AIoT：樹莓派應用

Chapter 3：Python X 基本影像辨識

學程式之前的知識

Outline

- Hello, World! 最像英文的 Python
- 物件、資料型別、變數、內建函式、匯入模組

Hello, World! 最像英文的 **Python**

Python

```python
#  印出 Hello World! 字串物件
print("Hello World!")
```

C

```c
/*  印出 Hello World! 字串物件*/
 include <stdio.h>

int main()
{
    printf("Hello, World!\n");
    return 0;
}
```

C++

```cpp
//印出 Hello World! 字串物件
#include <iostream>

using namespace std;

int main()
{
   cout << "Hello World" << endl;
   return 0;
```

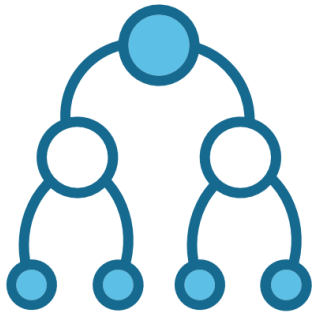
Java

```java
//印出 Hello World! 字串物件
public class HelloWorld{

     public static void main(String []args){
        System.out.println("Hello World");
     }
}
```

程式語言：五大語法結構

Sequence

循序執行

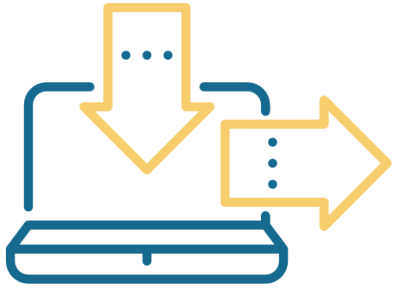


Conditional Statements

`if` 判斷式

Loops

迴圈



Input / Output

輸入/輸出

Functions

函數

Outline

- Hello, World! 最像英文的 Python
- 物件、資料型別、變數、內建函式、匯入模組

物件

- 一般寫英文句子時，會以名詞 + 動詞。Python 是以物件.方法來描述。

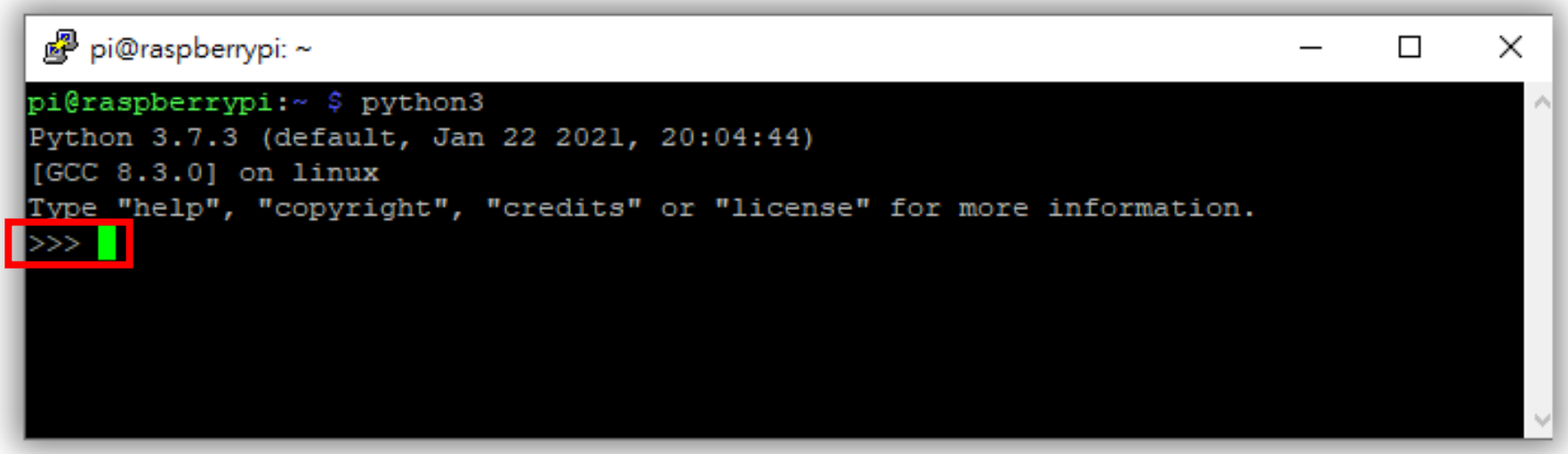
文章寫作	Python 編程	程式的解釋
車子	car	car 物件
車子向前進	car.start()	car 物件的 start 方法

- 方法後面會加上括號()，有些方法需要加入額外的參數。
  - ✓ 例如：car.go(100)，車子加速到 100。
- 若方法有多個參數，以逗點分隔。
  - ✓ 例如：car.left(50, 30)，以 50 的速度，向左轉 30 度。

進入互動模式

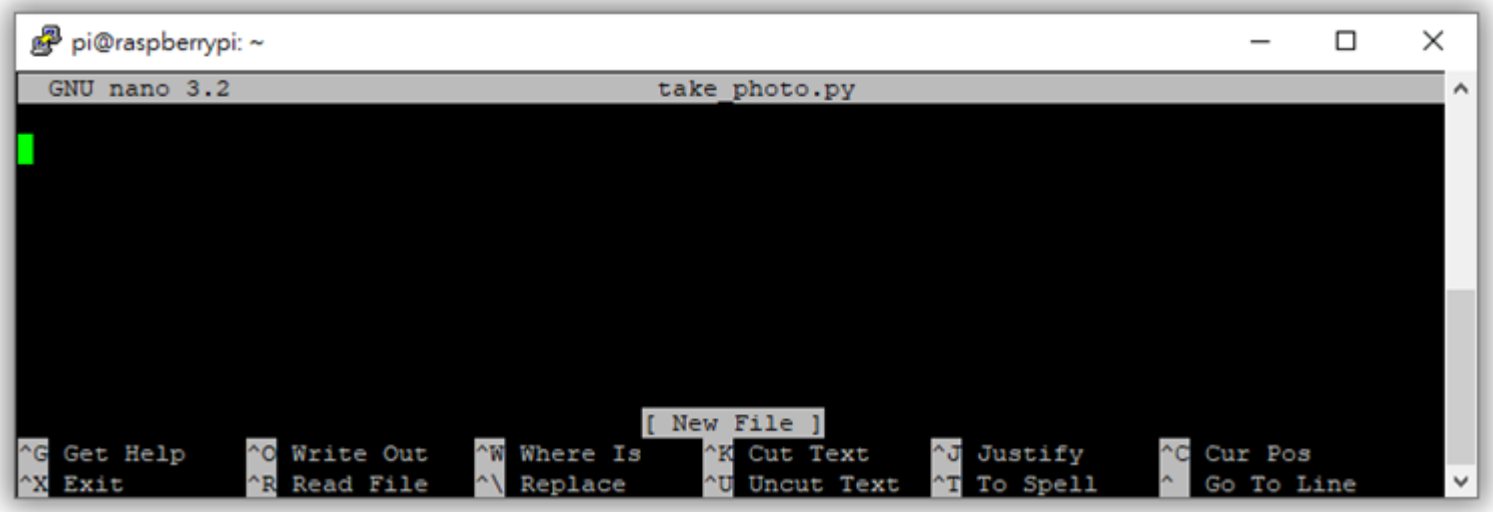
- 進到互動模式，撰寫程式碼直接看輸出結果（若執行 python，為 Python 2 的版本）

  >>> python3（>>> 是進入到互動模式）



編輯檔案：方法一（使用 nano）

- 使用 nano
  $ nano 檔名.副檔名 (例如：$ nano HelloWorld.py)
  - ✓ 強制離開：Ctrl + x
  - ✓ 存檔：Ctrl + o



- 存成檔案以後，用 python3 執行 (執行 python 為 Python 2)
  $ python3 檔案名稱.py (例如：$ python3 HelloWorld.py)

編輯檔案：方法二（leafpad 編輯器）

- 安裝 leafpad

  $ sudo apt-get update

  $ sudo apt-get install leafpad

- 在 X11 forwarding 連線成功下，執行 leafpad。

  $ leafpad &

使用 leafpad 編程

1. 開檔，輸入程式碼。(例如：在 ex3-8 底下)
2. 存檔，例如：Lab3-2-take_photo.py。
3. 執行：

```
$ python3 Lab3-2-take_photo.py
```

Pi 兩種執行 Python 的方法 (2/2)

- 使用內建的視窗程式 Thonny

```
$ thonny &
```

練習：操作字串物件的方法

ex3-1

- 在互動模式中，輸入下列敘述：(>>> 指的是在互動模式中，執行單行敘述)

```
>>> "Hello World!".upper()
'HELLO WORLD!'
>>> "Hello World!".find('r')
8
>>> "Hello World!".replace('r', 'u')
'Hello Would!'
```

使用字串物件 "Hello World!" 的 upper() 方法，將字串轉成大寫

find() 方法尋找 'b' 的位置 (從 0 開始)

replace() 方法將所有的 'r' 取代成 'b'

- 不同的物件會有不同的方法。例如：字串物件與整數物件。

資料型別

ex3-2

- 除字串物件以雙引號或單引號來表示，寫程式常有整數與浮點數(小數) 物件，例如：111 與 11.1。

```
>>> 111 + 111 ———— 整數物件相加
222
>>> "111" + "111" ———— 字串物件串聯
'111111'
```

- 上述 + 的運算，因物件的資料不同而產生不同的結果。物件的種類，程式語言稱之為『物件型態』或『資料型態』(data type)。

練習：要分清楚資料型別

ex3-3

- 兩個資料型別若不同，可能會導致程式錯誤。

```
>>> 111 + "111" —— 不同型別的資料相加，發生錯誤
Traceback (most recent call last):
  File "<stdin>”, line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int'
and 'str'
```

變數

- 『變數』(variable) 就像是掛在物件的名牌，幫物件取名之後，讓我們方便識別物件與操作，其語法為：

```
變數名稱 = 物件
```

- 例如：

```
>>> n1 = 123456789 ——— 將整數物件 123456789 指派給變數 n1
>>> n2 = 987654321 ——— 將整數物件 987654321 指派給變數 n2
>>> n1 + n2 ——————— 實際上是 123456789 + 987654321
1111111110
```

內建函式

ex3-5

- 『函式』(function) 是一段預先寫好的程式，方便重複使用。而程式先將經常使用到的功能以函式的形式先寫好，稱為『內建函式』。
- 例如：print() 是最常用的顯示函數：

```
>>> print("abc") ——— 顯示字串物件
abc
>>> print("abc".upper()) ——— 顯示字串物件.方法的執行結果
ABC
>>> print(111 + 111) ——— 顯示整數物件運算的結果
222
```

匯入模組

ex3-6

- 內建函式不就越多越好？若內建函式無限制增加，會導致啟動速度越來越慢，執行時佔用的記憶體越來越多。
- 『模組』(module)，就是將同一類的函式打包成模組，預設不會啟用。需要時，再用匯入 (import) 的方式啟用。預先寫好的稱為『內建模組』。

```
>>> import time ——— 匯入時間相關的 time 模組
>>> time.sleep(3)  ——— 執行 time 模組的 sleep() 函式，暫停 3 秒


>>> from time import sleep ——— 從 time 模組裡匯入 sleep() 函式
>>> sleep(5) ——— 執行 sleep() 函式，暫停 5 秒
```

練習：匯入模組

ex3-7

Lab3-1 暫停 5 秒後，印出 Hello World! 字串物件。

```
# 暫停 5 秒後，印出 Hello World!
from time import sleep


sleep(5)
print("Hello World!")
```

指的是在單獨一個檔案中，編輯程式。

結論

各種程式語言的語法邏輯都差不多，就像人類語言的文法。

1. 程式的每一個東西都是物件，有些物件有其特定的操作方法。
2. 基本物件有資料型別，型別不同，結果不同。甚至有時會錯誤。
3. 變數是物件的名牌而已，也方便程式設計師操作。
4. 內建函式是經常用的函式，預先寫好的。
5. 適當的匯入模組，能精簡效能。

exper3-1

實驗 3-1：寫程式控制 Camera

使用 picamera（Python 支援的模組）

https://picamera.readthedocs.io/en/release-1.13/

照相

ex3-8

- 預設的相片解析度為 720 x 480

Lab3-2 使用 picamera 函式庫照相。(Lab3-2-take_photo.py)

```
import picamera
import time

camera = picamera.PiCamera()
time.sleep(2)              #暫停 2 秒
camera.rotation = 180
camera.capture('Lab3-2-take_photo.jpg')
```

Demo

Lab3-2-take_photo.py

```
$ cd ~/exper3-1
$ nano Lab3-2-take_photo.py
$ python3 Lab3-2-take_photo.py
```

低光源拍照

ex3-10

Lab3-3 使用 picamera 函式庫，低光源拍照。(Lab3-3-low_light.py)

```
import picamera
import time
from fractions import Fraction
camera = picamera.PiCamera()
camera.resolution = (640, 480)
camera.framerate = Fraction(1, 6)
camera.shutter_speed = 6000000
camera.iso = 800
time.sleep(3)
camera.exposure_mode = 'off'
camera.rotation = 180
camera.capture('Lab3-3-low_light.jpg')
```

https://picamera.readthedocs.io/en/release-1.13/recipes1.html

Demo

Lab3-3-low_light.py

```
$ cd ~/exper3-1
$ nano Lab3-3-low_light.py
$ python3 Lab3-3-low_light.py
```

錄影

ex3-11

- 錄 3 秒鐘影像，儲存到檔案 Lab3-4-record_video.h264
- 預設錄影格式為 H.264/AVC 壓縮，解析度 1280 x 800

Lab3-4 使用 picamera 函式庫錄影。

```
import picamera

camera = picamera.PiCamera()
camera.rotation = 180
camera.start_recording('Lab3-4-record_video.h264')
camera.wait_recording(3)
camera.stop_recording()
```

https://picamera.readthedocs.io/en/release-1.13/recipes1.html

Demo

Lab3-4-record_video.py

```
$ cd ~/exper3-1
$ nano Lab3-4-record_video.py
$ python3 Lab3-4-record_video.py
```

練習

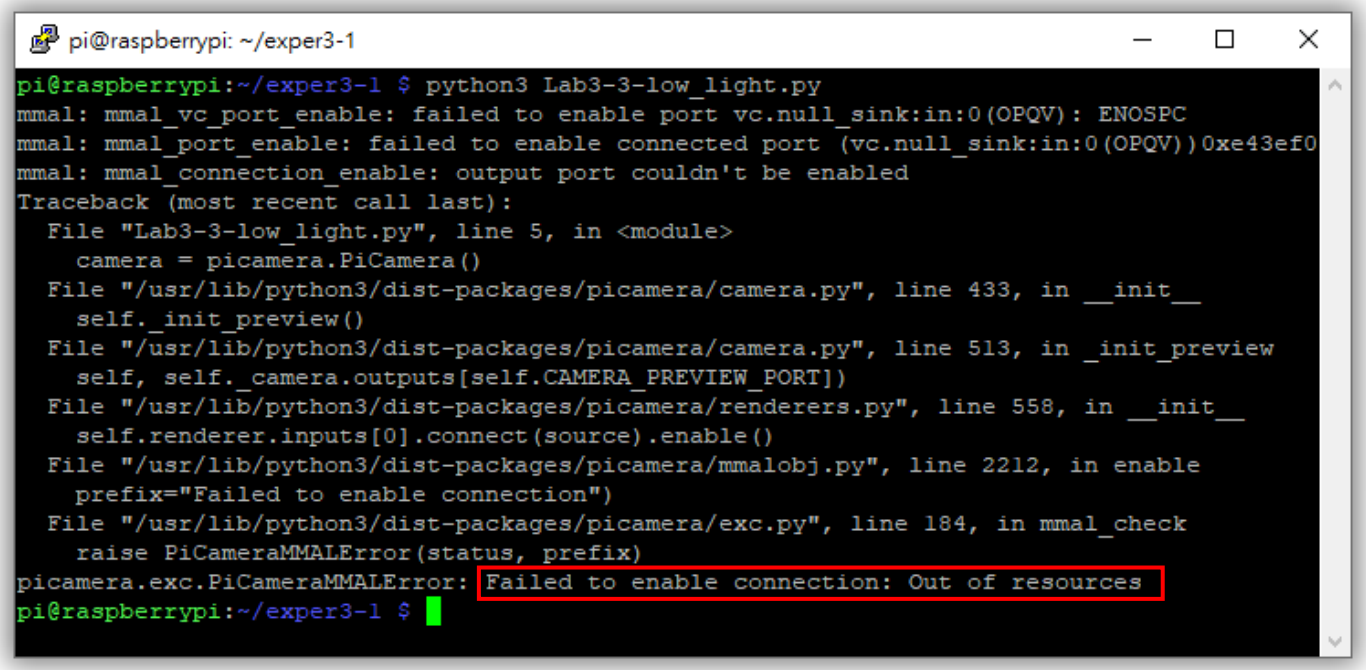
ex3-12

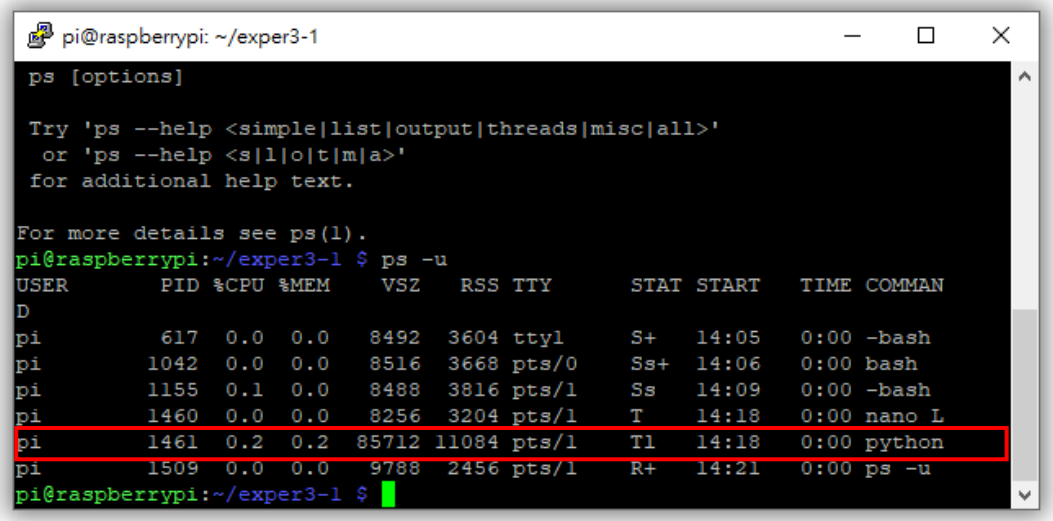
- 請查詢網頁文件，在拍照後的圖片疊上文字
  - ✓ https://picamera.readthedocs.io/en/release-1.13/recipes1.html
- 文字包含：日期與時間
  - ✓ 例如：年-月-日 時：分：秒
  - ✓ 提示：overlay 表示疊 (自己找解答)

小結

- picamera 是 Pi Camera 的 Python 套件
- 參考網頁文件：
  https://picamera.readthedocs.io/en/release-1.13/recipes1.html

若編譯時，發生以下錯誤





- 代表資源被卡在 python 程序裡：

  $ ps –u (找出 python 的程序)

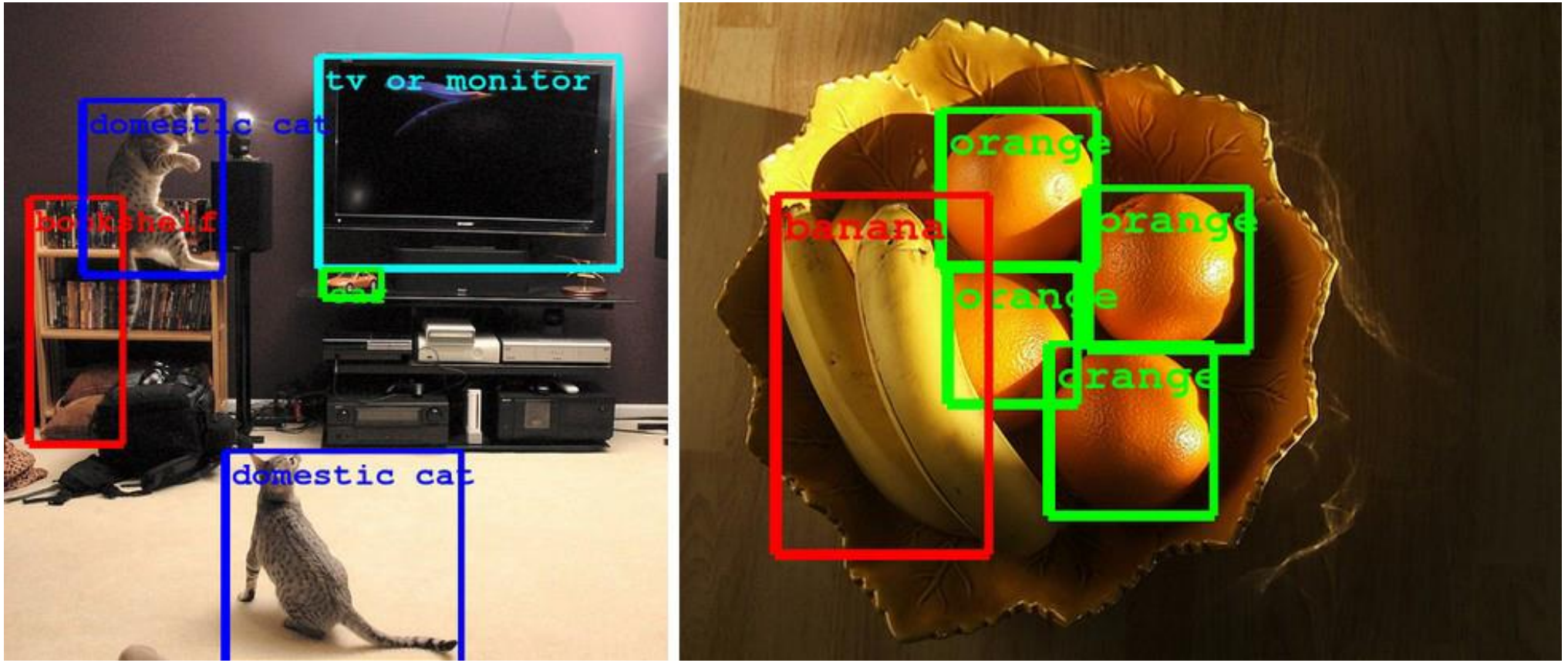
  $ kill -9 pid (pid：process id)

exper3-2

實驗 3-2：會認東西的 Camera

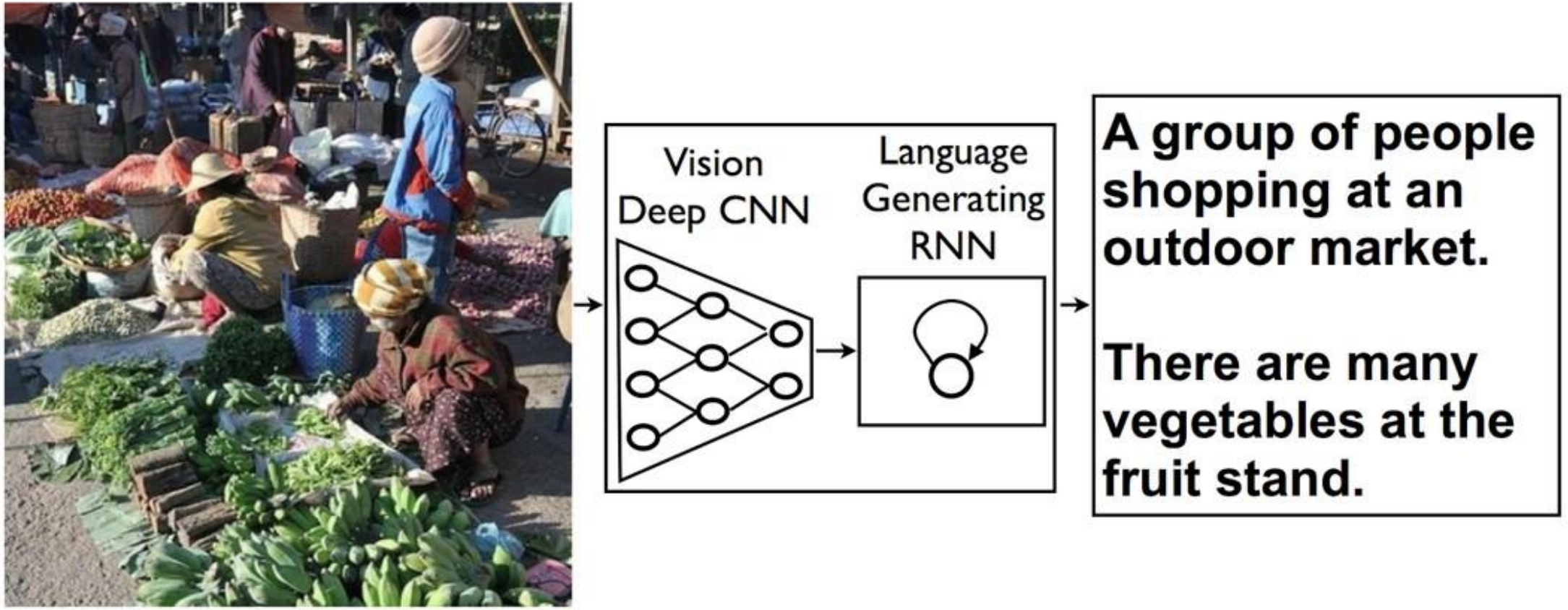
目的：串接網路服務

影像辨識



https://ai.googleblog.com/2014/09/building-deeper-understanding-of-images.html

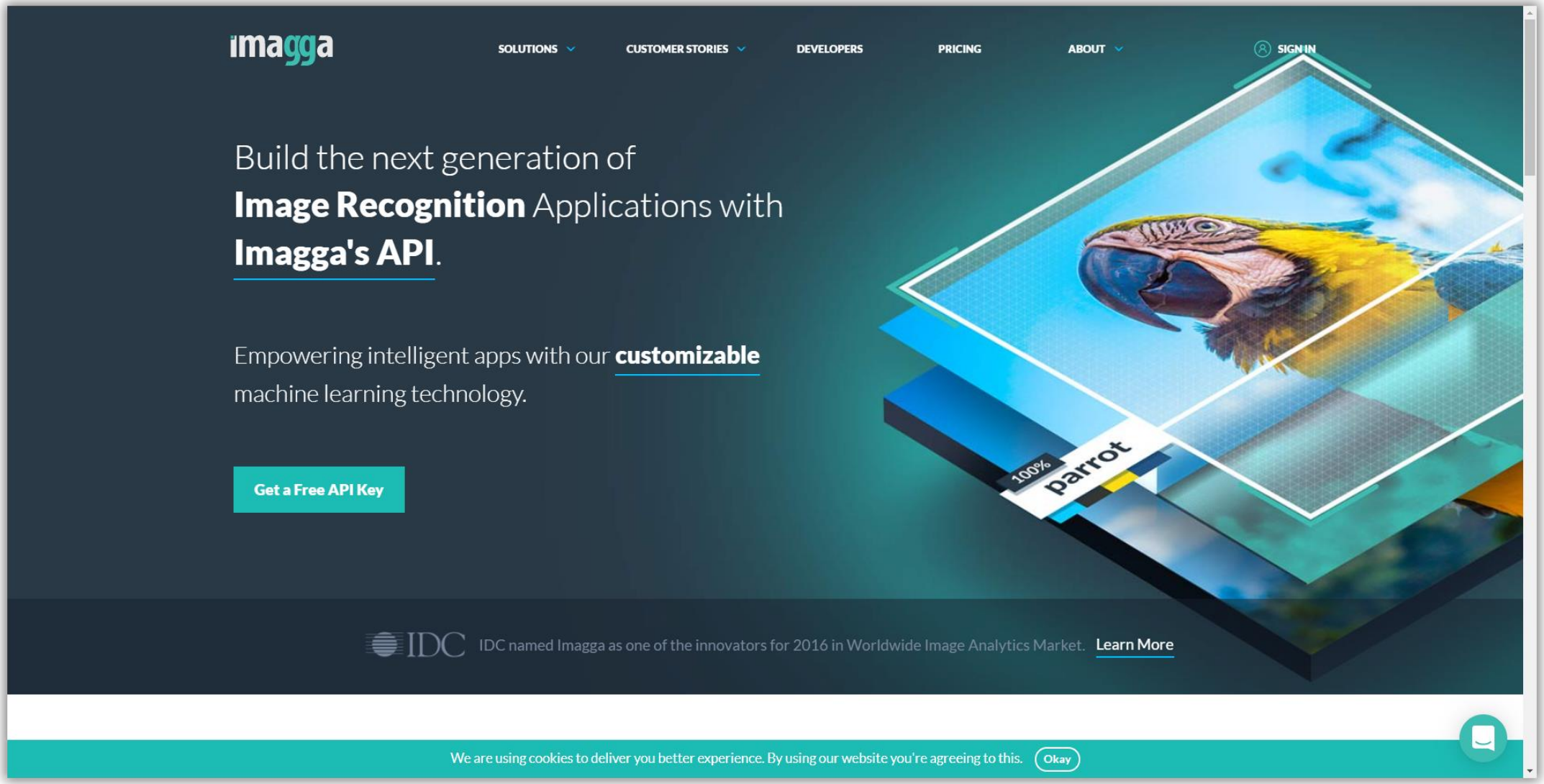
更強大的看圖說故事



CNN：Convolutional Neural Network (捲積式類神經網路)
RNN：Recurrent Neural Network (遞迴式類神經網路)

https://ai.googleblog.com/2014/11/a-picture-is-worth-thousand-coherent.html

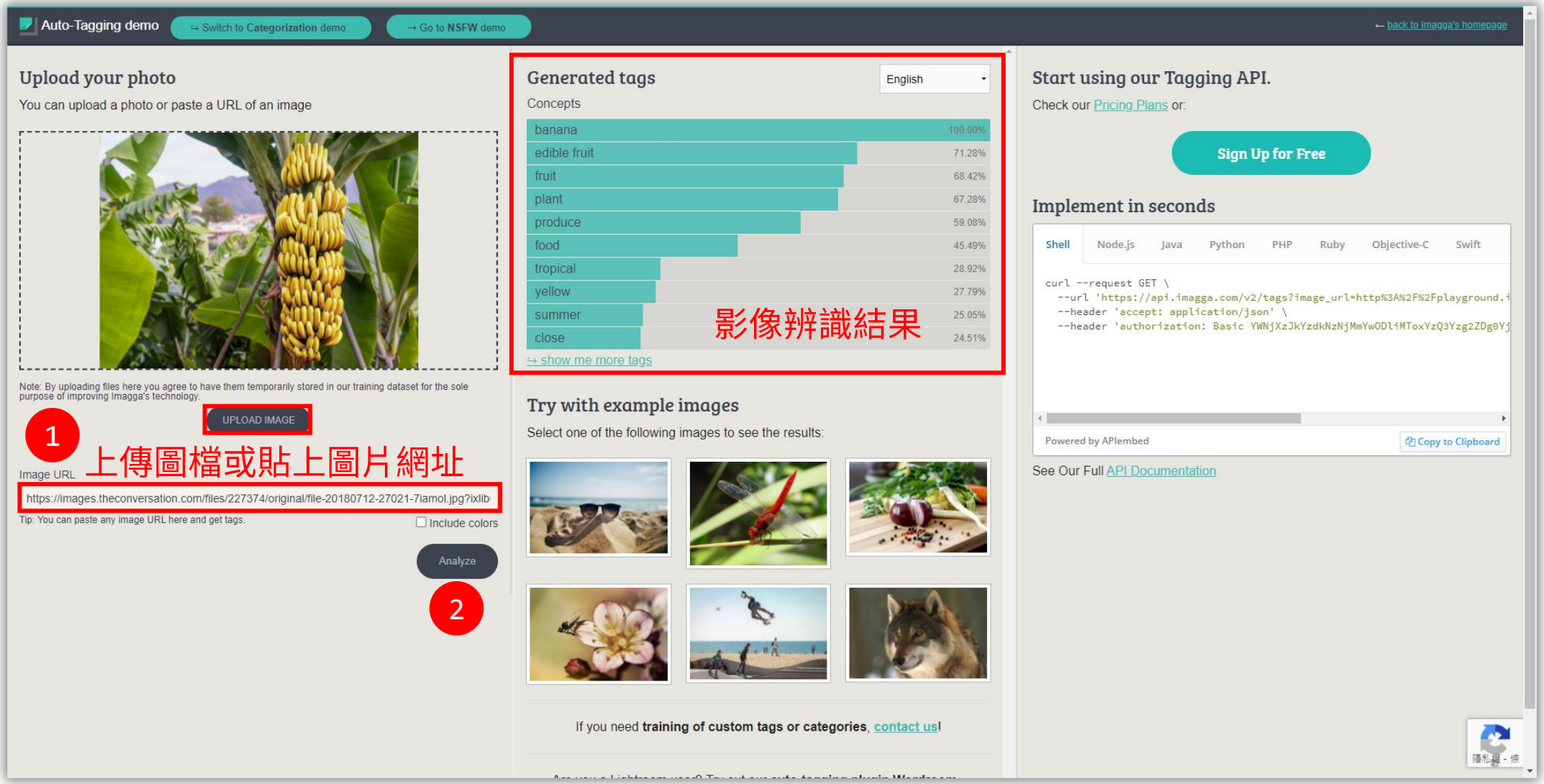
影像分類服務



https://imagga.com/

看 DEMO (Auto-Tagging)

ex3-13



https://imagga.com/auto-tagging-demo

如何開始使用服務？

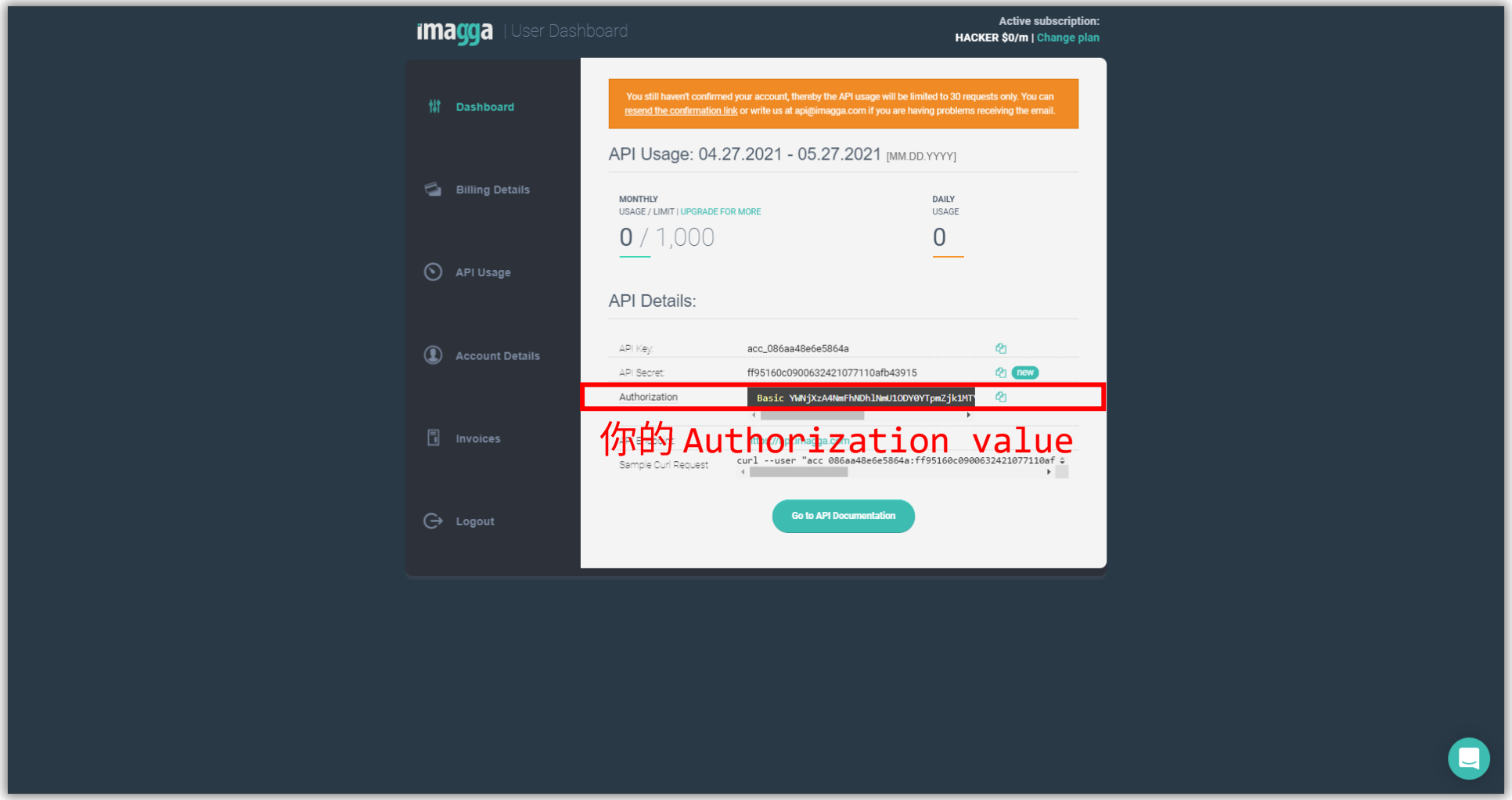
ex3-14

1. 註冊與認證
   - ✓ https://imagga.com/auth/signup
2. 取得 Authorization
   - ✓ https://imagga.com/profile/dashboard
3. 串接
   - ✓ input：程式指定圖片的 URL 或是上傳圖檔，進行影像辨識。
   - ✓ output：從傳回的 JSON 字串，擷取必要的資訊。

https://docs.imagga.com/

Dashboard

- API 所需要的資料



https://imagga.com/profile/dashboard

如何開始使用服務？

1. 註冊與認證
   - ✓ https://imagga.com/auth/signup
2. 取得 Authorization
   - ✓ https://imagga.com/profile/dashboard
3. 串接
   - ✓ input：程式指定圖片的 URL 或是上傳圖檔，進行影像辨識。
   - ✓ output：從傳回的 JSON 字串，分析並擷取必要的資訊。

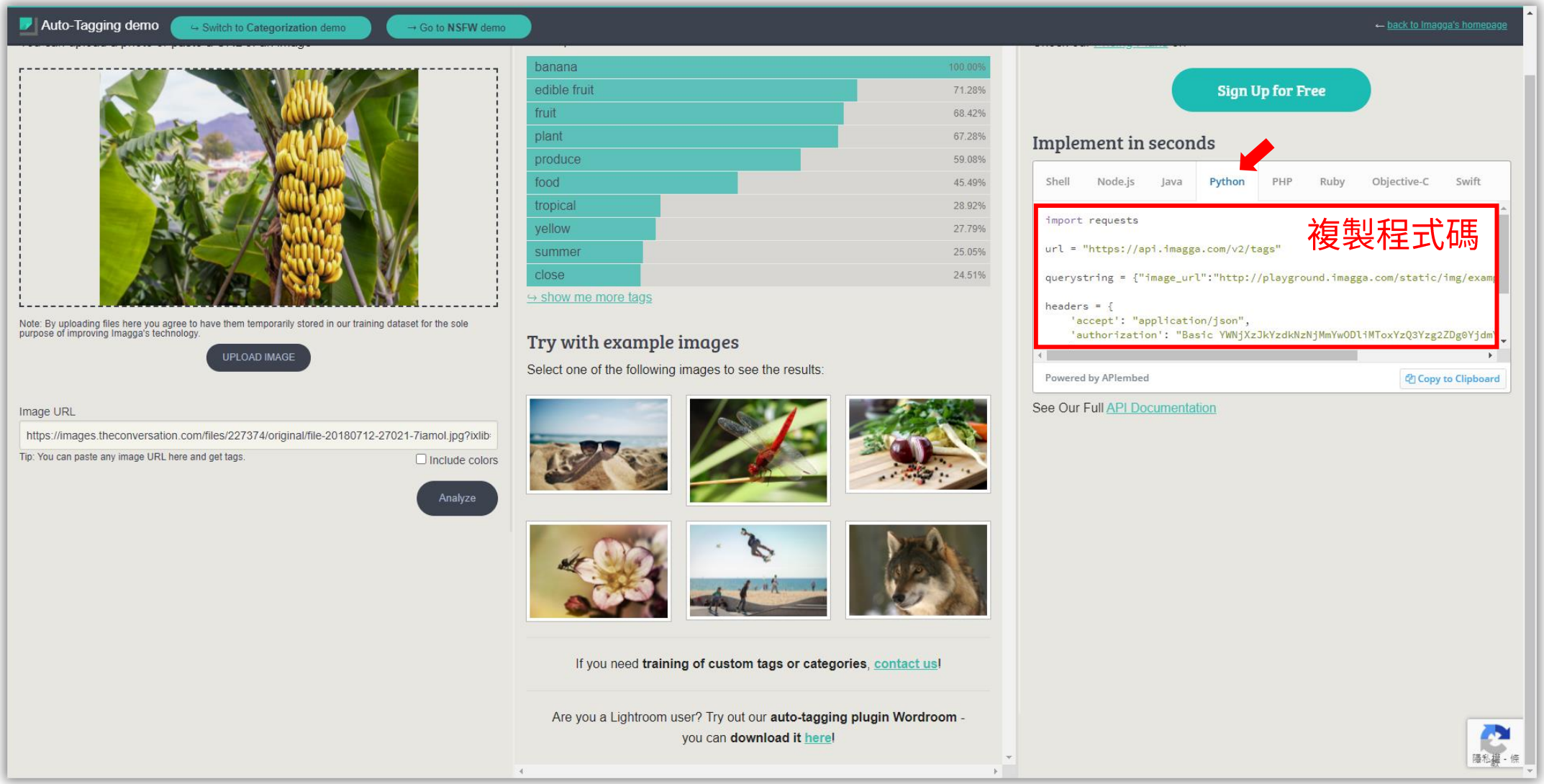
https://docs.imagga.com/

HTTP 兩三件事

GET & POST Method

- RFC 2616 / Hypertext Transfer Protocol - HTTP/1.1
- HTTP method
  - ✓ OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT
- GET 像明信片 (瀏覽網頁)
  - ✓ 資料 (query string) 在 URL 傳送
- POST 像信紙 + 信封 (收發 email)
  - ✓ 資料可以在 message-header
  - ✓ 也可以在 message-body

使用官網的 Python for imagga API



Python 程式串接

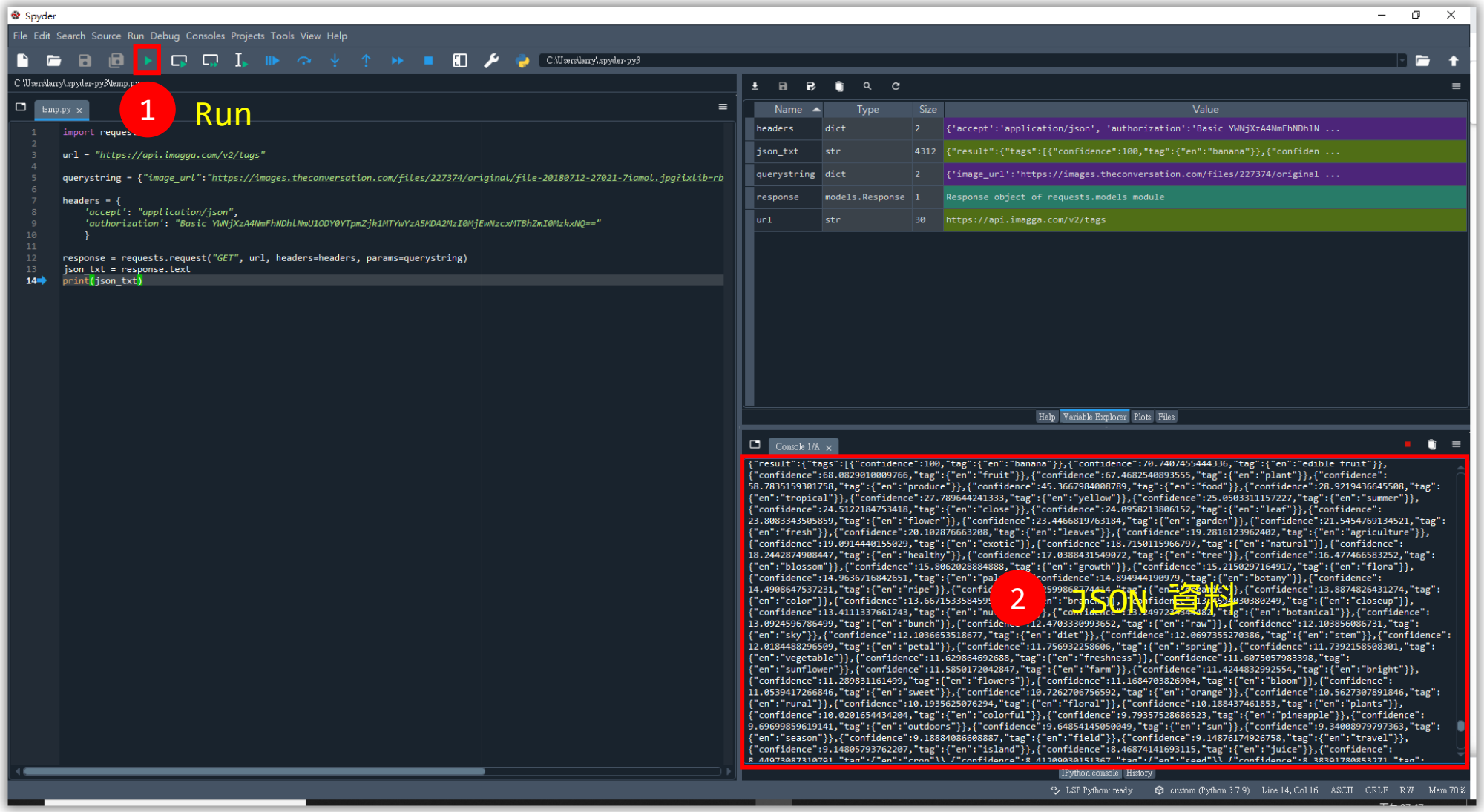
Lab3-5 imagga API 的程式。(Lab3-5-imagga_tag_file_url.py)

```
import requests

url = "https://api.imagga.com/v2/tags"

querystring =                                                    置換有效的圖片網址
{"image_url":"http://playground.imagga.com/static/img/example_photo.jpg","v
ersion":"2"}

headers = {
    'accept': "application/json",
    'authorization': "Basic                                  置換有效的 Authorization value
YWNjXzA4NmFhNDhlNmU1ODY0YTpmZjk1MTYwYzA5MDA2MzI0MjEwNzcxMTBhZmI0MzkxNQ=="
    }

response = requests.request("GET", url, headers=headers, params=querystring)
print(response.text)
```

Demo


pi@raspberrypi: ~/ex3-15
-bash: json_txt：命令找不到
pi@raspberrypi:~/ex3-15 $ nano Lab3-5.py
pi@raspberrypi:~/ex3-15 $
pi@raspberrypi:~/ex3-15 $
pi@raspberrypi:~/ex3-15 $ python3 Lab3-5.py
JSON 資料
{"result":{"tags":[{"confidence":100,"tag":{"en":"banana"}},{"confidence":70.740
7455444336,"tag":{"en":"edible fruit"}},{"confidence":68.0829010009766,"tag":{"e
n":"fruit"}},{"confidence":67.4682540893555,"tag":{"en":"plant"}},{"confidence":
58.7835159301758,"tag":{"en":"produce"}},{"confidence":45.3667984008789,"tag":{"
en":"food"}},{"confidence":28.9219436645508,"tag":{"en":"tropical"}},{"confidenc
e":27.789644241333,"tag":{"en":"yellow"}},{"confidence":25.0503311157227,"tag":{
"en":"summer"}},{"confidence":24.5122184753418,"tag":{"en":"close"}},{"confidenc
e":24.0958213806152,"tag":{"en":"leaf"}},{"confidence":23.8083343505859,"tag":{"
en":"flower"}},{"confidence":23.4466819763184,"tag":{"en":"garden"}},{"confidenc
e":21.5454769134521,"tag":{"en":"fresh"}},{"confidence":20.102876663208,"tag":{"
en":"leaves"}},{"confidence":19.2816123962402,"tag":{"en":"agriculture"}},{"conf
idence":19.0914440155029,"tag":{"en":"exotic"}},{"confidence":18.7150115966797,"
tag":{"en":"natural"}},{"confidence":18.2442874908447,"tag":{"en":"healthy"}},{"
confidence":17.0388431549072,"tag":{"en":"tree"}},{"confidence":16.477466583252,
"tag":{"en":"blossom"}},{"confidence":15.8062028884888,"tag":{"en":"growth"}},{"
confidence":15.2150297164917,"tag":{"en":"flora"}},{"confidence":14.963671684265
1,"tag":{"en":"palm"}},{"confidence":14.894944190979,"tag":{"en":"botany"}},{"co
nfidence":14.4908647537231,"tag":{"en":"ripe"}},{"confidence":14.2599868774414,"
tag":{"en":"organic"}},{"confidence":13.8874826431274,"tag":{"en":"color"}},{"co
nfidence":13.6671533584595,"tag":{"en":"branch"}},{"confidence":13.4594030380249
,"tag":{"en":"closeup"}},{"confidence":13.4111337661743,"tag":{"en":"nutrition"}
},{"confidence":13.2497234344482,"tag":{"en":"botanical"}},{"confidence":13.0924
596786499,"tag":{"en":"bunch"}},{"confidence":12.4703330993652,"tag":{"en":"raw"
}},{"confidence":12.103856086731,"tag":{"en":"sky"}},{"confidence":12.1036653518
677,"tag":{"en":"diet"}},{"confidence":12.0697355270386,"tag":{"en":"stem"}},{"c

Demo（使用 Spyder IDE 5 執行看看）



https://www.spyder-ide.org/

Python 的資料結構（容器）

Python 的資料結構

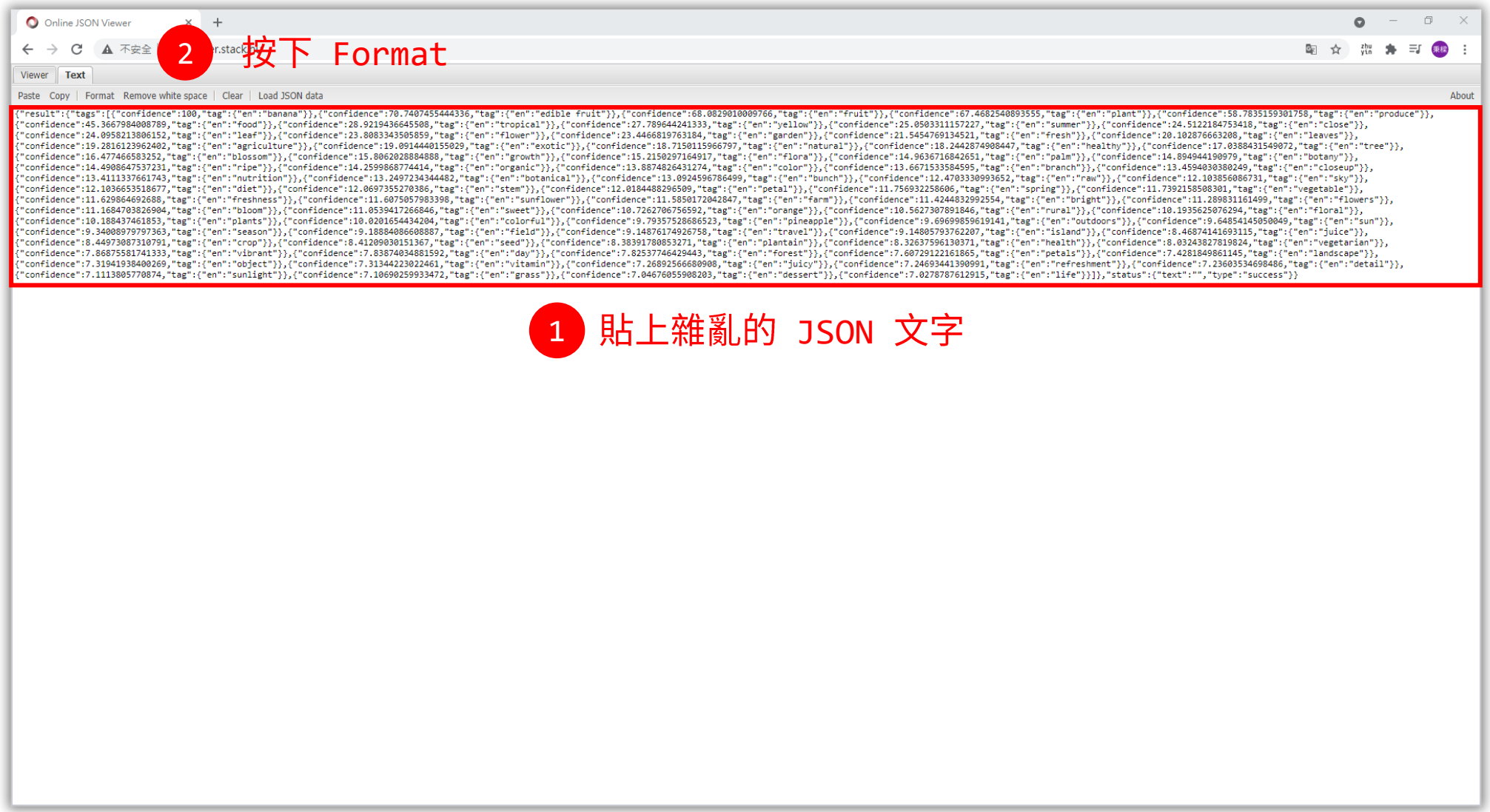
- 字串容器：由字元組成。例如：`string = "52python"`。
- `tuple` 容器：由資料物件組成。例如：`tuple = (1, '2', 3)`。
- 串列容器：由資料物件組成。例如：`list = [1, '2', 3]`。
- 集合容器：由資料物件組成。例如：`set = {1, '2', 3}`。
- 字典容器：由資料物件組成，以鍵：值表示。例如：`dick = {'A':1, 'B':'2', 'C':3}`。

JSON 資料格式解析

- JSON (JavaScript Object Notation) 是一種資料結構。
  - ✓ 原本是 JavaScript 中，以文字形式描述物件內容的格式。
  - ✓ 由於簡單易用，現在變成多層結構資料的常見格式。

Online JSON Viewer (1/2)

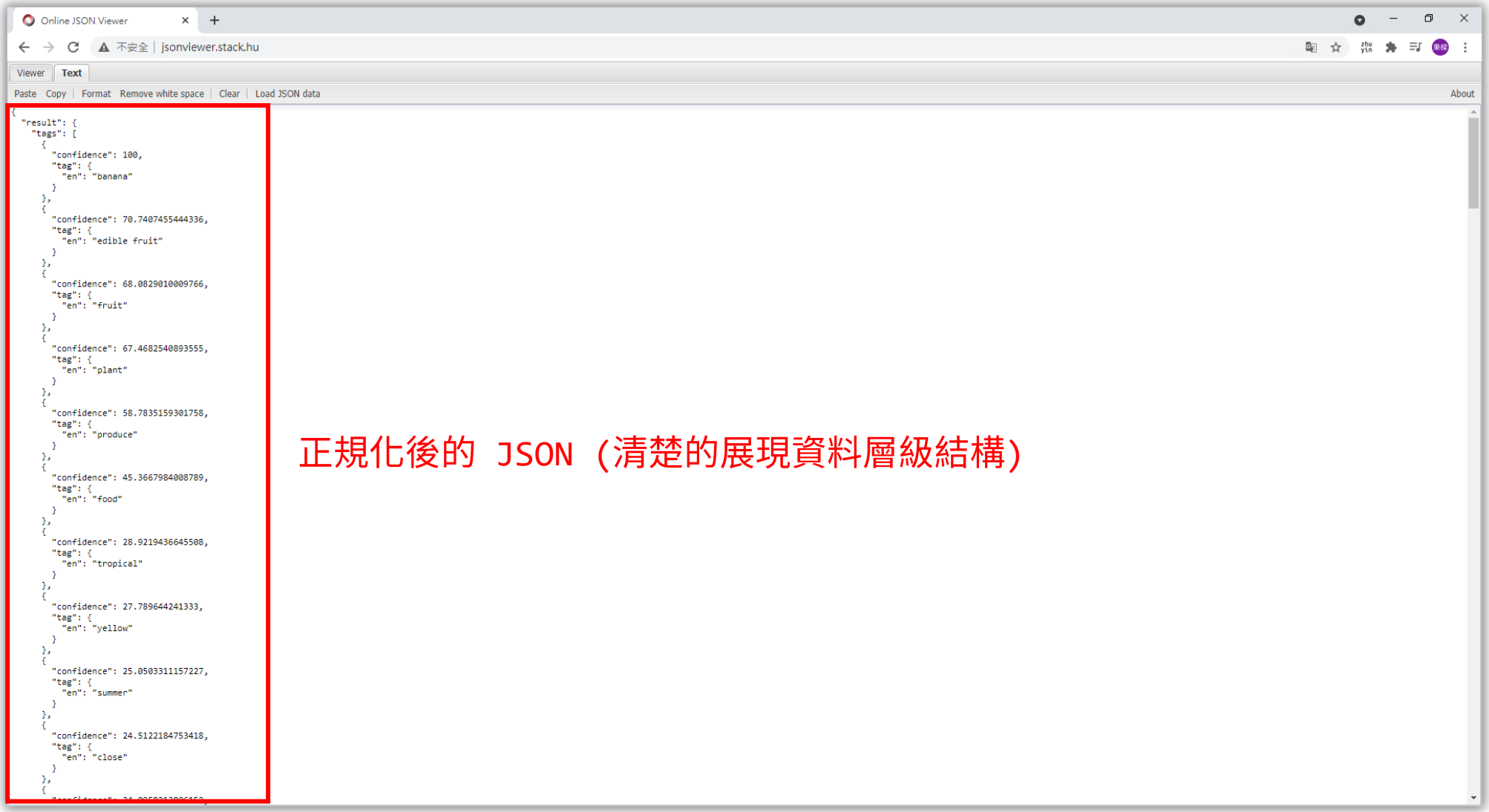
ex3-17



http://jsonviewer.stack.hu/

Online JSON Viewer (2/2)

ex3-18



正規化後的 JSON (清楚的展現資料層級結構)

http://jsonviewer.stack.hu/

Python 程式串接

Lab3-6 imagga API 的程式 (轉成 JSON)。

```
import requests
import json

url = "https://api.imagga.com/v2/tags"

querystring =
{"image_url":"https://images.theconversation.com/files/227374/original/file-
20180712-27021-7iamol.jpg?ixlib=rb-
1.1.0&q=45&auto=format&w=1200&h=900.0&fit=crop","version":"2"}

headers = {
    'accept': "application/json",
    'authorization': "Basic
YWNjXzA4NmFhNDhlNmU1ODY0YTpmZjk1MTYwYzA5MDA2MzI0MjEwNzcxMTBhZmI0MzkxNQ=="
    }

response = requests.request("GET", url, headers=headers, params=querystring)
data = json.loads(response.text)
print(data)
```

索引 JSON (1/6)

ex3-20

data

```
{
  'result': {
    'tags': [
      {
        'confidence': 100,
        'tag': {
          'en': 'banana'
        }
      },
      {
        'confidence': 70.7407455444336,
        'tag': {
          'en': 'edible fruit'
        }
      },
```

```
data = json.loads(response.text)
print(data)
```

索引 JSON (2/6)

ex3-20

data["result"]

```
{
  'result': {
    'tags': [
      {
        'confidence': 100,
        'tag': {
          'en': 'banana'
        }
      },
      {
        'confidence': 70.7407455444336,
        'tag': {
          'en': 'edible fruit'
        }
      },
```

```
data = json.loads(response.text)
print(data["result"])
```

索引 JSON (3/6)

ex3-20

data["result"]["tags"]

```
{
  'result': {
    'tags': [
      {
        'confidence': 100,
        'tag': {
          'en': 'banana'
        }
      },
      {
        'confidence': 70.7407455444336,
        'tag': {
          'en': 'edible fruit'
        }
      },
```

```
data = json.loads(response.text)
print(data["result"]["tags"])
```

索引 JSON (4/6)

ex3-20

```
{
  'result': {
    'tags': [
      {
        'confidence': 100,
        'tag': {
          'en': 'banana'
        }
      },
      {
        'confidence': 70.7407455444336,
        'tag': {
          'en': 'edible fruit'
        }
      },
```

data["result"]["tags"][0]

```
data = json.loads(response.text)
print(data["result"]["tags"][0])
```

索引 JSON (5/6)

ex3-20

```
{
  'result': {
    'tags': [
      {
        'confidence': 100,
        'tag': {
          'en': 'banana'
        }
      },
      {
        'confidence': 70.7407455444336,
        'tag': {
          'en': 'edible fruit'
        }
      },
```

data["result"]["tags"][0]["tag"]

```
data = json.loads(response.text)
print(data["result"]["tags"][0]["tag"])
```

索引 JSON (6/6)

ex3-20

```
{
  'result': {
    'tags': [
      {
        'confidence': 100,
        'tag': {
          'en': 'banana'
        }
      },
      {
        'confidence': 70.7407455444336,
        'tag': {
          'en': 'edible fruit'
        }
      },
```

data["result"]["tags"][0]["tag"]["en"]

```
data = json.loads(response.text)
print(data["result"]["tags"][0]["tag"]["en"])
```

Python 程式串接

Lab3-7 imagga API 的程式 (擷取 JSON 結果)。

```
import requests
import json

url = "https://api.imagga.com/v2/tags"

querystring =
{"image_url":"https://images.theconversation.com/files/227374/original/file-
20180712-27021-7iamol.jpg?ixlib=rb-
1.1.0&q=45&auto=format&w=1200&h=900.0&fit=crop","version":"2"}

headers = {
    'accept': "application/json",
    'authorization': "Basic
YWNjXzA4NmFhNDhlNmU1ODY0YTpmZjk1MTYwYzA5MDA2MzI0MjEwNzcxMTBhZmI0MzkxNQ=="
    }

response = requests.request("GET", url, headers=headers, params=querystring)
data = json.loads(response.text)
tag = data["result"]["tags"][0]["tag"]["en"]
print("Get tag... ")
print("<< " + tag + " >>")
```

exper3-2

Demo

Lab3-7-imagga_tag_file_url.py

```
$ cd ~/exper3-2
$ nano Lab3-7-imagga_tag_file_url.py
$ python3 Lab3-7-imagga_tag_file_url.py
```

Python 程式串接 (Upload File)

- 先拍張照片
- 根據文件得知查詢上傳的檔案，需要兩個步驟
  - ✓ 上傳檔案後取得檔案 uid
  - ✓ 將 uid 以參數方式送出查詢

上傳檔案取得檔案 uid

Lab3-8 取得上傳檔案的 uid。(Lab3-8-imagga_tag_upload_file.py)

```
import requests
import json

api_key = '你的 api_key'
api_secret = '你的 api_secret'
image_path = '照片檔的路徑'

response_post = requests.post(
    'https://api.imagga.com/v2/uploads',
    auth = (api_key, api_secret),
    files = {'image': open(image_path, 'rb')})

data_post = response_post.json()
querystring = data_post["result"]["upload_id"]
print(querystring)
```

將 uid 以參數方式送出查詢

```
# …接前頁
response_get = requests.get(
    'https://api.imagga.com/v2/tags?image_upload_id=' + querystring,
    auth=(api_key, api_secret))

data_get = json.loads(response_get.text)
tag = data_get["result"]["tags"][0]["tag"]["en"]
print("Get tag... ")
print("<< " + tag + " >>")
```

Demo

Lab3-8-imagga_tag_upload_file.py

```
$ cd ~/exper3-2
$ nano Lab3-8-imagga_tag_upload_file.py
$ python3 Lab3-8-imagga_tag_upload_file.py
```

小結

- Python 的 request 套件可提供 get 和 post 方法
- 網路 RESTful API 回傳結果通常為 JSON 格式

競賽題目一：實做雲端相機

實做雲端相機

- 實做能自動辨識物體的相機，可在拍照後的圖片疊上
  第一個 tag（文字）+ 第一個信心水準（confidence）
- 拆解功能：
  - ✓ 拍照 & 存檔
  - ✓ 將檔案上傳後取得 uid
  - ✓ 再將 uid 以參數方式送出查詢
  - ✓ 將文字疊在照片上，信心水準型態為 float 需轉型