**Documentation for telegram bot**

A screen shot of a computer program

AI-generated content may be incorrect.

**Purpose of each library:**

**Requests:** Used to make web requests to get the contents of the webpage (e.g. requests.get)

**BeautifulSoup:** Used to parse the HTML of the webpage so that the news article links and headline can be extracted

**Telegram & Constants:** Used to help initialize the telegram bot

**Logging:** Used to send error messages to the terminal if anything goes wrong (e.g. Scraper fails to fetch an article

**Asyncio:** Allows the bot to send telegram messages while still processing the next article without freezing. Not replaceable with the time module because time modules will freeze the whole program (i.e. blocking)

**Os:** Used to store links in a .txt file to prevent repeated news articles

A computer screen shot of text

AI-generated content may be incorrect.

**Scraping targets:**

Upon inspection of the webpage’s HTML, I found the common CSS selector (used to find both the title and link to the article). The URL provided is the section I am interested in

A computer screen shot of text

AI-generated content may be incorrect.

**Preventing duplicate articles**

To prevent duplicate articles, I decided to store a .txt in the same folder with the links that have already been sent to the telegram chat. To do this, I needed **save\_sent\_links()** and **load\_sent\_links().**

**Save\_sent\_links():** This function takes in an argument (link of the news article) and writes it into the .txt file

**Load\_sent\_links():** The first step this function does is to read the sent\_links.txt file to gain a memory of the links previously sent and returns a set of those links

A computer screen with text and images

AI-generated content may be incorrect.

**General flow of scrape & send function of tele bot**

The flow of the bot is to:

1. Loop through the different scraping targets, making a HTTP get request for each’s target HTML content.
2. The bot then uses BeautifulSoup to parse that HTML content, finding the article’s title and URL to send to the telegram channel using telegram API

The **sent\_links** .txt file is made accessible in this function using global keyword. The function starts a loop and extracts the URL and section selector (used to find both the title and link to the article). The **scrape\_and\_send** function is an async function to allow the bot to scrape and send without freezing.

A screen shot of a computer program

AI-generated content may be incorrect.

**Try and except block usage of main function:**

The first two lines of try block are to send a request to the scraping target’s url to retrieve its HTML content, throwing an error if the request fails. The ‘user-agent’: Mozilla/5.0 is to mimic a real browser to avoid the request getting blocked.

BeautifulSoup was then used to parse this HTML content, using the section\_selector (CSS selector) to find the desired articles from that HTML. logging.warning and telegram will both throw an error message if no articles were found using this CSS selector.

A **count** variable is then introduced to limit the amount of articles sent from each website to prevent overwhelming notifications.

The title and article link are then extracted into the **title** and **link** variables using the BeautifulSoup library (.get\_text() and ‘href’ attribute).

The first **if-statement** is to convert the link into a clickable https URL:  


The second **if-statement** is to leverage the memory (sent\_links) to prevent sending duplicate links. I then saved the extracted link from that scraping target into the memory file by calling **save\_sent\_links(link).**

An f-string is then use to craft the message with the news link and the title and bot.send\_message is used to send the message to telegram. The count variable is incremented by 1 to prevent spamming of articles, a logging.info is triggered to create a log of the article sent using the telegram API and a 1 second asyncio delay is implemented.

A screen shot of a computer program

AI-generated content may be incorrect.

**Except block:**

Any error that occurs in the Try block (e.g.) will redirect the control flow to this part, e.g. sending HTML request, parsing HTML, extracting the sections, getting the title & link and sending the tele message.

A black background with white text

AI-generated content may be incorrect.

Lastly, the **scrape\_and\_send** function is executed

Sources:

* ChatGPT
* <https://medium.com/@samuelmideksa4/web-scraping-using-python-c2924e3f0924>

Final note:

Also read up a bit about how to scale this up (using adaptive strategies to search for selectors), static vs dynamic websites, database storage of scraped articles for duplicate detection and usage of a queue to scale up the scraper