# Python 的 50+ 練習：資料科學學習手冊

> *簡介*

數據交點 | 郭耀仁 yaojenkuo@datainpoint.com

# 給同學的課前心理建設

- 雖然我們可能從很多資訊來源聽到 Python 簡單易學，但是對於初學者而言，Python 程式設計與資料科學是有難度的。
- 同學在修課過程可能會遭遇挫折，有不懂的地方可以重複觀看課程影片、練習題卡關不知從何下手時可以先看練習題詳解，善加使用 Hahow 的「問題討論」發問，我會盡快回覆。
- 覺得累了的時候，休息個幾天、轉換心情再回來，學習效率會更好。
- 我會陪伴同學克服初學 Python 程式設計與資料科學的難關！（握）

# 這個章節會登場的保留字、函數與模組

- `import` 保留字。
- `print()` 函數。
- `this` 模組。

# 程式設計與資料科學

# 「Python 的 50+ 練習」課程的兩個核心

1. 程式設計。
2. 資料科學。

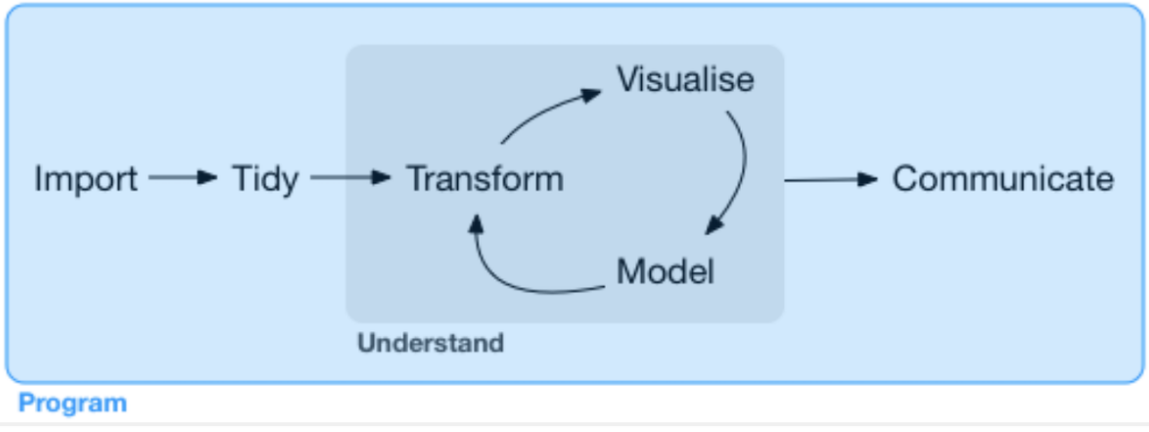
# 什麼是程式設計

以程式語言來指定電腦來解決特定問題的過程，常見的問題包含數值計算、文字搜尋取代、圖像或音訊處理等，程式設計過程包含分析、設計、寫作、測試與除錯等軟體開發的重要步驟。

# 什麼是資料科學

資料科學是一門將資料（Data）提煉為資訊（Information）的學科，提煉過程中可能包含資料載入、資料操作、探索性分析以及監督式學習等。

# 現代資料科學：以程式設計做資料科學的應用




來源：R for Data Science

# 有很多程式語言可以做到前述六項的資料科學應用，為什麼選 Python

- R
- Matlab
- Julia
- SAS
- Scala
- ...etc.

# 選擇最多人使用、推薦與使用人數上升的程式語言

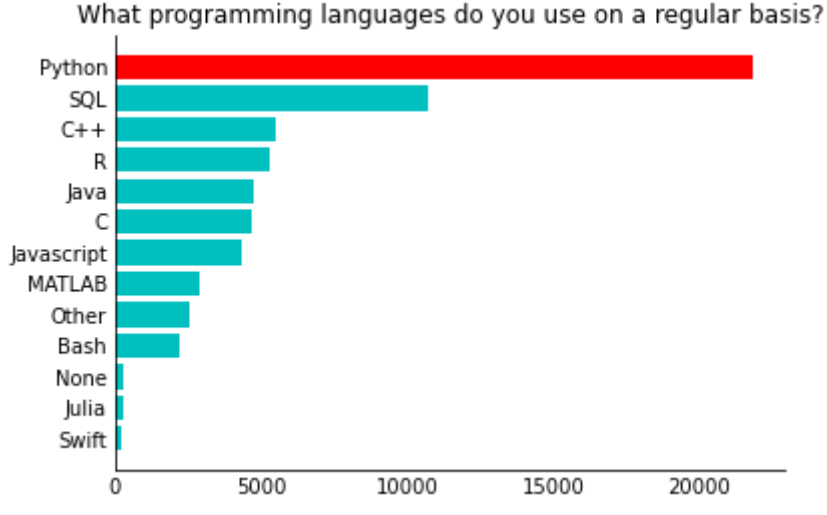
- 2021 Kaggle ML & DS Survey
- Stack Overflow Trends

# 2021 Kaggle ML & DS Survey 中的兩個問題

- Q7: What programming languages do you use on a regular basis? (Select all that apply)
- Q8: What programming language would you recommend an aspiring data scientist to learn first?

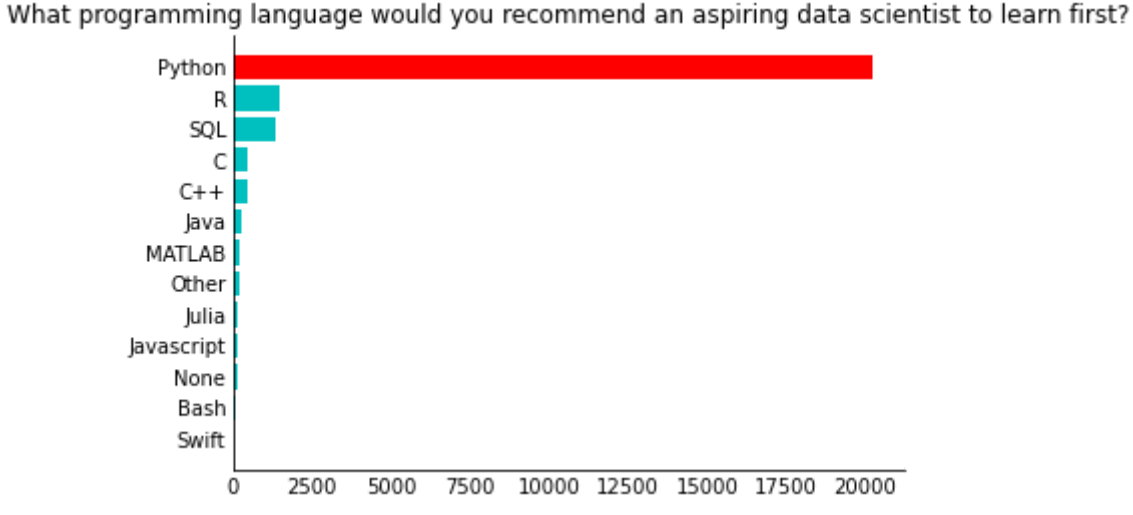
In [2]:

```
ks.plot_survey_summary(question_index="Q7", n=1)
```



In [3]:

```
ks.plot_survey_summary(question_index="Q8", n=1)
```



# Stack Overflow Trends

https://insights.stackoverflow.com/trends?tags=python%2Cr%2Cmatlab%2Cjulia%2Csas%2Cscala

# 關於 Python 的二三事

1. Python 的作者是荷蘭電腦科學家 Guido van Rossum
2. Python 的命名源於 Guido van Rossum 非常喜歡電視喜劇 Monty Python's Flying Circus
3. Python 的第一版釋出於 1991 年。

# 如何寫作與執行 Python 程式

# 寫作與執行 Python 程式需要三種類型的軟體

1. 純文字編輯器：寫作程式的軟體，例如記事本、Visual Studio Code 或 Notepad++。
2. 終端機：執行程式的軟體，例如 Windows 的命令提示字元、macOS 的 Terminal。
3. Python 直譯器：將 Python 程式翻譯為電腦語言的軟體。

# 將這三種類型的軟體整合在一起的軟體叫做整合開發環境（Integrated Development Environment, IDE）

受歡迎的 Python 整合開發環境有：

- **JupyterLab**/Jupyter Notebook
- PyCharm
- Spyder
- ...etc.

# 先從已經設定妥善的環境起步，之後會在「使用 conda 管理環境」章節介紹如何在自己的電腦建立 Python 資料科學環境

- 不需登入、不能儲存變更的 JupyterLab
- 需要使用 Google 帳號登入、能夠儲存變更的 Google Colab

# 初登場的兩個 Python 程式

1. 哈囉世界。
2. Python 禪學（The Zen of Python）。

# 哈囉世界

- `print()` 是 Python 的內建函數，可以將小括號中的輸入印出。
- `"Hello, world!"` 是屬於 `str` 類別的字面值（Literal value）。

In [4]:

```
print("Hello, world!")
```

```
Hello, world!
```

# Python 禪學（The Zen of Python）

- `import` 是 Python 的保留字（Keywords），可以載入模組。
- `this` 是 Python 的一個標準模組，可以印出 Python 禪學。

In [5]:

```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way t
o do it.
```

```
Although that way may not be obvious at first unless you're D
utch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good id
ea.
Namespaces are one honking great idea -- let's do more of tho
se!
```

# （沒什麼用的冷知識）Python 禪學在 `this` 模組中的內容是經過 ROT13 加密的文字

```
Gur Mra bs Clguba, ol Gvz Crgref

Ornhgvshy vf orggre guna htyl.
Rkcyvpvg vf orggre guna vzcyvpvg.
Fvzcyr vf orggre guna pbzcyrk.
Pbzcyrk vf orggre guna pbzcyvpngrq.
Syng vf orggre guna arfgrq.
Fcnefr vf orggre guna qrafr.
Ernqnovyvgl pbhagf.
Fcrpvny pnfrf nera'g fcrpvny rabhtu gb oernx gur ehyrf.
Nygubhtu cenpgvpnyvgl orngf chevgl.
Reebef fubhyq arire cnff fvyragyl.
Hayrff rkcyvpvgyl fvyraprq.
Va gur snpr bs nzovthvgl, ershfr gur grzcgngvba gb thrff.
Gurer fubhyq or bar-- naq cersrenoyl bayl bar --boivbhf jnl gb qb vg.
Nygubhtu gung jnl znl abg or boivbhf ng svefg hayrff lbh'er Qhgpu.
Abj vf orggre guna arire.
Nygubhtu arire vf bsgra orggre guna *evtug* abj.
Vs gur vzcyrzragngvba vf uneq gb rkcynva, vg'f n onq vqrn.
Vs gur vzcyrzragngvba vf rnfl gb rkcynva, vg znl or n tbbq vqrn.
Anzrfcnprf ner bar ubaxvat terng vqrn -- yrg'f qb zber bs gubfr!
```

# 在 JupyterLab 寫作與執行 Python 程式的兩種方式

1. 透過即時互動的筆記本介面寫作與執行。
2. 透過腳本（Script）寫作程式再以終端機執行。

# 兩種方式都很好，各有知名資料科學家支持

- fast.ai 的創辦人、Deep Learning for Coders with Fastai and PyTorch 的作者 Jeremy Howard 喜歡筆記本：https://youtu.be/9Q6sLbz37gk
- Data Science from Scratch 的作者 Joel Grus 不喜歡筆記本：https://youtu.be/7jiPeIFXb6U

# 課程中我們將透過寫作練習題在 JupyterLab 熟悉寫作與執行 Python 程式的兩種方式

1. 透過即時互動的筆記本介面寫作函數或類別。
2. 透過腳本（Script）在終端機批改測試。

# 在 Google Colab 寫作與執行 Python 程式

透過即時互動的筆記本介面寫作與執行：https://colab.research.google.com

# 如何寫作練習題並執行測試

# 練習題是「Python 的 50+ 練習」的課程重點

- 課程以 Exercise Based Learning 為主體。
- 練習題會利用類別提示（`typing`）來描述預期輸入以及預期輸出。
- LPAA 四步驟教學模型：透過 Learn（講師觀念講解）、Practice（講師範例操作）、Apply（同學寫作練習題）與 Assess（批改測試並觀看練習題詳解）循環強化學習效果。

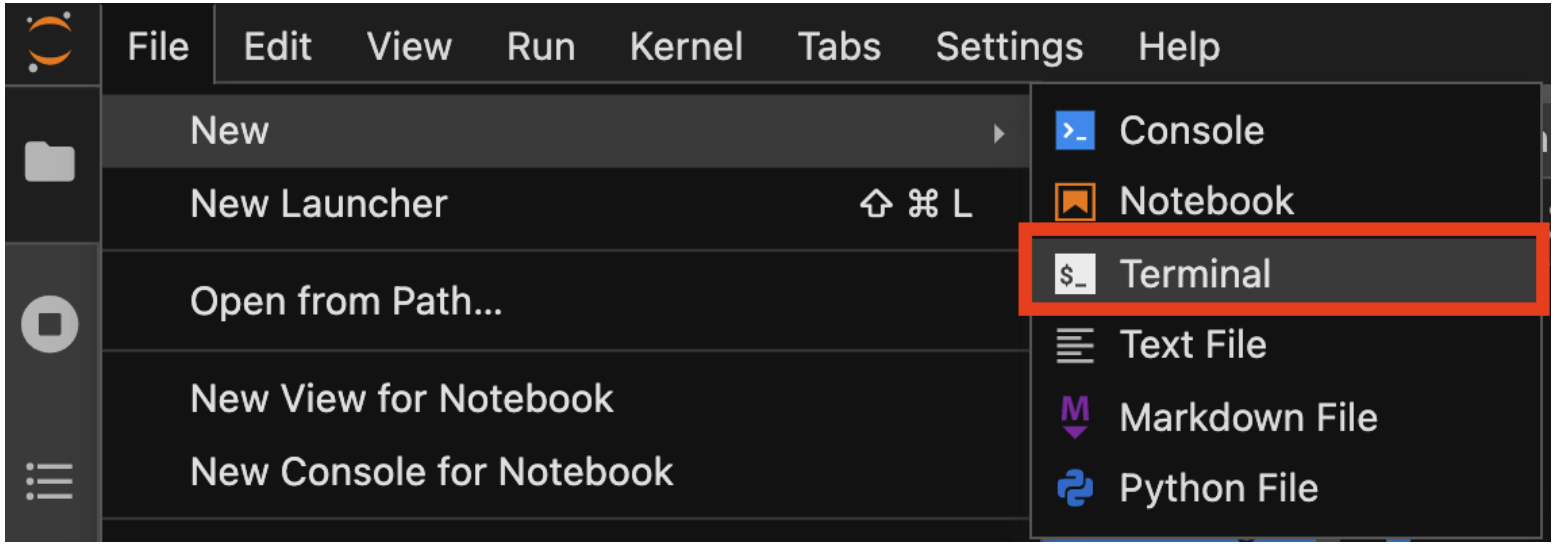
# 如何寫作練習題

- 在前幾個章節還不熟悉的時候仔細閱讀練習題指引並適時觀看練習題詳解。
- 在 `### BEGIN SOLUTION` 與 `### END SOLUTION` 之間運用預期輸入與參數寫出答案，並將答案寫在 `return` 保留字之後，若只是用 `print()` 印出預期輸出無法通過測試。
- 寫作完成之後點選 File -> Save Notebook 儲存 exercises.ipynb。

# 如何批改測試

- 點選上方選單的 File -> New -> Terminal 開啟終端機。
- 在 Terminal 輸入 `python 章節名稱/test_runner.py` 後按下 Enter 批改測試。
- 批改測試指令會在練習題指引中打好，複製在終端機貼上即可。

# File -> New -> Terminal 開啟終端機

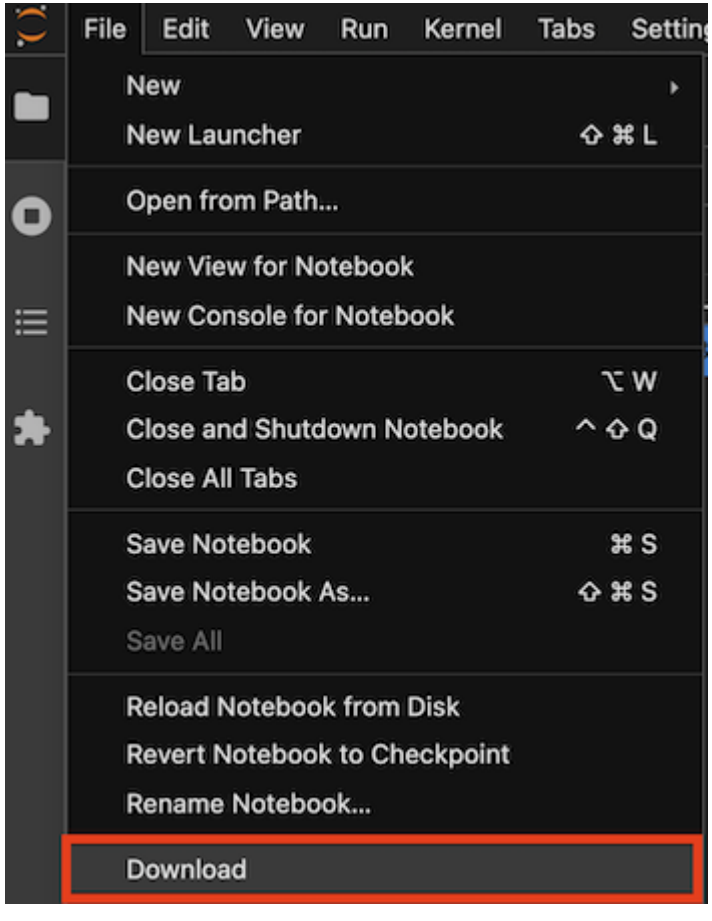


# 批改測試指令會在練習題指引中打好，複製在終端機貼上並按下 Enter 即可

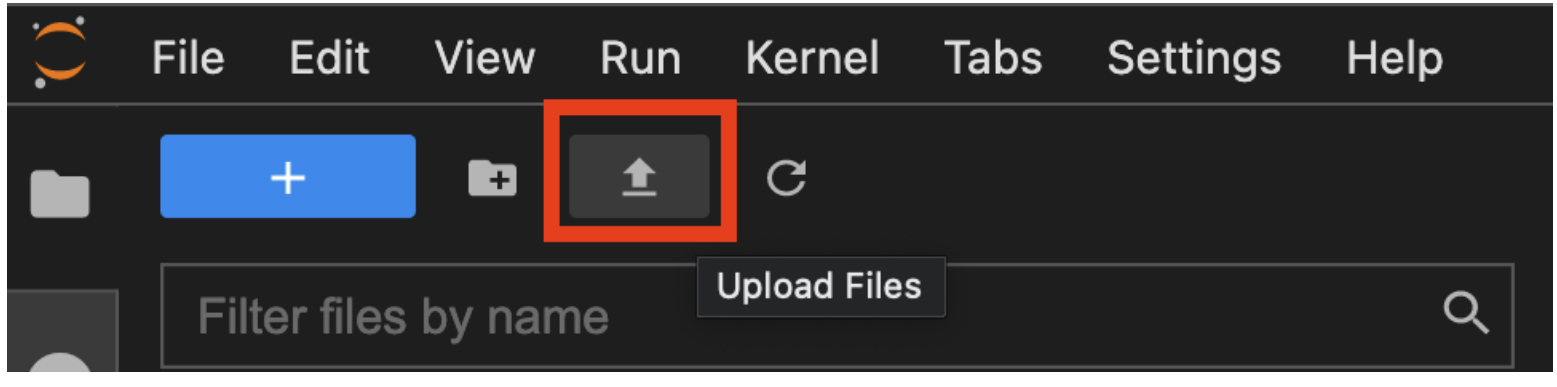
以第一章的練習題測試為例：

```
python 01-introduction/test_runner.py
```

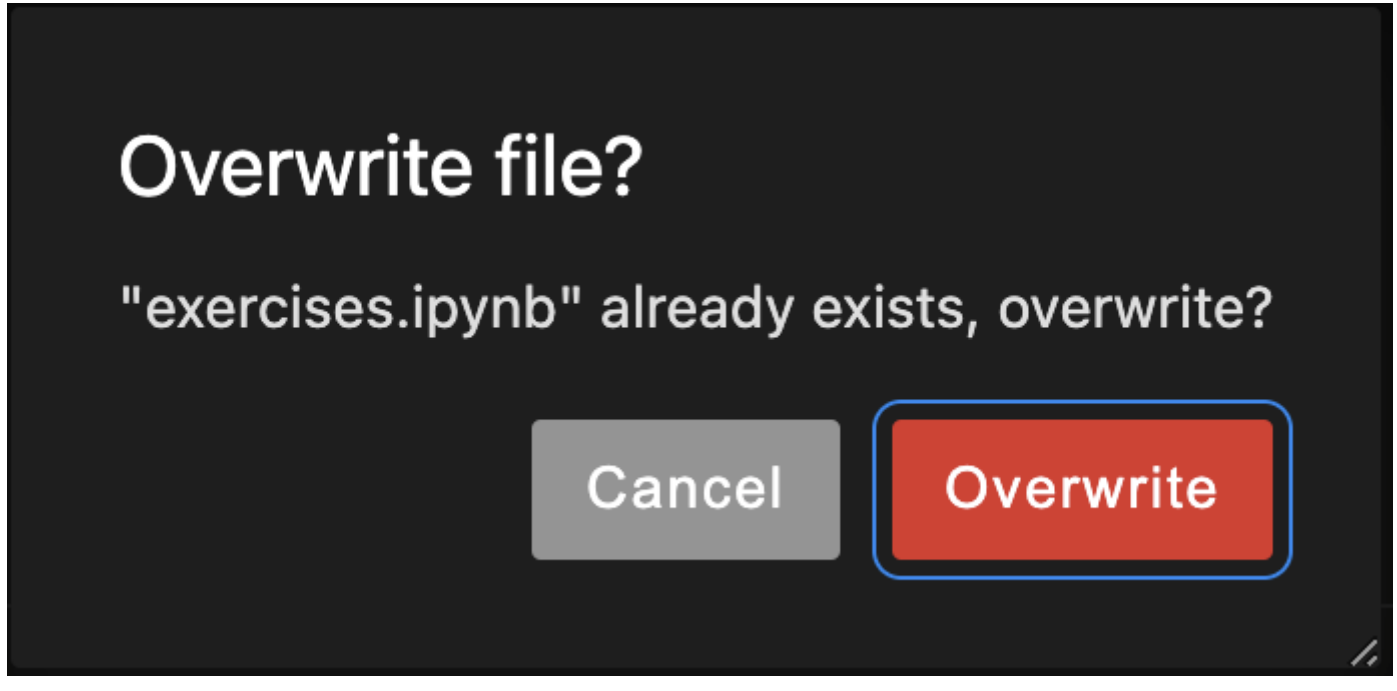
如果沒有足夠時間一次完成所有練習題，可以先將已完成部分的 exercises.ipynb 下載



# 下次再上傳 exercises.ipynb 繼續寫



點選 Overwrite 覆蓋原本空白的 exercises.ipynb



# 重點統整

- 「Python 的 50+ 練習」課程的兩個核心為程式設計與資料科學。
- 選擇 Python 作為資料科學應用的程式語言，因為這是目前最多人使用、推薦與使用人數上升的程式語言。
- 初登場的兩個 Python 程式為哈囉世界與 Python 禪學（The Zen of Python）。