```
# imports

import os
import requests
from dotenv import load_dotenv
from bs4 import BeautifulSoup
from IPython.display import Markdown, display
from openai import OpenAI

# If you get an error running this cell, then please head over to
the troubleshooting notebook!
```

- **import os:**

Thư viện os cung cấp các chức năng để làm việc với hệ điều hành. Ví dụ, bạn có thể sử dụng os để làm việc với các tệp tin, thư mục, và truy xuất các biến môi trường.

- **import requests:**

Thư viện requests được sử dụng để gửi HTTP requests đến các server và lấy dữ liệu từ các API hoặc trang web. Đây là công cụ quan trọng trong việc làm việc với web scraping hoặc tích hợp API từ các dịch vụ bên ngoài.

- **from dotenv import load_dotenv:**

dotenv được dùng để tải các biến môi trường từ file .env vào chương trình. Điều này giúp bảo mật các thông tin nhạy cảm như API keys, database credentials mà không phải cứng mã chúng trong mã nguồn.

- **from bs4 import BeautifulSoup:**

Thư viện BeautifulSoup trong bs4 được sử dụng để phân tích cú pháp HTML và XML. Đây là một công cụ mạnh mẽ để thực hiện web scraping và xử lý các trang web.

- **from IPython.display import Markdown, display:**

IPython.display cho phép bạn hiển thị các nội dung đặc biệt trong môi trường Jupyter notebook, như hiển thị văn bản dưới định dạng Markdown. Điều này hữu ích để tạo báo cáo hoặc hiển thị kết quả dễ đọc hơn.

- **from openai import OpenAI:**

Lưu ý: ở đây có thể sảy ra vấn đề lỗi vì câu lệnh này sẽ không hoạt động vì OpenAI không có lớp OpenAI. Đúng cú pháp là import openai. Thư viện openai cho phép bạn giao tiếp với API của OpenAI, như GPT-3 hoặc các mô hình AI khác để xử lý ngôn ngữ tự nhiên.

- **load_dotenv(override=True):**

Dòng này sử dụng thư viện dotenv để tải các biến môi trường từ file .env

```
# Kiển tra khóa trong API ở trong file .env

load_dotenv(override=True)
api_key = os.getenv('OPENAI_API_KEY')

# Check the key

if not api_key:
    print("No API key was found - please head over to the troubleshooting notebook in this folder to identify & fix!")
elif not api_key.startswith("sk-proj-"):
    print("An API key was found, but it doesn't start sk-proj-; please check you're using the right key - see troubleshooting notebook")
elif api_key.strip() != api_key:
    print("An API key was found, but it looks like it might have space or tab characters at the start or end - please remove them - see troubleshooting notebook")
else:
    print("API key found and looks good so far!")
```

- **api_key = os.getenv('OPENAI_API_KEY'):**

Dòng này lấy giá trị của biến môi trường OPENAI_API_KEY từ file .env và lưu trữ trong biến api_key. Nếu không tìm thấy khóa này, giá trị mặc định sẽ là None.

- **if not api_key:**
- **elif not api_key.startswith("sk-proj-"):**
- **elif api_key.strip() != api_key:**
- **else:**

Kiểm tra khóa có tồn tại hay không, khóa có đúng mẫu "sk-proj-" không và kiểm tra xem khóa có bị lỗi không VD: khoảng trắng trong khóa.
Nếu kiểm tra không có lỗi thì thống báo là khóa API good.

Kết quả của kiểm tra khóa:

"No API key was found - please head over to the troubleshooting notebook in this fold er to identify & fix!"

Do khóa API của ChatGpt cần phải trả phí nên mình không sử dụng khóa, mình định dùng **Ollama** làm mô hình cục bộ.

Các bạn có thể dùng khóa của:

- **Gemini**
- **Bard**

```
openai = OpenAI()
```

Hàm này dùng để khởi tạo OpenAI()

```
# To give you a preview -- calling OpenAI with these messages is
this easy. Any problems, head over to the Troubleshooting
notebook.

message = "Hello, GPT! This is my first ever message to you! Hi!"
response = openai.chat.completions.create(model="gpt-4o-mini",
messages=[{"role":"user", "content":message}])
print(response.choices[0].message.content)
```

Đoạn code này dùng để làm 1 **Example** về OpenAI

- **message = "Hello, GPT! This is my first ever message to you! Hi!"**

Đoạn code này chứa nội dung tin nhắn để gửi đến **OpenAI**

- **response = openai.chat.completions.create(model="gpt-4o-mini", messages=[{"role":"user", "content":message}])**

openai.chat.completions.create(...) là lệnh gửi yêu cầu đến OpenAI API.
model="gpt-4o-mini": Chỉ định mô hình AI GPT-4o-mini sẽ được sử dụng để xử lý tin nhắn.
messages=[{"role":"user", "content":message}]: Định nghĩa danh sách tin nhắn trong cuộc trò chuyện. Ở đây, chỉ có một tin nhắn từ người dùng (role: "user") gửi đến mô hình AI.

- **print(response.choices[0].message.content)**

Xuất nội dung mà OpenAI trả lời

- **headers = {**
- **"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36"**

```
headers = {
 "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0
Safari/537.36"
}

class Website:

    def __init__(self, url):
        """
        Create this Website object from the given url using the
BeautifulSoup library
        """
        self.url = url
        response = requests.get(url, headers=headers)
        soup = BeautifulSoup(response.content, 'html.parser')
        self.title = soup.title.string if soup.title else "No
title found"
        for irrelevant in soup.body(["script", "style", "img",
"input"]):
            irrelevant.decompose()
        self.text = soup.body.get_text(separator="\n", strip=True)
```

- **}**

Xác định headers giả lập trình duyệt:

- **class Website:**

Định nghĩa lớp Website:

- **def __init__(self, url):**

Khởi tạo đối tượng Website:

- **response = requests.get(url, headers=headers)**

Gửi yêu cầu HTTP để lấy nội dung trang web:

- **soup = BeautifulSoup(response.content, 'html.parser')**

Xử lý nội dung với BeautifulSoup:

- **self.title = soup.title.string if soup.title else "No title found"**

Trích xuất tiêu đề trang web:

- **for irrelevant in soup.body(["script", "style", "img", "input"]):**
- **irrelevant.decompose()**

Loại bỏ các phần tử <script>, <style>, <img>, <input> vì chúng không chứa nội dung văn bản hữu ích.

- **self.text = soup.body.get_text(separator="\n", strip=True)**

Lấy toàn bộ văn bản còn lại trong phần <body>, loại bỏ khoảng trắng không cần thiết.

```
ed = Website("https://edwarddonner.com")
print(ed.title)
print(ed.text)
```

- **ed = Website("https://edwarddonner.com")**
  Khởi tạo đối tượng Website với URL là: https://edwarddonner.com
- **print(ed.title)**

Hiển thị tiêu đề (<title>...</title>) của trang web, nếu có.

- **print(ed.text)**

In toàn bộ nội dung văn bản còn lại sau khi loại bỏ các phần tử không cần thiết (script, style, img, input).

```
system_prompt = "You are an assistant that analyzes the
contents of a website \
and provides a short summary, ignoring text that might be
navigation related. \
Respond in markdown."

def user_prompt_for(website):
    user_prompt = f"You are looking at a website titled
{website.title}"
    user_prompt += "\nThe contents of this website is as
follows; \
please provide a short summary of this website in markdown.
\
If it includes news or announcements, then summarize these
```

- **system_prompt**

Thiết lập prompt hệ thống (system_prompt)

- **user_prompt_for**

Hàm nhận một đối tượng Website làm tham số đầu vào.

- **user_prompt = f"You are looking at a website titled {website.title}"**

Bắt đầu prompt với tiêu đề của trang web (website.title).

- **user_prompt += "\nThe contents of this website is as follows; **
**please provide a short summary of this website in markdown. **
If it includes news or announcements, then summarize these too.\n\n"

Yêu cầu AI **tóm tắt nội dung trang web**, ưu tiên các tin tức hoặc thông báo quan trọng.

- **user_prompt += website.text**

Gắn phần nội dung đã trích xuất từ trang web (website.text) vào prompt.

- **return user_prompt**

Trả về user_prompt

- **print(user_prompt_for(ed))**

Chương trình sẽ in ra prompt mà bạn sẽ gửi đến mô hình AI.

def messages_for(website):
return [
{"role": "system", "content": system_prompt},
{"role": "user", "content": user_prompt_for(website)}
]

Dòng 1: Tin nhắn hệ thống (system): Thiết lập vai trò của AI bằng system_prompt.
Dòng 2: Tin nhắn người dùng (user): Gửi nội dung trang web cần tóm tắt.

messages_for(ed)

ed là một đối tượng Website, chứa tiêu đề và nội dung trang web "https://edwarddonner.com".

```
def summarize(url):
    website = Website(url)
    response = openai.chat.completions.create(
        model = "gpt-4o-mini",
        messages = messages_for(website)
    )
  return response.choices[0].message.content

def display_summary(url):
    summary = summarize(url)
  display(Markdown(summary))

display_summary("https://edwarddonner.com")
```

Xây dựng hàm summarize(url)

Tạo đối tượng Website(url), lấy tiêu đề và nội dung trang.
Gọi API OpenAI (gpt-4o-mini) với danh sách tin nhắn (messages_for(website)).
Trả về bản tóm tắt từ AI.

Xây dựng hàm display_summary(url):

Gọi summarize(url) để lấy bản tóm tắt.
Dùng display(Markdown(summary)) để hiển thị nội dung dưới dạng Markdown.

display_summary("https://edwarddonner.com")

Nếu trang có nội dung, hiển thị bản tóm tắt đẹp mắt bằng Markdown.
Nếu trang không tồn tại, thông báo lỗi.