 使用 list of dicts 建立資料框

In [13]: # 預設指標

```
In [14]: mydata = [{"a": 1, "b": 2}, {"a": 5, "b": 10, "c": 20}]\n....: df3 = pd.DataFrame(mydata)\n....: df3
```

Out[14]:

	a	b	c
0	1	2	NaN
1	5	10	20.0

In [15]: # 客製化指標

```
In [16]: df4 = pd.DataFrame(mydata, index=["first", "second"])\n....: df4
```

Out[16]:

	a	b	c
first	1	2	NaN
second	5	10	20.0

方法4. 使用 dict of tuples 建立資料框

- 使用 tuples dictionary, 可建立 MultiIndexed dataframe (階層式指標資料框)

```
In [17]: df5 = pd.DataFrame(  
    ....:     {  
    ....:         ("a", "b"): {("A", "B"): 1, ("A", "C"): 2},  
    ....:         ("a", "a"): {("A", "C"): 3, ("A", "B"): 4},  
    ....:         ("a", "c"): {("A", "B"): 5, ("A", "C"): 6},  
    ....:         ("b", "a"): {("A", "C"): 7, ("A", "B"): 8},  
    ....:         ("b", "b"): {("A", "D"): 10, ("A", "B"): 11},  
    ....:     }  
    ....: )  
....: df5
```

Out[17]:

		a	b	a	c	a	b
A	B	1.0	4.0	5.0	8.0	11.0	
C	2.0	3.0	6.0	7.0	NaN		
D	NaN	NaN	NaN	NaN	NaN	10.0	

方法5. 使用 list of dataclasses 建立資料框

- list of dataclasses 類似於 list of dictionaries

```
In [1]: import numpy as np # Python Scientific Computing Library
```

```
In [2]: import pandas as pd # Python Data Analysis Library
```

```
In [3]: from dataclasses import make_dataclass
```

pandas 1.1.0 新增功能
資料類別 (Data Classes)

```
In [4]: Mydata = make_dataclass("Stations", [("x", int), ("y", int)])
```

```
In [5]: Mydata
```

1

```
Out[5]: types.Stations
```

2

```
In [6]: df6 = pd.DataFrame([Mydata(0, 0), Mydata(0, 3), Mydata(2, 3), Mydata(1, 2)])
```

```
In [7]: df6
```

```
Out[7]:
```

	x	y
0	0	0
1	0	3
2	2	3
3	1	2

排序

A
↓

從 A 到 Z 排序(S)

遞增(預設值)

Z
↓

從 Z 到 A 排序(O)

遞減

(1).排序 sort_index

- 當對資料 "列" 進行排序時，axis必須設置為0.
- df.sort(["A"]) 新版不支援 sort, 改用 sort_values 或 sort_index
- ascending =**False**, 即遞增是**FALSE**, 表示**遞減是TRUE**, 結果為D,C,B,A

```
In [10]: df1.sort_index(axis=1, ascending=False)  
Out[10]:
```

	D	C	B	A
2021-08-01	-0.928378	1.885510	1.521155	0.695556
2021-08-02	-1.394514	1.209076	-0.733466	0.740674
2021-08-03	-0.237016	-0.642413	0.538803	0.903706
2021-08-04	0.649434	-0.741611	0.932843	-0.146590
2021-08-05	-0.899328	-0.933120	-0.800698	-0.347963
2021-08-06	-1.790927	0.159388	0.827578	0.080708

- 0表示橫列指標
- 1表示直行指標

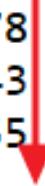
(2).排序 sort_values

- 依照 B 欄大小, 由小至大排序 (預設值是遞增)

```
In [12]: df1.sort_values(by='B')
```

```
Out[12]:
```

	A	B	C	D
2021-08-05	-0.347963	-0.800698	-0.933120	-0.899328
2021-08-02	0.740674	-0.733466	1.209076	-1.394514
2021-08-03	0.903706	0.538803	-0.642413	-0.237016
2021-08-06	0.080708	0.827578	0.159388	-1.790927
2021-08-04	-0.146590	0.932843	-0.741611	0.649434
2021-08-01	0.695556	1.521155	1.885510	-0.928378



(2).排序 sort_values (續)

- 依照 B 欄大小, 改為由大至小排序 (遞減)

```
In [13]: df1.sort_values(by='B', ascending = False)  
Out[13]:
```

	A	B	C	D
2021-08-01	0.695556	1.521155	1.885510	-0.928378
2021-08-04	-0.146590	0.932843	-0.741611	0.649434
2021-08-06	0.080708	0.827578	0.159388	-1.790927
2021-08-03	0.903706	0.538803	-0.642413	-0.237016
2021-08-02	0.740674	-0.733466	1.209076	-1.394514
2021-08-05	-0.347963	-0.800698	-0.933120	-0.899328

(2).排序 sort_values - nan

- 依照 B 欄大小, 將 nan 排在最前面

```
In [25]: df1.sort_values(by='A')
```

```
Out[25]:
```

	A	B	C	D
2021-08-05	-0.347963	-0.800698	-0.933120	-0.899328
2021-08-04	-0.146590	0.932843	-0.741611	0.649434
2021-08-06	0.080708	0.827578	0.159388	-1.790927
2021-08-01	0.695556	1.521155	1.885510	-0.928378
2021-08-02	0.740674	-0.733466	1.209076	-1.394514
2021-08-03	NaN	0.538803	-0.642413	-0.237016

```
In [26]: df1.sort_values(by='A', na_position = 'first')
```

```
Out[26]:
```

	A	B	C	D
2021-08-03	NaN	0.538803	-0.642413	-0.237016
2021-08-05	-0.347963	-0.800698	-0.933120	-0.899328
2021-08-04	-0.146590	0.932843	-0.741611	0.649434
2021-08-06	0.080708	0.827578	0.159388	-1.790927
2021-08-01	0.695556	1.521155	1.885510	-0.928378
2021-08-02	0.740674	-0.733466	1.209076	-1.394514

pandas 排序

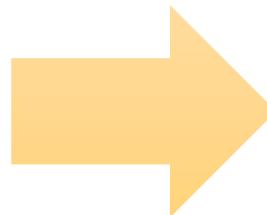


實作練習15



- 使用 mydf 進行A欄位遞增, B欄位遞減排序
- 排序後, 須一併更改 mydf 內容

	A	B	C
0	1	10	aa
1	2	24	bb
2	2	26	cc
3	4	9	dd
4	2	29	aa



	A	B	C
0	1	10	aa
4	2	29	aa
2	2	26	cc
1	2	24	bb
3	4	9	dd

資料列,行選取

選取行

```
In [1]: import numpy as np  
...: import pandas as pd  
...: np.random.seed(123)  
...: dates = pd.date_range('20210801', periods=6) # 日期指標  
...: df1 = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))  
...: df1  
Out[1]:
```

	A	B	C	D
2021-08-01	-1.085631	0.997345	0.282978	-1.506295
2021-08-02	-0.578600	1.651437	-2.426679	-0.428913
2021-08-03	1.265936	-0.866740	-0.678886	-0.094709
2021-08-04	1.491390	-0.638902	-0.443982	-0.434351
2021-08-05	2.205930	2.186786	1.004054	0.386186
2021-08-06	0.737369	1.490732	-0.935834	1.175829

```
In [2]:  
...: df1['A']  
...: df1.A
```

```
Out[2]:  
2021-08-01    -1.085631  
2021-08-02    -0.578600  
2021-08-03     1.265936  
2021-08-04     1.491390  
2021-08-05     2.205930  
2021-08-06     0.737369  
  
Freq: D, Name: A, dtype: float64
```

二種方法-

- df1['A'], df1[['A', 'B']]
- df1.A

選取列

- 選取列, `df[1:4]`選取第1至第3列($4-1=3$), 此功能與 R 不同.
- R: `df[1:4]` 表示選取第1至第4行

```
In [3]:  
....: df1  
....: df1[1:4]
```



		A	B	C	D
0	2021-08-01	-1.085631	0.997345	0.282978	-1.506295
1	2021-08-02	-0.578600	1.651437	-2.426679	-0.428913
2	2021-08-03	1.265936	-0.866740	-0.678886	-0.094709
3	2021-08-04	1.491390	-0.638902	-0.443982	-0.434351
4	2021-08-05	2.205930	2.186786	1.004054	0.386186
5	2021-08-06	0.737369	1.490732	-0.935834	1.175829

Out[3]:

	A	B	C	D
2021-08-02	-0.578600	1.651437	-2.426679	-0.428913
2021-08-03	1.265936	-0.866740	-0.678886	-0.094709
2021-08-04	1.491390	-0.638902	-0.443982	-0.434351

使用 loc, iloc

In [4]: # 使用 Loc

In [5]: df1.loc[:, ['A', 'B']]

Out[5]:

	A	B
2021-08-01	-1.085631	0.997345
2021-08-02	-0.578600	1.651437
2021-08-03	1.265936	-0.866740
2021-08-04	1.491390	-0.638902
2021-08-05	2.205930	2.186786
2021-08-06	0.737369	1.490732

- loc[列, 行]
- 如果列的位置是：
表示選取所有列

In [6]: # 使用 iloc

In [7]: df1.iloc[2] # 指標為第2列

Out[7]:

A 1.265936
B -0.866740
C -0.678886
D -0.094709

Name: 2021-08-03 00:00:00, dtype: float64

iloc[,] iloc[, :]

In [8]: df1.iloc[2:4,]

Out[8]:

	A	B	C	D
2021-08-03	1.265936	-0.866740	-0.678886	-0.094709
2021-08-04	1.491390	-0.638902	-0.443982	-0.434351

In [9]: df1.iloc[2:4, :]

Out[9]:

	A	B	C	D
2021-08-03	1.265936	-0.866740	-0.678886	-0.094709
2021-08-04	1.491390	-0.638902	-0.443982	-0.434351

iloc[:,] 逗號旁加上冒號:

```
In [11]: df1.iloc[:, 2] # OK
Out[11]:
2021-08-01    0.282978
2021-08-02   -2.426679
2021-08-03   -0.678886
2021-08-04   -0.443982
2021-08-05    1.004054
2021-08-06   -0.935834
Freq: D, Name: C, dtype: float64
```

```
In [12]: df1.iloc[:, 2:4] # OK
Out[12]:
          C          D
2021-08-01  0.282978 -1.506295
2021-08-02 -2.426679 -0.428913
2021-08-03 -0.678886 -0.094709
2021-08-04 -0.443982 -0.434351
2021-08-05  1.004054  0.386186
2021-08-06 -0.935834  1.175829
```

	A	B	C	D	
0	2021-08-01	-1.085631	0.997345	0.282978	-1.506295
1	2021-08-02	-0.578600	1.651437	-2.426679	-0.428913
2	2021-08-03	1.265936	-0.866740	-0.678886	-0.094709
3	2021-08-04	1.491390	-0.638902	-0.443982	-0.434351
4	2021-08-05	2.205930	2.186786	1.004054	0.386186
5	2021-08-06	0.737369	1.490732	-0.935834	1.175829

少了冒號

```
In [10]: df1.iloc[, 2] # ERROR
          File "<ipython-input-10-5ff903a3b603>", line 1
                    df1.iloc[, 2]      # ERROR
                                ^
SyntaxError: invalid syntax
```

Boolean Indexing 邏輯值(條件式)資料選取

```
In [2]: df1.loc[dates[2]]  
Out[2]:  
A    1.265936  
B   -0.866740  
C   -0.678886  
D   -0.094709  
Name: 2021-08-03 00:00:00, dtype: float64
```

```
In [3]: df1.loc['20210803']  
Out[3]:  
A    1.265936  
B   -0.866740  
C   -0.678886  
D   -0.094709  
Name: 2021-08-03 00:00:00, dtype: float64
```

邏輯值(條件式)資料選取 (續)

```
In [4]: df1.loc['20210803', ['A', 'B']]
```

```
Out[4]:
```

```
A    1.265936
```

```
B   -0.866740
```

```
Name: 2021-08-03 00:00:00, dtype: float64
```

```
In [5]: df1.loc['20210802':'20210804', ['A', 'B']]
```

```
Out[5]:
```

	A	B
2021-08-02	-0.578600	1.651437
2021-08-03	1.265936	-0.866740
2021-08-04	1.491390	-0.638902

```
In [6]: df1.iloc[[1,2,4], [0,2]] # 選取不連續範圍
```

```
Out[6]:
```

	A	C
2021-08-02	-0.578600	-2.426679
2021-08-03	1.265936	-0.678886
2021-08-05	2.205930	1.004054

邏輯值(條件式)資料選取 (續)

```
In [7]: df1.iloc[2,2]  
Out[7]: -0.6788861516220543
```

```
In [8]: df1.iat[2,2]  
Out[8]: -0.6788861516220543
```

```
In [9]: df1[df1 > 1.5]  
Out[9]:
```

	A	B	C	D
2021-08-01	NaN	NaN	NaN	NaN
2021-08-02	NaN	1.651437	NaN	NaN
2021-08-03	NaN	NaN	NaN	NaN
2021-08-04	NaN	NaN	NaN	NaN
2021-08-05	2.20593	2.186786	NaN	NaN
2021-08-06	NaN	NaN	NaN	NaN

```
In [10]: df1[df1.A > 1.5]  
Out[10]:
```

	A	B	C	D
2021-08-05	2.20593	2.186786	1.004054	0.386186

使用 .isin - 範例1

```
In [11]: # 使用 .isin - 範例1

In [12]: df1[df1.index.isin(['2021-08-02', '2021-08-04'])]
Out[12]:
          A          B          C          D
2021-08-02 -0.57860  1.651437 -2.426679 -0.428913
2021-08-04  1.49139 -0.638902 -0.443982 -0.434351
```

使用 .isin -範例2

```
In [13]: # 使用 .isin - 範例2
```

```
In [14]: df2 = df1.copy()
```

```
In [15]: df2['E'] = ['one', 'one', 'two', 'three', 'four', 'three']
```

```
In [16]: df2
```

Out[16]:

	A	B	C	D	E
2021-08-01	-1.085631	0.997345	0.282978	-1.506295	one
2021-08-02	-0.578600	1.651437	-2.426679	-0.428913	one
2021-08-03	1.265936	-0.866740	-0.678886	-0.094709	two
2021-08-04	1.491390	-0.638902	-0.443982	-0.434351	three
2021-08-05	2.205930	2.186786	1.004054	0.386186	four
2021-08-06	0.737369	1.490732	-0.935834	1.175829	three

```
In [17]: df2[df2['E'].isin(['two', 'four'])]
```

Out[17]:

	A	B	C	D	E
2021-08-03	1.265936	-0.866740	-0.678886	-0.094709	two
2021-08-05	2.205930	2.186786	1.004054	0.386186	four

Missing Data 遺漏值 NaN

Missing Data 遺漏值

- Python: np.NaN (np.nan)
- R: NA

In [20]: df3

Out[20]:

	A	B	C	D	E
2021-08-01	-1.085631	0.997345	0.282978	-1.506295	1.0
2021-08-02	-0.578600	1.651437	-2.426679	-0.428913	1.0
2021-08-03	1.265936	-0.866740	-0.678886	-0.094709	NaN
2021-08-04	1.491390	-0.638902	-0.443982	-0.434351	NaN

In [21]: pd.isnull(df3)

Out[21]:

	A	B	C	D	E
2021-08-01	False	False	False	False	False
2021-08-02	False	False	False	False	False
2021-08-03	False	False	False	False	True
2021-08-04	False	False	False	False	True

In [22]: df3.dropna(how='any')

Out[22]:

	A	B	C	D	E
2021-08-01	-1.085631	0.997345	0.282978	-1.506295	1.0
2021-08-02	-0.578600	1.651437	-2.426679	-0.428913	1.0

In [23]: df3.fillna(value=999)

Out[23]:

	A	B	C	D	E
2021-08-01	-1.085631	0.997345	0.282978	-1.506295	1.0
2021-08-02	-0.578600	1.651437	-2.426679	-0.428913	1.0
2021-08-03	1.265936	-0.866740	-0.678886	-0.094709	999.0
2021-08-04	1.491390	-0.638902	-0.443982	-0.434351	999.0

群組

Python -群組

- <https://github.com/rwepa/DataDemo/blob/master/Cars93.csv>

In [1]:

```
...: import pandas as pd
...: df = pd.read_csv('C:/rdata/Cars93.csv')
...
...: df = df[['Manufacturer', 'Price', 'AirBags', 'Horsepower', 'Origin']]
```

In [2]: df.head()

Out[2]:

	Manufacturer	Price	AirBags	Horsepower	Origin
0	Acura	15.9	None	140	non-USA
1	Acura	33.9	Driver & Passenger	200	non-USA
2	Audi	29.1	Driver only	172	non-USA
3	Audi	37.7	Driver & Passenger	172	non-USA
4	BMW	30.0	Driver only	208	non-USA

Python – groupby {pandas}

```
In [3]: df_AirBags = df.groupby('AirBags')
...: type(df_AirBags)
```

```
Out[3]: pandas.core.groupby.generic.DataFrameGroupBy
```

```
In [4]: print(df_AirBags.groups)
{'Driver & Passenger': [1, 3, 10, 13, 19, 20, 29, 40, 42, 49, 50, 51, 58, 74, 76, 92],
'Driver only': [2, 4, 5, 6, 7, 8, 9, 12, 17, 18, 21, 23, 24, 25, 26, 27, 33, 34, 35,
36, 37, 39, 41, 47, 48, 54, 56, 57, 59, 62, 63, 64, 66, 68, 70, 77, 78, 81, 83, 84,
85, 86, 91], 'None': [0, 11, 14, 15, 16, 22, 28, 30, 31, 32, 38, 43, 44, 45, 46, 52,
53, 55, 60, 61, 65, 67, 69, 71, 72, 73, 75, 79, 80, 82, 87, 88, 89, 90]}
```

Python - 群組 2個維度

In [5]:

```
.... df_AirBagsOrigin = df.groupby(['AirBags', 'Origin'])  
....  
.... # 群組大小  
.... df_AirBagsOrigin.size()
```

Out[5]:

AirBags	Origin	
Driver & Passenger	USA	9
	non-USA	7
Driver only	USA	23
	non-USA	20
None	USA	16
	non-USA	18

dtype: int64

群組大小 size()

In [6]:

```
...: df_AirBagsOrigin.size()
```

Out[6]:

AirBags	Origin	
Driver & Passenger	USA	9
	non-USA	7
Driver only	USA	23
	non-USA	20
None	USA	16
	non-USA	18

dtype: int64

篩選群組 get_group()

In [7]:

```
....: df_AirBags.get_group('Driver & Passenger')
```

Out[7]:

	Manufacturer	Price	AirBags	Horsepower	Origin
1	Acura	33.9	Driver & Passenger	200	non-USA
3	Audi	37.7	Driver & Passenger	172	non-USA
10	Cadillac	40.1	Driver & Passenger	295	USA
13	Chevrolet	15.1	Driver & Passenger	160	USA
19	Chrylser	18.4	Driver & Passenger	153	USA
20	Chrysler	15.8	Driver & Passenger	141	USA
29	Eagle	19.3	Driver & Passenger	214	USA
40	Honda	19.8	Driver & Passenger	160	non-USA
42	Honda	17.5	Driver & Passenger	140	non-USA
49	Lexus	35.2	Driver & Passenger	225	non-USA
50	Lincoln	34.3	Driver & Passenger	160	USA
51	Lincoln	36.1	Driver & Passenger	210	USA
58	Mercedes-Benz	61.9	Driver & Passenger	217	non-USA
74	Pontiac	17.7	Driver & Passenger	160	USA
76	Pontiac	24.4	Driver & Passenger	170	USA
92	Volvo	26.7	Driver & Passenger	168	non-USA

群組 總和 sum, 平均值 mean

In [8]:

```
....: df_AirBags.sum()
```

Out[8]:

AirBags	Price	Horsepower
Driver & Passenger	453.9	2945
Driver only	912.6	6527
None	447.9	3904

In [9]:

```
....: df_AirBags.mean()
```

Out[9]:

AirBags	Price	Horsepower
Driver & Passenger	28.368750	184.062500
Driver only	21.223256	151.790698
None	13.173529	114.823529

群組 計算 agg()

In [10]:

```
...: df_AirBags.agg('min')
```

Out[10]:

	Manufacturer	Price	Horsepower	Origin
AirBags				
Driver & Passenger	Acura	15.1	140	USA
Driver only	Audi	9.8	82	USA
None	Acura	7.4	55	USA

In [11]:

```
...: df_AirBags.agg('max')
```

Out[11]:

	Manufacturer	Price	Horsepower	Origin
AirBags				
Driver & Passenger	Volvo	61.9	295	non-USA
Driver only	Volvo	47.9	300	non-USA
None	Volkswagen	23.3	200	non-USA

摘要

Python – describe {pandas}

```
In [1]: import pandas as pd  
....: df = pd.read_csv('C:/mydata/Cars93.csv')  
....: df
```

Out[1]:

	Manufacturer	Model	Type	...	Weight	Origin	Make	
0	Acura	Integra	Small	...	2705	non-USA	Acura	Integra
1	Acura	Legend	Midsize	...	3560	non-USA	Acura	Legend
2	Audi	90	Compact	...	3375	non-USA	Audi	90
3	Audi	100	Midsize	...	3405	non-USA	Audi	100
4	BMW	535i	Midsize	...	3640	non-USA	BMW	535i
..
88	Volkswagen	Eurovan	Van	...	3960	non-USA	Volkswagen	Eurovan
89	Volkswagen	Passat	Compact	...	2985	non-USA	Volkswagen	Passat
90	Volkswagen	Corrado	Sporty	...	2810	non-USA	Volkswagen	Corrado
91	Volvo	240	Compact	...	2985	non-USA	Volvo	240
92	Volvo	850	Midsize	...	3245	non-USA	Volvo	850

[93 rows x 27 columns]

df.dtypes

```
In [2]: df.dtypes # object: 字串, float64: 含小數點數值
```

```
Out[2]:
```

Manufacturer	object
Model	object
Type	object
Min.Price	float64
Price	float64
Max.Price	float64
MPG.city	int64
MPG.highway	int64
AirBags	object
DriveTrain	object
Cylinders	object
EngineSize	float64
Horsepower	int64
RPM	int64
Rev.per.mile	int64
Man.trans.avail	object
Fuel.tank.capacity	float64
Passengers	int64
Length	int64
Wheelbase	int64
Width	int64
Turn.circle	int64
Rear.seat.room	float64
Luggage.room	float64
Weight	int64
Origin	object
Make	object
	dtype: object

df.describe(include='all')

```
In [3]: df.describe() # 無法顯示所有欄位
```

```
Out[3]:
```

	Min.Price	Price	...	Luggage.room	Weight
count	93.000000	93.000000	...	82.000000	93.000000
mean	17.125806	19.509677	...	13.890244	3072.903226
std	8.746029	9.659430	...	2.997967	589.896510
min	6.700000	7.400000	...	6.000000	1695.000000
25%	10.800000	12.200000	...	12.000000	2620.000000
50%	14.700000	17.700000	...	14.000000	3040.000000
75%	20.300000	23.300000	...	15.000000	3525.000000
max	45.400000	61.900000	...	22.000000	4105.000000

[8 rows x 18 columns]

```
In [4]: df.describe(include='all') # 顯示所有欄位
```

```
Out[4]:
```

	Manufacturer	Model	Type	...	Weight	Origin	Make
count	93	93	93	...	93.000000	93	93
unique	32	93	6	...	NaN	2	93
top	Chevrolet	Spirit	Midsize	...	NaN	USA	Mercury Capri
freq	8	1	22	...	NaN	48	1
mean	NaN	NaN	NaN	...	3072.903226	NaN	NaN
std	NaN	NaN	NaN	...	589.896510	NaN	NaN
min	NaN	NaN	NaN	...	1695.000000	NaN	NaN
25%	NaN	NaN	NaN	...	2620.000000	NaN	NaN
50%	NaN	NaN	NaN	...	3040.000000	NaN	NaN
75%	NaN	NaN	NaN	...	3525.000000	NaN	NaN
max	NaN	NaN	NaN	...	4105.000000	NaN	NaN

[11 rows x 27 columns]

None 沒有限制

```
In [5]: pd.set_option('display.max_rows', None, 'display.max_columns', None) # None 沒有限制
```

```
In [6]: df.describe()
```

```
Out[6]:
```

	Min.Price	Price	Max.Price	MPG.city	MPG.highway	EngineSize	\
count	93.000000	93.000000	93.000000	93.000000	93.000000	93.000000	
mean	17.125806	19.509677	21.898925	22.365591	29.086022	2.667742	
std	8.746029	9.659430	11.030457	5.619812	5.331726	1.037363	
min	6.700000	7.400000	7.900000	15.000000	20.000000	1.000000	
25%	10.800000	12.200000	14.700000	18.000000	26.000000	1.800000	
50%	14.700000	17.700000	19.600000	21.000000	28.000000	2.400000	
75%	20.300000	23.300000	25.300000	25.000000	31.000000	3.300000	
max	45.400000	61.900000	80.000000	46.000000	50.000000	5.700000	

	Horsepower	RPM	Rev.per.mile	Fuel.tank.capacity	Passengers	\
count	93.000000	93.000000	93.000000	93.000000	93.000000	
mean	143.827957	5280.645161	2332.204301	16.664516	5.086022	
std	52.374410	596.731690	496.506525	3.279370	1.038979	
min	55.000000	3800.000000	1320.000000	9.200000	2.000000	
25%	103.000000	4800.000000	1985.000000	14.500000	4.000000	
50%	140.000000	5200.000000	2340.000000	16.400000	5.000000	
75%	170.000000	5750.000000	2565.000000	18.800000	6.000000	
max	300.000000	6500.000000	3755.000000	27.000000	8.000000	

	Length	Wheelbase	Width	Turn.circle	Rear.seat.room	\
count	93.000000	93.000000	93.000000	93.000000	91.000000	
mean	183.204301	103.946237	69.376344	38.956989	27.829670	

顯示所有資料(全部列, 全部行)

In [7]: df

Out[7]:

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	\
0	Acura	Integra	Small	12.9	15.9	18.8	
1	Acura	Legend	Midsize	29.2	33.9	38.7	
2	Audi	90	Compact	25.9	29.1	32.3	
3	Audi	100	Midsize	30.8	37.7	44.6	
4	BMW	535i	Midsize	23.7	30.0	36.2	
5	Buick	Century	Midsize	14.2	15.7	17.3	
6	Buick	LeSabre	Large	19.9	20.8	21.7	
7	Buick	Roadmaster	Large	22.6	23.7	24.9	
8	Buick	Riviera	Midsize	26.3	26.3	26.3	
9	Cadillac	DeVille	Large	33.0	34.7	36.3	
10	Cadillac	Seville	Midsize	37.5	40.1	42.7	

檔案匯入 pandas

pandas - IO tools

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	Fixed-Width Text File	read_fwf	
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	OpenDocument	read_excel	
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	ORC Format	read_orc	
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata (統計)	read_stata	to_stata
binary	SAS (統計)	read_sas	
binary	SPSS (統計)	read_spss	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL (資料庫)	read_sql	to_sql
SQL	Google BigQuery	read_gbq	to_gbq

讀取 CSV(包括Tab區隔)

pandas-資料載入及匯出.py

```
# 讀入 CSV 檔案
import pandas as pd
print(pd.__version__) # 1.2.4

# 下載 marketing.csv 至 C:\pythondata\data 資料夾
# https://github.com/rwepa/DataDemo/blob/master/marketing.csv

# 讀入資料
marketing = pd.read_csv('C:/mydata/marketing.csv')
marketing # 200*4

# 資料摘要
marketing.describe()
```

- pd.read_csv()
- sep = ';'
- sep = '\t'
- sep = '|'

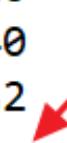
顯示結果

In [4]: marketing

Out[4]:

	youtube	facebook	newspaper	sales
0	276.12	45.36	83.04	26.52
1	53.40	NaN	54.12	12.48
2	20.64	55.08	83.16	11.16
3	181.80	49.56	70.20	22.20
4	216.96	12.96	70.08	15.48
..
195	45.84	4.44	16.56	9.12
196	113.04	5.88	9.72	11.64
197	212.40	11.16	7.68	15.36
198	340.32	50.40	79.44	30.60
199	278.52	10.32	10.44	16.08

[200 rows x 4 columns]



pd.describe

In [5]: marketing.describe()

Out[5]:

	youtube	facebook	newspaper	sales
count	200.000000	199.000000	200.000000	200.000000
mean	176.451000	27.820101	36.664800	16.827000
std	103.025084	17.808410	26.134345	6.260948
min	0.840000	0.000000	0.360000	1.920000
25%	89.250000	11.940000	15.300000	12.450000
50%	179.700000	27.000000	30.900000	15.480000
75%	262.590000	43.680000	54.120000	20.880000
max	355.680000	59.520000	136.800000	32.400000

df.isnull().sum()

```
In [6]: # 有NaN
...: marketing.isnull().sum()
Out[6]:
youtube      0
facebook     1
newspaper     0
sales         0
dtype: int64

In [7]: marketing['facebook']
Out[7]:
0      45.36
1      NaN
2      55.08
3      49.56
4      12.96
...
195     4.44
196     5.88
197    11.16
198    50.40
199    10.32
Name: facebook, Length: 200, dtype: float64
```

df['a'].fillna()

```
In [8]: # 將 facebook 變數的 NaN 資料，以中位數填滿
....: marketing['facebook'].fillna(marketing['facebook'].median, inplace = True)

In [9]: # 沒有NaN
....: marketing.isnull().sum()

Out[9]:
youtube      0
facebook     0
newspaper    0
sales        0
dtype: int64
```

讀取 Excel

pd.read_excel()

```
# 讀入 Excel 檔案，全國訂單明細.xlsx
# https://github.com/rwepa/DataDemo/blob/master/全國訂單明細.xlsx

sales = pd.read_excel(io = 'C:/mydata/全國訂單明細.xlsx', sheet_name = '全國訂單明細')
sales # 8568*19
sales.head()

In [6]: sales.head()
Out[6]:
   訂單號      訂單日期  顧客姓名 訂單等級 ...  產品包箱      運送日期 Unnamed: 19 Unnamed: 20
0     3 2010-10-13  李鵬農    低級 ...  大型箱子 2010-10-20        NaN        NaN
1     6 2012-02-20  王勇民    其它 ...  小型包裹 2012-02-21        NaN        NaN
2    32 2011-07-15  姚文文    高級 ...  中型箱子 2011-07-17        NaN        NaN
3    32 2011-07-15  姚文文    高級 ...  巨型紙箱 2011-07-16        NaN        NaN
4    32 2011-07-15  姚文文    高級 ...  中型箱子 2011-07-17        NaN        NaN

[5 rows x 21 columns]
```

讀取 SAS

pd.read_sas()

```
# 讀入 SAS 檔案, h_nhi_ipdte103.sas7bdat
# 資料說明: 103年模擬全民健保處方及治療明細檔_西醫住院檔
# 資料筆數: 14297
# 欄位個數: 80
# 檔案大小: 7.25MB
# https://github.com/rwepa/DataDemo/blob/master/h\_nhi\_ipdte103.sas7bdat

ipdate = pd.read_sas(filepath_or_buffer = 'C:/mydata/h_nhi_ipdte103.sas7bdat')
ipdate # 14297*80
ipdate.head()
```

	ID	PRSN_ID	...	CHILD_MARK	TW_DRGS_SUIT_MARK
0	b'])}))#*+;[*<'	b'%%%**#==_]~~'	...	NaN	b'1'
1	b'])}))+-^\$]\$[/'	b'~%~@&[>#*^#_'	...	NaN	b'C'
2	b'])]))/_^(&_'	b'*]*>=)&); }:+}~'	...	NaN	b'0'
3	b'])))~\$<_&#>'	b']]]:+\$/_\$};:'	...	NaN	b'1'
4	b']))))<@%/-@/\\"'	b'+%+[-"-*%]<<'	...	NaN	b'C'

[5 rows x 80 columns]

資料匯出

to_csv()

```
# 資料匯出
df = pd.DataFrame({'姓名': ['ALAN', 'LEE'],
                    '地址': ['台北市', '新北市'],
                    '年資': [10, 20]})

df
#      姓名    地址  年資
# 0   ALAN  台北市   10
# 1     LEE  新北市   20

df.to_csv('data/df.csv', index = False)
```

Excel有亂碼

df.csv			
1	姓名,地址,年資		
2	ALAN,台北市,10		
3	LEE,新北市,20		

	A	B	C
1	憊 ?	?噏?	擣渲?
2	ALAN	?噏?撣?10	
3	LEE	?噏?撣?20	

encoding = 'utf_8_sig'

```
df.to_csv('data/df.csv', index = False, encoding = 'utf_8_sig') # OK
```

	A	B	C
1	姓名	地址	年資
2	ALAN	台北市	10
3	LEE	新北市	20

Excel 中文正常

05.MySQL與SQL語法,Python連結MySQL應用



(6.4 GB)

MySQL 下載與安裝



MySQL 下載與安裝 – 2/8



Google 搜尋結果顯示了 MySQL 的下載頁面：

- 在搜索框中輸入 "mysql download"，標記為 1。
- 點擊 "MySQL Community Downloads - MySQL" 鏈接，標記為 2。

其他相關搜尋項目包括：

- MySQL download
- MySQL Community Server 8.0 17 download
- mysql 64 bit下載
- Mysql installer 5.7 12
- MySQL Installer 8.0 13
- Mysql server download archive

MySQL 下載與安裝 – 2/8

- <https://dev.mysql.com/downloads/>

① MySQL Community Downloads

- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench
- MySQL Installer for Windows
- MySQL for Visual Studio
- C API (libmysqlclient)
- Connector/C++
- Connector/J
- Connector/.NET
- Connector/Node.js
- Connector/ODBC
- Connector/Python
- MySQL Native Driver for PHP
- MySQL Benchmark Tool
- Time zone description tables
- Download Archives

MySQL 下載與安裝 – 3/8

MySQL Community Downloads

MySQL Installer

General Availability (GA) Releases Archives

MySQL Installer 8.0.23

Select Operating System: Microsoft Windows

Looking for previous GA versions?

Windows (x86, 32-bit), MSI Installer
(mysql-installer-web-community-8.0.23.0.msi)
8.0.23 2.4M Download
MD5: a3af6d91f93e046450b38a1e2589534c | Signature

Windows (x86, 32-bit), MSI Installer
(mysql-installer-community-8.0.23.0.msi)
8.0.23 422.4M Download
MD5: 8de85ced955631901829a1a363cdbf50 | Signature

We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

最新版 8.0.27



MySQL 下載與安裝 – 4/8

⬇ MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

[Login »](#)

using my Oracle Web account

[Sign Up »](#)

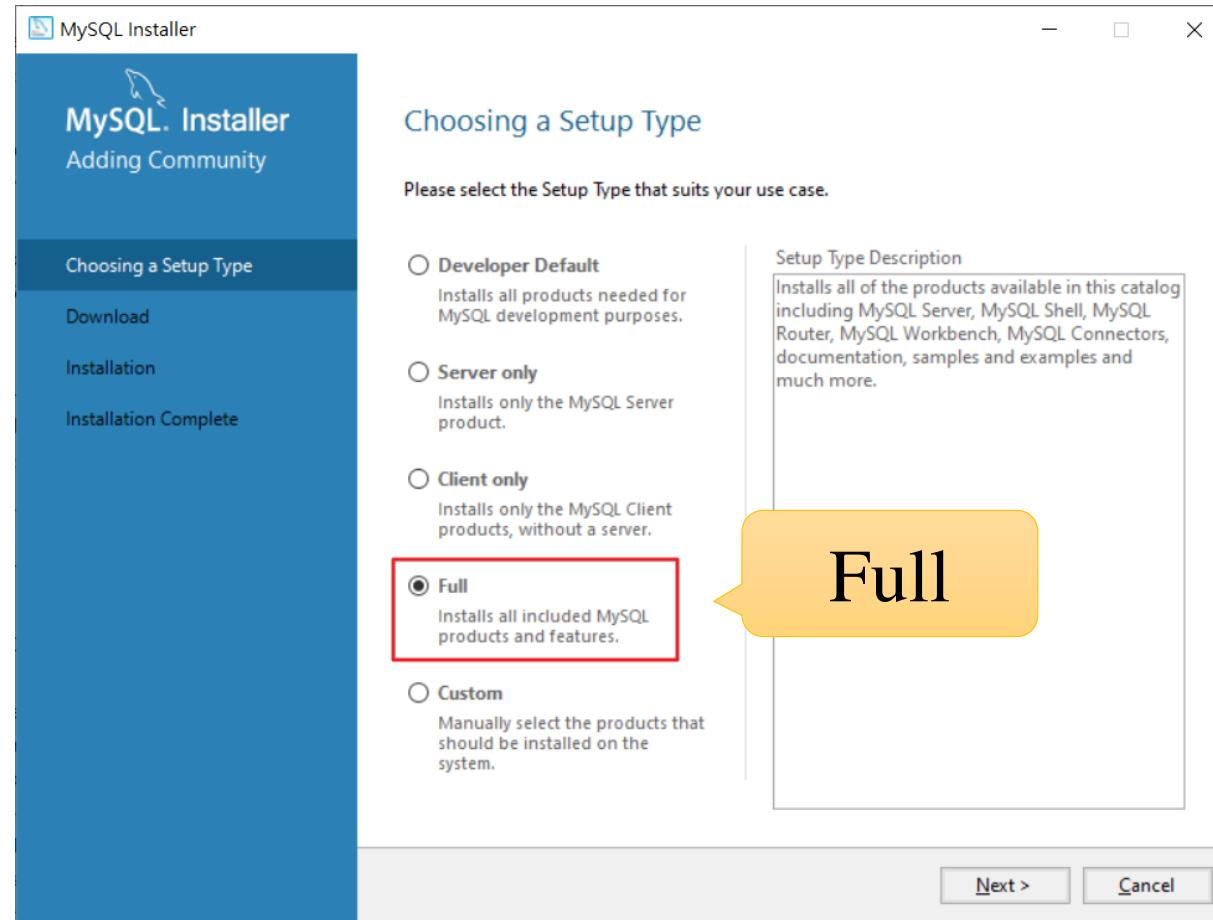
for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

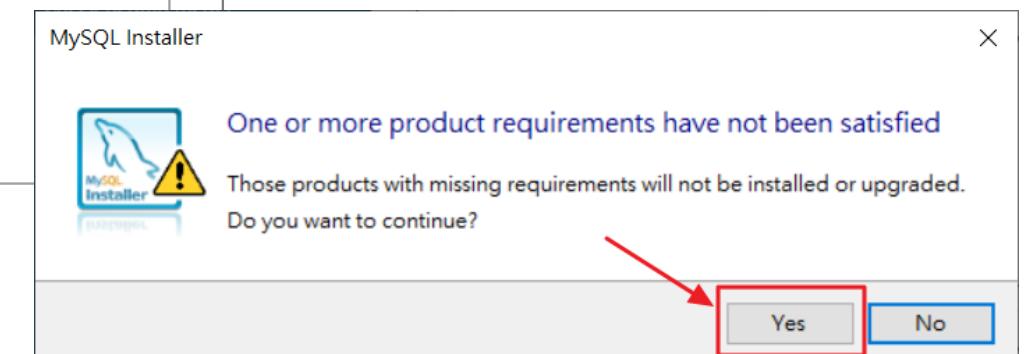
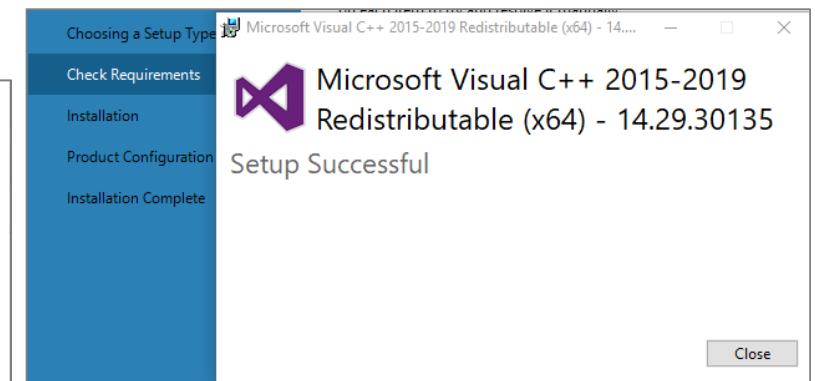
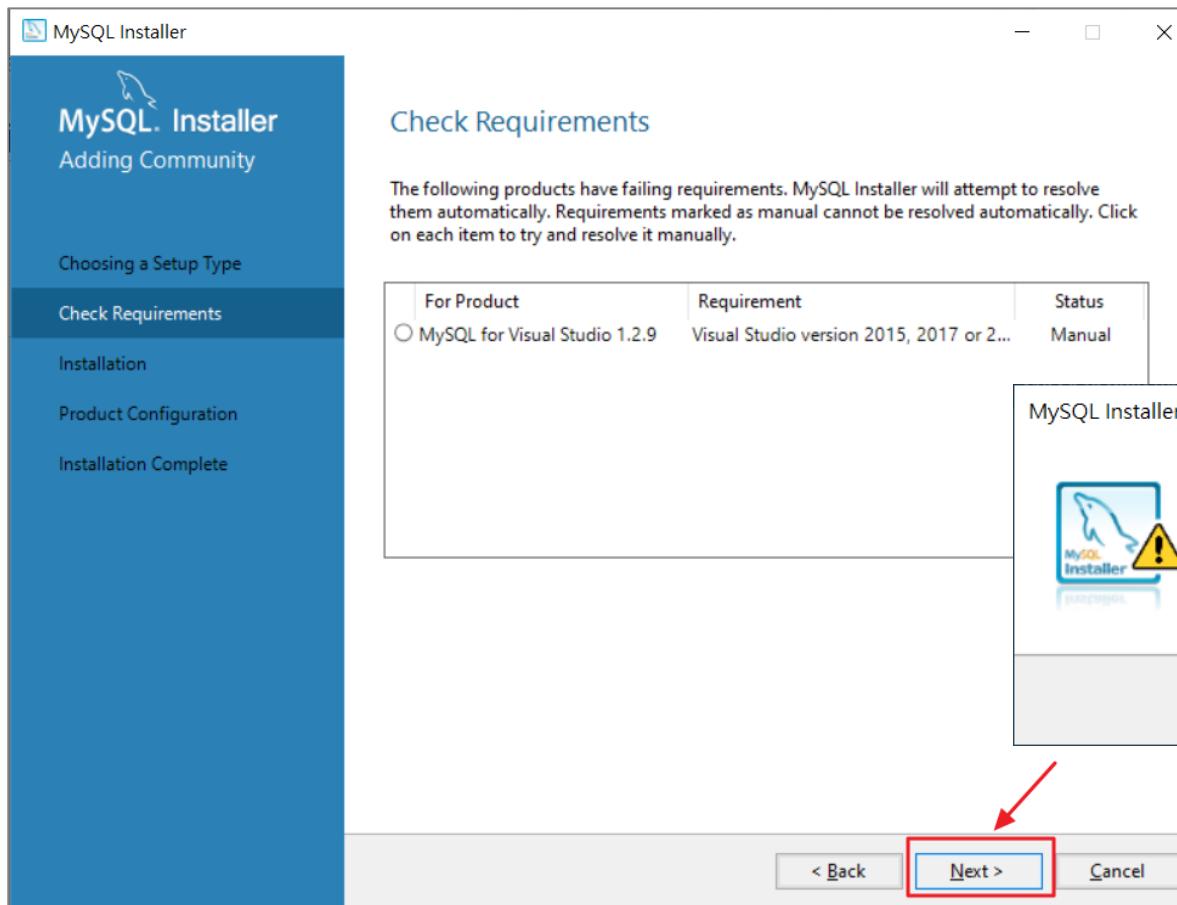
No thanks, just start my download.

mysql-installer-community-8.0.23.0.msi

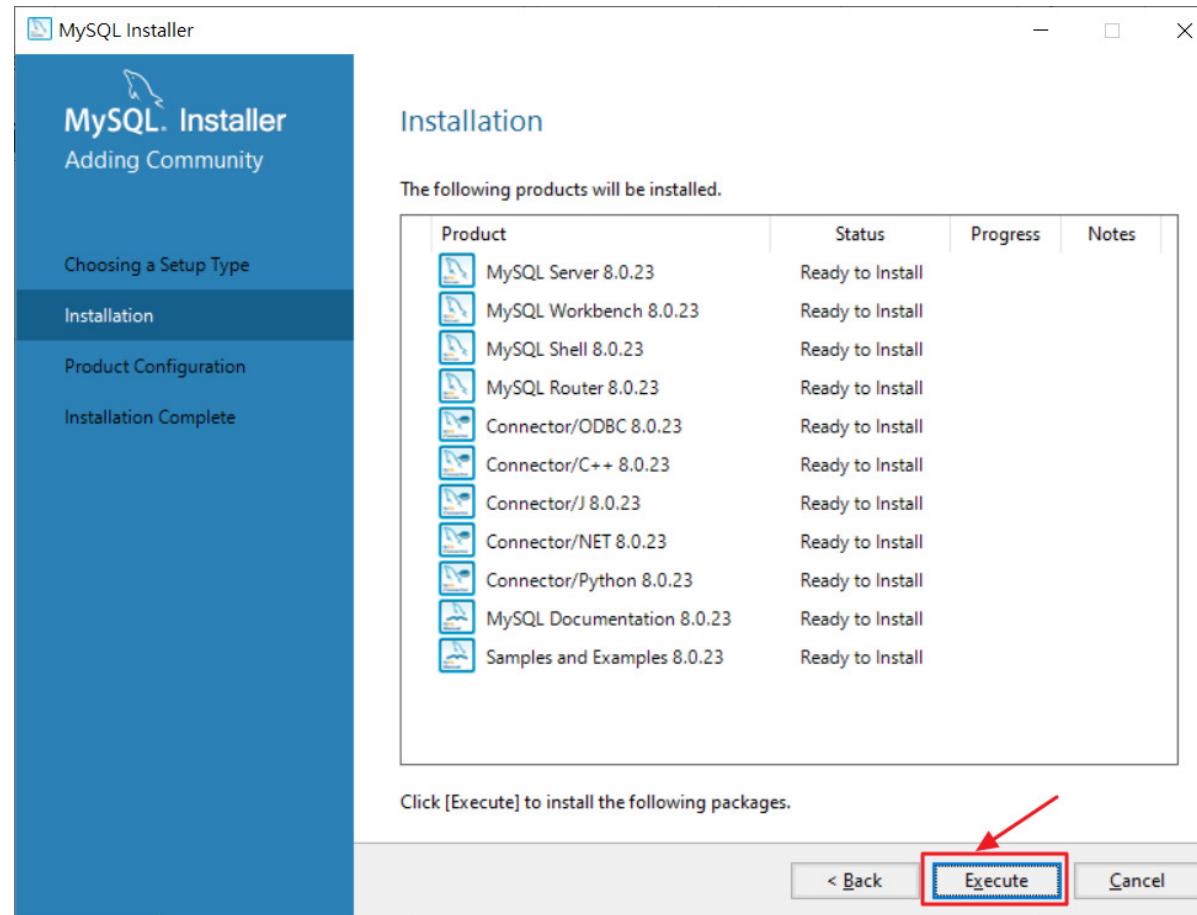
MySQL 下載與安裝 – 5/8



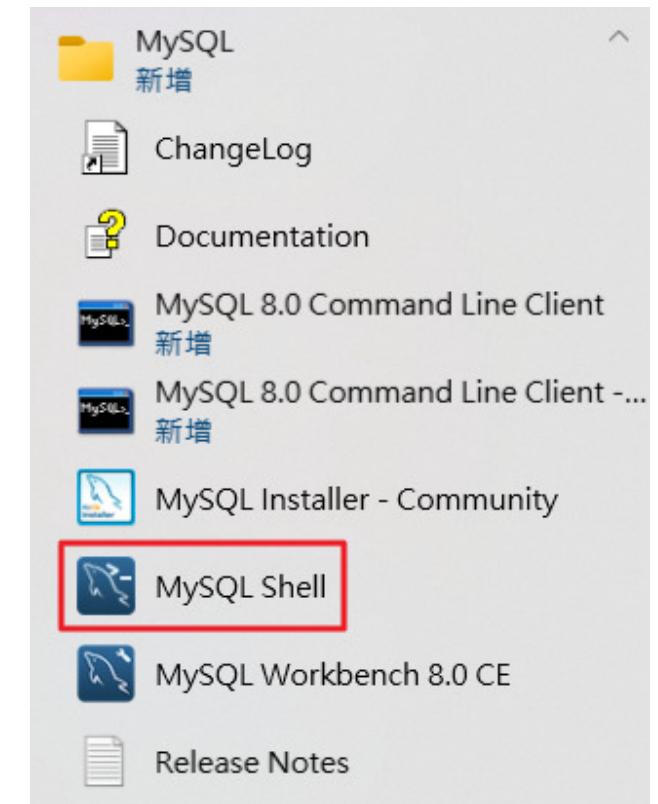
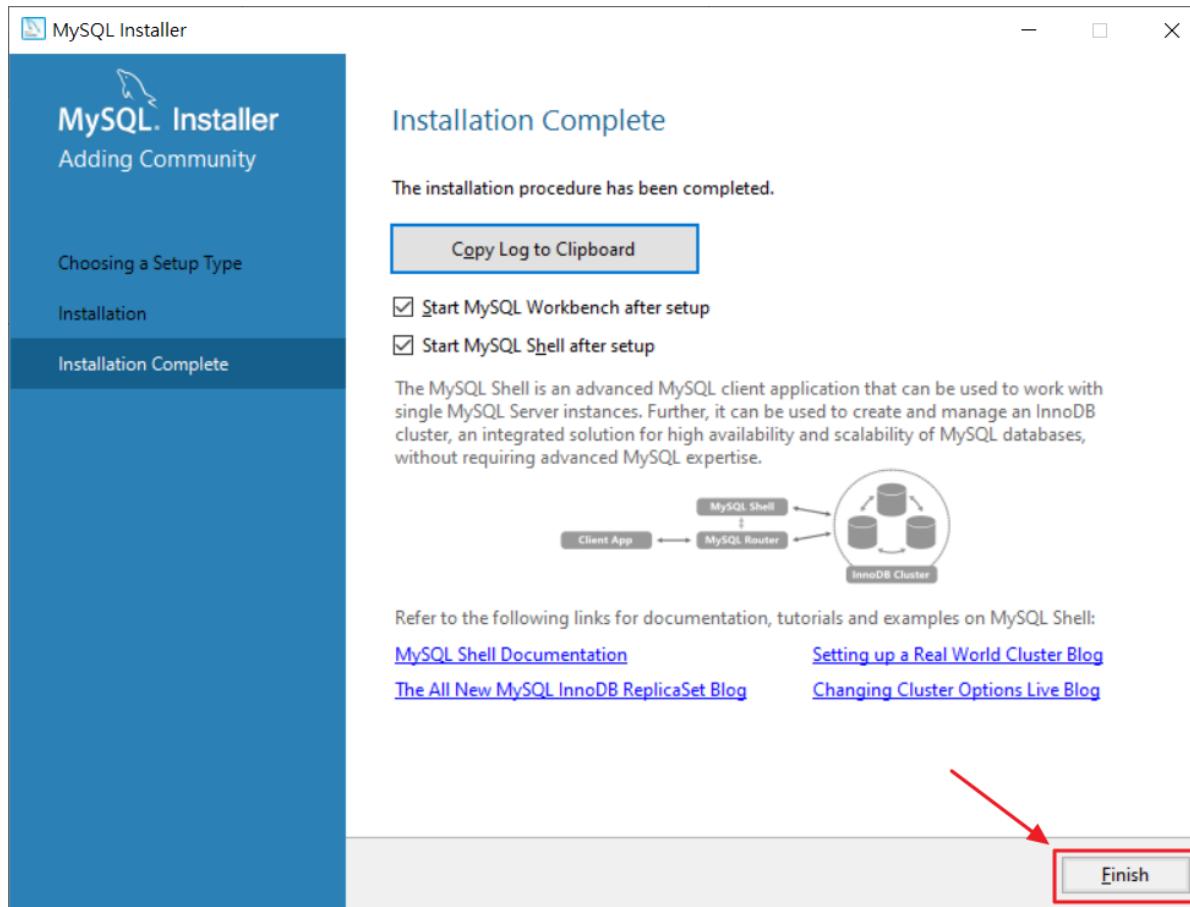
MySQL 下載與安裝 – 6/8



MySQL 下載與安裝 – 7/8

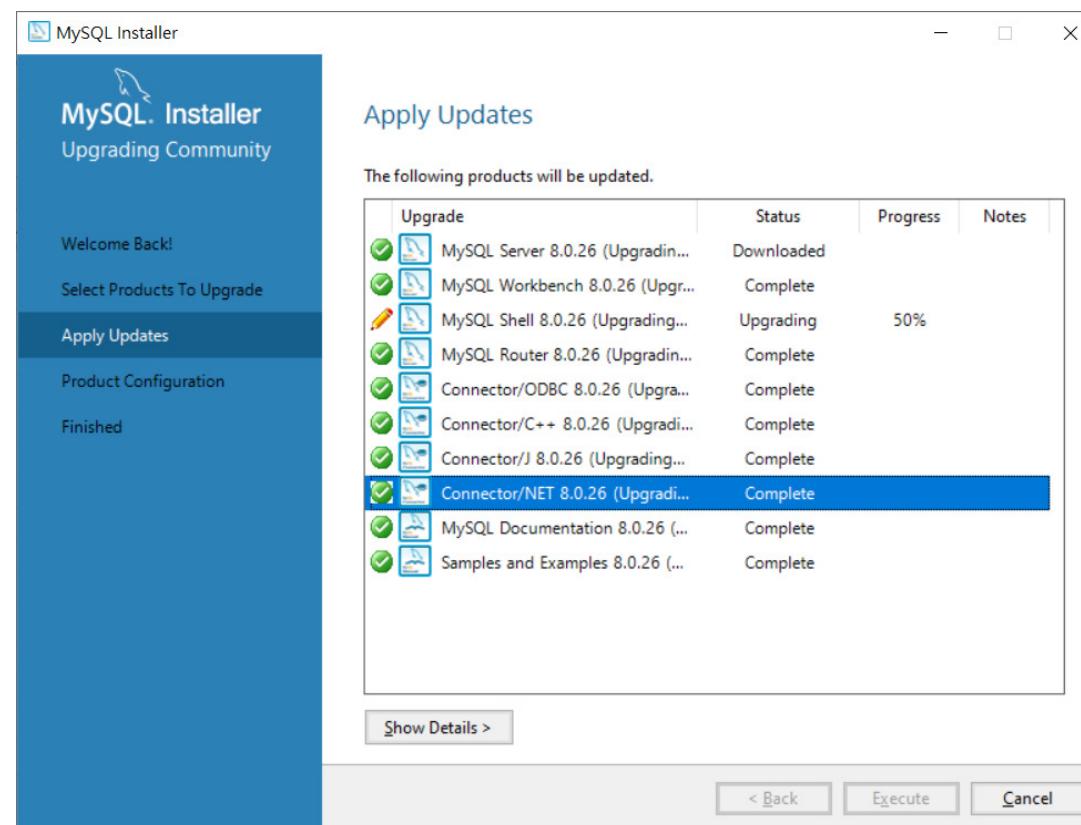


MySQL 下載與安裝 – 8/8 (Installation Complete)

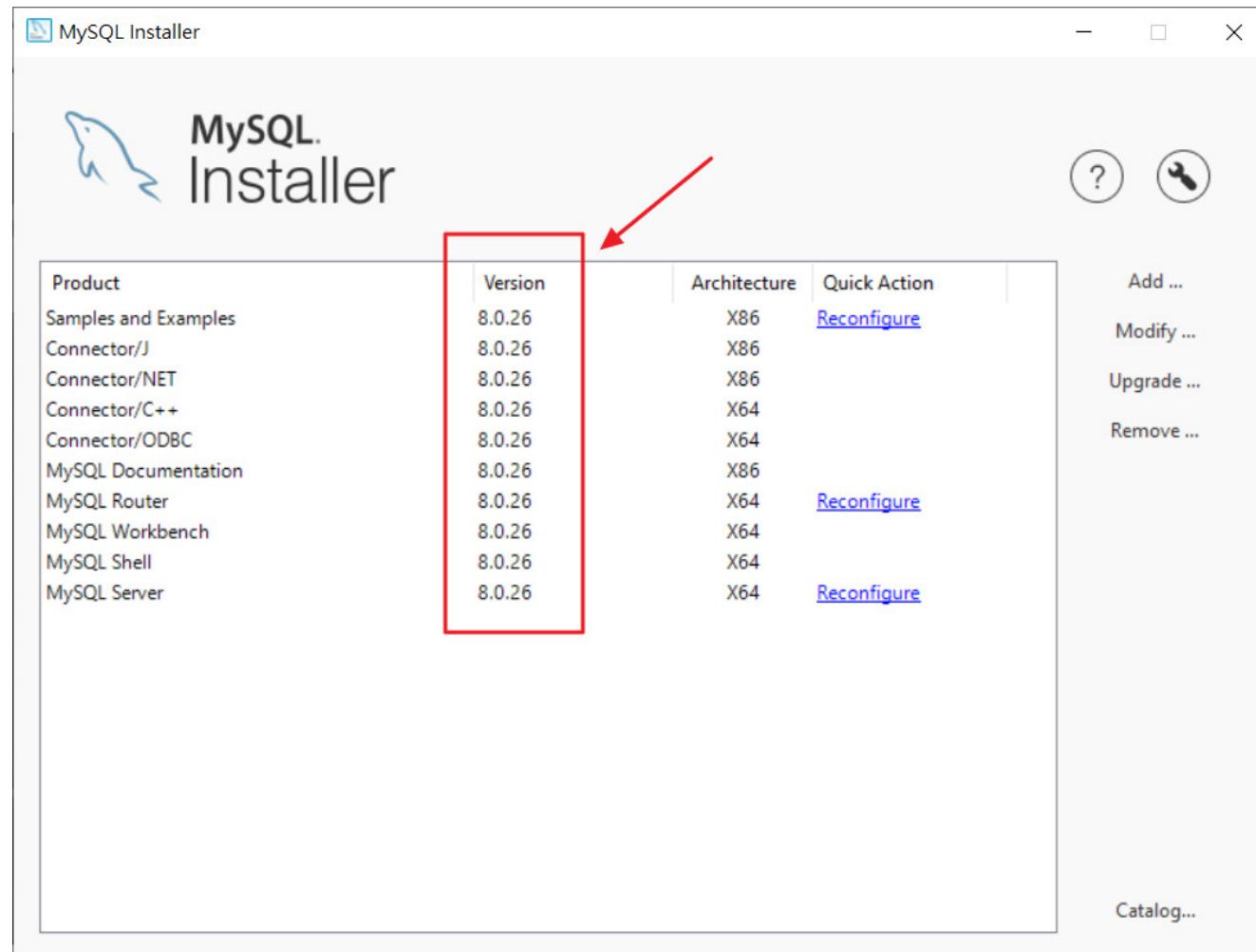


MySQL Installer – Community

-  MySQL Installer - Community 提供更新/升級功能



更新/升級至最新版 v8.0.26



MySQL Shell

1. MySQL Shell 簡介
2. MySQL Shell 啟動
3. MySQL Shell 指令
4. MySQL Shell – SQL 範例

1. MySQL Shell 簡介

- MySQL Shell是MySQL的進階客戶端和程式碼編輯器
- X DevAPI 支援關聯式資料與schema-less的文件資料
- 可執行 JavaScript, Python and SQL 等程式
- 8.0.18以後版本採用 Python 3 為主
- 支援互動式編碼執行 Interactive Code Execution
- 支援批次執行 Batch Code Execution
- 可自行定義報表與延伸功能

參考: <https://dev.mysql.com/doc/mysql-shell/8.0/en/>

2. MySQL Shell 啟動

- 程式集 \ MSQL \ MySQL Shell
- 命令提示字元輸入: mysqlsh



A screenshot of a Windows Command Prompt window. The title bar says '選取 命令提示字元'. The window content shows:

```
C:\>選取 命令提示字元
Microsoft Windows [版本 10.0.19042.746]
(c) 2020 Microsoft Corporation。著作權所有，並保留一切權利。
C:\Users\88697>mysqlsh
MySQL Shell 8.0.22

Copyright (c) 2016, 2020, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS > \quit
Bye!

C:\Users\88697>
```

The command 'mysqlsh' is highlighted with a red box. The MySQL logo 'MySQL JS' is also highlighted with a yellow box.



3. MySQL Shell 指令 (1/4)

中文功能	Command	Alias/Shortcut	Description
線上說明	\help	\h or \?	Print help about MySQL Shell, or search the online help.
離開	\quit	\q or \exit	Exit MySQL Shell.
切換至多行模式	\		In SQL mode, begin multiple-line mode. Code is cached and executed when an empty line is entered.
系統狀態	\status	\s	Show the current MySQL Shell status.
切換至 JavaScript	\js		Switch execution mode to JavaScript.
切換至 Python	\py		Switch execution mode to Python.
切換至 SQL	\sql		Switch execution mode to SQL.



3. MySQL Shell 指令 (2/4)

中文功能	Command	Alias/Shortcut	Description
連接至資料庫	\connect	\c	Connect to a MySQL instance.
重新連接至資料庫	\reconnect		Reconnect to the same MySQL instance.
中斷連線	\disconnect		Disconnect the global session.
設定資料庫綱要	\use	\u	Specify the schema to use.
執行腳本檔案	\source	\. or source (no backslash)	Execute a script file using the active language.

3. MySQL Shell 指令 (3/4)

中文功能	Command	Alias/Shortcut	Description
顯示警告訊息	\warnings	\W	Show any warnings generated by a statement.
不顯示警告訊息	\nowarnings	\w	Do not show any warnings generated by a statement.
歷史指令	\history		View and edit command line history.
手動更新	\rehash		Manually update the autocomplete name cache.
查詢與設定組態	\option		Query and change MySQL Shell configuration options.
顯示報表	\show		Run the specified report using the provided options and arguments.
監控報表	\watch		Run the specified report using the provided options and arguments, and refresh the results at regular intervals.

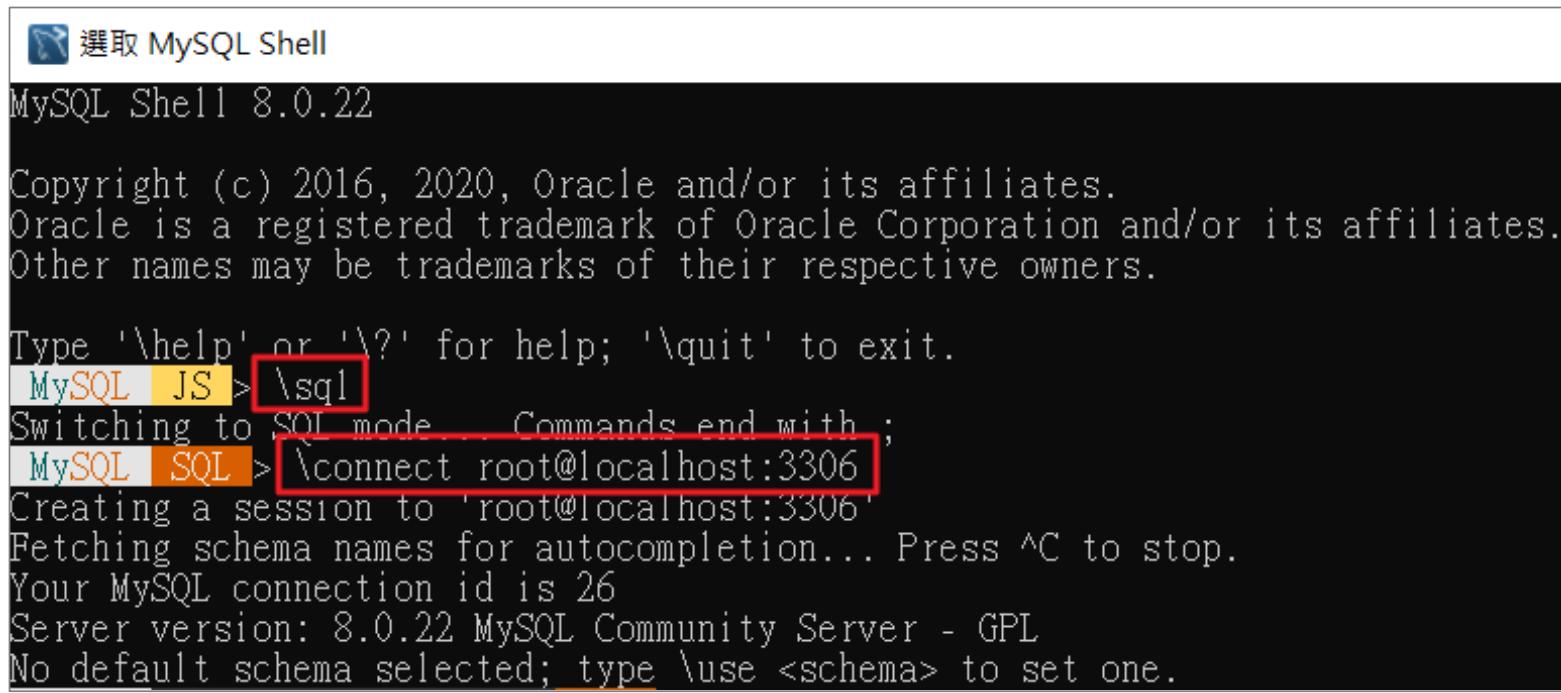


3. MySQL Shell 指令 (4/4)

中文功能	Command	Alias/Shortcut	Description
編輯	\edit	\e	Open a command in the default system editor then present it in MySQL Shell.
換頁設定	\pager	\P	Configure the pager which MySQL Shell uses to display text.
取消換頁	\nopager		Disable any pager which MySQL Shell was configured to use.
執行作業系統指令	\system	\!	Run the specified operating system command and display the results in MySQL Shell.

4. MySQL Shell – SQL 範例

- \sql 切換至SQL模式
- \connect root@localhost:3306 連接資料庫



The screenshot shows the MySQL Shell interface. It starts with a welcome message from Oracle. The user then types '\sql' at the prompt, which is highlighted with a red box. This command switches the mode to SQL. The user then types '\connect root@localhost:3306', which is also highlighted with a red box. The shell then creates a session for the root user on the local host. It fetches schema names for autocompletion and displays the MySQL connection ID (26). Finally, it shows the server version (8.0.22 MySQL Community Server - GPL) and a note about the default schema.

```
選取 MySQL Shell
MySQL Shell 8.0.22

Copyright (c) 2016, 2020, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS > \sql
Switching to SQL mode... Commands end with ;
MySQL SQL > \connect root@localhost:3306
Creating a session to 'root@localhost:3306'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 26
Server version: 8.0.22 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
```

\status 資料庫狀態

```
選取 MySQL Shell
MySQL | localhost:3306 ssl SQL > \status
MySQL Shell version 8.0.22

Connection Id: 26
Current schema: root@localhost
Current user: root@localhost
SSL: Cipher in use: TLS_AES_256_GCM_SHA384 TLSv1.3
Using delimiter: ;
Server version: 8.0.22 MySQL Community Server - GPL
Protocol version: Classic 10
Client library: 8.0.22
Connection: localhost via TCP/IP
TCP port: 3306
Server characterset: utf8mb4
Schema characterset: utf8mb4
Client characterset: utf8mb4
Conn. characterset: utf8mb4
Result characterset: utf8mb4
Compression: Disabled
Uptime: 2 days 5 hours 44 min 2.0000 sec

Threads: 4 Questions: 223 Slow queries: 0 Opens: 264 Flush tables: 3 Open tables: 185 Queries per second avg: 0.001
MySQL | localhost:3306 ssl SQL >
```

SHOW databases; 顯示資料庫

```
MySQL localhost:3306 ssl SQL > SHOW databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| mysql |  
| performance_schema |  
| sakila |  
| sys |  
| world |  
+-----+  
6 rows in set (0.0015 sec)
```

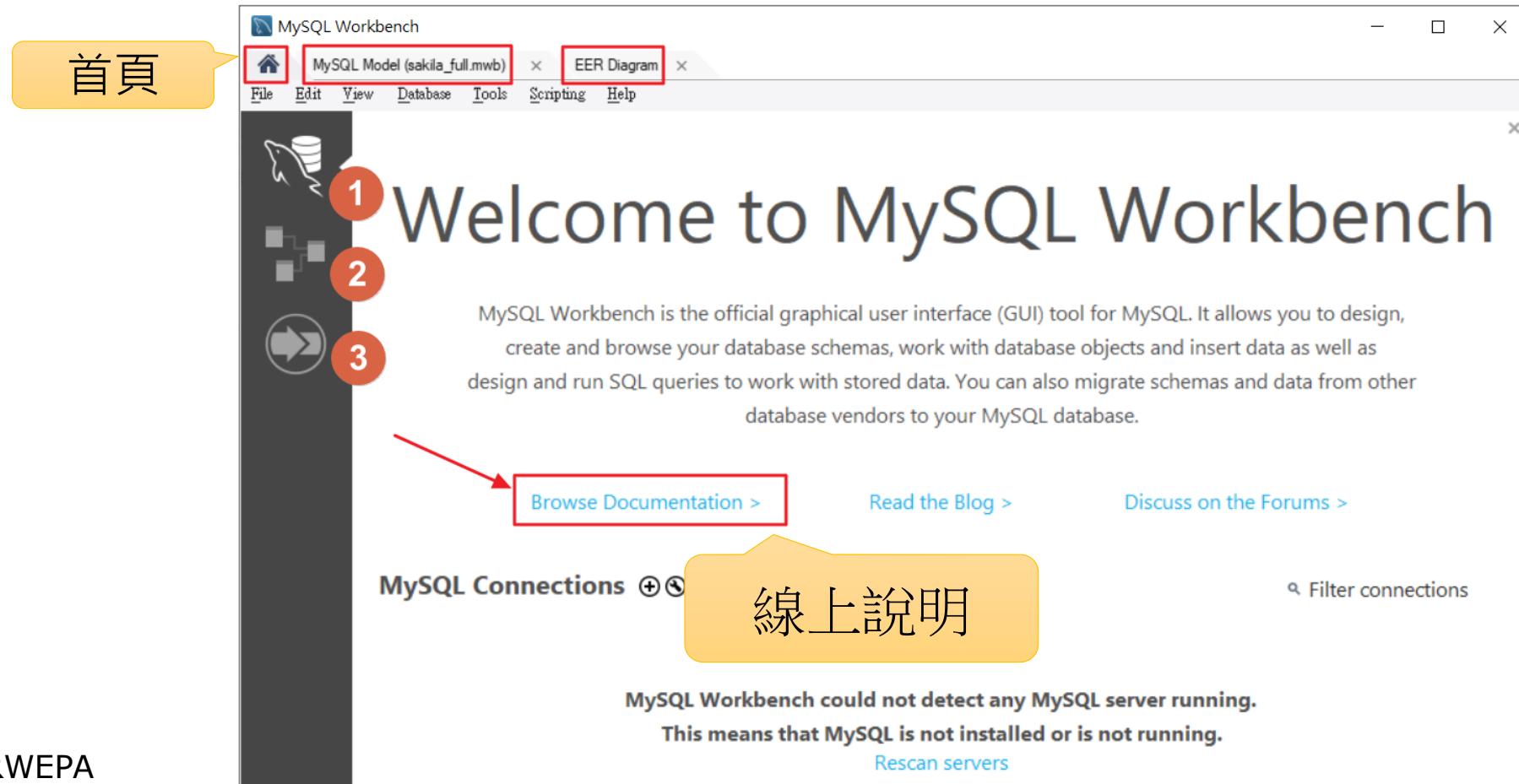
包括 sakila, world 等資料庫

USE sakila; 使用資料庫

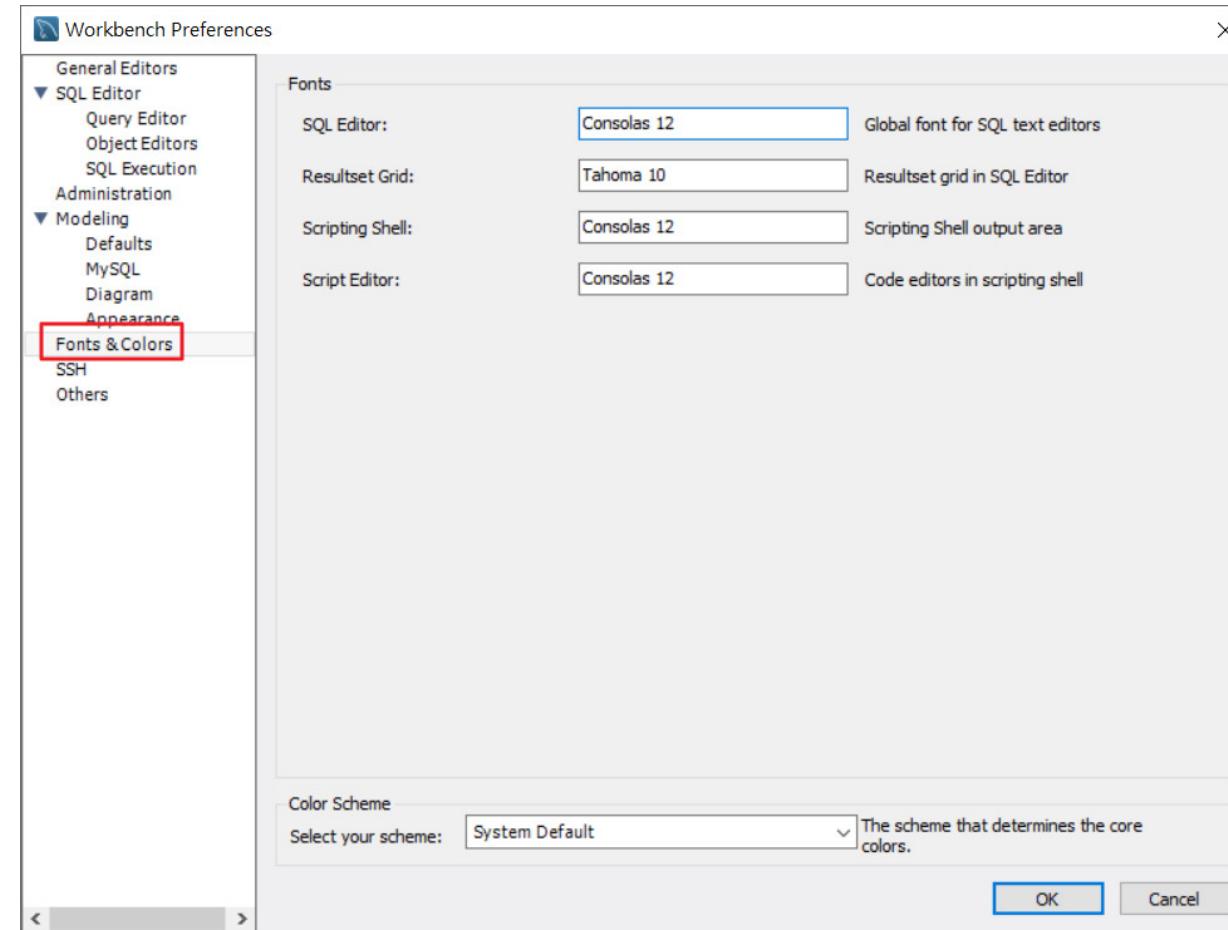
SELECT * FROM rental limit 6;

```
MySQL Shell
MySQL [localhost:3306 ssl] sakila SQL > USE sakila;
Default schema set to `sakila`.
Fetching table and column names from `sakila` for auto-completion... Press ^C to stop.
MySQL [localhost:3306 ssl] sakila SQL > SELECT * FROM rental limit 6;
+-----+-----+-----+-----+-----+-----+-----+
| rental_id | rental_date | inventory_id | customer_id | return_date | staff_id | last_update |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | 2005-05-24 22:53:30 | 367 | 130 | 2005-05-26 22:04:30 | 1 | 2006-02-15 21:30:53 |
| 2 | 2005-05-24 22:54:33 | 1525 | 459 | 2005-05-28 19:40:33 | 1 | 2006-02-15 21:30:53 |
| 3 | 2005-05-24 23:03:39 | 1711 | 408 | 2005-06-01 22:12:39 | 1 | 2006-02-15 21:30:53 |
| 4 | 2005-05-24 23:04:41 | 2452 | 333 | 2005-06-03 01:43:41 | 2 | 2006-02-15 21:30:53 |
| 5 | 2005-05-24 23:05:21 | 2079 | 222 | 2005-06-02 04:33:21 | 1 | 2006-02-15 21:30:53 |
| 6 | 2005-05-24 23:08:07 | 2792 | 549 | 2005-05-27 01:32:07 | 1 | 2006-02-15 21:30:53 |
+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.0007 sec)
MySQL [localhost:3306 ssl] sakila SQL > -
```

MySQL Workbench



Edit \ Preferences \ Fonts & Colors



歡迎視窗

Workbench Preferences X

General Editors

▼ SQL Editor

- Query Editor
- Object Editors
- SQL Execution

Administration

▼ Modeling

- Defaults
- MySQL
- Diagram
- Appearance

Fonts & Colors

SSH

Others

Home Screen

Show Welcome Message on Connections Screen

Timeouts

Migration Connection Timeout: Maximum time to wait before a connection is aborted.

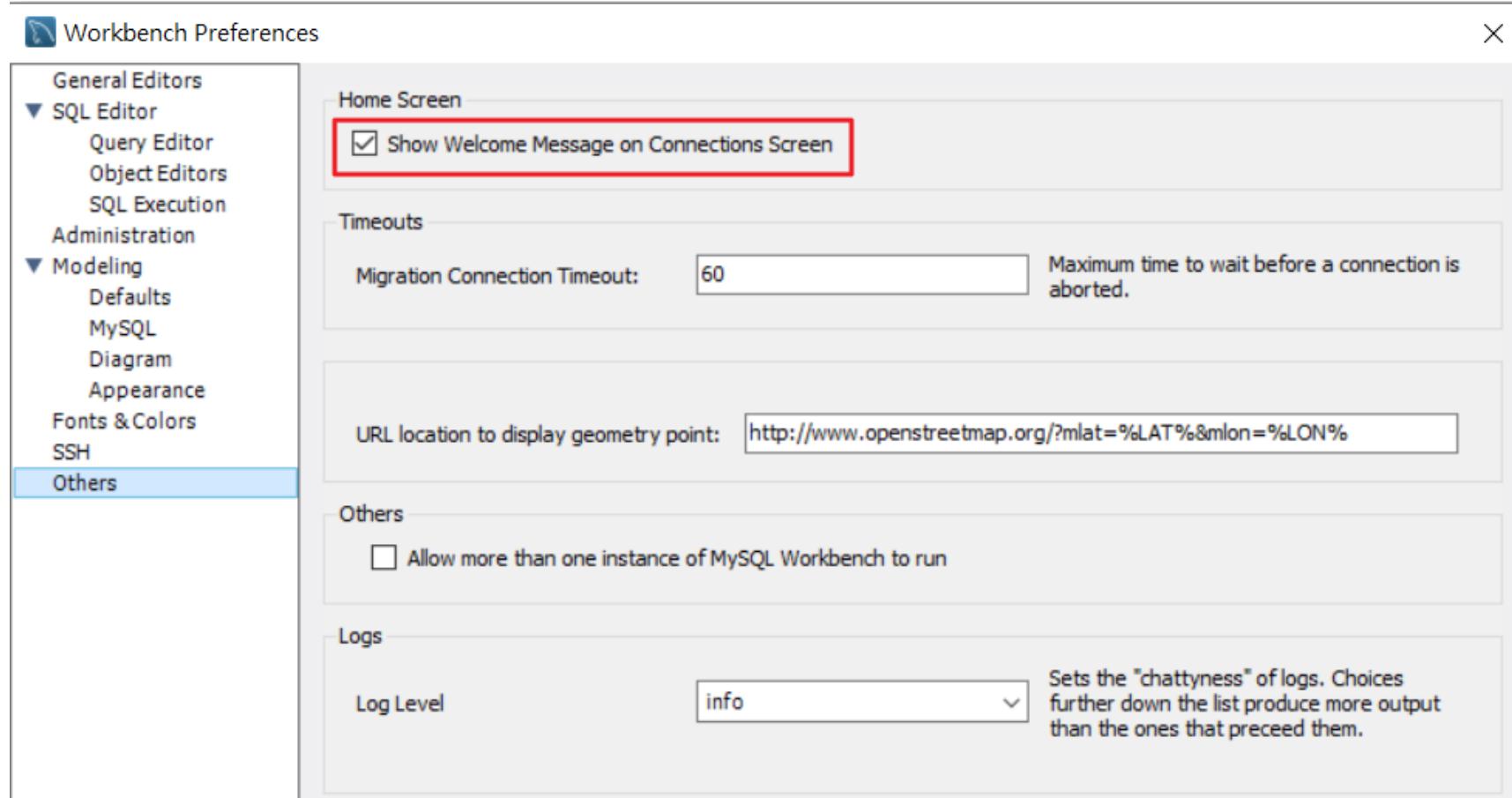
URL location to display geometry point:

Others

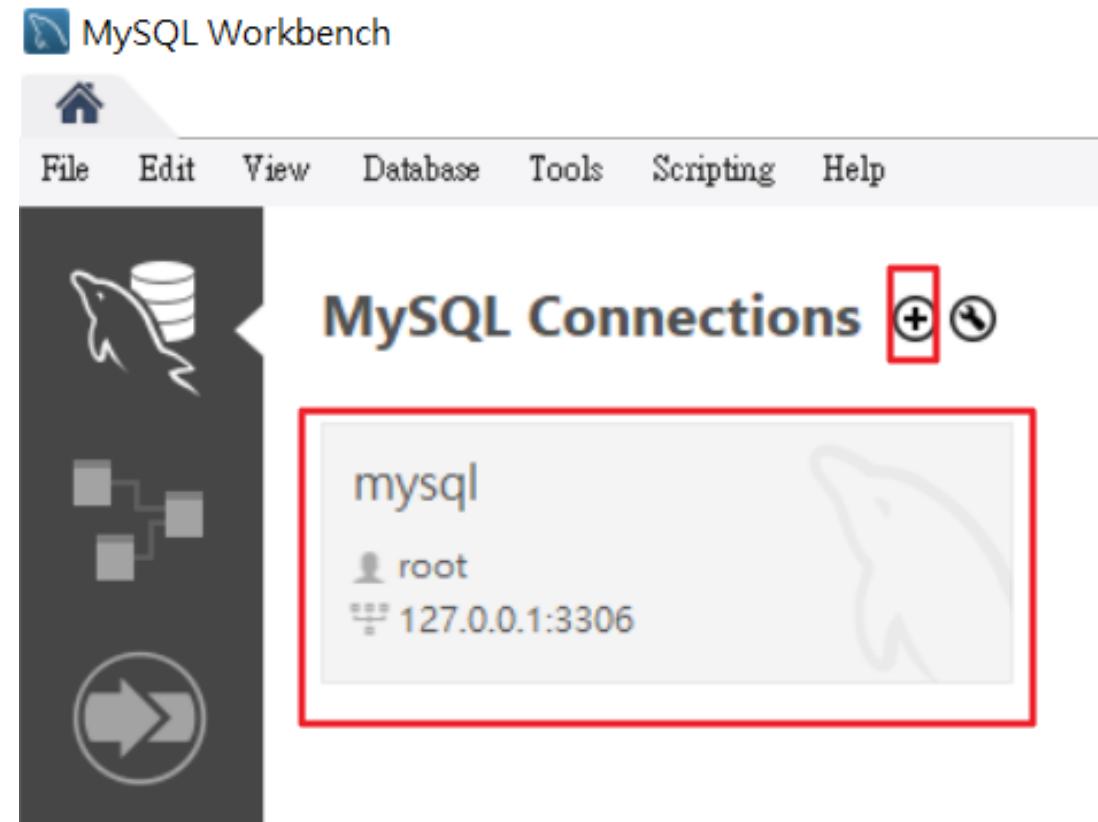
Allow more than one instance of MySQL Workbench to run

Logs

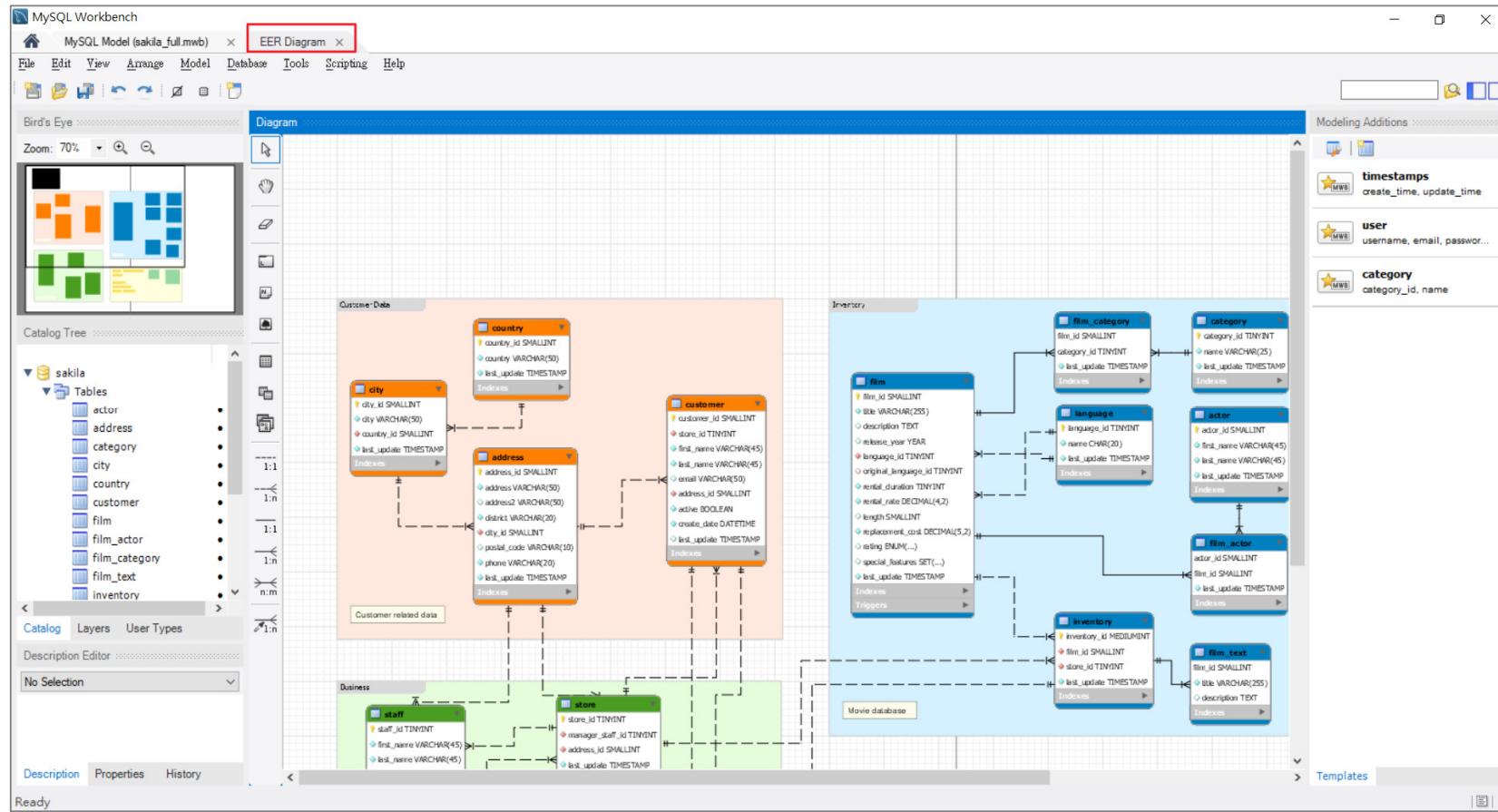
Log Level Sets the "chattiness" of logs. Choices further down the list produce more output than the ones that precede them.



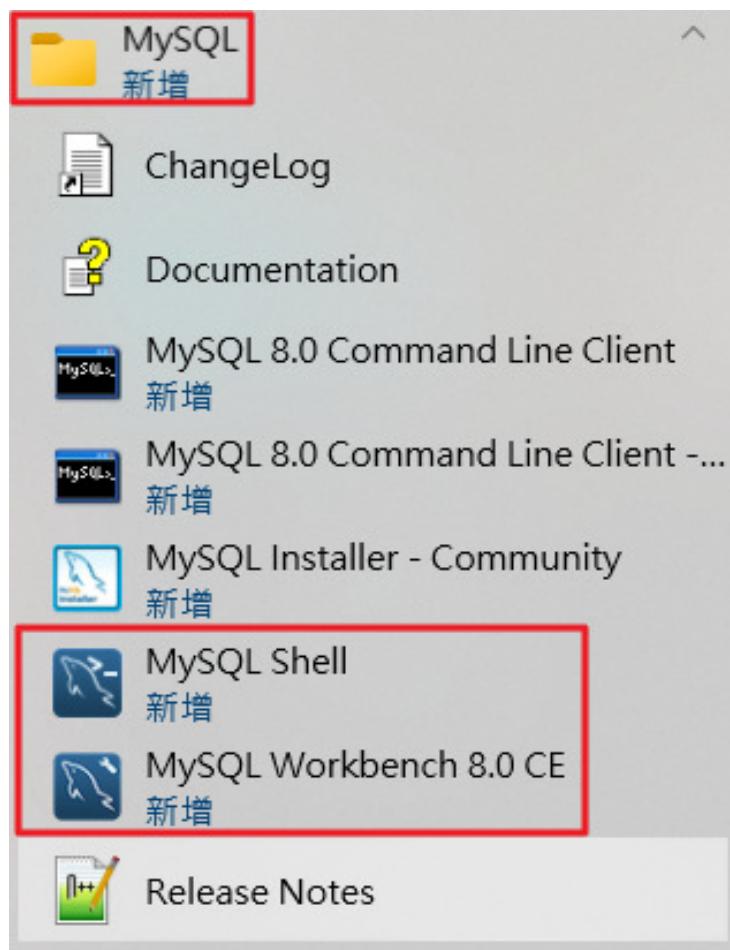
MySQL Connections



Enhanced Entity-Relationship (EER) Diagrams



程式集 - MySQL



- MySQL 線上說明
<https://dev.mysql.com/doc/refman/8.0/en/>

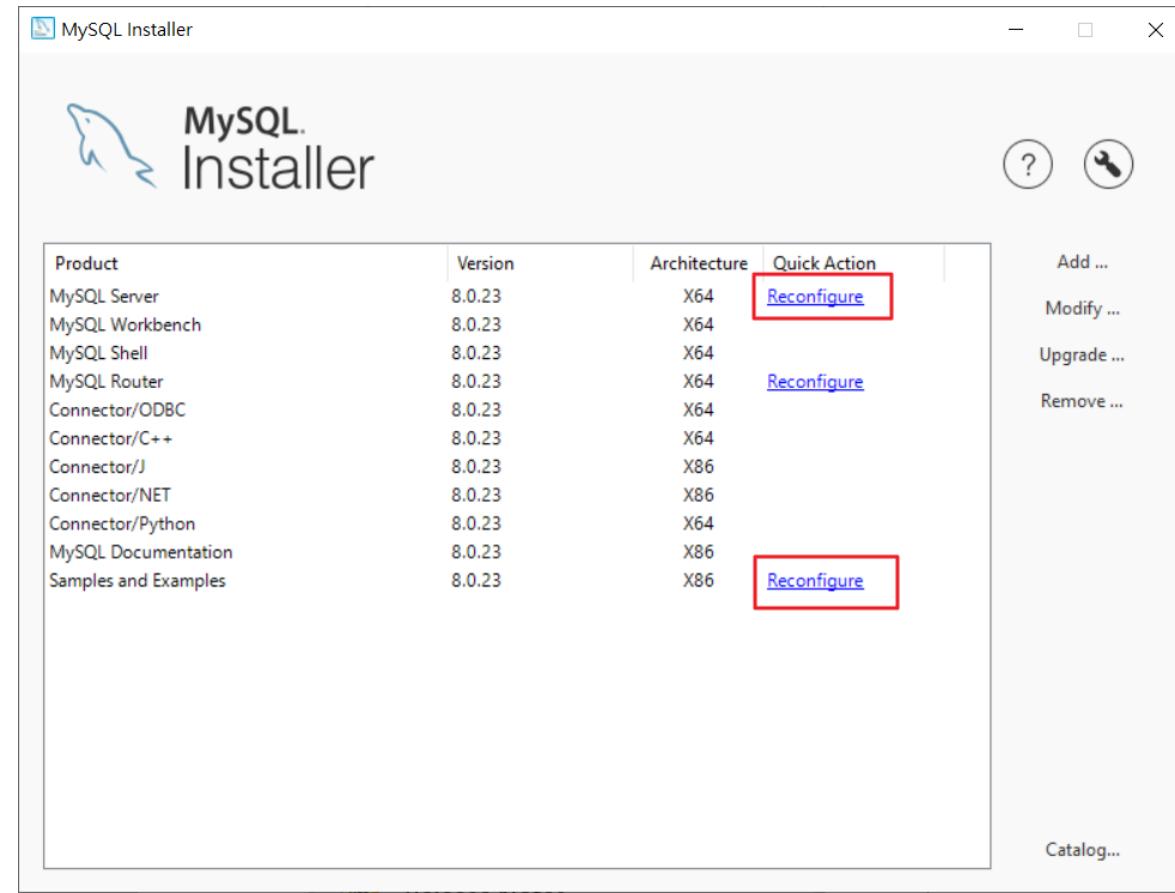
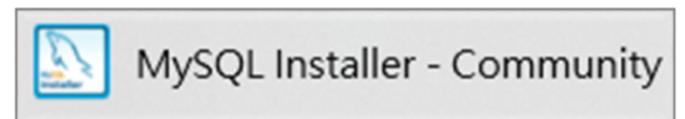
MySQL 8.0 Reference Manual

Including MySQL NDB Cluster 8.0

Abstract

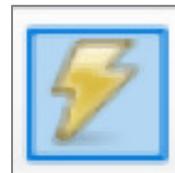
This is the MySQL™ Reference Manual. It documents MySQL 8.0 through 8.0.25, as well as I through 8.0.23-ndb-8.0.23, respectively. It may include documentation of features of MySQL information about which versions have been released, see the MySQL 8.0 Release Notes.

MySQL Installer - Community

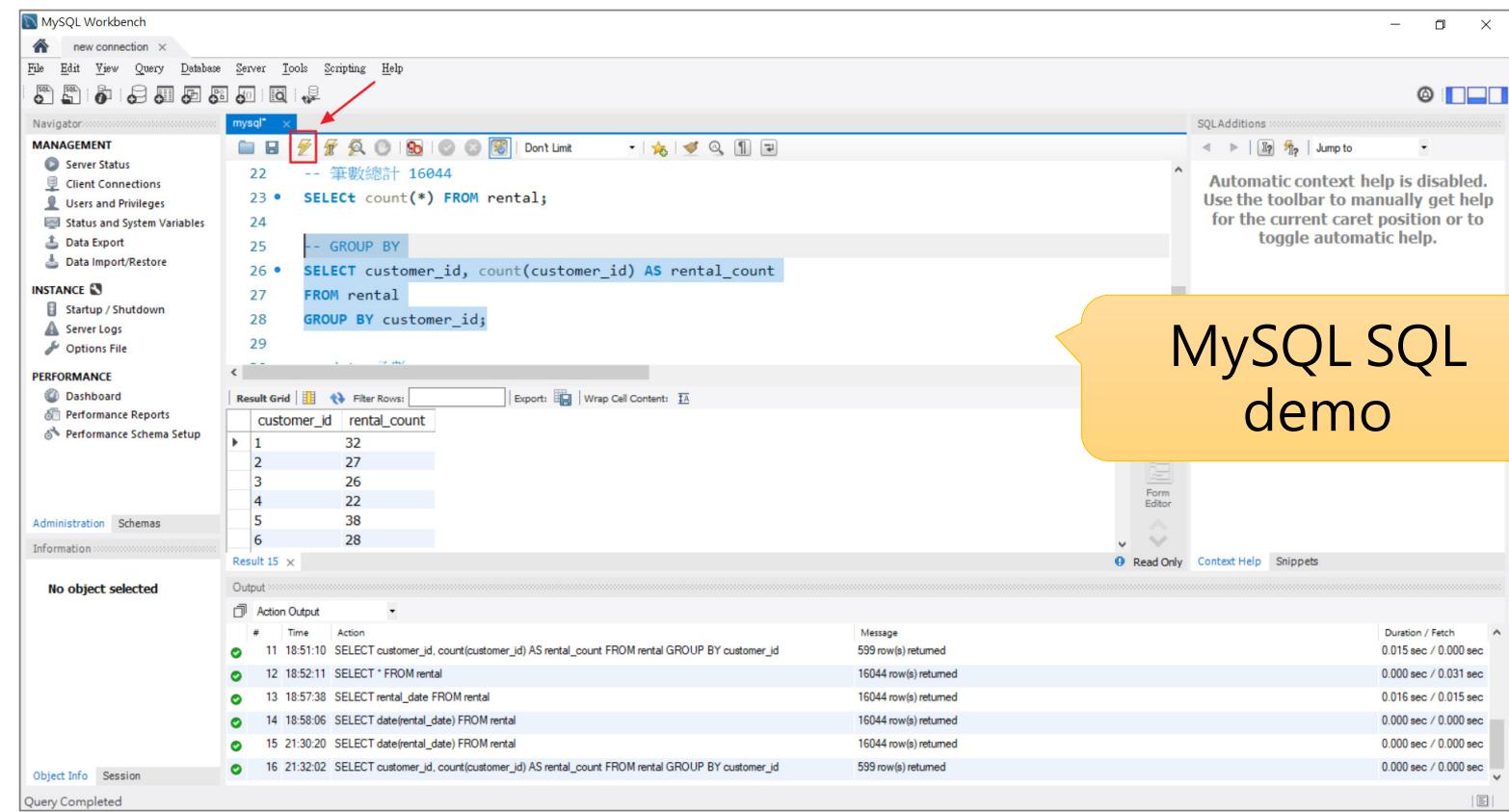


元件修改

MySQL 練習- 參考程式碼



Execute the selected portion of the script or everything, if there is no selection



The screenshot shows the MySQL Workbench interface. A red arrow points to the execute button in the toolbar, which is highlighted with a yellow box. The SQL editor contains the following code:

```
22 -- 筆數總計 16044
23 • SELECT count(*) FROM rental;
24
25 • -- GROUP BY
26 • SELECT customer_id, count(customer_id) AS rental_count
27 FROM rental
28 GROUP BY customer_id;
29
```

The Result Grid shows the output of the query:

customer_id	rental_count
1	32
2	27
3	26
4	22
5	38
6	28

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
11	18:51:10	SELECT customer_id, count(customer_id) AS rental_count FROM rental GROUP BY customer_id	599 row(s) returned	0.015 sec / 0.000 sec
12	18:52:11	SELECT * FROM rental	16044 row(s) returned	0.000 sec / 0.031 sec
13	18:57:38	SELECT rental_date FROM rental	16044 row(s) returned	0.016 sec / 0.015 sec
14	18:58:06	SELECT date(rental_date) FROM rental	16044 row(s) returned	0.000 sec / 0.000 sec
15	21:30:20	SELECT date(rental_date) FROM rental	16044 row(s) returned	0.000 sec / 0.000 sec
16	21:32:02	SELECT customer_id, count(customer_id) AS rental_count FROM rental GROUP BY customer_id	599 row(s) returned	0.000 sec / 0.000 sec

MySQL SQL
demo

Python 連結 MySQL

conda install -c conda-forge mysql-connector-python

```
Anaconda Prompt (anaconda3) - conda install -c conda-forge mysql-connector-python
The following packages will be downloaded:


| package                       | build          |        |             |
|-------------------------------|----------------|--------|-------------|
| conda-4.10.3                  | py38haa244fe_0 | 3.1 MB | conda-forge |
| dnspython-2.1.0               | pyhd8ed1ab_0   | 120 KB | conda-forge |
| libprotobuf-3.14.0            | h7755175_0     | 2.3 MB | conda-forge |
| mysql-common-8.0.22           | h57928b3_1     | 1.5 MB | conda-forge |
| mysql-connector-python-8.0.22 | py38hd95ec2e_1 | 653 KB | conda-forge |
| mysql-libs-8.0.22             | hb8c0690_1     | 1.7 MB | conda-forge |
| openssl-1.1.1h                | he774522_0     | 5.8 MB | conda-forge |
| protobuf-3.14.0               | py38h885f38d_1 | 262 KB | conda-forge |
| python_abi-3.8                | 2_cp38         | 4 KB   | conda-forge |


Total: 15.3 MB
The following NEW packages will be INSTALLED:
dnspython      conda-forge/noarch::dnspython-2.1.0-pyhd8ed1ab_0
libprotobuf     conda-forge/win-64::libprotobuf-3.14.0-h7755175_0
mysql-common    conda-forge/win-64::mysql-common-8.0.22-h57928b3_1
mysql-connector-p~ conda-forge/win-64::mysql-connector-python-8.0.22-py38hd95ec2e_1
mysql-libs       conda-forge/win-64::mysql-libs-8.0.22-hb8c0690_1
protobuf        conda-forge/win-64::protobuf-3.14.0-py38h885f38d_1
python_abi       conda-forge/win-64::python_abi-3.8-2_cp38
The following packages will be UPDATED:
conda          pkgs/main::conda-4.9.2-py38haa95532_0 --> conda-forge::conda-4.10.3-py38haa244fe_0
The following packages will be SUPERSEDED by a higher-priority channel:
openssl
pkgs/main --> conda-forge
Proceed ([y]/n)? 

```

按 y

mysql-connector-python - 安裝完成

```
Proceed ([y]/n)? y
```

```
Downloading and Extracting Packages
```

dnsPYTHON-2.1.0	120 KB	##### #####	##### #####	100%
protobuf-3.14.0	262 KB	##### #####	##### #####	100%
libprotobuf-3.14.0	2.3 MB	##### #####	##### #####	100%
mysql-common-8.0.22	1.5 MB	##### #####	##### #####	100%
python_abi-3.8	4 KB	##### #####	##### #####	100%
conda-4.10.3	3.1 MB	##### #####	##### #####	100%
mysql-connector-pyth	653 KB	##### #####	##### #####	100%
mysql-libs-8.0.22	1.7 MB	##### #####	##### #####	100%
openssl-1.1.1h	5.8 MB	##### #####	##### #####	100%

```
Preparing transaction: done
```

```
Verifying transaction: done
```

```
Executing transaction: done
```

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

Python 連結 MySQL

```
In [4]: cnx = mysql.connector.connect(user='root', password='123456', database='sakila')

In [5]: cursor = cnx.cursor()

In [6]: query = "SELECT rental_id, rental_date, customer_id FROM rental"

In [7]: cursor.execute(query)

In [8]: result1 = cursor.fetchone()

In [9]: print(result1)
(1, datetime.datetime(2005, 5, 24, 22, 53, 30), 130)

In [10]: result2 = cursor.fetchmany(6)    取出前6筆

In [11]: print(result2)
[(2, datetime.datetime(2005, 5, 24, 22, 54, 33), 459), (3, datetime.datetime(2005, 5, 24, 23, 3, 39), 408), (4, datetime.datetime(2005, 5, 24, 23, 4, 41), 333), (5, datetime.datetime(2005, 5, 24, 23, 5, 21), 222), (6, datetime.datetime(2005, 5, 24, 23, 8, 7), 510), (7, datetime.datetime(2005, 5, 24, 23, 11, 53), 269)]
```

Python demo

補充篇-行銷案例應用

線上交易銷售資料

- 下載網址：<https://www.kaggle.com/vijayuv/onlineretail>
- 下載檔名：OnlineRetail.csv.zip
- 解壓縮後檔名：**OnlineRetail.csv**
- 解壓縮後大小：43.4MB
- 參考資料：
<https://towardsdatascience.com/data-driven-growth-with-python-part-1-know-your-metrics-812781e66a5b>
- 下載：
<https://github.com/rwepa/DataDemo/blob/master/OnlineRetail.csv.zip>

檢視資料

- 檔案筆數：541909列
- 欄位數：8欄

A	B	C	D	E	F	G	H	
1	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
541905	581587	23256	CHILDRENS CUTLERY SPACEBOY	4	12/9/2011 12:50	4.15	12680	France
541906	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12/9/2011 12:50	0.85	12680	France
541907	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12/9/2011 12:50	2.1	12680	France
541908	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/2011 12:50	4.15	12680	France
541909	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/2011 12:50	4.15	12680	France
541910	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12/9/2011 12:50	4.95	12680	France

安裝與載入套件

```
# conda install plotly

# 載入套件
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from plotly.offline import plot # method 1: plot(fig)
import plotly.express as px # method 2: fig.show()
import plotly.graph_objs as go

# 設定 plotly 繪圖預設顯示於瀏覽器
import plotly.io as pio
pio.renderers.default='browser'

# 讀入CSV檔案
df = pd.read_csv('C:/mydata/OnlineRetail.csv')
```

使用 plotly 套件

前5筆資料 head

In [3]:

```
....: df.head(5)
```

Out[3]:

```
InvoiceNo StockCode ... CustomerID          Country
0      536365    85123A ...     17850.0  United Kingdom
1      536365      71053 ...     17850.0  United Kingdom
2      536365    84406B ...     17850.0  United Kingdom
3      536365    84029G ...     17850.0  United Kingdom
4      536365    84029E ...     17850.0  United Kingdom
```

[5 rows x 8 columns]

後5筆資料 tail

In [4]:

```
...: df.tail()
```

Out[4]:

```
InvoiceNo StockCode ... CustomerID Country
541904    581587   22613 ... 12680.0 France
541905    581587   22899 ... 12680.0 France
541906    581587   23254 ... 12680.0 France
541907    581587   23255 ... 12680.0 France
541908    581587   22138 ... 12680.0 France
```

[5 rows x 8 columns]

欄位名稱

In [5]:

```
....: df.columns
```

Out[5]:

```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

資料摘要 describe – 預設顯示數值摘要

數值八大摘要：

1. 計數
2. 平均值
3. 標準差
4. 最小值
5. 25百分位數
6. 50百分位數(中位數)
7. 75百分位數
8. 最大值

In [7]:

```
....: df.describe()
```

Out[7]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

資料好像怪怪的！
實務應用先進行預處理
^_~

資料摘要-類別, Country 唯一值為38個

In [8]:

```
....  
....: df.describe(include = 'object')
```

Out[8]:

	InvoiceNo	StockCode	...	InvoiceDate	Country
count	541909	541909	...	541909	541909
unique	25900	4070	...	23260	38
top	573585	85123A	...	10/31/2011 14:41	United Kingdom
freq	1114	2313	...	1114	495478

[4 rows x 5 columns]

資料摘要 - 數值 + 類別

```
In [9]: df.describe(include = 'all')  
Out[9]:
```

	InvoiceNo	StockCode	...	CustomerID	Country
count	541909	541909	...	406829.000000	541909
unique	25900	4070	...	NaN	38
top	573585	85123A	...	NaN	United Kingdom
freq	1114	2313	...	NaN	495478
mean	NaN	NaN	...	15287.690570	NaN
std	NaN	NaN	...	1713.600303	NaN
min	NaN	NaN	...	12346.000000	NaN
25%	NaN	NaN	...	13953.000000	NaN
50%	NaN	NaN	...	15152.000000	NaN
75%	NaN	NaN	...	16791.000000	NaN
max	NaN	NaN	...	18287.000000	NaN

[11 rows x 8 columns]



找出有用的特徵(有意義的屬性)

```
In [5]:  
....: df.columns  
Out[5]:  
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',  
       'UnitPrice', 'CustomerID', 'Country'],  
      dtype='object')
```

- InvoiceNo
- Quantity
- InvoiceDate
- Unit Price
- Customer ID

IMPORTANT



企業最想分析什麼類型資料?

將 InvoiceDate 欄位轉換為日期

```
In [10]: df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])  
...: df['InvoiceDate']
```

Out[10]:

```
0      2010-12-01 08:26:00  
1      2010-12-01 08:26:00  
2      2010-12-01 08:26:00  
3      2010-12-01 08:26:00  
4      2010-12-01 08:26:00
```

...

```
541904    2011-12-09 12:50:00  
541905    2011-12-09 12:50:00  
541906    2011-12-09 12:50:00  
541907    2011-12-09 12:50:00  
541908    2011-12-09 12:50:00
```

```
Name: InvoiceDate, Length: 541909, dtype: datetime64[ns]
```





新增評估欄位 InvoiceYearMonth 發票年月

In [11]:

```
.... df['InvoiceYearMonth'] = df['InvoiceDate'].map(lambda date: 100*date.year + date.month)
.... df['InvoiceYearMonth']
```

Out[11]:

```
0      201012
1      201012
2      201012
3      201012
4      201012
...
541904  201112
541905  201112
541906  201112
541907  201112
541908  201112
Name: InvoiceYearMonth, Length: 541909, dtype: int64
```

$$100 * 2010 + 12 = 201012$$





新增評估欄位 Revenue 收入

In [13]:

```
.... df[ 'Revenue' ] = df[ 'UnitPrice' ] * df[ 'Quantity' ]
.... df[ 'Revenue' ] _____ _____
```

Out[13]:

0	15.30
1	20.34
2	22.00
3	20.34
4	20.34
	...
541904	10.20
541905	12.60
541906	16.60
541907	16.60
541908	14.85

Name: Revenue, Length: 541909, dtype: float64



建立年月為群組,收入小計資料

In [14]:

```
....: df_revenue = df.groupby(['InvoiceYearMonth'])['Revenue'].sum().reset_index()  
....: df_revenue
```

Out[14]:

	InvoiceYearMonth	Revenue
0	201012	748957.020
1	201101	560000.260
2	201102	498062.650
3	201103	683267.080
4	201104	493207.121
5	201105	723333.510
6	201106	691123.120
7	201107	681300.111
8	201108	682680.510
9	201109	1019687.622
10	201110	1070704.670
11	201111	1461756.250
12	201112	433668.010

圖1.每月收入統計圖

```
# 每月收入統計圖
plot_data = [
    go.Scatter(
        x=df_revenue['InvoiceYearMonth'],
        y=df_revenue['Revenue'],
    )
]

plot_layout = go.Layout(
    xaxis={"type": "category"},
    title='圖1.每月收入統計圖')
fig = go.Figure(data=plot_data, layout=plot_layout)
```

繪圖 plot(fig)



繪圖 fig.show()



1. 每月收入向上增加趨勢.
2. 2011年2,4月達到最低點.



每月收入變化百分比 pct_change

```
#####
# 每月收入變化百分比
#####

# 使用 pct_change() 函數計算"每月收入百分比變化"

df_revenue[ 'MonthlyGrowth' ] = df_revenue[ 'Revenue' ].pct_change()
```

```
In [18]: df_revenue
```

```
Out[18]:
```

```
# (560000.260-748957.020)/748957.020 = -0.252293
```

	InvoiceYearMonth	Revenue	MonthlyGrowth
0	201012	748957.020	NaN
1	201101	560000.260	-0.252293
2	201102	498062.650	-0.110603
3	201103	683267.080	0.371850
4	201104	493207.121	-0.278163
5	201105	723333.510	0.466592
6	201106	691123.120	-0.044530
7	201107	681300.111	-0.014213
8	201108	682680.510	0.002026
9	201109	1019687.622	0.493653
10	201110	1070704.670	0.050032
11	201111	1461756.250	0.365228
12	201112	433668.010	-0.703324

繪圖

```
# 每月收入百分比變化統計圖
plot_data = [
    go.Scatter(
        x=df_revenue.query("InvoiceYearMonth < 201112")['InvoiceYearMonth'],
        y=df_revenue.query("InvoiceYearMonth < 201112")['MonthlyGrowth'],
    )
]

plot_layout = go.Layout(
    xaxis={"type": "category"},
    title='圖2.每月收入百分比變化統計圖'
)

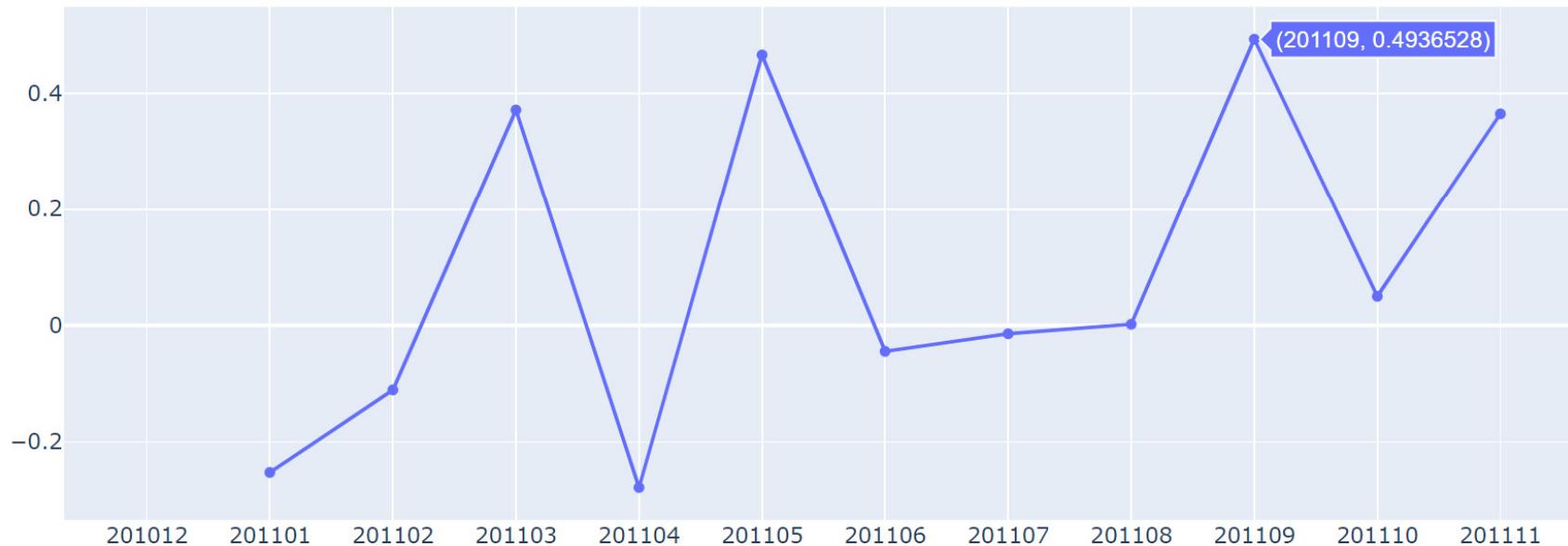
fig = go.Figure(data=plot_data, layout=plot_layout)

fig.show()
```

圖2.每月收入百分比變化統計圖

圖2.每月收入百分比變化統計圖

- 哪一月份達到最高月收入百分比變化?
 - 哪一月份達到最低月收入百分比變化?
- 應再深入研究 WHY?



pandas 資料處理技巧

- groupby 群組小計
- rename 欄位重新命名
- sort_values 排序

依國家別次數統計表

```
In [21]: df_country = df.groupby(['Country']).size().reset_index()
```

```
In [22]: df_country
```

Out[22]:

	Country	0
0	Australia	1259
1	Austria	401
2	Bahrain	19
3	Belgium	2069
4	Brazil	32
5	Canada	151

按字母排序，較難理解？

資料處理

```
# 欄位重新命名  
df_country.rename(columns={0: 'Count'}, inplace=True)  
  
# 排序  
df_country.sort_values(by=[ 'Count' ], ascending=False)  
  
# 建立篩選 United Kingdom 資料集  
df_uk = df.query("Country=='United Kingdom'").reset_index(drop=True)  
df_uk # 495478 rows x 10 columns
```

資料 df_uk

df_uk - DataFrame

Index	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	InvoiceYearMonth	Revenue
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850	United Kingdom	201012	15.3
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	201012	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850	United Kingdom	201012	22
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	201012	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	201012	20.34
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00	7.65	17850	United Kingdom	201012	15.3
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00	4.25	17850	United Kingdom	201012	25.5
7	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00	1.85	17850	United Kingdom	201012	11.1



每月活動客戶數 - unique

```
#####
# 建立每月活動客戶數統計表
#####
# nunique: Count distinct observations over requested axis.
df_monthly_active = df_uk.groupby('InvoiceYearMonth')[['CustomerID']].nunique().reset_index()
df_monthly_active
```

	InvoiceYearMonth	CustomerID
0	201012	871
1	201101	684
2	201102	714
3	201103	923
4	201104	817
5	201105	985
6	201106	943
7	201107	899
8	201108	867
9	201109	1177
10	201110	1285
11	201111	1548
12	201112	617

繪圖

```
# 每月活動客戶數長條圖
plot_data = [
    go.Bar(
        x=df_monthly_active['InvoiceYearMonth'],
        y=df_monthly_active['CustomerID'],
    )
]

plot_layout = go.Layout(
    xaxis={"type": "category"},
    title='圖3.每月活動客戶數長條圖'
)

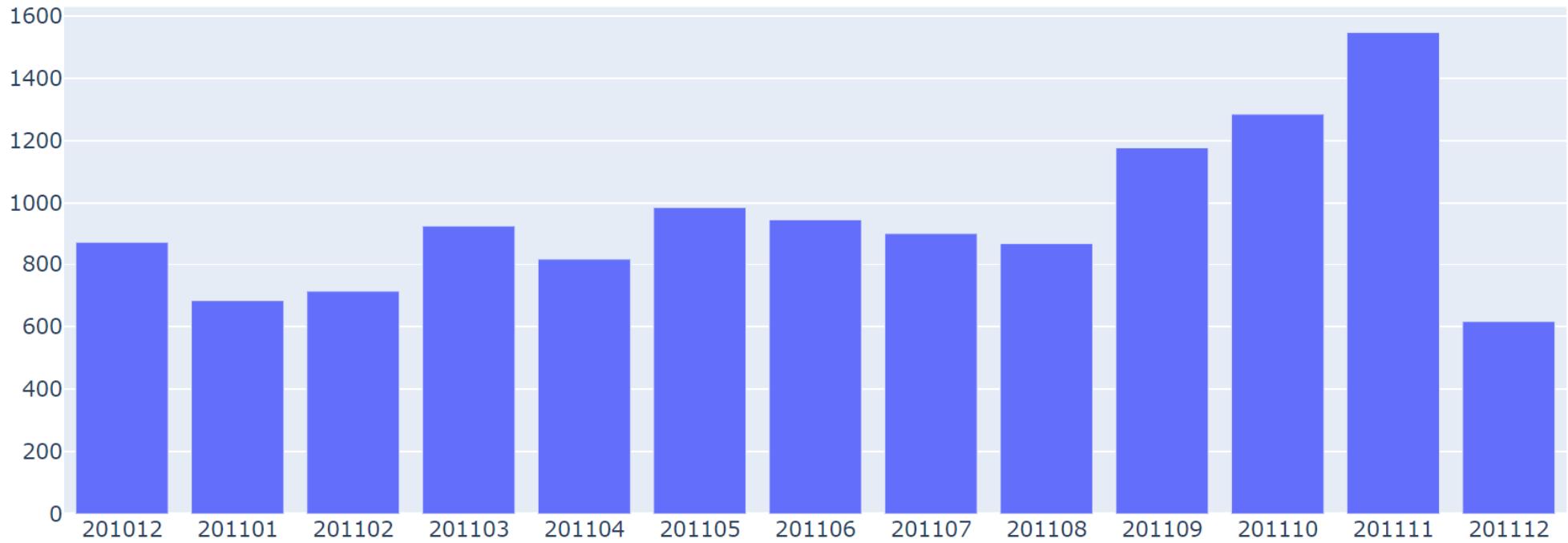
fig = go.Figure(data=plot_data, layout=plot_layout)

fig.show()
# 2011年4月客戶數從3月的 923降低至 817, 減少  $(817-923)/923 = -11.5\%$ 
```



圖3.每月活動客戶數長條圖

圖3.每月活動客戶數長條圖



每月每筆訂單平均收入

```
In [30]: df_monthly_order_avg = df_uk.groupby('InvoiceYearMonth')['Revenue'].mean().reset_index()
```

```
In [31]: df_monthly_order_avg
```

```
Out[31]:
```

	InvoiceYearMonth	Revenue
0	201012	16.865860
1	201101	13.614680
2	201102	16.093027
3	201103	16.716166
4	201104	15.773380
5	201105	17.713823
6	201106	16.714748
7	201107	15.723497
8	201108	17.315899
9	201109	18.931723
10	201110	16.093582
11	201111	16.312383
12	201112	16.247406

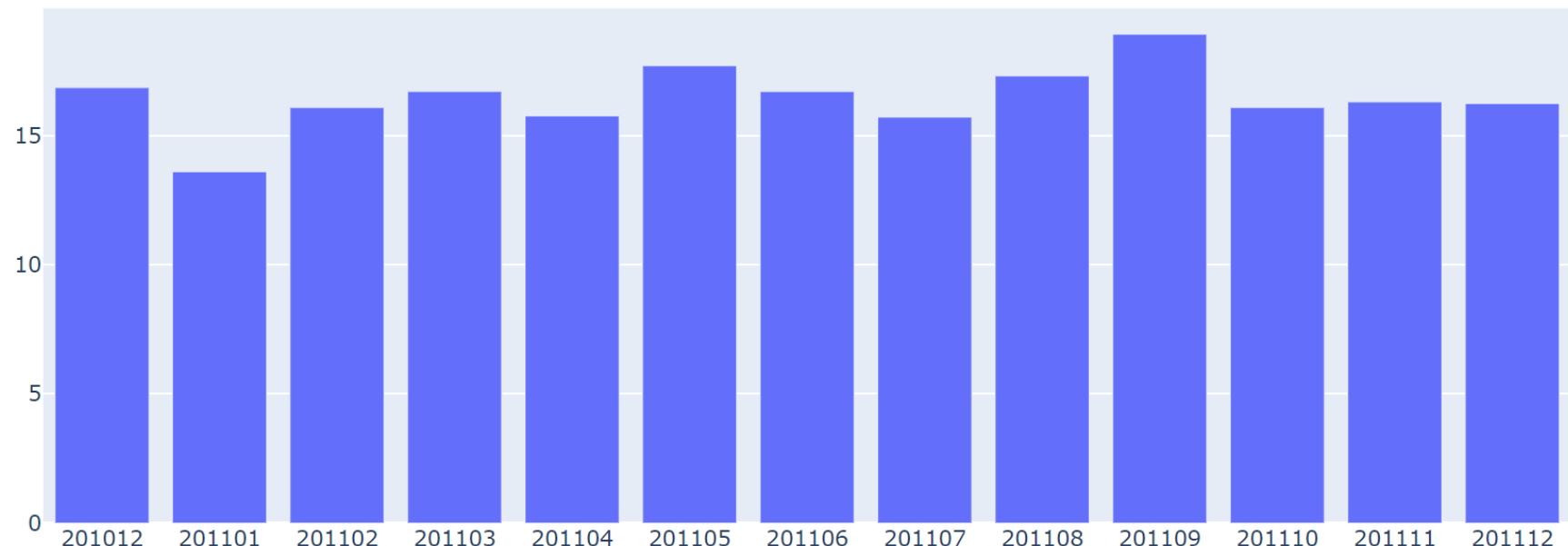
繪圖

```
# 每月每筆訂單平均收入長條圖
plot_data = [
    go.Bar(
        x=df_monthly_order_avg['InvoiceYearMonth'],
        y=df_monthly_order_avg['Revenue'],
    )
]

plot_layout = go.Layout(
    xaxis={"type": "category"},
    title='圖4.每月每筆訂單平均收入長條圖'
)
fig = go.Figure(data=plot_data, layout=plot_layout)
fig.show()
```

圖4.每月每筆訂單平均收入長條圖

圖4.每月每筆訂單平均收入長條圖



- 整體每月每筆訂單平均收入有逐漸下降趨勢。
- 最大值為2011年9月，每月每筆訂單平均收入為18.93。
- 最小值為2011年1月，每月每筆訂單平均收入為13.61。

建立評估欄位: 新客戶, 舊客戶

- 新客戶比率 (New Customer Ratio):
 - 紿定時間範圍
 - 找出第1次消費之客戶
- 現有客戶維持率或客戶保留率 (Retention rate), 客戶流失率 (Churn rate)
 - 紿定時間範圍
 - 客戶仍有消費情形

建立第1次消費之客戶

```
# 建立第1次消費之客戶  
# 以客戶編號為群組，計算 Min{發票日期}  
df_min_purchase = df_uk.groupby('CustomerID').InvoiceDate.min().reset_index()  
  
# 修改欄位名稱  
df_min_purchase.columns = ['CustomerID', 'MinPurchaseDate']  
df_min_purchase
```

使用 min

客戶之第1次消費統計表

	CustomerID	MinPurchaseDate
0	12346.0	2011-01-18 10:01:00
1	12747.0	2010-12-05 15:38:00
2	12748.0	2010-12-01 12:48:00
3	12749.0	2011-05-10 15:25:00
4	12820.0	2011-01-17 12:34:00
...
3945	18280.0	2011-03-07 09:52:00
3946	18281.0	2011-06-12 10:53:00
3947	18282.0	2011-08-05 13:35:00
3948	18283.0	2011-01-06 14:14:00
3949	18287.0	2011-05-22 10:39:00

[3950 rows x 2 columns]

新增日期轉換欄位

```
# 新增欄位 - 轉換日期格式: 西元年月
```

```
df_min_purchase['MinPurchaseYearMonth'] =  
df_min_purchase['MinPurchaseDate'].map(lambda date: 100*date.year + date.month)
```

df_min_purchase	CustomerID	MinPurchaseDate	MinPurchaseYearMonth
0	12346.0	2011-01-18 10:01:00	201101
1	12747.0	2010-12-05 15:38:00	201012
2	12748.0	2010-12-01 12:48:00	201012
3	12749.0	2011-05-10 15:25:00	201105
4	12820.0	2011-01-17 12:34:00	201101
...
3945	18280.0	2011-03-07 09:52:00	201103
3946	18281.0	2011-06-12 10:53:00	201106
3947	18282.0	2011-08-05 13:35:00	201108
3948	18283.0	2011-01-06 14:14:00	201101
3949	18287.0	2011-05-22 10:39:00	201105

[3950 rows x 3 columns]

df_uk

	InvoiceNo	StockCode	...	InvoiceYearMonth	Revenue
0	536365	85123A	...	201012	15.30
1	536365	71053	...	201012	20.34
2	536365	84406B	...	201012	22.00
3	536365	84029G	...	201012	20.34
4	536365	84029E	...	201012	20.34
...
495473	581585	22466	...	201112	23.40
495474	581586	22061	...	201112	23.60
495475	581586	23275	...	201112	30.00
495476	581586	21217	...	201112	214.80
495477	581586	20685	...	201112	70.80

[495478 rows x 10 columns]



合併 merge(左表格, 右表格, 共同欄位)

```
# 合併 {MinPurchaseDate, MinPurchaseYearMonth} 至 df_uk 之最右側  
df_uk = pd.merge(df_uk, df_min_purchase, on='CustomerID')  
df_uk # 361878 rows x 12 columns
```

合併後資料 df_uk

	InvoiceNo	StockCode	...	MinPurchaseDate	MinPurchaseYearMonth
0	536365	85123A	...	2010-12-01 08:26:00	201012
1	536365	71053	...	2010-12-01 08:26:00	201012
2	536365	84406B	...	2010-12-01 08:26:00	201012
3	536365	84029G	...	2010-12-01 08:26:00	201012
4	536365	84029E	...	2010-12-01 08:26:00	201012
...
361873	581416	22809	...	2011-12-08 14:58:00	201112
361874	581416	22807	...	2011-12-08 14:58:00	201112
361875	581416	72349B	...	2011-12-08 14:58:00	201112
361876	581416	22809	...	2011-12-08 14:58:00	201112
361877	581416	23487	...	2011-12-08 14:58:00	201112
[361878 rows x 12 columns]					

思考題:合併後資料筆數減少? 495478 rows --> 361878 rows! 消失13多萬列

建立評估欄位: UserType {New, Existing}

```
# 建立評估欄位: 客戶型態 UserType, 預設值為 New  
# 如果 InvoiceYearMonth 大於 MinPurchaseYearMonth, 表示之前為現有客戶, 將 UserType 改為 Existing  
  
df_uk['UserType'] = 'New'  
  
df_uk.loc[df_uk['InvoiceYearMonth'] > df_uk['MinPurchaseYearMonth'], 'UserType'] = 'Existing'
```

[列

, 行]

新,舊客戶的每月收入小計

```
# 新,舊客戶的每月收入小計  
  
# 計算每月新舊客戶收入小計  
df_user_type_revenue = df_uk.groupby(['InvoiceYearMonth', 'UserType'])['Revenue'].sum().reset_index()  
  
# 篩選資料並刪除前,後資料  
df_user_type_revenue = df_user_type_revenue.query("InvoiceYearMonth != 201012 and InvoiceYearMonth != 201112")  
df_user_type_revenue
```

	InvoiceYearMonth	UserType	Revenue
1	201101	Existing	195275.510
2	201101	New	156705.770
3	201102	Existing	220994.630
4	201102	New	127859.000
5	201103	Existing	296350.030
6	201103	New	160567.840
7	201104	Existing	268226.660

每月收入小計

	InvoiceYearMonth	UserType	Revenue
1	201101	Existing	195275.510
2	201101	New	156705.770
3	201102	Existing	220994.630
4	201102	New	127859.000
5	201103	Existing	296350.030
6	201103	New	160567.840
7	201104	Existing	268226.660

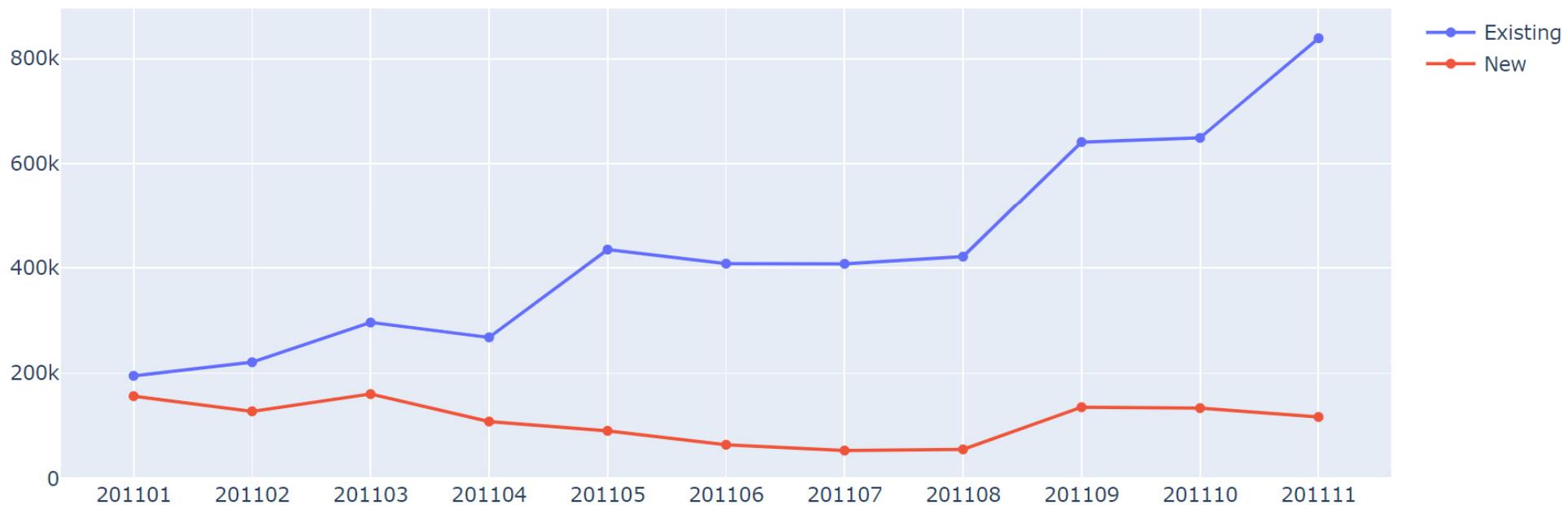
繪圖

```
# 繪圖
plot_data = [
    go.Scatter(
        x=df_user_type_revenue.query("UserType == 'Existing'")['InvoiceYearMonth'],
        y=df_user_type_revenue.query("UserType == 'Existing'")['Revenue'],
        name = 'Existing'
    ),
    go.Scatter(
        x=df_user_type_revenue.query("UserType == 'New'")['InvoiceYearMonth'],
        y=df_user_type_revenue.query("UserType == 'New'")['Revenue'],
        name = 'New'
    )
]

plot_layout = go.Layout(
    xaxis={"type": "category"},
    title='圖5.新,舊客戶的每月收入統計圖'
)
fig = go.Figure(data=plot_data, layout=plot_layout)
fig.show()
```

圖5.新,舊客戶的每月收入統計圖

圖5.新,舊客戶的每月收入統計圖



- 現有客戶(藍色)顯示每月收入有遞增的趨勢.
- 新客戶(紅色)則有些微的減少現象.

新客戶比率 (New Customer Ratio)

```
#####
# 新客戶比率 (New Customer Ratio)
#####

# 建立每月新客戶資料並刪除 NA
# 本例使用 unique 函數

df_user_ratio = df_uk.query("UserType == 'New'").groupby(['InvoiceYearMonth'])['CustomerID'].nunique()/
df_uk.query("UserType == 'Existing'").groupby(['InvoiceYearMonth'])['CustomerID'].nunique()

# 設定 index
df_user_ratio = df_user_ratio.reset_index()

# 刪除 NA
df_user_ratio = df_user_ratio.dropna()

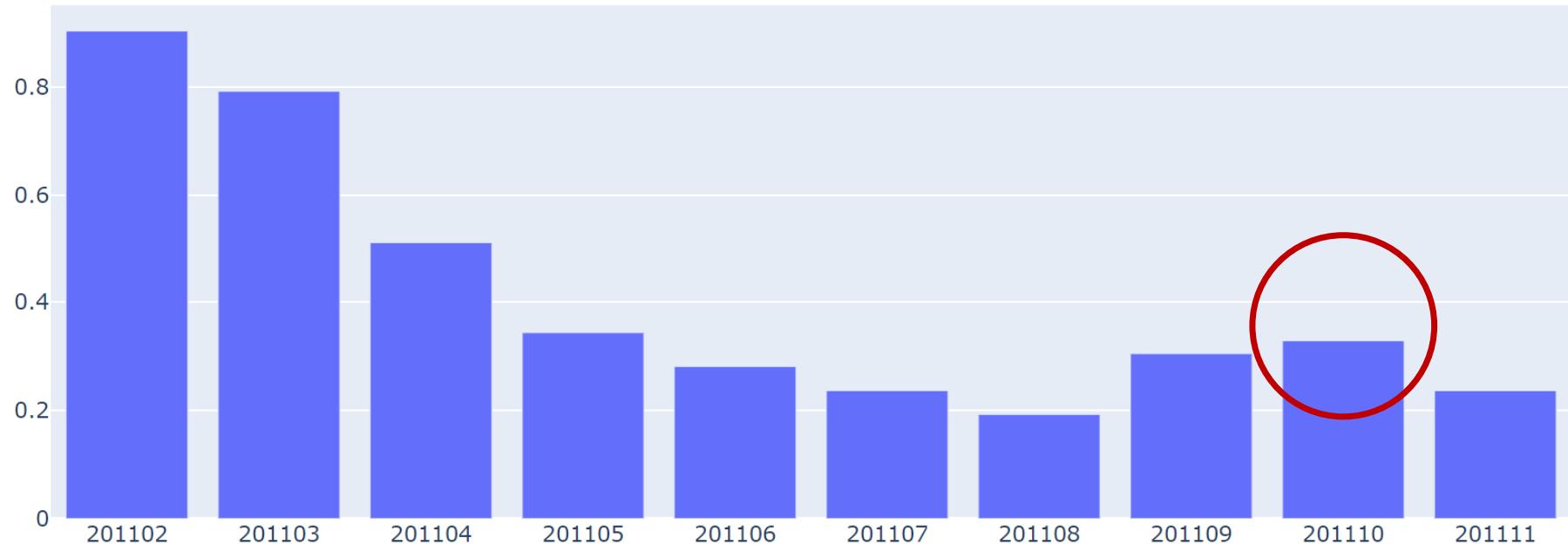
df_user_ratio
```

新客戶比率 df_user_ratio

	InvoiceYearMonth	CustomerID
1	201101	1.124224
2	201102	0.904000
3	201103	0.792233
4	201104	0.510166
5	201105	0.343793
6	201106	0.281250
7	201107	0.236589
8	201108	0.192572
9	201109	0.304878
10	201110	0.328852
11	201111	0.236422
12	201112	0.058319

圖6.新客戶比率統計圖

圖6.新客戶比率統計圖





每月客戶保留率 (Monthly Retention Rate)

```
#####
# 每月客戶保留率 (Monthly Retention Rate)
#####

# 每月客戶保留率=上個月以前保留的客戶/總活躍客戶數

# 計算活躍客戶數，依據 {CustomerID, InvoiceYearMonth} 為群組，計算收入總計。

df_user_purchase = df_uk.groupby(['CustomerID', 'InvoiceYearMonth'])['Revenue'].sum().reset_index()
df_user_purchase

# 使用 crosstab (交叉表) 建立客戶保留矩陣，crosstab 預設使用"發生次數"

df_retention = pd.crosstab(df_user_purchase['CustomerID'], df_user_purchase['InvoiceYearMonth']).reset_index()
df_retention
```

使用 crosstab 交叉表格

保留矩陣 df_retention

InvoiceYearMonth	CustomerID	201012	201101	...	201110	201111	201112
0	12346.0	0	1	...	0	0	0
1	12747.0	1	1	...	1	1	1
2	12748.0	1	1	...	1	1	1
3	12749.0	0	0	...	0	1	1
4	12820.0	0	1	...	1	0	1
...
3945	18280.0	0	0	...	0	0	0
3946	18281.0	0	0	...	0	0	0
3947	18282.0	0	0	...	0	0	1
3948	18283.0	0	1	...	1	1	1
3949	18287.0	0	0	...	1	0	0

[3950 rows x 14 columns]

保留率

```
# 建立字典物件，記錄月份，總活躍客戶數，保留客戶數
months = df_retention.columns[2:]
retention_array = []

for i in range(len(months)-1):
    retention_data = {}
    selected_month = months[i+1]
    prev_month = months[i]
    retention_data['InvoiceYearMonth'] = int(selected_month)
    retention_data['TotalUserCount'] = df_retention[selected_month].sum()
    retention_data['RetainedUserCount'] = df_retention[(df_retention[selected_month]>0) &
(df_retention[prev_month]>0)][selected_month].sum()
    retention_array.append(retention_data)
retention_array

# 轉換為資料框
df_retention = pd.DataFrame(retention_array)
df_retention

# 計算保留率
df_retention['RetentionRate'] = df_retention['RetainedUserCount']/df_retention['TotalUserCount']
df_retention
```

$$\text{每月客戶保留率} = \frac{\text{上個月以前保留的客戶}}{\text{總活躍客戶數}}$$

保留率 df_retention

	InvoiceYearMonth	TotalUserCount	RetainedUserCount	RetentionRate
0	201102	714	263	0.368347
1	201103	923	305	0.330444
2	201104	817	310	0.379437
3	201105	985	369	0.374619
4	201106	943	417	0.442206
5	201107	899	379	0.421580
6	201108	867	391	0.450980
7	201109	1177	417	0.354291
8	201110	1285	502	0.390661
9	201111	1548	616	0.397933
10	201112	617	402	0.651540

圖7.每月客戶保留率統計圖

圖7.每月客戶保留率統計圖

- 6月與8月達到2次高峰期
- 3月保留率最少



Python 模組

模組	功能	
Numpy	Large, multi-dimensional arrays and matrices	
Scipy	Optimization, linear algebra, integration, interpolation, FFT, signal and image processing	
Pandas	DataFrame object for data manipulation	
Matplotlib	Static, animated, and interactive visualizations	
Statsmodels	Statistical models	
Scikit-learn	Machine learning library	
Tensorflow	Deep learning	
Biopython	Biological computation	
Scanpy	Single-cell analysis	



參考資料

- RWEPA
 - <http://rwepa.blogspot.com/>
- Python 程式設計-李明昌 <免費電子書>
 - <http://rwepa.blogspot.com/2020/02/pythonprogramminglee.html>
- R入門資料分析與視覺化應用教學(付費)
 - <https://mastertalks.tw/products/r?ref=MCLEE>
- R商業預測與應用(付費)
 - <https://mastertalks.tw/products/r-2?ref=MCLEE>

謝謝您的聆聽

Q & A



李明昌

alan9956@gmail.com

<http://rwepa.blogspot.tw/>