深度學習

第 3 章 資料分析套件

講師：紀俊男

本章大綱

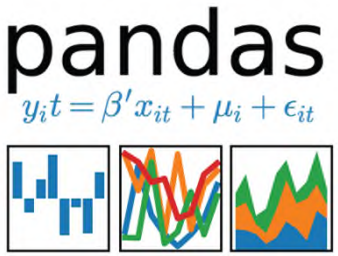
- NumPy 套件介紹
- Pandas 套件介紹
- Matplotlib 套件介紹
- 本章總結

資料分析套件簡介

NumPy

- 將數值組織成陣列
- 執行各式陣列運算

Pandas



- 載入外部檔案的資料
- 轉換成 NumPy 運算

MatPlotLib

- 繪製各類統計圖表

NumPy
套件介紹

範例完整原始碼：
https://ishort.ink/fQSX

NumPy 簡介

- **NumPy 的作用**
  - 處理「陣列」
  - 「陣列」= 相同資料型態元素的集合
  - 是 pandas 等套件的基礎架構

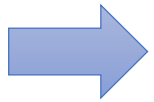
- **如何安裝**
  - Colab 預設已經安裝
  - 亦可用「! pip install numpy」安裝

- **如何引用**
  - import numpy as np

建立陣列

- 建立一維陣列

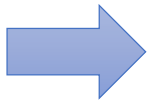
```
1  import numpy as np
2
3  ary1 = np.array([1, 2, 3])
4  print(ary1)
```



```
[1 2 3]
```

- 建立二維陣列

```
1  import numpy as np
2
3  ary2 = np.array([[1, 2, 3], [4, 5, 6]])
4  print(ary2)
```

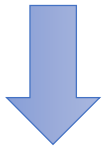


```
[[1 2 3]
 [4 5 6]]
```

建立陣列

- 故意輸入「資料型態不同」的資料

```
1  import numpy as np
2
3  ary3 = np.array([15, "Apple", True])
4  print(ary3)
```



['15' 'Apple' 'True']　通通變成「字串」了

隨堂練習：一維&二維陣列建立

- **請依照下列步驟，準備好 Colab 環境**
  - 開啟 Colab
  - 開一個新的 Colab 檔案，命名為「NumPy.py」

- **請用下列程式碼，試著建立一維&二維陣列，並將它印出：**
  - ary1 = np.array([1, 2, 3])
  - ary2 = np.array([[1, 2, 3], [4, 5, 6]])

- **請故意建立型態不同的串列，去建造一個 NumPy 陣列，並將它印出：**
  - ary3 = np.array([15, "Apple", True])
  - 觀察看看，它的陣列內容，是否還能保持不同的資料型態？

建立特殊陣列

- 建立「零陣列」

```
1  import numpy as np
2
3  zero1 = np.zeros((3,))
4  print(zero1)
5
6  zero2 = np.zeros((2, 3))
7  print(zero2)
```

預設是浮點數

一維零陣列 `[0. 0. 0.]`

二維零陣列
```
[[0. 0. 0.]
 [0. 0. 0.]]
```

- 建立「單位陣列」

```
1  import numpy as np
2
3  identity1 = np.eye(1)
4  print(identity1)
5
6  identity2 = np.eye(2)
7  print(identity2)
```

預設是浮點數

一維單位陣列 `[[1.]]`

二維單位陣列
```
[[1. 0.]
 [0. 1.]]
```

建立特殊陣列

- 建立全為 1 的「常數陣列」

```
1  import numpy as np
2
3  one1 = np.ones((3,))
4  print(one1)
5
6  one2 = np.ones((2, 3))
7  print(one2)
```

預設是浮點數

一維常數陣列

```
[1. 1. 1.]
```

二維常數陣列

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

- 建立全為特定數字的「常數陣列」

```
1  import numpy as np
2
3  const1 = np.full((3,), 7)
4  print(const1)
5
6  const2 = np.full((2,3), 7)
7  print(const2)
```

一維常數陣列

```
[7 7 7]
```

二維常數陣列

```
[[7 7 7]
 [7 7 7]]
```

隨堂練習：零陣列、單位陣列、常數陣列

- **用下列程式碼，建立「零陣列」，並且印出：**
  - zero1 = np.zeros((3,))
  - zero2 = np.zeros((2, 3))

- **用下列程式碼，建立「單位陣列」，並且印出：**
  - identity1 = np.eye(1)
  - identity2 = np.eye(2)

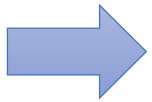
- **用下列程式碼，建立「常數陣列」，並且印出：**
  - 全為 1
    - one1 = np.ones((3,))
    - one2 = np.ones((2, 3))
  - 全為 7
    - const1 = np.full((3,), 7)
    - const2 = np.full((2,3), 7)

讀寫陣列元素

- **讀取**陣列元素

```
1  import numpy as np
2
3  ary1 = np.array([[1, 2, 3], [4, 5, 6]])
4  print(ary1[0, 0], ary1[1, 2])
```

[0, 0]　[1, 2]

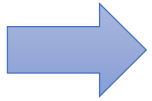


```
1  6
```

讀取 ary1 [0, 0] 與 [1, 2] 處的元素

- **寫入**陣列元素

```
1  import numpy as np
2
3  ary1 = np.array([[1, 2, 3], [4, 5, 6]])
4
5  ary1[0, 0] = 100
6  ary1[1, 2] = 600
7  print(ary1)
```



```
[[100   2   3]
 [  4   5 600]]
```

寫入 ary1 [0, 0] 與 [1, 2] 處的元素

隨堂練習：讀寫陣列元素

- 請先建立下列陣列
  - ary1 = np.array([[1, 2, 3], [4, 5, 6]])

- 用下列方法，讀取陣列 [0, 0] 與 [1, 2] 處的元素，並將之印出
  - print(ary1[0, 0], ary1[1, 2])

- 用下列方法，寫入陣列 [0, 0] 與 [1, 2] 處的元素，並將整個陣列印出
  - ary1[0, 0] = 100
  - ary1[1, 2] = 600
  - print(ary1)

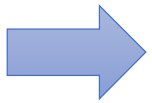
- 參考程式碼如右所示：

```
1  import numpy as np
2
3  ary1 = np.array([[1, 2, 3], [4, 5, 6]])
4  print(ary1[0, 0], ary1[1, 2])
5
6  ary1[0, 0] = 100
7  ary1[1, 2] = 600
8  print(ary1)
```

讀取陣列資訊

- 讀取陣列本身「資料型態（Type）」

```
1  import numpy as np
2
3  ary1 = np.array([[1, 2, 3], [4, 5, 6]])
4  print(type(ary1))
```

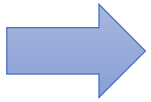


```
<class 'numpy.ndarray'>
```

NumPy 陣列型態為「ndarray」

- 讀取陣列「維度（N-Dimension）」

```
1  import numpy as np
2
3  ary1 = np.array([[1, 2, 3], [4, 5, 6]])
4  print(ary1.ndim)
```



```
2
```

ary1 的維度是「2 維」

讀取陣列資訊

- 讀取陣列本身「行列數（Shape）」

```
1  import numpy as np
2
3  ary1 = np.array([[1, 2, 3], [4, 5, 6]])
4  print(ary1.shape)
```

(2, 3)

ary1 的行列數是「2 x 3」

- 讀取陣列內容值「資料型態（Data Type）」

```
1  import numpy as np
2
3  ary1 = np.array([[1, 2, 3], [4, 5, 6]])
4  print(ary1.dtype)
```

int32

ary1 內容值的資料型態是「int32」

隨堂練習：讀取陣列資訊

- **請先建立下列陣列**
  - ary1 = np.**array**([[1, 2, 3], [4, 5, 6]])

- **用下列指令，取得 ary1 的各種資訊，並將之印出**
  - 陣列本身資料型態：**type(**ary1**)**
  - 陣列維度：ary1.**ndim**
  - 陣列行列數：ary1.**shape**
  - 陣列內容值資料型態：ary1.**dtype**

- **參考程式碼如下所示：**

```
1  import numpy as np
2
3  ary1 = np.array([[1, 2, 3], [4, 5, 6]])
4  print(type(ary1), ary1.ndim, ary1.shape, ary1.dtype)
```

產生「樣本點」

- 產生「線性」樣本點

```
1  import numpy as np
2
3  sample1 = np.arange(0., 5., 0.2)
4  print(sample1)
```

arange = Array RANGE = [0.0, 5.0) 浮點數序列

```
[0.  0.2 0.4 0.6 0.8
 1.  1.2 1.4 1.6 1.8
 2.  2.2 2.4 2.6 2.8
 3.  3.2 3.4 3.6 3.8
 4.  4.2 4.4 4.6 4.8]
```

- 產生「線性」樣本點 + 洗牌

```
1  import numpy as np
2
3  sample1 = np.arange(0., 5., 0.2)
4  np.random.shuffle(sample1)
5  print(sample1)
```

```
[2.6 3.6 4.4 4.6 1.6
 3.4 2.4 2.2 2.8 4.8
 1.  0.6 4.2 0.2 1.2
 1.4 0.8 3.  0.  0.4
 3.8 2.  4.  3.2 1.8]
```

產生「樣本點」

- 產生「線性」樣本點 + 更改維度

```
1  import numpy as np
2
3  sample1 = np.arange(0., 5., 0.2)
4  sample1 = sample1.reshape(5, 5)
5  print(sample1)
```

```
[[0.  0.2 0.4 0.6 0.8]
 [1.  1.2 1.4 1.6 1.8]
 [2.  2.2 2.4 2.6 2.8]
 [3.  3.2 3.4 3.6 3.8]
 [4.  4.2 4.4 4.6 4.8]]
```

- 產生「線性」樣本點 + 更改資料型態

```
1  import numpy as np
2
3  sample1 = np.arange(0., 5., 0.2)
4  sample1 = sample1.astype("unicode")
5  print(sample1)
```

```
['0.0' '0.2' '0.4' '0.6000000000000001' '0.8'
 '1.0' '1.2000000000000002' '1.4000000000000001' '1.6' '1.8'
 '2.0' '2.2' '2.4000000000000004' '2.6' '2.8000000000000003'
 '3.0' '3.2' '3.4000000000000004' '3.6' '3.8000000000000003'
 '4.0' '4.2' '4.4' '4.6000000000000005' '4.800000000000001']
```

隨堂練習：產生「線性」樣本點

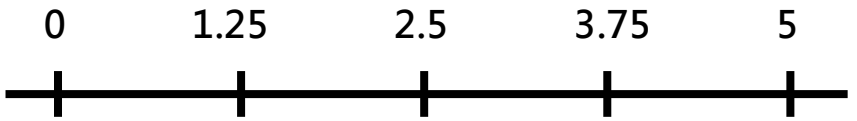
- 請先用 arange() 建立「線性樣本點」陣列，並將它印出
  - sample1 = np.arange(0., 5., 0.2)
- 用下列程式碼「弄亂」它，並將結果印出
  - np.random.shuffle(sample1)
- 用下列程式碼，將維度從 1x25 改成 5x5，並將之印出
  - sample1 = sample1.reshape(5, 5)
- 用下列程式碼，將元素型態改為「文字」，並將之印出
  - sample1 = sample1.astype("unicode")
- 參考程式碼如右所示：

```
1  import numpy as np
2
3  sample1 = np.arange(0., 5., 0.2)
4  print(sample1)
5
6  np.random.shuffle(sample1)
7  print(sample1)
8
9  sample1 = sample1.reshape(5, 5)
10 print(sample1)
11
12 sample1 = sample1.astype("unicode")
13 print(sample1)
```

產生「等差/等比樣本點」

- 產生「等差」樣本點

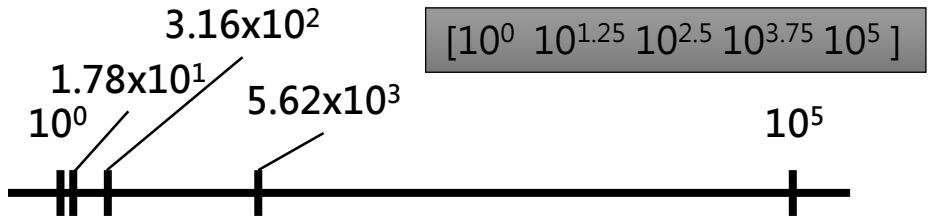
```
1   import numpy as np
2
3   sample1 = np.linspace(0., 5., 5)
4   print(sample1)
```



[0.0 1.25 2.5 3.75 5.0]

- 產生「等比」樣本點

```
1   import numpy as np
2
3   sample1 = np.logspace(0., 5., 5)
4   print(sample1)
```



[10^0 $10^{1.25}$ $10^{2.5}$ $10^{3.75}$ 10^5]

[1.00000000e+00 1.77827941e+01 3.16227766e+02
5.62341325e+03 1.00000000e+05]

隨堂練習：產生「等差/等比樣本點」

- 請用下列這幾道指令，建立「等差」的樣本點：

```
sample1 = np.linspace(0., 5., 5)
print(sample1)
```

- 再用下列這幾道指令，建立「等比」的樣本點：

```
sample1 = np.logspace(0., 5., 5)
print(sample1)
```

產生「亂數」樣本點

- 產生整數「亂數」樣本點

```
1  import numpy as np
2
3  sample2 = np.random.randint(1, 7, size=15)
4  print(sample2)
```

產生 [1, 7) 之間的 15 個樣本點 = 模擬擲骰子 15 次

[4 6 4 6 5 1 6 5 4 1 2 1 5 5 3]

- 產生浮點數「亂數」樣本點

```
1  import numpy as np
2
3  sample3 = np.random.rand(2, 3)
4  print(sample3)
```

產生 [0, 1) 之間的浮點亂數 2x3 個
(其它範圍亂數：以 r [0, 1) + b 製作之)

[[0.29940871 0.22833058 0.09572088]
 [0.22495211 0.15460668 0.47581877]]

隨堂練習：產生「亂數」樣本點

- 請用下列方法，模擬出擲 15 次骰子的結果，並將之印出
  - sample2 = np.random.randint(1, 7, size=15)
- 請用下列方法，製造出 2x3 個 0~1 之間的浮點數，並將之印出
  - sample3 = np.random.rand(2, 3)
- 參考程式碼如下所示

```
1  import numpy as np
2
3  sample2 = np.random.randint(1, 7, size=15)
4  print(sample2)
5
6  sample3 = np.random.rand(2, 3)
7  print(sample3)
```

產生「常態分佈」樣本點

- 產生「標準常態分佈」（平均值 = 0，標準差 = 1）

```
1  import numpy as np
2
3  normal1 = np.random.randn(3, 5)
4  print(normal1)
```

```
[[ 0.07383453  1.56612877 -1.72056335  0.79062657 -0.98145664]
 [ 0.75696738  0.03326342 -1.05400173  0.41382954  0.14313756]
 [ 0.03675955 -0.4943983  -0.01105739  0.64156388  0.01929711]]
```

產生 3x5 個符合「標準常態分佈」的樣本點

- 產生「一般常態分佈」

```
1  import numpy as np
2
3  normal2 = np.random.normal(10, 2, size=(3, 5))
4  print(normal2)
```

```
[[ 4.27554693 10.82599695 11.48964224 12.21061391 11.22502566]
 [11.93800476 11.45222262 11.23980293  9.62609497  9.04431152]
 [ 9.14795709 10.61995594  7.42692941  8.22977962 12.83596237]]
```

產生 3x5 個符合「平均值 = 10, 標準差 =2」的樣本點

隨堂練習：產生「常態分佈」樣本點

- 請輸入下列程式碼，產生 3x5 個「標準常態分佈」樣本點
  - normal1 = np.random.**randn**(3, 5)

- 請輸入下列程式碼，產生 3x5 個「一般常態分佈」樣本點
  - normal2 = np.random.**normal**(10, 2, size=(3, 5))

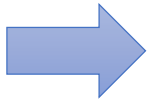
- 參考程式碼如下所示：

```
1  import numpy as np
2
3  normal1 = np.random.randn(3, 5)
4  print(normal1)
5
6  normal2 = np.random.normal(10, 2, size=(3, 5))
7  print(normal2)
```

切片運算（Slicing）

- 一維陣列切片運算

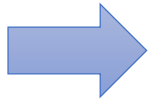
```
1  import numpy as np
2
3  slice1 = np.random.randint(20, size=20)
4  print(slice1)
5  print(slice1[:5])
```



```
[ 5 14 17 18 14  0  3  6 12  1 14  3  6 14 17  9 11  0  3 18]
```

- 二維陣列切片運算

```
1  import numpy as np
2
3  slice2 = np.random.randint(20, size=(5, 5))
4  print(slice2)
5  print(slice2[:3, :3])
```



```
[[ 9 15  3 13 15]
 [ 7 11 10  3  0]
 [14 13 17  3 10]
 [ 9  2 13 18 17]
 [14 10  7  6 15]]
```

隨堂練習：切片運算

- 請依照前一頁投影片的內容，練習一維、二維陣列的切片運算。

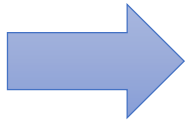
- 參考程式碼如下所示：

```
1  import numpy as np
2
3  slice1 = np.random.randint(20, size=20)
4  print(slice1)
5  print(slice1[:5])
6
7  slice2 = np.random.randint(20, size=(5, 5))
8  print(slice2)
9  print(slice2[:3, :3])
```

統計量的計算

```
1  import numpy as np
2
3  stat1 = np.arange(1, 11)
4  print(stat1)
5
6  print(stat1.min())
7  print(stat1.max())
8  print(stat1.sum())
9  print(stat1.mean())
10 print(stat1.std())
```

1
2
3
4
5
6



```
[1 2 3 4 5 6 7 8 9 10]
1
10
55
5.5
2.8722813232690143
```

1. 原始資料
2. 取陣列中最小元素
3. 取陣列中最大元素
4. 取陣列元素總和
5. 取陣列元素平均
6. 取陣列元素標準差

隨堂練習：統計量的計算

- 請先建造下列陣列，並將之印出：
  - stat1 = np.arange(1, 11)

- 請用下列函數，練習計算 stat1 內的各種統計量：
  - 最小元素：stat1.min()
  - 最大元素： stat1.max()
  - 總和：stat1.sum()
  - 平均值：stat1.mean()
  - 標準差：stat1.std()

- 參考程式碼如右圖所示：

```
1  import numpy as np
2
3  stat1 = np.arange(1, 11)
4  print(stat1)
5
6  print(stat1.min())
7  print(stat1.max())
8  print(stat1.sum())
9  print(stat1.mean())
10 print(stat1.std())
```

陣列運算：轉置（Transpose）

```
1  import numpy as np
2
3  X1 = np.array([1, 2, 3])
4  X2 = np.array([20, 36, 40])
5  features = np.concatenate((X1, X2)).reshape(2, 3).T
6  print(features)
```

features = np.concatenate((X1, X2)).reshape(2, 3).T

$([1 \quad 2 \quad 3] \quad [20 \quad 36 \quad 40])$

一定要用 Tuple 包住再丟入

$\begin{bmatrix} 1 & 2 & 3 \\ 20 & 36 & 40 \end{bmatrix}$ $\begin{bmatrix} 1 & 20 \\ 2 & 36 \\ 3 & 40 \end{bmatrix}$

隨堂練習：轉置（Transpose）

- 請先定義如下的陣列：
  - X1 = np.array([1, 2, 3])
  - X2 = np.array([20, 36, 40])

- 用下列指令，練習陣列「矩陣轉置」運算：
  - 陣列接合： .concatenate((X1, X2))
  - 維度變更：.reshape(2, 3)
  - 矩陣轉置：.T

- 參考程式碼如下圖所示：

```
1  import numpy as np
2
3  X1 = np.array([1, 2, 3])
4  X2 = np.array([20, 36, 40])
5  features = np.concatenate((X1, X2)).reshape(2, 3).T
6  print(features)
```

Pandas
套件介紹

範例完整原始碼：
https://ishort.ink/8FVv

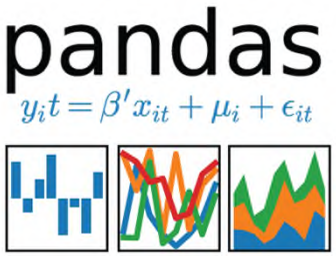
Pandas 簡介

- **Pandas 的作用**
  - 讀入「外部資料」，並以「表格」形式呈現
  - 讀入後，以下列資料型態表示：
    - 一維：Series
    - 二維：DataFrame
    - 三維：Panel

資料科學裡最常用！
本課程只介紹此類！

- **如何安裝**
  - Colab 預設已經安裝
  - 亦可用「! pip install pandas」安裝

- **如何引用**
  - import pandas as pd



建立 DataFrame 資料集

透過「複合資料結構」建立

```python
import pandas as pd

Rows = ["Row1", "Row2", "Row3"]
Columns = ["Column1", "Column2"]

# by list (Row-based)
dfList = pd.DataFrame([[1, 2], [3, 4], [5, 6]], index=Rows, columns=Columns)
print(dfList)

# by Dict (Column-based)
dfDict = pd.DataFrame({"Column1":[1, 3, 5], "Column2":[2, 4, 6]}, index=Rows)
print(dfDict)
```

index=指定列名 （預設：[0, 1, 2...]）　columns=指定欄名（預設：[0, 1, 2...]）

以列為主

	Column1	Column2
Row1	1	2
Row2	3	4
Row3	5	6

以欄為主

	Column1	Column2
Row1	1	2
Row2	3	4
Row3	5	6

隨堂練習：用「複合資料」建立 DataFrame

- **請先建立下列欄名、列名：**
  - Rows = ["Row1", "Row2", "Row3"]
  - Columns = ["Column1", "Column2"]
- **用串列建立一個 DataFrame，並將之印出**
  - dfList = pd.DataFrame([[1, 2], [3, 4], [5, 6]], index=Rows, columns=Columns)
- **用字典建立一個 DataFrame，並將之印出**
  - dfDict = pd.DataFrame({"Column1":[1, 3, 5], "Column2":[2, 4, 6]}, index=Rows)
- **參考原始碼如下所示：**

```
1  import pandas as pd
2
3  Rows = ["Row1", "Row2", "Row3"]
4  Columns = ["Column1", "Column2"]
5
6  # by list (Row-based)
7  dfList = pd.DataFrame([[1, 2], [3, 4], [5, 6]], index=Rows, columns=Columns)
8  print(dfList)
9
10 # by Dict (Column-based)
11 dfDict = pd.DataFrame({"Column1":[1, 3, 5], "Column2":[2, 4, 6]}, index=Rows)
12 print(dfDict)
```

建立 DataFrame 資料集

- 透過「CSV 檔」建立

	A	B	C	D
1	Country	Age	Salary	ToBuy
2	France	44	72000	No
3	Spain	27	48000	Yes
4	Germany	30	54000	No
5	Spain	38	61000	No
6	Germany	40		Yes
7	France	35	58000	Yes
8	Spain		52000	No
9	France	48	79000	Yes
10	Germany	50	83000	No
11	France	37	67000	Yes

CarSales.csv

```
1  import pandas as pd
2
3  dfCSV = pd.read_csv("CarSales.csv")
4  print(dfCSV)
5
6  print(dfCSV["Country"].mode())
7  print(dfCSV["Age"].median())
8  print(dfCSV["Salary"].mean())
```

```
   Country   Age   Salary ToBuy
0   France  44.0  72000.0    No
1    Spain  27.0  48000.0   Yes
2  Germany  30.0  54000.0    No
3    Spain  38.0  61000.0    No
4  Germany  40.0      NaN   Yes
5   France  35.0  58000.0   Yes
6    Spain   NaN  52000.0    No
7   France  48.0  79000.0   Yes
8  Germany  50.0  83000.0    No
9   France  37.0  67000.0   Yes
```

- NaN = Not a Number
- DataFrame 用來表示**缺失資料**的記號

- dfCSV[**“欄位名稱”**]：取出特定欄位的方法
- 針對特定欄位，計算**統計量**的方法
  - .mode()：取「眾數」
  - .median()：取「中位數」
  - .mean()：取「平均」

隨堂練習：用「CSV」建立 DataFrame

- 請依照講師的指示，將 CarSales.csv 檔案，上傳至 Colab。
- 上傳完畢後，請撰寫下列程式碼：

```
1  import pandas as pd
2
3  dfCSV = pd.read_csv("CarSales.csv")
4  print(dfCSV)
5
6  print(dfCSV["Country"].mode())
7  print(dfCSV["Age"].median())
8  print(dfCSV["Salary"].mean())
```

- 您已經學會下列方法了嗎？
  - 讀入 CSV
  - 取欄位眾數
  - 取欄位中位數
  - 取欄位平均值
- NumPy 的所有統計量公式，如：.max(), .min()...都可以使用。

取得資料集的「摘要描述」

- <Pandas資料集>.describe(include=“all”)
  - 可以列出欄位的「眾數、平均、標準差、極小/極大值、Q25/Q50/Q75」。
  - 能夠馬上對整個資料集的每一欄資料之樣貌，有個大概的掌握。

```
1  import pandas as pd
2
3  dfCSV = pd.read_csv("CarSales.csv")
4  dfCSV.describe(include="all")
```

	Country	Age	Salary	ToBuy
count	10	9.000000	9.000000	10
unique	3	NaN	NaN	2
top	France	NaN	NaN	No
freq	4	NaN	NaN	5
mean	NaN	38.777778	63777.777778	NaN
std	NaN	7.693793	12265.579662	NaN
min	NaN	27.000000	48000.000000	NaN
25%	NaN	35.000000	54000.000000	NaN
50%	NaN	38.000000	61000.000000	NaN
75%	NaN	44.000000	72000.000000	NaN
max	NaN	50.000000	83000.000000	NaN

NaN = Not a Number
- 相當於「空值」、「不適用」之意。

隨堂練習：取得資料集的「摘要描述」

- 請先撰寫下列程式碼，並執行看看：
  - dfCSV.describe(include= “all”)

- 你可以取得所有欄位的「摘要描述」嗎？

	Country	Age	Salary	ToBuy
count	10	9.000000	9.000000	10
unique	3	NaN	NaN	2
top	France	NaN	NaN	No
freq	4	NaN	NaN	5
mean	NaN	38.777778	63777.777778	NaN
std	NaN	7.693793	12265.579662	NaN
min	NaN	27.000000	48000.000000	NaN
25%	NaN	35.000000	54000.000000	NaN
50%	NaN	38.000000	61000.000000	NaN
75%	NaN	44.000000	72000.000000	NaN
max	NaN	50.000000	83000.000000	NaN

選取特定欄列

- 使用 .iloc[ ] 與 .loc[ ] 函數
  - .iloc[ ]：Index Locator 之意。使用「索引值」來指定特定欄列。
  - .loc[ ]：Locator 之意。使用「欄位名稱」來指定特定欄列。

dfCSV.iloc[: , 1:3]

[1] [2]

	Age	Salary
0	44.0	72000.0
1	27.0	48000.0
2	30.0	54000.0
3	38.0	61000.0
4	40.0	NaN
5	35.0	58000.0
6	NaN	52000.0
7	48.0	79000.0
8	50.0	83000.0
9	37.0	67000.0

索引值： [0] [1] [2] [3]

	Country	Age	Salary	ToBuy
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

dfCSV.loc[: , ["Country", "ToBuy"]]

	Country	ToBuy
0	France	No
1	Spain	Yes
2	Germany	No
3	Spain	No
4	Germany	Yes
5	France	Yes
6	Spain	No
7	France	Yes
8	Germany	No
9	France	Yes

隨堂練習：使用 iloc[ ] 與 loc[ ]

- 請先撰寫下列程式碼，並執行看看：

```
# 用 .iloc() 選取年齡與月薪兩個欄位（列數則全取）
print(dfCSV.iloc[:, 1:3])

# 用 .loc() 選取國別與是否購買兩個欄位（列數則全取）
print(dfCSV.loc[:, ["Country", "ToBuy"]])
```

- 你可以透過 iloc[ ] 或 loc[ ]，取得指定的欄位嗎？

	Age	Salary
0	44.0	72000.0
1	27.0	48000.0
2	30.0	54000.0
3	38.0	61000.0
4	40.0	NaN
5	35.0	58000.0
6	NaN	52000.0
7	48.0	79000.0
8	50.0	83000.0
9	37.0	67000.0

	Country	ToBuy
0	France	No
1	Spain	Yes
2	Germany	No
3	Spain	No
4	Germany	Yes
5	France	Yes
6	Spain	No
7	France	Yes
8	Germany	No
9	France	Yes

選取特定欄列

- 使用「條件」來選取
  - 先將「條件」儲存成一個變數。
  - 再將「條件」，塞入 dfCSV[ ] 的**中括號**裡面，過濾特定欄列。

索引值：　[0]　[1]　[2]　[3]

	Country	Age	Salary	ToBuy
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
# 條件=最後沒有買車的客戶
the_condition = (dfCSV["ToBuy"] == "No")

# 塞入 dfCSV[ ] 的中括號裡面
print(dfCSV[the_condition])
```

	Country	Age	Salary	ToBuy
0	France	44.0	72000.0	No
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
6	Spain	NaN	52000.0	No
8	Germany	50.0	83000.0	No

都是沒有買車的客戶

隨堂練習：使用「條件」指定欄列

- 請先撰寫下列程式碼，並執行看看：

```
1  # 選出最後沒有購買汽車的客戶
2  the_condition = (dfCSV["ToBuy"] == "No")
3  print(dfCSV[the_condition])
```

- 你可以透過「條件」，篩選出指定的欄位嗎？

	Country	Age	Salary	ToBuy
0	France	44.0	72000.0	No
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
6	Spain	NaN	52000.0	No
8	Germany	50.0	83000.0	No

轉換成 NumPy 陣列

```
1  import pandas as pd
2
3  ary = dfCSV.to_numpy()
4  print(ary)
```

法一

.values

法二

```
[['France' 44.0 72000.0 'No']
 ['Spain' 27.0 48000.0 'Yes']
 ['Germany' 30.0 54000.0 'No']
 ['Spain' 38.0 61000.0 'No']
 ['Germany' 40.0 nan 'Yes']
 ['France' 35.0 58000.0 'Yes']
 ['Spain' nan 52000.0 'No']
 ['France' 48.0 79000.0 'Yes']
 ['Germany' 50.0 83000.0 'No']
 ['France' 37.0 67000.0 'Yes']]
```

隨堂練習：將 DataFrame 轉為 NumPy 陣列

- 請用 .to_numpy()、或者是 .values 指令，將 DataFrame 轉成 NumPy 陣列後印出：
  - ary = dfCSV.to_numpy()
  - ary = dfCSV.values

- 參考原始碼如下所示：

```
1  import pandas as pd
2
3  ary = dfCSV.to_numpy()
4  print(ary)
```

Matplotlib
套件介紹

範例完整原始碼：
https://ishort.ink/6NJb

Matplotlib 套件下的 PyPlot 簡介

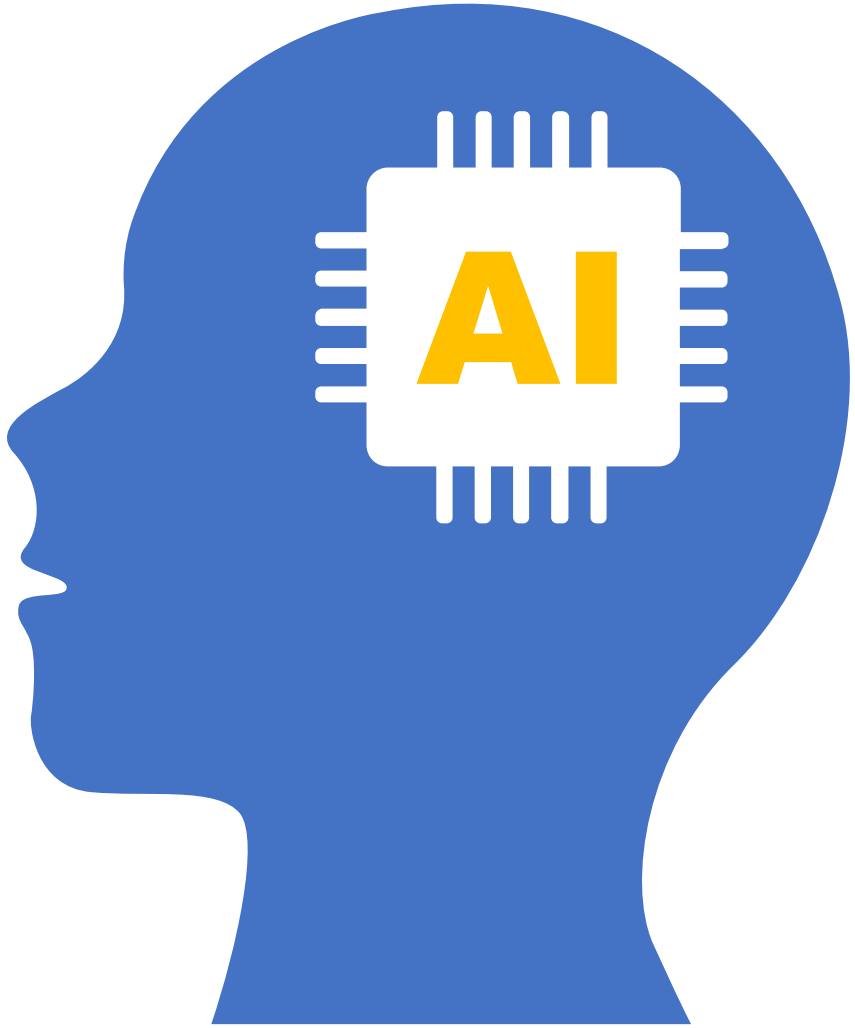
- **Matplotlib.pyplot 的作用**
  - 繪製各種「**統計圖表**」
  - 將資料「**視覺化**」

- **如何安裝**
  - Colab 預設已經安裝
  - 亦可用「! pip install matplotlib」安裝

- **如何引用**
  - import matplotlib.pyplot as plt

折線圖（Line Charts）

- 繪製指令：plot()
- 指定繪製格式
- 繪製多組資料

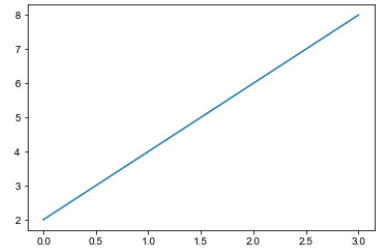


繪製指令：plot()

- 語法
  - plt.plot([X軸資料], Y軸資料, [線段格式] ...)

- 只有 Y 軸資料

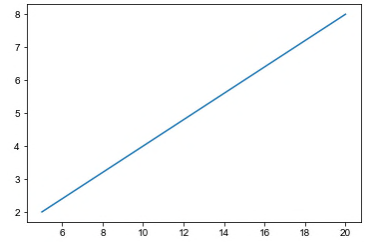
```
1  import matplotlib.pyplot as plt
2
3  plt.plot([2, 4, 6, 8])
4  plt.show()
```



X 軸預設：
[0, 1, 2, 3...]

- 有 X 軸 & Y 軸資料

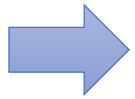
```
1  import matplotlib.pyplot as plt
2
3  plt.plot([5, 10, 15, 20], [2, 4, 6, 8])
4  plt.show()
```

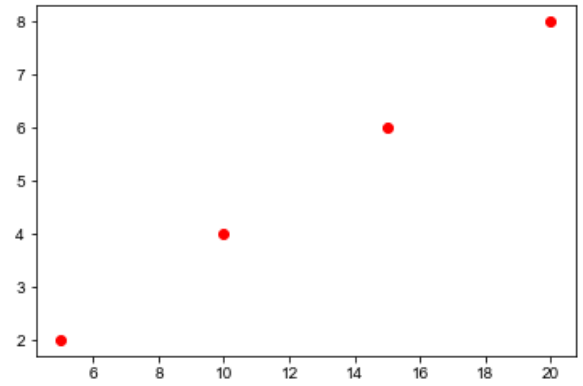


指定繪製格式

- 有 X 軸 、 Y 軸、以及線段格式資料

```
1  import matplotlib.pyplot as plt
2
3  plt.plot([5, 10, 15, 20], [2, 4, 6, 8], "ro")
4  plt.show()
```





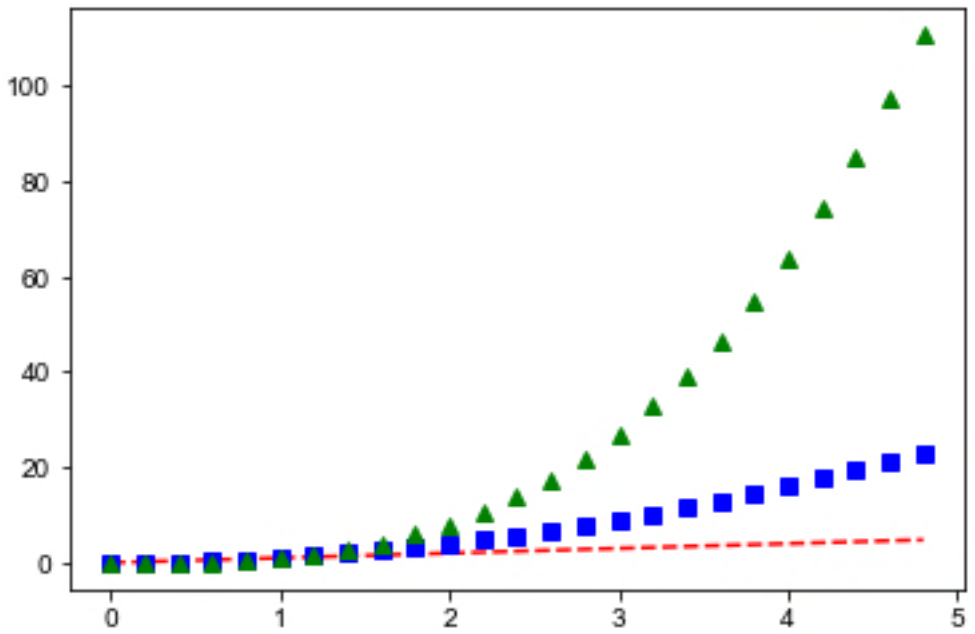
代號	顏色
“r”	紅色（Red）
“g”	綠色（Green）
“b”	藍色（Blue）
“y”	黃色（Yellow）
“c”	青色（Cyan）
“m”	洋紅色（Magenta）
“k”	黑色（Black）
“w”	白色（White）

代號	線段格式
“o” / “s”	圓點 / 方形
“^”, “V”, “<“, “>”	三角形（上下左右）
“p”	五角形（Pentagon）
“*”	星形
“x”	X 形
“D”	菱形（Diamond）
“-”, “--”, “-.”, “.”	實線、Dash-Dash 虛線, Dash-Dot 虛線, Dot 虛線

繪製多組資料

- 有多組資料

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  X = np.arange(0., 5., 0.2)
5  plt.plot(X, X, "r--")
6  plt.plot(X, X**2, "bs")
7  plt.plot(X, X**3, "g^")
8  plt.show()
```



隨堂練習：繪製折線圖

- 請先引入下列兩個套件
  - import matplotlib.pyplot as plt
  - import numpy as np

- 請用下列方法，製作 X 軸資料
  - X = np.*arange*(0., 5., 0.2)

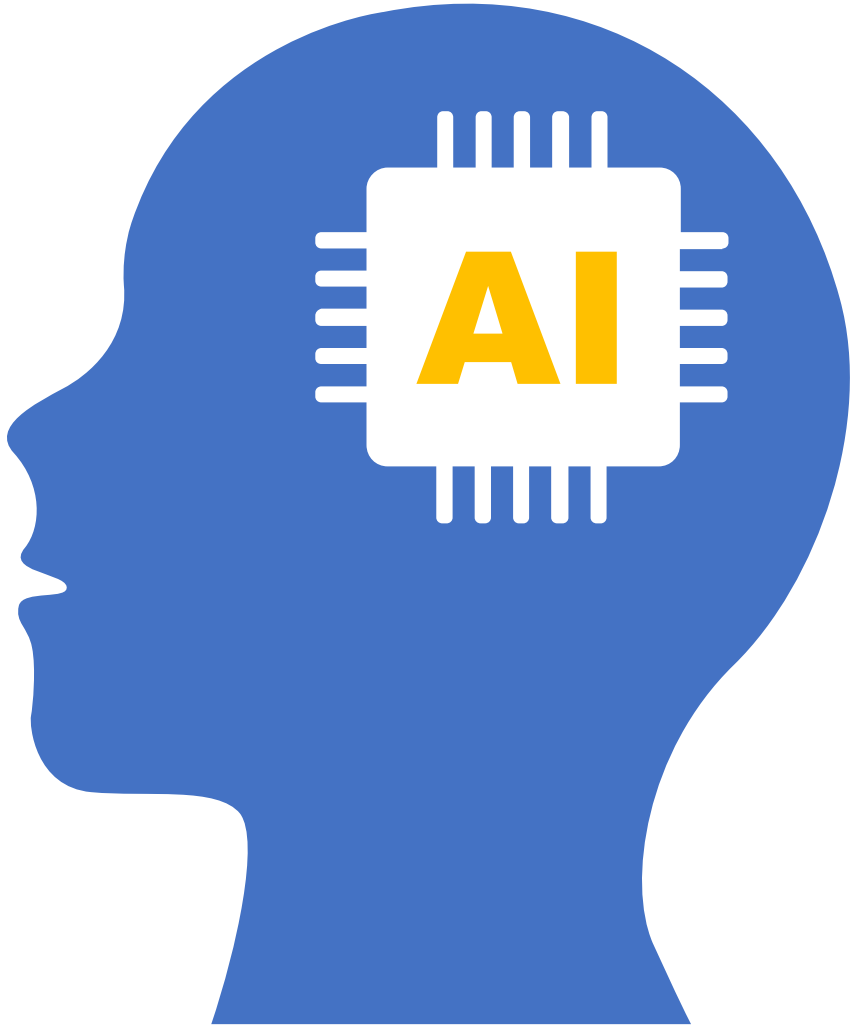
- 請分別繪製 X, X2, X3 等三組 Y 軸資料，於同一圖形上
  - plt.**plot**(X, X, "r--")
  - plt.**plot**(X, X**2, "bs")
  - plt.**plot**(X, X**3, "g^")

- 參考程式碼如右圖所示：

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  X = np.arange(0., 5., 0.2)
5  plt.plot(X, X, "r--")
6  plt.plot(X, X**2, "bs")
7  plt.plot(X, X**3, "g^")
8  plt.show()
```

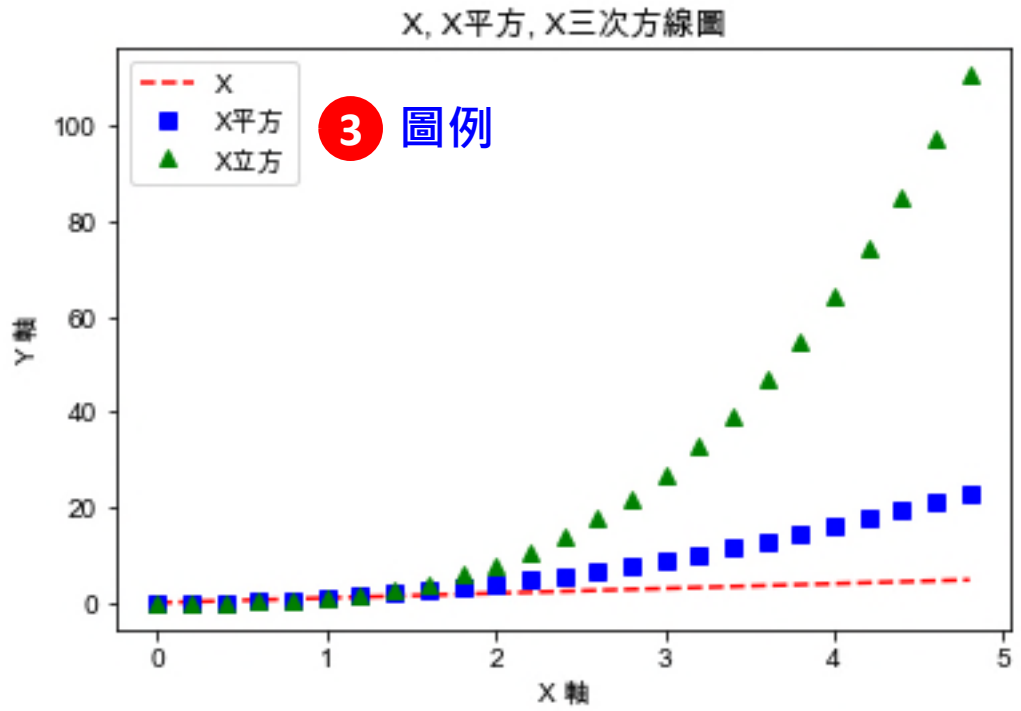
圖形標題、軸線標籤、圖例

- 相關指令介紹
- 「中文亂碼」成因
- 「中文亂碼」解決方法



相關指令介紹

1 圖形標題



3 圖例

2 軸線標籤

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  X = np.arange(0., 5., 0.2)
5
6  plt.title("X, X平方, X三次方線圖")
7  plt.xlabel("X 軸")
8  plt.ylabel("Y 軸")
9
10 plt.plot(X, X, "r--", label="X")
11 plt.plot(X, X**2, "bs", label="X平方")
12 plt.plot(X, X**3, "g^", label="X立方")
13 plt.legend(loc='best')
14
15 plt.show()
```

1 2 3

2 軸線標籤

圖例位置（loc=）選擇：

- “best”, “center”, “upper”, “lower” …

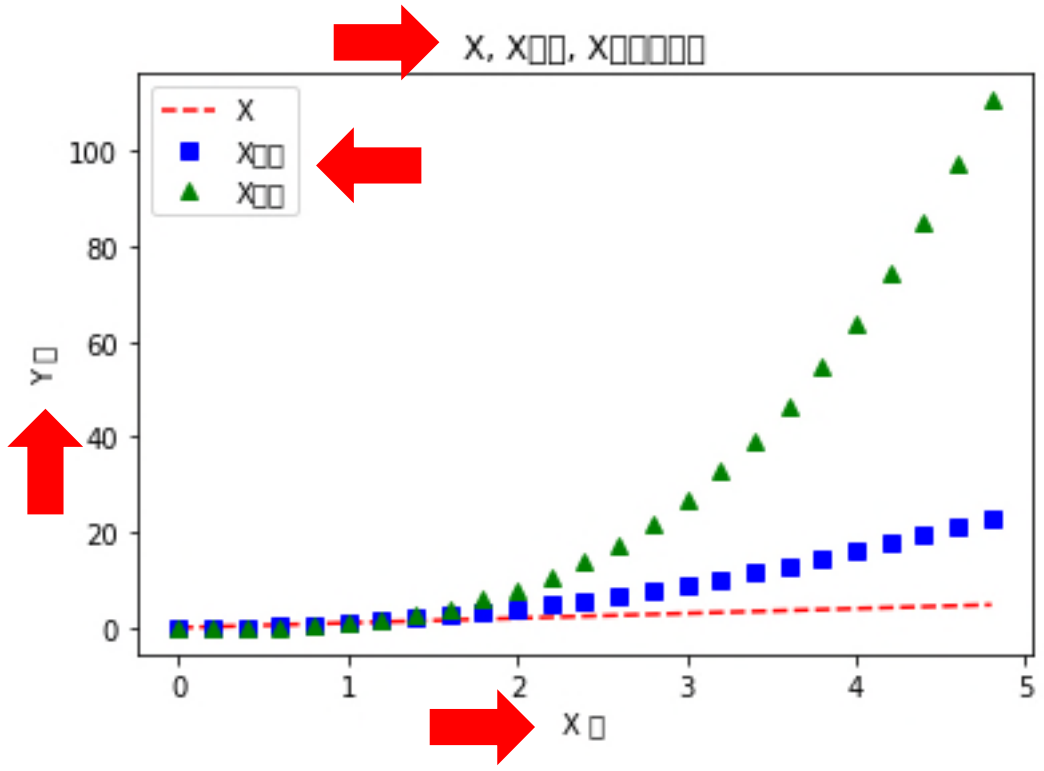
隨堂練習：顯示「圖形標題、軸線標籤、圖例」

- 請先引入下列兩個外掛套件
  - import matplotlib.pyplot as plt
  - import numpy as np
- 撰寫右方程式碼，繪製出相關圖示：

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  X = np.arange(0., 5., 0.2)
5
6  plt.title("X, X平方, X三次方線圖")
7  plt.xlabel("X 軸")
8  plt.ylabel("Y 軸")
9
10 plt.plot(X, X, "r--", label="X")
11 plt.plot(X, X**2, "bs", label="X平方")
12 plt.plot(X, X**3, "g^", label="X立方")
13 plt.legend(loc='best')
14
15 plt.show()
```

- 中文亂碼問題，將在下一個小節解決。

「中文亂碼」成因



成因：

- MatPlotLib 預設**沒有中文字型**

「中文亂碼」解決方法

- **解決步驟一**
  - 下載中文字形，並安裝至 Colab 正確位置

```
1  # 先檢查 Colab 目前 Python 版本（至小數點下兩位），得知將來字形檔安裝路徑
2  !python --version
3
4  # 下載台北思源黑體，並命名taipei_sans_tc_beta.ttf
5  !wget -O taipei_sans_tc_beta.ttf https://drive.google.com/uc?id=1eGAsTN1HBpJAkeVM57_C7ccp7hbgSz3_&export=download
6
7  # 移至指定路徑（若 Python 版本並非 3.10，則需配合更改）
8  !mv taipei_sans_tc_beta.ttf /usr/local/lib/python3.10/dist-packages/matplotlib/mpl-data/fonts/ttf
9
10 # 自定義字體變數（若 Python 版本並非 3.10，則需配合更改）
11 from matplotlib.font_manager import FontProperties
12 myfont = FontProperties(fname=r'/usr/local/lib/python3.10/dist-packages/matplotlib/mpl-data/fonts/ttf/taipei_sans_tc_beta.ttf')
13
14 # 後續在相關函式中，使用 myfont 做為指定字形即可。
15 # 如：plt.xlabel("時間", fontproperties=myfont)
16 # 如：plt.legend(loc='best', prop=myfont)
```

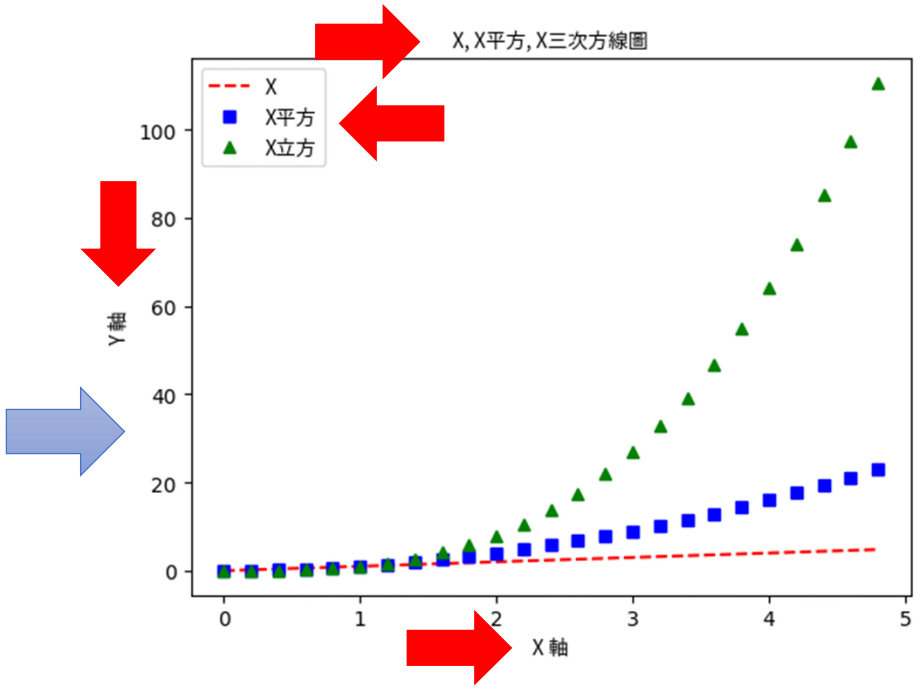
① ② ③ ④ ⑤

← 字形必須安裝至 /usr/local/lib/<Python版本>/dist-packages/matplotlib/mpl-data/fonts/ttf/ 之下

「中文亂碼」解決方法

- **解決步驟二**
  - 修改圖形繪製原始碼，指定使用 myfont 這個中文字形

```
1  # X 軸資料製作
2  X = np.arange(0., 5., 0.2)
3
4  # 設定圖形標題
5  plt.title("X, X平方, X三次方線圖", fontproperties=myfont)
6
7  # 設定軸線標籤
8  plt.xlabel("X 軸", fontproperties=myfont)
9  plt.ylabel("Y 軸", fontproperties=myfont)
10
11 # 繪製三組折線圖並設定圖例
12 plt.plot(X, X, "r--", label="X")
13 plt.plot(X, X**2, "bs", label="X平方")
14 plt.plot(X, X**3, "g^", label="X立方")
15 plt.legend(loc='best', prop=myfont)
16
17 plt.show()
```



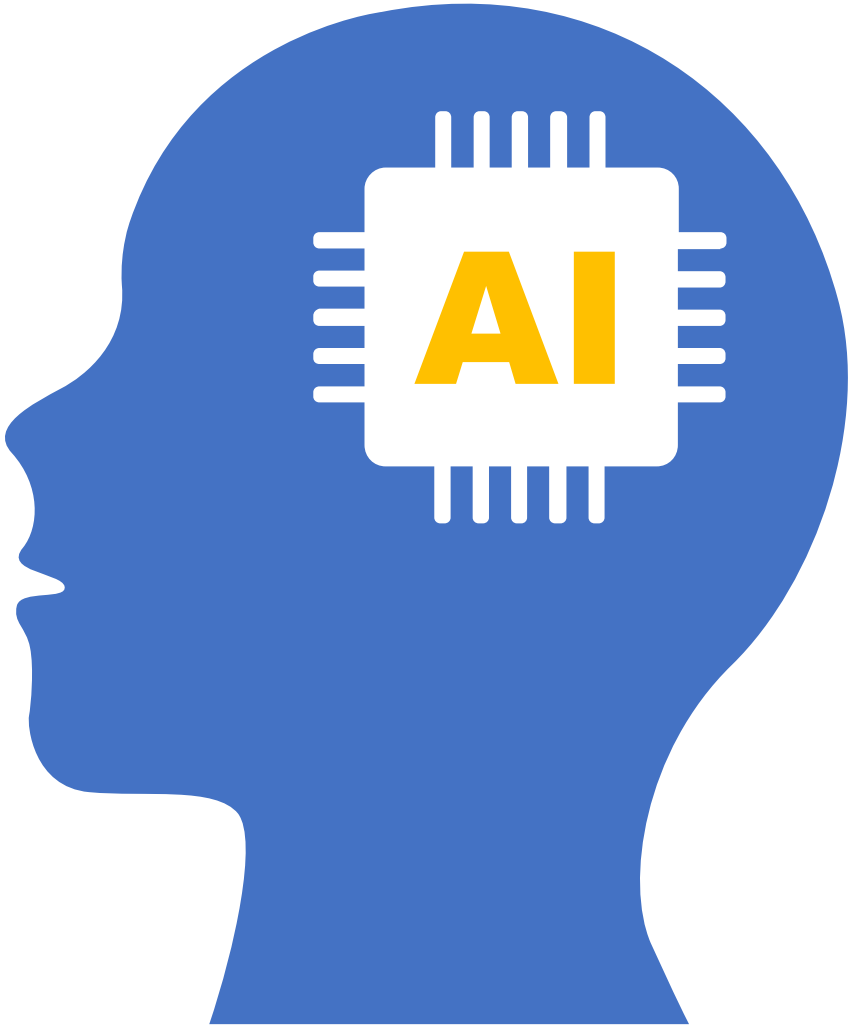
隨堂練習：解決「中文亂碼」問題

- 請依照前兩張投影片的說明，下載、並設定中文字形（可拷貝貼上正確答案之原始碼）。
- 接著依照前一張投影片的說明，修改原始碼，指定系統使用中文字形。

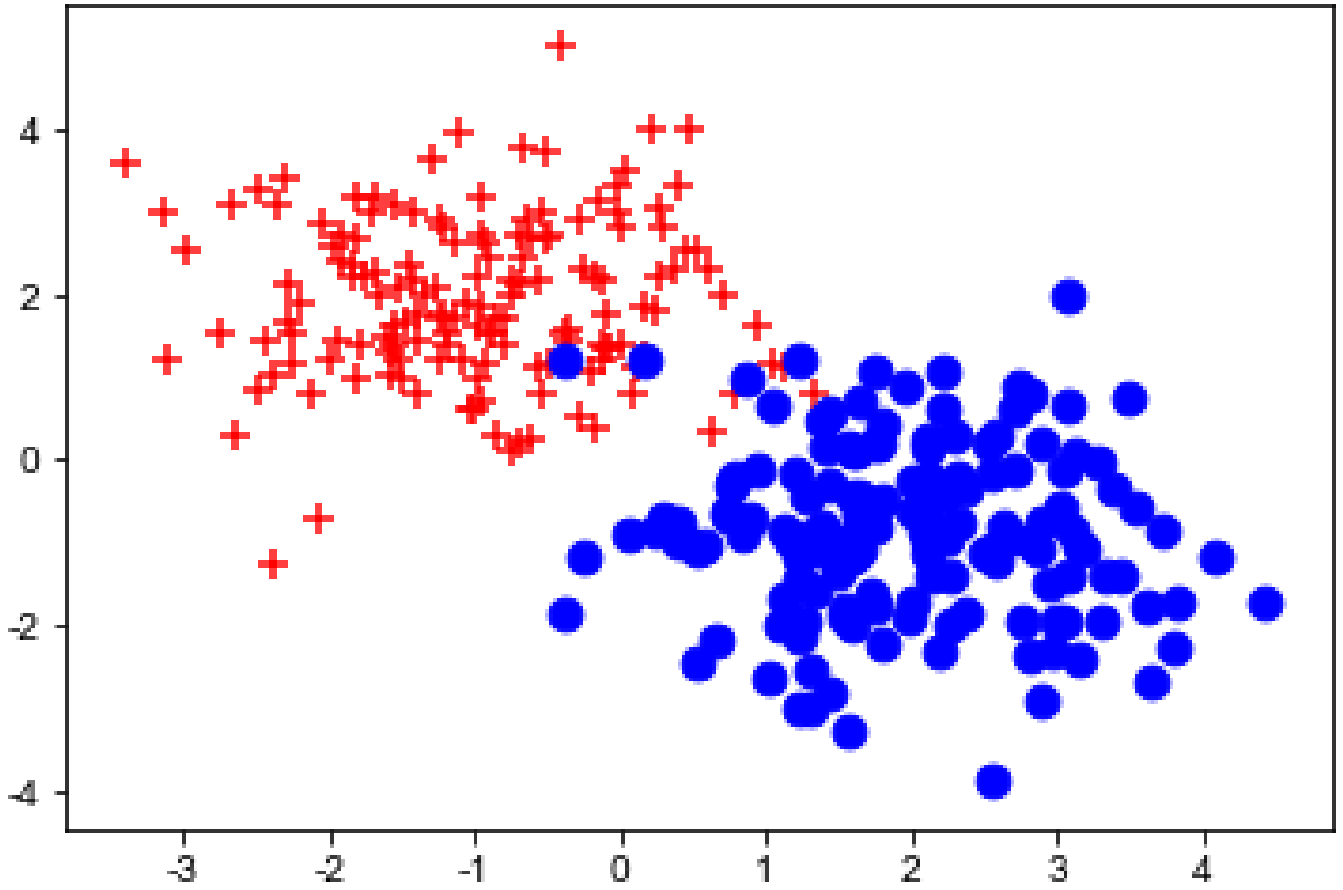
```
# X 軸資料製作
X = np.arange(0., 5., 0.2)

# 設定圖形標題
plt.title("X, X平方, X三次方線圖", fontproperties=myfont)

# 設定軸線標籤
plt.xlabel("X 軸", fontproperties=myfont)
plt.ylabel("Y 軸", fontproperties=myfont)

# 繪製三組折線圖並設定圖例
plt.plot(X, X, "r--", label="X")
plt.plot(X, X**2, "bs", label="X平方")
plt.plot(X, X**3, "g^", label="X立方")
plt.legend(loc='best', prop=myfont)

plt.show()
```

點狀圖（Scatter Charts）

- 何謂「點狀圖」
- 點狀圖繪製指令：scatter()



何謂「點狀圖」（Scatter Charts）



點狀圖繪製指令：scatter()

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

# Size of Sample Data
n = 150

# Group1 Data around (-1, 2) as Normal Distribution
X1 = np.random.normal(-1, 1, n)
Y1 = np.random.normal(2, 1, n)

# Group2 Data around (2, -1) as Normal Distribution
X2 = np.random.normal(2, 1, n)
Y2 = np.random.normal(-1, 1, n)

# Scatter Chart with Size(s), Color(c), Style(marker) of marker
plt.scatter(X1, Y1, s=75, c="red", marker="+")
plt.scatter(X2, Y2, s=75, c="blue", marker="o")
plt.show()
```

150 組中心點 (-1, 2)，
離散程度 1 的樣本點

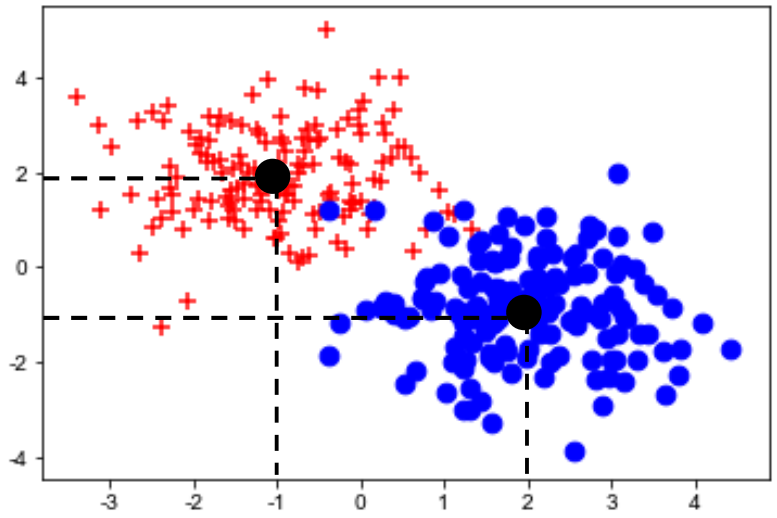
150 組中心點 (2, -1)，
離散程度 1 的樣本點

大小 75 px，顏色為 c，
樣式為 marker 的樣本圖示

任何「可迭代」的
複合資料結構

.scatter(X, Y, Style)

點狀圖
外觀設定



隨堂練習：繪製點狀圖

- 請輸入下列程式碼，練習「點狀圖」的繪製：

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  # Size of Sample Data
5  n = 150
6
7  # Group1 Data around (-1, 2) as Normal Distribution
8  X1 = np.random.normal(-1, 1, n)
9  Y1 = np.random.normal(2, 1, n)
10
11 # Group2 Data around (2, -1) as Normal Distribution
12 X2 = np.random.normal(2, 1, n)
13 Y2 = np.random.normal(-1, 1, n)
14
15 # Scatter Chart with Size(s), Color(c), Style(marker) of marker
16 plt.scatter(X1, Y1, s=75, c="red", marker="+")
17 plt.scatter(X2, Y2, s=75, c="blue", marker="o")
18 plt.show()
```

課後作業：氣象資料分析

- **資料說明**
  - 以下資料是由中央氣象局文山觀測站，下載之 2023 年 6 月氣象資料：

觀測日期(day)	當日氣壓(hPa)	氣溫(℃)	相對溼度(%)	風速(m/s)	降水量(mm)
1	996.3	25.7	90	0.3	6.5
2	1001.1	28.2	80	0.6	0

- **題目說明**
  - 請先將該檔案 WeatherData.csv 上傳至 Colab（手工上傳、或線上下載皆可），將之讀入後，直接印出 Pandas 讀入後的結果。
  - 請將該資料集，轉換成 NumPy 之後，再把它印出來。
  - 利用 Pandas 的 .describe() 函數，了解該資料集的特性。
  - 請計算出該月份的「平均溫度」、「最高濕度」、「最低氣壓」、以及「總雨量」。
  - 最後請將該月份的溫度，以折線圖的方式繪製出來。

課後作業：氣象資料分析

- 程式執行後之輸出：

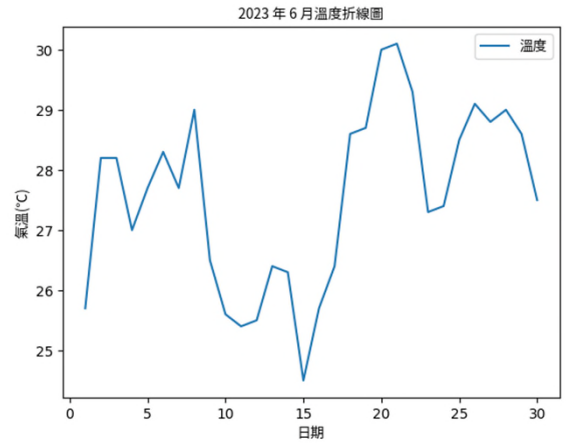
Pandas 載入的資料集之輸出

	觀測日期(day)	當日氣壓(hPa)	氣溫(℃)	相對溼度(%)	風速(m/s)	降水量(mm)
0	1	996.3	25.7	90	0.3	6.5
1	2	1001.1	28.2	80	0.6	0.0
2	3	1005.1	28.2	79	1.3	4.0
3	4	1005.9	27.0	86	0.6	91.5
4	5	1005.2	27.7	84	0.3	7.0

Pandas .describe() 之輸出

	觀測日期(day)	當日氣壓(hPa)	氣溫(℃)	相對溼度(%)	風速(m/s)	降水量(mm)
count	30.000000	30.000000	30.000000	30.000000	30.000000	30.000000
mean	15.500000	1003.420000	27.566667	82.466667	0.553333	11.933333
std	8.803408	2.844644	1.486568	5.929083	0.333976	22.894712
min	1.000000	996.300000	24.500000	73.000000	0.000000	0.000000
25%	8.250000	1001.400000	26.400000	78.250000	0.300000	0.000000
50%	15.500000	1004.450000	27.700000	82.000000	0.450000	0.500000
75%	22.750000	1005.650000	28.675000	86.750000	0.600000	8.125000
max	30.000000	1007.100000	30.100000	93.000000	1.500000	91.500000

全月份溫度折線圖



轉成 NumPy 陣列之後的輸出

```
[[1.0000e+00 9.9630e+02 2.5700e+01 9.0000e+01 3.0000e-01 6.5000e+00]
 [2.0000e+00 1.0011e+03 2.8200e+01 8.0000e+01 6.0000e-01 0.0000e+00]
 [3.0000e+00 1.0051e+03 2.8200e+01 7.9000e+01 1.3000e+00 4.0000e+00]
 [4.0000e+00 1.0059e+03 2.7000e+01 8.6000e+01 6.0000e-01 9.1500e+01]
 [5.0000e+00 1.0052e+03 2.7700e+01 8.4000e+01 3.0000e-01 7.0000e+00]
 [6.0000e+00 1.0048e+03 2.8300e+01 7.9000e+01 4.0000e-01 0.0000e+00]
```

統計量計算之後的輸出

```
平均溫度:  27.566666666666666 ℃
最高濕度:  93 %
最低氣壓:  996.3 hPa
總雨量:  358.0 mm
```

本章總結

- **NumPy 基本運算**
  - 建立矩陣： np.array([1, 2], [3, 4])
  - 讀寫元素： ary[0, 1]
  - 線性樣本點： np.arange(0.0, 0.5, 0.2)
  - 亂數樣本點： np.random.randint() / .rand() / .choice()
  - 常態分布： np.random.randn() / .normal()
  - 切片運算： ary[0:3, 1:5]
  - 統計量計算： ary.min() / .max() / .sum() / .mean() / .std()
  - 轉置： np.T
- **Pandas：讀取外部資料**
  - 建立： pd.DataFrame(X軸資料, Y軸資料, index=列標籤, columns=欄標籤)
  - 讀取 CSV： pd.read_csv()
  - 型態轉換： pd.to_numpy() / pd.values
- **Matplotlib：繪製圖形**
  - 折線圖： plt.plot(X軸資料, Y軸資料, 線段格式...)
  - 散佈圖： plt.scatter(X軸資料, Y軸資料, s=圖例大小, c=顏色, marker=樣式)