本書導讀

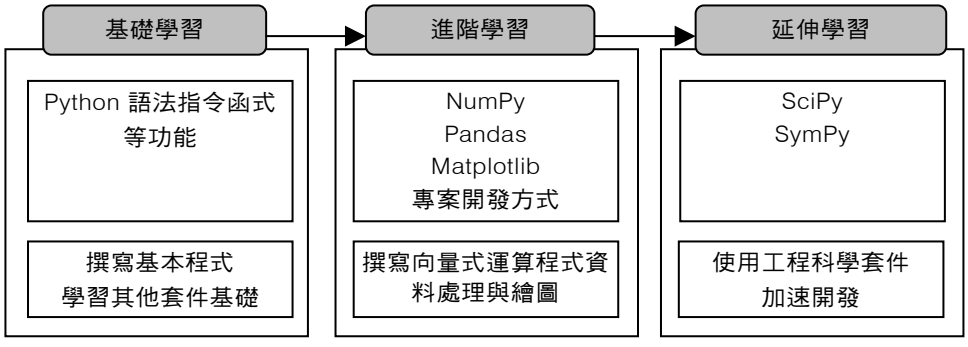
To 讀者：

本書旨在系統化的協助初學者從軟體工具安裝，Python 基本語法學習，至 NumPy、Pandas、Matplotlib、四項最重要之軟體套件，足夠用於自行開發之小型專案，此外再延伸至 SciPy 與 SymPy，得以進行更複雜之工程科學分析計算模擬等。

因此構想為初學主打造之工程科學分析之主要內涵如下：

PART 1：Python 基礎學習（Chapter 1 ~ Chapter 9）
1. Python：基本工具，語法與資料結構，程序流程，函式模組套件，類別，物件導向，檔案輸入輸出，GUI 設計等。是 Python 學習之基礎。
PART 2：進階學習（Chapter 10 ~ Chapter 13）
1. NumPy：引入後即可方便快速的處理工程科學中之向量矩陣計算。 2. Pandas：讓使用者方便的處理資料框之數據。 3. Matplotlib：提供多樣化之 2D/3D 繪圖功能，讓使用者進行視覺化資料表達。 4. 整合專案開發方式
PART 3：延伸學習（Chapter 14, 15）
1. SciPy：提供大量之工程科學分析工具，大量簡化常用分析函式之開發時間。 2. SymPy：提供使用者符號運算之處理，許多冗長重複之數學推導得以電腦協助完成。

學習流程圖如下：



To 教師：

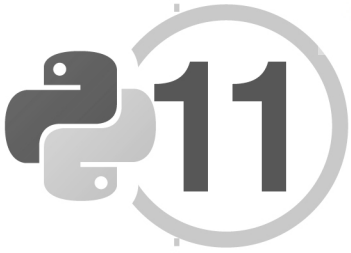
18 週之課程規畫安排之建議如下（期中考前為 Python 基本學習，期中考後為進階學習）：

週次	章節進度	名稱	備註
1	第 1 章 第 2 章	Python 簡介 Python 與 Anaconda 下載與安裝	*基礎學習 第 2 章可示範或講解後由學生自行回家安裝
2	第 3 章	資料處理	
3	第 4 章	流程控制	
4	第 5 章	資料結構	
5	第 6 章	檔案輸入輸出	
6	第 7 章	函式模組與套件	
7	第 8 章	物件導向	
8	第 9 章	Turtle 與 GUI 元件學習	
9	-	期中考	期中考
10	第 10 章	NumPy 套件	*進階學習
11	第 11 章	Pandas 套件（1）	
12	第 11 章	Pandas 套件（2）	
13	第 12 章	Matplotlib 繪圖套件（1）：figure,subplot	
14	第 12 章	Matplotlib 繪圖套件（2）：2D	
15	第 12 章	Matplotlib 繪圖套件（3）：3D	
16	第 13 章	專案開發（1）	
17	第 13 章	專案開發（2）	
18	-	期末考或分組專案報告	期末考/分組專案報告
19	第 14 章 第 15 章	SciPy SymPy	*延伸學習 （寒暑假自習）

* 時間足夠或學習較快者，教師可調整進度將第 14、15 章納入更佳。

適用版本

本書以 Python 3.6.4 為主要對象，但適用於 Python 2.7~ 3.*（版本會持續更新）程式開發界面，Anaconda 則為 5.01 for windows （版本會持續更新）。



11 Pandas 套件之基本使用

本章主旨 **介紹 Pandas 套件之基本使用，讓使用者可以對數據資料之整理與讀取更為方便。**

11-1 Pandas 資料處理套件簡介

Pandas 是 Python 語言的一個數據分析套件，2009 年底開源出來，相關資源可以參考官方網站 www.Pandas.org。Pandas 可以提供高效能、簡易使用的資料格式（Data Frame）讓使用者可以快速操作及分析資料，主要特色描述如下：

- 在異質數據的讀取、轉換和處理上，都讓分析人員更容易處理，例如：從列欄試算表中找到想要的值。
- Pandas 提供兩種主要的資料結構，Series 與 DataFrame。Series 顧名思義就是用來處理時間序列相關的資料（如感測器資料等），主要為建立索引的一維陣列。DataFrame 則是用來處理結構化（Table like）的資料，有列索引與欄標籤的二維資料集，例如關聯式資料庫、CSV 等等。
- 透過載入至 Pandas 的資料結構物件後，可以透過結構化物件所提供的方法，來快速地進行資料的前處理，如資料補值，空值去除或取代等。
- 更多的輸入來源及輸出整合性，例如：可以從資料庫讀取資料進入 DataFrame，也可將處理完的資料存回資料庫。

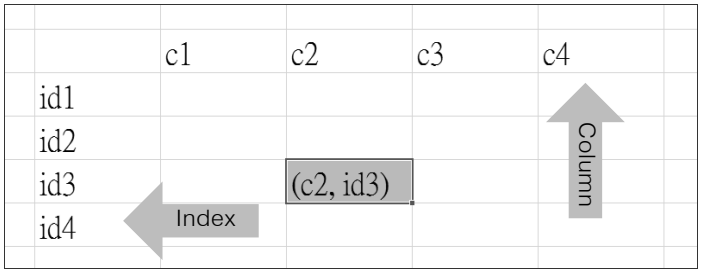
如果讀者按照第 2 章安裝 Python 及 Anaconda，則 Pandas 套件也會自動安裝進來，非常方便。以下為幾個相關資訊查詢指令：

	用途	指令
1	載入 Pandas 套件	import pandas as pd
2	查詢目前 Pandas 版本	pd.__version__
3	查閱套件內容	help(pd)

11-2 Pandas 使用的資料結構

Pandas 主要提供兩種資料結構以利後續數據分析使用，如下所述。

- Series：用來處理時間序列相關的資料（如感測器資料等），主要為建立索引的一維陣列。
- DataFrame：用來處理結構化（Table like）的資料，有列索引與欄標籤的二維資料集，例如關聯式資料庫、CSV 等等。



11-2-1 Series

1. 建立 Series

可以依據 Array 陣列、Dictionary 字典或是單一輸入資料來建立 Series。

使用方法。

```
# 載入 Pandas 套件並命名為 pd
import pandas as pd

#將資料 data 讀入 series 方法中
select = pd.series(data, index = idx)
```

當輸入資料為陣列範例如下：

```
#-------------------------------------------------
#   CH_11_2_1_EX_1_Pandas_Series_Input_Array.py
#-------------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----定義 Array 陣列資料
cars = ["BMW", "BENZ", "Toyota", "Nissan", "Lexus"]

select = pd.Series(cars)
print(select)
```

執行結果如下：

```
0     BMW
1     BENZ
2     Toyota
3     Nissan
4     Lexus
dtype: object
```

當輸入資料為字典範例如下：

```
#-------------------------------------------------------
#   CH_11_2_1_EX_2_Pandas_Series_Input_Dictionary.py
#-------------------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----定義 Dictionary 字典資料
dict = {
    "factory": "Taipei",
    "sensor1": "1",
    "sensor2": "2",
    "sensor3": "3",
    "sensor4": "4",
    "sensor5": "5"
}

#-----排序與原 dict 相同
select = pd.Series(dict, index = dict.keys())
print(select)
```

執行結果如下：

```
factory     Taipei
sensor1          1
sensor2          2
sensor3          3
sensor4          4
sensor5          5
dtype: object
```

2. Series 操作

(1) 資料選擇與篩選

可以針對字典或是陣列資料透過索引值或標籤，挑選出你所要的值。

Series 資料選擇與篩選範例如下：

```
#-----------------------------------------
#   CH_11_2_1_EX_3_Pandas_Series_Select.py
#-----------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----定義 Dictionary 字典資料
dict = {
    "factory": "Taipei",
    "sensor1": "1",
    "sensor2": "2",
    "sensor3": "3",
    "sensor4": "4",
    "sensor5": "5"
}

#-----排序與原 dict 相同
select = pd.Series(dict, index = dict.keys())
print(select[0])
print("-"*30)
print(select['sensor1'])
print("-"*30)
print(select[[0, 2, 4]])
print("-"*30)
print(select[['factory', 'sensor1', 'sensor3']])
```

執行結果如下：

```
Taipei
------------------------------
1
------------------------------
factory     Taipei
sensor2          2
sensor4          4
dtype: object
------------------------------
factory     Taipei
sensor1          1
sensor3          3
dtype: object
```

(2) 資料切片

透過透過索引值或標籤，切割需要的值。

Series 資料切片範例如下：

```
#-------------------------------------------
#   CH_11_2_1_EX_4_Pandas_Series_Slicing.py
#-------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----定義 Dictionary 字典資料
dict = {
    "factory": "Taipei",
    "sensor1": "1",
    "sensor2": "2",
    "sensor3": "3",
    "sensor4": "4",
    "sensor5": "5"
}

#-----排序與原 dict 相同
select = pd.Series(dict, index = dict.keys())
print(select[:2])
print("-"*30)
print(select['sensor2':])
```

執行結果如下：

```
factory    Taipei
sensor1         1
dtype: object
------------------------------
sensor2    2
sensor3    3
sensor4    4
sensor5    5
dtype: object
```

11-2-2 DataFrame

1. 建立 DataFrame

可以依據 Array 陣列或是 Dictionary 字典資料來建立 DataFrame。

當輸入資料為字典範例如下：

```
#-------------------------------------------------
#   CH_11_2_2_EX_1_Pandas_DataFrame_Dictionary.py
#-------------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

groups = ["Movies", "Sports", "Coding", "Fishing", "Dancing", "cooking"]
num = [46, 8, 12, 12, 6, 58]

dict = {"groups": groups,
      "num": num
     }

select_df = pd.DataFrame(dict)

#-----看看資料框的外觀
print(select_df)
```

執行結果如下：

```
    groups  num
0   Movies   46
1   Sports    8
2   Coding   12
3  Fishing   12
```

```
4  Dancing    6
5  cooking   58
```

當輸入資料為陣列範例如下：

```
#-------------------------------------------
#   CH_11_2_2_EX_2_Pandas_DataFrame_Array.py
#-------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

arr = groups = [["Movies", 46],["Sports", 8], ["Coding", 12],
["Fishing",12], ["Dancing",6], ["Cooking",8]]

#-----指定欄標籤名稱
df = pd.DataFrame(arr, columns = ["name", "num"])
print(df)
```

執行結果如下：

```
      name  num
0   Movies   46
1   Sports    8
2   Coding   12
3  Fishing   12
4  Dancing    6
5  Cooking    8
```

2. DataFrame 的操作

(1) 資料查詢

可以透過下列方法查看目前資料的資訊

方法	說明
shape	回傳列數與欄數
describe()	回傳描述性統計
head(n)	回傳前 n 筆觀測值
tail(n)	回傳後 n 筆觀測值
columns	回傳欄位名稱
index	回傳 index
info()	回傳資料內容

資料查詢範例如下：

```
#-------------------------------------------
#   CH_11_2_2_EX_3_Pandas_DataFrame_Info.py
#-------------------------------------------
#-----Pandas 的 Data Frame 資料結構有一些方法或屬性
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

groups = ["Movies", "Sports", "Coding", "Fishing", "Dancing", "Cooking"]
num = [46, 8, 12, 12, 6, 58]

dict = {"groups": groups,
        "num": num
        }

select_df = pd.DataFrame(dict)

#-----回傳列數與欄數
print(select_df.shape)
print("-"*30)

#-----回傳描述性統計
print(select_df.describe())
print("-"*30)

#-----回傳前三筆觀測值
print(select_df.head(3))
print("-"*30)

#-----回傳後三筆觀測值
print(select_df.tail(3))
print("-"*30)

#-----回傳欄位名稱
print(select_df.columns)
print("-"*30)

#-----回傳 index
print(select_df.index)
print("-"*30)

#-----回傳資料內容
print(select_df.info)
```

執行結果如下：

```
    (6, 2)
------------------------------
            num
count    6.000000
mean    23.666667
std     22.393451
min      6.000000
25%      9.000000
50%     12.000000
75%     37.500000
max     58.000000
------------------------------
   groups  num
0  Movies   46
1  Sports    8
2  Coding   12
------------------------------
    groups  num
3  Fishing   12
4  Dancing    6
5  cooking   58
------------------------------
Index(['groups', 'num'], dtype='object')
---
RangeIndex(start=0, stop=6, step=1)
------------------------------
<bound method DataFrame.info of     groups  num
0   Movies   46
1   Sports    8
2   Coding   12
3  Fishing   12
4  Dancing    6
5  Cooking   58>
```

(2) 資料選擇與篩選

可以透過下列方法選擇元素

- 中括號 [] 選擇元素

  範例如下：

```
#-----------------------------------------------
#   CH_11_2_2_EX_4_Pandas_DataFrame_Select_1.py
#-----------------------------------------------
```

```
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----定義字典資料
dict = {'A':[1,2,3],'B':[4,5,6],'C':[7,8,9]}

#-----資料匯入 DataFrame
select_df = pd.DataFrame(dict, index=["a","b","c"])

#-----選擇元素
print(select_df["A"])
```

執行結果如下：

```
a    1
b    2
c    3
Name: A, dtype: int64
```

- 將變數當作屬性選擇

範例如下：

```
#-------------------------------------------------
#   CH_11_2_2_EX_5_Pandas_DataFrame_Select_2.py
#-------------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----定義字典資料
dict = {'A':[1,2,3],'B':[4,5,6],'C':[7,8,9]}

#-----資料匯入 DataFrame
select_df = pd.DataFrame(dict, index=["a","b","c"])

#-----選擇元素
print(select_df.B)
```

執行結果如下：

```
a    4
b    5
c    6
Name: B, dtype: int64
```

- Loc[]及 iloc[]方法選擇

loc[]其中括號裡面是先行後列，以逗號分開，行和列分別是行標籤和列標籤，比如我要得到數字 5，那麼就就是：loc ["b","B"]。iloc[] 與 loc 一樣，中括弧裡面也是先行後列，行列標籤用逗號分割，與 loc 不同的之處是，.iloc 是根據行數與列數來索引的，比如上面提到的得到數字 5，那麼用 iloc 來表示就是 data.iloc[1,1],因為數字 5 是第 2 行第 2 列的值，注意索引值是從 0 開始的，同理 4 就是 data.iloc[0,1]，同樣如果我們需要選擇一個區域，比如我要選擇 5，8，6，9，那麼用 iloc 來選擇就是 iloc[1：3,1：3]。

範例如下：

```
#-------------------------------------------------
#   CH_11_2_2_EX_6_Pandas_DataFrame_Select_3.py
#-------------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----定義字典資料
dict = {'A':[1,2,3],'B':[4,5,6],'C':[7,8,9]}

#-----資料匯入 DataFrame
select_df = pd.DataFrame(dict, index=["a","b","c"])

#-----選擇元素
print(select_df.loc ["b","B"])

#-----選擇區域元素
print(select_df.iloc[1:3,1:3])
```

執行結果如下：

```
5
------------------------------
   B  C
b  5  8
c  6  9
```

- 使用邏輯運算篩選

範例如下：

```
#-------------------------------------------------
#   CH_11_2_2_EX_7_Pandas_DataFrame_Boolean.py
#-------------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

groups = ["Movies", "Sports", "Coding", "Fishing", "Dancing", "Cooking"]
num = [46, 8, 12, 12, 6, 58]

dict = {"groups": groups,
        "num": num
        }

select_df = pd.DataFrame(dict)

#-----選出人數超過 10 的群組
out_df = select_df[select_df.loc[:,"num"] > 10]
print(out_df)
```

執行結果如下：

```
    groups  num
0   Movies   46
2   Coding   12
3  Fishing   12
5  Cooking   58
```

(3) 資料排序

資料排序可以使用的方法如下：

- .sort_index()
- .sort_values()

sort_index()範例如下：

```
#-------------------------------------------------
#   CH_11_2_2_EX_8_Pandas_DataFrame_ Sort_Index.py
#-------------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd
```

```
groups = ["Movies", "Sports", "Coding", "Fishing", "Dancing", "Cooking"]
num = [46, 8, 12, 12, 6, 58]

dict = {"groups": groups,
        "num": num
        }

select_df = pd.DataFrame(dict)

#-----透過索引值做排序，axis 可以指定第幾欄，ascending 用於設定升冪或降冪
print(select_df.sort_index(axis = 0, ascending = True))
```

執行結果如下：

```
    groups  num
0   Movies   46
1   Sports    8
2   Coding   12
3  Fishing   12
4  Dancing    6
5  Cooking   58
```

sort_values()範例如下：

```
#------------------------------------------------------
#   CH_11_2_2_EX_9_Pandas_DataFrame_ Sort_Values.py
#------------------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

groups = ["Movies", "Sports", "Coding", "Fishing", "Dancing", "Cooking"]
num = [46, 8, 12, 12, 6, 58]

dict = {"groups": groups,
        "num": num
        }

select_df = pd.DataFrame(dict)

#-----透過指定欄位的數值排序
print(select_df.sort_values(by = 'num'))
```

執行結果如下：

```
    groups  num
4  Dancing    6
1   Sports    8
2   Coding   12
3  Fishing   12
```

```
0   Movies   46
5  Cooking   58
```

11-3 Pandas 讀取外部資料

可以從異質資料來源讀取檔案內容，並將資料放入 DataFrame 中，進行資料查看、資料篩選、資料切片等運算。

11-3-1 讀取文字檔

基本上可以以文字編輯器開啟檢視的檔案，可稱為文字檔如 TXT 檔或是 CSV 檔，CSV 檔其檔案格式主要是透過","進行分隔 CSV 中的 C 就是英文字 Comma 的縮寫，"\n"進行換行，第一行是 column headers。Pandas 套件可以將他處理成 DataFrame 格式供後續數據分析使用。

使用方式如下：

```
df = pd.read_csv("file_name")
```

讀取文字 CSV 檔範例如下：

```
#-----------------------------------------------
#   CH_11_3_1_EX_1_Pandas_Reading_CSV_File.py
#-----------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----讀取 CSV File
df = pd.read_csv('books.csv')
print(df)
```

執行結果如下：

```
                Name       ISBN                      Publisher  Language
0                BIM 1118942760                          Wiley  English
1       Steel Design 1337094749                CL Engineering  English
2 Sustainable Design 1568989415 Princeton Architectural Press  English
```

11-3-2 讀取電子試算表 EXCEL 檔

Pandas 套件支援 Excel 2003（.xls）及 Excel 2007 以上（.xlsx）兩種格式，可以將其資料處理成 DataFrame 格式供後續數據分析使用。

常用使用方式如下：

```
df = pd.read_excel ("file_name", "sheet_name")
```

讀取 EXCEL 檔範例如下：

```
#-----------------------------------------------
#   CH_11_3_2_EX_1_Pandas_Reading_Excel_File.py
#-----------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----讀取 EXCEL File，若沒指定工作表則讀取第一個工作表
df = pd.read_excel("markets.xlsx")
print(df)

print("-"*30)

#-----指定則讀取 Stock 工作表
df = pd.read_excel("markets.xlsx", sheetname="Stock")
print(df)
```

執行結果如下：

```
                        Market       Open     Day High      Day Low
0 Dow Jones Industrial Average   23995.18    24108.47    23509.06
1             NASDAQ Composite    7170.68     7194.31     6992.67
2                      S&P 500    2646.71     2657.67     2585.89
------------------------------
               Name    Price
0             Apple   164.95
1         Citigroup    67.91
2 General Electric    13.07
3            Google  1027.31
4         Microsoft    87.08
5          Facebook   159.39
```

11-3-3 讀取 HTML 檔

Pandas 套件可以讀取網頁 HTML 標籤格式內容將其資料處理成 DataFrame 格式供後續數據分析使用。必須注意回傳的是一個由很多 DataFrame 組成的 list，因為一般 html 檔中可能會出現很多個 table。

使用方式如下：

```
dfs= pd.read_html("URL")
```

讀取 HTML 檔範例如下：

```
#-----------------------------------------------
#   CH_11_3_3_EX_1_Pandas_Reading_HTML_File.py
#-----------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd
dfs = pd.read_html("http://rate.bot.com.tw/xrt?Lang=zh-TW")
print(dfs[0])
```

11-4 Pandas 資料處理

空值對於後續資料處理上會產生很大的問題，所以在資料處理前必須依序判斷各欄位值是否有空值，然後再進行後續處理或是填補空值。

1. 判斷空值

可以使用下列兩種方法來判斷空值。

方法	說明
isnull()	判斷是否為空值
notnull()	判斷是否不是空值

判斷空值方法範例如下：

```
#-----------------------------------------
#   CH_11_4_EX_1_Pandas_DataFrame_Null.py
#-----------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd
```

```
#-----讀取 CSV File
df = pd.read_csv('shops.csv')
print(df)

#-----讀取後放入 DataFrame
select_df = pd.DataFrame(df)

#-----可以透過 ix 方法(利用索引值或欄位)選擇 data frame 區段
#-----判斷哪些店名是遺失值
print(select_df.ix[:, "shop name"].isnull())
print("-"*30)

#-----判斷哪些市場規模不是遺失值
print(select_df.ix[:, "maket size"].notnull())
```

執行結果如下：

```
   shop id shop name  maket size
0        1  Wal mart   3000000.0
1        2    Costco   2000000.0
2        3       NaN   1500000.0
3        4    Pchome    300000.0
4        5     Yahoo          NaN
0    False
1    False
2     True
3    False
4    False
Name: shop name, dtype: bool
------------------------------
0     True
1     True
2     True
3     True
4    False
Name: maket size, dtype: bool
```

2. 處理空值

可以使用下列兩種方法來填補空值。

方法	說明
dropna()	判斷是否為空值
fillna()	判斷是否不是空值

填補空值方法範例如下：

```
#-----------------------------------------------
#   CH_11_4_EX_2_Pandas_DataFrame_FillData.py
#-----------------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd
#-----讀取 CSV File
df = pd.read_csv('shops.csv')
print(df)

#-----讀取後放入 DataFrame
select_df = pd.DataFrame(df)

#-----有遺失值的觀測值都刪除
drop_value = select_df.dropna()
print(drop_value)
print("-"*30)

#-----有遺失值的觀測值填補 0
filled_value = select_df.fillna(0)
print(filled_value)
print("-"*30)

#-----依欄位填補遺失值
filled_value_column = select_df.fillna({"shop name": "NULL", "maket
size": 0})
print(filled_value_column)
```

執行結果如下：

```
   shop id shop name  maket size
0        1  Wal mart   3000000.0
1        2    Costco   2000000.0
2        3       NaN   1500000.0
3        4    Pchome    300000.0
4        5     Yahoo         NaN
   shop id shop name  maket size
0        1  Wal mart   3000000.0
1        2    Costco   2000000.0
3        4    Pchome    300000.0
------------------------------
   shop id shop name  maket size
0        1  Wal mart   3000000.0
1        2    Costco   2000000.0
2        3         0   1500000.0
3        4    Pchome    300000.0
```

```
4         5      Yahoo          0.0
------------------------------
   shop id shop name  maket size
0         1  Wal mart   3000000.0
1         2    Costco   2000000.0
2         3      NULL   1500000.0
3         4    Pchome    300000.0
4         5     Yahoo          0.0
```

11-5 Pandas 資料視覺化

對於 Python 視覺化，第一個想到的就是 Matplotlib 套件，是很好用的視覺化出圖套件，主要將圖表元素，如座標軸、線、點或者文字分開處理，出圖時再用不同的方法一一拼湊起來，繪圖較彈性但是缺點就是繪製程序較複雜使用門檻較高，若想要快速繪製出基本的圖表來檢視資料，了解資料的含義，Pandas 基於這個問題，將 matplotlib.pyplot 的基礎圖形包裝起來成為一個方法讓使用者只要呼叫 df.plot()就能夠便利地繪圖，可以選擇的圖形種類相當豐富，如下所示。

圖表	說明
line	折線圖，為 plot 方法預設圖表。
bar	直條圖
barh	橫條圖
hist	直方圖
box	箱形圖
kde	Kernel density estimation 核密度估計圖
density	如 kde 圖，繪製密度圖可以簡易了解資料的機率分佈。
area	區域圖
pie	圓餅圖
scatter	散佈圖
hexbin	六角形的 binning 圖形

Plot 方法常用參數如下：

參數	說明
x	X 軸資料
y	Y 軸資料
kind	設定圖表類型
color	點顏色
label	標籤
title	圖表標題
ytitle	Y 軸標題
grid	是否顯示網格
legend	是否顯示圖例
xticks	指定 x 軸標籤的取值範圍
yticks	指定 y 軸標籤的取值範圍
fontsize	字體大小

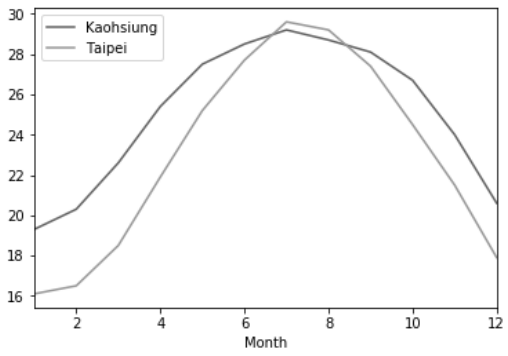
折線圖繪製方法範例如下：

```
#-------------------------------------
#   CH_11_5_EX_1_Pandas_Plot_Line.py
#-------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----讀取資料
df = pd.read_excel("temperture.xlsx")

#-----將結果快速繪出
df.plot(x='Month', y=['Kaohsiung', 'Taipei'])
```

執行結果如下：



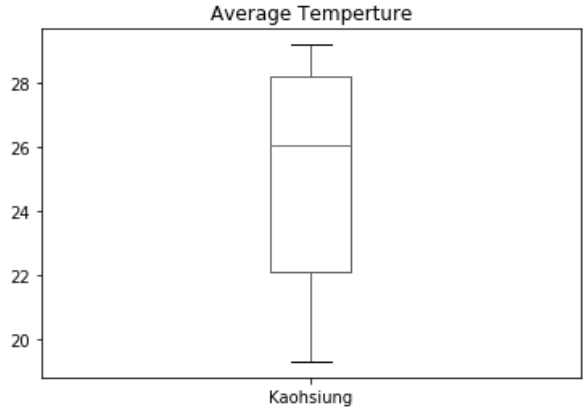
箱形圖繪製方法範例如下：

```
#-------------------------------------
#   CH_11_5_EX_2_Pandas_Plot_Box.py
#-------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----讀取資料
df = pd.read_excel("temperture.xlsx")

#-----繪圖
df.iloc[:,1].plot(kind = 'box', title = 'Average Temperture')
```

執行結果如下：



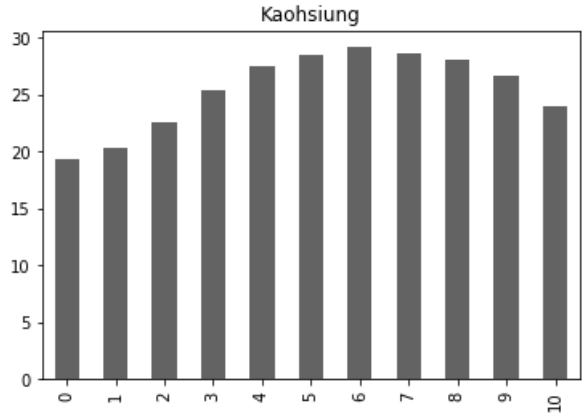
直條圖繪製方法範例如下：

```
#-------------------------------------
#   CH_11_5_EX_3_Pandas_Plot_Bar.py
#-------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----讀取資料
df = pd.read_excel("temperture.xlsx")

#-----繪圖
df.iloc[:,1].plot(kind = 'bar', title = 'Kaohsiung')
```

執行結果如下：



圖表格式化範例如下：

```
#----------------------------------------
#   CH_11_5_EX_4_Pandas_Plot_Format.py
#----------------------------------------
#-----載入 Pandas 套件並命名為 pd
import pandas as pd

#-----讀取資料
df = pd.read_excel("temperture.xlsx")

#-----繪圖
df.iloc[1:11,1].plot(kind = 'bar', title = 'Kaohsiung', grid=True,
legend=True)
```

執行結果如下：