**Python程式設計：Pytesseract車牌辨識**

**目錄**

[1. 基本環境建置 1](#_Toc206949514)

[2. Tesseract OCR 與 pyTesseract OCR 1](#_Toc206949515)

[2.1 Tesseract OCR 1](#_Toc206949516)

[2.2 PyTesseract OCR 2](#_Toc206949517)

[3. 安裝 pytesseract 2](#_Toc206949518)

[4. 車牌辨識 2](#_Toc206949519)

[5. 辨識繁體中文 2](#_Toc206949520)

[6. 簡體中文辨識 3](#_Toc206949521)

[Reference 4](#_Toc206949522)

[7. 小專題：模擬停車場收費系統 5](#_Toc206949523)

[系統完整化 7](#_Toc206949524)

[程式碼模組化 (Functions) 7](#_Toc206949525)

[計算停車時間 7](#_Toc206949526)

[增加金額的設計進去 7](#_Toc206949527)

[提升辨識率：圖片預處理 (Image Preprocessing) 7](#_Toc206949528)

[資料持久化 (Data Persistence) 7](#_Toc206949529)

# 基本環境建置

這邊提供的程式碼與環境，以colab為主

[Python程式設計：Pytesseract 車牌辨識.ipynb - Colab](https://colab.research.google.com/drive/1LaTubSmHdfWh5NRgdlKaHk8V-K9zuyag?usp=sharing)

# 掛載雲端硬碟

from google.colab import drive

drive.mount**(**'/content/drive'**)**

# Tesseract OCR 與 pyTesseract OCR

Tesseract OCR 與 pyTesseract OCR 兩者都是進行光學字元辨識（OCR）的工具，但扮演的角色不同，可以想像成是「核心引擎」與「操作介面」的關係。

## Tesseract OCR

* 核心引擎：Tesseract 是一個強大的開源 OCR 引擎，它負責進行所有複雜的辨識工作，包括影像預處理、文字區域偵測、字元辨識等。它是由 C++ 語言所開發的，本身是一個獨立的程式或函式庫。
* 如何使用：你可以在你的作業系統上直接安裝 Tesseract 程式，然後透過\*\*命令列（Command-Line）\*\*來執行它，例如輸入 tesseract image.png output 就能將圖片中的文字辨識並輸出到 output.txt 檔案中。
* 獨立運作：Tesseract 可以在沒有其他程式語言（如 Python）的情況下獨立運作。

## PyTesseract OCR

* Python 介面：pyTesseract 是一個 Python 函式庫，它扮演著「包裝器（Wrapper）」的角色。它的功能是讓 Python 程式碼能夠呼叫並使用底層的 Tesseract 引擎。
* 如何使用：開發者在 Python 程式中引入 pytesseract 函式庫，然後使用其提供的函式（例如 pytesseract.image\_to\_string()）來傳送圖片給 Tesseract 引擎進行辨識，並接收回傳的文字結果。
* 相依性：pyTesseract 本身不具備辨識功能，它必須依賴於你系統中已經安裝好的 Tesseract 核心程式才能運作。

# 安裝 pytesseract

!pip install pytesseract

# 引入方式 import pytesseract

# 車牌辨識

from PIL import Image

import pytesseract

from IPython.display import display # 在colab展示

image = Image.open('/content/drive/MyDrive/python tutor/Pytesseract 車牌辨識素材/AJV-1688.jpg')

text = pytesseract.image\_to\_string(image)

display(image) # 使用 display 顯示圖片

print(text)

|  |  |
| --- | --- |
|  | AJV-1 688 |

# 辨識繁體中文

(效果不是很好…)

需要將 lang 參數設定為 'chi\_tra

# 安裝套件

!sudo apt-get install tesseract-ocr

!sudo apt-get install tesseract-ocr-chi-tra

from PIL import Image

import pytesseract

from IPython.display import display

##### 設定 Tesseract 執行路徑 #####

pytesseract.pytesseract.tesseract\_cmd = r'/usr/bin/tesseract' # 在 Colab 中，執行檔的預設路徑通常是 /usr/bin/tesseract

image\_path = '/content/drive/MyDrive/python tutor/Pytesseract 車牌辨識素材/繁體中文辨識.png'

image = Image.open(image\_path)

text = pytesseract.image\_to\_string(image, lang="chi\_tra")

# 步驟7：顯示結果

display(image)

print("--- 辨識結果 ---")

print(text)

|  |  |
| --- | --- |
|  |  |

# 簡體中文辨識

(效果不是很好…)

需要將 lang 參數設定為 'chi\_sim

# 安裝套件

!sudo apt-get install tesseract-ocr

!sudo apt-get install tesseract-ocr-chi-sim

from PIL import Image

import pytesseract

from IPython.display import display

##### 設定 Tesseract 執行路徑 #####

pytesseract.pytesseract.tesseract\_cmd = r'/usr/bin/tesseract' # 在 Colab 中，執行檔的預設路徑通常是 /usr/bin/tesseract

image\_path = '/content/drive/MyDrive/python tutor/Pytesseract 車牌辨識素材/簡體中文辨識.jpg'

image = Image.open(image\_path)

text = pytesseract.image\_to\_string(image, lang="chi\_sim")

# 步驟7：顯示結果

display(image)

print("--- 辨識結果 ---")

print(text)

|  |  |
| --- | --- |
|  |  |

# Reference

[Python OCR 安裝手冊 : 圖片轉文字 超簡單上手 | 光學字元辨識 x Tesseract | 不會 AI 但可以用 AI 【Gamma Ray 軟體工作室】 - YouTube](https://www.youtube.com/watch?v=fCEtYBLZ_LA)

[[ 實用心得 ] Tesseract-OCR. 因為工作上的關係，接觸到了 Tesseract 由 Google… | by KC凱稱 | Medium](https://kaichenlab.medium.com/%E5%AF%A6%E7%94%A8%E5%BF%83%E5%BE%97-tesseract-ocr-eef4fcd425f0)

[Tesseract – Google開源的光學文字辨識系統 – Claire's Blog](https://claire-chang.com/2024/06/24/tesseract-google%E9%96%8B%E6%BA%90%E7%9A%84%E5%85%89%E5%AD%B8%E6%96%87%E5%AD%97%E8%BE%A8%E8%AD%98%E7%B3%BB%E7%B5%B1/)

# 小專題：模擬停車場收費系統

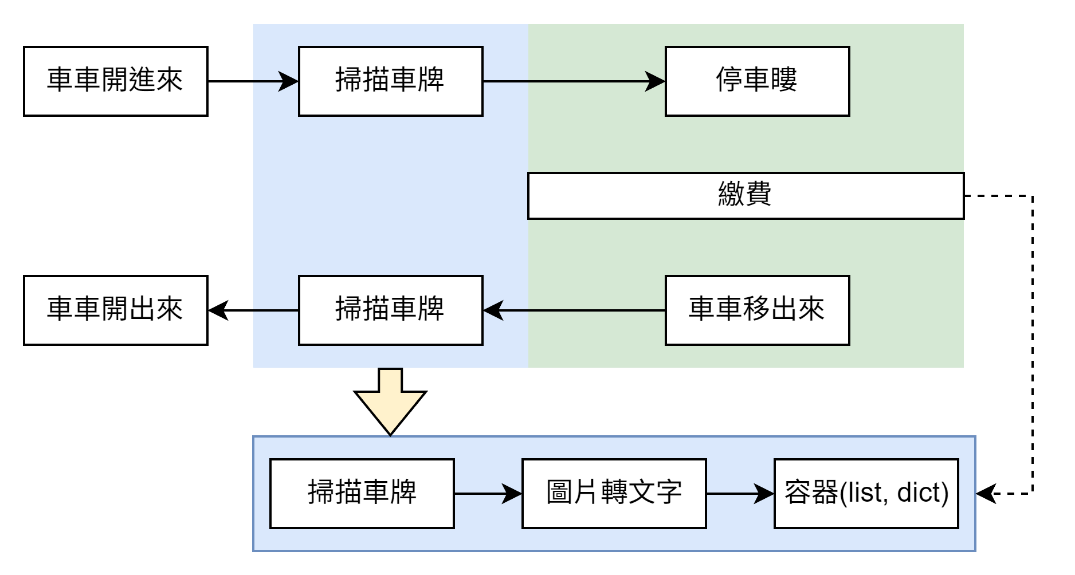


Figure ：停車系統流程圖

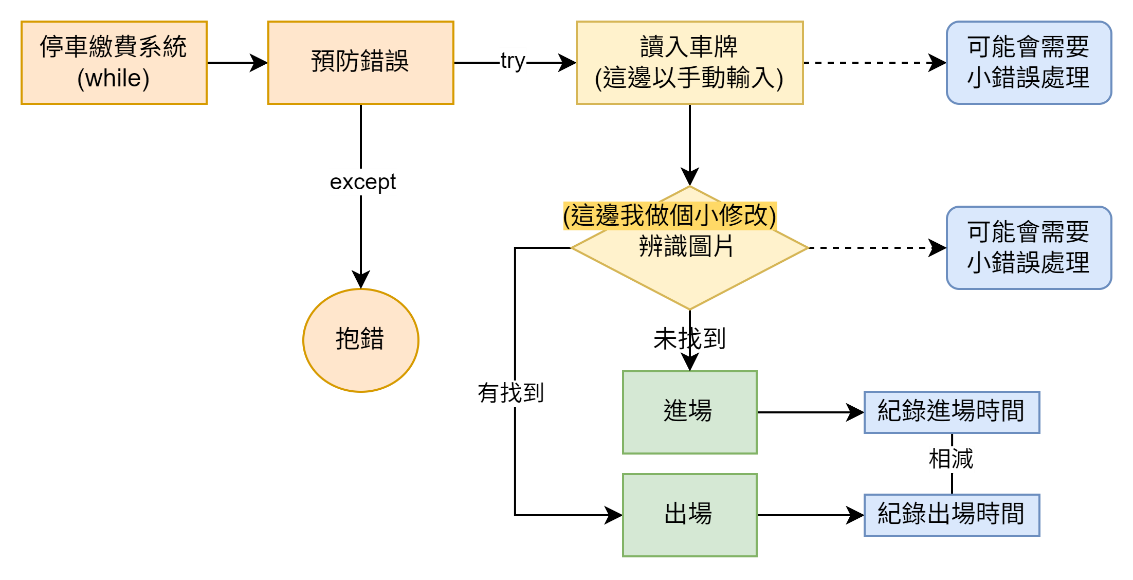


Figure ：收費系統邏輯

from PIL import Image

import pytesseract

import time

import re

PARKING\_LOT\_PATH = "/content/drive/MyDrive/python tutor/Pytesseract 車牌辨識素材/" # 所有的車牌圖片都會存放到這個資料夾

carDict = {} # 字典用於儲存車牌的進場時間

print("停車場系統已啟動...")

while True:

  try:

    carPlate = input("請輸入車牌名稱 (例如 AJV-1688) 或 Q 結束： ")

    if carPlate.upper() == 'Q': break

    ##### 輸入文字(車牌)處理 #####

    carPlate.strip() # 清除空格

    if not carPlate.lower().endswith('.jpg'): # 檢查輸入是否已經包含 .jpg 副檔名，如果沒有則手動補上

        carPlate += '.jpg'

    ##### 拼接完整圖片路徑 #####

    carPlatePath = PARKING\_LOT\_PATH + carPlate

    carText = pytesseract.image\_to\_string(Image.open(carPlatePath), config='--psm 8').strip()

    ##### 圖片辨識(車牌)處理 #####

    # STEP1 修正 pytesseract 辨識錯誤並統一格式

    carText = carText.replace('o', '0').replace('O', '0')

    carText = carText.replace('l', '1').replace('i', '1').replace('I', '1')

    carText = carText.replace('B', '8')

    # STEP2 只保留字母和數字

    carText = re.sub(r'[^A-Z0-9]', '', carText.upper())

    ##### 檢查車牌辨識結果是否為空 #####

    if not carText:

        print("無法辨識車牌，請檢查圖片。")

        continue

    print(*f*"辨識到的車牌號碼： {carText}")

    ##### 檢查 carDict 字典中是否已有此車牌 #####

    if carText in carDict: # 車牌已存在，視為出場

      print(*f*"車牌出場時間： {carText}")

      print(*f*"進場時間： {time.asctime(time.localtime(carDict[carText]))}")

      print(*f*"出場時間： {time.asctime(time.localtime())}")

      del carDict[carText]  # 移除已出場的車牌

    else: # 車牌不存在，視為進場

      entryTime = time.time()

      carDict[carText] = entryTime

      print(*f*"車牌進場時間： {carText}")

      # 使用 time.localtime() 將浮點數轉換為 struct\_time

      print(*f*"時間： {time.asctime(time.localtime(entryTime))}")

      print(*f*"車輛已進場。")

  except Exception as e:

    print(*f*"發生未知錯誤：{e}")

這邊就換你嘗試看看，可以盡你所有方式去完成

完成後，也可以跟我依樣，去整理一份word，好好描述這個小專題的內容、流程、你的思考方式、邏輯、遇到甚麼困難/問題 .etc，這就是一個很棒很棒的專題搂(實作跟表達，都是我們要去好好練習的部分)

## 系統完整化

可以看到，我上述的程式碼是有做過修改的，那是否可以做出更逼真的收費系統呈現

## 程式碼模組化 (Functions)

目前的程式碼都是包在主程式內，是否可以將重複或有特定功能的程式碼區塊封裝成函式，讓主程式的 while 迴圈更簡潔。例如：

* process\_plate\_image(image\_path): 接收圖片路徑，回傳辨識和清理後的車牌文字。
* display\_entry\_info(plate, entry\_time): 用於顯示車輛進場的資訊。
* display\_exit\_info(plate, entry\_time): 用於顯示車輛出場的資訊，並計算停車時間

## 計算停車時間

在車輛出場時，除了顯示進出場時間，還要計算並顯示總共停了多久（例如：xx 分 yy 秒）。

(提示：出場時間 time.time() 減去進場時間 carDict[carText] 會得到總秒數，需要將秒數轉換成更易讀的格式。)

## 增加金額的設計進去

## 提升辨識率：圖片預處理 (Image Preprocessing)

J個可以先放心底，之後有能力在完成即可。

直接辨識原始圖片的成功率不高。請在呼叫 pytesseract 之前，使用 OpenCV (cv2) 函式庫對車牌圖片進行預處理，以大幅提高辨識準確率。

建議步驟：

* 讀取圖片後，先將其轉換為灰階 (Grayscale)。
* 進行二值化 (Binarization) 處理，將圖片轉為只有純黑和純白的影像，讓文字輪廓更清晰。
* 將處理後的圖片物件（而不是檔案路徑）傳遞給 Pytesseract 進行辨識。

# 資料持久化 (Data Persistence)

J個可以先放心底，之後有能力在完成即可。

目前的 carDict 存在記憶體中，程式一關閉，所有在場車輛的紀錄都會消失。請修改程式，讓程式關閉時能將 carDict 的內容儲存到一個檔案（例如 parking\_log.json），並在程式啟動時自動載入該檔案，恢復先前的狀態。