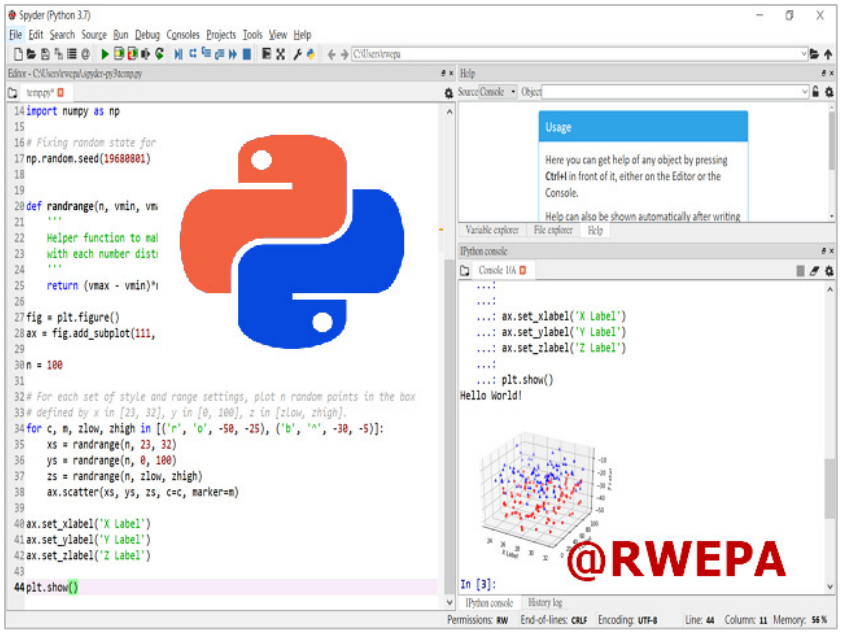
Python 程 式 設 計

育 達 科 技 大 學

資訊管理系所

主編：李明昌



108 年 12 月

國家圖書館出版品預行編目(CIP)資料

Python 程式設計 / 李明昌主編.

-- 苗栗縣造橋鄉 : 育達科大, 民 108.12 面 ; 公分

ISBN 978-986-5911-77-5(平裝)

1.Python(電腦程式語言)

312.32P97 108022920

目錄

第 1 章 Python 語言簡介 1

1.1 Python 簡介 1

1.2 Python 特性與應用 1

1.3 Python 安裝 2

1.4 Python 操作 5

第 2 章 Anaconda 簡介與安裝 8

2.1 Anaconda 特性 8

2.2 Anaconda 下載與安裝 8

2.3 Anaconda 套件管理 23

2.4 Spyder 操作 24

第 3 章 Python 語法與流程控制 26

3.1 資料型別與運算子 26

3.2 NumPy 模組的使用 36

3.3 reshape 應用 38

3.4 離群值處理 40

3.5 if 與 for 處理 42

第 4 章 資料型別與資料處理 62

4.1 Tuple 序列 62

4.2 List 串列 65

4.3 Set 集合 67

4.4 Dictionaries 字典 68

第 5 章 檔案匯入與匯出 71

5.1 認識 pandas 模組 71
5.2 資料輸入/輸出 91
第 6 章 視覺化應用 104
6.1 視覺化簡介 104
6.2 認識 matplotlib 模組 115
6.3 matplotlib 繪圖應用 116
6.4 seaborn 模組繪圖 137
6.5 互動式繪圖 150
第 7 章 迴歸分析 156
7.1 迴歸模型 Regression Model 156
7.2 迴歸分析 - 使用 scikit-learn 模組 157
第 8 章 決策樹 169
8.1 決策樹 169
8.2 鐵達尼號資料集-決策樹應用 174
第 9 章 關聯規則應用 181
9.1 購物籃分析（market-basket analysis） 181
9.2 mlxtend 模組 183
第 10 章 推薦系統 195
10.1 何謂推薦系統 Recommender System 195
10.2 電影推薦系統 197
參考文獻 216

第1章 Python語言簡介

本章節從基礎Python 語言介紹為開端, 包括以下內容:

- **1.1 Python簡介**
- **1.2 Python特性與應用**
- **1.3 Python安裝**
- **1.4 Python操作**

1.1 Python簡介

Python 是一種廣泛使用的直譯式、進階程式、通用型程式語言，由吉多·范羅蘇姆 (Guido van Rossum)創造，第一版釋出於1991年，可以視之為一種改良並加入一些其他程式語言的優點，如：LISP 物件導向程式語言。

Python的設計哲學強調代碼的可讀性和簡潔的語法（尤其是使用空格縮排劃分程式碼區塊，而非使用大括號{ }或者關鍵詞。相比於C++或Java，Python讓開發者能夠用更少的代碼表達想法。不管是小型還是大型程式，該語言都試圖讓程式的結構清晰明了。

Python與Scheme、Ruby、Perl、Tcl等動態型別程式語言一樣，Python擁有動態型別系統和垃圾回收功能，能夠自動管理記憶體使用，並且支援多種程式範式，包括物件導向、命令式、函數式和程序式程式。其本身擁有一個巨大而廣泛的標準庫。

Python 直譯器本身幾乎可以在所有的作業系統中執行。Python的其中一個直譯器CPython是用C語言編寫的、是一個由社群驅動的自由軟體，目前由Python軟體基金會管理。

參考資料: https://zh.wikipedia.org/wiki/Python (https://zh.wikipedia.org/wiki/Python)

1.2 Python特性與應用

Python特性

Python 具有以下的特性：

- 跨平台
- 開放性
- 易讀性
- 豐富套件(模組)
- 其他語言結合, 例: Cython 編譯成二進位執行檔

Python應用範圍

Python 包括以下應用範圍：

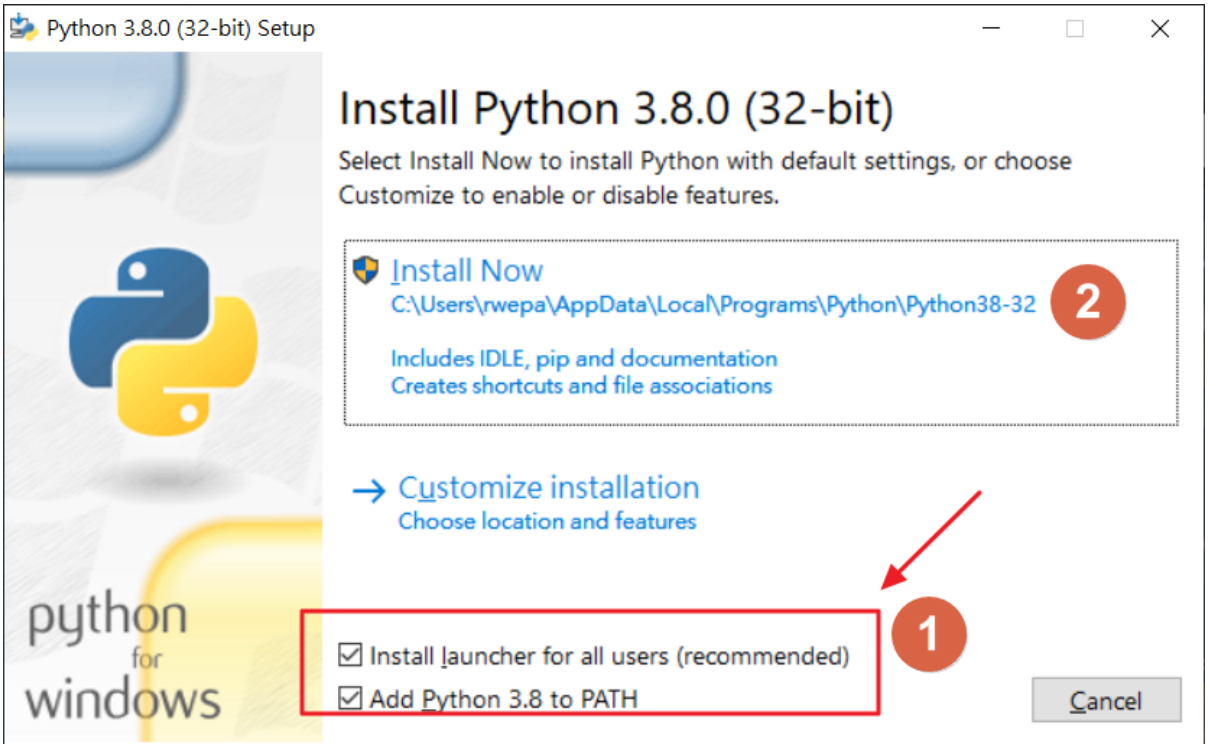
- 大數據分析
- 機器學習 (scikit-learn 模組)
- 深度學習 (TensorFlow 模組)
- 網路爬蟲
- 繪圖
- 網路應用
- 財金分析
- 物聯網應用
- 影像識別
- 科學計算
- GUI開發

1.3 Python安裝

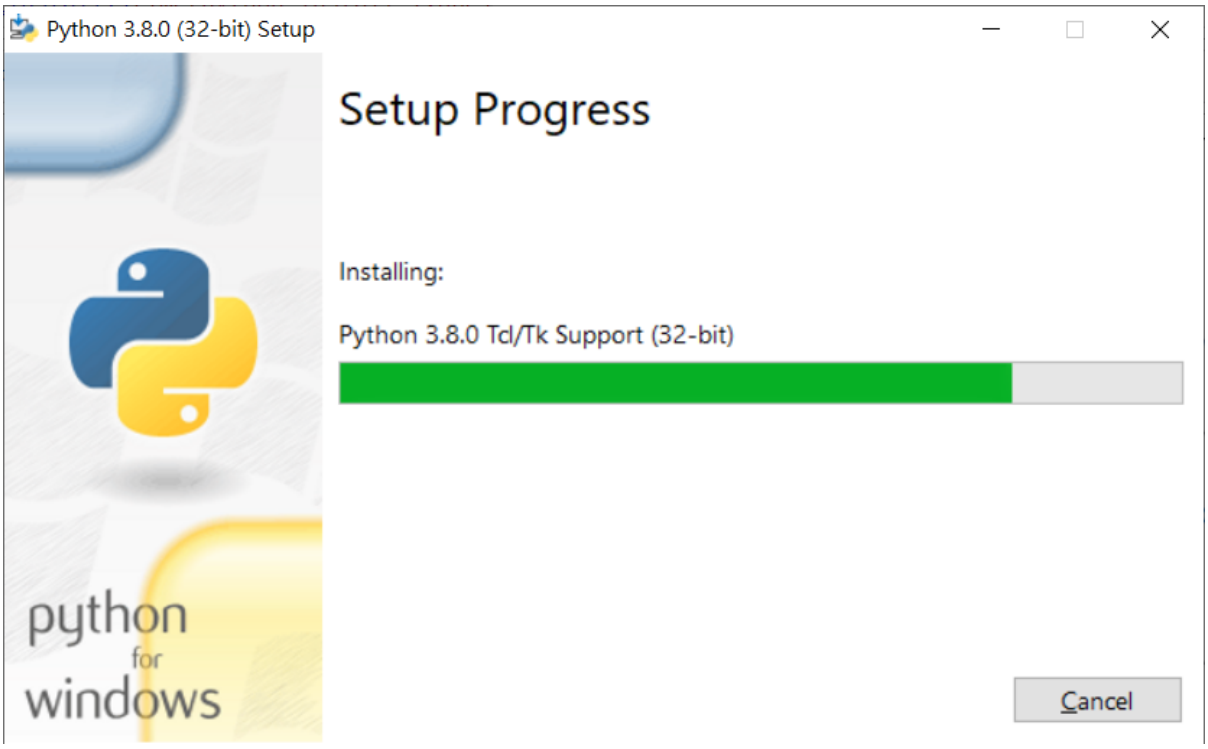
考慮 Windows 作業系統，其安裝步驟如下所示：

1. Python官網的下載頁面: https://www.python.org (https://www.python.org) ，選取中間 [Download Python 3.8.0]，下載檔案約25.1MB。

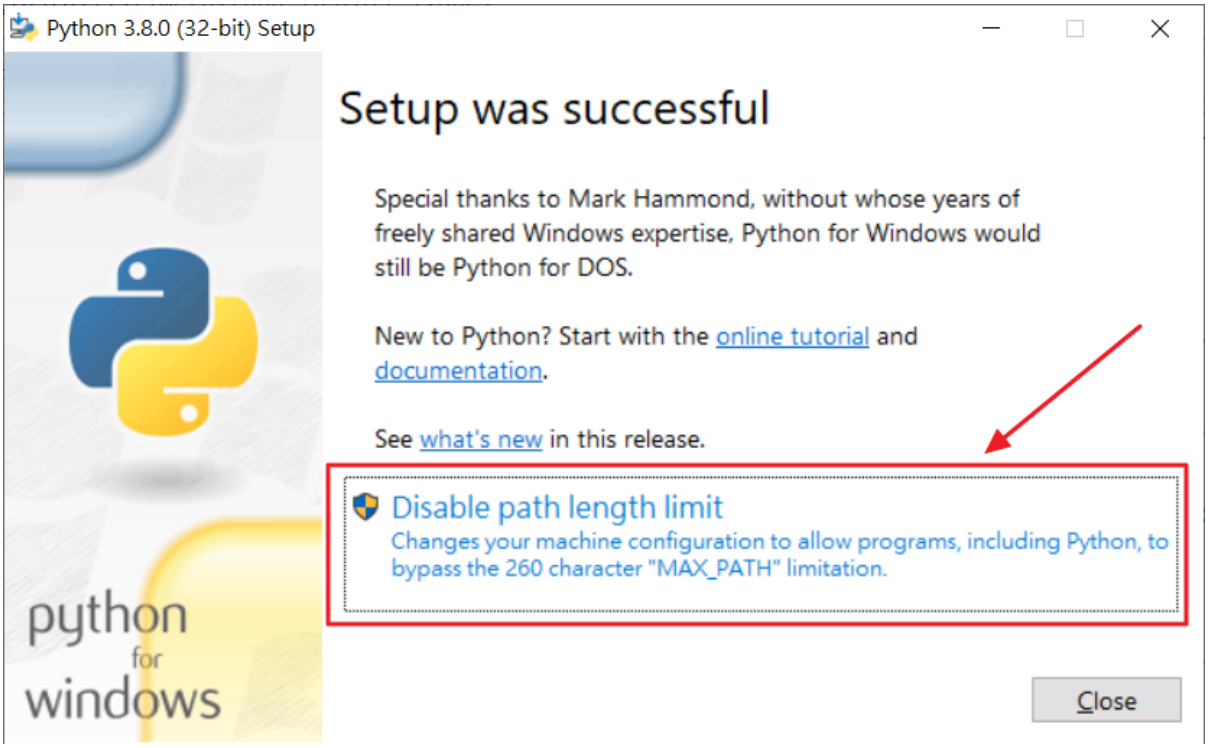
2. 執行該檔案，將二個選項打勾, 選取 [Install Now]。



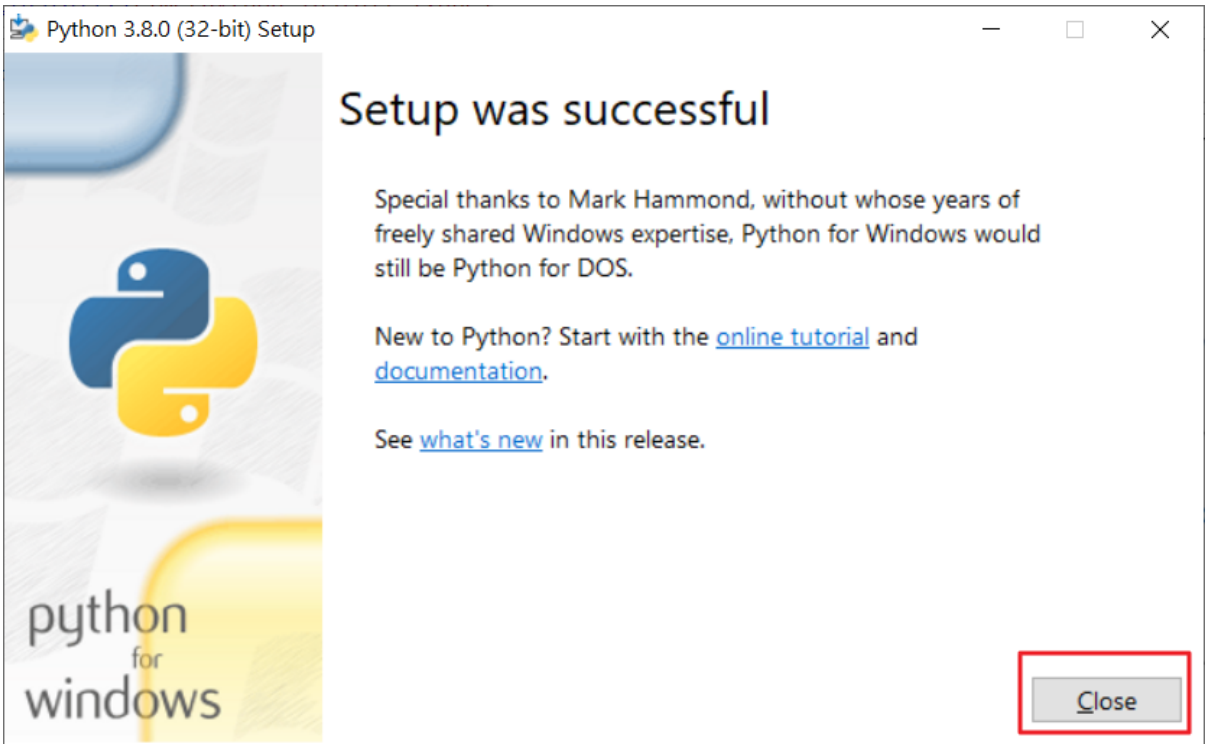
3. 安裝畫面。



4. 選取 [Disable path length limit]。



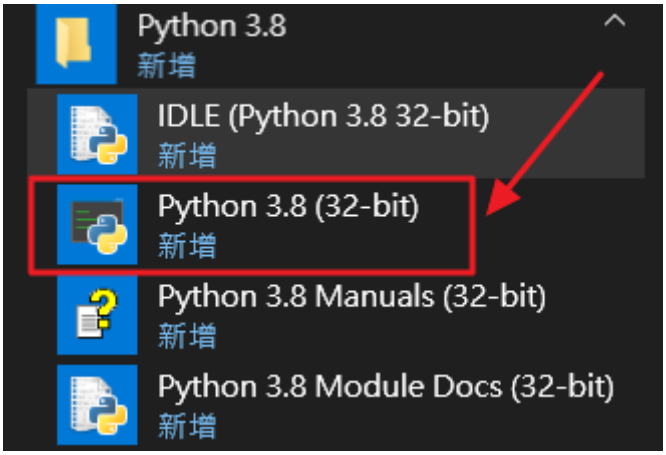
5. 安裝完成畫面，按 [Close]。



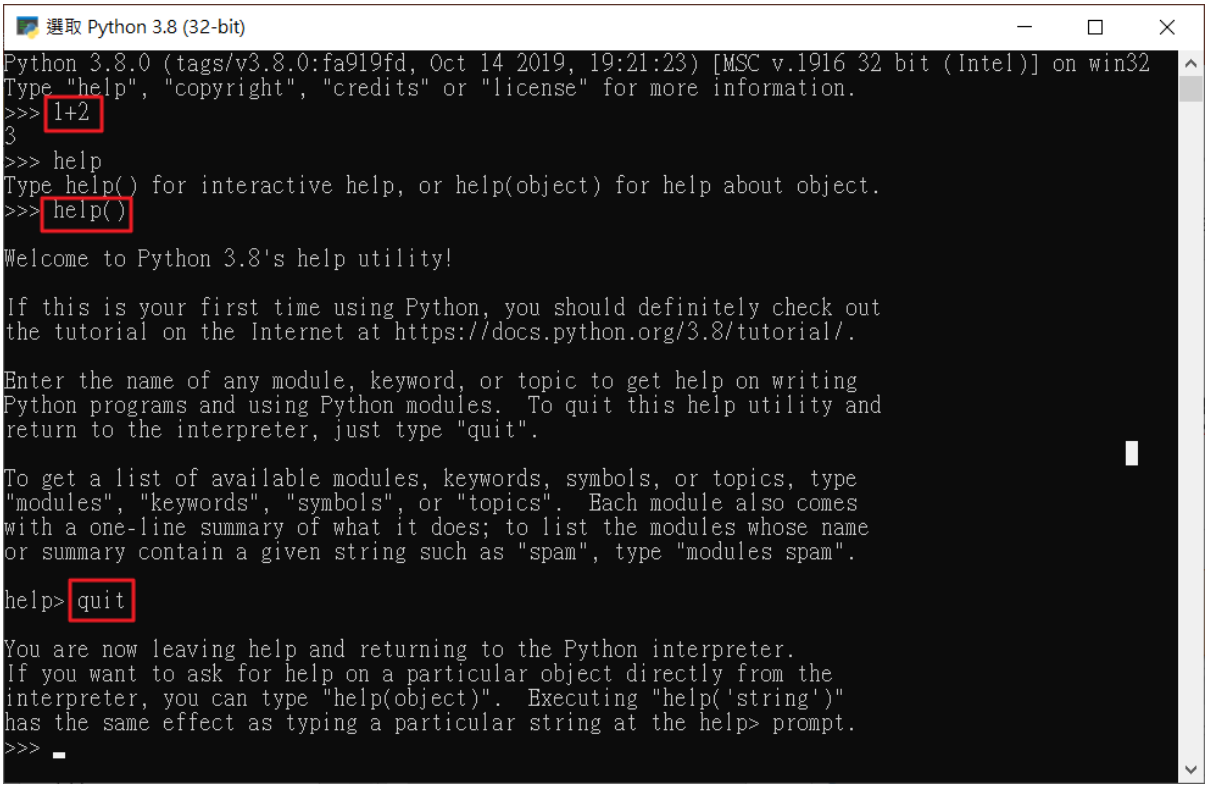
6. 安裝完成後會在程式集顯示4個程式，其中[Python 3.8(32-bit)] 為主要操作介面， [IDLE] 為視窗介面，其他二項分別是使用手冊與模組之說明。

1.4 Python操作

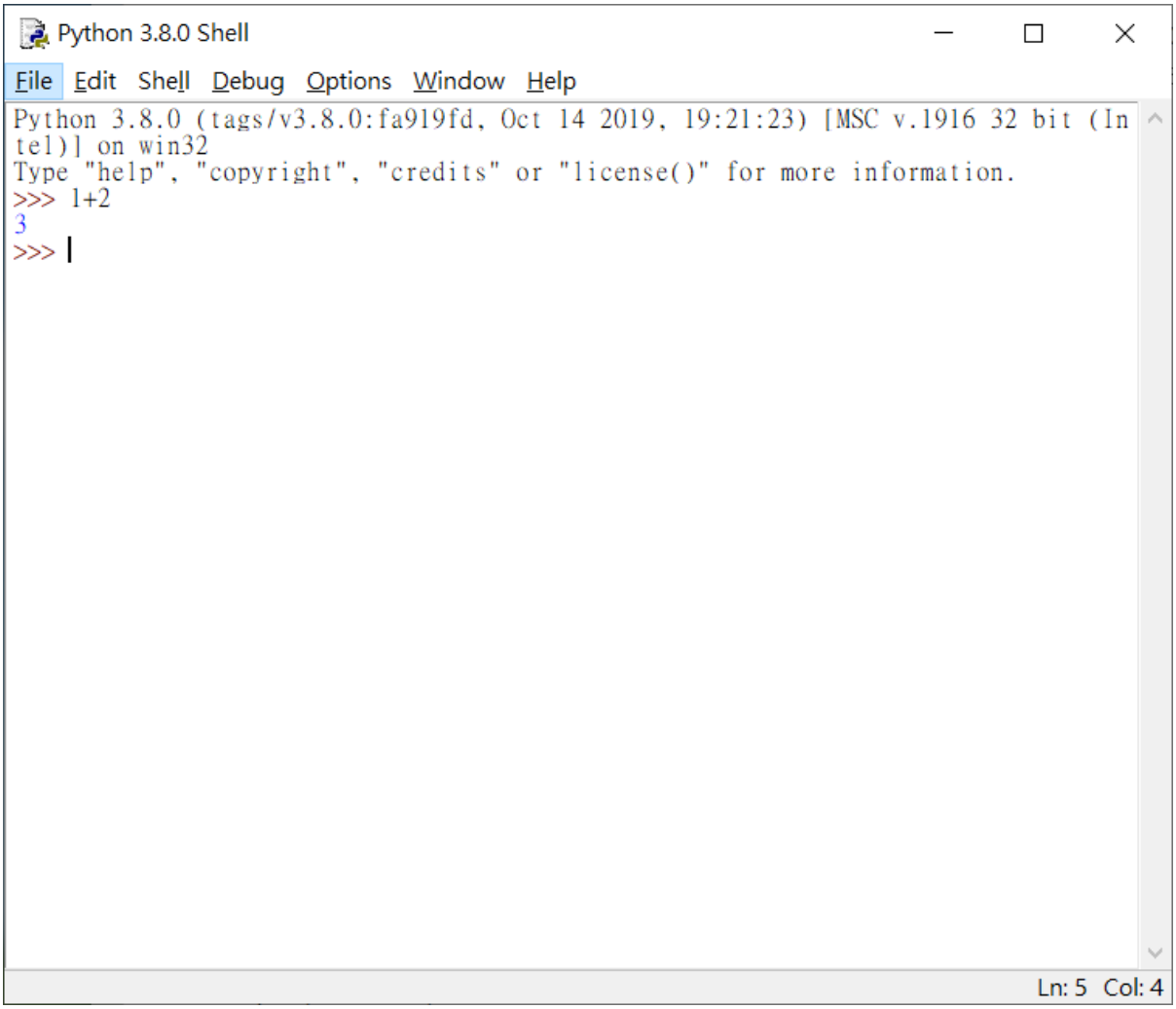
開啟 Python 3.8 會顯示互動式操作介面，輸入 1+2, 按[Enter]，結果會顯示3，輸入 help() 會顯示線上說明，輸入 quit會離開說明，輸入 exit() 會關閉視窗。



Python 命令提示列視窗



Python IDLE



注意! **Python**輸入時，有區分英文未母大小寫。

Python 撰寫特性

1. Python為提高程式可讀性，並且在數學影響下，其語法與英語具有某些相似之處。
2. Python與其他經常使用分號或括號的編程語言不同，Python使用換行符號來完成命令。
3. Python依靠縮排（使用空格）與冒號（:）來定義範圍，例如迴圈，函數和類別的範圍。其他程式語言通常使用括號 { }表示。

In [1]:

```
# 正確
if 2 > 1:
  print("2大於1!")
```

2大於1!

In [2]:

```
# 錯誤-沒有縮排
if 2 > 1:
print("2大於1!")
```

```
  File "<ipython-input-2-7dbe6b1a7a3c>", line 3
    print("2大於1!")
        ^
IndentationError: expected an indented block
```

In [ ]:

```
# 錯誤-縮排位置不同
if 2 > 1:
  print("2大於1!")
      print("縮排位置不同")
```

第2章 Anaconda 簡介與安裝

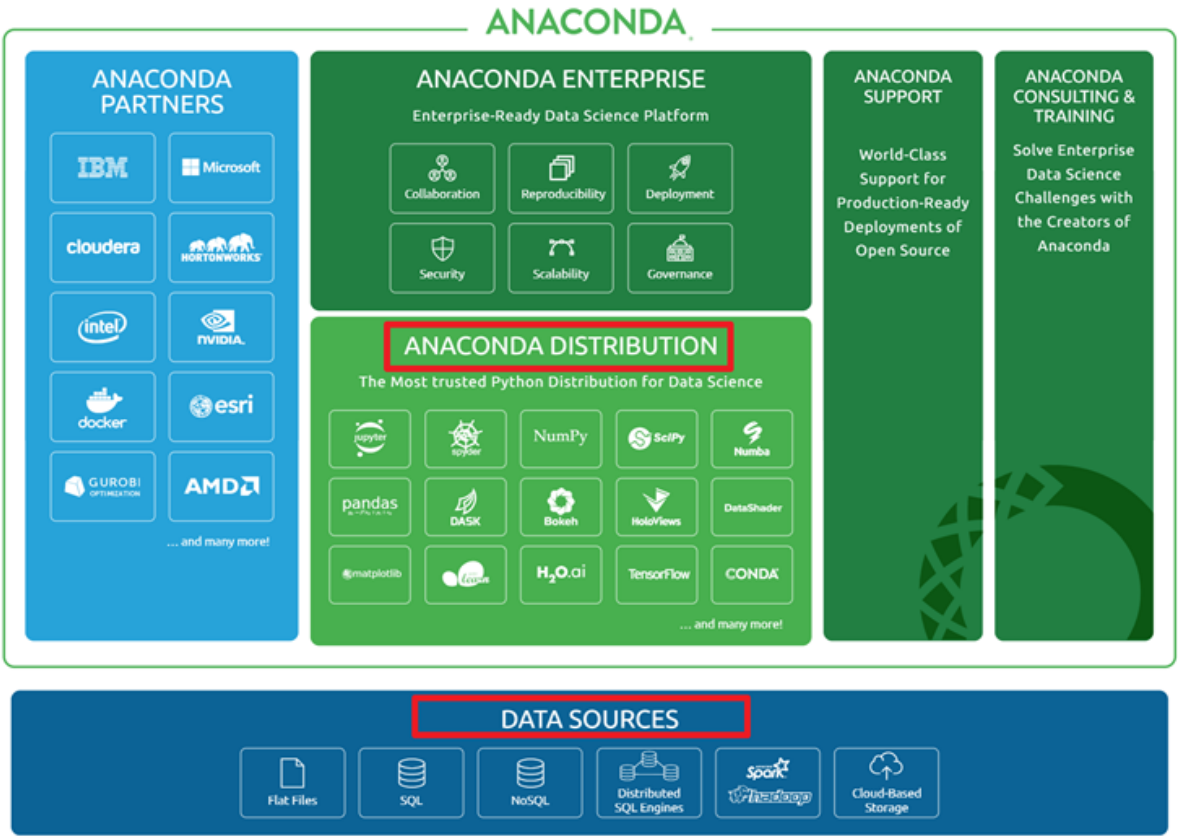
本章節從免費 **Anaconda** 軟體特性介紹為開端，包括以下內容:

- **2.1 Anaconda** 特性
- **2.2 Anaconda** 下載與安裝
- **2.3 Anaconda** 套件管理
- **2.4 Spyder** 操作

2.1 Anaconda特性

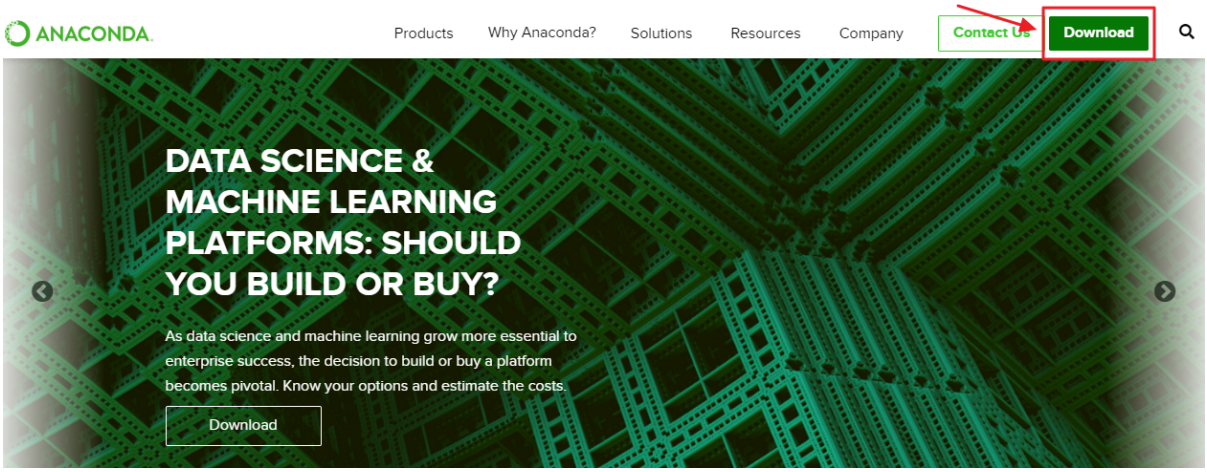
Anaconda 特性如下，參考下圖所示：

1. Anaconda是一個免費、易於安裝與管理並支援Python語言。
2. 支援1000個以上的開源套件(package)。
3. 支援 Spyder (支援 Python IDE)。
4. 支援 jupyter notebook。
5. 支援 Windows、Mac OS X和Linux。

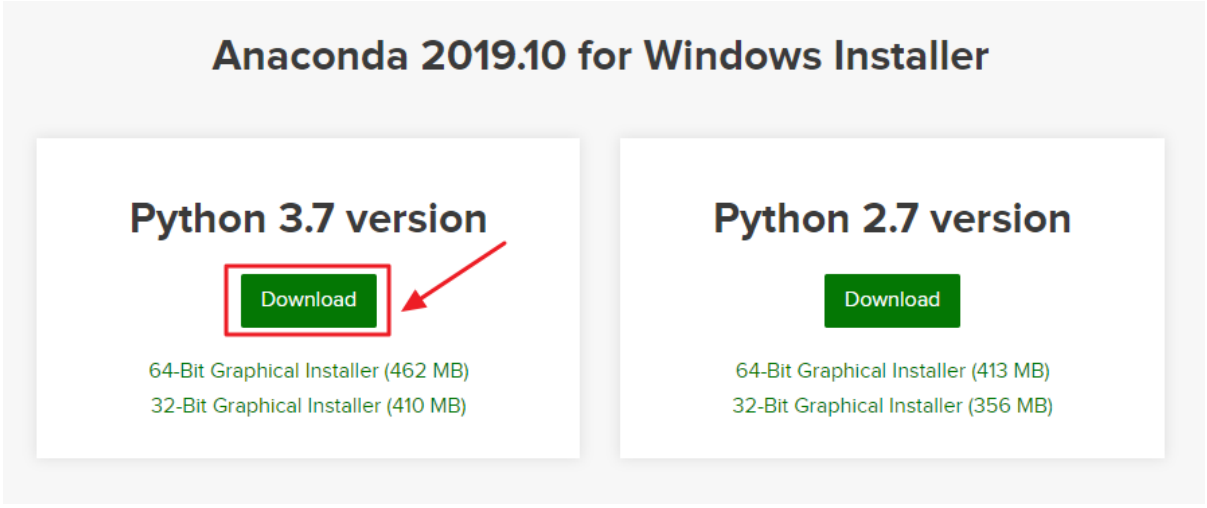


2.2 Anaconda下載與安裝

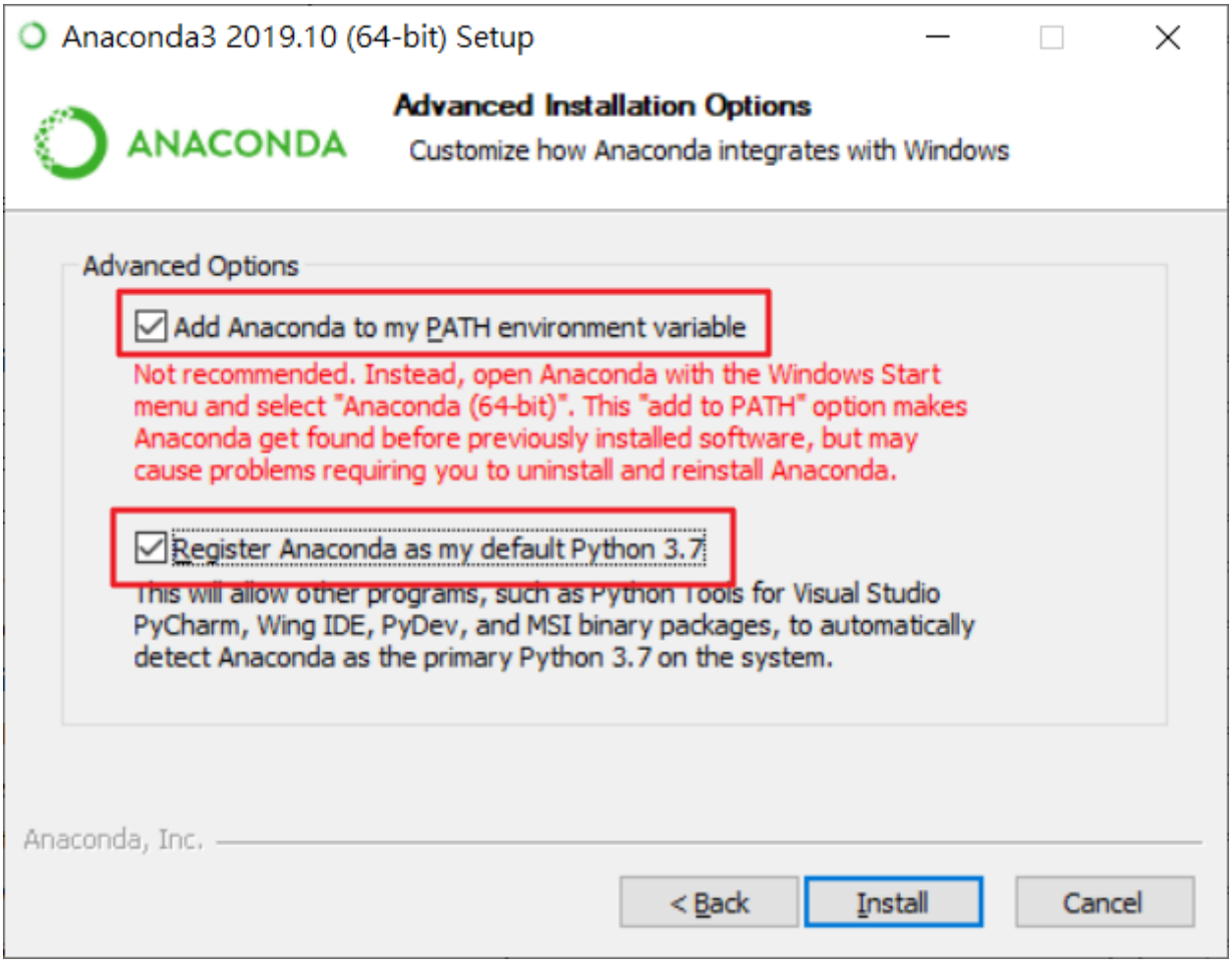
1. 連接至 https://www.anaconda.com/ (https://www.anaconda.com/) ，選取上方 Download。



2. 選取 Python 3.7 version [Download] 約462MB。



3. 安裝 Anaconda3-2019.10-Windows-x86_64.exe，其中二個選項須打勾，否則執行會有問題。

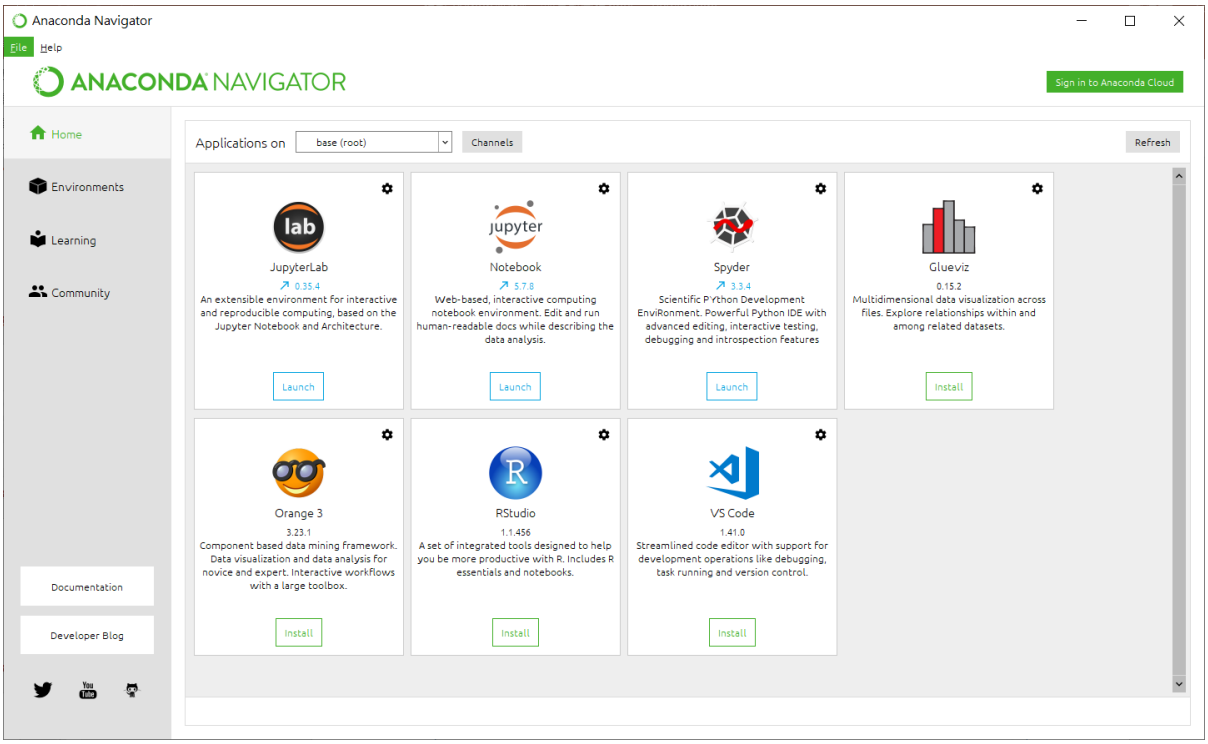


4. 安裝完成後程式集畫面如下圖所示。

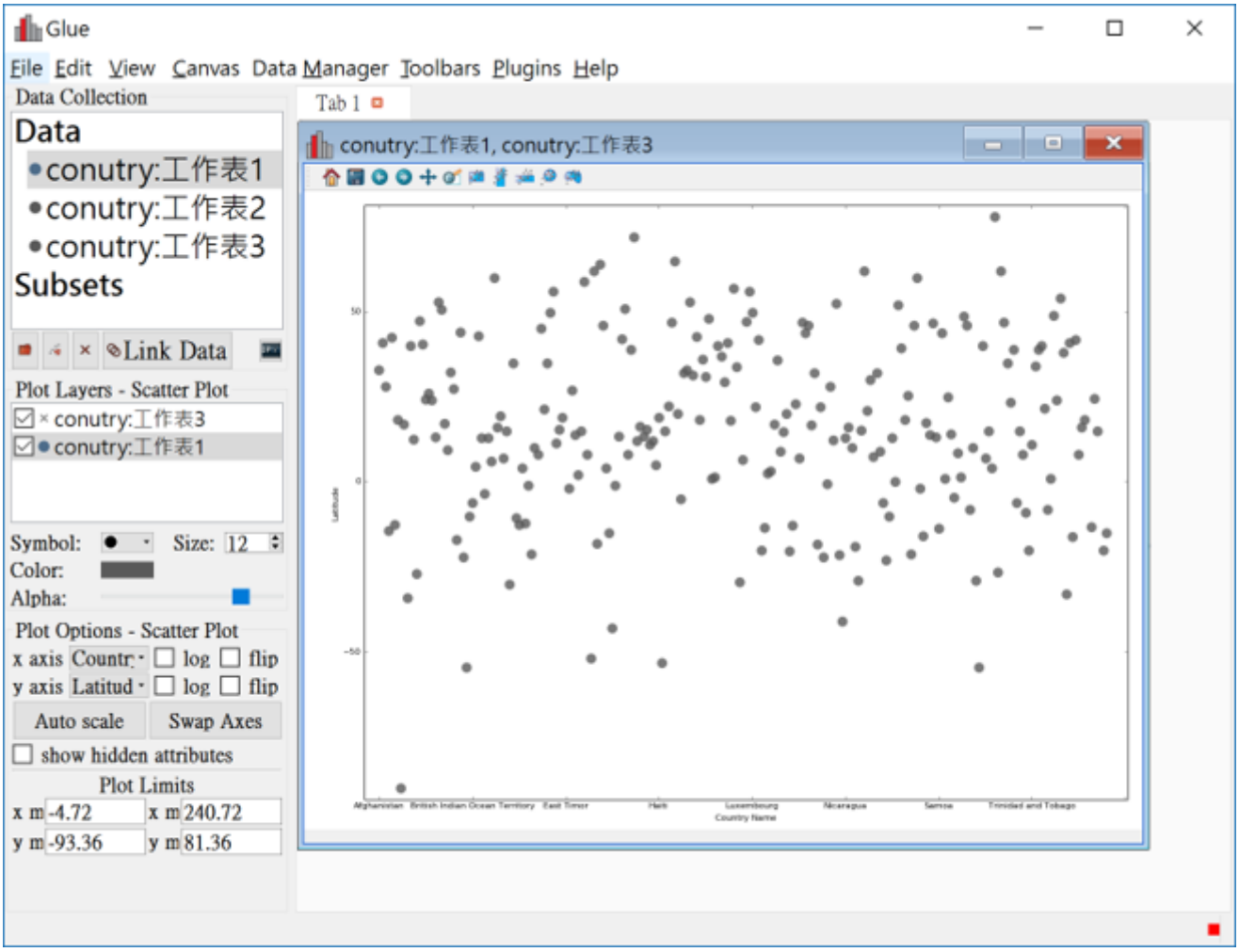
Anaconda 安裝六大程式，說明如下：

1. Anaconda Navigator：Anaconda 瀏覽程式視窗，可開啟額外功能。
2. Anaconda Powershell Prompt：Anaconda 超級命令列介面。
3. Anaconda Prompt：命令列介面視窗，可安裝額外模組。
4. Jupyter Notebook：網頁互動式程式編輯。
5. Reset Spyder Settings：重新設定Spyder功能。
6. Spyder：Python 程式整合開發環境，主要用於撰寫 Python 程式。

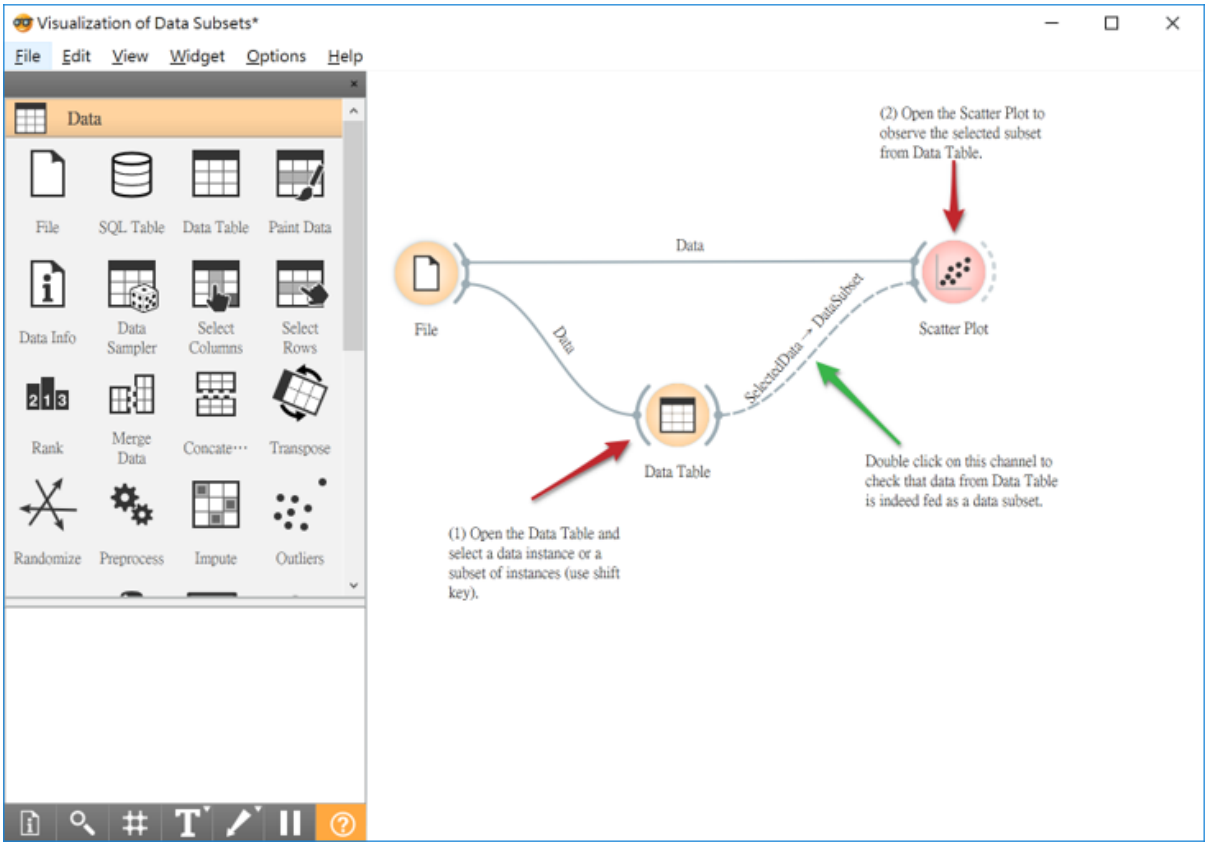
Anaconda Navigator



Glueviz 視覺化功能

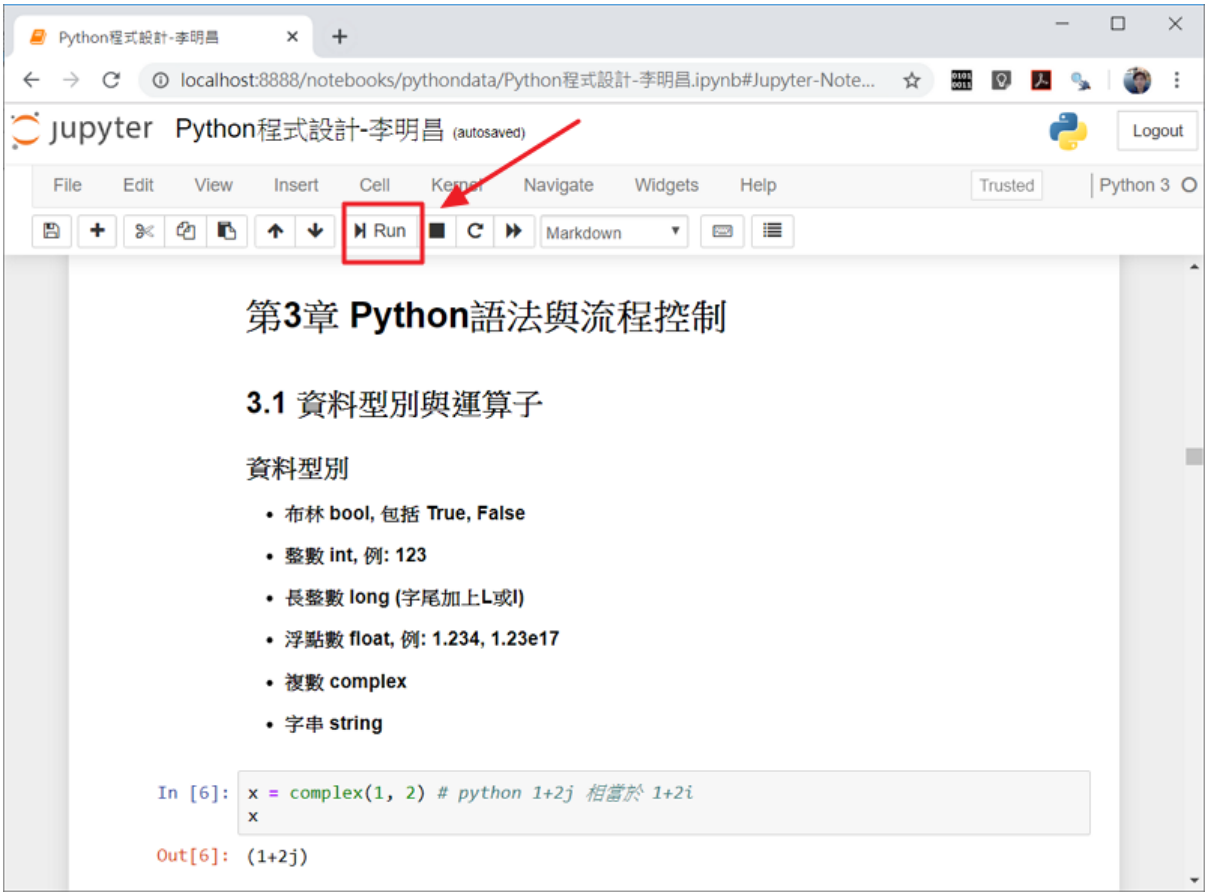


Orange 工作流程管理



Orange 工作流程管理完成圖 (Scatter Plot)

Jupyter Notebook完成圖



Jupyter Notebook 快速鍵

Jupyter Notebook 為網頁互動式執行環境，支援 Python，R，Julia　等多種程式撰寫。每次輸入程式的地方稱為儲存格 (Cell)。使用方式為先按 [Esc]，此時 cell 左側會顯示藍色，配合以下快速鍵。

快速鍵	功能
x	刪除當前選擇的cell
a	在當前選擇的上方新增一個cell
b	在當前選擇的下方新增一個cell
Shift + Enter	執行當前的cell並且選到下一個cell
Ctrl + Enter	執行當前cell
M	切換至 markdown 模式，可以看到紅色框框內容從code變成markdown
Y	切換至 code 模式，可以看到左側有程式碼編號

Jupyter Notebook 插入圖片

In [3]:

```
# <img src="img/matplotlib_01.png" width="600">
# ![anaconda](img/matplotlib_01.png)
```

Jupyter Notebook 加入數學式

方法1: 將LaTeX代碼括在美元符號 $ ... $中，表示在文字中顯示數學式。

參考資料 https://www.math.ubc.ca/~pwalls/math-python/jupyter/latex/ (https://www.math.ubc.ca/~pwalls/math-python/jupyter/latex/)

參考資料 https://en.wikibooks.org/wiki/LaTeX/Mathematics (https://en.wikibooks.org/wiki/LaTeX/Mathematics)

In [4]:

```
# 積分

# $\int_a^b f(x) = F(b) - F(a)$
```

$$\int_a^b f(x) = F(b) - F(a)$$

方法2: 將LaTeX代碼括在2個美元符號$$ ... &$中，表示在段落中顯示數學式。

In [5]:

```
# 極限

# $$f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x-a}$$
```

$$f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a}$$

In [6]:

```
# 常用數學式
```

Syntax	Output
`$x_n$`	x_n
`$x^2$`	x^2
`$\infty$`	∞
`$\frac{a}{b}$`	$\frac{a}{b}$
`$\partial$`	∂
`$\alpha$`	α
`$\beta$`	β
`$\gamma$`	γ
`$\Gamma$`	Γ
`$\Delta$`	Δ
`$\sin$`	$\sin$
`$\cos$`	$\cos$
`$\tan$`	$\tan$

`$\sum_{n=0}^{\infty}$`	$\sum_{n=0}^{\infty}$
`$\prod_{n=0}^{\infty}$`	$\prod_{n=0}^{\infty}$
`$\int_a^b$`	$\int_a^b$
`$\lim_{x \to a}$`	$\lim_{x \to a}$
`$\mathrm{Hom}$`	Hom
`$\mathbf{v}$`	$\mathbf{v}$
`$\mathbb{Z}$`	$\mathbb{Z}$
`$\mathscr{L}$`	$\mathscr{L}$
`$\mathfrak{g}$`	$\mathfrak{g}$
`$\dots$`	$\dots$
`$\vdots$`	$\vdots$
`$\ddots$`	$\ddots$

矩陣表示

In [7]:

```
# $$\begin{matrix} a & b \\ c & d \end{matrix}$$
```

$$\begin{matrix} a & b \\ c & d \end{matrix}$$

矩陣加上**( )** 表示

In [8]:

```
# $$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

矩陣加上**[ ]** 表示

In [9]:

```
# $$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 0 & 1 \\ 0 & 2 & 4 \end{bmatrix}$$
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 0 & 1 \\ 0 & 2 & 4 \end{bmatrix}$$

In [10]:

```
# $$\left( \frac{p}{q} \right)$$
```

$$\left(\frac{p}{q} \right)$$

In [11]:

```
# $$\lim_{x \to a^-} f(x) = f(a) = \lim_{x \to a^+} f(x)$$
```

$$\lim_{x \to a^-} f(x) = f(a) = \lim_{x \to a^+} f(x)$$

MacLaurin Series

In [12]:

```
# $$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$
```

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

Jacobian Matrix

```
$$
\mathbf{J}
=
\frac{d \mathbf{f}}{d \mathbf{x}}
=
\left[ \frac{\partial \mathbf{f}}{\partial x_1}
\cdots \frac{\partial \mathbf{f}}{\partial x_n} \right]
=
\begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \cdots &
\frac{\partial f_1}{\partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial f_m}{\partial x_1} & \cdots &
\frac{\partial f_m}{\partial x_n}
\end{bmatrix}
$$
```

$$\mathbf{J} = \frac{d\mathbf{f}}{d\mathbf{x}} = \left[\frac{\partial \mathbf{f}}{\partial x_1} \cdots \frac{\partial \mathbf{f}}{\partial x_n} \right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

In [13]:

```
# $$\forall x \in X, \quad \exists y \leq \epsilon$$
```

$$\forall x \in X, \quad \exists y \leq \epsilon$$

希臘字母

In [14]:

```
# $$\alpha$$,

# $$\beta$$,

# $$\gamma$$,

# $$\Gamma$$,

# $$\pi$$,

# $$\Pi$$,

# $$\phi$$,

# $$\varphi$$,

# $$\mu$$,

# $$\Phi$$
```

$$\alpha$$

,

$$\beta$$

,

$$\gamma$$

,

$$\Gamma$$

,

$$\pi$$

,

$$\Pi$$

,

$$\phi$$

,

$$\varphi$$

,

$$\mu$$

,

$$\Phi$$

三角函數

In [15]:

```
# $$\cos (2\theta) = \cos^2 \theta - \sin^2 \theta$$
```

$$\cos(2\theta) = \cos^2 \theta - \sin^2 \theta$$

其他符號

In [16]:

```
# x \equiv a \pmod{b}
```

$$x \equiv a \pmod{b}$$

In [17]:

```
# $$k_{n+1} = n^2 + k_n^2 - k_{n-1}$$
```

$$k_{n+1} = n^2 + k_n^2 - k_{n-1}$$

In [18]:

```
# $$n^{22}$$
```

$$n^{22}$$

In [19]:

```
# $$f(n) = n^5 + 4n^2 + 2 |_{n=17}$$
```

$$f(n) = n^5 + 4n^2 + 2|_{n=17}$$

In [20]:

```
# $$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$
```

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

In [21]:

```
# $$\frac{\frac{1}{x}+\frac{1}{y}}{y-z}$$
```

$$\frac{\frac{1}{x}+\frac{1}{y}}{y-z}$$

In [22]:

```
# $x^\frac{1}{2}$ % no error
# $x^\sfrac{1}{2}$ % error
# $x^{\sfrac{1}{2}}$ % no error
```

In [23]:

```
# $$\begin{equation}
#   x = a_0 + \cfrac{1}{a_1
#           + \cfrac{1}{a_2
#           + \cfrac{1}{a_3 + \cfrac{1}{a_4} } } }
# \end{equation}$$
```

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}$$

In [24]:

```
# $$\sqrt{\frac{a}{b}}$$
```

$$\sqrt{\frac{a}{b}}$$

In [25]:

```
# $$\sqrt[n]{1+x+x^2+x^3+\dots+x^n}$$
```

$$\sqrt[n]{1 + x + x^2 + x^3 + \cdots + x^n}$$

In [26]:

```
# $$\frac{\mathrm d}{\mathrm d x} \big( k g(x) \big)$$
```

$$\frac{\mathrm{d}}{\mathrm{d}x}\left(kg(x)\right)$$

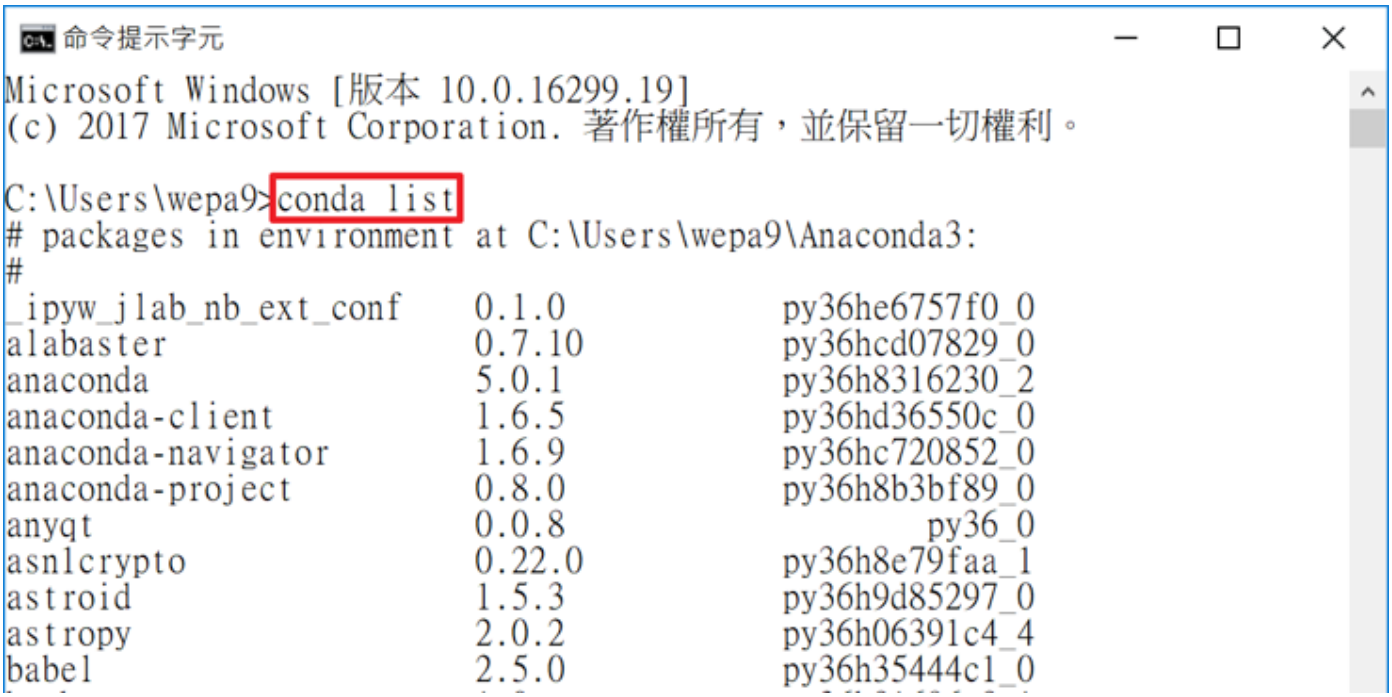
There are many other "big" commands which operate in a similar manner:

`\sum`	$\sum$	`\prod`	$\prod$	`\coprod`	$\coprod$	`\bigoplus`	$\bigoplus$	`\bigotimes`	$\bigotimes$
`\bigodot`	$\bigodot$	`\bigcup`	$\bigcup$	`\bigcap`	$\bigcap$	`\biguplus`	$\biguplus$	`\bigsqcup`	$\bigsqcup$
`\bigvee`	$\bigvee$	`\bigwedge`	$\bigwedge$	`\int`	$\int$	`\oint`	$\oint$	`\iint` [3]	$\iint$
`\iiint` [3]	$\iiint$	`\iiiint` [3]	$\iiiint$	`\idotsint` [3]	$\int\cdots\int$				

2.3 Anaconda 套件管理

顯示已安裝套件

conda list



尋找套件

conda search matplotlib

安裝模組

conda install 模組名稱

更新模組

conda update 模組名稱

範例: 更新 **anaconda** 模組

conda update anaconda

範例: 更新 **Spyder** 模組

conda update spyder

變更工作目錄

In [27]:

```
import os
```

In [28]:

```
os.getcwd() # 取得工作目錄
```

Out[28]:

```
'C:\\Users\\rwepa'
```

In [29]:

```
os.chdir("C:/pythondata") # 變更工作目錄, 注意 斜線的方向
```

In [30]:

```
os.getcwd() # 取得工作目錄
```

Out[30]:

```
'C:\\pythondata'
```

In [31]:

```
os.listdir(os.getcwd()) # 顯示檔案清單
```

Out[31]:

```
['.ipynb_checkpoints',
 '10m_pandas',
 '10m_pandas.zip',
 '2017年10月每小時網路流量.png',
 'data',
 'img',
 'mydata.csv',
 'mydata.h5',
 'mydata.xlsx',
 'random.plot.pdf',
 'random.plot.png',
 'regression_01.pdf',
 'Untitled.ipynb',
 'Untitled1.ipynb',
 'web_traffic.csv',
 '加入圖片.txt']
```

In [32]:

```
os.chdir("C:/Users/rwepa")
```

2.4 Spyder操作

Spyder 提供 Python 程式撰寫的整合開發環境 (Integrated Development Environment，簡稱IDE) 說明如下：

1. 最上方是功能表列

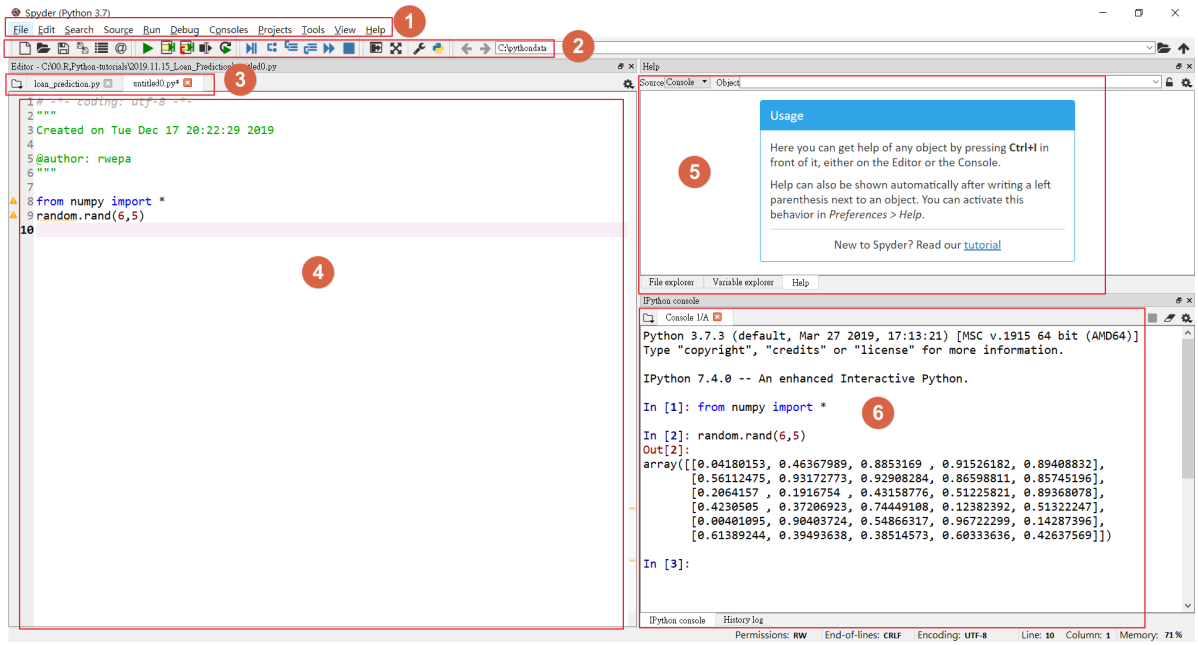
2. 中間是工具列

3. 視窗切換

4. 程式編輯區, 選取程式碼, 按 CTRL + ENTER 即可執行程式.

5. 檔案總管, 變數清單, 說明

6. 主控制台, 顯示執行結果



第3章 Python語法與流程控制

本章節從資料型別與運算子為開端, 包括以下內容:

- **3.1** 資料型別與運算子
- **3.2 NumPy**模組的使用
- **3.3 reshape** 應用
- **3.4** 離群值處理
- **3.5 if** 與 **for** 處理

3.1 資料型別與運算子

資料型別

- 布林 **bool,** 包括 **True, False**
- 整數 **int,** 例**: 123**
- 長整數 **long (**字尾加上**L**或**l)**
- 浮點數 **float,** 例**: 1.234, 1.23e17**
- 複數 **complex**
- 字串 **string**

資料型別範例

In [33]:

```
x = "育達科技大學"
type(x)
```

Out[33]:

```
str
```

In [34]:

```
x = 123
type(x)
```

Out[34]:

```
int
```

In [35]:

```
x = 123.456
type(x)
```

Out[35]:

```
float
```

In [36]:

```
x = complex(1, 2) # python 1+2j 相當於 1+2i
type(x)
```

Out[36]:

```
complex
```

In [37]:

```
x = ["apple", "banana", "cherry"]
type(x)
```

Out[37]:

```
list
```

In [38]:

```
x = ("apple", "banana", "cherry")
type(x)
```

Out[38]:

```
tuple
```

In [39]:

```
x = range(6)
type(x)
```

Out[39]:

```
range
```

In [40]:

```
x = {"name" : "John", "age" : 36}
type(x)
```

Out[40]:

```
dict
```

In [41]:

```
x = {"apple", "banana", "cherry"}
type(x)
```

Out[41]:

```
set
```

In [42]:

```
x = frozenset({"apple", "banana", "cherry"})
type(x)
```

Out[42]:

```
frozenset
```

In [43]:

```
x = True
type(x)
```

Out[43]:

```
bool
```

In [44]:

```
x = b"Hello"
type(x)
```

Out[44]:

```
bytes
```

In [45]:

```
x = bytearray(5)
type(x)
```

Out[45]:

```
bytearray
```

In [46]:

```
x = memoryview(bytes(5))
type(x)
```

Out[46]:

```
memoryview
```

強制轉換資料型別

In [47]:

```
str("Hello World")
```

Out[47]:

```
'Hello World'
```

In [48]:

```
int(123.789)
```

Out[48]:

```
123
```

In [49]:

```
float(20.567)
```

Out[49]:

```
20.567
```

In [50]:

```
complex(1j)
```

Out[50]:

```
1j
```

In [51]:

```
list(("apple", "banana", "cherry"))
```

Out[51]:

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

In [52]:

```
tuple(("apple", "banana", "cherry"))
```

Out[52]:

```
('apple', 'banana', 'cherry')
```

In [53]:

```
range(6)
```

Out[53]:

```
range(0, 6)
```

In [54]:

```
dict(name="John", age=36)
```

Out[54]:

```
{'name': 'John', 'age': 36}
```

In [55]:

```
set(("apple", "banana", "cherry"))
```

Out[55]:

```
{'apple', 'banana', 'cherry'}
```

In [56]:

```
frozenset(("apple", "banana", "cherry"))
```

Out[56]:

```
frozenset({'apple', 'banana', 'cherry'})
```

In [57]:

```
bool(5)
```

Out[57]:

```
True
```

In [58]:

```
bytes(5)
```

Out[58]:

```
b'\x00\x00\x00\x00\x00'
```

In [59]:

```
bytearray(5)
```

Out[59]:

```
bytearray(b'\x00\x00\x00\x00\x00')
```

In [60]:

```
memoryview(bytes(5))
```

Out[60]:

```
<memory at 0x00000242335ACF48>
```

布林資料型別

In [61]:

```
print(10 > 8)
print(10 == 8)
print(10 < 8)
```

```
True
False
False
```

註解

註解功能:

1. 註解可用於解釋 Python 程式碼。
2. 註解可用於使程式碼更具可讀性。
3. 註解可用於在測試代碼時暫時不執行某程式。

使用 # 表示註解

In [62]:

```
x = 5 # 設定初值
```

使用 """ """ 區塊註解

In [63]:

```
"""
這是區塊註解
註解超過1行
測試中
"""
print("Hello, World!")
```

```
Hello, World!
```

變數名稱

- 變數可以具有短名稱（如x和y）或更具描述性的名稱（如: age, customer_name, total_volume）。
- Python 變數命名規則：

1. 變數名稱必須以字母或下底線符號開頭
2. 變數名稱不能以數字開頭
3. 變數名稱只能包含字母,數字和下底線（A-z，0-9和_）
4. 變數名稱區分大小寫, 例: age, Age and AGE 是三個不同的變數
5. 雙下底線開頭並結尾的名稱已經由Python保留, 例: __init__

詳細編輯方式可參考 Google Python Style Guide http://google.github.io/styleguide/pyguide.html (http://google.github.io/styleguide/pyguide.html)

隨機樣本

In [64]:

```
import random
random.seed(10) # 輸入大於1的隨機種子, 如此每次結果皆相同
print(random.random()) # 回傳 0~1之間的隨機數值
```

```
0.5714025946899135
```

In [65]:

```
print(random.randrange(1,100)) # 傳回1~10 之暑的隨機整數
```

```
55
```

random 模組函數

方法	說明
seed()	Initialize the random number generator
getstate()	Returns the current internal state of the random number generator
setstate()	Restores the internal state of the random number generator
getrandbits()	Returns a number representing the random bits
randrange()	Returns a random number between the given range
randint()	Returns a random number between the given range
choice()	Returns a random element from the given sequence
choices()	Returns a list with a random selection from the given sequence
shuffle()	Takes a sequence and returns the sequence in a random order
sample()	Returns a given sample of a sequence
random()	Returns a random float number between 0 and 1
uniform()	Returns a random float number between two given parameters
triangular()	Returns a random float number between two given parameters, you can also set a mode parameter to specify the midpoint between the two other parameters
betavariate()	Returns a random float number between 0 and 1 based on the Beta distribution (used in statistics)
expovariate()	Returns a random float number between 0 and 1, or between 0 and -1 if the parameter is negative, based on the Exponential distribution (used in statistics)
gammavariate()	Returns a random float number between 0 and 1 based on the Gamma distribution (used in statistics)
gauss()	Returns a random float number between 0 and 1 based on the Gaussian distribution (used in probability theories)
lognormvariate()	Returns a random float number between 0 and 1 based on a log-normal distribution (used in probability theories)
normalvariate()	Returns a random float number between 0 and 1 based on the normal distribution (used in probability theories)

方法	說明
vonmisesvariate()	Returns a random float number between 0 and 1 based on the von Mises distribution (used in directional statistics)
paretovariate()	Returns a random float number between 0 and 1 based on the Pareto distribution (used in probability theories)
weibullvariate()	Returns a random float number between 0 and 1 based on the Weibull distribution (used in statistics)

指派運算子 **Assignment Operators**

運算子	範例	功能
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

In [66]:

```
# 指派運算子 +=
x = 5
print(x)
x += 3
print(x)
```

```
5
8
```

In [67]:

```
# 指派運算子 -=
x = 5
print(x)
x -= 3
print(x)
```

```
5
2
```

邏輯運算子 **Logical Operators**

運算子	功能	範例
==	等於 Equal	x == y
!=	不等於 Not equal	x != y
>	大於 Greater than	x > y
<	小於 Less than	x < y
>=	大於或等於 Greater than or equal to	x >= y
<=	小於或等於 Less than or equal to	x <= y

In [68]:

```
2 == 2.000
```

Out[68]:

```
True
```

In [69]:

```
3 != 2
```

Out[69]:

```
True
```

In [70]:

```
3 > 2
```

Out[70]:

```
True
```

In [71]:

```
1 < 3
```

Out[71]:

```
True
```

In [72]:

```
10 >= 10
```

Out[72]:

```
True
```

In [73]:

```
10 <= 55
```

Out[73]:

```
True
```

邏輯運算子 **Logical Operators**

運算子	功能	範例
and	如果二個都為真, 回傳 True	x < 5 and x < 10
or	如果有一個都為真, 回傳 True	x < 5 or x < 4
not	反運算	not(x < 5 and x < 10)

In [74]:

```
x < 5 and x < 10
```

Out[74]:

```
True
```

In [75]:

```
x < 5 or x < 4
```

Out[75]:

```
True
```

In [76]:

```
not(x < 5 and x < 10)
```

Out[76]:

```
False
```

會員運算子 **Membership Operators**

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

In [77]:

```
x = ["apple", "banana"]

print("banana" in x)
```

```
True
```

In [78]:

```
x = ["apple", "banana"]

print("apple" not in x)
```

```
False
```

常用運算子

- ** 次方, **2**3=8**
- * 乘法
- / 除法
- // 整除
- % 餘數
- + 加法
- - 減法
- | 或運算子 **OR**
- ^ 互斥運算子 **XOR**
- & **AND**運算子
- << 左移運算子
- >> 右移運算子

3.2 NumPy模組的使用

In [79]:

```
import numpy as np
a = np.array([0,1,2,3,4,5])
a
```

Out[79]:

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

In [80]:

```
a.ndim   # 1
```

Out[80]:

```
1
```

In [81]:

```
a.shape  # (6,)
```

Out[81]:

```
(6,)
```

In [82]:

```
# 建立副本, b之修改會影響a
b = a.reshape((3,2))
b
```

Out[82]:

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

In [83]:

```
b.ndim   # 2
```

Out[83]:

```
2
```

In [84]:

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

Out[84]:

```
(3, 2)
```

In [85]:

```
b[1][0] = 168
b
```

Out[85]:

```
array([[  0,   1],
       [168,   3],
       [  4,   5]])
```

In [86]:

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

Out[86]:

```
array([  0,   1, 168,   3,   4,   5])
```

In [87]:

```
c = a.reshape((3,2)).copy()
c
```

Out[87]:

```
array([[  0,   1],
       [168,   3],
       [  4,   5]])
```

In [88]:

```
c[0][0] = -999
```

In [89]:

```
c
```

Out[89]:

```
array([[-999,    1],
       [ 168,    3],
       [   4,    5]])
```

In [90]:

```
a # a物件沒有更改
```

Out[90]:

```
array([  0,   1, 168,   3,   4,   5])
```

3.3 reshape 應用

In [91]:

```
z = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
z
```

Out[91]:

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [92]:

```
z.reshape(-1) # -1: unknown dimension
# array([ 1,  2,  3, ..., 10, 11, 12])
```

Out[92]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [93]:

```
z.reshape(-1,1) # row -1: unknown , column 1
```

Out[93]:

```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10],
       [11],
       [12]])
```

In [94]:

```
z.reshape(-1, 2) # row -1: unknown , column 2
```

Out[94]:

```
array([[ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8],
       [ 9, 10],
       [11, 12]])
```

In [95]:

```
z.reshape(1,-1) # row 1 , column: unknown
```

Out[95]:

```
array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]])
```

In [96]:

```
z.reshape(2, -1) # row 2 , column: unknown
```

Out[96]:

```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12]])
```

In [97]:

```
z.reshape(3, -1) # row 3 , column: unknown
```

Out[97]:

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [98]:

```
# z.reshape(-1, -1) # ERROR
```

In [99]:

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

Out[99]:

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

In [100]:

```
a*2
```

Out[100]:

```
array([ 0,  2,  2,  4,  6, 10])
```

In [101]:

```
a**3 # 次方運算
```

Out[101]:

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

In [102]:

```
# indexing
a[np.array([1,3,5])]
a
```

Out[102]:

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

3.4 離群值處理

In [103]:

```
a = a**3
a
```

Out[103]:

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

In [104]:

```
a > 10
```

Out[104]:

```
array([False, False, False, False,  True,  True])
```

In [105]:

```
a[a > 10]
```

Out[105]:

```
array([ 27, 125], dtype=int32)
```

In [106]:

```
a[a > 10] = 10
```

In [107]:

```
a
```

Out[107]:

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

In [108]:

```
a.clip(0, 3) # 將a 範圍切換至 [0,3]
```

Out[108]:

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

In [109]:

```
# 處理 Na
x = np.array([1, 2, 3, np.NAN, 4])
x
```

Out[109]:

```
array([ 1.,  2.,  3., nan,  4.])
```

In [110]:

```
np.isnan(x) # R使用 is.na()函數
```

Out[110]:

```
array([False, False, False,  True, False])
```

In [111]:

```
x[~np.isnan(x)]
```

Out[111]:

```
array([1., 2., 3., 4.])
```

In [112]:

```
np.mean(x[~np.isnan(x)])
```

Out[112]:

```
2.5
```

In [113]:

```
np.mean(x) # nan
```

Out[113]:

```
nan
```

In [114]:

```
# 計算時間
import timeit
import numpy as np
```

In [115]:

```
normal_py_sec = timeit.timeit('sum(x*x for x in range(1000))', number=10000)
naive_np_sec = timeit.timeit('sum(na*na)', setup="import numpy as np; na=np.arange(1000)",
good_np_sec = timeit.timeit('na.dot(na)', setup="import numpy as np; na=np.arange(1000)", r
print("Normal Python: %f sec"%normal_py_sec)
print("Naive NumPy: %f sec"%naive_np_sec)
print("Good NumPy: %f sec(最快)"%good_np_sec)

# print "Hello World"    # python 2
print("Hello World")   # python 3
```

```
Normal Python: 1.169908 sec
Naive NumPy: 0.688709 sec
Good NumPy: 0.010764 sec(最快)
Hello World
```

3.5 if 與 for 處理

if 判斷式

In [116]:

```
a = 33
b = 200
if b > a:
    print("b 是大於 a")
```

```
b 是大於 a
```

elif 判斷式

In [117]:

```
x = 20
if x < 0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
     print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```

```
More
```

else 判斷式

In [118]:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

if 簡短用法

In [119]:

```
# 使用 :
if a > b: print("a is greater than b")
```

```
a is greater than b
```

if ... else 簡短用法

In [120]:

```
a = 90
b = 60
print("A") if a > b else print("B")
```

```
A
```

巢狀 if

In [121]:

```
x = 12

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

```
Above ten,
but not above 20.
```

pass

In [122]:

```
a = 33
b = 200

if b > a:
  pass
```

while 迴圈

In [123]:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

while 迴圈 + break

In [124]:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

```
1
2
3
```

while 迴圈 + continue

In [125]:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

```
1
2
4
5
6
```

while 迴圈 + else

In [126]:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

```
1
2
3
4
5
i is no longer less than 6
```

for 迴圈

In [127]:

```
# for
words = ['cat', 'window', 'defenestrate']
```

In [128]:

```
for w in words:
    print(w, len(w))
```

```
cat 3
window 6
defenestrate 12
```

In [129]:

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

```
0
1
2
3
4
```

for 迴圈 + break

In [130]:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
apple
banana
```

for 迴圈 + continue

In [131]:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

```
apple
cherry
```

巢狀 for 迴圈

In [132]:

```
adj = ["red", "small", "RWPEA"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, ",", y)
```

```
red , apple
red , banana
red , cherry
small , apple
small , banana
small , cherry
RWPEA , apple
RWPEA , banana
RWPEA , cherry
```

def 函數

In [133]:

```
def my_function(fname):
    print(fname + " 會員")

my_function("RWEPA")
my_function("李明昌")
my_function("ALAN")
```

```
RWEPA 會員
李明昌 會員
ALAN 會員
```

In [134]:

```
# 最大公因數
def gcd(m, n):
    if n == 0:
        return m
    else:
        return gcd(n, m % n)

print(gcd(60, 36)) # 顯示 12
```

```
12
```

Lambda 函數

Python 提供了一個簡易的 function define：lambda，用完即回收。可以實作出很簡單的 function (只處理一個運算式)。

lambda param1, param2, ... : expression

相當於使用 def

def fun(param1, param2, ...) : return expression

其中的 expression 不能放 assignment，也就是這一行指令不能放=等號。

In [135]:

```
def funsum(x, y, z):
    return x + y + z
funsum(1, 2, 3)
```

Out[135]:

```
6
```

In [136]:

```
func2 = lambda x,y,z : x+y+z
func2(1, 2, 3)
```

Out[136]:

```
6
```

def + Lambda 函數

In [137]:

```
def myfunc(n):
  return lambda a : a * n

mydoubler = myfunc(2)
mytripler = myfunc(3)

print(mydoubler(11))
print(mytripler(11))
```

```
22
33
```

create 建立物件

In [138]:

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

In [139]:

```
MyClass
```

Out[139]:

```
__main__.MyClass
```

In [140]:

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

```
5
```

__init__() 函數

In [141]:

```
# The self parameter is a reference to the current instance of the class, and is used to ac

class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

```
John
36
```

In [142]:

```
# 修改物件性質
p1.age = 40
```

In [143]:

```
# 刪除物件性質
del p1.age
```

In [144]:

```
print(p1.name)
```

```
John
```

In [145]:

```
# print(p1.age) # 已經沒有該成員, 會有 ERROR
```

In [146]:

```
# 刪除物件性質
del p1
```

模組

In [147]:

```
# mymodule.py

def greeting(name):
  print("Hello, " + name)
```

載入模組

import mymodule

mymodule.greeting("Jonathan")

日期

In [148]:

```
import datetime

x = datetime.datetime.now()
print(x)
```

```
2019-12-25 23:09:05.420603
```

In [149]:

```
print(x.year)
```

```
2019
```

In [150]:

```
print(x.strftime("%A"))
```

```
Wednesday
```

建立日期物件

In [151]:

```
import datetime

x = datetime.datetime(2019, 12, 19)

print(x)
```

```
2019-12-19 00:00:00
```

strftime 函數

In [152]:

```
x = datetime.datetime(2018, 6, 1)

print(x.strftime("%B"))
```

```
June
```

常用日期時間格式

格式	功能	範例
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	日 Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	月 Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	西元年 Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	08
%f	Microsecond 000000-999999	548513
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%x	Local version of date	12/31/18
%X	Local version of time	17:41:00
%%	A % character	%

JSON 資料

In [153]:

```
import json
```

In [154]:

```
# JSON 轉換至 Python:

# some JSON:
x =  '{ "name":"John", "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

print(y)

# the result is a Python dictionary:
print(y["age"])
```

```
{'name': 'John', 'age': 30, 'city': 'New York'}
30
```

In [155]:

```
# Python 轉換至 JSON:

# a Python object (dict):
x = {
  "name": "John",
  "age": 30,
  "city": "New York"
}

print(x)

# convert into JSON:
y = json.dumps(x)

# the result is a JSON string:
print(y)
```

```
{'name': 'John', 'age': 30, 'city': 'New York'}
{"name": "John", "age": 30, "city": "New York"}
```

Python 轉換至 JSON 對照表

Python	JSON
dict	Object
list	Array
tuple	Array
str	String
int	Number
float	Number
True	true
False	false
None	null

In [156]:

```
import json

x = {
  "name": "John",
  "age": 30,
  "married": True,
  "divorced": False,
  "children": ("Ann","Billy"),
  "pets": None,
  "cars": [
    {"model": "BMW 230", "mpg": 27.5},
    {"model": "Ford Edge", "mpg": 24.1}
  ]
}

print(json.dumps(x))
```

```
{"name": "John", "age": 30, "married": true, "divorced": false, "children":
["Ann", "Billy"], "pets": null, "cars": [{"model": "BMW 230", "mpg": 27.5},
{"model": "Ford Edge", "mpg": 24.1}]}
```

numpy 統計運算

In [157]:

```
import numpy

value = [99,86,87,88,111,86,103,87,94,78,77,85,86]

# 平均值 Mean

x = numpy.mean(value)

print(x)
```

```
89.76923076923077
```

In [158]:

```
# 中位數 Median

x = numpy.median(value)

print(x)
```

```
87.0
```

In [159]:

```
# 眾數 Mode

from scipy import stats

x = stats.mode(value)

print(x)
```

```
ModeResult(mode=array([86]), count=array([3]))
```

In [160]:

```
# 標準差 Standard Deviation

x = numpy.std(value)

print(x)
```

```
9.258292301032677
```

In [161]:

```
# 變異數 Variance

x = numpy.var(value)

print(x)
```

```
85.71597633136093
```

In [162]:

```
# 百分位數 Percentiles

ages = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]

x = numpy.percentile(ages, 75)

print(x)
```

```
43.0
```

In [163]:

```
x = numpy.percentile(ages, 90)

print(x)
```

```
61.0
```

正規表示式 **re**模組

re 模組（**R**egular **E**xpression 正規表示式）提供各種正規表示式的匹配操作，主要進行字串處理，使用這一內嵌於 Python 的語言工具，儘管不能滿足所有複雜的匹配情況，但足夠在絕大多數情況下能夠有效地對複雜字串的分析並提取出相關資訊。Python 會將正規表示式轉化為位元組碼，利用 C 語言的匹配引擎進行深度優先的匹

配。

re模組網頁: https://docs.python.org/3/library/re.html (https://docs.python.org/3/library/re.html)

search 與 match

兩種不同的操作：

re.match( ) 僅在字串的開頭檢查匹配項

re.search( ) 在字串的任何位置檢查匹配項（這是Perl的預設操作）

re.findall( ) 找出字串中含有 pattern 的所有字串

使用方式

import re

re.search(pattern, string)

In [164]:

```
# 載入模組
import re

# 參考資料 https://zh.wikibooks.org/zh-tw/Python/Regular_Expression
```

In [165]:

```
re.match("RWEPA", "RWEPA")    # match
```

Out[165]:

```
<re.Match object; span=(0, 5), match='RWEPA'>
```

In [166]:

```
re.match("c", "abcdef")    # No match
```

In [167]:

```
re.search("c", "abcdef")   # Match, 只要有包括 c 的字串即可
```

Out[167]:

```
<re.Match object; span=(2, 3), match='c'>
```

In [168]:

```
# . 表示任意字元，如果指定了 DOTALL 的標識，就表示包括新行在內的所有字元。

re.match('c.f', 'abcdefg')
```

In [169]:

```
# ^ 表示字串開頭

re.match("c", "abcdef")    # No match
```

In [170]:

```
re.match("^c", "abcdef")    # No match
```

In [171]:

```
re.match("^a", "abcdef")    # match
```

Out[171]:

```
<re.Match object; span=(0, 1), match='a'>
```

In [172]:

```
re.search("^c", "abcdef")  # No match
```

In [173]:

```
re.search("^a", "abcdef")  # Match
```

Out[173]:

```
<re.Match object; span=(0, 1), match='a'>
```

In [174]:

```
# $ 表示字串結尾

re.match("f$", "abcdef")    # No match
```

In [175]:

```
re.search("f$", "abcdef")    # match
```

Out[175]:

```
<re.Match object; span=(5, 6), match='f'>
```

In [176]:

```
# * 表示匹配前一個字元重複 0 次到無限次
# ’ ‘表示匹配前一個字元重複 1次到無限次
# ’?’表示匹配前一個字元重複 0 次到 1次

re.search('abc*', 'abcccd') # abccc
```

Out[176]:

```
<re.Match object; span=(0, 5), match='abccc'>
```

In [177]:

```
re.search('abc ', 'abcccabcd')
```

In [178]:

```
re.search('abc?', 'abcccd')
```

Out[178]:

```
<re.Match object; span=(0, 3), match='abc'>
```

In [179]:

```
# {m} 符合前一個字元 m 次

re.findall('c{3}', 'abcccd abcccgh')
```

Out[179]:

```
['ccc', 'ccc']
```

In [180]:

```
# {m,n} 符合前一個字元 m 到 n 次 (包括 m與n)

re.findall('a{1,2}b{2,3}', 'abbcccad defababcabbbccccgh')
```

Out[180]:

```
['abb', 'abbb']
```

In [181]:

```
# [] 表示任何一個字符集,所有特殊字元在其都失去特殊意義,只有： ^  -  ]  \ 為特殊含義

re.findall('a[bc]', 'abbcccad abc gfacd') # 找出 ab 或 ac
```

Out[181]:

```
['ab', 'ab', 'ac']
```

In [182]:

```
# 練習抓abbbbc, bc

test_string = 'find abbbbc, bc, skip c, acc'

pattern = 'a*b+c'

ans=re.findall(pattern,test_string)

print(ans)

# 參考資料 https://ithelp.ithome.com.tw/articles/10194954
```

```
['abbbbc', 'bc']
```

In [183]:

```
# 練習找數字

test_string = '12 Drummers Drumming, 11 Pipers Piping, 10 Lords a Leaping'

pattern = '[0-9]+'

ans=re.findall(pattern,test_string)

print(ans)
```

```
['12', '11', '10']
```

In [184]:

```
# 練習找文字

test_string = 'find: can, man, fan, skip: dan, ran, pan'

pattern = '[cmf]an'

ans=re.findall(pattern,test_string)

print(ans)
```

```
['can', 'man', 'fan']
```

In [185]:

```
# 練習跳脫符號

test_string = 'find: 591., dot., yes., skip: non!'

pattern = '.{3}\.'

ans=re.findall(pattern,test_string)

print(ans)
```

```
['591.', 'dot.', 'yes.']
```

In [186]:

```
# 條件式搜尋

test_string = 'find: I love cats, I love dogs, skip: I love logs, I love cogs'

pattern = 'I love cats|I love dogs'

ans=re.findall(pattern,test_string)

print(ans)
```

```
['I love cats', 'I love dogs']
```

re.search 函數

In [187]:

```
if re.search("H+","Hello"):
    print("找到H")
```

找到H

In [188]:

```
if re.search("A+","Hello"):
    print("找到A")
else:
    print("找不到A")
```

找不到A

re.match 函數

找出符合的字串，如果要找完全符合的字串可以在最後加 "$"

In [189]:

```
if not re.match("ell.","Hello"): # "."表示任合字元
    print(2)  # The beginning of the string has to match
```

2

In [190]:

```
if re.match(".el","Hello"):
    print(3)
```

3

In [191]:

```
if re.match("he..o","Hello",re.I):
    print(4)  # Case-insensitive match
```

4

In [192]:

```
re.match("wo$", "HELLO WORLD")
```

In [193]:

```
re.match("wo$", "Hello World")
```

In [194]:

```
re.sub("l+", "l", "Hello") # Prints "Helo"; replacement substitution
```

Out[194]:

```
'Helo'
```

In [195]:

```
re.sub(r"(.*)\1", r"\1", "HeyHey")    # Prints "Hey"; backreference
```

Out[195]:

```
'Hey'
```

In [196]:

```
re.sub("EY", "ey", "HEy", flags=re.I) # Prints "Hey"; case-insensitive sub
```

Out[196]:

```
'Hey'
```

In [197]:

```
re.sub(r"(?i)EY", r"ey", "HEy")       # Prints "Hey"; case-insensitive sub
```

Out[197]:

```
'Hey'
```

In [198]:

```
for match in re.findall("l+.", "Hello Dolly"):
  print(match)                                # Prints "llo" and then "lly"
```

```
llo
lly
```

In [199]:

```
for match in re.findall("e(l+.)", "Hello Dolly"):
  print(match)                                # Prints "llo"; match picks group 1
```

```
llo
```

In [200]:

```
for match in re.findall("(l+)(.)", "Hello Dolly"):
  print(match[0], match[1])                  # The groups end up as items in a tuple
```

```
ll o
ll y
```

In [201]:

```
re.match("(Hello|Hi) (Tom|Thom)","Hello Tom Bombadil")
```

Out[201]:

```
<re.Match object; span=(0, 9), match='Hello Tom'>
```

第4章 資料型別與資料處理

本章節從 序列資料物件介紹為開端，包括以下內容：

- **4.1 Tuple** 序列
- **4.2 List** 串列
- **4.3 Set** 集合
- **4.4 Dictionaries** 字典

4.1 Tuple 序列

- 基本型態 – 固定有序列(Tuples)：特色與list類似。
- 最大的不同tuple是一種唯讀且不可變更的資料結構不可取代tuple中的任意一個元素，因為它是唯讀不可變更的。

In [202]:

```
f = (2,3,4,5) # A tuple of integers
f
```

Out[202]:

```
(2, 3, 4, 5)
```

In [203]:

```
g = ()
g
```

Out[203]:

```
()
```

In [204]:

```
h = (2, [3,4], (10,11,12))  # A tuple containing mixed objects
h
```

Out[204]:

```
(2, [3, 4], (10, 11, 12))
```

In [205]:

```
# tuple operations
x = f[1] # Element access. x = 3
x
```

Out[205]:

```
3
```

In [206]:

```
y = f[1:3]  # Slices. y = (3,4)
y
```

Out[206]:

```
(3, 4)
```

In [207]:

```
z = h[1][1]     # Nesting. z = 4
z
```

Out[207]:

```
4
```

In [208]:

```
personal = ('Hannah',14,5*12+6)
personal
```

Out[208]:

```
('Hannah', 14, 66)
```

In [209]:

```
singleton = ("hello",)
singleton
```

Out[209]:

```
('hello',)
```

In [210]:

```
# Tuple Operations
("chapter",8) + ("strings","tuples","lists")
```

Out[210]:

```
('chapter', 8, 'strings', 'tuples', 'lists')
```

In [211]:

```
2*(3,"blind","mice")
```

Out[211]:

```
(3, 'blind', 'mice', 3, 'blind', 'mice')
```

In [212]:

```
# single format: tuple[index]
# index : 0  ~  len(tuple)-1
# index: -len(tuple)  ~  -1
f
```

Out[212]:

```
(2, 3, 4, 5)
```

In [213]:

```
f[0]
```

Out[213]:

```
2
```

In [214]:

```
f[-1] # 回傳最後一筆資料
```

Out[214]:

```
5
```

In [215]:

```
f[-2] # 回傳最後第2筆資料
```

Out[215]:

```
4
```

In [216]:

```
f[len(f)-1]
```

Out[216]:

```
5
```

In [217]:

```
# slice format: tuple [start:end ]. Items from start to (end -1)
t=((1,2), (2,"Hi"), (3,"RWEPA"), 2+3j, 6E23)
t
```

Out[217]:

```
((1, 2), (2, 'Hi'), (3, 'RWEPA'), (2+3j), 6e+23)
```

In [218]:

```
t[2]
```

Out[218]:

```
(3, 'RWEPA')
```

In [219]:

```
t[:3]
```

Out[219]:

```
((1, 2), (2, 'Hi'), (3, 'RWEPA'))
```

In [220]:

```
t[3:]
```

Out[220]:

```
((2+3j), 6e+23)
```

In [221]:

```
t[-1]
```

Out[221]:

```
6e+23
```

In [222]:

```
t[-3:]
# Tuple Comparison Operations
# standard comparisons ‘<’, ‘<=’, ‘>’, ‘>=’, ‘==’, ‘!=’, in, not in
```

Out[222]:

```
((3, 'RWEPA'), (2+3j), 6e+23)
```

4.2 List 串列

In [223]:

```
# 任意物件的串列
# 可以修改其內容
a = [2, 3, 4]            # A list of integer
```

In [224]:

```
b = [2, 7, 3.5, "Hello"] # A mixed list
b
```

Out[224]:

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

In [225]:

```
c = []                   # An empty list
c
```

Out[225]:

```
[]
```

In [226]:

```
d = [2, [a, b]]          # A list containing a list
d
```

Out[226]:

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

In [227]:

```
d[0]
```

Out[227]:

```
2
```

In [228]:

```
d[1]
```

Out[228]:

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

In [229]:

```
e = a + b # Join two lists
e
```

Out[229]:

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

In [230]:

```
#串列的操作
x = a[1] # Get 2nd element (0 is first)
x
```

Out[230]:

```
3
```

In [231]:

```
y = b[1:3]     # Return a sub-list
y
```

Out[231]:

```
[7, 3.5]
```

In [232]:

```
z = d[1][0][2] # Nested lists
z
```

Out[232]:

```
4
```

In [233]:

```
b[0] = 42 # Change an element
b
```

Out[233]:

```
[42, 7, 3.5, 'Hello']
```

4.3 Set 集合

In [234]:

```
# 集合與字典相似, 但字典沒有key,只有值
a = set() # An empty set
type(a)
```

Out[234]:

```
set
```

In [235]:

```
b = {"台北市", "新北市", "桃園市", "台中市", "台北市", "新北市", "高雄市"}
b # {'台中市', '台北市', '新北市', '桃園市', '高雄市'}
```

Out[235]:

```
{'台中市', '台北市', '新北市', '桃園市', '高雄市'}
```

In [236]:

```
# 集合運算
x = {1,2,3,4,5}
y = {1,3,5,7}
```

In [237]:

```
x
```

Out[237]:

```
{1, 2, 3, 4, 5}
```

In [238]:

```
y
```

Out[238]:

```
{1, 3, 5, 7}
```

In [239]:

```
x & y # {1, 3, 5}
```

Out[239]:

```
{1, 3, 5}
```

In [240]:

```
x | y # {1, 2, 3, 4, 5, 7}
```

Out[240]:

```
{1, 2, 3, 4, 5, 7}
```

In [241]:

```
x ^ y # {2, 4, 7}
```

Out[241]:

```
{2, 4, 7}
```

4.4 Dictionaries 字典

In [242]:

```
a = {}  # An empty dictionary
type(a) # dict
```

Out[242]:

```
dict
```

In [243]:

```
b = {'x': 3, "y": 4}
b
```

Out[243]:

```
{'x': 3, 'y': 4}
```

In [244]:

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

Out[244]:

```
{'uid': 168, 'login': 'marvelous', 'name': 'Alan Lee'}
```

In [245]:

```
u = c["uid"] # Get an element
u
```

Out[245]:

```
168
```

In [246]:

```
c["shell"] = "/bin/sh" # Add an element
c
```

Out[246]:

```
{'uid': 168, 'login': 'marvelous', 'name': 'Alan Lee', 'shell': '/bin/sh'}
```

In [247]:

```
# if c.has_key("directory"):    # Check for presence of a member 適合 v2.x
#     d = c["directory"]
# else:
#    d = None
# d
```

In [248]:

```
if "directory" in c:   # v3.x 直接使用 in
    d = c["directory"]
else:
    d = None
d
```

In [249]:

```
if "uid" in c:   # v3.x 直接使用 in
    d = c["uid"]
else:
    d = None
d
```

Out[249]:

```
168
```

In [250]:

```
d1 = c.get("directory", None) # 較簡潔
d1
```

In [251]:

```
d2 = c.get("uid", None) # 較簡潔
d2
```

Out[251]:

```
168
```

第5章 檔案匯入與匯出

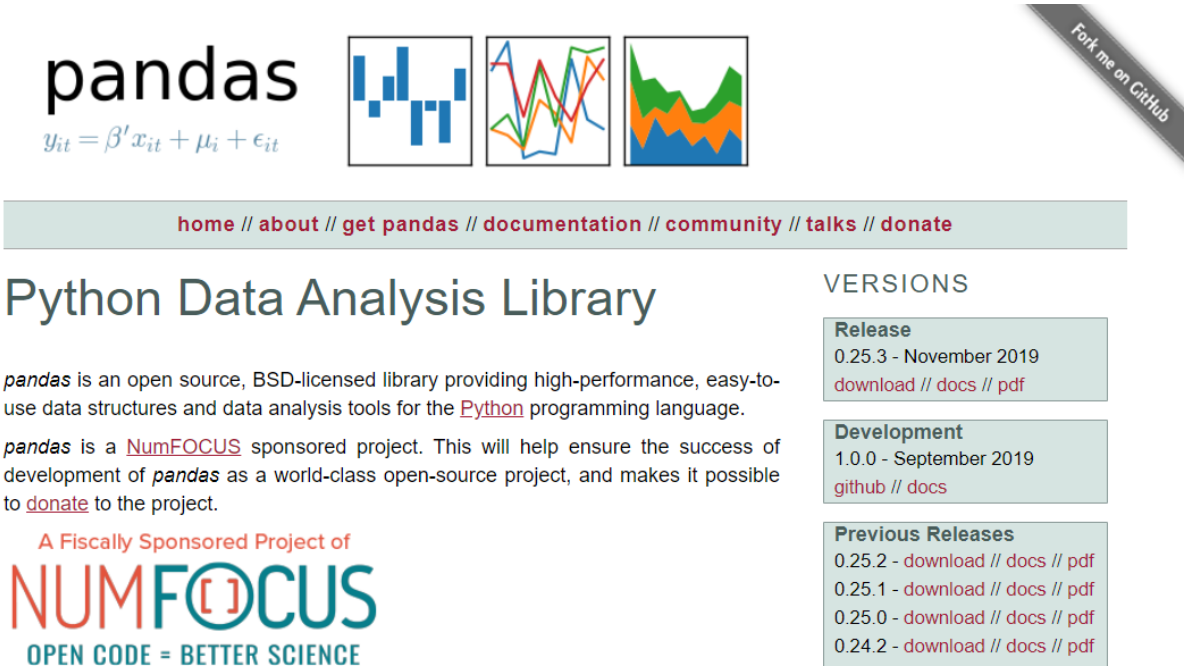
本章節從認識 **pandas** 模組為開端，包括以下內容：

- **5.1** 認識 **pandas** 模組
- **5.2** 資料輸入/輸出

5.1 認識 pandas 模組

pandas 取名自 pan(el)-da(ta)-s, pandas 提供資料讀取，資料整理，統計分析，繪圖等功能。

官方網站 https://pandas.pydata.org/ (https://pandas.pydata.org/) 參考下圖所示。



套件主要提供的二個資料結構: DataFrame 與 Series. 以下說明 pandas 的使用方式.

Source : https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html (https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html)

In [252]:

```
# 載入3大套件 (pandas, numpy, matplotlib)

import pandas as pd # Python Data Analysis Library

import numpy as np # Python Scientific Computing Library

import matplotlib.pyplot as plt # Python 2D Plotting Library
```

Object Creation 建立物件

In [253]:

```
# 使用串列(List)建立 序列(Series) 物件, 序列包括指標(Index) 與值(Value), 指標採用預設整數型態指
s = pd.Series([1,3,5,np.nan,6,8])
s
```

Out[253]:

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

In [254]:

```
type(s)
```

Out[254]:

```
pandas.core.series.Series
```

In [255]:

```
# 使用陣列(Array) 建立資料框(DataFrame)
dates = pd.date_range('20191101', periods=6) # 日期指標
dates
```

Out[255]:

```
DatetimeIndex(['2019-11-01', '2019-11-02', '2019-11-03', '2019-11-04',
               '2019-11-05', '2019-11-06'],
              dtype='datetime64[ns]', freq='D')
```

In [256]:

```
type(dates)
```

Out[256]:

```
pandas.core.indexes.datetimes.DatetimeIndex
```

In [257]:

```
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD')) # 文字欄位名稱
df
```

Out[257]:

	A	B	C	D
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840
2019-11-02	-0.279166	0.143542	0.375640	0.800522
2019-11-03	-1.615721	0.369622	1.644659	-0.304645
2019-11-04	0.345833	0.889531	-0.901462	-1.010893
2019-11-05	-0.232098	0.531953	-0.026177	-1.119629
2019-11-06	0.566963	-0.394570	0.550336	-1.308769

In [258]:

```
# 使用字典建立資料框 DataFrame
df2 = pd.DataFrame({ 'A' : 1.,
    'B' : pd.Timestamp('20190101'),
    'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
    'D' : np.array([3] * 4,dtype='int32'),
    'E' : pd.Categorical(["test","train","test","train"]),
    'F' : 'foo' })
df2
```

Out[258]:

	A	B	C	D	E	F
0	1.0	2019-01-01	1.0	3	test	foo
1	1.0	2019-01-01	1.0	3	train	foo
2	1.0	2019-01-01	1.0	3	test	foo
3	1.0	2019-01-01	1.0	3	train	foo

In [259]:

```
# dtypes: 表示資料型態
df2.dtypes # df2. 按 [Tab] 按鈕
```

Out[259]:

```
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

Viewing Data 資料檢視

In [260]:

```
df
```

Out[260]:

	A	B	C	D
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840
2019-11-02	-0.279166	0.143542	0.375640	0.800522
2019-11-03	-1.615721	0.369622	1.644659	-0.304645
2019-11-04	0.345833	0.889531	-0.901462	-1.010893
2019-11-05	-0.232098	0.531953	-0.026177	-1.119629
2019-11-06	0.566963	-0.394570	0.550336	-1.308769

In [261]:

```
# 檢視前幾筆資料, 後幾筆資料,
# head 顯示前 5 筆資料, 此功能與 R 顯示 6 筆不相同.
```

In [262]:

```
df.head()
```

Out[262]:

	A	B	C	D
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840
2019-11-02	-0.279166	0.143542	0.375640	0.800522
2019-11-03	-1.615721	0.369622	1.644659	-0.304645
2019-11-04	0.345833	0.889531	-0.901462	-1.010893
2019-11-05	-0.232098	0.531953	-0.026177	-1.119629

In [263]:

```
df.head(3)
```

Out[263]:

	A	B	C	D
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840
2019-11-02	-0.279166	0.143542	0.375640	0.800522
2019-11-03	-1.615721	0.369622	1.644659	-0.304645

In [264]:

```
df.tail()
```

Out[264]:

	A	B	C	D
2019-11-02	-0.279166	0.143542	0.375640	0.800522
2019-11-03	-1.615721	0.369622	1.644659	-0.304645
2019-11-04	0.345833	0.889531	-0.901462	-1.010893
2019-11-05	-0.232098	0.531953	-0.026177	-1.119629
2019-11-06	0.566963	-0.394570	0.550336	-1.308769

In [265]:

```
# 顯示指標(index)
# 欄名稱(columns)
# 資料值(values)
```

In [266]:

```
df.index
```

Out[266]:

```
DatetimeIndex(['2019-11-01', '2019-11-02', '2019-11-03', '2019-11-04',
               '2019-11-05', '2019-11-06'],
              dtype='datetime64[ns]', freq='D')
```

In [267]:

```
df.columns
```

Out[267]:

```
Index(['A', 'B', 'C', 'D'], dtype='object')
```

In [268]:

```
df.values
```

Out[268]:

```
array([[ 0.6602548 , -0.24468033, -1.0516002 , -1.13383992],
       [-0.27916631,  0.14354235,  0.37564014,  0.80052168],
       [-1.61572136,  0.36962166,  1.64465905, -0.30464495],
       [ 0.34583345,  0.88953124, -0.9014616 , -1.01089292],
       [-0.23209763,  0.53195342, -0.02617676, -1.11962945],
       [ 0.5669633 , -0.3945701 ,  0.55033552, -1.30876916]])
```

統計分析

In [269]:

```
# describe 統計摘要 statistic summary
# count 個數
# mean 平均值
# std  標準差 standard deviation, 一般希望愈小愈好
# min  最小值
# 25%  25百分位數
# 50%  50百分位數, 中位數 median
# 75%  75百分位數 (quantile)
# max  最大值
```

In [270]:

```
df.describe()
```

Out[270]:

	A	B	C	D
count	6.000000	6.000000	6.000000	6.000000
mean	-0.092322	0.215900	0.098566	-0.679542
std	0.844772	0.483186	1.000976	0.804602
min	-1.615721	-0.394570	-1.051600	-1.308769
25%	-0.267399	-0.147625	-0.682640	-1.130287
50%	0.056868	0.256582	0.174732	-1.065261
75%	0.511681	0.491370	0.506662	-0.481207
max	0.660255	0.889531	1.644659	0.800522

In [271]:

```
# T 資料轉置, 類似將原本長資料 (Long data), 轉換為寬資料 (Wide data)
# 資料轉置
# | 1 2 3 4|
# | 5 6 7 8|
# 轉換為
# | 1 5|
# | 2 6|
# | 3 7|
# | 4 8|
df.T
```

Out[271]:

	2019-11-01	2019-11-02	2019-11-03	2019-11-04	2019-11-05	2019-11-06
A	0.660255	-0.279166	-1.615721	0.345833	-0.232098	0.566963
B	-0.244680	0.143542	0.369622	0.889531	0.531953	-0.394570
C	-1.051600	0.375640	1.644659	-0.901462	-0.026177	0.550336
D	-1.133840	0.800522	-0.304645	-1.010893	-1.119629	-1.308769

In [272]:

```
# axis為排序的軸，0表示 rows index(列指標)，1表示columns index(行指標)，
# 當對數據 "列" 進行排序時，axis必須設置為0.
# df.sort(["A"]) 新版不支援 sort, 改用 sort_values 或 sort_index
```

In [273]:

```
# ascending =FALSE, 即遞增是FALSE, 表示遞減是TRUE
df.sort_index(axis=1, ascending=False)
```

Out[273]:

	D	C	B	A
2019-11-01	-1.133840	-1.051600	-0.244680	0.660255
2019-11-02	0.800522	0.375640	0.143542	-0.279166
2019-11-03	-0.304645	1.644659	0.369622	-1.615721
2019-11-04	-1.010893	-0.901462	0.889531	0.345833
2019-11-05	-1.119629	-0.026177	0.531953	-0.232098
2019-11-06	-1.308769	0.550336	-0.394570	0.566963

In [274]:

```
# 依照 B 欄大小, 由小至大排序
df.sort_values(by='B')
```

Out[274]:

	A	B	C	D
2019-11-06	0.566963	-0.394570	0.550336	-1.308769
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840
2019-11-02	-0.279166	0.143542	0.375640	0.800522
2019-11-03	-1.615721	0.369622	1.644659	-0.304645
2019-11-05	-0.232098	0.531953	-0.026177	-1.119629
2019-11-04	0.345833	0.889531	-0.901462	-1.010893

Selection 資料選取 .at, .iat, .loc, .iloc

In [275]:

```
# 選取行
df['A']
```

Out[275]:

```
2019-11-01    0.660255
2019-11-02   -0.279166
2019-11-03   -1.615721
2019-11-04    0.345833
2019-11-05   -0.232098
2019-11-06    0.566963
Freq: D, Name: A, dtype: float64
```

In [276]:

```
df.A # 與 df['A'] 相同
```

Out[276]:

```
2019-11-01    0.660255
2019-11-02   -0.279166
2019-11-03   -1.615721
2019-11-04    0.345833
2019-11-05   -0.232098
2019-11-06    0.566963
Freq: D, Name: A, dtype: float64
```

In [277]:

```
# 選取列, 此功能與 R 不同, df[1:4] 表示選取第1至第4行
df[0:4]
```

Out[277]:

	A	B	C	D
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840
2019-11-02	-0.279166	0.143542	0.375640	0.800522
2019-11-03	-1.615721	0.369622	1.644659	-0.304645
2019-11-04	0.345833	0.889531	-0.901462	-1.010893

In [278]:

```
df.loc[dates[0]]
```

Out[278]:

```
A    0.660255
B   -0.244680
C   -1.051600
D   -1.133840
Name: 2019-11-01 00:00:00, dtype: float64
```

In [279]:

```
# Selecting on a multi-axis by label 選取多軸(列,行)
# 如果列的位置是空白, 表示所有列皆選取.
```

In [280]:

```
df.loc[:, ['A','B']]
```

Out[280]:

	A	B
2019-11-01	0.660255	-0.244680
2019-11-02	-0.279166	0.143542
2019-11-03	-1.615721	0.369622
2019-11-04	0.345833	0.889531
2019-11-05	-0.232098	0.531953
2019-11-06	0.566963	-0.394570

In [281]:

```
df.loc['20191102':'20191104',['A','B']]
```

Out[281]:

	A	B
2019-11-02	-0.279166	0.143542
2019-11-03	-1.615721	0.369622
2019-11-04	0.345833	0.889531

In [282]:

```
df.loc['20191102',['A','B']] # 回傳值已降為1維
```

Out[282]:

```
A   -0.279166
B    0.143542
Name: 2019-11-02 00:00:00, dtype: float64
```

In [283]:

```
df.loc[dates[0],'A']
```

Out[283]:

```
0.6602548040200551
```

In [284]:

```
df.at[dates[0],'A'] # .at 與 .loc 如果相同
```

Out[284]:

```
0.6602548040200551
```

In [285]:

```
df.iloc[3] # [3] 表示選取指標為3的列, 實際為第4列.
```

Out[285]:

```
A    0.345833
B    0.889531
C   -0.901462
D   -1.010893
Name: 2019-11-04 00:00:00, dtype: float64
```

In [286]:

```
# [第3列:第4列, 第0行:第1行] , 結束位置須減1.
# 例:5-1=4, 即選取列指標第 3, 4列, 即2019-11-04, 2019-11-05.
df.iloc[3:5,0:2]
```

Out[286]:

	A	B
2019-11-04	0.345833	0.889531
2019-11-05	-0.232098	0.531953

In [287]:

```
df.iloc[[1,2,4],[0,2]] # "," 表示不連續範圍
```

Out[287]:

	A	C
2019-11-02	-0.279166	0.375640
2019-11-03	-1.615721	1.644659
2019-11-05	-0.232098	-0.026177

In [288]:

```
df.iloc[1:3,:]
```

Out[288]:

	A	B	C	D
2019-11-02	-0.279166	0.143542	0.375640	0.800522
2019-11-03	-1.615721	0.369622	1.644659	-0.304645

In [289]:

```
df.iloc[:,1:3]
```

Out[289]:

	B	C
2019-11-01	-0.244680	-1.051600
2019-11-02	0.143542	0.375640
2019-11-03	0.369622	1.644659
2019-11-04	0.889531	-0.901462
2019-11-05	0.531953	-0.026177
2019-11-06	-0.394570	0.550336

In [290]:

```
df.iloc[1,1]
```

Out[290]:

```
0.14354234577317534
```

In [291]:

```
df.iat[1,1]
```

Out[291]:

```
0.14354234577317534
```

Boolean Indexing 邏輯值(條件式)資料選取

In [292]:

```
df[df.A > 0]
```

Out[292]:

	A	B	C	D
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840
2019-11-04	0.345833	0.889531	-0.901462	-1.010893
2019-11-06	0.566963	-0.394570	0.550336	-1.308769

In [293]:

```
df[df > 0]
```

Out[293]:

	A	B	C	D
2019-11-01	0.660255	NaN	NaN	NaN
2019-11-02	NaN	0.143542	0.375640	0.800522
2019-11-03	NaN	0.369622	1.644659	NaN
2019-11-04	0.345833	0.889531	NaN	NaN
2019-11-05	NaN	0.531953	NaN	NaN
2019-11-06	0.566963	NaN	0.550336	NaN

In [294]:

```
# 使用 .isin
df[df.index.isin(['2013-01-02', '2013-01-06'])]
```

Out[294]:

	A	B	C	D

In [295]:

```
df.A
```

Out[295]:

```
2019-11-01    0.660255
2019-11-02   -0.279166
2019-11-03   -1.615721
2019-11-04    0.345833
2019-11-05   -0.232098
2019-11-06    0.566963
Freq: D, Name: A, dtype: float64
```

In [296]:

```
df2 = df.copy()
df2['E'] = ['one', 'one','two','three','four','three']
df2
```

Out[296]:

	A	B	C	D	E
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840	one
2019-11-02	-0.279166	0.143542	0.375640	0.800522	one
2019-11-03	-1.615721	0.369622	1.644659	-0.304645	two
2019-11-04	0.345833	0.889531	-0.901462	-1.010893	three
2019-11-05	-0.232098	0.531953	-0.026177	-1.119629	four
2019-11-06	0.566963	-0.394570	0.550336	-1.308769	three

In [297]:

```
df2[df2['E'].isin(['two','four'])]
```

Out[297]:

	A	B	C	D	E
2019-11-03	-1.615721	0.369622	1.644659	-0.304645	two
2019-11-05	-0.232098	0.531953	-0.026177	-1.119629	four

Missing Data 遺漏值 NaN, R方式為NA

In [298]:

```
# [0:4] 表示index為 0,1,2,3
df1 = df.reindex(index=dates[0:4], columns=list(df.columns) + ['E'])
df1
```

Out[298]:

	A	B	C	D	E
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840	NaN
2019-11-02	-0.279166	0.143542	0.375640	0.800522	NaN
2019-11-03	-1.615721	0.369622	1.644659	-0.304645	NaN
2019-11-04	0.345833	0.889531	-0.901462	-1.010893	NaN

In [299]:

```
df1.loc[dates[0]:dates[1],'E'] = 1
df1
```

Out[299]:

	A	B	C	D	E
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840	1.0
2019-11-02	-0.279166	0.143542	0.375640	0.800522	1.0
2019-11-03	-1.615721	0.369622	1.644659	-0.304645	NaN
2019-11-04	0.345833	0.889531	-0.901462	-1.010893	NaN

In [300]:

```
# 删除列中包括 NaN
df1.dropna(how='any')
```

Out[300]:

	A	B	C	D	E
2019-11-01	0.660255	-0.244680	-1.05160	-1.133840	1.0
2019-11-02	-0.279166	0.143542	0.37564	0.800522	1.0

In [301]:

```
# 将遗漏值填入值
df1.fillna(value=999)
```

Out[301]:

	A	B	C	D	E
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840	1.0
2019-11-02	-0.279166	0.143542	0.375640	0.800522	1.0
2019-11-03	-1.615721	0.369622	1.644659	-0.304645	999.0
2019-11-04	0.345833	0.889531	-0.901462	-1.010893	999.0

In [302]:

```
# 判斷何者為NaN
pd.isnull(df1)
```

Out[302]:

	A	B	C	D	E
2019-11-01	False	False	False	False	False
2019-11-02	False	False	False	False	False
2019-11-03	False	False	False	False	True
2019-11-04	False	False	False	False	True

In [303]:

```
df
```

Out[303]:

	A	B	C	D
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840
2019-11-02	-0.279166	0.143542	0.375640	0.800522
2019-11-03	-1.615721	0.369622	1.644659	-0.304645
2019-11-04	0.345833	0.889531	-0.901462	-1.010893
2019-11-05	-0.232098	0.531953	-0.026177	-1.119629
2019-11-06	0.566963	-0.394570	0.550336	-1.308769

In [304]:

```
# 計算每行平均
df.mean()
```

Out[304]:

```
A   -0.092322
B    0.215900
C    0.098566
D   -0.679542
dtype: float64
```

In [305]:

```
# 計算每列平均
df.mean(1)
```

Out[305]:

```
2019-11-01   -0.442466
2019-11-02    0.260134
2019-11-03    0.023479
2019-11-04   -0.169247
2019-11-05   -0.211488
2019-11-06   -0.146510
Freq: D, dtype: float64
```

In [306]:

```
# Apply 將資料套用至函數
df.apply(np.cumsum)
```

Out[306]:

	A	B	C	D
2019-11-01	0.660255	-0.244680	-1.051600	-1.133840
2019-11-02	0.381088	-0.101138	-0.675960	-0.333318
2019-11-03	-1.234633	0.268484	0.968699	-0.637963
2019-11-04	-0.888799	1.158015	0.067237	-1.648856
2019-11-05	-1.120897	1.689968	0.041061	-2.768486
2019-11-06	-0.553934	1.295398	0.591396	-4.077255

Merge 合併

In [307]:

```
df = pd.DataFrame(np.random.randn(10, 4))
df
```

Out[307]:

	0	1	2	3
0	1.468262	-0.404749	1.720550	0.528660
1	-0.518356	0.215257	0.712262	1.263178
2	-0.697867	-0.508577	-0.818798	-0.337099
3	1.108294	-1.866329	-1.002680	1.003886
4	-1.053475	-0.479908	0.136076	1.051824
5	1.182651	1.175810	0.479253	-1.450316
6	-1.177922	-0.536663	0.586324	0.707825
7	2.253403	-0.278183	-1.472198	-0.404129
8	-0.702423	1.801728	1.506089	-0.469449
9	-0.812168	0.639542	0.034242	-1.811065

In [308]:

```
pieces = [df[:3], df[4:7], df[8:]]
pieces
```

Out[308]:

```
[          0         1         2         3
 0  1.468262 -0.404749  1.720550  0.528660
 1 -0.518356  0.215257  0.712262  1.263178
 2 -0.697867 -0.508577 -0.818798 -0.337099,
           0         1         2         3
 4 -1.053475 -0.479908  0.136076  1.051824
 5  1.182651  1.175810  0.479253 -1.450316
 6 -1.177922 -0.536663  0.586324  0.707825,
           0         1         2         3
 8 -0.702423  1.801728  1.506089 -0.469449
 9 -0.812168  0.639542  0.034242 -1.811065]
```

In [309]:

```
# 列合併, 類似 R的 rbind
pd.concat(pieces)
```

Out[309]:

	0	1	2	3
0	1.468262	-0.404749	1.720550	0.528660
1	-0.518356	0.215257	0.712262	1.263178
2	-0.697867	-0.508577	-0.818798	-0.337099
4	-1.053475	-0.479908	0.136076	1.051824
5	1.182651	1.175810	0.479253	-1.450316
6	-1.177922	-0.536663	0.586324	0.707825
8	-0.702423	1.801728	1.506089	-0.469449
9	-0.812168	0.639542	0.034242	-1.811065

Grouping 群組計算

In [310]:

```
df = pd.DataFrame({
    'A' : ['foo', 'bar', 'foo', 'bar', 'foo', 'bar', 'foo', 'foo'],
    'B' : ['one', 'one', 'two', 'three', 'two', 'two', 'one', 'three'],
    'C' : np.random.randn(8),
    'D' : np.random.randn(8)})
df
```

Out[310]:

	A	B	C	D
0	foo	one	0.097453	2.618800
1	bar	one	1.930037	0.578673
2	foo	two	-0.411213	0.372851
3	bar	three	-1.535785	0.586691
4	foo	two	2.183012	0.425635
5	bar	two	0.113904	0.829784
6	foo	one	-0.231005	-0.778486
7	foo	three	0.493072	0.021465

In [311]:

```
df.groupby('A').sum() # 類似 R- aggregate
```

Out[311]:

A	C	D
bar	0.508156	1.995148
foo	2.131320	2.660265

In [312]:

```
df.groupby(['A','B']).sum()
```

Out[312]:

A	B	C	D
	one	1.930037	0.578673
bar	three	-1.535785	0.586691
	two	0.113904	0.829784
	one	-0.133552	1.840314
foo	three	0.493072	0.021465
	two	1.771799	0.798485

Plotting 繪圖

DataFrame.plot 包括常用繪圖方式，以下列出常用函數：

參考資料: https://pandas.pydata.org/pandas-docs/stable/reference/frame.html#plotting (https://pandas.pydata.org/pandas-docs/stable/reference/frame.html#plotting)

1. 繪圖 DataFrame.plot([x, y, kind, ax, ….]) DataFrame plotting accessor and method
2. 長條圖 DataFrame.plot.bar(self[, x, y]) Vertical bar plot.
3. 水平長條圖 DataFrame.plot.barh(self[, x, y]) Make a horizontal bar plot.
4. 盒鬚圖 DataFrame.plot.box(self[, by]) Make a box plot of the DataFrame columns.
5. 盒鬚圖 DataFrame.boxplot(self[, column, by, ax, …]) Make a box plot from DataFrame columns.
6. 直方圖 DataFrame.plot.hist(self[, by, bins]) Draw one histogram of the DataFrame’s columns.
7. 直方圖 DataFrame.hist(data[, column, by, grid, …]) Make a histogram of the DataFrame’s.
8. 區域圖 DataFrame.plot.area(self[, x, y]) Draw a stacked area plot.
9. 密度圖 DataFrame.plot.density(self[, bw_method, ind]) Generate Kernel Density Estimate plot using Gaussian kernels.

10. 六邊箱圖 DataFrame.plot.hexbin(self, x, y[, C, …]) Generate a hexagonal binning plot.
11. 核密度圖 DataFrame.plot.kde(self[, bw_method, ind]) Generate Kernel Density Estimate plot using Gaussian kernels.
12. 線圖 DataFrame.plot.line(self[, x, y]) Plot Series or DataFrame as lines.
13. 圓形圖 DataFrame.plot.pie(self, **kwargs) Generate a pie plot.
14. 散佈圖 DataFrame.plot.scatter(self, x, y[, s, c]) Create a scatter plot with varying marker point size and color.

In [313]:

```
ts = pd.Series(np.random.randn(1000),
               index=pd.date_range('1/1/2000', periods=1000))
ts
```

Out[313]:

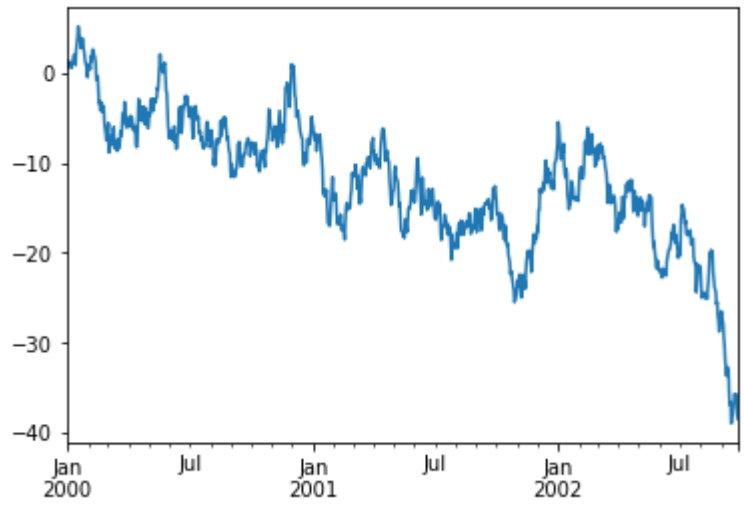
```
2000-01-01    0.512340
2000-01-02    0.838276
2000-01-03   -0.204208
2000-01-04   -0.048011
2000-01-05   -0.204895
                ...
2002-09-22   -0.187671
2002-09-23    0.152934
2002-09-24   -0.531606
2002-09-25   -1.883384
2002-09-26   -0.349447
Freq: D, Length: 1000, dtype: float64
```

In [314]:

```
ts = ts.cumsum()
ts.plot()
```

Out[314]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2423705e208>
```



In [315]:

```
df = pd.DataFrame(np.random.randn(1000, 4),
                  index=ts.index, columns=['A', 'B', 'C', 'D'])
```

In [316]:

```
df = df.cumsum()
```

In [317]:

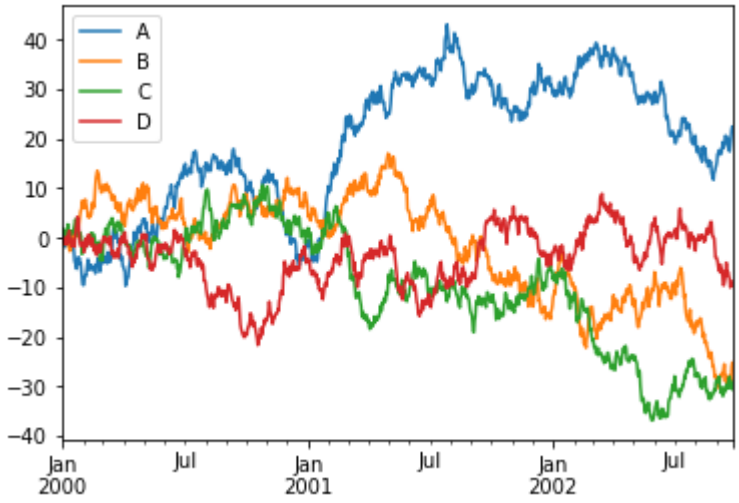
```
df.head()
```

Out[317]:

	A	B	C	D
2000-01-01	-0.471915	-0.982491	1.421004	-0.192351
2000-01-02	0.991406	-0.316590	1.917942	-0.093805
2000-01-03	-0.430281	-1.112225	0.985545	-0.991099
2000-01-04	-1.318746	-1.373021	1.205421	-0.766650
2000-01-05	-1.871198	-0.465319	-0.113678	-1.215254

In [318]:

```
# Python 可以使用 ; 區隔，將三個指令寫在一行，本例僅是示範用，實務操作不建議使用。

df.plot(); plt.legend(loc='best');plt.show()
```



5.2 資料輸入/輸出

In [319]:

```
df = pd.DataFrame(np.random.randn(5, 3), columns=['A','B','C'])
df
```

Out[319]:

	A	B	C
0	-0.058746	0.178172	0.037756
1	0.813678	1.414431	0.321252
2	0.536753	0.900358	-0.005535
3	-0.442518	1.257654	0.174161
4	-1.291784	-1.140809	0.032687

CSV 檔案 - pandas 模組

In [320]:

```
# 讀取 CSV 檔案
df.to_csv('mydata.csv')
```

In [321]:

```
mydf = pd.read_csv('mydata.csv')
mydf
```

Out[321]:

	Unnamed: 0	A	B	C
0	0	-0.058746	0.178172	0.037756
1	1	0.813678	1.414431	0.321252
2	2	0.536753	0.900358	-0.005535
3	3	-0.442518	1.257654	0.174161
4	4	-1.291784	-1.140809	0.032687

台灣電力公司_各縣市再生能源別購入情形

https://data.gov.tw/dataset/29936 (https://data.gov.tw/dataset/29936)

台灣電力公司_各縣市再生能源別購入情形

資料集評分:	★★★★★ 平均 4.2 (5 人次投票)
資料集描述:	各縣市歷年已建置再生能源別之躉購容量及購入度數（自103年起）
主要欄位說明:	年度、縣市、風力件數、風力躉購容量(KW)、風力本年累計購電度數(千度)、太陽光電件數、太陽光電躉購容量(KW)、太陽光電本年累計購電度數(千度)、水力件數、水力躉購容量(KW)、水力本年累計購電度數(千度)、合計件數、躉購容量(KW)、本年累計購電度數(千度)
資料下載網址:	CSV 檢視資料 各縣市歷年已建置再生能源別之躉購容量及購.....
提供機關:	台灣電力股份有限公司
提供機關聯絡人姓名:	林先生 (0223668489)
更新頻率:	每年
授權方式:	政府資料開放授權條款-第1版

RenewableEnergy.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	年度	縣市	風力件數	風力躉購容量(KW)	風力本年累計購電度數(千度)	太陽光電件數	太陽光電躉購容量(KW)	太陽光電本年累計購電度數(千度)	水力件數	水力躉購容量(KW)	水力本年累計購電度數(千度)	合計件數	躉購容量(KW)	本年累計購電度數(千度)
2	107	基隆市	0	0	0	5	470	151253	0	0	0	5	470	151253
3	107	台北市	0	0	0	108	10902	10927586	0	0	0	108	10902	10927586
4	107	新北市	0	0	0	266	20231	16992130	1	70000	214632842	267	90231	231624972
5	107	桃園市	5	48356	128656547	682	123008	132108345	2	130000	354417447	689	301364	615182339
6	107	新竹市	0	0	0	124	11453	11699468	0	0	0	124	11453	11699468
7	107	新竹縣	2	11500	24731588	341	53942	51161875	0	0	0	343	65442	75893463
8	107	苗栗縣	10	157100	344283314	592	53456	53382036	0	0	0	602	210556	397665350
9	107	台中市	9	89704	198470890	1244	128440	143459582	1	110	402911	1254	218254	342333383
10	107	彰化縣	4	96609	218053210	1963	306192	359066863	1	195	797591	1968	402996	577917664
11	107	南投縣	0	0	0	572	44100	47633301	1	16704	55948834	573	60804	103582135
12	107	雲林縣	5	2332	4577210	2687	395860	489640370	1	200	640936	2693	398392	494858516
13	107	嘉義市	0	0	0	288	12003	13695196	0	0	0	288	12003	13695196
14	107	嘉義縣	0	0	0	1672	187445	217281967	2	50130	184762634	1674	237575	402044601

In [322]:

```
# CSV 檔案 pandas 模組 - 匯入
import pandas as pd

filename = "data/RenewableEnergy.csv"

# read data
mydata = pd.read_csv(filename)
```

In [323]:

```
mydata
```

Out[323]:

	年度	縣市	風力件數	風力躉購容量(KW)	風力本年累計購電度數(千度)	太陽光電件數	太陽光電躉購容量(KW)	太陽光電本年累計購電度數(千度)	水力件數	水力躉購容量(KW)	水力本年累計購電度數(千度)	合計件數	躉購容量(KW)	本
0	107	基隆市	0	0.0	0	5	470	151253	0	0	0	5	470.0	
1	107	台北市	0	0.0	0	108	10902	10927586	0	0	0	108	10902.0	10
2	107	新北市	0	0.0	0	266	20231	16992130	1	70000	214632842	267	90231.0	231

In [324]:

```
# pandas 模組 - 匯出 CSV 檔案

# write data
mydata.to_csv("data/RenewableEnergy_new.csv", sep=",", index=False)
```

CSV 檔案 - csv 模組

In [325]:

```
# csv 模組 - 讀取
import csv

filename = "data/RenewableEnergy.csv"
```

In [326]:

```
fields = []
rows = []
```

In [327]:

```
with open(filename, "r", encoding="utf-8") as csvfile:

    csvreader = csv.reader(csvfile)

    fields = next(csvreader)

    for row in csvreader:
        rows.append(row)

    # print the first 6 rows
    for row in rows[:6]:
        print(row)
```

```
['107', '基隆市', '0', '0', '0', '5', '470', '151253', '0', '0', '0', '5', '4
70', '151253']
['107', '台北市', '0', '0', '0', '108', '10902', '10927586', '0', '0', '0',
'108', '10902', '10927586']
['107', '新北市', '0', '0', '0', '266', '20231', '16992130', '1', '70000', '2
14632842', '267', '90231', '231624972']
['107', '桃園市', '5', '48356', '128656547', '682', '123008', '132108345',
'2', '130000', '354417447', '689', '301364', '615182339']
['107', '新竹市', '0', '0', '0', '124', '11453', '11699468', '0', '0', '0',
'124', '11453', '11699468']
['107', '新竹縣', '2', '11500', '24731588', '341', '53942', '51161875', '0',
'0', '0', '343', '65442', '75893463']
```

In [328]:

```
# 匯出成 CSV
columns = ["id", "name", "Score"]

datas = [["1", "ALAN", 90],
         ["2", "JOHN", 80],
         ["3", "RWEPA", 75]]

filename = "data/score.csv"
```

In [329]:

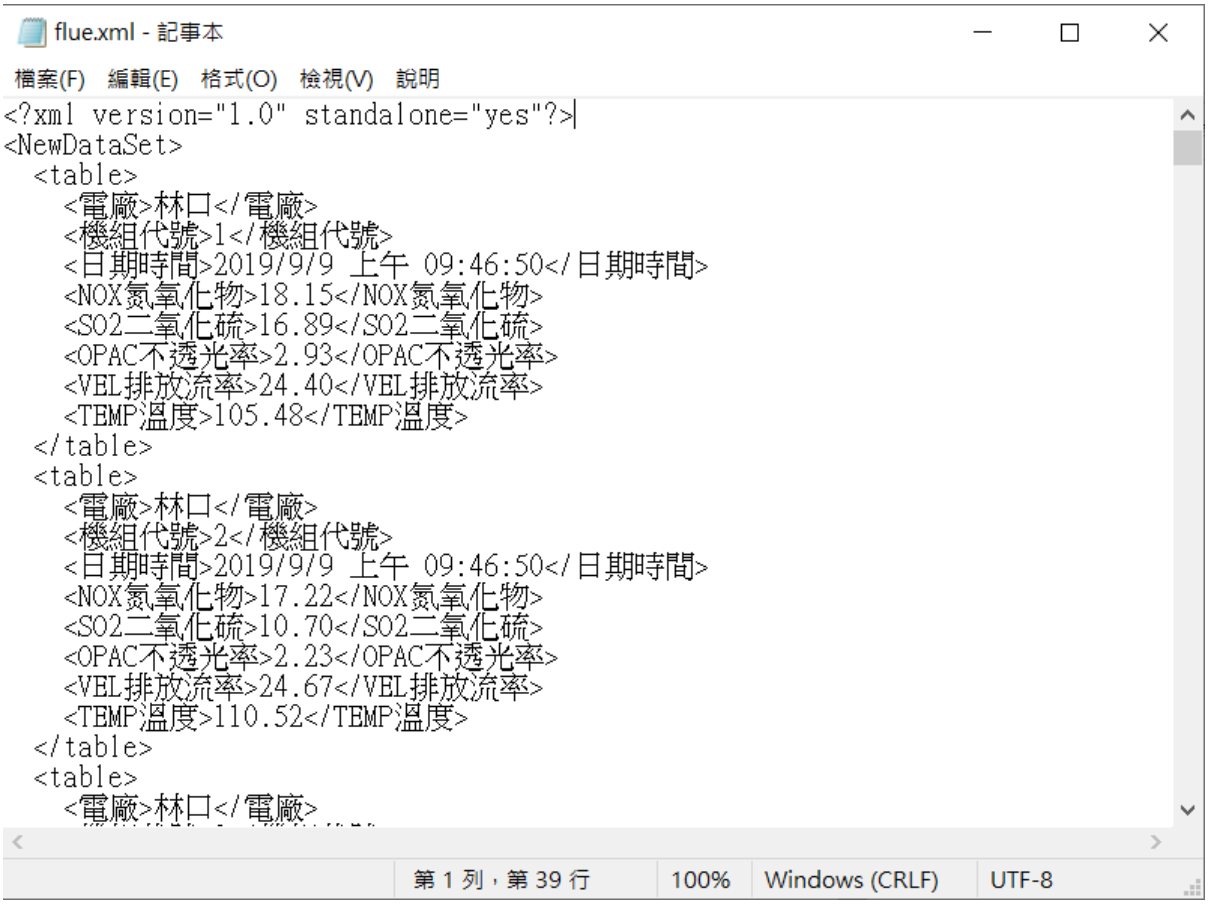
```
with open(filename, "w", newline="") as csvfile:

    csvwriter = csv.writer(csvfile, delimiter=",")

    csvwriter.writerow(columns)

    csvwriter.writerows(datas)
```

XML 檔案

台灣電力公司_煙道資料即時量測值

https://data.gov.tw/dataset/44163 (https://data.gov.tw/dataset/44163)

flue.xml



安裝 **xmltodict, dicttoxml** 二個模組

conda install xmltodict

conda install dicttoxml

In [330]:

```
import xml.etree.ElementTree as ET
import xmltodict
```

In [331]:

```
tree = ET.parse("data/flue.xml")

root = tree.getroot() # 取得XML表格

root # NewDataSet
```

Out[331]:

```
<Element 'NewDataSet' at 0x0000024238515AE8>
```

In [332]:

```
# 以XML檔案的index及迴圈取得XML檔案中的3個欄位資料

for i in range(len(root)):

    print(root[i][0].text, root[i][1].text, root[i][2].text)
```

```
林口 1 2019/9/9 上午 09:46:50
林口 2 2019/9/9 上午 09:46:50
林口 3 2019/9/9 上午 09:46:50
南部 1 2019/9/9 上午 09:46:50
南部 2 2019/9/9 上午 09:46:50
南部 3 2019/9/9 上午 09:46:50
南部 4 2019/9/9 上午 09:46:50
大林 1 2019/9/9 上午 09:46:52
大林 2 2019/9/9 上午 09:46:52
大林 5 2019/9/9 上午 09:46:52
大林 6 2019/9/9 上午 09:46:52
興達 1 2019/9/9 上午 09:46:51
興達 2 2019/9/9 上午 09:46:51
興達 3 2019/9/9 上午 09:46:51
興達 4 2019/9/9 上午 09:46:51
興達 5 2019/9/9 上午 09:46:51
興達 6 2019/9/9 上午 09:46:51
興達 7 2019/9/9 上午 09:46:51
興達 8 2019/9/9 上午 09:46:51
興達 9 2019/9/9 上午 09:46:51
協和 1 2019/9/9 上午 09:46:02
協和 2 2019/9/9 上午 09:46:02
協和 3 2019/9/9 上午 09:46:02
協和 4 2019/9/9 上午 09:46:02
通霄 1 2019/9/9 上午 09:46:51
通霄 2 2019/9/9 上午 09:46:51
通霄 3 2019/9/9 上午 09:46:51
通霄 4 2019/9/9 上午 09:46:51
通霄 5 2019/9/9 上午 09:46:51
通霄 6 2019/9/9 上午 09:46:51
台中 1 2019/9/9 上午 09:46:50
台中 2 2019/9/9 上午 09:46:50
台中 3 2019/9/9 上午 09:46:50
台中 4 2019/9/9 上午 09:46:50
台中 5 2019/9/9 上午 09:46:50
台中 6 2019/9/9 上午 09:46:50
台中 7 2019/9/9 上午 09:46:50
台中 8 2019/9/9 上午 09:46:50
台中 9 2019/9/9 上午 09:46:50
台中 10 2019/9/9 上午 09:46:50
大潭 1 2019/9/9 上午 09:46:02
大潭 2 2019/9/9 上午 09:46:02
大潭 3 2019/9/9 上午 09:46:02
大潭 4 2019/9/9 上午 09:46:02
大潭 5 2019/9/9 上午 09:46:02
大潭 6 2019/9/9 上午 09:46:02
```

In [333]:

```
xmlstr = ET.tostring(root, encoding="utf-8", method="xml")
```

In [334]:

```
data_dict = dict(xmltodict.parse(xmlstr))

print(data_dict)
```

```
{'NewDataSet': OrderedDict([('table', [OrderedDict([('電廠', '林口'), ('機
組代號', '1'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '18.1
5'), ('SO2二氧化硫', '16.89'), ('OPAC不透光率', '2.93'), ('VEL排放流率', '2
4.40'), ('TEMP溫度', '105.48')]), OrderedDict([('電廠', '林口'), ('機組代
號', '2'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '17.22'),
('SO2二氧化硫', '10.70'), ('OPAC不透光率', '2.23'), ('VEL排放流率', '24.6
7'), ('TEMP溫度', '110.52')]), OrderedDict([('電廠', '林口'), ('機組代號',
'3'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '5.60'), ('SO2
二氧化硫', '2.44'), ('OPAC不透光率', '2.42'), ('VEL排放流率', '24.69'), ('TE
MP溫度', '110.20')]), OrderedDict([('電廠', '南部'), ('機組代號', '1'), ('日
期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '6.00'), ('SO2二氧化硫',
'0.00'), ('OPAC不透光率', '0.00'), ('VEL排放流率', '0.00'), ('TEMP溫度', '11
4.00')]), OrderedDict([('電廠', '南部'), ('機組代號', '2'), ('日期時間', '20
19/9/9 上午 09:46:50'), ('NOX氮氧化物', '6.00'), ('SO2二氧化硫', '0.00'),
('OPAC不透光率', '0.00'), ('VEL排放流率', '0.00'), ('TEMP溫度', '115.00')]),
OrderedDict([('電廠', '南部'), ('機組代號', '3'), ('日期時間', '2019/9/9 上
午 09:46:50'), ('NOX氮氧化物', '6.00'), ('SO2二氧化硫', '0.00'), ('OPAC不透
光率', '0.00'), ('VEL排放流率', '0.00'), ('TEMP溫度', '114.00')]), OrderedD
ict([('電廠', '南部'), ('機組代號', '4'), ('日期時間', '2019/9/9 上午 09:46:
50'), ('NOX氮氧化物', '5.00'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.
00'), ('VEL排放流率', '0.00'), ('TEMP溫度', '97.00')]), OrderedDict([('電
廠', '大林'), ('機組代號', '1'), ('日期時間', '2019/9/9 上午 09:46:52'), ('N
OX氮氧化物', '21.65'), ('SO2二氧化硫', '19.55'), ('OPAC不透光率', '5.53'),
('VEL排放流率', '0.00'), ('TEMP溫度', '95.48')]), OrderedDict([('電廠', '大
林'), ('機組代號', '2'), ('日期時間', '2019/9/9 上午 09:46:52'), ('NOX氮氧化
物', '20.25'), ('SO2二氧化硫', '8.52'), ('OPAC不透光率', '5.13'), ('VEL排放
流率', '22.68'), ('TEMP溫度', '92.97')]), OrderedDict([('電廠', '大林'),
('機組代號', '5'), ('日期時間', '2019/9/9 上午 09:46:52'), ('NOX氮氧化物',
'71.30'), ('SO2二氧化硫', '0.10'), ('OPAC不透光率', '6.10'), ('VEL排放流率',
'0.40'), ('TEMP溫度', '124.40')]), OrderedDict([('電廠', '大林'), ('機組代
號', '6'), ('日期時間', '2019/9/9 上午 09:46:52'), ('NOX氮氧化物', '27.90'),
('SO2二氧化硫', '0.10'), ('OPAC不透光率', '5.10'), ('VEL排放流率', '8.80'),
('TEMP溫度', '90.80')]), OrderedDict([('電廠', '興達'), ('機組代號', '1'),
('日期時間', '2019/9/9 上午 09:46:51'), ('NOX氮氧化物', '37.83'), ('SO2二氧
化硫', '21.09'), ('OPAC不透光率', '10.68'), ('VEL排放流率', '19.12'), ('TEMP
溫度', '88.78')]), OrderedDict([('電廠', '興達'), ('機組代號', '2'), ('日期
時間', '2019/9/9 上午 09:46:51'), ('NOX氮氧化物', '20.66'), ('SO2二氧化硫',
'18.96'), ('OPAC不透光率', '9.55'), ('VEL排放流率', '20.75'), ('TEMP溫度',
'104.72')]), OrderedDict([('電廠', '興達'), ('機組代號', '3'), ('日期時間',
'2019/9/9 上午 09:46:51'), ('NOX氮氧化物', '25.07'), ('SO2二氧化硫', '13.8
7'), ('OPAC不透光率', '7.69'), ('VEL排放流率', '23.36'), ('TEMP溫度', '99.9
9')]), OrderedDict([('電廠', '興達'), ('機組代號', '4'), ('日期時間', '2019/
9/9 上午 09:46:51'), ('NOX氮氧化物', '24.67'), ('SO2二氧化硫', '14.14'), ('O
PAC不透光率', '7.86'), ('VEL排放流率', '22.39'), ('TEMP溫度', '93.42')]), Or
deredDict([('電廠', '興達'), ('機組代號', '5'), ('日期時間', '2019/9/9 上午
09:46:51'), ('NOX氮氧化物', '11.84'), ('SO2二氧化硫', '0.00'), ('OPAC不透光
率', '0.00'), ('VEL排放流率', '12.36'), ('TEMP溫度', '110.70')]), OrderedDi
ct([('電廠', '興達'), ('機組代號', '6'), ('日期時間', '2019/9/9 上午 09:46:5
1'), ('NOX氮氧化物', '11.35'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.
00'), ('VEL排放流率', '11.75'), ('TEMP溫度', '111.67')]), OrderedDict([('電
廠', '興達'), ('機組代號', '7'), ('日期時間', '2019/9/9 上午 09:46:51'), ('N
OX氮氧化物', '14.92'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.00'),
('VEL排放流率', '11.16'), ('TEMP溫度', '111.30')]), OrderedDict([('電廠',
'興達'), ('機組代號', '8'), ('日期時間', '2019/9/9 上午 09:46:51'), ('NOX氮
```

氧化物', '19.69'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.00'), ('VEL排放流率', '11.65'), ('TEMP溫度', '115.59')]), OrderedDict([('電廠', '興達'), ('機組代號', '9'), ('日期時間', '2019/9/9 上午 09:46:51'), ('NOX氮氧化物', '17.90'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.00'), ('VEL排放流率', '7.93'), ('TEMP溫度', '105.00')]), OrderedDict([('電廠', '協和'), ('機組代號', '1'), ('日期時間', '2019/9/9 上午 09:46:02'), ('NOX氮氧化物', '1.12'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '11.63'), ('VEL排放流率', '10343.14'), ('TEMP溫度', '31.13')]), OrderedDict([('電廠', '協和'), ('機組代號', '2'), ('日期時間', '2019/9/9 上午 09:46:02'), ('NOX氮氧化物', '0.00'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '14.50'), ('VEL排放流率', '23259.04'), ('TEMP溫度', '29.54')]), OrderedDict([('電廠', '協和'), ('機組代號', '3'), ('日期時間', '2019/9/9 上午 09:46:02'), ('NOX氮氧化物', '81.69'), ('SO2二氧化硫', '123.99'), ('OPAC不透光率', '11.85'), ('VEL排放流率', '543025.03'), ('TEMP溫度', '122.53')]), OrderedDict([('電廠', '協和'), ('機組代號', '4'), ('日期時間', '2019/9/9 上午 09:46:02'), ('NOX氮氧化物', '119.27'), ('SO2二氧化硫', '145.06'), ('OPAC不透光率', '14.18'), ('VEL排放流率', '686947.88'), ('TEMP溫度', '134.20')]), OrderedDict([('電廠', '通霄'), ('機組代號', '1'), ('日期時間', '2019/9/9 上午 09:46:51'), ('NOX氮氧化物', '1.00'), ('SO2二氧化硫', '1.00'), ('OPAC不透光率', '6.40'), ('VEL排放流率', '16153.00'), ('TEMP溫度', '33.00')]), OrderedDict([('電廠', '通霄'), ('機組代號', '2'), ('日期時間', '2019/9/9 上午 09:46:51'), ('NOX氮氧化物', '6.00'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.00'), ('VEL排放流率', '4059142.00'), ('TEMP溫度', '82.00')]), OrderedDict([('電廠', '通霄'), ('機組代號', '3'), ('日期時間', '2019/9/9 上午 09:46:51'), ('NOX氮氧化物', '8.00'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.00'), ('VEL排放流率', '1554528.00'), ('TEMP溫度', '72.00')]), OrderedDict([('電廠', '通霄'), ('機組代號', '4'), ('日期時間', '2019/9/9 上午 09:46:51'), ('NOX氮氧化物', '12.00'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.00'), ('VEL排放流率', '1355036.00'), ('TEMP溫度', '97.00')]), OrderedDict([('電廠', '通霄'), ('機組代號', '5'), ('日期時間', '2019/9/9 上午 09:46:51'), ('NOX氮氧化物', '10.00'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.00'), ('VEL排放流率', '2299313.00'), ('TEMP溫度', '108.00')]), OrderedDict([('電廠', '通霄'), ('機組代號', '6'), ('日期時間', '2019/9/9 上午 09:46:51'), ('NOX氮氧化物', '16.00'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.00'), ('VEL排放流率', '783705.00'), ('TEMP溫度', '105.00')]), OrderedDict([('電廠', '台中'), ('機組代號', '1'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '32.08'), ('SO2二氧化硫', '25.87'), ('OPAC不透光率', '15.70'), ('VEL排放流率', '23.07'), ('TEMP溫度', '136.12')]), OrderedDict([('電廠', '台中'), ('機組代號', '2'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '38.76'), ('SO2二氧化硫', '18.97'), ('OPAC不透光率', '16.41'), ('VEL排放流率', '21.59'), ('TEMP溫度', '134.42')]), OrderedDict([('電廠', '台中'), ('機組代號', '3'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '42.33'), ('SO2二氧化硫', '14.88'), ('OPAC不透光率', '17.11'), ('VEL排放流率', '20.87'), ('TEMP溫度', '134.19')]), OrderedDict([('電廠', '台中'), ('機組代號', '4'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '0.00'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '0.00'), ('VEL排放流率', '0.08'), ('TEMP溫度', '29.06')]), OrderedDict([('電廠', '台中'), ('機組代號', '5'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '31.76'), ('SO2二氧化硫', '36.28'), ('OPAC不透光率', '13.33'), ('VEL排放流率', '11.68'), ('TEMP溫度', '96.01')]), OrderedDict([('電廠', '台中'), ('機組代號', '6'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '46.29'), ('SO2二氧化硫', '34.05'), ('OPAC不透光率', '15.33'), ('VEL排放流率', '18.29'), ('TEMP溫度', '109.17')]), OrderedDict([('電廠', '台中'), ('機組代號', '7'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '39.84'), ('SO2二氧化硫', '33.19'), ('OPAC不透光率', '12.64'), ('VEL排放流率', '19.84'), ('TEMP溫度', '101.94')]), OrderedDict([('電廠', '台中'), ('機組代號', '8'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '44.38'), ('SO2二氧化硫', '26.75'), ('OPAC不透光率', '14.08'), ('VEL排放流率', '21.71'), ('TEMP溫度', '108.09')]), OrderedDict([('電廠', '台中'), ('機組代號', '9'), ('日期時間', '2019/9/9 上午 09:46:50'), ('NOX氮氧化物', '42.13'), ('SO2二氧化硫', '29.08'), ('OPAC不透光率', '7.24'), ('VEL排放流率', '19.28'), ('TEMP溫度', '98.98')]), OrderedDict

([('電廠', '台中'), ('機組代號', '10'), ('日期時間', '2019/9/9 上午 09:46:5
0'), ('NOX氮氧化物', '42.19'), ('SO2二氧化硫', '28.47'), ('OPAC不透光率', '1
3.74'), ('VEL排放流率', '21.40'), ('TEMP溫度', '101.97')]), OrderedDict
([('電廠', '大潭'), ('機組代號', '1'), ('日期時間', '2019/9/9 上午 09:46:0
2'), ('NOX氮氧化物', '14.50'), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', '10
343.14'), ('VEL排放流率', '0.00'), ('TEMP溫度', '29.54')]), OrderedDict
([('電廠', '大潭'), ('機組代號', '2'), ('日期時間', '2019/9/9 上午 09:46:0
2'), ('NOX氮氧化物', '543025.03'), ('SO2二氧化硫', '31.13'), ('OPAC不透光
率', '123.99'), ('VEL排放流率', '119.27'), ('TEMP溫度', '14.18')]), Ordered
Dict([('電廠', '大潭'), ('機組代號', '3'), ('日期時間', '2019/9/9 上午 09:4
6:02'), ('NOX氮氧化物', None), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', Non
e), ('VEL排放流率', None), ('TEMP溫度', None)]), OrderedDict([('電廠', '大
潭'), ('機組代號', '4'), ('日期時間', '2019/9/9 上午 09:46:02'), ('NOX氮氧化
物', None), ('SO2二氧化硫', '0.00'), ('OPAC不透光率', None), ('VEL排放流率',
None), ('TEMP溫度', None)]), OrderedDict([('電廠', '大潭'), ('機組代號',
'5'), ('日期時間', '2019/9/9 上午 09:46:02'), ('NOX氮氧化物', None), ('SO2二
氧化硫', '0.00'), ('OPAC不透光率', None), ('VEL排放流率', None), ('TEMP溫
度', None)]), OrderedDict([('電廠', '大潭'), ('機組代號', '6'), ('日期時間',
'2019/9/9 上午 09:46:02'), ('NOX氮氧化物', None), ('SO2二氧化硫', '0.00'),
('OPAC不透光率', None), ('VEL排放流率', None), ('TEMP溫度', None)])])])}

Excel 檔案

In [335]:

```
df.to_excel('mydata.xlsx', sheet_name='Sheet1')
```

In [336]:

```
myexcel = pd.read_excel('mydata.xlsx', 'Sheet1',
                        index_col=None, na_values=['NA'])
myexcel
```

Out[336]:

	Unnamed: 0	A	B	C
0	0	-0.058746	0.178172	0.037756
1	1	0.813678	1.414431	0.321252
2	2	0.536753	0.900358	-0.005535
3	3	-0.442518	1.257654	0.174161
4	4	-1.291784	-1.140809	0.032687

HDF5 檔案

In [337]:

```
df.to_hdf('mydata.h5', 'mydf')
```

In [338]:

```
mydatahdf5 = pd.read_hdf('mydata.h5')
mydatahdf5
```

Out[338]:

	A	B	C
0	-0.058746	0.178172	0.037756
1	0.813678	1.414431	0.321252
2	0.536753	0.900358	-0.005535
3	-0.442518	1.257654	0.174161
4	-1.291784	-1.140809	0.032687

HTML 檔案

In [339]:

```
# HTML to CSV

import requests

from bs4 import BeautifulSoup

import csv
```

In [340]:

```
myhtml = "https://docs.python.org/3/py-modindex.html"
r = requests.get(myhtml)
r
```

Out[340]:

```
<Response [200]>
```

In [341]:

```
soup = BeautifulSoup(r.text)

table = soup.find("table")

output_rows = []

for table_row in table.findAll('tr'):
    columns = table_row.findAll('td')
    output_row = []
    for column in columns:
        output_row.append(column.text)
    output_rows.append(output_row)
```

In [342]:

```
type(output_rows)
```

Out[342]:

```
list
```

In [343]:

```
with open("data/output-py-modules.csv", "w", newline="", encoding="utf-8") as csvfile:
    csvwriter = csv.writer(csvfile, delimiter=",")
    csvwriter.writerows(output_rows)
```

第6章 視覺化應用

本章節從視覺化簡介為開端，包括以下內容：

- **6.1** 視覺化簡介
- **6.2** 認識 **matplotlib** 模組
- **6.3 matplotlib** 繪圖應用
- **6.4 seaborn** 模組繪圖
- **6.5** 互動式繪圖

6.1 視覺化簡介

視覺化是指用於建立圖形、圖像或動畫，以便交流溝通訊息的任何技術和方法。在歷史上包括洞穴壁畫、埃及象形文字等，如今視覺化有不斷擴大的應用領域，如科學教育、工程、互動多媒體、醫學等。

參考資料 https://zh.wikipedia.org/wiki/可视化_(计算机图形学)
(https://zh.wikipedia.org/wiki/%E5%8F%AF%E8%A7%86%E5%8C%96_(%E8%AE%A1%E7%AE%97%E6%9C%

視覺化意義

- 不是僅有表面敘述或是畫出圖形。
- 是概念 (concepts), 想法 (ideas), 特性 (properties)轉換過程

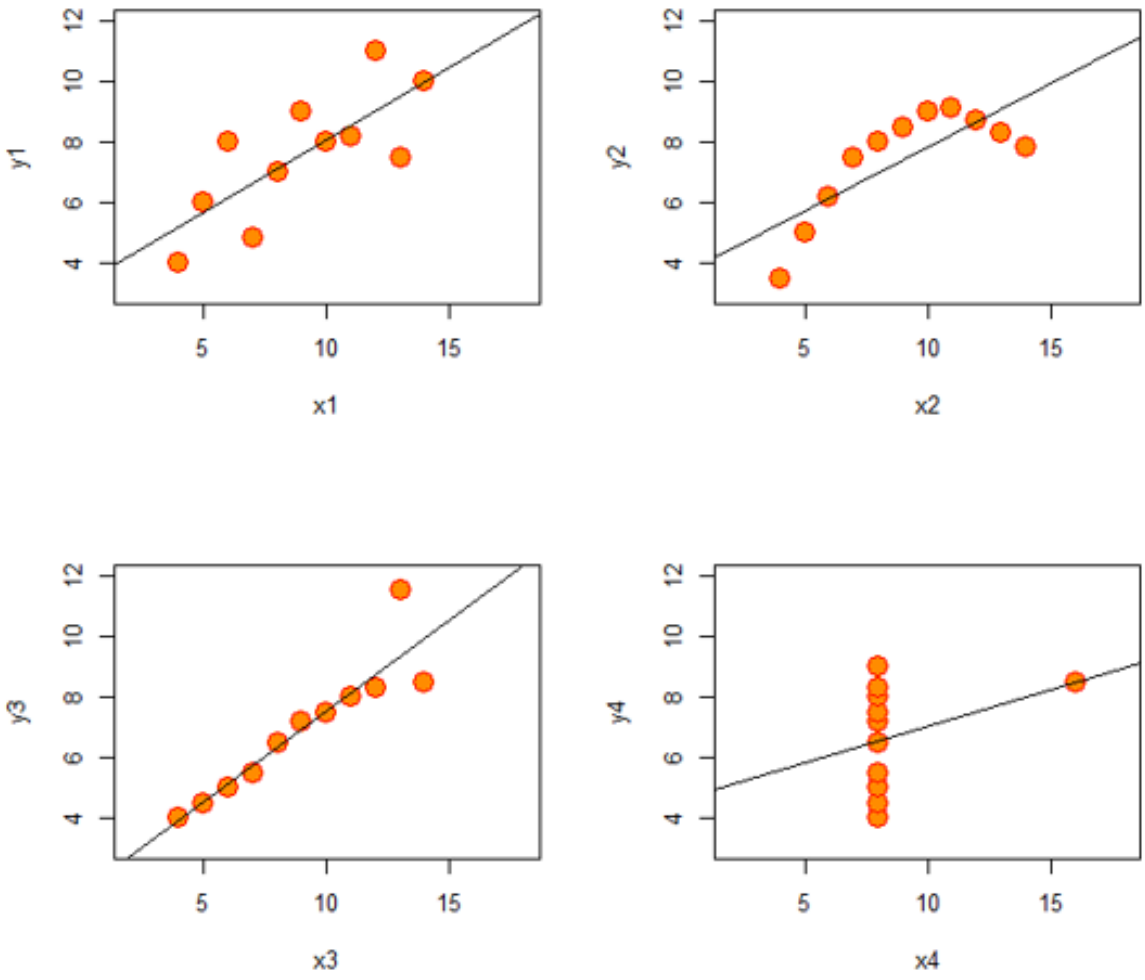
參考資料: Tufte, E. R., Professor Emeritus of Political Science, Statistics, and Computer Science at Yale University, maintains that “excellence in statistical graphics consists of complex ideas communicated with clarity, precision, and efficiency”, 1983.

為什麼需要視覺化

因為人腦不善於閱讀和分析大量資料。

	A	B	C	D	E	F	G
1	Country or Area	Subgroup	Year	Source	Unit	Value	Value Footnotes
2	Afghanistan	Male 5-14 yr	2002-2011	UN_Demographic a	Percent	11	1
3	Afghanistan	Female 5-14 yr	2002-2011	UN_Demographic a	Percent	10	1
4	Afghanistan	Total 5-14 yr	2002-2011	UN_Demographic a	Percent	10	1
5	Albania	Male 5-14 yr	2002-2011	UN_Demographic a	Percent	14	1
6	Albania	Female 5-14 yr	2002-2011	UN_Demographic a	Percent	9	1
7	Albania	Total 5-14 yr	2002-2011	UN_Demographic a	Percent	12	1
8	Algeria	Male 5-14 yr	2002-2011	UN_Demographic a	Percent	6	2
9	Algeria	Female 5-14 yr	2002-2011	UN_Demographic a	Percent	4	2
10	Algeria	Total 5-14 yr	2002-2011	UN_Demographic a	Percent	5	2
11	Angola	Male 5-14 yr	2002-2011	UN_Demographic a	Percent	22	3
12	Angola	Female 5-14 yr	2002-2011	UN_Demographic a	Percent	25	3
13	Angola	Total 5-14 yr	2002-2011	UN_Demographic a	Percent	24	3
14	Argentina	Male 5-14 yr	2002-2011	UN_Demographic a	Percent	8	2
15	Argentina	Female 5-14 yr	2002-2011	UN_Demographic a	Percent	5	2
16	Argentina	Total 5-14 yr	2002-2011	UN_Demographic a	Percent	7	2
17	Armenia	Male 5-14 yr	2002-2011	UN_Demographic a	Percent	5	2
18	Armenia	Female 5-14 yr	2002-2011	UN_Demographic a	Percent	3	2
19	Armenia	Total 5-14 yr	2002-2011	UN_Demographic a	Percent	4	2

但是人腦很善於閱讀和分析圖形。



人類是視覺動物，其視覺神經系統有強大的模式識別和分析能力，視覺化是啟動這套系統的途徑。

視覺化是一種高效的資訊壓縮和展示方法，能將大量資料快速傳輸給人的大腦。

視覺化能探索並提煉資料，並促進新的問題的提出和解決。

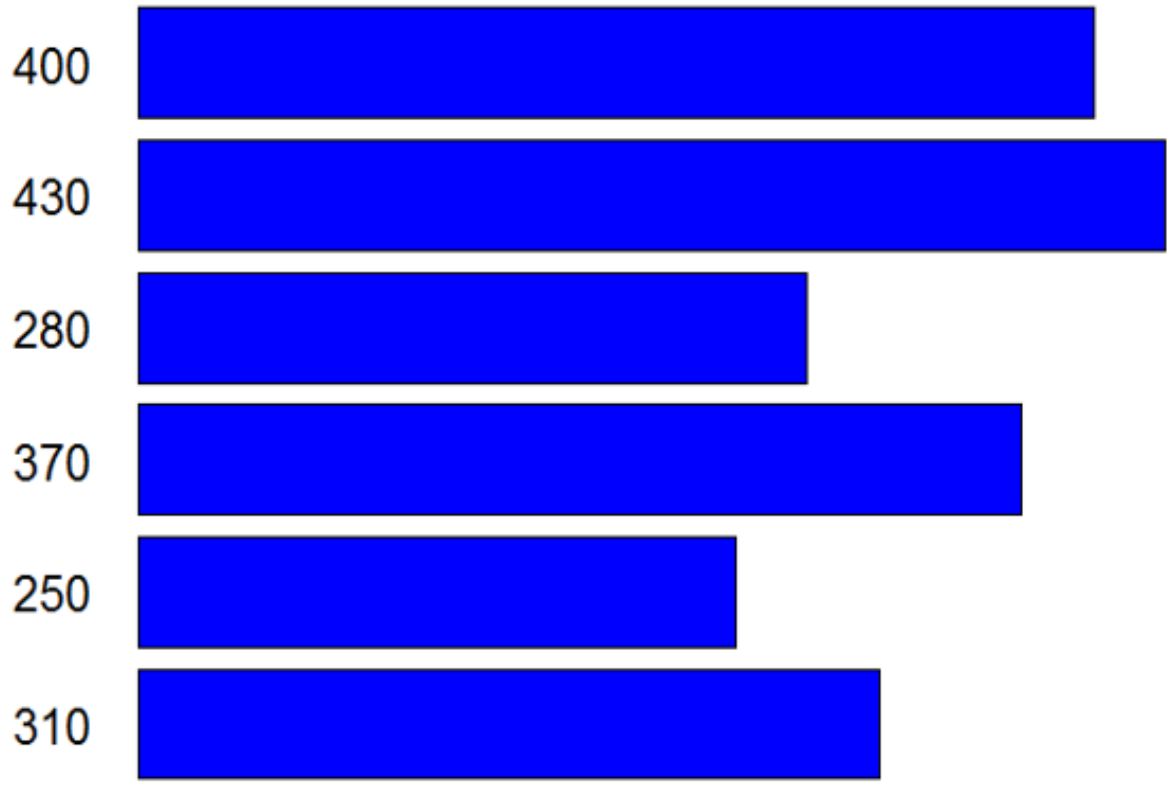
探索式資料分析是資料視覺化的重要應用。

資料視覺化目的

- **identify properties (特性)**
- **relationships (關係)**
- **regularities (規則)**
- **patterns (樣式)**

人們接收視覺屬性(顏色, 大小, 形狀)非常好。

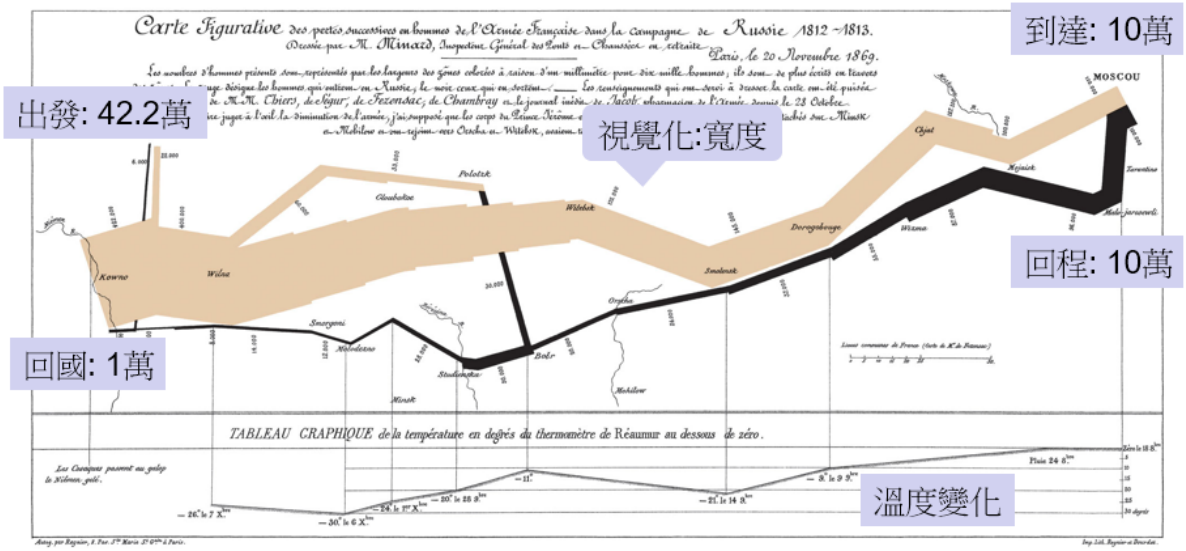
圖形認知優於心智思考。



第一個著名的視覺化-俄法戰爭。

俄法戰爭

第一個著名的視覺化



- Charles Joseph Minard, a French engineer, in 1869.
- Illustrate the number of losses suffered by Napoleon's army during the disastrous march toward Moscow in 1812.6 - 1812.11

視覺化的原則

- 明確視覺化的具體目標
- 可視化的類型？
- 要傳送什麼樣的資訊？哪些最重要？使用什麼資料？
- 考慮觀眾/聽眾之不同
- 閱讀者的角色和知識背景？
- 需要何類資訊？細節程度？
- 風格簡約，凸顯主題
- 閱讀者的注意力是有限的，提煉重點
- 最大化 **Data-ink ratio**
- 選擇合適的視覺編碼方法
- 位置、長度、尺寸、角度、顏色、形狀等

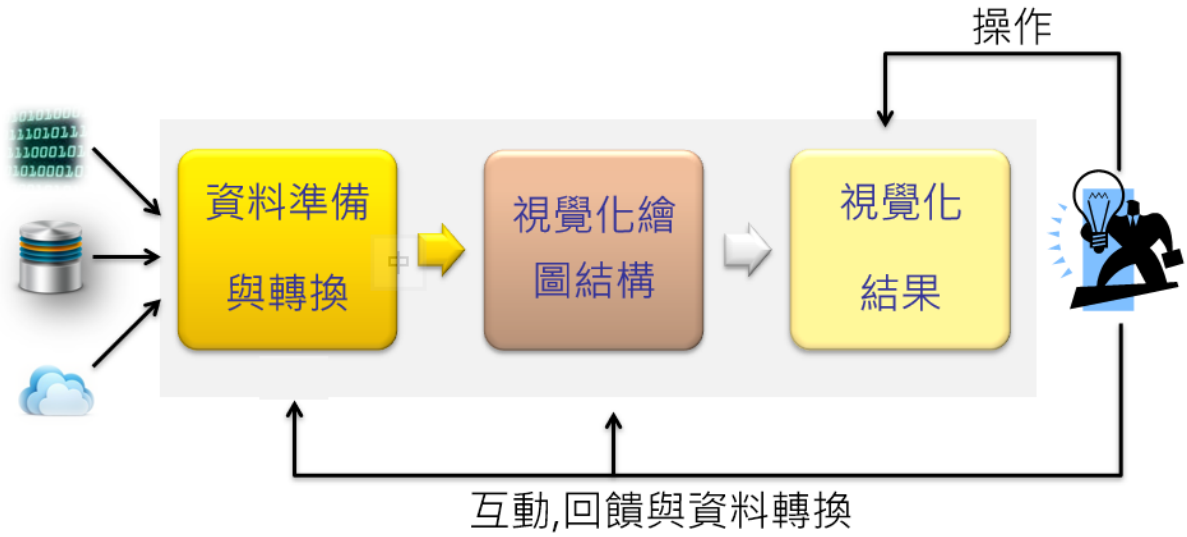
資料視覺化三大步驟

1. 資料準備與轉換
    - 數據獲取，從外部獲得視覺化所需要的原始資料。
    - 資料整理，對資料進行整理，形成所需的結構和文檔格式。
2. 視覺化繪圖結構

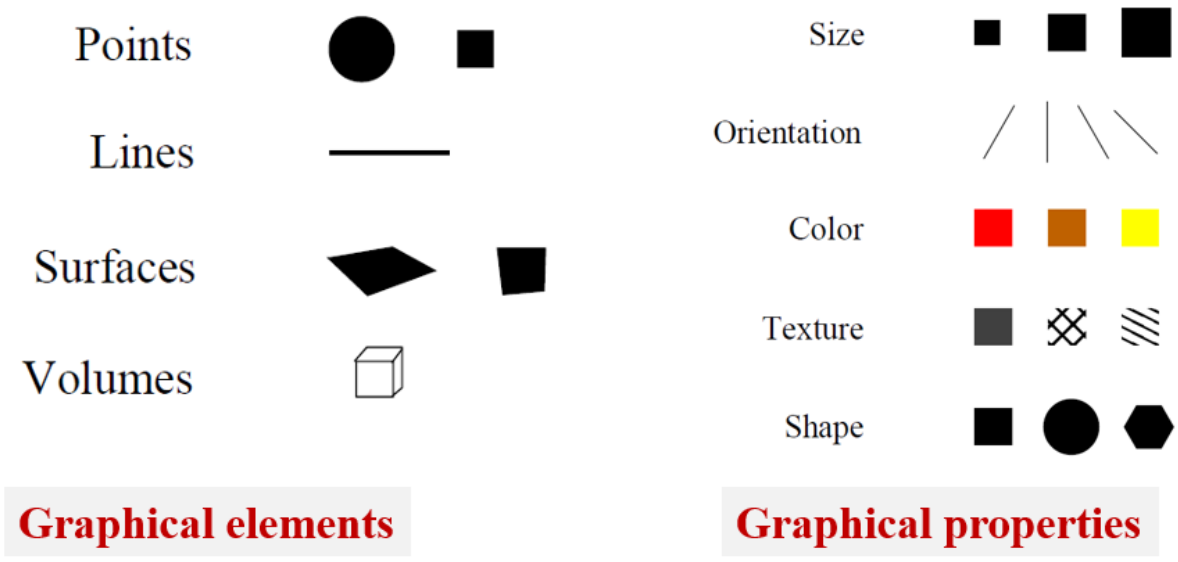
- 資料分析，用統計和挖掘方法對資料進行模式提煉，通常是解釋性視覺化的前提。
- 設計特徵映射，將資料特徵映射到圖形元素，可手繪草圖構思。

3. 視覺化結果

詳細流程參考下圖所示。



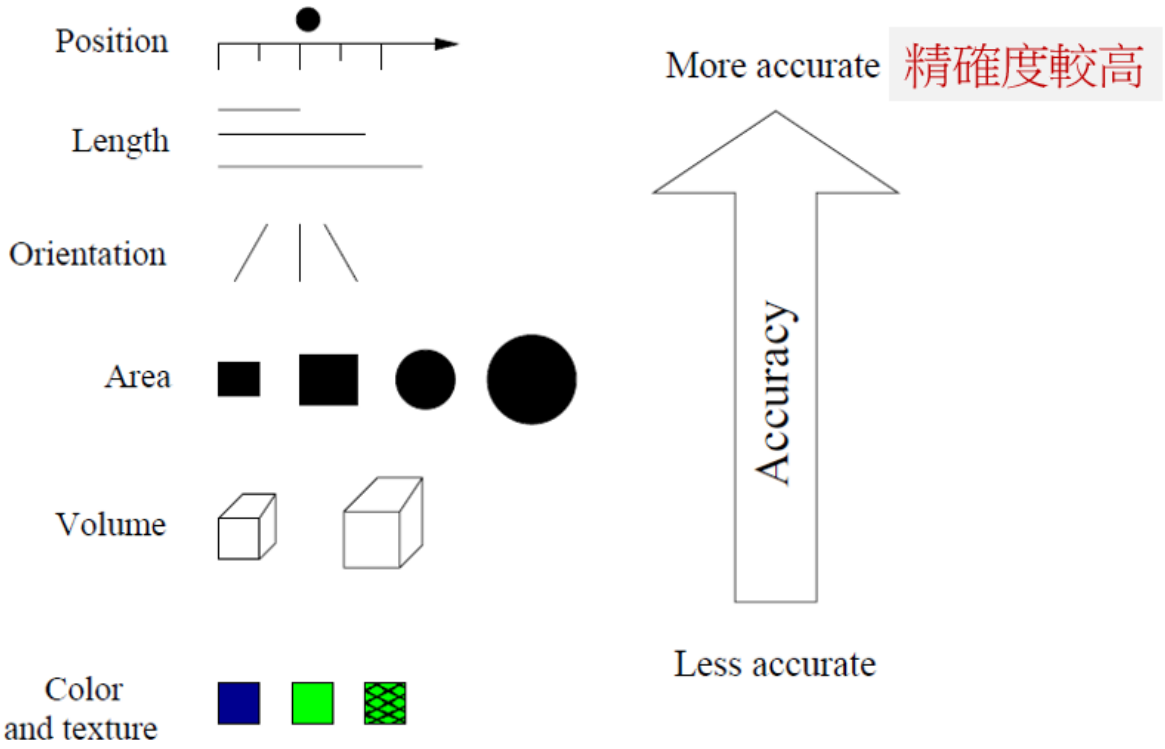
視覺化繪圖結構



視覺化繪圖結構 - 依精細度為區分

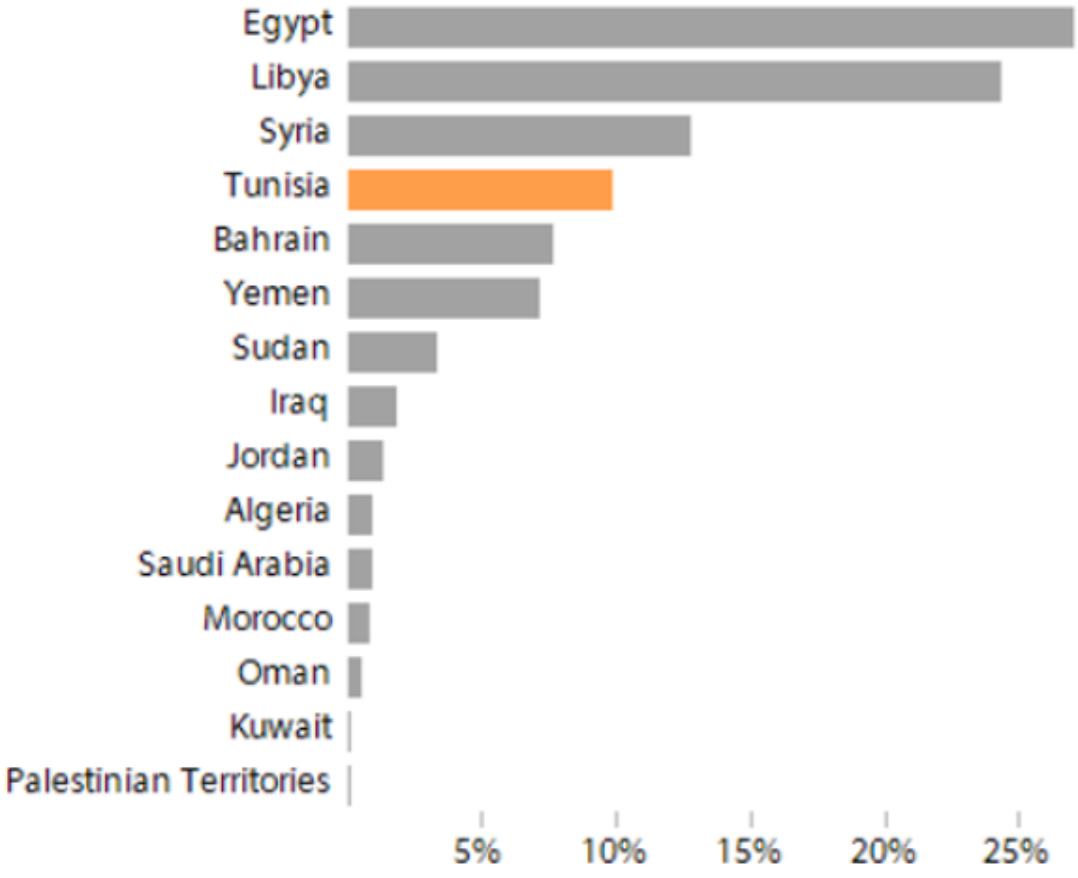
- 位置 (高精確度)
- 長度
- 方向
- 面積
- 體積

- 顏色 (低精確度)

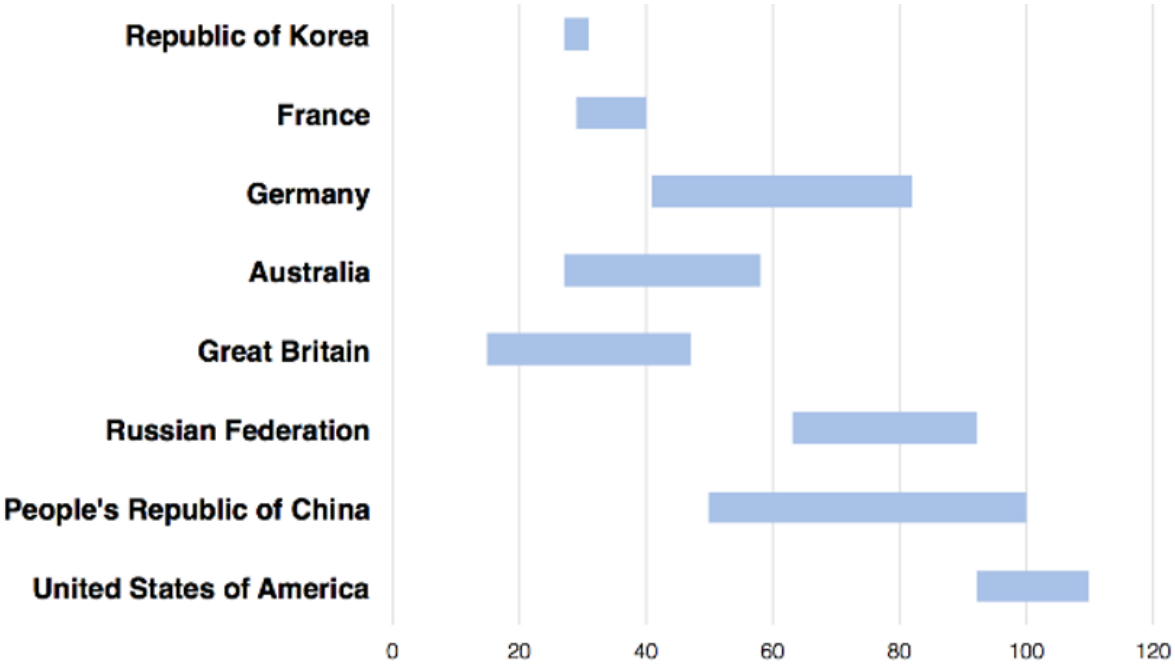


視覺化案例

類別變數的處理1



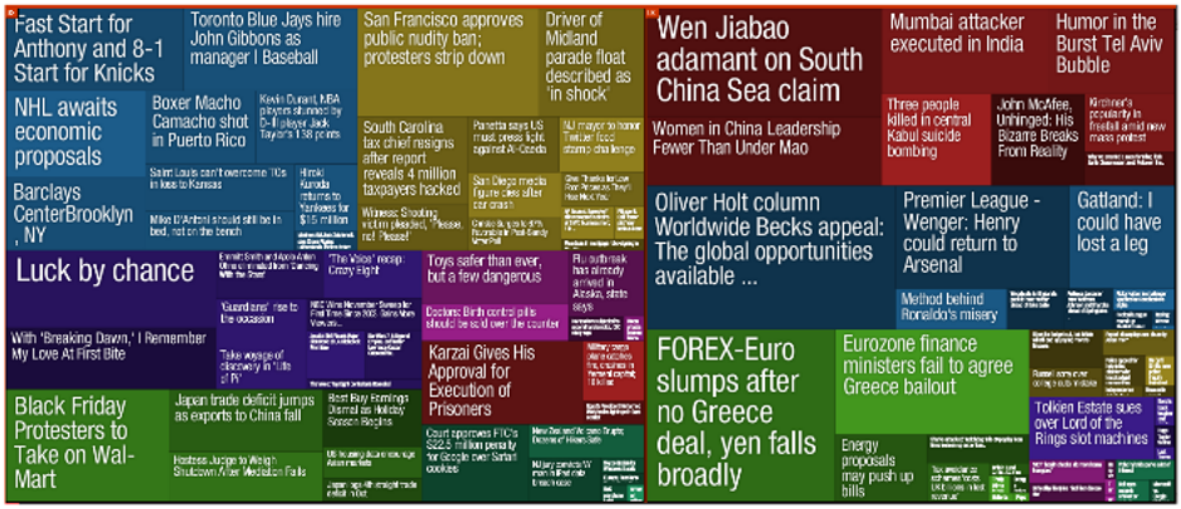
類別變數的處理**2**



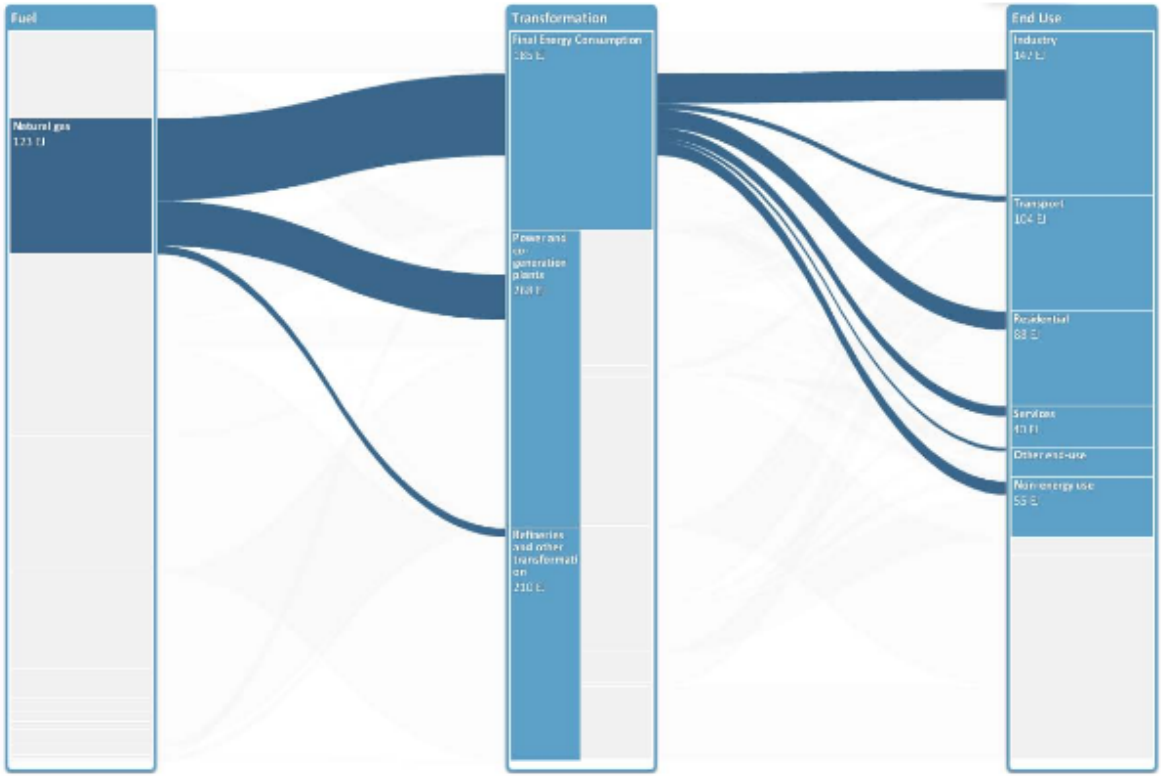
Type *Markdown* and LaTeX: α^2

類別變數的處理**3**

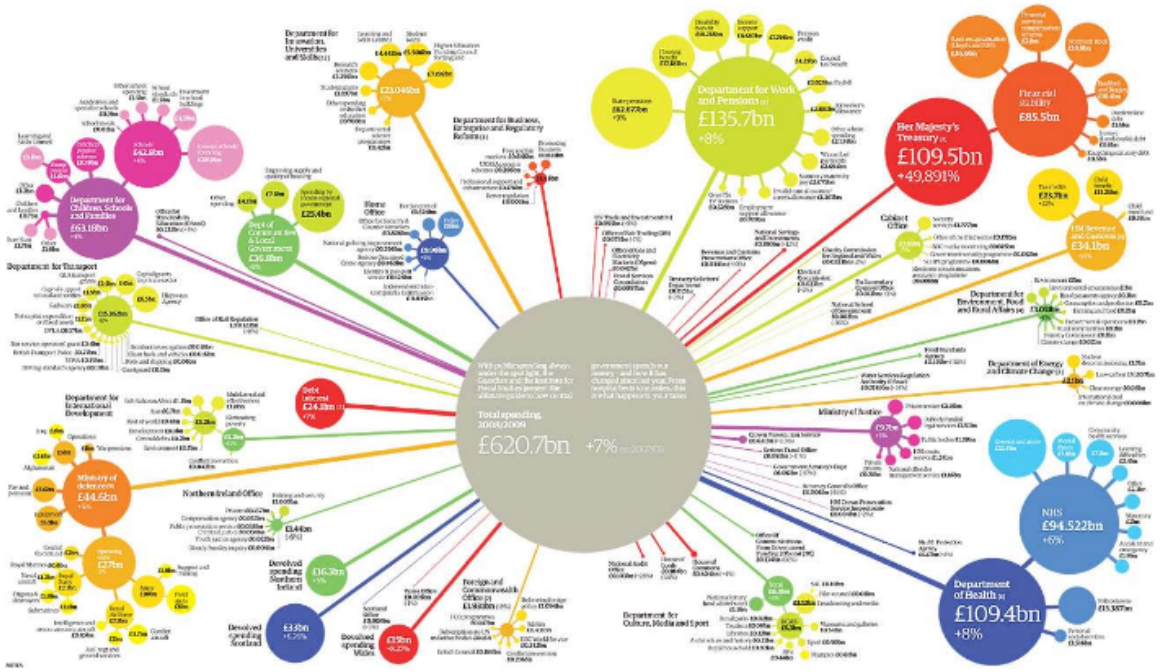
層次結構的展示**1**



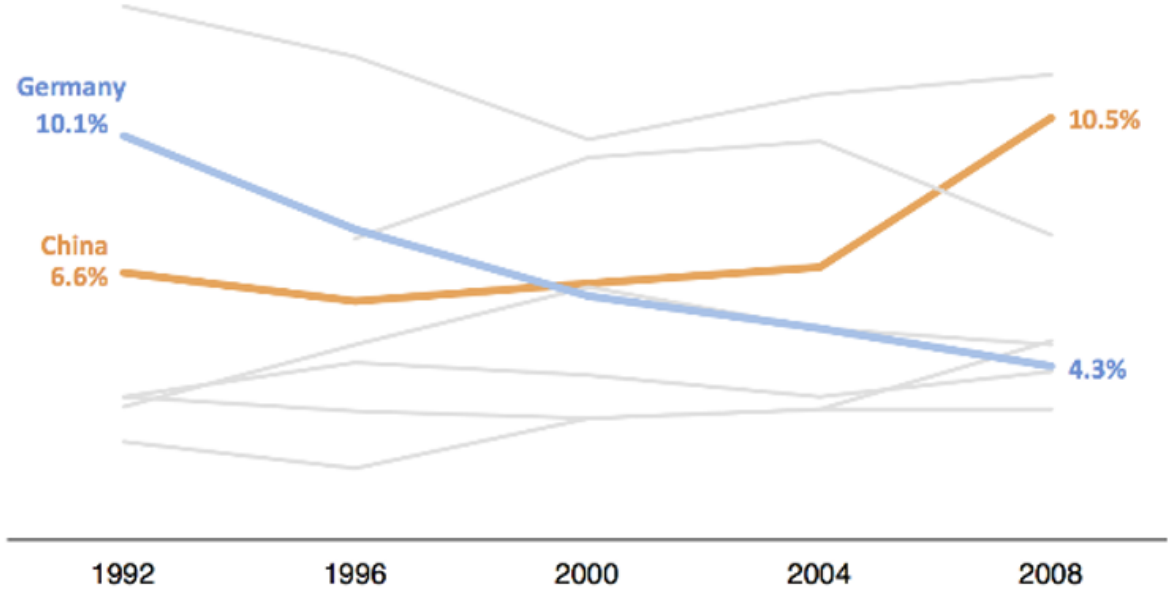
層次結構的展示**2**



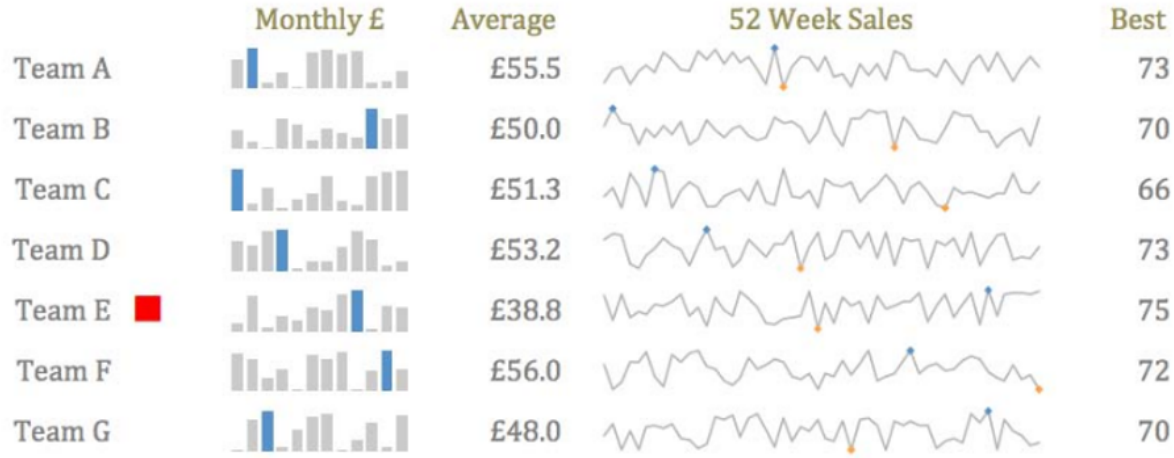
層次結構的展示**3**



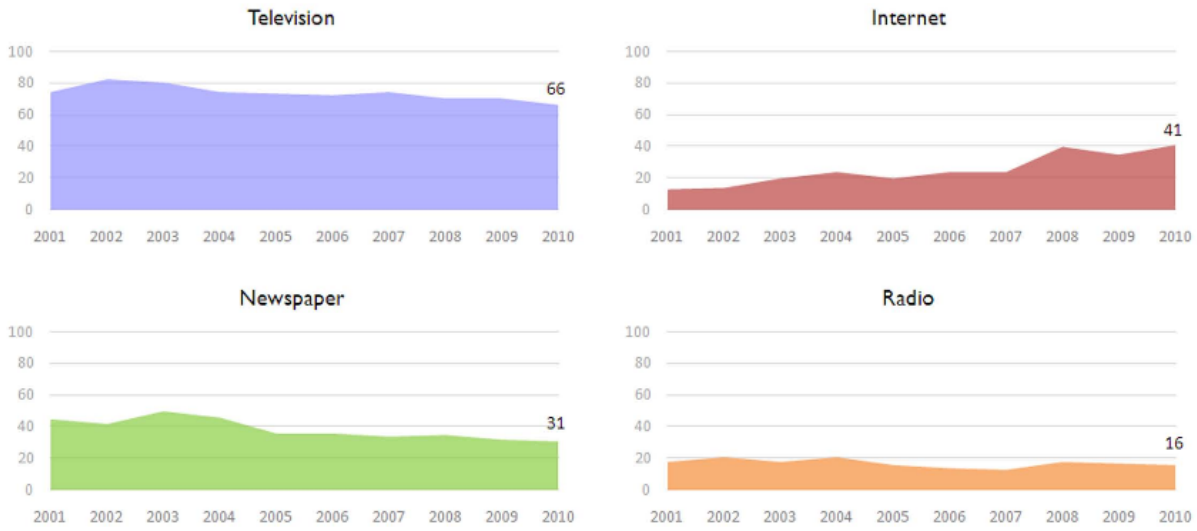
顯示時間的流動**1**



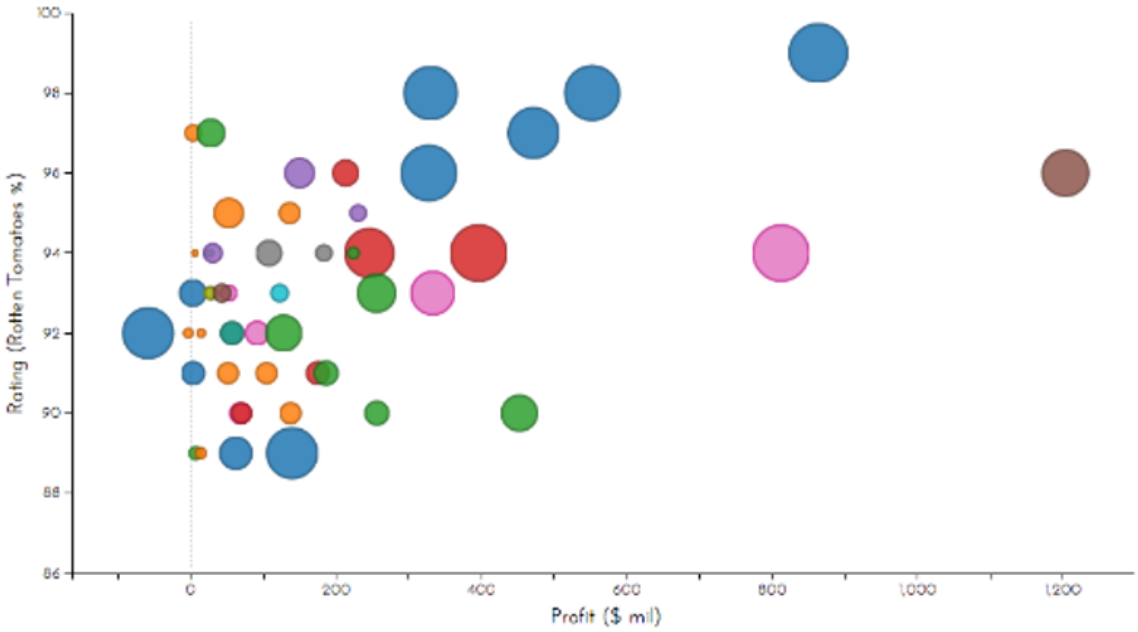
顯示時間的流動**2**



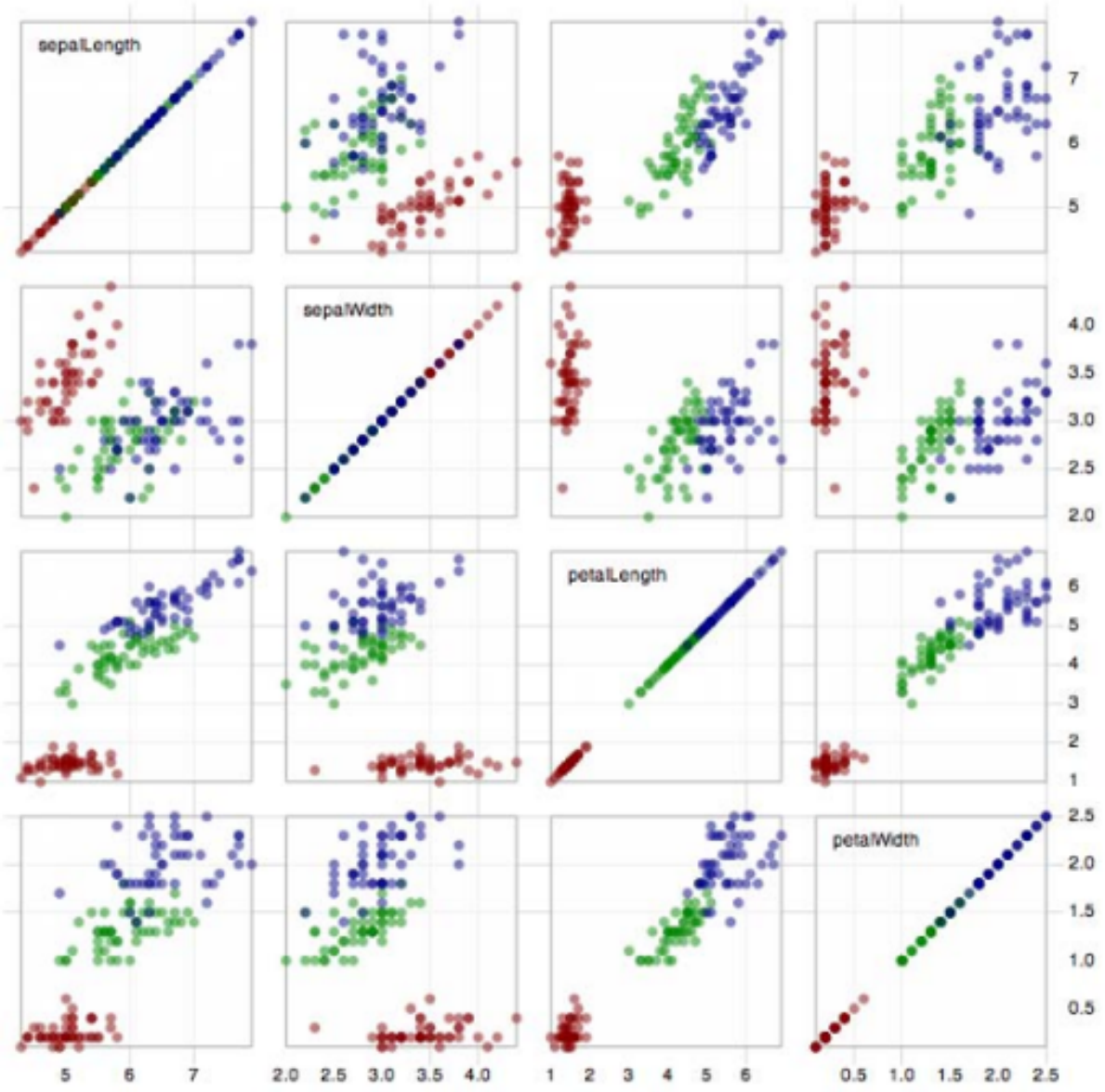
顯示時間的流動**3**



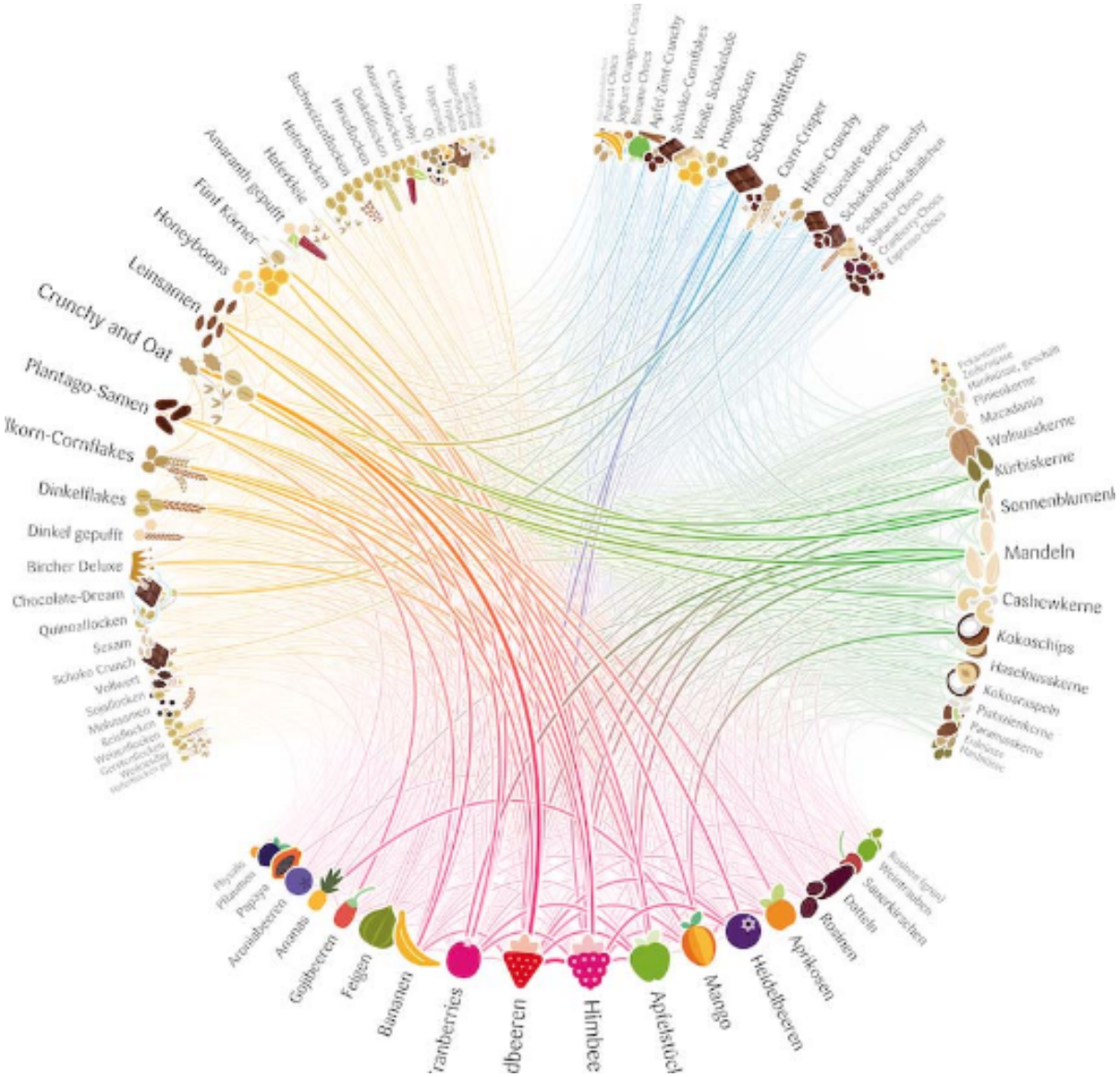
顯示關係的強弱**1**



顯示關係的強弱**2**


sepalLength
sepalWidth
petalLength
petalWidth
5
6
7
2.0
2.5
3.0
3.5
4.0
1
2
3
4
5
6
0.5
1.0
1.5
2.0
2.5

顯示關係的強弱**3**



視覺化的陷阱

- 避免使用太複雜三維圖形
- 圓形圖處理的類別數目不可過多
- 避免使用漸變色
- 避免使用陰影

6.2 認識 matplotlib 模組

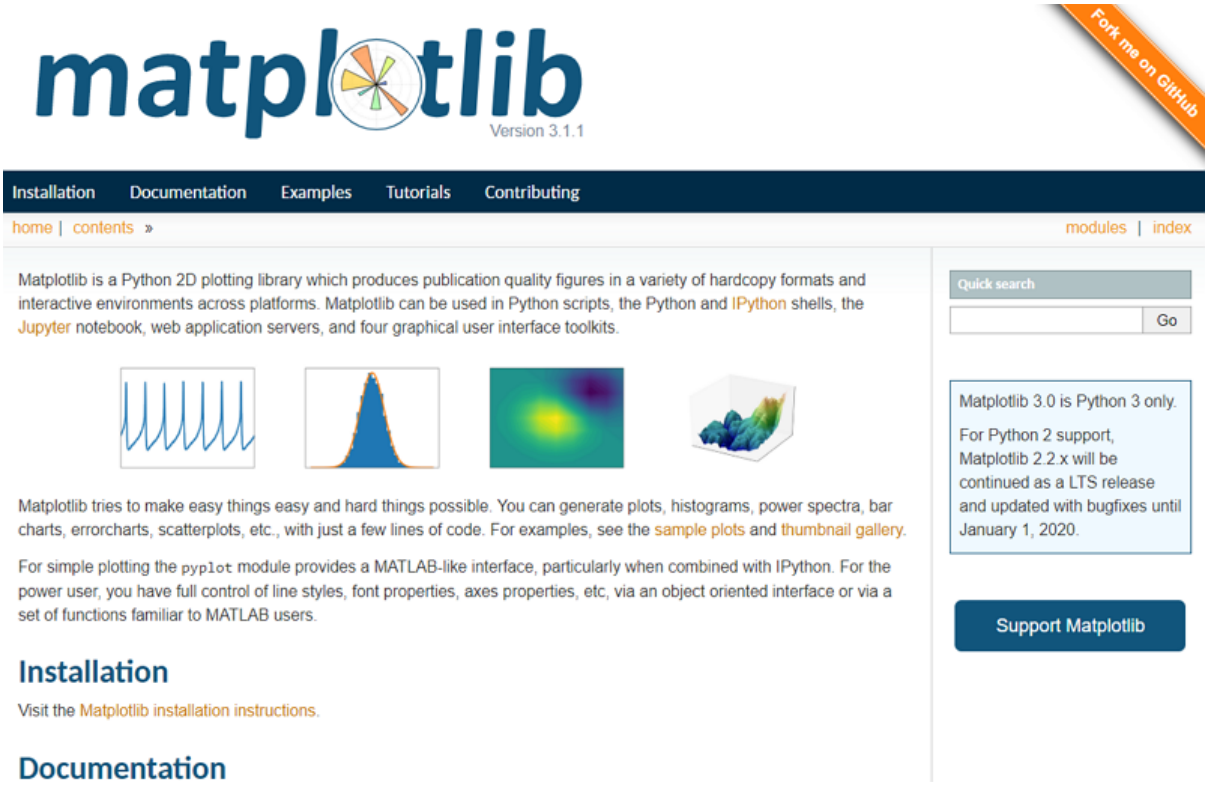
matplotlib 是Python程式語言及其數值數學模組 NumPy的可視化操作界面。它利用通用的圖形用戶界面工具包，如Tkinter, wxPython, Qt或GTK+，在應用程式嵌入式繪圖提供了應用程式接口（API）。此外，matplotlib還有一個基於圖像處理庫（如開放圖形庫OpenGL）的pylab接口，其設計與 MATLAB 非常類似。SciPy就是用matplotlib進行圖形繪製。

matplotlib最初由John D. Hunter撰寫，它擁有一個活躍的開發社區，並且根據BSD樣式許可證分發。 在John D. Hunter2012年去世前不久，Michael Droettboom被提名為matplotlib的主要開發者。

截至到2015年10月30日，matplotlib 1.5.x支持Python 2.7到3.5版本。Matplotlib 1.2是第一個支持Python 3.x的版本。Matplotlib 1.4是支持Python 2.6的最後一個版本。

資料來源: https://zh.wikipedia.org/wiki/Matplotlib (https://zh.wikipedia.org/wiki/Matplotlib)

matplotlib 官方網站 https://matplotlib.org/ (https://matplotlib.org/), 參考下圖所示.



6.3 matplotlib 模組繪圖

散佈圖

In [344]:

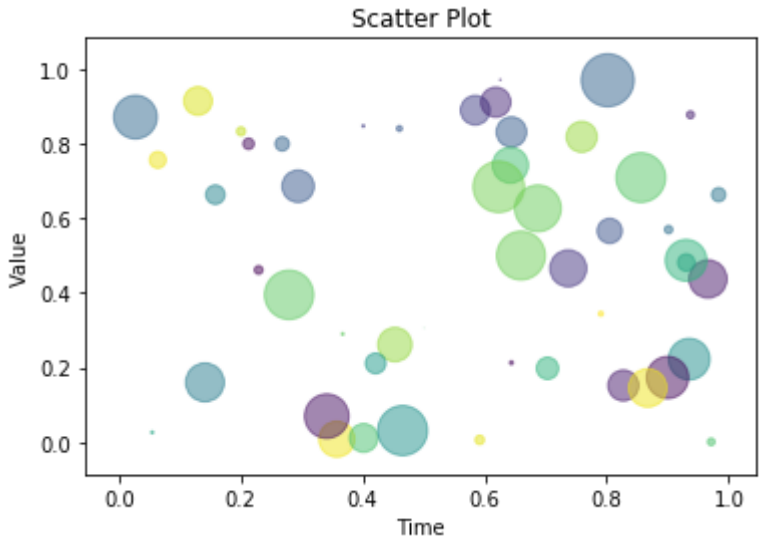
```
# matplotlib
import matplotlib.pyplot as plt
import numpy as np
```

In [345]:

```
N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = np.pi * (15 * np.random.rand(N))**2  # 半徑 0~15
```

In [346]:

```
# 散佈圖 plt.scatter
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.title('Scatter Plot')
plt.xlabel('Time')
plt.ylabel('Value')
plt.savefig('random.plot.png') # 儲存為 png
plt.savefig('random.plot.pdf') # 儲存為 pdf
```

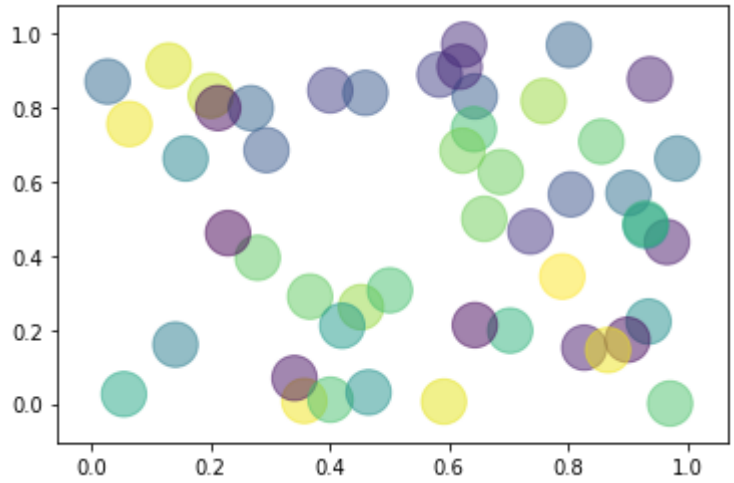


In [347]:

```
# 固定大小
plt.scatter(x, y, s=500, c=colors, alpha=0.5)
```

Out[347]:

```
<matplotlib.collections.PathCollection at 0x2423975dac8>
```



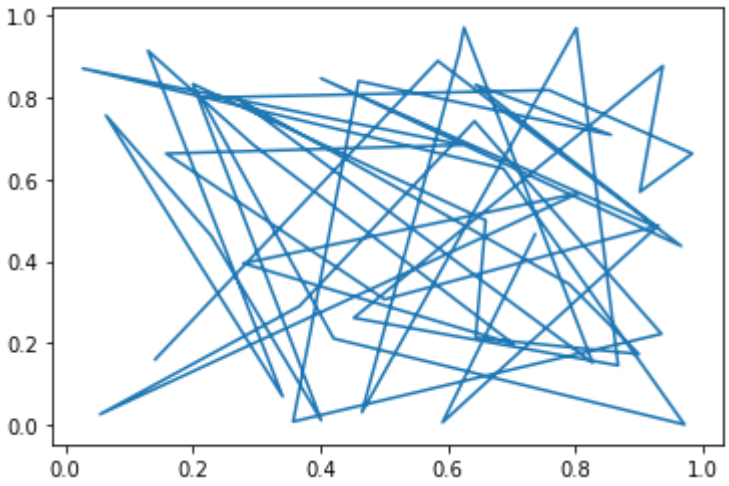
線圖

In [348]:

```
# 線圖 plt.plot
plt.plot(x,y)
```

Out[348]:

```
[<matplotlib.lines.Line2D at 0x242397bb788>]
```

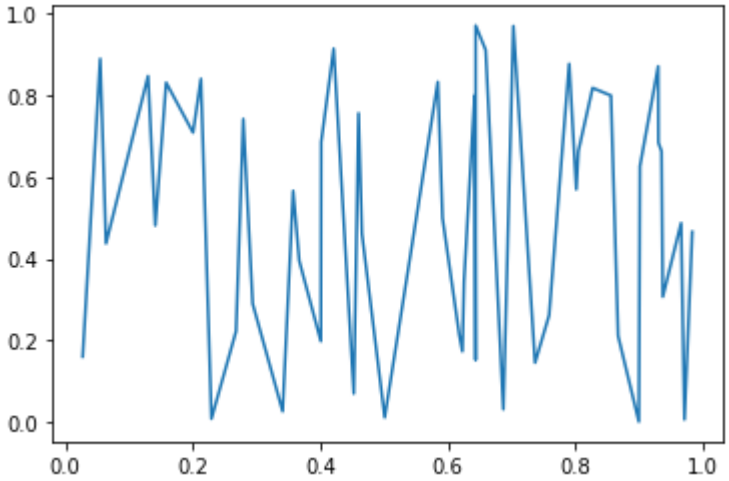


In [349]:

```
# 排序後線圖
plt.plot(sorted(x), y)
```

Out[349]:

```
[<matplotlib.lines.Line2D at 0x24239824f48>]
```

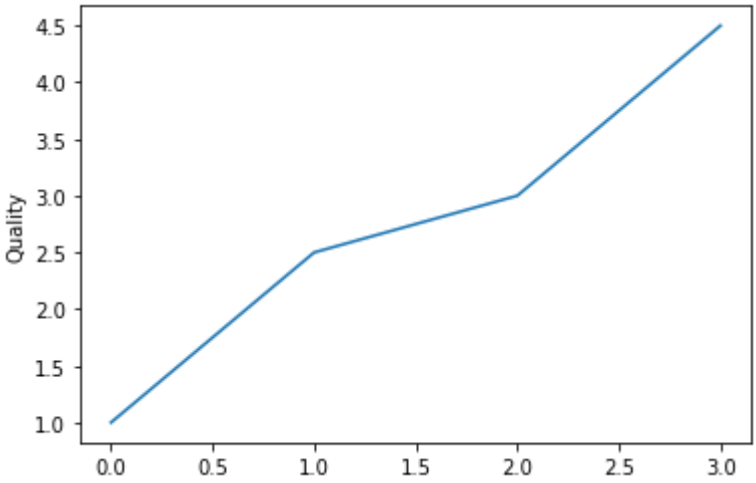


In [350]:

```
# 線圖
plt.plot([1,2.5,3,4.5])   # x 軸: 0,1,2,3
plt.ylabel("Quality")
```

Out[350]:

```
Text(0, 0.5, 'Quality')
```



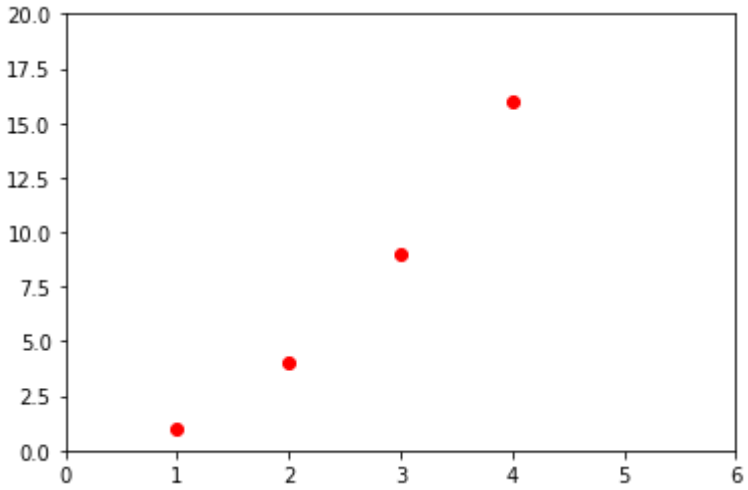
點圖

In [351]:

```
# 點圖
plt.plot([1,2,3,4], [1,4,9,16], 'ro')  # r:red, o:circle marker
plt.axis([0, 6, 0, 20]) # set xlim and ylim
```

Out[351]:

```
[0, 6, 0, 20]
```

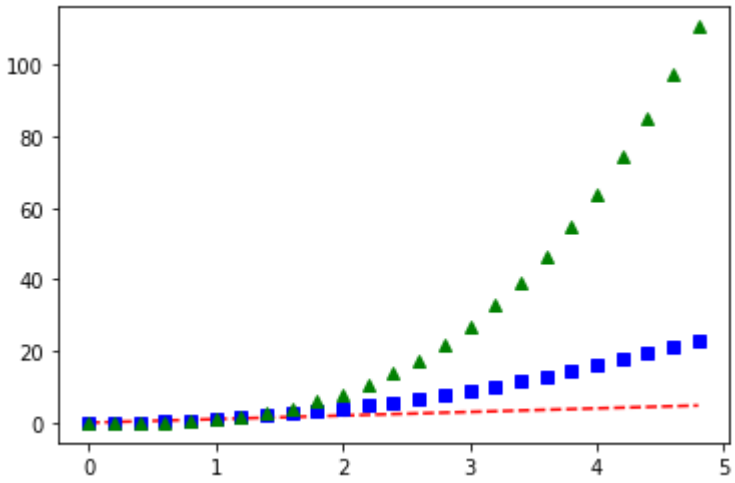


In [352]:

```
# 群組點圖
t = np.arange(0., 5., 0.2) # 等差級數, 區間包括啟始, 不包括結束
# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
```

Out[352]:

```
[<matplotlib.lines.Line2D at 0x2423996a3c8>,
 <matplotlib.lines.Line2D at 0x2423996a5c8>,
 <matplotlib.lines.Line2D at 0x2423996a7c8>]
```

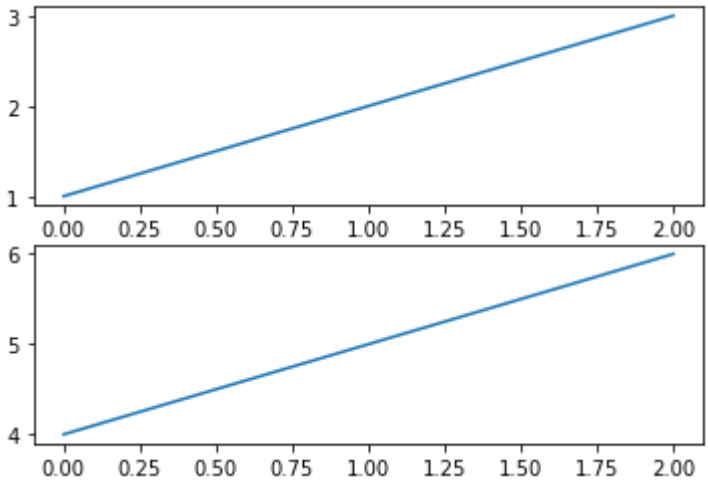


In [353]:

```
# 群組線圖
plt.figure(1)                # the first figure

# 211: (nrow, ncol, plot_number)
plt.subplot(211)             # the first subplot in the first figure
plt.plot([1, 2, 3])
plt.subplot(212)             # the second subplot in the first figure
plt.plot([4, 5, 6])
```

Out[353]:

```
[<matplotlib.lines.Line2D at 0x24238996888>]
```



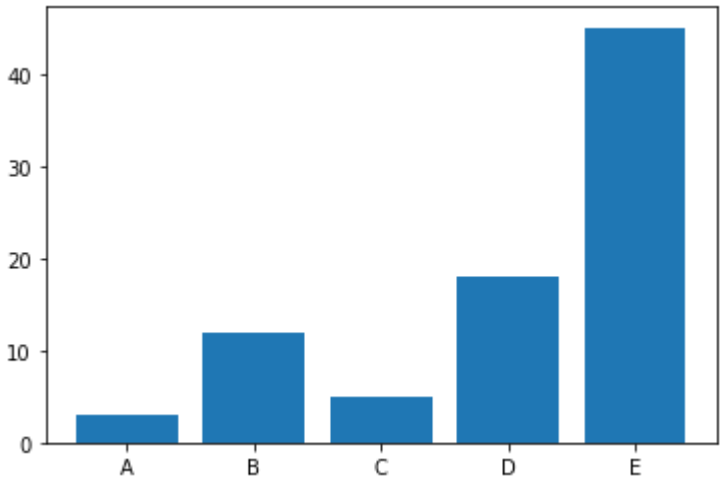
長條圖

In [354]:

```
# Make a fake dataset:
height = [3, 12, 5, 18, 45]
bars = ('A', 'B', 'C', 'D', 'E')
y_pos = np.arange(len(bars))

# Create bars
plt.bar(y_pos, height)

# Create names on the x-axis
plt.xticks(y_pos, bars)

# 參考資料 https://python-graph-gallery.com/
```

Out[354]:

```
([<matplotlib.axis.XTick at 0x242398c9f08>,
  <matplotlib.axis.XTick at 0x2423968fc08>,
  <matplotlib.axis.XTick at 0x2423968f788>,
  <matplotlib.axis.XTick at 0x2423a9b0548>,
  <matplotlib.axis.XTick at 0x2423a9b0c08>],
 <a list of 5 Text xticklabel objects>)
```



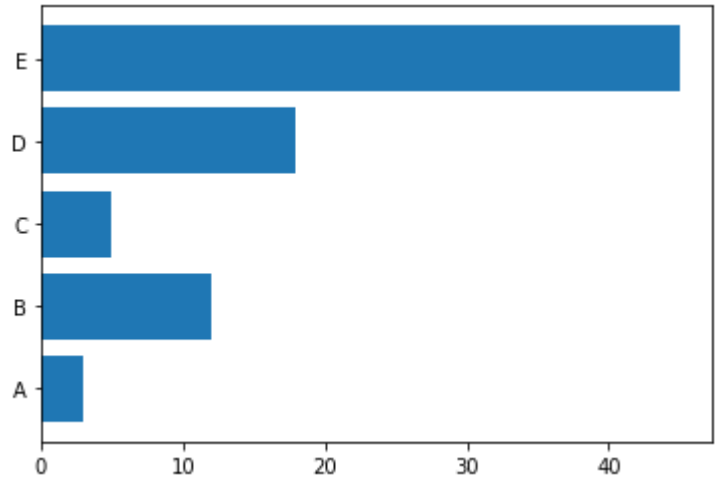
水平長條圖

In [355]:

```
# Make fake dataset
height = [3, 12, 5, 18, 45]
bars = ('A', 'B', 'C', 'D', 'E')
y_pos = np.arange(len(bars))

# Create horizontal bars
plt.barh(y_pos, height)

# Create names on the y-axis
plt.yticks(y_pos, bars)
```

Out[355]:

```
([<matplotlib.axis.YTick at 0x2423a9e61c8>,
  <matplotlib.axis.YTick at 0x2423969d948>,
  <matplotlib.axis.YTick at 0x2423a9e0488>,
  <matplotlib.axis.YTick at 0x2423aa16348>,
  <matplotlib.axis.YTick at 0x2423aa16c08>],
 <a list of 5 Text yticklabel objects>)
```



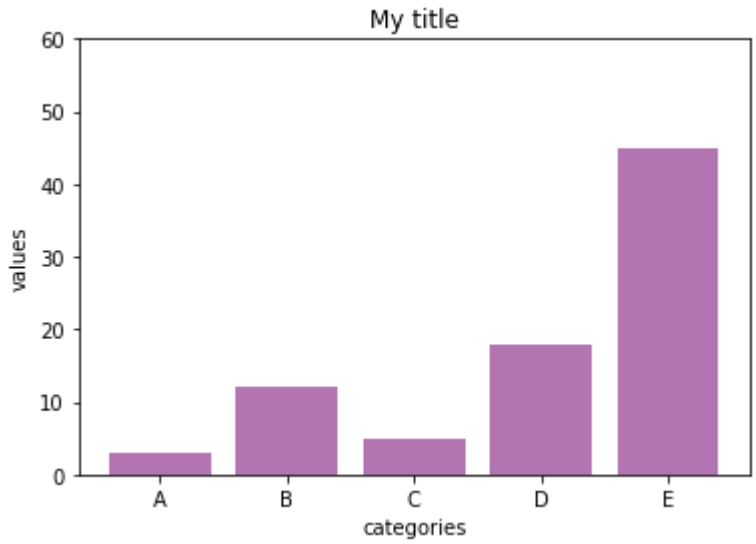
長條圖-有標題

In [356]:

```
# Fake dataset
height = [3, 12, 5, 18, 45]
bars = ('A', 'B', 'C', 'D', 'E')
y_pos = np.arange(len(bars))

# Create bars and choose color
plt.bar(y_pos, height, color = (0.5,0.1,0.5,0.6))

# Add title and axis names
plt.title('My title')
plt.xlabel('categories')
plt.ylabel('values')

# Limits for the Y axis
plt.ylim(0,60)

# Create names
plt.xticks(y_pos, bars)
```

Out[356]:

```
([<matplotlib.axis.XTick at 0x2423aa45988>,
  <matplotlib.axis.XTick at 0x2423aa45548>,
  <matplotlib.axis.XTick at 0x2423aa450c8>,
  <matplotlib.axis.XTick at 0x2423aa7bc08>,
  <matplotlib.axis.XTick at 0x2423aa7e308>],
 <a list of 5 Text xticklabel objects>)
```



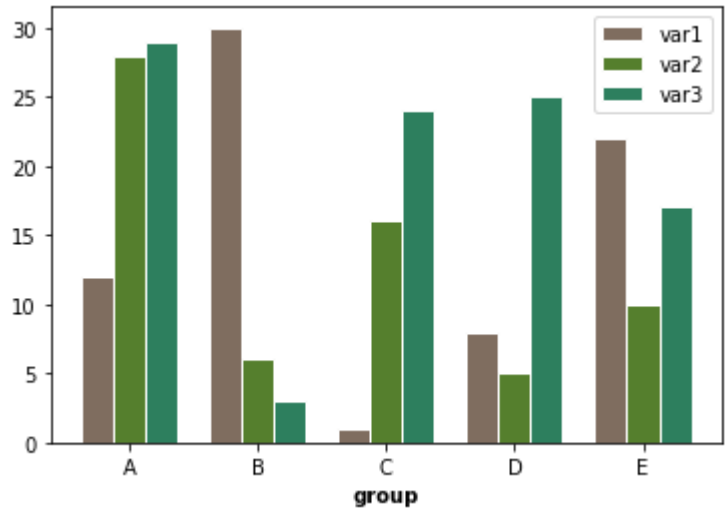
群組長條圖

In [357]:

```
# set width of bar
barWidth = 0.25

# set height of bar
bars1 = [12, 30, 1, 8, 22]
bars2 = [28, 6, 16, 5, 10]
bars3 = [29, 3, 24, 25, 17]

# Set position of bar on X axis
r1 = np.arange(len(bars1))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]

# Make the plot
plt.bar(r1, bars1, color='#7f6d5f', width=barWidth, edgecolor='white', label='var1')
plt.bar(r2, bars2, color='#557f2d', width=barWidth, edgecolor='white', label='var2')
plt.bar(r3, bars3, color='#2d7f5e', width=barWidth, edgecolor='white', label='var3')

# Add xticks on the middle of the group bars
plt.xlabel('group', fontweight='bold')
plt.xticks([r + barWidth for r in range(len(bars1))], ['A', 'B', 'C', 'D', 'E'])

# Create legend & Show graphic
plt.legend()
```

Out[357]:

```
<matplotlib.legend.Legend at 0x2423aa7e8c8>
```



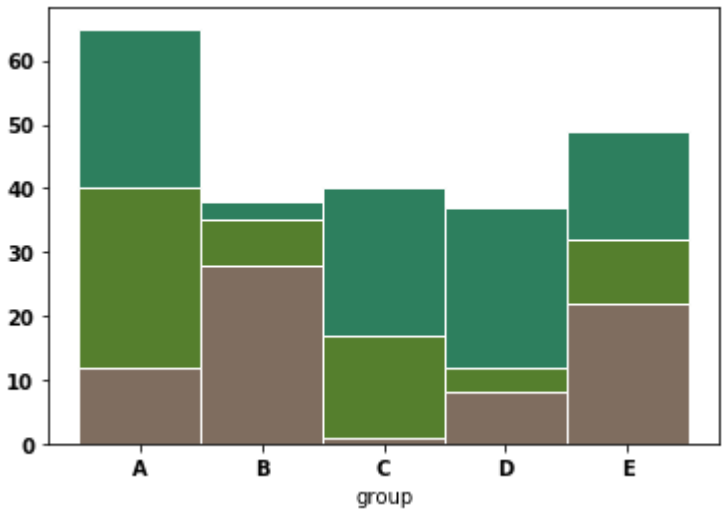
堆疊長條圖

In [358]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
import pandas as pd

# y-axis in bold
rc('font', weight='bold')

# Values of each group
bars1 = [12, 28, 1, 8, 22]
bars2 = [28, 7, 16, 4, 10]
bars3 = [25, 3, 23, 25, 17]

# Heights of bars1 + bars2
bars = np.add(bars1, bars2).tolist()

# The position of the bars on the x-axis
r = [0,1,2,3,4]

# Names of group and bar width
names = ['A','B','C','D','E']
barWidth = 1

# Create brown bars
plt.bar(r, bars1, color='#7f6d5f', edgecolor='white', width=barWidth)
# Create green bars (middle), on top of the firs ones
plt.bar(r, bars2, bottom=bars1, color='#557f2d', edgecolor='white', width=barWidth)
# Create green bars (top)
plt.bar(r, bars3, bottom=bars, color='#2d7f5e', edgecolor='white', width=barWidth)

# Custom X axis
plt.xticks(r, names, fontweight='bold')
plt.xlabel("group")
```

Out[358]:

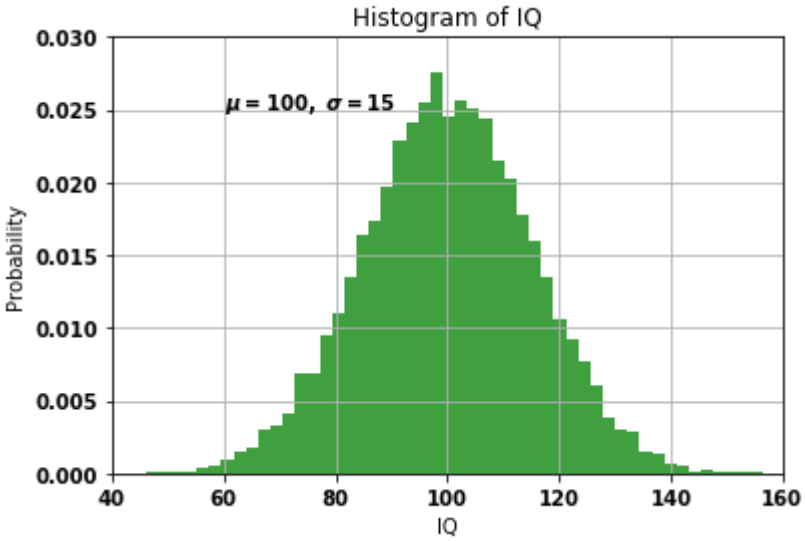
```
Text(0.5, 0, 'group')
```



直方圖

In [359]:

```
# 直方圖 histogram plot
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

n, bins, patches = plt.hist(x, 50, density=True, facecolor='g', alpha=0.75)

plt.xlabel("IQ")
plt.ylabel("Probability")
plt.title("Histogram of IQ")
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
```



繪圖中文字型

In [360]:

```
from matplotlib.font_manager import FontProperties
font = FontProperties(fname=r"c:\windows\fonts\mingliu.ttc", size=12)
import scipy as sp
```

In [361]:

```
# 匯入資料
# https://github.com/rwepa/DataDemo/blob/master/web_traffic.csv
myData = sp.genfromtxt("data/web_traffic.csv", delimiter="\t")
print(myData[:6])
myData.ndim # 2個維度
myData.shape # same as print(myData.shape) 743*2
```

```
[[1.000e+00 2.272e+03]
 [2.000e+00       nan]
 [3.000e+00 1.386e+03]
 [4.000e+00 1.365e+03]
 [5.000e+00 1.488e+03]
 [6.000e+00 1.337e+03]]
```

Out[361]:

```
(743, 2)
```

In [362]:

```
x = myData[:,0] # 743*1
y = myData[:,1] # 743*1

#檢查是否有Na
sp.sum(sp.isnan(y)) # 有8個值是nan
print("Number of invalid entries:", sp.sum(sp.isnan(y)))
```

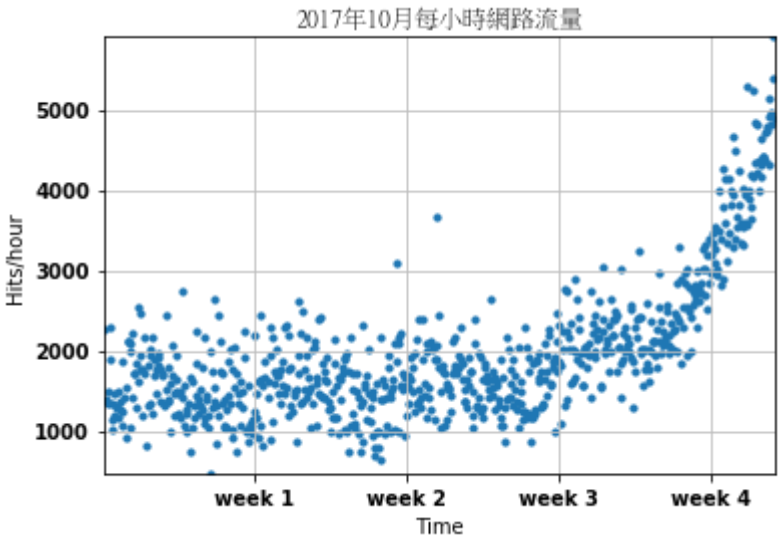
```
Number of invalid entries: 8
```

In [363]:

```
#取出非Na
x = x[~sp.isnan(y)] # 743-8=735
y = y[~sp.isnan(y)]
```

In [364]:

```
#繪圖
plt.scatter(x, y, s=10)
plt.title(u"2017年10月每小時網路流量", fontproperties=font) # 中文顯示
plt.xlabel("Time")
plt.ylabel("Hits/hour")
plt.xticks([w*7*24 for w in range(10)], ['week %i' % w for w in range(10)])
plt.autoscale(tight=True)

#加上網格線
plt.grid(True, linestyle='-', color='0.75')

#output png file
plt.savefig(u"2017年10月每小時網路流量.png", dpi=300, format="png")
plt.show()

#plt.xticks() :設定x軸刻度, e.g. plt.xticks([10,20,30,40,50])
#plt.yticks() :設定y軸刻度
```



基本顏色 **color**

- b: blue
- g: green
- r: red
- c: cyan
- m: magenta
- y: yellow
- k: black
- w: white

參考資料: https://matplotlib.org/users/colors.html (https://matplotlib.org/users/colors.html)

線的型態與符號設定

- '-' solid line style
- '--' dashed line style
- '-.' dash-dot line style

- ':' dotted line style
- '.' point marker
- ',' pixel marker
- 'o' circle marker
- 'v' triangle_down marker
- '^' triangle_up marker
- '<' triangle_left marker
- '>' triangle_right marker
- '1' tri_down marker
- '2' tri_up marker
- '3' tri_left marker
- '4' tri_right marker
- 's' square marker
- 'p' pentagon marker
- '*' star marker
- 'h' hexagon1 marker
- 'H' hexagon2 marker
- '+' plus marker
- 'x' x marker
- 'D' diamond marker
- 'd' thin_diamond marker
- '|' vline marker
- '_' hline marker

參考資料 http://matplotlib.org/users/pyplot_tutorial.html (http://matplotlib.org/users/pyplot_tutorial.html)

平行座標軸 **Parallel coordinates graph**

In [365]:

```
# 資料理解
import pandas as pd
# from pandas import DataFrame
```

In [366]:

```
import matplotlib.pyplot as plot
target_url = ("https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/conne

#資料有 208列, 61行
#第61行: R: rock  岩石
#第61行: M: mine 水雷
```

In [367]:

```
# read rocks versus mines data into pandas data frame
rocksVMines = pd.read_csv(target_url,header=None, prefix="V") # 208*61

#print head and tail of data frame
print(rocksVMines.head(n=10))
print(rocksVMines.tail(n=10))
```

```
      V0      V1      V2      V3      V4      V5      V6      V7      V8
\
0  0.0200  0.0371  0.0428  0.0207  0.0954  0.0986  0.1539  0.1601  0.3109
1  0.0453  0.0523  0.0843  0.0689  0.1183  0.2583  0.2156  0.3481  0.3337
2  0.0262  0.0582  0.1099  0.1083  0.0974  0.2280  0.2431  0.3771  0.5598
3  0.0100  0.0171  0.0623  0.0205  0.0205  0.0368  0.1098  0.1276  0.0598
4  0.0762  0.0666  0.0481  0.0394  0.0590  0.0649  0.1209  0.2467  0.3564
5  0.0286  0.0453  0.0277  0.0174  0.0384  0.0990  0.1201  0.1833  0.2105
6  0.0317  0.0956  0.1321  0.1408  0.1674  0.1710  0.0731  0.1401  0.2083
7  0.0519  0.0548  0.0842  0.0319  0.1158  0.0922  0.1027  0.0613  0.1465
8  0.0223  0.0375  0.0484  0.0475  0.0647  0.0591  0.0753  0.0098  0.0684
9  0.0164  0.0173  0.0347  0.0070  0.0187  0.0671  0.1056  0.0697  0.0962

      V9  ...     V51     V52     V53     V54     V55     V56     V57  \
0  0.2111  ...  0.0027  0.0065  0.0159  0.0072  0.0167  0.0180  0.0084
1  0.2872  ...  0.0084  0.0089  0.0048  0.0094  0.0191  0.0140  0.0049
2  0.6194  ...  0.0232  0.0166  0.0095  0.0180  0.0244  0.0316  0.0164
3  0.1264  ...  0.0121  0.0036  0.0150  0.0085  0.0073  0.0050  0.0044
4  0.4459  ...  0.0031  0.0054  0.0105  0.0110  0.0015  0.0072  0.0048
5  0.3039  ...  0.0045  0.0014  0.0038  0.0013  0.0089  0.0057  0.0027
6  0.3513  ...  0.0201  0.0248  0.0131  0.0070  0.0138  0.0092  0.0143
7  0.2838  ...  0.0081  0.0120  0.0045  0.0121  0.0097  0.0085  0.0047
8  0.1487  ...  0.0145  0.0128  0.0145  0.0058  0.0049  0.0065  0.0093
9  0.0251  ...  0.0090  0.0223  0.0179  0.0084  0.0068  0.0032  0.0035

      V58     V59  V60
0  0.0090  0.0032    R
1  0.0052  0.0044    R
2  0.0095  0.0078    R
3  0.0040  0.0117    R
4  0.0107  0.0094    R
5  0.0051  0.0062    R
6  0.0036  0.0103    R
7  0.0048  0.0053    R
8  0.0059  0.0022    R
9  0.0056  0.0040    R

[10 rows x 61 columns]
         V0      V1      V2      V3      V4      V5      V6      V7      V
8  \
198  0.0238  0.0318  0.0422  0.0399  0.0788  0.0766  0.0881  0.1143  0.159
4
199  0.0116  0.0744  0.0367  0.0225  0.0076  0.0545  0.1110  0.1069  0.170
8
200  0.0131  0.0387  0.0329  0.0078  0.0721  0.1341  0.1626  0.1902  0.261
0
201  0.0335  0.0258  0.0398  0.0570  0.0529  0.1091  0.1709  0.1684  0.186
5
202  0.0272  0.0378  0.0488  0.0848  0.1127  0.1103  0.1349  0.2337  0.311
3
203  0.0187  0.0346  0.0168  0.0177  0.0393  0.1630  0.2028  0.1694  0.232
```

```
8
204  0.0323  0.0101  0.0298  0.0564  0.0760  0.0958  0.0990  0.1018  0.103
0
205  0.0522  0.0437  0.0180  0.0292  0.0351  0.1171  0.1257  0.1178  0.125
8
206  0.0303  0.0353  0.0490  0.0608  0.0167  0.1354  0.1465  0.1123  0.194
5
207  0.0260  0.0363  0.0136  0.0272  0.0214  0.0338  0.0655  0.1400  0.184
3

         V9  ...     V51     V52     V53     V54     V55     V56     V57
\
198  0.2048  ...  0.0096  0.0071  0.0084  0.0038  0.0026  0.0028  0.0013
199  0.2271  ...  0.0141  0.0103  0.0100  0.0034  0.0026  0.0037  0.0044
200  0.3193  ...  0.0150  0.0076  0.0032  0.0037  0.0071  0.0040  0.0009
201  0.2660  ...  0.0120  0.0039  0.0053  0.0062  0.0046  0.0045  0.0022
202  0.3997  ...  0.0091  0.0045  0.0043  0.0043  0.0098  0.0054  0.0051
203  0.2684  ...  0.0116  0.0098  0.0199  0.0033  0.0101  0.0065  0.0115
204  0.2154  ...  0.0061  0.0093  0.0135  0.0063  0.0063  0.0034  0.0032
205  0.2529  ...  0.0160  0.0029  0.0051  0.0062  0.0089  0.0140  0.0138
206  0.2354  ...  0.0086  0.0046  0.0126  0.0036  0.0035  0.0034  0.0079
207  0.2354  ...  0.0146  0.0129  0.0047  0.0039  0.0061  0.0040  0.0036

        V58     V59  V60
198  0.0035  0.0060    M
199  0.0057  0.0035    M
200  0.0015  0.0085    M
201  0.0005  0.0031    M
202  0.0065  0.0103    M
203  0.0193  0.0157    M
204  0.0062  0.0067    M
205  0.0077  0.0031    M
206  0.0036  0.0048    M
207  0.0061  0.0115    M

[10 rows x 61 columns]
```

In [368]:

```
# print summary of data frame
summary = rocksVMines.describe()
print(summary)
```

```
              V0          V1          V2          V3          V4
V5  \
count  208.000000  208.000000  208.000000  208.000000  208.000000  208.000
000
mean     0.029164    0.038437    0.043832    0.053892    0.075202    0.104
570
std      0.022991    0.032960    0.038428    0.046528    0.055552    0.059
105
min      0.001500    0.000600    0.001500    0.005800    0.006700    0.010
200
25%      0.013350    0.016450    0.018950    0.024375    0.038050    0.067
025
50%      0.022800    0.030800    0.034300    0.044050    0.062500    0.092
150
75%      0.035550    0.047950    0.057950    0.064500    0.100275    0.134
125
max      0.137100    0.233900    0.305900    0.426400    0.401000    0.382
300

              V6          V7          V8          V9  ...         V50  \
count  208.000000  208.000000  208.000000  208.000000  ...  208.000000
mean     0.121747    0.134799    0.178003    0.208259  ...    0.016069
std      0.061788    0.085152    0.118387    0.134416  ...    0.012008
min      0.003300    0.005500    0.007500    0.011300  ...    0.000000
25%      0.080900    0.080425    0.097025    0.111275  ...    0.008425
50%      0.106950    0.112100    0.152250    0.182400  ...    0.013900
75%      0.154000    0.169600    0.233425    0.268700  ...    0.020825
max      0.372900    0.459000    0.682800    0.710600  ...    0.100400

             V51         V52         V53         V54         V55
V56  \
count  208.000000  208.000000  208.000000  208.000000  208.000000  208.000
000
mean     0.013420    0.010709    0.010941    0.009290    0.008222    0.007
820
std      0.009634    0.007060    0.007301    0.007088    0.005736    0.005
785
min      0.000800    0.000500    0.001000    0.000600    0.000400    0.000
300
25%      0.007275    0.005075    0.005375    0.004150    0.004400    0.003
700
50%      0.011400    0.009550    0.009300    0.007500    0.006850    0.005
950
75%      0.016725    0.014900    0.014500    0.012100    0.010575    0.010
425
max      0.070900    0.039000    0.035200    0.044700    0.039400    0.035
500

             V57         V58         V59
count  208.000000  208.000000  208.000000
mean     0.007949    0.007941    0.006507
std      0.006470    0.006181    0.005031
min      0.000300    0.000100    0.000600
25%      0.003600    0.003675    0.003100
```

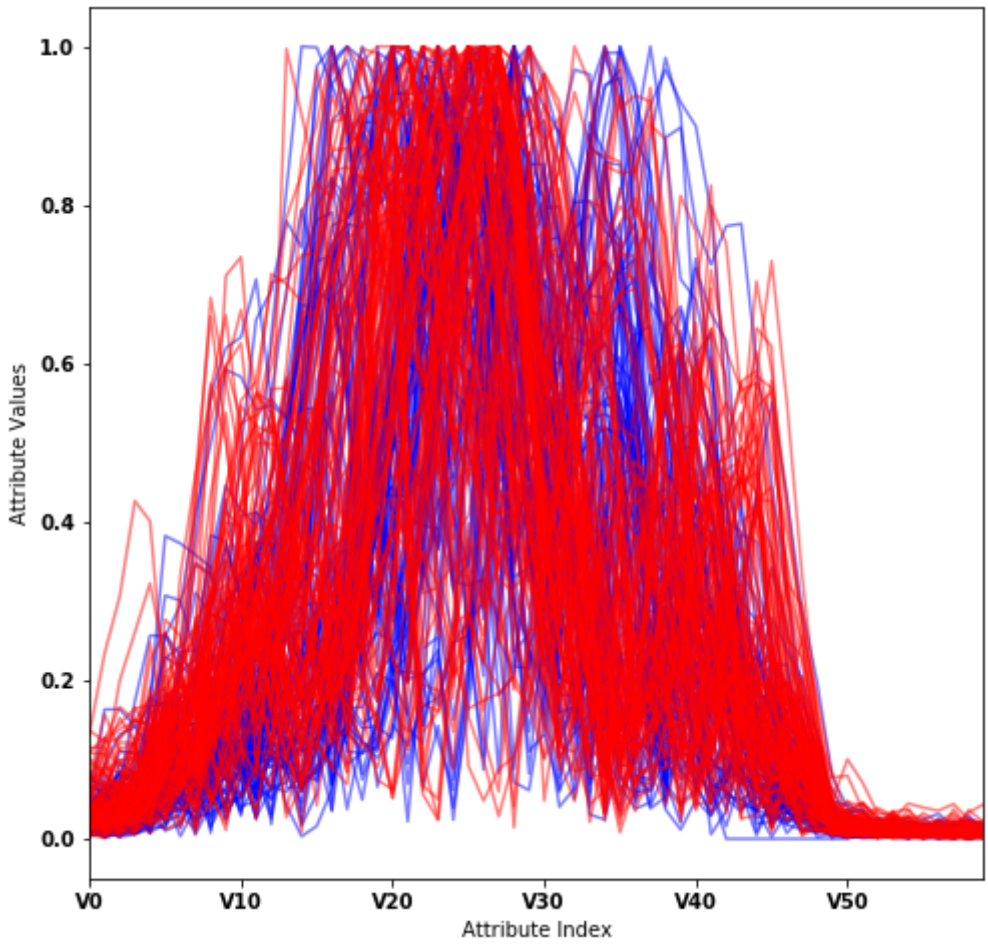
```
50%      0.005800    0.006400    0.005300
75%      0.010350    0.010325    0.008525
max      0.044000    0.036400    0.043900

[8 rows x 60 columns]
```

In [369]:

```
#平行座標軸 Parallel coordinates graph

for i in range(208):
    #assign color based on color based on "M" or "R" labels
    if rocksVMines.iat[i,60] == "M":
        pcolor = "red"
    else:
        pcolor = "blue"

    #plot rows of data as if they were series data
    dataRow = rocksVMines.iloc[i,0:60]
    plot.rcParams["figure.figsize"] = (8, 8) #(寬, 高)
    dataRow.plot(color=pcolor, alpha=0.5)
    #plot.figure(figsize=(16, 16))

plot.xlabel("Attribute Index")

plot.ylabel(("Attribute Values"))
```

Out[369]:

```
Text(0, 0.5, 'Attribute Values')
```



散佈圖矩陣 scatter_matrix{pandas}

In [370]:

```
import pandas as pd
from pandas.plotting import scatter_matrix
import numpy as np
```

In [371]:

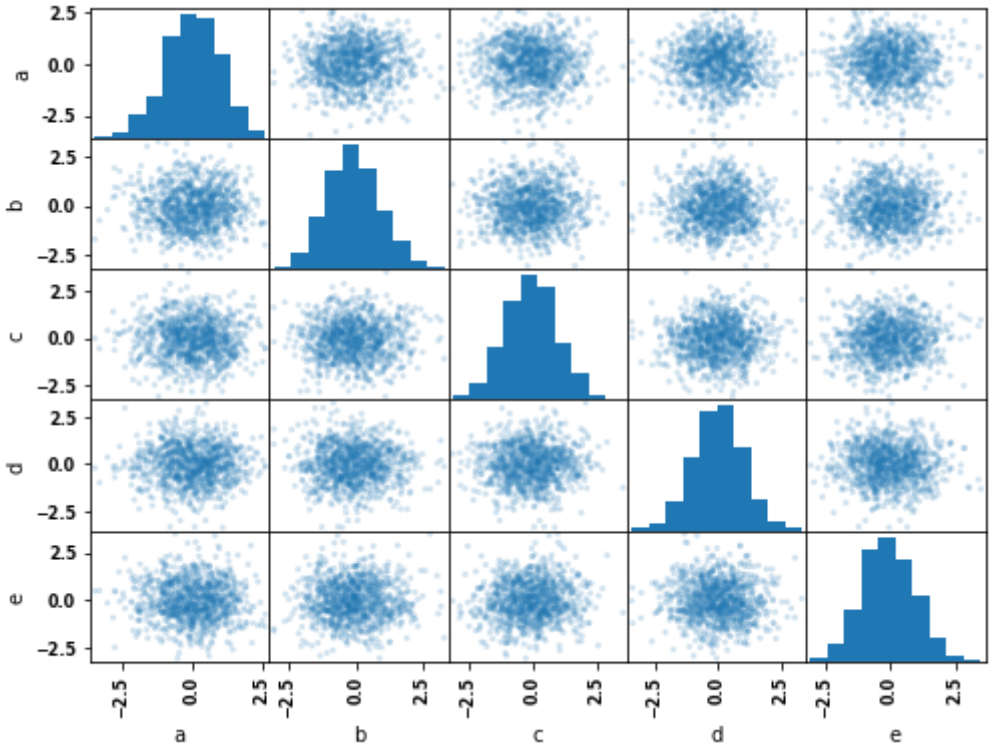
```
df = pd.DataFrame(np.random.randn(1000, 5), columns=['a', 'b', 'c', 'd', 'e'])
```

In [372]:

```
# diagonal matrix with histogram

scatter_matrix(df, alpha=0.2, figsize=(8, 6))
```

Out[372]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002423AF3FCC8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B138808
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423AF98D88
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423AFCFE88
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B007FC8
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002423B044108
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B07E188
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B0B52C8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B0BDE48
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B199048
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002423B200608
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B23DE48
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B270788
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B2A98C8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B2E09C8
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002423B319AC8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B353B88
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B38AC88
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B3C3D88
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B3FDEC8
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002423B43A048
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B4720C8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B4AB1C8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B4E4308
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002423B51D408
>]],
      dtype=object)
```



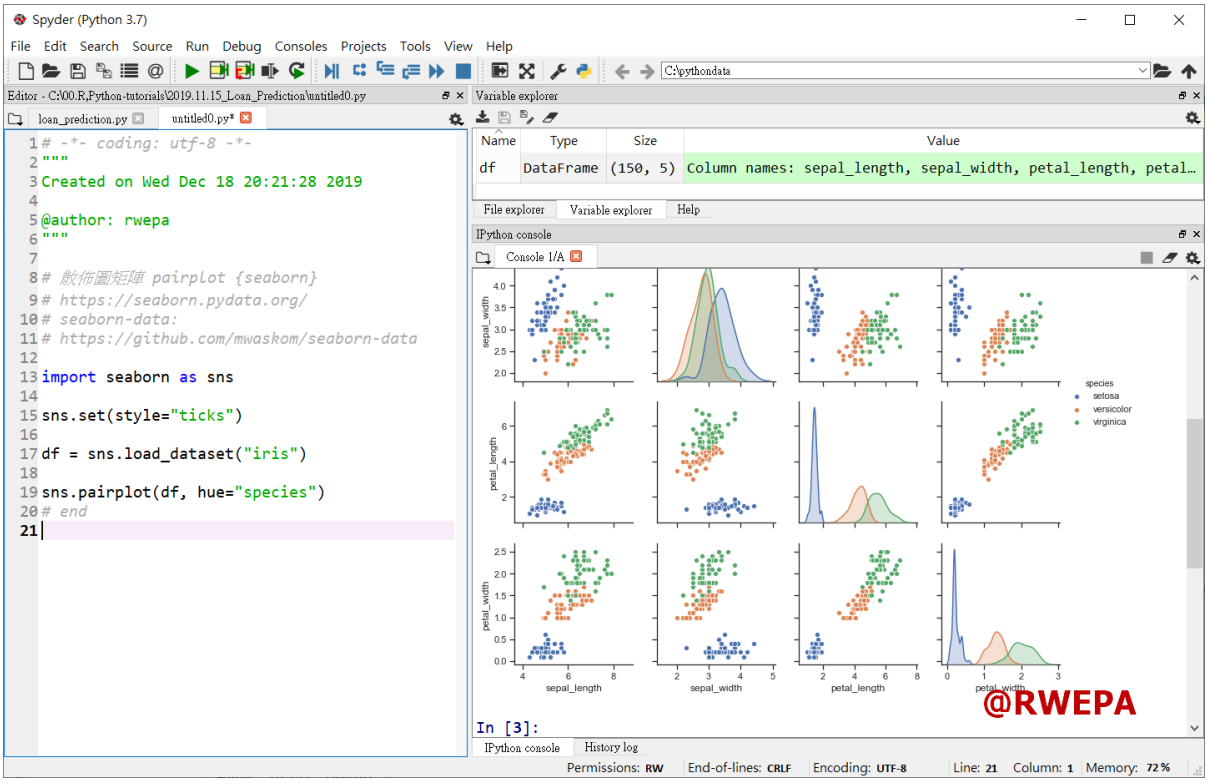
散佈圖矩陣 **pairplot {seaborn}**

https://seaborn.pydata.org/ (https://seaborn.pydata.org/)

seaborn-data: https://github.com/mwaskom/seaborn-data (https://github.com/mwaskom/seaborn-data)

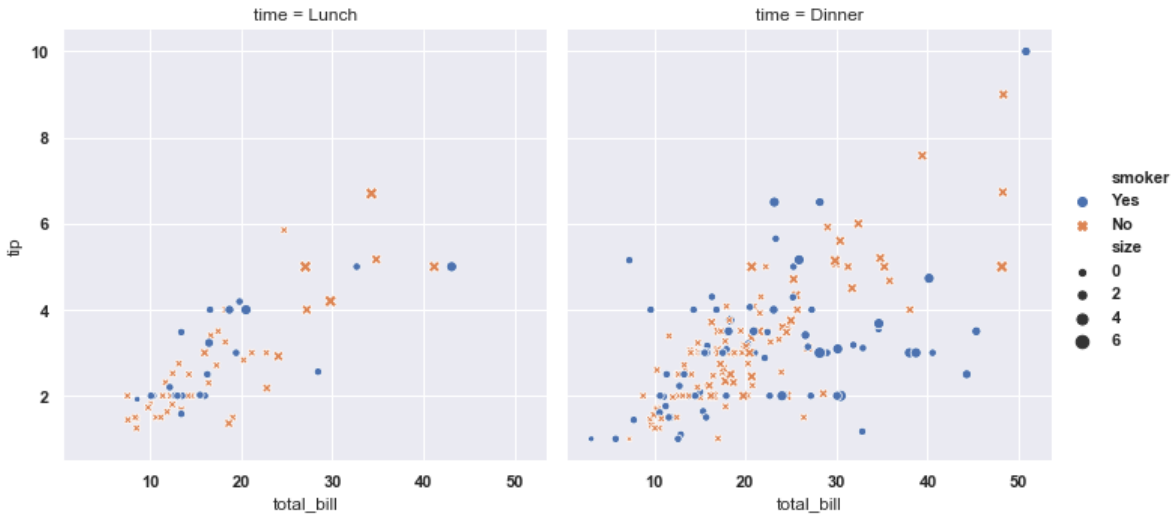
6.4 seaborn 模組繪圖

使用 **Spyder + seaborn** 模組



In [373]:

```
import seaborn as sns

sns.set()

tips = sns.load_dataset("tips")

sns.relplot(x="total_bill", y="tip", col="time",
            hue="smoker", style="smoker", size="size",
            data=tips);
```



In [374]:

```
dots = sns.load_dataset("dots")
dots
```

Out[374]:

	align	choice	time	coherence	firing_rate
0	dots	T1	-80	0.0	33.189967
1	dots	T1	-80	3.2	31.691726
2	dots	T1	-80	6.4	34.279840
3	dots	T1	-80	12.8	32.631874
4	dots	T1	-80	25.6	35.060487
...	...	...	...	...	...
843	sacc	T2	300	3.2	33.281734
844	sacc	T2	300	6.4	27.583979
845	sacc	T2	300	12.8	28.511530
846	sacc	T2	300	25.6	27.009804
847	sacc	T2	300	51.2	30.959302

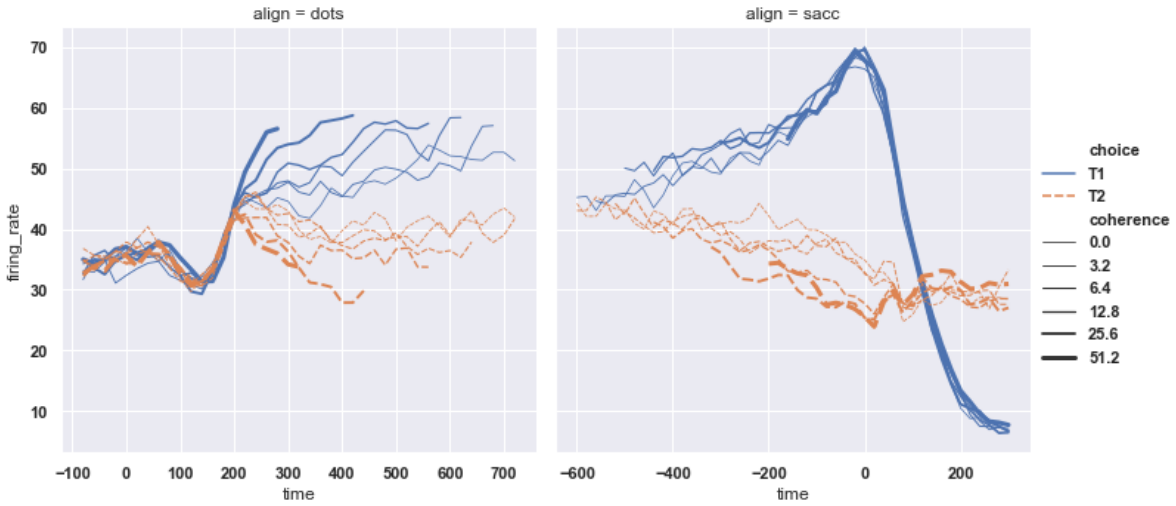
848 rows × 5 columns

In [375]:

```
sns.relplot(x="time", y="firing_rate", col="align",
            hue="choice", size="coherence", style="choice",
            facet_kws=dict(sharex=False),
            kind="line", legend="full", data=dots)
```

Out[375]:

```
<seaborn.axisgrid.FacetGrid at 0x2423b7e3ac8>
```

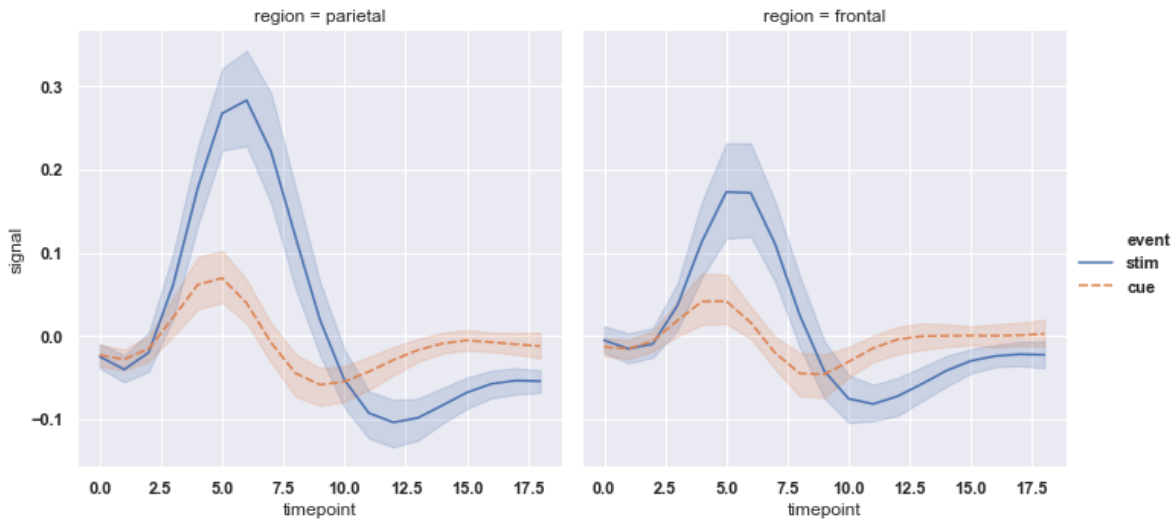


In [376]:

```
fmri = sns.load_dataset("fmri")
sns.relplot(x="timepoint", y="signal", col="region",
            hue="event", style="event",
            kind="line", data=fmri)
```

Out[376]:

```
<seaborn.axisgrid.FacetGrid at 0x2423cecd3c8>
```

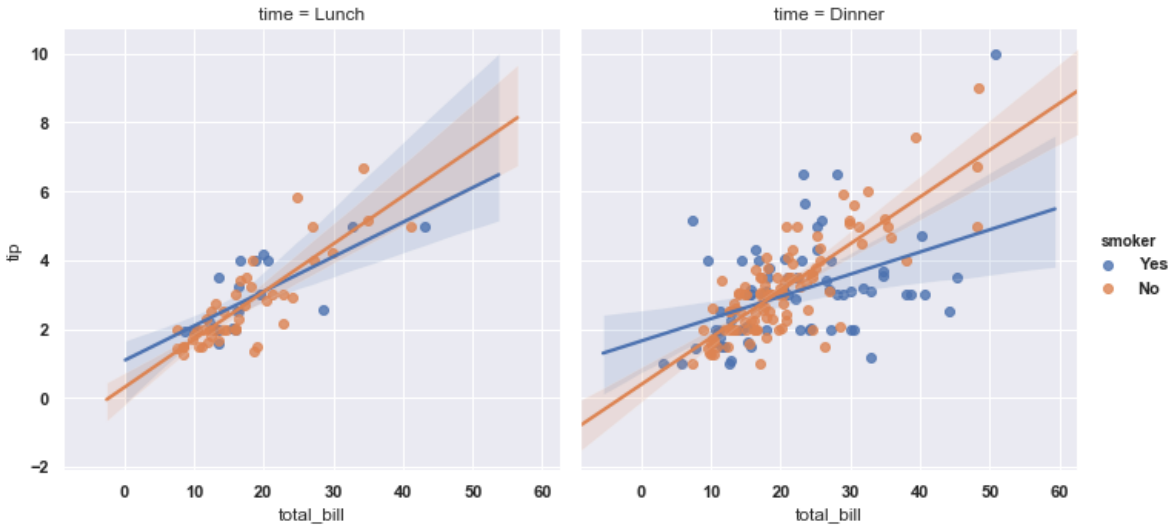


In [377]:

```
sns.lmplot(x="total_bill", y="tip", col="time", hue="smoker",
           data=tips)
```

Out[377]:

```
<seaborn.axisgrid.FacetGrid at 0x2423cfb4948>
```

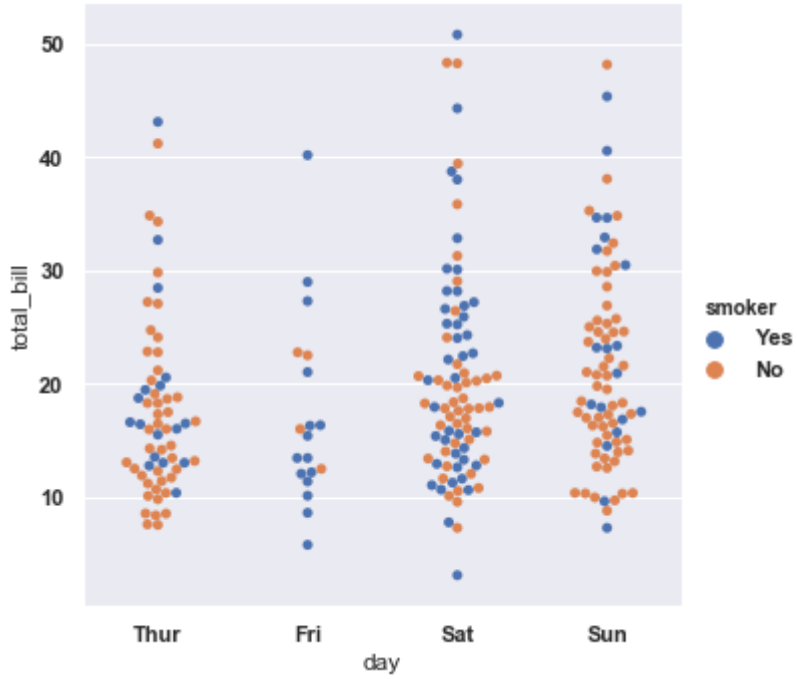


In [378]:

```
sns.catplot(x="day", y="total_bill", hue="smoker", kind="swarm", data=tips)
```

Out[378]:

```
<seaborn.axisgrid.FacetGrid at 0x2423cfaf808>
```

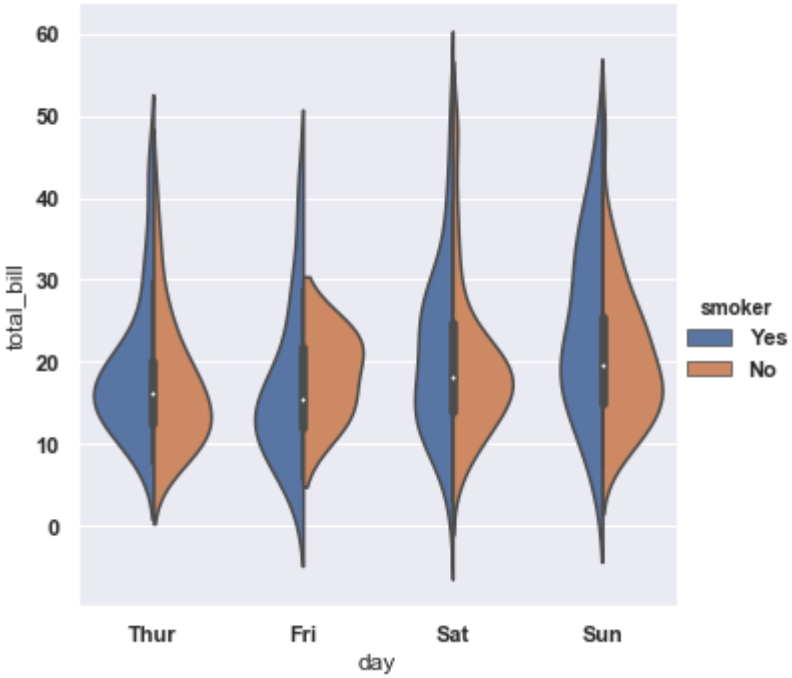


In [379]:

```
sns.catplot(x="day", y="total_bill", hue="smoker", kind="violin", split=True, data=tips)
```

Out[379]:

```
<seaborn.axisgrid.FacetGrid at 0x2423d3d1148>
```

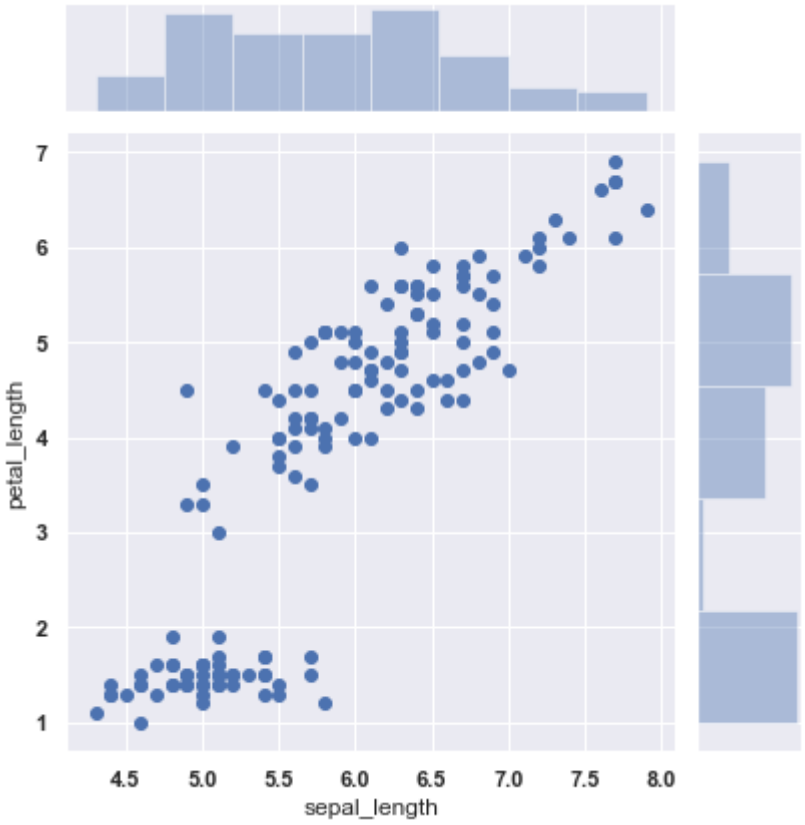


In [380]:

```
iris = sns.load_dataset("iris")

sns.jointplot(x="sepal_length", y="petal_length", data=iris)
```

Out[380]:

```
<seaborn.axisgrid.JointGrid at 0x2423d40d5c8>
```

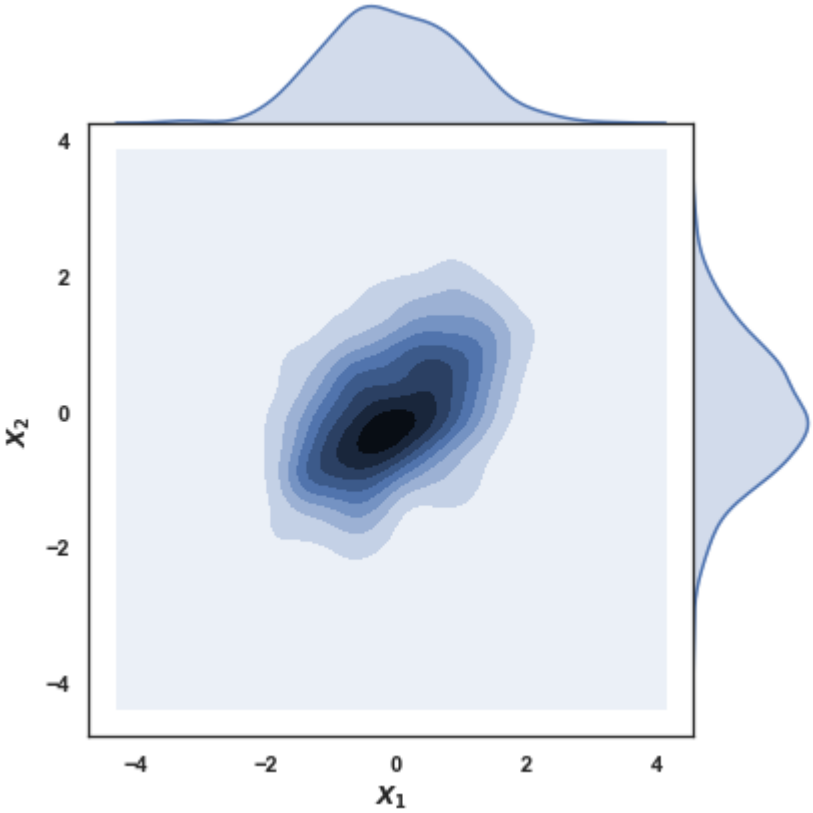


In [381]:

```
# Joint kernel density estimate

import numpy as np
import pandas as pd
import seaborn as sns
sns.set(style="white")

# Generate a random correlated bivariate dataset
rs = np.random.RandomState(5)
mean = [0, 0]
cov = [(1, .5), (.5, 1)]
x1, x2 = rs.multivariate_normal(mean, cov, 500).T
x1 = pd.Series(x1, name="$X_1$")
x2 = pd.Series(x2, name="$X_2$")

# Show the joint distribution using kernel density estimation
sns.jointplot(x1, x2, kind="kde", height=6, space=0)
```

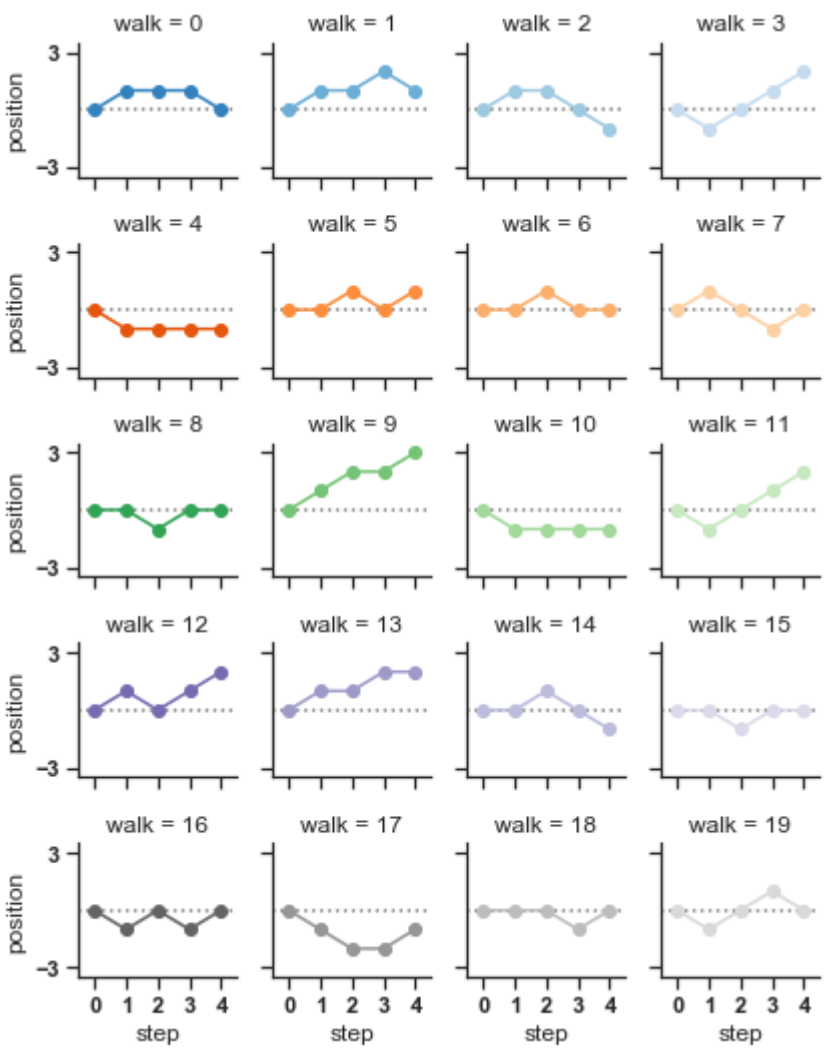
Out[381]:

```
<seaborn.axisgrid.JointGrid at 0x2423d6f1e88>
```



In [382]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

sns.set(style="ticks")

# Create a dataset with many short random walks
rs = np.random.RandomState(4)
pos = rs.randint(-1, 2, (20, 5)).cumsum(axis=1)
pos -= pos[:, 0, np.newaxis]
step = np.tile(range(5), 20)
walk = np.repeat(range(20), 5)
df = pd.DataFrame(np.c_[pos.flat, step, walk],
                  columns=["position", "step", "walk"])

# Initialize a grid of plots with an Axes for each walk
grid = sns.FacetGrid(df, col="walk", hue="walk", palette="tab20c",
                     col_wrap=4, height=1.5)

# Draw a horizontal line to show the starting point
grid.map(plt.axhline, y=0, ls=":", c=".5")

# Draw a line plot to show the trajectory of each random walk
grid.map(plt.plot, "step", "position", marker="o")

# Adjust the tick positions and labels
grid.set(xticks=np.arange(5), yticks=[-3, 3],
         xlim=(-.5, 4.5), ylim=(-3.5, 3.5))

# Adjust the arrangement of the plots
grid.fig.tight_layout(w_pad=1)
```

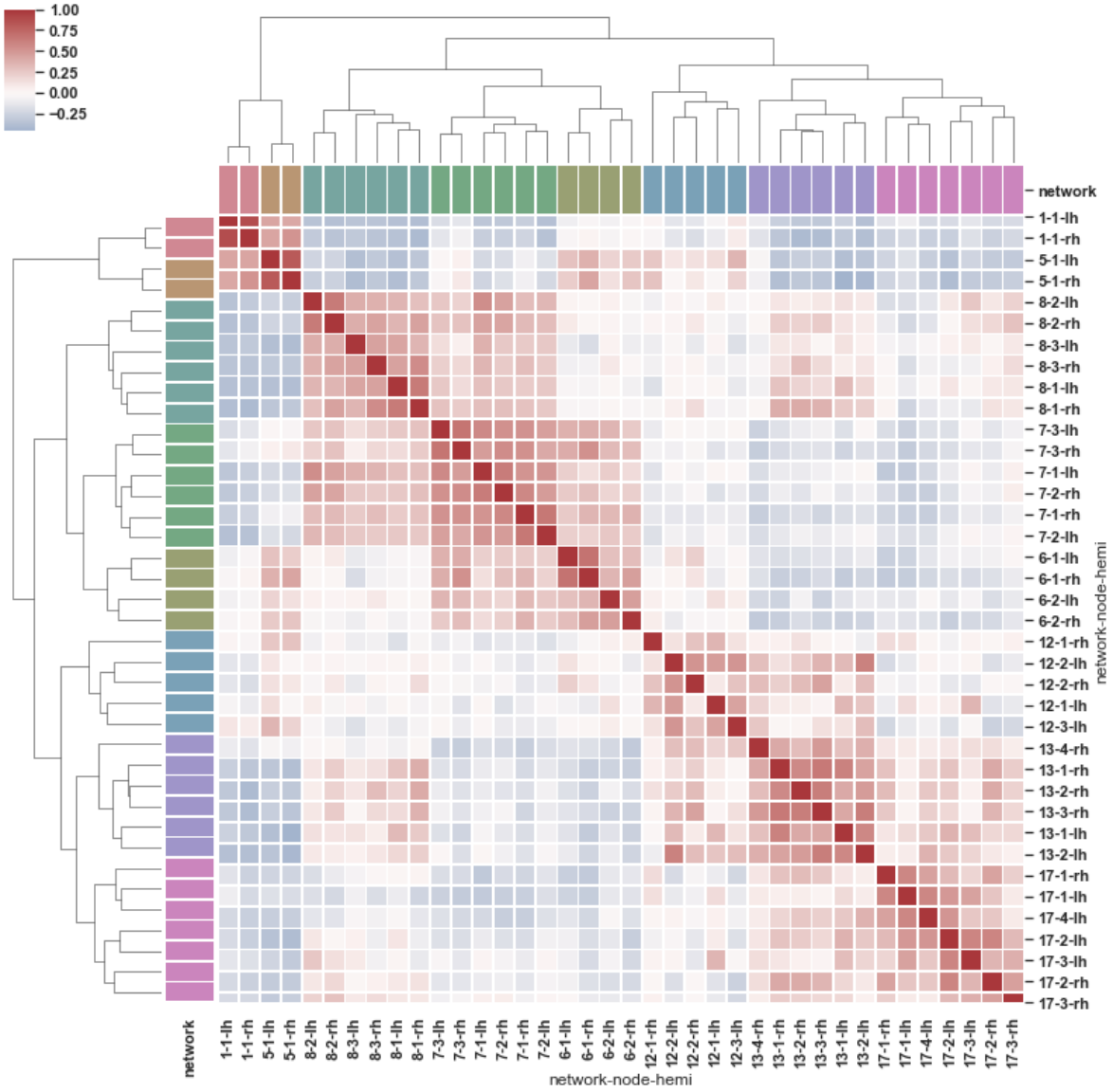

walk = 0
walk = 1
walk = 2
walk = 3
walk = 4
walk = 5
walk = 6
walk = 7
walk = 8
walk = 9
walk = 10
walk = 11
walk = 12
walk = 13
walk = 14
walk = 15
walk = 16
walk = 17
walk = 18
walk = 19
position
3
−3
0 1 2 3 4
step

In [383]:

```
# heatmap

import pandas as pd
import seaborn as sns
sns.set()

# Load the brain networks example dataset
df = sns.load_dataset("brain_networks", header=[0, 1, 2], index_col=0)

# Select a subset of the networks
used_networks = [1, 5, 6, 7, 8, 12, 13, 17]
used_columns = (df.columns.get_level_values("network")
                          .astype(int)
                          .isin(used_networks))
df = df.loc[:, used_columns]

# Create a categorical palette to identify the networks
network_pal = sns.husl_palette(8, s=.45)
network_lut = dict(zip(map(str, used_networks), network_pal))

# Convert the palette to vectors that will be drawn on the side of the matrix
networks = df.columns.get_level_values("network")
network_colors = pd.Series(networks, index=df.columns).map(network_lut)

# Draw the full plot
sns.clustermap(df.corr(), center=0, cmap="vlag",
               row_colors=network_colors, col_colors=network_colors,
               linewidths=.75, figsize=(13, 13))
```

Out[383]:

```
<seaborn.matrix.ClusterGrid at 0x2423ec0e948>
```

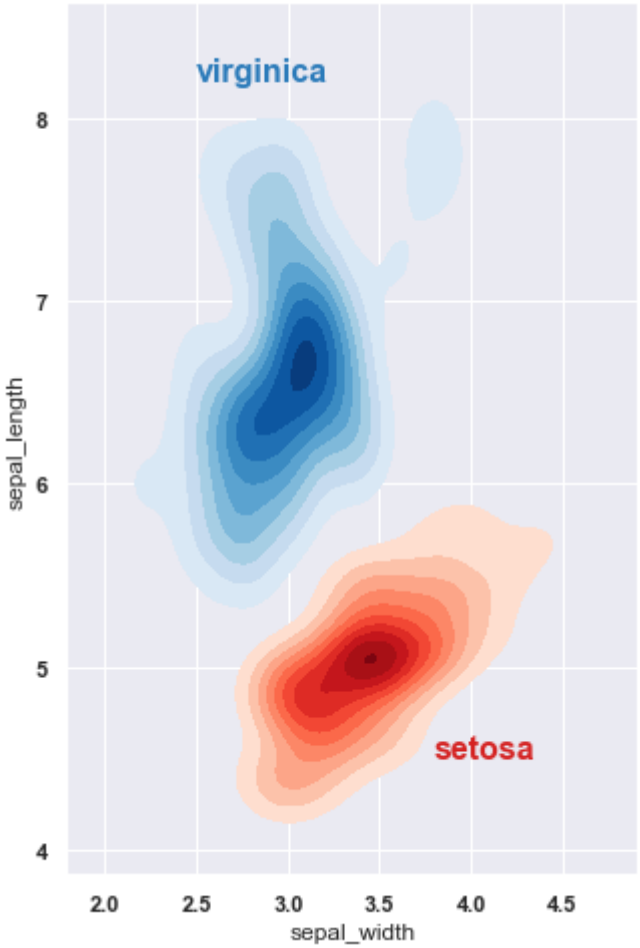

1.00
0.75
0.50
0.25
0.00
−0.25
network
1-1-lh
1-1-rh
5-1-lh
5-1-rh
8-2-lh
8-2-rh
8-3-lh
8-3-rh
8-1-lh
8-1-rh
7-3-lh
7-3-rh
7-1-lh
7-2-rh
7-1-rh
7-2-lh
6-1-lh
6-1-rh
6-2-lh
6-2-rh
12-1-rh
12-2-lh
12-2-rh
12-1-lh
12-3-lh
13-4-rh
13-1-rh
13-2-rh
13-3-rh
13-1-lh
13-2-lh
17-1-rh
17-1-lh
17-4-lh
17-2-lh
17-3-lh
17-2-rh
17-3-rh
network-node-hemi
network-node-hemi

In [384]:

```
# Multiple bivariate KDE plots

import seaborn as sns

import matplotlib.pyplot as plt

sns.set(style="darkgrid")
iris = sns.load_dataset("iris")

# Subset the iris dataset by species
setosa = iris.query("species == 'setosa'")
virginica = iris.query("species == 'virginica'")

# Set up the figure
f, ax = plt.subplots(figsize=(8, 8))
ax.set_aspect("equal")

# Draw the two density plots
ax = sns.kdeplot(setosa.sepal_width, setosa.sepal_length,
                 cmap="Reds", shade=True, shade_lowest=False)
ax = sns.kdeplot(virginica.sepal_width, virginica.sepal_length,
                 cmap="Blues", shade=True, shade_lowest=False)

# Add labels to the plot
red = sns.color_palette("Reds")[-2]
blue = sns.color_palette("Blues")[-2]
ax.text(2.5, 8.2, "virginica", size=16, color=blue)
ax.text(3.8, 4.5, "setosa", size=16, color=red)
```

Out[384]:

```
Text(3.8, 4.5, 'setosa')
```

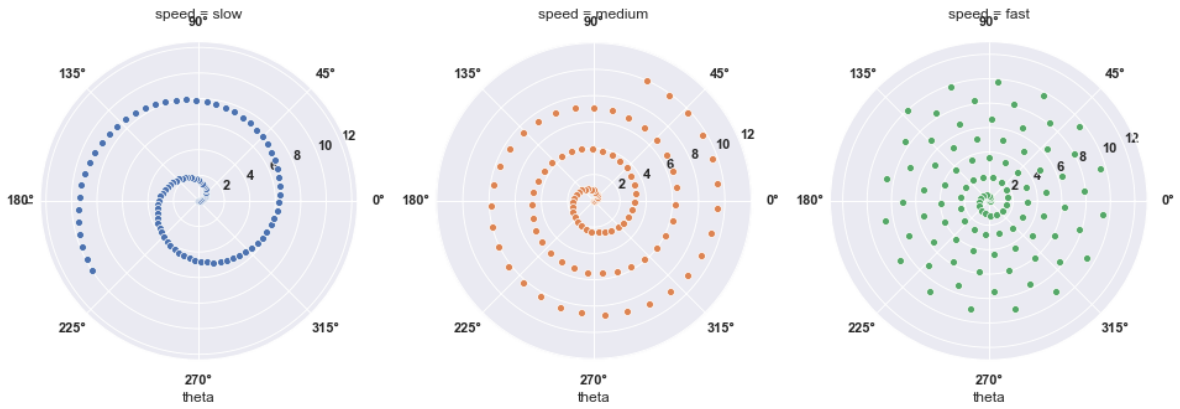

virginica
setosa
sepal_length
sepal_width
8
7
6
5
4
2.0
2.5
3.0
3.5
4.0
4.5

In [385]:

```
# FacetGrid with custom projection

import numpy as np
import pandas as pd
import seaborn as sns

sns.set()

# Generate an example radial datast
r = np.linspace(0, 10, num=100)
df = pd.DataFrame({'r': r, 'slow': r, 'medium': 2 * r, 'fast': 4 * r})

# Convert the dataframe to long-form or "tidy" format
df = pd.melt(df, id_vars=['r'], var_name='speed', value_name='theta')

# Set up a grid of axes with a polar projection
g = sns.FacetGrid(df, col="speed", hue="speed",
                  subplot_kws=dict(projection='polar'), height=4.5,
                  sharex=False, sharey=False, despine=False)

# Draw a scatterplot onto each axes in the grid
g.map(sns.scatterplot, "theta", "r")
```

Out[385]:

```
<seaborn.axisgrid.FacetGrid at 0x2423ec0eec8>
```

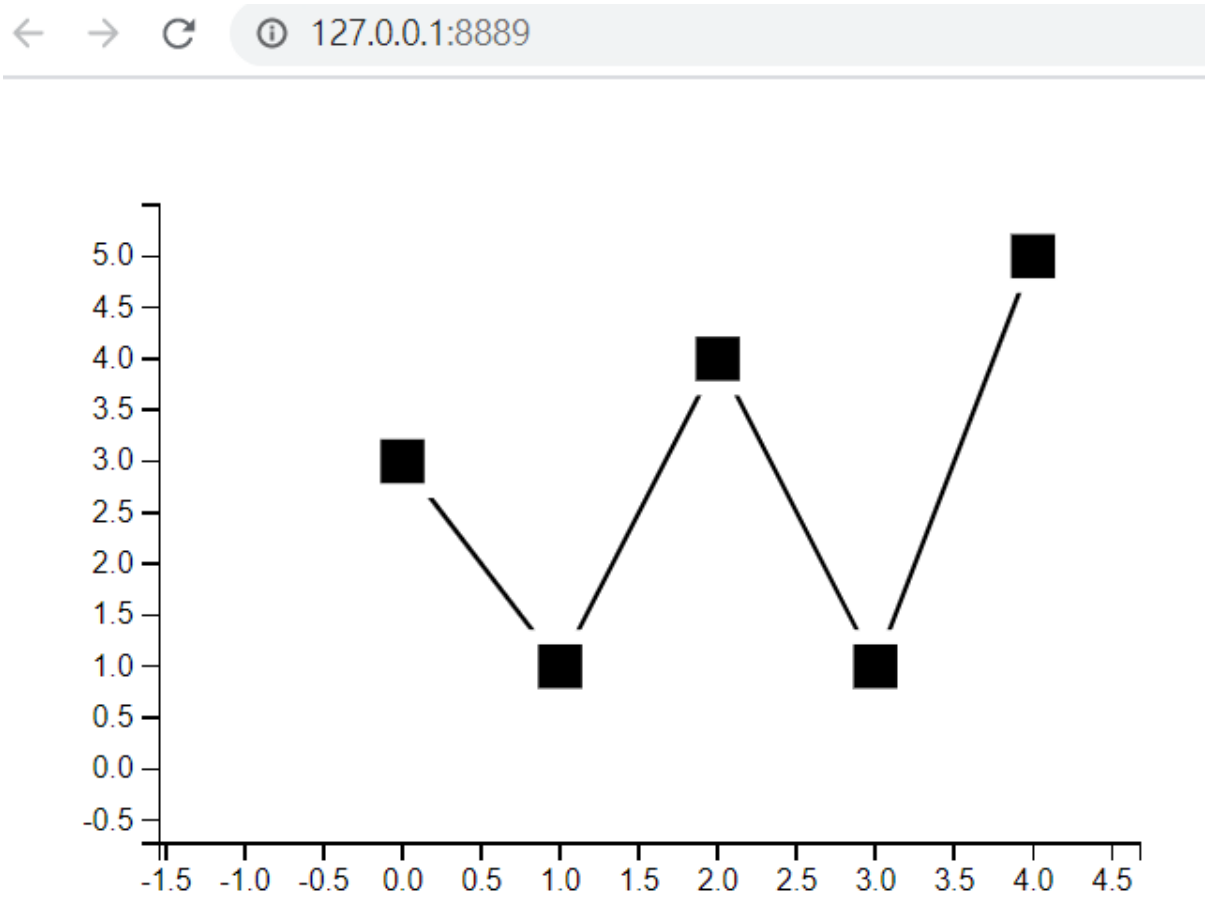


6.5 互動式繪圖

mpld3 模組

In [386]:

```
# pip install mpld3

# import matplotlib.pyplot as plt, mpld3

# plt.plot([3,1,4,1,5], 'ks-', mec='w', mew=5, ms=20)

# mpld3.show()
```

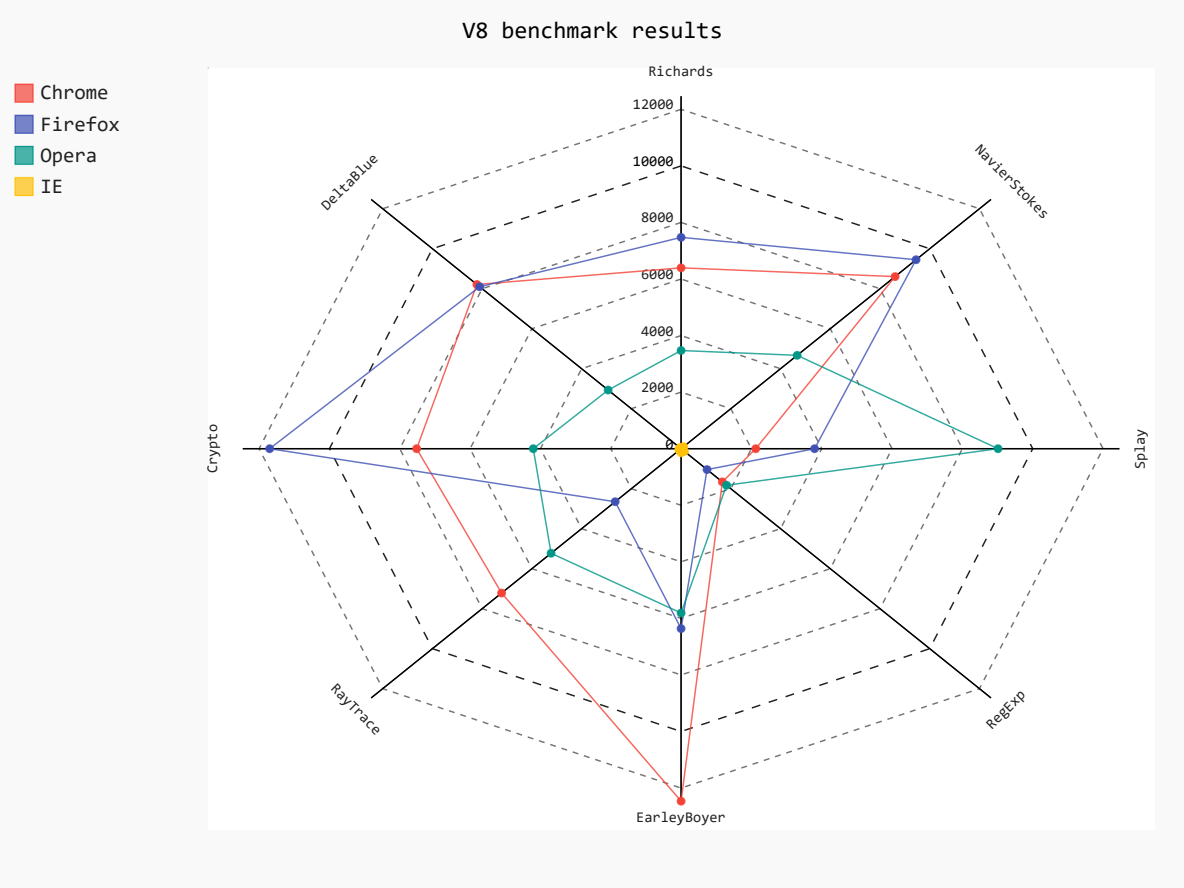

127.0.0.1:8889
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

pygal 模組

In [387]:

```
# pip install pygal
# pip install cairosvg

import pygal # First import pygal
radar_chart = pygal.Radar()
radar_chart.title = 'V8 benchmark results'
radar_chart.x_labels = ['Richards', 'DeltaBlue', 'Crypto', 'RayTrace', 'EarleyBoyer', 'RegE
radar_chart.add('Chrome', [6395, 8212, 7520, 7218, 12464, 1660, 2123, 8607])
radar_chart.add('Firefox', [7473, 8099, 11700, 2651, 6361, 1044, 3797, 9450])
radar_chart.add('Opera', [3472, 2933, 4203, 5229, 5810, 1828, 9013, 4669])
radar_chart.add('IE', [43, 41, 59, 79, 144, 136, 34, 102])
# radar_chart.render()
```

Out[387]:



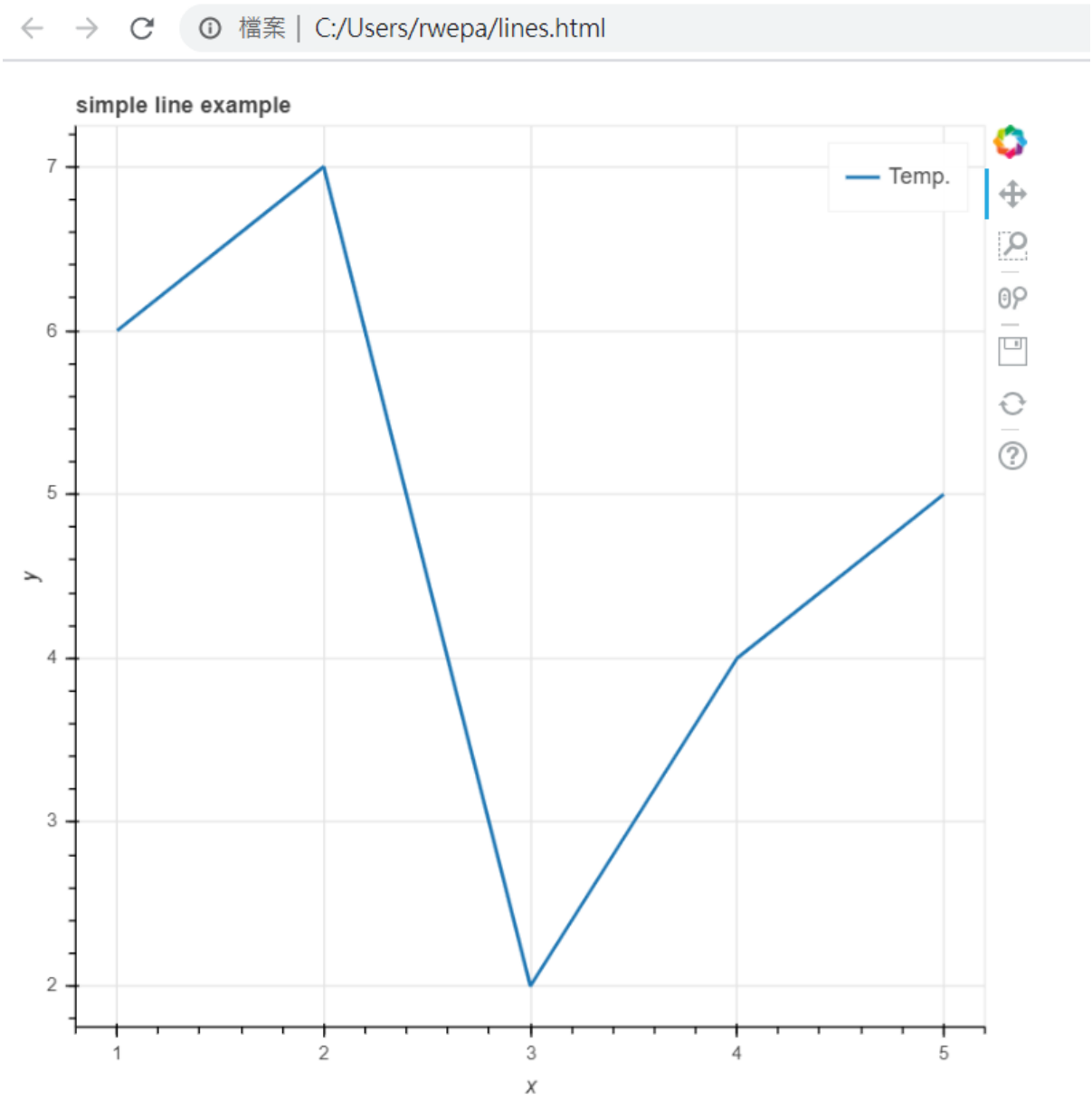
Bokeh 模組

In [388]:

```
# pip install bokeh

import numpy as np
import bokeh.sampledata
# bokeh.sampledata.download()

from bokeh.layouts import gridplot
from bokeh.plotting import figure, show, output_file
from bokeh.plotting import figure, output_file, show

# prepare some data
x = [1, 2, 3, 4, 5]
y = [6, 7, 2, 4, 5]

# output to static HTML file
output_file("lines.html")

# create a new plot with a title and axis labels
p = figure(title="simple line example", x_axis_label='x', y_axis_label='y')

# add a line renderer with legend and line thickness
p.line(x, y, legend="Temp.", line_width=2)

# show the results
# show(p)
```

Out[388]:

GlyphRenderer(id = '1038', …)



Plotly 模組

https://plot.ly/python/ (https://plot.ly/python/)

plotly | Graphing Libraries DEMO DASH

Plotly | Open Source Graphing Libraries | Python Edit this page on GitHub

- Quick Start
- Getting Started
- Is Plotly Free?
- Figure Reference
- API Reference
- Dash
- Examples
- Plotly Fundamentals
- Basic
- Statistical
- Scientific
- Financial
- Maps
- 3D
- Subplots
- Jupyter Widgets Interaction
- Transforms
- Custom Controls
- Animations

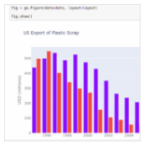
Plotly Python Open Source Graphing Library

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.
Plotly.py is free and open source and you can view the source, report issues or contribute on GitHub.

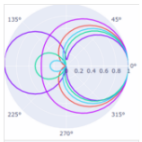
Search

Search Plotly's Python Docs

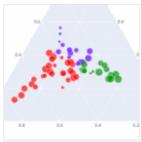
Plotly Fundamentals



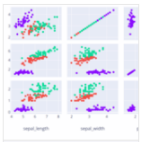
Displaying Figures



Creating and Updating Figures



Version 4 Migration Guide



Plotly Express

More Plotly Fundamentals

第7章 迴歸分析

本章節從迴歸模型簡介為開端，包括以下內容：

- **7.1** 迴歸模型 **Regression Model**
- **7.2** 迴歸分析 **-** 使用 **scikit-learn** 模組

7.1 迴歸模型 Regression Model

考慮 X 與 Y 二個隨機變數，其中的 X 表示「自變數」independent variables， Y 表示「依變數」dependent variables， ε表示誤差項，迴歸方程式表示如下：

$Y = \alpha + \beta X + \varepsilon$

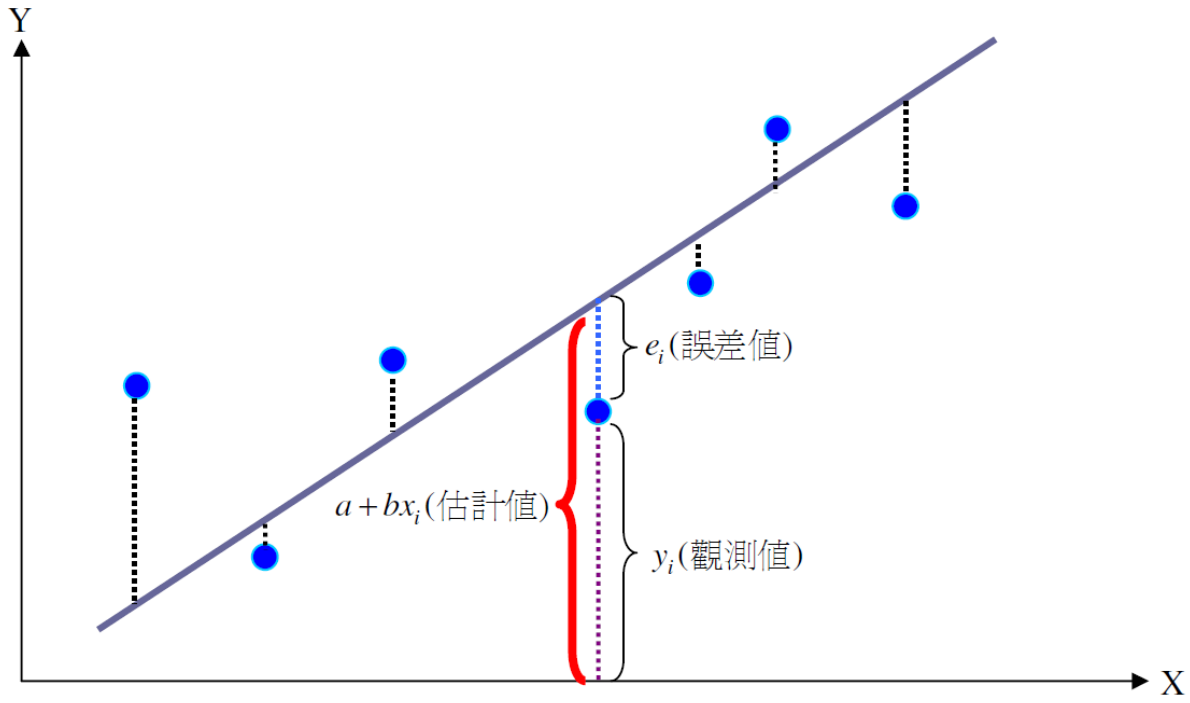
上式必須假設以下基本條件:

(1). Y_i是獨立的常態分佈 $N\left(\alpha + \beta X_i, \sigma^2\right), i = 1, 2, ..., n$

(2). ε_i是獨立的常態分佈 $N\left(0, \sigma^2\right), i = 1, 2, ..., n$

即 Y 的估計值為 $\hat{y} = a + bx$ ，其中^發音為 hat，而估計值的誤差，參圖 1 之說明 e_i = 觀測值(實際值) － 估計值 = $y_i - \hat{y}_i$， 一般估計的目標之一是讓估計值誤差愈小愈好，但要達成 Min e_i for i = 1 to n 不可行，故考慮 $Min\left\{\sum_{i=1}^{n} e_i\right\}$，並盡量達成 $\sum_{i=1}^{n} e_i \to 0$即考慮 $Min\left\{\sum_{i=1}^{n} e_i^2\right\}$，以下說明最小平方法找出 α, β的估計量

參考資料 https://github.com/rwepa/DataDemo/blob/master/regression_01.pdf (https://github.com/rwepa/DataDemo/blob/master/regression_01.pdf)



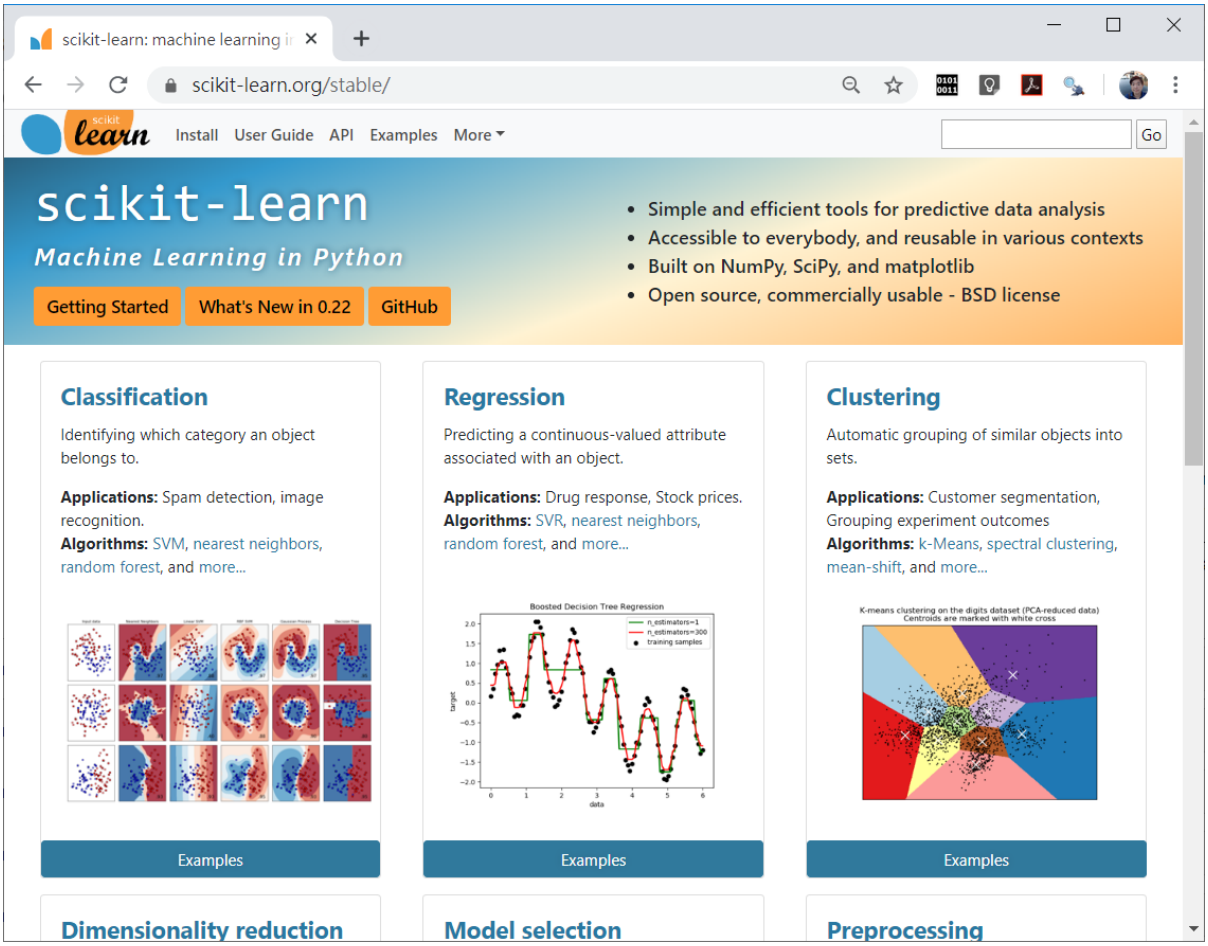
迴歸模式估計誤差說明圖

7.2 迴歸分析 - 使用 scikit-learn 模組

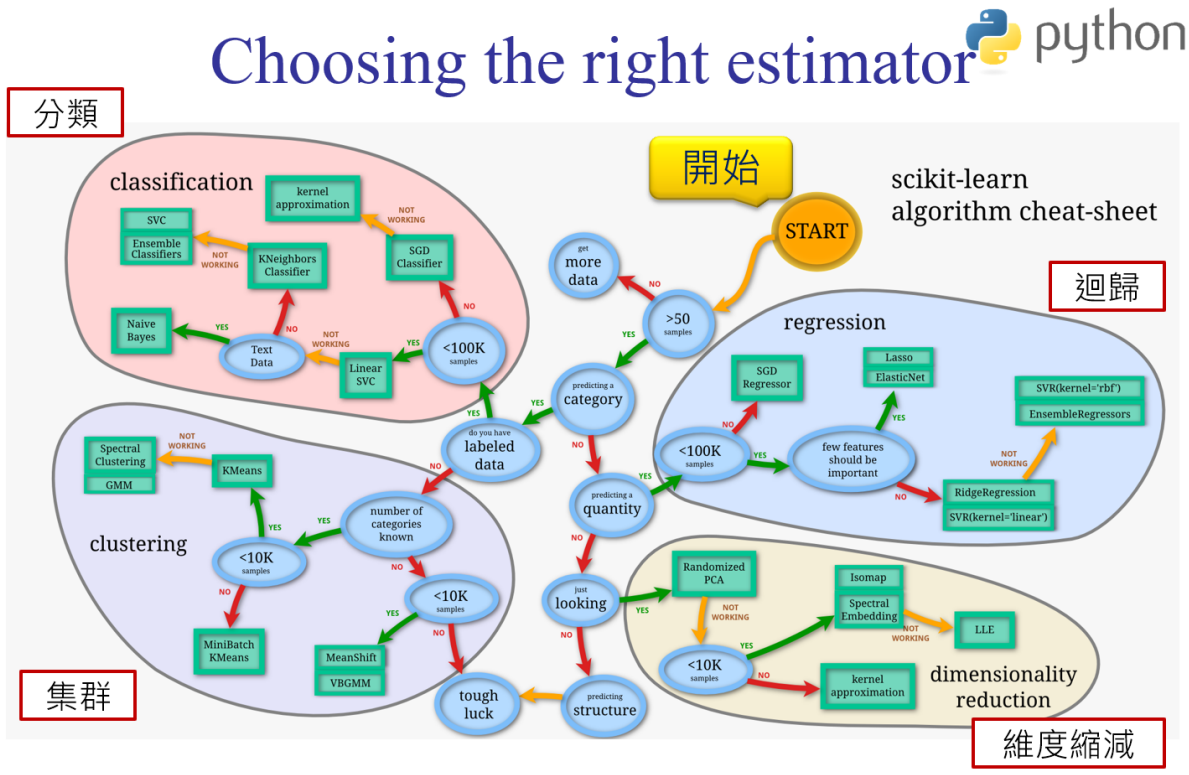
scikit-learn 模組

- Scikit-learn（以前稱為scikits.learn，也稱為sklearn）是針對Python語言的免費機器學習模組。
- 具有迴歸，集群法，支持向量機，隨機森林，梯度提升，k均值和DBSCAN，並且在與Python數值科學NumPy和SciPy 相互結合。
- scikit-learn項目始於scikits.learn，這是David Cournapeau的 Google Summer of Code項目。它的名稱源於它是“ SciKit”（SciPy工具包）的概念，它是SciPy的獨立開發和分佈式第三方擴展。
- 原始代碼庫後來被其他開發人員重寫。2010年費邊Pedregosa，蓋爾Varoquaux，亞歷山大Gramfort和Vincent米歇爾，全部由法國國家信息與自動化研究所的羅屈昂庫爾，法國，把該項目的領導和做出的首次公開發行在二月一日2010。在各種scikits中，scikit-learn和scikit-image在2012年11月被描述為“維護良好且受歡迎”。
- Scikit-learn是GitHub上最受歡迎的機器學習模組之一。
- Anaconda 安裝時已包括此模組。

scikit-learn 網站 https://scikit-learn.org/ (https://scikit-learn.org/)



scikit-learn 常用四大應用



匯入 **scikit-learn** 模組

In [389]:

```
#檢視 scikit-learn 版本
import sklearn

print("The scikit-learn version is", format(sklearn.__version__)) # 0.21.3
```

```
The scikit-learn version is 0.21.3
```

In [390]:

```
#匯入模組
import numpy as np
```

波士頓房價資料

In [391]:

```
# 方法1

from sklearn.datasets import load_boston

boston = load_boston()
```

In [392]:

```
import sklearn
```

In [393]:

```
# 方法2-使用 pandas

import pandas as pd

df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing
```

In [394]:

```
df.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS',
              'NOX', 'RM', 'AGE', 'DIS', 'RAD',
              'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

df # 506*14
```

Out[394]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	I
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	
...	...	...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	

506 rows × 14 columns

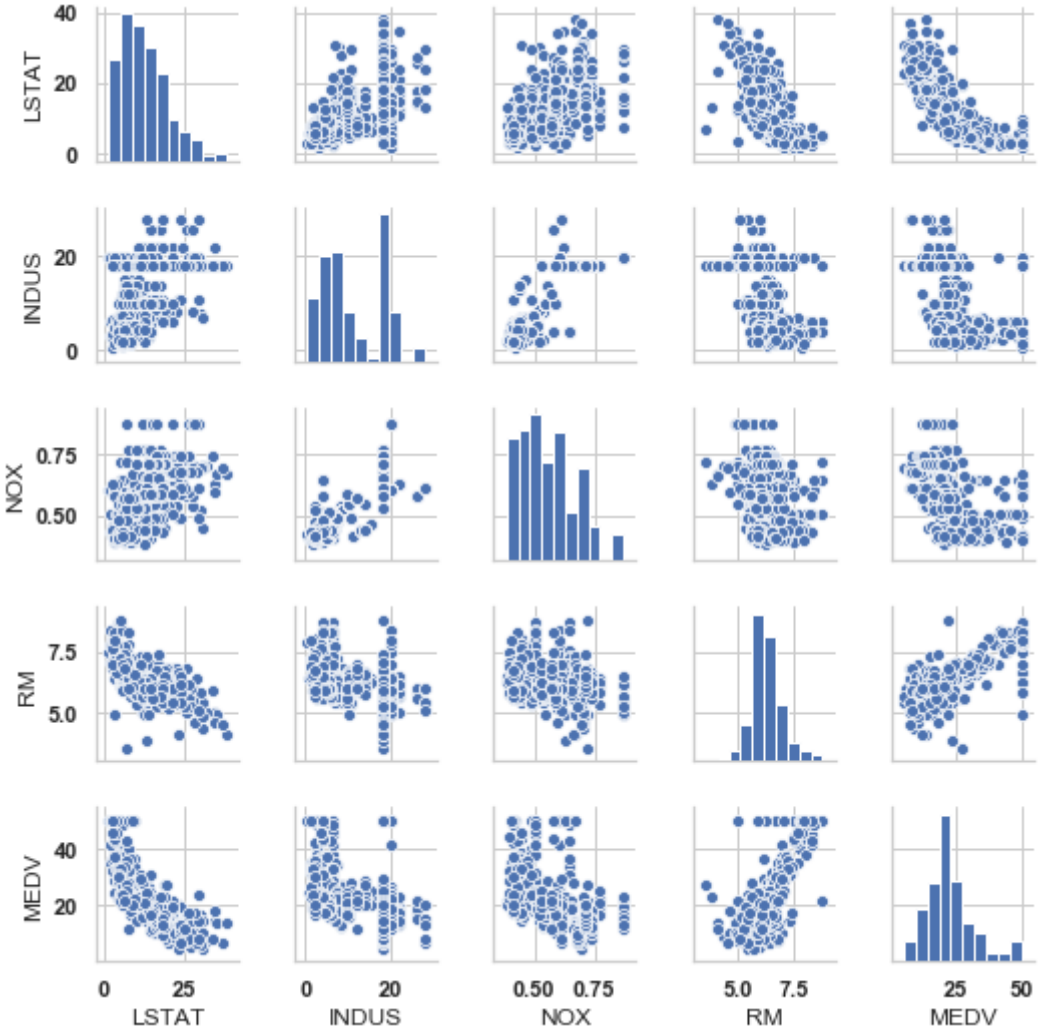
散佈圖矩陣

In [395]:

```
import matplotlib.pyplot as plt

import seaborn as sns

sns.set(style='whitegrid', context='notebook')

cols = ['LSTAT', 'INDUS', 'NOX', 'RM', 'MEDV']

sns.pairplot(df[cols], height=1.5)
# plt.savefig('./figures/scatter.png', dpi=300)
```

Out[395]:

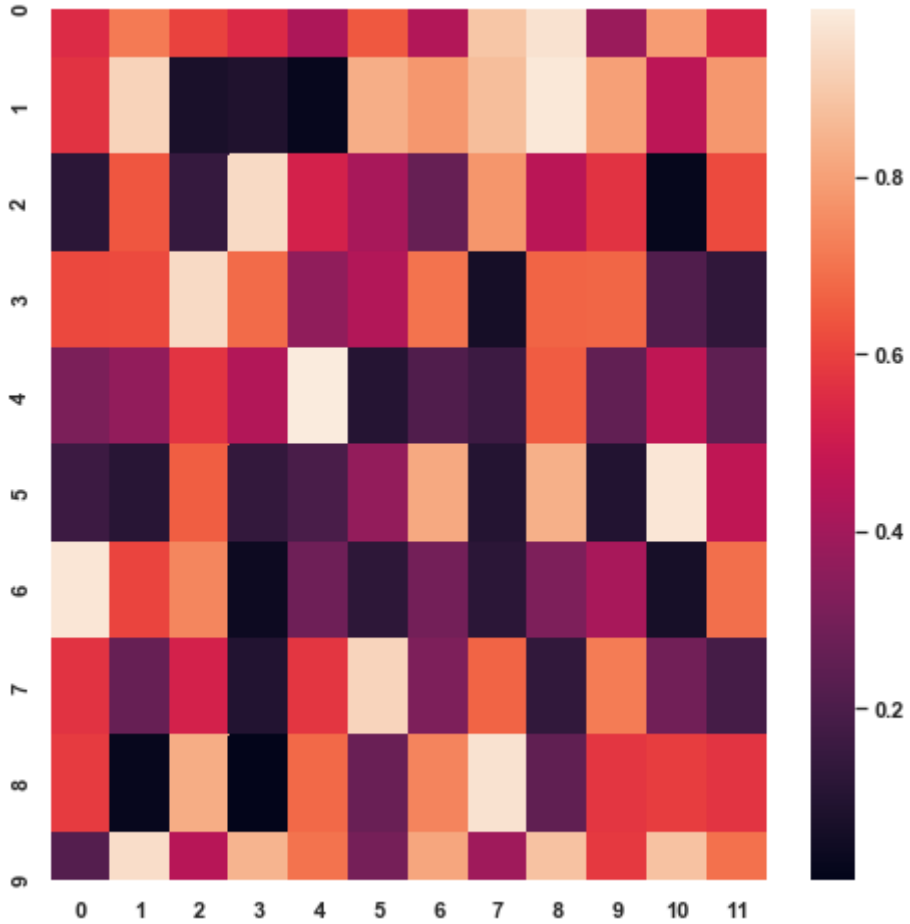
```
<seaborn.axisgrid.PairGrid at 0x24240822808>
```



相關分析

In [396]:

```
import numpy as np; np.random.seed(0)

import seaborn as sns; sns.set()

uniform_data = np.random.rand(10, 12)

ax = sns.heatmap(uniform_data)
```



資料分析

In [397]:

```
print(boston.data.shape) # x: (506, 13)
```

```
(506, 13)
```

In [398]:

```
# ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD'
#  'TAX' 'PTRATIO'  'B' 'LSTAT']
print(boston.feature_names)
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

In [399]:

```
print(np.max(boston.target), np.min(boston.target), np.mean(boston.target))
```

```
50.0 5.0 22.532806324110677
```

In [400]:

```
print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value
(attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,0
00 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds rive
r; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks
by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/ (http
s://archive.ics.uci.edu/ml/machine-learning-databases/housing/)


This dataset was taken from the StatLib library which is maintained at Car
negie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnost
ics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers
 that address regression
problems.

.. topic:: References
```

```
   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influenti
al Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning.
In Proceedings on the Tenth International Conference of Machine Learning,
 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
```

In [401]:

```
print(boston.data[0]) # 顯示第1筆記錄
```

```
[6.320e-03 1.800e+01 2.310e+00 0.000e+00 5.380e-01 6.575e+00 6.520e+01
 4.090e+00 1.000e+00 2.960e+02 1.530e+01 3.969e+02 4.980e+00]
```

In [402]:

```
print(np.max(boston.data), np.min(boston.data), np.mean(boston.data))
```

```
711.0 0.0 70.07396704469443
```

資料預處理

In [403]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target, test_size=0

#Normalize data 標準化資料
from sklearn.preprocessing import StandardScaler

scalerX = StandardScaler().fit(X_train) # fit: compute mean and sd

#scalery = StandardScaler().fit(y_train) # Warning
scalery = StandardScaler().fit(y_train.reshape(-1,1))

X_train = scalerX.transform(X_train) # 執行標準化

y_train = scalery.transform(y_train.reshape(-1,1))

X_test = scalerX.transform(X_test)

y_test = scalery.transform(y_test.reshape(-1,1))

print(np.max(X_train), np.min(X_train), np.mean(X_train), np.max(y_train), np.min(y_train),
```

```
10.190454845432923 -4.6670204084548 2.4732713452985016e-15 2.917749203673125
6 -1.931470986413033 3.5855223803197665e-16
```

In [404]:

```
# five-fold cross-validation and coefficient of determination
# from sklearn.cross_validation import * (舊版用法)
from sklearn.model_selection import KFold

from sklearn.model_selection import cross_val_score

def train_and_evaluate(clf, X_train, y_train):

    clf.fit(X_train, y_train)

    print("Coefficient of determination on training set:", clf.score(X_train, y_train))

    # create a k-fold croos validation iterator of k=5 folds
    cv = KFold(n_splits=5, shuffle=True,  random_state=33)

    scores = cross_val_score(clf, X_train, y_train, cv=cv)

    print("Average coefficient of determination using 5-fold crossvalidation:",np.mean(scor
```

線性模型 **Linear model**

In [405]:

```
from sklearn import linear_model

clf_sgd = linear_model.SGDRegressor(loss='squared_loss', penalty=None,  random_state=33)

train_and_evaluate(clf_sgd, X_train, y_train) # DataConversionWarning

train_and_evaluate(clf_sgd, X_train, y_train.ravel())
```

```
Coefficient of determination on training set: 0.750884717327877
Average coefficient of determination using 5-fold crossvalidation: 0.7135206
064415254
Coefficient of determination on training set: 0.750884717327877
Average coefficient of determination using 5-fold crossvalidation: 0.7135206
064415254

C:\Users\rwepa\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
C:\Users\rwepa\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
C:\Users\rwepa\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
C:\Users\rwepa\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
C:\Users\rwepa\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
C:\Users\rwepa\Anaconda3\lib\site-packages\sklearn\utils\validation.py:724:
DataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples, ), for example using ravel
().
  y = column_or_1d(y, warn=True)
```

In [406]:

```
print(clf_sgd.coef_)
```

```
[-0.09587016  0.09268611 -0.03213476  0.1013135  -0.11995974  0.34538717
 -0.0205987  -0.277162    0.13833327 -0.07844699 -0.19484804  0.05454065
 -0.40023775]
```

In [407]:

```
print(clf_sgd.coef_[0])
```

```
-0.09587016396506948
```

In [408]:

```
# penalty with L2 norm (the squared sums of the coefficients)
clf_sgd1 = linear_model.SGDRegressor(loss='squared_loss', penalty='l2',  random_state=33)
train_and_evaluate(clf_sgd1, X_train, y_train.ravel())
```

```
Coefficient of determination on training set: 0.7508810100754882
Average coefficient of determination using 5-fold crossvalidation: 0.7135250
155549039
```

第8章 決策樹

本章節從決策樹介紹為開端，包括以下內容：

- **8.1** 決策樹
- **8.2** 鐵達尼號資料集-決策樹應用

8.1 決策樹

決策樹是一種類似流程圖的樹狀結構，包含：

- 最上層：根節點(Root node)
- 中間層：節點(Node)
- 最底層：葉節點(Leaf node)，顯示分類／預測結果

每一節點表示一個屬性分類的測試條件，如同「IF-THEN」的控制結構，每個分支表示測試結果，並依此決定資料將分類於此節點的哪一棵子樹(Branch)，並繼續作為分類的條件和最後的決策。

預測明天是否會玩高爾夫

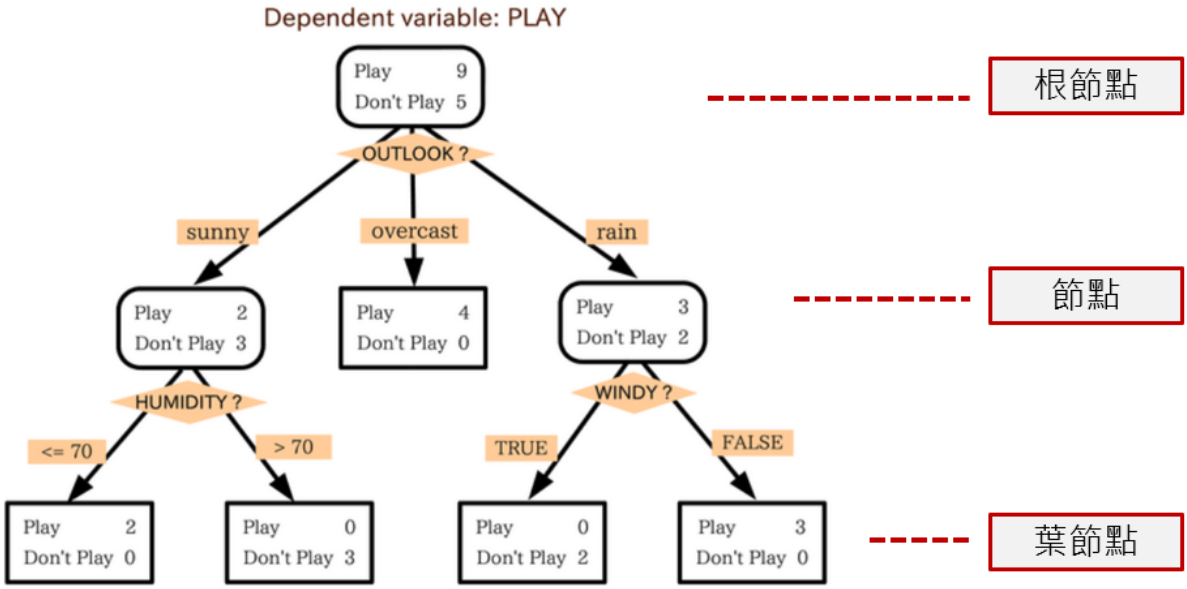
Independent variables				Dep. var
OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	FALSE	Don't Play
sunny	80	90	TRUE	Don't Play
overcast	83	78	FALSE	Play
rain	70	96	FALSE	Play
rain	68	80	FALSE	Play
rain	65	70	TRUE	Don't Play
overcast	64	65	TRUE	Play
sunny	72	95	FALSE	Don't Play
sunny	69	70	FALSE	Play
rain	75	80	FALSE	Play
sunny	75	70	TRUE	Play
overcast	72	90	TRUE	Play
overcast	81	75	FALSE	Play
rain	71	80	TRUE	Don't Play

- X: 4個
- Y: 1個
- 14列, 5行
- Y: Play 9
  Y: Don't 5

決策樹資料集

參考資料 https://zh.wikipedia.org/wiki/決策树

決策樹結構圖



決策樹

決策樹建立流程

1. 資料準備
2. 建立決策樹
3. 選取決策樹演算法

- 步驟1: 將原始資料分成兩組：訓練集、測試集。
- 步驟2: 將訓練集放入決策樹的樹根。
- 步驟3: 使用訓練集來建立決策樹，而在每一個內部節點，則依據資訊理論(Information Theory)來評估選擇哪個屬性繼續做分支的依據。
- 步驟4: 進行決策樹修剪(事前/事後)，以提升預測能力與速度。
- 將以上(1)-(4)步驟不斷遞迴進行，直到所有的新內部節點都是樹葉節點為止。

4. 決策樹修剪
5. 萃取分類規則

決策樹停止條件

1. 該群資料中，每一筆資料都已經歸類在同一類別下。
2. 該群資料中，已經沒有辦法再找到新的屬性來進行節點分割。
3. 該群資料中，已經沒有任何尚未處理的資料。
4. 滿足使用者定義的停上條件

常用的屬性選擇指標

1. 資訊獲利 (Information Gain) [熵愈小, 獲利愈大]

- ID3
- C4.5
- C5.0

2. 吉尼係數 (Gini Index) – CART [吉尼係數愈小者]
3. χ^2 獨立性檢定 – CHAID [卡方統計量愈大者]

決策樹演算法

- **ID3 (Iterative Dichotomizer 3, 疊代二元樹第3代, Quinlan, 1979)**

  + 可處理離散型資料。

  + 兼顧高分類正確率以及降低決策樹的複雜度。

  + 必須將連續型資料作離散化的程序。

- **CHAID (Chi-Square Automatic Interaction Detector, Gordon, 1980)**

  + 利用卡方分析(Chi-Square Test)預測二個變數是否需要合併，如能夠產生最大的類別差異的預測變數，將成為節點的分隔變數。

  + 計算節點中類別的 P值 (P-Value)，以P值大小來決定決策樹是否繼續生長，所以不需像C4.5或CART要再做決策樹修剪的動作。

- **CART (Classification and Regression Trees, Breiman, 1984)**

  + 是以每個節點的動態臨界值作為條件判斷式。

  + CART藉由單一輸入的變數函數，在每個節點分隔資料，並建立一個二元決策樹。

  + CART是使用 Gini Ratio來衡量指標，如果分散的指標程度很高，表示資料中分佈許多類別，相反的，如果指標程度越低，則代表單一類別的成員居多。

- **C4.5 (Quinlan, 1993)**

+ 改良自ID3演算法。

+ 先建構一顆完整的決策樹，再針對每一個內部節點，依使用者定義的預估錯誤率(Predicted Error Rate)來作決策樹修剪的動作。

+ 不同的節點，特徵值離散化結果是不相同的。

決策樹演算法之比較

決策樹	作者	資料屬性	屬性選取	修剪規則
ID3	Quinlan（1979）	離散型資料	Entropy, Gain Ratio	Predicted Error Rate
CHAID	Kass（1980）	離散型資料	Chi-Square Test	No Pruning
CART	Briemen（1984）	離散與連續型資料	Gini Index	Entire Error Rate
C4.5	Quinlan（1993）	離散型資料	Gain Ratio	Predicted Error Rate

資訊理論 (Information theory)

- 各種結果發生機率愈平均，提供資訊量也愈大。
- 資訊量可以當作亂度 (Entropy) 的指標，資訊量愈大，表示亂度愈大。
- Entropy = 1 表示該分類的雜亂度最高。
- 資訊理論可以解決屬性選擇的問題。

ID3 演算法(C4.5, C5.0-商業版)

- 昆蘭 (Quinlan)(1979)提出，以雪南 (Shannon) (1949)的資訊理論(Information theory)為依據。
- 資訊理論:若一事件有k種結果,對應的機率為 Pi。則此事件發生後所得到的資訊量 I (表示Entropy)。

$$I = \frac{x_1}{N} \times \left(-\log_2 \frac{x_1}{N}\right) + \frac{x_2}{N} \times \left(-\log_2 \frac{x_2}{N}\right) + \cdots + \frac{x_k}{N} \times \left(-\log_2 \frac{x_k}{N}\right)$$
$$= -\sum_{i=1}^{k} p_i \times \log_2(p_i), p_i = \frac{x_i}{N}$$

- 資訊增益 (Information Gain), 考慮屬性A作為分枝節點時, 對資訊的貢獻度。
- 一般資訊量最小的屬性為優先選取，也就是選擇資訊獲利最大的屬性。
- Gain(A) =原始資訊量 - 屬性A分枝節點資訊量 =I(D)- I(A)。

Gain 範例

id	年資	教育程度	相關經驗	表現
1	5年以下	研究所	是	優等
2	10年以上	研究所	否	普通
3	5年以下	研究所	是	優等
4	5年以下	大專	是	普通
5	5年以下	研究所	否	優等
6	10年以上	研究所	是	優等
7	5-10年	大專	否	普通
8	5-10年	研究所	是	優等
9	5-10年	大專	否	普通
10	5年以下	研究所	是	普通

- $I(D) = -\frac{5}{10}\log_2\frac{5}{10} - \frac{5}{10}\log_2\frac{5}{10}$
  $= -0.5 \times (-1) - 0.5 \times (-1) = 1$
- $I(\text{年資}.5\text{年以下}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$
- $I(\text{年資}.5-10\text{年}) = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918$
- $I(\text{年資}.10\text{年以上}) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$
- $I_{\text{年資}}(D) = \frac{5}{10} \times 0.971 + \frac{3}{10} \times 0.918 + \frac{2}{10} \times 1 = 0.961$

- $Gain_{\text{年資}} = I(D) - I(\text{年資}) = 1 - 0.961 = 0.039$
- $\boldsymbol{Gain_{\text{教育程度}} = 0.396}$
- $Gain_{\text{相關經驗}} = 0.125$

因為 $Gain_{\text{教育程度}} = 0.396$ 最大，第1層分枝選取教育程度

參考資料 資料挖礦與大數據分析, 簡禎富,許嘉裕, 前程文化

Gini Index 範例

- Gini係數是衡量資料對所有類別的不純度(Impurity, Breiman et al., 1984), Gini愈小愈適合.

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

$$Gini_{\text{年資}}(D) = \frac{5}{10} \times \left(1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right) + \frac{3}{10} \times \left(1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2\right) + \frac{2}{10} \times \left(1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2\right) = 0.473$$

- $Gini_{\text{教育程度}}(D) = 0.286$
- $Gini_{\text{相關經驗}}(D) = 0.417$

避免過度合適(over fit)的方法

1. 事前修剪(Pre-Pruning)

- 運用統計門檻值加以衡量，譬如卡方值或資訊獲得值等技術，評估是否該繼續分割某內部節點成數個子分支或是應該立刻停止。

2. 事後修剪(Post-Pruning)

- 允許決策樹超適情形的合理存在，當完成決策樹的建立之後，再來進行修剪的程序。

8.2 鐵達尼號資料集-決策樹應用

載入資料

In [409]:

```
# 鐵達尼號資料集 1912年
# https://github.com/rwepa/DataDemo/blob/master/titanic.csv
import csv

import numpy as np

# with open('data/titanic.csv', 'rb') as csvfile:
with open('data/titanic.csv') as csvfile:
    titanic_reader = csv.reader(csvfile, delimiter=',', quotechar='"')

    # Header contains feature names
    # row = titanic_reader.next()
    # feature_names = np.array(row)
    feature_names = next(titanic_reader)

    # Load dataset, and target classes
    titanic_X, titanic_y = [], []

    for row in titanic_reader:

        titanic_X.append(row)

        titanic_y.append(row[2]) # The target value is "survived"

    titanic_X = np.array(titanic_X)

    titanic_y = np.array(titanic_y)
```

讀取欄位名稱

In [410]:

```
print(feature_names)
```

```
['row.names', 'pclass', 'survived', 'name', 'age', 'embarked', 'home.dest',
'room', 'ticket', 'boat', 'sex']
```

讀取第1筆資料

In [411]:

```
print(titanic_X[0], titanic_y[0])
```

```
['1' '1st' '1' 'Allen, Miss Elisabeth Walton' '29.0000' 'Southampton'
 'St Louis, MO' 'B-5' '24160 L221' '2' 'female'] 1
```

資料預處理, 保留 **class, age and sex**

In [412]:

```
titanic_X = titanic_X[:, [1, 4, 10]]

feature_names = np.array(feature_names)[[1, 4, 10]]
```

In [413]:

```
print(feature_names)
```

```
['pclass' 'age' 'sex']
```

In [414]:

```
print(titanic_X[12], titanic_y[12])
```

```
['1st' 'NA' 'female'] 1
```

將 **ages** 為**NA**值以平均值填滿

In [415]:

```
ages = titanic_X[:, 1]

mean_age = np.mean(titanic_X[ages != 'NA', 1].astype(np.float))

mean_age
```

Out[415]:

```
31.19418104265403
```

In [416]:

```
titanic_X[titanic_X[:, 1] == 'NA', 1] = mean_age
```

將類別型資料編碼為數值型資料

In [417]:

```
# sex 編碼
from sklearn.preprocessing import LabelEncoder

enc = LabelEncoder()

label_encoder = enc.fit(titanic_X[:, 2])

# ['0' '1']
print("Categorical classes:", label_encoder.classes_)

integer_classes = label_encoder.transform(label_encoder.classes_)

print("Integer classes:", integer_classes)
t = label_encoder.transform(titanic_X[:, 2])

titanic_X[:, 2] = t

print(feature_names)

print(titanic_X[12], titanic_y[12])
```

```
Categorical classes: ['female' 'male']
Integer classes: [0 1]
['pclass' 'age' 'sex']
['1st' '31.19418104265403' '0'] 1
```

In [418]:

```
# pclass 編碼
from sklearn.preprocessing import OneHotEncoder

enc = LabelEncoder()

label_encoder = enc.fit(titanic_X[:, 0])

print("Categorical classes:", label_encoder.classes_)
```

```
Categorical classes: ['1st' '2nd' '3rd']
```

In [419]:

```
integer_classes = label_encoder.transform(label_encoder.classes_).reshape(3, 1)

print("Integer classes:", integer_classes)
```

```
Integer classes: [[0]
 [1]
 [2]]
```

In [420]:

```
enc = OneHotEncoder(categories='auto')

one_hot_encoder = enc.fit(integer_classes)

one_hot_encoder
```

Out[420]:

```
OneHotEncoder(categorical_features=None, categories='auto', drop=None,
              dtype=<class 'numpy.float64'>, handle_unknown='error',
              n_values=None, sparse=True)
```

In [421]:

```
# First, convert clases to 0-(N-1) integers using label_encoder
num_of_rows = titanic_X.shape[0]
t = label_encoder.transform(titanic_X[:, 0]).reshape(num_of_rows, 1)

# Second, create a sparse matrix with three columns, each one indicating
# if the instance belongs to the class
new_features = one_hot_encoder.transform(t)

# Add the new features to titanix_X
titanic_X = np.concatenate([titanic_X, new_features.toarray()], axis = 1)

#Eliminate converted columns
titanic_X = np.delete(titanic_X, [0], 1)

# Update feature names
feature_names = ['age', 'sex', 'first_class', 'second_class', 'third_class']

# Convert to numerical values
titanic_X = titanic_X.astype(float)
titanic_y = titanic_y.astype(float)

print(feature_names)
print(titanic_X[0], titanic_y[0])
```

```
['age', 'sex', 'first_class', 'second_class', 'third_class']
[29.  0.  1.  0.  0.] 1.0
```

訓練資料, 測試資料

In [422]:

```
# from sklearn.cross_validation import train_test_split (舊版本使用)
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    titanic_X, titanic_y, test_size=0.25, random_state=33)
```

訓練決策樹

In [423]:

```
from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion='entropy',
                                  max_depth=3, min_samples_leaf=5)
clf = clf.fit(X_train,y_train)

clf
```

Out[423]:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=3,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=5, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

計算訓練資料正確分類比例

In [424]:

```
from sklearn import metrics
def measure_performance(X,y,clf, show_accuracy=True,show_classification_report=True,show_cc
    y_pred=clf.predict(X)
    if show_accuracy:
        print("Accuracy:{0:.3f}".format(metrics.accuracy_score(y,y_pred)),"\n")

    if show_classification_report:
        print("Classification report")
        print(metrics.classification_report(y,y_pred),"\n")

    if show_confusion_matrix:
        print("Confusion matrix")
        print(metrics.confusion_matrix(y,y_pred),"\n")

measure_performance(X_train,y_train,clf, show_classification_report=False, show_confusion_m
```

```
Accuracy:0.838
```

使用 **Windows 64-bit** 繪製決策樹

- 步驟1 下載並安裝 Graphviz http://www.graphviz.org/ (http://www.graphviz.org/)
- 步驟2 環境變數 加入 PATH

  + 電腦 / 右鍵 / 內容 / 進階系統設定 /

  + 環境變數 / 系統變數 --> Path --> 編輯 --> 最後加上以下內容

  + Win7,8 --> ;C:\Program Files (x86)\Graphviz2.38\bin

  + Win10 --> 新增 --> C:\Program Files (x86)\Graphviz2.38\bin

- 步驟3 下載並安裝pydotplus

  + Anaconda 環境執行 conda install pydotplus

- 步驟4 重新啟動 Spyder

conda install pydotplus 安裝畫面

Microsoft Windows [版本 10.0.18363.535] (c) 2019 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\rwepa> conda install pydotplus Collecting package metadata (current_repodata.json): done Solving environment: done

"## Package Plan ##"

environment location: C:\Users\rwepa\Anaconda3

added / updated specs:

    - pydotplus

The following packages will be downloaded:

```
    package                    |            build
    ---------------------------|-----------------
    conda-4.8.0                |           py37_1         2.8 MB
    pydotplus-2.0.2            |           py37_1          42 KB
    ------------------------------------------------------------
                                           Total:         2.9 MB
```

The following NEW packages will be INSTALLED:

pydotplus pkgs/main/win-64::pydotplus-2.0.2-py37_1

The following packages will be UPDATED:

conda 4.7.12-py37_0 --> 4.8.0-py37_1

Proceed ([y]/n)? y

Downloading and Extracting Packages conda-4.8.0 | 2.8 MB |
| 100%
pydotplus-2.0.2 | 42 KB |
| 100%
Preparing transaction: done Verifying transaction: done Executing transaction: done

C:\Users\rwepa>

決策樹

In [425]:

```
import io

import pydotplus
```

In [426]:

```
dot_data = io.StringIO()

tree.export_graphviz(clf, out_file=dot_data, feature_names=['age','sex',
    '1st_class','2nd_class','3rd_class'])

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
```

匯出決策樹

In [427]:

```
graph.write_png('titanic-RWEPA2020.png')
```

Out[427]:

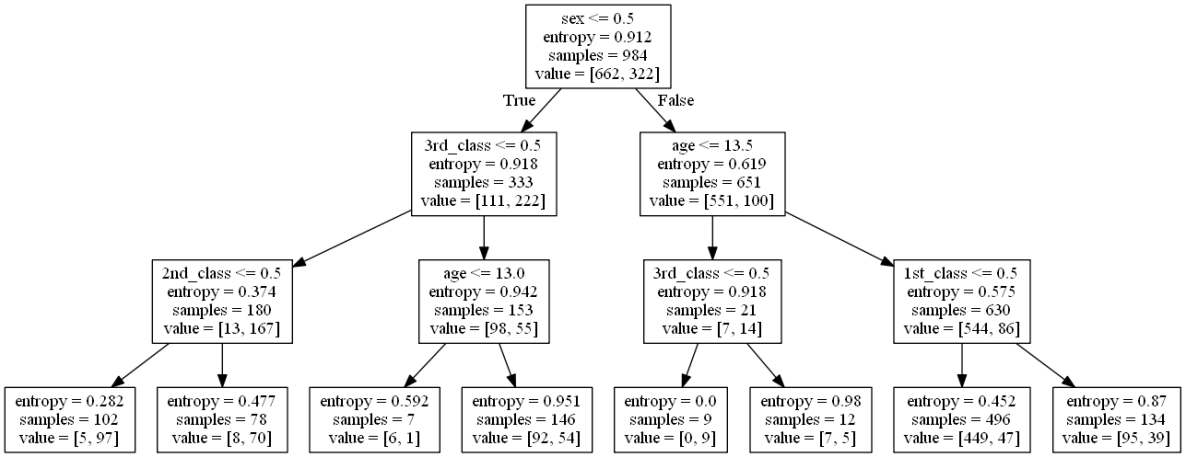
```
True
```

繪製決策樹

In [428]:

```
from IPython.core.display import Image
Image(filename='titanic-RWEPA2020.png')
```

Out[428]:



第9章 關聯規則應用

本章節從購物籃分析介紹為開端，包括以下內容：

- **9.1** 購物籃分析（**market-basket analysis**）
- **9.2 mlxtend** 模組

9.1 購物籃分析（market-basket analysis）

購物籃分析是在顧客的同一次購物活動中，對其所購買商品的組成，進行相關性研究的方法，是關聯規則探勘的一種應用。

- 分析對象為顧客購買的交易資料，其特點為:

  1. 資料量很大（某些行業經常有每天超過100 萬的交易）

  2. 資料矩陣呈現稀疏性，亦即每筆交易(橫列)只包含店內所有商品(縱行)的一小部分

  3. 購買者間存在著異質性，也就是說具有不同品味的人，往往傾向於購買一些特殊商品

  4. 所以對分析者而言，此種交易資料的分析是件富有挑戰性的任務

基本運作過程

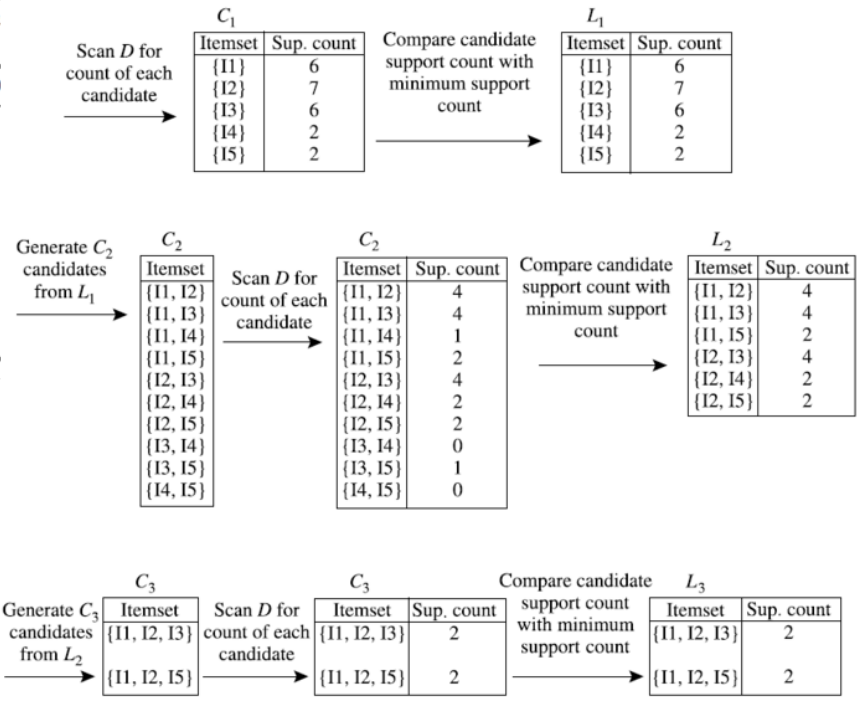
1. 選擇合適的商品項目及其組合
   - 必須在數以百計或千計的商品項目中選出商業上真正有用的品項
   - 可以對產品先進行各層次的分類(大分類、中分類、小分類)再進行組合
2. 尋找超過支持度門檻的商品
   - 即找出頻繁項目集(frequent itemsets)
   - 需要計算支持度(support)
   - 定義閥值（threshold）
3. 挖掘關聯規則
   - 透過相對次數的計算取得規則
   - 規則形如(如果A，則B)，其中A與B為產品項目形成的集合，稱為項目集 (itemsets)

Apriori 演算法

Apriori Algorithm

Transactional Data for an *AllElectronic* Branch

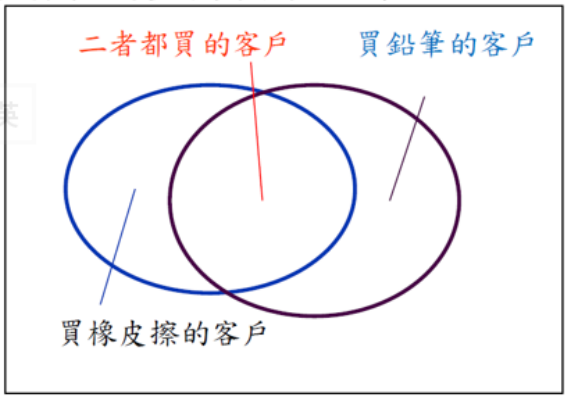
TID	List of item_ID
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



Support計算範例1

計算範例一

- 100筆交易資料中，若想要知道購買鉛筆的人會不會也購買橡皮擦，其計算方式如下:
  - 購買鉛筆的交易共有80筆
  - 買鉛筆同時也會買橡皮擦的交易有62筆



- 支持度(Support)
  - 62/100 =62%
- 信賴度(Confidence)
  - 62/80=77.5%

Support計算範例2

計算範例二

- 某商場記錄了下列顧客購買交易資料，請計算規則 $\{Milk, Diaper\} \Rightarrow \{Beer\}$ 的支持度s和信賴度c (其中|T|表交易資料總筆數)

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

$$\{Milk, Diaper\} \Rightarrow \{Beer\}$$

$$s = \frac{\sigma(Milk, Diaper, Beer)}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(Milk, Diaper, Beer)}{\sigma(Milk, Diaper)} = \frac{2}{3} = 0.67$$

9.2 mlxtend 模組

conda config --add channels conda-forge

conda install mlxtend

mlxtend 模組說明

apriori {mlxtend}

apriori

Definition : apriori(df, min_support=0.5, use_colnames=False, max_len=None)
Type : Function in mlxtend.frequent_patterns.apriori module

Get frequent itemsets from a one-hot DataFrame Parameters ---------- df : pandas DataFrame

pandas DataFrame the encoded format. For example,

```
      Apple Bananas Beer Chicken Milk Rice
0101101110100121010003110000400111150010116001010711000 0
```

min_support : float (default: 0.5)
A float between 0 and 1 for minumum support of the itemsets returned. The support is computed as the fraction transactions_where_item(s)_occur / total_transactions.
use_colnames : bool (default: False)
If true, uses the DataFrames' column names in the returned DataFrame instead of column indices.
max_len : int (default: None)
Maximum length of the itemsets generated. If *None* (default) all possible itemsets lengths (under the apriori condition) are evaluated.

匯入資料

In [429]:

```
import pandas as pd

from mlxtend.frequent_patterns import apriori

from mlxtend.frequent_patterns import association_rules
```

In [430]:

```
# https://github.com/rwepa/DataDemo/blob/master/OnlineRetail.xlsx
# download to C:\pythondata\data\Online Retail.xlsx (22.6MB)
```

In [431]:

```
df = pd.read_excel('data/OnlineRetail.xlsx') # 541909*8

df.head(20)
```

Out[431]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Ur King
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Ur King
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Ur King
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Ur King
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Ur King
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00	7.65	17850.0	Ur King
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00	4.25	17850.0	Ur King
7	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00	1.85	17850.0	Ur King
8	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00	1.85	17850.0	Ur King
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 08:34:00	1.69	13047.0	Ur King
10	536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	2010-12-01 08:34:00	2.10	13047.0	Ur King
11	536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	2010-12-01 08:34:00	2.10	13047.0	Ur King

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
12	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	2010-12-01 08:34:00	3.75	13047.0	Ur King
13	536367	22310	IVORY KNITTED MUG COSY	6	2010-12-01 08:34:00	1.65	13047.0	Ur King
14	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	2010-12-01 08:34:00	4.25	13047.0	Ur King
15	536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	2010-12-01 08:34:00	4.95	13047.0	Ur King
16	536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	2010-12-01 08:34:00	9.95	13047.0	Ur King
17	536367	21754	HOME BUILDING BLOCK WORD	3	2010-12-01 08:34:00	5.95	13047.0	Ur King
18	536367	21755	LOVE BUILDING BLOCK WORD	3	2010-12-01 08:34:00	5.95	13047.0	Ur King
19	536367	21777	RECIPE BOX WITH METAL HEART	4	2010-12-01 08:34:00	7.95	13047.0	Ur King

In [432]:

```
df.describe()
```

Out[432]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

In [433]:

```
print(df.dtypes)
```

```
InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID            float64
Country                object
dtype: object
```

資料清理

In [434]:

```
df['Description'] = df['Description'].str.strip() # 預設移除頭尾空白字元,eg.line 15

df.dropna(axis=0, subset=['InvoiceNo'], inplace=True) # 移除InvoiceNo有NA

df['InvoiceNo'] = df['InvoiceNo'].astype('str') # 資料型態轉換

df = df[~df['InvoiceNo'].str.contains('C')] # 移除InvoiceNo有 C (credit transactions), 5326
```

交易資料合併

In [435]:

```
basket = (df[df['Country'] =="France"] # 392*1563
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))

basket
```

Out[435]:

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	PENC SM TU SKI
InvoiceNo							
536370	0.0	0.0	0.0	0.0	0.0	0.0	
536852	0.0	0.0	0.0	0.0	0.0	0.0	
536974	0.0	0.0	0.0	0.0	0.0	0.0	
537065	0.0	0.0	0.0	0.0	0.0	0.0	
537463	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	
580986	0.0	0.0	0.0	0.0	0.0	0.0	
581001	0.0	0.0	0.0	0.0	0.0	0.0	
581171	0.0	0.0	0.0	0.0	0.0	0.0	
581279	0.0	0.0	0.0	0.0	0.0	0.0	
581587	0.0	0.0	0.0	0.0	0.0	0.0	

392 rows × 1563 columns

one hot 編碼

In [436]:

```
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1
```

In [437]:

```
basket_sets = basket.applymap(encode_units) # 編碼為0,1
basket_sets.drop('POSTAGE', inplace=True, axis=1) # 移除 POSTAGE 392*1562
basket_sets
```

Out[437]:

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	PENC SM TU SKI
InvoiceNo							
536370	0	0	0	0	0	0	
536852	0	0	0	0	0	0	
536974	0	0	0	0	0	0	
537065	0	0	0	0	0	0	
537463	0	0	0	0	0	0	
...	...	...	...	...	...	...	
580986	0	0	0	0	0	0	
581001	0	0	0	0	0	0	
581171	0	0	0	0	0	0	
581279	0	0	0	0	0	0	
581587	0	0	0	0	0	0	

392 rows × 1562 columns

建立頻繁項目集

In [438]:

```
# generate  frequent item sets
frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)

frequent_itemsets
```

Out[438]:

	support	itemsets
0	0.071429	(4 TRADITIONAL SPINNING TOPS)
1	0.096939	(ALARM CLOCK BAKELIKE GREEN)
2	0.102041	(ALARM CLOCK BAKELIKE PINK)
3	0.094388	(ALARM CLOCK BAKELIKE RED)
4	0.081633	(BAKING SET 9 PIECE RETROSPOT)
5	0.071429	(CHILDRENS CUTLERY DOLLY GIRL)
6	0.099490	(DOLLY GIRL LUNCH BOX)
7	0.096939	(JUMBO BAG RED RETROSPOT)
8	0.076531	(JUMBO BAG WOODLAND ANIMALS)
9	0.125000	(LUNCH BAG APPLE DESIGN)
10	0.084184	(LUNCH BAG DOLLY GIRL DESIGN)
11	0.153061	(LUNCH BAG RED RETROSPOT)
12	0.119898	(LUNCH BAG SPACEBOY DESIGN)
13	0.117347	(LUNCH BAG WOODLAND)
14	0.142857	(LUNCH BOX WITH CUTLERY RETROSPOT)
15	0.104592	(MINI PAINT SET VINTAGE)
16	0.102041	(PACK OF 72 RETROSPOT CAKE CASES)
17	0.081633	(PAPER BUNTING RETROSPOT)
18	0.168367	(PLASTERS IN TIN CIRCUS PARADE)
19	0.137755	(PLASTERS IN TIN SPACEBOY)
20	0.081633	(PLASTERS IN TIN STRONGMAN)
21	0.170918	(PLASTERS IN TIN WOODLAND ANIMALS)
22	0.188776	(RABBIT NIGHT LIGHT)
23	0.096939	(RED RETROSPOT CHARLOTTE BAG)
24	0.137755	(RED RETROSPOT MINI CASES)
25	0.071429	(RED RETROSPOT PICNIC BAG)
26	0.181122	(RED TOADSTOOL LED NIGHT LIGHT)
27	0.125000	(REGENCY CAKESTAND 3 TIER)
28	0.086735	(RETROSPOT TEA SET CERAMIC 11 PC)
29	0.107143	(ROUND SNACK BOXES SET OF 4 FRUITS)
30	0.158163	(ROUND SNACK BOXES SET OF4 WOODLAND)
31	0.076531	(SET/10 RED POLKADOT PARTY CANDLES)

	support	itemsets
32	0.132653	(SET/20 RED RETROSPOT PAPER NAPKINS)
33	0.137755	(SET/6 RED SPOTTY PAPER CUPS)
34	0.127551	(SET/6 RED SPOTTY PAPER PLATES)
35	0.071429	(SPACEBOY BIRTHDAY CARD)
36	0.125000	(SPACEBOY LUNCH BOX)
37	0.122449	(STRAWBERRY LUNCH BOX WITH CUTLERY)
38	0.094388	(TEA PARTY BIRTHDAY CARD)
39	0.073980	(WOODLAND CHARLOTTE BAG)
40	0.073980	(ALARM CLOCK BAKELIKE PINK, ALARM CLOCK BAKELI...
41	0.079082	(ALARM CLOCK BAKELIKE RED, ALARM CLOCK BAKELIK...
42	0.073980	(ALARM CLOCK BAKELIKE PINK, ALARM CLOCK BAKELI...
43	0.071429	(DOLLY GIRL LUNCH BOX, SPACEBOY LUNCH BOX)
44	0.089286	(PLASTERS IN TIN CIRCUS PARADE, PLASTERS IN TI...
45	0.102041	(PLASTERS IN TIN CIRCUS PARADE, PLASTERS IN TI...
46	0.104592	(PLASTERS IN TIN WOODLAND ANIMALS, PLASTERS IN...
47	0.102041	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...
48	0.102041	(SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...
49	0.122449	(SET/6 RED SPOTTY PAPER CUPS, SET/6 RED SPOTTY...
50	0.099490	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...

建立規則

In [439]:

```
# generate the rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

rules.head()
```

Out[439]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverag
0	(ALARM CLOCK BAKELIKE PINK)	(ALARM CLOCK BAKELIKE GREEN)	0.102041	0.096939	0.073980	0.725000	7.478947	0.06408
1	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE PINK)	0.096939	0.102041	0.073980	0.763158	7.478947	0.06408
2	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.096939	0.079082	0.837838	8.642959	0.06993
3	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.094388	0.079082	0.815789	8.642959	0.06993
4	(ALARM CLOCK BAKELIKE PINK)	(ALARM CLOCK BAKELIKE RED)	0.102041	0.094388	0.073980	0.725000	7.681081	0.06434

顯示規則

In [440]:

```
rules
print(rules.to_string())
```

```
                                        antecedents
consequents  antecedent support  consequent support   support  confidence
lift  leverage  conviction
0                        (ALARM CLOCK BAKELIKE PINK)
(ALARM CLOCK BAKELIKE GREEN)            0.102041           0.096939  0.0739
80    0.725000  7.478947  0.064088    3.283859
1                       (ALARM CLOCK BAKELIKE GREEN)
(ALARM CLOCK BAKELIKE PINK)             0.096939           0.102041  0.07398
0    0.763158  7.478947  0.064088    3.791383
2                         (ALARM CLOCK BAKELIKE RED)
(ALARM CLOCK BAKELIKE GREEN)            0.094388           0.096939  0.0790
82    0.837838  8.642959  0.069932    5.568878
3                       (ALARM CLOCK BAKELIKE GREEN)
(ALARM CLOCK BAKELIKE RED)              0.096939           0.094388  0.079082
0.815789  8.642959  0.069932    4.916181
4                         (ALARM CLOCK BAKELIKE PINK)
(ALARM CLOCK BAKELIKE RED)              0.102041           0.094388  0.073980
0.725000  7.681081  0.064348    3.293135
5                         (ALARM CLOCK BAKELIKE RED)
(ALARM CLOCK BAKELIKE PINK)             0.094388           0.102041  0.07398
0    0.783784  7.681081  0.064348    4.153061
6                             (DOLLY GIRL LUNCH BOX)
(SPACEBOY LUNCH BOX)             0.099490           0.125000  0.071429    0.
717949  5.743590  0.058992    3.102273
7                               (SPACEBOY LUNCH BOX)
(DOLLY GIRL LUNCH BOX)            0.125000           0.099490  0.071429
0.571429  5.743590  0.058992    2.101190
8                       (PLASTERS IN TIN CIRCUS PARADE)
(PLASTERS IN TIN SPACEBOY)              0.168367           0.137755  0.089286
0.530303  3.849607  0.066092    1.835747
9                         (PLASTERS IN TIN SPACEBOY)                    (PL
ASTERS IN TIN CIRCUS PARADE)            0.137755           0.168367  0.0892
86    0.648148  3.849607  0.066092    2.363588
10                      (PLASTERS IN TIN CIRCUS PARADE)                (PLAST
ERS IN TIN WOODLAND ANIMALS)            0.168367           0.170918  0.1020
41    0.606061  3.545907  0.073264    2.104592
11                (PLASTERS IN TIN WOODLAND ANIMALS)                    (PL
ASTERS IN TIN CIRCUS PARADE)            0.170918           0.168367  0.1020
41    0.597015  3.545907  0.073264    2.063681
12                (PLASTERS IN TIN WOODLAND ANIMALS)
(PLASTERS IN TIN SPACEBOY)              0.170918           0.137755  0.104592
0.611940  4.442233  0.081047    2.221939
13                        (PLASTERS IN TIN SPACEBOY)                (PLAST
ERS IN TIN WOODLAND ANIMALS)            0.137755           0.170918  0.1045
92    0.759259  4.442233  0.081047    3.443878
14                     (SET/6 RED SPOTTY PAPER CUPS)               (SET/20
RED RETROSPOT PAPER NAPKINS)            0.137755           0.132653  0.1020
41    0.740741  5.584046  0.083767    3.345481
15               (SET/20 RED RETROSPOT PAPER NAPKINS)
(SET/6 RED SPOTTY PAPER CUPS)            0.132653           0.137755  0.102
041    0.769231  5.584046  0.083767    3.736395
16               (SET/20 RED RETROSPOT PAPER NAPKINS)                    (SE
T/6 RED SPOTTY PAPER PLATES)            0.132653           0.127551  0.1020
41    0.769231  6.030769  0.085121    3.780612
17                     (SET/6 RED SPOTTY PAPER PLATES)               (SET/20
```

```
RED RETROSPOT PAPER NAPKINS)            0.127551           0.132653  0.1020
41    0.800000  6.030769  0.085121    4.336735
18                     (SET/6 RED SPOTTY PAPER CUPS)                    (SE
T/6 RED SPOTTY PAPER PLATES)            0.137755           0.127551  0.1224
49    0.888889  6.968889  0.104878    7.852041
19                    (SET/6 RED SPOTTY PAPER PLATES)
(SET/6 RED SPOTTY PAPER CUPS)            0.127551           0.137755  0.122
449    0.960000  6.968889  0.104878   21.556122
20  (SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...                    (SE
T/6 RED SPOTTY PAPER PLATES)            0.102041           0.127551  0.0994
90    0.975000  7.644000  0.086474   34.897959
21  (SET/6 RED SPOTTY PAPER CUPS, SET/6 RED SPOTTY...               (SET/20
RED RETROSPOT PAPER NAPKINS)            0.122449           0.132653  0.0994
90    0.812500  6.125000  0.083247    4.625850
22  (SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...
(SET/6 RED SPOTTY PAPER CUPS)            0.102041           0.137755  0.099
490    0.975000  7.077778  0.085433   34.489796
23                     (SET/6 RED SPOTTY PAPER CUPS)  (SET/20 RED RETROSPOT
PAPER NAPKINS, SET/6 RED...            0.137755           0.102041  0.09949
0    0.722222  7.077778  0.085433    3.232653
24               (SET/20 RED RETROSPOT PAPER NAPKINS)  (SET/6 RED SPOTTY PAP
ER CUPS, SET/6 RED SPOTTY...            0.132653           0.122449  0.0994
90    0.750000  6.125000  0.083247    3.510204
25                    (SET/6 RED SPOTTY PAPER PLATES)  (SET/6 RED SPOTTY PAP
ER CUPS, SET/20 RED RETRO...            0.127551           0.102041  0.0994
90    0.780000  7.644000  0.086474    4.081633
```

第10章 推薦系統

本章節從推薦系統介紹為開端，包括以下內容：

- **10.1** 何謂推薦系統 **Recommender System**
- **10.2** 電影推薦系統

10.1 何謂推薦系統 Recommender System

當今社會的每個人都面臨著各種各樣的選擇。例如，如果我漫無目的想找一本書讀，那麼關於如何搜索就會出現很多可能。其結果可能會浪費很多時間在網上瀏覽，並且在各種各樣的網站上搜尋，希望能找到有價值的書籍。這個時候我可能尋找別人的推薦。

如果有一家網站或者手機應用可以基於我以前閱讀的書籍向我推薦新的書籍，那對我肯定有很大的幫助。這時我會有如下愉快的體驗，登錄網站，就可以看到符合我興趣的10本書籍，不用浪費時間在網站上搜尋。

推薦系統的目的是通過發現數據集中的模式，為用戶提供與之最為相關的信息。當你訪問Netflix的時候，谷歌到谷歌閱讀，推薦引擎是機器學習技術中最廣泛的應用之一。

推薦系統是一種信息過濾系統，用於預測用戶對物品的「評分」或「偏好」。

推薦系統近年來非常流行，應用於各行各業。推薦的對象包括：電影、音樂、新聞、書籍、學術論文、搜索查詢、分眾分類、以及其他產品。也有一些推薦系統專門為尋找專家、合作者、笑話、餐廳、美食、金融服務、生命保險、網路交友，以及Twitter頁面設計。

推薦系統包括:

1. 協同過濾 (collaborative filtering)
2. 內容過濾 (content-based filtering) ，或者基於個性化推薦(personality-based approach)
3. 地理過濾 (Demographic Filtering)

協同過濾

- 協同過濾方法根據用戶歷史行為（例如其購買的、選擇的、評價過的物品等）結合其他用戶的相似決策建立模型。
- 這種模型可用於預測用戶對哪些物品可能感興趣（或用戶對物品的感興趣程度）。
- 基於內容推薦利用一些列有關物品的離散特徵，推薦出具有類似性質的相似物品。兩種方法經常互相結合（參考混合推薦系統）
- 協同過濾和基於內容推薦的區別可以比較兩個流行的音樂推薦系統 (Last.fm 和 Pandora Radio)
- Last.fm 建立通過觀察用戶日常收聽的樂隊或歌手，並與其它用戶的行為進行比對，建立一個「電台」，以此推薦歌曲。Last.fm 會播放不在用戶曲庫中，但其他相似用戶經常會播放的其它音樂。鑑於這種方式利用了用戶行為，因此可以認為它是協同過濾技術的一種應用範例。

- Pandora 使用歌曲或者藝人的屬性（由音樂流派項目提供的400個屬性的子集）從而生成一個電台，其中的樂曲都有相似的屬性。用戶的反饋用於精化電台中的內容。在用戶「不喜歡」某一歌曲時，弱化某一些屬性；在用戶喜歡某一歌曲時，強化另一些屬性。這是一種基於內容推薦的方式。

- 每一種系統都有其長處與弱點。在上面的例子中，為了提供精準推薦，Last.fm 需要大量用戶信息。這是一個冷啟動問題，在協同過濾系統中是常見的問題。而 Pandora 啟動時則僅需要很少信息，然而這種方法的局限性很大（例如，這類方法只能得出與原始種子相似的推薦）。
- 推薦系統是一種有效代替搜索算法的方式，因為他們幫助用戶找到一些他們自己沒有辦法找到的物品。有趣的是，推薦系統在實現之時通常使用搜尋引擎對非傳統數據索引。

內容的過濾

- 根據特定項目建議相似的項目。

- 該系統使用項目元數據（例如電影的流派，導演，描述，演員等）來提出這些建議。

- 這些推薦系統背後的總體思想是，如果某人喜歡某個特定項目，那麼他（她）也將喜歡與之相似的項目。

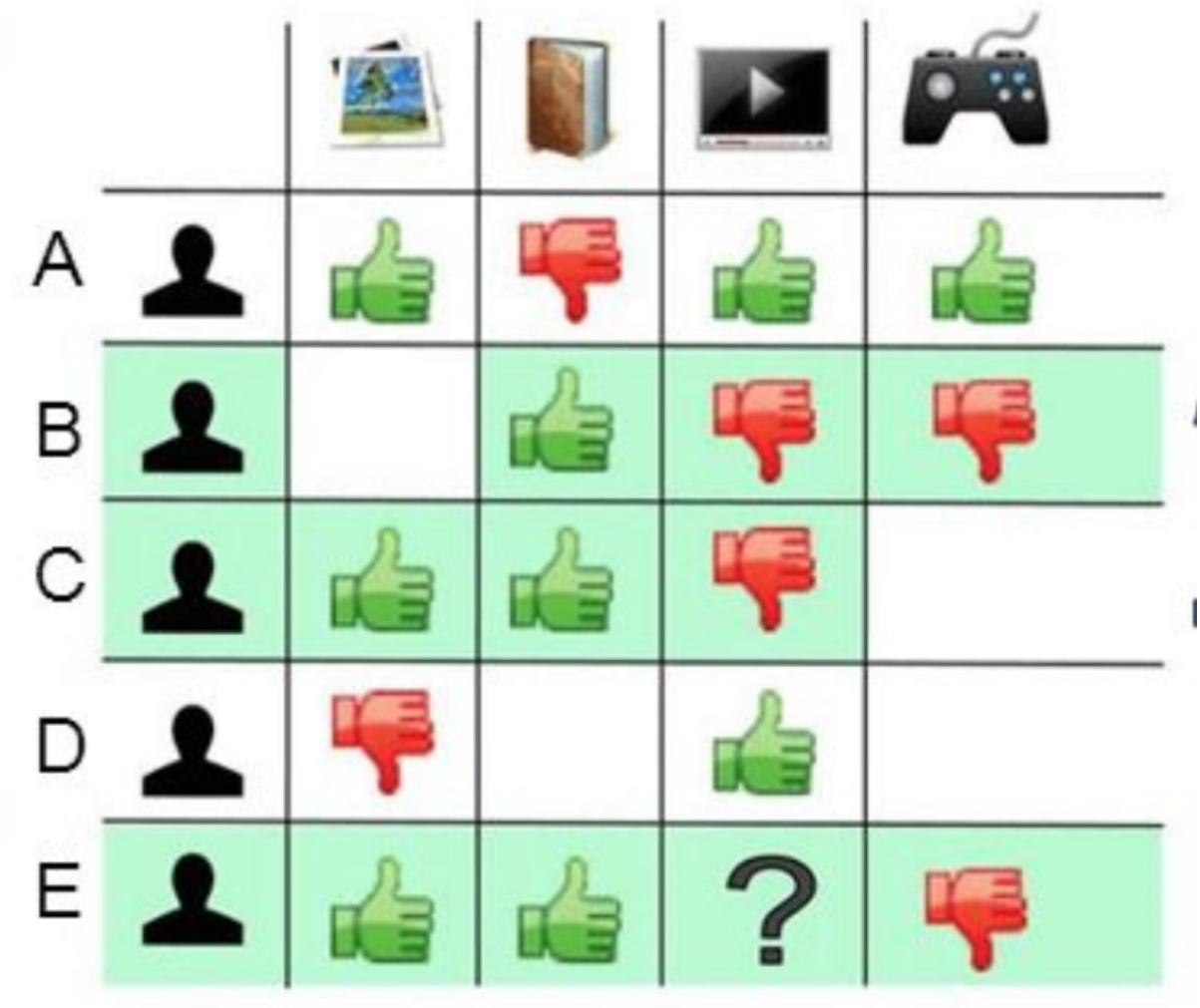
人口統計過濾

- 根據電影的受歡迎程度，它們向每個用戶提供通用建議。 系統向具有相似人口統計特徵的用戶推薦相同的電影。

- 由於每個用戶都不相同，因此該方法被認為過於簡單。

- 該系統背後的基本思想是，更受大眾歡迎和好評的電影具有更高的被普通觀眾喜歡的可能性。

參考資料 推薦系統, https://zh.wikipedia.org/wiki/推薦系統 (https://zh.wikipedia.org/wiki/%E6%8E%A8%E8%96%A6%E7%B3%BB%E7%B5%B1)



協同過濾圖

10.2 電影推薦系統

In [441]:

```
import pandas as pd

import numpy as np

# 參考資料: The Age of Recommender Systems, https://www.kaggle.com/ibtesama/getting-started-
```

In [442]:

```
# 電影資料 TMDB 5000 Movie Dataset

# https://www.kaggle.com/tmdb/tmdb-movie-metadata
```

匯入資料

In [443]:

```
# movies and credits card datatsets, 4803*4

# movie_id - A unique identifier for each movie.
# title
# cast - The name of lead and supporting actors.
# crew - The name of Director, Editor, Composer, Writer etc.

df1=pd.read_csv('data/tmdb-movie-metadata/tmdb_5000_credits.csv')

df1
```

Out[443]:

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

In [444]:

```
# Movie credits data, 4803*20

# budget - The budget in which the movie was made.
# genre - The genre of the movie, Action, Comedy ,Thriller etc.
# homepage - A link to the homepage of the movie.
# id - This is infact the movie_id as in the first dataset.
# keywords - The keywords or tags related to the movie.
# original_language - The language in which the movie was made.
# original_title - The title of the movie before translation or adaptation.
# overview - A brief description of the movie.
# popularity - A numeric quantity specifying the movie popularity.
# production_companies - The production house of the movie.
# production_countries - The country in which it was produced.
# release_date - The date on which it was released.
# revenue - The worldwide revenue generated by the movie.
# runtime - The running time of the movie in minutes.
# spoken_languages
# status - "Released" or "Rumored".
# tagline - Movie's tagline.
# title - Title of the movie.
# vote_average - average ratings the movie recieved.
# vote_count - the count of votes recieved.

df2=pd.read_csv('data/tmdb-movie-metadata/tmdb_5000_movies.csv')

df2
```

Out[444]:

	budget	genres	homepage	id	keywords	c
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	
2	245000000	[{"id": 28, "name": "Action"}, {"id":	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name	

In [445]:

```
df1.columns = ['id','tittle','cast','crew']
```

合併二筆資料

In [446]:

```
df2= df2.merge(df1, on='id') # 將 df1 合併至 df2

df2 # 4803 rows × 26 columns
```

Out[446]:

	budget	genres	homepage	id	
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	"nan cla
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 2 "ocean"
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 4 "spy"
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 8 "dc co
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 8 "basec
...	...	...	...	...	
4798	220000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	NaN	9367	"nai states\u
4799	9000	[{"id": 35, "name": "Comedy"}, {"id": 10749, "...	NaN	72766	
4800	0	[{"id": 35, "name": "Comedy"}, {"id": 18, "nam...	http://www.hallmarkchannel.com/signedsealeddel...	231617	[{"id": 2 "date"
4801	0	[]	http://shanghaicalling.com/	126186	

	budget	genres	homepage	id	
4802	0	[{"id": 99, "name": "Documentary"}]	NaN	25975	"obses

4803 rows × 23 columns

In [447]:

```
df2.head()
```

Out[447]:

	budget	genres	homepage	id	keywords	original
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id": 818, "name": "based on novel"}, {"id":...	

5 rows × 23 columns

人口統計過濾法

- 我們需要一個指標來給電影評分或評分
- 計算每部電影的分數
- 排序分數並向用戶推薦收視率最高的電影。
- 我們可以將電影的平均收視率作為得分，但使用它的評分不夠合理，因為電影的平均評分為8.9，只有3票不能被認為比電影的平均評分為7.8，但只有40票更好。 因此，我將使用IMDB的加權評分（WR, Weighted Rating）：

$$WR = (\frac{v}{v+m} * R) + (\frac{m}{v+m} * C)$$

其中:

- v是電影的票數
- m是圖表中需要列出的最低投票數
- R是電影的平均評分
- C是整個報告的平均投票

In [448]:

```
# 考慮已知v（vote_count）和R（vote_average），C可以計算為:

C= df2['vote_average'].mean()

C # 滿分10分, 整體平均 6.09分
```

Out[448]:

```
6.092171559442011
```

In [449]:

```
# 我們將使用第90個百分位數作為最小閥值。

# 即票數必須超過資料的 90%以上。

m= df2['vote_count'].quantile(0.9)

m
```

Out[449]:

```
1838.4000000000015
```

In [450]:

```
q_movies = df2.copy().loc[df2['vote_count'] >= m]

q_movies.shape # 481*29
```

Out[450]:

```
(481, 23)
```

In [451]:

```
# 定義加權評分

def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']

    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)
```

In [452]:

```
# 定義 score 與 weighted_rating()

q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
```

In [453]:

```
q_movies['score']
```

Out[453]:

```
0       7.050669
1       6.665696
2       6.239396
3       7.346721
4       6.096368
          ...
4291    6.693677
4300    7.366378
4302    7.210428
4337    7.198026
4602    7.210563
Name: score, Length: 481, dtype: float64
```

In [454]:

```
# 依 score 遞減排序
q_movies = q_movies.sort_values('score', ascending=False)

# 顯示前15個
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(15)
```

Out[454]:

	title	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239
3337	The Godfather	5893	8.4	7.851236
95	Interstellar	10867	8.1	7.809479
809	Forrest Gump	7927	8.2	7.803188
329	The Lord of the Rings: The Return of the King	8064	8.1	7.727243
1990	The Empire Strikes Back	5879	8.2	7.697884
262	The Lord of the Rings: The Fellowship of the Ring	8705	8.0	7.667341
2912	Star Wars	6624	8.1	7.663813
1818	Schindler's List	4329	8.3	7.641883
3865	Whiplash	4254	8.3	7.633781
330	The Lord of the Rings: The Two Towers	7487	8.0	7.623893

In [455]:

```
pop= df2.sort_values('popularity', ascending=False)

pop
```

Out[455]:

	budget	genres	homepage	id	keywords	orig
546	74000000	[{"id": 10751, "name": "Family"}, {"id": 16, "...	http://www.minionsmovie.com/	211672	[{"id": 3487, "name": "assistant"}, {"id": 179...	
95	165000000	[{"id": 12, "name": "Adventure"}, {"id": 18, "...	http://www.interstellarmovie.net/	157336	[{"id": 83, "name": "saving the world"}, {"id"...	
788	58000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.foxmovies.com/movies/deadpool	293660	[{"id": 2095, "name": "anti hero"}, {"id": 307...	
94	170000000	[{"id": 28, "name": "Action"}, {"id": 878, "na...	http://marvel.com/guardians	118340	[{"id": 8828, "name": "marvel comic"}, {"id": ...	
127	150000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.madmaxmovie.com/	76341	[{"id": 2964, "name": "future"}, {"id": 3713, ...	
...	...	...	...	...	...	
4625	0	[{"id": 27, "name": "Horror"}]	NaN	426067	[]	
4118	0	[]	NaN	325140	[]	
4727	0	[{"id": 28, "name": "Action"}, {"id": 18, "nam...	NaN	65448	[{"id": 378, "name": "prison"}, {"id": 209476,...	
3361	0	[{"id": 27, "name": "Horror"}, {"id": 28, "nam...	NaN	77156	[]	

	budget	genres	homepage	id	keywords	orig
4553	0	[]	NaN	380097	[]	

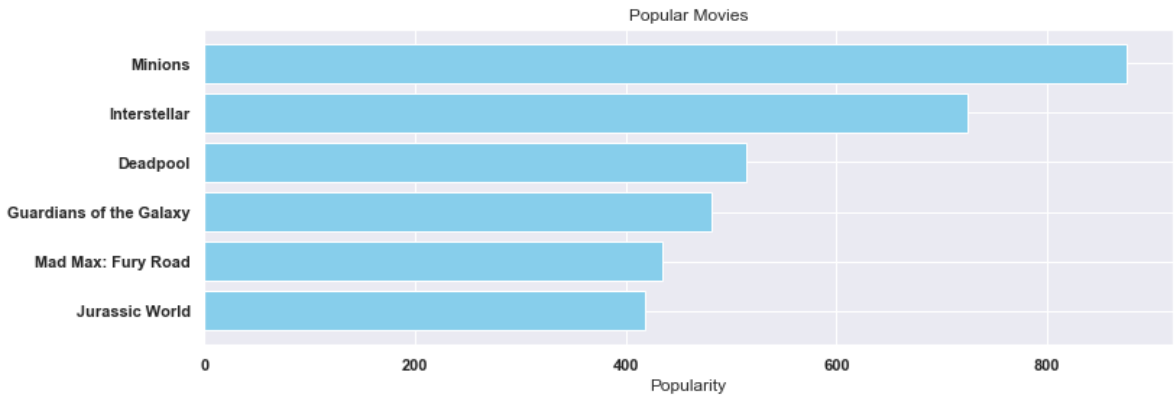
4803 rows × 23 columns

In [456]:

```
# 依照 popularity (人氣) 繪製水平長條圖
import matplotlib.pyplot as plt
plt.figure(figsize=(12,4))
plt.barh(pop['title'].head(6),pop['popularity'].head(6),
         align='center',
         color='skyblue')
plt.gca().invert_yaxis()
plt.xlabel("Popularity")
plt.title("Popular Movies")
```

Out[456]:

```
Text(0.5, 1.0, 'Popular Movies')
```

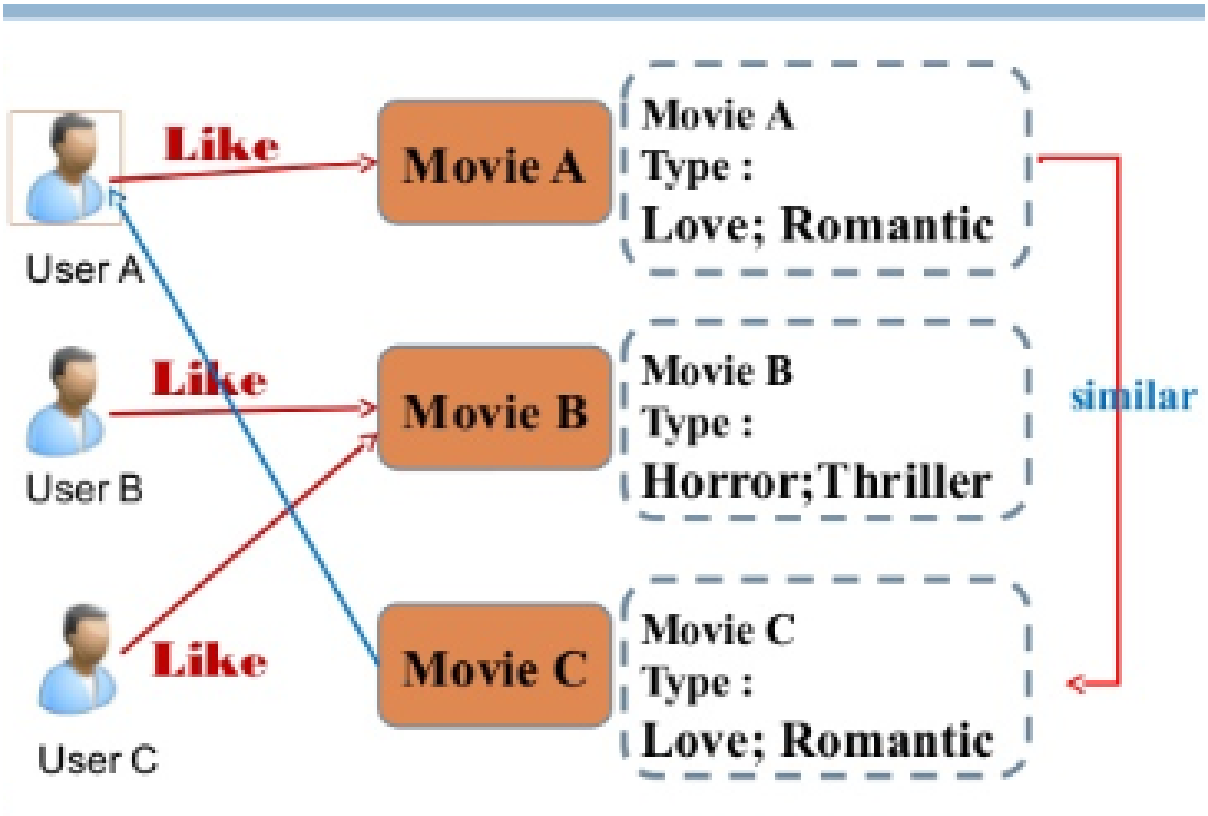


結論：

- 上面是針對所有用戶提供了推薦電影的一般圖表。
- 他們對特定用戶的興趣和愛好不敏感。
- 以下可採用更為完善的系統-基於內容的過濾。

基於內容的過濾

在此推薦系統中，電影的內容（概述，演員，工作人員，關鍵字，標語等）用於尋找其他電影的相似性。再推薦最可能相似的電影。



- 將根據所有電影的情節描述計算成對相似度得分，並根據相似度得分推薦電影。
- 在數據集的概覽功能中給出了圖解說明。

In [457]:

```
df2['overview'].head()
```

Out[457]:

```
0    In the 22nd century, a paraplegic Marine is di...
1    Captain Barbossa, long believed to be dead, ha...
2    A cryptic message from Bond’s past sends him o...
3    Following the death of District Attorney Harve...
4    John Carter is a war-weary, former military ca...
Name: overview, dtype: object
```

計算 **TF-IDF (Term Frequency-Inverse Document Frequency)**

In [458]:

```
# Import TfIdfVectorizer from scikit-learn

from sklearn.feature_extraction.text import TfidfVectorizer
```

In [459]:

```
# Define a TF-IDF Vectorizer Object. Remove all english stop words such as 'the', 'a'

tfidf = TfidfVectorizer(stop_words='english')
```

In [460]:

```
# Replace NaN with an empty string

df2['overview'] = df2['overview'].fillna('')
```

In [461]:

```
#Construct the required TF-IDF matrix by fitting and transforming the data

tfidf_matrix = tfidf.fit_transform(df2['overview'])
```

In [462]:

```
tfidf_matrix
```

Out[462]:

```
<4803x20978 sparse matrix of type '<class 'numpy.float64'>'
        with 125840 stored elements in Compressed Sparse Row format>
```

In [463]:

```
# Output the shape of tfidf_matrix

tfidf_matrix.shape
```

Out[463]:

```
(4803, 20978)
```

- 我們看到數據集中使用了20,000多個不同的詞來描述4800部電影。
- 有了這個矩陣，可以計算一個相似度分數 (Similarity)。例如歐幾里得，皮爾遜和餘弦相似度得分。
- 沒有哪個分數最好的正確答案。不同的分數在不同的情況下效果很好，嘗試使用不同的指標通常是一個好主意。

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

使用 **sklearn linear_kernel**

In [464]:

```
# Import linear_kernel

from sklearn.metrics.pairwise import linear_kernel

# Compute the cosine similarity matrix

cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

In [465]:

```
# identify the index of a movie in our metadata DataFrame, given its title.

# Construct a reverse map of indices and movie titles
indices = pd.Series(df2.index, index=df2['title']).drop_duplicates()
```

In [466]:

```
indices
```

Out[466]:

```
title
Avatar                                         0
Pirates of the Caribbean: At World's End       1
Spectre                                        2
The Dark Knight Rises                          3
John Carter                                    4
                                            ...
El Mariachi                                 4798
Newlyweds                                   4799
Signed, Sealed, Delivered                   4800
Shanghai Calling                            4801
My Date with Drew                           4802
Length: 4803, dtype: int64
```

建立推薦函數 **Recommendation functio**

- 根據標題獲得電影的索引。
- 獲取該特定電影與所有電影的餘弦相似度得分列表。將其轉換為元組列表 (list of tuples)，其中第一個元素是其位置，第二個元素是相似性分數。
- 根據相似度分數對上述元組列表進行排序；即是第二個元素。
- 獲取此列表的前10個元素。 忽略第一個涉及自我的元素（與特定電影最相似的電影是電影本身）。
- 回傳與頂部元素的索引對應的標題。

In [467]:

```
# Function that takes in movie title as input and outputs most similar movies

def get_recommendations(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return df2['title'].iloc[movie_indices]
```

In [468]:

```
get_recommendations('The Dark Knight Rises')
```

Out[468]:

```
65                              The Dark Knight
299                              Batman Forever
428                              Batman Returns
1359                                    Batman
3854    Batman: The Dark Knight Returns, Part 2
119                               Batman Begins
2507                                  Slow Burn
9            Batman v Superman: Dawn of Justice
1181                                       JFK
210                              Batman & Robin
Name: title, dtype: object
```

In [469]:

```
get_recommendations('The Avengers')
```

Out[469]:

```
7               Avengers: Age of Ultron
3144                            Plastic
1715                            Timecop
4124                This Thing of Ours
3311              Thank You for Smoking
3033                      The Corruptor
588     Wall Street: Money Never Sleeps
2136         Team America: World Police
1468                       The Fountain
1286                        Snowpiercer
Name: title, dtype: object
```

Credits, Genres and Keywords Based Recommender

- 基於片名，題材和關鍵字的推薦系統。
- 使用更好的元數據 (metadata) 將提高我們推薦程序的質量。 後續將基於以下元數據構建推薦系統：

  1. 前三個頂級演員

  2. 導演

  3. 相關題材

  4. 電影情節關鍵字

- 從演員，劇組和關鍵字功能中，我們需要提取三個最重要的演員，導演和與該電影相關的關鍵字。 數據以"字符串化"列表 (Stringified Lists) 的形式出現，我們需要將其轉換為安全且可用的結構

In [470]:

```
# Parse the stringified features into their corresponding python objects

from ast import literal_eval

features = ['cast', 'crew', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(literal_eval)
```

In [471]:

```
# Get the director's name from the crew feature. If director is not listed, return NaN

def get_director(x):
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan
```

In [472]:

```
# Returns the list top 3 elements or entire list; whichever is more.

def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
        #Check if more than 3 elements exist. If yes, return only first three. If no, retur
        if len(names) > 3:
            names = names[:3]
        return names

    #Return empty list in case of missing/malformed data
    return []
```

In [473]:

```
# Define new director, cast, genres and keywords features that are in a suitable form.

df2['director'] = df2['crew'].apply(get_director)

features = ['cast', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(get_list)
```

In [474]:

```
# Print the new features of the first 3 films

df2[['title', 'cast', 'director', 'keywords', 'genres']]
```

Out[474]:

	title	cast	director	keywords	genres
0	Avatar	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	James Cameron	[culture clash, future, space war]	[Action, Adventure, Fantasy]
1	Pirates of the Caribbean: At World's End	[Johnny Depp, Orlando Bloom, Keira Knightley]	Gore Verbinski	[ocean, drug abuse, exotic island]	[Adventure, Fantasy, Action]
2	Spectre	[Daniel Craig, Christoph Waltz, Léa Seydoux]	Sam Mendes	[spy, based on novel, secret agent]	[Action, Adventure, Crime]
3	The Dark Knight Rises	[Christian Bale, Michael Caine, Gary Oldman]	Christopher Nolan	[dc comics, crime fighter, terrorist]	[Action, Crime, Drama]
4	John Carter	[Taylor Kitsch, Lynn Collins, Samantha Morton]	Andrew Stanton	[based on novel, mars, medallion]	[Action, Adventure, Science Fiction]

In [475]:

```
# Function to convert all strings to lower case and strip names of spaces

def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        #Check if director exists. If not, return empty string
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
        else:
            return ''
```

In [476]:

```
# Apply clean_data function to your features.

features = ['cast', 'keywords', 'director', 'genres']

for feature in features:
    df2[feature] = df2[feature].apply(clean_data)
```

In [477]:

```
def create_soup(x):
    return ' '.join(x['keywords']) + ' ' + ' '.join(x['cast']) + ' ' + x['director'] + ' '
df2['soup'] = df2.apply(create_soup, axis=1)
```

In [478]:

```
# Import CountVectorizer and create the count matrix

from sklearn.feature_extraction.text import CountVectorizer

count = CountVectorizer(stop_words='english')

count_matrix = count.fit_transform(df2['soup'])
```

In [479]:

```
# Compute the Cosine Similarity matrix based on the count_matrix

from sklearn.metrics.pairwise import cosine_similarity

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
```

In [480]:

```
# Reset index of our main DataFrame and construct reverse mapping as before

df2 = df2.reset_index()
indices = pd.Series(df2.index, index=df2['title'])
```

In [481]:

```
get_recommendations('The Dark Knight Rises', cosine_sim2)
```

Out[481]:

```
65               The Dark Knight
119                Batman Begins
4638    Amidst the Devil's Wings
1196                 The Prestige
3073           Romeo Is Bleeding
3326              Black November
1503                      Takers
1986                      Faster
303                     Catwoman
747               Gangster Squad
Name: title, dtype: object
```

In [482]:

```
get_recommendations('The Godfather', cosine_sim2)
```

Out[482]:

```
867      The Godfather: Part III
2731      The Godfather: Part II
4638    Amidst the Devil's Wings
2649           The Son of No One
1525              Apocalypse Now
1018             The Cotton Club
1170     The Talented Mr. Ripley
1209               The Rainmaker
1394               Donnie Brasco
1850                    Scarface
Name: title, dtype: object
```

參考文獻

1. Coelho, L. P., Richert, W., Building Machine Learning Systems with Python, Second Edition, Packt Publishing, 2015.
2. Google Python Style Guide, http://google.github.io/styleguide/pyguide.html (http://google.github.io/styleguide/pyguide.html), 2019.
3. Matplotlib, https://matplotlib.org/ (https://matplotlib.org/), 2019.
4. Pandas, https://pandas.pydata.org/ (https://pandas.pydata.org/), 2019.
5. Python, https://www.python.org/ (https://www.python.org/), 2019.
6. Python Tutorial, https://www.w3schools.com/python/default.asp (https://www.w3schools.com/python/default.asp), 2019.
7. Scikit-learn in Python, https://scikit-learn.org/stable/ (https://scikit-learn.org/stable/), 2019.