

- Web Scraper of Trending Repositories on Github

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import base64
```

```
topics_url = 'https://github.com/topics'
```

```
response = requests.get(topics_url)
content = response.text
```

```
content[:1000]
```

```
<!DOCTYPE html><html lang="en" data-color-mode="auto" data-light-theme="light" data-dark-theme="dark" data-all-  
-animated-images="system" data-all-link-underlines="true"><head><meta charset="utf-8"><link rel="dns-p  
refetch" href="https://github.githubassets.com"><link rel="dns-prefetch" href="https://avatars.githubusercontent.com"><link  
rel="dns-prefetch" href="https://github-cloud.s3.amazonaws.com"><link rel="dns-prefetch" href="https://user-images.githubusercontent.  
com"/><link rel="preconnect" href="https://github.githubassets.com" crossorigin"><link rel="preconnect" href="https://  
avatars.githubusercontent.com"><link rel="preconnect" href="https://github-cloud.s3.amazonaws.com" crossorigin="all" rel="stylesheet" href="https://github.githuba
```

```
with open("webpage.htm", 'w') as f :
    f.write(content)
```

```
doc = BeautifulSoup(content, 'html.parser')
```

```
topic_titles = doc.find_all('p', {'class' : 'f3 lh-condensed mb-0 mt-1 Link--primary'})
topic_titles
```

```
[<p class="f3 lh-condensed mb-0 mt-1 Link--primary">3D</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Ajax</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Algorithm</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Amp</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Android</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Angular</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Ansible</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">API</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Arduino</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">ASP.NET</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Awesome Lists</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Amazon Web Services</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Azure</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Babel</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Bash</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Bitcoin</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Bootstrap</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Bot</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">C</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Chrome</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Chrome extension</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Command-line interface</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Clojure</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Code quality</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Code review</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Compiler</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Continuous integration</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">C++</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Cryptocurrency</p>,<br><p class="f3 lh-condensed mb-0 mt-1 Link--primary">Crystal</p>]
```

```
a = topic_titles[0].parent.parent.find_all('a',{'class':'no-underline flex-grow-0'}, href=True)
a[0]['href']
```

→ `'/topics/3d'`

```
topic_titles_ = []
topic_urls = []
for tag in topic_titles :
    topic_titles_.append(tag.text)
    topic_urls.append("https://github.com"+tag.parent.parent.find_all('a',{'class':'no-underline flex-grow-0'}, href=True)[0]['href'])
topic_urls
```

```
➦ ['https://github.com/topics/3d',  
   'https://github.com/topics/ajax',  
   'https://github.com/topics/algorithm',  
   'https://github.com/topics/amphp',
```

```
'https://github.com/topics/android',
'https://github.com/topics/angular',
'https://github.com/topics/ansible',
'https://github.com/topics/api',
'https://github.com/topics/arduino',
'https://github.com/topics/aspnet',
'https://github.com/topics/awesome',
'https://github.com/topics/aws',
'https://github.com/topics/azure',
'https://github.com/topics/babel',
'https://github.com/topics/bash',
'https://github.com/topics/bitcoin',
'https://github.com/topics/bootstrap',
'https://github.com/topics/bot',
'https://github.com/topics/c',
'https://github.com/topics/chrome',
'https://github.com/topics/chrome-extension',
'https://github.com/topics/cli',
'https://github.com/topics/clojure',
'https://github.com/topics/code-quality',
'https://github.com/topics/code-review',
'https://github.com/topics/compiler',
'https://github.com/topics/continuous-integration',
'https://github.com/topics/cpp',
'https://github.com/topics/cryptocurrency',
'https://github.com/topics/crystal']
```

```
df = pd.DataFrame({'topic_title': topic_titles_, 'topic_url': topic_urls})
```

```
df.head()
```

	topic_title	topic_url
0	3D	https://github.com/topics/3d
1	Ajax	https://github.com/topics/ajax
2	Algorithm	https://github.com/topics/algorithm
3	Amp	https://github.com/topics/amphp
4	Android	https://aithub.com/topics/android

Next steps:

[Generate code with df](#)
[View recommended plots](#)
[New interactive sheet](#)

```
def get_info_from_topic(url) :
    response = requests.get(url).text
    content = BeautifulSoup(response, 'html.parser')
    usernames = []
    repos = []
    repo_urls = []
    stars = []
    repositories = content.find_all('h3',{'class': "f3 color-fg-muted text-normal lh-condensed"})
    stars_raw = content.find_all('span', {'id' : "repo-stars-counter-star"})
    for i in range(len(repositories)) :
        repo = repositories[i]
        usernames.append(repo.find_all('a')[0].text.strip())
        repos.append(repo.find_all('a')[1].text.strip())
        repo_urls.append("https://github.com"+repo.find_all('a')[1]['href'])
        star = stars_raw[i].text.strip()
        if star[-1]=='k' :
            star = star[:-1]
            star = float(star)
            star = int(star*1000)
        else : star = int(star)
        stars.append(star)
    return {
        "usernames" : usernames,
        "repos" : repos,
        "stars" : stars,
        "repo_urls" : repo_urls,
    }
```

```
df = pd.DataFrame(get_info_from_topic("https://github.com/topics/Android"))
df.head()
```

	usernames	repos	stars	repo_urls
0	flutter	flutter	165000	https://github.com/flutter/flutter
1	facebook	react-native	119000	https://github.com/facebook/react-native
2	justjavac	free-programming-books-zh_CN	111000	https://github.com/justjavac/free-programming-...
3	Genymobile	scrcpy	110000	https://github.com/Genymobile/scrcpy
4	Hack-with-Github	Awesome-Hackin...	83900	https://github.com/Hack-with-Github/Awesome-Ha...

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

✓ save the data in to csv

```
for i in range(len(topic_titles_)) :
    df = pd.DataFrame(get_info_from_topic(topic_urls[i]))
    df.to_csv(topic_titles_[i])
```

scrap the readme file of repo.

```
import requests
from bs4 import BeautifulSoup

# URL of the README page
url = "https://github.com/bumptech/glide"

# Fetch the README page
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Extract the title (usually in <h1> or <h2>)
title = soup.find('h1').get_text(strip=True)

# Extract the description (first <p> after title or within a specific section)
description = soup.find('p').get_text(strip=True)

# Extract all paragraphs (for main content)
content_paragraphs = soup.find_all('p')
content = [para.get_text(strip=True) for para in content_paragraphs]

# Extract links from the content
links = []
for link in soup.find_all('a', href=True):
    links.append((link.get_text(strip=True), link['href']))

# Display the extracted content
print("Title:", title)
```

```
➦ Title: Search code, repositories, users, issues, pull requests...
```

```
print("Content:", content)
```

```
➦ ty libraries plug in to Google's Volley project or Square's OkHttp library instead.", "Glide's primary focus is on making scrolling
```

```
print("Links:", links)
```

```
➦ Links: [('Skip to content', '#start-of-content'), ('', '/'), ('Sign in', '/login?return_to=https%3A%2F%2Fgithub.com%2Fbumptech%2Fgli
```

merge both logic

```

def scrape_readme(url):
    response = requests.get(url)

    # Check if the request was successful
    if response.status_code != 200:
        print(f"Failed to fetch {url}: {response.status_code}")
        return None, "No description found", [], []

    soup = BeautifulSoup(response.text, 'html.parser')

    # Extract the title (usually in <h1>)
    title_tag = soup.find('h1')
    title = title_tag.get_text(strip=True) if title_tag else "No title found"

    # Find the first <p> after the title
    content_paragraphs = soup.find_all('p')

    # Extract the description (first <p> after title or within a specific section)
    description = soup.find('p').get_text(strip=True)

    # Extract all paragraphs (for main content)

    content = [para.get_text(strip=True) for para in content_paragraphs]

    # Extract links from the README content
    links = []
    for link in soup.find_all('a', href=True):
        links.append((link.get_text(strip=True), link['href']))

    return title, description, content, links


def scrape_trending_repositories(limit=100): # Set a default limit
    topics_url = 'https://github.com/topics'
    response = requests.get(topics_url)
    content = response.text

    doc = BeautifulSoup(content, 'html.parser')
    topic_titles = doc.find_all('p', {'class': 'f3 lh-condensed mb-0 mt-1 Link--primary'})

    topic_titles_ = []
    topic_urls = []

    for tag in topic_titles:
        topic_titles_.append(tag.text)
        topic_urls.append("https://github.com" + tag.parent.parent.find_all('a', {'class': 'no-underline flex-grow-0'}, href=True)[0]['href'])

    df_topics = pd.DataFrame({'topic_title': topic_titles_, 'topic_url': topic_urls})

    all_repo_data = []

    # Loop through each topic URL to get repository info
    for topic_url in topic_urls:
        response = requests.get(topic_url)
        content = BeautifulSoup(response.text, 'html.parser')
        repositories = content.find_all('h3', {'class': "f3 color-fg-muted text-normal lh-condensed"})

        for repo in repositories:
            if len(all_repo_data) >= limit: # Check if the limit is reached
                break

            user = repo.find_all('a')[0].text.strip()
            repo_name = repo.find_all('a')[1].text.strip()
            repo_url = "https://github.com" + repo.find_all('a')[1]['href']

            # Get the stars (if available)
            star_tag = repo.find_next('span', {'id': 'repo-stars-counter-star'})
            if star_tag:
                star_text = star_tag.text.strip().replace(',', '')
                if 'k' in star_text:
                    star_text = star_text.replace('k', '') # Remove the 'k'
                    stars = int(float(star_text) * 1000) # Convert to integer after multiplying by 1000
                else:
                    stars = int(float(star_text)) # Convert to integer directly
            else:
                stars = 0

            # Scrape the README content for the repository

```

```
readme_title, readme_description, readme_content, readme_links = scrape_readme(repo_url)

# Collect all data in a dictionary
repo_data = {
    'username': user,
    'repo': repo_name,
    'stars': stars,
    'repo_url': repo_url,
    'readme_content': readme_content,

    'readme_links': readme_links
}
all_repo_data.append(repo_data)

return all_repo_data

df_repos = pd.DataFrame(scrape_trending_repositories())
df_repos.head()
```



1 to 5 of 5 entries

Filter



index	username	repo	stars	repo_url	readme_content	readme_description
0	mrdoob	three.js	102000	https://github.com/mrdoob/three.js	We read every piece of feedback, and take your input very seriously.,To see all available qualifiers, see ourdocumentation.,JavaScript 3D Library.,The aim of the project is to create an easy-to-use, lightweight, cross-browser, general-purpose 3D library. The current builds only include a WebGL renderer but WebGPU (experimental), SVG and CSS3D renderers are also available as addons.,Examples—Docs—Manual—Wiki—Migrating—Questions—Forum—Discord,This code creates a scene, a camera, and a geometric cube, and it adds the cube to the scene. It then creates aWebGLrenderer for the scene and camera, and it adds that viewport to thedocument.bodyelement. Finally, it animates the cube within the scene for the camera.,If everything goes well, you should seethis.,Cloning the repo with all its history results in a ~2 GB download. If you don't need the whole history you can use thedepthparameter to significantly reduce download size.,Releases,JavaScript 3D Library.	We read every piece of feedback, and take your input very seriously.
1	pmndrs	react-three-fiber	27300	https://github.com/pmndrs/react-three-fiber	We read every piece of feedback, and take your input very seriously.,To see all available qualifiers, see ourdocumentation.,ch A React renderer for Three.js.,,react-three-fiber is aReact rendererfor threejs.,Build your scene declaratively with re-usable, self-contained components that react to state, are readily interactive and can participate in React's ecosystem.,None. Everything that works in Threejs will work here without exception.,No. There is no overhead. Components render outside of React. It outperforms Threejs in scale due to React's scheduling abilities.,Yes. It merely expresses Threejs in JSX,<mesh />dynamically turns into new THREE.Mesh(). If a new Threejs version adds, removes or changes features, it will be available to you instantly without depending on updates to this library.,Live demo: https://codesandbox.io/s/icy-tree-brnsm?file=/src/App.tsx ,This example relies on react 18 and usesexpo-cli, but you can create a bare project with their template or with thereact-nativeCLI.,Some configuration may be required to tell the Metro bundler about your assets if you useLoaderor Drei abstractions likeuseGLTFanduseTexture.,Visitdocs.pmnd.rs,You need to be versed in both React and Threejs before rushing into this. If you are unsure about React consult the officialReact docs, especiallythe section about hooks. As for Threejs, make sure you at least glance over the following links:,Some helpful material:,There is a vibrant and extensive eco system around three-fiber, full of libraries, helpers and abstractions.,Usage Trend of the @react-three Family,A small selection of companies and projects relying on three-fiber.,If you like this project, please consider helping out. All contributions are welcome as well as donations toOpencollective, or in cryptoBTC: 36fuguTPxGCNnYZSRdgdh6Ea94brCAjMbH,ETH: 0x6E3f79Ea1d0dcedeb33D3fC6c34d2B1f156F2682.,Thank you to all our backers! 🙌,This project exists thanks to all the people who contribute.,ch A React renderer for Three.js	We read every piece of feedback, and take your input very seriously.
2	libgdx	libgdx	23300	https://github.com/libgdx/libgdx	We read every piece of feedback, and take your input very seriously.,To see all available qualifiers, see ourdocumentation.,Desktop/Android/HTML5/iOS Java game development framework,,,,,libGDXis a cross-platform Java game development framework based on OpenGL (ES), designed for Windows, Linux, macOS, Android, web browsers, and iOS.It provides a robust and well-established environment for rapid prototyping and iterative development. Unlike other frameworks, libGDX does not impose a specific design or coding style, allowing you the freedom to create games according to your preferences.,libGDX is released under theApache 2.0 License, offering unrestricted usage in both commercial and non-commercial projects. While not mandatory, we appreciate any credit given to libGDX when you release a game or app using it. Check out ourshowcasefor a selection of popular libGDX-powered games. With libGDX, you gain access to a comprehensive set of tools and features to develop multi-platform 2D and 3D games using Java.,Moreover, libGDX boasts a vibrant third-party ecosystem, with numerous tools and libraries that streamline development tasks. Explore theawesome-libgdxrepository for a curated list of libGDX-centered libraries, serving as an excellent starting point for newcomers in the libGDX community.,Thanks to Gradle, you can easily set up libGDX without the need to download the framework itself. Your favorite build tool can handle everything for you. Additionally, we offer a convenient setup tool that automates project creation and downloads all the necessary components. Check out ourwebsitefor instructions on getting started or refer to our comprehensive wiki.,We provide the libGDX javadocsonline for easy reference. Additionally, the javadocs are bundled with every libGDX distribution, ensuring smooth integration with your preferred IDE.,Stay up to date with the latest libGDX news by following our blog. For engaging discussions and support, join our	We read every piece of feedback, and take your input very seriously.

					<p>officiallibGDX Discord.,Use theIssue Trackerhere on GitHub to report any issues you encounter. Before submitting, please read ourGetting Helpguide, which walks you through the process of reporting an issue effectively.,libGDX benefits greatly from contributions made by our dedicated developer community. We appreciate any assistance in making libGDX even better. Check out theCONTRIBUTING.mdfile for details on how to contribute. Note that contributing involves working directly with libGDX's source code, a process that regular users do not typically undertake. Refer to theWorking with the Sourcearticle for guidance.,You can also support our infrastructure (build server, web server, test devices) by contributing financially through ourPatreon!,Desktop/Android/HTML5/iOS Java game development framework</p>		
3	BabylonJS	Babylon.js	23200	https://github.com/BabylonJS/Babylon.js	<p>We read every piece of feedback, and take your input very seriously.,To see all available qualifiers, see ourdocumentation.,Babylon.js is a powerful, beautiful, simple, and open game and rendering engine packed into a friendly JavaScript framework.,Getting started? Play directly with the Babylon.js API using ourplayground. It also contains a lot of samples to learn how to use it.,Any questions?Here is our officialforum., ⚠️ WARNING: The CDN should not be used in production environments. The purpose of our CDN is to serve Babylon packages to users learning how to use the platform or running small experiments. Once you've built an application and are ready to share it with the world at large, you should serve all packages from your own CDN.,For the preview release, use the following URLs:A list of additional references can be foundhere.,BabylonJS and its modules are published on npm with full typing support. To install, use:,alternatively, you can now rely on ourES6 packages. Using the ES6 version will allow tree shaking among other bundling benefits.,This will allow you to import BabylonJS entirely using:,or individual classes using:,If using TypeScript, don't forget to add 'babylonjs' to 'types' intsconfig.json:,To add a module, install the respective package. A list of extra packages and their installation instructions can be found on thebabylonjs user on npm.,SeeGetting Started:,If you want to contribute, please read ourcontribution guidelinesfirst.,To get a complete list of supported features, please visit ourwebsite.,Babylon.js is a powerful, beautiful, simple, and open game and rendering engine packed into a friendly JavaScript framework.</p>	We read every piece of feedback, and take your input very seriously.	: : ((
4	ssloy	tinyrenderer	20400	https://github.com/ssloy/tinyrenderer	<p>We read every piece of feedback, and take your input very seriously.,To see all available qualifiers, see ourdocumentation.,A brief computer graphics / rendering course,The rendered image is saved toframebuffer.tga.,You can open the project in Gitpod, a free online dev environment for GitHub:,On open, the editor will compile & run the program as well as open the resulting image in the editor's preview. Just change the code in the editor and rerun the script (use the terminal's history) to see updated images.,My source code is irrelevant. Read the wiki and implement your own renderer. Only when you suffer through all the tiny details you will learn what is going on.,Inthis series of articles, I want to show the way OpenGL works by writing its clone (a much simplified one). Surprisingly enough, I often meet people who cannot overcome the initial hurdle of learning OpenGL / DirectX. Thus, I have prepared a short series of lectures, after which my students show quite good renderers.,So, the task is formulated as follows: using no third-party libraries (especially graphic ones), get something like this picture:.,Warning: this is a training material that will loosely repeat the structure of the OpenGL library. It will be a software renderer.I do not want to show how to write applications for OpenGL. I want to show how OpenGL works.I am deeply convinced that it is impossible to write efficient applications using 3D libraries without understanding this.,I will try to make the final code about 500 lines. My students need 10 to 20 programming hours to begin making such renderers. At the input, we get a test file with a polygonal wire + pictures with textures. At the output, we'll get a rendered model. No graphical interface, the program simply generates an image.,Since the goal is to minimize external dependencies, I give my students just one class that allows working withTGAfiles. It's one of the simplest formats that supports images in RGB/RGBA/black and white formats. So, as a starting point, we'll obtain a simple way to work with pictures. You should note that the only functionality available at the very beginning (in addition to loading and saving images) is the capability to set the color of one pixel.,There are no functions for drawing line segments and triangles. We'll have to do all of this by hand. I provide my source code that I write in parallel with students. But I would not recommend using it, as this doesn't make sense. The entire code is available on github, andhereyou will find the source code I give to my students.,output.tga should look something like this:,,,,,,A brief computer graphics / rendering course</p>	We read every piece of feedback, and take your input very seriously.	: : (

Show 25 per page

Next steps:

[Generate code with df_repos](#)[View recommended plots](#)[New interactive sheet](#)

summarization model

```

from transformers import pipeline

# Load the summarization model
summarization_model = pipeline("summarization", model="facebook/bart-large-cnn")

# Sample README content as a single string (you can combine from your list)
readme_content = """We read every piece of feedback, and take your input very seriously.
The aim of the project is to create an easy-to-use, lightweight, cross-browser, general-purpose 3D library.
The current builds only include a WebGL renderer but WebGPU (experimental), SVG, and CSS3D renderers are also available as addons.
This code creates a scene, a camera, and a geometric cube, and it adds the cube to the scene.
It then creates a WebGL renderer for the scene and camera, and it adds that viewport to the document.body element.
Finally, it animates the cube within the scene for the camera.
If everything goes well, you should see this.
Cloning the repo with all its history results in a ~2 GB download.
If you don't need the whole history you can use the depth parameter to significantly reduce download size.
JavaScript 3D Library."""

# Function to summarize the README content
def summarize_readme(text):
    # Summarize the text using the summarization model
    summary = summarization_model(text, max_length=300, min_length=30, do_sample=False)
    return summary[0]['summary_text'] # Extract the summary text

# Call the summarization function
summary = summarize_readme(readme_content)
print("Summary:")
print(summary)

```

 /usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning: The secret `HF_TOKEN` does not exist in your Colab secrets. To authenticate with the Hugging Face Hub, create a token in your settings tab (<https://huggingface.co/settings/tokens>), set it as : You will be able to reuse this secret in all of your notebooks. Please note that authentication is recommended but still optional to access public models or datasets.

/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:1601: FutureWarning: `clean_up_tokenization_spaces`

warnings.warn(

Your max_length is set to 300, but your input_length is only 201. Since this is a summarization task, where outputs shorter than the

Summary:

The aim of the project is to create an easy-to-use, lightweight, cross-browser, general-purpose 3D library. Current builds only incl

#merge this logic

```

# Function to summarize the README content
def summarize_readme(content):
    if not content:
        return "No content to summarize."

    # Join all paragraphs to form a single string
    full_text = " ".join(content)

    # Check if the length of the full text exceeds a reasonable length
    max_length = 1024 # Set a limit for the summarization model
    if len(full_text) > max_length:
        full_text = full_text[:max_length] # Truncate the text

    try:
        # Summarize the text using the summarization model
        summary = summarization_model(full_text, max_length=130, min_length=30, do_sample=False)
        return summary[0]['summary_text'] # Extract the summary text
    except Exception as e:
        print(f"Error during summarization: {e}")
        return "Error occurred during summarization."

# The rest of your existing code follows here...

```

```

def scrape_readme(url):
    response = requests.get(url)

    # Check if the request was successful
    if response.status_code != 200:
        print(f"Failed to fetch {url}: {response.status_code}")
        return None, "No description found", [], []

    soup = BeautifulSoup(response.text, 'html.parser')

```

```

# Extract the title (usually in <h1>)
title_tag = soup.find('h1')
title = title_tag.get_text(strip=True) if title_tag else "No title found"

# Find the first <p> after the title
content_paragraphs = soup.find_all('p')

# Extract the description (first <p> after title)
description = soup.find('p').get_text(strip=True) if soup.find('p') else "No description found"

# Extract all paragraphs for main content
content = [para.get_text(strip=True) for para in content_paragraphs]

# Extract links from the README content
links = []
for link in soup.find_all('a', href=True):
    links.append((link.get_text(strip=True), link['href']))

return title, description, content, links

# Scrape the trending repositories and extract README summaries
def scrape_trending_repositories(limit=10): # Set a default limit
    topics_url = 'https://github.com/topics'
    response = requests.get(topics_url)
    content = response.text

    doc = BeautifulSoup(content, 'html.parser')
    topic_titles = doc.find_all('p', {'class': 'f3 lh-condensed mb-0 mt-1 Link--primary'})

    topic_titles_ = []
    topic_urls = []

    for tag in topic_titles:
        topic_titles_.append(tag.text)
        topic_urls.append("https://github.com" + tag.parent.parent.find_all('a', {'class': 'no-underline flex-grow-0'}, href=True)[0]['href'])

    all_repo_data = []

    # Loop through each topic URL to get repository info
    for topic_url in topic_urls:
        response = requests.get(topic_url)
        content = BeautifulSoup(response.text, 'html.parser')
        repositories = content.find_all('h3', {'class': "f3 color-fg-muted text-normal lh-condensed"})

        for repo in repositories:
            if len(all_repo_data) >= limit: # Check if the limit is reached
                break

            user = repo.find_all('a')[0].text.strip()
            repo_name = repo.find_all('a')[1].text.strip()
            repo_url = "https://github.com" + repo.find_all('a')[1]['href']

            # Get the stars (if available)
            star_tag = repo.find_next('span', {'id': 'repo-stars-counter-star'})
            if star_tag:
                star_text = star_tag.text.strip().replace(',', '')
                if 'k' in star_text:
                    star_text = star_text.replace('k', '') # Remove the 'k'
                    stars = int(float(star_text) * 1000) # Convert to integer after multiplying by 1000
                else:
                    stars = int(float(star_text)) # Convert directly to integer
            else:
                stars = 0

            # Scrape the README content for the repository
            readme_title, readme_description, readme_content, readme_links = scrape_readme(repo_url)

            # Summarize the README content
            readme_summary = summarize_readme(readme_content)

            # Collect all data in a dictionary
            repo_data = {
                'username': user,
                'repo': repo_name,
                'stars': stars,
                'repo_url': repo_url,
                'readme_content': readme_content,
                'readme_summary': readme_summary,
                'readme_links': readme_links
            }

```

```
all_repo_data.append(repo_data)

return all_repo_data

# Execute the scraping and save to CSV
repo_data = scrape_trending_repositories()
df_repos = pd.DataFrame(repo_data)
df_repos.to_csv('trending_repositories_with_readme_summary.csv', index=False)
```

→ this is a summarization task, where outputs shorter than the input are typically wanted, you might consider decreasing max_length ma

df_repos.head()

1 to 5 of 5 entries Filter 📄 ?

index	username	repo	stars	repo_url	readme_content	readme_summary	
0	mrdoob	three.js	102000	https://github.com/mrdoob/three.js	We read every piece of feedback, and take your input very seriously.,To see all available qualifiers, see ourdocumentation.,JavaScript 3D Library.,The aim of the project is to create an easy-to-use, lightweight, cross-browser, general-purpose 3D library. The current builds only include a WebGL renderer but WebGPU (experimental), SVG and CSS3D renderers are also available as addons.,Examples—Docs—Manual—Wiki—Migrating—Questions—Forum—Discord,This code creates a scene, a camera, and a geometric cube, and it adds the cube to the scene. It then creates aWebGLrenderer for the scene and camera, and it adds that viewport to thedocument.bodyelement. Finally, it animates the cube within the scene for the camera.,If everything goes well, you should see this.,Cloning the repo with all its history results in a ~2 GB download. If you don't need the whole history you can use thedepthparameter to significantly reduce download size.,Releases,JavaScript 3D Library.	The aim of the project is to create an easy-to-use, lightweight, cross-browser, general-purpose 3D library. Current builds only include a WebGL renderer but WebGPU (experimental), SVG and CSS3D renderers are also available as addons.	Skip synt: Com libra state
1	pmndrs	react-three-fiber	27300	https://github.com/pmndrs/react-three-fiber	We read every piece of feedback, and take your input very seriously.,To see all available qualifiers, see ourdocumentation.,ch A React renderer for Three.js.,,react-three-fiber is aReact rendererfor threejs.,Build your scene declaratively with re-usable, self-contained components that react to state, are readily interactive and can participate in React's ecosystem.,None. Everything that works in Threejs will work here without exception.,No. There is no overhead. Components render outside of React. It outperforms Threejs in scale due to React's scheduling abilities.,Yes. It merely expresses Threejs in JSX,<mesh />dynamically turns into new THREE.Mesh(). If a new Threejs version adds, removes or changes features, it will be available to you instantly without depending on updates to this library.,Live demo:https://codesandbox.io/s/icy-tree-brnsm?file=/src/App.tsx,This example relies on react 18 and usesexpo-cli, but you can create a bare project with their template or with thereact-nativeCLI.,Some configuration may be required to tell the Metro bundler about your assets if you useuseLoaderor Drei abstractions likeuseGLTFanduseTexture;, Visitdocs.pmnd.rs,You need to be versed in both React and Threejs before rushing into this. If you are unsure about React consult the officialReact docs, especiallythe section about hooks. As for Threejs, make sure you at least glance over the following links:,Some helpful material:,There is a vibrant and extensive eco system around three-fiber, full of libraries, helpers and abstractions.,Usage Trend of the @react-three Family,A small selection of companies and projects relying on three-fiber.,If you like this project, please consider helping out. All contributions are welcome as well as donations toOpencollective, or in cryptoBTC: 36fuguTPxGCNnYZSRdgdh6Ea94brCAjMbH,ETH: 0x6E3f79Ea1d0dcedeb33D3fC6c34d2B1f156F2682.,Thank you to all our backers! 🙌,This project exists thanks to all the people who contribute.,ch A React renderer for Three.js	react-three-fiber is aReact rendererfor threejs. Build your scene declaratively with re-usable, self-contained components that react to state. It outperforms Threejs in scale due to React's scheduling abilities.	Skip synt: fiber. fiber three prop