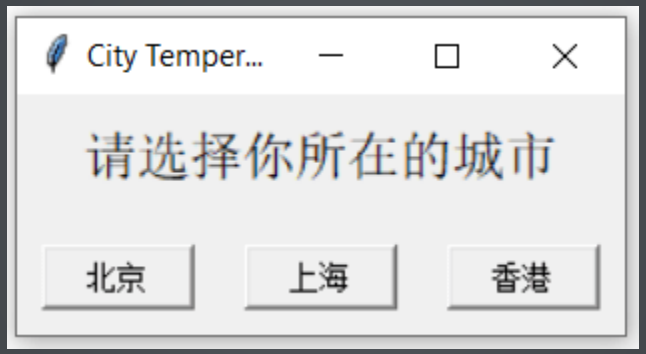
API

Demonstration and Vote!

This will be included as part of your in-class participation.

I developed an APP. Let's try it.



In your opinion, how does it get the temperature data?

This is done through an API.

API: application programming interface.

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.

You can view an API as a function (usually developed by a company), which helps you solve a task. You send your input to the function and it returns the output to you. You do not need to know how the function is computed.

This is done through an API.

You can view an API as a function (usually developed by a company), which helps you solve a task. You send your input to the function and it returns the output to you. You do not need to know how the function is computed. It is a black box.

In my example, there is a temperature API. I send the location to the API and it returns the temperature to me.

https://www.youtube.com/embed/s7wmiS2mSXY?enablejsapi=1

Let me show you the basics.

```python
import requests
url = "http://apis.juhe.cn/simpleWeather/query"
params = {
        "city": "北京",
        "key": "your API key",
    }
response = requests.get(url, params = params)
data = response.json()
temperature = data['result']['realtime']['temperature']
print(temperature)
```

You need to apply for your API key (password)

I will not share my key with you because a free key can only be used 50 times a day...

what is the request module in python

Completed 2 steps >

The `requests` module is a popular third-party Python library for sending HTTP requests to web servers easily and intuitively. It is **not** part of Python's standard library and must be installed separately with `pip install requests`. realpython

What It Does

`requests` abstracts the complexity of making HTTP connections behind simple, human-friendly function calls. It handles query strings, headers, authentication, cookies, and response parsing automatically. pypi +1

How to get the key?

1. Sign up an account here and activate it.
2. Sign up for an API key.

Exercise

Apply for your own API key and develop your own weather APP!

```python
import tkinter as tk
import requests
def get_temperature(city):
    url = "http://apis.juhe.cn/simpleWeather/query"
    params = { "city": city, "key": "your key"}
    response = requests.get(url, params=params)
    data = response.json()
    temperature = data['result']['realtime']['temperature']
    spaces_to_add = 3 - len(temperature)
    output_string = " " * spaces_to_add + temperature
    return output_string
def update_display(city):
    temperature = get_temperature(city)
    display_label.config(text="{}当前温度为: {}摄氏度".format(city, temperature))
root = tk.Tk()
root.title("City Temperature App")
display_label = tk.Label(root, text="    请选择你所在的城市    ", font=("Arial", 16))
display_label.grid(row=0, column=0, columnspan=3, pady=10)
cities = ["北京", "上海", "香港"]
for index, city in enumerate(cities):
    city_button = tk.Button(root, text=city, command=lambda c=city: update_display(c))
    city_button.grid(row=1, column=index, padx=10, pady=10, sticky="ew")
for i in range(3):
    root.grid_columnconfigure(i, weight=1)

root.grid_rowconfigure(1, weight=1)
root.mainloop()
```

Let’s try the following API for currency exchange

If you don’t understand the API, ask AI for help!

API can not only return data to you. It can also return images! Try the following API that returns a random dog image to you.

Try the following API for recognizing IP address.

AviationStack is a platform which provides an API to track the status of flights. Its website is here:

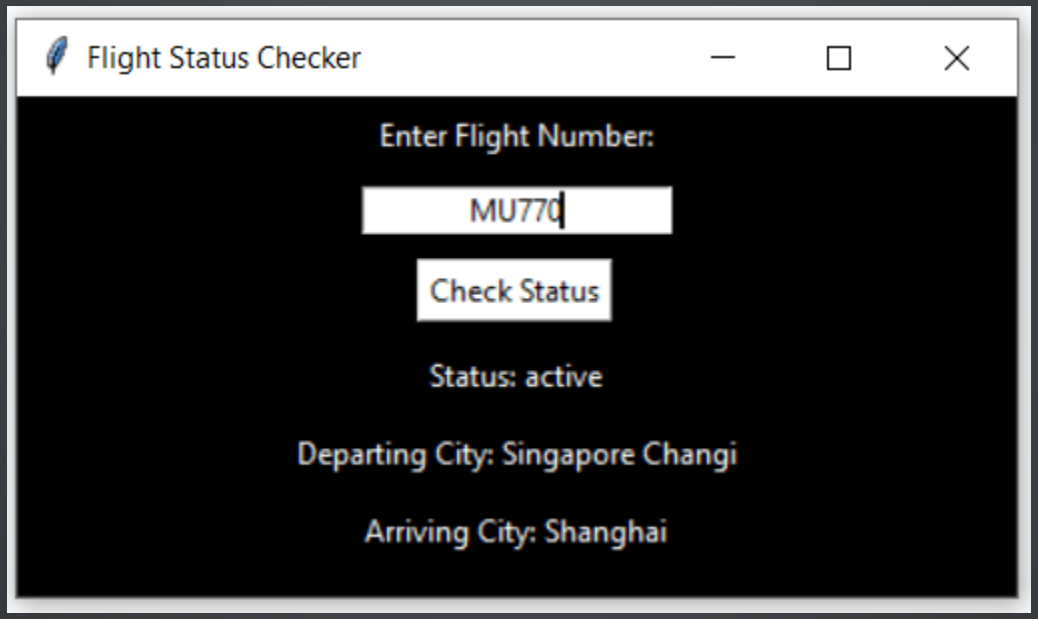
https://aviationstack.com/

You can find its API documentation here.

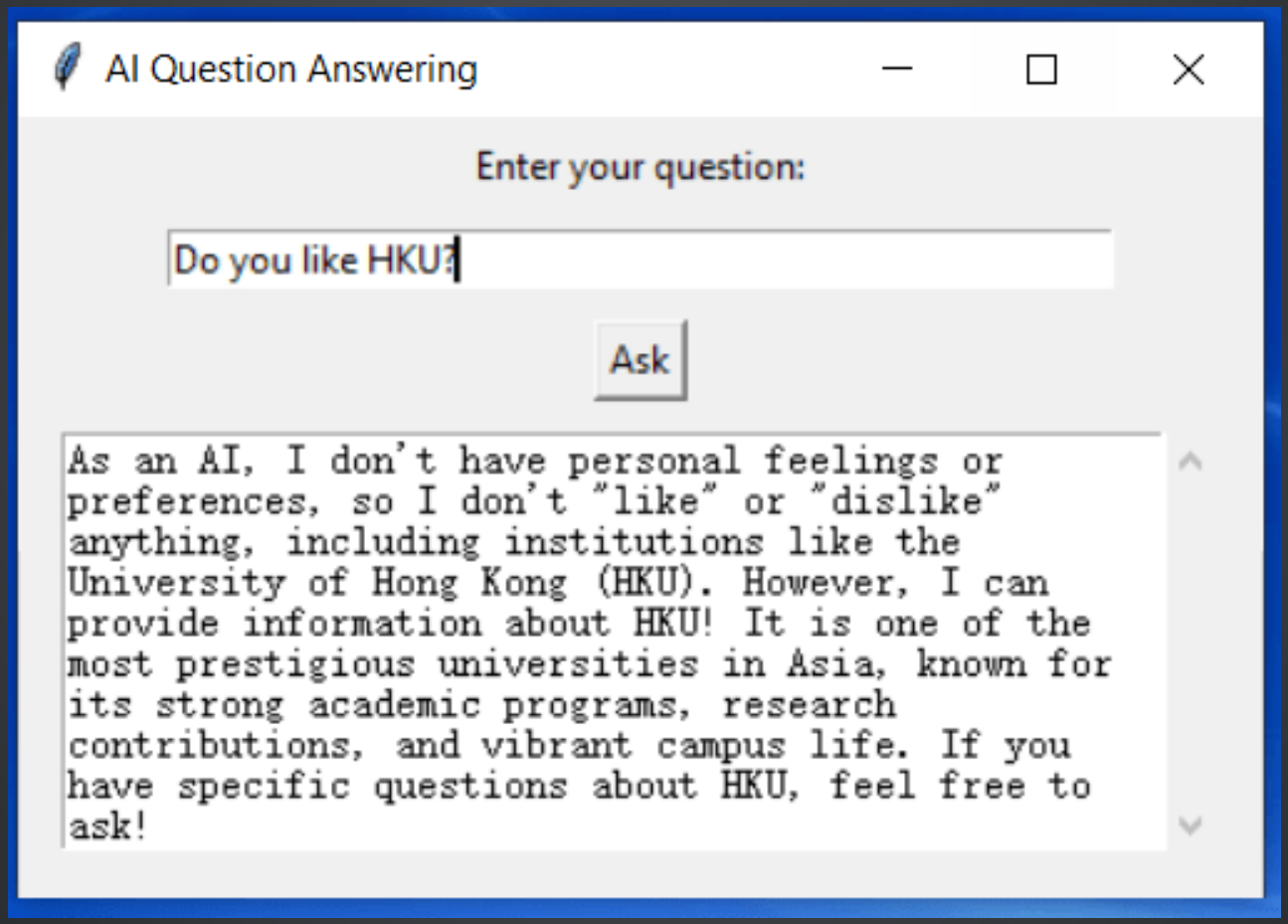
Exercise

- Acquire a free API Key from the platform
- Develop an APP, the APP takes flight number as input from the user, and displays its fight status, departing airport and arriving airport on the screen.
- Try to do everything yourself (with help from AI)!

This is my APP.

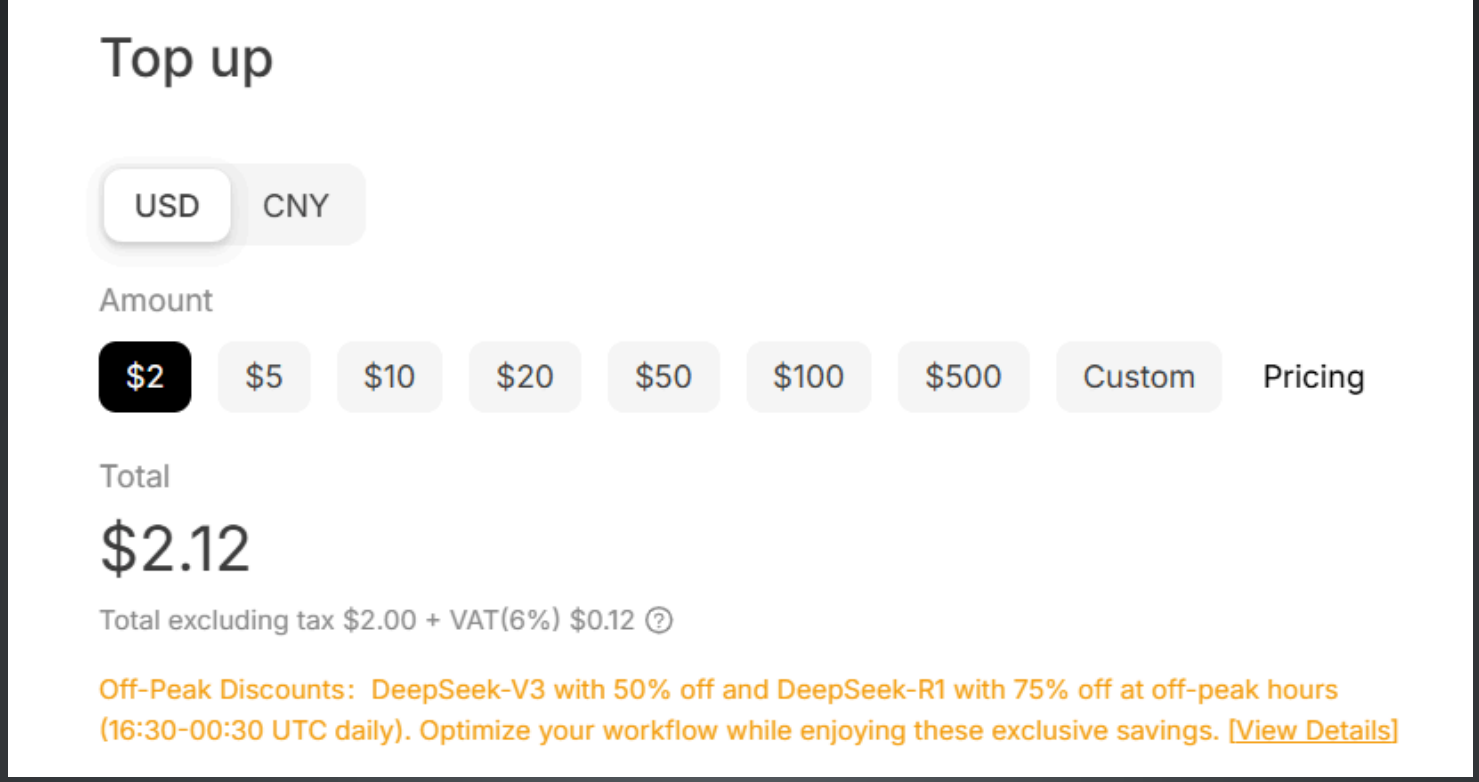


This is “my” AI (API)



Exercise

- Acquire an API Key from DeepSeek (https://platform.deepseek.com/)
- This is not free, but very cheap. You need to top up $2 or RMB 10 to use the API.
- Then, develop an APP which takes a question from user, sends the question to the API, and displays the answer from API to the user.



API keys

Your API keys are listed below. The API key is only visible and can be copied once at creation. Save it securely. Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, DeepSeek may also automatically disable any API key that we've found has leaked publicly. Usage of API keys created before April 25, 2024, was not tracked.

Name	Key	Created	Last used
ximarketing	sk-16926**********2e3e	2025-02-26	2025-02-26

If you need help on how to use the API, check the document here: https://api-docs.deepseek.com/

What are the main topics in modern economics?

Scraping made Simple

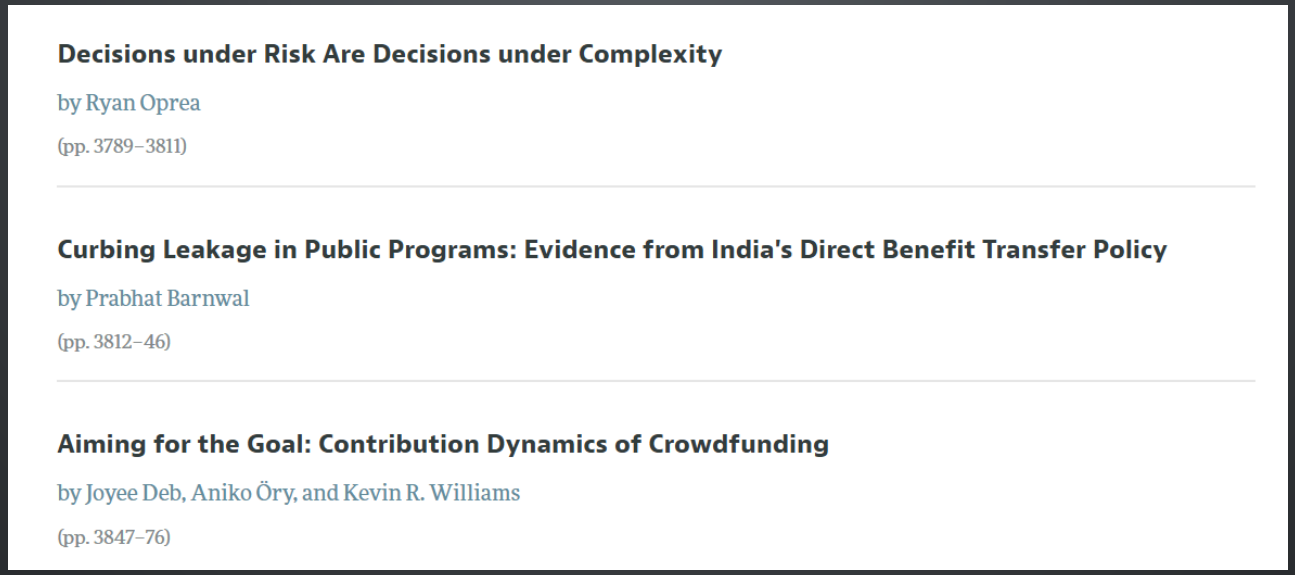
We will use Google Chrome as our browser.

Install one if you don’t have it.

Scraping

Consider a simple URL:
https://www.aeaweb.org/issues/785

American Economic Review is an academic journal in economics (we don't care it anyway). Let's try to write a scraper to download the article titles from the page:



Scraping

I am scraping article titles from https://www.aeaweb.org/issues/785. A title looks like Decisions under Risk Are Decisions under Complexity. Give me the python code

已完成 2 个步骤 >

To scrape the article titles from the provided American Economic Association (AEA) journal issue, you can use Python with the `requests` and `BeautifulSoup` libraries. The target webpage places article titles inside hyperlink tags that point to specific article IDs, which can be easily isolated .

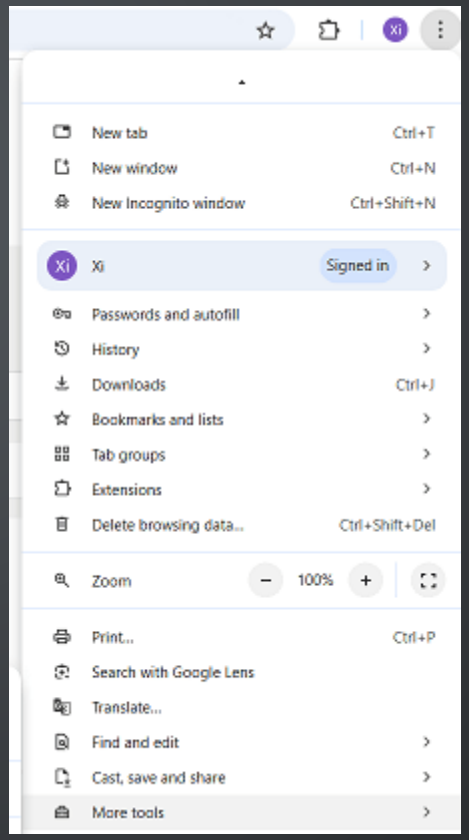
It sometimes works, but not always.

```python
import requests
from bs4 import BeautifulSoup
# URL of the page to scrape
url = "https://www.aeaweb.org/issues/785"
# Send a GET request to fetch the page content
response = requests.get(url)
# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Parse the page content with BeautifulSoup
    soup = BeautifulSoup(response.content, 'html.parser')

    # Find all article titles
    # Note: The exact HTML structure of the page needs to be inspected to find the correct tags and classes
    titles = soup.find_all('h2', class_='title')  # Example: Change these tags and classes based on actual HTML structure

    # Print the titles
    for title in titles:
        print(title.get_text().strip())
else:
    print(f"Failed to retrieve the page. Status code: {response.status_code}")
```

Check HTML with Chrome

Open the webpage in your Chrome browser.

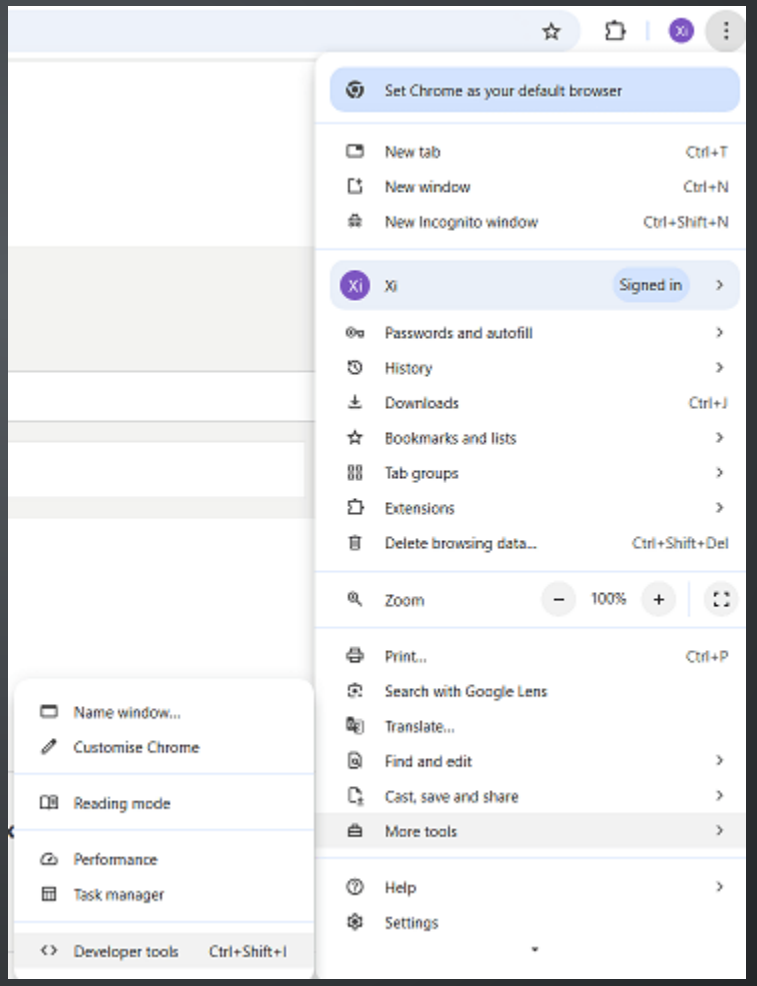
Click the upper right Chome setting button of your browser and you will be directed here.



Check HTML with Chrome

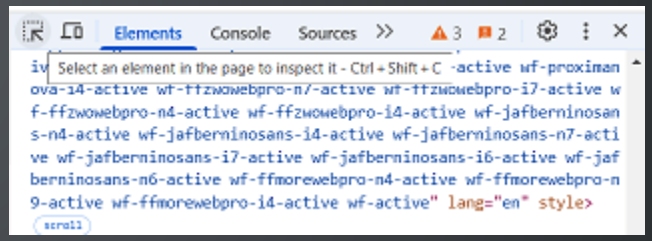
Choose “More tools”…

Choose “Developer tools”…



Check HTML with Chrome

Click the button and you will get to “select an element in the page to inspect it”.



Alternatively, use “Ctrl + Shift + C.”

If needed, also click the button to switch between desktop and mobile version.

Check HTML with Chrome

The title information is here.

Upload your screenshot to AI!

I am scraping article titles from https://www.aeaweb.org/issues/785. A title looks like Decisions under Risk Are Decisions under Complexity. Give me the python code


image.jpg

The code I got

```
import requests
from bs4 import BeautifulSoup
def scrape_aea_titles(url):
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
    }

    response = requests.get(url, headers=headers)
    if response.status_code != 200:
        print(f"Failed to retrieve page. Status code: {response.status_code}")
        return
    soup = BeautifulSoup(response.text, 'html.parser')
    title_elements = soup.select('article.journal-article h3.title a')
    print(f"Found {len(title_elements)} articles:\n")
    for element in title_elements:
        clean_title = ' '.join(element.text.split())
        print(clean_title)
target_url = "https://www.aeaweb.org/issues/785"
scrape_aea_titles(target_url)
```

Scraping

Consider a simple URL:
https://www.aeaweb.org/issues/785

Try to download the links of the articles now, with the help of AI. Here is an example of the link:

https://www.aeaweb.org/articles?id=10.1257/aer.20221227

Scraping

I am scraping article links from https://www.aeaweb.org/issues/785. A link looks like https://www.aeaweb.org/articles?id=10.1257/aer.20221227. Give me the python code


image.jpg

My Code

```python
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin
def scrape_aea_article_links(url):
    base_url = "https://www.aeaweb.org"
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
    }
    response = requests.get(url, headers=headers)
    response.raise_for_status()
    soup = BeautifulSoup(response.text, 'html.parser')
    article_links = []
    for link_tag in soup.select("article.journal-article h3.title a[href]"):
        full_url = urljoin(base_url, link_tag['href'])
        article_links.append(full_url)
    return article_links
if __name__ == "__main__":
    issue_url = "https://www.aeaweb.org/issues/785"
    links = scrape_aea_article_links(issue_url)
    print(f"Found {len(links)} links:\n")
    for link in links:
        print(link)
```

Exercise

- Go to https://www.aeaweb.org/issues/785 to download the links to each article.
- Go to each of the above links to download the abstract of the article.
- Print the all abstracts on the screen.

Exercise

First, I asked AI to scrape the abstract for a given URL.

I am scraping abstract from URL like https://www.aeaweb.org/articles?id=10.1257/aer.20221227. Given the URL, output abstract like "We provide evidence that classic lottery anomalies..."


image.jpg

Exercise

I got the following code.

```
import requests
from bs4 import BeautifulSoup

def get_aea_abstract(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    abstract_section = soup.find('section', class_='article-information
abstract')
    if abstract_section:
        # The abstract text is directly inside the section, after the <h4> tag
        return abstract_section.get_text(strip=True).replace('Abstract', '',
1).strip()
    return None

url = "https://www.aeaweb.org/articles?id=10.1257/aer.20221227"
print(get_aea_abstract(url))
```

Exercise

Next, ask AI to merge the two codes into one.

I have two codes. Code 1 allows me to get URLs and Code 2 allows me to get abstract from each URL. Merge them into a single python code so that it prints each abstract of each URL one by one.

Code 1:

显示更多

Exercise

```python
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin

def get_aea_abstract(url):
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36"
    }
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, "html.parser")
    sec = soup.find("section", class_="article-information abstract")
    if sec:
        return sec.get_text(strip=True).replace("Abstract", "", 1).strip()
    return "No abstract found"


issue_url = "https://www.aeaweb.org/issues/785"
base_url = "https://www.aeaweb.org"

response = requests.get(issue_url)
soup = BeautifulSoup(response.text, "html.parser")
links = [
    urljoin(base_url, a["href"])
    for a in soup.select("article.journal-article h3.title a[href]")
]

for url in links:
    print(url)
    print(get_aea_abstract(url))
    print("-" * 80)
```

We can do more!

- On page https://www.aeaweb.org/journals/aer/issues, we can scrape the URL of each issue.
- On the issue page, we can scrape the URL of each article.
- On the article page, we can scrape the abstract of each article (it took me two hours).
- Then, we can save all abstracts in one single file.
- After that, we can perform topic modeling!

We can do more!

I have already uploaded the txt file to the course website, which can be found here:
http://ximarketing.github.io/data/AER_abstracts.txt

Please try the top words and LDA algorithm yourself and demonstrate the main topics in modern economics.

Exercise: Download the Photos of All HKU Business School Professors from https://www.hkubs.hku.hk/people/faculty?pg=1&staff_type=faculty&subject_area=all&track=professoriate&region=hku

Downloading from a single page:

```python
import requests
from bs4 import BeautifulSoup
import os
url = "https://www.hkubs.hku.hk/people/faculty?pg=1&staff_type=faculty&subject_area=all&track=professoriate&region=hku"
response = requests.get(url)
if response.status_code == 200:
    soup = BeautifulSoup(response.text, 'html.parser')

    photo_divs = soup.find_all('div', class_='people-item')
    photo_urls = []

    for div in photo_divs:
        img_tag = div.find('img')
        if img_tag and 'src' in img_tag.attrs:
            photo_url = img_tag['src']
            photo_urls.append(photo_url)

    os.makedirs('profile_photos', exist_ok=True)

    for idx, photo_url in enumerate(photo_urls):
        img_response = requests.get(photo_url)

        if img_response.status_code == 200:
            # Save the photo to the directory
            with open(f'profile_photos/photo_{idx + 1}.jpg', 'wb') as f:
                f.write(img_response.content)
            print(f'Downloaded: photo_{idx + 1}.jpg')
        else:
            print(f'Failed to download photo from {photo_url}')
else:
    print(f'Failed to retrieve the page. Status code: {response.status_code}')
```

I further asked the following question...

I have 10 URLs each containing 16 photos, and I want to download all these photos. The URLs are https://www.hkubs.hku.hk/people/faculty?pg=1&staff_type=faculty&subject_area=all&track=professoriate&region=hku to https://www.hkubs.hku.hk/people/faculty?pg=10&staff_type=faculty&subject_area=all&track=professoriate&region=hku

I am done!


photo_1
photo_2
photo_3
photo_4
photo_5
photo_6
photo_7
photo_8
photo_9
photo_10
photo_11
photo_12
photo_13
photo_14
photo_15
photo_16
photo_page1_1
photo_page1_2
photo_page1_3
photo_page1_4
photo_page1_5
photo_page1_6
photo_page1_7
photo_page1_8
photo_page1_9
photo_page1_10
photo_page1_11
photo_page1_12
photo_page1_13
photo_page1_14
photo_page1_15
photo_page1_16
photo_page2_1
photo_page2_2
photo_page2_3
photo_page2_4
photo_page2_5
photo_page2_6
photo_page2_7
photo_page2_8

The code I used

```python
import requests
from bs4 import BeautifulSoup
import os
base_url = "https://www.hkubs.hku.hk/people/faculty?pg={}&staff_type=faculty&subject_area=all&track=professoriate&region=hku"
os.makedirs('profile_photos', exist_ok=True)
for page in range(1, 11):
    url = base_url.format(page)
    print(f"Fetching page: {url}")
    response = requests.get(url)
    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        photo_divs = soup.find_all('div', class_='people-item')
        for idx, div in enumerate(photo_divs):
            img_tag = div.find('img')
            if img_tag and 'src' in img_tag.attrs:
                photo_url = img_tag['src']
                img_response = requests.get(photo_url)
                if img_response.status_code == 200:
                    photo_filename = f'profile_photos/photo_page{page}_{idx + 1}.jpg'
                    with open(photo_filename, 'wb') as f:
                        f.write(img_response.content)
                    print(f'Downloaded: {photo_filename}')
                else:
                    print(f'Failed to download photo from {photo_url}')
    else:
        print(f'Failed to retrieve the page. Status code: {response.status_code}')
```

It becomes easier when I use more advanced LLMs: link

I would like to write Python codes to download photos from https://www.hkubs.hku.hk/people/faculty?pg=1&staff_type=faculty&subject_area=all&track=professoriate&region=hku and save them in my local folder. Here is a sample link to photo: https://www.hkubs.hku.hk/wp-content/uploads/fly-images/196951/Andy-Back-scaled-940x1000-ct.jpg Moreover, when saving the photos, name them using the faculty name, e.g., Andy_Back.jpg

Try to scrape article titles from the following page:
https://pubsonline.informs.org/toc/mnsc/71/1

Try to scrape article titles from the following page:
https://pubsonline.informs.org/toc/mnsc/71/1

In many cases, the simple scraper does not work. Possible reasons:

- They adopt anti-scraping technologies to prevent you from scraping their websites.
- The website offers different content to human users and scrapers.
- The website is dynamic (e.g., it requires password)

What else can we do with scraping?

What else can we do with scraping?

- Scraping a website with selenium.
- Web scraping with concurrency (parallel scraping).
- Changing IP addresses with VPN (using proxies/VPNs).

https://www.youtube.com/embed/cc26zFE8X1k?enablejsapi=1

Scraping with Selenium

```
import undetected_chromedriver as uc
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
url = "https://pubsonline.informs.org/toc/mnsc/71/1"
options = uc.ChromeOptions()
# options.add_argument("--headless=new")  # keep headed; Cloudflare blocks headless harder

driver = uc.Chrome(options=options, version_main=147)   # <-- pin to your Chrome major version

try:
    driver.get(url)

    WebDriverWait(driver, 30).until(
        EC.presence_of_all_elements_located((By.CSS_SELECTOR, "h5.issue-item__title a"))
    )

    titles = driver.find_elements(By.CSS_SELECTOR, "h5.issue-item__title a")
    for t in titles:
        print(t.text.strip())
finally:
    driver.quit()
```