AI與程式語言-第3章 Anaconda簡介與安裝

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/

主題

第1章 課程介紹 (iClass)

第2章 AI與程式語言簡介

第3章 Anaconda簡介與安裝

第4章 資料匯入探索

第5章 資料處理探索

第6章 資料視覺化探索

第7章 資料整合應用探索

大綱：

3.1 Python 簡介與安裝

3.2 Anaconda 簡介與安裝

3.3 Spyder 軟體簡介

3.4 變數

3.5 資料型別與運算子

3.6 資料物件

3.7 認識模組

3.1 Python 簡介與安裝

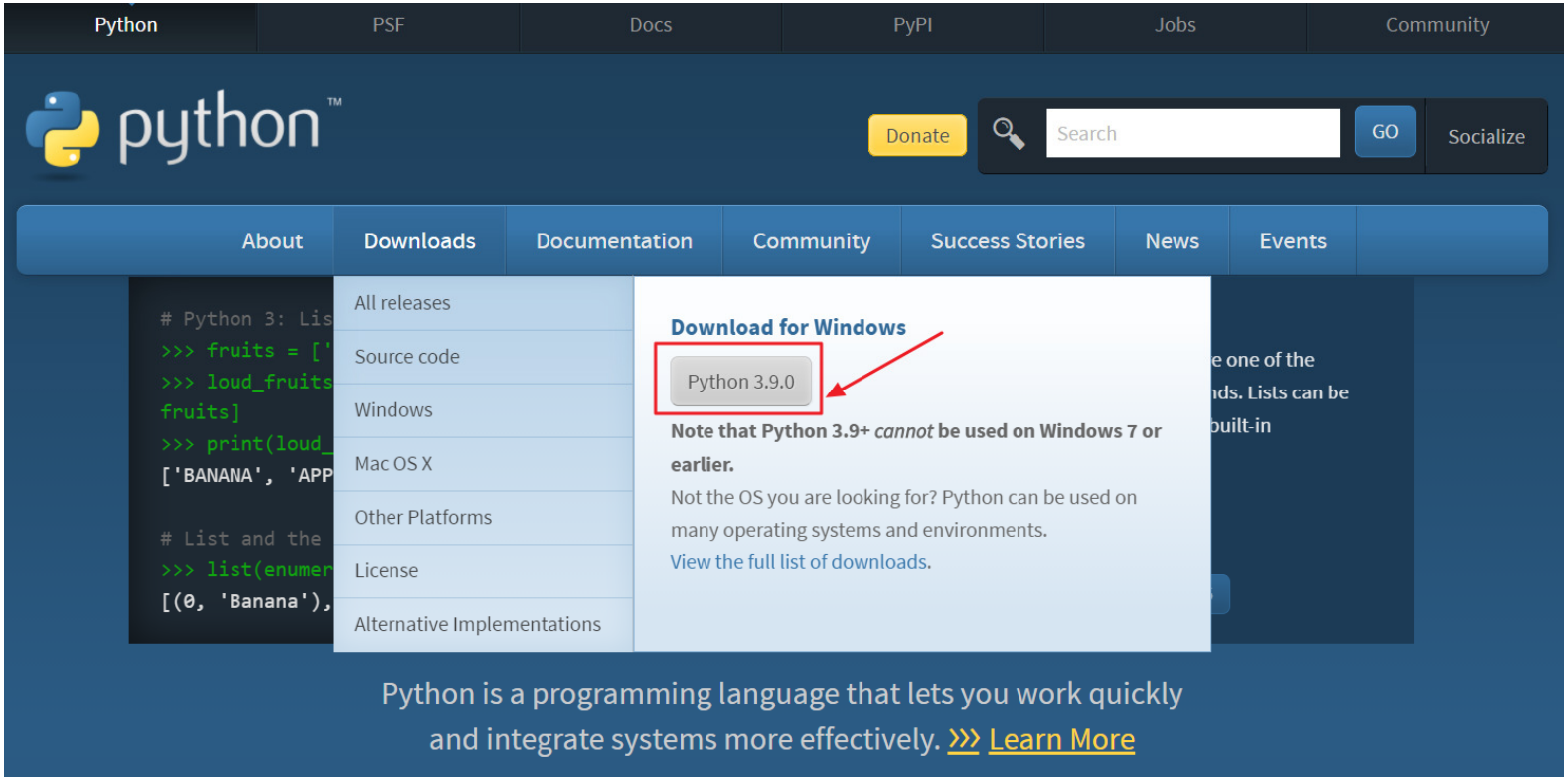
Python 簡介

- 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 3.10.6
- 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

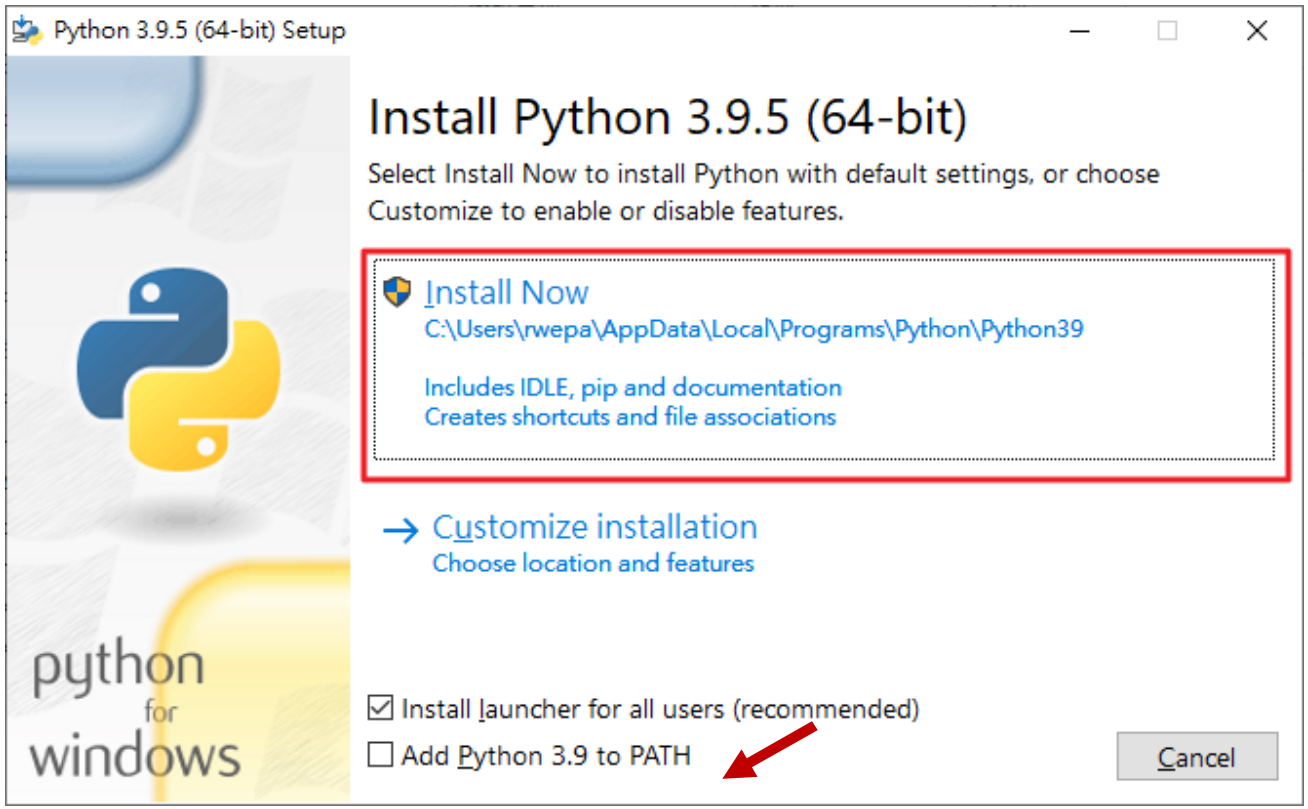
2022.9.6 最新版 3.10.6
【注意】本課程使用
Anaconda 軟體，已經內含
Python程式， 因此不用下載
Python 3.10.6

- https://www.python.org/

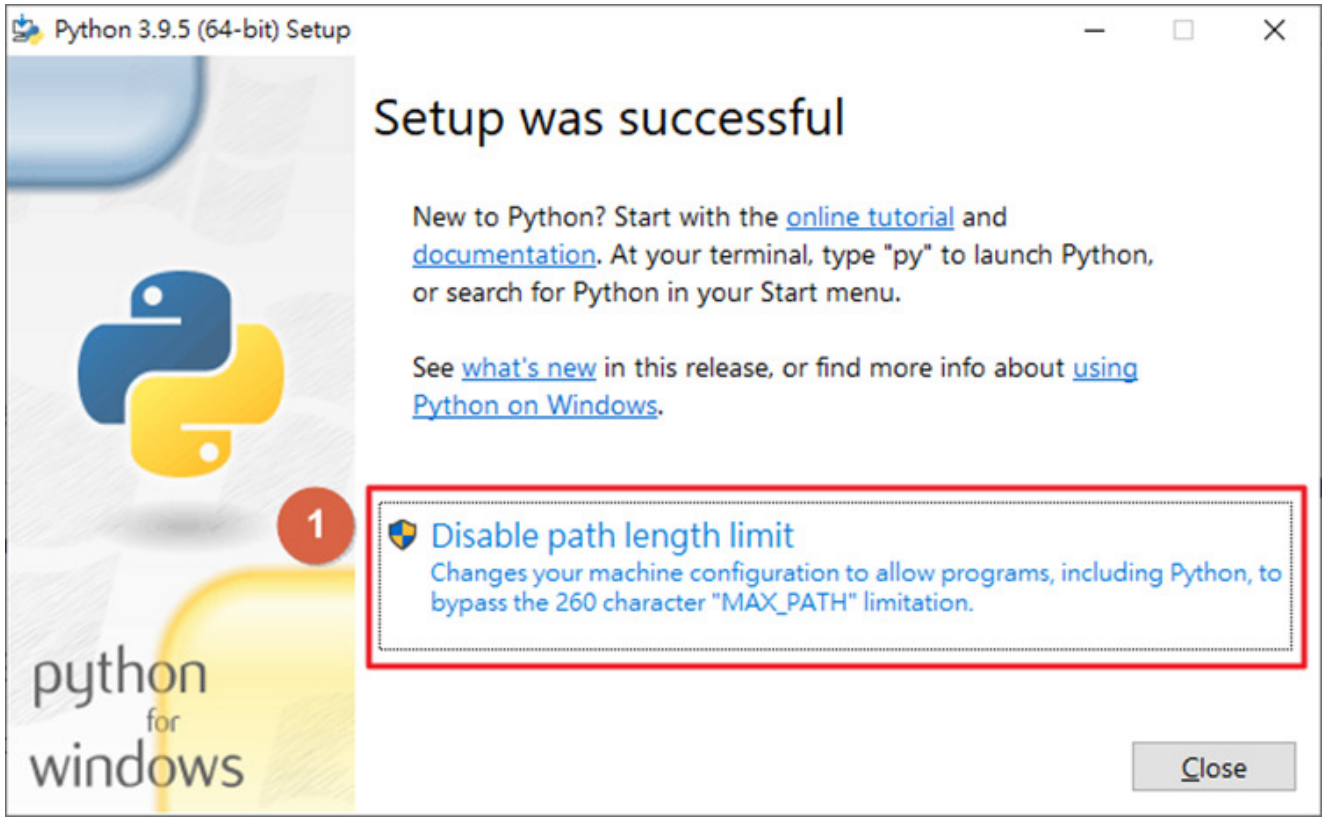


Python 安裝

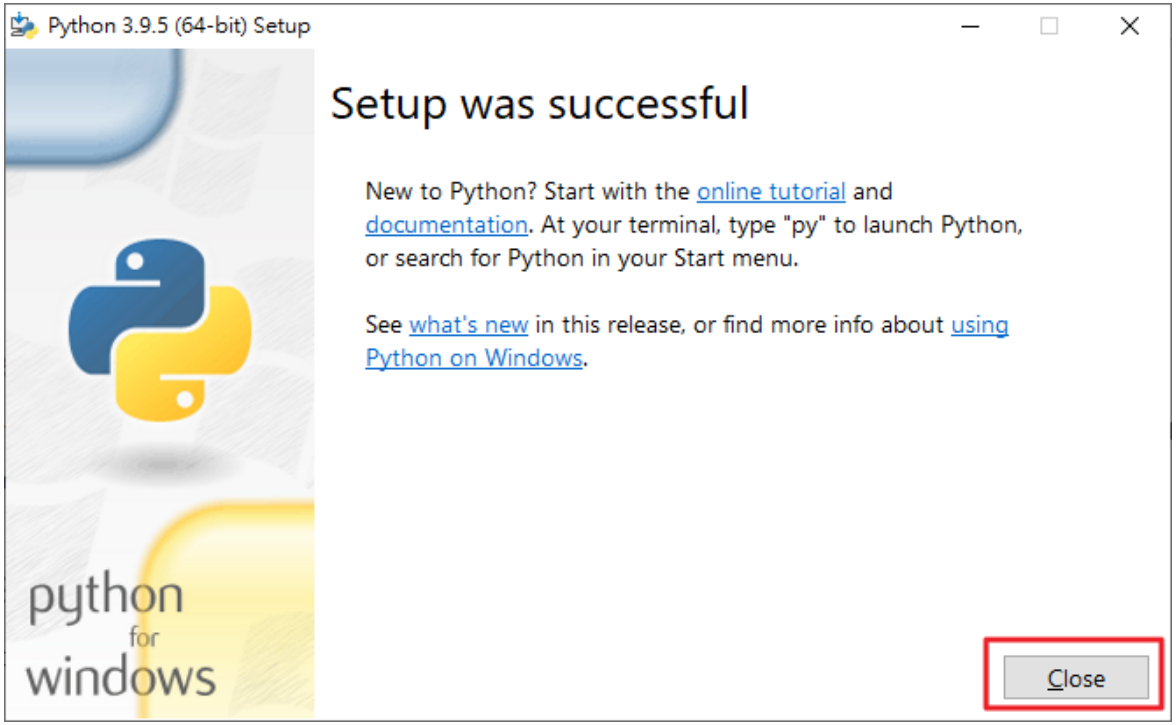
- python-3.9.5-amd64.exe



Python 安裝 – 取消字元長度限制

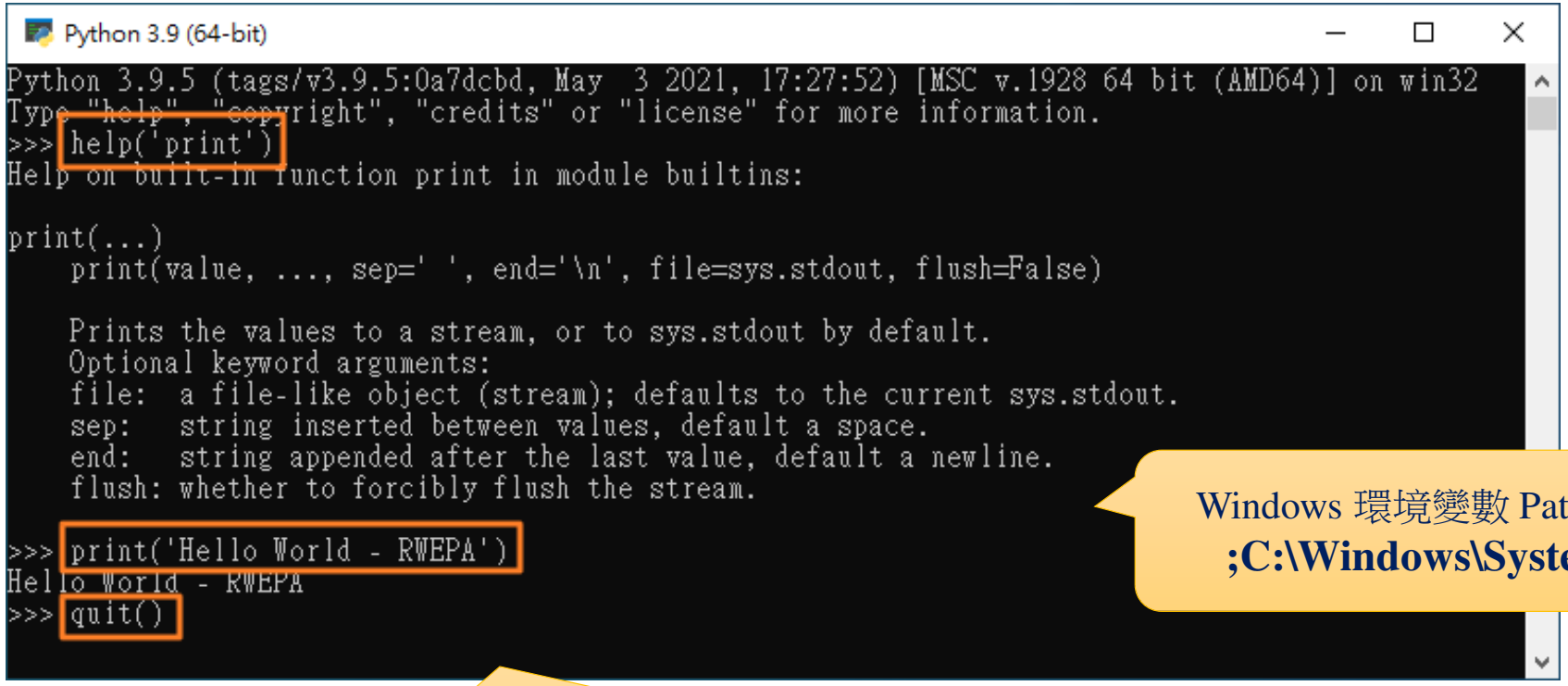


Python 安裝完成


Python 3.9.5 (64-bit) Setup
Setup was successful
New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.
See what's new in this release, or find more info about using Python on Windows.
python for windows
Close


Python 3.9
新增
IDLE (Python 3.9 64-bit)
新增
Python 3.9 (64-bit)
新增
Python 3.9 Manuals (64-bit)
新增
Python 3.9 Module Docs (64-bit)
新增

Python 執行 <方法1> Python 3.9(64-bit)



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

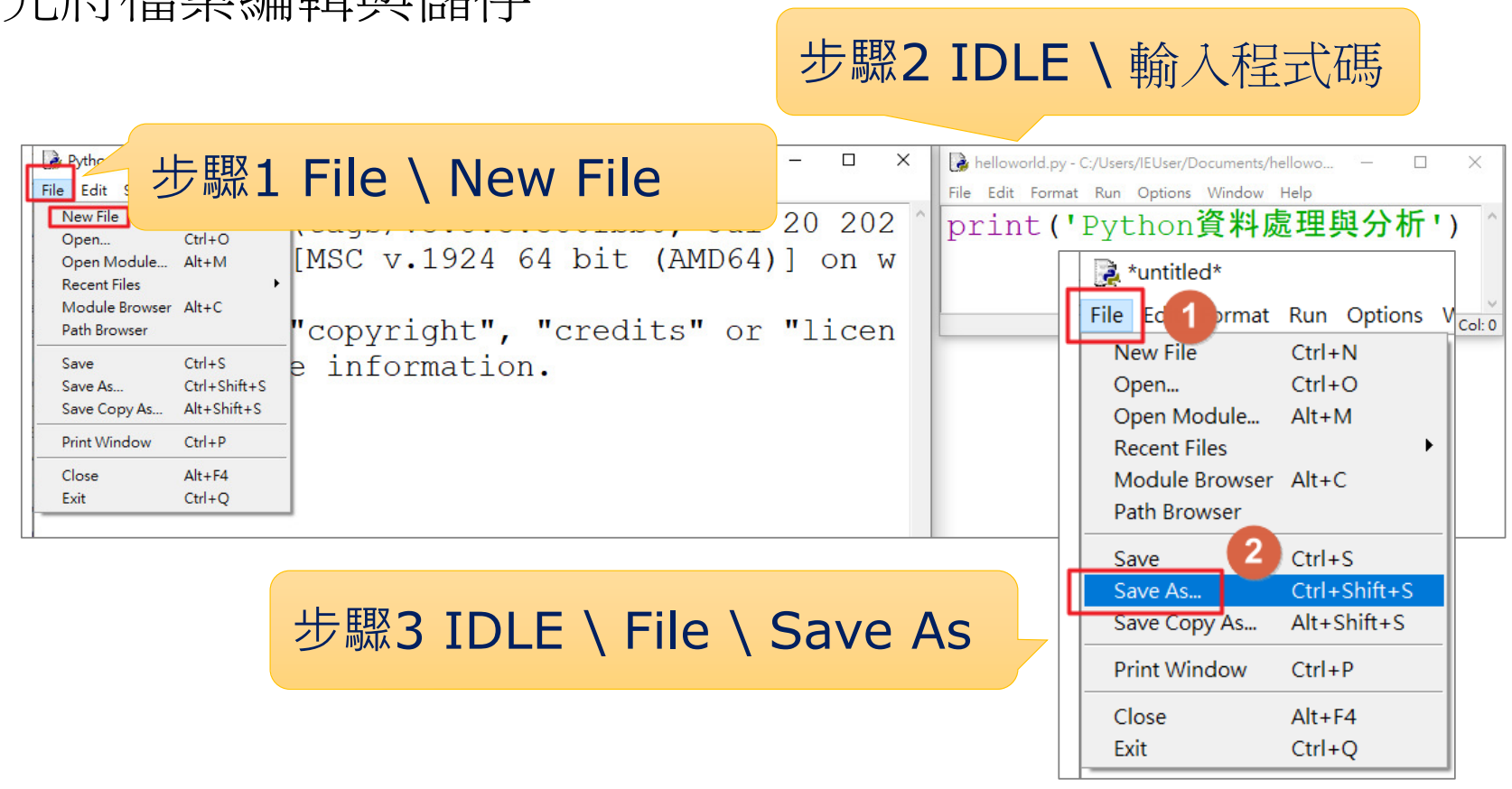
- help(print)
- print('Hello world')
- quit()

Python 執行 <方法2> 命令提示字元

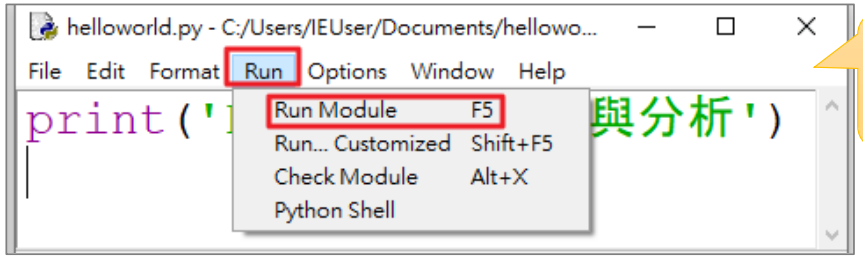
- python
- 1+2
- quit()

Python 執行 <方法3> IDLE模式

- IDLE (Integrated Development and Learning Environment)
  可先將檔案編輯與儲存

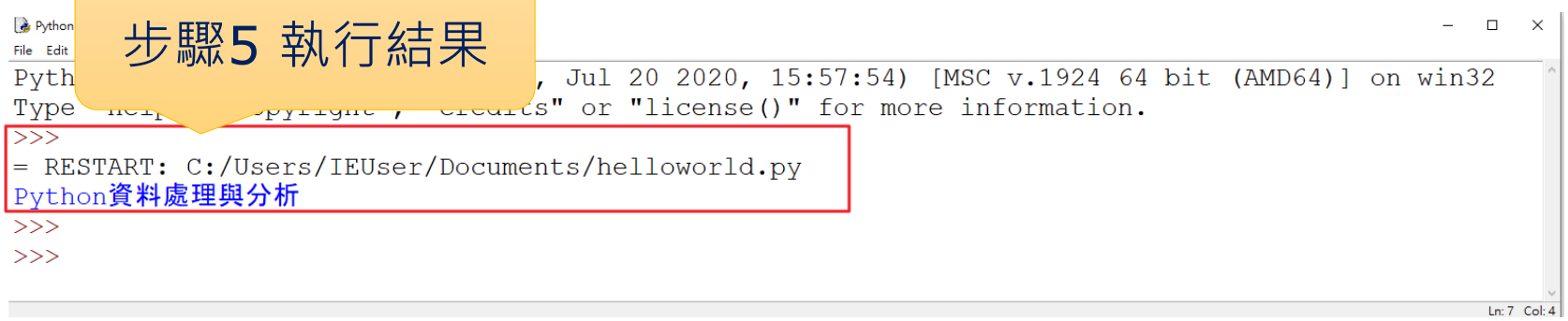


Python 執行 <方法3> IDLE模式 (續)



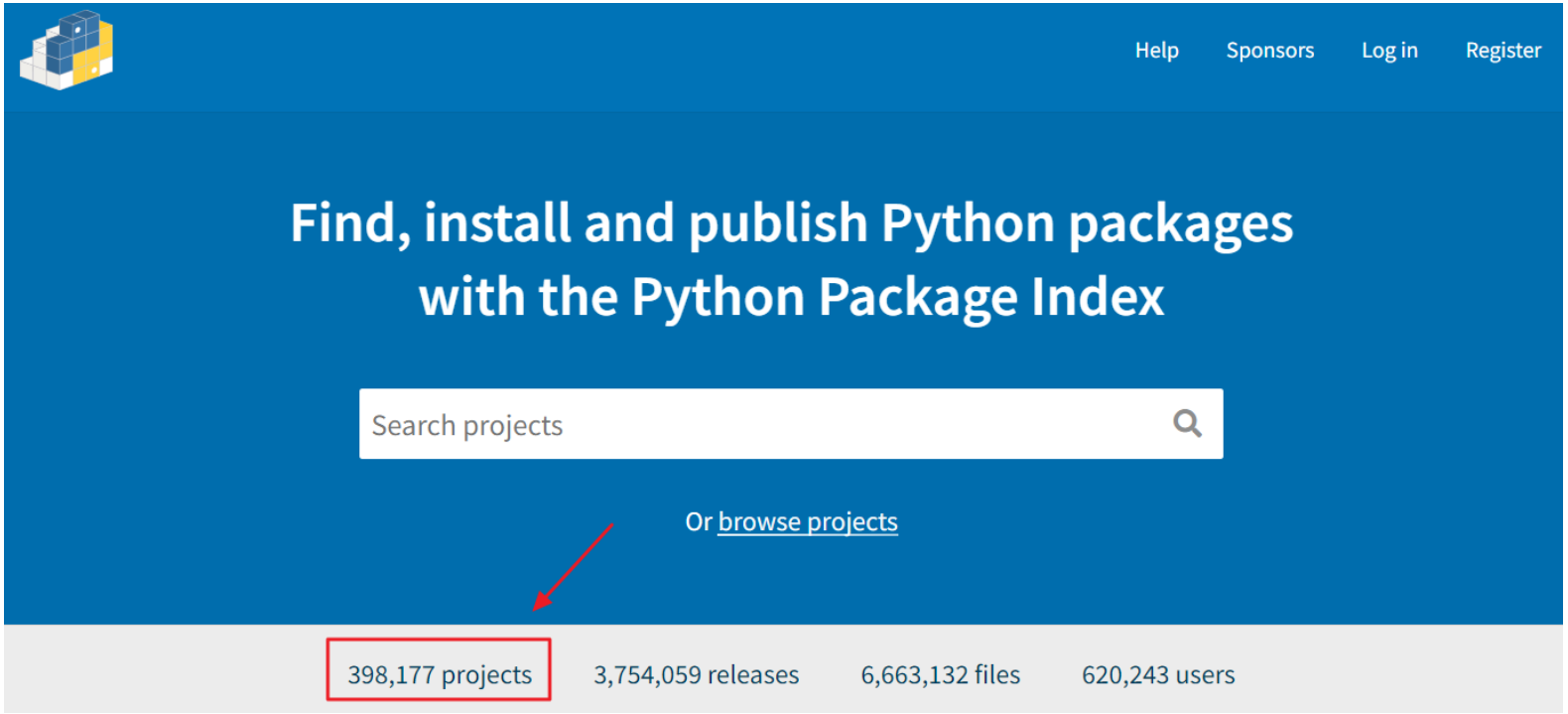
步驟4 Run \ Run Module

步驟5 執行結果

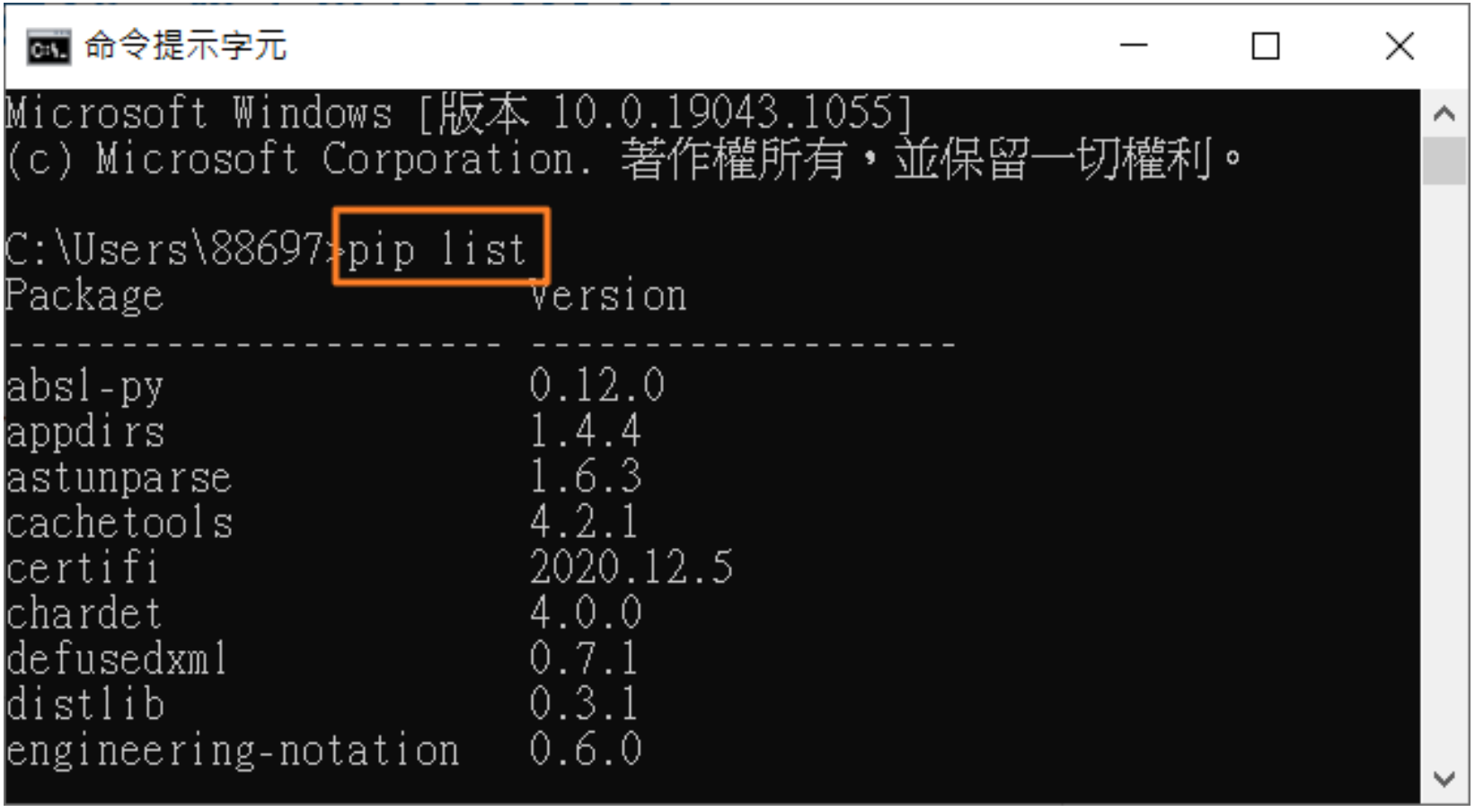


PyPI (Python Package Index)

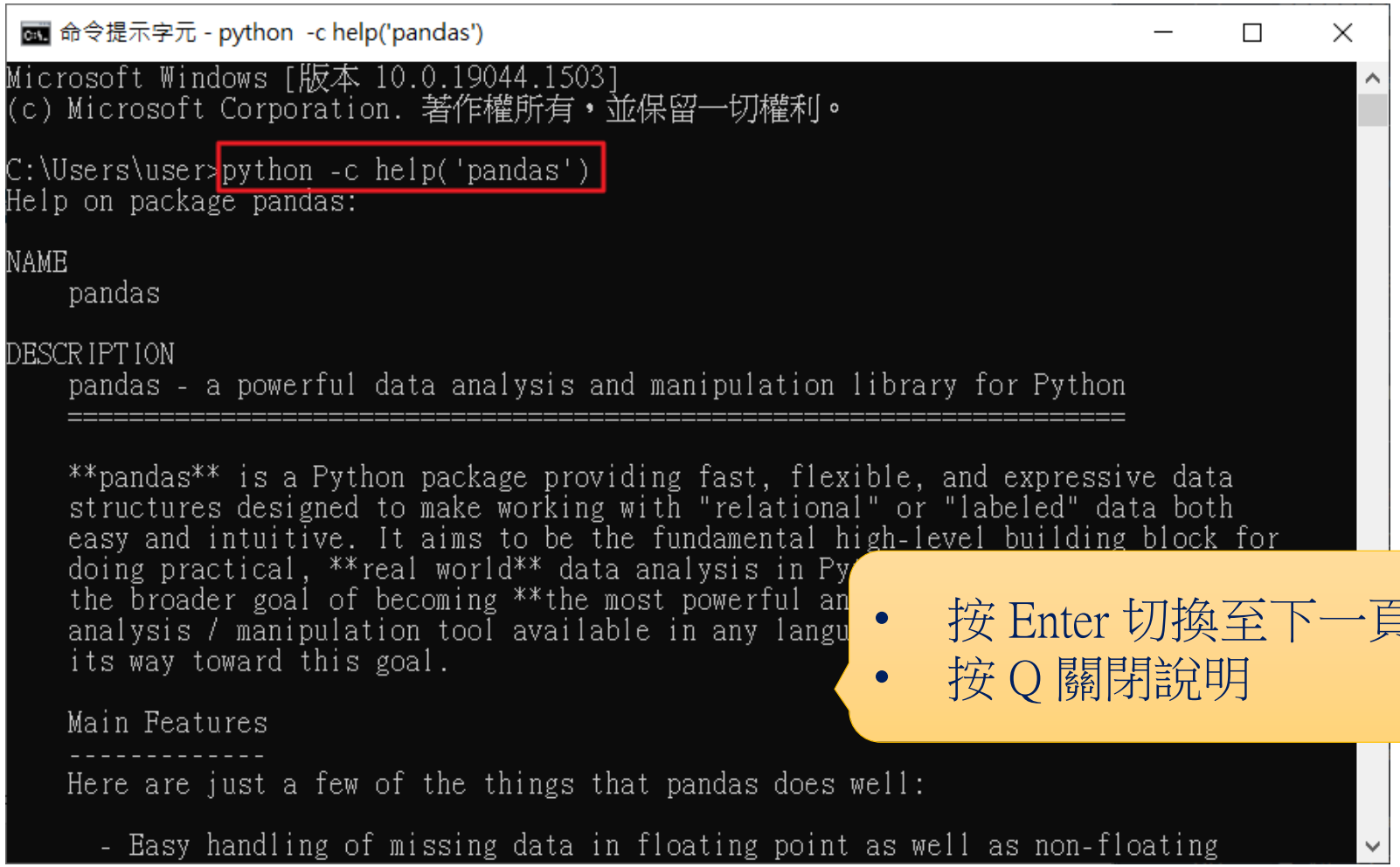
- https://pypi.org/
- 39萬多專案總表



已安裝模組 pip list



pandas 模組 - 線上說明 python -c help('pandas')



- 按 Enter 切換至下一頁
- 按 Q 關閉說明

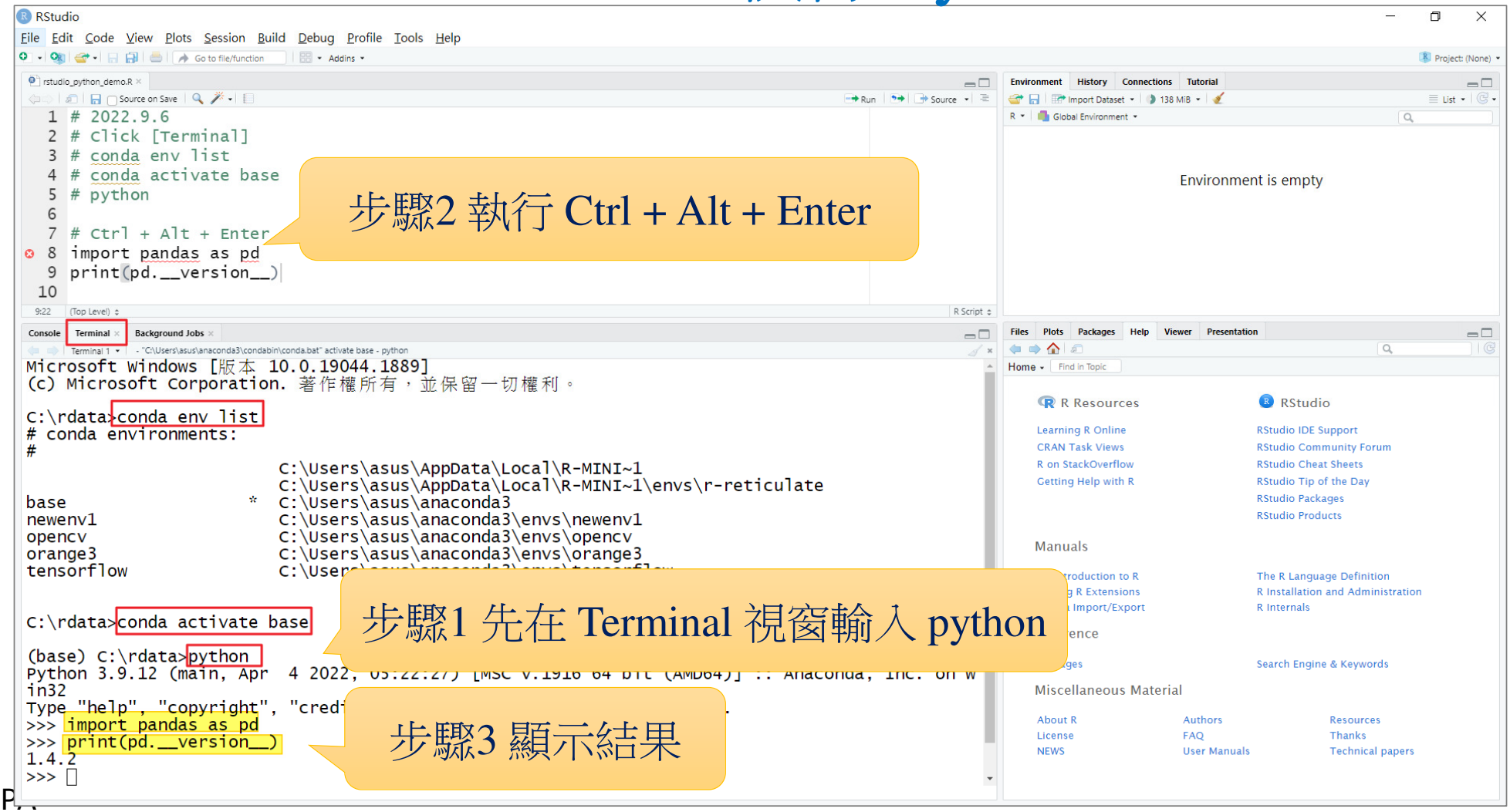
Python IDE

- Anaconda, 包括 **Spyder**, Jupyter notebook
  - 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



3.2 Anaconda 簡介與安裝

Anaconda 特性

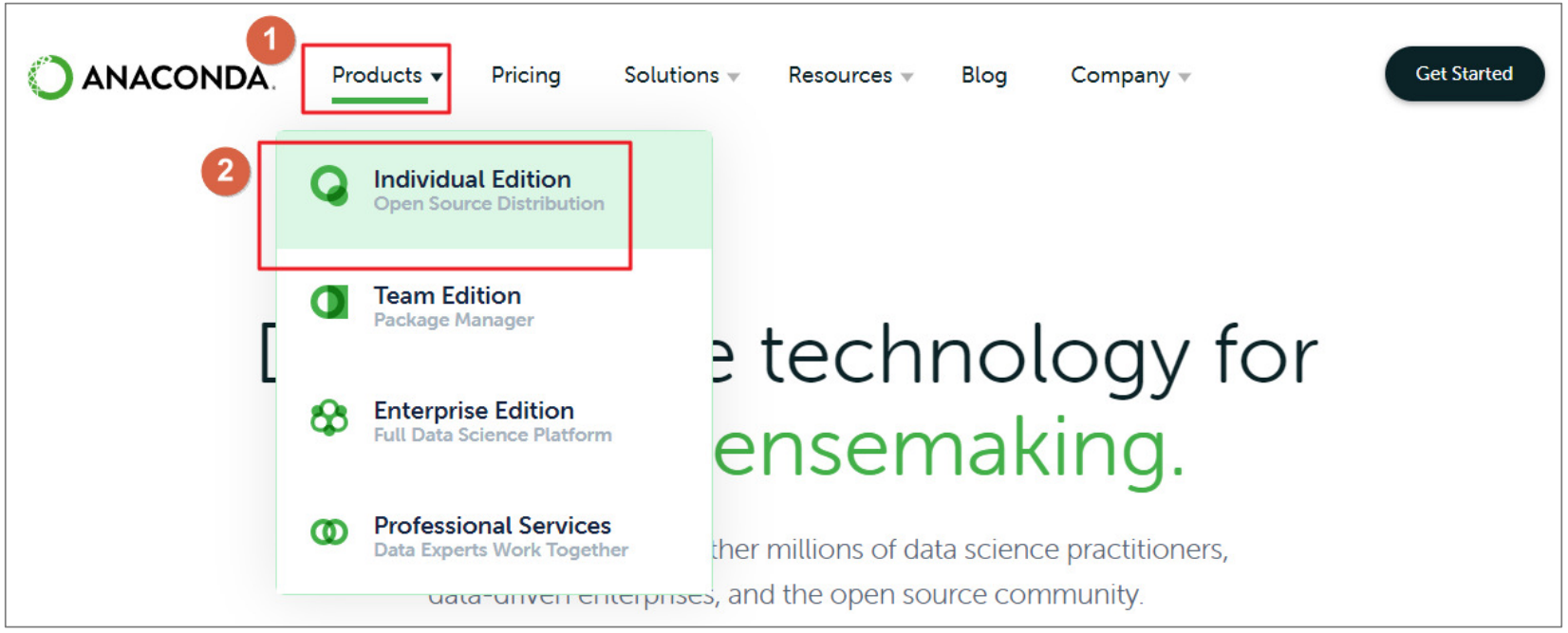
- Anaconda是一個免費、易於安裝/管理並支援Python語言
- 支援7500個以上的資料科學模組 (package)
- 支援模組的下載, 安裝, 更新
- 支援 Spyder (支援 Python IDE)
- 支援 jupyter notebook
- 支援機器學習模組 (scikit-learn)
- 支援深度學習模組 (Tensorflow, Keras)
- 支援 Windows、Mac OS X和Linux 跨平台

Anaconda 特性 (續)

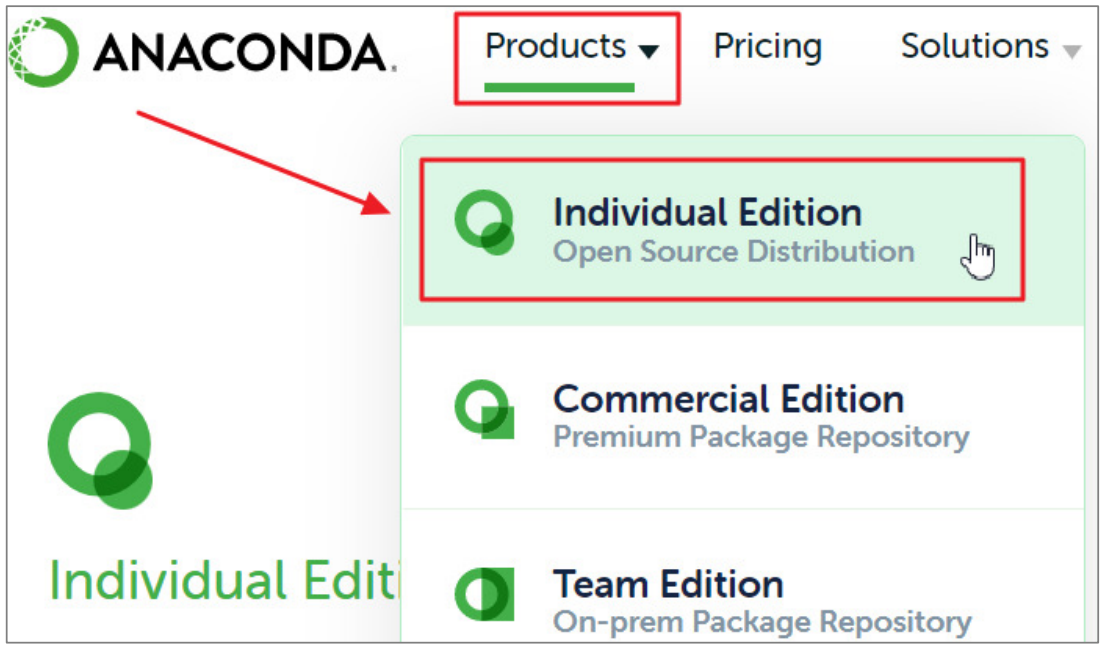
source: https://www.anaconda.com/distribution/

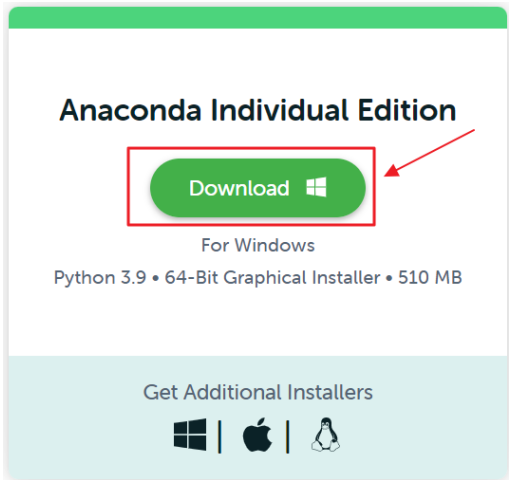
Anaconda 下載

- https://www.anaconda.com/



Anaconda 下載 (續)

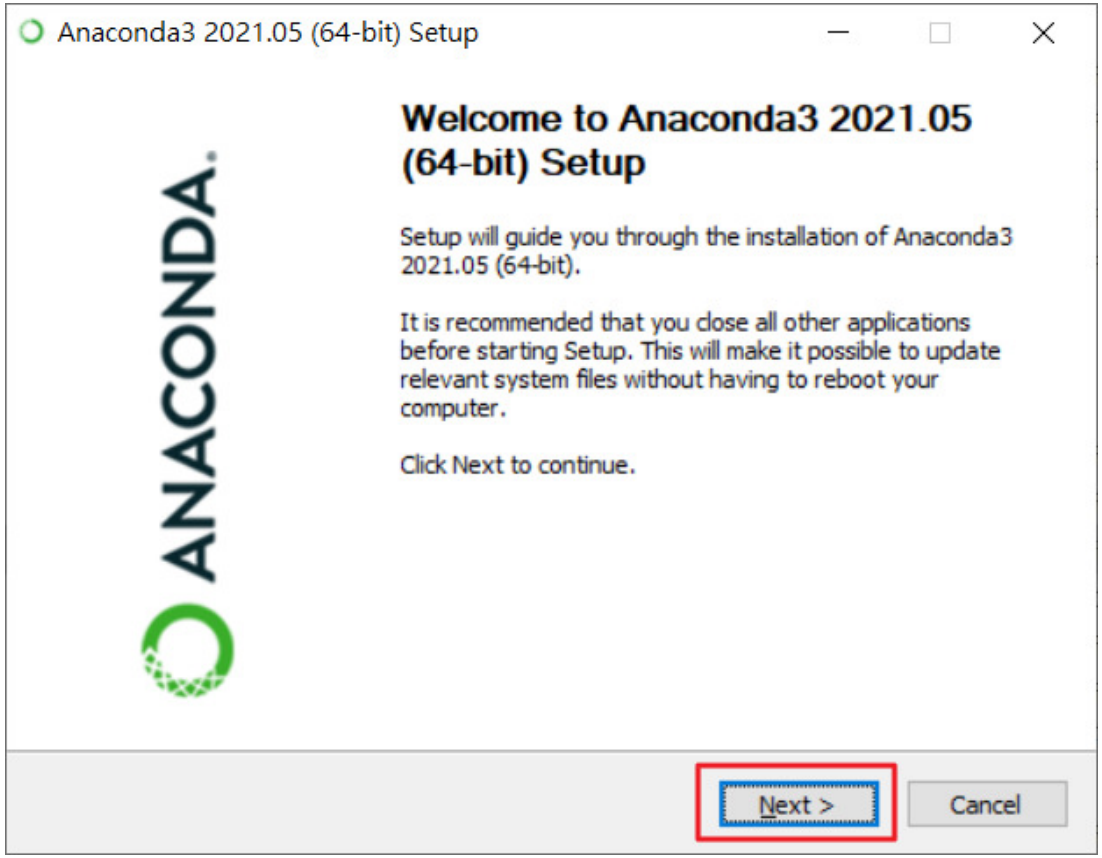




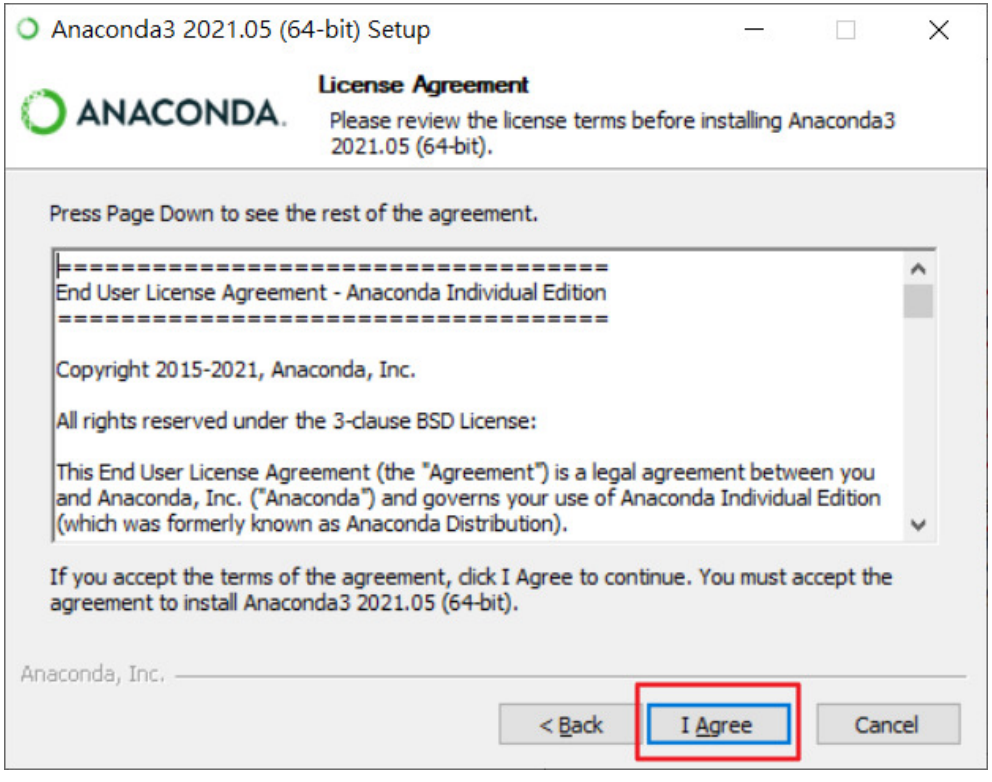
Anaconda3-2021.11-Windows-x86_64.exe

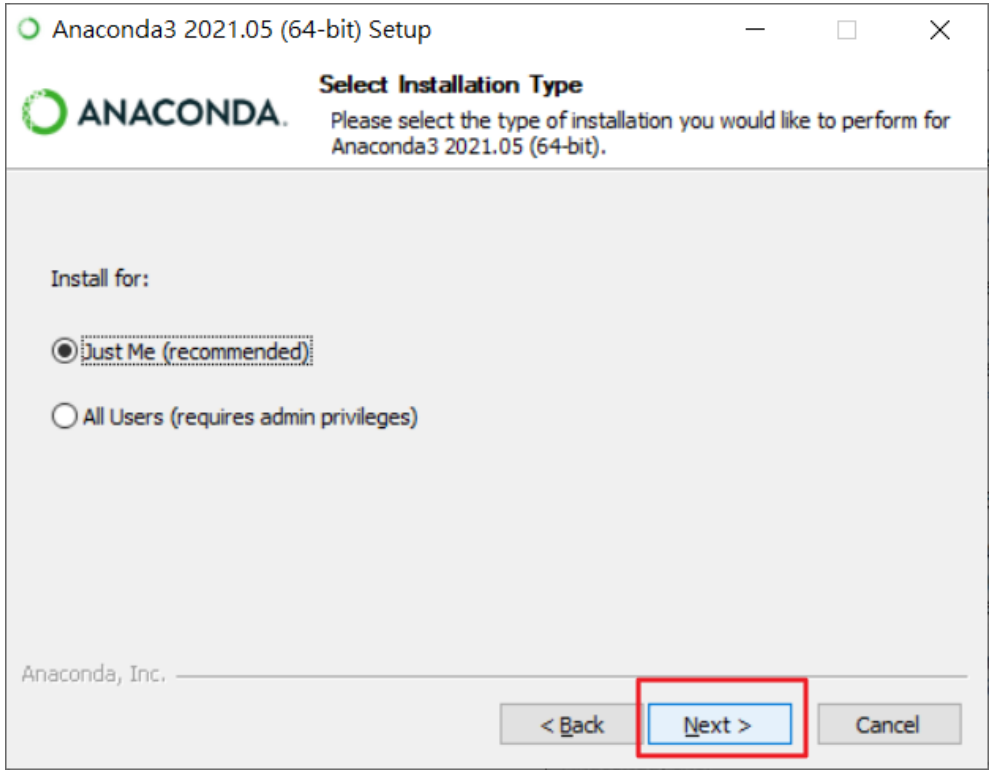
下載 64位元,
510 MB

安裝 Anaconda

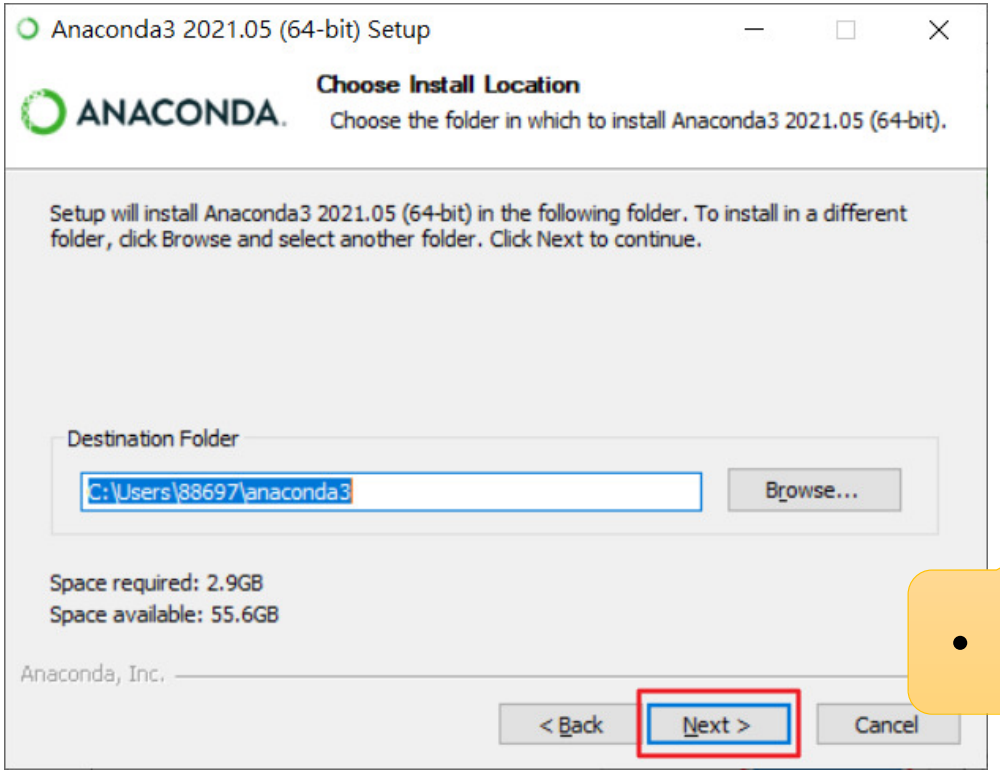

Anaconda3 2021.05 (64-bit) Setup
Welcome to Anaconda3 2021.05 (64-bit) Setup
Setup will guide you through the installation of Anaconda3 2021.05 (64-bit).
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
ANACONDA.
Next >
Cancel

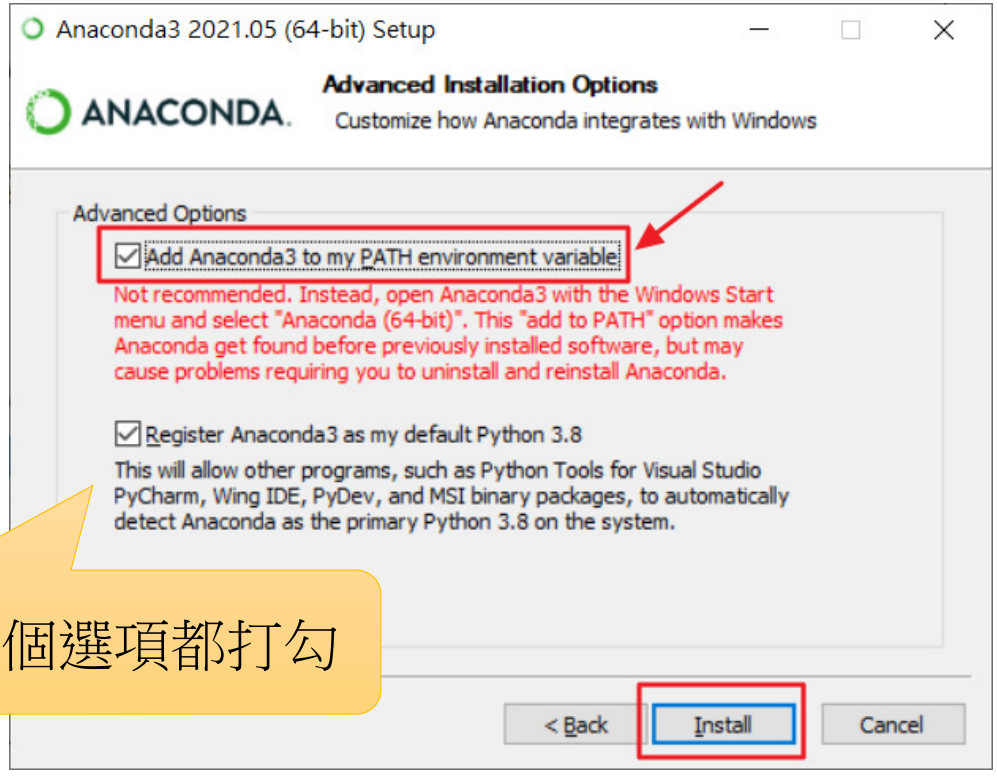
安裝畫面


Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
License Agreement
Please review the license terms before installing Anaconda3 2021.05 (64-bit).
Press Page Down to see the rest of the agreement.
End User License Agreement - Anaconda Individual Edition
Copyright 2015-2021, Anaconda, Inc.
All rights reserved under the 3-clause BSD License:
This End User License Agreement (the "Agreement") is a legal agreement between you and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Individual Edition (which was formerly known as Anaconda Distribution).
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2021.05 (64-bit).
Anaconda, Inc.
< Back
I Agree
Cancel


Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
Select Installation Type
Please select the type of installation you would like to perform for Anaconda3 2021.05 (64-bit).
Install for:
Just Me (recommended)
All Users (requires admin privileges)
Anaconda, Inc.
< Back
Next >
Cancel

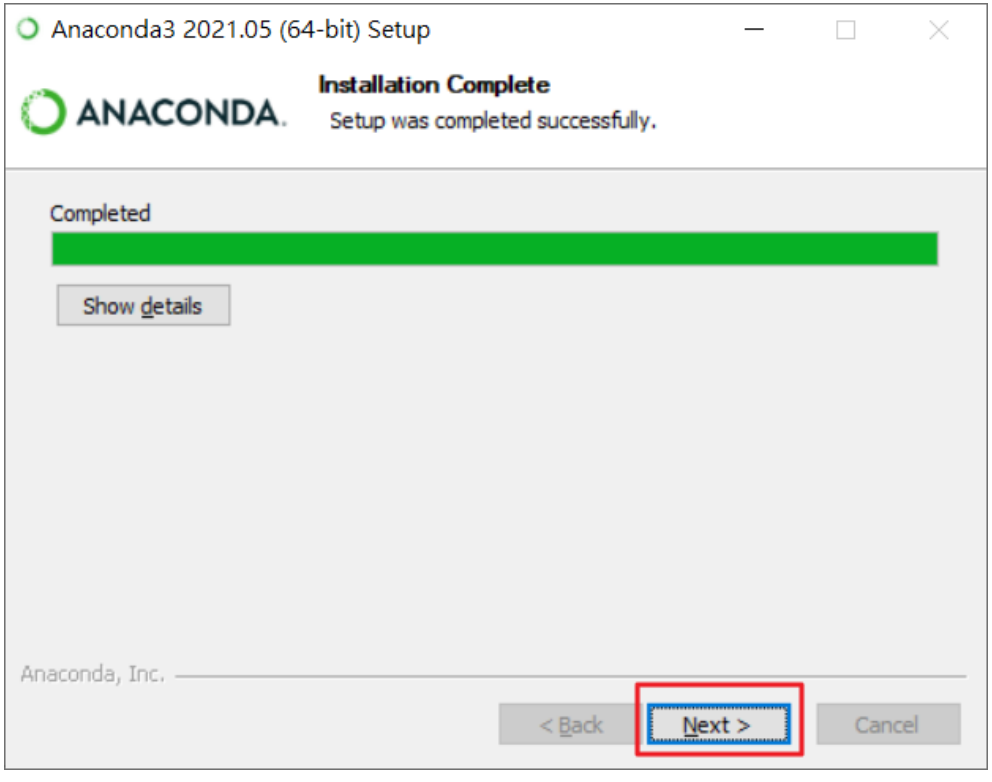
二個選項都打勾

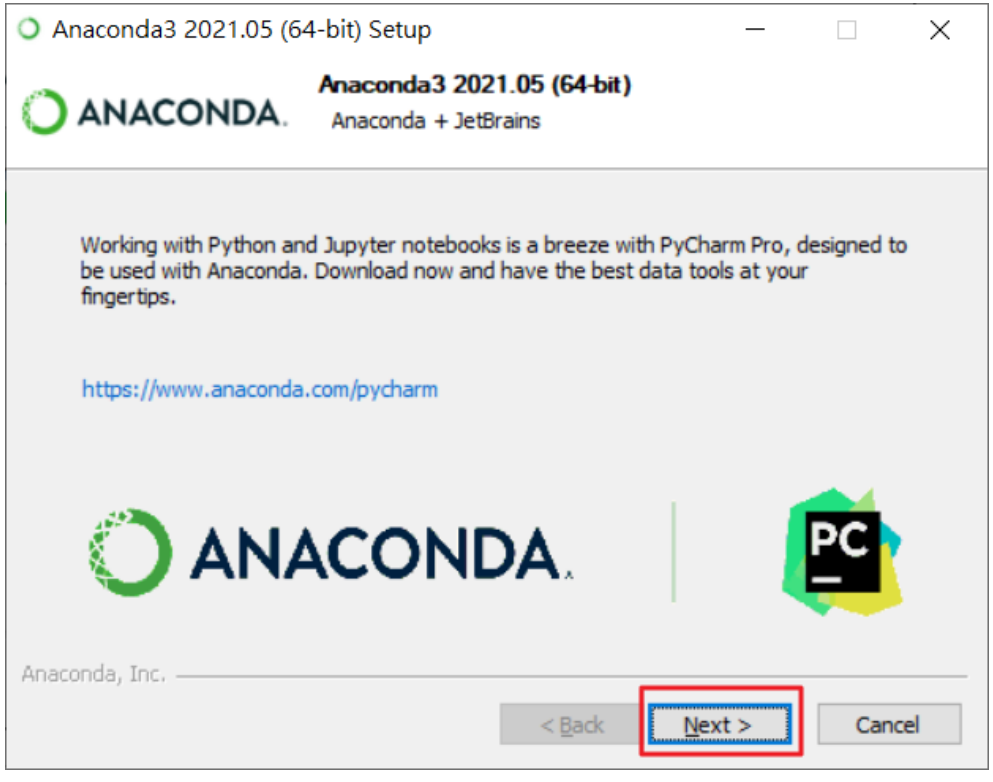




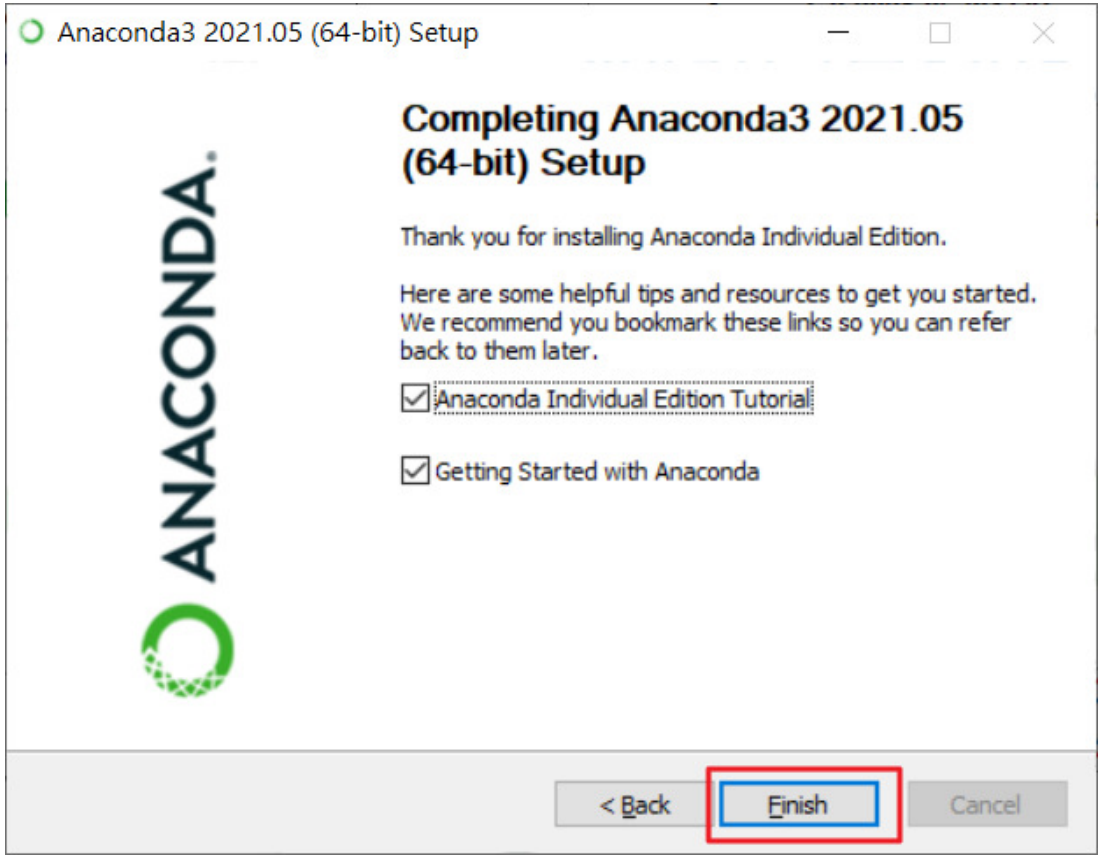
- 二個選項都打勾

安裝畫面

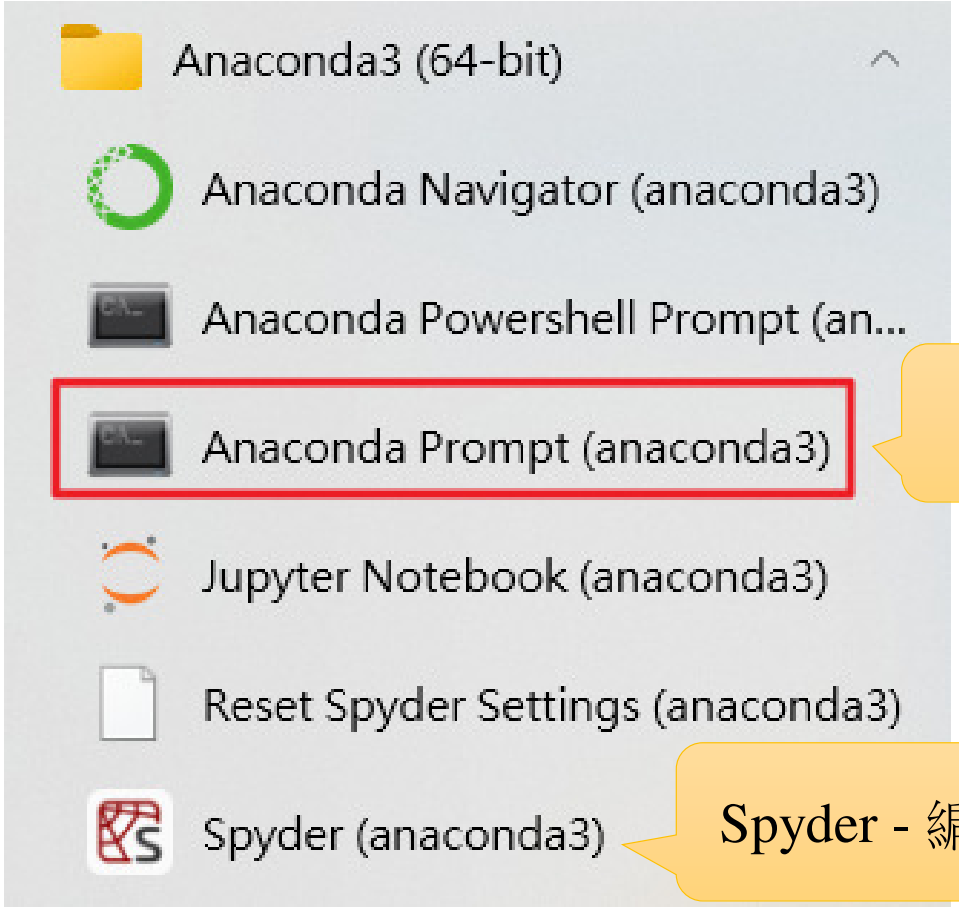

Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
Installation Complete
Setup was completed successfully.
Completed
Show details
Anaconda, Inc.
< Back
Next >
Cancel


Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
Anaconda3 2021.05 (64-bit)
Anaconda + JetBrains
Working with Python and Jupyter notebooks is a breeze with PyCharm Pro, designed to be used with Anaconda. Download now and have the best data tools at your fingertips.
https://www.anaconda.com/pycharm
ANACONDA.
PC
Anaconda, Inc.
< Back
Next >
Cancel

Anaconda 安裝完成

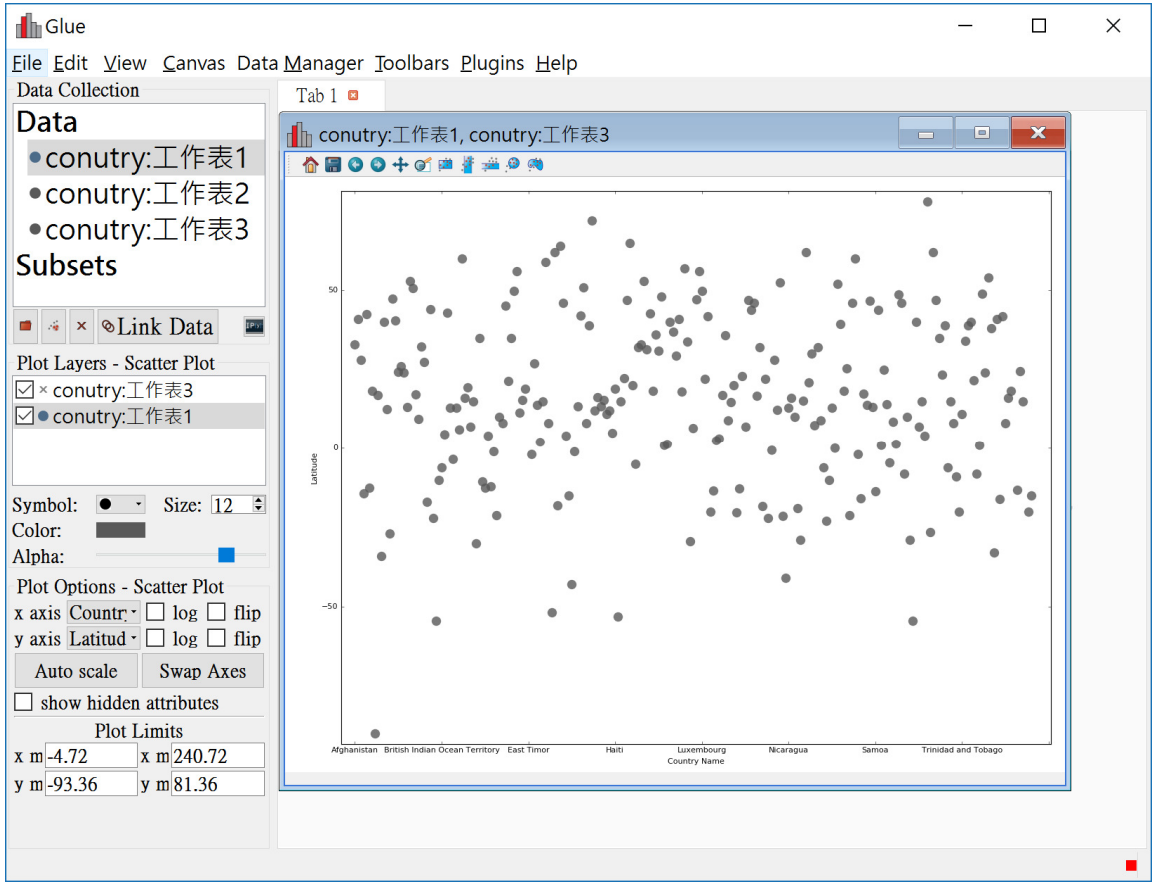


Anaconda3 (64-bit) -已安裝6個元件

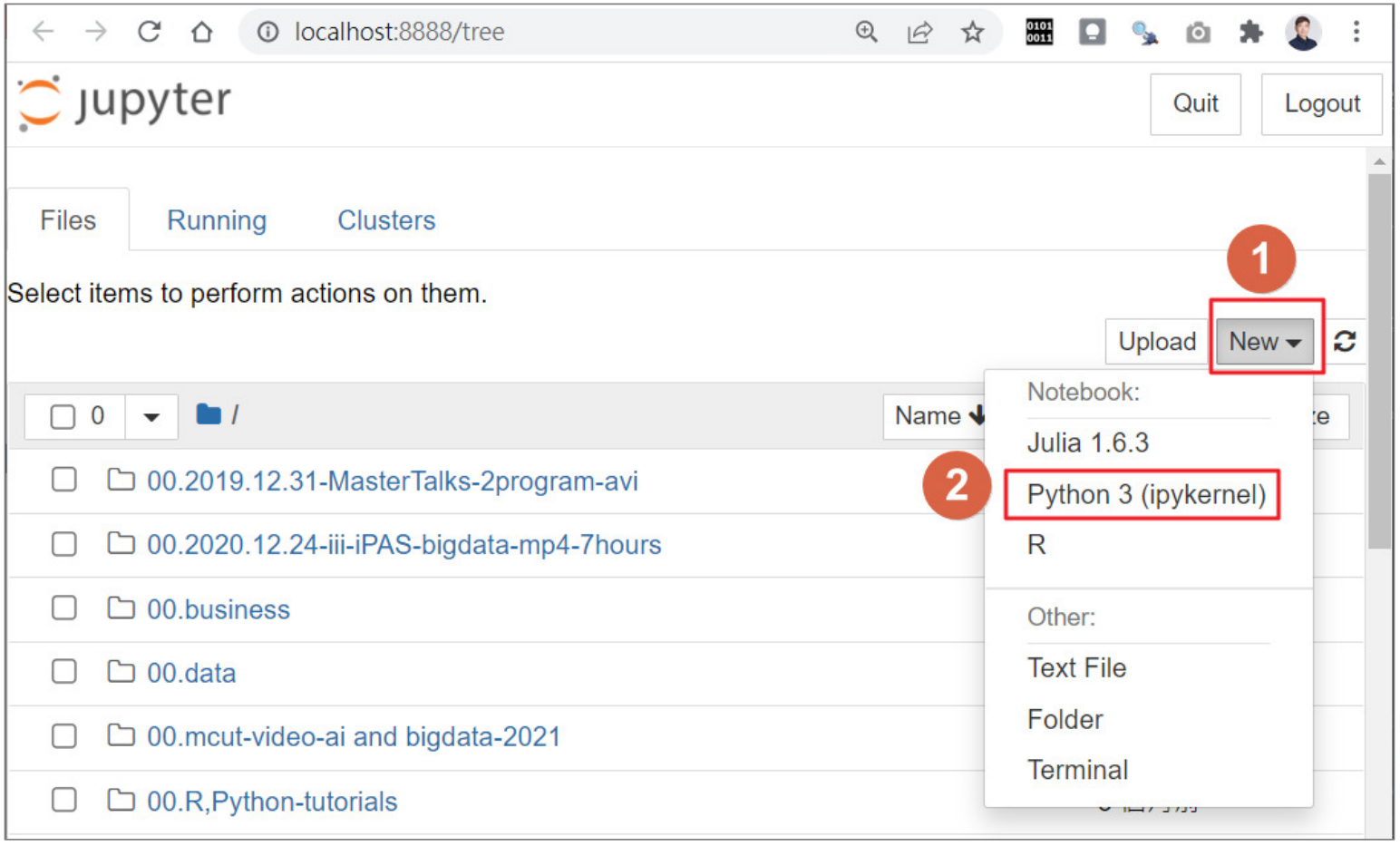


Anaconda Navigator

glueviz 視覺化


Glue
File Edit View Canvas Data Manager Toolbars Plugins Help
Data Collection
Data
conutry:工作表1
conutry:工作表2
conutry:工作表3
Subsets
Link Data
Plot Layers - Scatter Plot
conutry:工作表3
conutry:工作表1
Symbol:
Size: 12
Color:
Alpha:
Plot Options - Scatter Plot
x axis Countr log flip
y axis Latitud log flip
Auto scale
Swap Axes
show hidden attributes
Plot Limits
x m -4.72
x m 240.72
y m -93.36
y m 81.36
Tab 1
conutry:工作表1, conutry:工作表3
Latitude
Afghanistan
British Indian Ocean Territory
East Timor
Haiti
Luxembourg
Nicaragua
Samoa
Trinidad and Tobago
Country Name

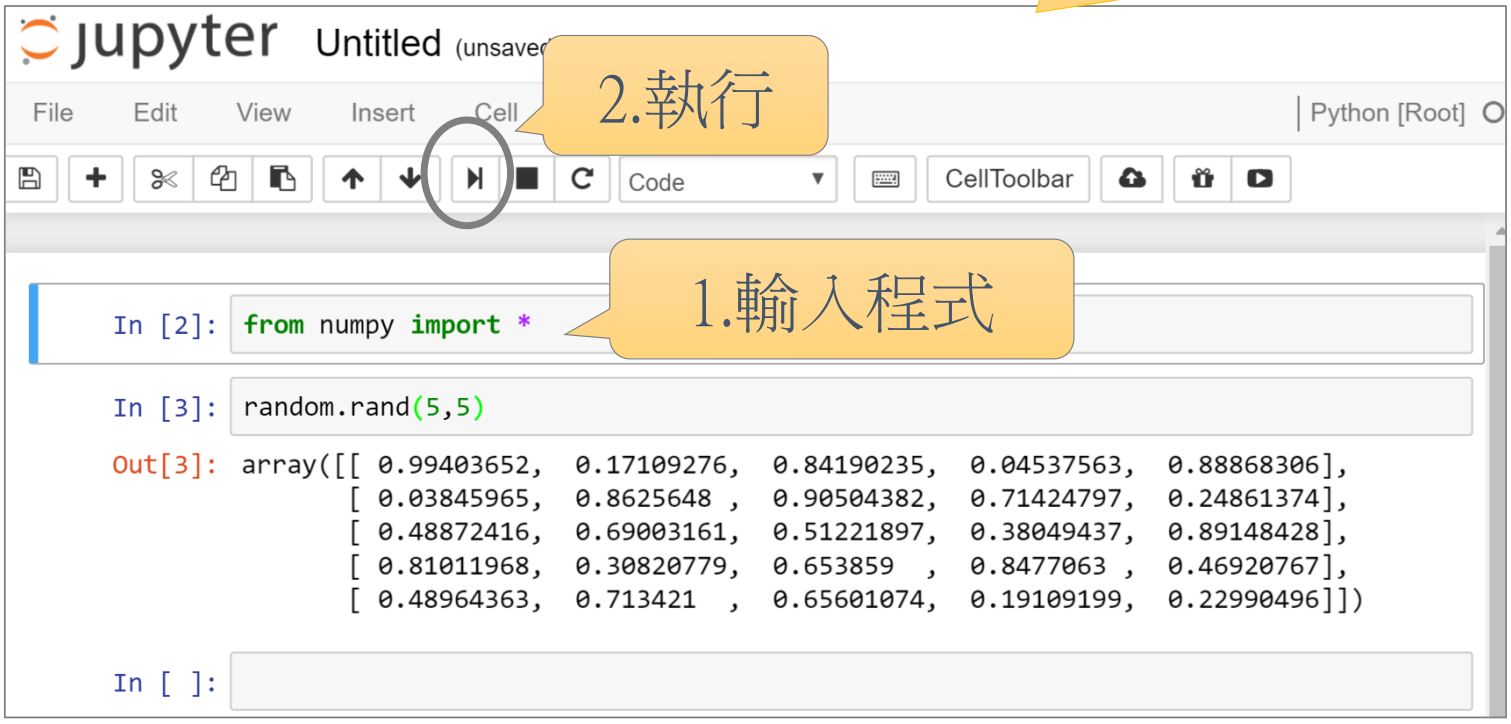
jupyter notebook



jupyter notebook (續)

實作練習

理解重新命名方式



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**

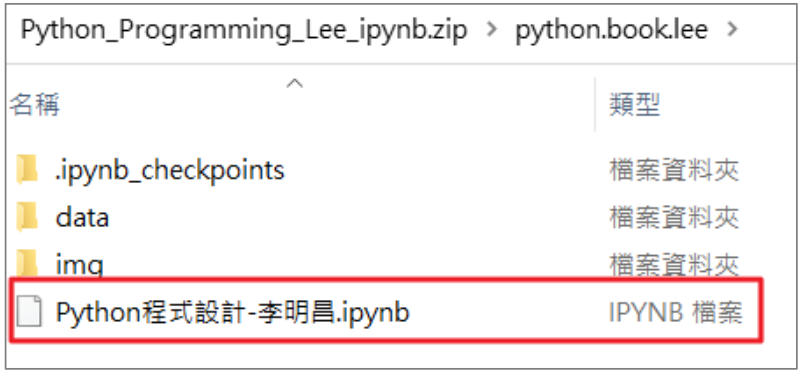
實作練習

主題: **Python 程式設計-李明昌 - ipynb**

檔名: Python_Programming_Lee_ipynb.zip

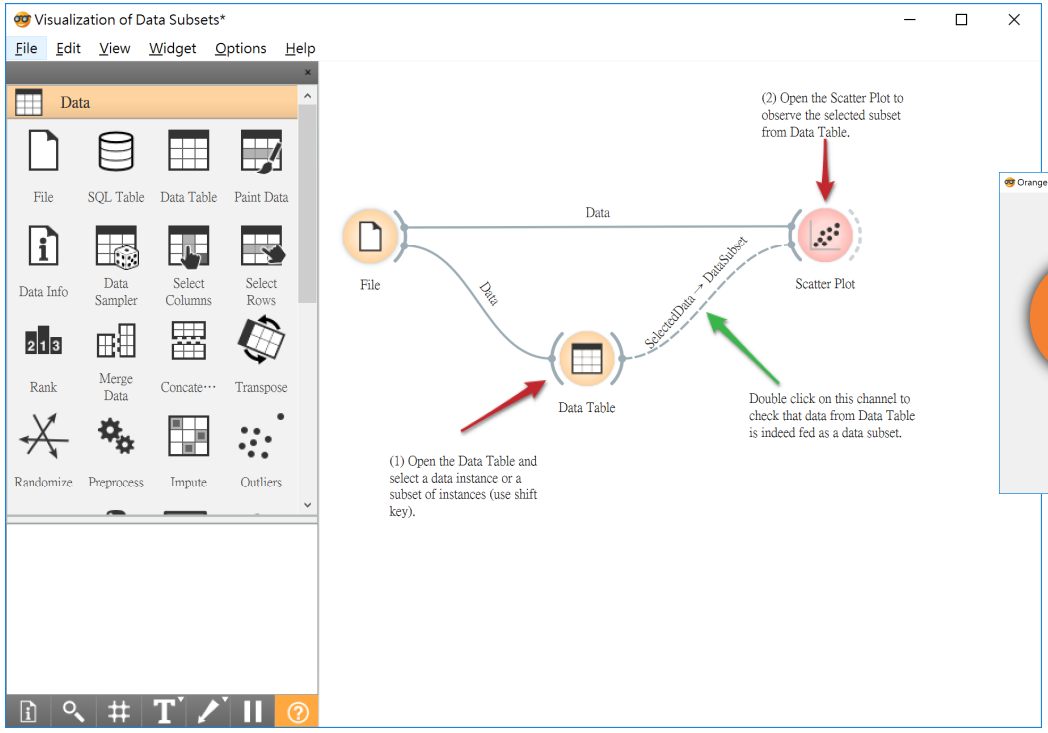
包括 Python 程式設計-李明昌電子書的原始 ipynb 檔案, 圖檔, 部分資料集

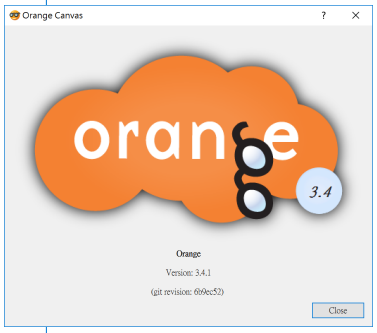
下載: https://github.com/rwepa/DataDemo/blob/master/Python_Programming_Lee_ipynb.zip



Orange

```
# 安裝 Orange
# conda install -c conda-forge orange3

# 使用命令提示列 開啟 Orange
# python -m Orange.canvas
```





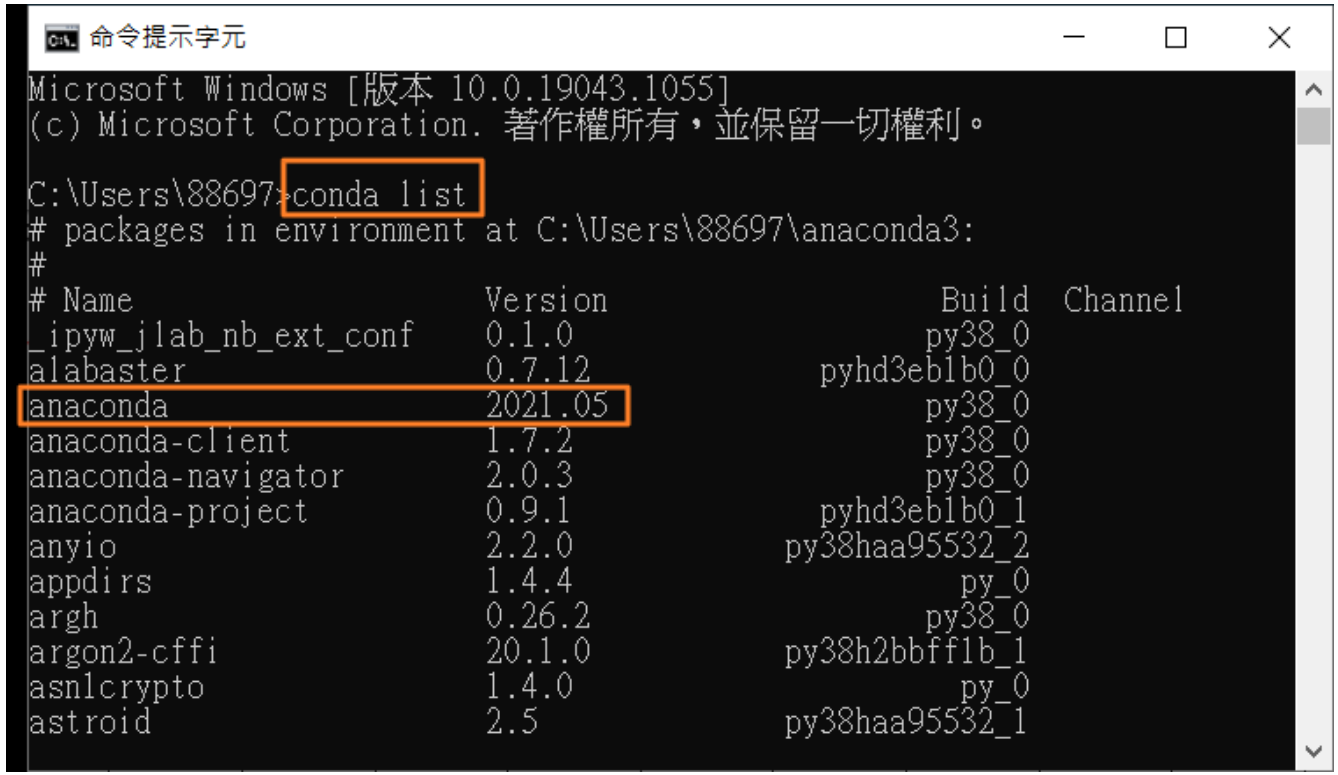
Python Orange - 關聯規則教學

- http://rwepa.blogspot.com/2022/07/python-orange-associate-tutorial.html
- https://youtu.be/rh5GxJamtNg

Orange - Scatter Plot

Anaconda 模組管理

- 顯示已安裝模組
  conda list



Anaconda 模組管理(續)

- 搜尋官網可以下載模組版本
  conda search matplotlib
- 安裝模組
  conda install 模組名稱
- 更新模組
  conda update 模組名稱

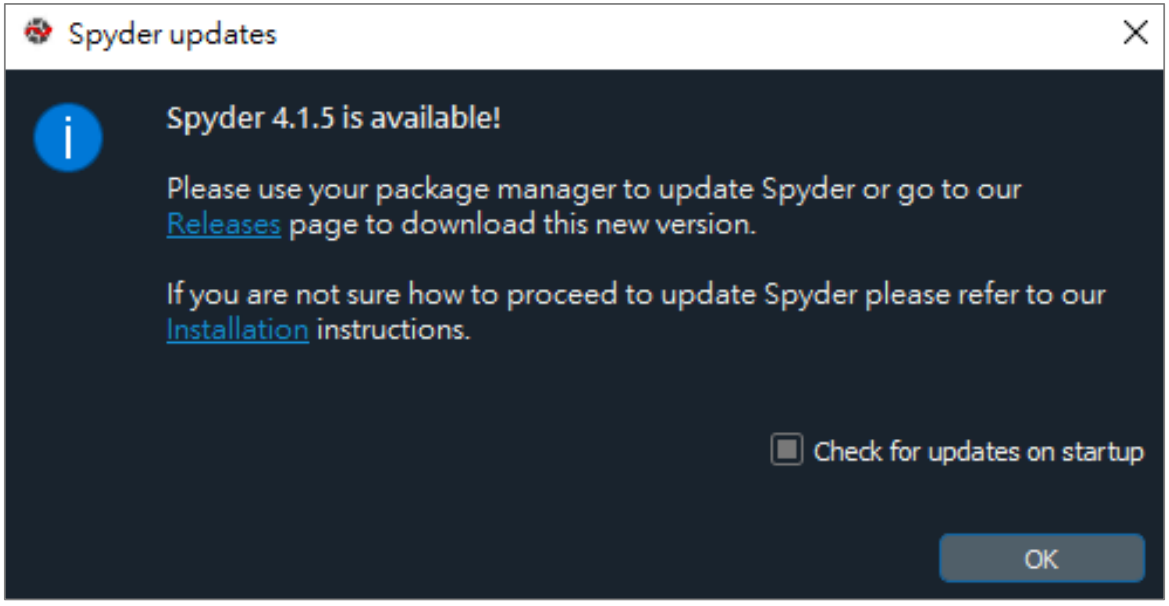
conda 虛擬環境

- 檢視所有虛擬環境清單 **conda env list**
- 啟用 myenv 虛擬環境 **conda activate myenv**
- 關閉虛擬環境 **conda deactivate**
- 建立 myenv 虛擬環境 **conda create --name myenv**
- 建立特定 python 版本的虛擬環境
  - **conda create -n myenv python=3.9**
- 建立特定 scipy 模組版本的虛擬環境
  - **conda create -n myenv scipy=1.9.0**

參考 https://github.com/rwepa/DataDemo/blob/10c8ab50a0871a6d30b212cef3fc865248532a39/iPAS-python-program.py#L2658

3.3 Spyder 軟體簡介

Spyder 更新

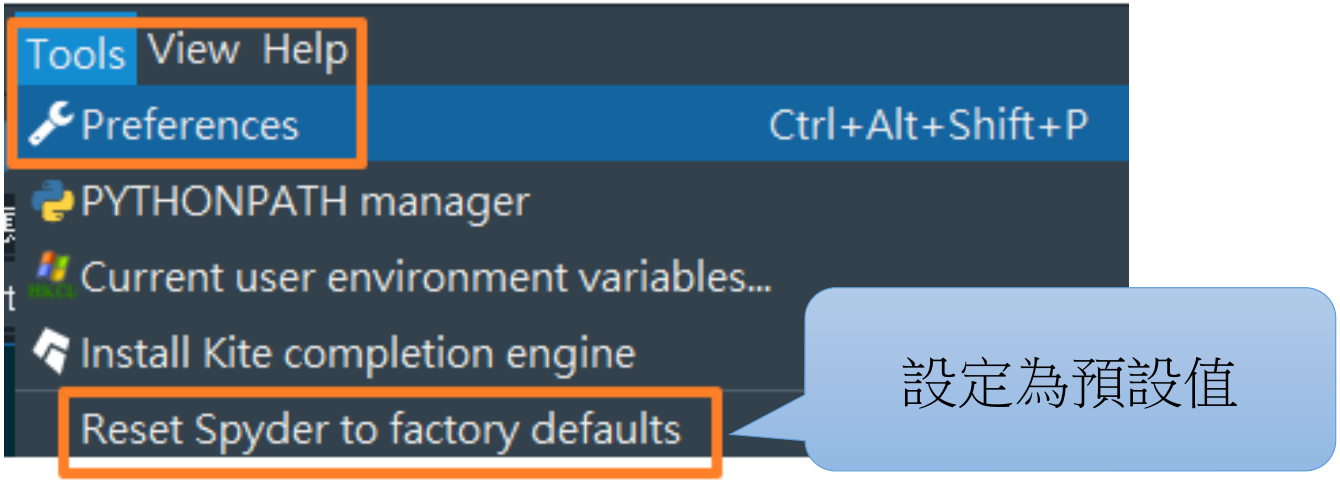


- Anaconda 整體更新　　conda update anaconda
- Spyder 更新　　conda update spyder

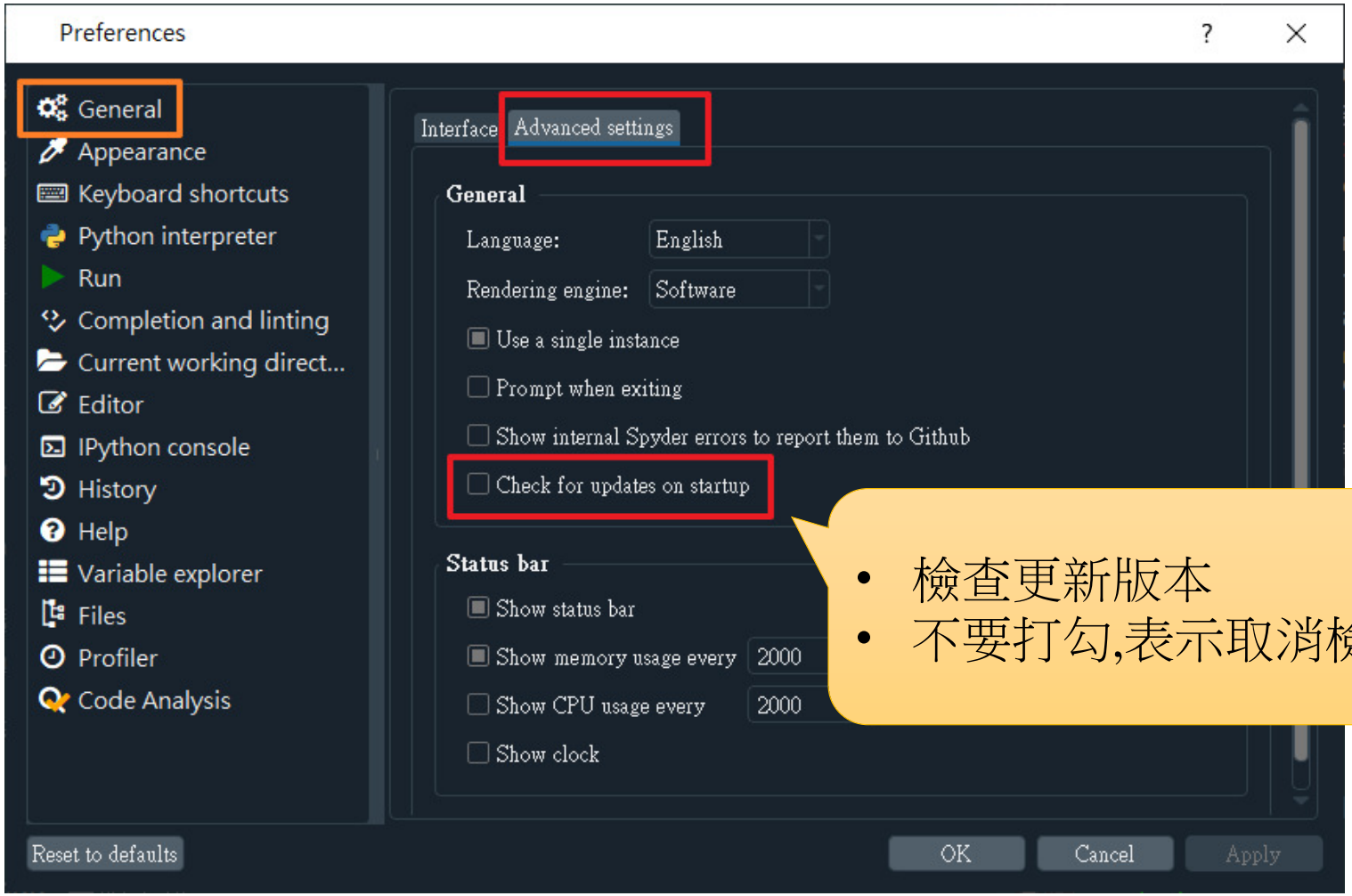
Spyder 畫面


功能表列
工具列
切換窗格
Spyder (Python 3.5)
File Edit Search Source Run Debug Consoles Tools View Help
C:\pythondata
Editor - C:\pythondata\python.sklearn.讀我.py
python.sklearn.讀我.py
type.operations.py
1
程式碼
物件瀏覽
2
Object inspector
Source Console Object
Object inspector Variable explorer File explorer
IPython console
Console 1/A
Python 3.5.2 |Anaconda 4.1.1 (64-bit)| (default, Jul 5 2016, 11:41:13) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 4.2.0 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
IPython
3
Console History log IPython console
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 8 Column: 4 Memory: 40 %

喜好設定 Tools\Preferences



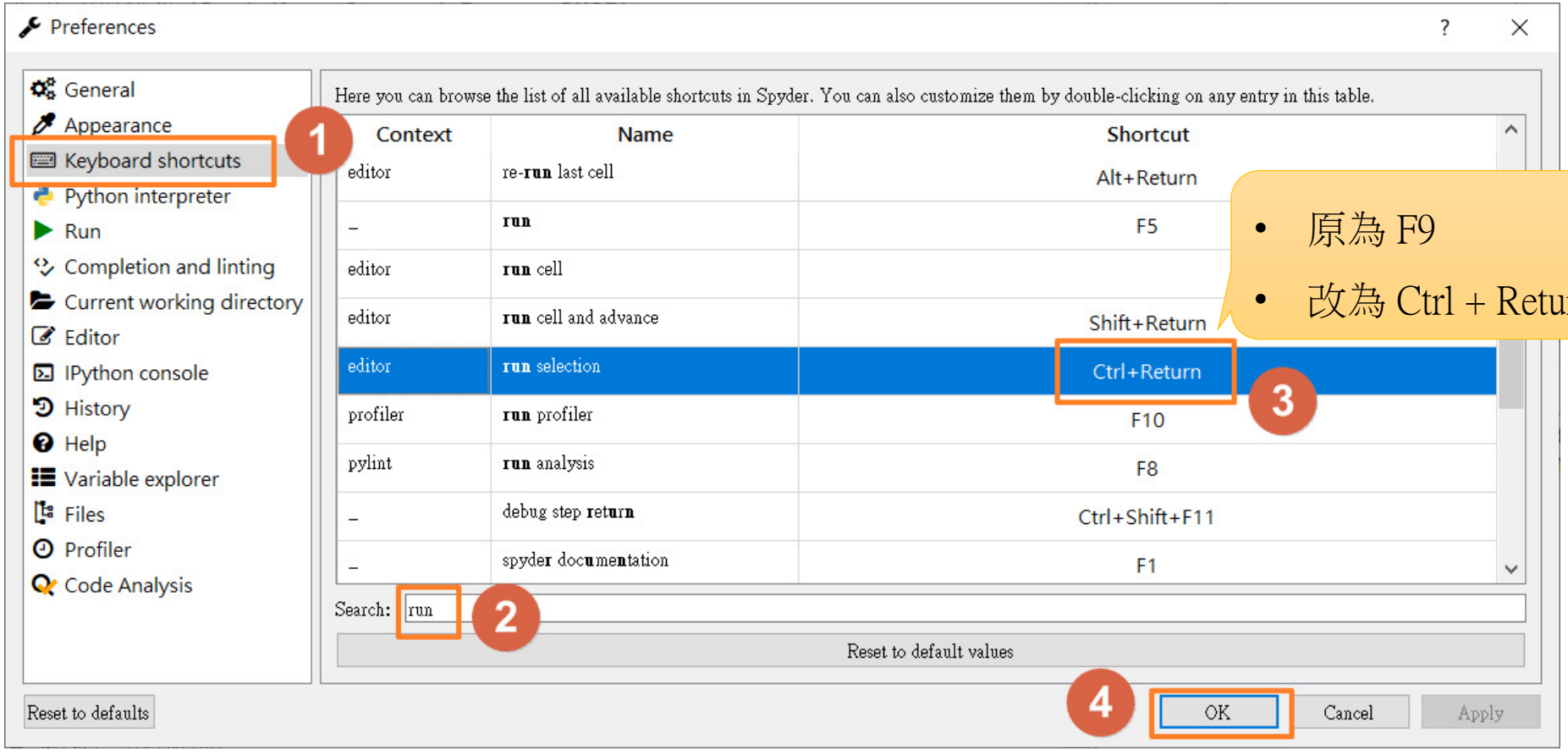
Preferences \ General \ Advanced setting



- 檢查更新版本
- 不要打勾,表示取消檢查

Preferences \ Appearance

Preferences \ Keyboard shortcuts

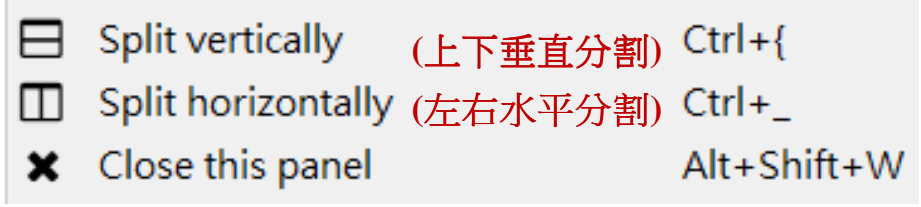


Spyder 好用的快速鍵

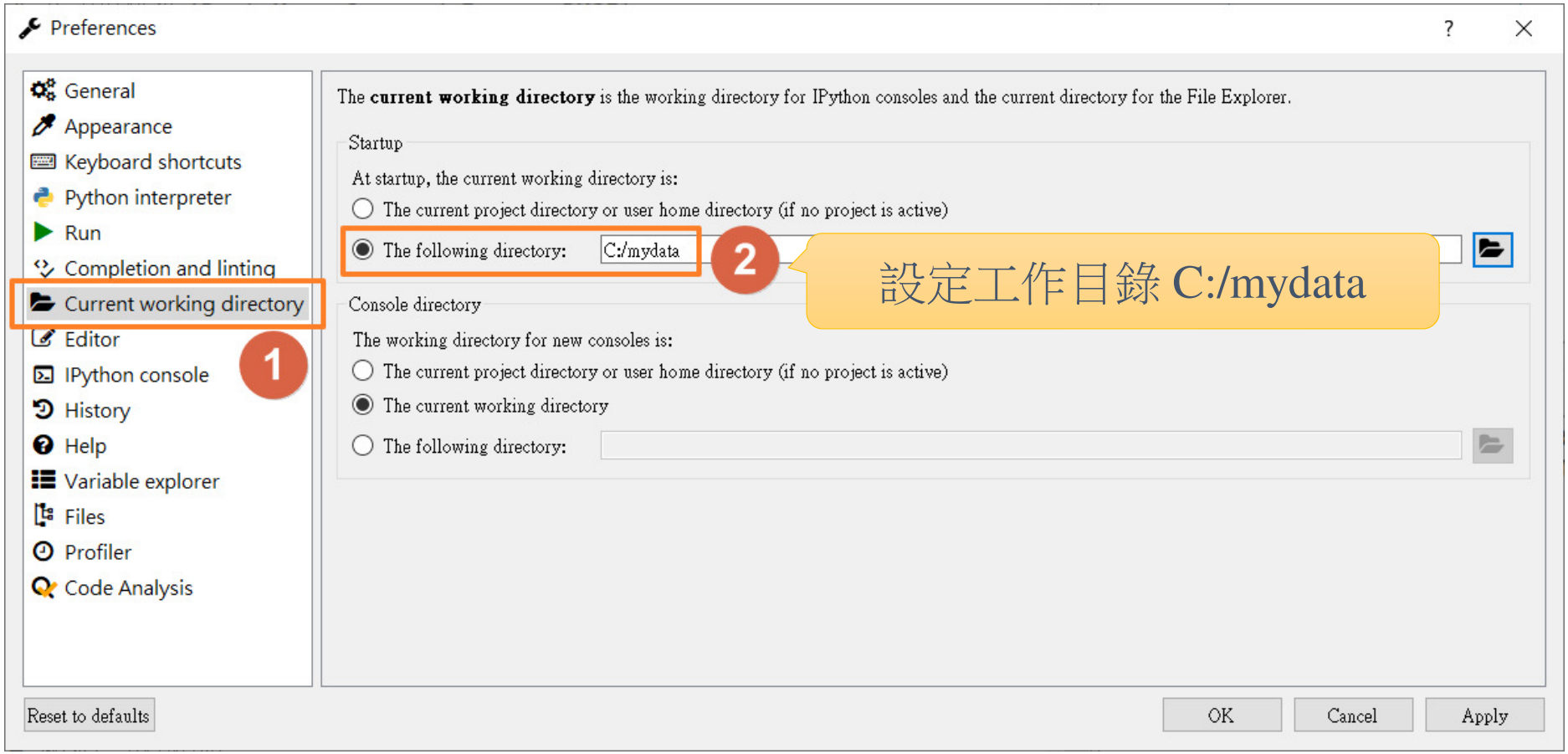
• **執行選取程式碼**	**Ctrl + Enter (原為 F9)**
• **註解切換**	**Ctrl + 1**
• 尋找	Ctrl + F
• 取代	Ctrl + R
• 切換至編輯視窗	Ctrl + Shift + E
• 切換至 Ipython Console	Ctrl + Shift + I
• 檔案切換	Ctrl + P
• **重新啟動 Ipython Console**	**Ctrl + .**
• 檢視放大	Ctrl + “+”
• 檢視縮小	Ctrl + “-”
• 重新啟動 Spyder	Alt + Shift + R

Ipython Console 視窗

- 清空: Ctrl + L
- 清空: %clear

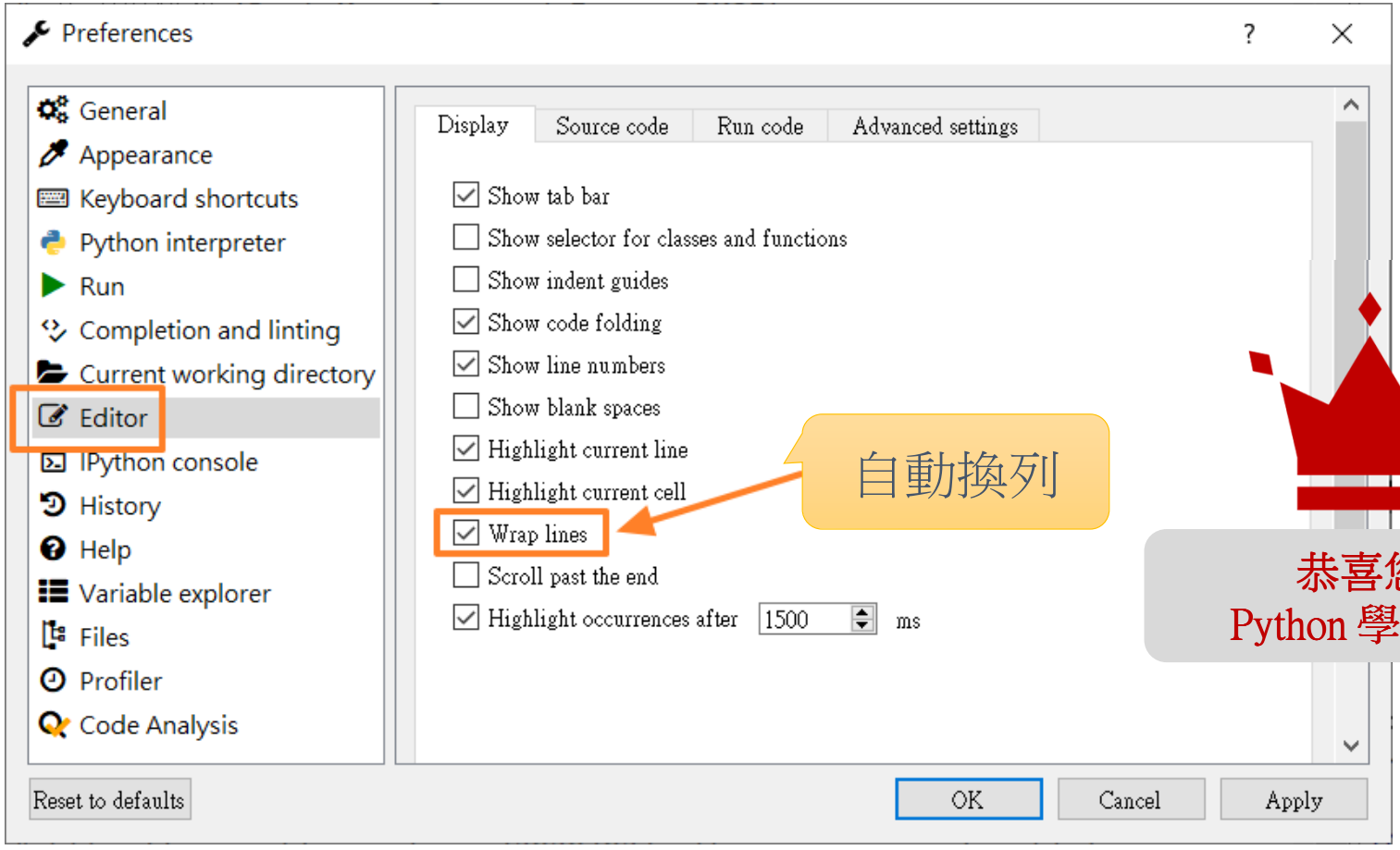


Preferences \ Current working directory



Preferences \ Editor \ Wrap lines

實作練習



恭喜您, 開啟
Python 學習之旅 ^_^

Python 執行 方法1.命令提示列

- 建立 C:\mydata\helloworld.py

C:\mydata\helloworld.py

helloworld.py

```
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")
```

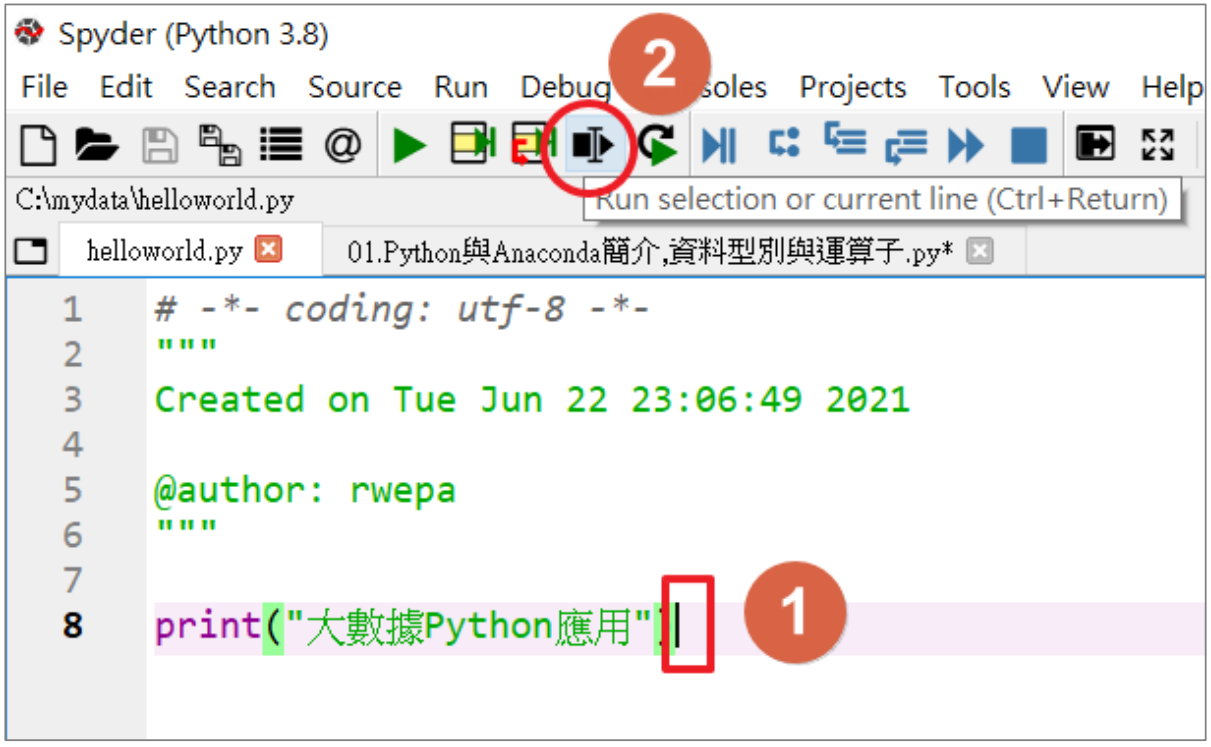
python helloworld.py

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 方法2.使用 Spyder

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



3.4 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
- 雙下底線開頭與結尾的名稱,部分已經由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

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

- 命名為中文, OK?
- 命名為Python保留字, OK?

內建保留字

- dir(__builtins__)
- len(dir(__builtins__)) # 160

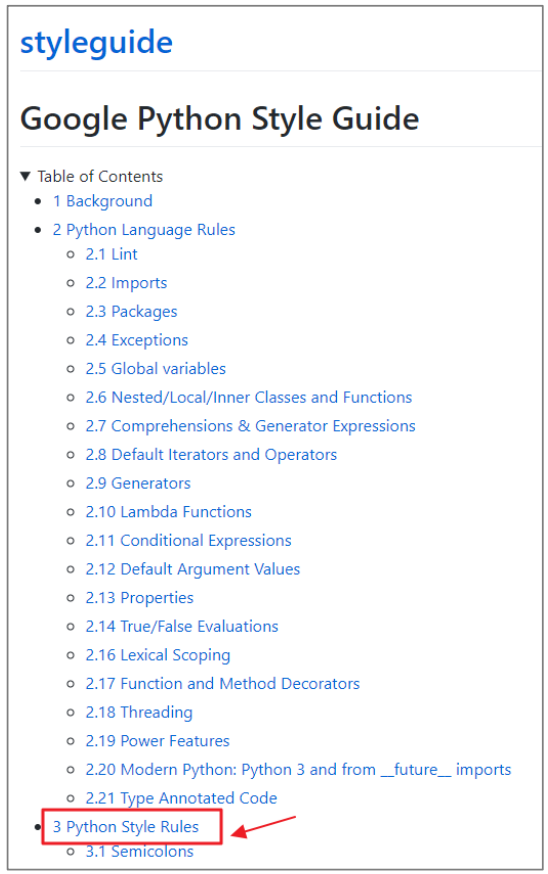
```
[in] len(dir(__builtins__)) # 160
[out] 160
```

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

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

3.5 資料型別與運算子

資料型別 (Data Types)

- 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: https://docs.python.org/3/library/stdtypes.html

廣義資料型別

- 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

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

亂數種子 random.seed()

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)

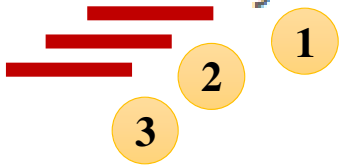
運算子 – 範例

```
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
```

3.6 資料物件

資料物件

★ ★ ★

- 四大容器型別 (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])
```

長度 len

- 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
```

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
   ...: print(tuple3)
('台北', '台中', '高雄', '男', '女', '女')

In [3]: # tuple - 重複

In [4]: tuple1*3
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')
```

使用 + 號

使用 * 號

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]
         0  1      2          3          4        5
In [3]: t[2]
Out[3]: (3, 'Hi')

In [4]: t[:3]
Out[4]: [1, 2, (3, 'Hi')]

In [5]: t[3:]
Out[5]: [[4, 'RWEPA'], (2+3j), 60000000.0]

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
```

[:3] 開始~指標3-1=2

[3:] 指標3 ~ 最後

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

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

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-結合: +

串列-重複: *

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']]

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']
```

附加為1個值

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

如何顯示list不以 __ 開始串列方法的總個數 11?

Out[1]: 11

```
In [1]: # 顯示方法

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("台南市")

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) # 較簡潔
   ...: print(d)
168
```

1 2

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

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

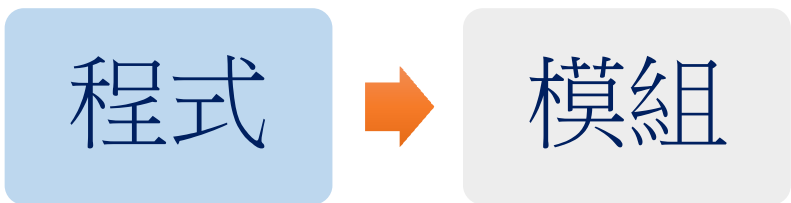
字典 – 方法

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

3.7 認識模組

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

- 匯入模組的所有函數
  - import 模組名稱
  - import 模組名稱 as 別名
- 匯入特定函數
  - 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']
```

實作練習

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
    r = a - q*b          # r 餘
    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( )

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

EMAIL: *alan9956@gmail.com*

WEB: http://rwepa.blogspot.com/