

Data Visualization(數據視覺化) with Python & JavaScript

~~※本教材不是以簡報為唯一目的，視為講義會更好！
為了方便初學者把握住重點，快速學習。
所以會補充較多字體較小的相關說明，請細細品嘗~~

Instructor: Jiun-Ting Jiang

Email: rjrj0510@gmail.com



上課，儘量往前座哦！

*Dept. of Computer Science and Information Engineering,
Tamkang University*

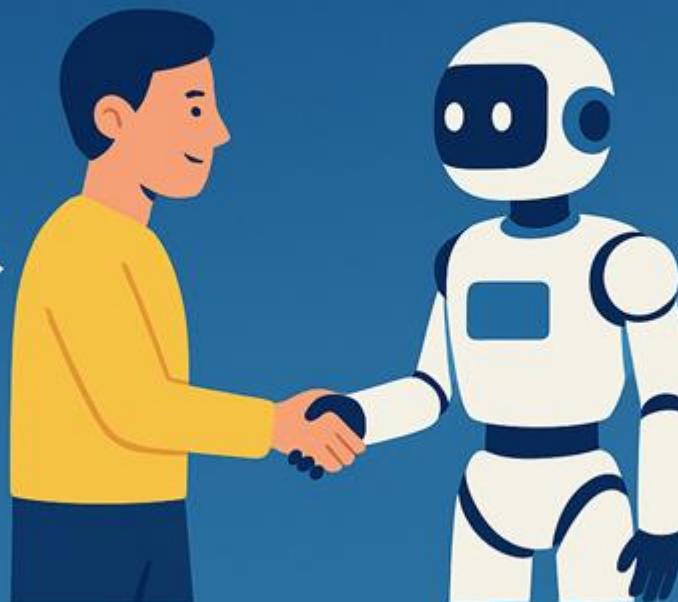
■ 在開始正式講課之前~~~

AI時代來臨~~

AI時代來臨，心態上有重大調整



不把AI視為敵人
而是夥伴
與AI共創價值

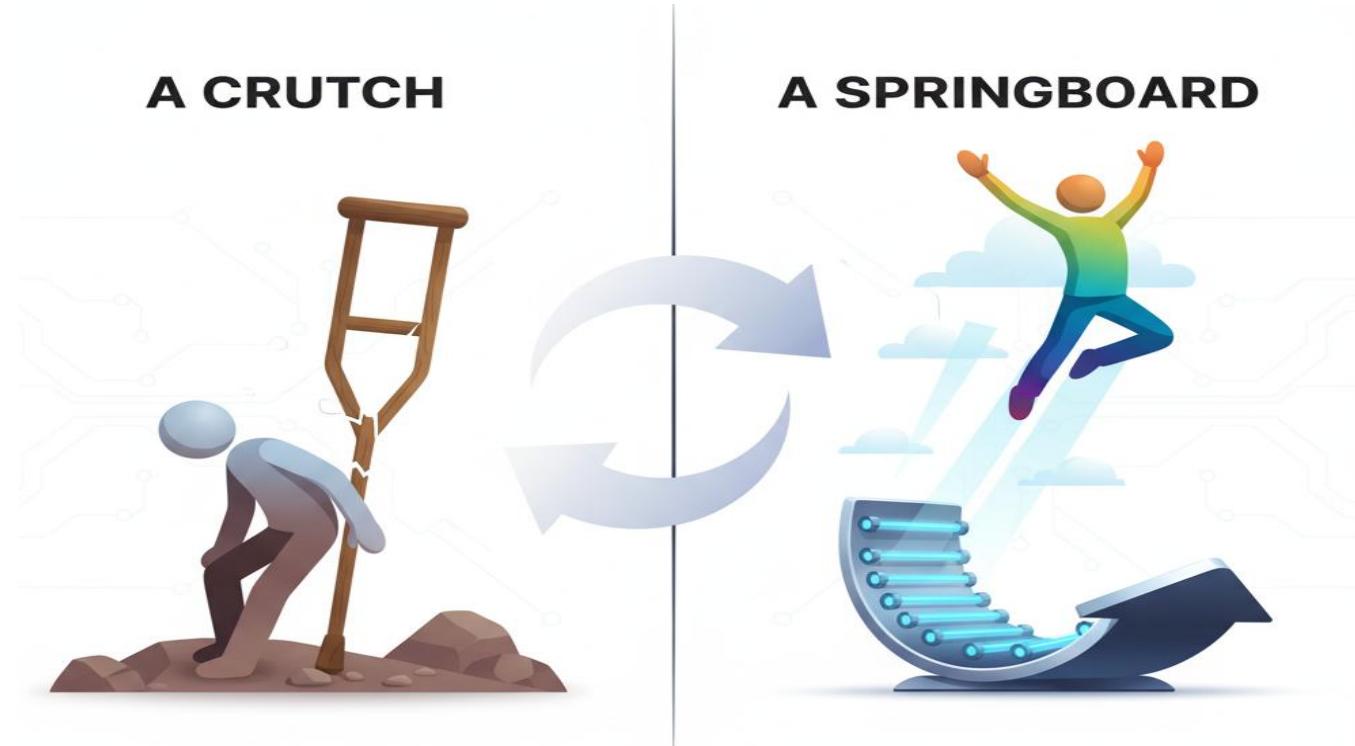


持續學習
成為常態！

AI會像工業革命
一樣，淘汰舊職位
但創造更多新工作

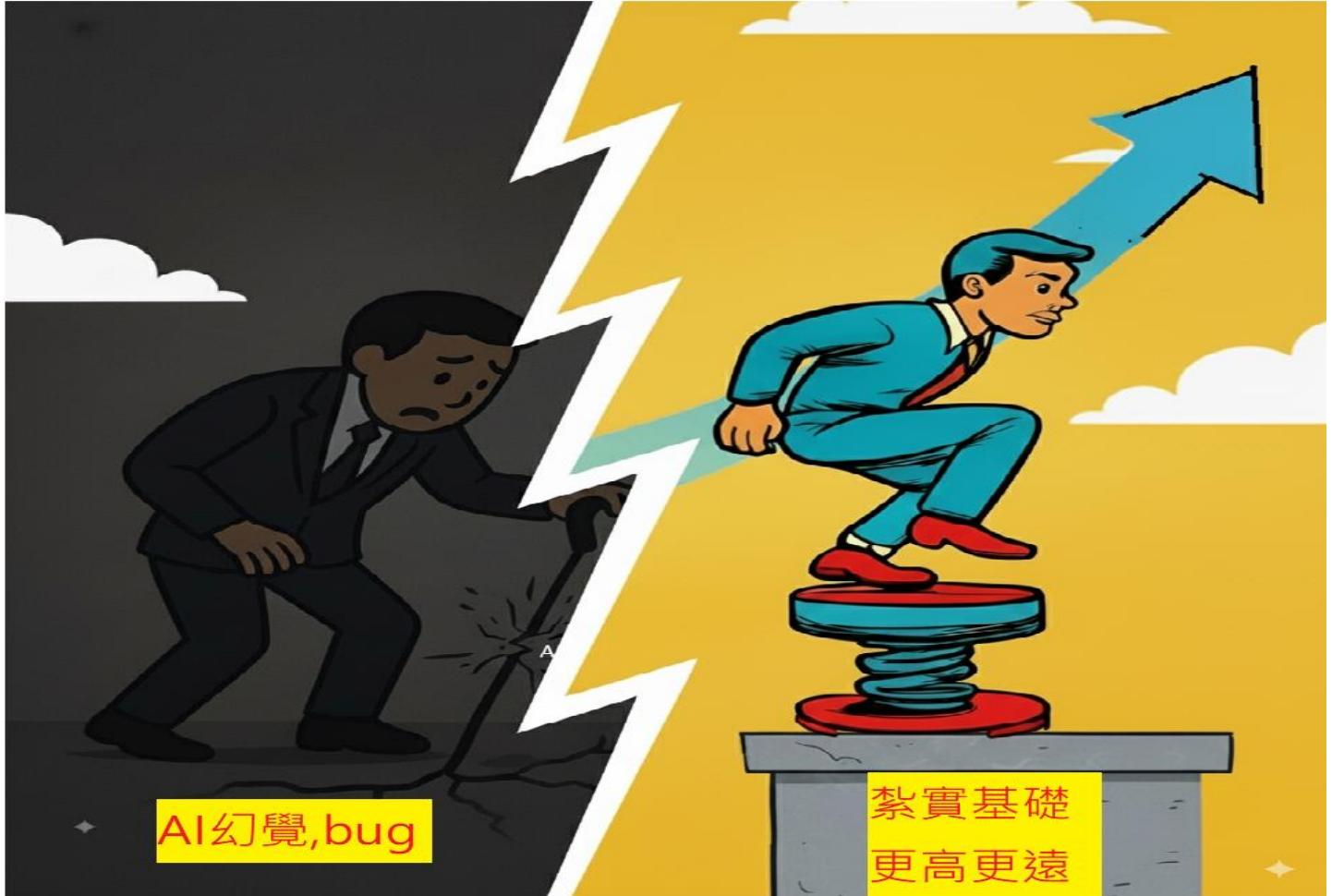
關鍵是擁抱變革，主動掌握AI工具
提升自身競爭力，迎接新機遇！

但是，AI是拐杖或跳板(a crutch or a springboard)



- AI正全球發熱中，AI工具是拐杖或跳板??
 - 若當作拐杖，則沒了AI工具，就寸步難行!
 - 個人「基本功」，與「該蹲的馬步」仍不可少！

希望AI是跳板~



- 擁抱變革，主動掌握AI工具，紮實基礎，迎接更高更遠新機遇！

資訊系基礎「CPE成績」：過去或本學期有答1題以上，期末前提供成績證明給老師，都有額外加分機會。

The screenshot shows a web browser window with the following details:

- Address Bar:** 大學程式能力檢定, CPE, Collegiate Programming Examination
- URL:** cpe.mcu.edu.tw
- Page Content:**
 - Header:** 大學 程式能力檢定
Collegiate Programming Examination
 - Language Selection:** 請選取語言
 - Powered By:** 由「Google 翻譯」技術提供
 - Section:** 簡介與規則
 - Links:**
 - [CPE完整簡介](#)
 - [大學程式能力檢定辦理要點](#)
 - [大學程式能力檢定考試規則](#)

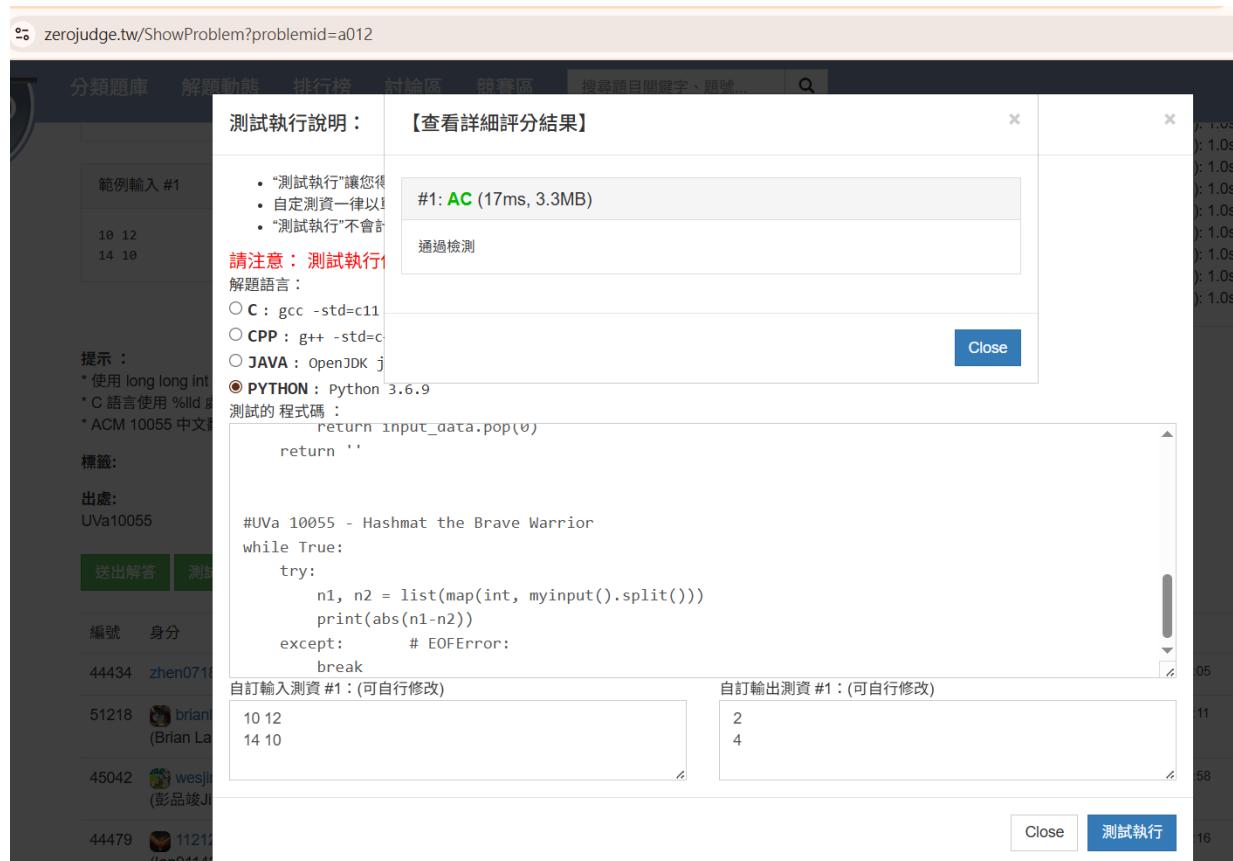
- CPE的目標是做為全台灣「程式能力檢定」的標準
 - CPE每年舉辦四次，跨校同步作業。
 - 報名費用：大專學生免費
 - <https://cpe.mcu.edu.tw/>

CPE一顆星選集49題_前兩題。

- prob1_uva10041: Vito's Family (維託的家人)
就是要找中位數並算出總距離 (★簡速版_可排序後，直接把「除除2」當中位數的正確位置)
 - Vito's family
 - a737. 10041 – Vito's family
- prob2_uva10055: Hashmat the Brave Warrior ★最簡單的相減絕對值
 - Hashmat the brave warrior
 - a012. 10055 – Hashmat the Brave Warrior

Lab#cpe_prob2_zerojudge測試AC的畫面： 此題(prob2)在zerojudge代號是a012

- 這是簡易測試，通過(Accept)的畫面。



Lab#cpe_prob2_瘋狂程設的畫面

※若想更真實練習，可用「瘋狂程設」

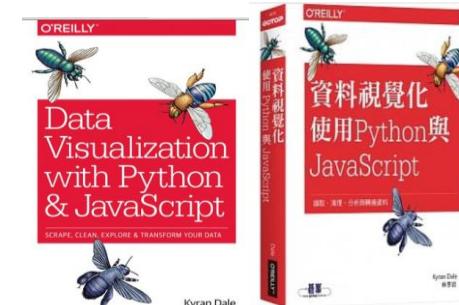
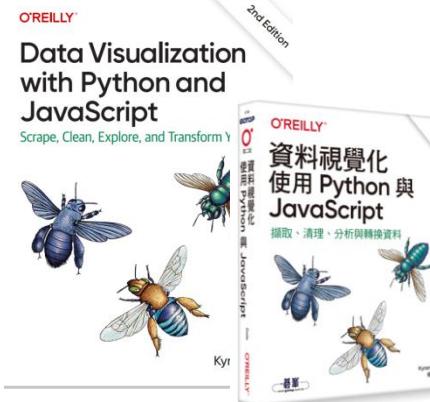
- 要使用本地程式(與考場版99%相似_但是會與jupyter相衝！)，測試完成，才可上傳到網路，就可由網路查詢，並且可以下載之前寫過的程式。

The screenshot shows a web browser displaying the 'coding-frenzy.arping.me' website. The page title is '瘋狂程設'. On the left, there's a sidebar with '同學功能' (Student Functions) including '在線' (Online), '同學' (Classmates), and '申請' (Apply). Below it are '課程列表' (Course List) and '加選課程' (Elective Courses). The main content area shows a list of programming problems under 'CPE49 共49題'. Each problem entry includes a '練習' (Exercise) button, a thumbnail image, a problem ID, a title, and a brief description. The first few entries are highlighted with blue boxes. The top right of the page shows the user's email 'rjrj0510@gmail.com'.

問題類別	題數	最後更新	問題名稱	描述
CPE49	共49題	2023-05-10 14:59	YKL49(UVA11321)	: Sort! Sort!! and Sort!!!
CPE一顆星	共96題	2023-05-10 14:59	YKL01(UVA10055)	: Hashmat the brave warrior
CPE二顆星	共28題	2023-05-10 14:59	YKL02(UVA10783)	: Odd Sum
CPE輸出入	共10題	2023-05-10 14:59	YKL03(UVA10812)	: Beat the Spread!

Textbook

- Kyran Dale, “Data Visualization with Python and JavaScript”, OReilly, 2017. (三隻蜜蜂！)
- 2ed, OReilly, 2023.



其他參考：

1. Dr. Pooja, Data Visualization with Python: Exploring Matplotlib, Seaborn, and Bokeh for Interactive Visualizations, BPB Publications (2023,7,11)
2. Cyrille Rossant, IPython Interactive Computing and Visualization Cookbook, 2ed, Packt, 2018.
※ 此作者，有公開9成的課文在網路上，也很值得參考。
https://github.com/ipython-books/cookbook-2nd/tree/master/chapter06_viz
3. 2. Sebastian Raschka & Vahid Mirjalili, "Python Machine Learning," (2nd edition), Packt, 2017.
4. 3. Igor Milovanović(英格蘭)等三人, "Python Data Visualization Cookbook", 2ed, Packt, 2015.
5. 4. Data Structures and Algorithms in C++, Michael T. Goodrich , Roberto Tamassia , David M. Mount, 2011.

Prerequisite

- Prerequisite
 - Programming language – Python & JavaScript Language.
 - 最好學過兩種程式語言(Python & Javascript)，可以較快了解本課程！
 - 應該要知道：
 - 可使用Python語言完成物件導向設計 (Class的設計與應用)
 - HTML5.0的骨架、CSS3.0、JavaScript、D3 (Data Driven Document)
 - 最好已經會簡單的數據爬蟲、清理資料、分析資料與呈現資料。
 - <若不曾學過Python與JavaScript語言基本語法，就要自己另外學習哦！在課程中，只有快速入門與複習！>
- Require the ability to program in Python & JavaScript
 - At least, write easy c++ or Pascal or Java and want to learn from now on.

Syllabus

- 教學大綱請參考教學計劃表(pdf)

資料可視化 dataViz的熱情 (dataViz, data Visualization)

- 在這堂課是理論、實作兼具！
 - 希望在完成課程後，所得到的不僅是「dataViz 的分數」，而且是「 dataViz的熱情」。
 - 由爬取資料、清理資料、分析資料、資料可視化。是大數據時代的重要技術哦！
-
- 實作上有問題，可Email，或在課堂上討論。
 - 希望這堂課後，會有很多收穫。

Grading

- Midterm Examination 30%
- Final Examination 30%
- Homework & Project 20%
- Quiz & Participation 15%
- Attendance 5%

- Other Bonus.

Course Work

- 兩次期考.(Midterm exam & Final exam) , 期考前一週，預計會有考前複習。
- 預計每章幾乎都有「實驗題」(LA, Lab. Assignment)。繳交期限(Due date)，各Lab在該章結束後，6天內繳交。信件主旨格式：[作業名稱#代號] 學號姓名.
- 若時間足夠，預計會有較大的PA (Project Assignment).
- 除了實作題，預計：每次上課都會有紙上作業，與2~3次的小考(Quiz).

- Bonus: Presentation. (choice topic today)
- 相信在實驗題完成後，會有很多的收穫。

LA, PA

- LA (Lab. Assignment): After you have completed the assignment of Lab, then , you export your project in an archive file, and send to my Email.
- PA (project assignment):

Chapter 1

Introduction for dataViz

數據爬蟲、清理、分析、呈現！

Outline

- 1-0a: 學Python太有趣了！(基礎認識)
- 1-0b: JavaScript的發展現況，學JavaScript很重要！
- 1-0c: python與JavaScript的距離？

- 1-1: ● Python overview
- 1.2: ● Objects in Python
- 1.3: ● Expressions and Operators
- 1.4 ● Program Structure
(程式結構)
- 1.5 ● Functions(函數)
- 1.6 Simple Output
(簡單輸出與輸入)

- 1.7 Exception Handling(例外處理)
- 1.8 Iterators
(疊代運算子)與Generators(生成器)
- 1.9 ●其他便利的寫法 Comprehension
- 1.10 Scopes and Namespaces(作用範圍和命名空間)-(※變數的生命週期、活動空間)
- 1.11 Modules and the Import Statement(模組區塊, 簡稱為模塊)

- 1-12: 如何快速--數據可視化？(以kaggle上執行iris為例)
- 1-13: 資料視覺化的工具鏈(The dataViz Tool chain)

因為時間關係，只介紹 6 節
，其他要靠自修哦！

1-0a 學Python太有趣了！(基礎認識)

■為何要學Python?

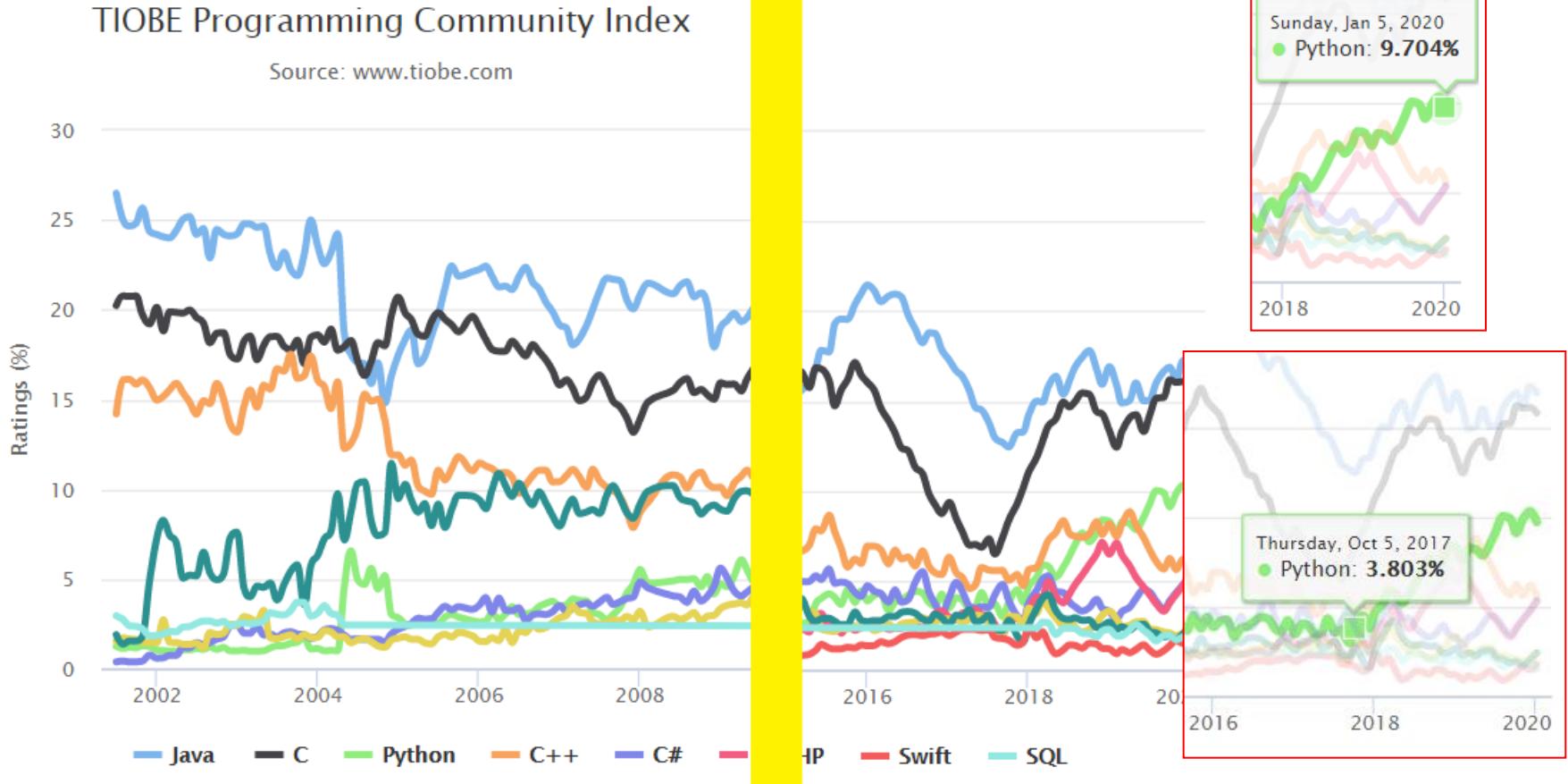
- 跨平台 (Cross Platform)
- 易學易用 (Easy Learning)
- 語法結構清晰 (Clear Syntax and Structure)
- 擴展與內嵌 (Extended and Embedded)
- 開放源碼 (Open Source)

光是跨平台，我就可以在”樹莓派、蘋果 (Mac)、Android 上開發應用與管理程式！

又是open source，可以永續發展

20年來程式語言佔有率

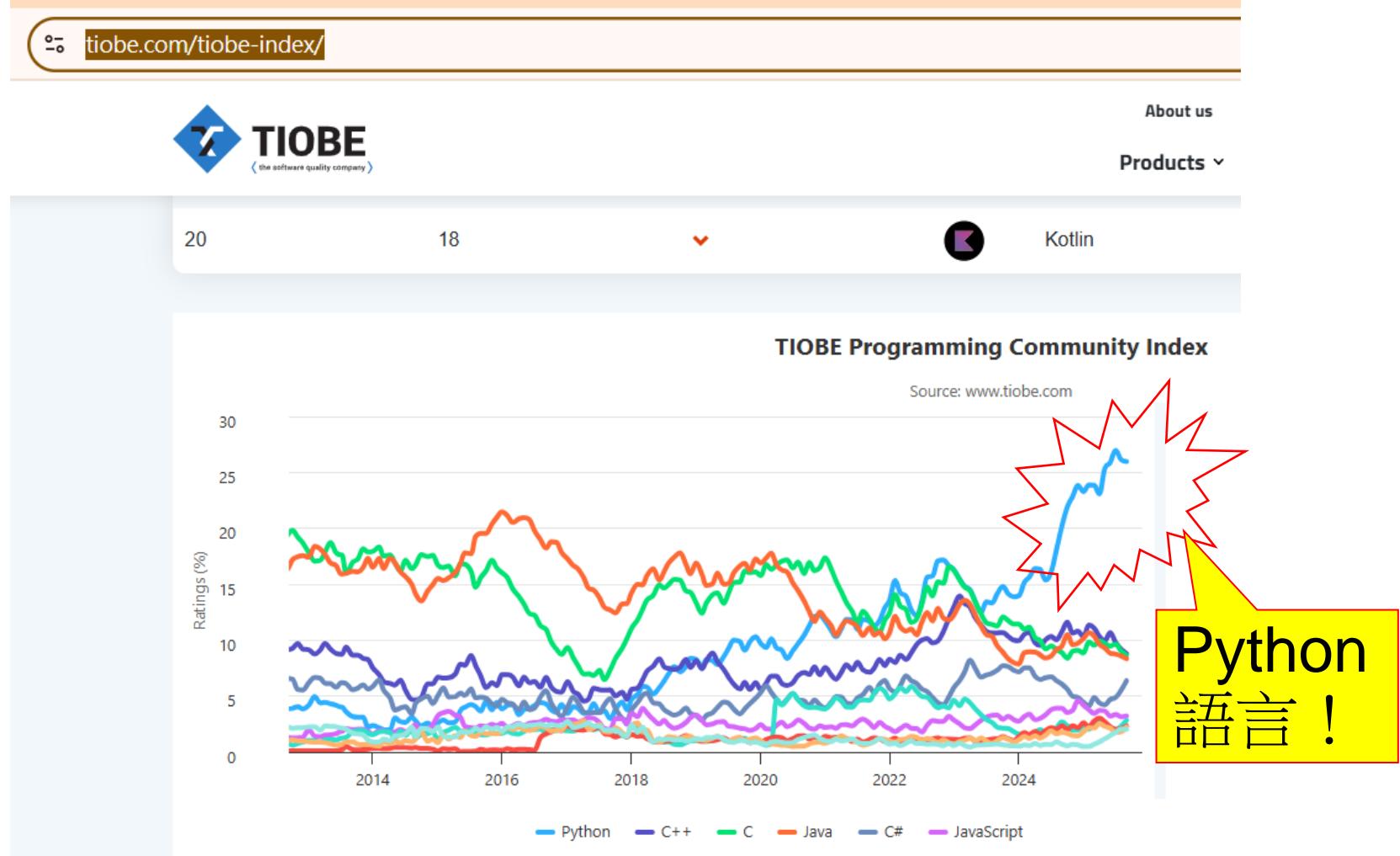
2017是3.8%, ,2020是9.7%! 幾乎是倍增！
那2021年呢？



■ TIOBE [https://www.tiobe.com/tiobe-index/programming community index](https://www.tiobe.com/tiobe-index/programming-community-index)

2024~25年唯一明顯人數大增的語言：

■ tiobe.com/tiobe-index/



眾多優點的一部份...

- 太多優點了：
 - 簡單易學、語法優美：Python 語言力求程式碼簡潔、優美
 - 豐富強大的Python 的類庫和開源專案，無論實現什麼功能，幾乎都有現成的程式庫可以使用。
 - 開發效率高，同樣程式，程式碼會精簡又清楚！
 - 應用領域廣泛：Python 語言的另一大優點就是應用領域廣泛，工程師可以使用 Python 做很多事情。例如，Web 開發、網路程式設計、自動化運維、Linux 系統管理、資料分析、科學計算、人工智慧、機器學習等等。
 - 可移植性、由於它的開源本質，Python已經被移植在許多平台上（經過改動使它能夠工作在不同平台上）。幾乎可以在市場上所有的系統平台上運行

Python 的各個優點是相輔相成的: ===>

因為「開發效率高」，所以「有很多豐富的程式庫」，也因為「有很多豐富的程式庫」，所以「開發效率高」

有點「雞與蛋」是誰生誰的關係！

比較明顯的缺點

- 1. 速度慢
- 2. 强制缩格 (**Indent**)
- 3. 功能太多，要學很久！

缺點算是很少的！

程式可不可以又要寫的快，又品質好？

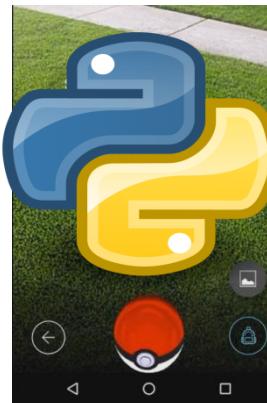
Yes.

-- >**Python+好的演算法！**

Python Shell

是最簡單互動式程設環境

- 進入Python Shell
 - >>> print ("hello")
 - hello



即使是python高手，會很多強大IDE，也不要小看python shell。

A screenshot of the Python 3.5.1 Shell window. The title bar reads "Python 3.5.1 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:

```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> print ("hello, 目前詢問度最高的程式語言：Python")
hello, 目前詢問度最高的程式語言：Python
>>> 9+2*10
29
>>>
>>>
```

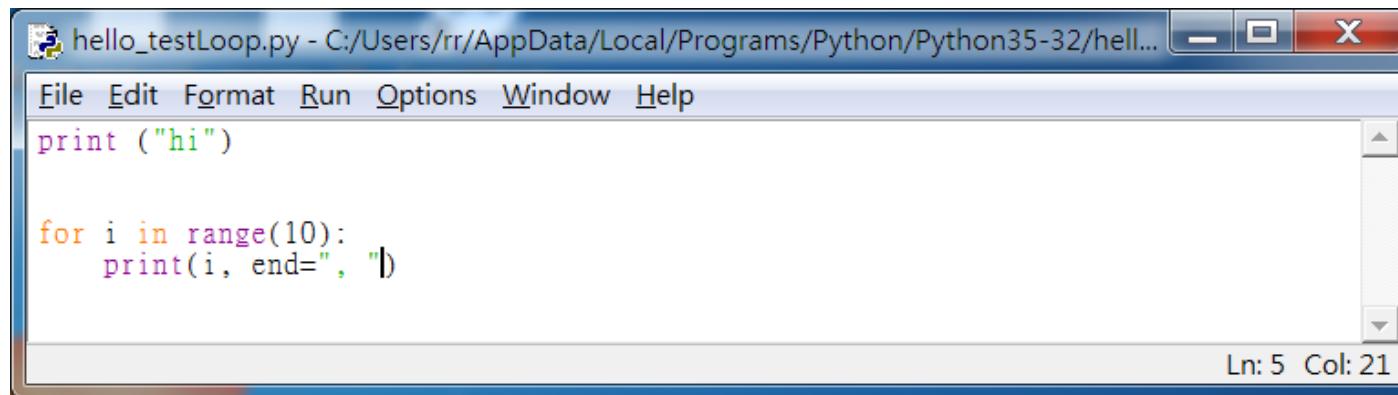
The status bar at the bottom right shows "Ln: 6 Col: 2".

Lab#1-0a: Python IDLE(Learning)

學習編寫程式檔與執行

- 編寫程式碼後按F5執行

編寫程式，會有適當提示，而且，如果夠熟悉，也會有行的除錯哦！

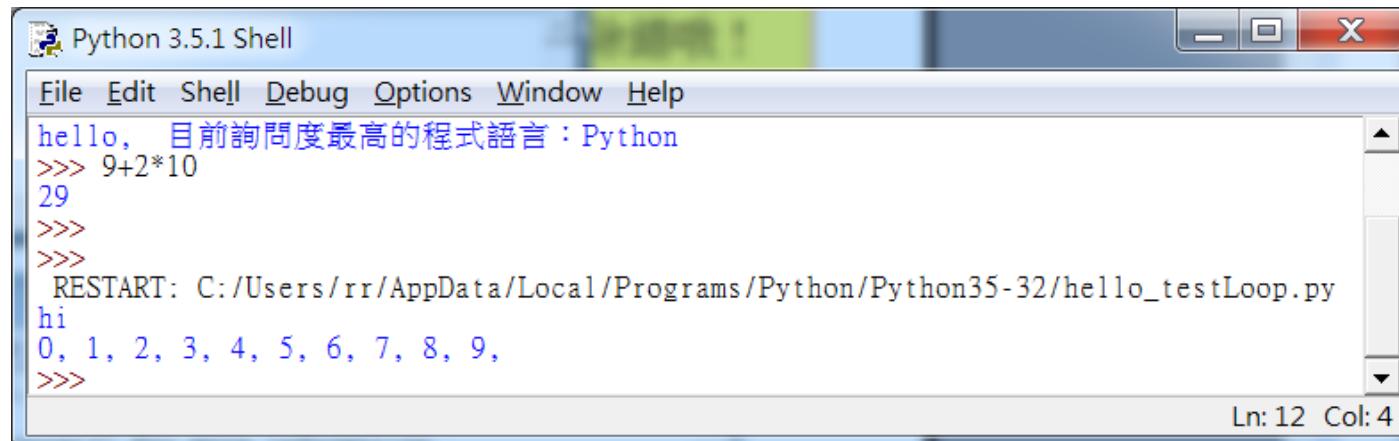


hello_testLoop.py - C:/Users/rr/AppData/Local/Programs/Python/Python35-32/hell...

```
File Edit Format Run Options Window Help
print ("hi")

for i in range(10):
    print(i, end=", ")
```

Ln: 5 Col: 21



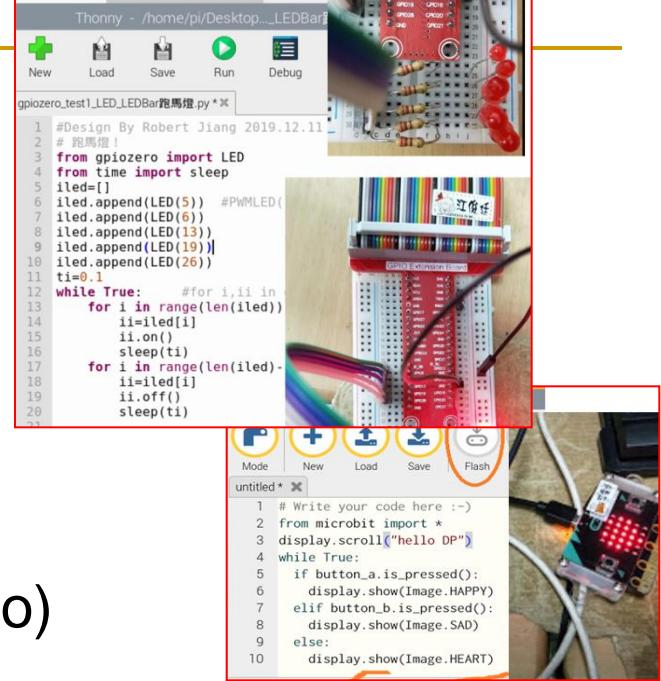
Python 3.5.1 Shell

```
File Edit Shell Debug Options Window Help
hello, 目前詢問度最高的程式語言：Python
>>> 9+2*10
29
>>>
>>>
RESTART: C:/Users/rr/AppData/Local/Programs/Python/Python35-32/hello_testLoop.py
hi
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
```

Ln: 12 Col: 4

Python近年為何這麼紅？

- 可低階硬體控制，例如：
樹莓派(Raspberry pi), microbit
- 普通的桌上圖形應用程式。
(ex: PyQt5 , Tkinter ...等)
- 手機程式設計 (apk)
- Web網站企業級的開發與維護。(Django)
- AI科學計算工具的接口。(OpenCV,
google的Tensorflow2.0)
- 爬蟲...
- 是一個靈活的粘合語言，可以通過C/C++
系統進行擴展，並能夠嵌入C/C++ 系統。



用途多
、樂趣多
、「錢途多」
，當然紅！

1-0b JavaScript的發展現況

- JavaScript 幾乎成為 Web 世界的標準程式語言：
 - 無論在你再怎樣不喜歡它，它就是在那裡，就是 **Number One**，所有瀏覽器唯一支持的程式語言。而這個世界，幾乎所有的軟體都在 Web 化
 - 除了網站之外，像是 Mobile App（Titanium, NativeScript, React Native）、網路遊戲（Web Game）、穿戴式裝置（Apple Watch）... 幾乎都朝著 Web 化的方向前進，可見得是值得投資在 JavaScript 的。

瀏覽器上的程式語言

- 1995年開始命名：
 - 最初命名為Mocha，1995年9月在Netscape Navigator 2.0的Beta版中改名為LiveScript，同年12月，Netscape Navigator 2.0 Beta 3中部署時被重新命名為JavaScript。
- 雖然目前瀏覽器JavaScript是主流，但是也不是沒有潛在敵人：
 - 最可能的潛在敵人那就是 ● Web Assembly (網路組合語言)
 - 不過 Web Assembly 還處於比較早期的階段
 - 自2017年以來，Web瀏覽器已支持WebAssembly(一種二進制格式)：該格式使JavaScript引擎能夠以接近本機速度的速度執行網頁腳本的關鍵部分。
 - 2019年12月 W3C(萬維網聯盟)已推薦 WebAssembly 與HTML,CSS和JavaScript是在瀏覽器上，可本機執行的第四種語言！
 - 因為僅以編譯形式出現在用戶電腦，這會“使惡意軟件檢測變得困難”，結果在使用情況分析後，竟然普遍使用的是惡意加密挖礦！而且還有其他安全的問題需解決。
 - ● JavaScript 是很值得學的！
 - 有人用其他語言寫，再轉成 JavaScript 才放上瀏覽器。（像是 CoffeeScript 就是一個長得有點像Python的語言，最後會轉成 JavaScript）
<http://coffeescript.org/>
 - 可看出：● JavaScript眾多優點中，最大的優勢是Web上面！

各種JS的應用場合

- 如果要開發手機板的瀏覽器應用 ● 可以採用 PhoneGap/Cordova 方案！
- 但，幾乎每年技術都有調整！



- 如果要開發手機遊戲 ● 可以採用 Unity 遊戲引擎搭配 Unity Script (一種加上型態標示的 JavaScript) 來寫程式.
- 其他node.js...

可以採用 Unity 遊戲引擎搭配 Unity Script
(一種加上型態標示的 JavaScript) 來寫程式



好用的 *.js 套件群

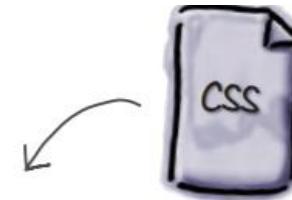
- 好用的套件，像是：**Showdown**: 將 markdown 轉為 HTML
- **MathJax**: 呈現 **latex** 數學式 **Highlight**: 為程式碼加上顏色！
- rlab 科學計算平台裡：
 - **lodash.js**: 基本常用函式庫
 - **numeric.js**: 線性代數矩陣函式庫
 - **jstat.js**: 機率統計函式庫
 - **algebrite**: 符號運算函式庫
- 前端繪圖套件
 - **C3.js (2D 繪圖)** – 這個用到 **d3.js** 的延伸套件。
 - **vis.js (3D 繪圖)**
- 讓程式編輯框顯示行號與顏色的 **CodeMirror.js** 套件
- 其他：深度學習技術專案**dnn.js**、決策樹、K-Mean、K近鄰法、SVM、分類分群等技術、人工智慧方法等等也都找得到專案！
- 不只**client**, 連**Server**端也可用 **JavaScript** 等。

JavaScript的運作原理

<html>



你已經知道我們可以使用 HTML 或 Hypertext Markup Language (超文字標記語言) 根據頁面的結構，像是段落、標題和章節，來指定它們的所有內容。

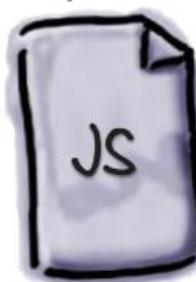


<style>



你已經知道我們可以使用 CSS 或 Cascading Style Sheets (疊層樣式表) 來告訴 HTML 如何呈現頁面的顏色、字體、邊框、邊距以及佈局。CSS 可以為頁面提供樣式，而且可以跟頁面的結構分離。

<script>



所以讓我們來瞭解 JavaScript 這個 HTML 和 CSS 善於計算的表親。JavaScript 讓你得以在網頁中建立行為。需要在使用者單擊你的 "On Sale for the next 30 seconds!" (接下來 30 秒開始拍賣！) 按鈕時做出回應嗎？需要即時檢查使用者的表單輸入嗎？需要從 Twitter 摚取推文並顯示它們嗎？那麼玩遊戲呢？此時 JavaScript 可以派上用場。JavaScript 為你提供了將程式加入頁面的方法，這樣你就可以進行計算、回應、繪圖、通訊、警報、修改、更新、變化等工作，讓使用者得以進行動態的操作，這就是 JavaScript 的行為。

要如何寫JavaScript程式？

```
<html>
<head>
<title>Icecream</title>
<script>
var x = 49;
</script>
<body>
<h1>Icecream Flavors</h1>
<h2><em>49 flavors</em></h2>
<p>All your favorite
flavors!</p>
</body>
</html>
```

撰寫

1

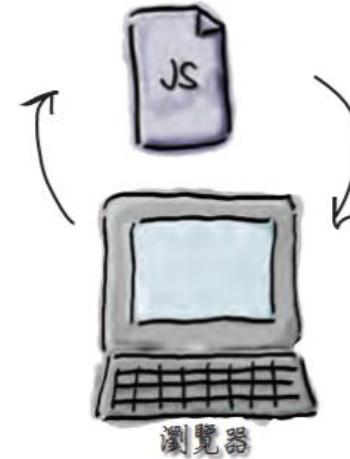
你可以像往常那樣，使用 HTML 的內容和 CSS 的樣式來建立你的頁面。你還可以在你的頁面中引用 JavaScript。如你所見，就像 HTML 和 CSS，你可以把這一切全都放到一個檔案，或是把 JavaScript 放到獨立的檔案，並在你的頁面加以引用。



載入

2

你可以像往常那樣，將瀏覽器指向你的頁面。當瀏覽器看到程式碼，它會立即著手解析它，以便執行它。請注意，就像 HTML 和 CSS，若瀏覽器在程式碼中發現錯誤，它將會盡其所能地繼續執行下去，並讀取更多的 JavaScript、HTML 和 CSS。它最不想做的事，就是無法讓使用者看到頁面。



執行

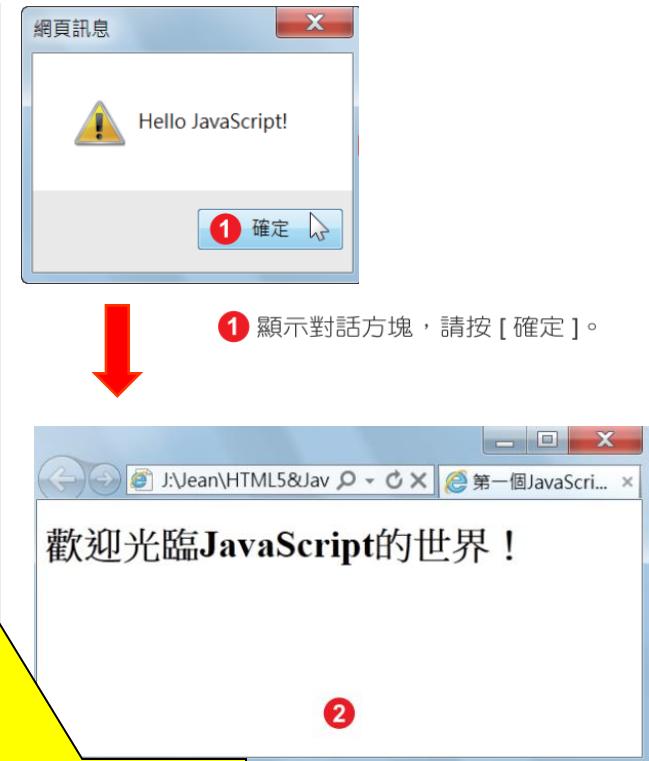
3

瀏覽器只要遇到頁面中的程式碼便會著手執行，而且會在頁面有效期間繼續執行。不同於 JavaScript 的早期版本，今日的 JavaScript 已是一個巨頭，採用先進的編譯技術，以近乎原生程式語言的速度來執行你的程式碼。

Lab#1-0b-1 撰寫第一個JavaScript 程式

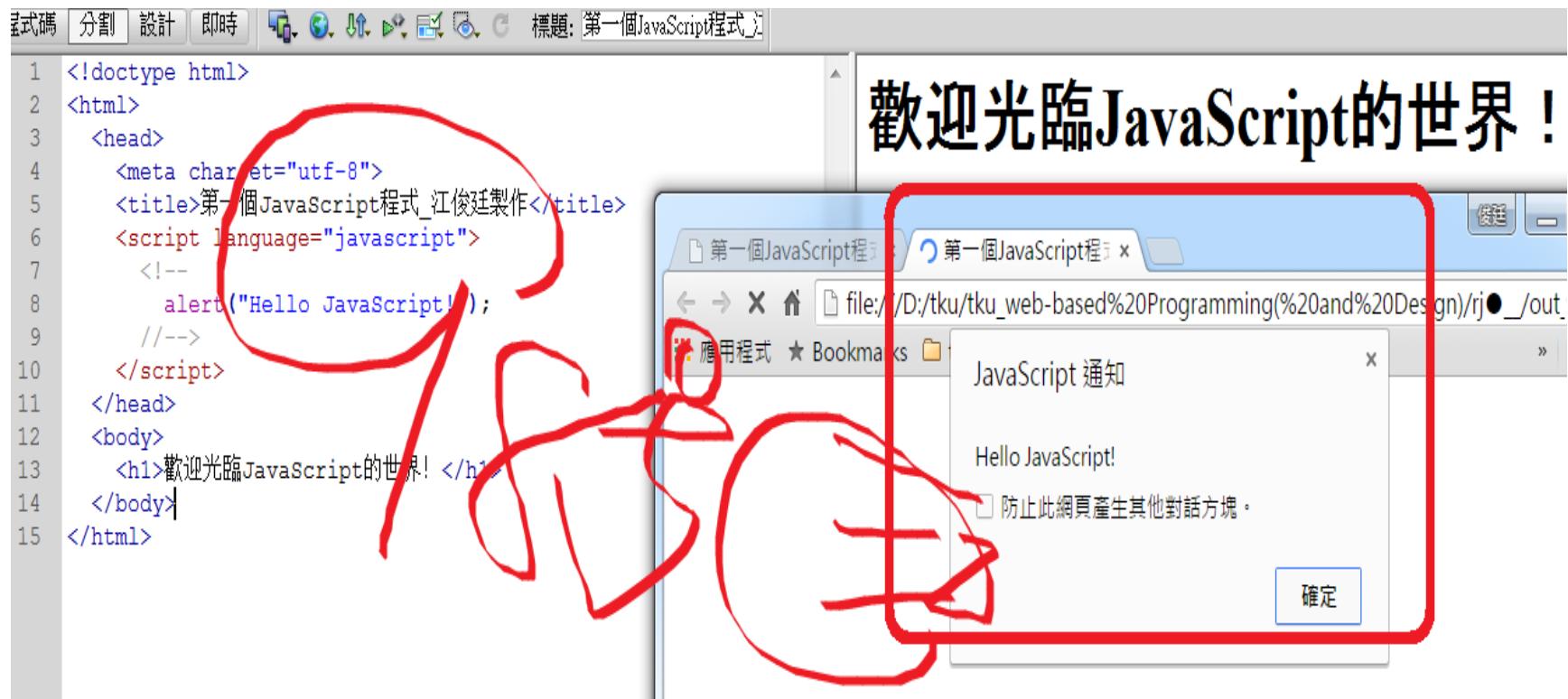
```
01:<!doctype html>
02:<html>
03:  <head>
04:    <meta charset="utf-8">
05:    <title>第一個 JavaScript 程式 </title>
06:    <script language="javascript">
07:      <!--
08:        alert("Hello JavaScript!");
09:      //-->
10:    </script>
11:  </head>
12:  <body>
13:    <h1>歡迎光臨 JavaScript 的世界！</h1>
14:  </body>
15:</html>
```

<script language=javascript>有常用的簡化
版寫法, 就是直接寫<script>即可.

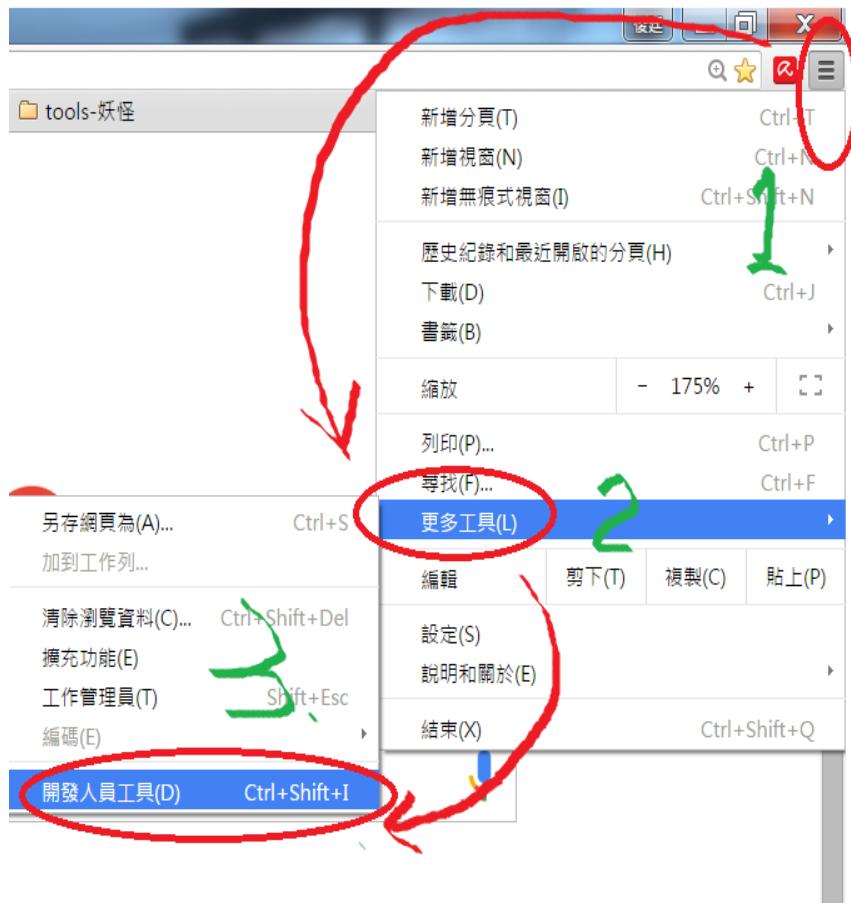


■Lab#1-0b-1 撰寫第一個JavaScript 程式.

- Alert()是學習Script時，最簡單的輸出訊息的方法。
- 他會產生新的視視窗來顯示結果。



※補充：Chrome新功能：開發人員工具(Ctrl+Shift+I) /同樣(Ctrl+Shift+I)再按一次就是關閉



在Console裡，
不但可以簡單實驗Script的指令
也具有很多相關的高級除錯的功能！
在此僅提供最簡單應用，讓初學者參考。

Lab#1-0b-2 數值的錯誤訊息用alert()快速進行 Script的實驗

- alert("abc"/3)



- alert(100/3)
- alert(-50/0)

※除了alert()，若不想產生新網頁，可以改為
console.log("2*4=%s", 2*4)
直接在console印出輸出！

The screenshot shows a browser developer tools console and several alert dialogs. The console log shows:

- < undefined
- > alert(10/0)
- < undefined
- > alert("abc"/3)
- < undefined
- > alert("abc"/3) https://www.google.com.tw 的網頁顯示：
 -Infinity
- < undefined
- > alert(100/0)
- < undefined
- > alert(-50/0)

Three alert dialogs are visible:

- "NaN" alert (top left)
- "Infinity" alert (top right)
- "-Infinity" alert (bottom right, with a red circle around the message)

The bottom status bar shows the time as "下午 01:11" and the date as "2015/11/26".

● 【check1】Lab#1-0b-3: 2乘4等於8在Chrome

← → C 檔案 | D:/tku/tku_dataViz(2020au)/js_入門/web_ch13_script/hello.html

應用程式 gMeet_遠端 Jcase_liu華_ML pyDataViz sphero py_DS micro:bit g5第五人格 H_Rina tools AI RWC tku(old)

歡迎光臨JavaScript的世界！

A screenshot of a Windows Notepad window titled "未命名 *". It contains the HTML code for "hello.html". The code includes a script block with a comment and two console.log statements. The second statement, which outputs "2*4= 8", is highlighted with a red oval.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>第一個JavaScript程式_江俊廷製作</title>
    <script language="javascript">
      <!--
        //alert("Hello JavaScript!");
        console.log(" 2*4= %d", 2*4)
      //-->
    </script>
  </head>
  <body>
    <h1>歡迎光臨JavaScript的世界！</h1>
  </body>
</html>
```

362 個位元組 (362 字節), 16 行 · HTML 行 9, 欄 37 UTF-8 不帶簽名

A screenshot of the Chrome DevTools Sources tab for "hello.html". The code is identical to the Notepad version. A red arrow points from the highlighted code in Notepad to the "Console" tab in DevTools, where the output "2*4= 8" is visible.

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>第一個JavaScript程式_江俊廷製作</title>
6     <script language="javascript">
7       <!--
8         //alert("Hello JavaScript!");
9         console.log(" 2*4= %d", 2*4)
10      //-->
11    </script>
12  </head>
13  <body>
14    <h1>歡迎光臨JavaScript的世界！</h1>
15  </body>
16 </html>
```

A screenshot of the Chrome DevTools Console tab. It shows the output "2*4= 8" from the previous screenshot. A red box highlights this output. Below it, the browser's rendered HTML code is shown, with a red box highlighting the "2*4= 8" part.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>第一個JavaScript程式_江俊廷製作</title>
    <script language="javascript">
      <!--
        //alert("Hello JavaScript!");
        console.log(" 2*4= %d", 2*4)
      //-->
    </script>
  </head>
  <body>
    <h1>歡迎光臨JavaScript的世界！</h1>
  </body>
</html>
```

實作畫面參考

The screenshot shows a Jupyter Notebook interface with three main panes:

- Left pane (Launcher):** A sidebar with a tree view of notebooks. A red box highlights the "check" node under "Lab#1-0b-3: 2乘4等於8在Chrome的console" in the "PYJS_CH01_第一段內容__預備知識_C0..." notebook.
- Middle pane (Code):** A code editor showing Python and JavaScript code. It includes a snippet of JavaScript at the top and a larger block of Python code for generating an HTML file with embedded JavaScript. A red box highlights the Python code block.

```
var people = "Alex";
var years = 42;
console.log(` ${people} is ${years} years old.`);
```

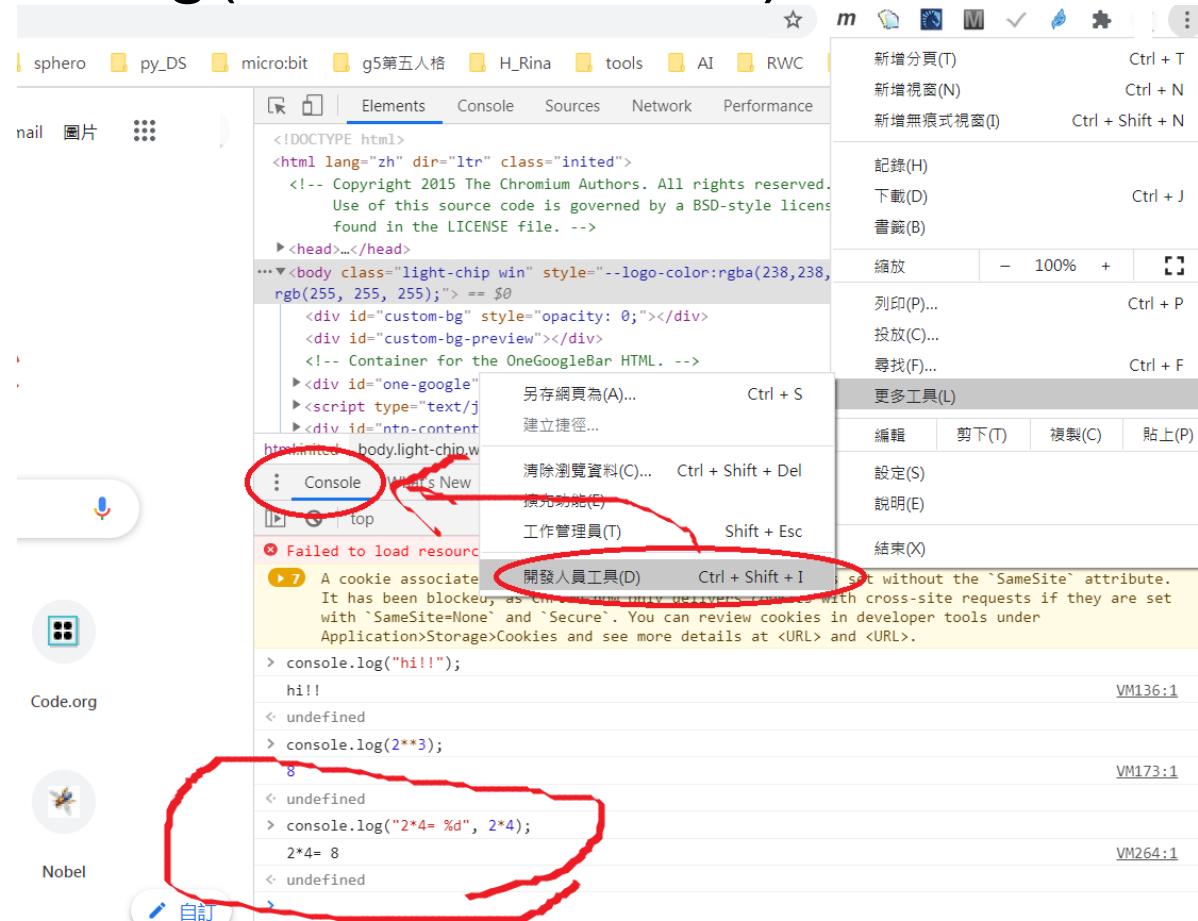
```
[31]: ## htm想在jupyter顯示的實驗
from IPython.display import display, HTML

# 測試基本 HTML 這章
test_html = """
!doctype html
<html>
<head>
<meta charset="utf-8">
<title>第一個JavaScript程式_江俊廷製作</title>
<script language="javascript">
<!--
//alert("Hello JavaScript!");
console.log(" 2*4= %d", 2*4)
//-->
</script>
</head>
<body>
<h1>歡迎光臨JavaScript的世界！</h1>
</body>
</html>
"""
```
- Right pane (Console):** A browser's developer tools console. A red box highlights the "Console" tab. The console output shows the result of the JavaScript alert and the Python log message. A red bracket groups the two alerts, and another red bracket groups the log message and the alert message.

```
DevTools is now available in Chinese
Don't show again Always match Chrome's language
[Console]
Default levels No Issues
2*4= 8
> alert(123456789/2)
< undefined
> alert(123456789/2)
```

Lab#1-0b-4: 2乘4等於8在Chrome的console
(只要有chrome，不必先寫程式，可直接實驗！)

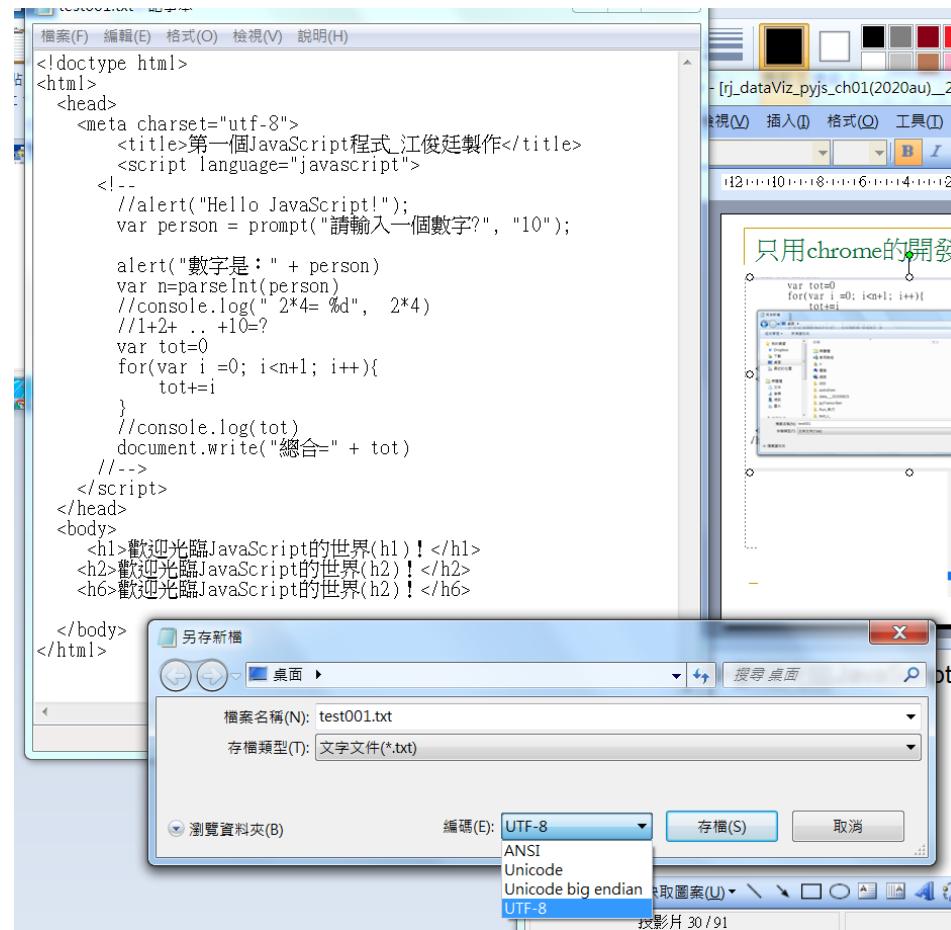
- `console.log("2*4= %d", 2*4);`



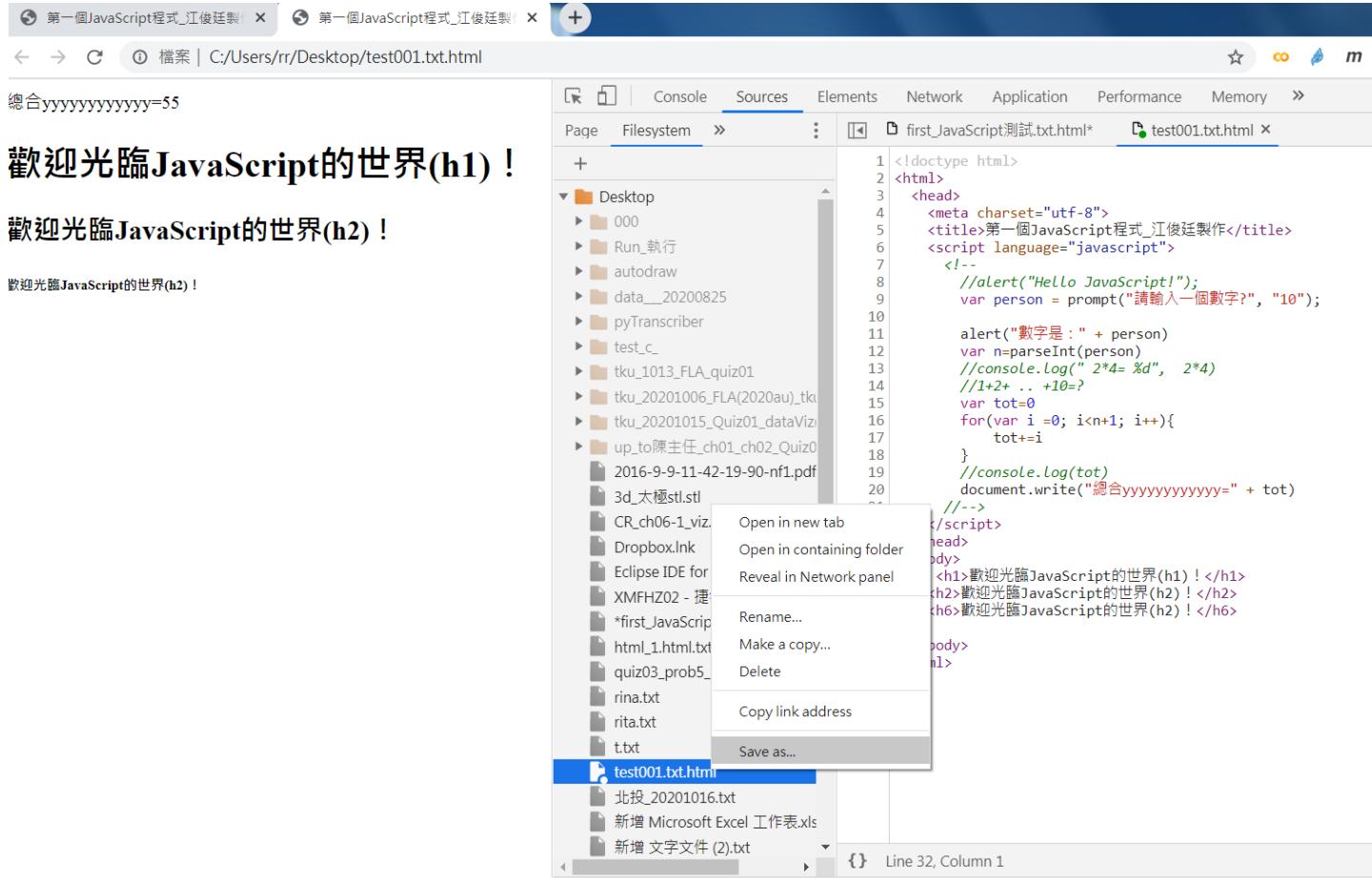
Instructor: Jiun-Ting Jiang

Lab#1-0b-5:只用chrome的開發人員工具寫JavaScript

- 文字檔，第一次要另存為 utf-8 或unicode否則可能中文會有亂碼！



Lab#1-0b-6: 檔案要 save..as 後，才可重新整理看最新網頁結果



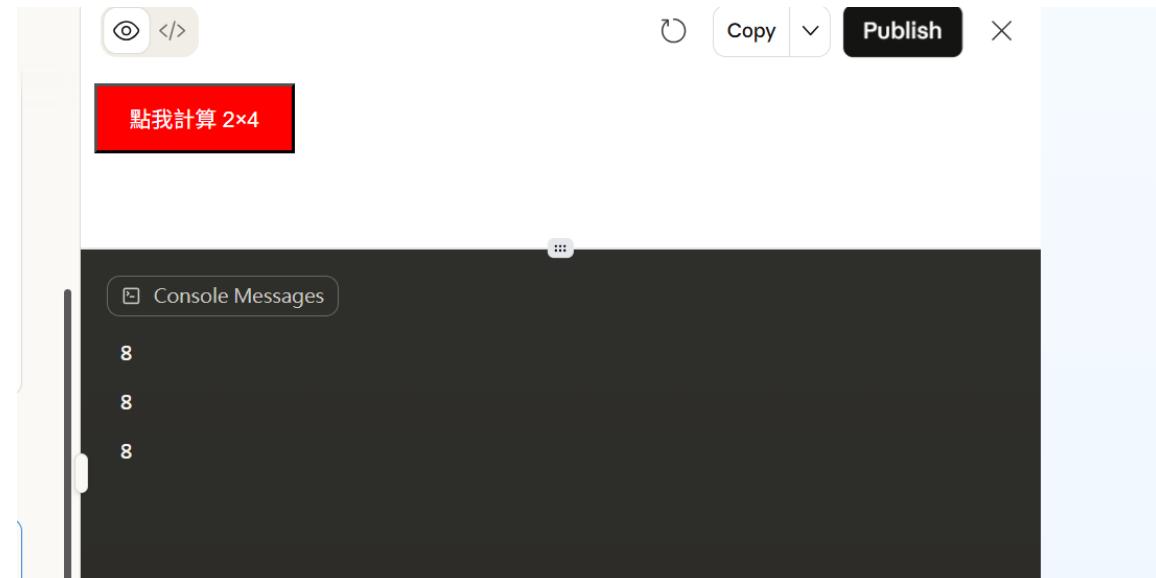
Exercise 1-0a, 1-0b

- Ex1-0aAdd1: Python有那些優點？有那些明顯的缺點？

- Ex1-0bAdd1: JS的最大優勢是那方面？可能的潛在競爭者是？簡單說明。
- Ex1-0bAdd2: JS裡的d3.js可以做到很多繪圖功能，但是這個函式庫更強調互動性介面。若要更完整的「3d資料」呈現，可考慮採用vis.js，因為會更專注於繪圖但互動性就沒那麼好。不過，一般情況，如果有3種資料要呈現，是否仍可用d3.js來呈現？
- Ex1-0bAdd3: 相對於JavaScript, 實驗中的WebAssembly在速度和隱蔽性都會更好，但是，是否也因此會產生其他問題？可以想到，很可能會被發展的惡意程式是什麼？
- Ex1-0bAdd4: AI時代來臨，心態上應該要有那些重大調整？

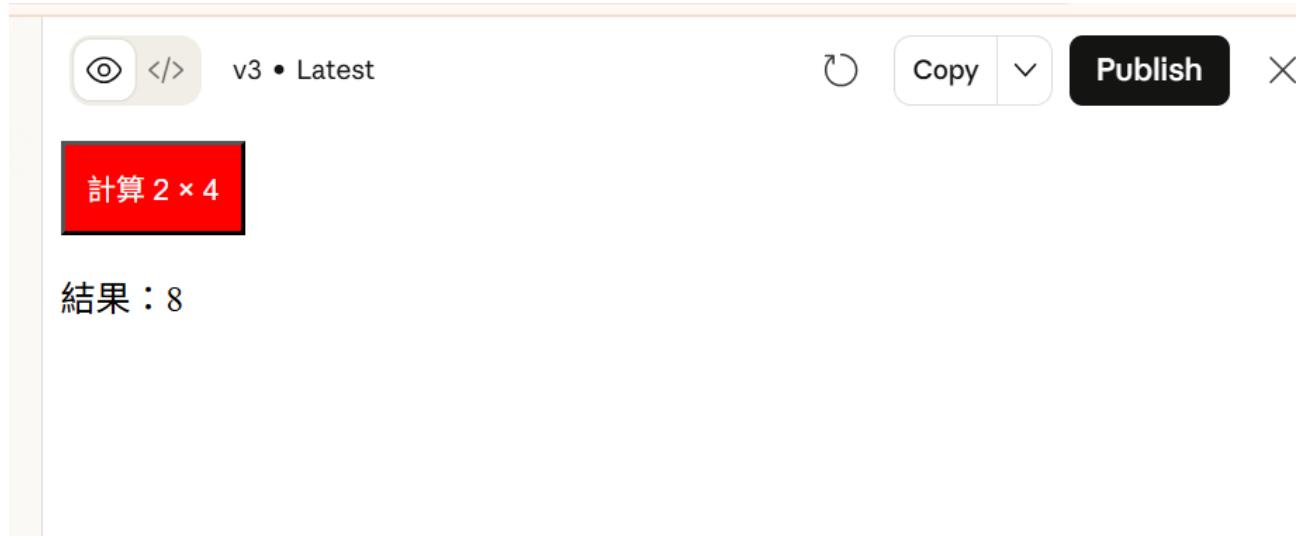
AI提示詞1a_輸出要放在console

- AI提示詞1a:html+css+javascript初學者練習：按鈕背景用css設為紅色，按鈕可執行「 $2*4$ 」的計算，並且顯示結果(輸出要放在console)。(注意：儘可能簡單，幾乎都3~5行，而且html, css,javascript放在單一檔案中，讓新手容易了解)



AI提示詞1b_輸出不要放在console

- AI提示詞1b:html+css+javascript初學者練習：按鈕背景用css設為紅色，按鈕可執行「 2^4 」的計算，並且顯示結果(**輸出不要放在console, 而且document.getElementById()要採用指定簡短id的方式來寫**)。(注意：儘可能簡單，幾乎都3~5行，而且html, css,javascript放在單一檔案中，讓新手容易了解)



參考寫法：輸出不要放在console, 而且document.getElementById()要採用指定簡短id的方式來寫

- <!DOCTYPE html>
- <html>
- <head>
- <style>p{color:red;}button{background:yellow;}</style>
- </head>
- <body>
- <p>這是紅色文字</p>
- <input id="n"><button id="btn">問候</button>
- <p id="msg"></p>
- <script>
- btn.onclick=()=>msg.innerText="Hi "+n.value;
- </script>
- </body>
- </html>



這是紅色文字



Hi aaa

同時有html, css, js的html實驗



JavaScript 初學者練習

輸入你的名字，按下「問候」；也可用計數器練習事件與狀態。

輸入你的名字，例如：江俊廷

問候（顯示 Hello）

清除

-

7

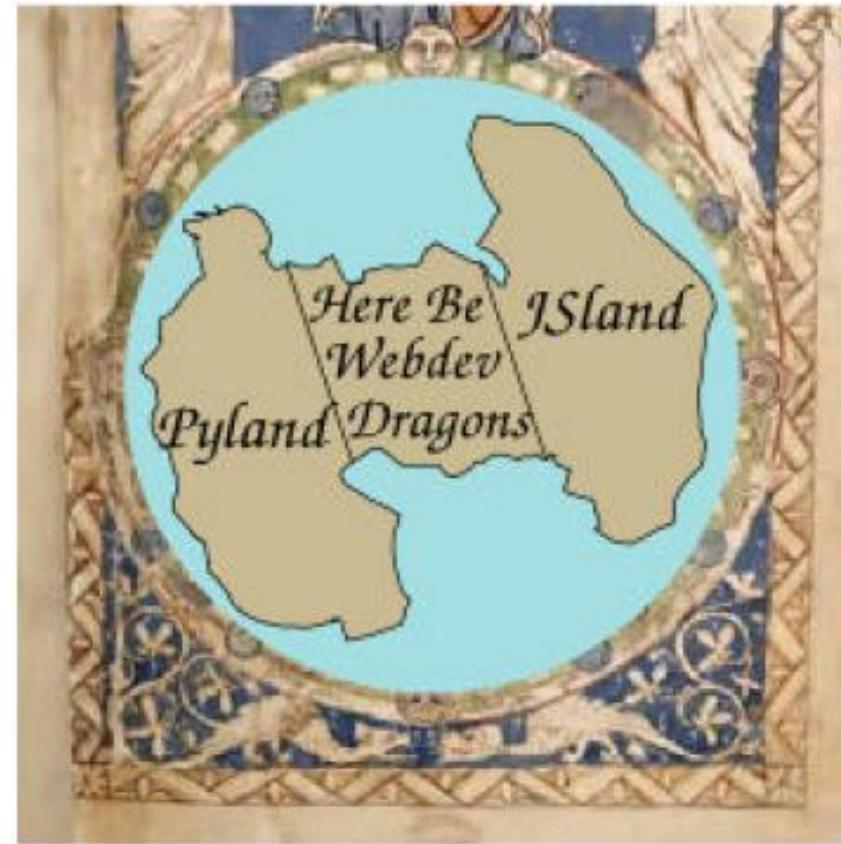
+

計數器：按 + 增加、- 減少。將狀態呈現在畫面上。

現在是正數：7

1-0c: python與JavaScript的距離？

- 由資料可視化Data_Viz領域來看
 - Python和JavaScript應該是一個很好互補！
- 在多數人的想像：
在Python-JS兩個世界之間的是「網站開發的巨大鴻溝
-就像大恐龍一般的恐怖」！



幸運的是：透過JSON已把大鴻溝變成細小河流。

- 要橋接Python - JS 之間的問題
 - 主要是:
 - 要有 HTML5骨架和一些CSS
 - JSON(javascript物件符號)(JavaScript Object Notation)
 - 要有一些小的 Flask RESTful API用來送出數據
 - 這些，都不算是很大的程序！

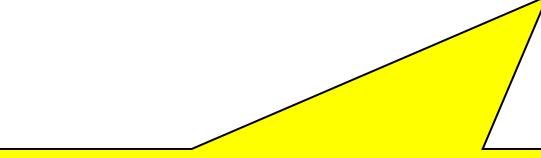


Exercise 1-0c

- Ex1-0cAdd1: 結合python與JavaScript這兩個著名語言有何優點？有那個關鍵技術，讓這兩種語言有方便交換資料的機會？

Chapter 1

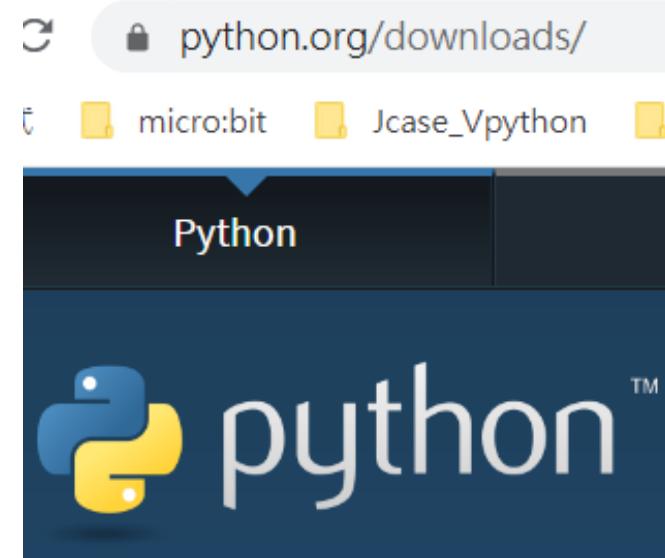
Python Primer



程式=資料結構(DS)+演算法(Algo.)

1-1: Python overview

Types and Operators



The Python Interpreter

- Python is an **interpreted language**. (是一種解譯(/直譯)語言)
- Commands are executed through the Python interpreter.
 - The interpreter receives a command, evaluates that command, and reports the result of the command. (解譯器:接到命令立刻執行)
- A programmer defines a series of commands in advance and saves those commands in a text file known as **source code** or a **script**.
(python的程式，有時也可直接稱為腳本(Script)！)
- For Python, source code is conventionally stored in a file named with the **.py** suffix (e.g., **demo.py**).

IDE (整合開發環境的工具)

- 在Python的學習過程中，必然不可缺少IDE，或代碼編輯器，或集成的開發編輯器。
- IDE能加快使用Python開發的速度，提高編程效率
 - IDLE: 下載官方版的python，就有內建的IDLE。
 - 但是有更方便的IDE整合開發工具，例如：
 - PyCharm 是JetBrains 開發的IDE，具有一般IDE具備的功能，也有版本控制，也有提供好的功能用於Django開發等。
 - Eclipse with PyDev:Eclipse曾經是非常流行的IDE，有久的歷史。
 - Visual Studio Code: Visual Studio Code (VSCode)為MS所開發的code editing tool，免費且開源，並支持Windows，Mac OS，Linux。
 - Spyder:是Anaconda科學計算工具中默認的開發工具。
 - ※在機器學習常用的是網頁版的jupyter notebook.
 - ※雲端可直接寫python的地方：如kaggle或 google的Colab都可直接Run.

Python 設計之前的須知

- 有幾個很基本的認識要先知道：
- 一、程式的縮排(Indent)：
 - 這是 python 很重要的特色！因為縮排代表一個區塊，是程式的結構，所以可以形成結構化程式設計的三大結構。在 python 判斷要縮排的最簡單方式是「：」，若見到冒號，則下一行必定要縮排。
- 二、程式碼的註解(Comment/ Remark)：
 - 可寫說明、作者、或備註事項。
 - 單行：#
 - 多行：用三個雙引號 開始 直到又發現三個雙引號 (※本功能有其他特殊用途) (雙引號改為單引號也可)
- 三、若某一行程式太長：
 - 則可用「\」將單一行，折成兩行或多行。而想要表示\'則要用連續的兩個反斜線，因為\代表逸出符號，例如要表示換行就是\n。
- 四、反之，若一行程式太短：
 - 想多行合成一行，則可以用「；」分號

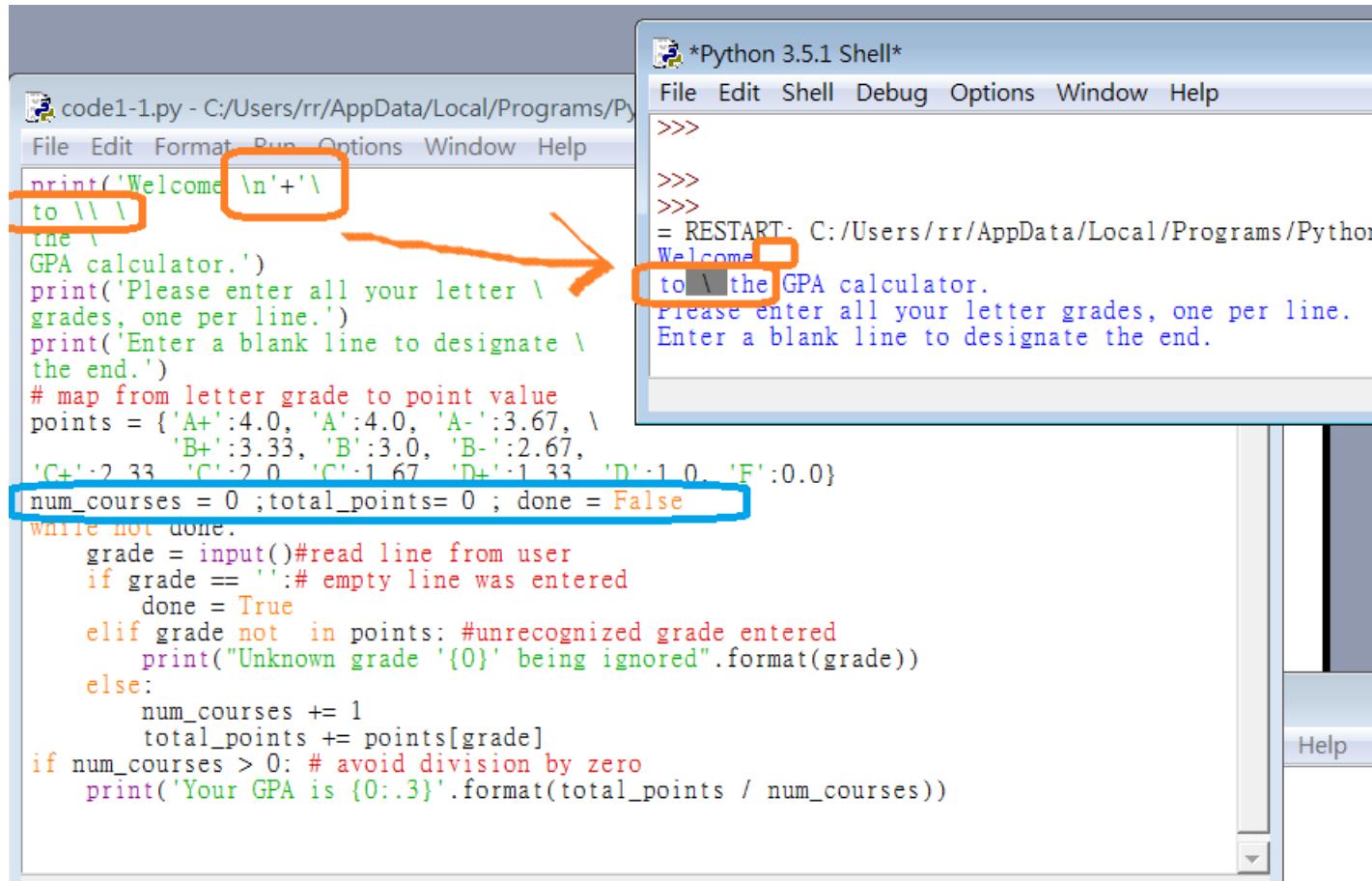
An Example Program GPA (Grade-Point Average) 計算學生成績平均 學生評分換算成績

```
1 print('Welcome to the GPA calculator.')
2 print('Please enter all your letter grades, one per line.')
3 print('Enter a blank line to designate the end.')
4 # map from letter grade to point value
5 points = {'A+':4.0, 'A':4.0, 'A-':3.67, 'B+':3.33, 'B':3.0, 'B-':2.6
6 'C+':2.33, 'C':2.0, 'C-':1.67, 'D+':1.33, 'D':1.0, 'F':0.0}
7 num_courses = 0
8 total_points = 0
9 done = False
10 while not done:
11     grade = input() #read line from user 每次讀一行
12     if grade == '': #empty line was entered 若讀到的是空白行，就結束輸入
13         done = True
14     elif grade not in points: #unrecognized grade entered
15         print("Unknown grade '{0}' being ignored".format(grade))
16     else:
17         num_courses += 1
18         total_points += points[grade]
19     if num_courses > 0: # avoid division by zero
20         print('Your GPA is {0:.3}'.format(total_points / num_courses))
```

```
Welcome to the GPA calculator.
Please enter all your letter grades, one per line.
Enter a blank line to designate the end.
A+
B
```

```
Your GPA is 3.5
```

Lab#1-1-同樣程式，可(1)加入#註解(2)加入;多行變一行或(3)加入\一行變多行



The screenshot shows the Python 3.5.1 Shell interface. On the left is the code editor window containing `code1-1.py`, and on the right is the shell window showing the execution results.

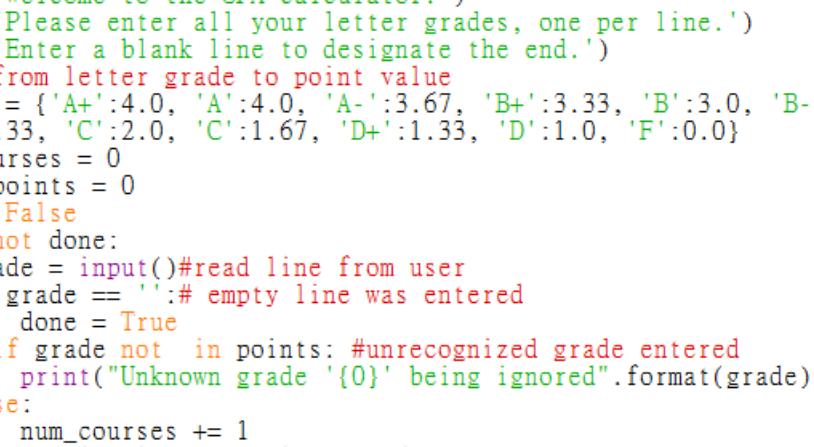
Code Editor (code1-1.py):

```
print('Welcome \n'+\
      'to \\\\'\
      'the \
      GPA calculator.')
print('Please enter all your letter \
grades, one per line.')
print('Enter a blank line to designate \
the end.')
# map from letter grade to point value
points = {'A+'.4.0, 'A':4.0, 'A-':3.67, \
          'B+'.3.33, 'B':3.0, 'B-':2.67, \
          'C+'.2.33, 'C':2.0, 'C-':1.67, 'D+'.1.33, 'D':1.0, 'F':0.0}
num_courses = 0 ;total_points= 0 ; done = False
while not done:
    grade = input()#read line from user
    if grade == '':# empty line was entered
        done = True
    elif grade not in points: #unrecognized grade entered
        print("Unknown grade '{0}' being ignored".format(grade))
    else:
        num_courses += 1
        total_points += points[grade]
if num_courses > 0: # avoid division by zero
    print('Your GPA is {0:.3}'.format(total_points / num_courses))
```

Shell Output:

```
>>>
>>>
>>>
= RESTART: C:/Users/rr/AppData/Local/Programs/Python
Welcome
to \\\' the GPA calculator.
Please enter all your letter grades, one per line.
Enter a blank line to designate the end.
```

Lab#1-1-DScode1-1_in_IDLE



The screenshot shows a Windows desktop environment with a Python code editor window open. The title bar of the window reads "code1-1.py - C:/Users/rr/AppData/Local/Programs/Python/Python35-32/code1-1.py (3)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The main code area contains a script for calculating GPA based on letter grades. The script uses a dictionary to map letter grades to their corresponding point values and includes logic to handle unrecognized grades and calculate the average.

```
print('Welcome to the GPA calculator.')
print('Please enter all your letter grades, one per line.')
print('Enter a blank line to designate the end.')
# map from letter grade to point value
points = {'A+':4.0, 'A':4.0, 'A-':3.67, 'B+':3.33, 'B':3.0, 'B-':2.67,
'C+':2.33, 'C':2.0, 'C-':1.67, 'D+':1.33, 'D':1.0, 'F':0.0}
num_courses = 0
total_points = 0
done = False
while not done:
    grade = input()#read line from user
    if grade == '':# empty line was entered
        done = True
    elif grade not in points: #unrecognized grade entered
        print("Unknown grade '{0}' being ignored".format(grade))
    else:
        num_courses += 1
        total_points += points[grade]
if num_courses > 0: # avoid division by zero
    print('Your GPA is {0:.3}'.format(total_points / num_courses))
```

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
>>>
>>>
= RESTART: C:/Users/rr/AppData/Local/Programs/Python
Welcome to the GPA calculator.
Please enter all your letter grades, one per line.
Enter a blank line to designate the end.
A
A
A-
A-
Your GPA is 3.83
>>>
= RESTART: C:/Users/rr/AppData/Local/Programs/Python
Welcome to the GPA calculator.
Please enter all your letter grades, one per line.
Enter a blank line to designate the end.
A
zzz
Unknown grade 'zzz' being ignored
B
Your GPA is 3.5
```

● 【check2】Lab#1-1-DScode1-1_in_jupyter

TextBook Code1-1

```
In [2]: 1 #code1-1
2 print('Welcome to the GPA calculator.')
3 print('Please enter all your letter grades, one per line.')
4 print('Enter a blank line to designate the end.')
5 # map from letter grade to point value
6 points = {'A+':4.0, 'A':4.0, 'A-':3.67, 'B+':3.33, 'B':3.0, 'B-':2.67,
7 'C+':2.33, 'C':2.0, 'C-':1.67, 'D+':1.33, 'D':1.0, 'F':0.0}
8 num_courses = 0
9 total_points = 0
10 done = False
11 while not done:
12     grade = input()#read line from user
13     if grade == '':# empty line was entered
14         done = True
15     elif grade not in points: #unrecognized grade entered
16         print("Unknown grade '{0}' being ignored".format(grade))
17     else:
18         num_courses += 1
19         total_points += points[grade]
20     if num_courses > 0: # avoid division by zero
21         print('Your GPA is {0:.3}'.format(total_points / num_courses))
22
23
```

```
Welcome to the GPA calculator.
Please enter all your letter grades, one per line.
Enter a blank line to designate the end.
A
zz
Unknown grade 'zz' being ignored
B
```

```
Your GPA is 3.5
```

Lab#1-1-DScode1-1_Google Colab可以寫python

The screenshot shows the Google Colab interface. At the top, there's a search bar labeled "搜尋雲端硬碟" and a sidebar with a "我的雲端硬碟" dropdown menu. The "更多" (More) option in this menu is circled in orange. Below it, the "Google Colaboratory" option is also circled in orange. The main workspace displays a Python notebook titled "DS-code1-1.ipynb". The code in the notebook is a GPA calculator:

```
1 print('Welcome to the GPA calculator.')
2 print('Please enter all your letter grades, one per line.')
3 print('Enter a blank line to designate the end.')
4 # map from letter grade to point value
5 points = {'A+':4.0, 'A':4.0, 'A-':3.67, 'B+':3.33, 'B':3.0, 'B-':2.67,
6 'C+':2.33, 'C':2.0, 'C-':1.67, 'D+':1.33, 'D':1.0, 'F':0.0}
7 num_courses = 0
8 total_points = 0
9 done = False
10 while not done:
11     grade = input()#read line from user
12     if grade == '':# empty line was entered
13         done = True
14     elif grade not in points: #unrecognized grade entered
15         print("Unknown grade '{0}' being ignored".format(grade))
16     else:
17         num_courses += 1
18         total_points += points[grade]
19 if num_courses > 0: # avoid division by zero
20     print('Your GPA is {0:.3}'.format(total_points / num_courses))
```

The output cell shows the execution results:

```
Welcome to the GPA calculator.
Please enter all your letter grades, one per line.
Enter a blank line to designate the end.
A
B
A
zz
Unknown grade 'zz' being ignored
B

Your GPA is 3.5
```

At the bottom right, there's a screenshot of the TensorFlow website showing a guide for building a neural network model.

Lab#1-1-DScode1-1_Kaggle可以寫python

kernel_DS_code1-1 Draft saved

File Edit Insert Run Add-ons Help

+ | Run All

```
# Any results you write to the current directory are saved as output.
print('Welcome to the GPA calculator.')
print('Please enter all your letter grades, one per line.')
print('Enter a blank line to designate the end.')
# map from letter grade to point value
points = {'A+':4.0, 'A':4.0, 'A-':3.67, 'B+':3.33, 'B':3.0, 'B-':2.67,
          'C+':2.33, 'C':2.0, 'C-':1.67, 'D+':1.33, 'D':1.0, 'F':0.0}
num_courses = 0
total_points = 0
done = False
while not done:
    grade = input()#read line from user
    if grade == '':# empty line was entered
        done = True
    elif grade not in points: #unrecognized grade entered
        print("Unknown grade '{0}' being ignored".format(grade))
    else:
        num_courses += 1
        total_points += points[grade]
if num_courses > 0: # avoid division by zero
    print('Your GPA is {0:.3}'.format(total_points / num_courses))
```

Welcome to the GPA calculator.
Please enter all your letter grades, one per line.
Enter a blank line to designate the end.
A
B
zz
Unknown grade 'zz' being ignored
Your GPA is 3.5

Exercises: Ex1-1

- Ex1.1Add1:高階語言可分為解譯(Interpreter) 與編譯(Compiler) ,
請問python是那一種？而且python要加入註解、要多行合一行、要
一行分為多行的特殊符號是什麼？
 - Ex1.1Add2:python常用的IDE有那些？
 - Ex1.1Add3:python語言有那些優點？
 - Ex1.1Add4:Python近年為何這麼紅？這麼流行與受到重視的可能原
因是？
 - Ex1.1Add5:結構化程式設計(Structured programming)是那三種控制
結構(**Control structures**)？以python為例各舉一個簡單例子說明。
 - Ex1.1Add6:遞迴結構可視為更抽象的重覆結構，請舉例說明。
-
- 作業題：**Ex1.1Add1, Ex1.1Add2, Ex1.1Add3 Ex1.1Add5**
 - 補充：**Ex1.1Add4, Ex1.1Add6**

1.2 Objects in Python

- Python is an object-oriented language and classes form the basis for all data types.
 - Python是一種物件導向(OO)的語言，而類(class)是全部資料型態的基礎。
- Python's built-in classes:
 - the **int** class for integers,
 - the **float** class for floating-point values,
 - the **str** class for character strings.

Identifiers, Objects, and the Assignment Statement

識別字、物件、賦值語句

- The most important of all Python commands is an assignment statement:
- `temperature = 98.6`
 - 此命令建立`temperature`作為標識符(ID, Identifier)，然後將其與等號右側表示的物件相關聯，在這種情況下，該對象是值為98.6的浮點數物件。



ID (Identifiers)識別字, 與最重要的33個**保留字** (Reserved Word) (在pyhton3.5以後增加為35個)

- Identifiers in Python are **case-sensitive**, so temperature and Temperature are distinct names.
(大小寫是敏感的，表示，只要有大小寫不同，就視為不相同)
- Identifiers can be composed of almost any combination of letters, numerals, and underscore characters.
(複合字的ID，常會用底線來分辨，如num_courses，※有時也可用開頭大寫來分辨)
- An identifier cannot begin with a numeral and that there are 33 specially reserved words that cannot be used as identifiers:

Reserved Words									
False	as	continue	else	from	in	not	return	yield	
None	assert	def	except	global	is	or	try		
True	break	del	finally	if	lambda	pass	while		
and	class	elif	for	import	nonlocal	raise	with		

35 個保留字，多了兩個保留字！

■ Import keyword:

- 在打入 `import keyword` 之後，".keyword"可以見到 `python` 裡面最重要的 35 個保留字！

■ 原來是33 個，在`python3.5`之後多加入 `async` 和 `await` 兩個協程指令(協同程序**coroutine**)。

※ 使用場景是多行程非同步，而且 `await`語法只能出現在通過`async`修飾的函數中。

■ Lab#1-2-import_keyword 練習：

- 其中的 `enumerate()`，是加上流水號的常用函數
- (請參考exercise1-2)

Types

- Python是一種動態類型化的語言(**dynamically typed language**)，因為沒有將**ID**(標識符)與特定數據類型相關聯的預先聲明。
- **ID**可以與任何類型的對象相關聯，以後可以將其重新分配給相同（或不同）類型的另一個對象。
- 儘管**ID**沒有聲明的類型，但它所引用的對象具有確定的類型。在我們的第一個示例中，字符串“98.6”被識別為浮點數，因此**ID**標識符“temperature”與具有該值的**float**類的實例相關聯。

Objects(物件)

- 創建類的新實例(instance)的過程稱為實例化.instantiation)
 -
- 為了實例化一個對象(object)，我們通常調用一個類的構造函數(constructor)：
 - `w = Widget ()`
 - 這是假設構造函數不需要任何參數。
- 如果構造函數確實需要參數，則可以使用如下語法：
 - `w =Widget (a , b , c)`
- Python的許多內建類(built-in class)都是用於指定新實例的文字形式。例如，命令
 - `temperature = 98.6`
- 導致創建float類的新實例，並指定給temperature。

Calling Methods

- Python 支持函數的語法，例如 `sorted (data)`，在這種情況下，`data` 是傳給函數的參數。
- Python 的類還可以定義一個或多個方法（也稱為成員函數(*member function*)），這些方法可以使用點（“.”）運算符在類的特定實例上調用。
 -
- 例如，Python 的串列類(*list class*)有一個名為 `sort` 的方法，可以用諸如 `data.sort()` 的語法來調用。
 - 這個 `sort` 的方法會重新排列串列的內容，以便對其進行排序。

Built-In Classes(內建類別)

Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

Bool,int
,float,str
是最基本的
資料型態。
都是具有固
定值(
immutable)

如果該類的每個對像在實例化時具有固定值 (*immutable*)，而該固定值隨後無法更改，則該類是不可變的。例如，**float class**是不可變的。

The bool Class

- **bool**類用於邏輯（布林）值，該類的僅有兩個實例(instance)文字為：
 - True , False
- 默認的構造函數**bool()** 返回**False**。
- Python允許使用 語法 **bool(foo)**，從非布林類型創建布林值。結果的解釋取決於參數的類型。
 - 如果是零則結果**False**；
如果非零則結果為**True**。
 - 若傳入序列(**Sequences**)或
其他容器類型(**container**)
(例如字符串**string**或串列**list**)
，如果為空則為**False**，如果
非空則為**True**。

■※註：python的bool型態，是直接用integer來設計的：

```
>>> isinstance(True, int)
True
>>> True+5
6
>>> False+3
3
>>> False + True
1
>>> isinstance(True, float)
False
>>> True == 1
True
>>>
```

The int Class

- Int類，用來表示具有任意大小的整數值。
 - Python根據其值的大小自動為整數選擇內部表示形式。
- 整數構造函數int() 默認情況下返回0。
- 該構造函數還可以構造出一個整數值。
- 例如，如果f表示浮點值，則語法int(f)會生成f的截斷值(無條件去尾法)。例如int(3.14)得到3，而int(-3.9)得到 -3。
- 構造函數還可用於解析表示整數的字符串。例如，表達式int('137') 產生整數值137。

The float Class

- **float**類是Python中的浮點數。
 - 整數2的浮點等效項可以直接表示為2.0。
 - 用於浮點值的文字的另一種形式使用科學計數法。例如，文字6.022e23表示數學值 6.022×10^{23}
- 構造函數**float()**返回0.0。
- 紿定參數時，構造函數**float**將返回等效的浮點值。
 - **Float(2)**返回浮點值2.0
 - **Float('3.14')**返回3.14

The list Class

- 串列(**list**)實例存儲一系列對象，即，對串列中對象的一系列引用(或指標)。
- 串列的元素(**element**)可以是任意對象（包括**None**）。
- 串列是基於數組(**array-based**)的序列(**sequence**)，長度為n的串列具有從0到n-1（含0）的索引。
- 串列可根據需要動態擴展或收縮其容納量。
- Python使用字符`[]`作為串列文字的分界符號
 - `[]`是一個空串列。
 - `['red', 'green', 'blue']`是包含三個字符串實例的串列。
- `list()`構造函數默認產生一個空串列。
- 串列構造函數將接受任何可迭代的參數。
 - `list('hello')`生成單個字符的串列，`['h', 'e', 'l', 'l', 'o']`。

The tuple Class

- 元組類(**tuple**)：可以視為具有不變性（**immutable**）的**list**。因此**tuple**的物件在實例化時，可以比串列(**list**)有更簡化的內部表示(更精簡的記憶體配置)。
- 用括號()來界定一個元組。
 - 空元組是()
- 要表示「長度為1的元組」，必須在元素後但括號內放置一個逗號。
 - 例如(17,)是只有一元素的**tuple**。
 - ※如此，才可以與單一參數計算的括號有分辨。

The str Class

- 字符串(**String**)文字可以用單引號括起來，如'hello'，也可以用雙引號括起來，如“hello”。
- 如果字符串中包含換行符(**newline**)，則字符串也可以以三個單引號或雙引號引起來。

```
print(""" Welcome to the GPA calculator.  
Please enter all your letter grades, one per line.  
Enter a blank line to designate the end.""")
```

The set Class

- Python的set類表示一個集合，即元素的集合，沒有重複，也沒有固定順序(order)！
- 放入set的元素：
 - 只能將不可變類型的實例添加到Python集。因此，諸如整數，浮點數和字符串之類的對像有資格成為集合的元素。
- Frozenset類，就是set類型，但必須不可變形式。可視為具有不變性(immutable)的set。
- Python使用花括號{和}作為集合的分界符號。
 - 例如，為{17}或{'red'， 'green'， 'blue' }
 - 此規則的例外是，「{}不代表空集合」。只有構造函數set()才是一個空集合。例如:set1 = set() - ※因為{}代表空字典！

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> l
>>> letters = {'a', 'b', 'c'}
>>> 'c' in letters
True
>>> letters.add('d')
>>> letters
{'d', 'b', 'c', 'a'}
>>> letters = frozenset({'a', 'b', 'c'})
>>> letters.add('d')
Traceback (most recent call last):
  File "<pyshell#38>", line 1, in <module>
    letters.add('d')
AttributeError: 'frozenset' object has no attribute 'add'
>>>
Ln: 82 Col: 4
```

```
Python 3.5.1...
File Edit Shell Debug
Options Window Help
>>>
>>> di={}
>>> se=set()
>>> type(di)
<class 'dict'>
>>> type(se)
<class 'set'>
>>>
Ln: 492 Col: 4
```

The dict Class

- Python的**dict**類(字典類)代表從一組相異鍵(key)到關聯值(value)的字典或映射。
- Python使用與集合幾乎相同的方法來實現**dict**，但要存儲相關的值(value)。
 - 文字形式{}產生一個空字典。
- 非空字典使用”，”逗號分隔的一系列{key : value}。
 - 例如，字典{“ga”：“Irish”，“de”：“German”}，就可以將“ga”映射為“Irish”，“de”映射為“German”。
 - 另外，構造函數也可以接受鍵值對(key-value pair)序列作為參數，如**dict(pairs)**中，**pairs = [('ga', 'Irish'), ('de', 'German')]**。

Exercises: Ex1-2

- Ex1.2Add1: python語言，想要查詢保留字(Reserved Word)，可行的方法是什麼？
- Ex1.2Add2: python語言中，最重要的保留字(Reserved Word) 原來有33個在pyhton3.5以後增加為35個，是增加那兩個新指令？主要是在那種場合使用？
- Ex1.2Add3: python語言中，最基本的四種具有固定值(不變量 **immutable**)的是那四種類型？而最常用的可變序列，又可想成是「可裝貨的火車」是那一種類型？
- Ex1.2Add4: 串列(list),元組類(tuple),字典類(dict)是三種具容器效果的資料型態，要分辨這三種類可從建構新實例時的符號就看得出來。請問這三種類在宣告新實例時，代表的符號是什麼？以List與Dict為例，要增加與刪除資料的常見的用法是？
- 作業題：**Ex1.2Add1, Ex1.2Add2, Ex1.2Add3 , Ex1.2Add4**
- 補充：

Lab#1-2-import_keyword

- 在打入 import keyword 之後，keyword 可以見到 python 裡面最重要的 35 個保留字！
- ※原來 33 個，在 python 3.5 之後才加入 async 和 await 兩個協同程序（co-routine）的新語法，是應用在多行程非同步的場景。

The image shows three separate Python shells side-by-side, each displaying the output of the command `print(keyword.kwlist)`. The outputs are as follows:

- Python 3.7.4 Shell:** Shows 35 keywords: False, None, True, and, as, assert, async, await, break, class, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield.
- Python 3.5.1 Shell:** Shows 33 keywords: False, None, True, and, as, assert, async, await, break, class, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield.
- Python 3.8.1 Shell:** Shows 35 keywords: False, None, True, and, as, assert, async, await, break, class, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield.

Annotations highlight several differences:

- A red dashed circle highlights the word `assert` in the 3.7.4 output, which is circled again in the 3.8.1 output.
- A red circle highlights the number `35` in the 3.8.1 output.
- A red circle highlights the word `break` in the 3.8.1 output, which is also circled again in the 3.8.1 output.

1.3 Expressions and Operators

- 可以使用特殊符號和稱為運算符(operator)的關鍵字將現有值組合成表達式(expression)。
- 運算符(operator)的語義取決於其運算元(operand)的類型。
- 例如，當a和b是數字時，語法 $a + b$ 表示加法，而如果a和b是字符串，則運算符+表示字串連結。

Logical Operators

- Python 支持以下用於布林值的關鍵字運算符：
 - not** unary negation
 - and** conditional and
 - or** conditional or
- 和(And), 或(Or)運算符支援短路(**short-circuit**)判斷。因為如果可以根據第一個操作數的值就可以確定結果，則它們不評估第二個操作數。
 -

Equality Operators

- Python 支持以下運算符來測試兩個相等性概念：

is	same identity
is not	different identity
==	equivalent
!=	not equivalent

- 恰好在標識符'a is b'是同一對象的別名時，表達式a為b的計算結果為True。
- 表達式'a == b'測試了更一般的等價概念。

Comparison Operators

- 數據類型可以通過以下運算符定義自然順序：
 - $<$ less than
 - \leq less than or equal to
 - $>$ greater than
 - \geq greater than or equal to
- 這些運算符對於數字類型具有預期的行為，
 - 012..9所以9>.. >2>1>0
- 並且對於字符串在字典順序上且區分大小寫地進行定義。
 - ABC... Z 所以Z>..>C>B>A
 - Z比A大，全部小寫比大寫大

Arithmetic Operators

- Python 支持以下算術運算符：

+	addition
-	subtraction
*	multiplication
/	true division
//	integer division
%	the modulo operator

加減乘除
整數除的商，餘

- 對於加，減和乘，如果兩個操作數的類型均為int，則結果為int；否則，結果為float。如果一個或兩個操作數的類型為float，則結果為float。
- 真除法始終為float類型，整數除法(integer division)始終為int（結果會截斷小數）

Bitwise Operators

- Python為整數提供以下「位元運算符 (bitwise operators)」：

~	bitwise complement (prefix unary operator)
&	bitwise and
	bitwise or
^	bitwise exclusive-or
<<	shift bits left, filling in with zeros
>>	shift bits right, filling in with sign bit

補運算，and, Or
Ex-or, 左移,右移

Sequence Operators

(序列資料切片 運算slice運算)

- Python的每種內置(built-in)序列類型(str, tuple和list)

都支持以下運算符語法：

<code>s[j]</code>	element at index <i>j</i>
<code>s[start:stop]</code>	slice including indices [start,stop)
<code>s[start:stop:step]</code>	slice including indices start, start + step, start + 2*step, ..., up to but not equalling or stop
<code>s + t</code>	concatenation of sequences
<code>k * s</code>	shorthand for <code>s + s + s + ... (k times)</code>
<code>val in s</code>	containment check
<code>val not in s</code>	non-containment check

Slice的結構，是包括
start但不包含stop. 而且
負號代表由後往前數。

序列的資料(包括string, list, tuple)。都可以進行的操作
包括索引(index)，切片(slice)，加，乘，檢查成員(in)。

Sequence Comparisons(序列資料比較)

- 序列(sequence)根據字典順序比較操作，逐個元素進行比較直到找到第一個差異之處。
 - 例如， $[5, 6, 9] < [5, 7]$ ，因為 index由0開始，在1的內容 $6 < 7$

$s == t$ equivalent (element by element)

$s != t$ not equivalent

$s < t$ lexicographically less than

$s <= t$ lexicographically less than or equal to

$s > t$ lexicographically greater than

$s >= t$ lexicographically greater than or equal to

Operators for Sets(集合的運算)

- 集(Set)和凍結集(frozenset)支持以下運算符：

<code>key in s</code>	containment check
<code>key not in s</code>	non-containment check
<code>s1 == s2</code>	<code>s1</code> is equivalent to <code>s2</code>
<code>s1 != s2</code>	<code>s1</code> is not equivalent to <code>s2</code>
<code>s1 <= s2</code>	<code>s1</code> is subset of <code>s2</code>
<code>s1 < s2</code>	<code>s1</code> is proper subset of <code>s2</code>
<code>s1 >= s2</code>	<code>s1</code> is superset of <code>s2</code>
<code>s1 > s2</code>	<code>s1</code> is proper superset of <code>s2</code>
<code>s1 s2</code>	the union of <code>s1</code> and <code>s2</code>
<code>s1 & s2</code>	the intersection of <code>s1</code> and <code>s2</code>
<code>s1 - s2</code>	the set of elements in <code>s1</code> but not <code>s2</code>
<code>s1 ^ s2</code>	the set of elements in precisely one of <code>s1</code> or <code>s2</code>

這些集合運與數學的集合相同！例如聯集|，交集&，差集-

Operators for Dictionaries(字典的運算)

- `dict`類型的對象支持的運算符如下：

<code>d[key]</code>	value associated with given key
<code>d[key] = value</code>	set (or reset) the value associated with given key
<code>del d[key]</code>	remove key and its associated value from dictionary
<code>key in d</code>	containment check
<code>key not in d</code>	non-containment check
<code>d1 == d2</code>	<code>d1</code> is equivalent to <code>d2</code>
<code>d1 != d2</code>	<code>d1</code> is not equivalent to <code>d2</code>

[]的用法，可想成是陣列的索引。
可把key直接當作index來使用。
至少要會：取值，加newValue，移除三種操作。

Operator Precedence(運算子的優先權)

Operator Precedence		
	Type	Symbols
1	member access	<code>expr.member</code>
2	function/method calls container subscripts/slices	<code>expr(...)</code> <code>expr[...]</code>
3	exponentiation	<code>**</code>
4	unary operators	<code>+expr</code> , <code>-expr</code> , <code>~expr</code>
5	multiplication, division	<code>*, /, //, %</code>
6	addition, subtraction	<code>+, -</code>
7	bitwise shifting	<code><<, >></code>
8	bitwise-and	<code>&</code>
9	bitwise-xor	<code>^</code>
10	bitwise-or	<code> </code>
11	comparisons containment	<code>is, is not, ==, !=, <, <=, >, >=</code> <code>in, not in</code>
12	logical-not	<code>not expr</code>
13	logical-and	<code>and</code>
14	logical-or	<code>or</code>
15	conditional	<code>val1 if cond else val2</code>
16	assignments	<code>=, +=, -=, *=, etc.</code>

指數`**`,
 ->單元運算:正負反
 ->乘除、整除商、餘
 ->加減。

位元運算:左右移
 ->位元 `& ^ |` (`and,xor,or`)
 ->比較與包含
 ->邏輯 `not, and, or`

Python 串列(List)補充說明-1

- 串列就相當於其他語言的陣列，而且資料項目可放入不相同類型。
- 創建一個串列，只要把逗號分隔的不同的資料項目使用方括號括起來即可。如下所示：
- `li1=['tku','edu',2020,131]`
- `li2=[1,2,3,4,5,6,7]`
- `li3 = ["a", "b", "c", "d"]`
- 與字串的索引一樣，List的索引從0開始。List可以進行截取、組合等。
- 訪問串列中的值

- 使用下標索引來訪問串列中的值，同樣你也可以使用方括號的形式截取字元，如下所示：

- 實例

```
li1 = ['tku', 'edu', 2020, 131]
li2 = [1, 2, 3, 4, 5, 6, 7]
```

```
print ( li1[0])
print ( li2[1:5])
```

■ ●[1:5]就是 切片 (Slice)運算！ li2[-1]就是倒數第1個字(最末尾的字)

- 結果：

- tku

- [2, 3, 4, 5]

The screenshot shows the Python 3.5.1 Shell window. The code entered is:

```
>>> li1=['tku','edu',2020,131]
>>> li2=[1,2,3,4,5,6,7]
>>> li3=['a','b','c','d']
>>> print(li1[0])
tku
>>> print(li2[1:5])
[2, 3, 4, 5]
>>> print(li1[3])
131
>>>
```

Ln: 425 Col: 17

Python 串列(List)補充說明-2

■ 動態新增的更新串列元素

- 你可以對串列的資料項目進行修改或更新，你也可以使用append()方法來添加串列項，如下所示：

```
□ li = []          ## 空串列
    li.append('Go') ## 使用 append() 添加元素
    li.append('Run')
    print (li)
```

- 以上實例輸出結果：
 - ['Go', 'Run']

■ 動態刪除串列元素

- 可以使用 `del` 語句來刪除串列的元素，如下實例：

□ li1 = ['tku', 'edu', 2020, 131]

print(li1)

del (li1[2]) #del 是重要的保留字哦！ ! ~

```
print ("After deleting value at index 2 : "
```

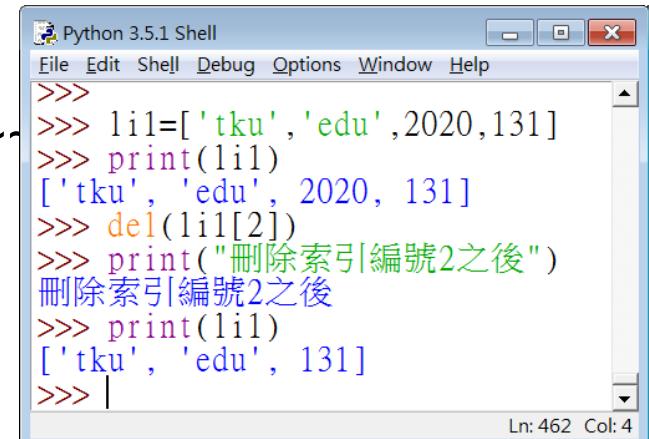
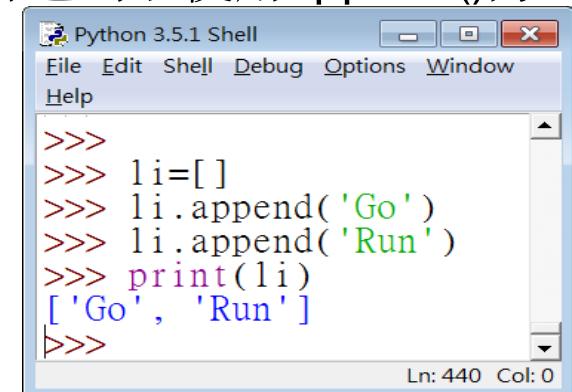
print (li1)

- #### □ 以上實例輸出結果：

- ['tku', 'edu', 2020, 131]

After deleting value at index 2 :
[tku', 'edu', 131]

■ 注意：還有另外，先搜尋再刪除的方法是remove(), .index()...



Exercises: Ex1-3

- Ex1.3Add1:**List**(串列)就相當於其他語言的陣列，非常important。請舉簡單例子來說明**List**中，元素的訪問、新增、刪除(訪問串列中的值，動態新增串列元素，動態刪除串列元素)
- Ex1.3Add2:**dict**(字典)是高效的資料結構。至少要會：取值,加新值,移除三種操作。請舉簡單的例子來示範這三種操作。
- 作業題：**Ex1.3Add1, Ex1.3Add2**
- 補充：

Lab#1-3-List實驗的slice_新增_刪除

- 要求：把2020換為個人學號，而131換為個人在本班內的座號

```
: 1 li1=['tku','edu',2020,131]
  2 li2=[1,2,3,4,5,6,7]
  3 li3=['a','b','c','d']
  4 print( li1[0])
  5 print( li2[1:5])
  6 print( li1[3])
  7
```

```
tku
[2, 3, 4, 5]
131
```

```
1 li1=['tku','edu',2020,131]
2 print(li1)
3 del(li1[2])
4 #del 是重要的保留字哦！！~~~~~
5 print("刪除索引編號 2之後:")
6 print(li1)
```

```
['tku', 'edu', 2020, 131]
刪除索引編號 2之後:
['tku', 'edu', 131]
```

```
: 1 li=[]
  2 ## 空串列
  3 li.append('Go') ## 使用 append() 添加元素
  4 li.append('Run')
  5 print(li)
  6 #以上實例輸出結果：
  7 ['Go', 'Run']
```

```
['Go', 'Run']
```

1.4 Program Structure(程式結構)

- 對於所有控制結構而言，”:” 冒號字符，用來界定控制結構主體的代碼區塊的開頭。
- 如果可以將主體程式用單行執行語句來表示，則可放在冒號右側的同一行上。
- 但是，通常將主體排版為從冒號之後的行開始的縮排區塊。
- Python依靠縮排的級別來指定該代碼塊或其中的任何巢狀(嵌套)代碼塊的範圍。

Conditionals(條件結構的一般形式)

```
if first_condition:  
    first_body  
  
elif second_condition:  
    second_body  
  
elif third_condition:  
    third_body  
  
else:  
    fourth_body
```

條件結構裡的每個條件
都是布林表達式
要滿足條件才會執行。

比較複雜的條件結構(以機器人開門，前進為例)

- 結構可巢狀：
也就是大選擇結構中，
可以有中選擇結構，
中選擇結構裡面，還
可以有小選擇結構！
- 巢狀(嵌套)結構，要靠縮
排(Indent)來確定區塊。
- 所以在python語言，縮
排的位置是很重要的！
- ※註：縮排區塊的專用
術語是”SUITE”，縮進是
”InDent”，縮進相反是
”DeDent”

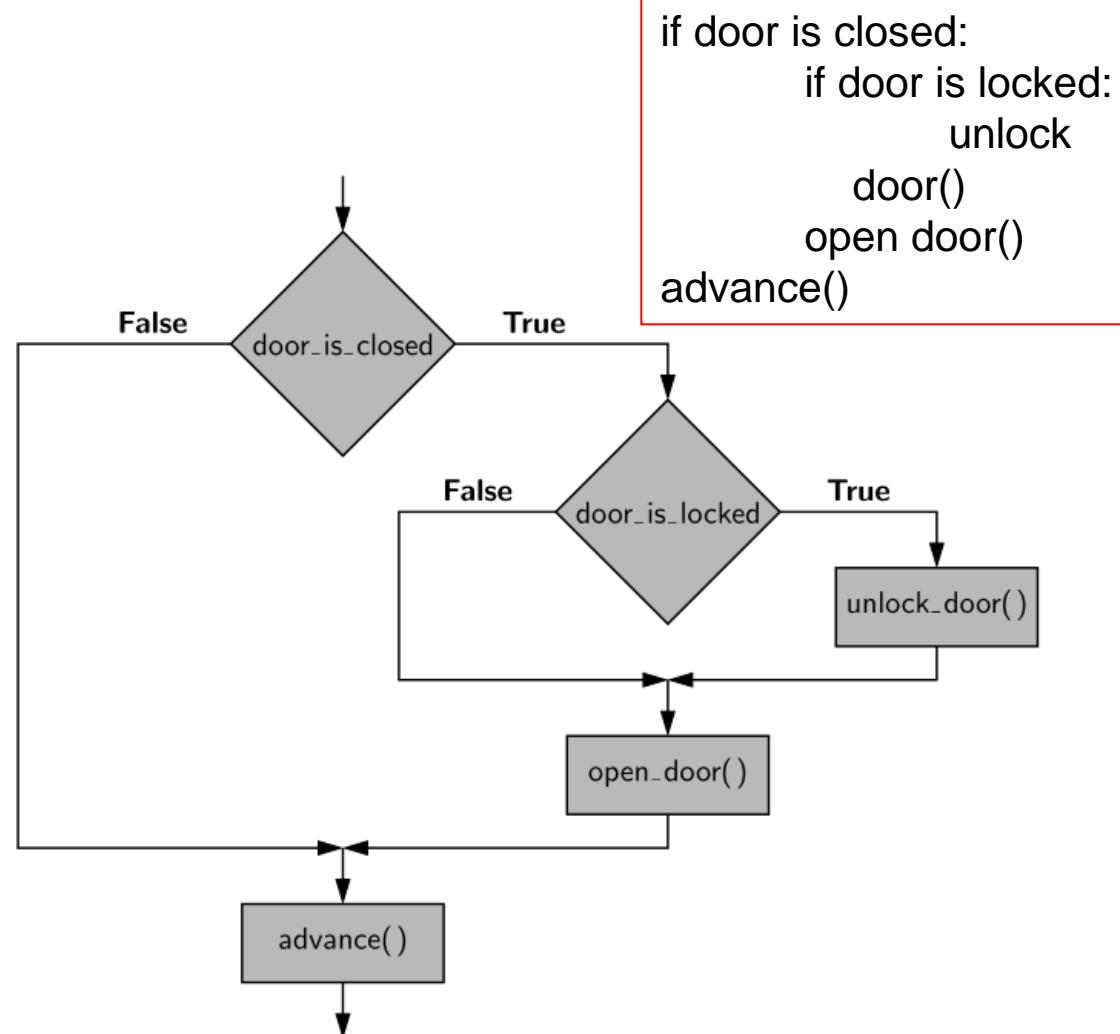


Figure 1.6: A flowchart describing the logic of nested conditional statements.

Loops(重覆結構)-有兩種不同的循環結構

■ While loop:

while condition:
body

data是字串，要尋找第一個X出現的位置

```
j = 0  
while j < len(data) and data[j] != 'X':  
    j += 1
```

■ For loop:

```
for element in iterable:  
    body # body may refer to 'element' as an identifier
```

■ Indexed For loop:

● **for**常用於基於索引的循環，不單循環，還**希望知道索引位置**。
● **range**代表檢查次數 **len(data)** 利用**index**代表目前索引到的位置 此程式是想知道最大值的位置。

While:以布林條件為重覆測試的循環結構。

For:適合把循環的語法使用在一個已定義的序列中，或任何疊代器(Iterator)類型的結構中(如 List, Dict, Str, Tuple...)

※**for** 適合用於知道重覆次數的情況(配合**range()**)。

```
big_index = 0  
for j in range(len(data)):  
    if data[j] > data[big_index]:  
        big_index = j
```

Break and Continue(立即終止全部循環，或，僅停止本次疊代而循環過程的後序要繼續)

- Python支持**break**語句，當它在其主體中執行時，該語句立即終止**While**或**for**的循環。

```
found = False
for item in data:
    if item == target:
        found = True
        break
```

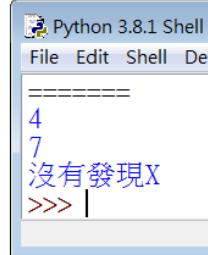
- Python還支持**continue**語句，該語句使循環主體的當前疊代停止，但隨後的循環則按預期進行。

Exercises: Ex1-4

- Ex1.4Add1:python的控制結構,用來界定控制結構主體的代碼區塊的開頭的字符是什麼？接下來要進入縮排區，請問縮排區習慣上按鍵盤上的那個鍵？或相當於是幾個空白？
 - Ex1.4Add2:在python中，每個東西都算是一個物件，除了有物件導向特性，也具有傳統的「結構化程式設計」的重要特色。結構化的三種程式控制基本架構，是那三種？
 - Ex1.4Add3:在python的重複結構，有提供兩種寫法，請問是那兩種？並說明，要如何分辨這兩種寫法適合使用的時機？
-
- 作業題：[Ex1.4Add1](#), [Ex1.4Add2](#), [Ex1.4Add3](#)
 - 補充：

Lab#1-4a-Loop-while

- Case1:X有在data的字串中：
 - 可找到data字串裡X的位置。
- Case2:若X不在data字串裡：
 - 則因為超過末尾將會產生異常(IndexError)此時可分為兩種情況
 - Case2a:
因為python的短路測試效應，
沒有任何錯誤，但是，
得到數字會是超過末尾的數字！
 - Case2b:
要準備接收異常 IndexError.



```
# 尋找data字串裡X的位置.py - C:/Users/rr/AppData/Local/Programs/Python/Pyth...
File Edit Format Run Options Window Help
#尋找data字串裡X的位置。
#若X不在字串裡，則因為超過末尾
# 而會產生異常(IndexError)
#data是有X，而data2是找不到X的情況。

#case1:找的到X
j=0
data="0123XYZ"
while j< len(data) and data[j]!='X':
    j+=1
print(j)

#case2a:找不到X，但沒錯誤，只是j會是len.
# 因為python的 "and"有短路效應！
j=0
data2="0123456"
while j< len(data2) and data2[j]!='X':
    j+=1
print(j)

#case2b:找不到X，但會產生異常，而可進行找不到的處理
# 因為索引位置超過data2的末尾，而丟出IndexError異常
j=0
try:
    data2="0123456"
    ###下面這行，只是 and 的左右換位！
    while data2[j]!='X' and j< len(data2):
        j+=1
    print(j)
except IndexError:
    print("沒有發現X")
```

Lab#1-4b-Loop-for

The screenshot shows two windows: a code editor and a Python shell.

Code Editor: The file `lab#1-4-for_loop.py` contains the following Python code:

```
data=[1,3,2,6,9,8,4]
print(data)
big_index=0
for j in range(len(data)):
    if data[j] > data[big_index]:
        big_index=j
print(big_index)
print("因為編號4的值是9，是最大值")
```

Python 3.5.1 Shell: The shell window shows the following output:

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
RESTART: C:/Users/rr/AppData/Local/...
[1, 3, 2, 6, 9, 8, 4]
4
因為編號4的值是9，是最大值
>>> |
```

1.5 Functions(函數)

- 定義函數簽名(function signature):
 - 使用關鍵字**def**定義函數Function。

```
def count(data, target):
    n = 0
    for item in data:
        if item == target:
            n += 1
    return n
```

Python的函數與其他語言(C, Java)不同！不必指明參數的型態，也不必指明傳回的型態。

※但是在最新語法中，有加入「箭頭函數」，就可加入指定型態的註解哦！

- 上述程式，將建立一個新的**ID**(標識符)作為函數的名稱(在本例中為**count**)，並建立期望兩個參數(本例是**data**與**target**)
- **return**語句返回此函數的值並終止函數。

Information Passing(訊息傳送)

- Python中傳遞參數(Parameter passing)

遵循標準賦值語句的語義。

- 例如呼叫前面寫的函數

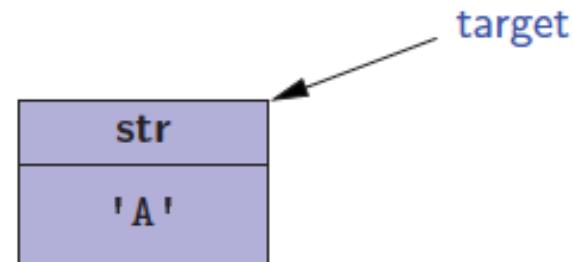
```
prizes = count(grades, 'A')
```

- 會把呼叫的實際參數分配給函數定義的形式參數：

- 結果如下：

```
def count(data, target):  
    n = 0  
    for item in data:  
        if item == target:  
            n += 1  
    return n
```

```
data = grades  
target = 'A'
```



python參數傳遞是用"Call by Sharing"

Return 的結果就像是賦值給呼叫者

- 從函數到調用者的返回值的通信效果，就像是賦值運算的實現。
 - `prizes= count(grade,'A')`
 - `prizes`會得到函數執行結束後`return` 的結果就是 `n`.
- Python訊息傳送機制的優點是，「不會複制對象(not copy)」。這樣即使在參數或返回值是複雜對象的情況下，也可以確保有效地調用函數。

參數是「可變的參數」(mutable)的情況

- 當參數是可變對象時，參數傳遞會有其他含義：
- 因為形式參數是實際參數的別名，所以函數主體「可以」通過更改其狀態的方式與對象進行交互。
- ●容易誤用的情況：
 - 函數內，將新值「重新分配」給形式參數，例如通過設置**data = []**，則不會更改實際參數；這樣的重新分配只會破壞別名。
 - 沒有理由期待將參數重新分配！因為會對參數產生意外效果，這是「一個很糟糕的設計。」
- ●合理使用的情況：(可修改參數)
 - 簡單的例子”
名為**scale**的方法的實現，其主要目的是將數值數據集的所有條目乘以給定因子。。

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
>>> def scale(data, factor):
    for j in range(len(data)):
        data[j] *= factor

>>> a=[1,3,5,10]
>>> scale(a,100)
>>> a
[100, 300, 500, 1000]
>>> |
```

The screenshot shows a Python 3.5.1 Shell window. It displays the definition of a function named 'scale' which takes two parameters: 'data' and 'factor'. Inside the function, a loop iterates over the indices of the 'data' list, multiplying each element by the 'factor'. Below the function definition, a list 'a' is created with elements [1, 3, 5, 10]. When 'scale(a, 100)' is run, the list 'a' is modified in place, resulting in [100, 300, 500, 1000]. The status bar at the bottom right indicates 'Ln: 587 Col: 4'.

Default Parameter Values (用=等號來代表預設值！)

- 例如： `def foo(a,b=15,c=27)`
- 此時，如果調用方僅發送一個參數，`foo(4)`，該函數將使用參數值 $a = 4$ ， $b = 15$ ， $c = 27$ 執行。
- 如果調用方，發送了兩個參數，則假定它們是前兩個，第三個是默認參數。因此 `foo(8, 20)` 就相當於是 $a=8$, $b=20$, $c=27$.
- 注意。Python 規定出現有預設值參數面的參數也必須有。下面是不合法的函數簽名：`def foo(a,b=15,c)`

```
a+b+c.py - C:/Users/rr/AppData/Local/Programs/Python/Python38-32/a+b+c.py (3.8.1)
File Edit Format Run Options Window Help
def foo(a,b,c):
    return a+b+c
print(foo(10,20,30))
def foo(a,b,c=30):
    return a+b+c
print(foo(10,20))
print(foo(10,20,30))
def foo(a,b=20,c):
    return a+b+c
print(foo(10,20,30))
```

一旦有預設值參數，後面的參數也必須有預設值

SyntaxError
non-default argument follows default argument
確定

```
d+b+c.py - C:/Users/rr/AppData/Local/Programs/Python/Python38-32/d+b+c.py (3.8.1)
File Edit Format Run Options Window Help
def foo(a,b,c):
    return a+b+c
print(foo(10))

Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
=====
RESTART: C:/Users/rr/AppData/Local/Programs/Python/Python38-32/d+b+c.py
Traceback (most recent call last)
  File "C:/Users/rr/AppData/Local/Programs/Python/Python38-32/d+b+c.py", line 1, in <module>
    print(foo(10))
TypeError: foo() missing 2 required positional arguments: 'b' and 'c'
>>>
```

使用默認(預設)參數的例子(計算學生GPA)

Points可直接用預設，也可自己再重新設定。

```
def compute_gpa(grades, points={ 'A+':4.0, 'A':4.0, 'A-':3.67, 'B+':3.33,
                                'B':3.0, 'B-':2.67, 'C+':2.33, 'C':2.0,
                                'C-':1.67, 'D+':1.33, 'D':1.0, 'F':0.0}):
    num_courses = 0
    total_points = 0
    for g in grades:
        if g in points:                               # a recognizable grade
            num_courses += 1
            total_points += points[g]
    return total_points / num_courses
```

Code Fragment 1.2: A function that computes a student's GPA with a point value system that can be customized as an optional parameter.

Range(start, stop, step)

- Range()裡的參數，可以1個，2個，3個。這也是因為有預設參數的效果：
 - 可以把range(start, stop, step)想成：

```
def range(start, stop=None, step=1):  
    if stop is None:  
        stop = start  
        start = 0
```

- ※特別技巧：如果只傳1個參數，則會調整參數 start=0, 而stop=start

Keyword Parameters(關鍵字參數)

- 實際參數與形式參數進行匹配的傳統機制是基於位置參數(*positional arguments*)的概念。
- Python 支持將參數發送到稱為關鍵字參數(*Keyword argument*)的機制。例如：
 - `def foo(a=10, b=20, c=30),`
 - `foo(c=5)` 就會是合法的。
- 其他實例：
 - 默認情況下，`max()`函數，是根據該類型的“運算符號”，並根據元素的自然順序進行操作。但是，最大值可以通過比較元素的其他方面來計算。
 - 如果希望 `+20` 可以比 `-35` 小，那麼就可以用下列方式呼叫`max()`
 - `max(a, b, key=abs)` #因為key傳入的函數，會在比較大小前都加上 `abs()`
 - 也就是 `abs(+20) < abs(-35)`

The screenshot shows the Python 3.8.1 Shell interface. The menu bar includes File, Edit, Shell, Debug, Options, and Windows. In the command line area, the user types `>>> max(-35, 20)`, which results in the output `20`. Then, the user types `>>> max(-35, 20, key=abs)`, which results in the output `-35`. This demonstrates how the `key=abs` parameter changes the comparison logic of the `max()` function.

※補充：py3.8最新增

■ Positional-only parameters

- /之前，指示參數必須是位置指定參數。

■ *之後，是僅關鍵字參數

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> def to_fahrenheit(*, celsius):
    return 32 + celsius * 9 / 5

>>> to_fahrenheit(40)
Traceback (most recent call last):
  File "<pyshell#155>", line 1, in <module>
    to_fahrenheit(40)
TypeError: to_fahrenheit() takes 0 positional arguments but 1 was given
>>> to_fahrenheit(celsius=40)
104.0
>>> |
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> def incr_addy(x,/,y=0):
    return x+1 + y

>>> incr_addy(2.3)
3.3
>>> incr_addy(2.3,y=20)
23.3
>>> incr_addy(x=2.3,y=20)
Traceback (most recent call last):
  File "<pyshell#151>", line 1, in <module>
    incr_addy(x=2.3,y=20)
TypeError: incr_addy() got some positional-only arguments passed as keyword arguments: 'x'
>>> |
```

1.5.2 Python's Built-In Functions

- 内建函數：可參考：<https://docs.python.org/3/library/functions.html>

Common Built-In Functions	
Calling Syntax	Description
<code>abs(x)</code>	Return the absolute value of a number.
<code>all(iterable)</code>	Return True if <code>bool(e)</code> is True for each element e.
<code>any(iterable)</code>	Return True if <code>bool(e)</code> is True for at least one element e.
<code>chr(integer)</code>	Return a one-character string with the given Unicode code point.
<code>divmod(x, y)</code>	Return $(x // y, x \% y)$ as tuple, if x and y are integers.
<code>hash(obj)</code>	Return an integer hash value for the object (see Chapter 10).
<code>id(obj)</code>	Return the unique integer serving as an “identity” for the object.
<code>input(prompt)</code>	Return a string from standard input; the prompt is optional.
<code>isinstance(obj, cls)</code>	Determine if obj is an instance of the class (or a subclass).
<code>iter(iterable)</code>	Return a new iterator
<code>len(iterable)</code>	Return the number of elements in the iterable.
<code>map(f, iter1, iter2, ...)</code>	Return an iterator yielding elements from iterables based on their position in the arguments.
<code>max(iterable)</code>	Return the largest element in the iterable.
<code>max(a, b, c, ...)</code>	Return the largest of the arguments.
<code>min(iterable)</code>	Return the smallest element in the iterable.
<code>min(a, b, c, ...)</code>	Return the smallest of the arguments.
	<code>min(a, b, c, ...)</code>
	<code>next(iterator)</code>
	<code>open(filename, mode)</code>
	<code>ord(char)</code>
	<code>pow(x, y)</code>
	<code>pow(x, y, z)</code>
	<code>print(obj1, obj2, ...)</code>
	<code>range(stop)</code>
	<code>range(start, stop)</code>
	<code>range(start, stop, step)</code>
	<code>reversed(sequence)</code>
	<code>round(x)</code>
	<code>round(x, k)</code>
	<code>sorted(iterable)</code>
	<code>sum(iterable)</code>
	<code>type(obj)</code>

Exercises: Ex1-5

- Ex1.5 Add1: Python的參數傳遞是用"Call by Sharing" 的機制。(a)與其他語言裡的call by reference, or call by value 比較有何不同？(b)如果要要call by value 或call by reference的效果，在python要如何表達？請分別考慮參數的資料型態是(b1)可變(mutable)，與(b2)不可變 (immutable)的可行寫法。(c)另外，考慮函數呼叫的特性，若想call by reference的「改變」效果，又「不想用可變(mutable)資料型態」時，是否有可行的寫法？

```
>>> def clear_a(x):  
    x = []  
  
>>> def clear_b(x):  
    while x: x.pop()  
  
>>> z = [1, 2, 3]  
>>> clear_a(z)  
>>> clear_b(z)
```
- Ex1.5 Add2:其實真正影響傳值效果的，並不僅是看資料型態是否"immutable"？而是改變參數的操作！例如有下列兩個簡單函數clear_a()與clear_b()。請問呼叫函數返回後，那個會真正把z清為空的list？
- Ex1.5 Add3:python的built-in function都是很重要很常用的函數。例如print, max.. 請問，其中可以查詢字元符的Unicode的函數是？可以把Unicode轉為字元符的函數是？
- Ex1C1.17(章末):我們是否按以下方式實現了之前提過的縮放函數Scale,它是否正常工作？

```
def scale(data, factor):  
    for val in data:  
        val *= factor
```

解釋為什麼能或者為什麼不能。
- 作業題：**Ex1.5Add1, Ex1.5Add2 , Ex1.5Add3, Ex1C1.17(章末):**
- 補充：

Lab#1-5 built-in-function 官網查詢py3.9版顯示有69個
(若不計算，最末的”`__import__()`”->有68個「最重要的」內建函數)

docs.python.org/3.9/library/functions.html#breakpoint				
<code>abs()</code>	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	<code>__import__('aaa')</code> 的更常見的簡單寫法 <code>import aaa</code>

Lab#1-5 built-in function_用dir來查「最重要的」內建函數73個

要求：查詢與自己座號同編號的內建函數，例如10號要查詢編號10的chr(), 52號則查詢 52的ord()

The screenshot shows the Python 3.8.1 Shell interface. In the top window, the user has run the command `>>> built_in_dir=dir(__builtins__)` and then `>>> len(built_in_dir)`, which returns 153. Below this, they have created a list of built-in functions: `>>> built_in_fun=[x for x in built_in_dir if 'a'<=x[0]<='z']`. They are then iterating over this list to print each function name and its corresponding index: `>>> for k,v in enumerate(built_in_fun): print(k,v)`. The output lists all 73 built-in functions from index 0 to 72. A cursor is visible in the code editor area, and a tooltip for the word `ord` is shown, indicating it is the 52nd function in the list.

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC
on win32
Type "help", "copyright", "credits" or "license()" for more inf
>>> built_in_dir=dir(__builtins__)
>>> len(built_in_dir)
153
>>> built_in_fun=[x for x in built_in_dir if 'a'<=x[0]<='z']
>>> for k,v in enumerate(built_in_fun):
    print(k,v)

0 abs
1 all
2 any
3 ascii
4 bin
5 bool
6 breakpoint
7 bytearray
8 bytes
9 callable
10 chr
11 callable
12 compile
13 complex
14 copyright
15 credits
16 delattr
17 dict
18 dir
19 divmod
20 enumerate
21 eval
22 exec
23 exit
24 filter
25 float
26 format
27 frozenset
28 getattr
29 globals
30 hasattr
31 hash
32 help
33 hex
34 id
35 input
36 int
37 isinstance
38 issubclass
39 iter
40 len
41 license
42 list
43 locals
44 map
45 max
46 memoryview
47 min
48 next
49 object
50 oct
51 open
52 ord
53 pow
54 print
55 property
56 quit
57 range
58 repr
59 reversed
60 round
61 set
62 setattr
63 slice
64 sorted
65 staticme
66 str
67 sum
68 super
69 tuple
70 type
71 vars
72 zip
>>>
```

- 此處的73個:
- (原來68個再加以下這5個)
- 14 copyright
- 15 credits
- 23 exit
- 41 license
- 56 quit

This screenshot shows the Python 3.8.1 Shell again. The user has run `>>> __builtins__.ord.__doc__` and `>>> ord.__doc__`, both of which return the same documentation: 'Return the Unicode code point for a one-character string.' The user then runs `>>> chr.__doc__`, which returns the documentation: 'Return a Unicode string of one character with ordinal i; 0 <= i <= 0x10ffff.' Finally, they run `>>> ord('A')` and `>>> chr(65)`, both of which return the character 'A'. The bottom right corner of the window shows the status bar with 'Ln: 209 Col: 4'.

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> __builtins__.ord.__doc__
'Return the Unicode code point for a one-character string.'
>>> ord.__doc__
'Return the Unicode code point for a one-character string.'
>>> chr.__doc__
'Return a Unicode string of one character with ordinal i; 0 <= i <= 0x10ffff.'
>>> ord('A')
65
>>> chr(65)
'A'
>>>
```

1.6 Simple Output(簡單輸出與輸入)

- 內建函數**print()**用於生成標準輸出到控制台。
- 它以最簡單的形式打印任意參數序列，以空格分隔，最後會有一個換行符號。
 - 例如，命令**print ('maroon' , 5)** 就會輸出字符串 '**maroon 5\n**'。(※其中5的前面會有一個空白)
 - 非字符串參數 **x**，則會顯示為 **str(x)** 例如：**print(35)** 就會顯示**str(35)**。
 - **print()**有一個常用的參數"**end**"，就是決定印完之後的結尾。預設是**end='\n'**(換行)。若不換行，則可把**end**改為其他，例如**end=' '**就不會換行。

Simple Input

- 從用戶控制台獲取信息的主要方式是名為**input()**的內建函數。
- 如果要得到可選(optional)的參數。此函數將顯示提示(prompt)，然後等待用戶輸入一些字符序列，再按回車鍵(return)完成。
- 該函數的返回值，會是在按下返回鍵之前輸入的字符串。
 - 這樣的字符串可以加上**int()**，就可立即轉換成整數

```
year = int(input('In what year were you born? '))
```

- ※也可用**float()**轉為浮點數，或**eval()**轉為數值。

Lab#1-6a-簡單輸出與輸入的小應用程式

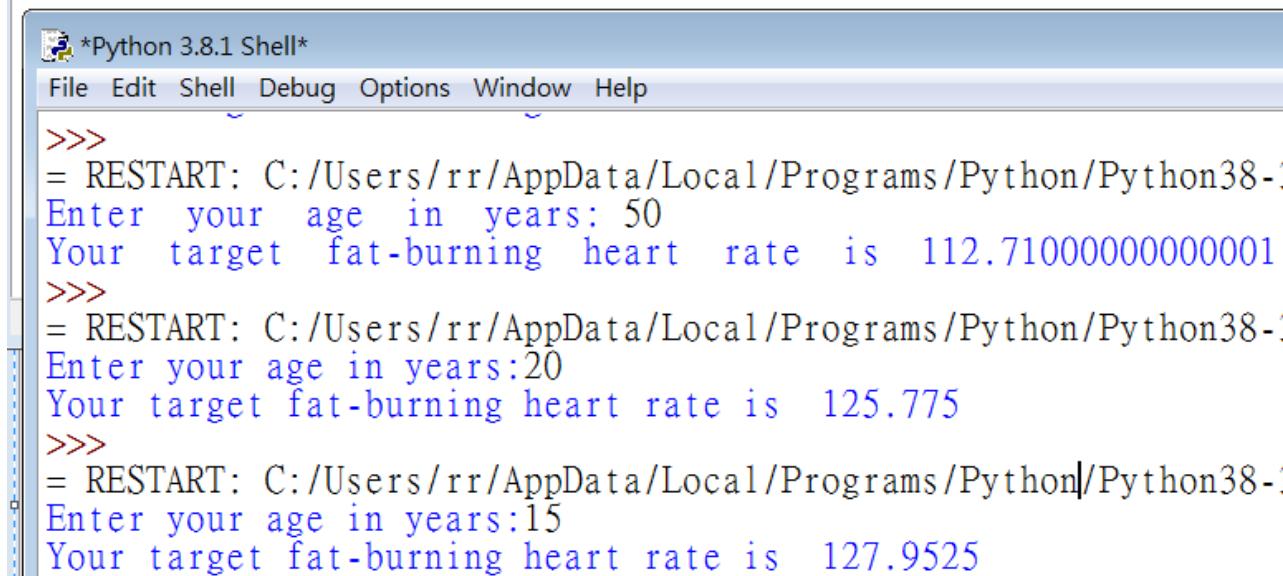
：輸入您的年齡，再查詢適當的目標燃脂心跳率

■ $\text{max_heart_rate} = 206.9 - (0.67 * \text{年齡})$

```
age = int(input('Enter your age in years: '))
max_heart_rate = 206.9 - (0.67 * age) # as per Med Sci Sports Exerc.
target = 0.65 * max_heart_rate
print('Your target fat-burning heart rate is', target)
```

```
age = int(input("Enter your age in years: "))
max_heart_rate = 206.9 - (0.67* age)

target = 0.65 *max_heart_rate
print( "Your target fat-burning heart rate is ", target)
```



The screenshot shows the Python 3.8.1 Shell interface with three separate runs of the script. Each run prompts for an age input, calculates the target heart rate, and prints the result. The first run is for age 50, the second for age 20, and the third for age 15.

```
*Python 3.8.1 Shell*
File Edit Shell Debug Options Window Help
>>>
= RESTART: C:/Users/rr/AppData/Local/Programs/Python/Python38-3
Enter your age in years: 50
Your target fat-burning heart rate is 112.71000000000001
>>>
= RESTART: C:/Users/rr/AppData/Local/Programs/Python/Python38-3
Enter your age in years:20
Your target fat-burning heart rate is 125.775
>>>
= RESTART: C:/Users/rr/AppData/Local/Programs/Python/Python38-3
Enter your age in years:15
Your target fat-burning heart rate is 127.9525
```

Files (檔案文件)

- 使用內建函數open打開文件檔案，該函數返回檔案的物件
 -
- 例如，命令fp = open ('sample.txt') 嘗試打開名為 sample.txt的文件。
- 文件檔案使用方法：

產生文字檔的簡單三步驟

- (1) 開檔:open 「可寫入」 檔
- (2) 使用檔案: Print轉向，
- (3) 關檔:close

以下是最簡單，逐行寫資料的方法：

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> import sys
>>> fp = open('c:\\\\in.txt', 'w')
>>> print("aaa,aaa2,aaa3", file=fp)
>>> print("bbb    bbb2    bbb3", file=fp)
>>> print("ccc  ccc2  ccc3", file=fp)
>>> print("ddd  ddd2,ddd3", file=fp)
>>> fp.close()
>>>
```

read()是讀完全部內容

read(k)是讀k個byte

Calling Syntax	Description
fp.read()	Return the (remaining) contents of a readable file as a string.
fp.read(k)	Return the next <i>k</i> bytes of a readable file as a string.
fp.readline()	Return (remainder of) the current line of a readable file as a string.
fp.readlines()	Return all (remaining) lines of a readable file as a list of strings.
for line in fp:	Iterate all (remaining) lines of a readable file.
fp.seek(k)	Change the current position to be at the <i>k</i> th byte of the file.
fp.tell()	Return the current position, measured as byte-offset from the start.
fp.write(string)	Write given string at current position of the writable file.
fp.writelines(seq)	Write each of the strings of the given sequence at the current position of the writable file. This command does <i>not</i> insert any newlines, beyond those that are embedded in the strings.
print(..., file=fp)	Redirect output of print function to the file.

Lab#1-6b-FileOpen逐行讀資料的方法

- 有很多方法，建議是 **with**取代**open**，就不必 **try**，也不必管**close()**

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> fp = open('c:\\\\in.txt')
>>> line=fp.readline()
>>> while line:
    print(line, end='')
    line=fp.readline()

aaa,aaa2,aaa3
bbb bbb2 bbb3
ccc ccc2 ccc3
ddd ddd2,ddd3

>>>
>>> fp = open('c:\\\\in.txt')
>>> for line in fp.readlines():
    print(line, end='')

aaa,aaa2,aaa3
bbb bbb2 bbb3
ccc ccc2 ccc3
ddd ddd2,ddd3
>>> |
```

The screenshot shows three code snippets in the Python 3.8.1 Shell:

- Top Snippet:** Demonstrates reading the entire file at once using `read()`. A yellow callout box points to the `read()` method with the text "read()是讀完全部內容".
- Middle Snippet:** Demonstrates reading line by line using a `for` loop with `readline()`.
- Bottom Snippet:** Demonstrates reading line by line using a `for` loop with `readlines()`.

A red dashed box highlights the middle snippet, and a yellow callout box points to the first line of the output with the text "這個寫法是高效、快速又簡單的程式寫法".

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> fp = open('c:\\\\in.txt')
>>> print(fp.read())
aaa,aaa2,aaa3
bbb bbb2 bbb3
ccc ccc2 ccc3
ddd ddd2,ddd3

Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
>>> fp=open('c:\\\\in.txt')
>>> for line in fp:
    print(line, end='')

aaa,aaa2,aaa3
bbb bbb2 bbb3
ccc ccc2 ccc3
ddd ddd2,ddd3

Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> with open('c:\\\\in.txt') as fp:
    for line in fp:
        print(line,end='')

aaa,aaa2,aaa3
bbb bbb2 bbb3
ccc ccc2 ccc3
ddd ddd2,ddd3
>>>
```

Lab#1-6c-File用轉向stdin,stdout的方法讀資料

```
*Python 3.8.1 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18
C v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "h
nformation.
>>> import sys
>>> temp=sys.stdout
>>> fp_stdout=open('stdout.log','w')
>>> sys.stdout=fp_stdout
>>> print('HI!!!!')
>>> print('20200202')
>>> sys.stdout.close()
>>> print('hi')
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    print('hi')
ValueError: I/O operation on closed file.
>>> sys.stdout=temp
>>> print('hi')
hi
>>> print('要先記在temp，才有正確console可返回')
要先記在temp，才有正確console可返回顯示
>>>
```

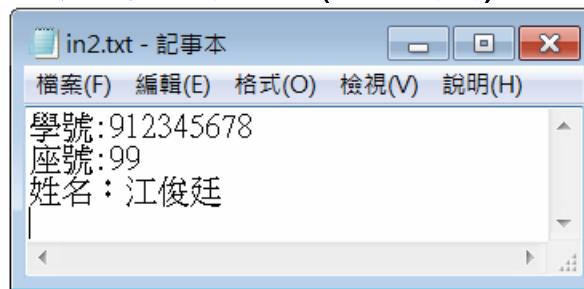
```
#把檔案轉入，就可直接當作鍵盤輸入使用。py - D:/gcj2018_rj/#把檔案
File Edit Format Run Options Window Help
#####
#轉入為檔案一成功#####
import sys
#把檔案轉入，就可直接當作鍵盤輸入使用。
try:
    fp = open('c:\\\\in.txt')
    sys.stdin=fp
    i=0
    while True:
        i+=1
        #x=input(str(i))
        x=input(str(i)+"-->")
        print(x)
except:
    print("exex")
    fp.close()
finally:
    print("fi")
>>>
```

```
Python 3.8.2 Shell
File Edit Shell Debug Opt
py =====
1-->hi 202002021
2-->hi 202002021
3-->aaa,aaa2,aaa3
4-->bbb bbb2 bbb3
5-->ccc ccc2 ccc3
6-->ddd ddd2,ddd3
7-->exex
fi
>>>
```

```
out_from_stdout轉向為檔案.py - D:/gcj2018_rj/out_from_stdout轉向為檔案.py (3.8.2)
File Edit Format Run Options Window Help
#####
#轉出，由stdio轉為檔案一成功#####
#將標準輸出從 console 轉到 檔案名稱：stdout.log
import sys
temp=sys.stdout #要先記在temp，完成後，才可還原被轉向的stdout
try:
    fp_stdout = open('stdout.log', 'w')
    sys.stdout = fp_stdout
    # 輸出字串，但是不會在 console 顯示
    #   ，而是被寫到 stdout.log 這個檔案
    print('Test output to ./stdout.log')
except:
    print("exex")
finally:
    print("fin")
    fp_stdout.close() #要close()才算檔案完成。
    sys.stdout = temp #此時要把被轉向的stdout還原為原始的正常情況。
```

Exercises: Ex1-6

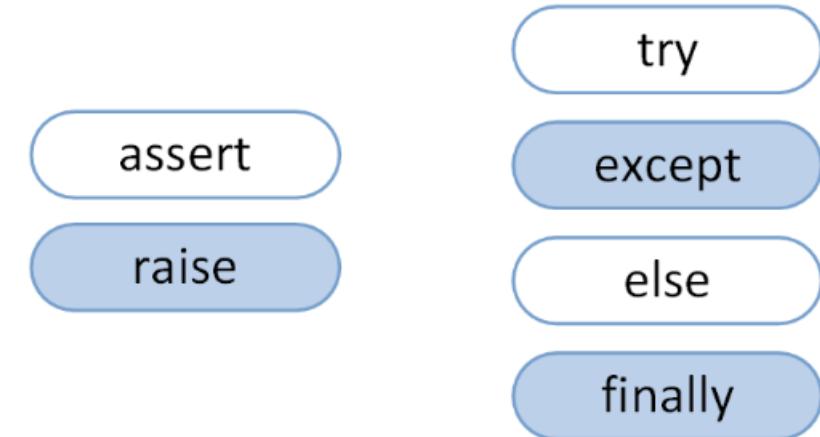
- Ex1.6 Add1: 請問要產生簡單文字檔的(a)簡單三步驟是什麼？(b)如果要在文字檔中，寫入三行資料，是自己的學號、座號、姓名(如右圖)。可執行的完整程式要如何寫？



- Ex1.6 Add2: 如果有一個文字檔，檔名是c:\\in.txt. 請問可以一行一行逐步讀入資料的程式要如何寫？
- Ex1.6 Add3: 一般程式語言，(a)所謂的「標準輸入」「標準輸出」是什麼？(b)請問python裡要如何引用標準輸出就可以轉出到特定檔案文件，又要如何引用標準輸入？簡單寫法是什麼？
- Ex1.6 Add4: 檔案，(a)可分兩類，第一類是文字文件(例如utf-8格式)，第二類是二進位檔式的binary. 請問在開檔上，如何分辨？(b)二進位檔要如何設定位置？如何查知目前位置？(c)請舉實例簡單示範binary的操作。
- Ex1.6 Add5: python建議開檔是用with取代open主要好處有那些？
- Ex1.6 Add6: 在Windows中使用記事本編輯有中文的文字檔案時，預設的編碼是cp950, 若要python開檔，對於中文字常會有解碼錯誤的問題。請問可行的解決方案是？
- 作業題：**Ex1.6Add1 ~~**
- 補充：

1.7 Exception Handling(例外處理)

- 例外(或稱為異常)(Exceptions)是在程序執行期間發生的意外事件。
- 邏輯錯誤或意外情況，都可能導致異常。
- 在Python中，異常（也稱為錯誤）也是一種物件(object)，是由遇到意外情況的代碼所引發(raised)或稱為丟出(thrown)。
 - Python解譯器也可以引發異常。
- 周圍環境會以適當的方式“處理(Handles)”異常，從而引發錯誤。如果未被捕獲，則異常會導致解釋器停止執行(stop)程序，並向控制台報告適當的消息。



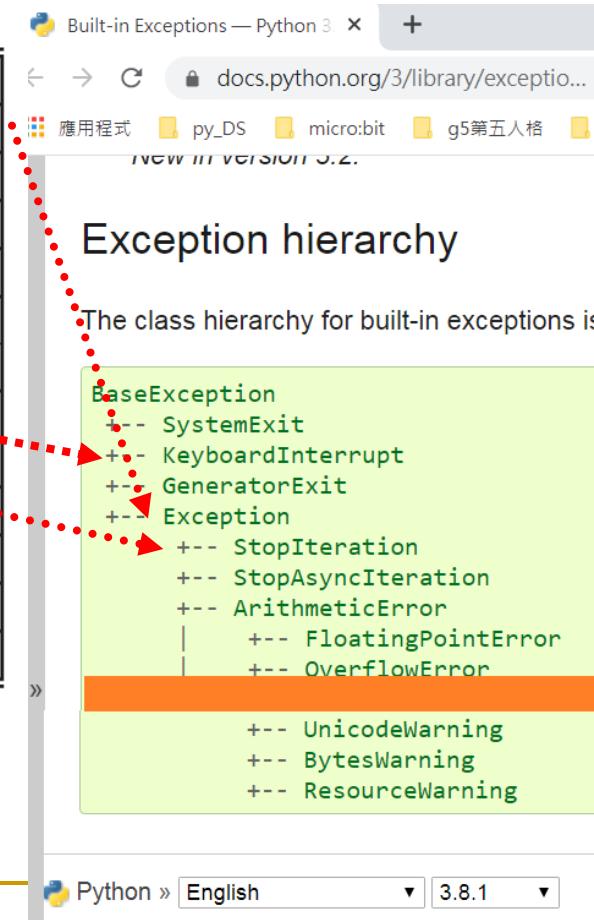
Common Exceptions(例外)(或稱為異常)

IOError已
被併到
OSSError

Python有豐富的異常類別，甚至可用分層
(hierarchy)結構來說明，用來指定各種類別的錯誤

Class	Description
Exception	A base class for most error types
AttributeError	Raised by syntax <code>obj.foo</code> , if <code>obj</code> has no member named <code>foo</code>
EOFError	Raised if “end of file” reached for console or file input
IOError	Raised upon failure of I/O operation (e.g., opening file)
IndexError	Raised if index to sequence is out of bounds
KeyError	Raised if nonexistent key requested for set or dictionary
KeyboardInterrupt	Raised if user types <code>ctrl-C</code> while program is executing
NameError	Raised if nonexistent identifier used
StopIteration	Raised by <code>next(iterator)</code> if no element; see Section 1.8
TypeError	Raised when wrong type of parameter is sent to a function
ValueError	Raised when parameter has invalid value (e.g., <code>sqrt(-5)</code>)
ZeroDivisionError	Raised when any division operator used with 0 as divisor

因為全部例外都是BaseException的子類別，所以
`except`後面沒加類型，就會包含Ctrl+C。要注意，
`ctrl+C`是不算在Exception類之內哦！



1.7.1 Raising an Exception(丟出例外)

- 通過執行**raise**語句會引發例外，並將例外類的實例(**instance**)作為指定問題的參數。
- 例如下面的例子：
 - 有一個用來計算平方根的函數，卻傳發送為負值作為參數，則該函數應該會引發以下例外(異常)：
`raise ValueError('x cannot be negative')`

exception ValueError

Raised when an operation or function receives an argument that has the **right type but an inappropriate value**, and the situation is not described by a more precise exception such as [IndexError](#)

example

- 檢查函數sqrt()參數是否有效：
驗證參數的(1)類型(Type) (2)值(Value)。

```
def sqrt(x):
    if not isinstance(x, (int, float)):
        raise TypeError('x must be numeric')
    elif x < 0:
        raise ValueError('x cannot be negative')
    # do the real work here...
```

Type 必須是int 或 float

Value範圍必須「非負數」

- 前述的sqrt()，在驗證Type與Value後，才進入正式程式。

- 另外Sum的例子：左邊是詳細版，右邊是幾乎同效果精簡版
(因為會有預設的例外處理(雖沒明顯的例外
檢查，仍隱藏有檢查效果))

```
def sum(values):
    if not isinstance(values, collections.Iterable):
        raise TypeError('parameter must be an iterable type')
    total = 0
    for v in values:
        if not isinstance(v, (int, float)):
            raise TypeError('elements must be numeric')
        total = total + v
    return total
```

```
def sum(values):
    total = 0
    for v in values:
        total = total + v
    return total
```

會發生TypeError的例外。
因為total初值=0，
所以error是: int + string

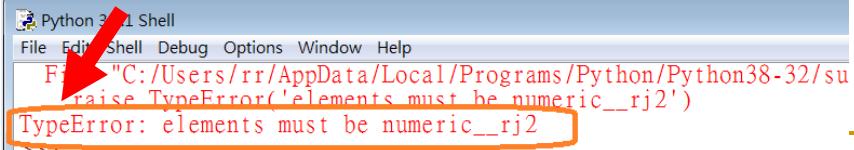
```
def sum(vals):
    total=0
    for v in vals:
        total=total +v
    return total
print("a",end='')
print(sum([1,2]))
print("b",end='')
#print(sum([1,"xyz"])) # int +str error!
print()
print("c",end='')
print(sum(["uvw", "xyz"]))

Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>= RESTART: C:/Users/rr/AppData/Local/Programs/Python/Python38-32/ssum.py
a3
b
cTraceback (most recent call last):
      total=total +
      TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> |
```

Lab#1.7_Exception

- 註，因為版本更新，所以collection要改為collection.abc

```
def sum(values):
    total = 0
    for v in values:
        if not isinstance(v, (int, float)): #必須是int 或float
            raise TypeError('elements must be numeric__rj2')
        total = total + v
    return total
print( sum( [1,2] ))
print( sum( [1, 2, 3.14] ))
##print( sum(3.14) ) #==>TypeError: parameter must be an iterable
print( sum( [1, "bbb"] ))#==>TypeError: elements must be numeric__rj2
```



A screenshot of the Python 3.8.1 Shell window. The title bar says "Python 3.8.1 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The command line starts with "In [1]: ". The user types "sum([1, 2])" and presses Enter. The output shows the function definition and then "Out[1]: 3". The user then types "sum([1, 3.14])" and presses Enter. The output shows the function definition again, followed by "raise TypeError('elements must be numeric__rj2')", and then "TypeError: elements must be numeric__rj2". A red arrow points from the explanatory text above to the "raise" line in the code, and another red arrow points from the explanatory text to the error message in the shell.

```
In [1]: def sum(values):
    total = 0
    for v in values:
        if not isinstance(v, (int, float)): #必須是int 或float
            raise TypeError('elements must be numeric__rj2')
        total = total + v
    return total
Out[1]: 3
In [2]: sum([1, 3.14])
raise TypeError('elements must be numeric__rj2')
TypeError: elements must be numeric__rj2
```

Catching an Exception(捕捉例外)

- 處理例外的兩種理念：第一是三思而後行，想完全避免例外。第二是不維護可能發生的例外，而是等例外發生，有處理機制就可以了！
- Python處理例外的機制，是用try-except控制結構來測試和捕獲例外/異常。

try:

```
ratio = x / y
except ZeroDivisionError:
    ... do something else ...
```

- 在這種結構中，“try”區塊是要執行的主要代碼。
- 此處的例子中，例外時，會執行單個命令(也可以有區塊)。
- 在try塊之後是一個或多個“例外”情況，每種情況都有一個已標識的錯誤類型和一個縮進的代碼區塊，如果在try區塊內出現指定的錯誤，則執行該對應的程式碼。
- 最後一個except子句「可省略」例外名稱，代表任一例外。

Exercises: Ex1-7

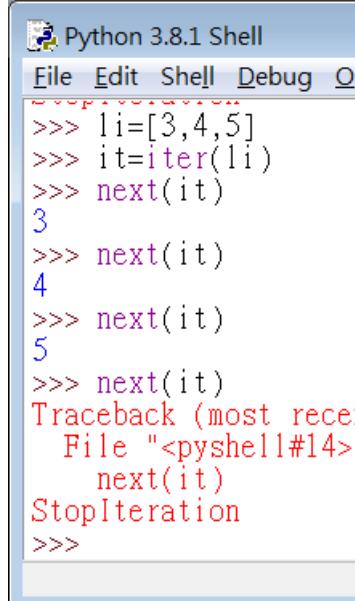
- Ex1.7Add1: 處理例外(a)有兩種理念是什麼？(b)python是採用那種程式結構來達成。
- Ex1.7Add2: 函數的參數檢查中，最常見的驗證是要驗證參數的類型(Type)與值(Value)，請以sqrt()求根號為例，寫出可以例外檢查的主要程式碼。
- Ex1.7Add3: 有一個Sum函數如下：
 - def sum(vals):
 - total=0
 - for v in vals:
 - total=total +v
 - return total
 - print(sum([1, 2.4]))
 - print(sum(["uvw", "xyz"]))
- 執行後可能發生什麼例外？請簡單解釋原因。
- 作業題： Ex1.7Add1 , Ex1.7Add2, Ex1.7Add3
- 補充：

1.8 Iterators(疊代運算子)與Generators(生成器)

- 基本的容器類型(container types) (例如list, tuple和set) 符合可疊代的類型(iterable types)，這使它們可以在for循環中用作可疊代的對象。

for element in iterable:

- 疊代器(iterator)是通過一系列值管理疊代的對象。如果變量it標識一個疊代器對象，則每次對內置函數next(it)的調用都會從基礎系列中生成一個後續元素，若沒下一個元素，就會引發StopIteration例外的異常。
- 一個可疊代對象是一個對象obj，它通過語法iter(obj)就可生成一個疊代器obj。



The screenshot shows a Python 3.8.1 Shell window. The code demonstrates the creation of an iterator from a list and its subsequent iteration using the next() function. The output shows the values 3, 4, and 5 being printed sequentially, followed by a StopIteration exception.

```
Python 3.8.1 Shell
File Edit Shell Debug ?
>>> li=[3,4,5]
>>> it=iter(li)
>>> next(it)
3
>>> next(it)
4
>>> next(it)
5
>>> next(it)
Traceback (most recent call last):
  File "<pyshell#14>" in <module>
    next(it)
StopIteration
>>>
```

Generators(生成器)

- 在Python中創建疊代器最方便的技術是使用生成器(generators)。
- ※生成器，一樣def開始，但發現有yield關鍵字，就與一般def不一樣行為哦！
- 實現生成器的語法與傳統函數非常相似，但是必須執行yield語句，來傳回該系列的每個元素，而不是return。
- 下面示範，由傳統方法改寫出生成器(印出全部因數)：
 - 參考傳統方法：如果n=100，則results =[1,2,4,5,10,20,25,50,100]

```
def factors(n):          # traditional function that computes factors
    results = []           # store factors in a new list
    for k in range(1,n+1):
        if n % k == 0:      # divides evenly, thus k is a factor
            results.append(k) # add k to the list of factors
    return results          # return the entire list
```

- factor公因數生成器方法1、直接改為生成器(幾乎就是把return 改為yield)

```
def factors(n):          # generator that computes factors
    for k in range(1,n+1):
        if n % k == 0:      # divides evenly, thus k is a factor
            yield k           # yield this factor as next result
```

- 此時，factors(100)會得到1,2,4,5,10,20,25,50,100

Factor()公因數生成器方法2：改為加速版

- 比較前面兩個程式，可看出，生成器的程式，就是在產生資料的地方改為**yield**。再把**return**也改為**yield**。
- 表面上看**yield**，有點像**return**，但是函式不會因**yield**而結束，只是暫時將流程控制權暫時交給函式呼叫者。
- **有加**yield**的函數，就像是可暫停執行的函數**(這是需要OS協助Swap程序哦！)，而且下次呼叫調用函數，就會由暫停處繼續執行！
- 當生成器達到最後一個元素，或程式末尾(也可以是0參數的**return**)會丟出**StopIteration**例外。
- 為了提高效率，前頁**factors**改寫如右圖：
 - 此時，**factors(100)**會得到1,100,2,50,4,25,5,20,10。
 - (因為k=1時， $100//1$ 會是100) ，(因為k=2時， $100//2$ 會是50)

```
def factors(n):      # generator that computes factors
    k = 1
    while k * k < n: # while k < sqrt(n)
        if n % k == 0:
            yield k
            yield n // k
        k += 1
    if k * k == n:    # special case if n is perfect square
        yield k
```

example1 :疊代物件iter()

- lst=[1,3,4]
- it物件=iter(lst) #產生疊代物件，類似 generator(生成器)物件
- while True:
 - try:
 - print(next(it物件))
 - except Exception as e:
 - print(e)
 - break

```
: 1 lst=[1,3,4]
 2 it物件=iter(lst) #產生疊代物件，類似 generator
 3 while True:
 4     try:
 5         print(next(it物件))
 6     except Exception as e:
 7         print(e)
 8         break
```

1
3
4

Example2:疊代物件iter() + next()

- # diction字典的 comprehension()
- d = {0: '7', 1: '9', 2: 'a', 3: 'b', 4: '9'}
- # 使用生成器表達式找到值為 '9' 的鍵

- # 類似list-comprehension的 generator-comprehension () 產生「生成器物件」以後可以用 next弓|用！
- gen0=(key for key, value in d.items() if value == '9')
- print(f'gen->{list(gen0)}') #讀過就沒資料了。---有點像「量子電腦」讀過就確定狀態！
- gen=(key for key, value in d.items() if value == '9')
- while True:
- key_to_find = next(gen, None) #None是defaultp值
- if key_to_find == None:
- break
- print(f'key_to_find->{key_to_find}')
- # gen->[1, 4]
- # key_to_find->1
- # key_to_find->4

example3生成器_反轉reverse() +next()

- def reverse(data):
- for index in range(len(data)-1, -1, -1):
- yield data[index]
- for c in reverse('abcde'):
- print(c)

- def reverse(data):
- for index in range(len(data)-1, -1, -1):
- yield data[index]
- rrObj=reverse('abcde') # 第一次呼叫def是
- aa=rrObj.__next__() #以後，遇到 __next__
- print(f'yield傳回的第一個資料是={aa}')
- for c in rrObj:
- print(c)

```
: 1 def reverse(data):
2     for index in range(len(data)-1, -1, -1):
3         yield data[index]
4 for c in reverse('abcde'):
5     print(c)
```

e
d
c
b
a

```
: 1 def reverse(data):
2     for index in range(len(data)-1, -1, -1):
3         yield data[index]
4 rrObj=reverse('abcde') # 第一次呼叫def是產生「生成器」
5 aa=rrObj.__next__() #以後，遇到 __next__()就傳回一個值
6 print(f'yield傳回的第一個資料是={aa}')
7 for c in rrObj:
8     print(c)
```

yield傳回的第一個資料是=e
d
c
b
a

Exercises: Ex1-8

- Ex1.8Add1: 請簡述，由傳統程式簡單改為生成器(generators)的方法。
- Ex1.8Add2: `yield`是python裡面三十多個重要保留字之一，可用來寫出生成器(generators)的程式，請簡介`yield`的作用是什麼？
- Ex1.8Add3: python的疊代子(Iterators)是在容器可用for語句循環的運算子。請設計一個簡單的實驗。
 - (a) 請以字串‘xyz’為例子，執行`iter()`運算子與`next()`運算子的操作，
 - (b) 請問`next`到最後會出現什麼例外？
- Ex1.8Add4: python的生成器(Generators)好用又好寫，只要用`yield`返回所要資料即可。主要理由是因為python會記住所有數據值以及最後執行的語句。所以每次呼叫`next()`，生成器都會從上次中斷的地方繼續。另外一個生成器可以很簡單完成的原因是`__iter__()`and `__next__()`方法是會自動創建！。請設計一個簡單的生成器。目的是把可以把輸入的序列資料反轉。
- Ex1.8Add5: python生成器，一樣`def`開始，但發現有`yield`關鍵字，就與一般`def`不一樣行為，(a)請說明生成器是什麼？(b)並且比較生成器與一般函式的區別
- Ex1-C1.27章末●: 在討論factors時，有討論到`factors(100)`會得到1,100,2,50,4,25,5,20,10，因為產生的順序不是由小到大，請改寫這個生成器`factors(n)`，讓他的元素，可以按照由小到大的順序產生。
- ※可以有無限大範圍的生成器。

- 作業題： Ex1.8Add1,~ Ex1.8Add5, Ex1章末C1.27，
- 補充：

1.9 其他便利的寫法

Conditional Expressions, Comprehension, pack, unpack

- Python 支持可以替換簡單控件結構的條件表達式語法(conditional expression syntax)。
 - (1) 三元運算子(**ternary conditional operator**)是 Pythonic 寫法之一
 - 通用語法是下列形式：`expr1 if condition else expr2`
 - 如果條件為真，則此複合表達式的計算結果為expr1，否則，計算 `param = n if n >= 0 else -n` *# pick the appropriate value*
 - 例如：`result = foo(param)` *# call the function*
- `result = foo(n if n >= 0 else -n)`

※註：在py3.8版，新加入「海象語法:=」，就可以在if的條件內放入assign的指定敘述，有機會更精簡！

●●Comprehension Syntax(理解語法)

- (2) **List Comprehension**列表推導 或 列表理解！
~~~ 這是很 **python**風格(**Pythonic**)的寫法！
- 在程設中，是常見任務特徵：任務是根據一系列資料，可推導出另一個系列資料。(也代表list使用上的高度理解！)
- 通常，這時候**List Comprehension**就可簡單完成任務。
- 下列是 **list**的理解語法  
*[ expression for value in iterable if condition ]*
- 下列程式與前一行是等效的。

```
result = []
for value in iterable:
    if condition:
        result.append(expression)
```

例子之1: 1到n的平方的列表

```
squares = []
for k in range(1, n+1):
    squares.append(k*k)
```

下列是等效寫法:

```
squares = [k*k for k in range(1, n+1)]
```

## 例子之2: 根據一系列資料，推導出另一系列資料

- 例如：從0~9資料中拿出偶數的資料

- `[x for x in range(10) if x % 2 == 0]`

```
1 [ x for x in range(10) if x % 2 == 0]
[0, 2, 4, 6, 8]
```

- 例如：將偶數乘以 2，並篩選出大於 10 的數

- `[n * 2 for n in range(10) if n % 2 == 0 if n * 2 > 10]`

```
[12, 16]
```

- 例如：將偶數放入字典，鍵為偶數本身，值為其平方

- `{n: n**2 for n in range(1,11) if n % 2 == 0}`

- `{n: n**2 for n in range(1,11) if n % 2 == 0}`

```
{2: 4, 4: 16, 6: 36, 8: 64, 10: 100}
```

# 例子之3：要產生因數的列表factor

As a second example, Section 1.8 introduced the goal of producing a list of factors for an integer  $n$ . That task is accomplished with the following list comprehension:

```
factors = [k for k in range(1,n+1) if n % k == 0]
```

Python supports similar comprehension syntaxes that respectively produce a set, generator, or dictionary. We compare those syntaxes using our example for producing the squares of numbers.

|                                    |                          |
|------------------------------------|--------------------------|
| [ k*k for k in range(1, n+1) ]     | list comprehension       |
| { k*k for k in range(1, n+1) }     | set comprehension        |
| ( k*k for k in range(1, n+1) )     | generator comprehension  |
| { k : k*k for k in range(1, n+1) } | dictionary comprehension |

Comprehension不只是list, 包括set, generator, dictionary也可用這個語法。

- 若不需將結果存儲在記憶體，生成器(generator) 語法特別具吸引力！
  - 例如，要計算1到 $n$ 平方後的和，最好使用生成器語法total = sum(k\*k for range(1,n+1))，會優於用顯式實例化列表理解作為參數哦！

# Packing (automatic packing)自動打包

- 如果在較大的程式文件中，有一系列用逗號分隔的表達式，則即使沒有提供任何括號，也可「自動」將它們視為一個元組(tuple)。
- 例如，考慮指定運算(assignment)

```
data = 2, 4, 6, 8
```

- 這導致將標識符data被指定為元組(2,4,6,8)。此行為稱為元組的自動打包(**automatic packing of a tuple**)。
- ※等號左邊，也有類似效果：
  - 例如： a,b=10,30，結果是a=10, b=30
  - ※用;分行，可得到相同效果： a=10; b=30

# Unpacking(解包)

- 作為打包行為的對偶功能，Python可以自動解壓縮序列(**automatically unpack a sequence**)，從而允許將一系列單獨的標識符分配給序列的元素。
  - 例如，程式：`a,b,c,d=range(7,11)`
    - 就會有指定`a = 7, b = 8, c = 9`和`d = 10`的效果。
  - 例如：`for x, y in [ (7, 2), (5, 8), (6, 4) ]:`
    - 就會指定`(x,y)=(7, 2), (x,y)=(5, 8)...`的效果。

## Simultaneous Assignments(同時分配)

，就是同時打包又解包。例如：`x,y,z=6,7,8`

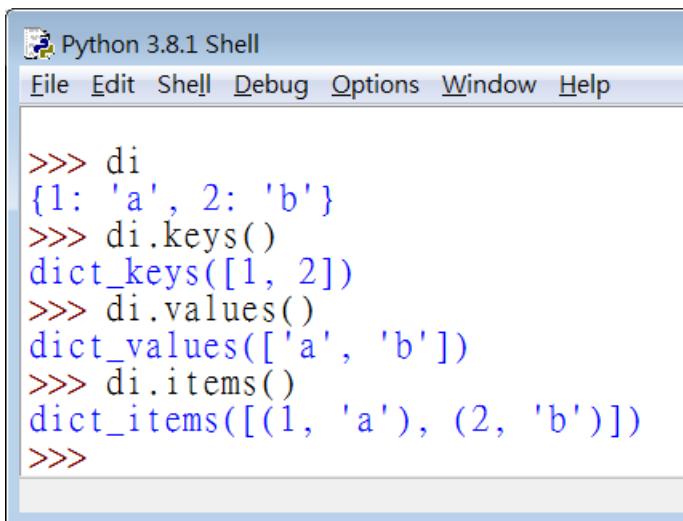
因為同時分配時，是先計算完右手邊，再分給左手邊，所以可以`x,y=y,x`進行swap的動作！

# Exercises: Ex1-9

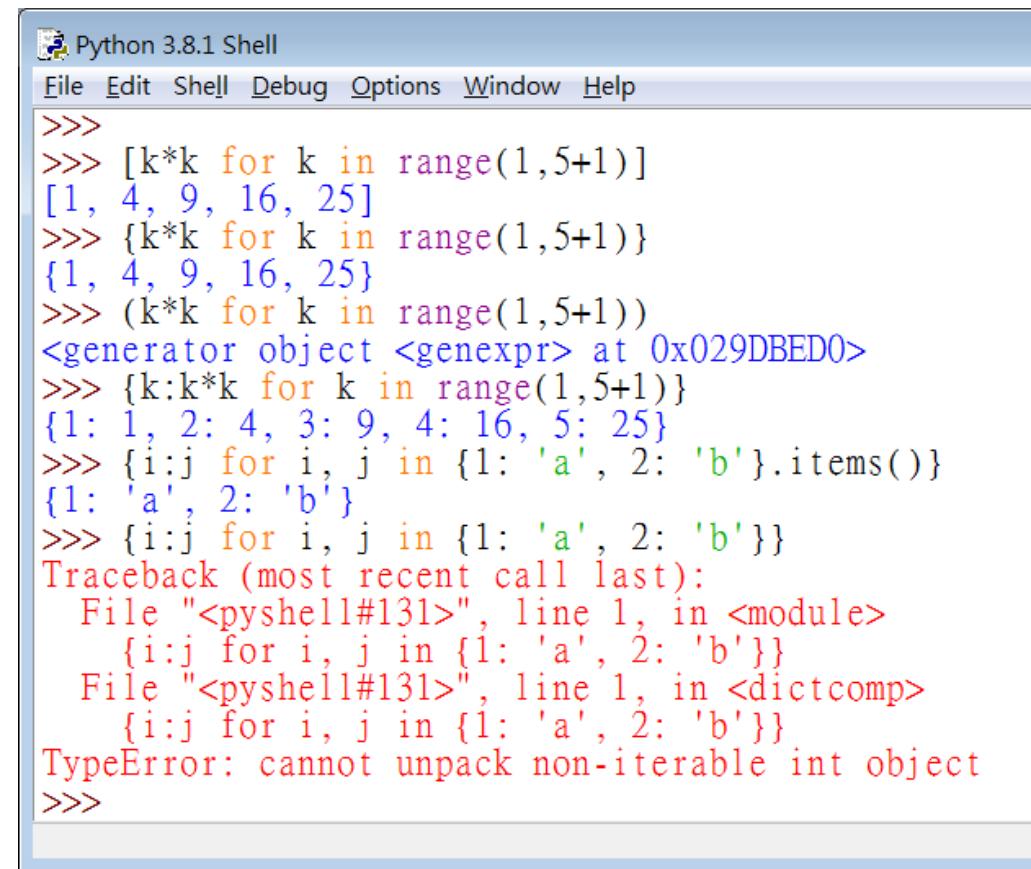
- **Ex1.9Add1:**如果程式任務具有下列特徵：任務是根據另一個list處理後的結果，再來產生一系列的list當作輸出。(a)此時可考慮用什麼語法？(b)請舉出不用此語法與使用此語法的比較。
- **Ex1.9Add2:**寫一個函數最後會有return，正常是傳回一個變數或一個物件，但是在python可以寫成 return a,b,c，請問這是那種python的那種特性？會視為傳送那種資料型態？
- **Ex1.9Add3:**一般在進行排序演算法設計時，都會用到swap(交換)，請問一般程式語言(例如 c++)，要兩變數交換的寫法是什麼？而在python，因為有自動打包(Packing)與解包(unpacking)，所以兩變數交換的寫法是什麼？
- **Ex1-R1.5:**章末● :用理解語法(comprehension syntax)寫出一行程式的小函數，可返回所有小於n的正整數的平方和。※可使用內置的sum函數。(與R1.4同)
- **Ex1-R1.7**章末● :用理解語法(comprehension syntax)寫出一行程式的小函數，僅需要一個正整數n，並返回所有小於n的奇數的平方和。※(目的與R1.6同)
- **EX1-C1.18** ●章末 Demonstrate how to use Python's list comprehension syntax to produce the list [0, 2, 6, 12, 20, 30, 42, 56, 72, 90].
- **EX1-C1.19** ●章末 Demonstrate how to use Python's list comprehension syntax to produce the list [a ,b, c, ...,z ] , but without having to type all 26 such characters literally.
- 作業題： Ex1.9Add1, Ex1.9Add2 , Ex1.9Add3, Ex1-R1.5:章末●, Ex1-R1.7:章末●, EX1-C1.18●章末, EX1-C1.19●章末
- 補充：

## ● 【check3】Lab#1-9-Comprehension語法測試

- ※注意1，沒有tuple comprehension哦！而是generator comprehension哦！
- ※注意2，Dict取值要加上`.items()`才是正確寫法。



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> di
{1: 'a', 2: 'b'}
>>> di.keys()
dict_keys([1, 2])
>>> di.values()
dict_values(['a', 'b'])
>>> di.items()
dict_items([(1, 'a'), (2, 'b')])
>>>
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>>
>>> [k*k for k in range(1,5+1)]
[1, 4, 9, 16, 25]
>>> {k*k for k in range(1,5+1)}
{1, 4, 9, 16, 25}
>>> (k*k for k in range(1,5+1))
<generator object <genexpr> at 0x029DBED0>
>>> {k:k*k for k in range(1,5+1)}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
>>> {i:j for i, j in {1: 'a', 2: 'b'}.items()}
{1: 'a', 2: 'b'}
>>> {i:j for i, j in {1: 'a', 2: 'b'}}
Traceback (most recent call last):
  File "<pyshell#131>", line 1, in <module>
    {i:j for i, j in {1: 'a', 2: 'b'}}
  File "<pyshell#131>", line 1, in <dictcomp>
    {i:j for i, j in {1: 'a', 2: 'b'}}
TypeError: cannot unpack non-iterable int object
>>>
```

## 1.10 Scopes and Namespaces(作用範圍和命名空間)-(※變數的生命週期、活動空間)

- Python使用變數名稱前，要先確定是否有定義，例如要計算 $x + y$ 總和時，名稱x和y必須事先與用作值的對象相關聯；如果找不到這樣的定義，將會引發**NameError**。
- 確定與標識符關聯的值的過程稱為名稱解析(**Name resolution**)。

# 變數範圍：

- 在python無需事先宣告就可直接使用變數
- 所以：
  - 在某名稱(變數)指定值時，就可自動成為變數，並建立自己的範圍(Scope)
  - 在取用某變數時，會先看目前範圍(local)，若無則向外尋找。(而且會以module為界)
  - 變數簡單分為兩個範圍：全域(Global), 區域函式(local function)。
- 每當將ID標識符「分配」給一個值時，該定義就會在特定範圍內進行
- 「最外層的分配」通常在所謂的全域範圍內進行。
- 在「函數主體內的分配」，通常具有該函數調用本地的作用域(local)。
  - 因此，函數內的賦值(例如 $x = 5$ )，在更大範圍外，是對標識符x不認識。因為x的活動範圍只在函數內。

# Scope(變數作用範圍)

在函數內部和外部，如果同時，都有「相同變量名」進行操作，Python會將它們視為兩個單獨的變量，

- Local(區域範圍): 在函數內部創建的變量屬於該函數，只能在該函數內部使用。

- ```
def myfunc():
    x = 300
    print(x)
myfunc()
```

```
def myfunc():
    x = 300
    print(x)
myfunc()
300
>>>
```

- Global(全域範圍)

- 在Python代碼主體中創建的變量是全局變量，屬於全局範圍。
- 全局變量可從任何範圍（全局和局部）中獲得。

- ```
X=300
def myfunc():
    print(x)
myfunc()
print(x)
```

```
x =300
def myfunc():
    print(x)
myfunc()
print(x)
300
300
>>>
```

## 全域關鍵字global

global關鍵字可以使函數內變量當作全域變量：

```
def myfunc():
    global x
    x = 300
myfunc()
print(x)
```

```
def myfunc():
    global x
    x = 300
myfunc()
print(x)
300
>>>
```

在函數內部，若要更改全域變量，請使用global關鍵字引用該變量：

```
x = 300
def myfunc():
    global x
    x = 200
myfunc()
print(x)
```

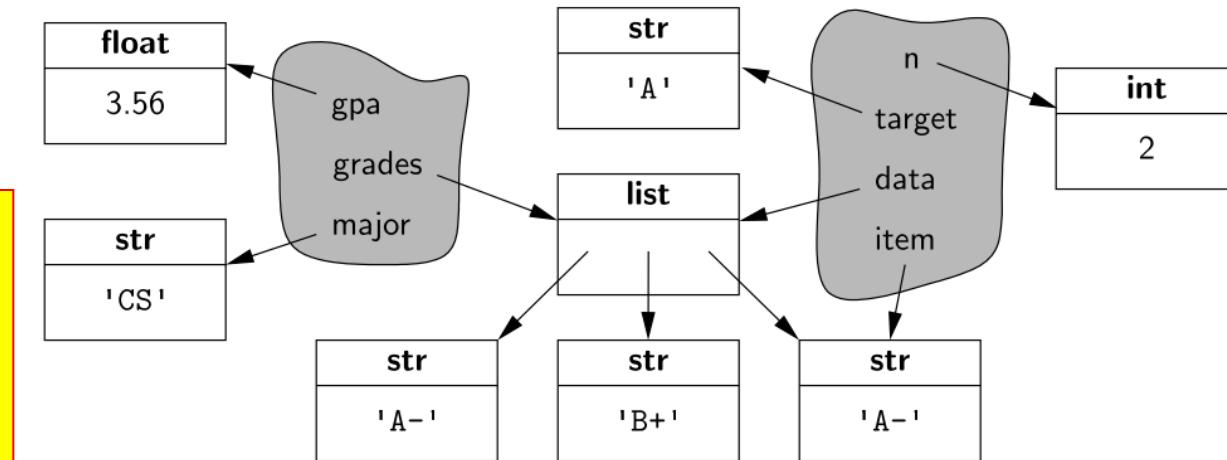
```
x=300
def myfunc():
    global x
    x = 200
myfunc()
print(x)
200
>>>
```

# Namespace(命名空間)

- Python中每個不同的作用域都使用稱為名稱空間(name space)的抽象表示。命名空間管理當前在給定範圍內定義的所有標識符。
- 圖1.8描繪了兩個名稱空間，一個是在1.5節中調用我們的**count**函數的名稱空間，另一個是在**執行該函數期間的本地名稱空間**.

在最接近本地  
區域範圍內  
搜索名稱。

若此處找不到  
，則搜索下  
一個外部作  
用區域。



# 用dir()和vars()查詢給定空間的命名集

- Python使用自己的字典(dictionary)實現名稱空間，該字典將每個標識字符串（例如n）映射到其關聯值。Python提供了幾種檢查給定名稱空間的方法。
- 函數dir()報告給定名稱空間中標識符的名稱（即dictionary的key）
- 函數vars()返回完整的字典。
- 默認情況下，對dir()和vars()的調用，會在執行它們的最接近本地的名稱空間上報告。
- 當在命令中指示標識符時，Python在名稱解析過程中搜索一系列名稱空間。
  - 先在本地的範圍內搜索名稱。
  - 如果在此處找不到，則搜索下一個外部作用域，依此類推。

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'a', 'b', 'c', 'func', 'gg']
>>> dir(__builtins__)
['ArithError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'Connecti
Ln: 533 Col: 4
```

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
==== RESTART: C:/Users/rr/AppData/Local/Programs/Python/Python38-32/global.py ====
{'a': 901, 'b': 902, 'c': 903, 'f': 10, 'f2': 12, 'f3': 13}
-----
{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__': <class '_frozen_importlib.BuiltinImporter'>, '__spec__': None, '__annotations__': {}, '__builtins__': <module 'builtins' (built-in)>, '__file__': 'C:/Users/rr/AppData/Local/Programs/Python/Python38-32/global.py', 'a': 1, 'b': 2, 'c': 3, 'gg': 9020, 'func': <function func at 0x02A001D8>}
=====

global.py - C:/Users/rr/AppData/Local/Programs/Python/Python38-32/global.py (3.8.1)
File Edit Format Run Options Window Help
a=1
b,c=2,3
gg=9001
def func():
    a,b,c=901,902,903 #會是新區域變數！
    f=10
    f2=12
    f3=13
    global gg #才可用到fun以外的廣域變數！
    gg=9020
    print(vars())
    print("----")
print("^^"*20)
func()
print(vars())
print("=="*20)
Ln: 15
```

# First-Class Objects(第一級物件)

- 在程式語言的術語中，第一級物件，指的是可分配給標識符(ID)，也可直接作為參數傳遞，或由函數返回的一種資料類型的實例。
  - 也表示可將第一級物件放在列表，並作為其他函數的參數。
- 在1.2.3節，曾介紹的所有資料類型(例如int, float, list, string)顯然都是Python中的第一級物件。
- Python中，函數(function)和類別(class)也被視為第一級物件。
  - 例如，我們可以編寫以下代碼：

```
scream = print      # assign name 'scream' to the function denoted as 'print'  
scream('Hello')    # call that function
```

    - 在這種情況下，我們沒有創建新函數！我們只是將scream定義為現有打印函數的別名。
- 另一個常見，是用函數當第一級物件的例子：
  - 例如：max(a,b, key=abs)就是把abs函數當作一級物件參數來傳遞。

The screenshot shows the Python 3.8.1 Shell window. The title bar says "rand.py - C:/Users/rr/AppData/Local/Temp". The menu bar includes File, Edit, Format, Run, and Options. The main area displays the following code and its output:

```
scream = print  
scream('Hello')  
印出 = print  
def func4():  
    return 4  
印出(func4()) # 4  
印出(func4) # <function func4 at 0x029C1CD0>  
also_func4 = func4  
印出(also_func4())  
印出(also_func4)
```

The screenshot shows a Windows Notepad window titled "rand.py - C:/Users/rr/AppData/Local/Temp". The code in the window is identical to the one shown in the Python shell:

```
scream = print  
scream('Hello')  
印出 = print  
def func4():  
    return 4  
印出(func4()) # 4  
印出(func4) # <function func4 at 0x029C1CD0>  
also_func4 = func4  
印出(also_func4())  
印出(also_func4)
```

# Exercises: Ex1-10

- Ex1.10Add1: First-Class Objects(第一級物件)是可以當作函數的參數，也可當作函數return的資料，請問在python中有那些First-Class Objects(第一級物件)?
- Ex1.10Add2: 第一類函數(First-Class Objects)可以幫助我們非常方便的實現很多有用的功能。主要的特徵是那四點？
  
- 作業題： Ex1.10Add1, Ex1.10Add2
- 補充：

# 1.11 Modules and the Import Statement

## (模組區塊, 簡稱為模塊)

- 除了**內定定義(built-in definitions)**之外，標準的Python發行版，在外加的程式庫(也可稱為**modules**)，還可能包含有成千上萬的其他的值(**values**)、函數(**functions**)和類別(**classes**)。這些值，可以從其他程序中導入(**import**)。
  - 例如：
    - `import math`
  - 如果：不僅要導入模塊名稱，也要導入模組內的函數名稱，可以配合 `from 模組 import *`：
    - `from math import *`
  - 備註：
    - ※ 唯一**不需import就可直接使用的函數是built-in function**

# Existing Modules(現有可用的模塊)

Python可import這些模塊。(下列是常用而且與DS(資料結構)相關的現有模塊)。還有很多要先自行安裝進python後再import才可使用的模塊。

- Some useful existing modules include the following:

為原始類型提供緊湊的數組(array)存儲。

| Existing Modules |                                                                                                |
|------------------|------------------------------------------------------------------------------------------------|
| Module Name      | Description                                                                                    |
| array            | Provides compact array storage for primitive types.                                            |
| collections      | Defines additional data structures and abstract base classes involving collections of objects. |
| copy             | Defines general functions for making copies of objects.                                        |
| heapq            | Provides heap-based priority queue functions (see Section 9.3.7).                              |
| math             | Defines common mathematical constants and functions.                                           |
| os               | Provides support for interactions with the operating system.                                   |
| random           | Provides random number generation.                                                             |
| re               | Provides support for processing regular expressions.                                           |
| sys              | Provides additional level of interaction with the Python interpreter.                          |
| time             | Provides support for measuring time, or delaying a program.                                    |

# Pseudo-Random Number Generation

- 簡單的偽隨機數生成器公式:  $\text{next} = (\text{a} * \text{current} + \text{b}) \% \text{n};$
- Python使用了稱為(Mersenne twister)梅森旋轉算法，是比前面公式，更高級的技術。對於大多數需要隨機數的應用(例如遊戲)而言已經夠好了。
- 因偽隨機數生成器中的下一個數字是由前一個（或多個）數字確定的，因此此類生成器始終需要有第一個開始的地方，稱為其種子(seed)。
- 同樣的Seed會得到相同順序，所以為了每次取得Seed會不同的常見技巧是使用當前系統時間當Seed.
- Python的random模塊

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window
>>> import random
>>> random.random()
0.3015910533067516
>>> random.random()
0.8587774767255003
>>> random.randint(10,19)
11
>>> random.randint(10,19)
13
>>> random.choice([3,4,5])
3
>>> random.choice([3,4,5])
4
>>> |
```

| Syntax                       | Description                                                                                                                                                     |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| seed(hashable)               | Initializes the pseudo-random number generator based upon the hash value of the parameter                                                                       |
| random()                     | Returns a pseudo-random floating-point value in the interval [0.0, 1.0). <span style="border: 2px solid red; border-radius: 50%; padding: 2px;">interval</span> |
| randint(a,b)                 | Returns a pseudo-random integer in the closed interval [a, b].                                                                                                  |
| randrange(start, stop, step) | Returns a pseudo-random integer in the standard Python range indicated by the parameters.                                                                       |
| choice(seq)                  | Returns an element of the given sequence chosen pseudo-randomly.                                                                                                |
| shuffle(seq)                 | Reorders the elements of the given sequence pseudo-randomly.                                                                                                    |

# Lab#1-11a-array\_module\_test

The screenshot shows a Windows desktop environment with three windows open:

- File Explorer:** Shows files in the "Python38-32" folder. Two files are highlighted: "array2.bin" and "array.bin".
- Windows File Explorer:** Shows the contents of "array.bin". The data is displayed in hex format: 01 00 00 00 02 00 00 00 03 00 00 00 04 00 00 00.
- Python 3.8.1 Shell:** Displays the following session:

```
>>> dir(array)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    dir(array)
NameError: name 'array' is not defined
>>> import array
>>> dir(array)
['ArrayType', '__doc__', '__loader__', '__name__', '__package__', '__spec__', '__array_reconstructor__', 'array', 'typecodes']
>>> ar_int=array('i',[1,3,5,7])
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    ar_int=array('i',[1,3,5,7])
TypeError: 'module' object is not callable
>>> ar_int=array.array('i',[1,3,5,7])
>>> ar_int.append(8)
>>> ar_int
array('i' [1 3 5 7 8])
>>> f = open("array.bin", "wb")
>>> array("i", [1, 2, 3, 4]).tofile(f)
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    array("i", [1, 2, 3, 4]).tofile(f)
TypeError: 'module' object is not callable
>>> from array import *
>>> array("i", [1, 2, 3, 4]).tofile(f)
>>> f.close()
>>> f = open("array2.bin", "wb")
>>> ar_int.tofile(f)
>>> f.close()
```

Annotations with arrows and boxes explain the code:

- A yellow box with a red arrow points to the first `dir(array)` command: **要import，才可認識array** (You need to import it to recognize it).
- A yellow box with a red arrow points to the line where `array` is imported: **Import後，要建立陣列，要使用array.array()來建立新陣列** (After import, to create an array, use array.array()).
- A green box with a red arrow points to the line where `array` is imported: **要from import，就可以把原來的array.array()省略為array()** (With from import, you can omit array.array() and just use array()).

# Lab#1-11b-array\_(numpy\_module)\_test

- Python內定的array module，速度雖快，但，須相同類型資料，不是很方便使用！
- 更多的情況，會在需要array時，選擇numpy，只是在使用numpy前：  
「先用pip安裝好numpy」  
「才可以import numpy」
- 因為陣列的使用  
更方便！如右圖所示。
- ※numpy習慣的簡稱是np.

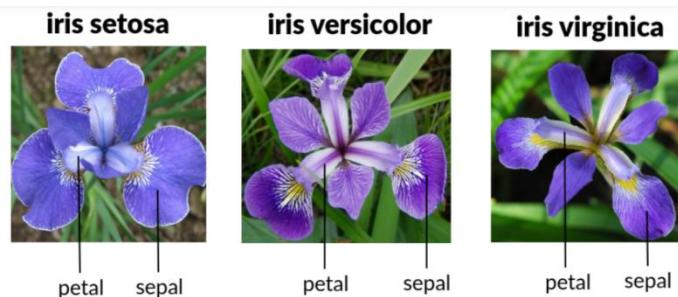
命令提示字元  
C:\Users\rr\AppData\Local\Programs\Python\Python38-32\Scripts> C:\Users\rr\AppData\Local\Programs\Python\Python38-32\Scripts> C:\Users\rr\AppData\Local\Programs\Python\Python38-32\Scripts> pip3.8.exe install numpy  
19kB/s  
Installing collected packages: numpy  
 WARNING: The script f2py.exe is installed in 'c:\users\rr\appdata\local\programs\python\python38-32\scripts' which is not on PATH.  
 Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.  
Successfully installed numpy-1.18.1  
WARNING: You are using pip version 19.2.3, however version 20.0.2 is available.  
You should consider upgrading via the 'python -m pip install --upgrade pip' command.  
C:\Users\rr\AppData\Local\Programs\Python\Python38-32\Scripts>

Python 3.8.1 Shell  
File Edit Shell Debug Options Window Help  
>>> import array  
>>> ar=array.array("i",[1,2,3])  
>>> ar2=array.array("i",[1,2,3.4])  
Traceback (most recent call last):  
 File "<pyshell#29>", line 1, in <module>  
 ar2=array.array("i",[1.2.3.4])  
TypeError: integer argument expected, got float  
>>> import numpy  
Traceback (most recent call last):  
 File "<pyshell#30>", line 1, in <module>

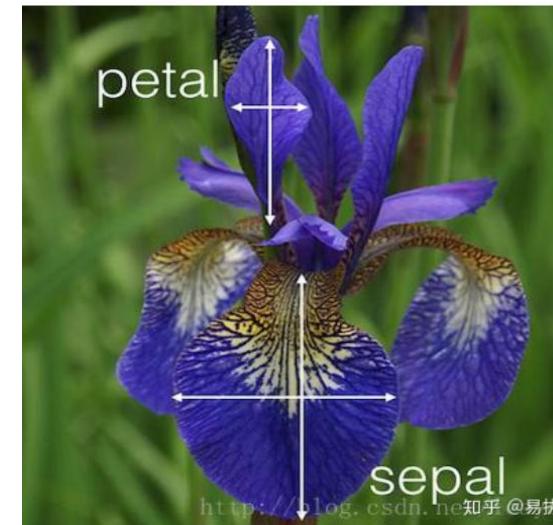
- 在開始正式「**1-12: 數據快速Explore與可視化(kaggle)**」之前~~~
- 最新的AI工具非常多，你有想過或有用過嗎？

# 1-12: 數據快速Explore與可視化(kaggle)

- 初學c/c++語言，會先學一個"hello world"的程式。
- 初學「數據」，幾乎就是"Hello world"等級的資料庫，就是 **iris** (**iris dataset**,鳶尾花資料集).
  - Iris裡有150筆資料，共有三種鳶尾花，每朵花都有四種屬性。
    - 花瓣(petal)之長與寬
    - 花萼(sepal)之長與寬。



這個 Iris 數據集是任何數據科學學生都在研究的第一個數據集。



- 如果只是要一般的可視化，只要幾分鐘就可以達到專業標準了，請看：

# 當今最熱門的數據專家聚集的網站

■ 請先到kaggle的網站，搜尋關鍵字：

The screenshot shows two side-by-side browser windows. Both windows have the URL `kaggle.com/search?q=Python+Iris+Data+Visualizations` in the address bar. The left window shows the search results page with various filters applied: Date (Last 90 days), Dataset Size (small), Dataset File Types (csv), and Kernel Language (Python). The right window shows the detailed search results for 'Python Iris Data Visualizations'. A red circle highlights the search term in the URL and the title of the top result. Another red circle highlights the 'Python' filter in the left window's sidebar. Red dots connect the highlighted elements between the two windows. A red box in the bottom right corner contains the text: '為了方便搜尋，請指定kernel為python'.

為了方便搜尋，請指定kernel為python

# kaggle官方下載iris可視化版本之1: 可見到這個 notebook的原始程式碼

The screenshot shows a Kaggle notebook interface. The title is 'Python Iris Data Visualizations'. The notebook content includes a descriptive text and three code snippets for importing libraries:

```
In [1]:  
# First, we'll import pandas, a data processing and CSV file I/O library  
import pandas as pd  
  
# Next, we import matplotlib, which is a go-to Python plotting library  
import matplotlib.pyplot as plt  
  
# Lastly, we'll import seaborn. Seaborn is a powerful Python graphing  
# library based on matplotlib that attempts to optimize matplotlib's ou  
tput  
import seaborn as sns  
# This will switch from matplotlib's default graph style to seaborn's de  
fault graph style.  
# I chose to use seaborn's default set which displays graph lines inside
```

A yellow callout box on the left contains the text: '將會使用這3個在python中很著名的程式庫。' (Will use these 3 famous Python libraries).

Annotations highlight the notebook's purpose, its fork status, and the imported libraries.

實驗方法：(1)可下載notebook(附加名是.ipynb)  
方法(2)也可免費註冊帳號，在kaggle上直接實驗！

TKU 淡江大學-Welcome to Tam | Python Iris Data Visualizations | Python Iris Data Visualizations | +

kaggle.com/zachgold/python-iris-data-visualizations

Sign In Register

Search

Python Iris Data Visualizations

Python notebook using data from Iris Species · 1,503 views · 2y ago

0 Copy and Edit 15 ...

Download code

Copy API command

Upgrade to Google Cloud AI Notebooks

This notebook demos Python data visualizations on the Iris dataset

This is my fork of Ben Hamner's well-regarded Iris dataset Python notebook and is my first Kernel on Kaggle. This fork is a learning experience for me since I'm new to data science!

I'm using three go-to Python libraries for this tutorial which are described in more detail in

copied from Python Data Visualizations (+99 -47)

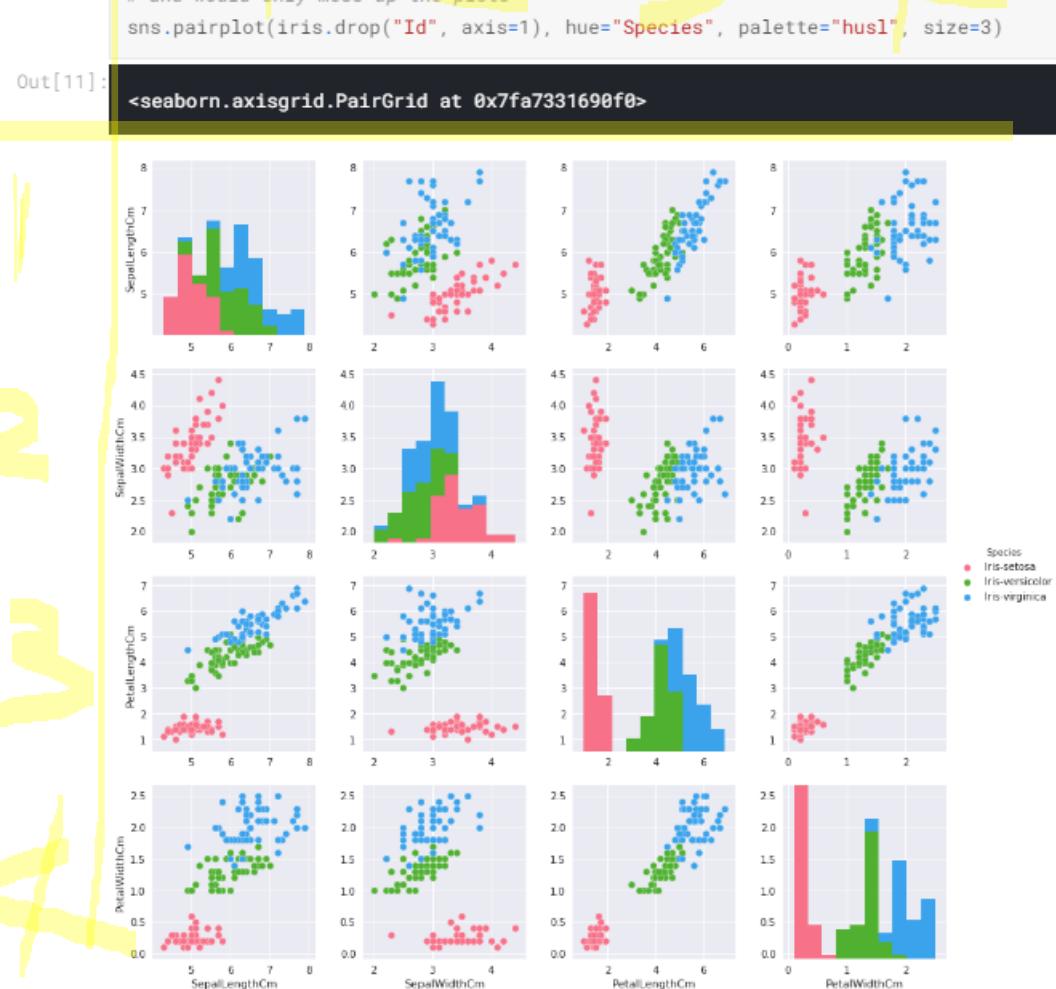
Notebook

https://www.kaggle.com/.../scriptcontent/3146274/download

python-iris-dat....ipynb

# 用seaborn(官方建議會簡稱為sns) 所呈現4個屬性的散點圖矩陣(pairplot)

- 「散點圖矩陣(pairplot)」主要展現的是變數兩兩之間的關係（線性或非線性，有無較為明顯的相關關係）



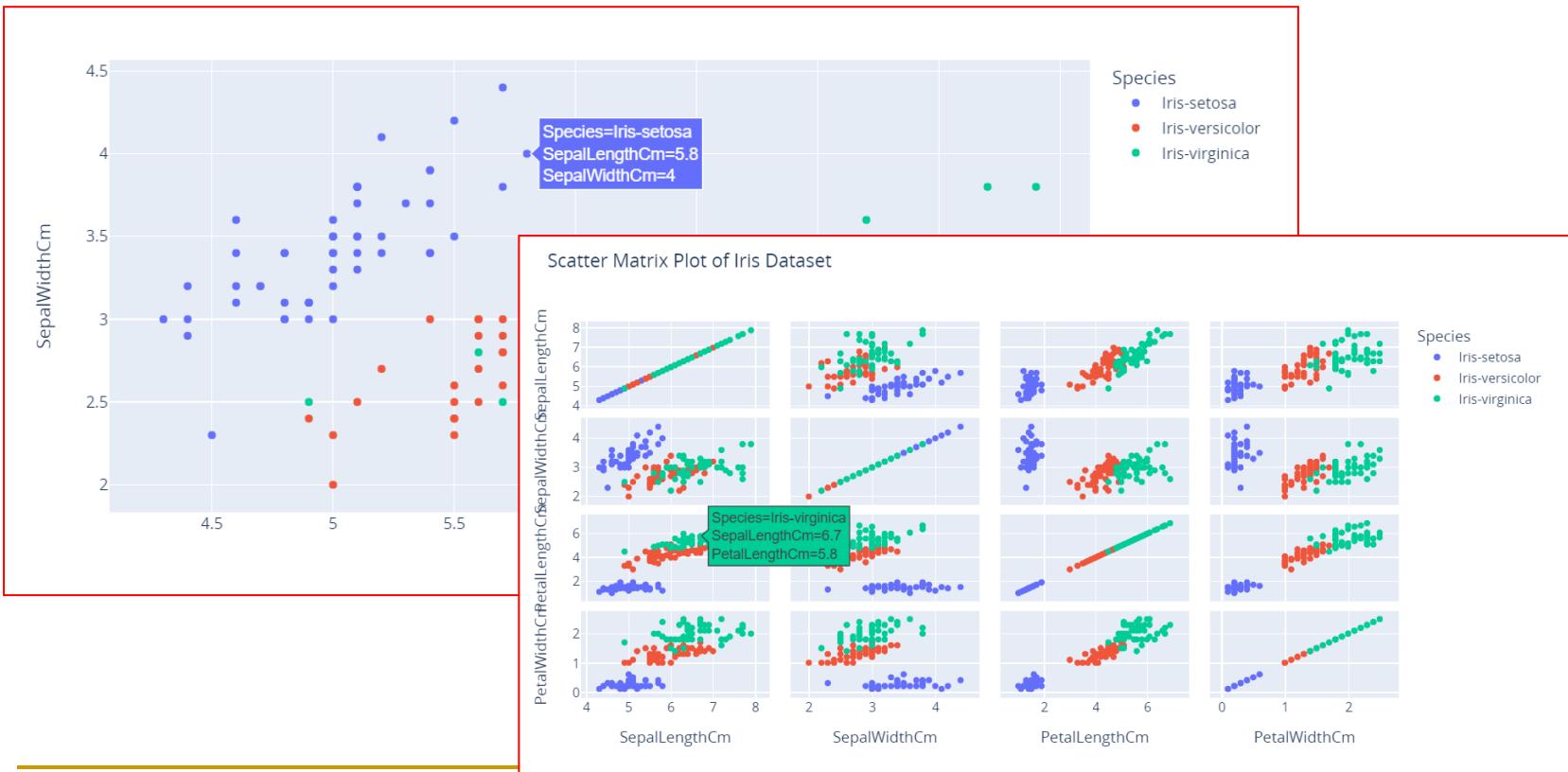
- ※註：若要再仔細計算出相關係數，就可變為Heat圖

○

# kaggle官方下載iris可視化版本之2

## ■ 包含seaborn+plotly的可視化+ML機器學習

- <https://www.kaggle.com/code/beltagymohamedsaleh/exploring-and-modeling-the-iris-dataset/notebook>
- 透過plotly的模組，增加使用者的互動的功能：



# 補充：iris dataset 的最快速認識方法：

- 因為iris是知名的入門資料庫
- 所以，直接在kaggle，只要下列三行程式，就可看出iris的官方版的介紹。

如果安裝Anaconda，  
則開新的python，  
寫入這三行，也可得到同樣的訊息哦！

← notebook1527934d0a Draft saved  
File Edit View Run Add-ons Help  
+ ▶ Run All

[4]: from sklearn import datasets  
iris = datasets.load\_iris()

▶ #接著我們嘗試將這個機器學習資料之描述檔顯示出來  
print(iris['DESCR'])

```
.. _iris_dataset:  
  
Iris plants dataset  
-----  
  
**Data Set Characteristics:**  
  
:Number of Instances: 150 (50 in each of three classes)  
:Number of Attributes: 4 numeric, predictive attributes and the class  
:Attribute Information:  
    - sepal length in cm  
    - sepal width in cm  
    - petal length in cm  
    - petal width in cm  
    - class:  
        - Iris-Setosa  
        - Iris-Versicolour  
        - Iris-Virginica  
  
:Summary Statistics:  
  
===== Min Max Mean SD Class Correlation  
===== ===== ===== ===== =====  
sepal length: 4.3 7.9 5.84 0.83 0.7826  
sepal width: 2.0 4.4 3.05 0.43 -0.4194  
petal length: 1.0 6.9 3.76 1.76 0.9490 (high!)  
petal width: 0.1 2.5 1.20 0.76 0.9565 (high!)  
===== ===== ===== ===== =====  
  
:Missing Attribute Values: None  
:Class Distribution: 33.3% for each of 3 classes.  
:Creator: R.A. Fisher  
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)  
:Date: July, 1988  
  
The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken  
from Fisher's paper. Note that it's the 4th column (Petal Width) that is missing, not the last one (Sepal Width).
```

# Iris資料集\_至少有3種容易取得的來源

## iris資料來源1：從 sklearn的內建資料庫下載

```
1 from sklearn import datasets  
2 iris = datasets.load_iris()  
3 print(iris['DESCR'])
```

資料來源3:到官方網站(UCI大學)

The screenshot shows the UCI Machine Learning Repository homepage. A red box highlights the 'Iris Data Set' section. Below it, a smaller red box highlights the 'Abstract' section which reads: 'Famous database; from Fisher, 1936'.

## iris資料來源2：從seaborn內建資料庫下載

```
1 # 在引入seaborn後，只要四行程式碼，就有可視化了！  
2 # 以iris的散點圖矩陣 (Pairs Plots) 為例：  
3 import seaborn as sns  
4 df = sns.load_dataset("iris")  
5 df
```

|   | sepal_length | sepal_width | petal_length | petal_width | species |
|---|--------------|-------------|--------------|-------------|---------|
| 0 | 5.1          | 3.5         | 1.4          | 0.2         | setosa  |
| 1 | 4.9          | 3.0         | 1.4          | 0.2         | setosa  |
| 2 | 4.7          | 3.2         | 1.3          | 0.2         | setosa  |

```
1 # QA 可查看第35個樣本的內容是 [[4.9 3.1 1.5 0.1 'Iris-setosa'  
2 import pandas as pd  
3 df = pd.read_csv('https://archive.ics.uci.edu/ml/'  
4 'machine-learning-databases/iris/iris|.data', header=None)  
5 #df.tail()  
6 print("統計表(最大、最小、平均、樣本標準差")  
7 import numpy as np  
8 df[:4]
```

統計表(最大、最小、平均、樣本標準差

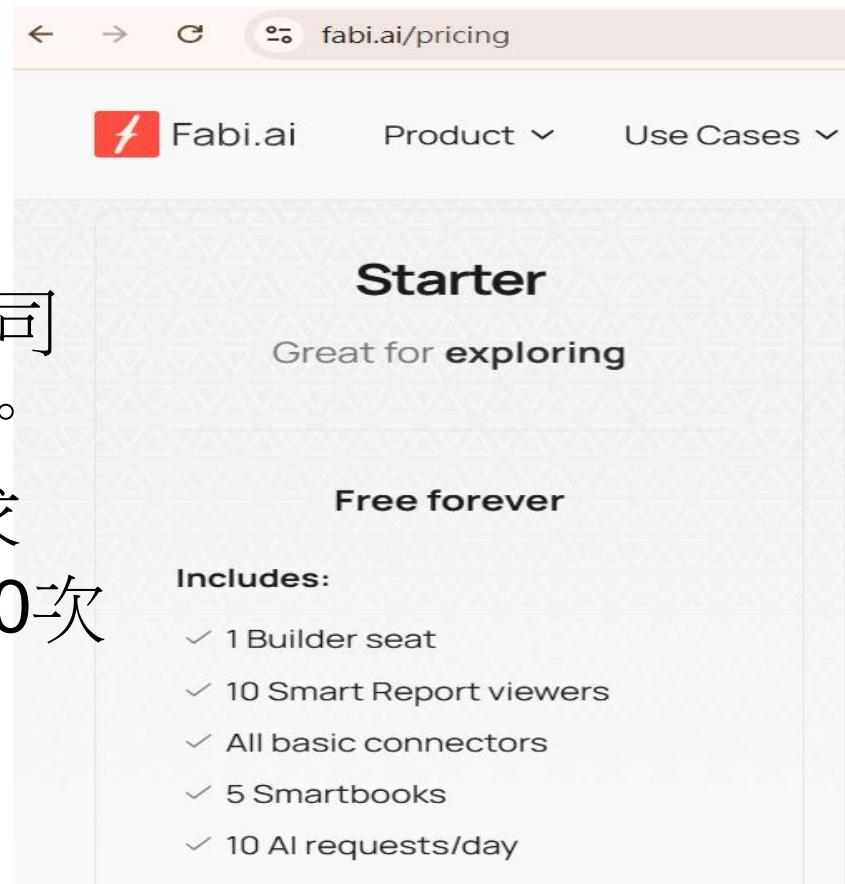
|   | 0   | 1   | 2   | 3   | 4           |
|---|-----|-----|-----|-----|-------------|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |

# ●在開始正式「 1-12: 數據快速Explore與可視化(kaggle)」之後

■ 最新的AI工具非常多，你有想過或有用過嗎？

■ 例如**fabi.ai**

- 是**2023年成立的公司**，google帳號即可。
- 終生免費版: AI請求(AI requests)每天10次

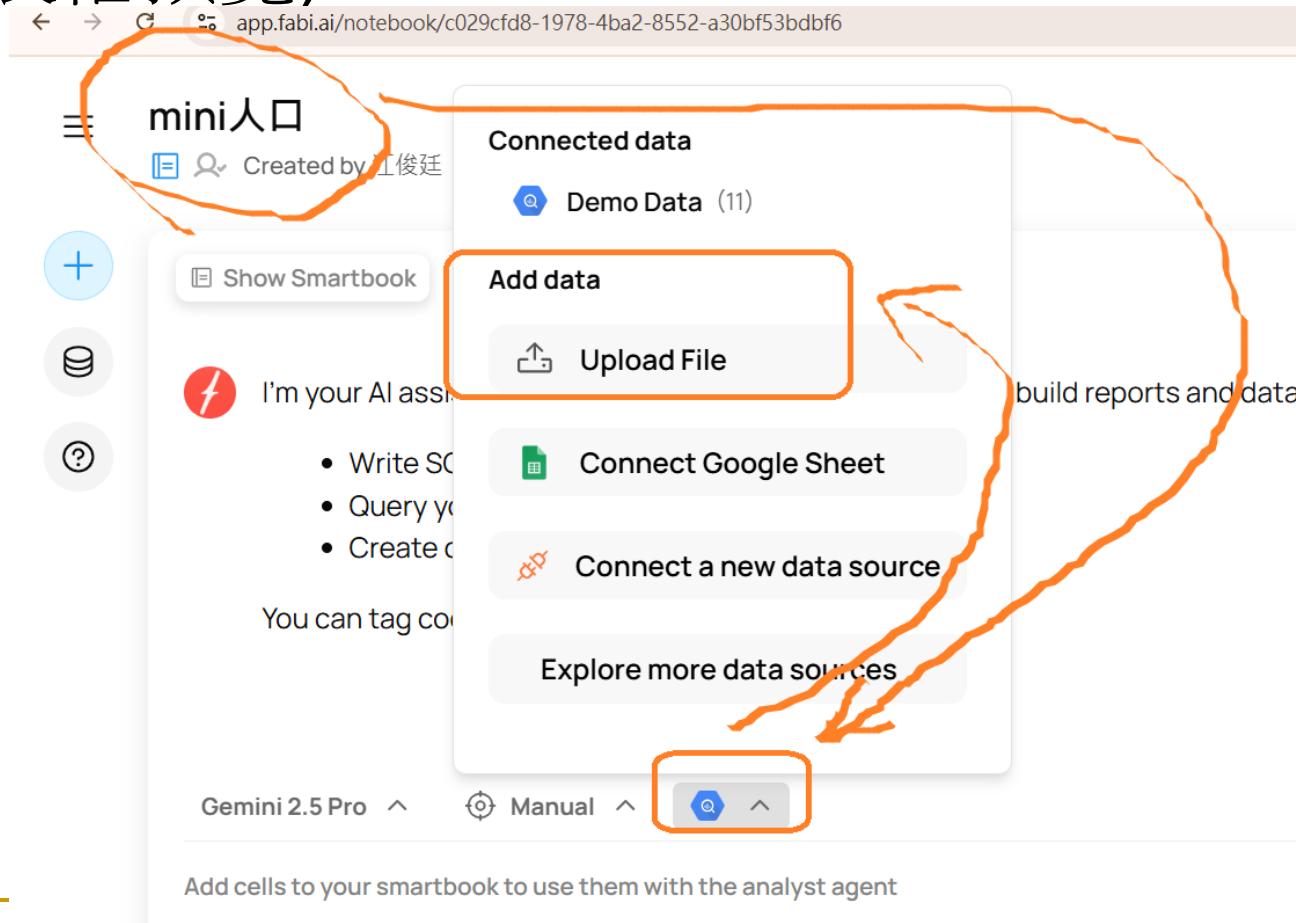


# Fabi.AI 是超好用的 AI 助手

- 對於數據，可挖掘洞察，建立報告，還可以：
  - 編寫 SQL 和 Python
  - 查詢資料庫和上傳的文件
  - 創建圖表和視覺化效果
- 「**智慧本(Smartbook)**」是Fabi.ai 中的一個「資料 + 視覺化 + AI 分析」的集合體
  - 是AI輔助資料分析工具，被設計為「下一代 Jupyter Notebook」  
(※可惜無法直接下載為jupyter notebook)
- 操作流程：上傳 CSV → 輸入 prompt → AI 生成圖表 → 使用者檢查 → 點「Accept / Confirm」就寫入智慧本的一個格子。

# 在fabi.ai 建立「mini人口」專案

- 找到上傳，(上傳完成後，你會看到python程式寫的表格預覽)



## ● 【check4】「mini人口」專案的Ai操作示範：

### ■ ※注意：Title要有自己的名字座號

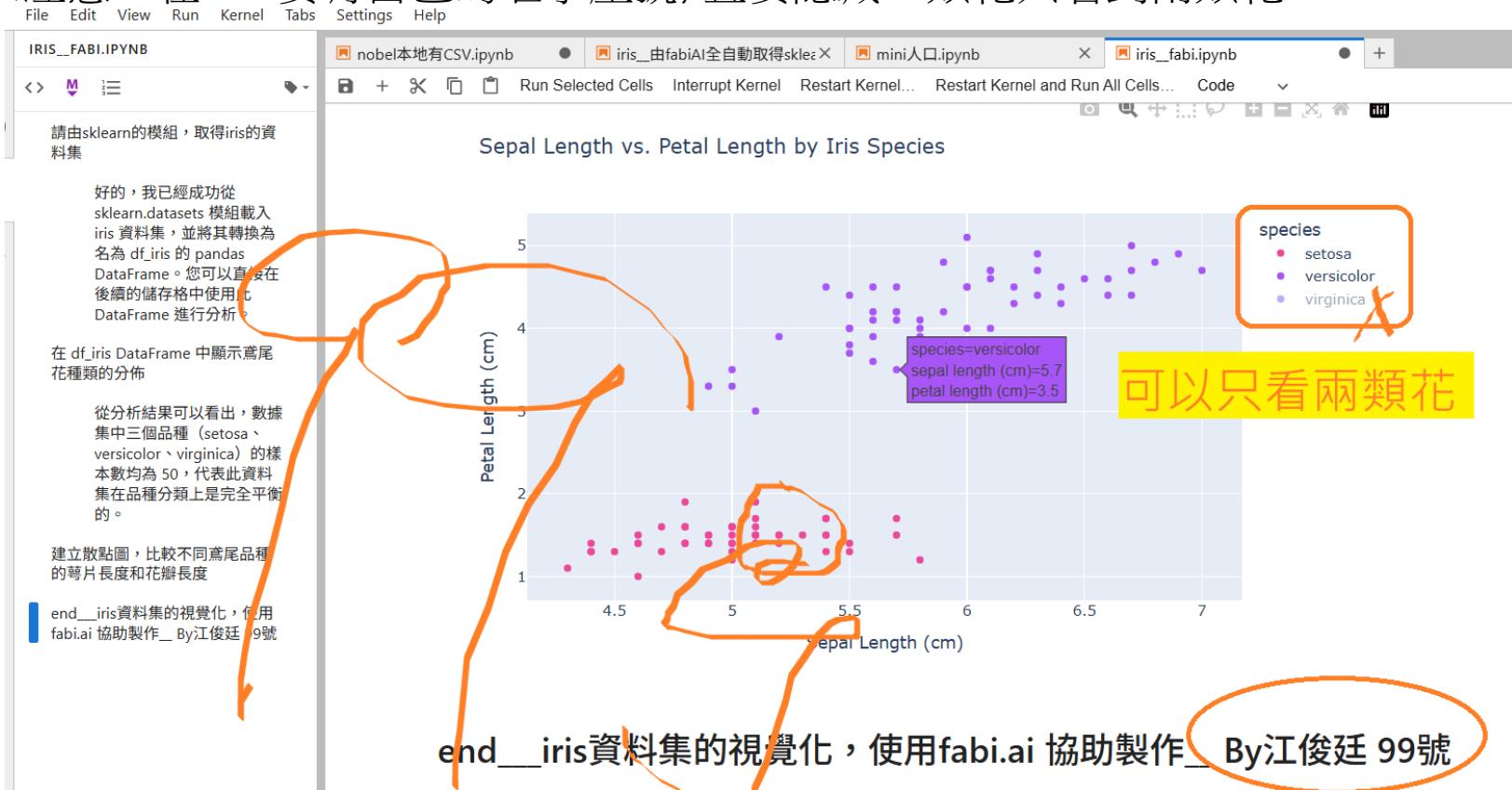
顯示一個人口比較圖表，其中以百分比取代絕對數字

```
[5]: import plotly.express as px  
  
# Make a copy to avoid SettingWithCopyWarning  
df = population_sample.copy()  
  
# Calculate the total population  
total_population = df['Population'].sum()  
  
# Calculate the percentage for each country  
df['Percentage'] = (df['Population'] / total_population) * 100  
  
# Create a bar chart using the population_sample dataframe with percentages  
fig = px.bar(df, x='Country', y='Percentage', title='各國人口百分比_作者江俊廷99號', text=df['Percentage'].apply(lambda x: f'{x:.2f}%'))  
  
# Display the chart  
fig.show()
```



# ● 【check5】「iris」專案的Ai操作示範

- Prompt: 請由sklearn的模組，取得iris的資料集
- Prompt: 在DataFrame中顯示鳶尾花種類的分佈
- Prompt: 建立散點圖，比較不同鳶尾品種的萼片長度和花瓣長度
- ※注意：在end要有自己的名字座號，且要隱藏一類花只看到兩類花。

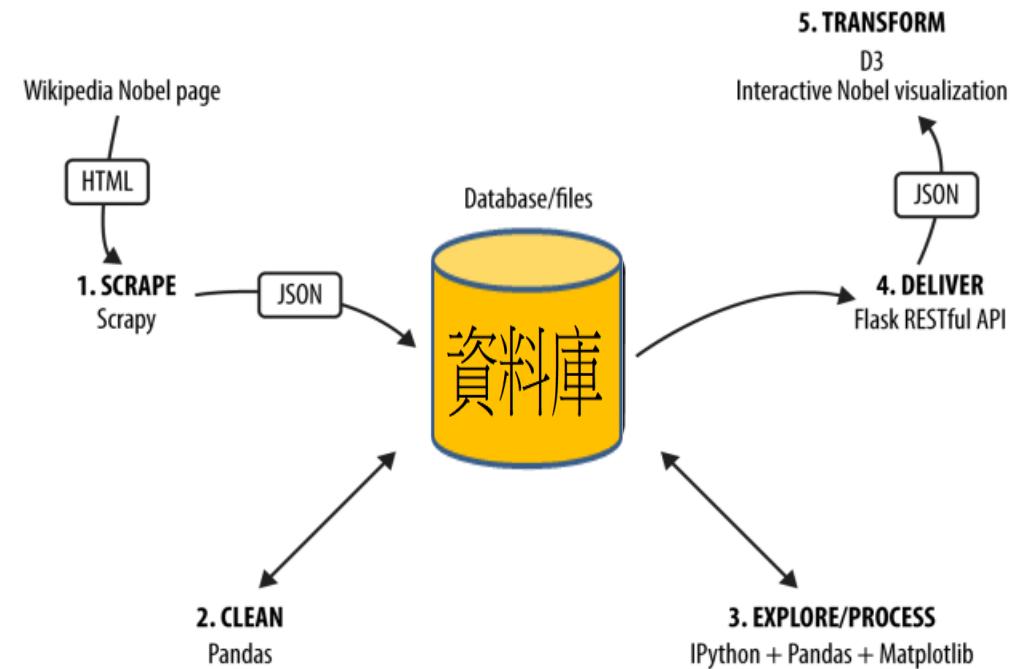


# Exercise 1-12

- Ex1-12Add1:因為iris是知名的入門資料庫，是否有簡單方法就可印出iris的官方版介紹？如果有，請寫出這段程式。
- Ex1-12Add2: Seaborn是基于matplotlib的Python可視化庫。提供高級界面來繪制有吸引力的統計圖形。他的簡稱比較奇怪！在官方網站上建議的縮寫是什麼？以iris為例，如何快速畫出散點圖矩陣（pairplot）？在圖上，可見到任兩屬性的關係，至少可看出三種關係是「正相關」「無關」還有是什麼關係？
- Ex1-12Add3:簡述三種容易取得的Iris資料集的來源，與相對應程式。
- 作業題：**Ex1.12Add1 ~ Ex1.12Add3**
- 補充：

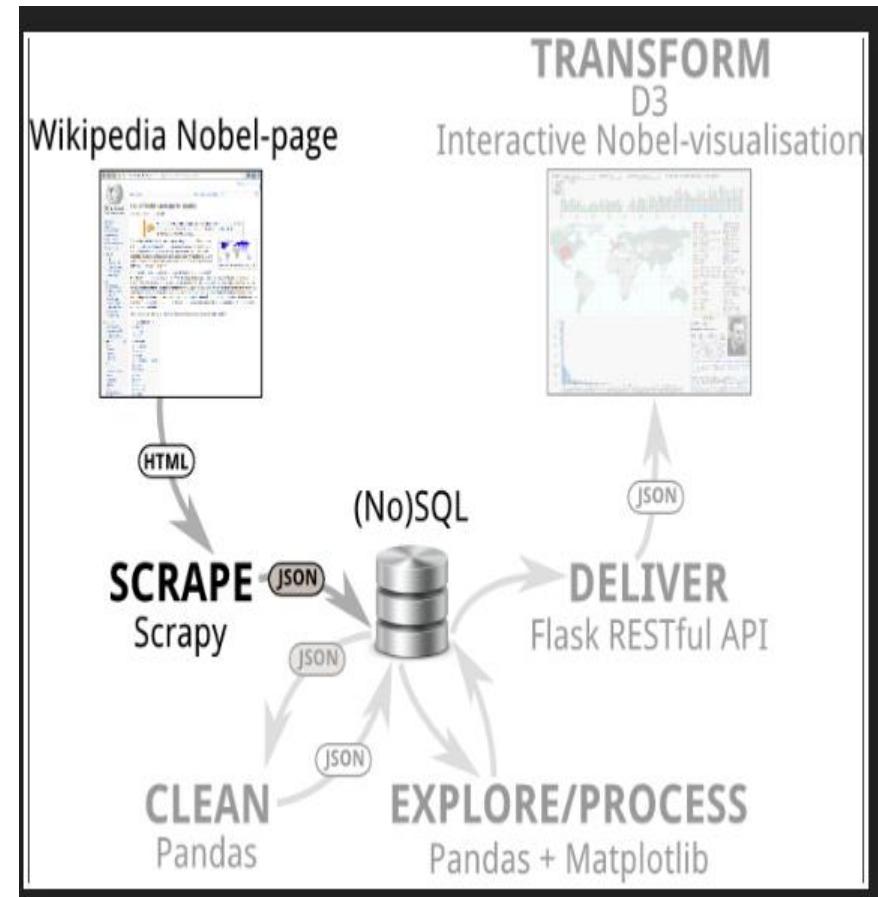
# 1-13: 資料視覺化的工具鏈 (The dataViz Tool chain)

- 這個數據可視化工具鏈，使用Python從Web上抓取數據(可用Scrapy)，進行數據處理(用Pandas, Seaborn, Matplotlib等)，然後使用Python將其傳送至Web瀏覽器。網站服務器(建議使用RESTful數據API)，最後選擇適當方式在網頁上呈現互動(建議，可採用D3)



# 1、用(Scraping)來爬取數據的結果

- Python已內建有不少好用的爬蟲工具。
- Scraping是具工業等級的爬蟲程式！



# 通過可識別的目標(target)，來爬取數據

The collage illustrates the scraping process:

- Top Left:** A screenshot of a browser's developer tools showing the element tree. A specific element is highlighted with the text "<p></p> stop-point".
- Top Right:** A screenshot of a Wikipedia page for "François Truffaut" with a large arrow pointing from the browser window to the code editor below.
- Bottom Left:** A screenshot of the browser developer tools element tree again, showing the same "<p></p> stop-point" element.
- Bottom Center:** A code editor window titled "nobel\_winner\_bio\_spider.py" containing Scrapy spider code. The code defines a spider "NwinnerSpider" that scrapes the URL "http://en.wikipedia.org/wiki/List\_of\_Nobel\_laureates\_by\_country". It includes a parse method to extract data from the page.
- Bottom Right:** A code editor window titled "pipeline.py" containing a Python pipeline class. It defines a "process\_item" method to handle scraped items and a "process\_completed" method to store images in a 'bio image' field.

## 用(Scraping)來爬取數據結果：

- 有不錯的JSON物件數據。
- 但是數據可能還很髒！

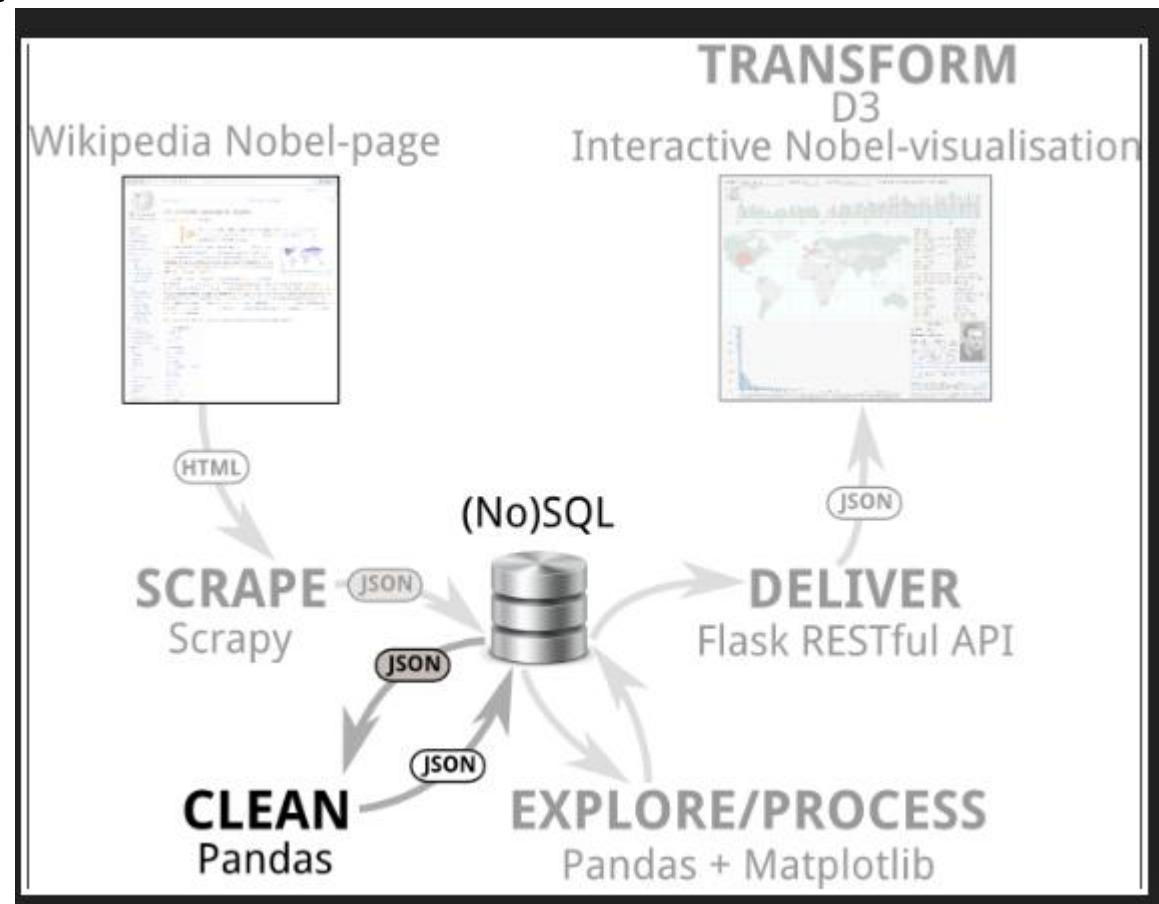
## 所以，接下來，要換

- 熊貓(Pandas)上場囉！

■ 需要識別並消除異常，例如重複項，缺少字段，格式化錯誤等...。

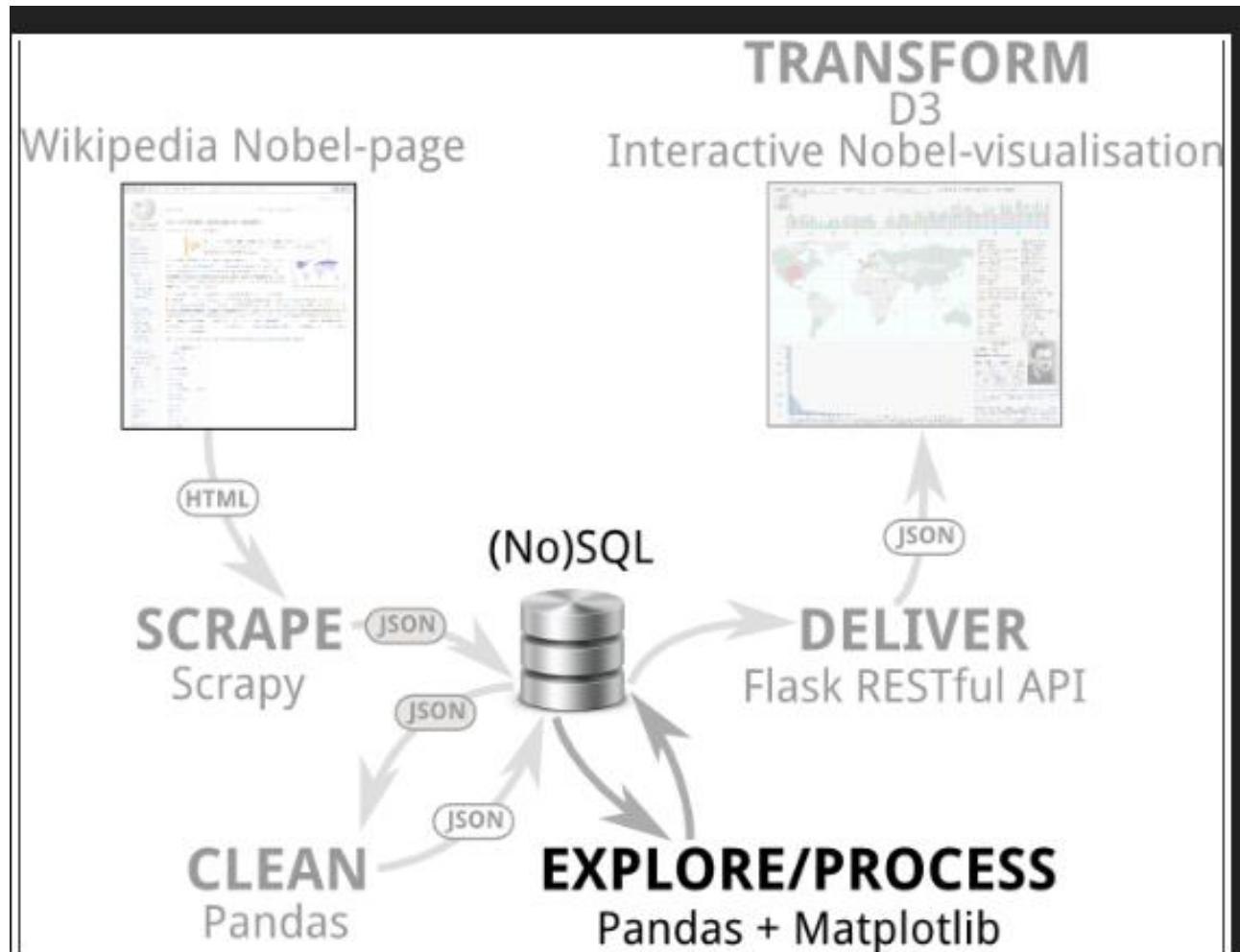
## 2、用Pandas清洗資料(Clean data)

- Pandas裡的重要元件**DataFrame**，就像是可程式控制的試算表。
  - 可找出重覆的資料、遺失的欄位、雜亂的日期...



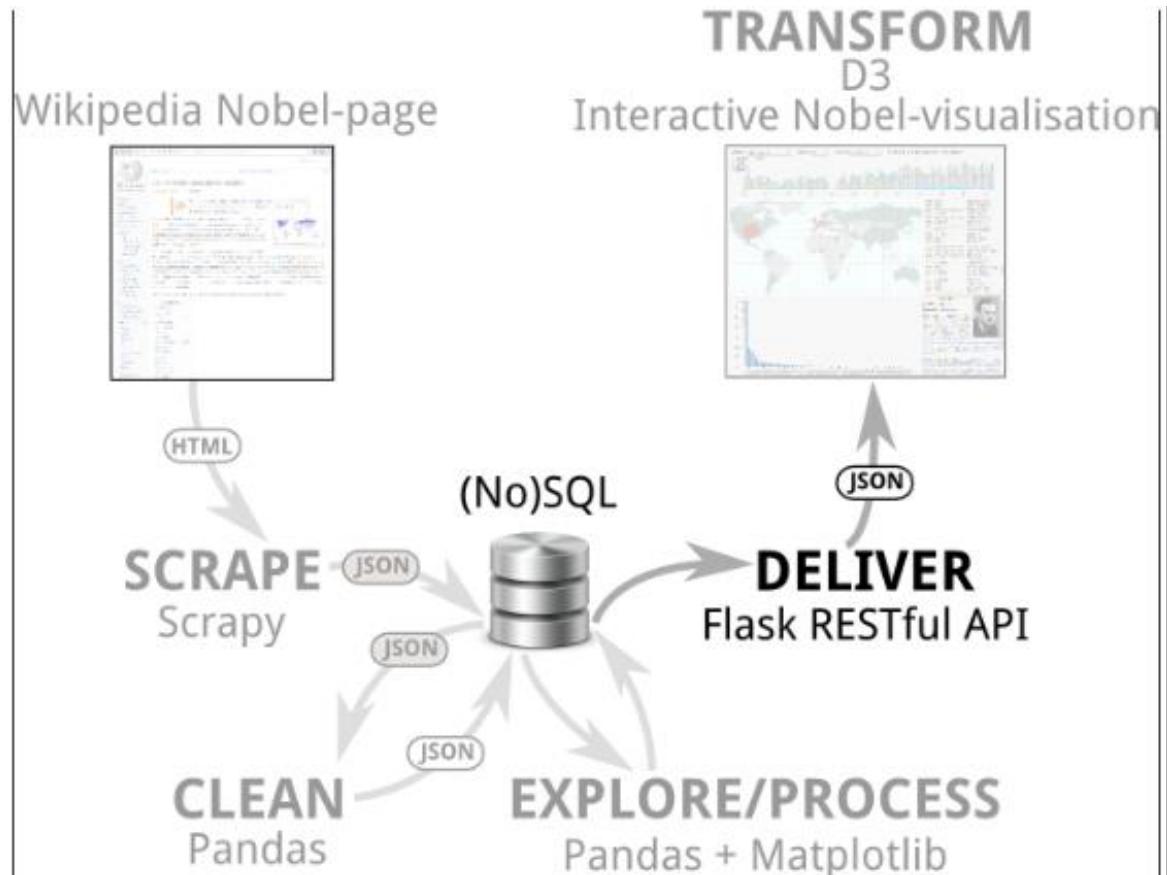
### 3. 用Ipython+Pandas+Matplotlib探索、處理數據

- 在製作視覺化之前，應該要先：
  - 懂得資料內容
  - 知道隱藏在資料裡的共通形式。
  - 傾向、離群值
  - ...
- 因為這些會影響試圖要說的故事。
  - 以iris 資料集為例，裡面的資料具有什麼特徵？可說出怎樣的故事？



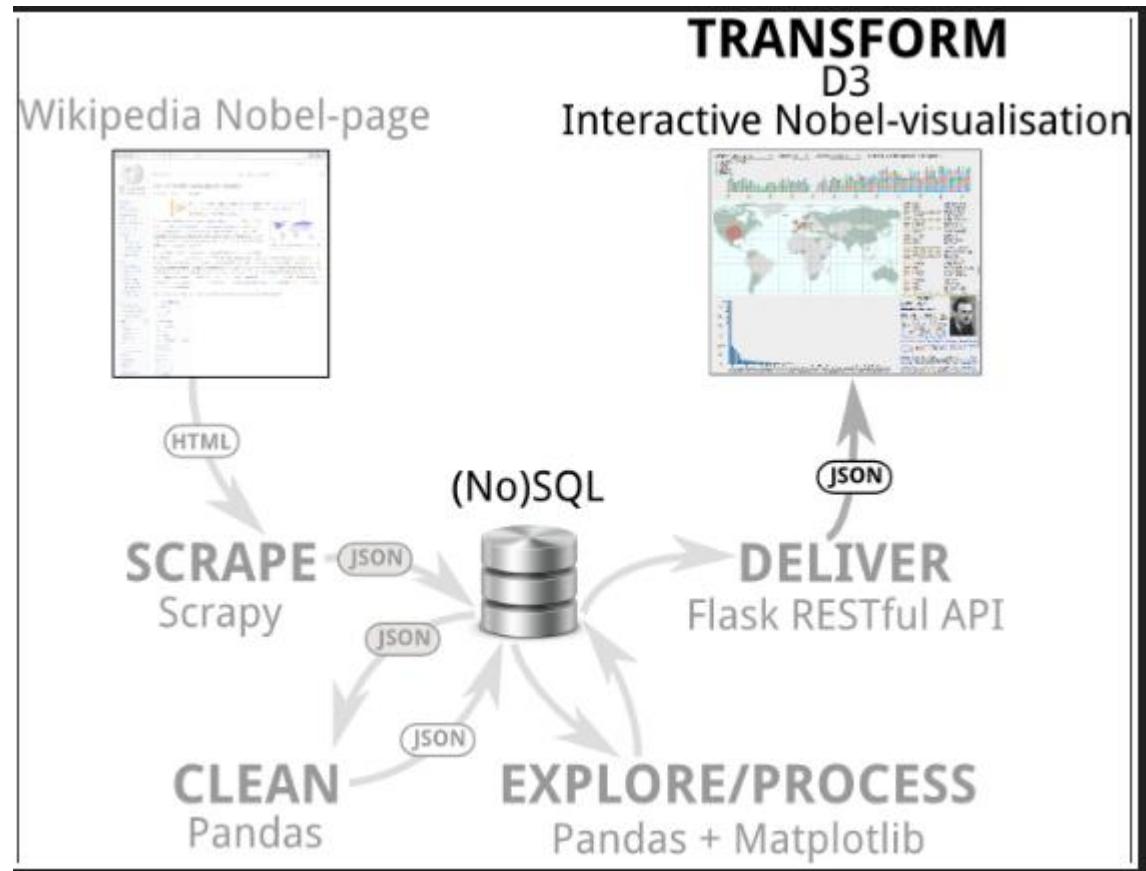
## 4. 用FLASK提供數據(Delivering DATA )

- 預計採用Flask的外掛程式庫Eve，來建立存取MongoDB資料庫的RESTful API.
- 就可提供資料給D3囉！

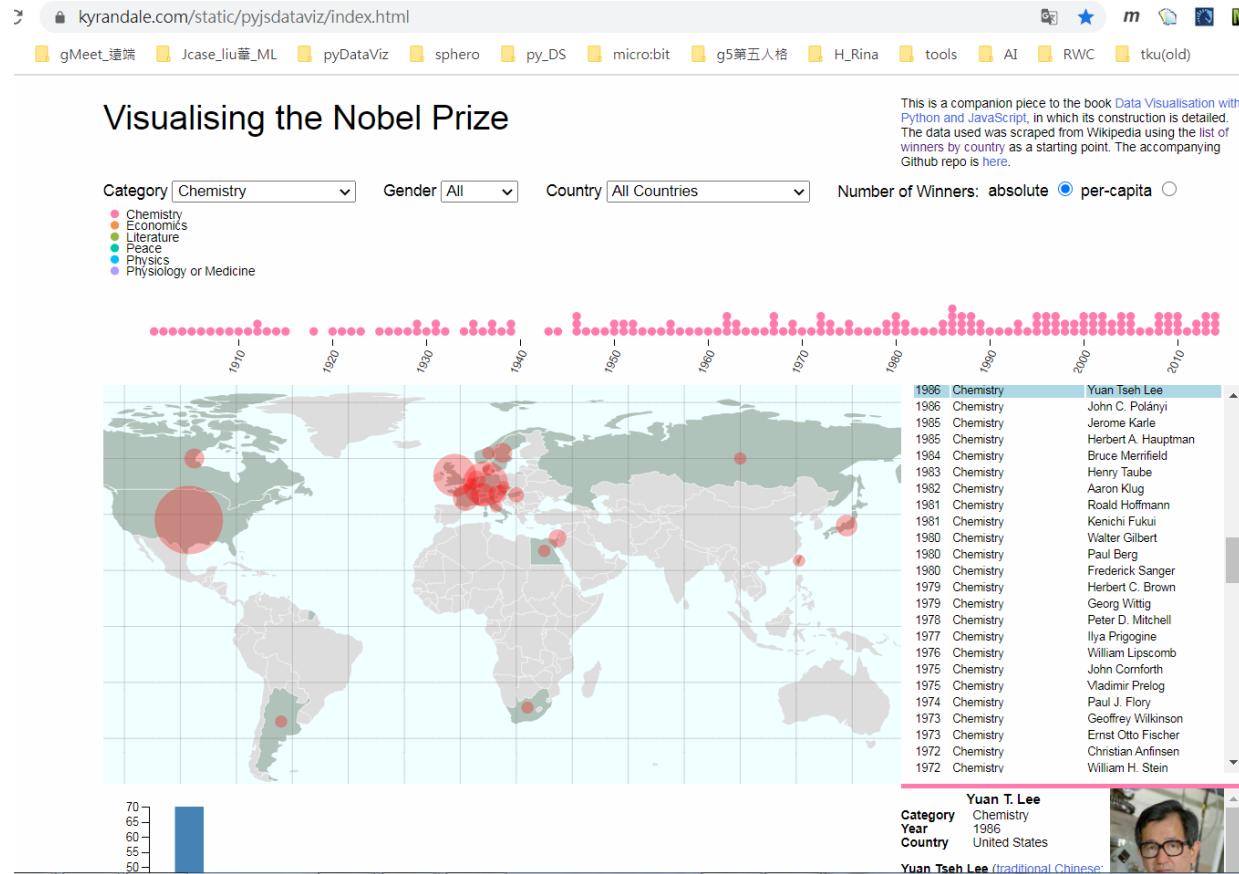


# 5、使用D3轉換數據(Transform data)

- 透過D3程式庫，轉成可互動操作視覺化網站介面。



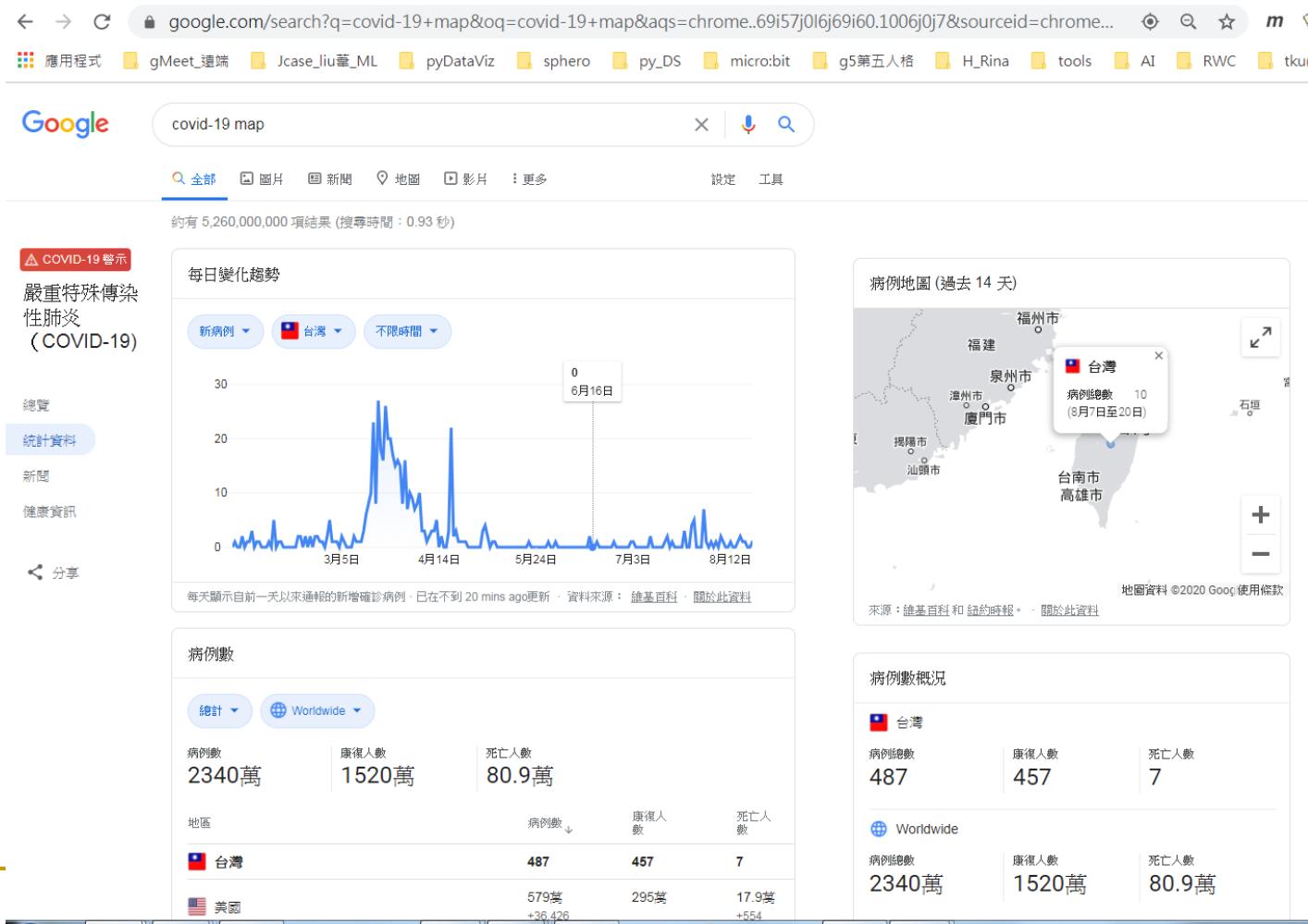
# 完成畫面的展示：



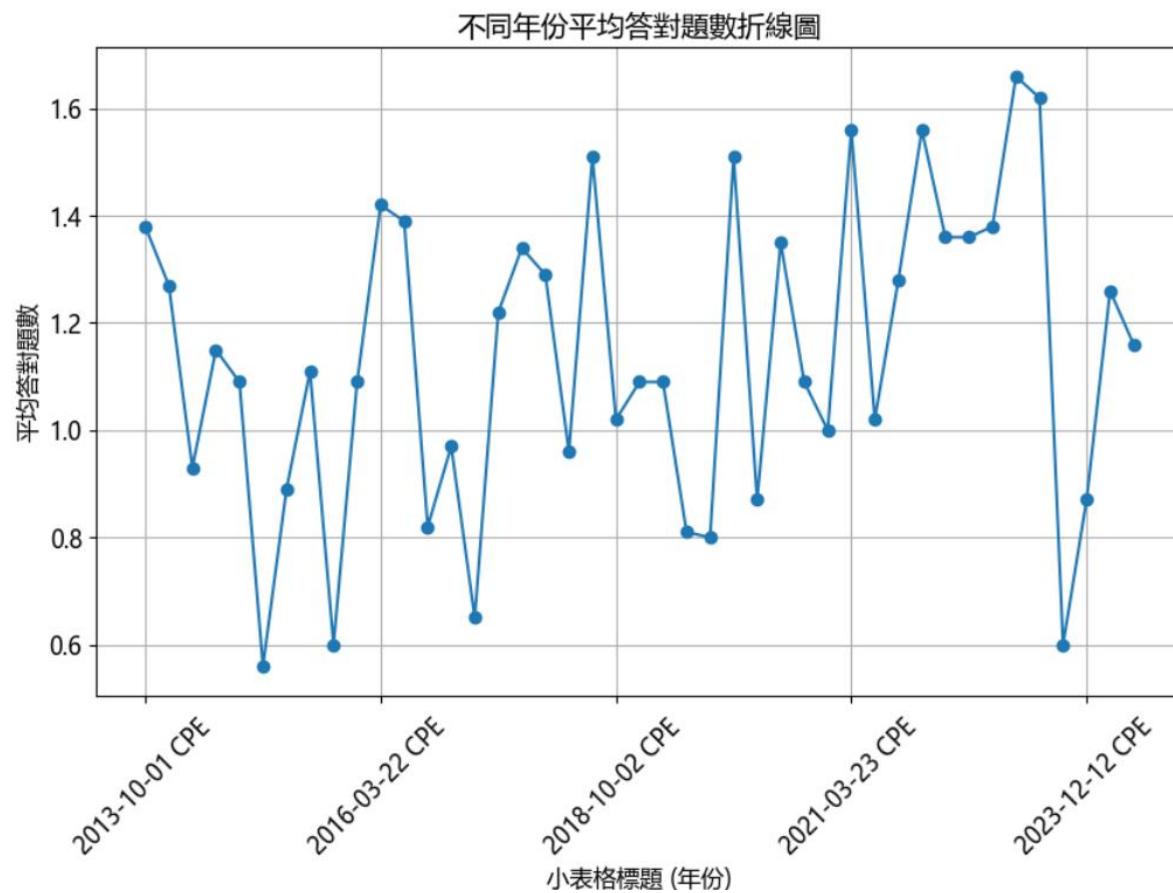
- 作者推薦的網址：<https://www.kyrdale.com/>
- <https://www.kyrdale.com/visualizing-the-nobel-prize-winners/>

# 範例1:covid-19 map在google直接搜尋簡易版

- <https://www.google.com/search?q=covid-19+map&oq=covid-19+map&aqs=chrome..69i57j0l6j69i60.1006j0j7&sourceid=chrome&ie=UTF-8>



## 範例2:台灣cpe近10年平均答對題目折線圖



# 著名的可視化：拿破崙1812年俄國戰役的數據

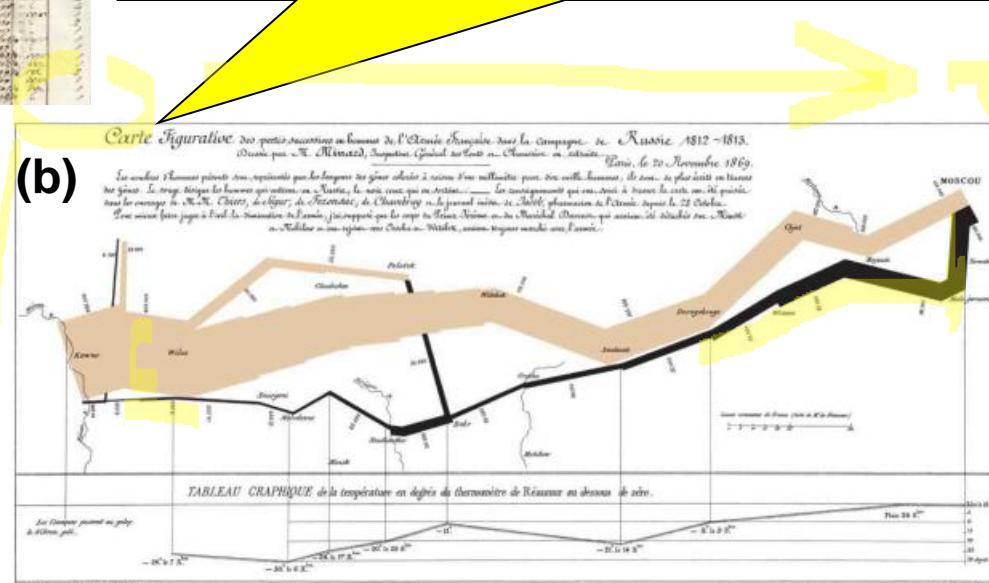
(a) 早期的電子表格的資料。

(b) 查爾斯·約瑟夫·米納德（法語Charles Joseph Minard）對這個事件可視化。

a.



(b)



# Exercise 1-13

- Ex1-13Add1: 本課程預定要介紹的「資料視覺化的工具鏈(The dataViz Tool chain)」有那5個步驟？請簡單說明。
- Ex1-13Add2: 關於 RESTful web API，(a) REST(Representational state transfer)是什麼，簡單說明。(b)請問「REST」與「RESTful」的ful，代表什麼？
- 作業題：**Ex1.13Add1 Ex1.13Add2**
- 補充：

# Exercise 1 (章末)

- **Ex1-1(章末):** 用python完成爬蟲工作有很多方法，(1)如果是一般爬蟲，已經有很多相關工具，請舉出至少兩種。(2)如果使用爬蟲框架(Crawler framework)來解決較複雜的爬蟲，請舉出工業等級的框架一種。
- **Ex1-2(章末):** 基本上，要開發python是可以不必使用IDE(整合開發環境)的，但是如果冇合適的IDE，是可以事半功倍的。請舉出3個好用的IDE。
- **Ex1-3(章末) :** 在開發javascript，想要使用外掛程式庫，可以下載到本地。也有另一種技術，可以免下載就直接使用？請問是什麼技術。
- **Ex1-4(章末) :** 用python開發網站的框架(Web Framework)有很多，請簡單畫出兩種並簡單說明。
  
- 作業題：**Ex1-1(章末), Ex1-2(章末), Ex1-3(章末), Ex1-4(章末)**
- 補充：



- Thank for your attention.

# 補充：Lab#用pycharm的Debug仔細觀察python的變化

- 綠色字與框是直接執行。
- 橘色字與框是**Debug**相關的功能，可直接觀察變數目前的值，也可觀察**Debug**的**console**視窗。

