**Google Vision API**

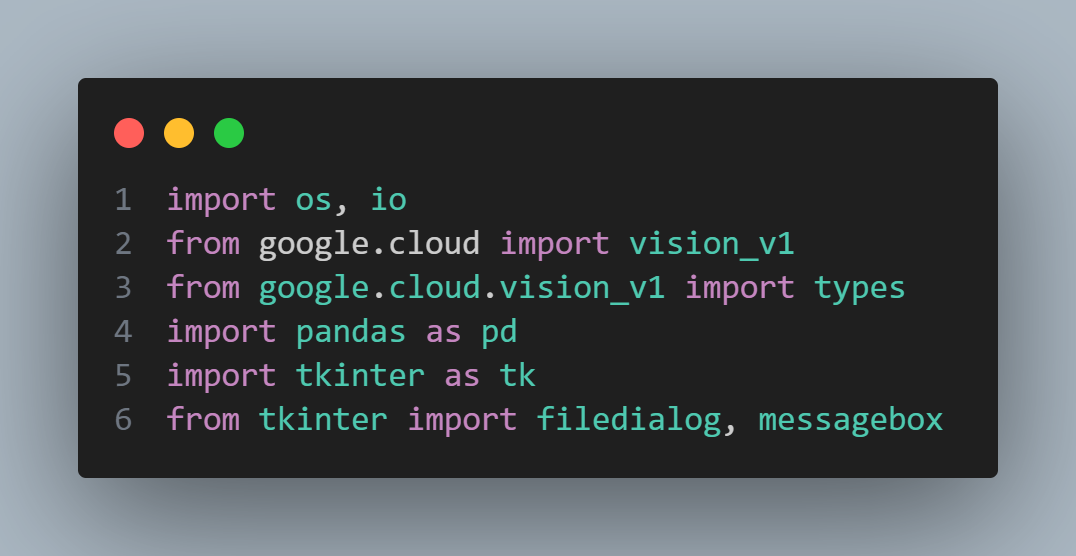
|  |  |
| --- | --- |
| Bước 1 | Truy cập Google Cloud để có thể lấy được API.  <https://console.cloud.google.com>    Lưu ý: Cần chọn vào Start Free( Bắt đầu dùng thử), thêm thông tin thanh toán. Thêm thẻ visa. Khi thành công hệ thống sẽ trừ tài khoản 1$ và sẽ hoàn lại ngay sau đó |
| Bước 2 | Nhấp chọn vào **Google Vision API** như hình bên dưới    Rồi nhấp chọn vào **New project**    Đặt tên cho project rồi nhấn **CREATE** |
| Bước 3 | Tại bảng APIs, nhấp chọn vào **Go to APIs overview**    Tiếp tục nhấp chọn **ENABLE APIS AND SERVICES**    Tìm kiếm **Vision** trên thanh tìm kiếm    Và chọn vào **Cloud Vision API** như hình bên dưới |
| Bước 3 | Sau khi API được bật, chọn **Credentials** như hình bên dưới    Tiếp tục nhấp chọn **CREATE CREDENTIAL**s để tạo mới thông tin xác thực    Tiếp theo chọn **Service account** như hình bên dưới    Đặt tên rồi tiếp tục nhấn **Create** và **Continue** rồi **Done** đến khi hoàn thành |
| Bước 4 | Nhấn vào tên service account vừa tạo    Chọn vào **KEY** và nhấp vào **ADD KEY**    Tiếp tục nhấp vào **Create new key**    Chọn JSON, 1 file JSON sẽ tự động được tải xuống (*Lưu ý: cần đặt file này ở thư mục của dự án)*.  Nên đặt lại tên file để dễ hiểu và sử dụng. |

**Cài đặt các gói cần thiết**

|  |  |
| --- | --- |
| Bước 1 | Tạo thư mục **GoogleVisionAPI** |
| Bước 2 | Chuyển file JSON đã tải về ở phần trước vào thư mục **GoogleVisionAPI** |
| Bước 3 | Đi vào trong thư mục **GoogleVisionAPI** |
| Bước 4 | Cài thư viện google-cloud-vision bằng câu lệnh: **pip install google-cloud-vision** |
| Bước 5 | Cài thư viện pandas bằng câu lệnh: **pip install pandas** |

**Giải thích chi tiết code**

1. **Import các thư viện**

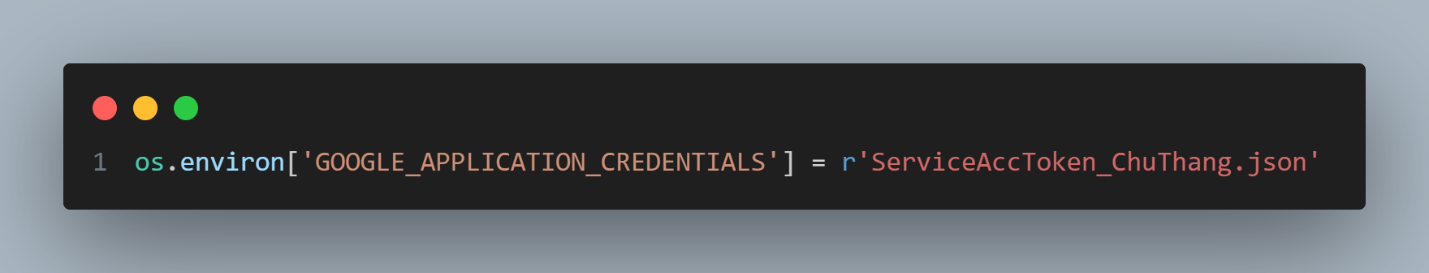


* **os, io**: Thư viện hỗ trợ quản lý hệ thống tệp tin và đọc/ghi dữ liệu dưới dạng luồng (stream).
* **google.cloud.vision\_v1**: Thư viện Google Cloud Vision API cho việc xử lý hình ảnh và nhận diện văn bản.
* **pandas as pd**: Thư viện để làm việc với dữ liệu dạng bảng.
* **tkinter as tk**: Thư viện để tạo giao diện đồ họa.
* **from tkinter import filedialog, messagebox**: Import các hàm từ thư viện tkinter cho hộp thoại chọn tệp và thông báo.

Chi tiết code tại đây:

|  |
| --- |
| import os, io  from google.cloud import vision\_v1  from google.cloud.vision\_v1 import types  import pandas as pd  import tkinter as tk  from tkinter import filedialog, messagebox |

1. **Đặt biến môi trường**



Câu lệnh trên được sử dụng để thiết lập biến môi trường **GOOGLE\_APPLICATION\_CREDENTIALS** trong môi trường thực thi của chương trình.

Giải thích chi tiết hơn:

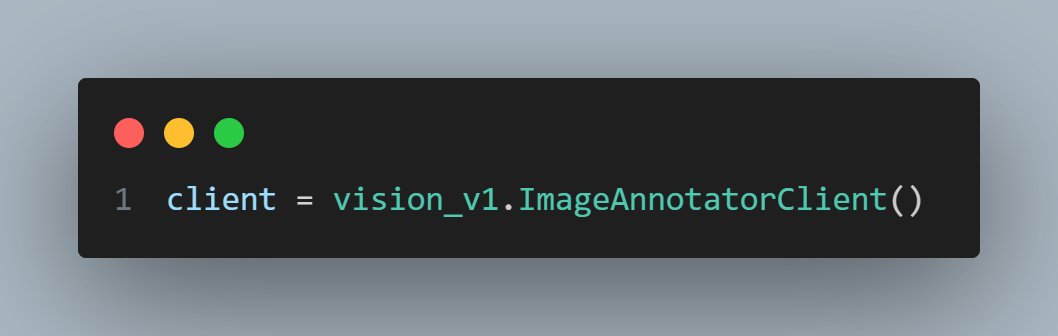
* **os.environ** là một từ điển chứa các biến của môi trường hệ thống
* **['GOOGLE\_APPLICATION\_CREDENTIALS']**: Đây là cách truy cập hoặc thiết lập giá trị của biến môi trường cụ thể trong từ điển **os.environ**
* **ServiceAccToken\_ChuThang.json** Đây là giá trị mà biến môi trường **['GOOGLE\_APPLICATION\_CREDENTIALS']** sẽ được thết lập thành. Trong trường hợp này, giá trị là một đường dẫn đến tệp JSON (tệp này được tải về tự động sau khi hoàn thành bước 4 của cách hướng dẫn dùng GGAPI ở phía trên)

Khi biến môi trường **['GOOGLE\_APPLICATION\_CREDENTIALS']** được thiết lập với giá trị của đường dẫn đến tệp JSON chứa thông tin xác thực, các thư viện và API của google cloud sẽ tự động sử dụng thông tin trong tệp JSON này để xác thực và cho phép ứng dụng truy cập vào các dịch vụ của google Cloud.

Chi tiết code tại dây:

|  |
| --- |
| os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = r'ServiceAccToken\_ChuThang.json' |

1. **Tạo đối tượng ImageAnnotatorClient**



Dòng mã này để tạo một đối tượng client, tương tác với dịch vụ Google Cloud Vision API

Giải thích chi tiết hơn:

* **client**: là tiên biến sử dụng để lưu trữ một đối tượng của lớp **ImageAnotatoClient** đã được import từ thư viện **google.cloud.vision\_v1**. Đối tượng nay sẽ là cầu nối giữa ứng dụng của mình và dịch vụ bên Google Vision API. Nó cung cấp các phương thức để gửi yêu cầu đến API và xử lý phản hồi từ API.

Chi tiết code tại đây:

|  |
| --- |
| client = vision\_v1.ImageAnnotatorClient() |

1. **Định nghĩa hàm detectText(FILE\_PATH)**



|  |  |
| --- | --- |
| Bước 1 | Mở và đọc tệp hình ảnh     * Sử dụng **io.open** để mở tệp hình ảnh được truyền vào (**FILE\_PATH**) ở chế độ đọc nhị phân (‘**rb’**) * Nội dung của tệp được vào biến **content**. Đây là dữ liệu nhị phân của hình ảnh. |
| Bước 2 | Tạo đối tượng hình ảnh     * Đối tượng **Image** được tạo bằng cách cung cấp nội dung của hình ảnh content cho thuộc tính content. |
| Bước 3 | Gửi yêu cầu nhận diện văn bản đến dịch vụ Vision API chứa thông tin về văn bản đã được nhận diện     * Hàm text\_detection của đối tượng client (ImageAnnotatorClient) được gọi để gửi yêu cầu nhận diện văn bản bằng cách sử dụng đối tượng hình ảnh đã được tạo. |
| Bước 4 | Trích xuất thông tin văn bản từ phản hồi     * Kết quả được phản hồi từ dịch vụ Vision API chứa thông tin về văn bản đã được nhận diện và sẽ được lưu vào biến **texts** |
| Bước 5 | Xử lý thông tin văn bản     * Tạo từ diển **data** để lưu trữ thông tin như ngôn ngữ (**locale**) và nội dung văn bản (**description**) của mỗi phần văn bản trong danh sách **texts** * Dùng vòng lặp for để duyệt qua danh sách **texts** và trích xuất thông tin cần thiết vào từ điển **data**. |
| Bước 6 | Tạo DataFrame từ thông tin văn bản     * Sau khi thu thập thông tin từ danh sách **texts**, dữ liệu được chuyển vào một DataFrame Pandas để dễ dàng xử lý và truy vấn |
| Bước 7 | Lấy nội dung từ văn bản     * Biến **texts** được gán bằng nội dung văn bản đầu tiên trong DataFrame, tức là phần văn bản đầu tiên được nhận diện từ hình ảnh |
| Bước 8 | Hiển thị kết quả     * Nội dung văn bản nhận diện được in ra màn hình console thông qua **print**. * Hộp thoại thông báo (**messagebox.showinfo**) được sử dụng để hiển thị nội dung văn bản nhận diện ra giao diện đồ họa với tiêu đề "Kết quả". |

Chi tiết code tại đây:

|  |
| --- |
| def detectText(FILE\_PATH):      with io.open(FILE\_PATH, 'rb') as image\_file:          content = image\_file.read()      image = vision\_v1.types.Image(content=content)      response = client.text\_detection(image=image)      texts = response.text\_annotations      data = {'locale': [], 'description': []}      for text in texts:          data['locale'].append(text.locale)          data['description'].append(text.description)      df = pd.DataFrame(data)      texts = df['description'][0]      print(texts)      messagebox.showinfo("Kết quả", texts) |

1. **Định nghĩa hàm open\_and\_detect()**

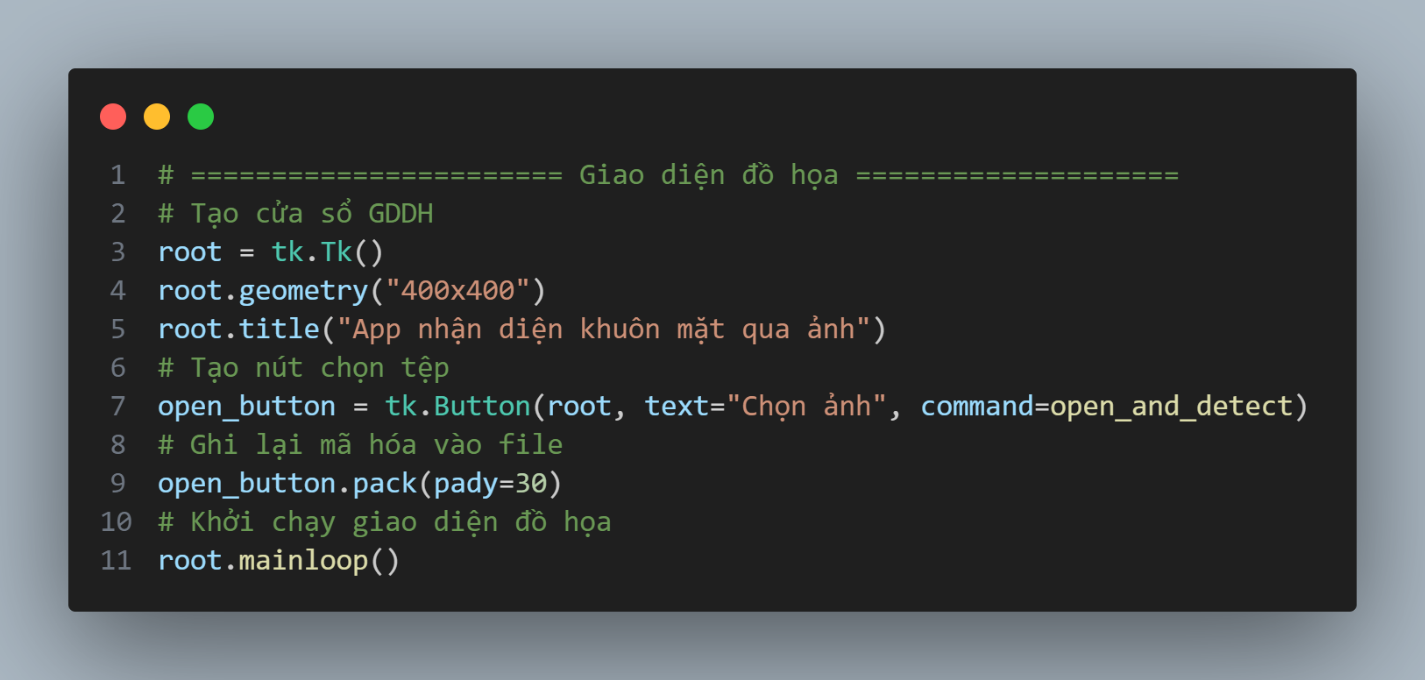


|  |  |
| --- | --- |
| Bước 1 | Câu lệnh này sẽ khiến hộp thoại file được bật lên và cho phép người dùng chọn file ảnh (Chỉ nhận các định dạng file .jpg .jpeg .png .gif .bmp) và ảnh được chọn sẽ có đường dẫn được lưu vào biến **file\_path** |
| Bước 2 | Nếu file ảnh được chọn đúng định dạng, đường dẫn sẽ được in ra trong terminal và sau đó chương trình sẽ gọi tới hàm **detectText(file\_path)** |

Chi tiết code tại đây:

|  |
| --- |
| def open\_and\_detect():      file\_path = filedialog.askopenfilename(filetypes=[("Image files", "\*.jpg \*.jpeg \*.png \*.gif \*.bmp")])      if file\_path:          print('Đường dẫn:', file\_path)          detectText(file\_path) |

1. **Tạo giao diện đồ họa với Tkinter**



Mã nguồn tạo một cửa sổ giao diện đồ họa sử dụng Tkinter và một nút "Chọn ảnh". Khi người dùng nhấn vào nút này, ứng dụng sẽ bắt đầu quá trình nhận diện văn bản từ hình ảnh. Chi tiết của mã:

**root = tk.Tk()**: Tạo cửa sổ giao diện đồ họa.

**root.geometry("400x400")**: Đặt kích thước cửa sổ là 400x400 pixel.

**root.title("App nhận diện khuôn mặt qua ảnh")**: Đặt tiêu đề cho cửa sổ.

**open\_button = tk.Button(root, text="Chọn ảnh", command=open\_and\_detect)**: Tạo nút "Chọn ảnh" và liên kết nó với hàm open\_and\_detect.

**open\_button.pack(pady=30)**: Đặt nút vào cửa sổ và đặt khoảng cách dưới nút là 30 pixel.

**root.mainloop()**: Khởi chạy giao diện và chờ sự tương tác từ người dùng.

Chi tiết code tại đây:

|  |
| --- |
| root = tk.Tk()  root.geometry("400x400")  root.title("App nhận diện khuôn mặt qua ảnh")  open\_button = tk.Button(root, text="Chọn ảnh", command=open\_and\_detect)  open\_button.pack(pady=30)  root.mainloop() |

**CODE ĐẦY ĐỦ**

|  |
| --- |
| import os, io  from google.cloud import vision\_v1  from google.cloud.vision\_v1 import types  import pandas as pd  import tkinter as tk  from tkinter import filedialog, messagebox  os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = r'ServiceAccToken\_ChuThang.json'  client = vision\_v1.ImageAnnotatorClient()  def detectText(FILE\_PATH):      with io.open(FILE\_PATH, 'rb') as image\_file:          content = image\_file.read()      image = vision\_v1.types.Image(content=content)      response = client.text\_detection(image=image)      texts = response.text\_annotations      data = {'locale': [], 'description': []}      for text in texts:          data['locale'].append(text.locale)          data['description'].append(text.description)      df = pd.DataFrame(data)      texts = df['description'][0]      print(texts)      messagebox.showinfo("Kết quả", texts)  # Hàm xử lý sự kiện người dùng nhấn nút "Chọn ảnh"  def open\_and\_detect():      file\_path = filedialog.askopenfilename(filetypes=[("Image files", "\*.jpg \*.jpeg \*.png \*.gif \*.bmp")])      if file\_path:          print('Đường dẫn:', file\_path)          detectText(file\_path)  # ======================= Giao diện đồ họa ====================  # Tạo cửa sổ GDDH  root = tk.Tk()  root.geometry("400x400")  root.title("App nhận diện chữ qua ảnh")  # Tạo nút chọn tệp  open\_button = tk.Button(root, text="Chọn ảnh", command=open\_and\_detect)  open\_button.pack(pady=30)  # Khởi chạy giao diện đồ họa  root.mainloop() |