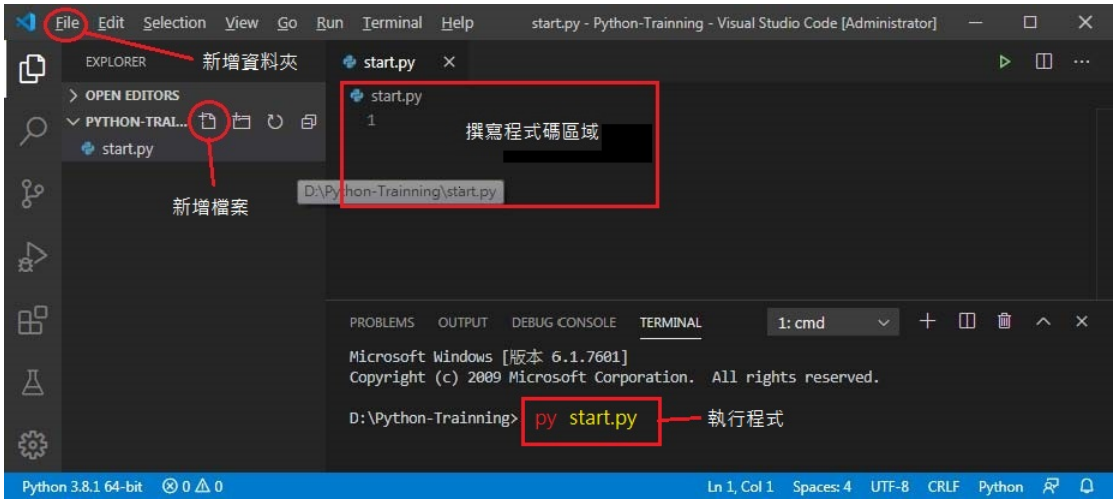
Python3 初接觸

一、前言

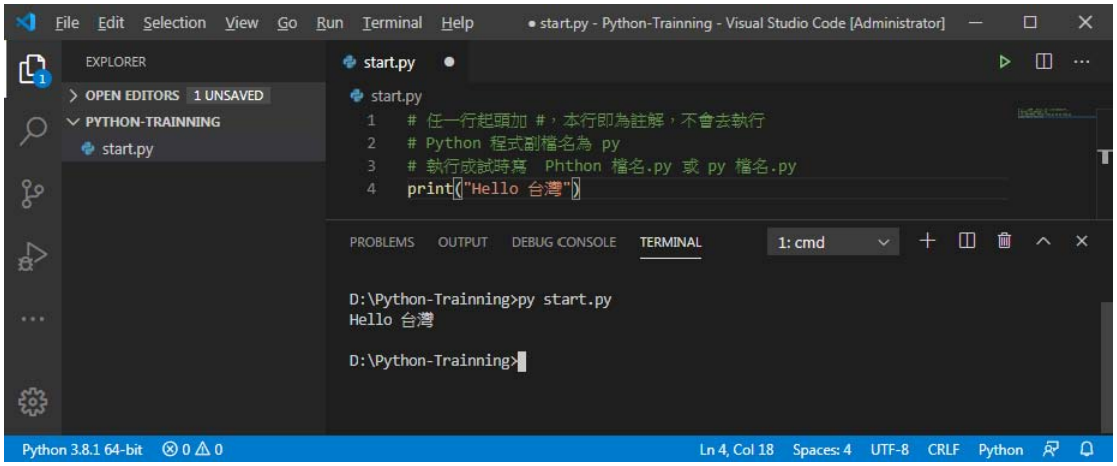
去年八月份在數位社一個活動課程中，洪駿逸技師講述由 ETABS 轉出資料來自動繪圖成為 ACAD 檔的成果，原來他是利用 Python3 程式來撰寫這種功能。第一次聽到這個軟體，據說 Python 比以前各種程式都好寫，而且基本上是開放式的，網路上也可以找到有許多人提供的副程式供公眾使用，這個軟體完全免費沒版權問題，也引起我的好奇心。不過卻隔一陣子才去下載來摸索，由於沒先上網去找教學影片，直接就打開程式執行檔，出現一個類似 DOS 文字模式指令欄之黑色畫面，完全茫然不知要怎麼去處理，不夠積極也就中斷學習，直到今年二月份才再去接觸。這次學乖了，先上網去搜尋 Python 教學，果然跳出一大堆教學網站及影片。嘗試幾個後，覺得有一位彭彭老師一系列的教學影片，解釋得還蠻仔細。初接觸時，就一系列看下去，然而就只是看卻懶得真動手去跟著操作，結果還是白看，只得到一個模糊概念。所以要學習還是必須一步一腳印，每看一集就應該做點紀錄才容易再回顧。

二、目前經驗及整理紀錄

1.Python 原來是 1991 才出現的泛用程式語言，各種情況都可以用，據說數據分析、人工智慧等都可以利用到它，但目前剛接觸只是想要踏進門檻一窺究竟而已。要學習首先得安裝編譯器，可上 Python 官方網站去下載。至於寫程式時的文字編輯器，教學影片是建議使用 Visual Studio Code，也跟著上官網去下載安裝，跟著學習比較方便。安裝 Python 時，新增 Python 到 Path 的選項建議要打勾，說是實際操作時會方便許多。啟動 Visual Studio Code 文字編輯器後，畫面如下。



第一步由左框處先點下拉式選單 File 新增一個檔案夾，然後再新增一檔案，副檔名要用 .py。當你打開該檔案，會在上列右框上半出現可撰寫程式之區塊，並會自動把你寫的每行程式依序編號。右半畫面可由最下往上拉出一個下框，當你試寫一小段程式碼存檔後，在下面標籤為 Terminal 畫面上，打上 **py 檔名.py** 就會自動執行所寫的程式了，如下:



如果程式碼中有任何錯誤，例如用不符規定的名稱或邏輯錯誤等，會停止執行程式，並指出錯誤發生在第幾行程式碼，方便你去除錯。用 Visual Studio Code 文字編輯器作業，還有一個好處是會用字體顏色來區分程式碼中各性值的字，例如註解就會使用綠色等。

2.設置好上述環境配置後，接著就是要去了解 Python 程式撰寫的一些觀念及指令了。依循彭彭君網路教學影片一一去學習及撰寫該主題練習程式碼，並跟著一步步去操作，覺得還算容易了解，但仍需要就所學到的觀念及指令留下一點紀錄，以利後續要撰寫程式時比較方便回顧。在程式碼中有需要做註解時，在註解文字前加上 # 該行自會變綠色，亦不會當成是程式碼執行。

a.首先學的是各種資料型態及數字及字串的基本運算：

配合撰寫的程式碼檔案：datatype.py　number-string.py　參考影片為第 2、3 / 26 集。

數字有　整數、長整數及浮點數(小數)。

字串可以是任意的文字內容　"Hello Taiwan"　'Hi Taiwan '　需加單或雙引號。

布林值　是一種表示　正確(True)　或　不正確(False)　的意思，用在判斷式中。

可變列表(List)　是有順序、內容可變動的資料集合　例如[3,4,5]　["台北","花蓮","台東"]　要使用中括號包著，中括號中各資料要以逗點區隔。

固定列表(Tuple)　有順序、但內容不可變動的資料集合　例如　(4,5,6) ("市長","局長","課長")　要使用小括號包著，其內各資料要以逗點隔開。

集合(Set)　沒有順序的一種資料集合　{6,100,35.2}　是使用大括號包起來，其內各資料間要以逗點隔開。

字典　這個資料型態比較特別，影片中稱它為　鍵值對(Key-Value Pair)　的集合，也就是此集合有一系列索引值對應到另一資料的功能，例如可利用來三號鋼筋對應其面積，或某型號鋼梁對應其 I 值等。影片中是以{"flower":"花","water":"水"}做例子。整個集合還是以大括號包著，其中各資料是成對的，如 “flower”對應到”花”，寫法是 “flower” :”花”，用冒號對應。各對資料間一樣是以逗點隔開。

變數(Variable)　大家比較孰悉，是可儲存資料的一個東西，其名稱可自訂。可使用字母或數字當名稱，但開頭不可以用數字。

各數字之基本運算如下：

加、減、乘 的運算跟 xls 中沒兩樣，符號 + - * ，先乘除後加減。

小數除法 7/3=3.5，**整數除法** 7 // 3=2 捨去小數位，即 2.5 只變成 2。

取餘數 % 7%3=1。

次方 ** X**Y 2**3=8。

在迴圈作業常會用到 x=x+1 x=x-1 等，在 python 可以有較簡之寫法，如下：

x=x+1 --> x+ = 1 。

x=x-1 --> x- = 1 。

x=x*2 --> x* = 2 。

字串連結可用 + 使其連起來，或接續寫兩個字串，也可以接起來，很特別。

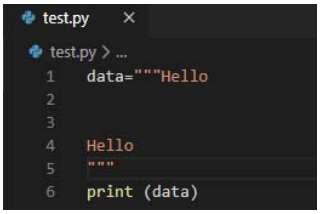
s="abc"+"def" t= "abc""def" s 跟 t 都是 “abcdef"。

s="abc\"c" 若字串要呈現特殊字元 ”，可以加 \ 來跳脫 即 s 可顯示成 abc"c。

s="Hello\nHello" 其中\n 是跳行符號，所以會有兩行 Hello。

s="""Hello Hello""" 用三個 " 或 ' 來寫的字串是很特殊的換行方式。

寫法如下截圖，將字串分幾行來寫，就會依所寫方式跳行顯示。



S="Hello"*3+"World" 寫法也很特別，等同於 "HelloHelloHelloWorld"。

重複相同文字，可用乘法。而且遵行先乘除後加減規矩。

字串中每一個字都有編號或叫索引，由 0 開始編起。

字串中的字元或子字串，可以利用編號(或叫索引)來取得。

s="Hello"

則 s[2] 為編號 2 的字元 就是第一個 l。

s[1:4] 則可取得第 1~3 之子字串 ell ，包含第 1 不包含第 4。

s[1:] 不寫結尾編號，則由第 1 取到結尾之子字串 ello。

s[:4] 不寫起頭編號，則由起頭取到第 4 前一個位置 Hell。

b. List、Tuple 有序列表的使用

配合撰寫的程式碼檔案：list-tuple.py 參考影片為第 4 / 26 集

有序列表在括號裡邊的資料有順序關係，也是由 0 開始編號。

grades=[12,60,15,70,90] 要用 中括號 闊起來，資料間 逗號 隔開。

grades[0]=55 把 55 去替換編號為 0 的資料，變成[55,60,15,70,90]。

grades[1:4]=[] 把位置 1~3 資料清除掉，上面 grades 變成[55,90]。

grades=grades+[12,33] 列表的串接，變成 [55,90,12,33]。

grades=[5,6,9,8]時，其資料編號是 0~3。

len(grades) 用 len 可取得列表的長度即資料個數。總長度就是 4 個資料。

Tuple 操作方法跟 List 完全一樣，但要注意 無法更動資料。

Data(3,4,5) Tuple 要用小括號。

c.集合&字典

配合撰寫的程式碼檔案：set-dictionary.py 參考影片為第 5/ 26 集。

集合係一群資料，用大括號括起來，沒有順序性。應用上重點在於，某資訊是存在或不存在此集合中。使用 in or notin 指令來判斷。

如 s1={3,4,5}　3 in s1 則為 True。

集合還有交集、聯集、差集 與 反交集 等運算方式。

使用 & 做交集運算，取得兩集合中相同的資料。

使用 | 做聯集運算，取得兩集合中全部資料，但不重複取。

使用 - 做差集運算。s1-s2 由 s1 中把 s2 有的資料都除去。

使用 ^ 做反交集運算，取得兩集合中不重複之資料。

set(字串) 前述指令亦可把字串拆解為集合。

字典基本觀念：如前所說明，字典是一種資料可配對的一種集合。

dic ={"apple":"蘋果","bug":"蟲蟲"} 需配大括號、分號及逗號組成。

上述字典 dic["apple"] 會顯示出 "蘋果" 其中 "apple" 就是 Key。

如同集合運算用 in or notin 可判斷資料是否在字典內，但只能判斷 Key 是否存在。Print("apple" in dic) 因存在，所以會印出 True。

dic["apple"]="小蘋果" 即可將字典中"apple" 所對應得值替換成"小蘋果"。

用 del 可刪除字典中的配對，如 del dic["apple"] 會把"apple"這 Key 刪除。

另外影片中也簡單提到利用 List 為基礎來建立字典，是比較特別之處。

dic{x:x*2 for x in [3,4,5]} 此例即利用[3,4,5]的集合及 x 對應 x*2 的對應公式來建立一個字典。跟下行直接寫出各 Key 及所對應的值，答案是一樣。

dic{3: 3*2 , 4: 4*2 , 5: 5*2}。。

d.流程控制 if 判斷式

配合撰寫的程式碼檔案：condition.py 參考影片為第 6/ 26 集。

if 判斷式，只要寫過程式的都很清楚，python 跟其他程式差不多，比較特別之處，在於使用冒號跟縮排來得知判斷式程式碼範圍，一般 if、elif、else 用法都一樣，但無需在最後加 endif。

```
if   x>200 :
     print ("大於 200")
elif   x>100 :
     print ("大於 100 小於等於 200")
else:
     print ("小於 100")
```

e.流程控制 迴圈基礎 while 迴圈 for 迴圈

配合撰寫的程式碼檔案：loop-basic.py 參考影片為第 7/ 26 集。

while 迴圈及 for 迴圈也是用冒號及縮排處理，基本語法如下例。

```
while n<=3 :
     True 執行命令
     n=n+1  (或 n+=1)
```

```
for x in [4,1,2] :
     print("逐一取得字串中的字元印出", x )
```

```
for c in "Hello":
    print("逐一取得字串中的字元印出", c )
```

使用 range() 是另一較特別的一種寫法。

```
for x in range(3) :
    print x
```
--> 等同於寫 for x in [0,1,2] :

```
for x in range(3,6) :
    print x
```
--> 等同於 for x in [3,4,5]:

◎ 練習時不小心遇到無限迴圈一直跑時，可用強制中止鍵 ctr+c 停止。

f.迴圈進階控制 與迴圈搭配的指令 break continue else

配合撰寫的程式碼檔案：loop-control.py 參考影片為第 8/ 26 集。

執行迴圈過程，可設定在某特定狀況時強制結束迴圈。

```
n=1
while n<5 :
    if n==3 :
        break          # 當 n 是 3 時跳出迴圈，n 也不繼續增加
    n+=1
print(n)               # 會印出值 n 值 3，因為 n==3 即跳出沒再增加動作。
```

```
n=0
for x in [0,1,2,3,4] :
    if   x%2==0 :      # 判斷 x 是否偶數
        continue       # 偶數會忽略 n+=1 所以 0 2 4 會跳出故 n 只加兩次
        print(x)       # 不是偶數的 x 被印出 即印出 1  3 兩個
    n+=1               # 遇到奇數才加
print(n)               # 最終會印出 n 為 2
```

```
n=1
while n<5 :
    print("變數 n : ",n)
    n+=1
else :                 # 最後跳出迴圈前會先執行 else 下的程式
    print(n)           #  n 已變成 5 才會結束迴圈，跳出前印出是 5
```

```
for c in "Hello" :
    print("字串字元", c)
else :                 # 跳出迴圈前會執行 else 下的程式
    print(c)           #  o 取完會結束迴圈，c 內容仍為 o，故會印出 o
```

◎ *在練習撰寫程式時，可以框一區域用 ctr+/ 一起把程式碼註解掉*

```
n=input("輸入一個正整數: ")   # 左例同時出現 break 及 else 之綜合應用
n=int(n)
for i in range(n) :
    if i*i== n :
        print ("整數平方根")
        break                 #用 break 強制結束迴圈，跳出前不執行 else
else :
    print ("沒有整數平方根")
```

g. 函式基礎：定義並呼叫函式

配合撰寫的程式碼檔案：function-basic.py 參考影片為第 9/ 26 集。

函式是程式碼包裝在一個區塊中，方便隨時呼叫使用。

首要先定義(建立)函式，然後才能呼叫(使用)函式。

定義含式之基本語法用 def。

```
def 函式名稱(參數名稱):      # 參數可有可無，需要時才加
    函式內部程式碼
```

下兩個都是定義一個印出訊息的函式：

```
def say():                  # 沒使用參數
    print("Hello")          # 程式寫印什麼就印什麼，用處不大
def say(msg):               # 參數為變數 msg
    print(msg)              # 依 msg 內容印
def add(n1,n2):             # 參數為 n1 及 n2
    rusult=n1+n2            # 可把參數拿出來做計算，放入變數中
    print(result)
```

呼叫含式之基本語法，只要直接打出

```
函式名稱(參數資料)
```

即可執行，例如執行此程式會去印什麼資料等，但若做程式不是要他回傳計算後資料，意義不大。

通常含式內之程式碼最後有回傳值 之指令，即有 return 指令，如下例：

```
def add(n1,n2):             # 程式 add 會利用參數 n1  n2 做計算
    rusult = n1+n2          # 如計算出加法，把結果放入 result 程式內變數中
    return "Hello"          # 程式碼中回傳值可以為任何資料，例如寫 "Hello"
value=add(3,4)              # 用參數 3 及 4 呼叫此程式並把結果放入 value 中
print(value)                # 會印出 回傳值 "Hello" 不是帶出 result 的值
```

另如果最後是只寫 return 指令沒寫資料，會僅結束程式，回傳值為 None。

較有意義使用回傳值之功能，應是用內部程式碼把計算結果用回傳值代出。

```
def add(n1,n2):
    rusult=n1+n2            #  result 屬程式內變數，其值要靠 return 帶出來
    return result
value=add(3,4)              # 執行 add(3,4) 回傳結果放入外部變數值 value
print(value)                # 即可以印出計算結果 7 比較有意義。
```

h.函式參數詳解：參數預設值、名稱對應、任意長度參數

配合撰寫的程式碼檔案：function-args.py 參考影片為第 10/ 26 集。

```
def 函式名稱(參數名稱=預設資料):    # 參數可以設定預設資料
    函式內部的程式碼
```

下例參數 msg 預設資料為"Hello"

```
def say(msg="Hello"):
    print(msg)
say()     # 有預設值時，呼叫若不寫 msg 值，即會印出預設資料"Hello"
say("Hello Function")     # 呼叫時若有填，就會印填入值"Hello function"
```

定義函式時，可以預設多個參數，用逗號隔開，預設值係視需要再設即可。

```
def   power(base,exp=0):
    print (base**exp)
```

```
power(3,2)        # 會計算 3**2 依參數位置順序帶入來計算
power(4)          # 沒填入第二參數，會用預設值計算 4**0
power()           # 因沒 base 的值又無預設值，會產生錯誤
power(exp=2,base=6)  # 計算出 6**2，會依參數名稱對應，次序就沒關係
```

另一較特別的主題是 無限參數 應該不是無限個，而是不預設幾個參數。

```
def 函式名稱(*無限參數):          # 無限參數名稱前要加上星號 *
    無限參數以 Tuple 資料型態處理  # 無限參數要用 Tuple 資料型態
    函式內部的程式碼               # 內部程式碼通常會用到迴圈
```

```
def say(*msgs):                   #  msgs 是 Tuple 資料型態
    for c in msgs:
        print(c)
say("Hello","Abritrary")          # 依序印出 "Hello","Abritrary"
say("a","b","c","d")              # 依序印出 "a","b","c","d"
```

```
def avg(*ns):                     # 利用此功能計算平均值例
    sum=0
    for n in ns:
        sum=sum+n
    print(sum/len(ns))
avg (3,5,13)                      # (3+5+13)/3=7.0
avg (2,4,6,8,10)                  # (2+4+6+8+10)/5=6.0
```

i. Module 模組的載入與使用

配合撰寫的程式碼檔案：module.py 及 geometry.py 參考影片為第 11/ 26 集。

模組是個獨立的程式檔案，將程式寫在一個檔案中此檔案可重複載入使用。

由 python 主程式中載入另一個程式，來使用該模組的函式或變數。

基本語法 要先 載入模組 相當於 載入副程式之意義。

```
import 模組名稱
import 模組名稱 as 模組別名      # 載入時可以順便簡化為模組別名
```

使用模組名稱或別名呼叫均可，呼教之基本語法有兩種：

```
模組名稱(或模組別名).函式名稱(參數資料)     # 記得有一個點 .
模組名稱或別名 . 變數名稱
```

Python 中有許多內建模組，影片只先介紹 sys 模組，可取得系統相關資料。

```
import sys              # 首先要載入模組
print(sys.platform)     # 可印出作業系統
print(sys.maxsize)      # 可印出整數型態的最大值
print(sys.path)         # 可印出搜尋模組的路徑
import sys as s         # 載入時可簡化名稱為 s
```

你也可以撰寫一些常用公式，自訂成為一個模組。

教學影片建立一個幾何運算模組為例，名為 geometry 模組。

模組中示範了兩點距離 distance 及 兩點斜率 slop 兩函式寫法。

```
def distance(x1,y1,x2,y2):
    return ((x2-x1)**2+(y2-y1)**2)**0.5
def slope(x1,y1,x2,y2):
    return (y2-y1)/(x2-x1)
```

上述函式是寫在 geometry 模組內，也就是存在 geometry.py 檔案中。
主程式要使用它時，需先載入再呼叫利用之。

```
import geometry                    # 載入所需之模組
result=geometry.distance(1,1,5,5)  # 呼叫 geometry 中 distance 函式
result=geometry.slop(1,2,5,6)      # 呼叫 geometry 中 slop 函式
```

主程式檔跟模組檔案有呼叫關係，若不小心會發生找不找得到路徑的問題。
python 內建 sys 模組中有 sys.path 函式，可去呼叫印出來看，其用 中括號 括起來的列表模式，載明著會按順序搜尋每一個資料夾。
當要呼叫的模組檔案跟主程式放在同一資料夾，當然找得到。不過為管理方便，通常會新建一專門放模組的檔案夾，例如稱為 modules。這樣就可能需要修正搜尋路徑，把新建的檔案夾也加入搜尋路徑中。使用的指令為：
sys.path.append("modules") # 把檔案夾 modules 加在前面搜尋路徑最後
上面指令中"modules"為相對路徑，也可以用絕對資料夾路徑來做新增。

j. Package 封包的設計與使用

配合撰寫的程式碼檔案：main.py __init__.py line.py point.py 參考影片為第 12/ 26 集。

封包是用來整理、分類模組。當模組非常多，例如 100 個模組，就有需要做此整理。檔案系統中的資料夾對應到封包，系統中的檔案對應到模組。

```
專案資料夾配置方式
    主程式.py          #  main.py
    封包資料夾         #  影片中封包資料夾名稱 geometry
        __init__.py    # 封包資料夾必須存在此檔案 注意是用兩條底線
        模組一.py      # 如 point.py
        模組二.py      # 如 line.py
```

基本語法：

import 封包名稱.模組名稱
import 封包名稱.模組名稱 as 模組別名
教學影片中把模組切割為 線 跟 點 包裝在封包 geometry 裡面

◎練習撰寫程式時，建議盡量用提示之名稱中去選要使用的字，因為練習時曾發現打的字詞明明沒錯，但不知何故卻發生錯誤。

```
import geometry.point
result=geometry.point.distance(3,4)  # 呼叫使用 line 這模組
print (result)
import geomery.line as line          # 也可以設定別名
result1=line.slope(1,1,3,3)          # 用別名就更簡便
print (result1)
```

封包其實還可以設很多層 只要邏輯一樣，有必要時再去試。

k.亂數與統計模組

配合撰寫的程式碼檔案：random-statistics.py 參考影片為第 14/ 26 集。

random 及 statictics 係 python 兩內建模組

```
import random                  # 載入亂數模組後有下列一些使用功能
random.choice([0,1,5,8])       # 從列表中隨機選取 1 個資料
random.sample([0,1,5,8],2)     # 從列表中隨機選取多個資料，不能超過總數
```

```
data=[0,1,5,8]
random.shuffle(data)        # 可將列中表中的資料「就地」隨機調換順序
print(data)                 # 印出 data 列表就可看到已掉換順序
random.random()     # 取得 0.0~1.0 之間的隨機亂數，其出現的機率相同
random.uniform(0.0,1.0)     # 意義同上取 0.0~1.0 間亂數，出現的機率相同
random.mormalvariate(100,10) #取得平均數 100 標準差 10 的常態分配亂數
import   statistics     # 載入統計模組後有下列一些使用功能
statistics.mean([1,4,6,9])     # 計算列表中數字的平均數
statistics.median([1,4,6,9])   # 計算列表中數字的中位數
statistics.stdev([1,4,6,9])    # 計算列表中數字的標準差
```

I.類別的定義與使用 Class Attributes

配合撰寫的程式碼檔案：test-class.py 參考影片為第 16/ 26 集。

類別是 python 中一種語法的結構，可以把變數或函式封裝在裡面，封裝在裡面的變數或函式，統稱類別的屬性。

先要定義(建立)類別,然後才能使用類別中封裝的屬性。

```
class 類別名稱 :          #名稱用英文或數字,首字不能用數字且習慣用大寫。
    定義封裝的變數
    定義封裝的函式
```

```
class Test :                # 定義類別
    x=3                     # 定義變數 x
    def say() :             # 定義函式 say()
        print ("Hello")
```

使用類別的方式如下

```
Test.x+3     # 取得類別 Test 中屬性 x 的資料 3 來做運算
Test.say()   # 呼叫類別 Test 中屬性 say() 函式
```

```
class IO :                          # 定義一個類別用來讀取及寫入
    supportedSrcs=["console","file"]  # 定義的變數是資源的來源，用列表
    def read(src) :
    print("read from",src)
IO.read("file")                     # 呼叫類別 IO 中的資料來源
```

上面案例"console","file"是資源的資料來源，即可由終端機或檔案讀取資料

m. 實體物件的建立與使用

配合撰寫的程式碼檔案：instance.py data1.txt data2.txt 參考影片為第 17 18/ 26 集。

實體物件是類別的另一種使用方式，蠻奇特的規定與功能。

定義一個類別，目的是為了產生一個實體物件，並利用實體物件來包裝實體的屬性，並去操作實體的屬性。

```
calss 類別名稱 :
    def __init__(self) :  # 定義初始化函式，透過操作 self 來定義實體屬性
obj=類別名稱()            # 呼叫初始化函式來建立實體物件放入變數 obj 中
```

```
class Point :
    def __init__(self,x,y)  # 這實體物件包括兩個實體屬性 x 和 y
        self . x=x
        self . y=y
```

```
p=Point(1,5)      # 把 1,5 包裝在實體物件內的 x y 屬性中就可取到來利
                    用資料，如 print(p.x+p.y) --> 1+5=6
```

```
class FullName:
    def __init__(self,first,last)
        self.first=first
        self.last=last
name1=FullName("S.Y","Chen")
print FullName(name1.first,name1.last) -->  印出  "S.Y Chen"
```

本主題前半段是談實體屬性(封裝在實體物件中的變數)，後半段則是介紹實體方法(封裝在實體物件中的函式)

```
class  類別名稱 :
    def __init__(self) :                  # 第一個參數固定為 self
        封裝在實體物件中的變數              # 定義實體屬性
    def  方法名稱(self,更多自訂參數) :      # 定義實體方法/函式
        方法主體，透過 self 操作實體物件
obj=  類別名稱()
```

```
class Point :
    def __init__(self,x,y) :
        self.x=x
        sefl.y=y
    def show(self) :
        print(self.x , self.y)
p=Point(1,5)                    # 建立實體物件
p.show()                        # 呼叫實體方法
```

n. 文字檔案的讀取與儲存

配合撰寫的程式碼檔案： file.py {}config.json data.txt

參考影片為第 13/ 26 集。

開啟檔案-->讀取或寫入-->關閉檔案 三個流程。

檔案物件=open(檔案路徑,mode=開啟模式) # open 是 python 內建程式

讀取模式 r 寫入模式 w 讀寫模式 r+ 其他模式暫略。

```
檔案物件. read()                    # 是一次讀取全部文字
for 變數 in 檔案物件:
    從檔案依序讀取每行文字到變數中
```

JSON 格式 python 也可以讀取。

```
import json                 # 但要先載入內建模組 json
json.load(檔案物件)         # 讀取 json 格式的資料
```

◎ 彭君有一影片專門介紹 json。網路交換資料常用到 json javascript 格式

```
檔案物件.write(字串)                    # 寫入或叫儲存檔案
檔案物件.write("這是範例文字\n")        # 可利用換行符號來一行行寫入
import json
json.dump(要寫入的資料,檔案物件)       # 將 json 格式之資料寫進檔案中
關閉檔案才能釋放資源
檔案物件.close()
```

```
with open(檔案物件,mode=開啟模式) as 檔案物件:
    讀取或寫入檔案的程式
```
此為最佳實務的寫法，以上區塊會自動、安全關閉檔案

```
file=open("data,txt",mode="w",encoding="utf-8")   # 開啟檔案
file.write("測試中文\n 好棒棒")                    # 寫入兩行 \n 是換行
file.close()                                      # 關閉檔案
```
◎ 要寫中文必須加 encoding="utf-8" 之參數，英文以外語言大致都要加。

```
with open("data.txt", mode="r",encoding="utf-8") as file :
    data=file.read()
print(data)          # 此寫法讀出 data.txt 內容印出，無需再 close 動作
```

```
sum=0
with open("data1.txt", mode="r",encoding="utf-8") as file :
    for nn in file :                                  # nn 是所讀字串
        sum+=int(nn)
print(sum)                    # 讀出一行行之數字字串轉成數字再加總印出
```

o. Pandas 資料分析 基礎教學 單維度資料 雙維度資料 篩選資料

配合撰寫的程式碼檔案：pandas-practice.py pandas-Series.py pandas-DataFrame.py pandas-filter.py 參考影片為第 23 24 25 26/ 26 集。

Pasdas 套件就是類似試算表的資料分析軟體，來對資做處理。

想使用 Pandas 套件，必須先做安裝動作。

在安裝 python 時，RIP 套件管理工具就已經安裝在電腦中了。

只要在命令列中輸入 pip install pandas 即可自行安裝。

Series 是單維度資料

```
import pandas as pd          # 安裝過 pandas 套件後即可在程式中載入
```

	A	B	C
1	John	18	
2	Bob	16	
3	Mary	20	
4	Andy	25	
5	Cathy	16	
6			
7			

如 xls 黃色區塊資料就是 Series

```
pd.Series(列表)                  # 以列表資料為基底，建立 Series
data=pd.Series(列表)             # 列表放進 data 變數中
data.max()                       # 找到最大值
data.median()                    # 計算中位數
data=data*2                      # 資料放大兩倍
```

這些都是 pandas 提供的資源。

DataRrame 雙維度的資料

	A	B	C
1	John	18	
2	Bpb	16	
3	Mary	20	
4	Andy	25	
5	Cathy	16	
6			
7			

如 xls 綠色區塊表格，有欄和列的概念

pd.DataRrame(字典)　# 以字典資料為底，建立 DataFrame
data=pd.DataFrame(字典)
data["欄位名稱"]　# 取得直向特定欄位
data.iloc[列編號]　# 列編號按順序由 0 開始編起，取得特定橫向列資料

實例操作:

```
import pandas as pd
data=pd.Series([20,10,15])
print("Median",data.midian())　# 印出此 Series 列表之中位數
TorF=data==20　# 比對此列表中各筆資料是否是 20，把結果放入變數中
print(TorF)　# 印出自訂名稱變數 TorF 內容 只第二筆是 T
data=pd.DataFrame({"name":["Amy","John","Bob"],"salary":[10,21,35]})
```

雙維度資料建立例，name 對應三個名字，salary 對應三筆錢，如下表

	name	salary
0	Amy	10
1	John	21
2	Bob	35

```
print(data["name"])　# 即可印出標籤為"nme"那欄資料
print(data.iloc[1])　# 即可印出索引為 1 的那列資料
```

==========

用 pandas 建立的 Series 會自動建內建索引，也可以自訂索引。

Pd.Series(資料列表,index=索引列表)

可以觀察此 Series 的 dtype(資料型態)、size(幾筆資料)、index(索引)。

```
Data=pd.Series
print(data.dtype)　# 印出資料型態
Print(data[1])　# 依順序編號取得資料
```

建好 Series 後，有下列幾種數字之運算之利用。

```
Data=pd.Series([3,10,20,5,-12])
print(data.sum()) , print(data.max())
print(data.mean()) , print(data.median()) , print(std())
print(data.nlargest(3)) , print(data.nsmallest(2))　# 列出前幾個最大最小值
print(prod())　# 印出全部相乘的值
```

字串運算則都定義在 str 底下，都有加一個 str

```
data=pd.Series(["你好","Python","Pandas"])
print(data.str.lower() , data.str.upper()) , data.str.len()) #變小寫大寫 字串長度
print(data.str.cat(sep=","))　# 把字串在一起，並用逗點做區隔
print( data.str.contains("P")))　# 判斷是否有大寫 P
print(data.str.replace("您好" ,"Hello"))　# 把"您好"改為"Hello"
```

DataFrame 一樣可以建立索引列表

data=Pd.DataFrame(字典,index=索引列表)

觀察資料有這些方式

```
Print(data.size)　# 資料數量，如 3 行 2 列，就是 6 個資料數量
Print(data.shape)　# 資料形狀，會告訴你是幾列幾欄
Print(data.index)　# 會印出索引資訊
```

取得列的資料　係整列的資料 列(Row) 有兩種方式。

其一是依據**順序**取一整列資料 由 0 起編號。

```
data=Pd.DataFrame(字典)  # 沒去訂索引，一樣會自動由編起
Print(data.iloc[1])      # 取出後相當 Series 型態
```

也可依據**索引**取得一整列資料。

```
data=Pd.DataFrame(字典,index=索引列表)
Print(data.loc[索引])    # Series 型態
```

取得欄(Column)的資料係根據欄位名稱取一整欄。

```
Print(data[欄位名稱])    # Series 型態
```

有下列兩種語法建立新的欄位。

```
data[“新欄位名稱”]=列表資料          # 直接用列表方式建立
data["rvenue"]=[50000,40000,30000]
data[“新欄位名稱”]=Series 型態資料   # 用 Series 型態資料建立
data["rank"]=pd.Series([3,6,1], index["a","b","c"])
# 要注意如果原來 DataFrame 有自訂 index，比須完全一致
# 建新欄位，其資例內容也可以如 xls 一般，由現有欄位計算得來
data["rank"]=data["rvenue"]/["salary"]
```

資料篩選之功能：

```
Series 跟 DataFrame 資料篩選要分開來討論
import pandas as pd
```

先建 Series 資料 data=pd.Series(列表)。

續建立篩選條件(個數與資料對應的布林值)。

```
condition[True , False , True]     # 表示只要第 1、3 資料，第 2 個不要
filteredData=data[condition]       # 即可根據條件完成 data 內之篩選
```

建立篩選條件(比較有意義的是直接透過比較運算產生)。

```
condition=data>5
filteredData=data[condition]    # 依各資料是否大於 5 做篩選
```

先建立 DataFrame 資料

```
data=pd.DataFrame(字典)
```

再建立篩選條件，有幾個列，就要有同數量對應的布林值。

篩選條件是透過特定欄位的比較運算產生，例如：

```
condition=data[欄位名稱]>5
filteredData=data[condition]
condition(True,False,True...)   # 要跟列的數量一致，False 的那列就整列刪掉
依條件來產生布林值是比較實務的做法
condition=data["salary"]>=40000   # 舉例用 salary 那欄數字跟 40000 比對的
condition=data["name"]=="Amy"   # 用 name 那欄字串跟 Amy 比對做篩選
```

3.前面是依據彭彭君公開分享之 26 集入門教學課程中，除了跟網路有關的第 15、19~22 集外，所介紹之資訊整理的摘要紀錄。對 python 程式已有一點初步概念。但還沒看到工程數學上，例如三角函數、對數、複數等之程式介紹，以及繪圖方面之資訊。網路上是有看到 NumPy 數值擴展包外掛套件，以及繪圖相關套件。但都還只大致去跟著瀏覽一遍，還要繼續去專心學習，才能開始去嘗試寫想要的程式。

三、結語

1. Python 屬於開放式的程式語言，又有許多前輩能提供教學影片或外掛套件程式，放在網上供大眾利用，學習練習使用方便多了，實在是感激在心。
2. 近日自學，覺得還是要挑一套好的教學影片或文件親自花時間慢慢磨，順便做點紀錄，主要是方便後續撰寫程式時之程式各指令功能回顧。也能順便分享給朋友快速瀏覽一下此程式入門大概會介紹那些東西。
3. 依據彭彭君入門教學影片過程，實際撰寫做演練的練習用 .py 程式檔案，也已跟著撰寫，將放附錄中，以利後續編撰程式時可以更細節來參考各主題指令及其功能。
4. 本文僅屬個人學習過程摘要紀錄，或仍有記載失誤之處。如想學習，建議仍應找一教學影片觀看學習才較務實。

四、附錄

1. https://www.youtube.com/watch?v=wqRlKVRUV_k&list=PL-g0fdC5RMboYEyt6QS2iLb_1m7QcgfHk
   點此連結可觀看彭彭君全部教學影片，看完第一部影片後，會自動依序繼續播放。影片視窗旁邊還列有後續各集教學影片的連結，也可以跳著去點看想先學的內容。

2. 演練時配合撰寫之 .py 程式檔案。配合用 Visual studio code 讀取較佳

   start.py
   datatype.py 及 number-string.py
   list-tuple.py
   set-dictionary.py
   condition.py
   loop-basic.py
   loop-control.py
   function-basic.py
   function-args.py
   module.py 及 geometry.py
   main.py 及 __init__.py 及 line.py point.py
   random-statistics.py
   test-class.py
   instance.py 及 data1.txt 及 data2.txt
   file.py 及 {}config.json 及 data.txt
   pandas-practice.py *pandas-Series.py pandas-DataFrame.py pandas-filter.py*