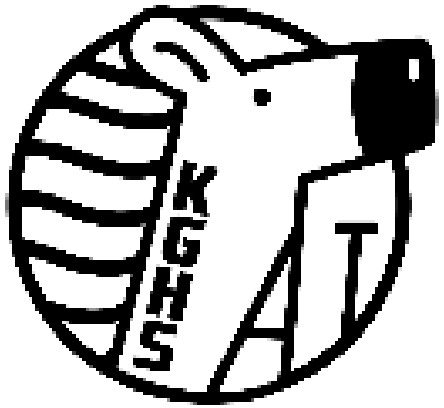


AI與Python快速教學

高雄女中新興科技推廣中心
資訊執行秘書 物理科邱崑山

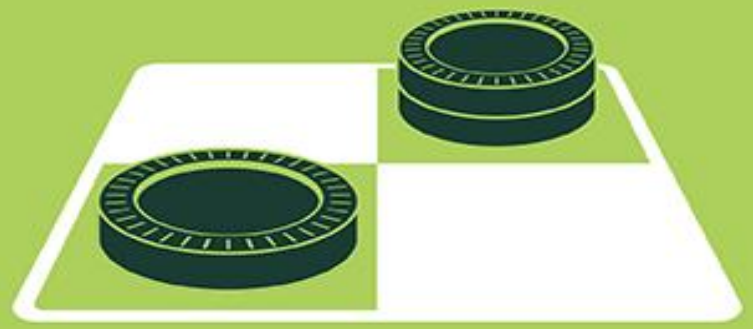
Part 0

人工智慧發展趨勢和影響

人工智慧 機器學習 和 深度學習

人工智慧(Artificial Intelligence)

使電腦具有類似人類學習及解決複雜問題、抽象思考、展現創意等能力



機器學習(Machine learning)

透過從過往的資料和經驗中學習並找到其運行規則，最後達到人工智慧的方法。

深度學習(Deep learning)

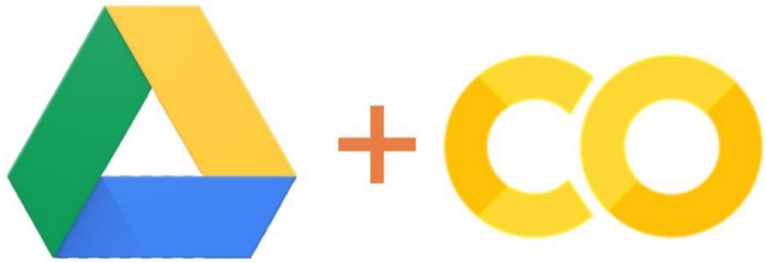
模仿人類大腦的學習模式，一種實現機器學習的技術。

1950 — 1980 — 2010

受限硬體，未能實現 | 大數據 + 分散式儲存 | GPU TPU + 平行運算

1-3 Colab 簡易操作介紹

- Google Colab 是 Google 所開發類似 Jupyter Notebook 的 Python 線上執行環境，它整合於 Google Drive 中，以外掛的形式存在。

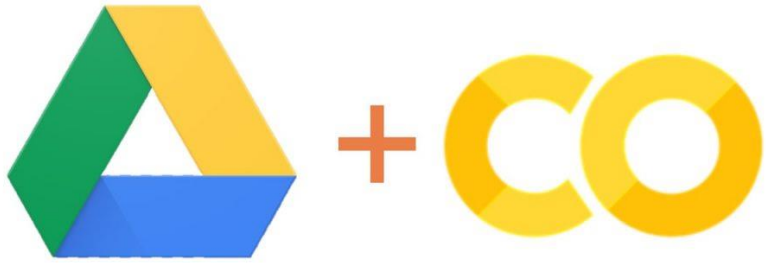


- Google Colab 的優點：
  - 跨平台、作業系統，不需要建立環境，打開瀏覽器登入 Google 就可以使用。
  - 預先安裝了常用的套件和模組，只要 import 就能使用。
  - 免費的 GPU、TPU 和記憶體可以使用，不用擔心自己的電腦等級不足以負荷。
  - 依附 Google Drive 中，方便備份、存取，也方便和他人一起協作。

準備實作環境

1-3 Colab 簡易操作介紹

- Google Colab 的限制：
  - 最多 12 小時的連續運算。
  - 在閒置一段時間後，虛擬機會被停止並回收運算資源(包括上傳的檔案)，此時只需再重新連接，但檔案要重新上傳。



準備實作環境

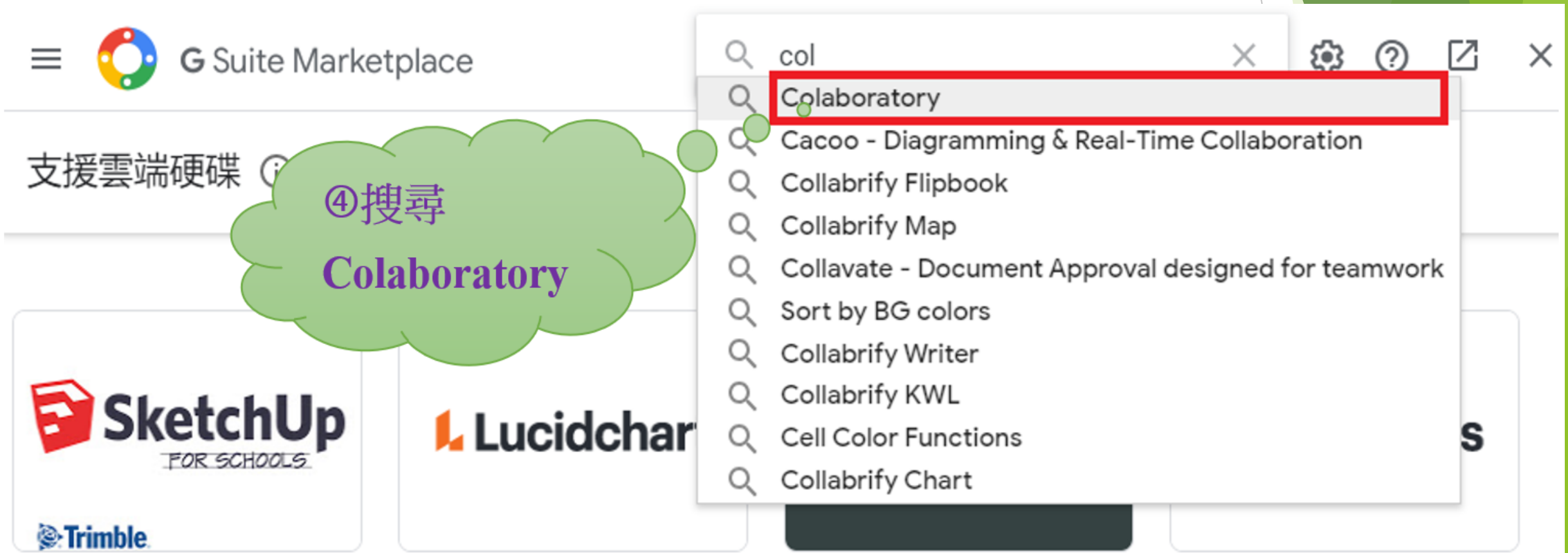
1-3 Colab 簡易操作介紹

- 如何連結 Colab 應用程式：

準備實作環境

1-3 Colab 簡易操作介紹

- 如何連結 Colab 應用程式：



準備實作環境

1-3 Colab 簡易操作介紹

- 如何連結 Colab 應用程式：

準備實作環境

1-3 Colab 簡易操作介紹

- 如何連結 Colab 應用程式：

1-3 Colab 簡易操作介紹

- Colab 掛載雲端硬碟資料夾：
  - 在 Colab 中運行 Python 程式如果需要讀取資料集檔案或字型，透過 Google 提供的套件，我們可以讓 Colab 上的程式直接讀取自己的雲端硬碟。
    - 優點：檔案存在於自己的雲端硬碟，就隨時都可以直接存取，不用每次上傳。
    - 缺點：程式運行時要輸入連結 Google Drive 所需要的授權碼。

準備實作環境

1-3 Colab 簡易操作介紹

- Colab 掛載雲端硬碟資料夾：
  - 步驟：

準備實作環境

1-3 Colab 簡易操作介紹

- 當開啟或新增一個 Colab 檔案，其附檔名為 ipynb，雲端伺服器將自動啟動一個獨立執行程序 RunTime。
  - Menu 中的 RunTime 功能包含管理這個執行程序的命令。
  - 當執行程序重新啟動 Restart 時，所有與導入套件、變量賦值等相關的內存都會丟失。
  - 若發生無預期錯誤，可以 Restart RunTime，重新再執行程式。

準備實作環境

1-3 Colab 簡易操作介紹

- ipynb 筆記本由文本 Text 和代碼 Code 兩種“單元格”組成：
  - 文本 Text：可以編輯程式重要的說明。
  - 代碼 Code：程式的主體，點擊筆記本左邊的運行按鈕或鍵盤快捷鍵[CTRL+ENTER]，將執行該單元格中的程式碼，執行完畢會顯示執行編號數值。

```
import numpy as np
from IPython import display
import matplotlib.pyplot as plt
import matplotlib.font_manager as plt_font
twfont1 = plt_font.FontProperties(fname="drive/My Drive/colab/kaiu.ttf")
```

```
### (2)將原始資料分為訓練資料集和驗證資料集
```

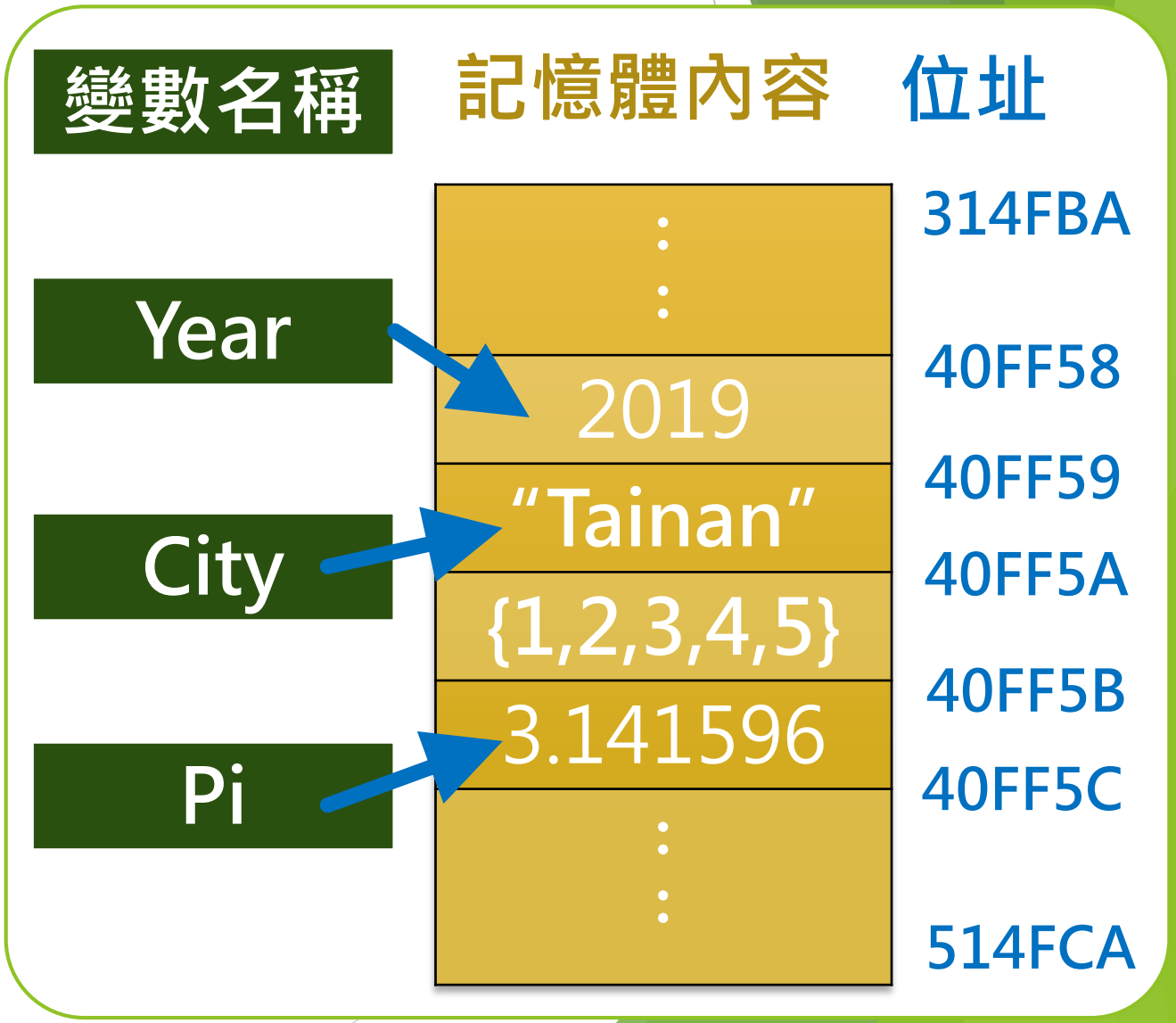
(2)將原始資料分為訓練資料集和驗證資料集

Part 1

變數與Python的資料型態

什麼是變數(variable)?

- 程式執行期間用到的資料(數值、文字、、、)，會先記錄在記憶體的某個位址中，並給它一個名稱，這就是變數。



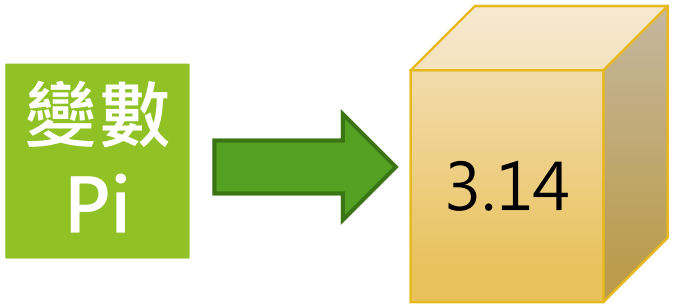
Python變數命名規則

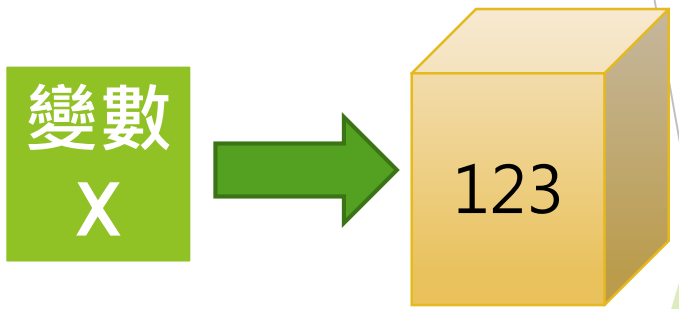
- 變數名稱由字母(a~z或A~Z) 、數字(0~9)和底線(_)組成，但首字不可為數字。
- 大小寫相異(Year和year是不同的變數名稱)。
- 用意義的英文單字來命名，以增加程式的可讀性
  例如：儲存數學成績的變數math_score或MathScore。
- Python3版本中，變數也可以用中文命名。

✗ 2year　✓ _Year　✓ _year　✓ Year2

Python變數的賦值

- 把資料貼上一個標籤(變數)
- 對變數進行運算，就是對其指向的資料作運算。





Pi=3.14

這不是等號
是賦值

X=123

y=X+1

Python的基本資料型態（Data type）

資料型態	種類	例子
數值型態 (Numeric)	整數(int)	1、2、3、4、、、
	浮點數(float)	3.1415、9.8
	虛數(complex)	2+3j、4-5j
字串型態 (String)	單行字串	“Hello world”或‘hello world’
	多行字串	“””第一行..... 第二行..... 第三行.....””” ‘’’第一行 ..... 第二行..... 第三行.....‘’’
布林值 (bool)		True(真)、False(假)

Part 2

運算子
處理資料的符號

Python的算術運算子

運算子	功能	例
+	兩數相加(亦可做字串相接)	3+4、”3”+”4”
-	兩數相減	3-4
*	兩數相乘	3*4
**	取某數的次方	3**4
/	兩數相除	10/3
%	兩數相除取餘數	10%3
//	兩數相除取整數商	10//3

Python的比較（關係）運算子

運算子	例	說明
==	3==4	• 比較運算子的結果必為布林值(bool) • 比較結果成立為True • 比較結果不成立為False
!=	3!=4	
>	3>4	
<	3<4	
>=	3>=4	
<=	3<=4	

Python的邏輯運算子

運算子	說明	舉例
A and B	A和B均為True 結果為True，其它為False	3>2 and 4>3 3>2 and 4>5
A or B	A和B均為False結果為False，其它為True	3>2 or 4>3 3>2 or 4>5
not A	A為Ture 結果為False，A為False結果為True	not 3>2 not 2>3

Python的賦值運算子

運算子	說明	舉例
=	簡單的賦值運算子	c = a + b 將 a + b 的運算結果賦值為 c
+=	加法賦值運算子	c += a 等效於 c = c + a
-=	減法賦值運算子	c -= a 等效於 c = c - a
*=	乘法賦值運算子	c *= a 等效於 c = c * a
/=	除法賦值運算子	c /= a 等效於 c = c / a
%=	取模賦值運算子	c %= a 等效於 c = c % a
**=	冪賦值運算子	c **= a 等效於 c = c ** a
//=	取整除賦值運算子	c //= a 等效於 c = c // a

Part 3 Python 的輸出和輸入

Python的輸出：使用函式print()

- 語法：print(資料1,資料2,資料3, sep=' ', end='\n')
- 參數：
  - 可以輸出多個資料，資料間需要用逗號分隔。
  - sep：指定資料間的分隔符號，預設值是一個空格。
  - end ： 指定輸出的結尾符號，預設值是分行符號 \n。
- 範例：
  - print(“Hello World Python")
  - print(“Hello” , “World” ,” Python")
  - print(“Hello” , “World” ,” Python “,sep= “!!”)
  - print(3.14*2**2)

Python的輸入：使用函式input()

- 語法：變數=input([提示字串])
- 參數：
  - 提示字串 ：提示輸入資料的字串。
- 由於input回傳給變數的資料型態是字串，若此資料要做算術運算，必須先經過int()轉為整數或float()轉為浮點數。
- 範例：
  - uname=input(“請輸入註冊的帳號")
  - Age=Input(“請輸入你的年齡”)

請看程式範例💻3-1、3-2

實作時間：計算BMI值

- 讓使用者輸入體重(kg)和身高(m)，計算出BMI值
- BMI = 體重 (kg) / 身高^2(m^2)

- 參考畫面如下：

```
請輸入體重(kg):70
請輸入身高(m):1.75
你的BMI值為 22.857142857142858
```

Part 4

Python 的選擇(分支)結構

選擇結構(執行分支)

- 依據條件的不同，執行不同的程式區塊，稱為選擇結構。
- Python中選擇結構只有 if语句。
- 每個條件後面要使用冒號 :表示接下來是滿足條件後要執行的程式區塊。
- 使用縮排來劃分語句塊，相同縮排數的語句在一起組成一個語句塊。

If條件判斷，選擇程式執行分支

- 當條件1成立時,進行區塊1的運算；不成立時,進行條件2的判斷
- 當條件2成立時,進行區塊2的運算；不成立時,進行條件3的判斷、、、、
- 若所有條件都不成立，可執行else裡的程式區塊

```
if 條件1成立:
    條件1成立要執行的區塊1
    ..................
elif 條件2成立:(不一定要有)
    條件2成立要執行的區塊2
    ..................
else:(不一定要有)
    不滿足上面條件要執行的區塊
    ..................
```

縮排

請看程式範例4-1、4-2、4-3

實作時間：計算BMI並輸出分級評等

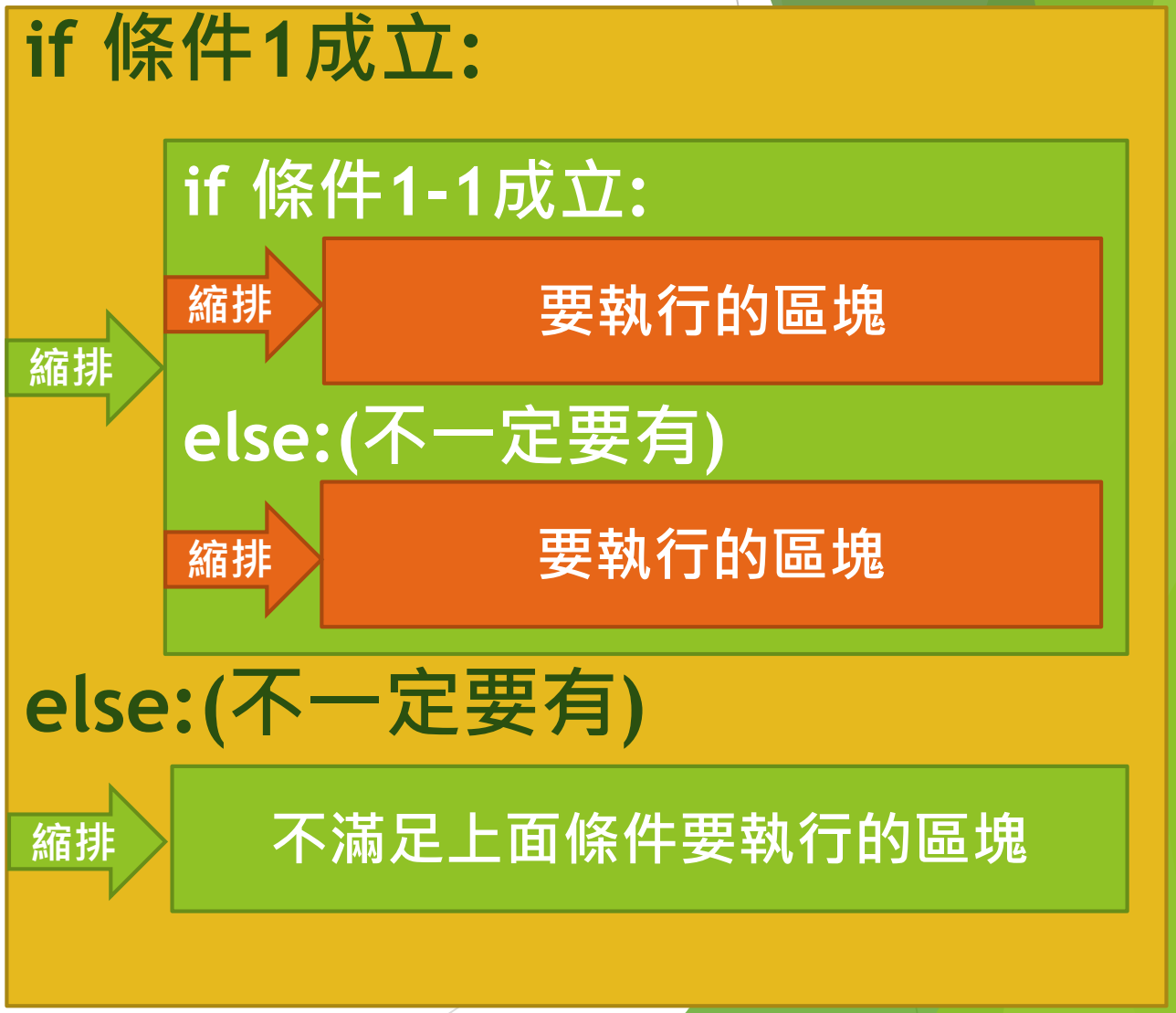
- 讓使用者輸入體重(kg)和身高(m)，計算出BMI值，並依據BMI值顯示其分級
- BMI = 體重 (kg) / 身高^2(m^2)
- 參考畫面如下：

```
請輸入體重(kg):70
請輸入身高(m):1.75
正常範圍
```

分 級	身體質量指數
體重過輕	BMI ＜ 18.5
正常範圍	18.5 ≦ BMI ＜24
過 重	24 ≦ BMI ＜ 27
輕度肥胖	27 ≦ BMI ＜ 30
中度肥胖	30 ≦ BMI ＜ 35
重度肥胖	BMI ≧ 35

多層If條件判斷，選擇程式執行分支

- 當條件1成立時,要執行的程式區塊內又有選擇結構。
- 第二層選擇結構必須再縮排一次



Part 5

Python 的迴圈(循環)結構

循環結構(迴圈)

- 當需要重複進行運算的時候使用迴圈。
- Python中的迴圈語句有 for 和 while。
- 當重複的次數可以清楚被計算或當迭代的表現明顯時使用for迴圈。
- 當重複的次數難以計算(但條件清楚)或是有條件的重複時使用while。
- 迴圈執行的條件要改變，最後能跳出迴圈，否則會形成無窮迴圈。

while 迴圈---不知循環次數的迴圈

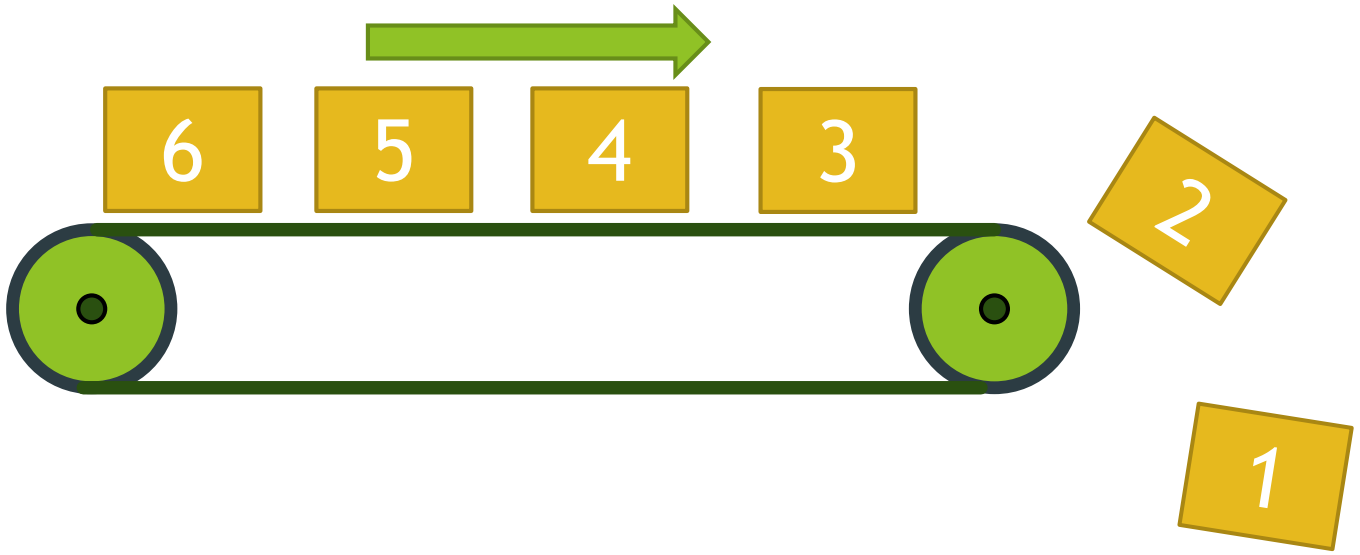
- 當條件判斷成立時,進行區塊的程式區塊
- 程式區塊執行完畢後, 再次檢查判斷條件, 若依然成立則繼續執行區塊
- 若條件判斷不成立，不再繼續執行區塊的動作稱為跳出迴圈

請看程式範例💻5-1、5-2

實作時間：
計算(1+2+3、、、+n) >10^6,最小n是多少？

迭代物件

- 可迭代物件：可以想像成這種物件裡面有可數的項目可依特定順序一個一個取出。



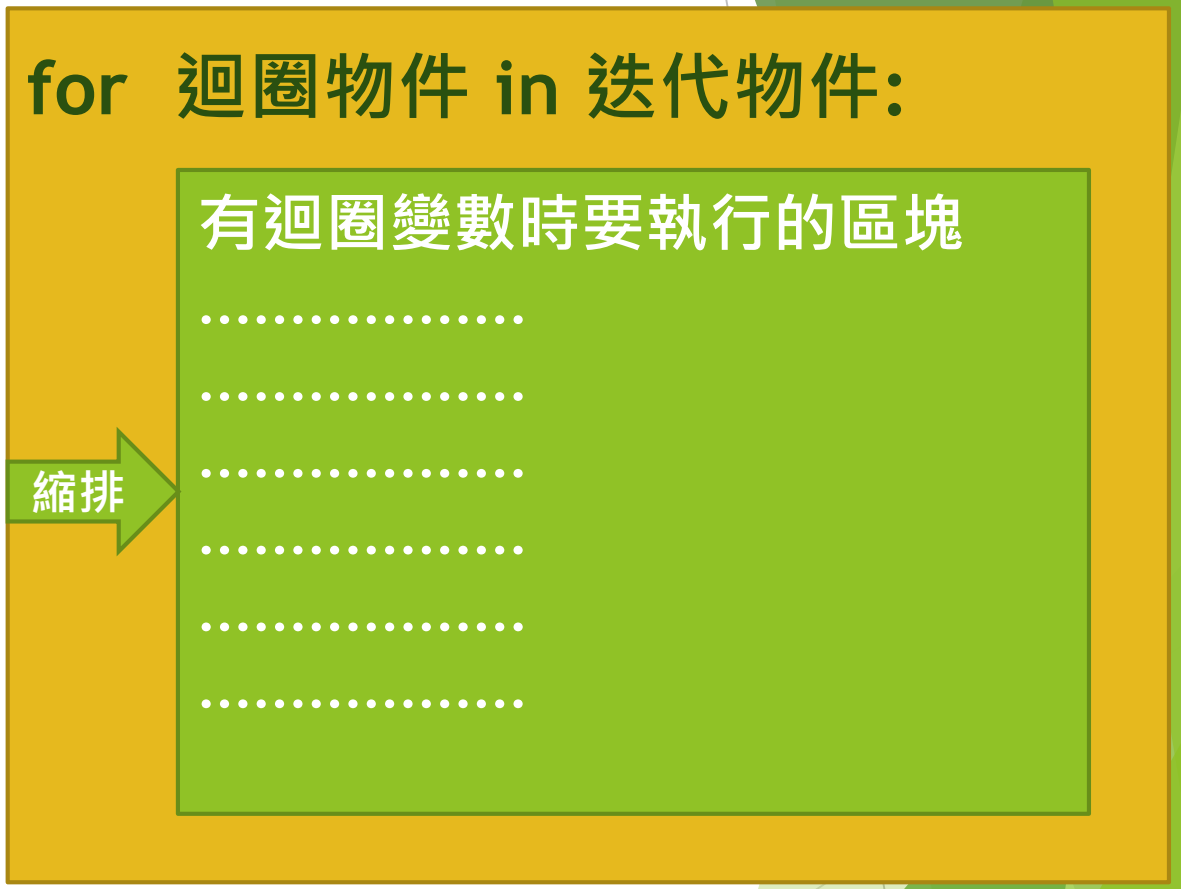
range() 函數產生迭代物件

語法：range(stop)或range(start, stop[, step])

- 參數說明：
  - start: 計數從 start 開始。默認是從 0 開始。
    例如range（5）等價於range（0， 5）;
  - stop: 計數到 stop 結束，但不包括 stop。
    例如：range（0， 5） 是[0, 1, 2, 3, 4]沒有5
  - step：步長，默認為1。
    例如：range（0， 5） 等價於 range(0, 5, 1)

for 迴圈---知道循環次數的迴圈

- for迴圈的執行流程從可迭代物件中取出一個項目給迴圈物件， 執行程式區塊
- 執行完成式區塊，再從可迭代物件中取出下一個項目，在執行程式區塊
- 可迭代物件的項目盡皆取出跳出迴圈



請看程式範例💻5-3

多層迴圈

- while迴圈或for迴圈都支援多層迴圈，各層之間的縮排務必清楚，冒號也要記得加上

```
for  迴圈物件1 in 迭代物件1:
    For 迴圈物件2 in 迭代物件2:
    ……
    ……
    ……
```

縮排

請看程式範例💻5-4

實作時間：請印出九九乘法表

- 請用二維迴圈印出九九乘法表
- 提示：print()參數中
  - sep=“”，可設定沒有分隔字符
  - end=“\t”，可以讓輸出後移到下一個定位點。

1x1=1	1x2=2	1x3=3	1x4=4	1x5=5	1x6=6	1x7=7	1x8=8	1x9=9
2x1=2	2x2=4	2x3=6	2x4=8	2x5=10	2x6=12	2x7=14	2x8=16	2x9=18
3x1=3	3x2=6	3x3=9	3x4=12	3x5=15	3x6=18	3x7=21	3x8=24	3x9=27
4x1=4	4x2=8	4x3=12	4x4=16	4x5=20	4x6=24	4x7=28	4x8=32	4x9=36
5x1=5	5x2=10	5x3=15	5x4=20	5x5=25	5x6=30	5x7=35	5x8=40	5x9=45
6x1=6	6x2=12	6x3=18	6x4=24	6x5=30	6x6=36	6x7=42	6x8=48	6x9=54
7x1=7	7x2=14	7x3=21	7x4=28	7x5=35	7x6=42	7x7=49	7x8=56	7x9=63
8x1=8	8x2=16	8x3=24	8x4=32	8x5=40	8x6=48	8x7=56	8x8=64	8x9=72
9x1=9	9x2=18	9x3=27	9x4=36	9x5=45	9x6=54	9x7=63	9x8=72	9x9=81

Part 6

Python 的結構資料型態

Python的結構(容器)資料型態（ Data type ）

一個變數指向的記憶體位址中，存放多個資料，容器資料(Container))屬於資料結構

種類	例子	備註
列表 (list)	[1,2,3,4,5,6,7]	資料可變，有序資料
元組(tuple)	(1,2,3,4,5,6,7)	資料不可變，有序資料
集合(set)	{1,2,3,4,5,6,7}	資料可變，無序資料
字典(dict)	{“香蕉”:20,“蘋果”:30}	資料可變，關鍵資料

列表(list) –以數字為索引的一串資料

- 使用中括弧[]創建列表，逗號分開各資料。
  - 語法：變數名稱=[資料1,資料2,資料3 、、、]
  - 舉例：

    list1 = ['Google', ‘facebook', 1997, 2000]
    list2 = [1, 2, 3, 4, 5];
    list3 = ["a", "b", "c", "d"]
  - 列表的資料項目不需要具有相同的類型。
- 第一個索引是0，第二個索引是1，依此類推。
- range() 函數返回的是一個可迭代物件而不是列表類型，可以利用內建函list將可迭代物件中的資料取出組成列表
  例：list(range(0, 10, 2))

取出列表中的資料

- 使用索引來取出列表中的值
  語法：列表變數[索引值]
  - 規則：索引編號有以下兩種：
    由左到右 | 0 | 1 | 2 | 3 | .... |
  - 由右到左 | .... | -4 | -3 | -2 | -1 |
  - 練習：

```
list1= ["a", "b", "c", "d"]
print(list1[0])
print(list1[1])
```

請看程式範例💻6-1

取出列表中的資料

- 同樣你也可以使用方括號的形式截取列表
  語法：列表變數[起始索引值:結束索引值]
  - 規則：遵循左取右棄原則，前索引省略由頭開始；後索引省略到尾結束
  - 練習：

```
list1= ["a", "b", "c", "d"];
print(list1[0:3])
print(list1[1:])
Print(list[:3])
```

Python二(多)維列表

- 在列表裡創建其它列表
  - 語法：變數名稱=[列表1,列表2 ,列表3 、、、]
  - 練習：

```
a=[[1, 2, 3] , [4, 5, 6],[7,8,9]]
b=[["a", "b", "c"] , ["d", "e", "f"] ]
```

- 取出二維列表中的資料
  - 練習：

```
print(a[0])
print(a[0][2])
```

- 以此類推，亦可建三維列表、四維列表、、、多維列表

請看程式範例💻6-2

Python內建的列表函式

函式和功能	範例
len(列表變數) 返回列表中元素個數	list1 = ['Google', 'yahoo', 'Kimo'] Print(len(list1))
列表物件.append(新的元素資料) 在列表末尾添加新的物件	list1 = ['Google', 'yahoo', 'Kimo'] list1.append('facebook') print ("更新後的列表：", list1)

請看程式範例💻6-3、6-4

元組(tuple)-不能改變資料的列表

- 使用小括弧 () 創建元組，逗號分開各資料。
  - 語法：變數名稱=(資料1,資料2,資料3 、、、)
  - 練習：

```
month= (' January' , ' February ', ' March ', ' March')
print(month)
```

- 元組可避免資料被修改。
- 元組中只包含一個元素時，需要在元素後面添加逗號，否則括弧會被當作運算元使用。
- 物件方法和列表相同，只要是不會變動資料的也都可以使用。
- list和tuple均可視為可迭代物件。

Part 7

函數(Function)

什麼是函式(Function)?

- 依據輸入的資料作處理並返回處理結果的程式區塊
- 函式可以被重複使用，不同的輸入資料做不同的處理，產生不同結果。
- 簡潔、容易維護的程式架構是由一群函式和簡短的主程式組成。
- 類似數學的函數用法，但程式上的函式不僅僅只是處理數學的運算。

$$Y=f(V_0,t)=V_0xt-0.5x9.8xt^2$$

自訂函式--定義一個由自己想要功能的函式

- 函式程式碼塊以 def 關鍵字開頭，後接函式識別字名稱和圓括號 ()。
  - 函式名稱的的命名方式如同變數命名。
  - 任何傳入參數和引數必須放在圓括號中間，圓括號之間可以用於定義參數。
- 函數內容以冒號：起始，並且縮排。
- 函式結尾可以用return ，選擇性地返回資料。
  - 沒有要回傳的資料，可以省略。
  - 回傳多個資料，可以使用tuple來包裝。

自訂函數語法格式如下

def 函數名(參數1,參數2,、、):

縮排

函式內程式區塊

.....

.....

.....

return 要返回資料

def 函數名(參數1,參數2,、、):

縮排

函式內程式區塊

.....

.....

.....

return (資料1,資料2)

函式的呼叫

- 變數=函式名稱(參數1,參數2,、、、)
  - 變數可以接收函式回傳的資料。
  - 小括號內逗號分開數個要傳遞到函式作處理的資料，稱為參數。
  - 預設情況下，參數值(個數、型態)是按函式宣告中定義的順序匹配起來的。
- 變數1,變數2,變數3=函式名稱(參數1,參數2,、、、)
  - 多個變數逗號分開來接收函式回傳的tuple來包裝資料。

請看程式範例💻7-1、7-2

實作時間：
設計一個函式用來計算每個月要繳的房貸金額，這個函式輸入三個參數，分別是貸款本金、利率、貸款年數。

然後在主程式中讓使用者可以輸入這三個參數資料，然後顯示出每月應繳房貸。

呼叫函式時的參數種類

- 必須參數:
  - 須以正確的順序傳入函式。
  - 調用時的數量必須和聲明時的一樣。
- 關鍵字參數：
  - 使用關鍵字來匹配傳入的參數值。
  - 使用關鍵字參數允許函數調用時參數的順序與聲明時不一致。

請看程式範例💻7-3、7-4

呼叫函式時的參數種類

- 默認參數:
  - 調用函數時，如果沒有傳遞關鍵字參數，則會使用默認參數。
  - 默認值在定義函式時在小刮號內賦值。
- 不定長度參數：
  - 不定長度參數：同時傳入幾個資料作為參數。
  - 函式的參數前使用*：將傳入函數內的多個資料組成元組
  - 函式的參數前使用**：將傳入函數內的多個資料群組成字典

請看程式範例💻7-5、7-6、7-7

函式與變數的作用範圍

- 宣告在最外層的稱作全域變數，全域變數作用範圍為整個檔案。
- 宣告在函式內的變數稱作區域變數，函式內若沒有那個變數就會往函式外找尋。
- 函式內使用global宣告變數，該變數將明確指向全域變數。

請看程式範例🖳7-8、7-9、7-10

Part 8

使用模組、套件

Python的模組和套件

- Python擁有功能完備、豐富套件和模組生態系。
  例：讀寫檔案、自然語言處理、網路爬蟲、網站開發、機器學習、、、等
- 可以在程式中使用import，匯入符合需求的Python套件或模組，節省程式開發的時間。
- 模組是單獨一個Python程式檔案，而套件是由許多Python程式檔案放在同一個資料夾中所組成。

import模組的方式

- 在主程式檔中匯入整個模組：
  - import 模組名稱
    #匯入一個模組
  - import 模組名稱1,模組名稱2
    #匯入多個模組
  - import 模組名稱 as 模組別名
    #匯入一個模組，並使用別名

- 執行匯入模組的函式：
  - 使用「模組名稱.函式名稱()」
  - 使用「模組別名.函式名稱()」

Import套件的方式

- 在主程式檔中匯入整個模組：
  - from 套件名稱 import 模組名稱
    #匯入套件中的一個模組
  - from 套件名稱 import 模組名稱1,模組名稱2
    #匯入套件中多個模組
  - from 套件名稱 import 模組名稱 as 模組別名
    #匯入套件中一個模組，並使用別名

- 執行匯入模組的函式：
  - 使用「模組名稱.函式名稱()」
  - 使用「模組別名.函式名稱()」

Part 9

Numpy套件介紹

什麼是Numpy

- Numpy 是 Python 的一個重要模組，主要用於資料處理上。
- Numpy 底層以 C 和 Fortran 語言實作，所以能快速操作多重維度的陣列。
- Numpy 具備平行處理的能力，可以將操作動作一次套用在大型陣列上。
- 其餘重量級的資料科學相關套件（例如：Pandas、SciPy、Scikit-learn 等）都幾乎是奠基在 Numpy 的基礎上。

使用Numpy模組

- 匯入numpy模組
  語法：import numpy as np
- Numpy主要處理的資料型態為陣列(nd.array)，所以要先建立陣列資料。
- 陣列(nd.array)的取值和擷取部分資料的方法和列表相同，這裡不再述。

建立陣列的幾種方式

- 將列表資料轉成陣列資料
  語法：陣列名稱=np.array(列表資料)
  - A=np.array([1,2,3]) #1維陣列
  - B=np.array([[1,2,3],[4,5,6]]) #2維陣列
- 將產生可迭代物件轉成陣列資料
  語法：陣列名稱=np. arange(起始值,結束值[,間隔])
  - C=np.arange(3,33,5)
  - D=np.arange(1,10)

建立陣列的幾種方式

- 產生亂數實數(0~1)的陣列資料
  語法：陣列名稱=np. random.random(陣列維度)
  - A=np.random.random(3)
    #產生一維陣列，有3個元素
  - B=np.random.random((2,3))
    #產生二維陣列，有2x3個元素
- 產生元素都是0陣列資料
  語法：陣列名稱=np. zeros(陣列維度)
  - A=np.zeros(3)　　#產生一維陣列，有3個都是0的元素
  - B=np.zeros((2,3))　#產生二維陣列，2x3個都是0的元素

建立陣列的幾種方式

- 產生元素都是1陣列資料
  語法：陣列名稱=np. ones(陣列維度)
  - A=np.ones(3)　　#產生一維陣列，有3個都是1的元素
  - B=np.ones((2,3))　　#產生二維陣列，2x3個都是1的元素
- (含)最大值和(含)最小值間取幾個等差數值
  語法：陣列名稱=np. linspace(最小值,最大值,個數)
  - A=np.linspace(0,5,6)
    #產生0到5(包含0,5)之間的6個數字，前後兩數字間差距相同(等差)

陣列的重要屬性

- 關於陣列，有什麼基本屬性可以使用
  - ndim()：取得陣列的維度數量
  - shape()：陣列的形狀
  - size()：陣列的元素數量
  - dtype()：資料型態
  - itemsize()：陣列中元素的大小(位元組為單位)
  - nbytes()：陣列的大小(位元組為單位) 一般來說 nbytes = itemsize * size

修改陣列的維度- reshape

- 語法：np.reshape(a, b)或 np.reshape(a, b, -1)
- 可以修改ndarray每個維度的元素個數
- 使用-1，代表這個維度就交給NumPy自動計算
- 舉例：

```
a1 = np.arange(12)
print(a1)
print(a1.shape)
a2 = a1.reshape(3,4)
print(a2)
print(a2.shape)
```

陣列基本算術運算

- 相對應位置元素做運算：
  - 陣列相加：x+y
  - 陣列相減：x-y
  - 陣列相乘：x*y
  - 陣列相乘：x/y
  - 陣列次方：x**y
- 對陣列每一個元素做賦值運算：
  - x+=1,x-=1,x*=2,x=x/2, x**=2
- 矩陣轉置(行、列互換)：X.T

陣列使用函數

- 取平方根：**np.sqrt(x)**
- 取得正弦：**np.sin(x)**　　其他三角函數用法相同。
- 取得對數：**np.log(x)**
- 取得e的次方：**np.exp(x):** 。
- 取得最小值：**np.min(x)或x.min()**
- 取出每一行的最小值：**np.min(x, axis=0)或x.min(axis=0)**
- 取出每一列的最小值：**np.min(x, axis=1)或x.min(axis=1)**
- **max,sum,mean,std用法相同。**

陣列運算

- 矩陣乘法：np.dot(X,Y)
  - X是m列xn行 矩陣 ，Y是n列xk行矩陣，則np.dot(X,Y) 得到m列xk行矩陣。
  - 運算方法，前矩陣X的整列元素和後矩陣Y的整行元素，相乘後求和，放到對應的列、行位置為新矩陣。
  - 前矩陣的行數務必要等於後矩陣的列數

陣列運算

$$\begin{bmatrix}1 & 2\\3 & -2\end{bmatrix}\begin{bmatrix}2 & -1\\1 & 3\end{bmatrix} = \begin{bmatrix}1\times2+2\times1 & 1\times(-1)+2\times3\\3\times2+(-2)\times1 & 3\times(-1)+(-2)\times3\end{bmatrix} = \begin{bmatrix}4 & 5\\4 & -9\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\3 & -2\end{bmatrix}\begin{bmatrix}2 & -1\\1 & 3\end{bmatrix} = \begin{bmatrix}1\times2+2\times1 & 1\times(-1)+2\times3\\3\times2+(-2)\times1 & 3\times(-1)+(-2)\times3\end{bmatrix} = \begin{bmatrix}4 & 5\\4 & -9\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\3 & -2\end{bmatrix}\begin{bmatrix}2 & -1\\1 & 3\end{bmatrix} = \begin{bmatrix}1\times2+2\times1 & 1\times(-1)+2\times3\\3\times2+(-2)\times1 & 3\times(-1)+(-2)\times3\end{bmatrix} = \begin{bmatrix}4 & 5\\4 & -9\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\3 & -2\end{bmatrix}\begin{bmatrix}2 & -1\\1 & 3\end{bmatrix} = \begin{bmatrix}1\times2+2\times1 & 1\times(-1)+2\times3\\3\times2+(-2)\times1 & 3\times(-1)+(-2)\times3\end{bmatrix} = \begin{bmatrix}4 & 5\\4 & -9\end{bmatrix}$$

陣列的疊加(Stacking)

- 疊加:
  - np.vstack(A, B)：將A, B矩陣沿著垂直軸堆疊！

$$A=\begin{bmatrix}11 & 2 & 23\\ 4 & 25 & 6\end{bmatrix}$$

$$B=\begin{bmatrix}7 & 8 & 9\\ 10 & 11 & 12\end{bmatrix}$$

vstack

$$\begin{bmatrix}11 & 2 & 23\\ 4 & 25 & 6\\ 7 & 8 & 9\\ 10 & 11 & 12\end{bmatrix}$$

陣列的疊加(Stacking)

- 疊加:
  - np.hstack(a, b)：將a, b矩陣沿著水平軸堆疊！

$$A=\begin{bmatrix}1 & 4\\ 2 & 5\\ 3 & 6\end{bmatrix}$$

$$B=\begin{bmatrix}7 & 0\\ 8 & 2\\ 9 & 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 4 & 7 & 0\\ 2 & 5 & 8 & 2\\ 3 & 6 & 9 & 1\end{bmatrix}$$

從csv檔案載入陣列

- 語法：陣列名稱=np.loadtxt(‘檔名.csv', 關鍵字參數)
- 參數：
  - delimiter = ‘,‘ 表示資料間分隔符號為逗號
  - dtype = int 或float 載入資料格式
  - unpack=False 每一列成一個向量, 而不是合併在一起
  - skiprows=0 忽略某些列

實作0-1 將csv檔資料讀進numpy 矩陣

- 利用numpy.loadtxt()函式從目錄『資料集』中的data1.csv檔，將資料讀進numpy矩陣中。

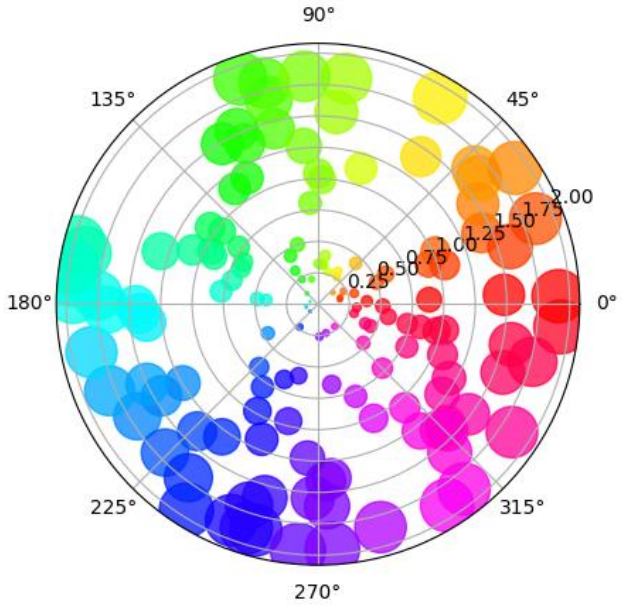
- 分別印出資料檔中的第一列、第二列、第三列。

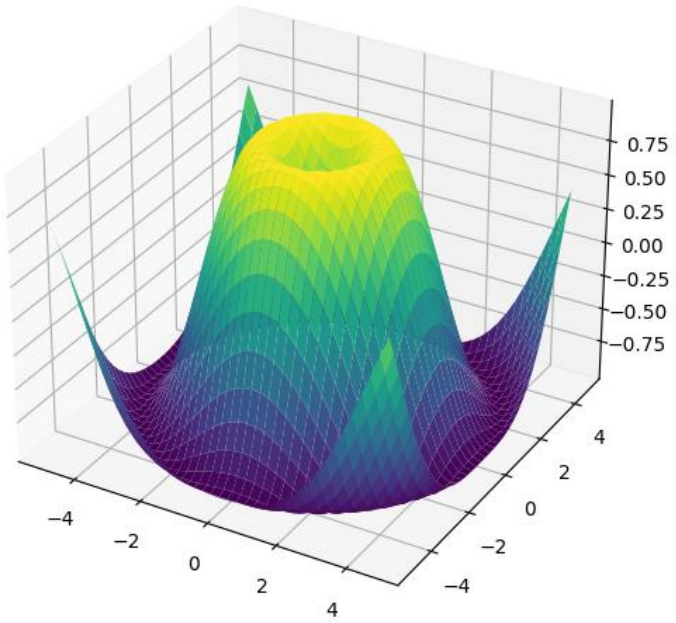
Part 10

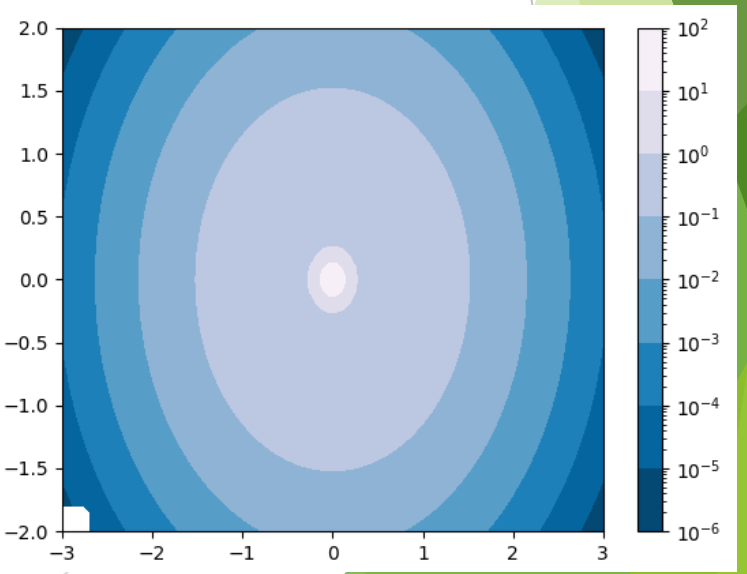
可視化資料分析 Matplotlib

什麼是Matplotlib

- Matplotlib 是 Python 的繪圖套件。
- 歷史悠久，因此有很多的教學文章或是範例可參考。
- 畫圖功能最齊全，基本上沒什麼圖表畫不出來的。
- 它的文檔相當完備，並且Gallery頁中有上百幅縮略圖，打開之後都有來源程式。
  - ◆網站https://matplotlib.org/gallery.html







Matplotlib畫平面圖(關係、散佈、等高)

■ 載入必要模組

```
#數據、矩陣處理套件numpy
import numpy as np
#繪圖處理套件matplotlib
import matplotlib.pyplot as plt
#繪圖處理套件顯示中文matplotlib.font_manager
import matplotlib.font_manager as plt_font
#設定中文字體物件和字型檔案路徑
twfont1 = plt_font.FontProperties(fname="字型路徑")
```

使用Matplotlib流程

```
#設定繪圖區大小
plt.figure(figsize=(12,6))
#繪圖區的標題，設定用中文字體twfont1，字體大小20
plt.title(“標題說明文字",fontproperties=twfont1,fontsize=20)
#設定橫軸和縱軸的標題
plt.xlabel(“x軸標題文字",fontproperties=twfont1,fontsize=20)
plt.ylabel(“y軸標題文字",fontproperties=twfont1,fontsize=20)
```

2D繪圖指令

```
#若顯示多個繪圖物件的label圖示
plt.legend(prop=twfont1)
```

Matplotlib繪製關係圖-plot()

- 語法：plt.plot([x軸資料],y軸資料,格式,label=“標籤”)
- 可用格式如下：

顏色	字元
藍色	b
綠色	g
紅色	r
青色	c
紫色	m
黃色	y
黑色	k
白色	w

數據標示	字元
點	.
像素	,
圓	o
倒三角	v
正三角	^
左三角	<
右三角	>

標示	字元
正方形	s
五邊形	p
星形	*
X	x
菱形	D
窄菱形	d
加號	+

線條	字元
實線	-
短橫線	--
點劃線	-.
虛線	:

請看程式範例💻

實作0-2 劃出資料間的關係圖

- 利用loadtxt()函式從目錄『資料集』中的data1.csv檔，將資料讀進numpy矩陣中。
- 資料檔中的第1列為自變數x、第2列為y1、第3列為y1。
- 用plot()函數劃出y1-x和y2-x關係圖

實作0-3 劃出自定義函數的函數圖

- 先自定義一個多項式函數。
- 用plot()函數劃出個種自定義函數的函數圖。

Matplotlib繪製散佈圖-scatter()

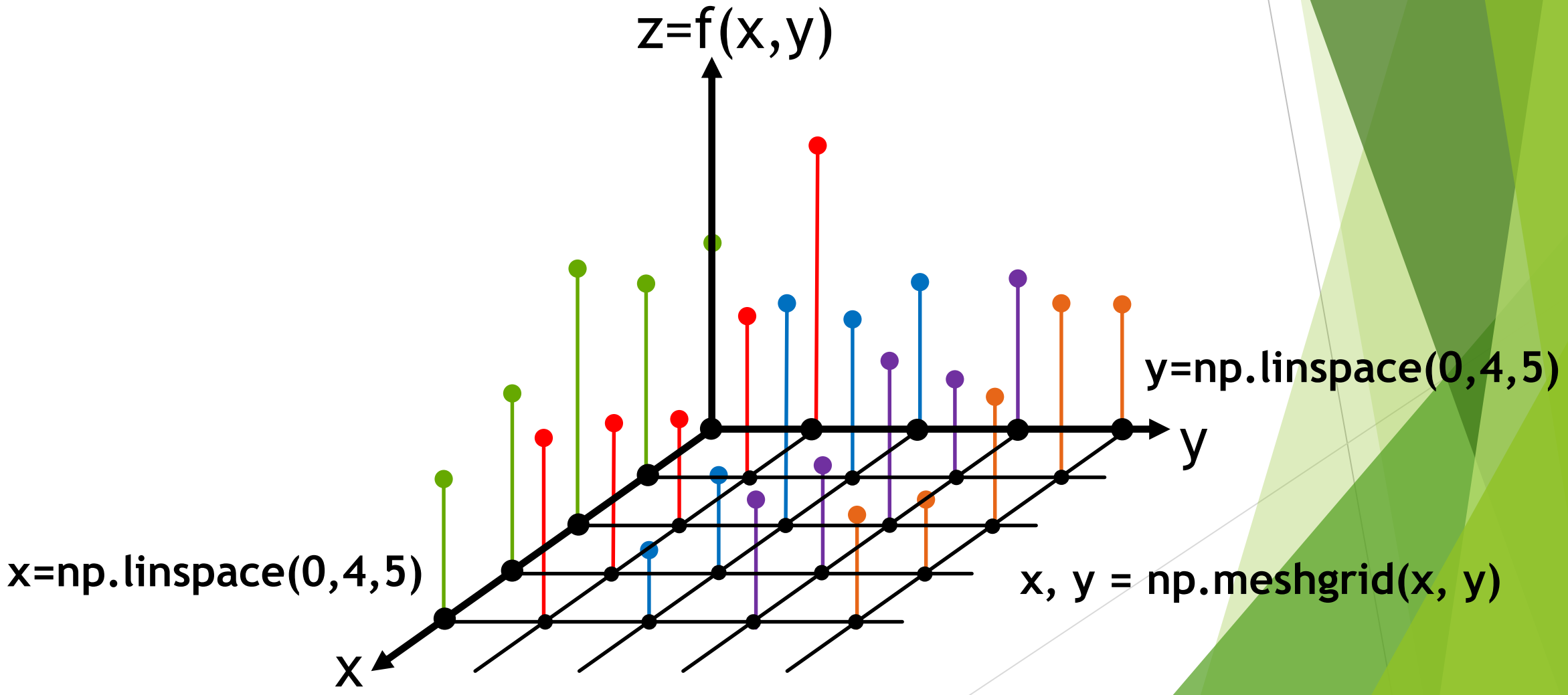
- 語法：scatter(x軸資料,y軸資料,c='顏色',s= 大小,marker='圖示')
  - 顏色使用單色：c='r'或c='y'
  - 顏色使用多色：c=['r','y','g','b','r','y','g','b','r']
  - 顏色使用隨機色盤：c=y軸資料
  - 大小隨y軸資料而變：s=y軸資料運算結果

實作0-4 劃出資料間的關係圖

- 利用loadtxt()函式從目錄『資料集』中的data3.csv檔，將資料讀進numpy矩陣中。
- 資料檔中的第1列為x軸座標、第2列為y軸座標、第3列為此座標的環境汙染程度。
- 用scatter()函數劃出散佈圖，用顏色或點大小表達汙染程度。

Matplotlib-鋪設網格點

- 等高線圖或3D曲面圖都是描述三個量值的關係，要畫圖先由x軸和y軸構成網格點為底，投射到z軸的高度。



Matplotlib-鋪設網格點

- 等高線圖或3D曲面圖都是描述三個量值的關係，要畫圖先由x軸和y軸構成網格點為底，投射到z軸的高度。
- 鋪設網格點

#產生橫軸、縱軸座標的數據點矩陣

x=np.arange(始,末,間隔)或x=np.linspace(始,末,數量)

y=np.arange(始,末,間隔)或y=np.linspace(始,末,數量)

#利用橫軸、縱軸座標的數據點鋪設網點

x, y = np.meshgrid(x, y)

- 分別將x,y對應的點，計算出Z的大小，z=f(x,y)

請看程式範例💻10-1

Matplotlib畫等高圖流程-contour()

- 語法：CS=plt.contour(x網格,y網格,Z值,等高線數量)
- 等高線值標示
  plt.clabel(CS,inline=1,fontsize=10)

實作0-5 劃出自定義函數的等高線圖

- 先自定義一個雙變數函數。
- 用plt.contour()函數劃出這個雙變數函數的等高線分布圖。

Matplotlib畫3D曲面圖流程

```
fig=plt.figure(figsize=(寬度幾吋,高度幾吋)) # 開啟繪圖區域
ax=fig.gca(projection='3d')                  # 設定為 3D 繪圖物件
ax.set_zlim(最小值,最大值)                    # 設定x坐標軸範圍
ax.set_xlim(最小值,最大值)                    # 設定y坐標軸範圍
ax.set_ylim(最小值,最大值)                    # 設定z坐標軸範圍
# 設定坐標軸說明文字
ax.set_xlabel('X軸說明文字',fontproperties=twfont1)
ax.set_ylabel('y軸說明文字',fontproperties=twfont1)
ax.set_zlabel('Z軸說明文字',fontproperties=twfont1)
```

3D繪圖指令

```
#設定視線和水平面的夾角，設定立體圖順時針旋轉角度
ax.view_init(elev=俯視角度, azim=側視角度)
```

使用Matplotlib畫3D曲面-plot_surface()

- 語法：ax.plot_surface(x軸,y軸,z軸, cmap=“色盤”,alpha=“透明度”)

- 畫關係圖語法：ax.plot(x軸,y軸,z軸, “格式")

- 顯示數值及色階對應方式：
  fig.colorbar(surf, shrink=0.5, aspect=5)
  - ◆ shrink 設定 colorbar 長度為圖片高度的幾倍
  - ◆ aspect 設定 colorbar 寬、高比

實作0-6 劃出自定義函數的3D曲面圖

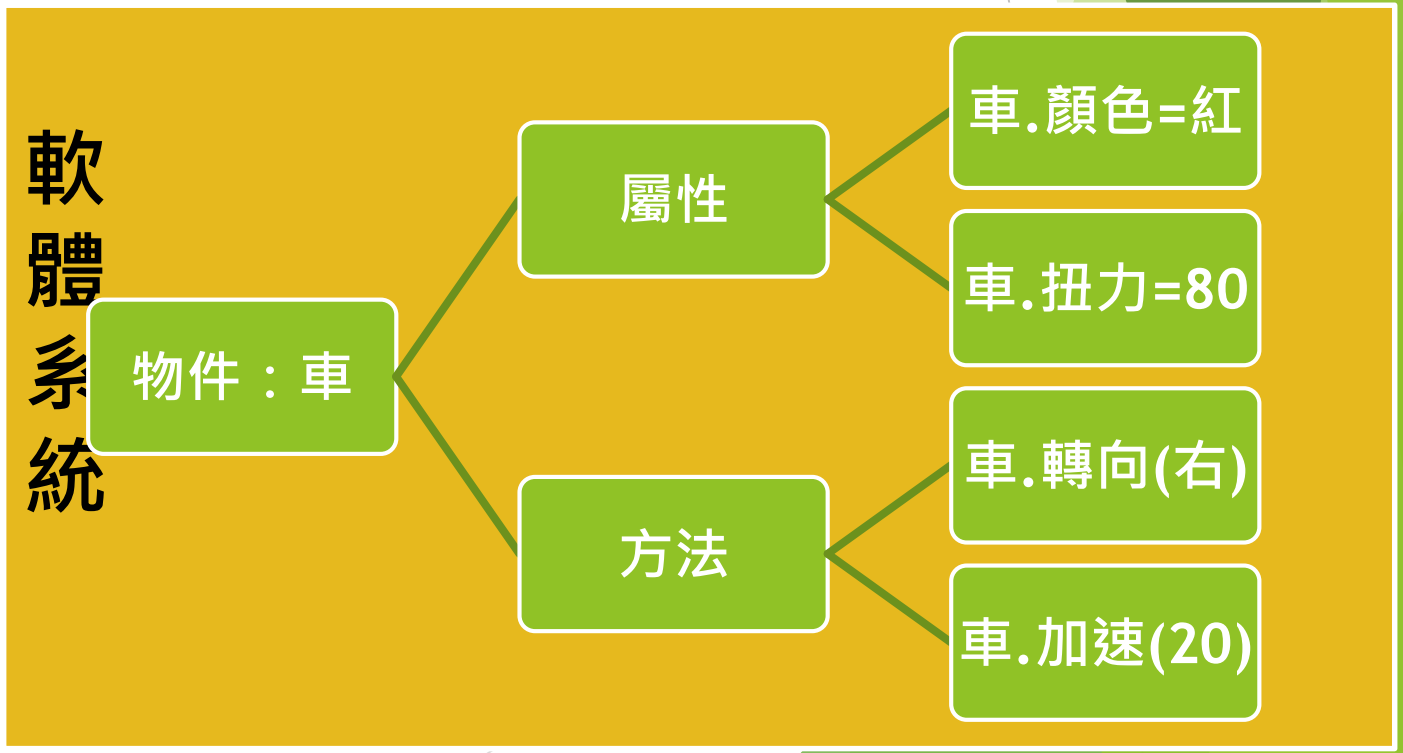
- 先自定義一個雙變數函數。
- 用plot_surface()函數劃出這個雙變數函數的3D曲面圖。

Part 11 物件(Object)導向程式語言

物件(Object)導向程式語言

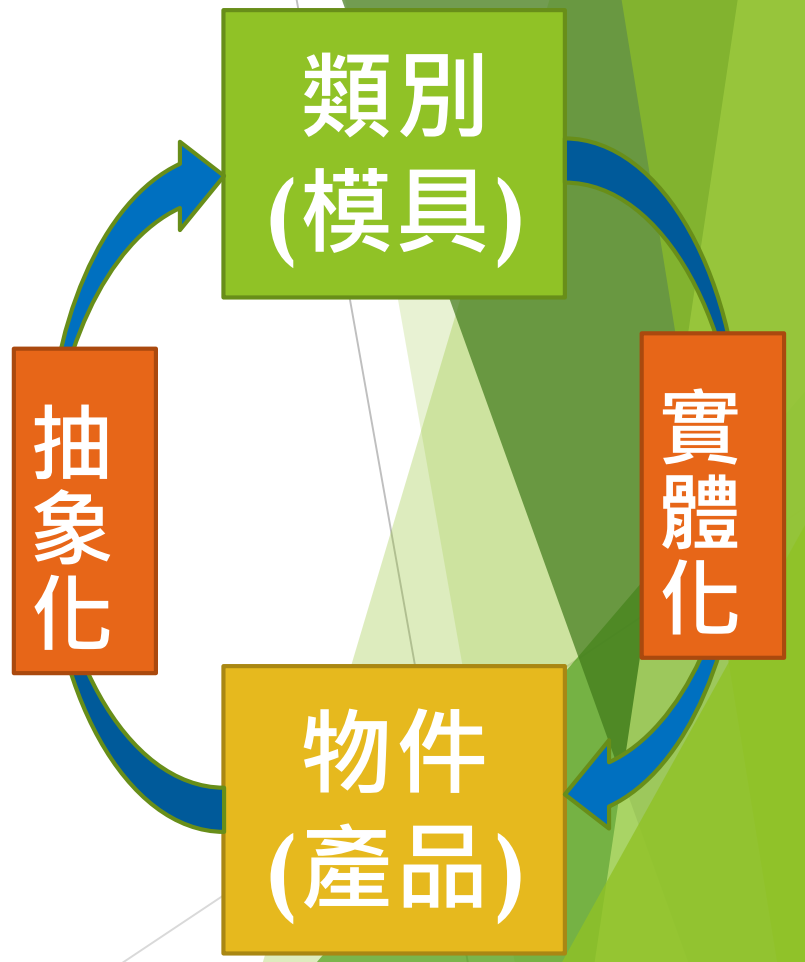
---模擬真實世界的運作方式，在軟體系統內實體化，解決問題。

- 傳統程序導向的程式語言，將程式看成一系列函式的集合，但在軟硬體日益複雜的情況下，程式難以維護。
- 物件包括一群獨立又互相呼叫的(屬性)變數、(方法)函式。
- 在Python中所有的資料、變數、函式都是物件。



類別(class)和物件(object)

- 類別(Class)用來描述具有相同的屬性和方法的物件，它定義了該類別中每個物件所共有的屬性和方法。
- 類別就像是可以快速、大量產生物件的模具。
- 每個物件都是某個類別 (Class) 所產生的一個實體 (Instance)產品。
- 類別裡可包含屬性 (Attribute) 及方法 (Method)
  - 屬性：儲存物件的資訊，也就是變數
  - 方法：操作物件的資訊，也就是函式



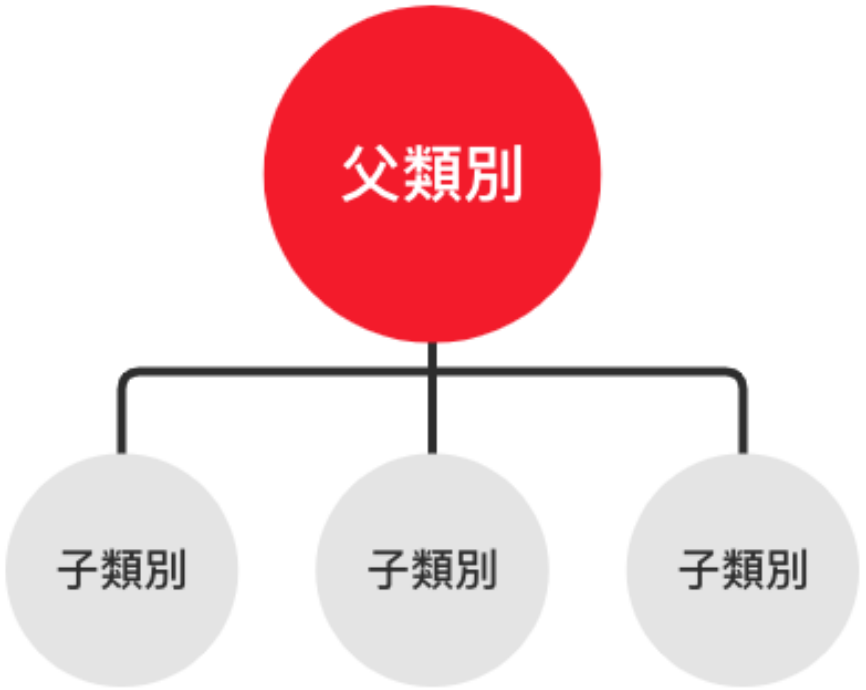
Python的類別實作

- Python中類別就是使用class定義
- 類別內的資料與操作的函式：
  - 建構函式「_ _init_ _()」表示宣告物件時會自動執行的函式
    - 第一個參數為self，表示自己
    - 第二個參數為輸入類別的資料
    - 傳入的資料可以指定給「self.變數名稱」，表示該物件有了儲存資料的變數
  - 其他函式建立方法和建構函式相同，惟名稱不是_ _init_ _()，且是呼叫時才執行
- 賦予物件屬性可以藉由類別變數、物件變數來儲存。

請看程式範例💻11-1

Python的類別繼承

- 繼承是物件導向程式設計的主要特性之一
- 當我們定義一個class的時候，可以從某個現有的class繼承
  - 新產生的class稱為子類別（Subclass）
  - 被繼承的class稱為父類別（Superclass）
- 繼承可以把父類的所有功能都直接拿過來，這樣就
  - 不必重零做起
  - 子類別只需要新增自己特有的方法
  - 可以把父類不適合的方法覆蓋重寫
- 語法
  - class 子類別名稱(繼承的父類別名稱)



Python的類別多重繼承

- 父類別 (superclass) 可以有多個，這是說子類別 (subclass) 能夠繼承 (inherit) 多個父類別，使子類別可以有多種特性。
- 多重繼承會以寫在最左邊的父類別優先繼承， 多個父類別如果有相同名稱的屬性 (attribute) 與方法(method) ，就會以最左邊的父類別優先。
- 語法

  class 子類別名稱(父類別1,父類別2,父類別3)：

請看程式範例🖳11-2