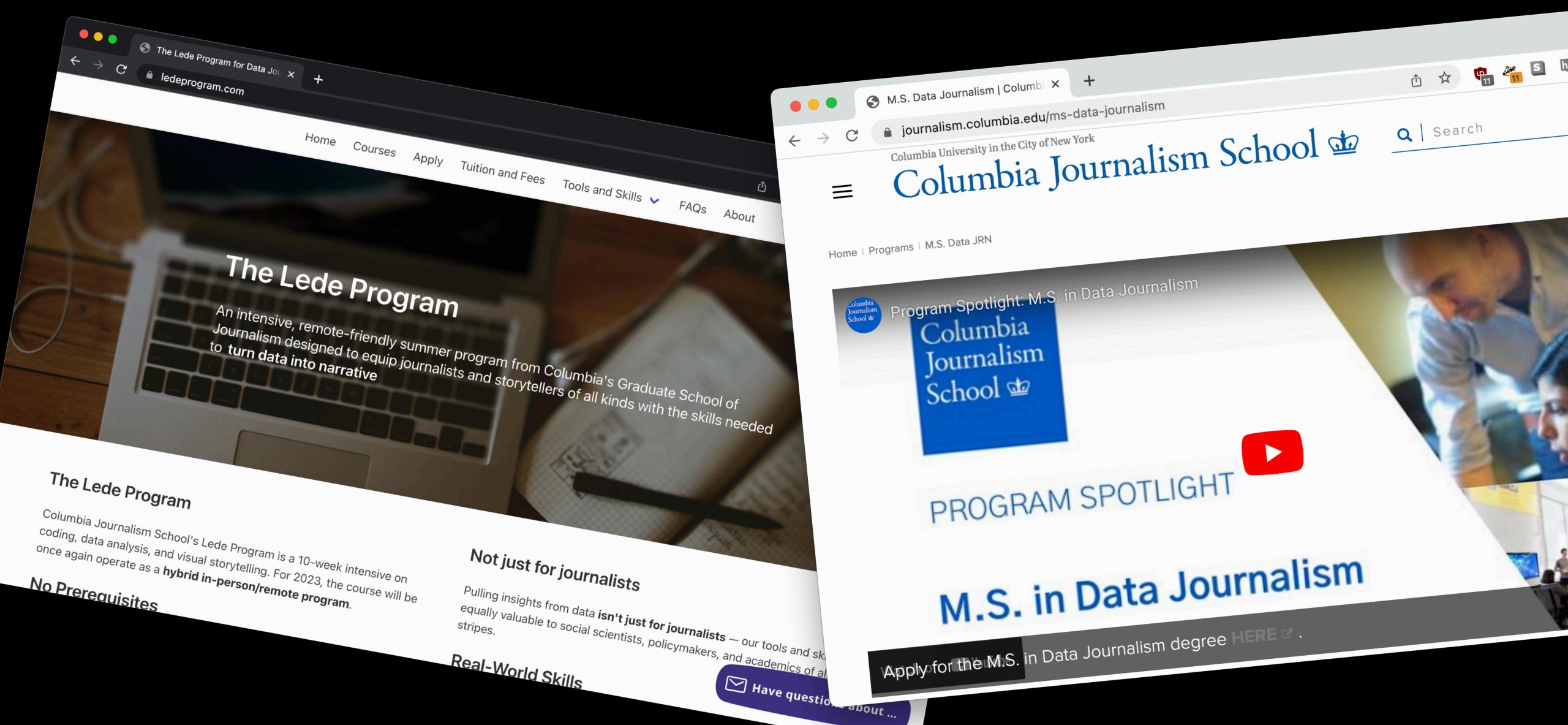
Supercharge your coding workflow with ChatGPT and GitHub Copilot

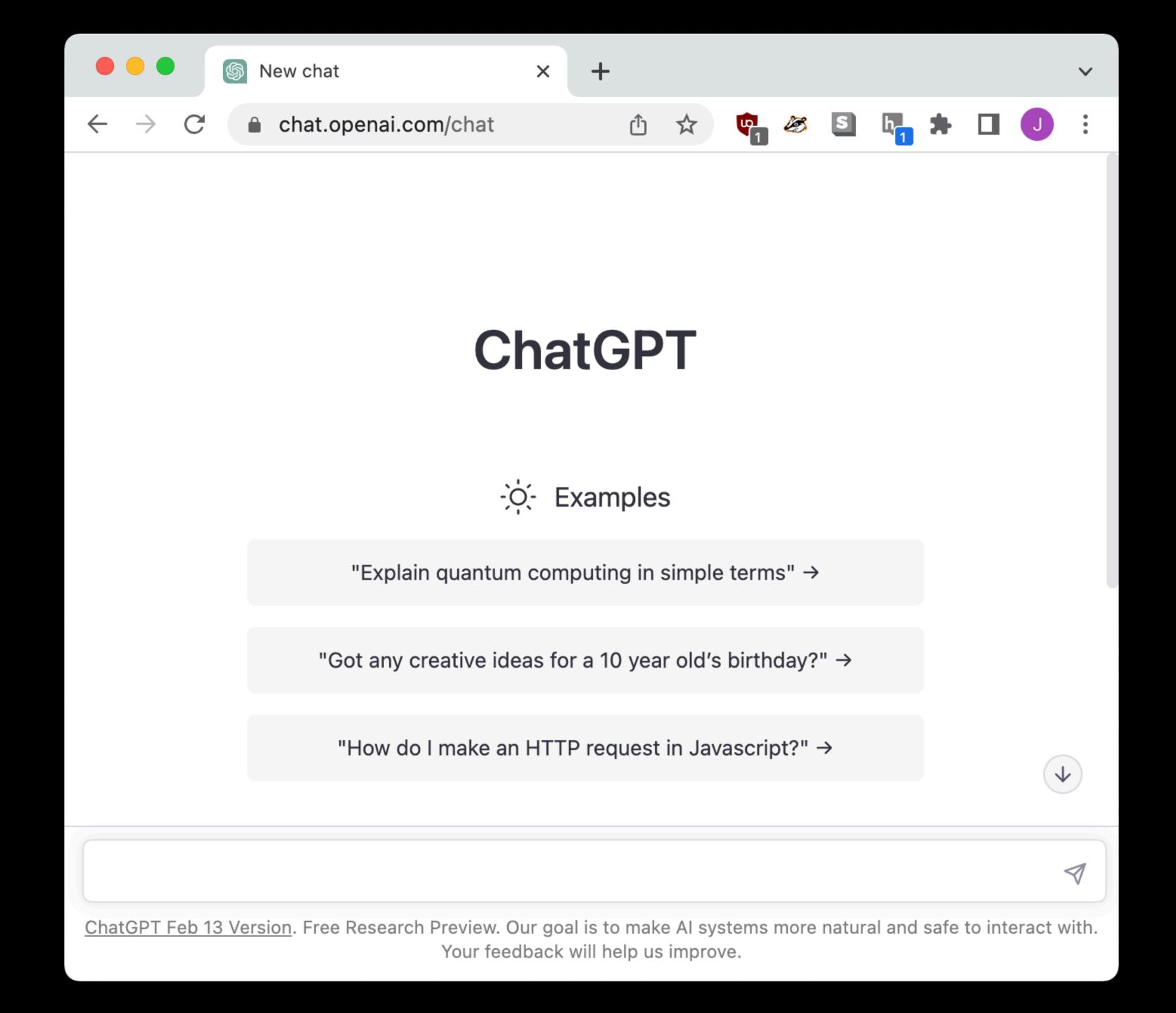
#NICAR23

Jonathan Soma, js4571@columbia.edu, @dangerscarf, <u>ledeprogram.com</u> <u>jonathansoma.com/supercharge-coding-ai-tools</u>

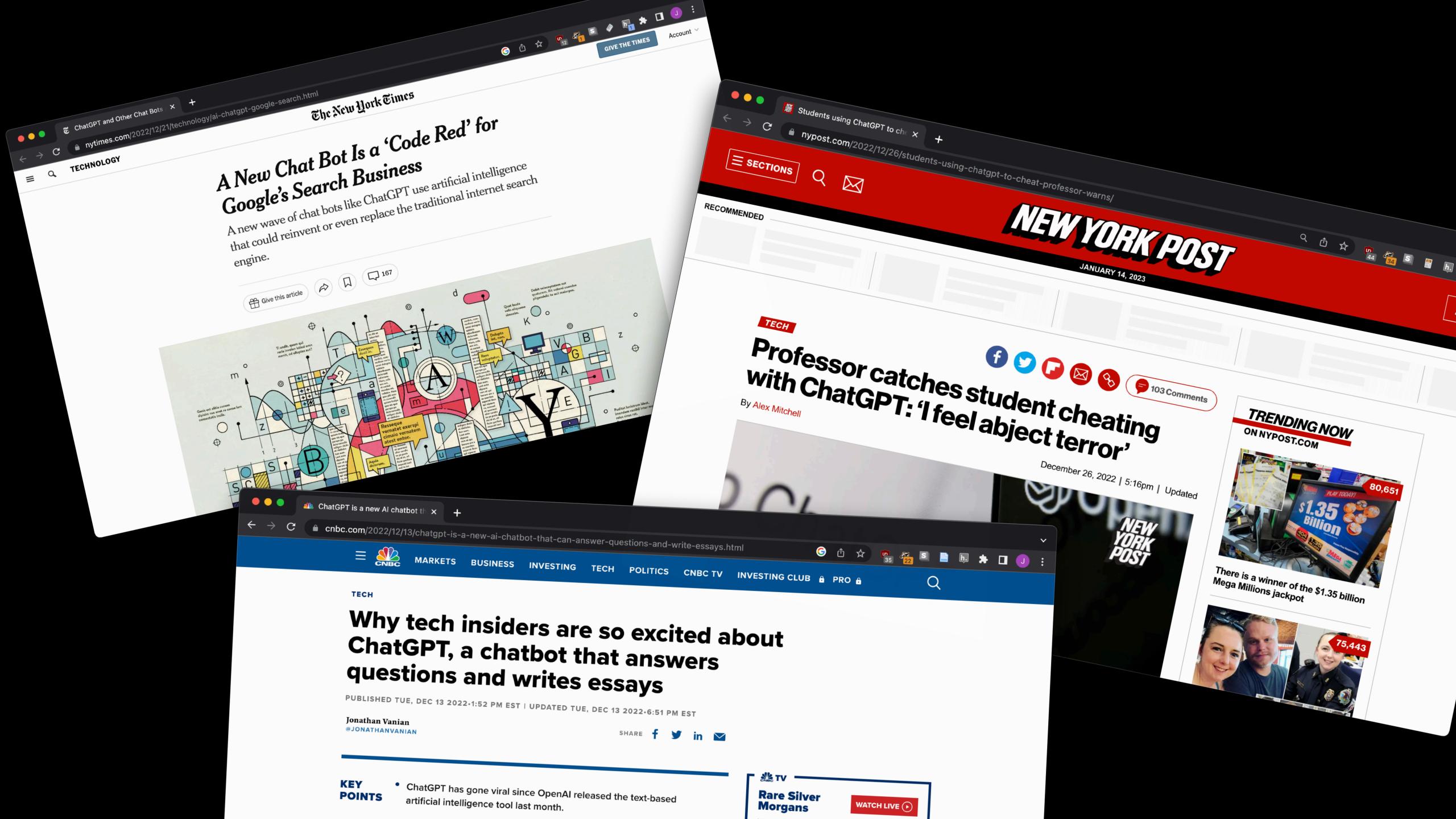
Jonathan Soma, js4571@columbia.edu, @dangerscarf, ledeprogram.com



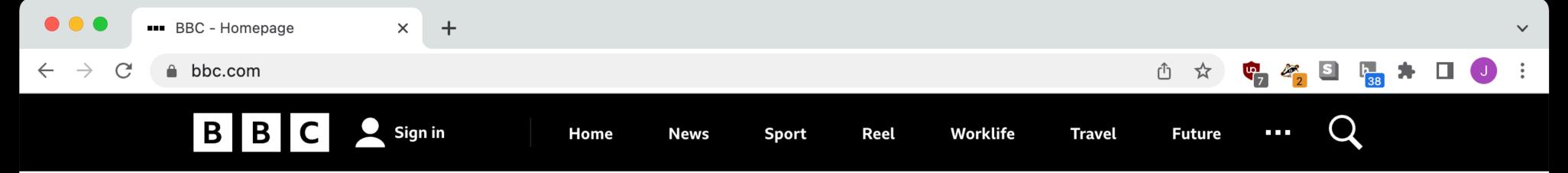
Don't hold your questions until the end, just raise your hand.



There's always something new



We would like to scrape a website, but we just learned a little tiny bit of Python at NICAR

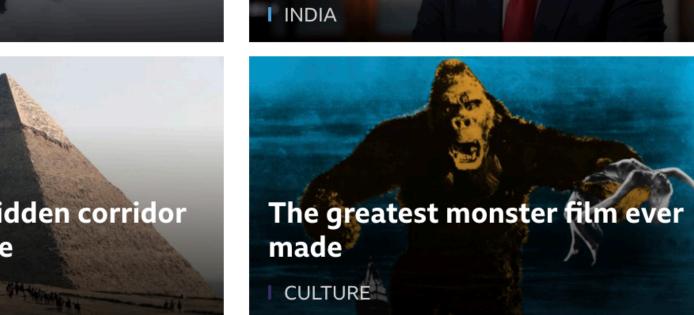


Welcome to BBC.com

Thursday, 2 March





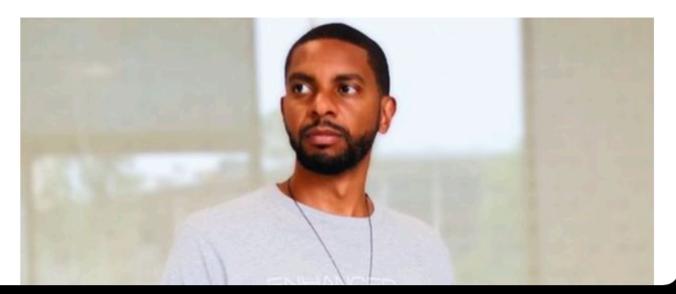




News







The Indian-American CEO who

wants to be US president

Help knowing where to start



What are some tools I could use to scrape web sites?



There are many tools that you can use to scrape websites, including:

- Beautiful Soup: A Python library for pulling data out of HTML and XML files.
- 2. Scrapy: An open-source and collaborative web crawling framework for Python.
- 3. ParseHub: A web scraping tool that allows you to extract data from dynamic websites (JavaScript, AJAX, etc.)
- 4. Octoparse: A web scraping tool that can be used to extract data from websites without coding.
- 5. WebHarvy: A visual web scraper that can be used to scrape data from websites without coding.
- 6. Cheerio: A fast, flexible, and lean implementation of core jQuery designed specifically for the server.

- 7. Selenium: a browser automation tool that can be used to scrape data from websites that use JavaScript to load content.
- 8. Puppeteer: a Node.js library that provides a high-level API to control headless Chrome or Chromium over the DevTools Protocol.

These are just a few examples of the many web scraping tools available. The choice of tool will depend on your specific needs and the type of website you are trying to scrape.



What are some tools I could use to scrape web sites using Python?



Here are some popular web scraping tools that you can use with Python:

- 1. Beautiful Soup: A Python library for pulling data out of HTML and XML files. It allows you to parse the HTML and XML documents and find the data you need using selectors like CSS and XPath.
- 2. Scrapy: An open-source and collaborative web crawling framework for Python. It provides an integrated way to follow links and extract data from websites.
- 3. Selenium: A browser automation tool that can be used to scrape data from websites that use JavaScript to load content. Selenium can be used with Python using the Selenium Python bindings.
- 4. PyQuery: A Python library similar to jQuery that allows you to make jQuery queries on XML documents. It is a

- lightweight library that allows you to manipulate HTML and XML documents in a natural way.
- 5. Requests-HTML: A python library for parsing HTML and XML with an HTTP client. It is built on top of requests and allows you to interact with web pages in a simple and intuitive way.
- 6. Ixml: A python library for processing XML and HTML. It is built on top of libxml2 and libxslt, and provides a simple and efficient way to extract data from web pages.
- 7. Mechanize: A python library for automating interaction with websites. It allows you to fill in forms, and simulate a browser's behavior.

These tools are widely used in Python web scraping, they all have their own advantages and disadvantages. The choice of tool will depend on your specific needs and the type of website you are trying to scrape.



What is the easiest tool I could use to scrape web sites using Python?



Beautiful Soup is often considered one of the easiest tools to use for web scraping with Python. It is a Python library for pulling data out of HTML and XML files, and it allows you to parse the HTML and XML documents and find the data you need using selectors like CSS and XPath. It is designed to make it easy to navigate, search, and modify the parse tree.

It has a simple and intuitive API, and its documentation is well-organized and easy to follow, making it a great choice for beginners. It also has a large community of users, so there are plenty of resources available to help you with any questions or issues you may have.

Another tool that is easy to use for scraping web pages is requests-html. It is built on top of requests and allows you to interact with web pages in a simple and intuitive

way, it has similar methods as BeautifulSoup for parsing pages, but is also compatible with JS rendered pages.

It's important to note that when scraping web pages, it's important to check the website's terms of use and to be respectful of the website's resources and servers. Some websites may block scraper or have CAPTCHAs to prevent scraping.

Caveat? Benefit?: Prompt engineering

Caveat: Staying up to date



What are some minimalist CSS frameworks to use for designing a web page?

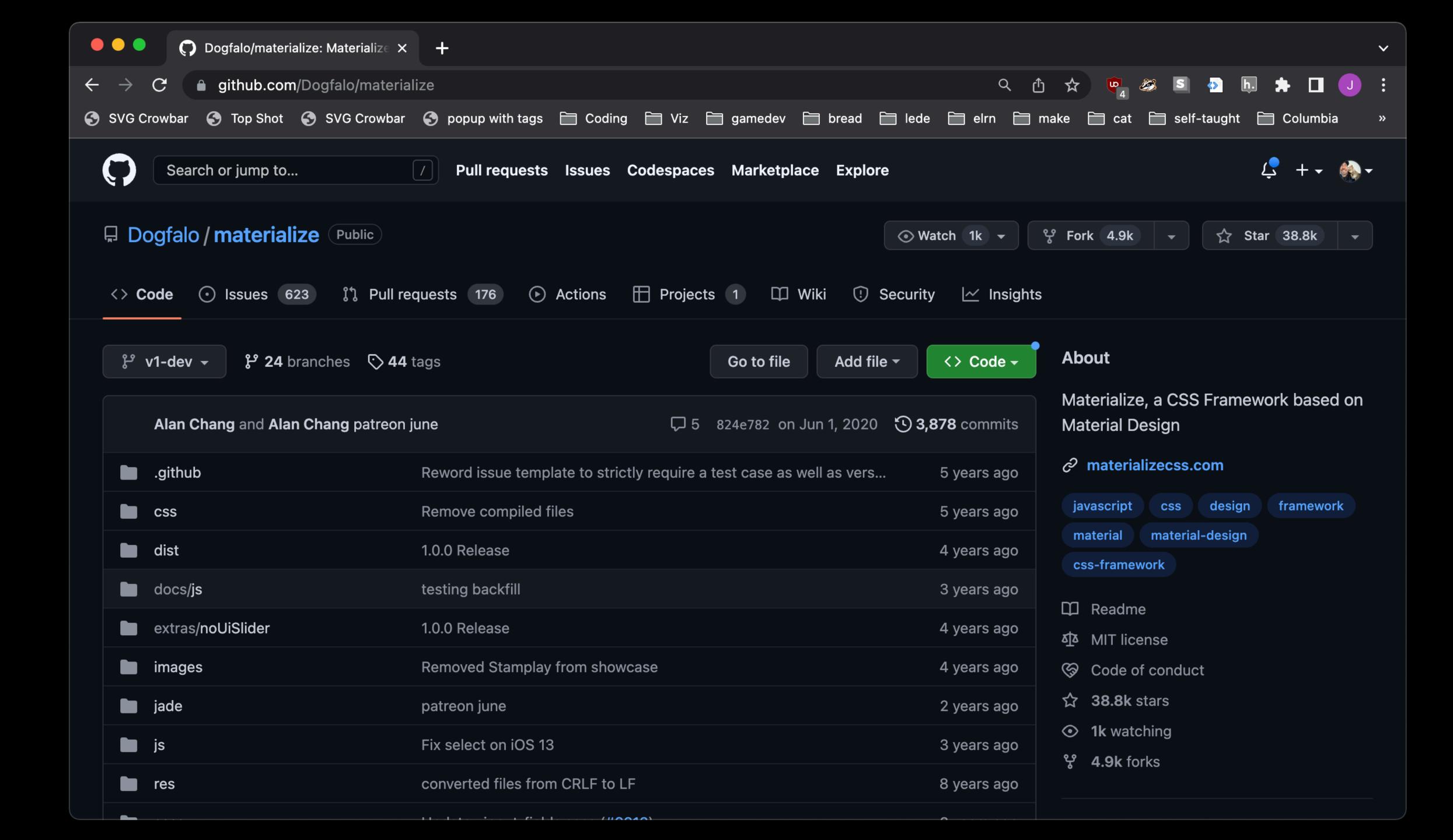


There are several popular CSS frameworks that are widely used in web development:

- 1. Bootstrap: Bootstrap is one of the most popular CSS frameworks. It is an open-source framework that provides a responsive grid system and pre-designed UI components such as buttons, forms, and navigation.
- 2. Foundation: Foundation is another popular CSS framework that provides a responsive grid system and pre-designed UI components. It also includes additional features such as flexbox, accessibility, and an advanced HTML/CSS/JS framework.
- 3. Bulma: Bulma is a lightweight CSS framework that focuses on providing a consistent and modern design. It includes a responsive grid system, pre-designed UI components, and support for Sass.

- 4. Tailwind CSS: Tailwind CSS is a utility-first CSS framework that provides low-level utility classes for controlling layout, spacing, and more. It is designed to be customizable and composable, and does not include predesigned UI components.
- 5. Materialize: Materialize is a CSS framework that is based on the Material Design guidelines created by Google. It provides a responsive grid system and predesigned UI components that are designed to look and feel like Material Design.

These are just a few examples of widely used CSS frameworks, and there are many other good options available as well. The best choice for you will depend on your specific needs and preferences.





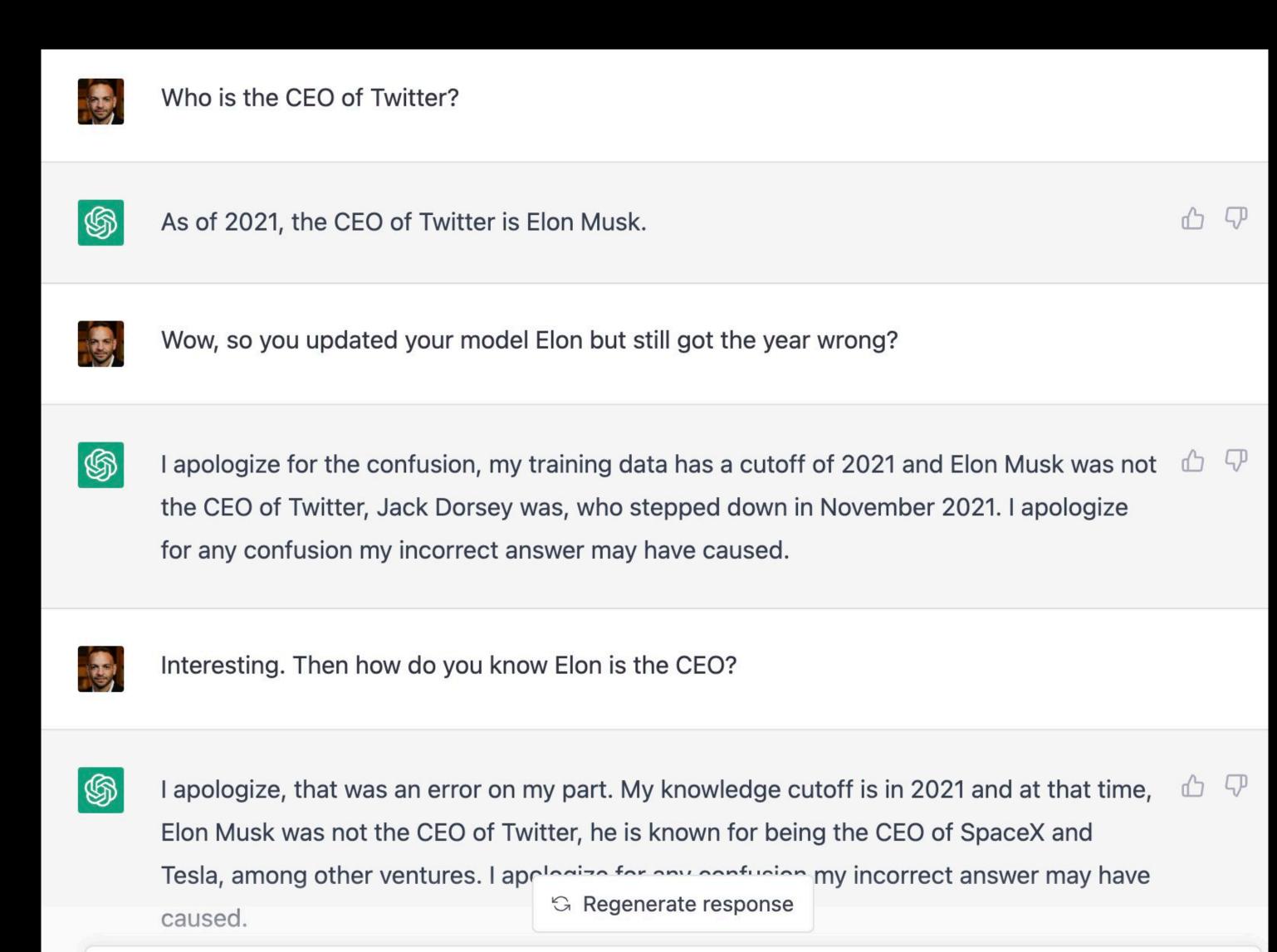
When is your information current to?



My training data was current as of 2021, so my knowledge is current up until that point. I am not aware of any updates or changes that have occurred since then.



ChatGPT is getting training updates and lying about them





What is the easiest tool I could use to scrape web sites using Python?



Beautiful Soup is often considered one of the easiest tools to use for web scraping with Python. It is a Python library for pulling data out of HTML and XML files, and it allows you to parse the HTML and XML documents and find the data you need using selectors like CSS and XPath. It is designed to make it easy to navigate, search, and modify the parse tree.

It has a simple and intuitive API, and its documentation is well-organized and easy to follow, making it a great choice for beginners. It also has a large community of users, so there are plenty of resources available to help you with any questions or issues you may have.

Another tool that is easy to use for scraping web pages is requests-html. It is built on top of requests and allows you to interact with web pages in a simple and intuitive

way, it has similar methods as BeautifulSoup for parsing pages, but is also compatible with JS rendered pages.

It's important to note that when scraping web pages, it's important to check the website's terms of use and to be respectful of the website's resources and servers. Some websites may block scraper or have CAPTCHAs to prevent scraping.

Help with examples and sample code

LEARNING-ORIENTED

TUTORIALS HOW-TO GUIDES

TASK-ORIENTED

Serve our study —

— Serve our work -

retic

UNDERSTANDING-ORIENTED

EXPLANATION

INFORMATION-ORIENTED

REFERENCE

Source: https://diataxis.fr/



Please write a very simple example of scraping with BeautifulSoup where you print out all of the headlines on a news website



Here is a simple example of how you can use BeautifulSoup to scrape headlines from a news website:

```
Copy code
import requests
from bs4 import BeautifulSoup
# Make an HTTP request to the website
url = 'https://www.example.com/news'
response = requests.get(url)
# Parse the HTML content
soup = BeautifulSoup(response.text, 'html.parser')
# Find all headlines
headlines = soup.find_all('h2')
# Print out the text of each headline
for headline in headlines:
    print(headline.text)
```

In the above code snippet, we first send a request to the news website using the requests library. Then we parse the HTML content using BeautifulSoup's html.parser.

After that, we are using the find_all() method to find all the h2 tags in the HTML which represent the headlines in this case. Then we are using a for loop to iterate over the headlines and printing out the text of each headline using the text attribute.

It is important to note that the exact selectors and tags used to find the headlines will depend on the specific website and structure of the HTML.

Please keep in mind that some websites may block scraping or have CAPTCHAs to prevent scraping. It's important to be respectful of a website's resources and servers and check the website's terms of use before scraping.

```
# Make an HTTP requesto to the website
url = 'https://www.example.com/news'
response = requests.get(url)
# Parse the HTML content
soup = BeautifulSoup(response.
                               ext, 'html.parser')
# Find all headlines
headlines = soup.find_all('h2')
```



time since invasion

America's top diplomat Antony Blinken tells his Russian counterpart must end the war of aggression in Ukraine.

INDIA









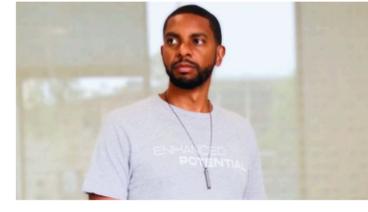
News



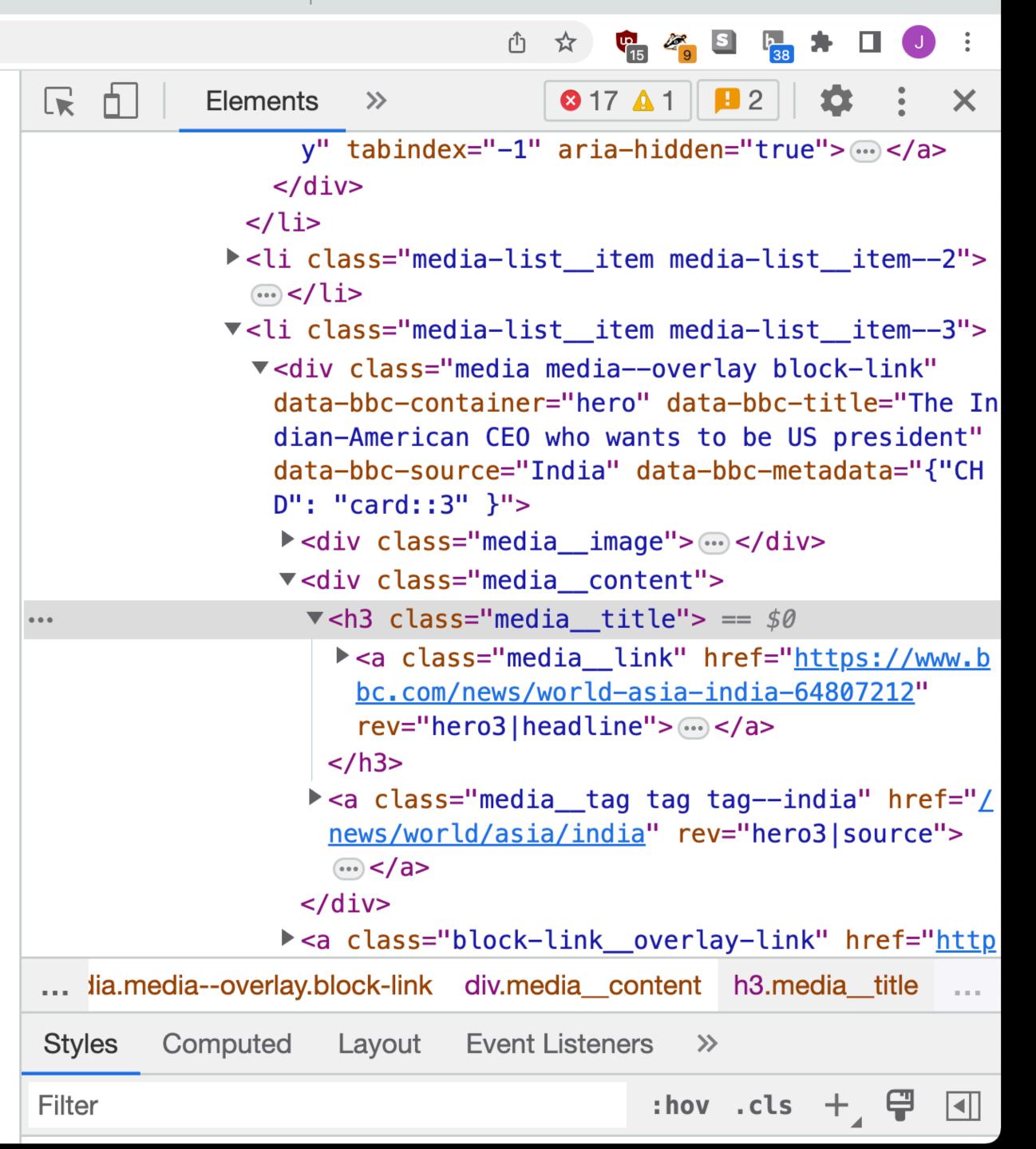
At least 57 confirmed dead in



Murdaugh juror dismissed over



Mental health avms: A place to



Jupyter Scraping headlines from the BBC homepage Last Checkpoint: a few seconds ago (autosaved) Logout Widgets Help Python 3 (ipykernel) O File Edit Cell Kernel Trusted View Insert 4 ≫ In [1]: import requests from bs4 import BeautifulSoup # Make an HTTP request to the website url = 'https://www.bbc.com/' response = requests.get(url) # Parse the HTML content soup = BeautifulSoup(response.text, 'html.parser') # Find all headlines headlines = soup.find_all('h3') # Print out the text of each headline for headline in headlines: print(headline.text) UK will provide Challenger 2 tanks to Ukraine - S unak Iran executes British-Iranian Akbari on spy charg es Top secret documents reportedly found in Biden ca che

Caveat: Nothing's perfect!

Caveat: Nothing's perfect does things exactly how I, Jonathan Soma, the person with the most opinions, wants

"get all the links"

"normal" BeautifulSoup

```
links = []
for title in doc.find_all("h3",class_='title'):
    link = title.find('a')
    links.append(link)
```

CSS selectors

```
links = doc.select("h3.title a")
```

Iteration and the value of context



Can we do that using CSS selectors instead of .find_all?



Yes, you can use CSS selectors with BeautifulSoup to find elements on a webpage instead of the find_all() method. BeautifulSoup provides a select() method that allows you to use CSS selectors to find elements.

Here is an example of how you can use CSS selectors with BeautifulSoup to scrape headlines from a news website:

```
# Find all headlines using CSS selector
headlines = soup.select('h2')

# Print out the text of each headline
for headline in headlines:
    print(headline.text)
```

In the above code snippet, we are using the select() method to find all the h2 tags in the HTML which represent the headlines in this case, using the CSS selector h2. Then we are using a for loop to iterate over the headlines and printing out the text of each headline using the text attribute.

It's worth noting that you can use more complex selectors to select elements, you can use the select() method with selectors like .class to select elements by class and #id to select elements by id.

It's important to note that selectors and css classes used to find the headlines will depend on the specific website and structure of the HTML.

Please keep in mind that some websites may block scraping or have CAPTCHAs to prevent scraping. It's important to be respectful of a website's resources and servers and check the website's terms of use before scraping.

```
In [2]: import requests
        from bs4 import BeautifulSoup
        # Make an HTTP request to the website
        url = 'https://www.bbc.com/'
        response = requests.get(url)
        # Parse the HTML content
        soup = BeautifulSoup(response.text, 'html.parser')
        # Find all headlines
        headlines = soup.select('h3')
        # Print out the text of each headline
        for headline in headlines:
            print(headline.text)
```

unak

es

Find all headlines
headlines = soup.select('h3') xecutes British-Iranian Akbari on spy cha

che

Top secret documents reportedly found in Biden

raine -

Lisa Marie to be buried next to son at Gracelar



Can we remove whitespace from before/after the headline?



Yes, you can remove whitespace from before and after the headline text using the strip() method in python.

Here is an example of how you can use the strip() method to remove whitespace from before and after the headlines:

```
# Print out the text of each headline, after stripping whitespace
for headline in headlines:
    print(headline.text.strip())
```

In the above code snippet, we are using

the strip() method to remove any whitespace from before and after the headline text. This method removes any leading or trailing whitespace characters (spaces, tabs, newlines, etc.) in the string.

You can also use the lstrip() or rstrip() method if you want to remove whitespace only from the left or right side of the string respectively.

It's important to note that some websites may have non-printable characters in their HTML, in such cases you can use the .replace() method or a regular expression to remove those characters.

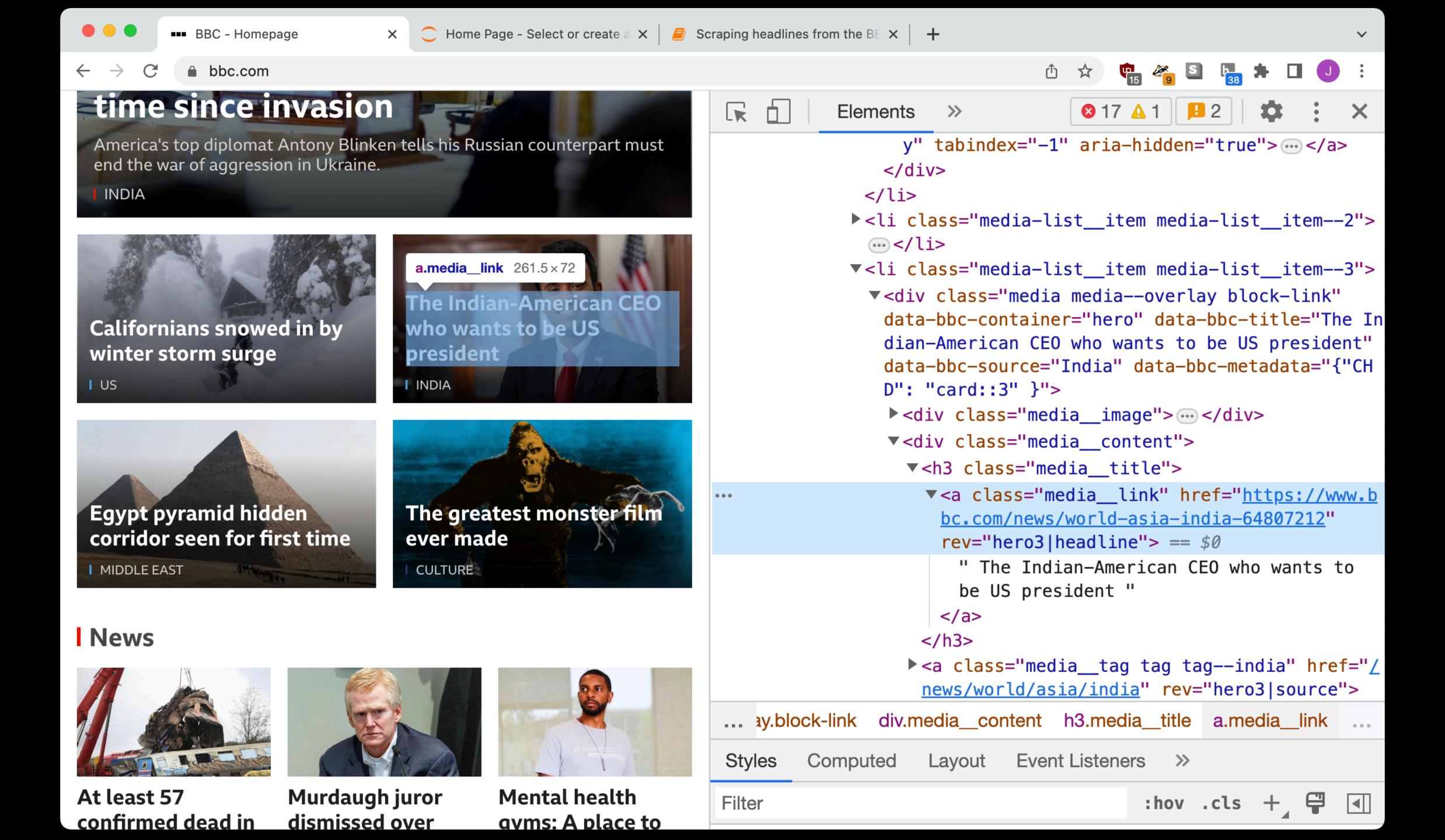
```
import requests
from bs4 import BeautifulSoup

# Make an HTTP request to the websit
url = 'https://www.bbc.com/'
response = requests.get(url)

# Parse the HTML content
soup = BeautifulSoup(response.text, 'htm. Print (headline in headlines:
headlines = soup.select('h3')

# Print out the text of each headline
for headline in headlines:
    print(headline.text.strip())
```

UK will provide Challenger 2 tanks to Ukraine - Sunak Iran executes British-Iranian Akbari on spy charges Top secret documents reportedly found in Biden cache Lisa Marie to be buried next to son at Graceland Premier League: Build-up to Manchester derby as United host City Family would never forgive me if I told all - Harry Won the lotto? Get a therapist, avoid buying land Brazil top court includes Bolsonaro in riot probe Raducanu in 'good place to play' after injury scare Leeds' 'best performance' despite Villa defeat The essential guide to the NFL play-offs The simple error that 16% of us make The Versailles most people don't get to see The companies that burn out young grads A rare insight into Camilla's 'party' life before Charles The strange phenomenon that has taken over the internet A simple exercise that gives your brain an unexpected boost The ancient road that bridged kingdoms The covert winter wave of Covid-19





What is the CSS selector for an a tag with the class of "media__link" immediately inside of an h3 tag with the class of "media__title"?



The CSS selector for an a tag with the class of "media__link" immediately inside of an h3 tag with the class of "media__title" is h3.media__title > a.media__link.

This selector uses the child combinator > to match only a tags that are immediate children of h3 tags. The h3.media__title selector targets an h3 tag with the class of "media__title" and a.media__link targets an a tag with the class of "media__link".

You can also use the space combinator to select a tags that are descendants of h3 tags. For example h3.media__title a.media__link will select all the a tags with class of "media__link" that are inside of an h3 tag with class of "media__title".

It's important to note that the above selectors assume that the website's HTML structure follows the exact format as described, in some cases the website's structure may be different.

```
# Find all headlines
headlines = soup.select('h3.media_title > a.media_link')

# Parse the HTML content
soup = BeautifulSoup(response.text, 'html.parser')

# Find all headlines
headlines = soup.select('h3.media_title > a.media_link')

# Print out the text of each headline
for headline in headlines:
    print(headline.text.strip())
```

UK will provide Challenger 2 tanks to Ukraine - Sunak Iran executes British-Iranian Akbari on spy charges Top secret documents reportedly found in Biden cache Lisa Marie to be buried next to son at Graceland Premier League: Build-up to Manchester derby as United host City Family would never forgive me if I told all - Harry Won the lotto? Get a therapist, avoid buying land Brazil top court includes Bolsonaro in riot probe Everton directors to miss match over safety threat Raducanu in 'good place to play' after injury scare EFL: Rotherham v Blackburn starts busy Saturday The simple error that 16% of us make The Versailles most people don't get to see The companies that burn out young grads The ancient road that bridged kingdoms The covert winter wave of Covid-19 Why siblings are becoming less common

```
data-bbc-source="India" data-bbc-metadata="{"CH
                D": "card::3" }">
                <div class="media__image"> - </div>
                ▼<div class="media__content">
                  ▼<h3 class="media__title">
                    ▼<a class="media___link" href="https://www.b</pre>
                     bc.com/news/world-asia-india-64807212"
                      rev="hero3|headline"> == $0
                       " The Indian-American CEO who wants to
                       be US president "
                     </a>
                    </h3>
                  <a class="media___tag tag tag--india" href="/</pre>
                    news/world/asia/india" rev="hero3|source">
... ay.block-link div.media__content h3.media__title a.media__link ...
```



How do I print out the href for an element selected by BeautifulSoup?



You can use the get() method to access the href attribute of an a tag that is selected by BeautifulSoup. Here is an example of how you can print out the href attribute of an a tag:

```
import requests
from bs4 import BeautifulSoup

# Make an HTTP request to the website
url = 'https://www.example.com/news'
response = requests.get(url)

# Parse the HTML content
soup = BeautifulSoup(response.text, 'html.parser')

# Find all links
links = soup.select('a.media__link')

# Print out the href of each link
for link in links:
    print(link.get('href'))
```

In the above code snippet, we are using the select() method to find all the a tags with class of "media__link" and then using a for loop to iterate over the links, and using the get() method to access the href attribute of each link.

It's worth noting that if the href attribute is not present in the HTML, the get() method will return None so you should check the returned value of get() method before using it.

```
[9]: import requests
                                                  # Print out the text of each headline
      from bs4 import BeautifulSoup
      url = "https://www.bbc.com/"
                                                 for link in links:
      # Make a GET request to the website
      response = requests.get(url)
                                                       print(link.text.strip())
                                                      print(link.get("href"))
      # Parse the HTML content
      soup = BeautifulSoup(response.text, "html.par
      # Find all headline elements on the page
      links = soup.select("h3.media title > a.media link")
      # Print out the text of each headline
      for link in links:
         print(link.text.strip())
         print(link.get("href"))
      People pulled from rubble as Russian missile hits flats
      /news/live/world-64274704
      Top secret documents reportedly found in Biden cache
      https://www.bbc.com/news/world-us-canada-64270906
      Family would never forgive me if I told all - Harry
      https://www.bbc.com/news/uk-64269194
      Premier League: Onana gives Everton lead, goalless elsewhere
      /sport/live/football/63881817
      Gen Z's latest surprising obsession
      https://www.bbc.com/culture/article/20230113-gen-z-and-young-millennials-surprising-obsession
```

/news/world-us-canada-64271642
Won the lotto? Get a therapist, avoid buying land

Lisa Marie to be buried next to son at Graceland

Widespread outrage after British-Iranian executed

/news/world-middle-east-64273969

Caveat: Not everything is understandable...

Anti-caveat: Al accepts infinite follow-ups

```
import requests
from bs4 import BeautifulSoup
url = "https://www.bbc.com/"
# Make a GET request to the website
response = requests.get(url)
# Parse the HTML content
soup = BeautifulSoup(response.text, "html.parser")
```

90

Why is it requests.get?



requests.get() is used to make an HTTP GET request to a specified URL. The get() method is part of the requests library, which is a popular library for sending HTTP requests in Python.

An HTTP GET request is the most common type of request and is used to retrieve data from a server. The request contains a URL, which specifies the location of the resource to be retrieved. The server then responds with the requested resource, if it exists, and a status code indicating whether the request was successful.

The get() method takes one argument, which is the URL of the resource to be retrieved. It returns a response object, which contains the server's response to the request. The response object has various properties and methods, such as text, content, status_code, etc. that can be used to access and analyze the response.

You can also use the post() method to make a HTTP POST request, put() to make a HTTP PUT request and delete() to make a HTTP DELETE request etc.

It's important to note that the requests library is not part of the python standard library, so you need to install it before using it in your code using pip install requests.



Explain that in much less technical language



The requests.get() function is used to retrieve information from a website. It is a function from a python library called requests. When you use this function, you provide the website's address, and the function sends a request to that website asking for information. The website then sends back a response which includes the information you asked for. The get() method is just one of the many HTTP requests that you can use to interact with a website. There are also post(), put(), delete(), etc.

It's important to note that requests is not part of the python standard library and you need to install it before using it in your code by running pip install requests in your command line.

Please keep in mind that some websites may block scraping or have CAPTCHAs to prevent scraping. It's important to be respectful of a website's resources and servers and check the website's terms of use before scraping.



I have a dataframe with two columns, county and social spending. I am using the following R code to plot a horizontal bar graph of each country's social spending

```
library(ggplot2)
ggplot(df) +
  aes(x=`Social Spending`, y=Country) +
  geom_col()
```

how do i make it sort the bars by social spending



You can sort the bars in descending order of social spending by adding the "desc" argument to the "arrange" function within the ggplot call. Here's the modified code

```
Copy code
SCSS
library(ggplot2)
ggplot(df) +
 aes(x=Social Spending, y=Country) +
 geom_col() +
 coord_flip() +
  labs(x = "Social Spending") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5)) +
 scale_x_continuous(labels = scales::dollar) +
 scale_y_discrete(limits = rev(df$Country)) +
  geom_text(aes(label = scales::dollar(Social Spending), x = Social Spending)
            hjust = -0.1, size = 3) +
  geom_col(data = arrange(df, desc(Social Spending)))
```

The "coord_flip" function flips the x and y axes to create a horizontal bar chart. The "labs" function sets the x-axis label to "Social Spending". The "theme_bw" function sets the plot background to white, and the "theme" function centers the plot title. The "scale_x_continuous" function formats the x-axis labels as dollar amounts. The "scale_y_discrete" function reverses the order of the y-axis categories. Finally, the "geom_text" function adds dollar values as labels to the bars, and the "geom_col" function



is there a way to do this that doesn't add as much codew



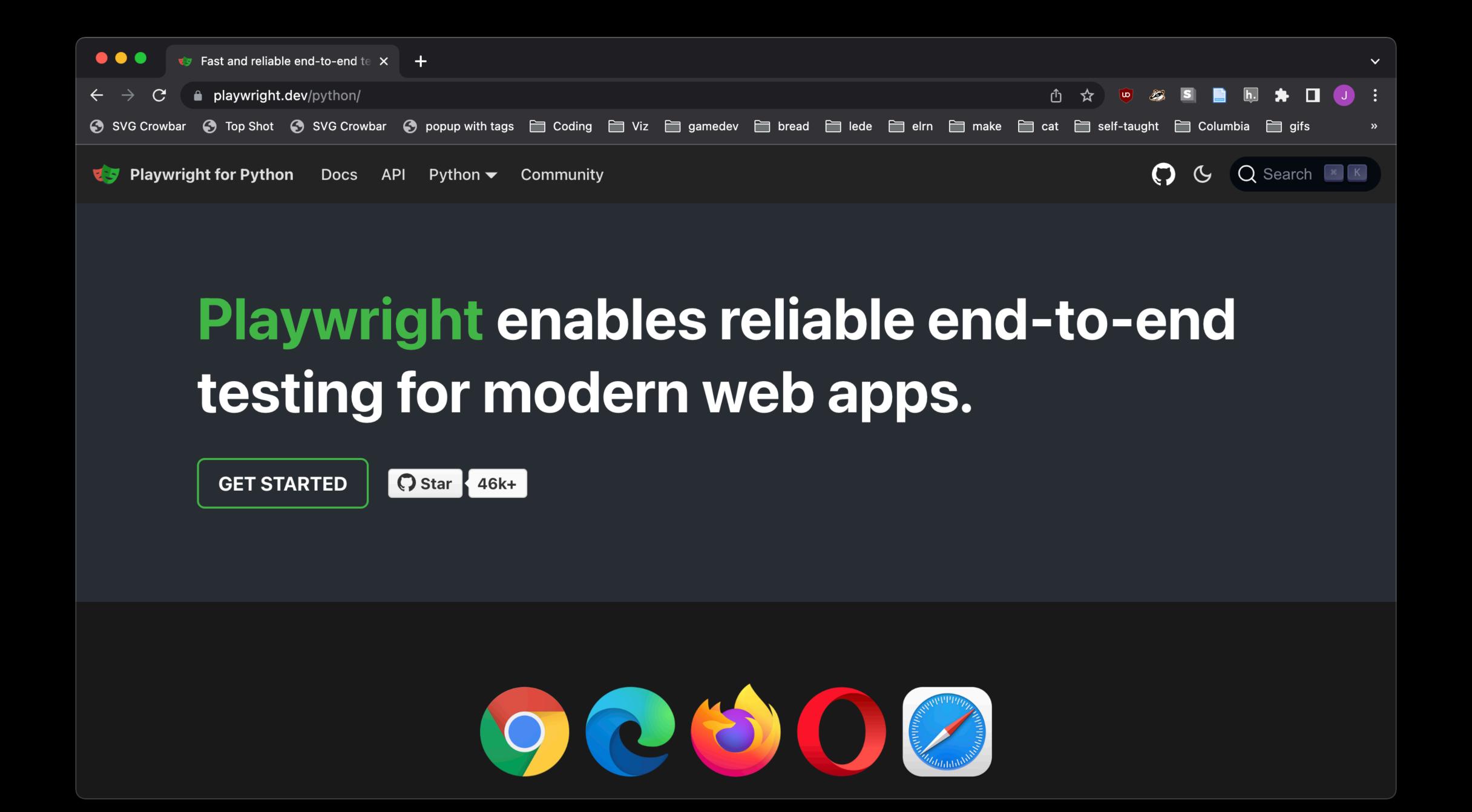
Yes, you can add the reorder function inside the aes call to sort the bars by the Social Spending column. Here's the modified code with this approach:

```
library(ggplot2)

ggplot(df) +
  aes(x = reorder(Country, SocialSpending), y = SocialSpending) +
  geom_col() +
  labs(x = "Country", y = "Social Spending") +
  theme_bw()
```

In this code, the reorder function sorts the Country variable by the SocialSpending variable, which is done inside the aes call. This creates a bar chart with bars sorted by social spending, without the need for additional geom_col or arrange calls. The labs function adds axis labels, and the theme_bw function sets a white background.

Translating your skills





Translate the code below into using playwright instead of using BeautifulSoup:



Here is an example of how you can use Playwright to scrape headlines and hrefs from a news website, instead of using BeautifulSoup:

```
Copy code
import asyncio
from playwright import async_playwright
async def main():
   # initialize Playwright
   playwright = await async_playwright()
   # create a browser context
   context = await playwright.newContext()
   # create a new page in the context
   page = await context.newPage()
   # navigate to the website
   await page.goto("https://www.bbc.com/")
   # Find all headlines and hrefs using Playwright's querySelectorAll
   headlines = await page.querySelectorAll('h3.media__title >
a.media__link')
   for headline in headlines:
       #get the text and href of each headline
       text = await page.evaluate("(el) => el.textContent", headline)
       href = await page.evaluate("(el) => el.getAttribute('href')",
headline)
       print(text, href)
   # close the browser context
   await context.close()
#run the function
asyncio.get_event_loop().run_until_complete
```

Caveat: Very much not perfect

Fixing bugs



I'm getting the error "ImportError: cannot import name 'async_playwright' from 'playwright'"



I apologize for the confusion.

You should use import playwright instead of from playwright import async_playwright in the first line of the code.

```
Copy code
import asyncio
import playwright
async def main():
    # initialize Playwright
    playwright = await playwright.async_playwright()
    # create a browser context
    context = await playwright.newContext()
    # create a new page in the context
    page = await context.newPage()
    # navigate to the website
    await page.goto("https://www.bbc.com/")
    # Find all headlines and hrefs using Playwright's querySelectorAll
   headlines = await page.querySelectorAll('h3.media__title >
a.media__link')
    for headline in headlines:
       #get the text and href of each headline
       text = await page.evaluate("(el) => el.textContent", headline)
       href = await page.evaluate("(el) => el.getAttribute('href')",
headline)
        print(text, href)
    # close the browser context
    await context.close()
#run the function
asyncio.get_event_loop().run_until_complete(main())
```

```
RuntimeError
                                           Traceback (most recent call las
t)
/var/folders/_m/b8tjbm6n4zs1q2mvjvg25x1m0000gn/T/ipykernel_59147/15954043
64.py in <cell line: 30>()
     28
     29 #run the function
---> 30 asyncio.get_event_loop().run_until_complete(main())
~/.pyenv/versions/3.10.3/Library/Frameworks/Python.framework/Versions/3.1
0/lib/python3.10/asyncio/base_events.py in run_until_complete(self, futur
e)
                11 11 11
    620
                self. check closed()
    621
                self._check_running()
--> 622
    623
    624
                new_task = not futures.isfuture(future)
~/.pyenv/versions/3.10.3/Library/Frameworks/Python.framework/Versions/3.1
0/lib/python3.10/asyncio/base_events.py in _check_running(self)
            def _check_running(self):
    580
                if self.is_running():
    581
                    raise RuntimeError('This event loop is already runnin
--> 582
g')
                if events._get_running_loop() is not None:
    583
    584
                    raise RuntimeError(
RuntimeError: This event loop is already running
```



I'm getting the error "RuntimeError: This event loop is already running"



The error "RuntimeError: This event loop is already running" is raised because the event loop is already running and you can't create a new one.

You could fix this error by removing the line asyncio.get_event_loop().run_until_complete(main()) from the code, because it is trying to run the event loop again, and the event loop is already running.

```
Copy code
import asyncio
import playwright
async def main():
    # initialize Playwright
    playwright = await playwright.async_playwright()
    # create a browser context
    context = await playwright.newContext()
    # create a new page in the context
    page = await context.newPage()
    # navigate to the website
    await page.goto("https://www.bbc.com/")
   # Find all headlines and hrefs using Playwright's querySelectorAll
    headlines = await page.querySelectorAll('h3.media__title >
a.media__link')
    for headline in headlines:
        #get the text and href of each headline
        text = await page.evaluate("(el) => el.textContent", headline)
        href = await page.evaluate("(el) => el.getAttribute('href')",
headline)
        print(text, href)
   # close the browser context
    await context.close()
#run the function
await main()
```

```
UnboundLocalError
                                          Traceback (most recent call las
t)
/var/folders/_m/b8tjbm6n4zs1q2mvjvg25x1m0000gn/T/ipykernel_59147/59500773
4.py in <cell line: 30>()
     28
     29 #run the function
---> 30 await main()
/var/folders/ m/b8tjbm6n4zs1q2mvjvg25x1m0000gn/T/ipykernel 59147/59500773
4.py in main()
      4 async def main():
       # initialize Playwright
       playwright = await playwright.async_playwright()
---> 6
            # create a browser context
UnboundLocalError: local variable 'playwright' referenced before assignme
nt
```



I'm getting the error "UnboundLocalError: local variable 'playwright' referenced before assignment"



The error "UnboundLocalError: local variable 'playwright' referenced before assignment" is raised because the variable playwright is not defined before it is used.

You should define playwright variable before you use it.

```
Copy code
import asyncio
import playwright
async def main():
   # initialize Playwright
   _playwright = await playwright.async_playwright()
   # create a browser context
   context = await _playwright.newContext()
   # create a new page in the context
   page = await context.newPage()
   # navigate to the website
   await page.goto("https://www.bbc.com/")
   # Find all headlines and hrefs using Playwright's querySelectorAll
   headlines = await page.querySelectorAll('h3.media__title >
a.media__link')
   for headline in headlines:
       #get the text and href of each headline
        text = await page.evaluate("(el) => el.textContent", headline)
       href = await page.evaluate("(el) => el.getAttribute('href')",
headline)
        print(text, href)
   # close the browser context
   await context.close()
#run the function
await main()
```



This doesn't work, but online I found the code "from playwright.async_api import async_playwright"



You are correct, my apologies for the confusion.

The correct import statement for the async_playwright function from the playwright library is from playwright.async_api import async_playwright.

```
Copy code
import asyncio
from playwright.async_api import async_playwright
async def main():
    # initialize Playwright
    _playwright = await async_playwright()
    # create a browser context
    context = await _playwright.newContext()
    # create a new page in the context
    page = await context.newPage()
    # navigate to the website
    await page.goto("https://www.bbc.com/")
    # Find all headlines and hrefs using Playwright's querySelectorAll
    headlines = await page.querySelectorAll('h3.media__title >
a.media__link')
    for headline in headlines:
        #get the text and href of each headline
        text = await page.evaluate("(el) => el.textContent", headline)
        href = await page.evaluate("(el) => el.getAttribute('href')",
headline)
        print(text, href)
    # close the browser context
   await context.close()
#run the function
await main()
```



I'm getting the error "TypeError: object PlaywrightContextManager can't be used in 'await' expression"



The error "TypeError: object PlaywrightContextManager can't be used in 'await' expression" is raised because you are trying to use await on a non-awaited object, such as playwright.async_playwright().

You should first call async_playwright() function to get an awaitable object, and then use await on the returned object.

Here is the corrected code:

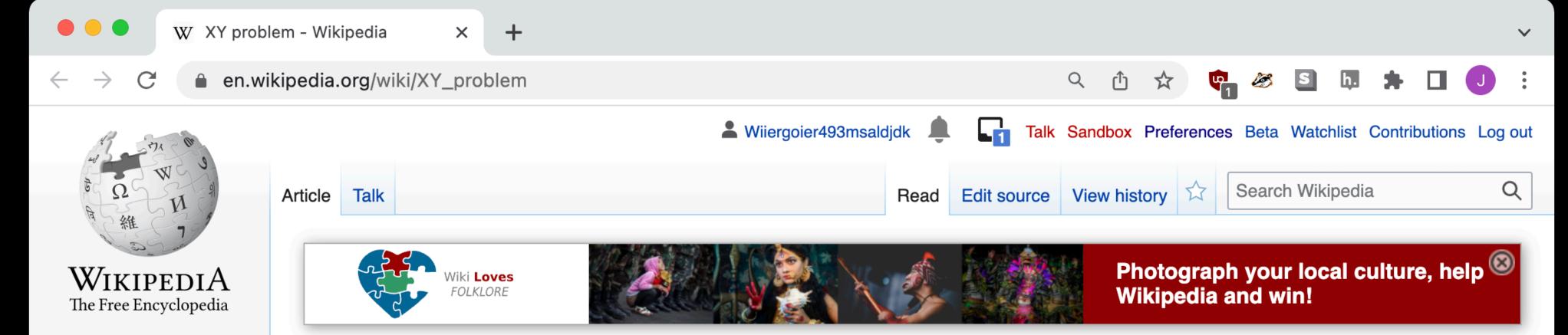
It's important to note that Playwright is a browser automation tool, so it runs slower than parsing HTML directly and it may not work with all the websites, some websites may block scraping or have CAPTCHAs to prevent scraping. It's important to be respectful of a website's resources and servers and check the website's terms of use before scraping.



Caveat: It's remarkably confident even when it's wrong, and has infinite patience with itself and its errors.

(and hopes that you do, too)

Caveat: The trap of the XY Problem



XY problem

Main page

Contents

Current events

Random article

About Wikipedia

Contact us

Contribute

Learn to edit

Upload file

Tools

Community portal

Recent changes

What links here

Related changes

Special pages

Permanent link

Cite this page

Wikidata item

Print/export

Download as PDF

Page information

Donate

Help

From Wikipedia, the free encyclopedia

The **XY problem** is a communication problem encountered in help desk, technical support, software engineering, or customer service situations where the question is about an end user's attempted solution (Y) rather than the root problem itself (X).^[1]

The XY problem obscures the real issues and may even introduce secondary problems that lead to miscommunication, resource mismanagement, and sub-par solutions. The solution for the support personnel is to ask probing questions as to why the information is needed in order to identify the root problem X and redirect the end user away from an unproductive path of inquiry. [2][3]

Contents [hide]

- 1 Terminology
- 2 Examples
- 3 See also
- 4 References

Terminology [edit source]

The term XY problem was implicitly coined by Eric S. Raymond in How To Ask Questions The Smart Way when he wrote "How can I use X to do Y?" in the "Questions Not To Ask" section (note that in this original version X and Y are swapped):^[4]

Q: How can I use X to do Y?

A: If what you want is to do Y, you should ask that question without pre-supposing the use of a method that may not be appropriate.

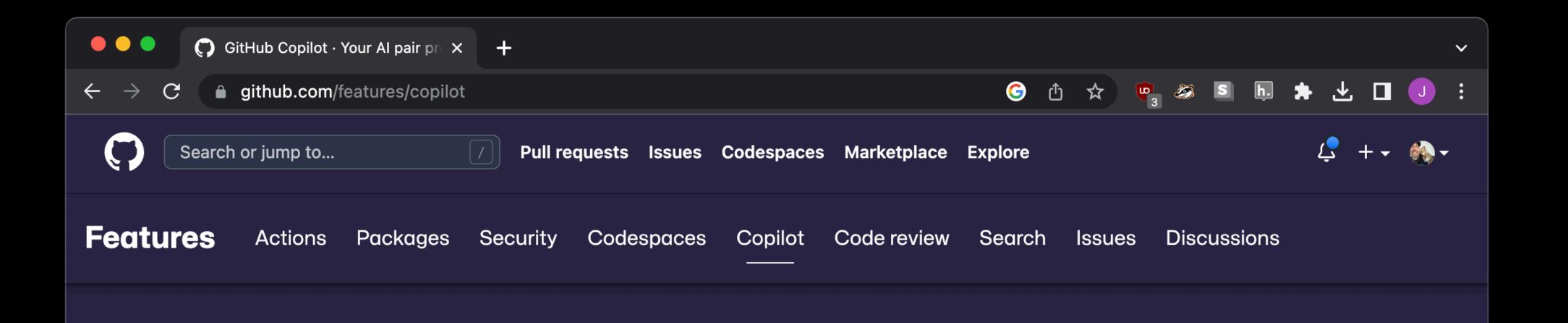
Questions of this form often indicate a person who is not merely ignorant about X, but confused about what problem V they are

The actual answer

https://jonathansoma.com/everything/scraping/selenium-playwright-conversion/https://jonathansoma.com/everything/scraping/loops-in-playwright/

```
import asyncio
from playwright.async_api import async_playwright
playwright = await async_playwright().start()
browser = await playwright.chromium.launch(headless = False)
page = await browser.new page()
await page.goto('https://www.bbc.com')
# Find all headlines and hrefs using Playwright's querySelectorAll
headlines = await page.query_selector_all('h3.media__title > a.media__link')
for headline in headlines:
    #get the text and href of each headline
   text = await headline.text_content()
    href = await headline.get_attribute('href')
    print(text.strip(), href)
# close the browser context
await browser.close()
```

Bonus time: we don't need to work in a chatbox

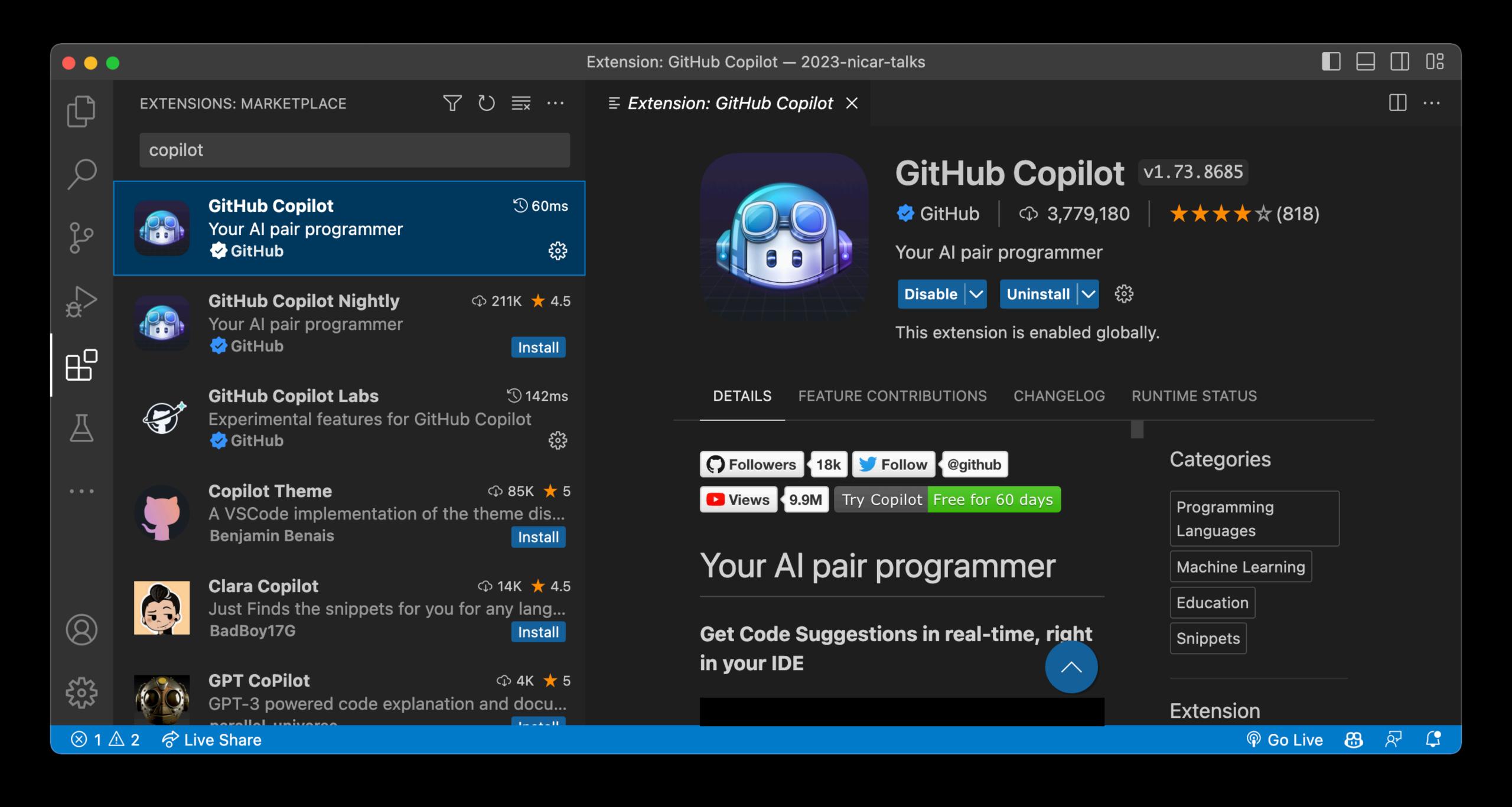


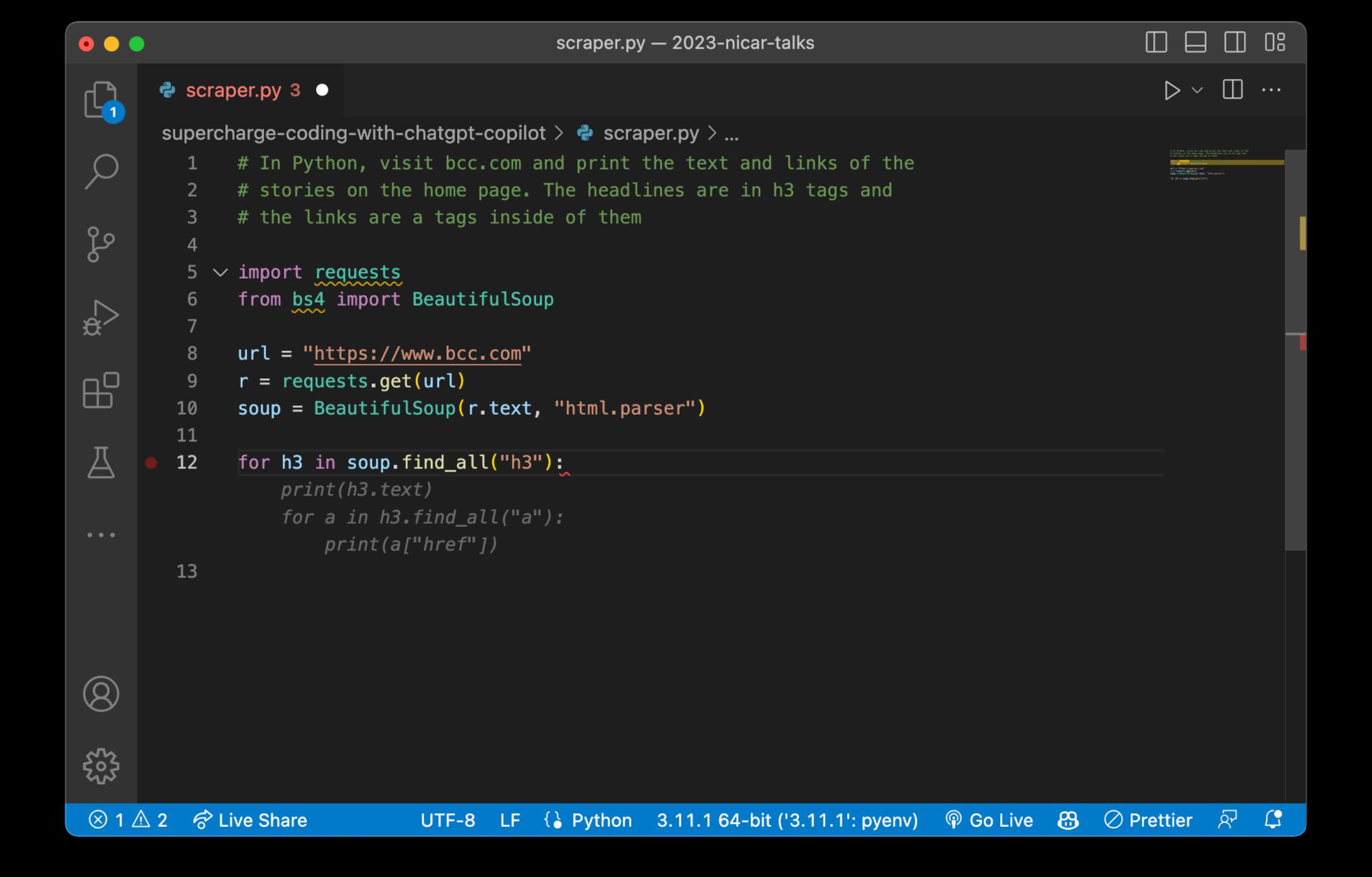
Your Al pair programmer

GitHub Copilot uses the OpenAl Codex to suggest code and entire functions in real-time, right from your editor.

Get Copilot for Business >

Compare plans

















A talk about AI-assisted programming

I'm a big booster for AI-assisted programming. Even if you get stuck along the way, it's a "force multiplier" for the knowledge that you have. It will take a tiny tiny bit of foundational skill that you have and (potentially) expand it into an **infinite universe** of remarkable accomplishments.

Some of my favorite tools are ChatGPT for discussion-oriented coding and GitHub Copilot for autocomplete. If you're in VS Code, GitHub Copilot will not only complete your .py files, but also your .md files. I'm using it right now to write this talk.

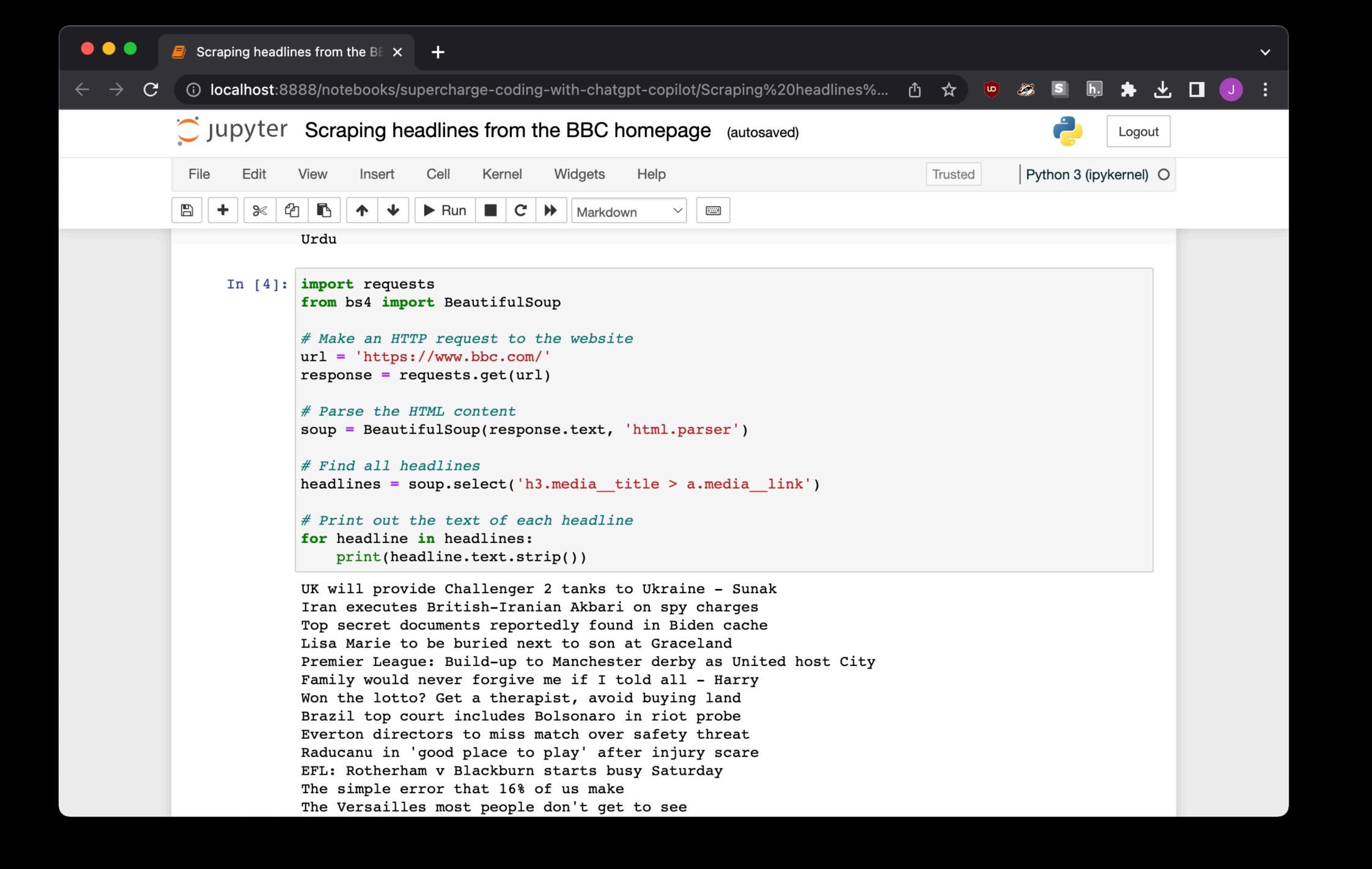


0 0 0

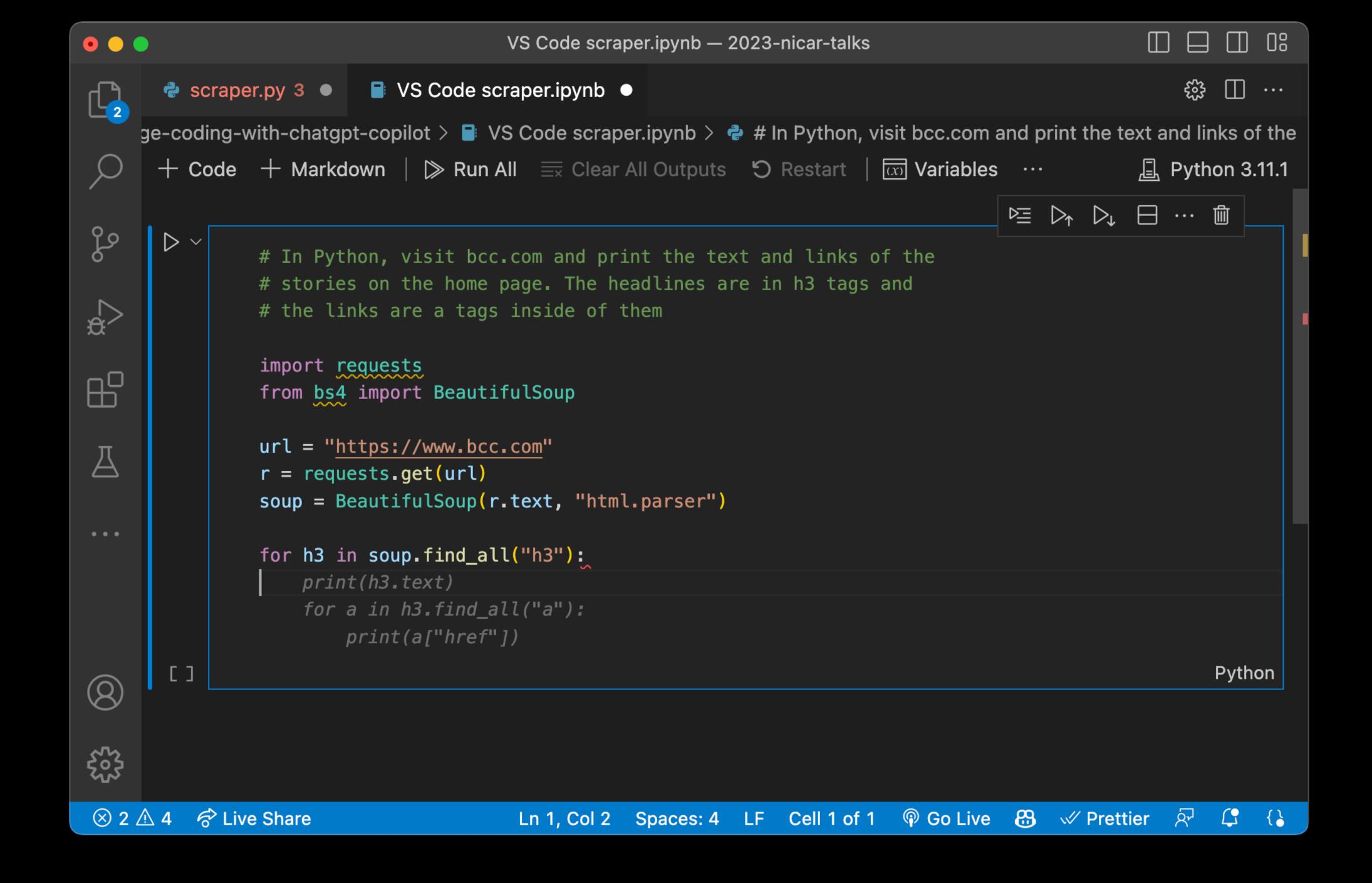




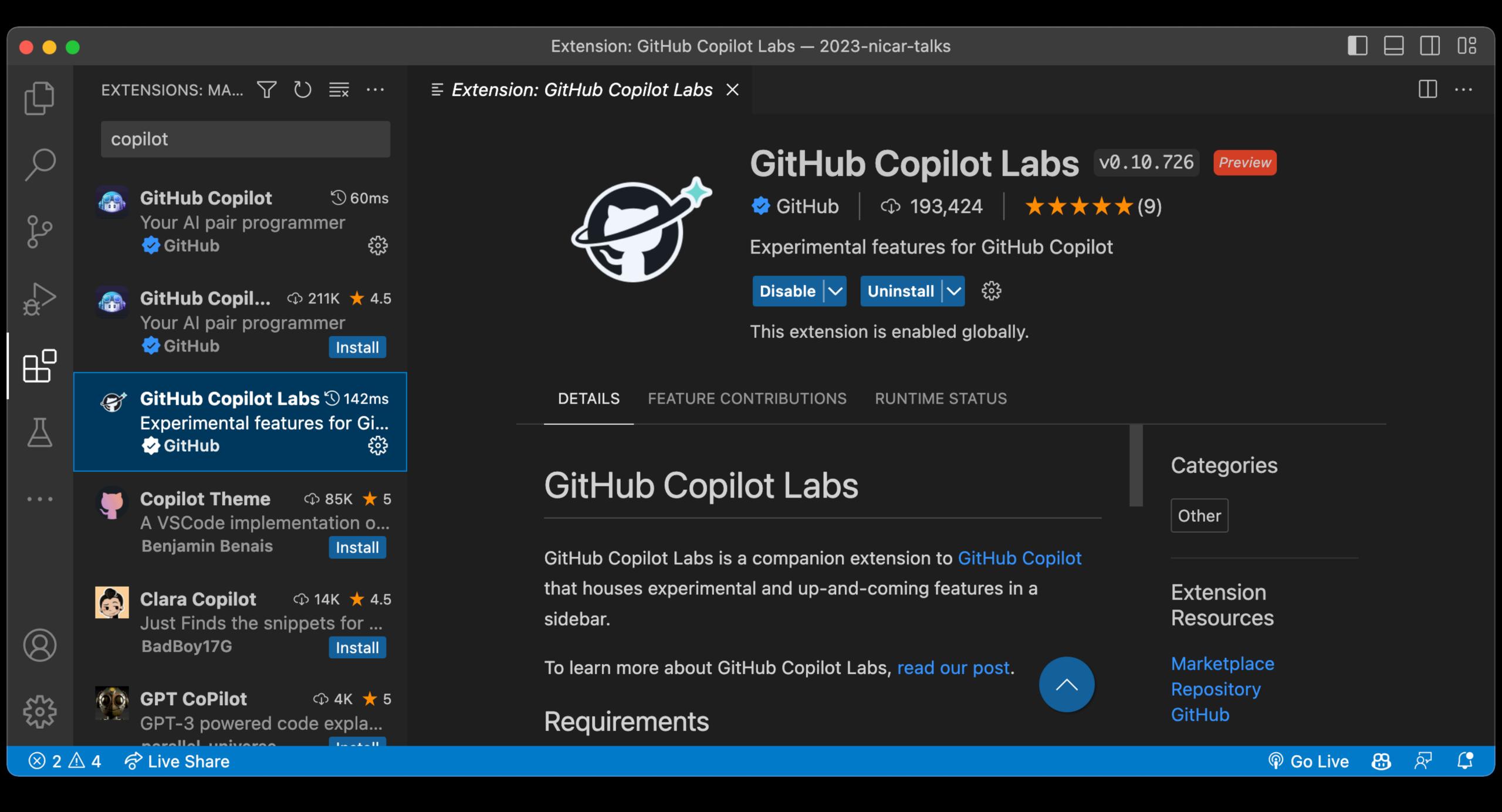
Caveat: GitHub Copilot only officially works in a few editors: Visual Studio, VS Code, Jetbrains, Neovim

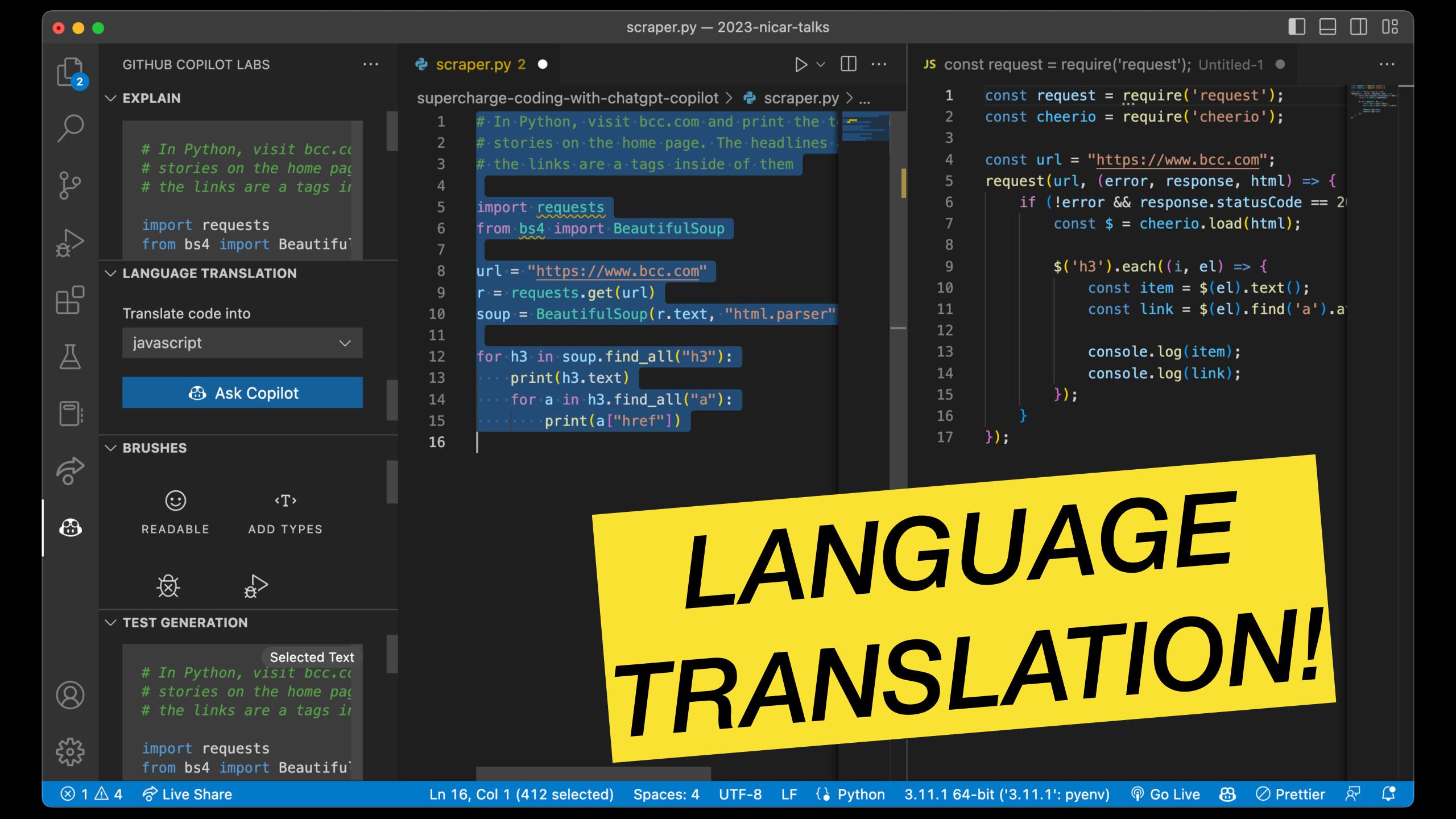


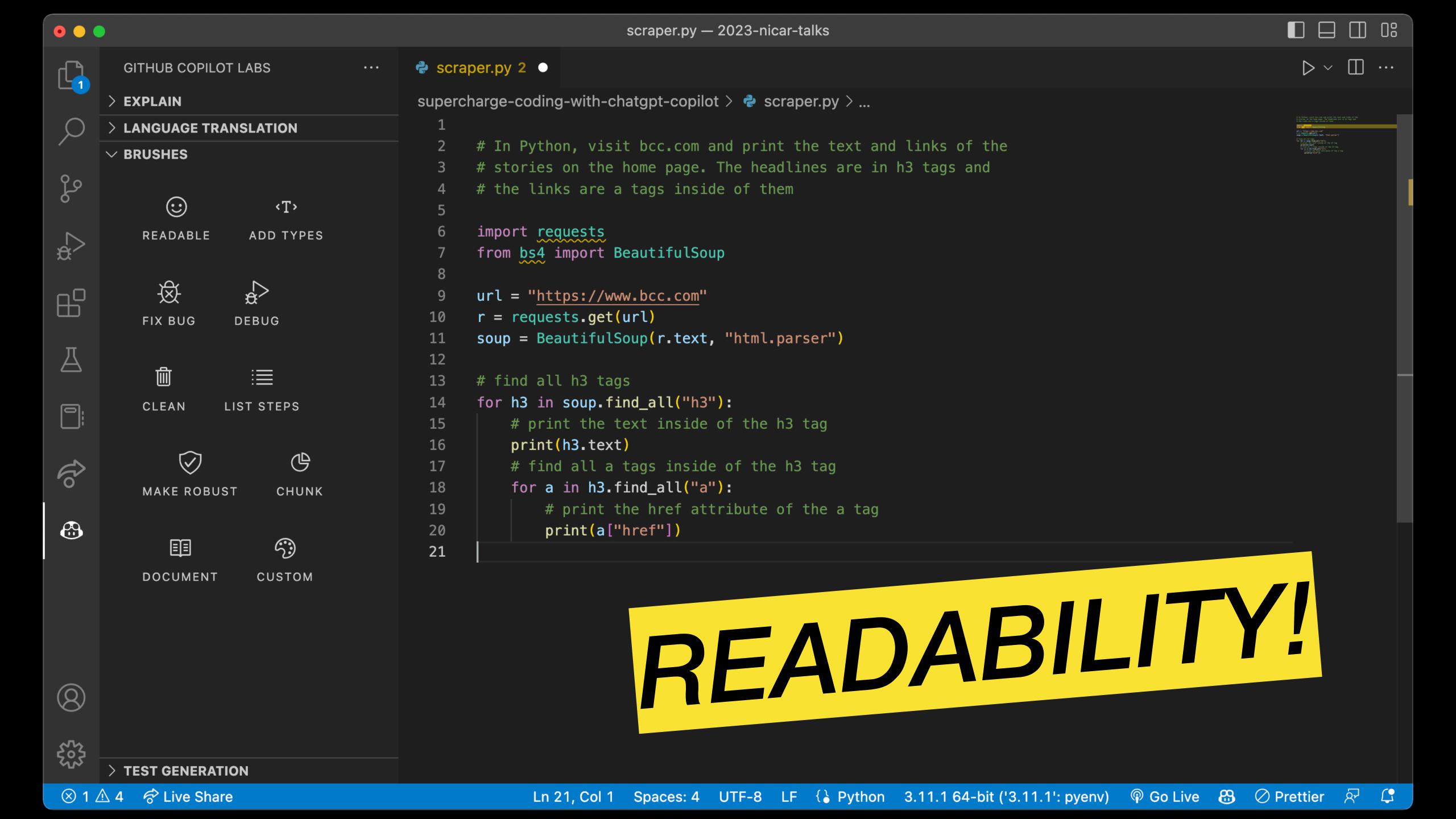
Fortunately, VS Code can do anything!

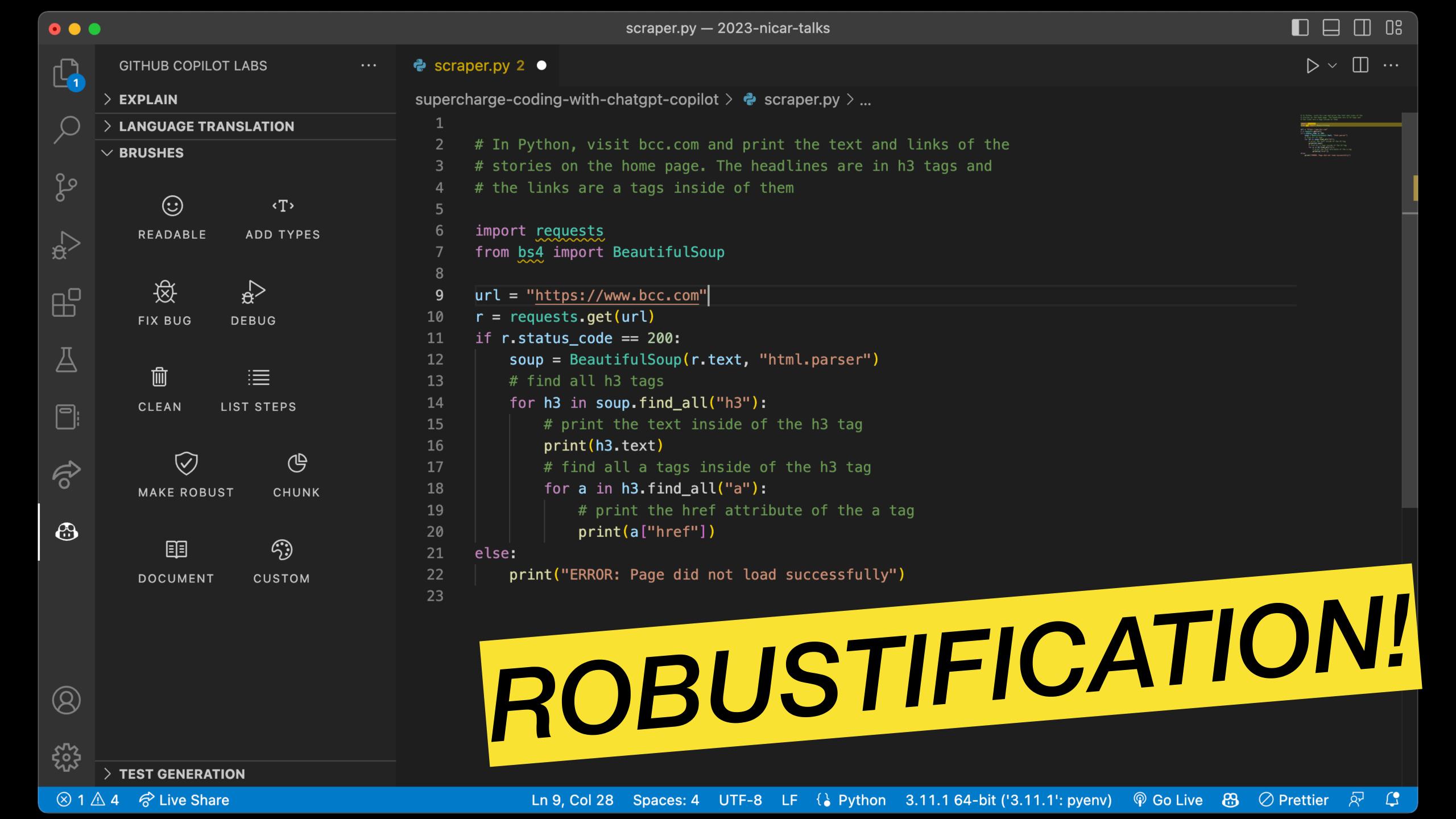


But wait there's more!









Privacy – Copilot for Business

What data does Copilot for Business collect?

GitHub Copilot relies on file content and additional data to work. It collects data to provide the service, some of which is then retained for further analysis and product improvements.

Copilot for Business collects data as described below:

User Engagement Data

When you use GitHub Copilot it will collect usage information about events generated when interacting with the IDE or editor. These events include user edit actions like completions accepted and dismissed, and error and general usage data to identify metrics like latency and features engagement. This information may include personal data, such as pseudonymous identifiers.

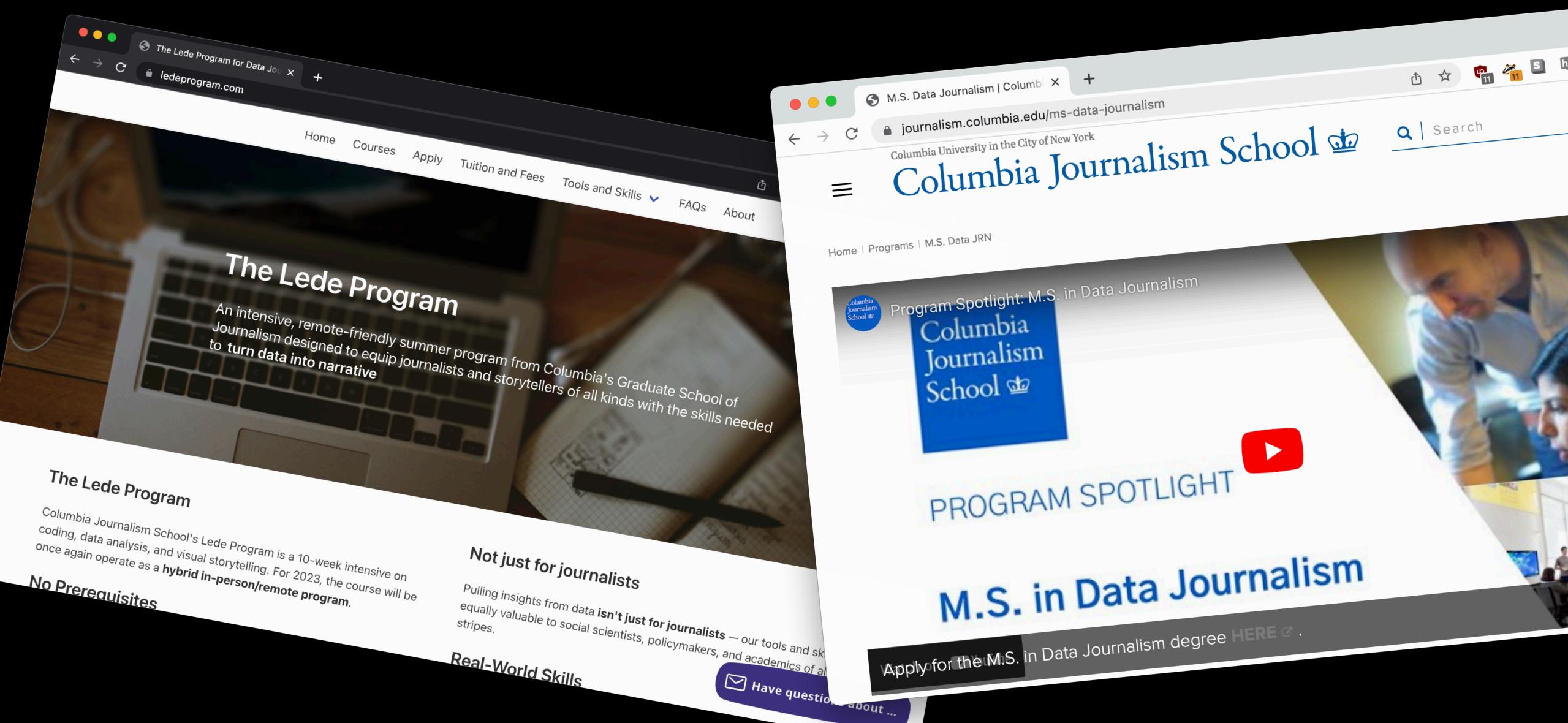
Code Snippets Data

GitHub Copilot transmits snippets of your code from your IDE to GitHub to provide Suggestions to you. Code snippets data is only transmitted in real-time to return Suggestions, and is discarded once a Suggestion is returned. Copilot for Business does not retain any Code Snippets Data.

How is the data in Copilot for Business used and shared?

We live in a crazy future

Jonathan Soma, js4571@columbia.edu, @dangerscarf, <u>ledeprogram.com</u>



Supercharge your coding workflow with ChatGPT and GitHub Copilot

#NICAR23

Jonathan Soma, js4571@columbia.edu, @dangerscarf, <u>ledeprogram.com</u> <u>jonathansoma.com/supercharge-coding-ai-tools</u>