```
import os
import requests
import json
from typing import List
from dotenv import load_dotenv
from bs4 import BeautifulSoup
from IPython.display import Markdown, display, update_display
```

Dòng lệnh	Chức năng
import os	Làm việc với biến môi trường.
import requests	Gửi yêu cầu HTTP (API, Web scraping).
import json	Xử lý dữ liệu JSON.
from typing import List	Xác định kiểu danh sách.
from dotenv import load_dotenv	Đọc biến môi trường từ .env.
from bs4 import BeautifulSoup	Phân tích HTML/XML (web scraping).
from IPython.display import Markdown, display, update_display	Hiển thị Markdown trong Jupyter Notebook.
from openai import OpenAI	Gọi API OpenAI để sử dụng GPT.

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

if api_key and api_key.startswith('sk-proj-') and len(api_key)>10:
    print("API key looks good so far")
else:
    print("There might be a problem with your API key? Please visit the
troubleshooting notebook!")

MODEL = 'gpt-4o-mini'
```

Dòng lệnh	Chức năng
load_dotenv(override=True)	Nạp biến môi trường từ file .env.
api_key = os.getenv('OPENAI_API_KEY')	Lấy API Key từ hệ thống.
if api_key and api_key.startswith('sk-proj-') and len(api_key) > 10:	Kiểm tra API Key hợp lệ.
print("API key looks good so far")	Xác nhận API Key hợp lệ.
print("There might be a problem with your API key?...")	Cảnh báo nếu API Key sai.
MODEL = 'gpt-4o-mini'	Xác định mô hình GPT sử dụng.
openai = OpenAI()	Khởi tạo SDK OpenAI.

```
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:
    """
    A utility class to represent a Website that we have scraped, now with links
    """

    def __init__(self, url):
        self.url = url
        response = requests.get(url, headers=headers)
        self.body = response.content
        soup = BeautifulSoup(self.body, 'html.parser')
        self.title = soup.title.string if soup.title else "No title found"
        if soup.body:
            for irrelevant in soup.body(["script", "style", "img", "input"]):
                irrelevant.decompose()
            self.text = soup.body.get_text(separator="\n", strip=True)
        else:
            self.text = ""
        links = [link.get('href') for link in soup.find_all('a')]
        self.links = [link for link in links if link]
```

Dòng lệnh	Chức năng
headers = { "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) ..." }	Định nghĩa User-Agent để giả lập trình duyệt, tránh bị chặn khi gửi request.
class Website:	Định nghĩa lớp Website để trích xuất thông tin từ trang web.
def __init__(self, url):	Hàm khởi tạo, nhận url của trang web cần lấy dữ liệu.
self.url = url	Lưu url vào thuộc tính của đối tượng Website.
response = requests.get(url, headers=headers)	Gửi request GET đến url với User-Agent.
self.body = response.content	Lưu nội dung HTML của trang vào self.body.
soup = BeautifulSoup(self.body, 'html.parser')	Chuyển nội dung HTML thành đối tượng BeautifulSoup để xử lý dễ dàng hơn.
self.title = soup.title.string if soup.title else "No title found"	Trích xuất tiêu đề <title> của trang, nếu không có thì trả về "No title found".
if soup.body:	Kiểm tra xem trang có thẻ <body> không.
for irrelevant in soup.body(["script", "style", "img", "input"]): irrelevant.decompose()	Loại bỏ các thẻ không cần thiết (script, style, img, input) để giữ lại nội dung chính.
self.text = soup.body.get_text(separator="\n", strip=True)	Lấy văn bản từ <body>, loại bỏ khoảng trắng và xuống dòng hợp lý.
else: self.text = ""	Nếu không có <body>, đặt self.text là chuỗi rỗng.
links = [link.get('href') for link in soup.find_all('a')]	Lấy tất cả liên kết (href) từ thẻ <a>.
self.links = [link for link in links if link]	Lọc bỏ các liên kết rỗng hoặc None.
def get_contents(self):	Hàm trả về nội dung của trang web dưới dạng chuỗi.
return f"Webpage Title:\n{self.title}\nWebpage Contents:\n{self.text}\n\n"	Trả về tiêu đề và nội dung văn bản của trang web.

```
ed = Website("https://edwarddonner.com")
ed.links
```

Dòng lệnh	Chức năng
ed = Website("https://edwarddonner.com")	Khởi tạo một đối tượng Website với URL chỉ định.
ed.links	Lấy danh sách các liên kết (links) từ trang web.

Kết quả:

```
['https://edwarddonner.com/',
 'https://edwarddonner.com/connect-four/',
 'https://edwarddonner.com/outsmart/',
 'https://edwarddonner.com/about-me-and-about-nebula/',
 'https://edwarddonner.com/posts/',
 'https://edwarddonner.com/',
 'https://news.ycombinator.com',
 'https://nebula.io/?utm_source=ed&utm_medium=referral',
 'https://www.prnewswire.com/news-releases/wynden-stark-group-acquires-nyc-venture-backed-tech-startup-untapt-301269512.html',
 'https://patents.google.com/patent/US20210049536A1/',
 'https://www.linkedin.com/in/eddonner/',
 'https://edwarddonner.com/2025/01/23/llm-workshop-hands-on-with-agents-resources/',
 'https://edwarddonner.com/2025/01/23/llm-workshop-hands-on-with-agents-resources/',
 'https://edwarddonner.com/2024/12/21/llm-resources-superdatascience/',
 'https://edwarddonner.com/2024/12/21/llm-resources-superdatascience/',
 'https://edwarddonner.com/2024/11/13/llm-engineering-resources/',
 'https://edwarddonner.com/2024/11/13/llm-engineering-resources/',
 'https://edwarddonner.com/2024/10/16/from-software-engineer-to-ai-data-scientist-resources/',
 'https://edwarddonner.com/2024/10/16/from-software-engineer-to-ai-data-scientist-resources/',
 'https://edwarddonner.com/',
 'https://edwarddonner.com/connect-four/',
 'https://edwarddonner.com/outsmart/',
 'https://edwarddonner.com/about-me-and-about-nebula/',
 'https://edwarddonner.com/posts/',
 'mailto:hello@mygroovydomain.com',
 'https://www.linkedin.com/in/eddonner/',
 'https://twitter.com/edwarddonner',
 'https://www.facebook.com/edward.donner.52']
```

```
link_system_prompt = "You are provided with a list of links found on a webpage. \
You are able to decide which of the links would be most relevant to include in a brochure about the company, \
such as links to an About page, or a Company page, or Careers/Jobs pages.\n"
link_system_prompt += "You should respond in JSON as in this example:"
link_system_prompt += """
{
   "links": [
      {"type": "about page", "url": "https://full.url/goes/here/about"},
      {"type": "careers page": "url": "https://another.full.url/careers"}
   ]
}
```

Dòng lệnh	Chức năng
ed = Website("https://edwarddonner.com")	Khởi tạo một đối tượng `Website` với URL chỉ định.
ed.links	Lấy danh sách các liên kết (`links`) từ trang web.
link_system_prompt = "You are provided with a list of links found on a webpage. ..."	Tạo một chuỗi prompt mô tả cách chọn lọc các liên kết phù hợp.
link_system_prompt += """{ "links": [{...}] }"""	Thêm định dạng JSON mẫu vào prompt.

```
print(link_system_prompt)
```

Dòng lệnh	Chức năng
print(link_system_prompt)	Xuất ra dữ liệu của biến

	link_system_prompt

Kết Quả:

```
You are provided with a list of links found on a webpage. You are able to decide which of the links would be most relevant to include in a brochure abo
ut the company, such as links to an About page, or a Company page, or Careers/Jobs pages.
You should respond in JSON as in this example:
{
    "links": [
        {"type": "about page", "url": "https://full.url/goes/here/about"},
        {"type": "careers page": "url": "https://another.full.url/careers"}
    ]
}
```

```
def get_links_user_prompt(website):
    user_prompt = f"Here is the list of links on the website of {website.url} - "
    user_prompt += "please decide which of these are relevant web links for a
brochure about the company, respond with the full https URL in JSON format. \
Do not include Terms of Service, Privacy, email links.\n"
    user_prompt += "Links (some might be relative links):\n"
    user_prompt += "\n".join(website.links)
    return user_prompt
```

Dòng lệnh	Chức năng
def get_links_user_prompt(website):	Định nghĩa hàm `get_links_user_prompt` nhận vào một đối tượng `website`.
user_prompt = f"Here is the list of links..."	Tạo chuỗi nhắc (`user_prompt`) mô tả danh sách liên kết trên trang web của `website.url`.
user_prompt += "please decide which..."	Hướng dẫn người dùng chọn các liên kết phù hợp để sử dụng trong brochure công ty, trả về định dạng JSON.
user_prompt += "Links (some might be..."	Thêm tiêu đề danh sách liên kết vào chuỗi `user_prompt`.
user_prompt += "\n".join(website.links)	Thêm danh sách các liên kết từ `website.links` vào `user_prompt`, mỗi liên kết trên một dòng.
return user_prompt	Trả về `user_prompt` đã tạo.

```
print(get_links_user_prompt(ed))
```

Dòng lệnh	Chức năng
print(get_links_user_prompt(ed))	Hàm lấy danh sách các liên kết từ đối tượng ed và có thể định dạng chúng dưới dạng prompt.

Kết quả:

```
Here is the list of links on the website of https://edwarddonner.com - please decide which of these are relevant web links for a brochur
pany, respond with the full https URL in JSON format. Do not include Terms of Service, Privacy, email links.
Links (some might be relative links):
https://edwarddonner.com/
https://edwarddonner.com/connect-four/
https://edwarddonner.com/outsmart/
https://edwarddonner.com/about-me-and-about-nebula/
https://edwarddonner.com/posts/
https://edwarddonner.com/
https://news.ycombinator.com
https://nebula.io/?utm_source=ed&utm_medium=referral
https://www.prnewswire.com/news-releases/wynden-stark-group-acquires-nyc-venture-backed-tech-startup-untapt-301269512.html
https://patents.google.com/patent/US20210049536A1/
https://www.linkedin.com/in/eddonner/
https://edwarddonner.com/2025/01/23/llm-workshop-hands-on-with-agents-resources/
https://edwarddonner.com/2025/01/23/llm-workshop-hands-on-with-agents-resources/
https://edwarddonner.com/2024/12/21/llm-resources-superdatascience/
https://edwarddonner.com/2024/12/21/llm-resources-superdatascience/
https://edwarddonner.com/2024/11/13/llm-engineering-resources/
https://edwarddonner.com/2024/11/13/llm-engineering-resources/
https://edwarddonner.com/2024/10/16/from-software-engineer-to-ai-data-scientist-resources/
https://edwarddonner.com/2024/10/16/from-software-engineer-to-ai-data-scientist-resources/
https://edwarddonner.com/
https://edwarddonner.com/connect-four/
https://edwarddonner.com/outsmart/
https://edwarddonner.com/about-me-and-about-nebula/
https://edwarddonner.com/posts/
mailto:hello@mygroovydomain.com
https://www.linkedin.com/in/eddonner/
https://twitter.com/edwarddonner
```

```
def get_links(url):
    website = Website(url)
    response = openai.chat.completions.create(
        model=MODEL,
        messages=[
            {"role": "system", "content": link_system_prompt},
            {"role": "user", "content": get_links_user_prompt(website)}
      ],
        response_format={"type": "json_object"}
    )
    result = response.choices[0].message.content
```

Dòng lệnh	Chức năng
def get_links(url):	Định nghĩa hàm `get_links` để lấy danh sách liên kết từ một URL.
website = Website(url)	Tạo đối tượng `Website` từ URL được cung cấp.
response = openai.chat.completions.create(...)	Gửi yêu cầu đến OpenAI API để xử lý và chọn lọc liên kết.
result = response.choices[0].message.content	Lấy nội dung phản hồi từ API.
return json.loads(result)	Chuyển đổi kết quả JSON từ API thành đối tượng Python.

```
def get_all_details(url):
    result = "Landing page:\n"
    result += Website(url).get_contents()
    links = get_links(url)
    print("Found links:", links)
    for link in links["links"]:
        result += f"\n\n{link['type']}\n"
        result += Website(link["url"]).get_contents()
```

Dòng lệnh	Chức năng
def get_all_details(url):	Định nghĩa hàm `get_all_details` nhận vào một tham số `url`.
result = "Landing page:\n"	Khởi tạo biến `result` với tiêu đề 'Landing page:'.
result += Website(url).get_contents()	Gọi phương thức `get_contents()` của lớp `Website` để lấy nội dung trang đích và nối vào `result`.
links = get_links(url)	Gọi hàm `get_links(url)` để lấy danh sách các liên kết từ trang web.
print("Found links:", links)	In danh sách các liên kết tìm thấy ra màn hình.
for link in links["links"]:	Lặp qua từng liên kết trong danh sách `links["links"]`.
result += f"\n\n{link['type']}\n"	Thêm loại liên kết (`type`) vào `result`.
result += Website(link["url"]).get_contents()	Gọi `get_contents()` cho từng liên kết để lấy nội dung và nối vào `result`.
return result	Trả về nội dung đã tổng hợp.

```
print(get_all_details("https://huggingface.co"))
system_prompt = "You are an assistant that analyzes the contents of several relevant pages from a company website \
and creates a short brochure about the company for prospective customers, investors and recruits. Respond in markdown.\
Include details of company culture, customers and careers/jobs if you have the information."
def get_brochure_user_prompt(company_name, url):
    user_prompt = f"You are looking at a company called: {company_name}\n"
    user_prompt += f"Here are the contents of its landing page and other relevant pages; use this information to build a short brochure of the company in markdown.\n"
    user_prompt += get_all_details(url)
    user_prompt = user_prompt[:5_000] # Truncate if more than 5,000 characters
        return user_prompt
get_brochure_user_prompt("HuggingFace", "https://huggingface.co")
```

Dòng lệnh	Chức năng
print(get_all_details("https://huggingface.co"))	In ra tất cả thông tin thu thập được từ trang web của Hugging Face.
system_prompt = "You are an assistant that analyzes..."	Định nghĩa chuỗi prompt cho AI, hướng dẫn tạo brochure về công ty.
def get_brochure_user_prompt(company_name, url):	Định nghĩa hàm tạo prompt dành cho người dùng dựa trên URL và tên công ty.
user_prompt = f"You are looking at a company called: {company_name}"	Xây dựng chuỗi prompt mô tả công ty.
user_prompt += f"Here are the contents of its landing page..."	Thêm nội dung từ trang web vào prompt.
user_prompt += get_all_details(url)	Gọi hàm `get_all_details()` để lấy thông tin từ trang web.
user_prompt = user_prompt[:5_000]	Cắt ngắn prompt nếu vượt quá 5.000 ký tự.
return user_prompt	Trả về prompt đã tạo.
get_brochure_user_prompt("HuggingFace", "https://huggingface.co")	Gọi hàm `get_brochure_user_prompt` để tạo nội dung nhắc cho việc tạo brochure về HuggingFace.

```
def create_brochure(company_name, url):
    response = openai.chat.completions.create(
        model=MODEL,
        messages=[
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": get_brochure_user_prompt(company_name, url)}
        ],
    )
    result = response.choices[0].message.content
    display(Markdown(result))

create_brochure("HuggingFace", "https://huggingface.co")
```

Dòng lệnh	Chức năng
def create_brochure(company_name, url):	Định nghĩa hàm `create_brochure` nhận vào tên công ty và URL.
response = openai.chat.completions.create(...)	Gọi API của OpenAI để tạo phản hồi dựa trên mô hình `MODEL`.
messages=[{"role": "system", "content":	Truyền vào danh sách tin nhắn gồm lời nhắc hệ

system_prompt}, ...]	thống (`system_prompt`) và lời nhắc từ người dùng.
{"role": "user", "content": get_brochure_user_prompt(company_name, url)}	Sử dụng nội dung từ `get_brochure_user_prompt` để gửi vào API dưới vai trò người dùng.
result = response.choices[0].message.content	Lấy nội dung phản hồi từ API và lưu vào biến `result`.
display(Markdown(result))	Hiển thị kết quả dưới dạng Markdown.
create_brochure("HuggingFace", "https://huggingface.co")	Gọi hàm `create_brochure` để tạo brochure cho HuggingFace.

```
def stream_brochure(company_name, url):
    stream = openai.chat.completions.create(
        model=MODEL,
        messages=[
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": get_brochure_user_prompt(company_name, url)}
          ],
        stream=True
    )

    response = ""
    display_handle = display(Markdown(""), display_id=True)
    for chunk in stream:
        response += chunk.choices[0].delta.content or ''
        response = response.replace("```","").replace("markdown", "")
        update_display(Markdown(response), display_id=display_handle.display_id)
```

Dòng lệnh	**Chức năng**
def stream_brochure(company_name, url):	Định nghĩa hàm `stream_brochure` để tạo brochure công ty theo luồng dữ liệu.
stream = openai.chat.completions.create(...)	Gửi yêu cầu đến OpenAI API để tạo nội dung brochure.
messages=[{"role": "system", "content": system_prompt}, ...]	Truyền vào hệ thống prompt và user prompt.
stream=True	Kích hoạt chế độ stream (nhận dữ liệu từng phần).
response = ""	Khởi tạo chuỗi rỗng để lưu phản hồi.
display_handle = display(Markdown(""), display_id=True)	Hiển thị markdown rỗng và lưu ID để cập nhật nội dung.
for chunk in stream:	Lặp qua từng phần dữ liệu nhận được.
response += chunk.choices[0].delta.content or ""	Cập nhật phản hồi với nội dung mới từ API.
response = response.replace("```","").replace("markdown", "")	Xóa ký tự không mong muốn từ phản hồi.
update_display(Markdown(response), display_id=display_handle.display_id)	Cập nhật nội dung hiển thị với phản hồi mới.
stream_brochure("HuggingFace", "https://huggingface.co")	Gọi hàm để tạo brochure cho Hugging Face.