Fop Lab Project (Report)

Group Members:

1: Muhammad Usman Bhutto (453891).

2: Muhammad Usman Abdullah (461513).

3: Muhammad Maaz (479510).

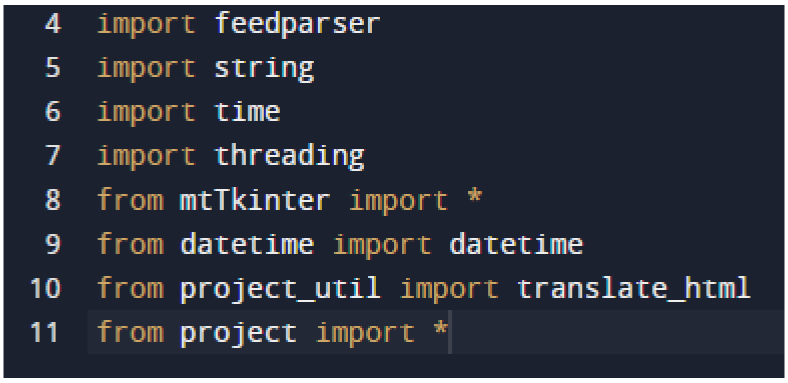
4: Zahoor Azam (453972).

INSTRUCTOR: SAQIB NAZIR/MUHAMMAD AFFAN.

DATE: 09-April-2024

Explanation of code

Part 1: Importing Necessary Modules

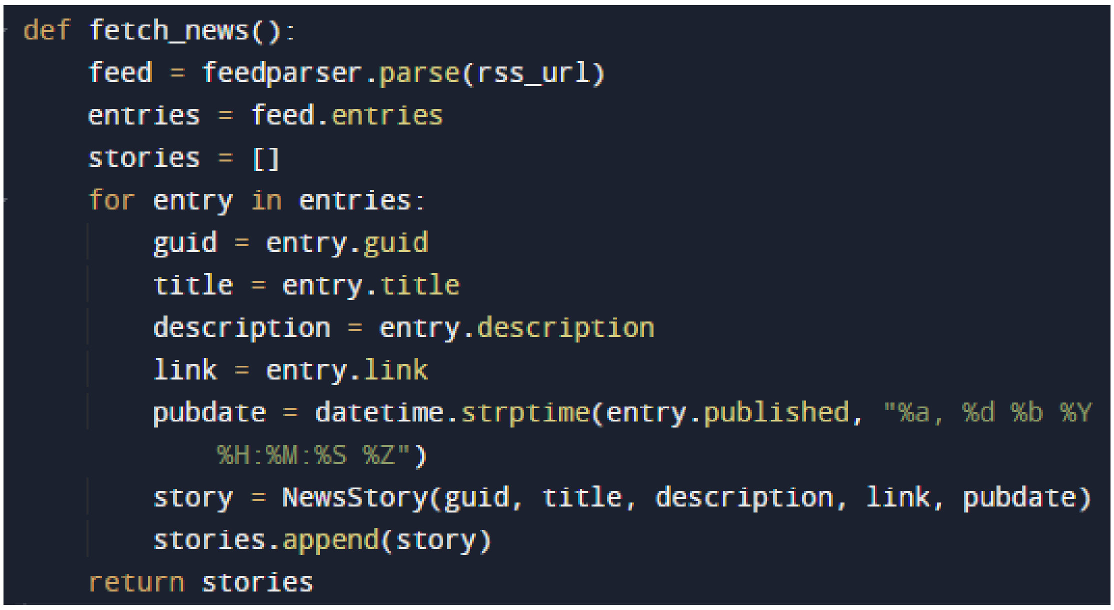


```python
import feedparser
import string
import time
import threading
from mtTkinter import *
from datetime import datetime
from project_util import translate_html
from project import *
```

Explanation:

- feedparser: A library to parse RSS feeds.
- string: Provides string manipulation utilities.
- time: Provides time-related functions.
- threading: Allows running multiple threads (used here for Tkinter).
- mtTkinter: A thread-safe version of Tkinter, used for GUI.
- datetime: Provides date and time handling functions.
- project_util: Contains utility functions (e.g., translate_html).
- project: Contains the main project classes and functions defined earlier.

Part 2: Fetching News Function

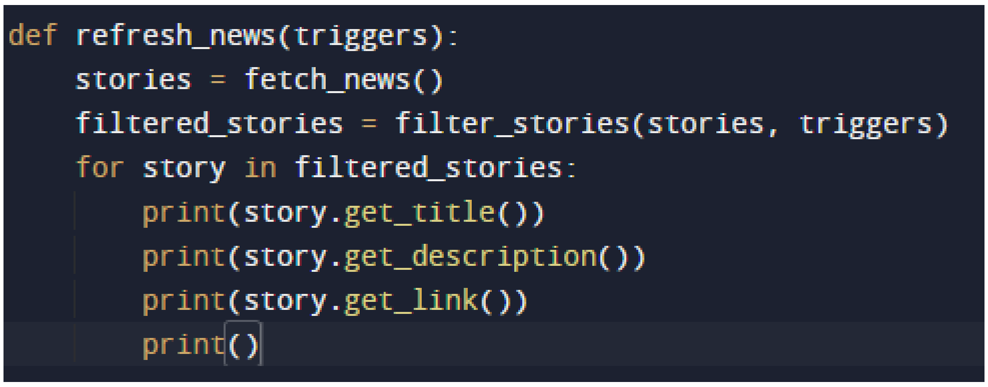


```python
def fetch_news():
    feed = feedparser.parse(rss_url)
    entries = feed.entries
    stories = []
    for entry in entries:
        guid = entry.guid
        title = entry.title
        description = entry.description
        link = entry.link
        pubdate = datetime.strptime(entry.published, "%a, %d %b %Y %H:%M:%S %Z")
        story = NewsStory(guid, title, description, link, pubdate)
        stories.append(story)
    return stories
```

Explanation:

- fetch_news: Function to fetch and parse the RSS feed.
- feedparser.parse(rss_url): Parses the RSS feed from the provided URL.
- entries: List of feed entries (news stories).
- stories: List to hold NewsStory objects.
- For each entry in entries, extract the relevant fields (guid, title, description, link, pubdate), create a NewsStory object, and append it to the stories list.
- Return the list of NewsStory objects.

Part 3: Refreshing News based on triggers

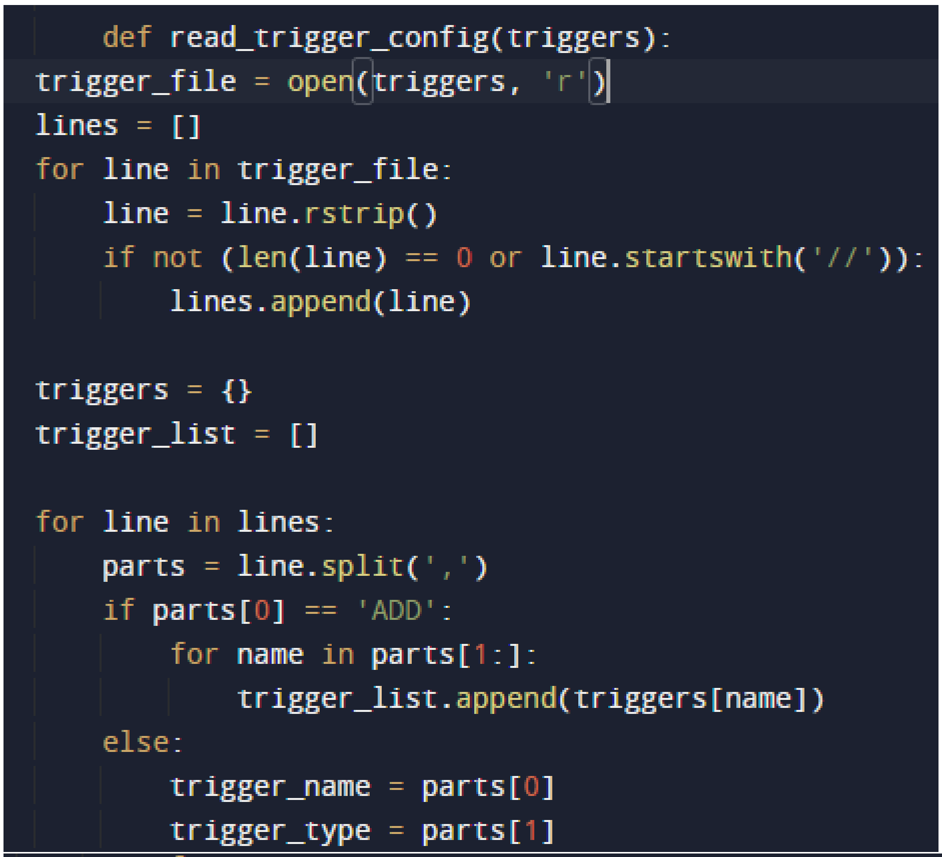


```python
def refresh_news(triggers):
    stories = fetch_news()
    filtered_stories = filter_stories(stories, triggers)
    for story in filtered_stories:
        print(story.get_title())
        print(story.get_description())
        print(story.get_link())
        print()
```

Explanation:

- refresh_news: Function to refresh news stories based on triggers.
- stories = fetch_news(): Fetch the latest news stories.
- filtered_stories = filter_stories(stories, triggers): Filter the fetched stories using the given triggers.
- Print the title, description, and link of each filtered story.

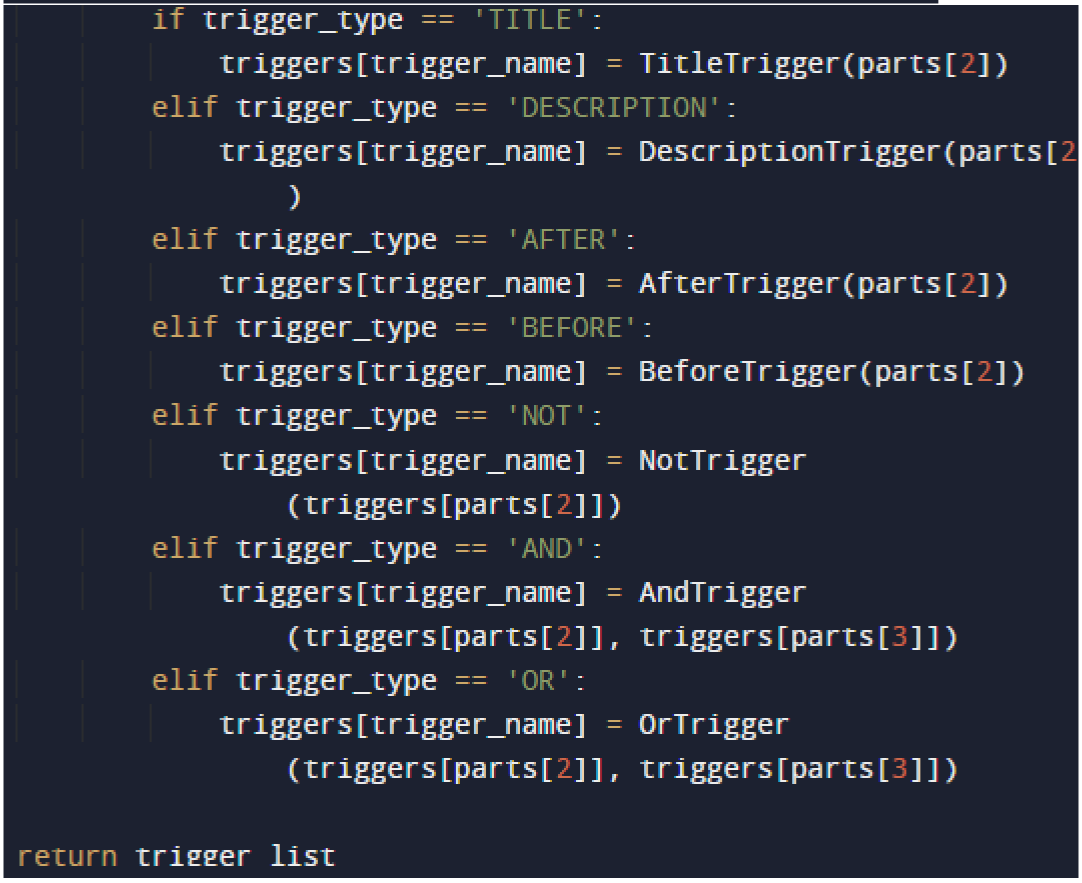
Part 4: Reading Trigger Configuration



```python
    def read_trigger_config(triggers):
trigger_file = open(triggers, 'r')
lines = []
for line in trigger_file:
    line = line.rstrip()
    if not (len(line) == 0 or line.startswith('//')):
        lines.append(line)

triggers = {}
trigger_list = []

for line in lines:
    parts = line.split(',')
    if parts[0] == 'ADD':
        for name in parts[1:]:
            trigger_list.append(triggers[name])
    else:
        trigger_name = parts[0]
        trigger_type = parts[1]
```



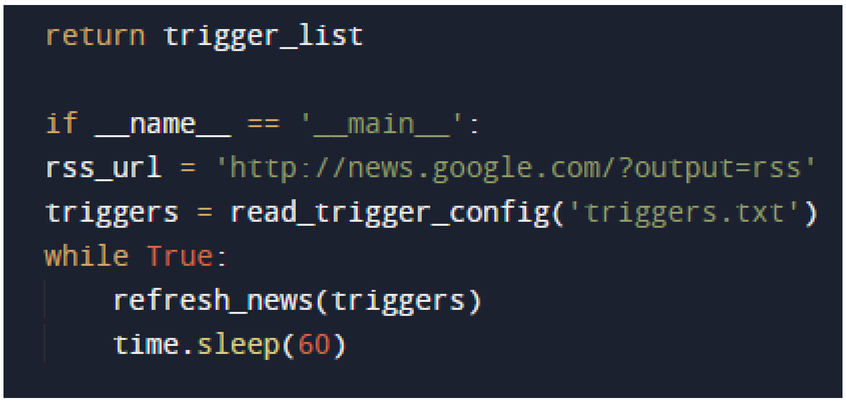
```python
        if trigger_type == 'TITLE':
            triggers[trigger_name] = TitleTrigger(parts[2])
        elif trigger_type == 'DESCRIPTION':
            triggers[trigger_name] = DescriptionTrigger(parts[2
                )
        elif trigger_type == 'AFTER':
            triggers[trigger_name] = AfterTrigger(parts[2])
        elif trigger_type == 'BEFORE':
            triggers[trigger_name] = BeforeTrigger(parts[2])
        elif trigger_type == 'NOT':
            triggers[trigger_name] = NotTrigger
                (triggers[parts[2]])
        elif trigger_type == 'AND':
            triggers[trigger_name] = AndTrigger
                (triggers[parts[2]], triggers[parts[3]])
        elif trigger_type == 'OR':
            triggers[trigger_name] = OrTrigger
                (triggers[parts[2]], triggers[parts[3]])

return trigger_list
```

Explanation:

- read_trigger_config: Function to read trigger configurations from triggers.txt.
- Opens the triggers.txt file and reads its content.
- Parses each line to create triggers and add them to the triggers dictionary.
- Adds specified triggers to the trigger_list.
- Returns the list of triggers.

Part 5: Main program execution



```
return trigger_list

if __name__ == '__main__':
rss_url = 'http://news.google.com/?output=rss'
triggers = read_trigger_config('triggers.txt')
while True:
    refresh_news(triggers)
    time.sleep(60)
```

Explanation:

- if __name__ == '__main__': Ensures the code runs only if the script is executed directly, not when imported as a module.
- Defines the RSS feed URL (rss_url).
- Calls read_trigger_config to read the trigger configurations from triggers.txt.
- Enters an infinite loop to continuously fetch and filter news stories every 60 seconds using refresh_news and time.sleep(60).

Summary:

1. Importing Modules: Import necessary libraries and modules.
2. Fetching News: Define a function (fetch_news) to fetch and parse news from the RSS feed.
3. Refreshing News: Define a function (refresh_news) to refresh news stories based on triggers.
4. Reading Trigger Configuration: Define a function (read_trigger_config) to read trigger configurations from triggers.txt.
5. Main Program Execution: Define the main program execution to continuously fetch and filter news stories based on triggers.