

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

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

C

```
/* 印出 Hello World! 字串物件*/  
include <stdio.h>  
  
int main()  
{  
    printf("Hello, World!\n");  
    return 0;  
}
```

C++

```
//印出 Hello World! 字串物件  
#include <iostream>  
  
using namespace std;  
  
int main()  
{  
    cout << "Hello World" << endl;  
    return 0;  
}
```

Java

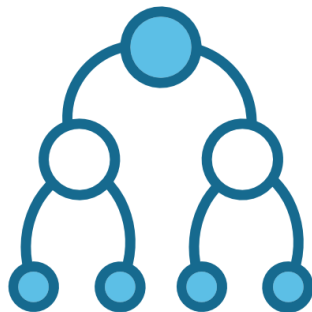
```
//印出 Hello World! 字串物件  
public class HelloWorld{  
  
    public static void main(String []args){  
        System.out.println("Hello World");  
    }  
}
```

程式語言：五大語法結構



Sequence

循序執行



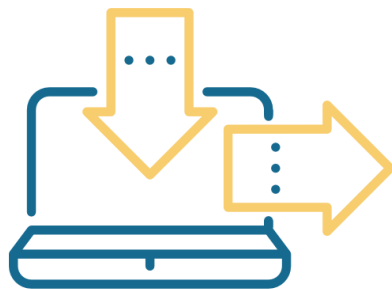
Conditional Statements

if 判斷式



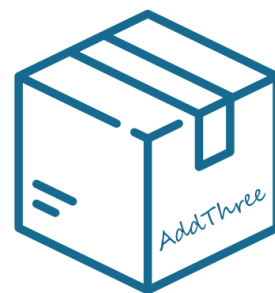
Loops

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

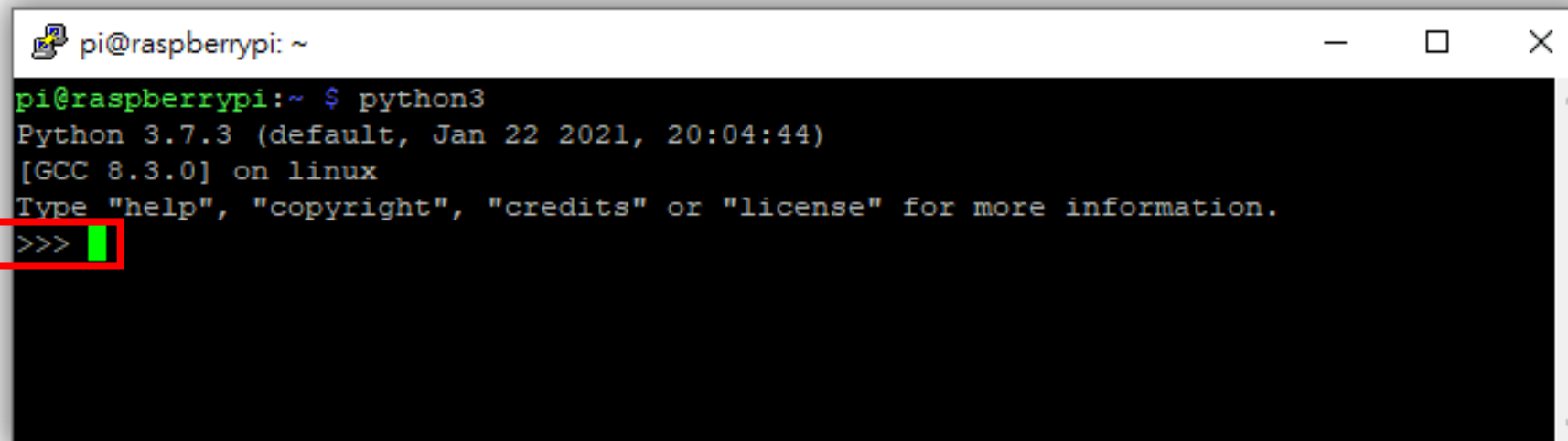
- 一般寫英文句子時，會以**名詞 + 動詞**。Python 是以**物件.方法**來描述。

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）

A terminal window titled "pi@raspberrypi: ~" showing the execution of the command "python3". The output displays the Python version "Python 3.7.3 (default, Jan 22 2021, 20:04:44)", the compiler "[GCC 8.3.0] on linux", and instructions to type "help", "copyright", "credits", or "license" for more information. The interactive prompt ">>>" is highlighted with a red box, and a green cursor is visible to its right.

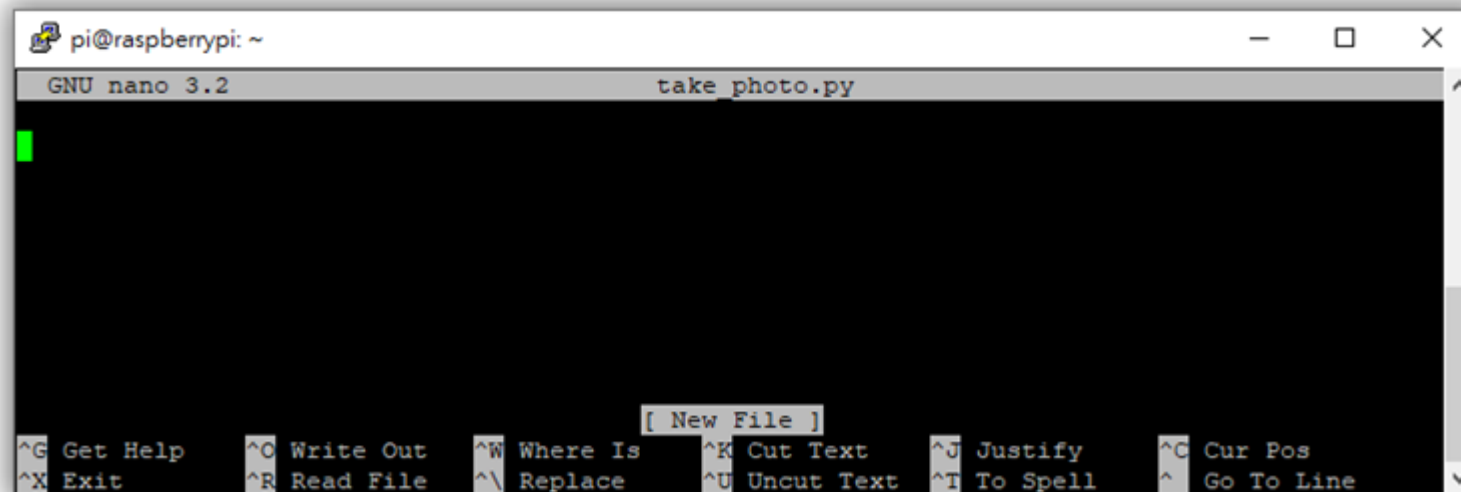
```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ python3  
Python 3.7.3 (default, Jan 22 2021, 20:04:44)  
[GCC 8.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

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）

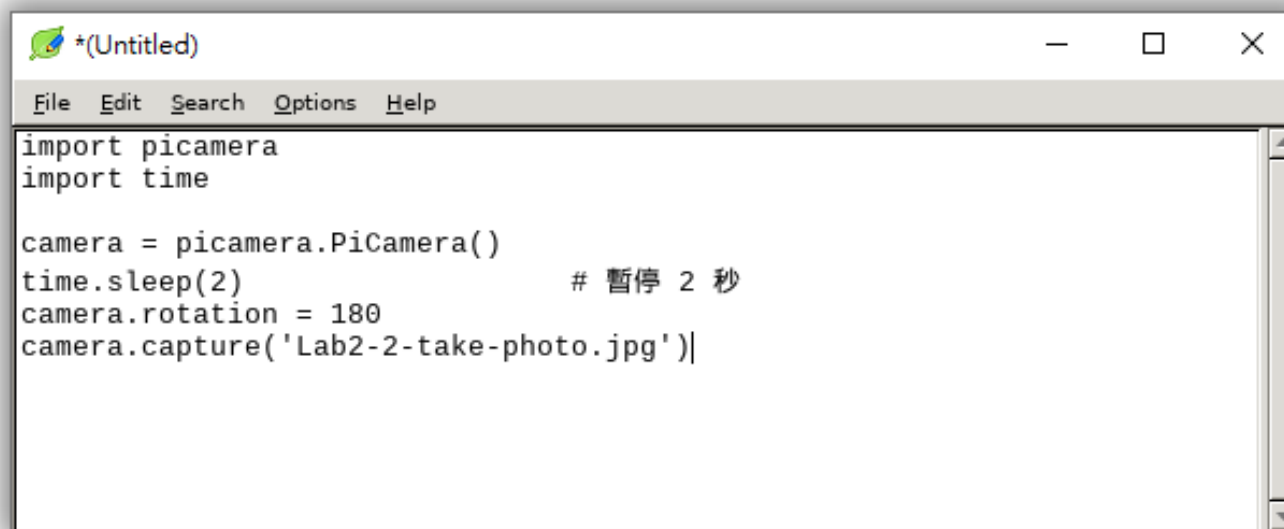
- 安裝 leafpad

```
$ sudo apt-get update
```

```
$ sudo apt-get install leafpad
```

- 在 X11 forwarding 連線成功下，執行 leafpad。

```
$ leafpad &
```



```
*(Untitled)
File Edit Search Options Help
import picamera
import time

camera = picamera.PiCamera()
time.sleep(2)           # 暫停 2 秒
camera.rotation = 180
camera.capture('Lab2-2-take-photo.jpg')
```

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

- 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()—— 使用字串物件 "Hello World!" 的  
upper() 方法，將字串轉成大寫  
'HELLO WORLD!'  
  
>>> "Hello World!".find('r')—— find() 方法尋找 'r' 的位置  
(從 0 開始)  
8  
  
>>> "Hello World!".replace('r', 'u')—— replace() 方法將所有的  
'r' 取代成 'u'  
'Hello Would!'
```

- 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

- 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

- 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』(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

- 『**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

- 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
```

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。

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/>

- 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
```


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')
```

Demo

Lab3-3-low_light.py

```
$ cd ~/exper3-1
```

```
$ nano Lab3-3-low_light.py
```

```
$ python3 Lab3-3-low_light.py
```

- 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()
```

Demo

Lab3-4-record_video.py

```
$ cd ~/exper3-1
```

```
$ nano Lab3-4-record_video.py
```

```
$ python3 Lab3-4-record_video.py
```

練習

- 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

```
pi@raspberrypi: ~/exper3-1
pi@raspberrypi:~/exper3-1 $ python3 Lab3-3-low_light.py
mmal: mmal_vc_port_enable: failed to enable port vc.null_sink:in:0(OPQV): ENOSPC
mmal: mmal_port_enable: failed to enable connected port (vc.null_sink:in:0(OPQV))0xe43ef0
mmal: mmal_connection_enable: output port couldn't be enabled
Traceback (most recent call last):
  File "Lab3-3-low_light.py", line 5, in <module>
    camera = picamera.PiCamera()
  File "/usr/lib/python3/dist-packages/picamera/camera.py", line 433, in __init__
    self._init_preview()
  File "/usr/lib/python3/dist-packages/picamera/camera.py", line 513, in _init_preview
    self, self.camera.outputs[self.CAMERA_PREVIEW_PORT])
  File "/usr/lib/python3/dist-packages/picamera/renderers.py", line 558, in __init__
    self.renderer.inputs[0].connect(source).enable()
  File "/usr/lib/python3/dist-packages/picamera/mmalobj.py", line 2212, in enable
    prefix="Failed to enable connection")
  File "/usr/lib/python3/dist-packages/picamera/exc.py", line 184, in mmal_check
    raise PiCameraMMALError(status, prefix)
picamera.exc.PiCameraMMALError: Failed to enable connection: Out of resources
pi@raspberrypi:~/exper3-1 $
```

```
pi@raspberrypi: ~/exper3-1
ps [options]

Try 'ps --help <simple|list|output|threads|misc|all>'
or 'ps --help <s|l|o|t|m|a>'
for additional help text.

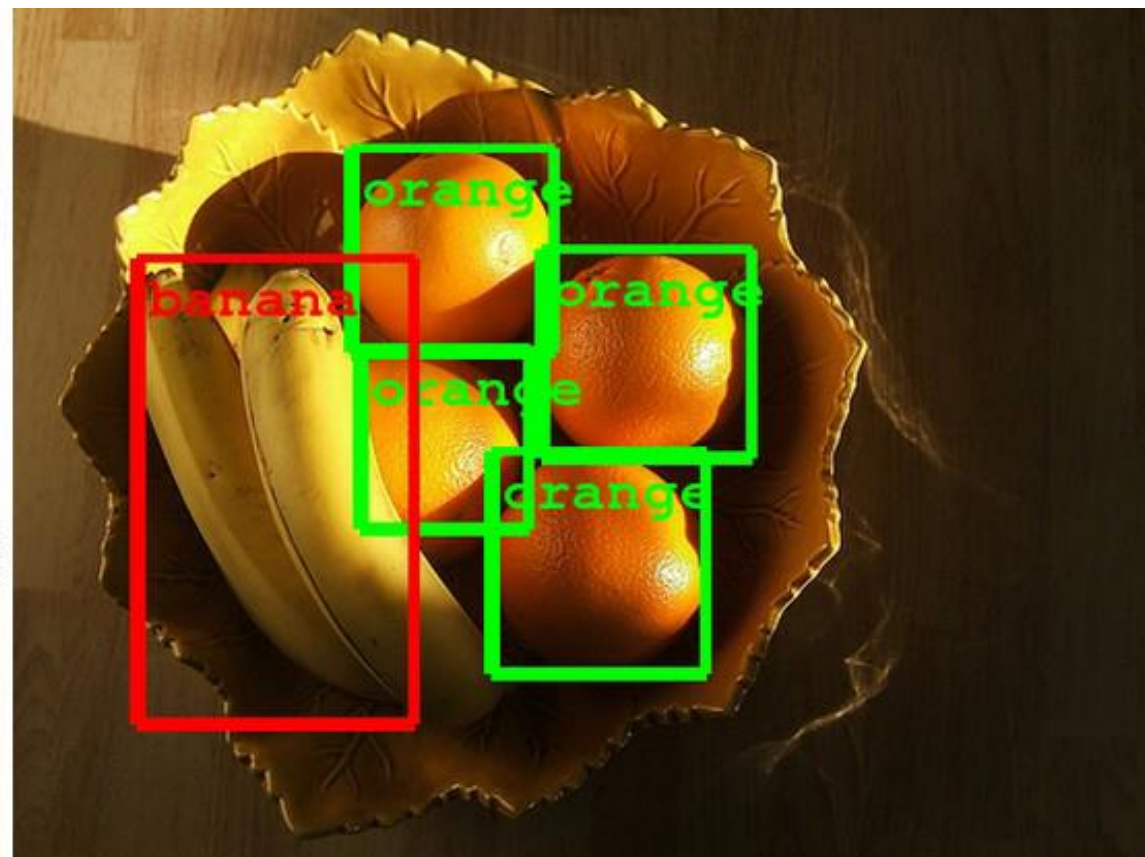
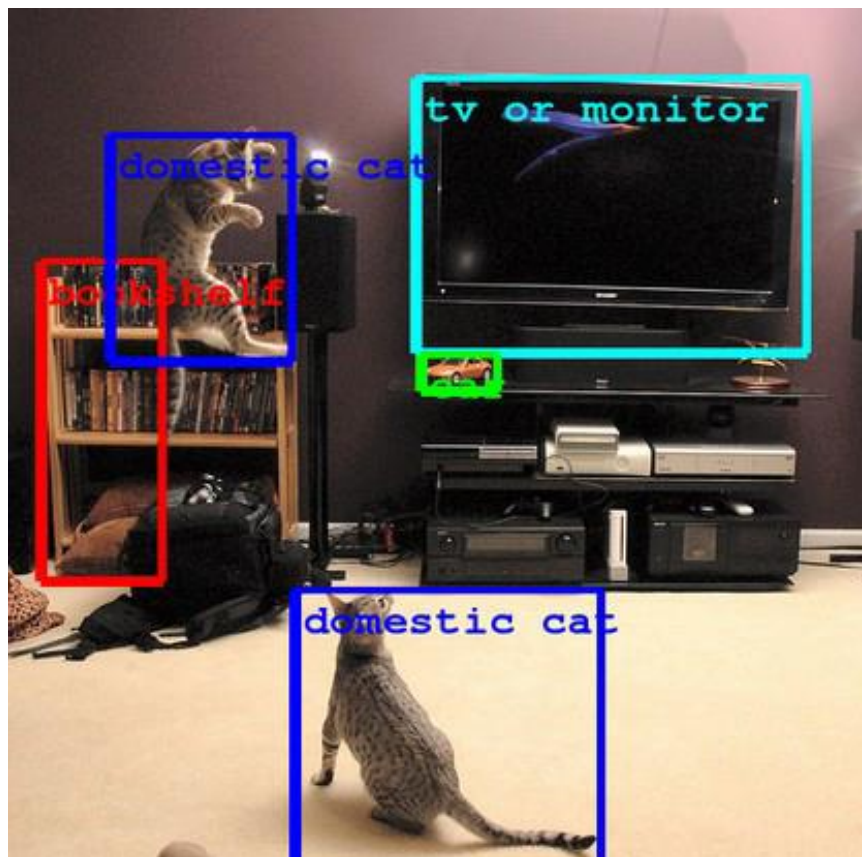
For more details see ps(1).
pi@raspberrypi:~/exper3-1 $ ps -u
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
D
pi         617  0.0  0.0   8492   3604 tty1    S+   14:05   0:00 -bash
pi        1042  0.0  0.0   8516   3668 pts/0    Ss+  14:06   0:00 bash
pi        1155  0.1  0.0   8488   3816 pts/1    Ss   14:09   0:00 -bash
pi        1460  0.0  0.0   8256   3204 pts/1    T    14:18   0:00 nano L
pi        1461  0.2  0.2  85712 11084 pts/1    Tl   14:18   0:00 python
pi        1509  0.0  0.0   9788   2456 pts/1    R+   14:21   0:00 ps -u
pi@raspberrypi:~/exper3-1 $
```

- 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)

實驗 3-2：會認東西的 Camera

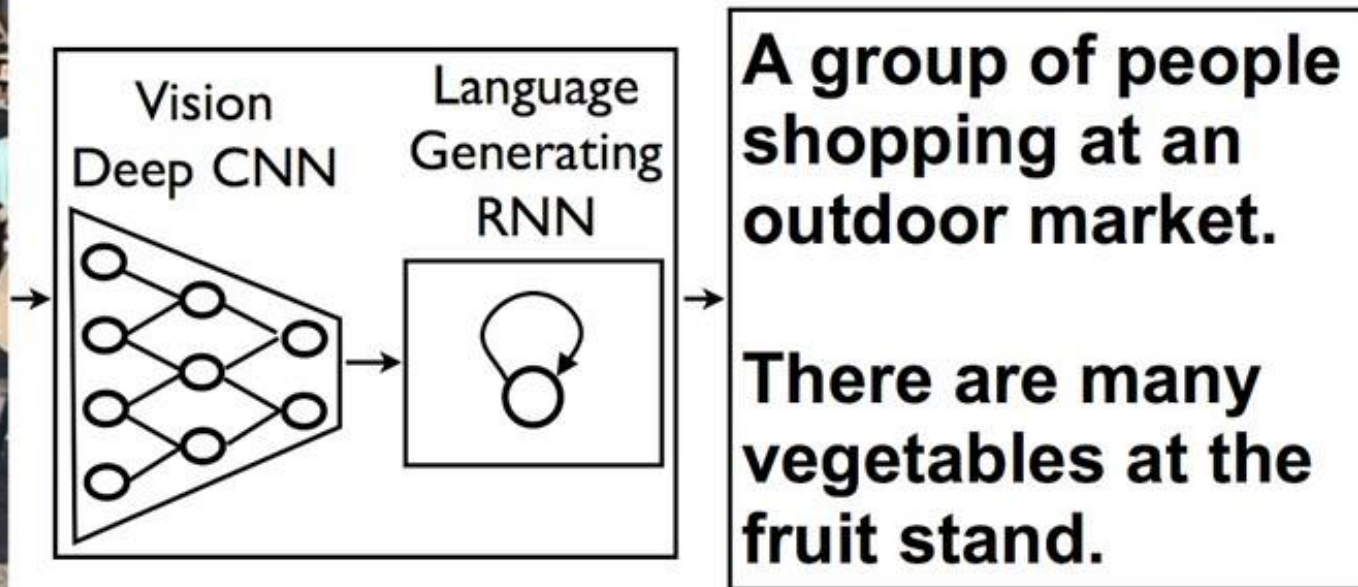
目的：串接網路服務

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

imagga SOLUTIONS CUSTOMER STORIES DEVELOPERS PRICING ABOUT SIGN IN

Build the next generation of **Image Recognition** Applications with **Imagga's API**.

Empowering intelligent apps with our **customizable** machine learning technology.

[Get a Free API Key](#)

100% **parrot**

IDC IDC named Imagga as one of the innovators for 2016 in Worldwide Image Analytics Market. [Learn More](#)

We are using cookies to deliver you better experience. By using our website you're agreeing to this. [Okay](#)

<https://imagga.com/>


看 DEMO (Auto-Tagging)

ex3-13

Auto-Tagging demo ← Switch to Categorization demo → Go to NSFW demo ← back to Imagga's homepage

Upload your photo

You can upload a photo or paste a URL of an image



Generated tags English

Concepts

banana	100.00%
edible fruit	71.28%
fruit	68.42%
plant	67.28%
produce	59.08%
food	45.49%
tropical	28.92%
yellow	27.79%
summer	25.05%
close	24.51%

[show me more tags](#)

Note: By uploading files here you agree to have them temporarily stored in our training dataset for the sole purpose of improving Imagga's technology.

1 **UPLOAD IMAGE** **上傳圖檔或貼上圖片網址**

Image URL: <https://images.theconversation.com/files/227374/original/file-20180712-27021-7iamol.jpg?ixlib=>







Tip: You can paste any image URL here and get tags. Include colors

Analyze

2

Try with example images

Select one of the following images to see the results:



If you need **training of custom tags or categories**, [contact us!](#)

Powered by APLebed [Copy to Clipboard](#)

See Our Full [API Documentation](#)

影像辨識結果

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

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>

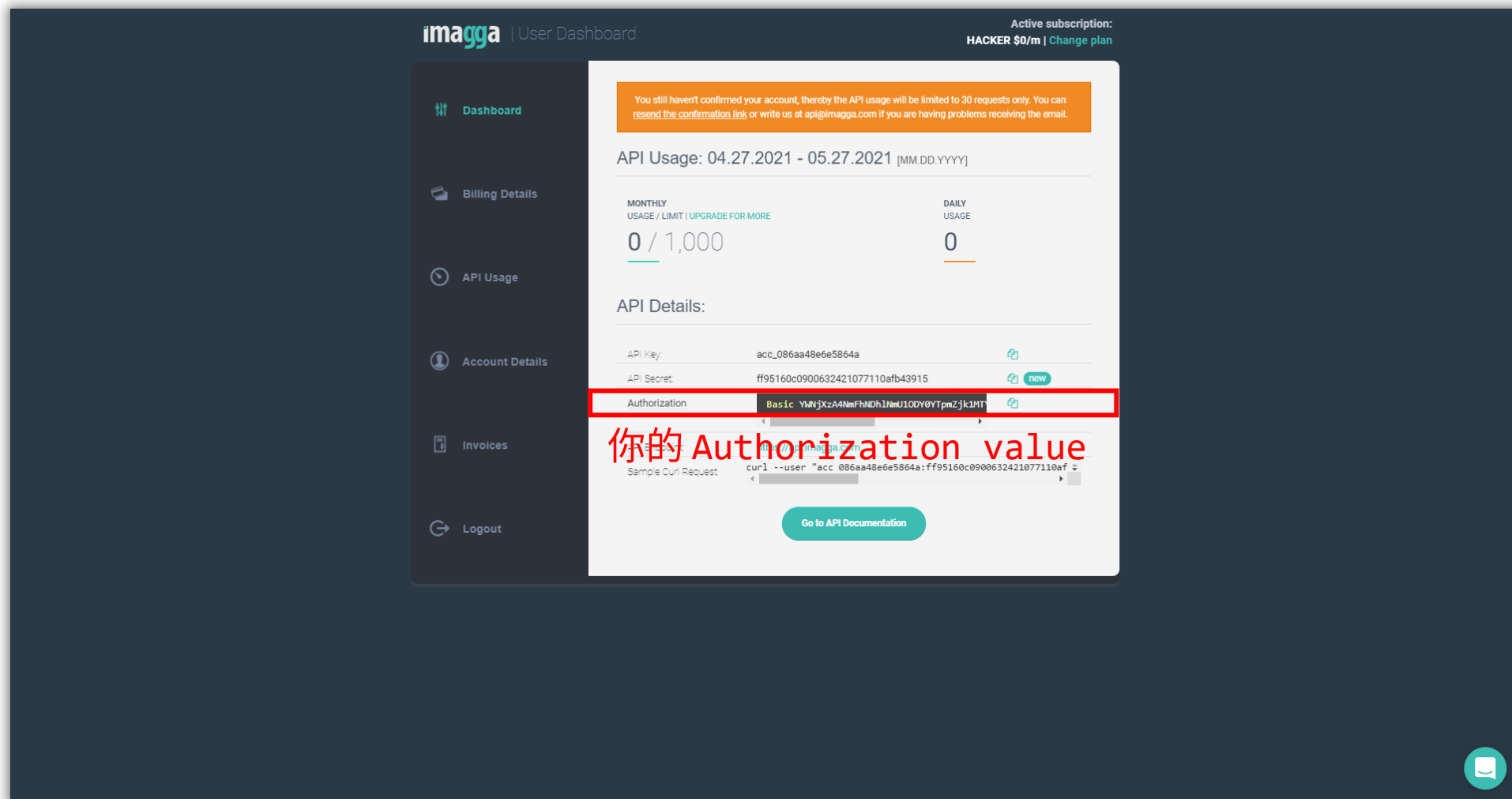
3. 串接

- ✓ 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，擷取必要的資訊。

<https://docs.imagga.com/>

Dashboard

- API 所需要的資料



The screenshot displays the 'imagga | User Dashboard' interface. At the top right, it indicates an 'Active subscription: HACKER \$0/m | Change plan'. A prominent orange warning box states: 'You still haven't confirmed your account, thereby the API usage will be limited to 30 requests only. You can resend the confirmation link or write us at api@imagga.com if you are having problems receiving the email.'

The main content area shows 'API Usage: 04.27.2021 - 05.27.2021 [MM.DD.YYYY]'. Below this, usage statistics are presented in two columns: 'MONTHLY USAGE / LIMIT | UPGRADE FOR MORE' showing '0 / 1,000' and 'DAILY USAGE' showing '0'. The 'API Details' section lists the following information:

API Key:	acc_086aa48e6e5864a	🔗
API Secret:	ff95160c0900632421077110afb43915	🔗 view
Authorization:	Basic YWVjXzA4NmFhNDh1NmU1ODY0YTpmZjk1MTI=	🔗

The 'Authorization' field is highlighted with a red box. Below the table, a red text overlay reads '你的 Authorization value'. A 'Sample Curl Request' is shown as: `curl --user "acc_086aa48e6e5864a:ff95160c0900632421077110af"`. A 'Go to API Documentation' button is located at the bottom of the details section.

<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

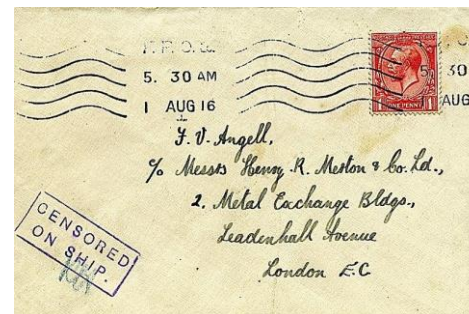
- ✓ **input**：程式指定圖片的 URL 或是上傳圖檔，進行影像辨識。
- ✓ **output**：從傳回的 JSON 字串，分析並擷取必要的資訊。

<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

Auto-Tagging demo ← Switch to Categorization demo → Go to NSFW demo ← back to Imagga's homepage

Note: By uploading files here you agree to have them temporarily stored in our training dataset for the sole purpose of improving Imagga's technology.

UPLOAD IMAGE

Image URL
https://images.theconversation.com/files/227374/original/file-20180712-27021-7iamol.jpg?ixlib=...
Tip: You can paste any image URL here and get tags. Include colors Analyze

banana	100.00%
edible fruit	71.28%
fruit	68.42%
plant	67.28%
produce	59.08%
food	45.49%
tropical	28.92%
yellow	27.79%
summer	25.05%
close	24.51%

→ show me more tags

Try with example images

Select one of the following images to see the results:

If you need training of custom tags or categories, [contact us!](#)

Are you a Lightroom user? Try out our auto-tagging plugin Wordroom - you can [download it here!](#)

Sign Up for Free

Implement in seconds

Shell Node.js Java Python PHP Ruby Objective-C Swift

```
import requests
url = "https://api.imagga.com/v2/tags"
querystring = {"image_url": "http://playground.imagga.com/static/img/exam..."}
headers = {
    'accept': "application/json",
    'authorization': "Basic YWNjXzJkYzdkNzNjMmYwOD1lMToxYzQ3YzgzZDg0Yjdm..."
}
```

複製程式碼

Powered by APIEmbed Copy to Clipboard

See Our Full [API Documentation](#)

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

```
import requests

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

querystring = {"image_url": "http://playground.imagga.com/static/img/example_photo.jpg", "version": "2"}

headers = {
    'accept': "application/json",
    'authorization': "Basic YWNjXzA4NmFhNDh1NmU1ODY0YTpmZjk1MTYwYzA5MDA2MzI0MjEwNzcxMTBhZmI0MzIxNQ=="
}

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

置換有效的圖片網址

置換有效的 Authorization value

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
{"result":{"tags":[{"confidence":100,"tag":{"en":"banana"}}, {"confidence":70.7407455444336,"tag":{"en":"edible fruit"}}, {"confidence":68.0829010009766,"tag":{"en":"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"}}, {"confidence":27.789644241333,"tag":{"en":"yellow"}}, {"confidence":25.0503311157227,"tag":{"en":"summer"}}, {"confidence":24.5122184753418,"tag":{"en":"close"}}, {"confidence":24.0958213806152,"tag":{"en":"leaf"}}, {"confidence":23.8083343505859,"tag":{"en":"flower"}}, {"confidence":23.4466819763184,"tag":{"en":"garden"}}, {"confidence":21.5454769134521,"tag":{"en":"fresh"}}, {"confidence":20.102876663208,"tag":{"en":"leaves"}}, {"confidence":19.2816123962402,"tag":{"en":"agriculture"}}, {"confidence":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":14.9636716842651,"tag":{"en":"palm"}}, {"confidence":14.894944190979,"tag":{"en":"botany"}}, {"confidence":14.4908647537231,"tag":{"en":"ripe"}}, {"confidence":14.2599868774414,"tag":{"en":"organic"}}, {"confidence":13.8874826431274,"tag":{"en":"color"}}, {"confidence":13.6671533584595,"tag":{"en":"branch"}}, {"confidence":13.4594030380249,"tag":{"en":"closeup"}}, {"confidence":13.4111337661743,"tag":{"en":"nutrition"}}, {"confidence":13.2497234344482,"tag":{"en":"botanical"}}, {"confidence":13.0924596786499,"tag":{"en":"bunch"}}, {"confidence":12.4703330993652,"tag":{"en":"raw"}}, {"confidence":12.103856086731,"tag":{"en":"sky"}}, {"confidence":12.1036653518677,"tag":{"en":"diet"}}, {"confidence":12.0697355270386,"tag":{"en":"stem"}}, {"c
```

JSON 資料

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)

Online JSON Viewer

2 按下 Format

Viewer Text

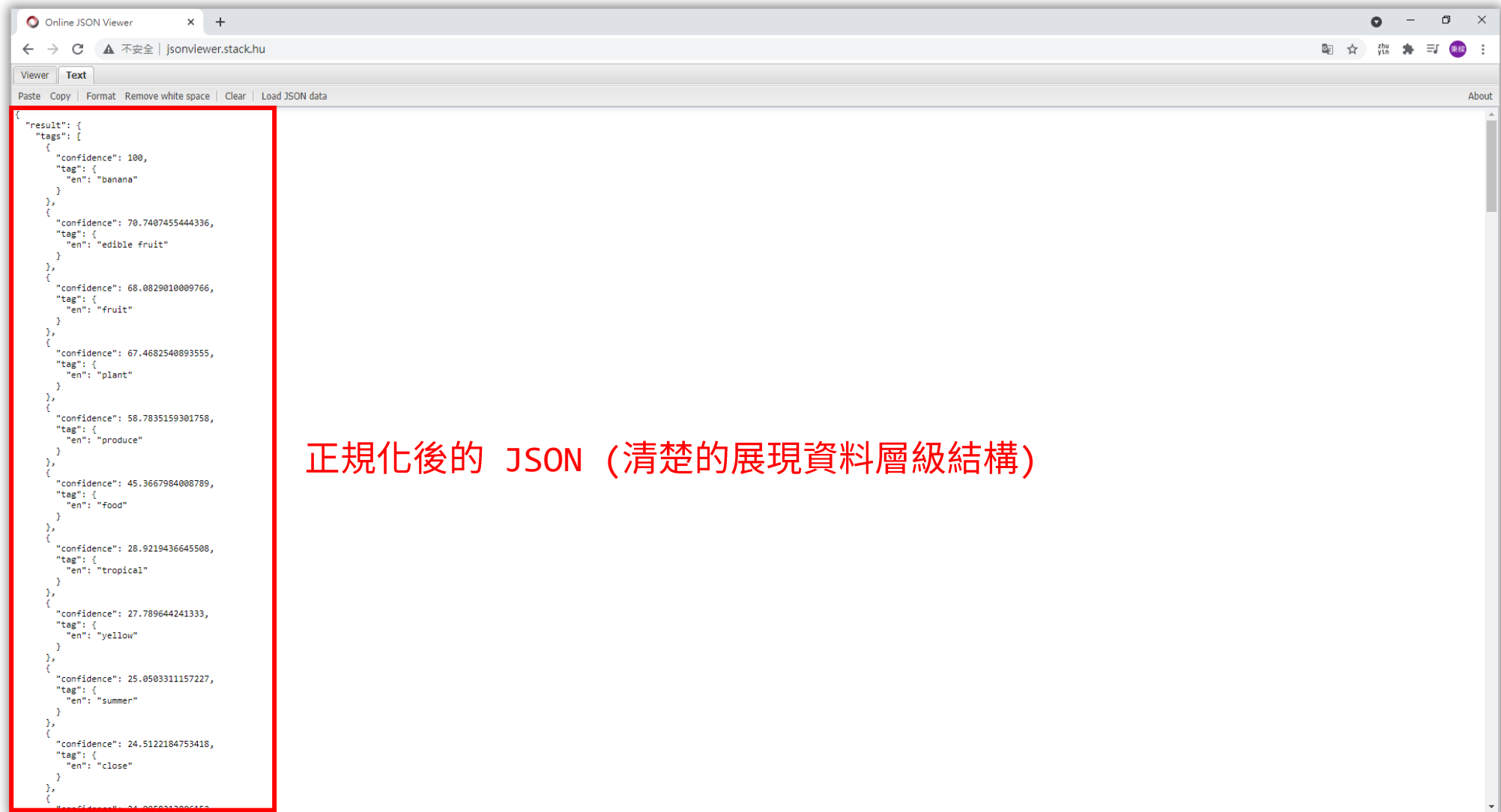
Paste Copy Format Remove white space Clear Load JSON data About

```
{
  "result": {
    "tags": [
      {
        "confidence": 100,
        "tag": {
          "en": "banana"
        }
      },
      {
        "confidence": 70.7407455444336,
        "tag": {
          "en": "edible fruit"
        }
      },
      {
        "confidence": 68.0829010009766,
        "tag": {
          "en": "fruit"
        }
      },
      {
        "confidence": 67.4682540893555,
        "tag": {
          "en": "plant"
        }
      },
      {
        "confidence": 58.7835159301758,
        "tag": {
          "en": "produce"
        }
      },
      {
        "confidence": 45.36679840808789,
        "tag": {
          "en": "food"
        }
      },
      {
        "confidence": 28.9219436645508,
        "tag": {
          "en": "tropical"
        }
      },
      {
        "confidence": 27.789644241333,
        "tag": {
          "en": "yellow"
        }
      },
      {
        "confidence": 25.050331157227,
        "tag": {
          "en": "summer"
        }
      },
      {
        "confidence": 24.5122184753418,
        "tag": {
          "en": "close"
        }
      },
      {
        "confidence": 24.0958213806152,
        "tag": {
          "en": "leaf"
        }
      },
      {
        "confidence": 23.8083343595859,
        "tag": {
          "en": "flower"
        }
      },
      {
        "confidence": 23.4466819763184,
        "tag": {
          "en": "garden"
        }
      },
      {
        "confidence": 21.5454769134521,
        "tag": {
          "en": "fresh"
        }
      },
      {
        "confidence": 20.102876663208,
        "tag": {
          "en": "leaves"
        }
      },
      {
        "confidence": 19.2816123962402,
        "tag": {
          "en": "agriculture"
        }
      },
      {
        "confidence": 19.0914440155829,
        "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": 15.2150267164917,
        "tag": {
          "en": "flora"
        }
      },
      {
        "confidence": 14.9636716842651,
        "tag": {
          "en": "palm"
        }
      },
      {
        "confidence": 14.89494190079,
        "tag": {
          "en": "botany"
        }
      },
      {
        "confidence": 14.4908647537231,
        "tag": {
          "en": "ripe"
        }
      },
      {
        "confidence": 14.2599868774414,
        "tag": {
          "en": "organic"
        }
      },
      {
        "confidence": 13.8874826431274,
        "tag": {
          "en": "color"
        }
      },
      {
        "confidence": 13.6671533584595,
        "tag": {
          "en": "branch"
        }
      },
      {
        "confidence": 13.45940380249,
        "tag": {
          "en": "closeup"
        }
      },
      {
        "confidence": 13.4111337661743,
        "tag": {
          "en": "nutrition"
        }
      },
      {
        "confidence": 13.2497234344482,
        "tag": {
          "en": "botanical"
        }
      },
      {
        "confidence": 13.0924596786499,
        "tag": {
          "en": "bunch"
        }
      },
      {
        "confidence": 12.4703330993652,
        "tag": {
          "en": "raw"
        }
      },
      {
        "confidence": 12.103856086731,
        "tag": {
          "en": "sky"
        }
      },
      {
        "confidence": 12.1036653518677,
        "tag": {
          "en": "diet"
        }
      },
      {
        "confidence": 12.0697355270386,
        "tag": {
          "en": "stem"
        }
      },
      {
        "confidence": 12.0184488296509,
        "tag": {
          "en": "petal"
        }
      },
      {
        "confidence": 11.756932258606,
        "tag": {
          "en": "spring"
        }
      },
      {
        "confidence": 11.7392158508301,
        "tag": {
          "en": "vegetable"
        }
      },
      {
        "confidence": 11.629864692688,
        "tag": {
          "en": "freshness"
        }
      },
      {
        "confidence": 11.6075057983398,
        "tag": {
          "en": "sunflower"
        }
      },
      {
        "confidence": 11.5850172042847,
        "tag": {
          "en": "farm"
        }
      },
      {
        "confidence": 11.4244832992554,
        "tag": {
          "en": "bright"
        }
      },
      {
        "confidence": 11.289831161499,
        "tag": {
          "en": "flowers"
        }
      },
      {
        "confidence": 11.1684703826904,
        "tag": {
          "en": "bloom"
        }
      },
      {
        "confidence": 11.0539417266846,
        "tag": {
          "en": "sweet"
        }
      },
      {
        "confidence": 10.7262706756592,
        "tag": {
          "en": "orange"
        }
      },
      {
        "confidence": 10.5627307891846,
        "tag": {
          "en": "rural"
        }
      },
      {
        "confidence": 10.1935625076294,
        "tag": {
          "en": "floral"
        }
      },
      {
        "confidence": 10.188437461853,
        "tag": {
          "en": "plants"
        }
      },
      {
        "confidence": 10.0201664434204,
        "tag": {
          "en": "colorful"
        }
      },
      {
        "confidence": 9.79357528686523,
        "tag": {
          "en": "pineapple"
        }
      },
      {
        "confidence": 9.69699859619141,
        "tag": {
          "en": "outdoors"
        }
      },
      {
        "confidence": 9.64854145050049,
        "tag": {
          "en": "sun"
        }
      },
      {
        "confidence": 9.34008979797363,
        "tag": {
          "en": "season"
        }
      },
      {
        "confidence": 9.1888408608887,
        "tag": {
          "en": "field"
        }
      },
      {
        "confidence": 9.14876174926758,
        "tag": {
          "en": "travel"
        }
      },
      {
        "confidence": 9.14805793762207,
        "tag": {
          "en": "island"
        }
      },
      {
        "confidence": 8.46874141693115,
        "tag": {
          "en": "juice"
        }
      },
      {
        "confidence": 8.44973087310791,
        "tag": {
          "en": "crop"
        }
      },
      {
        "confidence": 8.41209030151367,
        "tag": {
          "en": "seed"
        }
      },
      {
        "confidence": 8.30391780853271,
        "tag": {
          "en": "plantain"
        }
      },
      {
        "confidence": 8.32637596130371,
        "tag": {
          "en": "health"
        }
      },
      {
        "confidence": 8.05243827819624,
        "tag": {
          "en": "vegetarian"
        }
      },
      {
        "confidence": 7.86075581741333,
        "tag": {
          "en": "vibrant"
        }
      },
      {
        "confidence": 7.83874034081592,
        "tag": {
          "en": "day"
        }
      },
      {
        "confidence": 7.82537746429443,
        "tag": {
          "en": "forests"
        }
      },
      {
        "confidence": 7.66729122161865,
        "tag": {
          "en": "petals"
        }
      },
      {
        "confidence": 7.4281849861145,
        "tag": {
          "en": "landscape"
        }
      },
      {
        "confidence": 7.31941938400269,
        "tag": {
          "en": "object"
        }
      },
      {
        "confidence": 7.31344223022461,
        "tag": {
          "en": "vitamin"
        }
      },
      {
        "confidence": 7.26892566680908,
        "tag": {
          "en": "juicy"
        }
      },
      {
        "confidence": 7.24693441390991,
        "tag": {
          "en": "refreshment"
        }
      },
      {
        "confidence": 7.23603534698486,
        "tag": {
          "en": "detail"
        }
      },
      {
        "confidence": 7.1113805770874,
        "tag": {
          "en": "sunlight"
        }
      },
      {
        "confidence": 7.1069025993472,
        "tag": {
          "en": "grass"
        }
      },
      {
        "confidence": 7.0467605998203,
        "tag": {
          "en": "dessert"
        }
      },
      {
        "confidence": 7.0278787612915,
        "tag": {
          "en": "life"
        }
      }
    ]
  },
  "status": {
    "text": "",
    "type": "success"
  }
}
```

1 貼上雜亂的 JSON 文字

<http://jsonviewer.stack.hu/>

Online JSON Viewer (2/2)



```
{
  "result": {
    "tags": [
      {
        "confidence": 100,
        "tag": {
          "en": "banana"
        }
      },
      {
        "confidence": 70.7407455444336,
        "tag": {
          "en": "edible fruit"
        }
      },
      {
        "confidence": 68.0829010009766,
        "tag": {
          "en": "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"
        }
      },
      {
        "confidence": 27.789644241333,
        "tag": {
          "en": "yellow"
        }
      },
      {
        "confidence": 25.0503311157227,
        "tag": {
          "en": "summer"
        }
      },
      {
        "confidence": 24.5122184753418,
        "tag": {
          "en": "close"
        }
      }
    ]
  }
}
```

正規化後的 JSON (清楚的展現資料層級結構)

<http://jsonviewer.stack.hu/>

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
YWNjXzA4NmFhNDhINmU1ODY0YTpmZjk1MTYwYzA5MDA2MzI0MjEwNzcxMTBhZmI0MzIxNQ=="
}

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

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

data

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

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

`data["result"]`

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

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

`data["result"]["tags"]`

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

```
{  
  '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])
```

```
{
  '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"])
```



```
{
  '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"])
```

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
YWNjXzA4NmFhNDh1NmU1ODY0YTpmZjk1MTYwYzA5MDA2MzI0MjEwNzcxMTBhZmI0MzIxNQ=="
}

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 + " >>")
```

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