

homework918

The same parts:

Prompt TYPE	Prompt
Writer agent	<pre>writer_prompt = ChatPromptTemplate.from_messages([("system", """</pre> <p>Role: You are a versatile creator responsible for producing high-quality content based on client needs. Your work includes creating program code, web novels, academic papers, advertising designs, and more. You pay attention to detail, strive for excellence, and are willing to continuously improve your work based on feedback from the reviewer. Your goal is to enhance the quality of your work through multiple rounds of interaction to ensure client satisfaction.</p> <p>Skills:</p> <ol style="list-style-type: none">Content Creation:<ul style="list-style-type: none">Produce high-quality content based on client requirements.Ensure originality and creativity in your work.Adapt your style and approach to suit different types of content (e.g., code, novels, papers, ads).Responsiveness to Feedback:<ul style="list-style-type: none">Carefully review feedback from the reviewer.Make necessary adjustments to improve the quality of your work.Maintain a positive attitude towards constructive criticism.Communication:<ul style="list-style-type: none">Clearly communicate any questions or clarifications needed from the reviewer.

	<ul style="list-style-type: none">◦ Provide explanations for any creative decisions made in your work. <p>Constraints:</p> <ul style="list-style-type: none">• Focus solely on content creation tasks.• Respect the feedback provided by the reviewer.• Maintain a collaborative and professional relationship with the reviewer. <pre> """,), MessagesPlaceholder(variable_name="messages"),])</pre>
Reflection agent	<pre>reflection_prompt = ChatPromptTemplate.from_messages([("system", ""</pre> <p>Role: You are a meticulous reviewer responsible for evaluating the content created by the creator. Your job is to provide specific and actionable feedback to help improve the quality of the work. You will review various types of content, including program code, web novels, academic papers, advertising designs, and more. Your goal is to enhance the overall quality of the content through multiple rounds of review and feedback.</p> <p>Skills:</p> <ol style="list-style-type: none">1. Content Review:<ul style="list-style-type: none">◦ Thoroughly review the content created by the creator.◦ Identify any errors, inconsistencies, or areas for improvement.◦ Evaluate the content based on originality, creativity, and adherence to client requirements.2. Providing Feedback:<ul style="list-style-type: none">◦ Offer specific, actionable, and constructive feedback.◦ Highlight both strengths and areas for improvement in the content.

- Ensure feedback is clear, concise, and respectful.

3. Collaboration:

- Maintain a collaborative and supportive relationship with the creator.
- Be open to discussions and clarifications regarding feedback provided.
- Encourage continuous improvement and excellence in the creator's work.

Constraints:

- Focus solely on reviewing and providing feedback on content.
- Avoid making changes to the content directly; instead, provide suggestions for improvement.
- Maintain a professional and respectful tone in all interactions with the creator.

```
""",  
  
),  
    MessagesPlaceholder(variable_name="messages"),  
]  
)
```

Human Message

```
inputs = {  
    "messages": [  
        # HumanMessage(content="design a professional solution for  
video generation using The state-of-the-art technology and The  
cutting-edge academic research results ")  
        HumanMessage(content=""  
        需要开发一个 Sentinel程序，能帮助用户自动跟踪和分析 GitHub 开  
源项目 的最新动态，还能快速扩展到其他信息渠道，如 Hacker News  
的热门话题，提供更全面的信息挖掘与分析能力。  
        主要功能  


- 订阅管理：轻松管理和跟踪您关注的 GitHub 仓库。
- 更新检索：自动检索并汇总订阅仓库的最新动态，包括提交记录、问  
题和拉取请求。
- 通知系统：通过电子邮件等方式，实时通知订阅者项目的最新进展。

  
    ]  
}
```

- 报告生成：基于检索到的更新生成详细的项目进展报告，支持多种格式和模板，满足不同需求。
- 多模型支持：结合 OpenAI 和 Ollama 模型，生成自然语言项目报告，提供更智能、精准的信息服务。
- 定时任务：支持以守护进程方式执行定时任务，确保信息更新及时获取。
- 图形化界面：基于 Gradio 实现了简单易用的 GUI 操作模式，降低使用门槛。
- 容器化：项目支持 Docker 构建和容器化部署，便于在不同环境中快速部署和运行。
- 持续集成：实现了完备的单元测试，便于进一步配置生产级 CI/CD 流程，确保项目的稳定性和高质量交付。

这是已经实现好的一些python code which you can refer to :
dogecoin_news_client.py

```
import os
import logging
from datetime import datetime
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup

LOG = logging.getLogger(__name__)
LOG.setLevel(logging.DEBUG)
ch = logging.StreamHandler()
ch.setLevel(logging.DEBUG)
formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
ch.setFormatter(formatter)
LOG.addHandler(ch)
```

```
class DogeNewsClient:
    def __init__(self):
        self.url = 'https://cryptopanic.com/news/dogecoin/' # Dogecoin
        news URL

    def fetch_top_news(self):
        LOG.debug("准备获取Dogecoin的热门新闻。")
        try:
            # Set up Chrome WebDriver with headless mode
            chrome_options = Options()
            chrome_options.add_argument("--headless")
            chrome_options.add_argument("--no-sandbox")
            chrome_options.add_argument("--disable-dev-shm-usage")
            driver =
webdriver.Chrome(service=Service(ChromeDriverManager().install()
), options=chrome_options)

            driver.get(self.url)
            # Wait for the page to load
            WebDriverWait(driver, 10).until(
                EC.presence_of_element_located((By.CSS_SELECTOR,
'news-row-link'))
            )
            LOG.debug("页面加载完毕，开始解析新闻。")
            page_content = driver.page_source
            driver.quit()

            return self.parse_news(page_content)

        except Exception as e:
            LOG.error(f"获取Dogecoin的热门新闻失败：{str(e)}")
            return []

    def parse_news(self, html_content):
        LOG.debug("解析Dogecoin新闻页面。")
```

```

soup = BeautifulSoup(html_content, 'html.parser')

news_rows = soup.find_all('div', class_='news-row news-row-link')


top_news = []
for row in news_rows:
    title_tag = row.find('a', class_='news-cell nc-title')
    if title_tag:
        title = title_tag.get_text(strip=True)
        link = title_tag['href']
        full_url = f"

```

```

# Define the file path
file_path = os.path.join(dir_path, f'{hour}.md')
with open(file_path, 'w') as file:
    file.write(f"# Dogecoin Top News ({date} {hour}:00)\n\n")
    for idx, news in enumerate(top_news, start=1):
        file.write(f"{idx}. [{news['title']}]({news['link']})\n")

LOG.info(f"Dogecoin热门新闻文件已生成: {file_path}")
return file_path

def main():
    client = DogeNewsClient()
    client.export_top_news() # 默认情况下使用当前日期和时间

if name == "__main__":
    main()

```

Now you need to add a new information channel:

<https://36kr.com/information/AI/>

并解析网页中的 articles, 能够提取到列表中每一条article 的title、description、URL, for example:

title:三星AI新功能曝光, “重点通知” 将成AI OS杀手铜功能?

description:从被动到主动, 每一步都困难重重。

URL: <https://36kr.com/p/3042712457494023>

title:大模型狂飙两年后, “六小龙” 开始做减法

description:变大还是变小

URL: <https://36kr.com/p/3042747744009984>

Now write python code for the adding this information channel

```

"""
],

```

```
}
```

Output under different llms:

model	llama3.1:8b-instruct-q8_0	gpt-4o-mini
output	<div><h2>Round 1</h2><p>创作生成:</p><ul style="list-style-type: none">可以使用BeautifulSoup和Scrapy来解析这个网页，并提取需要的信息。<p>首先，我们需要安装所需的库：</p><pre>pip install beautifulsoup4 scrapy</pre><p>然后，我们需要写一个Spide器（即scrapy的Spider）来提取数据。由于这是一个简单的任务，因此我们可以使用Scrapy提供的基本spider模板。</p><p>创建一个新的python文件，例如36kr_spider.py，并添加以下代码：</p><pre>import scrapy class ArticleItem(scrapy.Item): title = scrapy.Field() description = scrapy.Field() url = scrapy.Field() class ThirtySixKRSpyder(scrapy.Spider): name = "36kr_spider" start_urls = ['https://36kr.com/information/AI/',] def parse(self, response): # 解析网页内容 soup = BeautifulSoup(response.body.decode('utf-8'), 'html.parser') # 提取每一篇文章的信息 article_elements = soup.find_all('li', class_='item-wrap')</pre></div>	<div><h2>Round 1</h2><p>创作生成:</p><ul style="list-style-type: none">To add a new information channel for extracting articles from the specified URL, you can create a new class that follows a similar structure to the DogeNewsClient class. Below is the Python code that implements this functionality.<pre>import os import logging from datetime import datetime from selenium import webdriver from selenium.webdriver.chrome.service import Service from selenium.webdriver.chrome.options import Options from selenium.webdriver.common.by import By from selenium.webdriver.support.ui import WebDriverWait from selenium.webdriver.support import expected_conditions as EC from webdriver_manager.chrome import ChromeDriverManager from bs4 import BeautifulSoup # Set up logging LOG = logging.getLogger(__name__) LOG.setLevel(logging.DEBUG) ch = logging.StreamHandler() ch.setLevel(logging.DEBUG) formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s') ch.setFormatter(formatter) LOG.addHandler(ch)</pre></div>


```

        for element in article_elements:
            title_element = element.find('a',
class_='issue-link')
            if title_element:
                title =
title_element.get_text(strip=True)
                url = title_element['href']
                description_element =
title_element.find_next('div',
class_='issue-desc')
                if description_element:
                    description =
description_element.get_text(strip=True)
                else:
                    description = ""

                # 创建并yield一个Item
                article_item = ArticleItem(
                    title=title,
                    description=description,
                    url=url,
                )
                yield article_item

```

```

        # 如果还有下一页，则继续爬取
        next_page = soup.find('a',
class_='next')
        if next_page:
            url =
response.urljoin(next_page['href'])
            yield scrapy.Request(url,
callback=self.parse)

```

这里，我们定义了一个Item类ArticleItem，用来储存每一条文章的信息，然后创建一个Spide器类ThirtySixKRSpyder，用来解析网页并提取数据。

在这个spider中，我们首先使用BeautifulSoup解析网页内容，找到所有的文章元素（由class为item-wrap的列表项组成）。然后，对每个元素进行处理，提

```

class KrAIClient:
    def __init__(self):
        self.url = 'https://36kr.com/information/AI/'
        # 36Kr AI news URL

    def fetch_top_articles(self):
        LOG.debug("准备获取36Kr AI 信息的热门文章。")
        try:
            # Set up Chrome WebDriver with headless
            mode
            chrome_options = Options()
            chrome_options.add_argument("--headless")
            chrome_options.add_argument("--no-sandbox")
            chrome_options.add_argument("--disable-dev-shm-usage")

            driver =
webdriver.Chrome(service=Service(ChromeDrive
rManager().install()), options=chrome_options)
            driver.get(self.url)

            # Wait for the page to load
            WebDriverWait(driver, 10).until(
EC.presence_of_element_located((By.CSS_SELE
CTOR, 'article-item')))
        )
        LOG.debug("页面加载完毕，开始解析文
章。")

        page_content = driver.page_source
        driver.quit()

        return self.parse_articles(page_content)

    except Exception as e:
        LOG.error(f"获取36Kr AI 信息的热门文章失
败: {str(e)}")
        return []

    def parse_articles(self, html_content):
        LOG.debug("解析36Kr AI 信息页面。")

```

取标题、描述和URL。最后，如果存在下一页，则继续爬取。

要运行这个spider，可以在命令行中执行以下命令：

```
scrapy crawl 36kr_spider
```

Scrapy将会根据配置的设置（例如是否保存到文件等）生成数据。数据可以使用 scrapy shell 命令检查，具体使用方法如下：

```
scrapy shell -s LOG_LEVEL=DEBUG -O articles.csv
```

```
http://localhost:6800/schedule.json -a settings=debug
```

或者，你也可以直接修改 spider 代码来输出到其他地方，比如保存到 csv 或 txt 文件中。

同时，我们需要将这个Spider整合到我们的原来的爬虫代码中。具体步骤如下：

1. 在原来的爬虫文件（例如 DogNewsCrawler.py）中，导入新创建的spider模块：

```
import scrapy
from . import ThirtySixKRSpyder
```

1. 创建一个新的spider对象并使用它来进行数据爬取。

```
spider = ThirtySixKRSpyder()
result_items = spider.crawl(start_urls)
```

1. 最后，将结果存储到文件中或进一步处理即可。

这里是一个完整的示例代码：

```
import scrapy
from . import ThirtySixKRSpyder

class DogNewsCrawler(scrapy.Spider):
    name = "dog_news_crawler"
    start_urls = [
        'https://36kr.com/information/AI',
    ]

    def crawl(self, urls):
        spiders = []
```

```
soup = BeautifulSoup(html_content,
                        'html.parser')
    article_rows = soup.find_all('div',
                                class_='article-item')
```

```
    articles = []
    for row in article_rows:
        title_tag = row.find('h3', class_='article-
            title')
        description_tag = row.find('p',
            class_='article-description')
        link_tag = row.find('a')
```

```
        if title_tag and description_tag and
            link_tag:
            title = title_tag.get_text(strip=True)
            description =
                description_tag.get_text(strip=True)
            link = link_tag['href']
```

```
        articles.append({
            'title': title,
            'description': description,
            'link': link
        })

        LOG.info(f"成功解析 {len(articles)} 条36Kr AI
            信息文章。")
        return articles
```

```
    def export_top_articles(self, date=None,
        hour=None):
        LOG.debug("准备导出36Kr AI 信息的热门文章。")
        articles = self.fetch_top_articles()

        if not articles:
            LOG.warning("未找到任何36Kr AI 信息的文
                章。")
            return None

        if date is None:
            date = datetime.now().strftime('%Y-%m-
                %d')
        if hour is None:
```

```

for url in urls:
    spider = ThirtySixKRSpyder()

spider.start_requests.append(scrapy.Request(url=url))
spiders.append(spider)

result_items = []
for spider in spiders:
    yield from spider.parse()

if __name__ == "__main__":
    client = DogNewsCrawler()
    client.crawl(client.start_urls)

```

这个示例代码结合了我们之前的原爬虫文件，使用 `ThirtySixKRSpyder` 对象进行数据爬取，并将结果存储到 `result_items` 列表中。

现在，我们就可以将这个新的 `spider` 模块集成到我们的原来的爬虫代码中，以爬取更多类型的数据！

Round 2

评论反思：

- 你的代码看起来很清晰，易于理解。以下是我的一些建议和反馈：

1. 命名规范

: 你的类和函数名称使用了英文单词，这是很好的实践。但是，你也可以考虑使用中文名称，以便更容易地与其他开发者沟通。

2. 注释

: 在代码中添加注释可以帮助其他人快速了解你的代码的逻辑。特别是在复杂的部分或需要解释的部分，注释会非常有用。

3. 错误处理

: 当 `BeautifulSoup` 或 `Scrapy` 遇到异常时，你可能想考虑添加一些错误处理机制，以便在出现问题时能够捕获并记录错误信息。

4. 性能优化

```

hour = datetime.now().strftime('%H')

# Create the directory path
dir_path = os.path.join('kr_ai_top_articles', date)
os.makedirs(dir_path, exist_ok=True)

# Define the file path
file_path = os.path.join(dir_path, f'{hour}.md')

with open(file_path, 'w') as file:
    file.write(f"# 36Kr AI Top Articles ({date} {hour}:00)\n\n")
    for idx, article in enumerate(articles, start=1):
        file.write(f"{idx}. **[{article['title']}] ({article['link']})**\n")
        file.write(f" * {article['description']}*\n\n")

LOG.info(f"36Kr AI 信息热门文章文件已生成: {file_path}")
return file_path

def main():
    client = KrAIClient()
    client.export_top_articles() # 默认情况下使用当前日期和时间

```

```

if __name__ == "__main__":
    main()

```

Explanation of the Code:

1. **Class Definition:** The `KrAIClient` class is created to handle the new information channel.
2. **Fetching Articles:** The `fetch_top_articles` method retrieves the HTML content from the 36Kr AI URL and waits for the articles to load.
3. **Parsing Articles:** The `parse_articles` method extracts the title, description, and URL of each article using `BeautifulSoup`.

: 如果你爬取的数据量很大, 建议使用 Scrapy 提供的缓存功能来减少网络请求次数, 从而提高爬取速度。

5. 日志记录

: 在生产环境中, 你可能想考虑使用 Scrapy 提供的日志记录功能, 以便能够监控和调试你的爬虫。

6. 代码重构

: 如果你需要爬取多个网站, 建议将网站信息抽象成一个类或函数, 使得代码更易于维护和扩展。

7. 测试

: 最后, 但并非最不重要的, 建议在开发完成后编写一些测试用例, 以便能够确保你的爬虫正常工作, 并且能够快速发现问题。

总之, 你的代码已经非常好, 继续保持这种风格和实践!

Round 3

创作生成:

- 非常感谢你的反馈和建议! 我会逐一考虑并实施这些意见, 以使得我的代码更加完善和高效。
1. 命名规范: 我会考虑使用中文名称, 以便更容易地与其他开发者沟通, 特别是在中文开发环境下。但是, 也会继续保持英文单词作为主要名称。
 2. 注释: 我会增加注释的数量和详细性, 以便其他人快速了解代码的逻辑, 尤其是在复杂的部分或需要解释的部分。
 3. 错误处理: 我会添加错误处理机制, 以便在出现问题时能够捕获并记录错误信息。Scrapy 提供的异常类和错误处理方法将是我的参考。
 4. 性能优化: 我将考虑使用 Scrapy 提供的缓存功能, 减少网络请求次数, 从而提高爬取速度。在需要高效数据爬取时, 这种优化很重要。

4. Exporting Articles: The `export_top_articles` method formats the extracted articles into a Markdown file.
5. Logging: The code includes logging to track the progress and errors.

You can integrate this code with your existing project structure to add the new functionality. Ensure you have the necessary libraries installed, such as `selenium`, `beautifulsoup4`, and `webdriver-manager`.

Round 2

评论反思:

- Your implementation of the new information channel for extracting articles from the specified URL is well-structured and effectively follows the design of the existing `DogeNewsClient` class. Here are some specific feedback points, strengths, and areas for improvement:

Strengths:

1. Consistent Structure: You maintained a consistent coding style and structure with the `DogeNewsClient`, making it easy to integrate and understand.
2. Logging: The use of logging for debugging and tracking the flow of execution is excellent. This will help in identifying issues during runtime.
3. Error Handling: You have included exception handling which is crucial for making the program robust against failures during web scraping.
4. Markdown Export: The export functionality is well-implemented, which can be useful for users wanting to save the output in a readable format.

Areas for Improvement:

1. Dynamic URL Handling

: Consider allowing the URL to be passed as a parameter to the class or methods. This would

5. 日志记录: 在生产环境中, 我会使用 Scrapy 提供的日志记录功能, 以便能够监控和调试我的爬虫。这将帮助我识别问题并解决它们。
6. 代码重构: 我会考虑抽象出网站信息, 使用类或函数来实现, 这样就可以更容易地维护和扩展我的爬虫代码。这种设计模式有助于可扩展性和灵活性。
7. 测试: 最后, 但并非最不重要的, 我会编写一些测试用例, 以便能够确保我的爬虫正常工作, 并且能够快速发现问题。在开发完成后, 测试是一个非常关键的步骤。

感谢你的指导和建议! 这些反馈将有助于我进一步改进我的代码并提高编码质量。

enhance the flexibility of the `KrAIClient` class, enabling it to be reused for other URLs if needed.

- Suggestion: Modify the `__init__` method to accept a `url` parameter.

```
def __init__(self,
url='https://36kr.com/information/AI/'):
    self.url = url
```

2. CSS Selectors

: Ensure that the CSS selectors used in `fetch_top_articles` and `parse_articles` are up-to-date with the website's structure, as web pages often change. It may be useful to verify these selectors regularly.

- Suggestion: Add a brief comment above the CSS selector lines, indicating that they may need updates if the website changes.

3. Documentation

: While the code is self-explanatory, adding docstrings to your class and methods would enhance readability and maintainability.

- Suggestion: Use standard Python docstring conventions to describe the purpose, parameters, and return values of each method.

4. Dependency Management

: Ensure that the necessary libraries (like `selenium`, `beautifulsoup4`, etc.) are mentioned in a `requirements.txt` file or documentation for easy installation.

- Suggestion: Create a `requirements.txt` file with the required packages and their versions.

5. Testing

: Implement unit tests for the `KrAIClient` class to ensure that methods like `fetch_top_articles`, `parse_articles`, and `export_top_articles` work as expected. This is especially important since the functionality relies heavily on external web content.

- Suggestion: Use a testing framework like `unittest` or `pytest` to facilitate this.

6. File Overwrite Handling

: Currently, the code will overwrite existing files without warning. It may be beneficial to check if the file already exists and prompt the user or create a unique filename.

- Suggestion: Implement a check using `os.path.exists(file_path)` before writing the file, and adjust the naming accordingly.

Conclusion:

Overall, the `KrAIClient` code is a solid addition to your project, with clear functionality and robustness. By addressing the areas for improvement, you can further enhance the maintainability and flexibility of your code. Implementing these suggestions will not only improve user experience but also facilitate easier updates and testing in the future. Keep up the great work!

Round 3

创作生成:

- Thank you for your detailed feedback and positive remarks regarding the implementation of the `KrAIClient` class. I appreciate your insights and suggestions for improvement. Here's how I would implement your suggestions:

Updated Implementation

1. Dynamic URL Handling

: Modify the `__init__` method to accept a URL parameter.

2. CSS Selectors Comments

: Add comments indicating the potential need for updates.

3. Documentation

: Add docstrings to the class and its methods.

4. Dependency Management

: Mention required libraries and provide a sample `requirements.txt`.

5. Testing

: Outline a simple structure for unit tests.

6. File Overwrite Handling

: Implement a check to avoid overwriting existing files.

Here's the revised code:

```
import os
import logging
from datetime import datetime
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
```

```
# Set up logging
LOG = logging.getLogger(__name__)
LOG.setLevel(logging.DEBUG)
ch = logging.StreamHandler()
ch.setLevel(logging.DEBUG)
formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
ch.setFormatter(formatter)
LOG.addHandler(ch)
```

```
class KrAIClient:
```

```
    """
```

A client to fetch and parse articles from a specified URL related to AI news.

Attributes:

url (str): The URL to fetch articles from.

```
    """
```

```

def __init__(self,
url='https://36kr.com/information/AI/'):
    """
    Initializes the KrAIClient with the given URL.

    Parameters:
        url (str): The URL from which to fetch
articles. Defaults to 36Kr AI news.
    """
    self.url = url

def fetch_top_articles(self):
    """
    Fetch the top articles from the specified URL.

    Returns:
        list: A list of dictionaries containing article
titles, descriptions, and links.
    """
    LOG.debug("准备获取36Kr AI 信息的热门文
章。")
    try:
        # Set up Chrome WebDriver with headless
mode
        chrome_options = Options()
        chrome_options.add_argument("--
headless")
        chrome_options.add_argument("--no-
sandbox")
        chrome_options.add_argument("--disable-
dev-shm-usage")

        driver =
webdriver.Chrome(service=Service(ChromeDrive
rManager().install()), options=chrome_options)
        driver.get(self.url)

        # Wait for the page to load
        WebDriverWait(driver, 10).until(

EC.presence_of_element_located((By.CSS_SELE
CTOR, 'article-item'))
        )
        LOG.debug("页面加载完毕，开始解析文
章。")

```



```

page_content = driver.page_source
driver.quit()

return self.parse_articles(page_content)

except Exception as e:
    LOG.error(f"获取36Kr AI 信息的热门文章失败: {str(e)}")
    return []

def parse_articles(self, html_content):
    """
    Parse the articles from the HTML content.

    Parameters:
        html_content (str): The HTML content to parse.

    Returns:
        list: A list of dictionaries containing article titles, descriptions, and links.
    """
    LOG.debug("解析36Kr AI 信息页面。")
    soup = BeautifulSoup(html_content, 'html.parser')
    article_rows = soup.find_all('div', class_='article-item')

    articles = []
    for row in article_rows:
        title_tag = row.find('h3', class_='article-title')
        description_tag = row.find('p', class_='article-description')
        link_tag = row.find('a')

        if title_tag and description_tag and link_tag:
            title = title_tag.get_text(strip=True)
            description = description_tag.get_text(strip=True)
            link = link_tag['href']

            articles.append({

```

```
'title': title,
'description': description,
'link': link
})
```

```
LOG.info(f"成功解析 {len(articles)} 条36Kr AI
信息文章。")
```

```
return articles
```

```
def export_top_articles(self, date=None,
hour=None):
```

```
    """
```

```
    Export the top articles to a Markdown file.
```

```
    Parameters:
```

```
        date (str): The date for the file name.
```

```
    Defaults to today's date.
```

```
        hour (str): The hour for the file name.
```

```
    Defaults to the current hour.
```

```
    Returns:
```

```
        str: The file path of the generated
    Markdown file.
```

```
    """
```

```
    LOG.debug("准备导出36Kr AI 信息的热门文
章。")
```

```
    articles = self.fetch_top_articles()
```

```
    if not articles:
```

```
        LOG.warning("未找到任何36Kr AI 信息的文
章。")
```

```
        return None
```

```
    if date is None:
```

```
        date = datetime.now().strftime('%Y-%m-
%d')
```

```
    if hour is None:
```

```
        hour = datetime.now().strftime('%H')
```

```
    # Create the directory path
```

```
    dir_path = os.path.join('kr_ai_top_articles',
date)
```

```
    os.makedirs(dir_path, exist_ok=True)
```

```
    # Define the file path
```

```

        file_path = os.path.join(dir_path,
f'{hour}.md')

        # Check if the file already exists
        if os.path.exists(file_path):
            LOG.warning(f"文件 {file_path} 已存在，将
使用新文件名。")
            file_path = os.path.join(dir_path,
f'{hour}_{datetime.now().timestamp()}.md')

        with open(file_path, 'w') as file:
            file.write(f"# 36Kr AI Top Articles ({date}
{hour}:00)\n\n")
            for idx, article in enumerate(articles,
start=1):
                file.write(f"{idx}. **[{article['title']}]
({article['link']})**\n")
                file.write(f" *
{article['description']}*\n\n")

        LOG.info(f"36Kr AI 信息热门文章文件已生成：
{file_path}")
        return file_path

def main():
    client = KrAIClient()
    client.export_top_articles() # 默认情况下使用当前日期和时间

if __name__ == "__main__":
    main()

```

Additional Considerations

1. requirements.txt

Example:

```

selenium==4.3.0
beautifulsoup4==4.10.0
webdriver-manager==0.8.5

```

2. Testing Structure

: You can create a separate testing file (e.g., test_kr_ai_client.py) and use unittest or pytest to create test cases for the methods in KrAIClient.

Conclusion

	<p>By implementing these improvements, the KrAIClient class becomes more flexible, maintainable, and robust. Thank you again for your constructive feedback; it significantly enhances the quality of the code. If you have any further suggestions or questions, feel free to ask!</p>
--	---