Assignment: Pastebin Keyword Crawler (Crypto / t.me)

Objective:

Build a Python script that scrapes **Pastebin's public archive** for pastes containing **keywords related to crypto** (e.g., "crypto", "bitcoin", "ethereum") or **Telegram links** (e.g., "t.me"). Extract these pastes and store the relevant information in a structured format (JSON).

Requirements:

1. **Scrape Pastebin's archive** (`https://pastebin.com/archive`) to extract the latest 30 Paste IDs.

2. **Fetch the content** of each paste from the archive. Use the raw content URL format: `https://pastebin.com/raw/{paste_id}`.

3. **Check for mentions** of the following keywords in the paste content:

    - **Crypto-related terms**: "crypto", "bitcoin", "ethereum", "blockchain", etc.

    - **Telegram links**: "t.me"

4. If any of the keywords are found, **store the paste's information** in the following **JSON format** (one JSON object per line):

```
{
  "source": "pastebin",
  "context": "Found crypto-related content in Pastebin paste ID abc123",
  "paste_id": "abc123",
```

```
  "url": "https://pastebin.com/raw/abc123",
  "discovered_at": "2025-05-12T10:00:00Z",
  "keywords_found": ["crypto", "t.me"],
  "status": "pending"
}
```

5. **Output the results** to a file called `keyword_matches.jsonl`.

Validation Strategy:

- The candidate should validate their tool by ensuring that pastes containing **crypto-related terms** or **Telegram links** are correctly identified and saved.
- The output file (`keyword_matches.jsonl`) should only contain the pastes that include at least one of the keywords.
- To validate the tool, they can **manually check** the content of a few pastes to ensure the keywords are correctly detected.

Bonus (Optional):

- Implement a **rate-limiting mechanism** or **delays** between requests to avoid being blocked.
- Use **proxy rotation** to prevent rate-limiting issues.
- Implement **logging** to record which pastes were checked and whether any keywords were found.
- **Multi-threading** or **asynchronous programming** to speed up the crawling process.

Setup Hints for Candidates:

Install the required libraries:

```
pip install requests beautifulsoup4
```

1.
2. **Scraping the Archive**: Start by scraping the archive at `https://pastebin.com/archive` to extract the latest Paste IDs (you can get a list of pastes by parsing the HTML).

3. **Fetching Paste Content**: Once you have a Paste ID, use `https://pastebin.com/raw/{paste_id}` to get the raw paste content.

4. **Keyword Detection**: Check for the presence of the following keywords in the paste content:

    - **Crypto-related**: "crypto", "bitcoin", "ethereum", "blockchain", etc.

    - **Telegram links**: "t.me".

5. **Store Results**: If a paste contains any of the keywords, store the data in the required JSON format in the file `keyword_matches.jsonl`.

Example Use Case:

- The script should start by scraping the archive URL: `https://pastebin.com/archive`, which contains the 30 most recent pastes.

- It will extract Paste IDs from the archive page (e.g., `abc123`, `xyz456`).

- For each paste, the script will attempt to detect **crypto-related terms** or **Telegram links**.

- If any keywords are found, the script will save the paste information in the output file, `keyword_matches.jsonl`.

Expected output in the file `keyword_matches.json`:


```
{
  "source": "pastebin",
  "context": "Found crypto-related content in Pastebin paste ID abc123",
  "paste_id": "abc123",
  "url": "https://pastebin.com/raw/abc123",
  "discovered_at": "2025-05-12T10:00:00Z",
  "keywords_found": ["crypto", "bitcoin"],
  "status": "pending"
}
```


Validation Example:

To validate that the script works:

- The candidate can **check the logs** or the output file (`keyword_matches.jsonl`) to see if any relevant keywords were found.

- If no keywords are found in a paste, the script should log it as skipped but continue to the next one.