目次

- GitHub 專案連結
  https://github.com/telunyang/python_data_analysis
- YouTube 頻道
  https://www.youtube.com/@darreninfo-boatman

Module 1. Python 與資料分析應用

為什麼要使用 Python 進行資料分析

Python 是一種廣泛使用的程式語言，因為它具有易於學習、易於閱讀、易於編寫和開發高效應用程序等優點。另外，Python 也擁有豐富的資料分析函數庫和工具，這使得它成為一個流行的資料科學和機器學習工具。

在 Python 中進行資料分析的目的是從大量的資料中提取有價值的資訊。資料分析可以幫助我們發現趨勢、模式、異常情況和關聯性，這些資訊對企業、學術機構、政府等各種組織來說都具有重要的意義。

使用 Python 進行資料分析的好處之一是可以利用 Python 的數學和統計函數庫進行數據處理和分析。例如，NumPy、Pandas 等函數庫提供了豐富的數據處理和統計分析工具，可以進行數據操作、統計推斷、機器學習等任務。

另一個好處是 Python 的視覺化函式庫，例如 Matplotlib，可以幫助我們視覺化資料，這對於發現趨勢、研究關聯性和提供洞察非常有用。

總之，資料分析是 Python 的重要應用之一，它可以幫助人們從大量的資料中提取有價值的資訊，並且可以幫助我們更好地理解我們周圍的世界。

需要有微積分、線性代數、統計學等先備知識來作為基礎

定義問題	→	取得資料	→	資料前處理	→	模型訓練	→	模型評估	→	決策應用
想解決什麼問題		定義資料格式		資料清理與調整		模型選擇		確定評估指標		領域/產業知識
問題的背景		資料蒐集與儲存		資料探索與視覺化		參數調整		評估結果視覺化		加值應用
你的動機和目的				特徵工程						

表: 資料科學的基礎流程

資料分析步驟說明

資料分析通常可以分成以下幾個步驟：

- 資料蒐集：收集要分析的資料，可以從網路上下載或者從資料庫中讀取。
- 資料清理：將資料中的錯誤或者缺失值進行處理，包括刪除或者補值等。
- 資料分析：使用 numpy 和 pandas 對資料進行操作和統計分析。

- 資料視覺化：使用 matplotlib 將資料以圖表的形式呈現出來，以便更好地理解和分析。

以下是每個步驟的說明：

資料蒐集

在資料蒐集階段，你需要確定你的資料來源和收集方式，並且將資料儲存到一個可讀取的格式中，例如 csv、Excel 或者 JSON 等。資料蒐集的過程需要將資料收集到本地端，進行一些基本的資料檢查，並且進行下一步的資料清理。

資料清理

在資料清理階段，你需要檢查資料是否有缺失值、重複值、錯誤值等問題。若有這些問題需要進行處理，例如刪除或者補值等。在這個步驟中，可以使用 pandas 的函數來處理這些問題，例如 dropna()、fillna()、drop_duplicates()、replace() 等等。

資料分析

在資料分析階段，你需要使用 numpy 和 pandas 提供的函數對資料進行操作和統計分析，例如計算平均值、標準差、變異數等等。這個階段也可以對資料進行進一步的整理，例如資料篩選、資料排序、資料分組等等。

資料視覺化

在資料視覺化階段，你需要使用 matplotlib 將資料以圖表的形式呈現出來，以便更好地理解和分析。可以使用各種不同的圖表類型，例如散點圖、柱狀圖、線圖等等。在這個步驟中，需要選擇最適合的圖表類型，以便更好地表達資料的含意和關係。

總結起來，資料分析是一個循序漸進的過程，每個步驟都相互依賴，資料的品質和結果的準確性也取決於前一個步驟的執行。在實際的應用中，還需要不斷地調整和優化每個步驟的操作和結果，以達到最佳的分析效果。

Jupyter Notebook 環境設定

使用 Jupyter Kernel

Jupyter kernel 的主要功能是在 Jupyter 環境中執行並解釋不同的程式語言。Kernel 作為一個後端引擎 (它會自行開啟一個 http://localhost:8888/... 的

網頁伺服器，提供一個網頁版本的線上編輯器)，接收使用者在前端的輸入並進行運算或計算，然後將結果傳回給前端顯示。因此，Jupyter kernel 的主要功能是讓使用者能夠在 Jupyter 環境中使用不同的程式語言進行互動式的編程和數據分析。

Jupyter kernel 支持的程式語言包括但不限於 Python、R、Julia、MATLAB、C++ 等。使用者可以通過在 Jupyter Notebook 或 JupyterLab 中選擇不同的 kernel 來使用不同的程式語言。

安裝 conda 環境
conda create --name da python=3.10 notebook ipykernel

進入 conda 環境
conda activate da

新增 Kernel
python -m ipykernel install --user --name da --display-name "Python3@da"

檢視 Jupyter Notebook kernel
jupyter kernelspec list

註:

1. python -m 指的是直接使用模組 (module) 的預設功能。
2. --user 代表將 kernel 安裝在個人使用者目錄 (或是個人的家目錄) 當中，而非預設的系統環境。
3. --name da 與 conda env 的 da 沒有直接關係，僅是 kernel 在系統當中的別名。
4. --display-name "Python3@da" 是新增 Jupyter Notebook 時的選項。

刪除 Kernel
jupyter kernelspec uninstall da

```
C:\Users\Owner>conda activate da

(da) C:\Users\Owner>jupyter kernelspec list
Available kernels:
  python3    C:\Users\Owner\anaconda3\envs\da\share\jupyter\kernels\python3

(da) C:\Users\Owner>python -m ipykernel install --user --name da --display-name "Python3@da"
Installed kernelspec da in C:\Users\Owner\AppData\Roaming\jupyter\kernels\da

(da) C:\Users\Owner>jupyter kernelspec list
Available kernels:
  python3    C:\Users\Owner\anaconda3\envs\da\share\jupyter\kernels\python3
  da         C:\Users\Owner\AppData\Roaming\jupyter\kernels\da

(da) C:\Users\Owner>
```

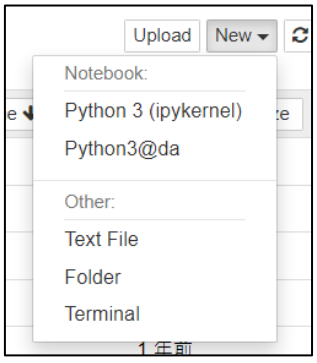
圖: 安裝完 Jupyter Notebook Kernel，再次檢視，會發現新增的 Kernel

注意：

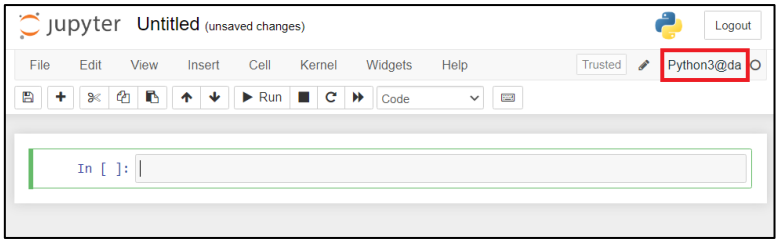
- Jupyter Notebook 建立的 kernel，跟 conda 的 env 概念是不一樣的。
  - Kernel 會啟動網頁版本的線上編輯器，並且使用目前 conda env 的 Python 版本；而 conda 的 env 單純是環境和套件的管理器。
  - 例如在 python=3.10 的 conda env 裡面安裝 kernel，此時該 kernel 用的 python 版本就會是 3.10。
  - 無論在任何 conda env 下，只要啟動 Jupyter Notebook，就會以當下目錄作為主目錄/工作路徑。

執行 Jupyter Notebook (在虛擬環境 da 下)
conda activate da jupyte notebook

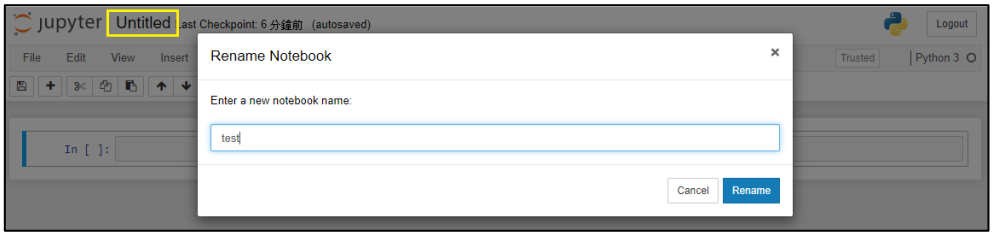
新增 Notebook



圖：按下 New，新增 Notebook（選擇 Python3@da）



圖：此時進入新建的 Jupyter Notebook 中



圖：按下「Untitled」，命名「test」，按下「Rename」

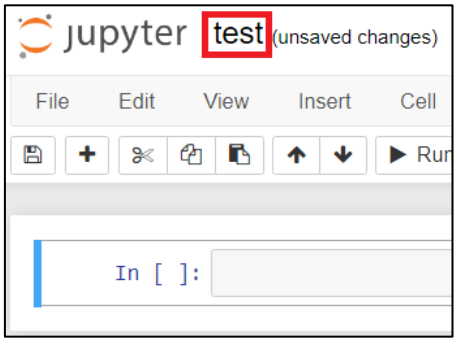
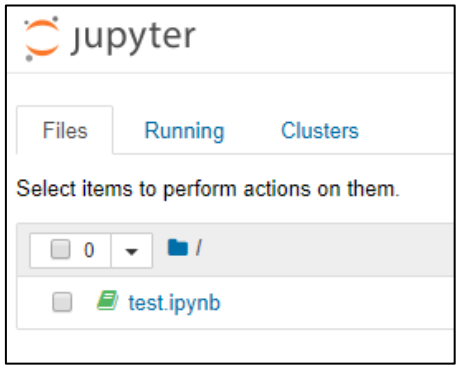


圖: Notebook 的名稱被改成 test



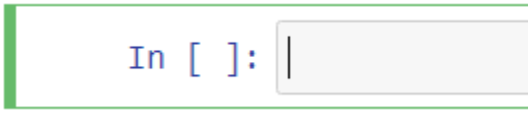
圖：工作目錄中，有對應名稱的「.ipynb」檔（Jupyter Notebook 檔）

Cell 模式

在 Jupyter Notebook 中，有兩種模式可以切換，分別是 command mode 和 edit mode。command mode 和 edit mode 是 Jupyter Notebook 中非常重要的兩個模式。command mode 可以讓使用者快速執行各種操作，而 edit mode

則可以讓使用者方便地編輯程式碼或文字內容。：

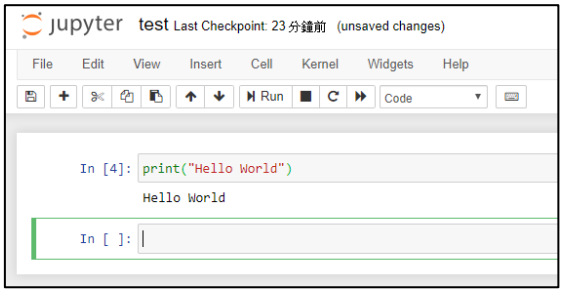
- Command Mode (指令模式)
  - 當進入 Jupyter Notebook 時，預設就是 command mode。在這個模式下，可以使用鍵盤快捷鍵來執行各種操作，例如新增、刪除或複製儲存格、改變儲存格類型、執行儲存格內的程式碼、儲存 Notebook 等等。
  - 以下是一些常見的鍵盤快捷鍵：
    - Enter：進入 edit mode。
    - Esc：回到 command mode。
    - A：在當前儲存格的上方新增一個儲存格。
    - B：在當前儲存格的下方新增一個儲存格。
    - D + D：刪除當前選取的儲存格。
    - M：將當前儲存格的類型改為 Markdown (可參考這裡)。
    - Y：將當前儲存格的類型改為 Code。
    - Shift + Enter：執行當前儲存格並跳到下一個儲存格。
    - Ctrl + S：儲存 Notebook。
- Edit Mode
  - 當在儲存格中進入編輯模式時，就會進入 edit mode。在這個模式下，可以編輯儲存格中的內容。在 edit mode 中，可以使用一些鍵盤快捷鍵，例如 Tab 鍵自動補齊程式碼、Ctrl + / 註解程式碼等等。
  - 以下是一些常見的鍵盤快捷鍵：
    - Esc：回到 command mode。
    - Ctrl + Enter：執行當前儲存格。
    - Shift + Enter：執行當前儲存格並跳到下一個儲存格。
    - Tab：自動補齊程式碼。
    - Ctrl + /：註解或取消註解程式碼。

編輯模式 (Edit Mode)	指令模式 (Command Mode)
	

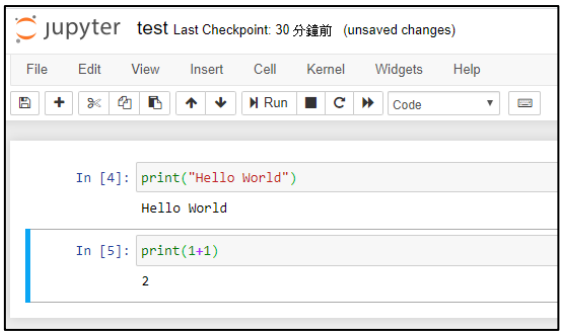
Esc 進入指令模式 Shift + Enter 執行該 cell，同時選擇下方 cell Alt + Enter 執行該 cell，同時向下新增 cell Ctrl + Enter 執行該 cell Tab 向右縮排 (indent) Shift + Tab 向左縮排 (indent) Ctrl + Z 取消 (放棄目前編輯結果;回上一步) Ctrl + Y 重做 (重現上一次編輯結果) Ctrl + Shift + - 將目前程式碼獨立成一個 cell 程式碼後方按 Tab 鍵 自動完成 (類似語法提示)	Enter 進入編輯模式 Shift + Enter 執行該 cell，同時選擇下方 cell Alt + Enter 執行該 cell，同時向下新增 cell Ctrl + Enter 執行該 cell M Markdown 模式 (可參考這裡) Y Cell 模式 A (above) 或 B (below) 新增 cell 在 上/下 方 Shift + L 顯示/隱藏行號 D, D (連按兩次 D) 刪除 cell Ctrl + S 儲存 Notebook

表: cell 有分兩種模式

執行 cell 中的程式碼

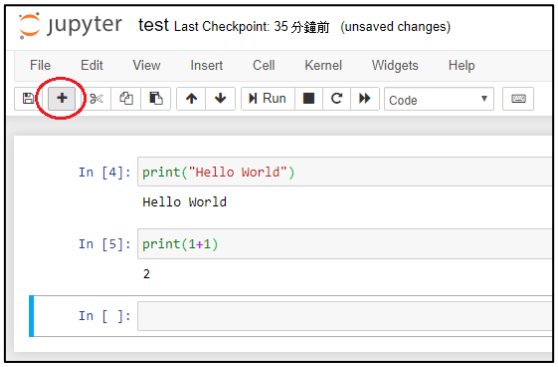


圖：在第一個 cell 輸入「print("Hello World")」，按下「Alt + Enter」



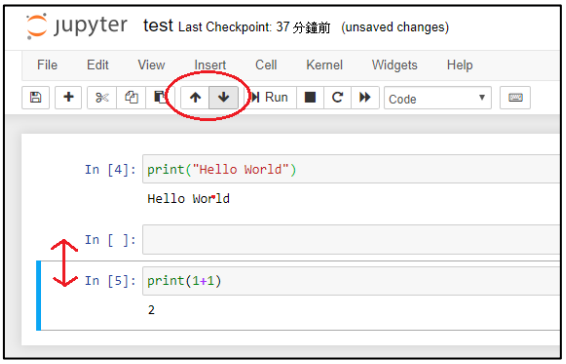
圖：在第二個 cell 輸入「print(1+1)」，按下 Ctrl + Enter 的輸出結果

新增 cell



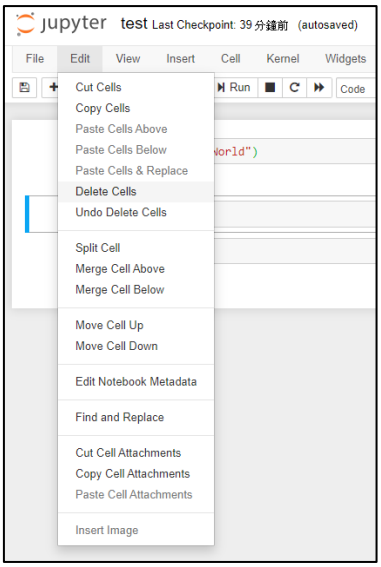
圖：按下「+」符號，會在當前 cell 下新增空白的 cell

移動 cell 位置



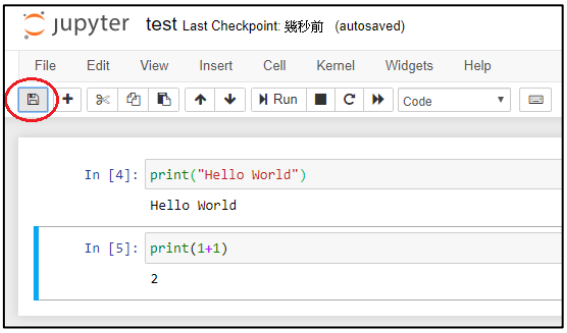
圖：選定 cell 後，按下「↑」或「↓」，可以移動 cell 的位置

刪除 cell



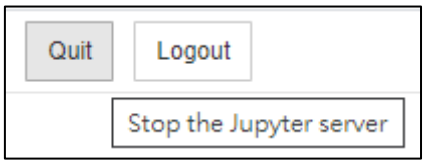
圖：選定 cell 後，按下「Edit」，選擇「Delete Cells」，即可刪除 cell

儲存 notebook

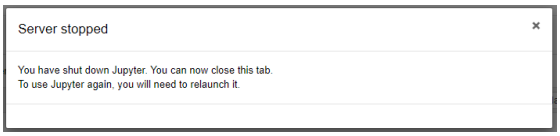


圖：按下儲存鈕，即可儲存 notebook

關閉 Jupyter Server



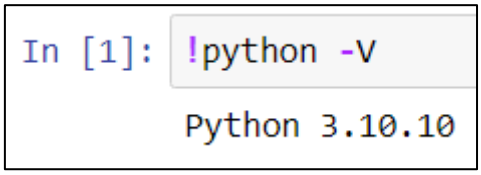
圖：按下 Quit，即可關閉 Jupyter Server



圖：Jupyter Server 關閉成功的訊息

在 Jupyter Notebook 中執行指令

我們可以直接在 cell 中，以「!」開頭，讓 Jupyter Notebook 了解到我們輸入的不是程式碼，而是指令。因為可以在 cell 中執行指令的關係，意味著也可以在 Jupyter Notebook 裡安裝套件。



圖：在 Jupyter Notebook 中執行指令

參考資料

[1] Installing Jupyter
https://jupyter.org/install
[2] Jupyter Notebook 使用技巧彙整：建置不同程式語言的核心
https://pyradise.com/jupyter-notebook-tricks-kernels-9350502ccb69
[3] 如何使用 Markdown 語言撰寫技術文件
https://experienceleague.adobe.com/docs/contributor/contributor-guide/writing-essentials/markdown.html?lang=zh-Hant
[4] 15 Tips and Tricks for Jupyter Notebook that will ease your Coding Experience
https://towardsdatascience.com/15-tips-and-tricks-for-jupyter-notebook-that-will-ease-your-coding-experience-e469207ac95c
[5] Using R language with Anaconda
https://docs.anaconda.com/anaconda/user-guide/tasks/using-r-language/
[6] Anaconda 建立 JupyterLab 的 Python 與 R 混合環境教學
https://officeguide.cc/anaconda-jupyterlab-python-r-kernel-tutorial/
[7] 使用機器學習解決問題的五步驟：定義問題
https://datasciocean.tech/machine-learning-basic-concept/machine-learning-define-problem/
[8] 真・資料團隊與分工
https://blog.v123582.tw/2020/10/31/真・資料團隊與分工/
[9] 机器学习模型评估指标
https://zhuanlan.zhihu.com/p/86120987

Module 2. 資料處理

Numpy

NumPy (Numerical Python) 是一個用於 Python 程式語言的科學計算套件，具有以下特性：

- 多維度陣列：
  - NumPy 提供了 ndarray (n-dimensions 的 array，透過 Array creation routine 來自動建立陣列) 這個物件，可以讓使用者方便地操作多維度的陣列資料，比起原生的 Python List 或 Tuple 更有效率。
- 數學函式庫：
  - NumPy 包含許多數學函數，如線性代數、隨機數產生、統計函數等，方便用戶進行科學計算。
- 廣播功能：
  - NumPy 允許在不同形狀的陣列上進行操作，例如相加、減去、乘以等，其機制稱為「廣播」，大幅減少了開發人員的程式碼量。
- 效能優化：
  - NumPy 是用 C 語言實現的，透過 Python 語言對其進行高階的操作，使得其在效能上有顯著的提升，尤其是在處理大量數據時更為明顯。

Numpy.ndarray 物件的屬性

屬性	說明
ndarray.ndim	rank，即軸 (axis) 的數量或維度的數量
ndarray.shape	Array 的維度，對於矩陣，n 列 m 行
ndarray.size	Array 元素的總個數，相當於 .shape 中 n*m 的值
ndarray.dtype	ndarray 物件的元素類型
ndarray.itemsize	ndarray 物件中每個元素的大小，以位元組為單位
ndarray.flags	ndarray 物件的記憶體資訊
ndarray.real	ndarray 元素的實部
ndarray.imag	ndarray 元素的虛部
ndarray.data	包含實際 Array 元素的 buffer，由於一般通過 Array 的索引獲取元素，所以通常不需要使用這個屬性。

表: ndarray 屬性

Numpy 的資料型態 (date-type, dtype)

名稱	描述
bool_	布林型態資料型態（True 或者 False）
int_	預設的整數型態（類似於 C 語言中的 long，int32 或 int64）
intc	與 C 的 int 型態一樣，一般是 int32 或 int 64
intp	用於索引的整數型態（類似於 C 的 ssize_t，一般情況下仍然是 int32 或 int64）
int8	位元組（-128 to 127）
int16	整數（-32768 to 32767）
int32	整數（-2147483648 to 2147483647）
int64	整數（-9223372036854775808 to 9223372036854775807）
uint8	無符號整數（0 to 255）
uint16	無符號整數（0 to 65535）
uint32	無符號整數（0 to 4294967295）
uint64	無符號整數（0 to 18446744073709551615）
float_	float64 型態的簡寫
float16	半精度浮點數，包括：1 個符號位，5 個指數位，10 個尾數位
float32	單精度浮點數，包括：1 個符號位，8 個指數位，23 個尾數位
float64	雙精度浮點數，包括：1 個符號位，11 個指數位，52 個尾數位
complex_	complex128 類型的簡寫，即 128 位複數
complex64	複數，表示雙 32 位浮點數（實數部分和虛數部分）
complex128	複數，表示雙 64 位浮點數（實數部分和虛數部分）

表: Numpy 的資料類型

陣列生成(創建)函式 (Array creation routine)

函式	說明
numpy.array	從 Python 原生的列表或元組中建立 Numpy array。
numpy.zeros	建立全部為 0 的 Numpy array。
numpy.ones	建立全部為 1 的 Numpy array。
numpy.empty	建立指定形狀和大小但元素為未初始化的 Numpy array。
numpy.arange	建立一維 Numpy array，其中元素從指定的開始值到結束值（不包括結束值）之間連續生成。
numpy.linspace	建立一維 Numpy array，其中元素從指定的開始值到

	結束值（包括結束值）之間平均生成。
numpy.random.rand	建立給定形狀的 Numpy array，其中元素是從均勻分布中隨機生成的。
numpy.random.randint	建立給定形狀的 Numpy array，其中元素是從指定的整數數值範圍生成的。

表: 陣列生成函式

2-1 Numpy 建立陣列

安裝 Numpy

- 若有語法不了解的地方，可以參考以下連結:
  - NumPy Reference
  - NumPy Tutorial
  - NumPy 教程

```
'''
註:
- 也可以在 conda 的虛擬環境中 (如 da) 裡面安裝 Numpy。
  - 指令: pip install -U numpy。
- 如果直接安裝 pandas，會連同 numpy 一起安裝。
  - 指令: pip install -U pandas
'''
!pip install -U pandas
```

```
# 匯入套件
import numpy as np
```

建議遇到不確定如何使用的語法，可以經常查詢官方的文件

- doc 版本列表
- 當前 stable 版本

np.array(): 產生陣列 (Array)

```
# 語法查詢
np.array?
```

```
# 在未完成的語法後面按下 tab 鍵，會開啟自動完成功能 (語法提示)
np.ar
```

```
'''
numpy.array(object, dtype=None, *, copy=True,
order='K', subok=False, ndmin=0, like=None)
'''
# 0 維陣列
```

```
66
```

a = np.array(33) print(a + a)	

# 一維陣列 	
a = np.array([0,1,2,3])
'''
註:
np.array() 裡面的 [0,1,2,3]，是 Python 的 List (串列) 變數，
而使用 np.array([0,1,2,3]) 賦值之後的 arr1，
它是真正的 Array (陣列)
'''
 	
# 在 cell 裡面，程式碼最後一行是變數名稱時，可以自動預覽結果 	
a
	array([0, 1, 2, 3])

# 如果希望宣告變數後，直接預覽變數，可以用「;」隔開，寫在同一行: 	
a = np.array([0,1,2,3]); a
	array([0, 1, 2, 3])

# 二維陣列 	
'''
長這個樣子:
arr2 = [
  [0,1,2],
  [3,4,5]
]
'''
a = np.array([[0,1,2], [3,4,5]]); a
	array([[0, 1, 2],
	  [3, 4, 5]])

Numpy 的 shape, ndim, dtype	
# shape (形狀，以 tuple 格式呈現) ''' (2, 3) 代表 2 維陣列: - 有 2 列，每 1 列有 3 個元素， - 也可看成幾個 row、幾個 column ''' a.shape	(2, 3)
# ndim (n-dimensions，維度)	2

a.ndim	
# dtype (data-type，陣列當中每一個元素的屬性) a.dtype	dtype('int32')
# 1 維陣列的 shape 長什麼樣子? ''' 說明: - (4) 代表一個數字 - (4,) 代表一個 tuple (1 維陣列) - (1, 4) 代表 2 維陣列，裡面只有 1 維的資料，該維有 4 個元素，例如 np.array([[1,2,3,4]]) ''' a.shape	(2, 3)

np.arange(): 從數值範圍來建立陣列	
# 跟 range() 用法一樣 ''' 用法: numpy.arange([start,]stop, [step,]dtype=None, *, like=None) ''' a = np.arange(10); a	array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

# 取得陣列 [1, 2, 3, 4, 5, 6, 7, 8, 9] 	
a = np.arange(1, 10); a
	array([1, 2, 3, 4, 5, 6, 7, 8, 9])

# 1 ~ 10 之間，每隔 3 個元素，加入資料到陣列: [1, 4, 7] a = np.arange(1, 10, 3); a	array([1, 4, 7])

Numpy 的資料型態 (data-type)	

# 自動判斷資料型態 	
a = np.array([1,2,3,4]); a.dtype
	dtype('int32')

# 自動判斷資料型態 (其中一個元素變成浮點數，dtype 會自動轉型) 	
a = np.array([1., 2, 3, 4]); a.dtype
	dtype('float64')

# 指定陣列每一個資料的資料型態 (指定 float 預設為 float64) 	
a = np.array([1, 2, 3, 4], dtype='float')
a
	array([1., 2., 3., 4.])

# arr8.dtype	
# 字元/字串用於 numpy ''' 雖然 NumPy 宣稱數值型 (Numeric) 的 Python，但也可以處理字串/字元 dtype('<U4') = Unicode + 最多放 4 個字元 ''' a = np.array(['Good', 'job', '!']); a.dtype	dtype('<U4')

# 布林值用於 numpy 	
a = np.array([True, True, False]); a.dtype
	dtype('bool')

# 類型轉換 (float 轉成 int) 	
a = np.array([1, 2, 3.14]) # dtype('float64')
a_ = a.astype(int); a_
	array([1, 2, 3])

Array creation routine 用於創建多維陣列的函數	
# np.linspace(): 建立等距陣列的 1 維陣列 ''' np.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None) ''' a = np.linspace(1, 9, num=3); a	array([1., 5., 9.])
# np.ones(): 建立填滿 1 的陣列，透過 shape 指定幾維 ''' numpy.ones(shape, dtype=None, order='C', *, like=None)[source] ''' a = np.ones((2,3)); a	array([[1., 1., 1.], [1., 1., 1.]])
# np.zeros(): 建立填滿 0 的陣列，透過 shape 指定幾維 ''' numpy.zeros(shape, dtype=float, order='C', *, like=None) ''' a = np.zeros((5,)); a	array([0., 0., 0., 0., 0.])

# np.empty(): 建立尚未初始化的陣列，裡面的元素是隨機產生的結果，換句話說，只產生陣列，值都是隨機產生的 # 註: 若需要初始化 (同時給元素預設的值)，建議使用 np.zeros、np.ones 或 np.fill ''' numpy.empty(shape, dtype=float, order='C', *, like=None) ''' a = np.empty((5, 6), dtype=float); a	array([[9.70329585e-312, 9.70301637e-312, 0.00000000e+000, 0.00000000e+000, 4.81884868e+252, 5.02034658e+175], [6.58544438e-066, 3.88451646e-057, 2.62251230e+180, 3.76874995e-061, 2.76761163e-057, 7.43172351e-144], [3.59751658e+252, 3.96046095e+246, 1.04918621e-153, 7.69165785e+218, 5.04621343e+180, 1.04917822e-153], [9.08366793e+223, 6.58544438e-066, 3.88451646e-057, 2.62251230e+180, 3.76874995e-061, 1.04917592e-153], [1.94918966e-153, 2.57707508e-057, 2.26368443e-076, 6.23583732e-038, 2.59027896e-144, 7.79952704e-143]])
# np.full(): 建立填滿 fill_value 的陣列，透過 shape 指定幾維 ''' numpy.full(shape, fill_value, dtype=None, order='C', *, like=None) ''' a = np.full((3,4), 10); a	array([[10, 10, 10, 10], [10, 10, 10, 10], [10, 10, 10, 10]])

2-2 Numpy 一維陣列.ipynb
import numpy as np

一維陣列的四則運算	

# 初始化陣列 	
x = np.array([2, 4, 6, 8, 10])
y = np.array([10, 8, 6, 4, 2])	

# 整數與陣列: 加法 	
a = x + 4; a
	array([6, 8, 10, 12, 14])

# 整數與陣列: 減法 	
a = x - 2; a
	array([0, 2, 4, 6, 8])

# 整數與陣列: 乘法 	
a = x * 2; a
	array([4, 8, 12, 16, 20])

# 整數與陣列: 除法 	
a = x / 2; a
	array([1., 2., 3., 4., 5.])

# 陣列加法運算 	
a = x + y; a
	array([12, 12, 12, 12, 12])

# 陣列減法運算	array([-8, -4, 0, 4, 8])

a = x - y; a	

# 陣列乘法運算 	
a = x * y; a
	array([20, 32, 36, 32, 20])

# 陣列除法運算 	
a = x / y; a
	array([0.2, 0.5, 1. , 2. ,
	5.])

# 陣列元素平方 ''' 也可以使用 A = np.square(x) B = np.square(y) ''' A = x ** 2 B = y ** 2	
A	array([4, 16, 36, 64, 100])
B	array([100, 64, 36, 16, 4])

# 陣列元素開根號 	
A = np.sqrt(x)
B = np.sqrt(y)	

A	array([1.41421356, 2., 2.44948974, 2.82842712, 3.16227766])
B	array([3.16227766, 2.82842712, 2.44948974, 2., 1.41421356])

關係運算子運算 以 大於 (>)、小於 (<)、等於 (==) 為例	

# 大於 	
a = x > y; a
	array([False, False, False,
	True, True])

# 小於 	
a = x < y; a
	array([True, True, False,
	False, False])

# 等於 	
a = x == y; a
	array([False, False, True,
	False, False])

陣列索引與切片 (Indexing & Slicing)	
# 指定索引 x[0]	2
# 切片 (範圍: [start, end-1, step]) x[0:2]	array([2, 4])
# 切片，指定每幾步算一次 ''' 這樣也可以 x[::2] ''' x[0:5:2]	array([2, 6, 10])
# 切片 (使用負號) x[-1]	10
# 切片 (範圍: [start, end-1, step]) x[-3:-1]	array([6, 8])

陣列的結合與加入	
# 陣列結合 ''' numpy.concatenate((a1, a2, ...), axis=0, out=None, dtype=None, casting="same_kind") ''' z = np.concatenate((x, y)); z	array([2, 4, 6, 8, 10, 10, 8, 6, 4, 2])

# 將陣列元素加入其它陣列 	
z = np.concatenate((x, [12, 14,
16])); z
	array([2, 4, 6, 8, 10, 12,
	14, 16])

在陣列指定索引插入元素 insert() 原本 x 是 [2,4,6,8,10]	
# 在陣列指定索引 2 插入元素 9 ''' numpy.insert(arr, obj, values, axis=None) ''' z = np.insert(x, 2, 9); z	array([2, 4, 9, 6, 8, 10])

# 在陣列指定索引 1 跟 3，分別插入元素 7 跟 9 	
z = np.insert(x, [1,3], [7,9]); z
	array([2, 7, 4, 6, 9, 8, 10])

刪除指定索引的陣列元素 delete()	
# 刪除索引 1 的元素 ''' numpy.delete(arr, obj, axis=None) ''' z = np.delete(x, 1); z	array([2, 6, 8, 10])

# 刪除索引 1 和 3 所放置的元素 	
z = np.delete(x, [1, 3]); z
	array([2, 6, 10])

向量內部元素總和	
# 計算元素總和 ''' numpy.sum(a, axis=None, dtype=None, ...) ''' c = np.array([1, 3, 5, 7]) d = np.sum(c); d	16

向量內積 (inner product)	
''' 內積在機器學習的領域，是很重要的概念， 無論是取得向量特徵，或是向量間的關係， 內積都扮演重要的角色 numpy.inner(a, b, /) 其它還有 numpy.cross 叉積 numpy.outer 外積 可以自行嘗試 ''' a = np.array([4,5,1,3,4]) b = np.array([2,3,5,4,2])	48

```
c = np.inner(a, b); c
```

牛刀小試: 餘弦相似度 (Cosine Similarity)

參考網頁: https://zh.wikipedia.org/wiki/余弦相似性

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

假設某 A 的向量為 [4,5,1,3,4]，某 B 的向量為 [2,3,5,4,2]，餘弦相似度的算法為:

$$\frac{4*5 + 5*3 + 1*5 + 3*4 + 4*2}{\sqrt{4^2 + 5^2 + 1^2 + 3^2 + 4^2} * \sqrt{5^2 + 3^2 + 5^2 + 4^2 + 2^2}}$$

其值介於 0 到 1 之間的浮點數。

2-3 Numpy 二維陣列

```
import numpy as np
# 建立二維陣列
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6], [7, 8]])
```

二維陣列相對位置與四則運算

```
# 與整數的加法運算
a = x + 10; a
```

```
array([[11, 12],
       [13, 14]])
```

```
# 加法運算
a = x + y; a
```

```
# 加法運算
a = x + y; a
```

```
# 相對位置乘法運算
'''
註: 這裡是陣列運算，與矩陣 (matrix)
運算是不一樣的作法
'''
```

```
array([[ 5, 12],
       [21, 32]])
```

a = x * y; a	

# 相對位置除法運算 	
a = x / y; a
	array([[0.2 , 0.33333333],
	[0.42857143, 0.5]])

關係運算子運算	

# 大於 	
a = x > y; a
	array([[False, False],
	[False, False]])

# 小於 	
a = x < y; a
	array([[True, True],
	[True, True]])

取得與設定二維陣列元素 (Indexing)	
# 取得二維陣列某元素內容 x[0, 1] # row = 0, col = 1	2

# 設定二維陣列某元素內容 	
tmp = np.array([[5, 5, 6, 6], [3, 3, 1, 2]])
tmp[1, 2] = 11; tmp
	array([[5, 5, 6, 6],
	[3, 3, 11, 2]])

# 取得 row = 0 的元素 ''' x[0] 的內容: [[1,2], [3,4]] 以下方式同樣效果 x[0,] x[0,:] ''' x[0]	array([1, 2])
# 取得 column = 0 的元素 ''' x[0] 的內容: [[1,2], [3,4]] 以下方式同樣效果 x[:,0] ''' x[:,0]	array([1, 3])

切片 (Slicing)
在這裡使用 xx = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

xx = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])	
# 取得 row = 0 的前 3 個元素 xx[0, :3]	array([1, 2, 3])
# 取得 row=0:2, column=2:4 ''' 實際上是 row 0 ~ row 1，以及 column 2 和 column 3 的資料 ''' xx[0:2, 2:4]	array([[3, 4], [7, 8]])
# 取得前 2 個 row 的元素 xx[:2]	array([[1, 2, 3, 4], [5, 6, 7, 8]])
# 取得 row = 1 之後的元素 xx[1:]	array([[5, 6, 7, 8], [9, 10, 11, 12]])

更改陣列形狀 (即是改變陣列維度)	

# 建立一維陣列 	
x1 = np.array([1, 2, 3, 4, 5, 6])	

# 轉成 2x3 陣列 (2 列 3 行，2 rows，3 columns) 	
y1 = x1.reshape(2, 3); y1
	array([[1, 2, 3],
	[4, 5, 6]])

# 將 2x3 陣列轉成一維陣列 ''' numpy.ravel(a, order='C') ''' x2 = y1.ravel(); x2	array([1, 2, 3, 4, 5, 6])
# 將 2x3 陣列，改成 3x2 陣列 y1.resize(3,2); y1	array([[1, 2], [3, 4], [5, 6]])

轉置矩陣 transpose() 把 n x m 的矩陣，變成 m x n 的矩陣	

# 生成一維陣列 (0 - 7)，reshape 成 4 列 2 行後，再 transpose 成為 2 列 4 行 	
a = np.arange(8).reshape(4, 2); a
'''
reshape 後的結果:
	array([[0, 2, 4, 6],
	[1, 3, 5, 7]])

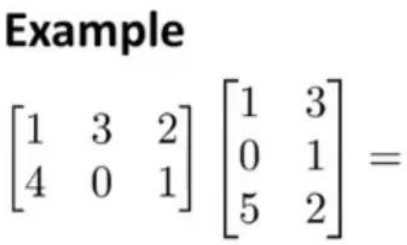
```
[[0, 1],
 [2, 3],
 [4, 5],
 [6, 7]]
'''
b = a.transpose(); b
```

```
# 也可以使用 T 來進行 transpose
a = np.arange(8).reshape(4, 2); a
b = a.T; b
```

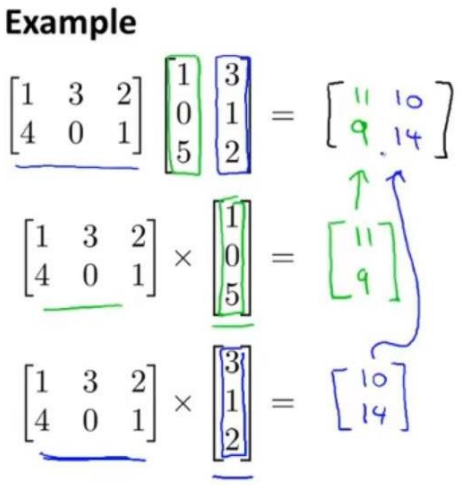
```
array([[0, 2, 4, 6],
       [1, 3, 5, 7]])
```

矩陣乘法運算

這裡所說的矩陣乘法運算，就是線性代數的乘法，以下圖為例:



計算過程:



參考資料: 线性代数基础——矩阵和矩阵的乘法

```
# 使用 dot() 來執行 2 個 2x2 矩陣乘法運算
a = np.dot(x, y); a
```

```
array([[19, 22],
       [43, 50]])
```

# 使用 @ 來執行 2 個 2x2 矩陣乘法運算 	
a = x @ y; a
	array([[19, 22],
	[43, 50]])

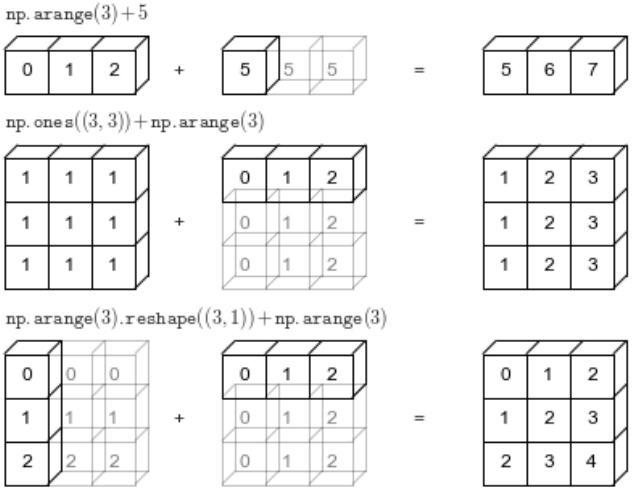
# 使用 @ 來執行上方圖片中的算式 	
l = np.array([[1,3,2], [4,0,1]])
r = np.array([[1,3],[0,1],[5,2]])
a = l @ r; a
	array([[11, 10],
	[9, 14]])

簡單線性代數運算	
# 一元二次方程式 ''' 方程式: a^2 + bx + c = 0 公式: x = (-b + math.sqrt(b ** 2 - 4 * a * c)) / 2 * a x = (-b - math.sqrt(b ** 2 - 4 * a * c)) / 2 * a 求: 3 * x**2 + 5 * x + 1 = 0 的根 (root) 計算結果: import math x = (-5 + math.sqrt(13)) / 6; x x = (-5 - math.sqrt(13)) / 6; x array([-1.43425855, -0.23240812]) ''' r = np.roots([3, 5, 1]); r	array([-1.43425855, -0.23240812])
# 聯立線性方程式 ''' 求下最公式的 x 和 y: 3 * x + 5 * y = 18 2 * x + 3 * y = 11 ''' coefficient_matrix = np.array([[3, 5], [2, 3]]) dependent_variable = np.array([18, 11]) a = np.linalg.solve(	array([1., 3.])

coefficient_matrix, dependent_variable); a	

廣播 (broadcast)

執行 2 個陣列運算時，原則上必須外形相同才能運算，如果不同，可以使用廣播 (broadcasting) 機制，將小陣列「擴大」到兩個陣列都相同，之後再進行運算。



參考資料: Computation on Arrays: Broadcasting

# np.arange(3) + 5 是否等於 [5,6,7] ? 	
a = np.arange(3) + 5; a
	array([5, 6, 7])

# np.ones((3, 3)) + np.arange(3) 是否等於 [[1,2,3],[1,2,3],[1,2,3]] ? 	
a = np.ones((3, 3)) + np.arange(3); a
	array([[1., 2., 3.],
	[1., 2., 3.],
	[1., 2., 3.]])

# np.arange(3).reshape((3,1)) + np.arange(3) 是否等於 [[0,1,2],[1,2,3],[2,3,4]] ? 	
a = np.arange(3).reshape((3,1)) + np.arange(3); a
	array([[0, 1, 2],
	[1, 2, 3],
	[2, 3, 4]])

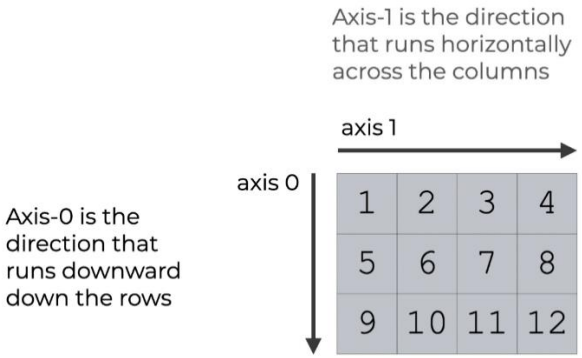
2-4 Numpy 簡單數學運算

「加總」sum()、「乘積」prod()、「差」diff()

```
import numpy as np
```

`# 一維陣列的「和」 (加總)` `np.sum([1, 2, 3, 4])`	10
`# 二維陣列的「和」 (加總)` `np.sum([[1,2], [3,4]])`	10

軸 (axis) 的概念



參考資料: Numpy Axes, Explained

`# 二維陣列的「和」，指定軸 (axis) 來計算`
`'''`
`[`
`  [1,2],`
`  [3,4],`
`  [5,6]`
`]`
 	
`axis = None:`
`- 所有元素，不分列 (row) 或行 (column)。預設值。`
`axis = 0:`
`- 順著 [0][0], [1][0], [2][0], ... 等索引號碼增加的方向`
`- 類似座標的 y 軸，針對每一個元素 y 軸對應的位置 (由上而下) 進行計算`
`axis = 1:`
`- 順著 [0][0], [0][1], [0][2], ... 等索引號碼增加的方向`
`- 類似座標的 x 軸，針對每一個元素 x 軸對應的位置 (由左而右) 進行計算`
`'''`
`np.sum([[1,2], [3,4], [5,6]], axis=None)`
	21

`# 一維陣列的「乘積」`	24

`np.prod([1, 2, 3, 4])`	
`# 二維陣列的「乘積」` `np.prod([[1, 2], [3, 4]])`	24

`# 二維陣列的「乘積」，指定軸 (axis) 來計算`
`'''`
`[`
`  [1,2],`
`  [3,4],`
`  [5,6]`
`]`
 	
`axis = None:`
`- 所有元素，不分列 (row) 或行 (column)。預設值。`
`axis = 0:`
`- 順著 [0][0], [1][0], [2][0], ... 等索引號碼增加的方向`
`- 類似座標的 y 軸，針對每一個元素 y 軸對應的位置 (由上而下) 進行計算`
`axis = 1:`
`- 順著 [0][0], [0][1], [0][2], ... 等索引號碼增加的方向`
`- 類似座標的 x 軸，針對每一個元素 x 軸對應的位置 (由左而右) 進行計算`
`'''`
`np.prod([[1,2], [3,4], [5,6]], axis=None)`
	720

`# 一維陣列的「差」(後一個元素的值，減去前一個元素的值)`
`'''`
`numpy.diff(a, n=1, axis=-1, prepend=<no value>, append=<no value>)`
`'''`
`x = np.array([7, 5, 3, 3, 9, 6, 7])`
`np.diff(x)`
	`array([-2, -2,  0,  6, -3,  1])`

`# 一維陣列的「差」(指定 n，代表 diff 幾次)` `np.diff(x, n=2)`	`array([ 0,  2,  6, -9,  4])`

# 二維陣列的「差」，指定軸 (axis) 來計算 	
'''
[
    [1,2],
    [3,4],
    [5,6]
]
 	
axis = None:
- 所有元素，不分列 (row) 或行 (column)。預設值。
axis = 0:
- 順著 [0][0], [1][0], [2][0], ... 等索引號碼增加的方向
- 類似座標的 y 軸，針對每一個元素 y 軸對應的位置 (由上而下) 進行計算
axis = 1:
- 順著 [0][0], [0][1], [0][2], ... 等索引號碼增加的方向
- 類似座標的 x 軸，針對每一個元素 x 軸對應的位置 (由左而右) 進行計算
'''
x = np.array([[1,2], [3,4], [5,6]])
np.diff(x, axis=0)
	array([[2, 2],
	       [2, 2]])

捨去函數	
# around(): 四捨五入到最近的「偶數值」 ''' numpy.around(a, decimals=0, out=None) decimals: 指定小數位數 (小數點後面第幾位) ''' np.around([0.45, 1.85], 1)	array([0.4, 1.8])

# rint(): 回傳最近的整數 (四捨五入到最近的「偶數值」) 	
np.rint([1.5, 2.5, 1.6, 3.3])
	array([2., 2., 2., 3.])

# floor(): 無條件捨去 	
np.floor([1.5, 2.5, 1.6, 3.3])
	array([1., 2., 1., 3.])

# ceil(): 無條件進位 	
np.ceil([1.1, 2.5, 1.6, 3.3])
	array([2., 3., 2., 4.])

# trunc(): 捨棄小數 	
np.trunc([1.1, 2.5, 1.6, 3.3])
	array([1., 2., 1., 3.])

指數 (exp) 與對數 (log)	
# exp(): 自然對數 e 的幾次方 ''' e^1 = 2.71828183 e^2 = 7.3890561 e^3 = 20.08553692 e^4 = 54.59815003 . . . ''' np.exp([1, 2, 3, 4])	array([2.71828183, 7.3890561 , 20.08553692, 54.59815003])

# exp2(): 2 的幾次方 	
np.exp2([1, 2, 3, 4])
	array([2., 4., 8., 16.])

# log(): 計算自然對數的值 ''' np.e = 自然對數 註: log(1) = 0 (任何數的 0 次方，等於 1) log(e) = 1 log(e^2) = 2 ''' np.log([1, np.e, np.e ** 2])	array([0., 1., 2.])

# log2(): 計算以 2 為底的值 	
np.log2([1, 2, 4, 8])
	array([0., 1., 2., 3.])

# log10(): 計算以 10 為底的值 	
np.log10([1, 10, 100, 1000])
	array([0., 1., 2., 3.])

其它

# absolute(): 回傳絕對值 	
np.absolute([-3, 3])
	array([3, 3])

# square(): 回傳平方值 	
np.square([4, 9])
	array([16, 81])

# sqrt(): 回傳平方根 	
np.sqrt([4, 9])
	array([2., 3.])

隨機函數 ● np.random.rand(dim0[, dim1, dim2, ...]) ● np.random.randint(low, high=None, size=None) ● 參考資料: ■ Random sampling ■ python numpy 常用随机数的产生方法	
''' np.random.seed(42) numpy.random.rand(): 設定種子後，取得隨機亂數 註: np.random.seed() 的引數固定，隨機產生的值也會固定，亂數的值介於 [0, 1) ''' np.random.seed(42) a = np.random.rand(3); a	array([0.37454012, 0.95071431, 0.73199394])
''' np.random.randint(start, end - 1, size=數量): 產生 start 到 end - 1 之間的整數值 隨機產生 100 位同學的成績 ''' a = np.random.randint(0, 100, size=(1, 100), dtype='int32'); a	array([[60, 20, 82, 86, 74, 74, 87, 99, 23, 2, 21, 52, 1, 87, 29, 37, 1, 63, 59, 20, 32, 75, 57, 21, 88, 48, 90, 58, 41, 91, 59, 79, 14, 61, 61, 46, 61, 50, 54, 63, 2, 50, 6, 20, 72, 38, 17, 3, 88, 59, 13, 8, 89, 52, 1, 83, 91, 59, 70, 43, 7, 46, 34, 77, 80, 35, 49, 3, 1, 5, 53, 3, 53, 92, 62, 17, 89, 43, 33, 73,

	61, 99, 13, 94, 47, 14, 71, 77, 86, 61, 39, 84, 79, 81, 52, 23, 25, 88, 59, 40]])
''' np.random.shuffle(): 陣列元素重新排列 ''' x = np.arange(10) np.random.shuffle(x) x	array([1, 5, 2, 4, 3, 7, 9, 6, 8, 0])

一點統計的概念	

# mean(): 元素平均值 	
x = np.array([[1, 2], [3, 4]])
np.mean(x)
	2.5

# mean() 加上 axis 	
np.mean(x, axis = 0)
	array([2., 3.])

# median(): 中位數 	
x = np.array([[12, 7, 4], [3, 2,
6]])
np.median(x)
	5.0

# median() 加上 axis 	
np.median(x, axis = 0)
	array([7.5, 4.5, 5.])

# var(): 變異數 (variance) (離散隨機變數) (一維陣列) ''' import math mean = (1 + 2 + 3 + 4) / 4 (math.pow(1 - mean, 2) + math.pow(2 - mean, 2) + math.pow(3 - mean, 2) + math.pow(4 - mean, 2)) / 4 補充: var() 也可以設定 axis ''' x = np.array([1, 2, 3, 4])	1.25

`np.var(x)`	

`# var(): 變異數 (variance) (離散隨機變數) (二維陣列)`
`x = np.array([[1, 2, 3, 4], [2, 4, 6, 8]])`
`np.var(x)`
	`4.6875`

`# var() 加上 axis` `np.var(x, axis=0)`	`array([0.25, 1.  , 2.25, 4.  ])`

`# std(): 標準差 (standard deviation)`
`'''`
`變異數 = 「標準差」的平方`
`標準差 = 「變異數」開根號`
`'''`
`np.std(x)`
	`2.165063509461097`

`# std() 加上 axis` `np.std(x, axis=0)`	`array([0.5, 1. , 1.5, 2. ])`

Pandas

Pandas 是一個用於資料處理和分析的 Python 套件，其主要特性如下：

- 強大的資料結構：
  - Pandas 支持多種資料結構，包括 Series、DataFrame 等，其中最常用的是 DataFrame。
  - DataFrame 是一個二維表格，類似於關聯式資料庫中的表格，可以方便地處理和分析資料。
- 資料清洗：
  - Pandas 提供了一系列函數，用於處理資料中的缺失值、重複值、異常值等問題，可以進行資料清洗和預處理。
- 資料操作：
  - Pandas 支持對資料進行聚合、分組、選擇、過濾、排序等操作，能夠對大量資料進行高效的處理。
- 資料視覺化：
  - Pandas 可以與 Matplotlib 等資料視覺化函式庫配合使用，可以輕鬆地對資料進行視覺化呈現。
- 資料輸入輸出：
  - Pandas 支持多種資料輸入輸出格式，包括 CSV、Excel、JSON、XML

等，方便用戶對不同格式的資料進行處理。

2-5 Pandas 使用 Series

安裝 pandas

- 若有語法不了解的地方，可以參考以下連結:
  - API reference
  - Pandas Tutorial
  - Pandas 教程

```
# 安裝套件
!pip install pandas
```

```
import pandas as pd
import numpy as np
```

使用 series

```
# 使用 list 建立 series 物件
s = pd.Series([11, 22, 33, 44, 55]); s
```

```
0    11
1    22
2    33
3    44
4    55
dtype: int64
```

```
# 使用 dict 建立 series 物件
myDict = {"蘋果": 60, "水梨": 50}
s = pd.Series(myDict); s
```

```
蘋果    60
水梨    50
dtype: int64
```

```
# 使用 Numpy 的 ndarray 建立 series 物件
s = pd.Series(np.arange(13, 24)); s
```

```
0     13
1     14
2     15
3     16
4     17
5     18
6     19
7     20
8     21
9     22
10    23
dtype: int32
```

```
# 建立含索引的 series 物件
```

```
3    10
6    20
```

myIndex = [3, 6, 9] # 也可以用字串當 index (key)
price = [10, 20, 30]
s = pd.Series(price, index=myIndex); s
	9 30
	dtype: int64

# 使用純量 (scalar) 建立 series 	
s = pd.Series(7, index=[1,2,3]); s
	1 7
	2 7
	3 7
	dtype: int64

# 列出 series 物件索引與值 	
s = pd.Series([10, 20, 30], index=['apple', 'orange', 'pear']); s
	apple 10
	orange 20
	pear 30
	dtype: int64

# 承上，印出所有的值 s.values	array([10, 20, 30], dtype=int64)
# 承上，印出所有的索引編號 s.index	Index(['apple', 'orange', 'pear'], dtype='object')

# 將切片觀念用在 series 物件 	
s = pd.Series([0, 1, 2, 3, 4, 5])	

s[3]	3
s[2:4]	2 2 3 3 dtype: int64
s[:3]	0 0 1 1 2 2 dtype: int64
s[2:]	2 2 3 3 4 4 5 5 dtype: int64
# series 只能使用 slicing，無法直接使用 -1 來取得最後一個元素的值 s[-1:]	5 5 dtype: int64

# series 物件相加 	
x = pd.Series([1, 2])
	0 4
	1 6

y = pd.Series([3, 4]) x + y	dtype: int64
# series 物件相乘 x * y	0 3 1 8 dtype: int64
# 邏輯判斷: 大於 (可以嘗試其它判斷) x > y	0 False 1 False dtype: bool

# 擁有相同的 index (或 key)，執行相加 	
fruits = ['apple', 'orange', 'pear']
x1 = pd.Series([20, 30, 40], index=fruits)
x2 = pd.Series([15, 35, 55], index=fruits)
x1 + x2
	apple 35
	orange 65
	pear 95
	dtype: int64

# 擁有不同的 index (或 key)，執行相加，不同索引之間的值相加，會填上 NaN (Not a Number) 	
fruits1 = ['apple', 'orange', 'pear']
fruits2 = ['apple', 'banana', 'pear']
x1 = pd.Series([20, 30, 40], index=fruits1)
x2 = pd.Series([15, 35, 55], index=fruits2)
x1 + x2
	apple 35.0
	banana NaN
	orange NaN
	pear 95.0
	dtype: float64

# series 的索引是字串 (key)，取得元素內容 	
fruits = ['apple', 'orange', 'pear']
x = pd.Series([20, 30, 40], index=fruits); x
	apple 20
	orange 30
	pear 40
	dtype: int64

# 取得單一 key 的資料 x['apple']	20

# 取得多個 key 的資料，要使用 list 包起來 (在 pandas 很常用) x[['apple', 'orange']]	apple 20 orange 30 dtype: int64
# 將純量與函式應用在 series 上 (x + 10) * 2	apple 60 orange 80 pear 100 dtype: int64

# 每個元素都取得平方值 	
np.square(x)
	apple 400
	orange 900
	pear 1600
	dtype: int64

2-6 Pandas 建立 DataFrame
import pandas as pd

建立 dataframe

```
# 新增一個二維 list
list_students = [
    ['Alex', 19],
    ['Bill', 22],
    ['Carl', 14],
    ['Darren', 18]
]

# 建立基本的 dataframe
df = pd.DataFrame(list_students)

# 設定 dataframe 的欄位
df.columns = ['name', 'age']; df
```

	name	age
0	Alex	19
1	Bill	22
2	Carl	14
3	Darren	18

```
# 使用 dict 來建立 dataframe
df = pd.DataFrame({
    'name': ['Alex', 'Bill', 'Carl',
'Darren'],
    'age': [19, 22, 14, 18]
}); df
```

	name	age
0	Alex	19
1	Bill	22
2	Carl	14
3	Darren	18

```python
# 使用 list of dict 來建立 dataframe
data = [
    {'name': 'Alex', 'age': 19},
    {'name': 'Bill', 'age': 22},
    {'name': 'Carl', 'age': 14},
    {'name': 'Darren', 'age': 18},
]

df = pd.DataFrame(data); df
```

	name	age
0	Alex	19
1	Bill	22
2	Carl	14
3	Darren	18

```python
'''建立「2020 ~ 2023」年「臺北、臺中、高雄」某月平均溫度 的 dataframe'''

# 建立欄位
years = range(2020, 2024)

# 臺北、臺中、高雄 某個月的平均溫度
taipei = pd.Series([20, 21, 19, 21],
index=years)
taichung = pd.Series([25, 26, 27, 28],
index=years)
kaohsiung = pd.Series([30, 29, 31, 32],
index=years)

# 建立 dataframe (axis=0 是上下合併，axis=1 是左右合併)
df = pd.concat([taipei, taichung,
kaohsiung], axis=1)

# 設定欄位
df.columns = ['taipei', 'taichung',
'kaohsiung']; df
```

	taipei	taichung	kaohsiung
2020	20	25	30
2021	21	26	29
2022	19	27	31
2023	21	28	32

2-7 Pandas 資料分析與處理

```python
import pandas as pd
import numpy as np
```

索引參照屬性

- df.**at**: 使用 index (key) 和 columns **名稱** 來「取得或設定」單一元素內容或陣列內容。
- df.**iat**: 使用 index (key) 和 columns **編號** 來「取得或設定」單一元素內容。
- df.**loc**: 使用 index (key) 或 columns **名稱** 來「取得或設定」整個 row 或 columns 的資料或陣列內容。
- df.**iloc**: 使用 index (key) 或 columns **編號** 來「取得或設定」整個 row 或 columns 的資料。

```
'''建立「2020 ~ 2023」年「臺北、臺中、高雄」某月平均溫度 的 dataframe'''

# 建立欄位
years = range(2020, 2024)

# 臺北、臺中、高雄 某個月的平均溫度
taipei = pd.Series([20, 21, 19, 21], index=years)
taichung = pd.Series([25, 26, 27, 28], index=years)
kaohsiung = pd.Series([30, 29, 31, 32], index=years)

# 建立 dataframe (axis=0 是上下合併，axis=1 是左右合併)
df = pd.concat([taipei, taichung, kaohsiung], axis=1)

# 設定欄位
df.columns = ['taipei', 'taichung', 'kaohsiung']; df
```

	taipei	taichung	kaohsiung
2020	20	25	30
2021	21	26	29
2022	19	27	31
2023	21	28	32

```
# 使用 at: 取得 row 是 2020 且 column 是 taipei 的值
df.at[2020, 'taipei']
```

```
20
```

```
# 使用 at: 取得 row 是 2023 且 column 是 kaohsiung 的值
df.at[2023, 'kaohsiung']
```

```
32
```

```
# 使用 iat: 取得 row 是 2, column 是 1 的值
```

```
27
```

```
'''
index 是 zero-based
'''
df.iat[2, 1]
```

```
# 使用 loc: 取得 row 是 2021 的資料
'''
只有指定 2021，所以回傳 series 型態
'''
df.loc[2021]
```

```
taipei       21
taichung     26
kaohsiung    29
Name: 2021, dtype: int64
```

```
# 使用 loc: 取得 row 是 2020 和 2023 的資料
'''
指定兩個 row，資料將自帶 column，回傳 dataframe
'''
df.loc[ [2020, 2023] ]
```

	taipei	taichung	kaohsiung
2020	20	25	30
2023	21	28	32

```
# 使用 loc: 取得 row 是 2021 到 2023、column 是 taichung 到 kaohsiung 的資料
'''
沒有 slicing 有關 end 需要減 1 的問題
'''
df.loc[2021:2023, "taichung":"kaohsiung"]
```

	taichung	kaohsiung
2021	26	29
2022	27	31
2023	28	32

```
# 使用 iloc: 取得 row 是 0 的資料
'''
只有指定 [0]，所以回傳 series
'''
df.iloc[0]
```

```
taipei       20
taichung     25
kaohsiung    30
Name: 2020, dtype: int64
```

直接索引

不用使用 df.at[]、df.loc[] 等方式取得資料，直接使用 `df[index 或 key]`，來取得對應的行或列的資料。

```
# 取得 column 為 taipei 的資料
df['taipei']
```

```
2020    20
2021    21
2022    19
2023    21
Name: taipei, dtype: int64
```

```
# 取得 column 是 taipei、row 是 2022 的資料
df['taipei'][2022]
```

```
19
```

```
# 取得 column 是 taipei 和 taichung 的資料
df[ ['taipei', 'taichung'] ]
```

	taipei	taichung
2020	20	25
2021	21	26
2022	19	27
2023	21	28

```
# 取得 row 索引編號 1 ~ 2 的資料
'''
透過 slicing 方式取得資料
'''
df[1:3]
```

	taipei	taichung	kaohsiung
2021	21	26	29
2022	19	27	31

```
# 取得 row 索引編號 3 之前的資料
'''
透過 slicing 方式取得資料
'''
df[:3]
```

	taipei	taichung	kaohsiung
2020	20	25	30
2021	21	26	29
2022	19	27	31

```
# 取得 taipei 溫度大於 20 的資料
'''
類似在 df 索引位置進行邏輯判斷
'''
df[ df['taipei'] > 20 ]
```

	taipei	taichung	kaohsiung
2021	21	26	29
2023	21	28	32

```
# 取得 taipei 溫度大於 20、高雄溫度大於 30 的資料
'''
判斷陳述句，要用括號包起來

&: and
|: or
'''
df[ (df['taipei'] > 20) & (df['kaohsiung'] > 30) ]
```

	taipei	taichung	kaohsiung
2023	21	28	32

四則運算

- add(): 加法
- sub(): 減法
- mul(): 乘法
- div(): 除法

註: series 跟 dataframe 都可以用

```
# 初始化
```

```
df1 = pd.DataFrame([
    {'a': 15, 'b': 15},
    {'a': 12, 'b': 18}
])
df2 = pd.DataFrame([
    {'a': 15, 'b': 13},
    {'a': 11, 'b': 22}
])
```

```
df1
```

	a	b
0	15	15
1	12	18

```
df2
```

	a	b
0	15	13
1	11	22

```
# 加法
df = df1.add(df2); df
```

	a	b
0	30	28
1	23	40

```
# 减法
df = df1.sub(df2); df
```

	a	b
0	0	2
1	1	-4

```
# 乘法
df = df1.mul(df2); df
```

	a	b
0	225	195
1	132	396

```
# 除法
df = df1.div(df2); df
```

	a	b
0	1.000000	1.153846
1	1.090909	0.818182

邏輯運算方法

- **gt()**: 大於
- **lt()**: 小於
- **ge()**: 大於等於
- **le()**: 小於等於
- **eq()**: 等於
- **ne()**: 不等於

註: series 跟 dataframe 都可以用

```
# gt(): 大於
df = df1.gt(df2); df
```

	a	b
0	False	True
1	True	False

```
# lt(): 小於
df = df1.lt(df2); df
```

	a	b
0	False	False
1	False	True

```
# ge(): 大於等於
df = df1.ge(df2); df
```

	a	b
0	True	True
1	True	False

```
# le(): 小於等於
df = df1.le(df2); df
```

	a	b
0	True	False
1	False	True

```
# eq(): 等於
df = df1.eq(df2); df
```

	a	b
0	True	False
1	False	False

```
# ne(): 不等於
df = df1.ne(df2); df
```

	a	b
0	False	True
1	True	True

Numpy 也可以用在 Pandas

```
# 把 df1 所有值平方
df = np.square(df1); df
```

	a	b
0	225	225
1	144	324

NaN 的處理

- df.**dropna()**: 將 NaN 刪除，再回傳新的 series 或 dataframe 物件。
- df.**fillna()**: 將 NaN 由特定的 value 取代，再回傳新的 series 或 dataframe 物件。
- df.**isna()**: 判斷是否為 NaN，如果是，就回傳 True，如果不是，就回傳 False。
- df.**notna()**: 判斷是否為 NaN，如果不是，就回傳 True，如果是，就回傳 False。
- df.**isnull()**: 跟 df.**isna()** 一樣。

```
# 初始化
df = pd.DataFrame([
    [1, 2, 3],
    [4, np.nan, 6],
    [7, 8, np.nan]
]); df
```

	0	1	2
0	1	2.0	3.0
1	4	NaN	6.0
2	7	8.0	NaN

```
# isna(): 判斷是否為 NaN，如果是，就回傳 True，如果不是，就回傳 False。
'''
isna() 和 isnull() 是一樣的效果，
主要原因是因為 pandas 的 dataframe 概念，
是基於 R 的 dataframe，
在 R 裡面，na 和 null 是不一樣的東西，
而 pandas 是基於 numpy，
numpy 沒有 na 也沒有 null，只有 nan，
所以 pandas
'''
df.isna()
# isnull()
```

	0	1	2
0	False	False	False
1	False	True	False
2	False	False	True

```
# notna(): 判斷是否為 NaN，如果不是，就回傳 True，如果是，就回傳 False。
df.notna()
```

	0	1	2
0	True	True	True
1	True	False	True
2	True	True	False

```python
# 在 NaN 的位置上，補上 0
df_ = df.fillna(0); df_.astype('int32')
```

	0	1	2
0	1	2	3
1	4	0	6
2	7	8	0

```python
# dropna(): 刪除含 NaN 的 row (預設)
df_ = df.dropna(); df_
```

	0	1	2
0	1	2.0	3.0

```python
# dropna(axis='columns'): 刪除含 NaN 的 column
df_ = df.dropna(axis='columns'); df_
```

	0
0	1
1	4
2	7

簡單的統計函數

- **cummax**(axis=None): 回傳指定軸所累計的最大值。
- **cummin**(axis=None): 回傳指定軸所累計的最小值。
- **cumsum**(axis=None): 回傳指定軸所累計的總和。
- **max**(axis=None): 回傳指定軸的最大值。
- **min**(axis=None): 回傳指定軸的最小值。
- **sum**(axis=None): 回傳指定軸的總和。
- **mean**(axis=None): 回傳指定軸的平均數。
- **median**(axis=None): 回傳指定軸的中位數。
- **std**(axis=None): 回傳指定軸的標準差。

```python
# 初始化
from random import randint

# dataframe 的 columns
course = ['國文', '英文', '數學', '自然', '社會']

# 各科的分數 (隨機產生)
chinese = [randint(60, 100) for x in range(5)]
```

	國文	英文	數學	自然	社會
1	73	84	98	78	70
2	64	89	61	65	94
3	61	71	65	87	64
4	100	92	87	88	92
5	100	74	75	66	61

```
english = [randint(60, 100) for x in
range(5)]
math = [randint(60, 100) for x in
range(5)]
nature = [randint(60, 100) for x in
range(5)]
society = [randint(60, 100) for x in
range(5)]

# 建立 dataframe
df = pd.DataFrame(
    [chinese, english, math, nature,
society],
    columns=course,
    index=range(1,6)
); df
```

```
# 累計國文的分數，另外新增一個欄位來放置它
x = df['國文'].cumsum()
df['小計_國文'] = x; df
```

	國文	英文	數學	自然	社會	小計_國文
1	73	84	98	78	70	73
2	64	89	61	65	94	137
3	61	71	65	87	64	198
4	100	92	87	88	92	298
5	100	74	75	66	61	398

```
# 補充: 刪除 column 的資料
'''
也可以刪除多個 columns:
df = df.drop(columns=['column_nameA',
'column_nameB'])
'''
df = df.drop('小計_國文', axis=1); df
```

	國文	英文	數學	自然	社會
1	73	84	98	78	70
2	64	89	61	65	94
3	61	71	65	87	64
4	100	92	87	88	92
5	100	74	75	66	61

```
# 列出每一個學生的總分，另外新增一個欄位來放置
它
total = [df.iloc[i].sum() for i in
range(0, 5)]
df['總分'] = total; df
```

	國文	英文	數學	自然	社會	總分
1	73	84	98	78	70	403
2	64	89	61	65	94	373
3	61	71	65	87	64	348
4	100	92	87	88	92	459
5	100	74	75	66	61	376

```
# 列出各科平均分數 (包括總分的平均分數)
avg = df.mean(); avg
```

```
國文    79.6
英文    82.0
數學    77.2
自然    76.8
社會    76.2
```

	總分　　391.8 dtype: float64
# 增加 index: 在 df 下方增加平均分數 df.loc['平均分數'] = avg; df	 | | 國文 | 英文 | 數學 | 自然 | 社會 | 總分 | |---|---|---|---|---|---|---| | 1 | 73.0 | 84.0 | 98.0 | 78.0 | 70.0 | 403.0 | | 2 | 64.0 | 89.0 | 61.0 | 65.0 | 94.0 | 373.0 | | 3 | 61.0 | 71.0 | 65.0 | 87.0 | 64.0 | 348.0 | | 4 | 100.0 | 92.0 | 87.0 | 88.0 | 92.0 | 459.0 | | 5 | 100.0 | 74.0 | 75.0 | 66.0 | 61.0 | 376.0 | | 平均分數 | 79.6 | 82.0 | 77.2 | 76.8 | 76.2 | 391.8 |
# 刪除 index: 刪除 平均分數 的 row ''' 也可以這樣寫: df = df.drop(index=['平均分數']) ''' df = df.drop('平均分數', axis=0); df	 | | 國文 | 英文 | 數學 | 自然 | 社會 | 總分 | |---|---|---|---|---|---|---| | 1 | 73.0 | 84.0 | 98.0 | 78.0 | 70.0 | 403.0 | | 2 | 64.0 | 89.0 | 61.0 | 65.0 | 94.0 | 373.0 | | 3 | 61.0 | 71.0 | 65.0 | 87.0 | 64.0 | 348.0 | | 4 | 100.0 | 92.0 | 87.0 | 88.0 | 92.0 | 459.0 | | 5 | 100.0 | 74.0 | 75.0 | 66.0 | 61.0 | 376.0 |

# 排序: 將 dataframe 物件的 總分 欄位，從大排到小 	
df = df.sort_values(by='總分',
ascending=False); df
	
	| | 國文 | 英文 | 數學 | 自然 | 社會 | 總分 |
	|---|---|---|---|---|---|---|
	| 4 | 100.0 | 92.0 | 87.0 | 88.0 | 92.0 | 459.0 |
	| 1 | 73.0 | 84.0 | 98.0 | 78.0 | 70.0 | 403.0 |
	| 5 | 100.0 | 74.0 | 75.0 | 66.0 | 61.0 | 376.0 |
	| 2 | 64.0 | 89.0 | 61.0 | 65.0 | 94.0 | 373.0 |
	| 3 | 61.0 | 71.0 | 65.0 | 87.0 | 64.0 | 348.0 |

# 經過大到小的排序後，增加名次欄位 	
rank = range(1, 6)
df['名次'] = rank; df
	
	| | 國文 | 英文 | 數學 | 自然 | 社會 | 總分 | 名次 |
	|---|---|---|---|---|---|---|---|
	| 4 | 100.0 | 92.0 | 87.0 | 88.0 | 92.0 | 459.0 | 1 |
	| 1 | 73.0 | 84.0 | 98.0 | 78.0 | 70.0 | 403.0 | 2 |
	| 5 | 100.0 | 74.0 | 75.0 | 66.0 | 61.0 | 376.0 | 3 |
	| 2 | 64.0 | 89.0 | 61.0 | 65.0 | 94.0 | 373.0 | 4 |
	| 3 | 61.0 | 71.0 | 65.0 | 87.0 | 64.0 | 348.0 | 5 |

# 依 index 從新排序 	
df = df.sort_index(ascending=True); df
	
	| | 國文 | 英文 | 數學 | 自然 | 社會 | 總分 | 名次 |
	|---|---|---|---|---|---|---|---|
	| 1 | 73.0 | 84.0 | 98.0 | 78.0 | 70.0 | 403.0 | 2 |
	| 2 | 64.0 | 89.0 | 61.0 | 65.0 | 94.0 | 373.0 | 4 |
	| 3 | 61.0 | 71.0 | 65.0 | 87.0 | 64.0 | 348.0 | 5 |
	| 4 | 100.0 | 92.0 | 87.0 | 88.0 | 92.0 | 459.0 | 1 |
	| 5 | 100.0 | 74.0 | 75.0 | 66.0 | 61.0 | 376.0 | 3 |

參考資料

[1] NumPy Reference

https://numpy.org/doc/stable/reference/index.html

[2] NumPy 教程

https://www.runoob.com/numpy/numpy-tutorial.html

[3] NumPy Tutorial

https://www.w3schools.com/python/numpy/default.asp

[4] pandas - API reference

https://pandas.pydata.org/docs/reference/index.html

[5] Pandas 教程
https://www.runoob.com/pandas/pandas-tutorial.html
[6] Pandas Tutorial
https://www.w3schools.com/python/pandas/default.asp

Module 3. 不同資料型態存取

認識資料型態-結構、半結構與非結構化資料

當談到資料型態時，通常可以分為三種不同的類型：結構化、半結構化和非結構化資料。以下是這三種類型的簡單介紹：

- 結構化資料
  - 結構化資料是指具有固定結構和明確定義的數據，通常使用表格、關聯式資料庫或電子表格等形式進行組織和儲存。常見的結構化數據格式包括 CSV、Excel、SQL 等。
  - CSV(Comma-Separated Values)是一種純文本格式，用於將資料進行表格化處理，以逗號作為字段分隔符號，每行代表一條資料記錄。例如【臺北市資料大平臺「垃圾車點位路線資訊」】(https://data.taipei/dataset/detail?id=6bb3304b-4f46-4bb0-8cd1-60c66dcd1cae)

- 半結構化資料
  - 半結構化資料是指具有某種程度上的結構化數據，但是其結構不像傳統結構化資料那麼嚴格和明確。通常使用標記語言（如 XML、HTML）或 JavaScript 物件表示法（JSON）等格式進行儲存和組織。常見的半結構化數據格式包括 JSON、XML 等。
    - JSON（JavaScript Object Notation）是一種常用的半結構化資料格式，具有簡潔輕便、易於閱讀和解析等優點，通常用於網際網路資料傳輸。例如【Café Nomad 咖啡廳遊牧民族】(https://cafenomad.tw/)
    - XML（eXtensible Markup Language）是一種用於描述資料的標記語言，可描述各種不同種類的數據，並且可進行有效的資料驗證。例如【臺北市資料大平臺「臺北市政府求職徵才職缺資訊」】(https://data.taipei/dataset/detail?id=f2f3f0d3-9e84-4fc5-af4d-5814563e17b3)

- 非結構化資料
  - 非結構化資料是指缺乏固定結構或明確定義的資料，例如文字、圖像、影音等。由於這些資料通常缺乏組織和格式，因此儲存和分析這些資料不太容易的。
  - 在這種情況下，常用的方法是使用資訊檢索、自然語言處理、電腦視覺等技術來進行資料分析。圖像和影音資料可以使用電腦視覺技術進行分析和處理，以取得圖像中的特徵和內容，以及識別和跟踪物件。文字資料可以使用自然語言處理技術 (Natural Language Processing) 來進行分析，以取得關鍵詞、標記實體和情感等信息。
  - 以取得 PDF 文字而言，可以考慮使用 tika (https://tika.apache.org/) 這個工具 (需要有 Java 環境)，來取得檔案當中的文字。

處理 CSV,JSON,XML,Excel 格式資料

有關檔案編碼的部分，請參考這個連結: standard-encodings

3-1 Pandas 檔案輸入與輸出

```
# 安裝套件
!pip install pandas openpyxl
```

```
# 匯入套件
import pandas as pd
```

讀寫 csv 檔

- CSV: Comma-Separated Values
  - pd.read_csv(): 讀取 csv 檔
  - df.to_csv(): 寫入 csv 檔
- 請另存下載 csv 檔: 垃圾車點位路線資訊 到專案目錄當中。

```
# 讀取 csv 格式檔案
'''
pd.read_csv(
    filepath_or_buffer="你的檔案路徑",
    sep="分隔符號",
    header=True 或 False,
    index=True 或 False,
    encoding=None,
    ...
```

```
)

說明:
- 引數 filepath_or_buffer 不一定要寫:
  - df = pd.read_csv("./output.csv")
- 近年許多檔案或資料，都使用 UTF-8 進行編碼
  - 一般來說，只要是 utf-8 編碼格式儲存的檔案，讀取成 dataframe 時候，不用特別指定編碼。
  - 然而本例使用的 csv 檔，需要用 ANSI 編碼來開啟。
'''
df = pd.read_csv(filepath_or_buffer="./垃圾車點位資訊.csv",
encoding="ANSI"); df
```

```
# 寫入 csv 檔案
'''
預設是 row number 和 column name 都包括進去，不習慣加進 csv 檔的話，可以這樣寫:
df.to_csv(path_or_buf="./output.csv", index=False, header=False)

另外，引數 path_or_buf 不一定要寫:
df.to_csv("./output.csv", index=False, header=False)
'''
df.to_csv(path_or_buf="./output.csv", index=False, encoding="ANSI")
```

讀寫 json 檔

- JSON（JavaScript Object Notation）
  - pd.read_json(): 讀取 json 檔
  - df.to_json(): 寫入 json 檔
- 使用 Cafe Nomad 的資料來測試。
  - 請另存下載: https://cafenomad.tw/api/v1.2/cafes/taipei

```
# 讀取 json 格式檔案
'''
參考連結:
[1] pandas.read_json
https://pandas.pydata.org/docs/reference/api/pandas.read_json.html
[2] pandas.json_normalize
https://pandas.pydata.org/docs/reference/api/pandas.json_normalize.html
```

```
'''
df = pd.read_json('./taipei.json'); df
```

```
# 寫入 json 檔案
'''
參考連結:
[1] pandas.DataFrame.to_json
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_json
.html
'''
df.to_json('./output.json', indent=None, orient='records',
lines=False, force_ascii=False)
```

讀寫 xml

- eXtensible Markup Language
  - pd.read_xml(): 讀取 xml 檔
  - df.to_xml(): 寫入 xml 檔
- 使用 臺北市政府求職徵才職缺資訊 的資料來測試
  - 請另存下載: https://dop.blob.core.windows.net/ipsnworkcontainer/jobs.xml

```
# 讀取 xml 格式檔案
'''
參考連結:
[1] pandas.read_xml
https://pandas.pydata.org/docs/reference/api/pandas.read_xml.html
'''
df = pd.read_xml("./jobs.xml"); df
```

```
# 寫入 xml 檔案
'''
參考連結:
[1] pandas.DataFrame.to_xml
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_xml.
html
'''
df.to_xml("./output.xml")
```

讀寫 excel

- 語法:
  - pd.read_excel(): 讀取 excel 檔

- df.to_excel(): 寫入 excel 檔

- 記得先安裝套件: `pip install openpyxl`

```
# 讀取 excel 格式檔案
'''
參考連結:
[1] openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files
https://openpyxl.readthedocs.io/en/stable/
[2] pandas.read_excel
https://pandas.pydata.org/docs/reference/api/pandas.read_excel.html
'''
df = pd.read_excel("./files/通訊錄.xlsx"); df
```

```
# 寫入 excel 檔案
'''
參考連結:
[1] pandas.DataFrame.to_excel
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_excel.html
'''
df.to_excel("./output.xlsx", index=False)
```

參考資料

[1] Café Nomad 咖啡廳遊牧民族
https://cafenomad.tw/
[2] 臺北市資料大平臺「垃圾車點位路線資訊
https://data.taipei/dataset/detail?id=6bb3304b-4f46-4bb0-8cd1-60c66dcd1cae
[3] 臺北市資料大平臺「臺北市政府求職徵才職缺資訊」
https://tika.apache.org/

Module 4. Matplotlib

Matplotlib 是一個 Python 繪圖套件，其主要特性如下：

- 多種繪圖風格：
  - Matplotlib 支持多種繪圖風格，包括折線圖、散點(佈)圖、長條圖、圓餅圖、直方圖等，能夠滿足不同的繪圖需求。

- 完整的繪圖功能：
  - Matplotlib 提供了完整的繪圖功能，包括圖形標籤、軸標籤、圖例、多子圖等，能夠方便地對圖形進行樣式設置和修改。
- 繪圖美學：
  - Matplotlib 具有豐富的繪圖美學，可以對圖形的顏色、線型、填充、字體等進行自定義設置，能夠繪製高質量的圖形。
- 良好的互動性：
  - Matplotlib 支持與 Jupyter Notebook 等交互式環境配合使用，能夠方便地對圖形進行互動式操作。
- 支持多種輸出格式：
  - Matplotlib 支持多種輸出格式，包括 PNG、PDF、SVG、EPS 等，方便用戶將繪製的圖形保存或用於出版。

Matplotlib

- 若有語法不了解的地方，可以參考以下連結:
  - API Reference
  - Matplotlib Pyplot
  - Matplotlib Pyplot

```
# 安裝 matplotlib
!pip install matplotlib
```

```
# 匯入套件(模組)
import matplotlib.pyplot as plt
```

Line 折線圖

4-1 Matplotlib 折線圖

Matplotlib

- 若有語法不了解的地方，可以參考以下連結:
  - API Reference
  - Matplotlib Pyplot
  - Matplotlib Pyplot

```
# 安裝 matplotlib
!pip install matplotlib
```

```
# 匯入套件(模組)
import matplotlib.pyplot as plt
```

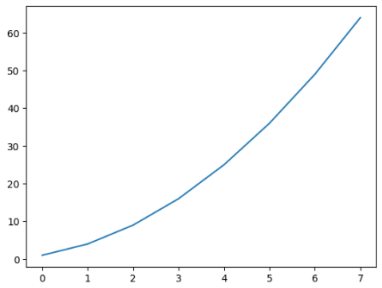
```

```

畫線

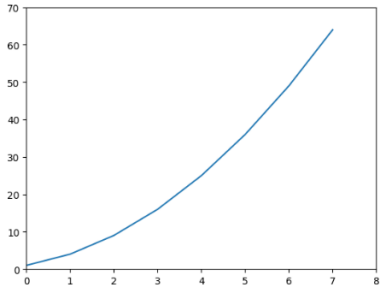
- 參考網頁: matplotlib.pyplot.plot

```
# 準備 y 軸的值
y = [x**2 for x in range(1, 9)]

# 準備 x 軸的值
x = [x for x in range(1, 9)]
```

```
# 繪製線條 (回傳 matplotlib.lines.Line2D 物件)
plt.plot(y)

# 顯示圖片
plt.show()
```

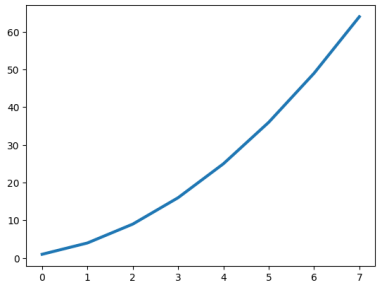



```
# 將 x 軸範圍設定成 0 到 8，y 軸設定成 0 到 70
plt.plot(y)

# 設定 x 跟 y 值的刻度
'''
plt.axis([xmin, xmax, ymin, ymax])
'''
plt.axis([0, 8, 0, 70])

# 顯示圖片
plt.show()
```



線條寬度 linewidth

```
# 設定線寬為 3
plt.plot(y, linewidth=3)
plt.show()
```

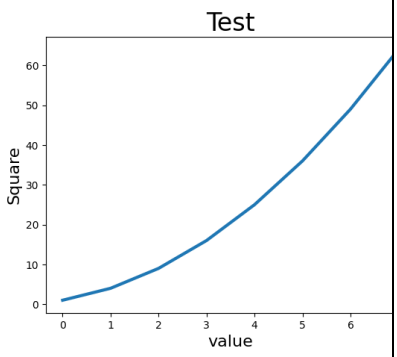


顯示標題

- title(): 圖表標題
- xlabel(): x 軸標題

- ylabel(): y 軸標題
- 引數可以用 `fontsize` 來改變標題的字型大小

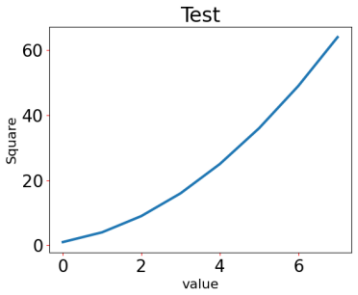
```
# 使用字型大小 24 與 16，分別為圖表與 x, y 軸建立標題
'''
fontsize 沒設定，預設 12
plt.title("Test")
plt.xlabel("value")
plt.ylabel("Square")
'''
plt.plot(y, linewidth=3)
plt.title("Test", fontsize=24)
plt.xlabel("value", fontsize=16)
plt.ylabel("Square", fontsize=16)
plt.show()
```



座標軸刻度設定

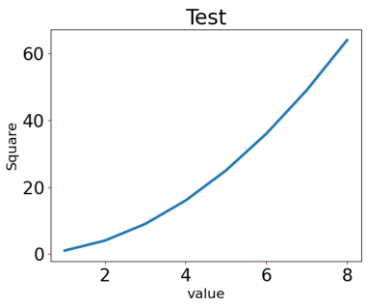
- tick_params(axis='both', labelsize=12, color='red')
  - axis: 'both'(x,y 都適用), 'x' (僅適用 x), 'y' (僅適用 y)
  - labelsize: 刻度大小
  - color: 刻度顏色，例如 'r' (紅色)
  - colors: 刻度與標題的顏色
- 參考網頁: matplotlib.pyplot.tick_params

```
# 使用不同座標軸刻度大小與顏色
plt.plot(y, linewidth=3)
plt.title("Test", fontsize=24)
plt.xlabel("value", fontsize=16)
plt.ylabel("Square", fontsize=16)
plt.tick_params(axis='both', labelsize=20, color='r') # color 成改 colors 會如何?
plt.show()
```



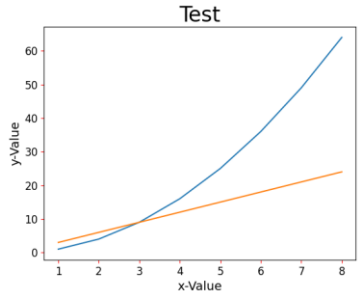
修改圖表的起始值

```
# x 軸標計從 1 開始
plt.plot(x, y, linewidth=3)
plt.title("Test", fontsize=24)
plt.xlabel("value", fontsize=16)
plt.ylabel("Square", fontsize=16)

# color 成改 colors 會如何?
plt.tick_params(axis='both', labelsize=20,
color='r')

plt.show()
```



多組數據的應用

```
# 設計多組數據的應用
'''
plt.plot(x, y_1, x, y_2, ...)
'''
y_1 = [x ** 2 for x in range(1, 9)]
y_2 = [x * 3 for x in range(1, 9)]
x = [x for x in range(1, 9)]
plt.plot(x, y_1, x, y_2)
plt.title("Test", fontsize=24)
plt.xlabel("x-Value", fontsize=14)
plt.ylabel("y-Value", fontsize=14)
plt.tick_params(axis='both', labelsize=12,
color='r')
plt.show()
```

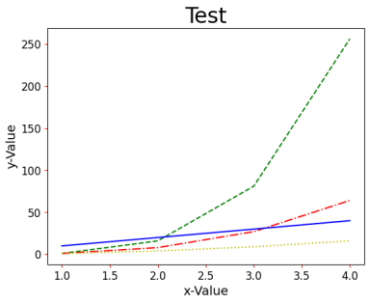


線條色彩與樣式

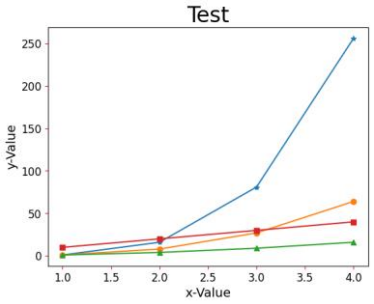
色彩字元	樣式字元
'b': blue (藍色)	'-' 或 'solid': 實線 (預設值)
'c': cyan (青色)	'--' 或 'dashed': 虛線
'g': green (綠色)	'-.' 或 'dashdot': 虛點線
'k': black (黑色)	':' 或 'dotted': 點線
'm': magenta (品紅色)	'.': 點
'r': red (紅色)	',': 像素
'w': white (白色)	'o': 圓

'y': yellow (黃色)	'v': 反三角形 '^': 三角形 '<': 左三角形 '>': 右三角形 's': 方形 'p': 五角 '*': 星號 '+': 加號 '-': 減號 'x': X 標記 'H': 六邊形 1 'h': 六邊形 2

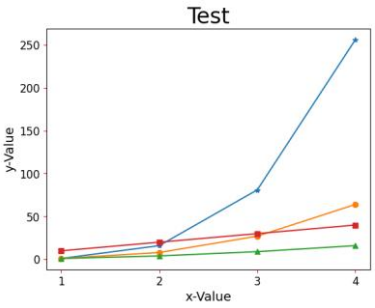
```
# 用不同顏色與線條樣式繪製圖表
y_1 = [x**4 for x in range(1, 5)]
y_2 = [x**3 for x in range(1, 5)]
y_3 = [x**2 for x in range(1, 5)]
y_4 = [x*10 for x in range(1, 5)]
x = [x for x in range(1, 5)]

# 色彩字元和樣式字元可以一起用
plt.plot(
    x, y_1, 'g--',
    x, y_2, 'r-.',
    x, y_3, 'y:',
    x, y_4, 'b-'
)

plt.title("Test", fontsize=24)
plt.xlabel("x-Value", fontsize=14)
plt.ylabel("y-Value", fontsize=14)
plt.tick_params(axis='both', labelsize=12,
color='r')
plt.show()
```



```
# 承上，額外在線條加入對應 y 值的標記
plt.plot(
    x, y_1, '-*',
    x, y_2, '-o',
    x, y_3, '-^',
    x, y_4, '-s'
)

plt.title("Test", fontsize=24)
plt.xlabel("x-Value", fontsize=14)
plt.ylabel("y-Value", fontsize=14)
plt.tick_params(axis='both', labelsize=12,
color='r')
plt.show()
```




```
# 承上，直接限定刻度範圍
plt.xticks(x)

plt.plot(
    x, y_1, '-*',
    x, y_2, '-o',
    x, y_3, '-^',
    x, y_4, '-s'
)

plt.title("Test", fontsize=24)
plt.xlabel("x-Value", fontsize=14)
plt.ylabel("y-Value", fontsize=14)
plt.tick_params(axis='both', labelsize=12,
color='r')
plt.show()
```

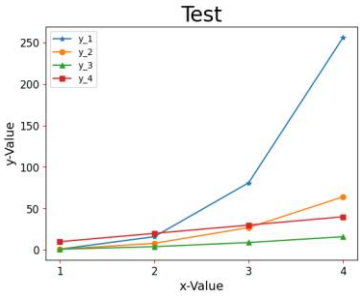


圖例 legend()

- 引數 loc:
- 'best': 0,
- 'upper right': 1,
- 'upper left': 2,
- 'lower left': 3,

- 'lower right': 4,
- 'right': 5 (等同 'center right'),
- 'center left': 6,
- 'center right': 7,
- 'lower center': 8,
- 'upper center': 9,
- 'center': 10

```
# 限定刻度範例
plt.xticks(x)

# plot 要分開來寫
plt.plot(x, y_1, '-*', label='y_1')
plt.plot(x, y_2, '-o', label='y_2')
plt.plot(x, y_3, '-^', label='y_3')
plt.plot(x, y_4, '-s', label='y_4')

# 設定圖例位置
plt.legend(loc='best') # 等同於設定 0

plt.title("Test", fontsize=24)
plt.xlabel("x-Value", fontsize=14)
plt.ylabel("y-Value", fontsize=14)
plt.tick_params(axis='both', labelsize=12,
color='r')
plt.show()
```

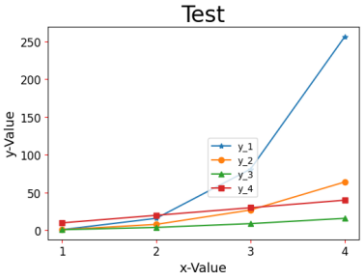


將圖例放在圖表外側

plt.legend(loc='upper left', `bbox_to_anchor(x0, y0, width, height)`):

- 圖表的左下角是 (0, 0)，右上角是 (1, 1)。
- 這裡的 loc='upper center'，是指 anchor 在 legend 的位置。
- loc='`upper left`' 在這裡指的是 `anchor 在 legend 的左上角`，如果 anchor 在 (1, 1)，代表 legend 在 anchor 的右下角，legend 在圖表 (1,1) 座標偏右下一點的位置。

```
# 限定刻度範例
plt.xticks(x)

# plot 要分開來寫
plt.plot(x, y_1, '-*', label='y_1')
plt.plot(x, y_2, '-o', label='y_2')
plt.plot(x, y_3, '-^', label='y_3')
plt.plot(x, y_4, '-s', label='y_4')

# 設定圖例位置
'''
這裡的 loc='upper left'，是指 anchor 在 legend 左上的位置，
因為 anchor 在 (0.5, 0.5) 的位置，所以 legend 是在圖表中間偏
右下的地方。
'''
plt.legend(loc='upper left', bbox_to_anchor=(0.5, 0.5))

# 增加圖表 padding
'''
如果 legend 消失不見，可以設定圖表 padding，讓 legend 有顯示
的空間，
通常用在 Terminal 執行時，圖表會顯示在一個視窗上，
視窗可能因為不夠寬、不夠高，造成 legend 無法正常顯示，
這時候可以嘗試設定 tight_layout()

參考網頁:
https://matplotlib.org/stable/api/_as_gen/matplotlib.pypl
ot.tight_layout.html
'''
plt.tight_layout(pad=5)

plt.title("Test", fontsize=24)
plt.xlabel("x-Value", fontsize=14)
plt.ylabel("y-Value", fontsize=14)
plt.tick_params(axis='both', labelsize=12, color='r')

# 儲存圖片
```



```
'''
plt.savefig('./output1.jpg', bbox_inches='tight')
第一個引數是儲存檔案的路徑，第二個引數 bbox_inches='tight'
會把先前的 padding 空白處移除掉。

參考網頁:
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.savefig.html
'''
plt.savefig('./output1.jpg', bbox_inches='tight')

plt.show()
```

Scatter 散點圖(散佈圖)

4-2 Matplotlib 散點圖

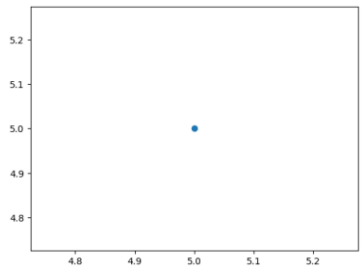
```
# 匯入套件(模組)
import matplotlib.pyplot as plt
import numpy as np
```

繪製散點圖 scatter()

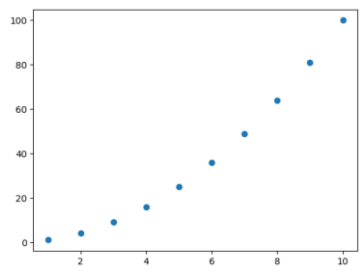
若有語法不了解的地方，可以參考以下連結:

- matplotlib.pyplot.scatter

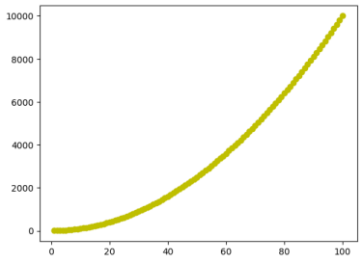
```
# 在座標 (5, 5) 繪製一個點
plt.scatter(5, 5)
plt.show()
```



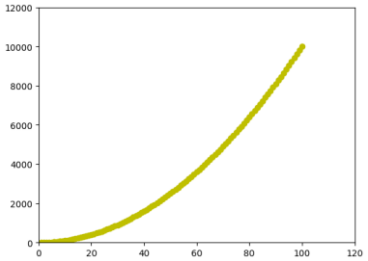

```
# 繪製連續的點
x_points = [x for x in range(1, 11)]
y_points = [x**2 for x in range(1, 11)]
plt.scatter(x_points, y_points)
plt.show()
```



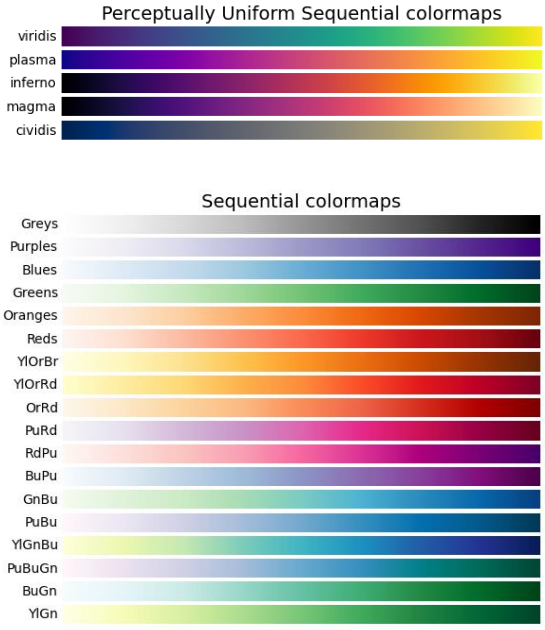
```python
# 繪製連續的黃色點，共 100 個點，x 軸由
range(1, 101) 產生，相對應的 y 軸是 x 的平方
x_points = list(range(1, 101))
y_points = [x**2 for x in x_points]
plt.scatter(x_points, y_points, color='y')
plt.show()
```

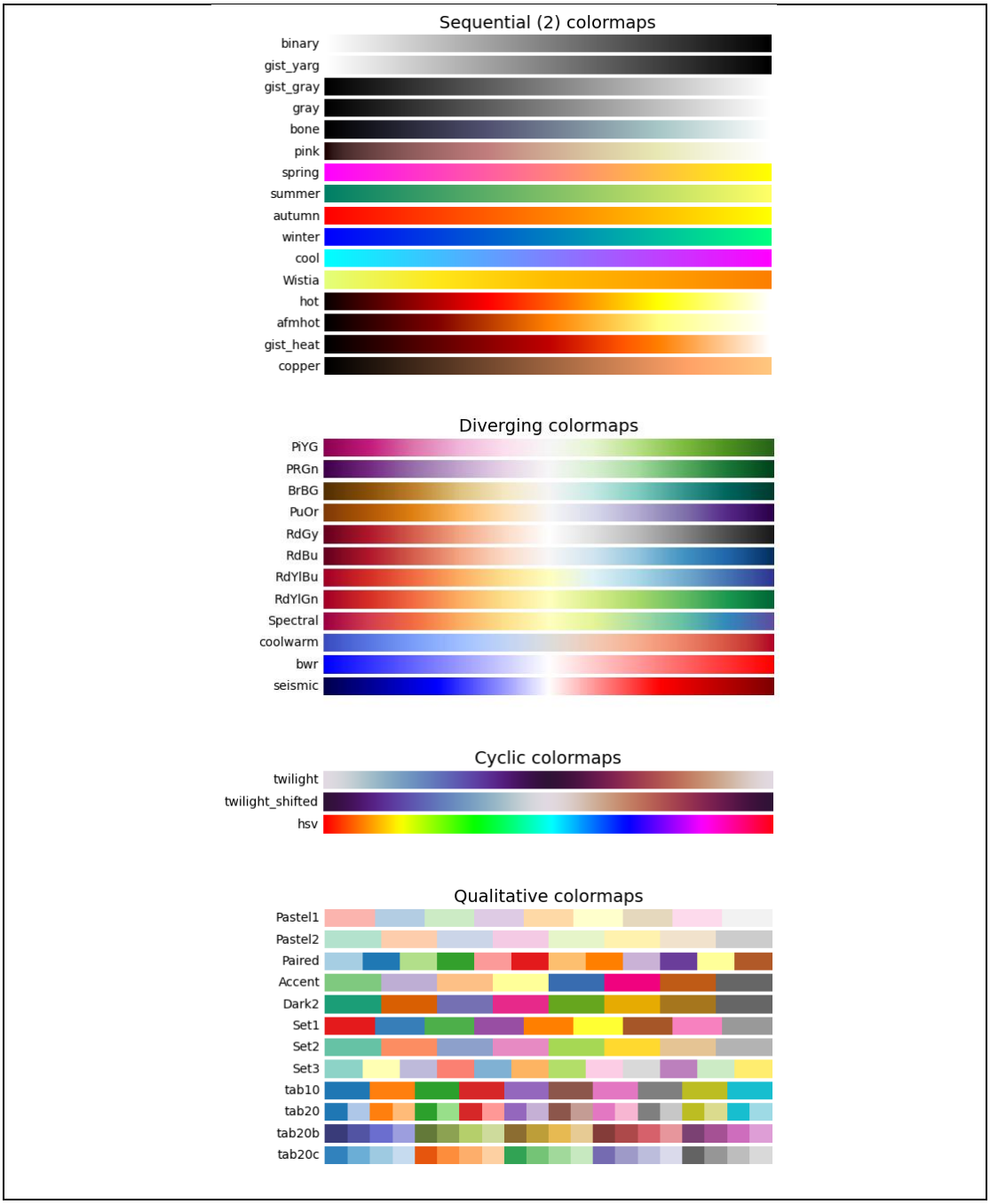


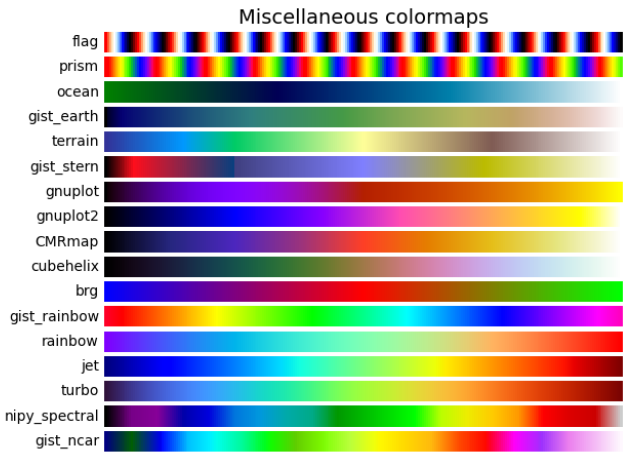

```python
# 設定 x 跟 y 軸的區間
'''
plt.axis([xmin, xmax, ymin, ymax])
'''
x_points = list(range(1, 101))
y_points = [x**2 for x in x_points]
plt.axis([0, 120, 0, 12000])
plt.scatter(x_points, y_points, color='y')
plt.show()
```



色彩映射



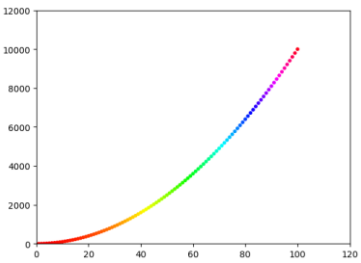

Sequential (2) colormaps
binary
gist_yarg
gist_gray
gray
bone
pink
spring
summer
autumn
winter
cool
Wistia
hot
afmhot
gist_heat
copper
Diverging colormaps
PiYG
PRGn
BrBG
PuOr
RdGy
RdBu
RdYlBu
RdYlGn
Spectral
coolwarm
bwr
seismic
Cyclic colormaps
twilight
twilight_shifted
hsv
Qualitative colormaps
Pastel1
Pastel2
Paired
Accent
Dark2
Set1
Set2
Set3
tab10
tab20
tab20b
tab20c



參考網頁

- Colormap reference
- Choosing Colormaps in Matplotlib
- matplotlib.pyplot.scatter

```
# 繪製散點圖時，產生色彩映射的效果
'''
plt.scatter(x_points, y_points, s=50,
c=x_points, cmap='hsv')
引數
- s: 散點大小
- c: x_points(根據 x 軸的值增減作變化) 或
y_points (根據 y 軸的值增減作變化)
- cmap: (請查閱 Choosing Colormaps in
Matplotlib)
'''
x_points = list(range(1, 101))
y_points = [x**2 for x in x_points]
plt.axis([0, 120, 0, 12000])
plt.scatter(x_points, y_points, s=10,
c=y_points, cmap='hsv')
plt.show()
```

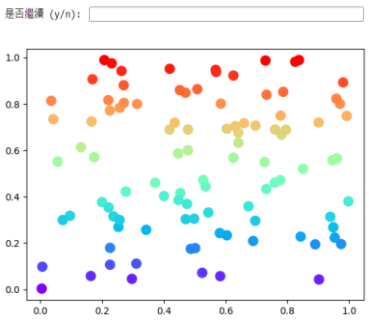



```
# 隨機數的應用
'''
傳回 N 個 [0.0, 1.0) 之間的數字
np.random.random(N)
```

```
array([0.0126898 ,
0.6636076 ,
0.10664936,
0.11458897,
0.26218278])
```

```
參考網頁:
https://numpy.org/doc/stable/reference/random/generated/numpy.random.random.html
'''
np.random.random(5)
```

```
# 隨機數整合 scatter 應用
while True:
    # 隨機生成 x 和 y 的資料
    x = np.random.random(100)
    y = np.random.random(100)

    # 建立散點圖
    plt.scatter(x, y, s=100, c=y,
cmap='rainbow')
    plt.show()

    # 互動
    ans = input('是否繼續 (y/n): ')
    if ans == 'n' or ans == 'N':
        break
```



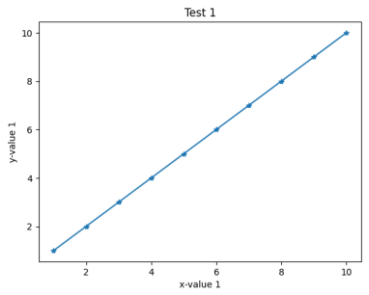
多個圖表

4-3 Matplotlib 多個圖表

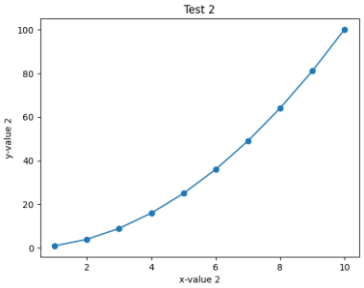
```
# 匯入套件(模組)
import matplotlib.pyplot as plt
import numpy as np
```

一個程式有多個圖表

```
# 一次顯示兩張圖表
'''
先前我們一直使用 [x for x in range(1, 11)] 來建立 list，這次使用 np.arange(1, 11) 來產生陣列
'''
y_1 = np.arange(1, 11)
y_2 = np.arange(1, 11) ** 2
x = np.arange(1, 11)
```



```
# 設定第 1 張圖的編號
plt.figure(1)
plt.plot(x, y_1, '-*')

# 這裡是設定第 1 張圖的編號 (可以自行決定要不要加 fontsize)
plt.title('Test 1')
plt.xlabel('x-value 1')
plt.ylabel('y-value 1')

# 設定第 2 張圖的編號
plt.figure(2)
plt.plot(x, y_2, '-o')

# 這裡是設定第 2 張圖的編號 (可以自行決定要不要加 fontsize)
plt.title('Test 2')
plt.xlabel('x-value 2')
plt.ylabel('y-value 2')

# 顯示圖表
plt.show()
```

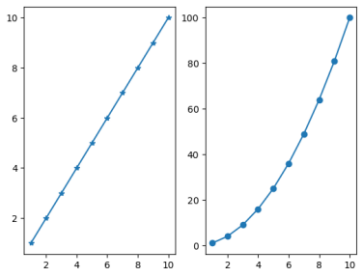


含有子圖表的圖表

參考連結:

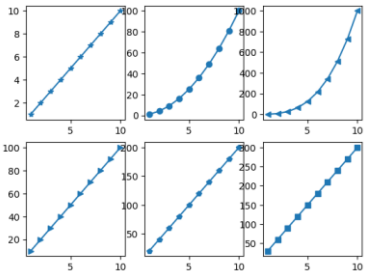
- matplotlib.pyplot.subplot
- Matplotlib 绘制多图

```
# 使用 plt.subplot(nrows, ncols, index)，繪製 1 列 2 欄的子圖表
'''
nrows = 1
ncols = 2
index = 1
代表一共 1 列 2 行(欄)的子圖表，目前是
figure(1)
'''
```



```
# 初始化資料
y_1 = np.arange(1, 11)
y_2 = np.arange(1, 11) ** 2
x = np.arange(1, 11)

# 設定第 1 張子圖表
plt.subplot(1,2,1)
plt.plot(x, y_1, '-*')

# 設定第 2 張子圖表
plt.subplot(1,2,2)
plt.plot(x, y_2, '-o')

# 顯示圖表
plt.show()
```

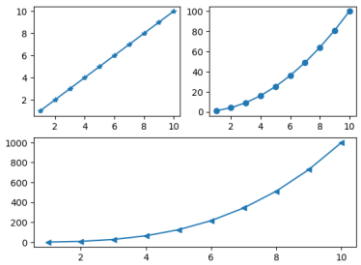
```
# 使用 plt.subplot(nrows, ncols, index)，繪製 2 列 3 欄的子圖表
'''
nrows = 2
ncols = 3
index = 1
代表一共 2 列 3 行(欄)的子圖表，目前是
figure(1)
'''
# 初始化資料
y_1 = np.arange(1, 11)
y_2 = np.arange(1, 11) ** 2
y_3 = np.arange(1, 11) ** 3
y_4 = np.arange(1, 11) * 10
y_5 = np.arange(1, 11) * 20
y_6 = np.arange(1, 11) * 30
x = np.arange(1, 11)

# 設定第 1 張子圖表
plt.subplot(2,3,1)
plt.plot(x, y_1, '-*')
```



```
# 設定第 2 張子圖表
plt.subplot(2,3,2)
plt.plot(x, y_2, '-o')

# 設定第 3 張子圖表
plt.subplot(2,3,3)
plt.plot(x, y_3, '-<')

# 設定第 4 張子圖表
plt.subplot(2,3,4)
plt.plot(x, y_4, '->')

# 設定第 5 張子圖表
plt.subplot(2,3,5)
plt.plot(x, y_5, '-p')

# 設定第 6 張子圖表
plt.subplot(2,3,6)
plt.plot(x, y_6, '-s')

# 顯示圖表
plt.show()
```

繪製上方一個子圖表、下方兩個子圖表的圖表

```
# 初始化
y_1 = np.arange(1, 11)
y_2 = np.arange(1, 11) ** 2
y_3 = np.arange(1, 11) ** 3
x = np.arange(1, 11)

# 設定第 1 張子圖表 (左上)
ax1 = plt.subplot(2,2,1)
ax1.plot(x, y_1, '-*')

# 設定第 2 張子圖表 (右上)
ax2 = plt.subplot(2,2,2)
ax2.plot(x, y_2, '-o')
```

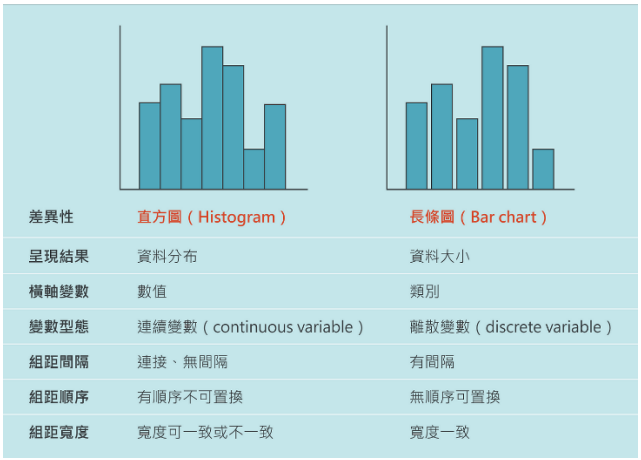


```
# 設定第 3 張子圖表 (正下，因為跨兩欄，所以編號為 2)
ax3 = plt.subplot(2,1,2)
ax3.plot(x, y_3, '-<')

# 顯示圖表
plt.show()
```

Bar 長條圖、Hist 直方圖 和 Pie 圓餅圖

4-4 Matplotlib 長條圖、直方圖、圓餅圖

```
# 匯入套件(模組)
import matplotlib.pyplot as plt
import numpy as np
```



差異性	直方圖（Histogram）	長條圖（Bar chart）
呈現結果	資料分布	資料大小
橫軸變數	數值	類別
變數型態	連續變數（continuous variable）	離散變數（discrete variable）
組距間隔	連接、無間隔	有間隔
組距順序	有順序不可置換	無順序可置換
組距寬度	寬度可一致或不一致	寬度一致

參考資料:

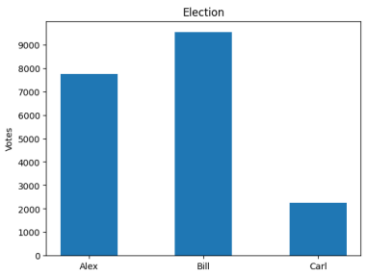
- Python 學習筆記：Matplotlib 資料視覺化（二）統計圖
- 如何分辨長相近似的孿生兄弟-直方圖（Histogram）與長條圖（Bar chart）之差異
- Examples

長條圖 bar()

- 通常用於「類別型」資料 (categorical data)
  - 資料之間是離散的 (discrete)、間斷的
- plt.bar(x, height, width=0.8, ...)
  - `x`: array-like 的資料，例如 list、array、tuple 等，作為 x 軸的標題
  - `height`: array-like 的資料，作為 y 軸的值
  - `width`: bar 的寬度

- matplotlib.pyplot.xticks(ticks=None, labels=None, ...)
  - `ticks`: x 軸的刻度
  - `label`: 對應 x 軸的標題，類似取代 x 軸的刻度，用其它名稱代替
- 參考連結:
  - matplotlib.pyplot.bar
  - matplotlib.pyplot.xticks
  - Python matplotlib 畫圖 – 柱狀圖 (長條圖)

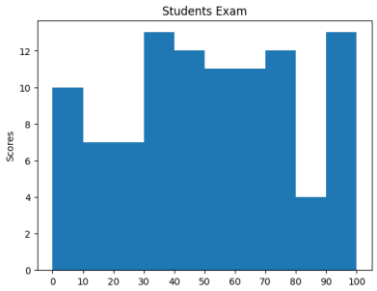
```
'''
班上投票，Alex 得票 7749，Bill 得票 9527，
Carl 得票 2266，請用長條圖表示
'''

# 得票數 (因為有 3 名候選人)
votes = [7749, 9527, 2266]
candidates = ['Alex', 'Bill', 'Carl']

# x 軸的刻度，例如 [0, 1, 2]
x_index = np.arange(len(votes))

# bar 的寬度
width = 0.5

# 繪製長條圖
plt.bar(x_index, votes, width)

# 設定圖表標題
plt.title('Election')
plt.ylabel('Votes')

# 設定 x 和 y 軸的刻度
plt.xticks(x_index, candidates)
plt.yticks(np.arange(0, 10000, 1000))

# 顯示圖表
plt.show()
```



直方圖 hist()

- 通常用於「數值型」資料 (numeric data)
  - 資料之間是連續的 (continueous)
- plt.hist(x, bins, ...)
  - x: 你的資料，list、array 或是 sequece
  - bin: 組距，預設 10
- 參考資料:
  - matplotlib.pyplot.hist
  - Python 學習筆記：Matplotlib 資料視覺化 (二) 統計圖

```
'''
請以班上 100 個學生考試成績，最低 0 分，最高
100 分，繪製直方圖
'''

# 隨機產生 100 位同學的成績 (用
np.random.randint 的話，要轉成 list)
x_scores = list(np.random.randint(0, 100,
size=(1, 100), dtype='int32'))

# 組距
x_bins = np.arange(0, 101, 10)

# 繪製直方圖
plt.hist(x_scores, bins=x_bins)

# 設定標題
plt.title('Students Exam')
plt.xticks(x_bins, x_bins)
plt.ylabel('Scores')

# 顯示圖表
plt.show()
```



圓餅圖 pie()

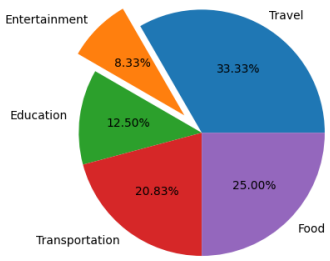
- 用於描述「類別型」資料
- plt.pie(x, explode=None, labels=None, colors=None, autopct=None,shadow=False, labeldistance=1.1, radius=1, counterclock=True, center=(0, 0), ...)
  - x: 你的資料。

- explode: 餅分離的程度，list 格式，其中的每個元素值介於 0 到 1 之間。
- labels: 圓餅圖的標題，list 格式。
- autopct: 每一個項目的百分比格式，例如「%2.2f%%」，代表整數 2 位數，小數 2 位數。
- labeldistance: 項目標題與圓餅圖中間距離是半徑的多少倍。
- center: 圓中心座標，預設 (0, 0)。
- shadow: 圓餅圖是否有陰影，預設為 False。

■ 參考資料:

- matplotlib.pyplot.pie

```
'''
有一家庭支出費用如下，以此設計圓餅圖:
旅行 (Travel): 8000
娛樂 (Entertainment): 2000
教育 (Education): 3000
交通 (Transportation): 5000
餐費 (Food): 6000
'''

# 設定支出項目
items = ['Travel', 'Entertainment',
'Education', 'Transportation', 'Food']

# 設定支出費用
expenses = [8000, 2000, 3000, 5000, 6000]

# 繪製圓餅圖
plt.pie(
    expenses,
    labels=items,
    explode=(0,0.2,0,0,0),
    autopct="%1.2f%%"
)

# 顯示圖表
plt.show()
```

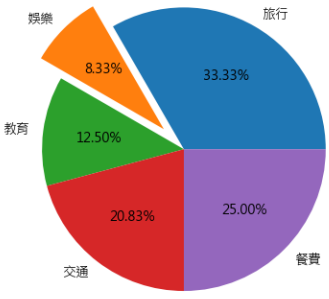


補充: 確認電腦內部字型

```
'''
補充：確認電腦內部字型
'''
from matplotlib.font_manager import
fontManager
for i in
sorted(fontManager.get_font_names()):
    print(i)
```

```
?????(P)
Agency FB
Algerian
Arial
Arial Nova
Centaur
Century
...
MS Gothic
M
Maiandra GD
Malgun Gothic
Matura MT Script Capitals
Microsoft Himalaya
Microsoft JhengHei
Microsoft New Tai Lue
Microsoft PhagsPa
Microsoft Sans Serif
Microsoft Tai Le
Microsoft YaHei
Microsoft Yi Baiti
...
Noto Sans TC
OCR A Extended
Old English Text MT
Times New Roman
Trebuchet MS
Tw Cen MT
Tw Cen MT Condensed
Tw Cen MT Condensed Extra Bold
Verdana
...
```

```
'''
承上，將標題改成中文來顯示
'''

# 設定字型
import matplotlib
matplotlib.rc('font', family='Microsoft
JhengHei')

# 設定支出項目
items = ['旅行', '娛樂', '教育', '交通', '餐
費']

# 設定支出費用
expenses = [8000, 2000, 3000, 5000, 6000]

# 繪製圓餅圖
plt.pie(
    expenses,
    labels=items,
    explode=(0,0.2,0,0,0),
    autopct="%1.2f%%"
)
```



```
# 顯示圖表
plt.show()
```

參考資料

[1] Python 學習筆記 : Matplotlib 資料視覺化 (二) 統計圖
http://yhhuang1966.blogspot.com/2020/05/python-matplotlib.html
[2] 如何分辨長相近似的孿生兄弟-直方圖（Histogram）與長條圖（Bar chart）之差異
https://medium.com/marketingdatascience/如何分辨長相近似的孿生兄弟-直方圖-histogram-與長條圖-bar-chart-之差異-154602ac0ba6
[3] matplotlib - Examples
https://matplotlib.org/stable/gallery/
[4] matplotlib.pyplot.bar
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html
[5] matplotlib.pyplot.xticks
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.xticks.html
[6] Python matplotlib 畫圖 – 柱狀圖 (長條圖)
https://python-ecw.com/2020/11/12/matplotlib 柱狀圖/
[7] matplotlib.pyplot.hist
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html
[8] Python 學習筆記 : Matplotlib 資料視覺化 (二) 統計圖
http://yhhuang1966.blogspot.com/2020/05/python-matplotlib.html
[9] matplotlib.pyplot.pie
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.pie.html#matplotlib.pyplot.pie