目次

Module 1. Python 與資料分析應用 1
為什麼要使用 Python 進行資料分析 1
資料分析步驟說明 1
Jupyter Notebook 環境設定 2
參考資料 10
Module 2. 資料處理 11
Numpy 11
Pandas 28
參考資料 39
Module 3. 不同資料型態存取 40
認識資料型態-結構、半結構與非結構化資料 40
處理 CSV,JSON,XML,Excel 格式資料 41
參考資料 44
Module 4. Matplotlib 45
Line 折線圖 45
Scatter 散點圖(散佈圖) 53
多個圖表 57
Bar 長條圖、Hist 直方圖 和 Pie 圓餅圖 61
Module 5. 資料庫基礎應用 74
MySQL 環境建置 74
資料庫設計議題 86
開啟 MySQL Workbench 91
建立資料庫、資料表 93
新增、修改、刪除資料 99
查詢資料 101
子查詢 106
函式 107
完整性原則與關聯介紹 109
Pandas 與 SQL 用法對照表 112

- GitHub 專案連結
  https://github.com/telunyang/python_data_analysis

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、機率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=3.11 ipykernel notebook

進入 conda 環境
conda activate da

說明：原先 ipykernel 在網頁的操作環境 (Jupyter Notebook)，換成 Visual Studio Code 裡面執行。

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 jupyter 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。

```
In [1]: !python -V
        Python 3.10.10
```

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，m 列 n 行
ndarray.size	Array 元素的總個數，相當於 .shape 中 m*n 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	整數（-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 (half-precision floating-point)，包括：1 個符號位，5 個指數位，10 個尾數位
float32	單精度浮點數 (single-precision floating-point)，包括：1 個符號位，8 個指數位，23 個尾數位
float64	雙精度浮點數 (double-precision floating-point)，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(建立)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 a = np.array(33) print(a + a) ```	66

``
# 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 a，
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:
arr = [
    [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: ```	(2, 3)

- 有 2 列，每 1 列有 3 個元素， - 也可看成幾個 row、幾個 column ''' a.shape	
# ndim (n-dimensions，維度) a.ndim	2
# 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)	dtype('float64')

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

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

# 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.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)	array([[10, 10, 10, 10], [10, 10, 10, 10], [10, 10, 10, 10]])

''' a = np.full((3,4), 10); a	

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 	
a = x - y; a
	array([-8, -4, 0, 4, 8])

# 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 '''	16

```
numpy.sum(a, axis=None,
dtype=None, ...)
'''
c = np.array([1, 3, 5, 7])
d = np.sum(c); d
```

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, /)
'''
a = np.array([4,5,1,3,4])
b = np.array([5,3,5,4,2])
c = np.inner(a, b); c
```

```
60
```

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 [5,3,5,4,2]，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`	`array([[ 6  8],` `  [10 12]])`

`# 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`
`'''`
`a = x * y; a`
	`array([[ 5, 12],`
	`  [21, 32]])`

`# 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]]`
	`array([1, 2])`

以下方式同樣效果 x[0,] x[0,:] ''' x[0]	
# 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)	array([[1, 2, 3], [4, 5, 6]])

# 註：「不會」修改原始陣列，「會」回傳轉換後的新陣列 	
y1 = x1.reshape(2, 3); y1	

# 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 # 註：「會」修改原始陣列，「不會」回傳轉換後的新陣列 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:
[[0, 1],
[2, 3],
[4, 5],
[6, 7]]
'''
b = a.transpose(); b
	array([[0, 2, 4, 6],
	[1, 3, 5, 7]])

# 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]])

點積 dot()	
# 如果是一維，計算結果等同於 inner product # = np.inner(a, b) # = a[0]*b[0] + a[1]*b[1] + a[2]*b[2] + a[3]*b[3] + a[4]*b[4] + a[5]*b[5] + a[6]*b[6] + a[7]*b[7] # = 1*2 + 2*4 + 3*6 + 4*8 + 5*10 + 6*12 + 7*14 + 8*16 # = 2 + 8 + 18 + 32 + 50 + 72 + 98 + 128 # = 408	408

a = np.array([1, 2, 3, 4, 5, 6, 7, 8]) b = np.array([2, 4, 6, 8, 10, 12, 14, 16]) c = np.dot(a, b); c	

# 如果是二維，計算結果等同於 matrix multiplication 	
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6], [7, 8]])
c = np.dot(a, b); c
	array([[19, 22],
	[43, 50]])

矩陣相乘 matmul()	

# 典型的矩陣乘法 	
a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
b = np.array([[10, 11], [12, 13], [14, 15]])
c = np.matmul(a, b); c
	array([[76, 82],
	[184, 199],
	[292, 316]])

對應矩陣相乘 multiply()	

# 如果是一維，計算結果等同於 element-wise product 	
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = np.multiply(a, b); c
	array([4, 10, 18])

# 如果是二維，計算結果等同於 element-wise product 	
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6], [7, 8]])
c = np.multiply(a, b); c
	array([[5, 12],
	[21, 32]])

【一些遇到就懵的概念】point-wise, pair-wise, element-wise 是什么

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 簡單運算	
陣列簡單加總	
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

捨去函數	
# around(): 四捨五入到最近的「偶數值」 ''' numpy.around(a, decimals=0, out=None)	array([0.4, 1.8])

decimals: 指定小數位數 (小數點後面第幾位) ''' np.around([0.45, 1.85], 1)	

# 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.])

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) '''	array([0.37454012, 0.95071431, 0.73199394])

np.random.seed(42) a = np.random.rand(3); a	
''' np.random.randint(start, end, size=數量): 產生 start 到不包含 end 之間的整數值 隨機產生 100 位同學的成績 ''' a = np.random.randint(0, 101, size=100); 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])

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 或 Excel 當中的表格，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 	
myIndex = [3, 6, 9] # 也可以用字串當 index (key)
price = [10, 20, 30]
s = pd.Series(price, index=myIndex); s
	3 10
	6 20
	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([11, 22, 33, 44, 55, 66])	

s[3]	44
s[2:4]	2 33 3 44 dtype: int64
s[:3]	0 11 1 22 2 33 dtype: int64
s[2:]	2 33

	3 44 4 55 5 66 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 66 dtype: int64

# series 物件相加 	
x = pd.Series([1, 2])
y = pd.Series([3, 4])
x + y
	0 4
	1 6
	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)
	apple 35.0
	banana NaN
	orange NaN
	pear 95.0
	dtype: float64

x1 + x2	

# 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

# 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

```python
# 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

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。

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

# 自訂索引
years = range(2020, 2024)

# 臺北、臺中、高雄 某個月的平均溫度
```

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

<table>
<tr><td>

```python
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
```

</td><td></td></tr>
<tr><td>

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

</td><td>

```
20
```

</td></tr>
<tr><td>

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

</td><td>

```
32
```

</td></tr>
<tr><td>

```python
# 使用 iat: 取得 row 是 2, column 是 1 的值
'''
index 是 zero-based
'''
df.iat[2, 1]
```

</td><td>

```
27
```

</td></tr>
<tr><td>

```python
# 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]
```

</td><td>

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

</td></tr>
<tr><td>

```python
# 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] ]
```

</td><td>

	taipei	taichung	kaohsiung
2020	20	25	30
2023	21	28	32

</td></tr>
</table>

```
# 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

```python
# 取得 taipei 溫度大於 20 的資料
'''
類似在 df 的 taipei 欄位當中，
將每一個值進行邏輯判斷，
符合條件的整筆資料，
才隨同 df 回傳回來
'''
df[ df['taipei'] > 20 ]
```

	taipei	taichung	kaohsiung
2021	21	26	29
2023	21	28	32

```python
# 取得 taipei 溫度大於 20、高雄溫度大於 30 的資料
'''
多個判斷語法，要用括號包起來

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

	taipei	taichung	kaohsiung
2023	21	28	32

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.**isnull()**: 跟 df.**isna()** 一樣。

```python
# 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

```python
# 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，
```

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

<table>
<tr><td>

```
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()
```

</td><td></td></tr>
<tr><td>

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

</td><td>

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

</td></tr>
<tr><td>

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

</td><td>

	0	1	2
0	1	2.0	3.0

</td></tr>
<tr><td>

```
# dropna(axis=1): 刪除含 NaN 的 column
df_ = df.dropna(axis=1); df_
```

</td><td>

	0
0	1
1	4
2	7

</td></tr>
</table>

<table>
<tr><td colspan="2">

簡單的統計函數

- **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。

</td></tr>
<tr><td>

```
# 初始化
from random import randint

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

# 5 個學生的各科分數 (隨機產生)
```

</td><td>

	國文	英文	數學	自然	社會	其它
1	88	68	100	81	92	78
2	85	69	60	72	65	94
3	66	76	75	74	61	87
4	90	66	61	79	80	83
5	85	83	97	62	65	85

</td></tr>
</table>

```python
s1 = [randint(60, 100) for x in range(6)]
s2 = [randint(60, 100) for x in range(6)]
s3 = [randint(60, 100) for x in range(6)]
s4 = [randint(60, 100) for x in range(6)]
s5 = [randint(60, 100) for x in range(6)]

grades = [
    s1, # 代表 s1 學生的 5 科成績
    s2, # 代表 s2 學生的 5 科成績
    s3, # 代表 s3 學生的 5 科成績
    s4, # 代表 s4 學生的 5 科成績
    s5  # 代表 s5 學生的 5 科成績
]

# 建立 dataframe
df = pd.DataFrame(
    grades,
    columns=courses,
    index=range(1,6)
); df
```

```python
# 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('其它', axis=1); df
```

	國文	英文	數學	自然	社會
1	88	68	100	81	92
2	85	69	60	72	65
3	66	76	75	74	61
4	90	66	61	79	80
5	85	83	97	62	65

```python
# 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	88	68	100	81	92	429
2	85	69	60	72	65	351
3	66	76	75	74	61	352
4	90	66	61	79	80	376
5	85	83	97	62	65	392

```python
# 列出各科平均分數 (包括總分的平均分數)
'''
df 的平均預設由上而下
df.mean(axis=0)
'''
```

```
國文      82.8
英文      72.4
數學      78.6
自然      73.6
社會      72.6
```

<table>
<tr><td>

```python
avg = df.mean(); avg
```

</td><td>

```
總分     380.0
dtype: float64
```

</td></tr>
<tr><td>

```python
# 增加 index: 在 df 下方增加平均分數
df.loc['平均分數'] = avg; df
```

</td><td>

	國文	英文	數學	自然	社會	總分
1	88.0	68.0	100.0	81.0	92.0	429.0
2	85.0	69.0	60.0	72.0	65.0	351.0
3	66.0	76.0	75.0	74.0	61.0	352.0
4	90.0	66.0	61.0	79.0	80.0	376.0
5	85.0	83.0	97.0	62.0	65.0	392.0
平均分數	82.8	72.4	78.6	73.6	72.6	380.0

</td></tr>
<tr><td>

```python
# 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
```

</td><td>

	國文	英文	數學	自然	社會	總分
1	88	68	100	81	92	429
2	85	69	60	72	65	351
3	66	76	75	74	61	352
4	90	66	61	79	80	376
5	85	83	97	62	65	392

</td></tr>
<tr><td>

```python
# 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
```

</td><td>

	國文	英文	數學	自然	社會	總分
1	88.0	68.0	100.0	81.0	92.0	429.0
5	85.0	83.0	97.0	62.0	65.0	392.0
4	90.0	66.0	61.0	79.0	80.0	376.0
3	66.0	76.0	75.0	74.0	61.0	352.0
2	85.0	69.0	60.0	72.0	65.0	351.0

</td></tr>
<tr><td>

```python
# 經過大到小的排序後，增加名次欄位
rank = range(1, 6)
df['名次'] = rank; df
```

</td><td>

	國文	英文	數學	自然	社會	總分	名次
1	88.0	68.0	100.0	81.0	92.0	429.0	1
5	85.0	83.0	97.0	62.0	65.0	392.0	2
4	90.0	66.0	61.0	79.0	80.0	376.0	3
3	66.0	76.0	75.0	74.0	61.0	352.0	4
2	85.0	69.0	60.0	72.0	65.0	351.0	5

</td></tr>
<tr><td>

```python
# 依 index 從新排序
df = df.sort_index(ascending=True); df
```

</td><td>

	國文	英文	數學	自然	社會	總分	名次
1	88.0	68.0	100.0	81.0	92.0	429.0	1
2	85.0	69.0	60.0	72.0	65.0	351.0	5
3	66.0	76.0	75.0	74.0	61.0	352.0	4
4	90.0	66.0	61.0	79.0	80.0	376.0	3
5	85.0	83.0	97.0	62.0	65.0	392.0	2

</td></tr>
</table>

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，每行代表一筆資料記錄。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="big5"); 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="big5")
```

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.ht
ml
'''
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_exce
l.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、線形、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

```python
# 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

```python
# 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

```python
# 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.pyplot.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()
```

```python
# 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)

```python
'''
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 軸文字的座標
x_ticks = [1, 2, 3]

# bar 的寬度
width = 0.5

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

# 設定數值文字在圖表上
for index in range(len(x_ticks)):
    # matplotlib.pyplot.text(圖表的 x 座標,
圖表的 y 座標, 顯示的文字)
    plt.text(
        x_ticks[index],
        votes[index],
        votes[index])

# 設定圖表標題
plt.title('Election')
```

```
plt.ylabel('Votes')

# 設定 x 和 y 軸的刻度
plt.xticks(x_ticks, candidates)
plt.yticks(np.arange(0, 10000, step=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, 101, size=100))

# 組距 (將成績 0 到 100 分，分成 10 個組距，回傳 11 個數字)
x_bins = np.arange(0, 101, 10)

# 繪製直方圖 (會回傳每個組距的數量，以及對應組距)
arr_counts, arr_bins, obj = plt.hist(
    x_scores,
    bins=x_bins
```

```
)

# 設定數值文字在圖表上
for i in range(len(arr_counts)):
    plt.text(
        arr_bins[i], # x 為 0 ~ 90 的座標上寫字
        arr_counts[i],
        arr_counts[i].astype('int32')
    )

# 設定標題
plt.title('Students Exam')
plt.xticks(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

```python
'''
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="%.2f%%"
)

# 顯示圖表
plt.show()
```

補充：確認電腦內部字型

```python
'''
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
...
```

```python
'''
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="%.2f%%"
)

# 顯示圖表
plt.show()
```

其它範例

```
'''Stacked Bar chart'''

# 匯入套件
import matplotlib.pyplot as plt
import numpy as np

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

# 建立資料
'''
y 用 np.array() 來產生數據，
是因為 plt.bar() 當中的 bottom 參數，
可以透過與上一個陣列相加來賦值
'''
x = ['A', 'B', 'C', 'D']
y1 = np.array([10, 20, 10, 30])
```

```
y2 = np.array([20, 25, 15, 25])
y3 = np.array([12, 15, 19, 6])
y4 = np.array([10, 29, 13, 19])

# 繪製 stacked bar
# (重點在 bottom 屬性; y1 和 y2 y3 可以相加，
是因為 numpy)
plt.bar(x, y1, color='r')
plt.bar(x, y2, bottom=y1, color='b')
plt.bar(x, y3, bottom=y1+y2, color='y')
plt.bar(x, y4, bottom=y1+y2+y3, color='g')

# 圖表標題
plt.xlabel("隊伍")
plt.ylabel("分數")
plt.legend(["Round 1", "Round 2", "Round
3", "Round 4"])
plt.title("每個隊伍在 4 輪中的分數")

# 顯示圖表
plt.show()
```

```
'''
Horizontal Bar chart
'''
# 匯入套件
import matplotlib.pyplot as plt

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

# 設定類別項目
y_tick_labels = [
    'Python',
    'Scala',
    'C#',
```

```
    'Java',
    'PHP'
]

# 設定類別項目的 x 軸座標
x = [1, 2, 3, 4, 5]

# 每個類別項目的值
y = [50, 10, 13, 27, 22]

# 繪製 Horizontal Bar
plt.barh(x, y, tick_label=y_tick_labels)

# 設定數值文字在圖表上
for i in range(len(x)):
    # plt.text() 依然使用 x, y text 的順序
    plt.text(y[i], x[i], y[i])

# 圖表標題
plt.ylabel("使用次數")
plt.xlabel("程式語言")

# 顯示圖表
plt.show()
```

```
'''
Grouped Bar chart
'''
# 匯入套件
import matplotlib.pyplot as plt

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

# 設定群組資料的值 (第 1 隊、第 2 隊)
y_score_1 = [35, 60, 18, 29, 77]
```

```
y_score_2 = [28, 33, 93, 44, 32]

# 設定 x 軸的順序，這裡要注意的是，group 的
bar chart，要另外指定 x 軸的位置
BAR_WIDTH = 0.4
x_axis_1 = [1, 2, 3, 4, 5]
x_axis_2 = [i + BAR_WIDTH for i in
x_axis_1]
'''
x_axis_2 = [
    1 + BAR_WIDTH,
    2 + BAR_WIDTH,
    3 + BAR_WIDTH,
    4 + BAR_WIDTH,
    5 + BAR_WIDTH
]
'''

# 設定 x 刻度的中心位置
ticks = [i + BAR_WIDTH / 2 for i in
x_axis_1]
'''
ticks = [
    1 + BAR_WIDTH / 2,
    2 + BAR_WIDTH / 2,
    3 + BAR_WIDTH / 2,
    4 + BAR_WIDTH / 2,
    5 + BAR_WIDTH / 2
]
'''

# 設定 x 標題
tick_labels = ["Round 1", "Round 2", "Round
3", "Round 4", "Round 5"]

# 繪製圖表
plt.bar(
```

```
    x_axis_1,
    y_score_1,
    BAR_WIDTH,
    color="y",
    label="第 1 隊"
)
plt.bar(
    x_axis_2,
    y_score_2,
    BAR_WIDTH,
    color="b",
    label="第 2 隊"
)

# 設定數值文字在圖表上
for i in range(len(x_axis_1)):
    plt.text(
        x_axis_1[i],
        y_score_1[i],
        y_score_1[i]
    )
    plt.text(
        x_axis_2[i],
        y_score_2[i],
        y_score_2[i]
    )

# 圖表標題
plt.xlabel("回合數")
plt.ylabel("分數")
plt.title("隊伍競賽分數")
plt.xticks(ticks, tick_labels)
plt.legend(loc='upper left', ncols=2)

# 顯示圖表
plt.show()
```

```python
import matplotlib.pyplot as plt
import pandas as pd

# 匯入資料。記得先安裝 openpyxl
df = pd.read_excel('./src01.xlsx')

# 資料
x_name = df['name']
y_num01 = df['num01']
y_num02 = df['num02']
y_num03 = df['num03']
y_num04 = df['num04']

# 設定每組的間距
bar_width = 0.2
x_indices = np.arange(len(x_name))

# 建立兩個 subplot，共享 x 軸
plt.figure(figsize=(10, 10), dpi=100)

# 上方長條圖 (num01, num02, num03)
plt.subplot(2, 1, 1)
plt.bar(x_indices - 0.2, y_num01,
width=bar_width, label='num01')
plt.bar(x_indices, y_num02,
width=bar_width, label='num02')
plt.bar(x_indices + 0.2, y_num03,
width=bar_width, label='num03')
plt.xticks([])  # 移除 x 軸標籤 (保留下方)
plt.ylabel('y_label_for_01_02_03')  # y 軸標
籤 (上方)
plt.title('Title01')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)

# 下方長條圖 (num04) - 值較小獨立展示
plt.subplot(2, 1, 2)
```

```
plt.bar(x_indices, y_num04,
width=bar_width, color='orange',
label='num04')
plt.xticks(x_indices, x_name)
plt.xlabel('Name')
plt.ylabel('y_label_for_04')  # y 軸標籤 (下方)
plt.title('Title02')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)

# 調整子圖之間的間距。tight_layout() 可以調整子圖之間的間距，使得子圖之間不會重疊。
plt.tight_layout()
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

Module 5. 資料庫基礎應用

MySQL 環境建置

簡介

MySQL (https://www.mysql.com/)是一個關聯式資料庫管理系統（Relational Database Management System，RDBMS），它是一個常用的資料庫工具，被廣泛應用於許多網站和應用程式的開發中。開發者可以使用結構化查詢語言 (Structured Query Language，SQL) 來新增、修改和查詢資料庫中的資料表 (table) 和記錄 (record / tuple)。此外，MySQL 還提供了諸如索引 (index)、觸發器 (trigger)、預存程序 (stored procedure) 和檢視表 (view) 等進階功能，使得資料庫的管理更加靈活和強大。

安裝 MySQL

圖：到 MySQL 首頁 (https://www.mysql.com/)，點選 DOWNLOADS

圖：移到 DOWNLOADS 首頁下方，有個 MySQL Community (GPL) 連結，按下去

圖：按下 MySQL Installer for Windows

圖：選擇檔案較大的 Windows MSI Installer，按下 Download

圖：直接按下 No thanks, just start my download，下載 MySQL 安裝檔

圖：下載完成後，對檔案連點兩下後進行安裝，中間確認訊息都按「是」

圖：選擇 Developer Default，按下 Next

圖：按下 Execute

圖： 等待一段時間，自動安裝 MySQL 相關產品和服務

圖： 全部安裝完後，按下 Next

圖： 準備開始進階設定，按下 Next

圖： 依需求修改預設設定，或是直接按下 Next

圖： 準備自訂 root 密碼，按下 Next

圖：輸入 root 密碼，同時可以新增 MySQL User Accounts，或是直接按下 Next

圖：依需求修改預設設定，或是直接按下 Next

圖：確認是否要讓開啟 MySQL 的 Window 帳號，以及管理者群組，都能存取 data 資料夾，可直接按下 Next

圖：準備執行先前的設定，按下 Execute

圖： 沒有錯誤訊息發生，可按下 Finish

圖： 設定 Router，按下 Next

圖：按下 Finish

圖：按下 Next

圖：輸入 root 的 Password，按下 Check，確認右上方 Connection succeeded，按下 Next

圖：按下 Execute

圖： 按下 Finish

圖： 按下 Next

圖： 兩個都勾選的話，在按下 Finish 後會開啟；按下 Finish

圖： 這是對 MySQL 伺服器下指令的地方，沒有用到的話，可以先關閉或略過

圖：MySQL Workbench 是操作 MySQL 資料庫的圖型化介面

資料庫設計議題

資料庫的設計原則如下：

- 確認綱要（schema）中屬性（attributes）的語意（semantic）清楚無誤。
- 減少值組中的重覆資訊。
- 減少值組中的 NULL 值。

異常（anomalies）原則上可以分成三種：

- 新增異常（insertion anomaly）
- 刪除異常（delection anomaly）
- 修改異常（modification anomaly）

正規化（normalization）是在結構化分析與設計中，建立資料模式所用的技術，一種資料表分割的法則，目的是為了降低資料的重覆性，同時避免異常（anomalies）情況的發生。

（圖）正規化的常見形式

正規化步驟	規則
1NF	● 每一個欄位只能有一個值。 ● 沒有任何資料完全重覆。

	● 資料表中有主鍵，其它欄位相依與主鍵。
2NF	● 符合 1NF。 ● 每一個非鍵的屬性（例如姓名、性別）必須完全相依於主鍵（例如學號）。 ● 也就是不可以部分功能相依於主鍵。
3NF	● 符合 2NF。 ● 各欄位與主鍵之間，沒有遞移相依的關係。 ● 若要找出資料表中的遞移相依性，簡單的方法就正從左到右掃瞄各個欄位，有沒有與主鍵無關的相依性存在。

（表）實務上最多進行到 3NF

（圖）正規化步驟

備註:

功能相依（X→Y），代表欄位 X 可以決定或找到欄位 Y，或是稱欄位 Y 相依於欄位 X，例如

- 「學生編號」可以決定「學生姓名」或「學生暱稱」等。
- 「老師編號」可以決定「老師姓名」等。
- 「課程編號」可以決定「學分數」或「必選修」等。

尚未正規化的資料表：學生選課資料表

學號	姓名	性別	暱稱	課程代號	課程名稱	學分數	必選修	成績	老師編號	老師姓名

087	楊○○	男	好人	C001 C002	程式設計 網頁設計	4 3	必 選	74 93	T001 T002	曾○○ 林○○
088	陳○○	女	小白	C002 C003 C004	網頁設計 視覺設計 網路教學	3 2 4	選 必 選	63 82 94	T002 T003 T005	林○○ 王○○ 謝○○

1NF 後的資料表

學號	姓名	性別	暱稱	課程代號	課程名稱	學分數	必選修	成績	老師編號	老師姓名
087	楊○○	男	好人	C001	程式設計	4	必	74	T001	曾○○
087	楊○○	男	好人	C002	網頁設計	3	選	93	T002	林○○
088	陳○○	女	小白	C002	網頁設計	3	選	63	T002	林○○
088	陳○○	女	小白	C003	視覺設計	2	必	82	T003	王○○
088	陳○○	女	小白	C004	網路教學	4	選	94	T005	謝○○

新增異常：

學號	姓名	性別	暱稱	課程代號	課程名稱	學分數	必選修	成績	老師編號	老師姓名
NULL				C005	系統分析				NULL	

若要新增一門課程，要等學生選課、確定授課老師之後，才能新增。

修改異常：

學號	姓名	性別	暱稱	課程代號	課程名稱	學分數	必選修	成績	老師編號	老師姓名
087	楊○○	男	好人	C001	程式設計	4	必	74	T001	曾○○
087	楊○○	男	好人	**C002**	**網頁設計**	3	選	~~93~~ 98	T002	林○○
088	陳○○	女	小白	**C002**	**網頁設計**	3	選	63	T002	林○○
088	陳○○	女	小白	C003	視覺設計	2	必	82	T003	王○○
088	陳○○	女	小白	C004	網路教學	4	選	94	T005	謝○○

若是希望有修「網頁設計」的同學，全班加 5 分，卻可能因為加分行為重覆多次，有些人加到、有些人沒加到，而造成資料不一致的異常現象（意即是沒有辦法透過指令進行一次性更新，甚至在更新過程中，因疏忽而缺漏修改資料，便是一種異常）。

刪除異常：

學號	姓名	性別	暱稱	課程	課程	學分數	必選修	成績	老師	老師

				代號	名稱				編號	姓名
087	楊○○	男	好人	C001	程式設計	4	必	74	T001	曾○○
087	楊○○	男	好人	C002	網頁設計	3	選	93	T002	林○○
088	陳○○	女	小白	C002	網頁設計	3	選	63	T002	林○○
~~088~~	~~陳○○~~	~~女~~	~~小白~~	~~C003~~	~~視覺設計~~	~~2~~	~~必~~	~~82~~	~~T003~~	~~王○○~~
088	陳○○	女	小白	C004	網路教學	4	選	94	T005	謝○○

當我們刪除學生記錄時，同時也會刪除課程、老師的資訊，造成資料遺失的現象。

2NF 後的資料表

將部分功能相依的欄位分割出去，組成新的資料表。

學號	姓名	性別	暱稱	課程代號	課程名稱	學分數	必選修	成績	老師編號	老師姓名
087	楊○○	男	好人	C001	程式設計	4	必	74	T001	曾○○
087	楊○○	男	好人	C002	網頁設計	3	選	93	T002	林○○
088	陳○○	女	小白	C002	網頁設計	3	選	63	T002	林○○
088	陳○○	女	小白	C003	視覺設計	2	必	82	T003	王○○
088	陳○○	女	小白	C004	網路教學	4	選	94	T005	謝○○

分割出去的結果：

學生資料表

學號	姓名	性別	暱稱
087	楊○○	男	好人
088	陳○○	女	小白

成績資料表

學號	課程代號	成績
087	C001	74
087	C002	93
088	C002	63
088	C003	82
088	C004	94

課程資料表

課程代號	課程名稱	學分數	必選修	老師編號	老師姓名
C001	程式設計	4	必	T001	曾○○
C002	網頁設計	3	選	T002	林○○
C003	視覺設計	2	必	T003	王○○
C004	網路教學	4	選	T005	謝○○

3NF 後的資料表

從左到右掃瞄各個欄位，有沒有**與主鍵無關的相依性**存在。

將「老師編號、老師姓名」獨立成一個資料，僅保留「老師編號」在課程資料表中，以便在資料查詢的過程中，還原資料的原貌。

老師資料表

老師編號	老師姓名
T001	曾○○
T002	林○○
T003	王○○
T005	謝○○

課程資料表

課程代號	課程名稱	學分數	必選修	老師編號
C001	程式設計	4	必	T001
C002	網頁設計	3	選	T002

C003	視覺設計	2	必	T003
C004	網路教學	4	選	T005

資料表正規化完成結果

學生資料表

學號	姓名	性別	暱稱
087	楊○○	男	好人
088	陳○○	女	小白

成績資料表

學號	課程代號	成績
087	C001	74
087	C002	93
088	C002	63
088	C003	82
088	C004	94

老師資料表

老師編號	老師姓名
T001	曾○○
T002	林○○
T003	王○○
T005	謝○○

課程資料表

課程代號	課程名稱	學分數	必選修	老師編號
C001	程式設計	4	必	T001
C002	網頁設計	3	選	T002
C003	視覺設計	2	必	T003
C004	網路教學	4	選	T005

開啟 MySQL Workbench

在課程中，將使用 MySQL Workbench 作為練習使用的資料庫操作軟體；在 Windows 搜尋裡，輸入「workbench」，可找到「MySQL Workbench 8.0

CE」，CE 代表 Community Edition。

圖：搜尋 workbench，可找到 MySQL Workbench 8.0 CE，請點選它

圖：Workbench 開啟畫面，請點選 Local instance MySQL80

圖：輸入先前建立的密碼，按下 OK

圖：Workbench 登入畫面

建立資料庫、資料表

建立資料庫（Database）

圖：新增 schema（類似新增資料庫）

圖：產生新頁籤（tab，叫作 my_db - Schema），Name 輸入 my_db，Charset

選擇 utf8mb4，Collation 選擇 utf8mb4_unicode_ci，按下 Apply

圖：預覽即將執行的 SQL 語法，按下 Apply

圖：確認執行成功，按下 Finish

圖：輸出成功訊息。如果當前頁籤 (tab) 不會用到，可以先關掉。

圖：在左側看到新增的資料庫 my_db，此時尚未建立資料表 (Tables)

5-1 建立資料庫.sql

```sql
/* 如果資料庫存在就刪除 */
DROP DATABASE IF EXISTS `my_db`;

/* 若資料庫不存在則新增，預設字元集為 utf8mb4，定序為 utf8mb4_unicode_ci
*/
CREATE DATABASE IF NOT EXISTS `my_db`
DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

/* 指定資料庫 */
USE `my_db`;
```

資料型態 (Data Type)

DATA TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

圖： 資料型態

建立資料表 (Table)

依照我們先前的範例，我們可以建立正規化後的那四張表。

- students（學生資料表）
- scores（成績資料表）
- teachers（老師資料表）
- courses（課程資料表）

圖： 對資料庫 my_db 連點兩下，變成粗體字，再點選上方的新增 table

圖： 以 students 資料表為例，依序建立資料表欄位 (Field)，再按下 Apply

圖： 預覽新增資料表的 SQL 語法，確認無誤後，按下 Apply

圖： 確認執行成功後，按下 Finish

圖： 會看到多了一個 students 資料表

雖然圖形使用者介面 (Graphical User Interface) 很方便，建議一開始先從練習 SQL 開始，以下是四張資料表的 SQL，提供給大家練習參考：

新增資料表語法

```
/* students（學生資料表） */
CREATE TABLE `my_db`.`students` (
    `sId` VARCHAR(3) NOT NULL COMMENT '學生編號',
    `sName` VARCHAR(20) NOT NULL COMMENT '學生姓名',
    `sGender` VARCHAR(1) NOT NULL COMMENT '學生性別',
    `sNickname` VARCHAR(50) NOT NULL COMMENT '學生暱稱',
    PRIMARY KEY (`sId`)
) COMMENT = '學生資料表';
```

```
/* scores（成績資料表） */
CREATE TABLE `my_db`.`scores` (
    `sId` VARCHAR(3) NOT NULL COMMENT '學生編號',
    `cId` VARCHAR(4) NOT NULL COMMENT '課程編號',
    `score` TINYINT(3) NOT NULL COMMENT '成績',
    PRIMARY KEY (`sId`,`cId`)
) COMMENT = '成績資料表';
```

```
/* teachers（老師資料表） */
CREATE TABLE `my_db`.`teachers` (
    `tId` VARCHAR(4) NOT NULL COMMENT '老師編號' ,
    `tName` VARCHAR(10) NOT NULL COMMENT '老師姓名',
    PRIMARY KEY (`tId`)
) COMMENT = '老師資料表';
```

```
/* courses（課程資料表） */
CREATE TABLE `my_db`.`courses` (
    `cId` VARCHAR(4) NOT NULL COMMENT '課程編號',
    `cName` VARCHAR(10) NOT NULL COMMENT '課程名稱',
    `credit` TINYINT(1) NOT NULL COMMENT '學分',
    `isCompulsory` TINYINT(1) NOT NULL COMMENT '是否必修',
    `tId` VARCHAR(4) NOT NULL COMMENT '老師編號',
    PRIMARY KEY (`cId`, `tId`)
) COMMENT = '課程資料表';
```

圖：完成後，刷新所有資料庫 (Fresh All，看到資料庫新增的 tables

新增、修改、刪除資料

新增資料

語法
INSERT INTO `資料表名稱` (`欄位 1`,`欄位 2`, … ,`欄位 N`) VALUES (值 1, 值 2, 值 3, … , 值 N), (值 1, 值 2, 值 3, … , 值 N), (值 1, 值 2, 值 3, … , 值 N);

SQL：寫入學生資料表
INSERT INTO `students` (`sId`,`sName`,`sGender`,`sNickname`) VALUES ('003', '王oo', '男', '小王'), ('004', '江oo', '女', '小江'), ('005', '周oo', '女', '小周'), ('006', '黃oo', '男', '小黃'), ('007', '丁oo', '男', '小丁'), ('008', '鄭oo', '男', '小鄭'), ('087', '楊oo', '男', '好人'), ('088', '陳oo', '女', '小白');

SQL：寫入課程資料表
INSERT INTO `courses` (`cId`, `cName`, `credit`, `isCompulsory`, `tId`) VALUES

```
('C001', '程式設計', 4, 1, 'T001'),
('C002', '網頁設計', 3, 1, 'T002'),
('C003', '視覺設計', 2, 1, 'T003'),
('C004', '網路教學', 4, 1, 'T005');
```

SQL: 寫入成績資料表

```
INSERT INTO `scores`
(`sId`, `cId`, `score`)
VALUES
('087', 'C001', 74),
('087', 'C002', 93),
('088', 'C002', 63),
('088', 'C003', 82),
('088', 'C004', 94);
```

SQL: 寫入老師資料表

```
INSERT INTO `teachers`
(`tId`, `tName`)
VALUES
('T001', '曾oo'),
('T002', '林oo'),
('T003', '王oo'),
('T005', '謝oo');
```

修改資料

語法

```
UPDATE `資料表名稱`
SET
`欄位 1` = 值 1,
`欄位 2` = 值 2,
`欄位 3` = 值 3
WHERE `欄位 A` = 值 A
AND `欄位 B` = 值 B
OR `欄位 C` = 值 C
```

SQL: 將老師編號為 T001 的姓名，改成黃○○

```
UPDATE `teachers`
```

```
SET `tName` = '黃oo'
WHERE `tId` = 'T001';
```

刪除資料

語法
DELETE FROM `資料表名稱` WHERE `欄位 1` = 值 1 AND `欄位 B` = 值 B OR `欄位 C` = 值 C;

SQL: 刪除學生編號為 088，且課程編號為 C004 的資料刪除
DELETE FROM `scores` WHERE `sId` = '088' AND `cId` = 'C004';

提醒：將先前修改、刪除的資料更新、新增回來
/* 將老師編號為 T001 的姓名，改成 曾oo */ UPDATE `teachers` SET `tName` = '曾oo' WHERE `tId` = 'T001'; /* 新增一筆資料，學生編號為 088、課程編號為 C004，成績為 94 */ INSERT INTO `scores` (`sId`, `cId`, `score`) VALUES ('088', 'C004', 94);

查詢資料

基本查詢

語法
SELECT * FROM `資料表名稱` WHERE `欄位 A` = 值 1 AND `欄位 B` > 值 B OR `欄位 C` != 值 C ORDER BY `欄位 D` ASC

語法
SELECT `欄位 1`,`欄位 2`, … ,`欄位 N` FROM `資料表名稱` WHERE `欄位 A` = 值 1 AND `欄位 B` > 值 B OR `欄位 C` != 值 C ORDER BY `欄位 D` ASC

語法
SELECT `資料庫`.`資料表名稱`.`欄位 1`, `資料庫`.`資料表名稱`.`欄位 2`, … , `資料庫`.`資料表名稱`.`欄位 N` FROM `資料庫`.`資料表名稱` WHERE `資料庫`.`資料表名稱`.`欄位 A` = 值 1 AND `資料庫`.`資料表名稱`.`欄位 B` > 值 B OR `資料庫`.`資料表名稱`.`欄位 C` != 值 C ORDER BY `資料庫`.`資料表名稱`.`欄位 D` ASC

M0: 列出所有的學生資料
SELECT * FROM `students`

M1: 列出學生的學號、姓名、暱稱
SELECT `sId`, `sName`, `sNickname` FROM `students`

M2: 找出老師編號為 T001 或 T003 的老師姓名
SELECT `tName` FROM `teachers` WHERE `tId` = 'T001' OR `tId` = 'T003'

M3: 將學生依學生編號排序，號碼最大的排前面，全部列出來
SELECT * FROM `students` ORDER BY `students`.`sId` DESC

M4: 找出性別為女、暱稱為小白的學生姓名

`SELECT` `` `sName` ``
`FROM` `` `students` ``
`WHERE` `` `sGender` `` `= '女'`
`AND` `` `sNickname` `` `= '小白';`

M4A: 找出性別為女、暱稱為小白的學生姓名（用 AS 來替代資料表名稱）

`SELECT` `` `s`.`sName` ``
`FROM` `` `students` `` `AS` `` `s` ``
`WHERE` `` `s`.`sGender` `` `= '女'`
`AND` `` `s`.`sNickname` `` `= '小白';`
 	
補充說明：
’資料表’ AS ‘資料表代稱‘，其中 <u>AS 可以省略</u>。

以下將透過 LIKE 關鍵字，搭配 SQL 內建的「%、_」模糊查詢功能，來查詢資料：

- %：配對任何文字，同時字數長度不限，可以放置在字串的前後，或是在每個分隔文字之間。例如
  - %人：可配對「好人」、「擺渡人」、「這裡沒有人」等。
  - 好%：可配對「好人」、「好東西」、「好像很厲害」等。
  - %好人：可配對「做好人」、「當個好人」、「你真是個好人」等。
  - 好人%：可配對「好人家」、「好人好事」、「好人村村長」等
  - %好人%：可配對「是好人嗎」、「走入好人村」、「當個好人村村長」。
  - 好%人：可配對「好男人」、「好嚇唬人」等。
  - %好%人%：可配對「新好男人！」、「好好照顧人家」等。
- _:配對任何文字，一個「_」代表一個字，可以放置在字串的前後，或是在每個分隔文字之間。例如
  - _人：可配對「好人」、「男人」、「女人」等。
  - 好_：可配對「好人」、「好事」、「好高」等。
  - _好人：可配對「當好人」、「做好人」等。
  - 好人_：可配對「好人家」、「好人選」、「好人村」等。
  - _好人_：可配對「是好人嗎」、「良好人選」等。
  - 好_人：可配對「好男人」、「好女人」等。
  - _好_人_：可配對「當好男人吧」、「有好男人嗎」等。

M5: 找出暱稱是「小」開頭的學生姓名與性別

```
SELECT `sName`, `sGender`
FROM `students`
WHERE `sNickname` LIKE '小%'
```

M5A: 找出暱稱是「人」結尾的學生姓名與性別

```
SELECT `sName`, `sGender`
FROM `students`
WHERE `sNickname` LIKE '%人'
```

M6: 找出學號（三位數）第二碼為 8 的學生編號、姓名與暱稱

```
SELECT `sId`, `sName`, `sNickname`
FROM `students`
WHERE `sId` LIKE '_8_'
```

M7: 找出學生資料表當中，學生編號為 003、005、007、087 等四位學生的所有資料。

```
SELECT *
FROM `students`
WHERE `sId` IN ('003', '005', '007', '087');
```

接下來，我們將使用 LIMIT 語法，模擬資料「分頁」(pagination)的情形：

M8: 依學生編號由小到大排序，取前三筆所有欄位資料

```
SELECT *
FROM `students`
ORDER BY `sId` ASC
LIMIT 3
```

M8-1: 依學生編號由小到大排序，從第 1 筆開始，取得 3 筆所有欄位資料

```
SELECT *
FROM `students`
ORDER BY `sId` ASC
LIMIT 0, 3
```

M8-2: 依學生編號由小到大排序，從第 3 筆開始，取得 3 筆所有欄位資料

```
SELECT *
FROM `students`
ORDER BY `sId` ASC
```

```
LIMIT 3, 3
```

M8-3: 依學生編號由小到大排序，從第 6 筆開始，取得 3 筆所有欄位資料

```
SELECT *
FROM `students`
ORDER BY `sId` ASC
LIMIT 6, 3
```

進階查詢

圖： 進階的查詢語法

語法

```
SELECT `欄位 1`, `欄位 2`
FROM `資料表 1` INNER JOIN `資料表 2`
ON `資料表 1`.`欄位 A` = `資料表 2`.`欄位 A`
```

說明：

INNER JOIN 兩個資料表的交集

LEFT JOIN 以左邊資料表為主，進行結合 RIGHT JOIN 以右邊資料表為主，進行結合 OUTTER JOIN 兩張資料表的聯集

M0: 查詢各個課程中的授課老師姓名

`SELECT` `` `cName` `` , `` `tName` ``
`FROM` `` `courses` `` `INNER JOIN` `` `teachers` ``
`ON` `` `courses` `` . `` `tId` `` = `` `teachers` `` . `` `tId` ``

M1: 列出所有擁有成績的學生姓名、課程名稱與分數

`SELECT` `` `sName` `` , `` `cName` `` , `` `score` ``
`FROM` `` `scores` ``
`INNER JOIN` `` `students` ``
`ON` `` `scores` `` . `` `sId` `` = `` `students` `` . `` `sId` ``
`INNER JOIN` `` `courses` ``
`ON` `` `scores` `` . `` `cId` `` = `` `courses` `` . `` `cId` `` ;

子查詢

進行多資料表查詢，除了使用 join 外，也可以使用 SQL 子查詢(sub query)。子查詢就是在一個 select 指令內再放入一個 select 查詢指令進行查詢，通常是位在 select 的 where 子句，可以透過子查詢取得查詢條件；子查詢的效能較差，除非資料不大，否則不建議頻繁使用。

語法
`SELECT` 欄位名稱 1,欄位名稱 2,...,欄位名稱 n `FROM` 資料表名稱 1 `WHERE` 欄位名稱 `IN` (或是 =) (　　`SELECT` 欄位名稱 　　`FROM` 資料表名稱 2 　　`WHERE` 條件)

M2: 查詢所有考過成績 90 分以上的學生編號和學生姓名

`SELECT` `` `sId` `` , `` `sName` ``
`FROM` `` `students` ``
`WHERE` `` `sId` `` `IN` (

```
    SELECT `sId`
    FROM `my_db`.`scores`
    WHERE `score` >= 90
);
```

函式

MySQL 的內建函式族繁不及備載，以下僅列出課程會用到函式。

參考連結：
MySQL Functions
https://www.w3schools.com/mysql/mysql_ref_functions.asp

語法
SELECT 函式(`欄位`) FROM `資料表` 備註：也可以加上 WHERE 條件來過濾資料

M1: 查詢所有課程中，最大的學分數是多少
SELECT MAX(`credit`) FROM `courses`

把離散的資料加以群組起來：

語法
SELECT `欄位 1`,`欄位 2`, 函式(`欄位 3`) FROM `資料表` WHERE `欄位 A` = `欄位 B` GROUP BY `欄位 1`,`欄位 2` HAVING 函式(`欄位 3`) > 值

M2: 查詢各個學分數，相對應的課程各有幾個
SELECT `credit`, COUNT(`cId`) FROM `courses` GROUP BY `credit`

M3: 計算平均成績

```
SELECT AVG(`score`)
FROM `scores`
```

M4: 查詢各個課程的平均成績

```
SELECT `cId`, AVG(`score`)
FROM `scores`
GROUP BY `cId`
```

函式	說明
CASE()	SELECT CASE column WHEN a THEN b WHEN c THEN d ELSE e END 若 column 等於 a，則返回 b 若 column 等於 c，則返回 d 否則返回 e 另外，a 與 c 可為運算式，例如 a < 10，只要是 a 小於 10 都返回 b
IF()	SELECT IF(input, x, y) IF()返回一個數字或字符串值。 如果 input 是 TRUE (input 不為 0 且 x 不為 NULL)，那麼 IF() 返回 x，否則它返回 y。 類似 PHP 中的 三元運算子

CASE()

範例

```
SELECT `sId`, `cId`, `score`,
CASE
    WHEN `score` >= 90 THEN '成績在 90 分以上'
    WHEN `score` < 90 AND `score` >= 70 THEN '成績在 70 到 89 分之間'
    ELSE '成績低於 70'
END AS `msg`
FROM `scores`
```

F1: 找出所有課程的成績，判斷是否高於或低於平均成績並印出提示文字

```
SELECT `sId`, `cId`, `score`,
CASE
    WHEN `score` >= 90 THEN '成績在 90 分以上'
    WHEN `score` < 90 AND `score` >= 70 THEN '成績在 70 到 89 分之間'
    ELSE '成績低於 70'
END AS `msg`,
(SELECT AVG(`score`) FROM `scores`) AS `avg_scores`,
CASE
    WHEN `score` >= (SELECT AVG(`score`) FROM `scores`) THEN '大於等於平均'
    ELSE '小於平均'
END AS `avg_msg`
FROM `scores`
```

IF()

範例
`SELECT `sId`, `cId`, `score`, IF(`score` >= 90, "大於等於 90 分", "低於 90 分") AS `msg``   `FROM `scores`;`

F2: F1 的 IF 版本，其中判斷是否大於等於 90、小於 90 即可
`SELECT `sId`, `cId`, `score`,`   `IF(`score` >= 90, '大於等於 90', '小於 90') AS `msg`,`   `(SELECT AVG(`score`) FROM `scores`) AS `avg_scores`,`   `IF(`score` >= (SELECT AVG(`score`) FROM `scores`), '大於等於平均成績', '小於平均成績`   `') AS `avg_msg``   `FROM `scores``

完整性原則與關聯介紹

在規劃資料庫的過程中，需要滿足完整性與資料表間對應的關聯性，才能設計出合適的資料庫結構。

完整性原則

- **參考完整性***

  資料表 A 的 Foreign Key（FK，外來鍵）集合，其值域應該落在資料表 B 的 Primary Key（PK，主鍵）當中，否則資料表 A 的 FK 無法參考到資料表 B 的 PK。

課程資料表

P.K = Primary Key　　F.K = Foreign Key

cId	cName	credit	isCompulsory	tId
C001	程式設計	4	1	T001
C002	網頁設計	3	1	T002
C003	視覺設計	2	1	T003
C004	網路教學	4	1	T005
…	…	…	…	T001
…	…	…	…	T005
…	…	…	…	T002
…	…	…	…	T087

老師資料表

tId	tName
T001	黃○○
T002	林○○
T003	王○○
T005	謝○○

圖: T087 並不存在於 tId 的值域當中

- **實體完整性***

  資料表 B 的 Primary Key，其值不得為空值（NULL），否則無法讓資料表 A 的 Foregin Key 參考到。

課程資料表

P.K = Primary Key　　F.K = Foreign Key

cId	cName	credit	isCompulsory	tId
C001	程式設計	4	1	T001
C002	網頁設計	3	1	T002
C003	視覺設計	2	1	T003
C004	網路教學	4	1	T005
…	…	…	…	T001
…	…	…	…	T005
…	…	…	…	T002

老師資料表

tId	tName
T001	黃○○
T002	林○○
T003	王○○
T005	謝○○
NULL	翁○○

圖: 老師資料主鍵為空值，無法讓課程資料表的外鍵進行參考

- **值域完整性**

  若是欄位的結構設定中，指定型態為數值（例如 credit），就不能寫入字串或日期時間（例如 varchar 或 datetime）。

課程資料表

P.K = Primary Key　　F.K = Foreign Key

cId	cName	credit	isCompulsory	tId	
C001	程式設計	4	1	T001	
C002	網頁設計	3	1	T002	
C003	視覺設計	2	1	T003	
C004	網路教學	4	1	T005	
…	…	3	…	T001	
…	…	3	…	T005	
…	…	4	…	T002	
…	…	讚	…	…	← 該筆資料無法被新增

圖：因 credit 欄位的類型是 tinyint，因此無法寫入字串類型的資料

資料表關聯

- 一對一

	A	B	C	D	E	F
1	sId	sName		sId	sNickname	
2	003	王〇〇		003	小王	
3	004	江〇〇		004	小江	
4						

圖：一對一關係

- 一對多

B	C	D	E	F	G	H	I	J
sName	sId					sId	cId	score
楊〇〇	087					087	C001	74
陳〇〇	088					087	C002	93
						088	C002	63
						088	C004	94

students　　scores

圖：一對多關係

- 多對多

圖:多對多關係

Pandas 與 SQL 用法對照表

avichawla.substack.com

Operation	Pandas	SQL
Read CSV	pd.read_csv(file)	LOAD DATA INFILE 'data.csv' INTO TABLE table FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 ROWS;
Print first 10 (or k) rows	df.head(10)	SELECT * FROM table LIMIT 10;
Dimensions	df.shape	SELECT count(*) FROM table;
Datatype	df.dtyes	DESCRIBE table;
Filter Data	df[df.column>10]	SELECT * FROM table where column>10;
Select column(s)	df.column	SELECT column FROM table;
Sort	df.sort_values("column")	SELECT * FROM table ORDER BY column;
Fill NaN	df.column.fillna(0)	UPDATE table SET column=0 WHERE column IS NULL;
Join	pd.merge(df1, df2, on ="col", how = "inner")	SELECT * FROM table1 JOIN table2 ON (table1.col = table2.col);
Concatenate	pd.concat((df1, df2))	SELECT * FROM table1 UNION ALL table2;
Group	df.groupby("column"). agg_col.mean()	SELECT column, avg(agg_col) FROM table GROUP BY column;
Unique values	df.column.unique()	SELECT DISTINCT column FROM table;
Rename column	df.rename(columns = {"old_name": "new_name"})	ALTER TABLE table RENAME COLUMN old_name TO new_name;
Delete column	df.drop(columns = ["column"])	ALTER TABLE table DROP COLUMN column;

圖: 用法對照表