Fundamentals of Programming-II

Project Report

2nd Semester

Instructor: Dr Saqib Nazeer/Affan Ahmed

Session: ME-15 **Section: C** **Group :**

SUBMITTED BY

Name	CMS ID
Haseeb Tahir	453901
Hasnain Ali	478806
Salar Azam	479001
Sakhawat Ali	470935

School of Mechanical & Manufacturing Engineering SMME

➢ Complete Code

```
import feedparser
import string
import time
import threading
from tkinter import *
from datetime import datetime
import html
import base64

# -----------------------------------------------------------------------

# ======================
# Code for retrieving and parsing
# Google and Yahoo News feeds
# ======================

def translate_html(text):
"""
Translates HTML entities to their corresponding characters.
Also handles decoding if text is base64 encoded.
"""
# First, decode HTML entities
text = html.unescape(text)

# Check if the text is base64 encoded and decode it if necessary
try:
decoded_bytes = base64.decodebytes(text.encode('utf-8'))
text = decoded_bytes.decode('utf-8')
except (base64.binascii.Error, UnicodeDecodeError):
# If it's not base64 encoded, just return the HTML unescaped text
pass

return text


def process(url):
"""
Fetches news items from the rss url and parses them.
Returns a list of NewsStory instances.
"""
feed = feedparser.parse(url)
entries = feed.entries
ret = []
for entry in entries:
guid = entry.guid
```

```
title = translate_html(entry.title)
link = entry.link

# Check if description field exists
if 'description' in entry:
description = translate_html(entry.description)
else:
description = ""

# Handling different date formats
if 'published' in entry:
pubdate_str = entry.published
elif 'published_parsed' in entry:
pubdate_str = time.strftime('%a, %d %b %Y %H:%M:%S %Z', entry.published_parsed)
else:
continue

try:
pubdate = datetime.strptime(pubdate_str, "%a, %d %b %Y %H:%M:%S %Z")
except ValueError:
pubdate = datetime.strptime(pubdate_str, "%Y-%m-%dT%H:%M:%SZ")

newsStory = NewsStory(guid, title, description, link, pubdate)
ret.append(newsStory)
return ret


# ======================
# Data structure design
# ======================
#Problem 1
class NewsStory:
def __init__(self, guid, title, description, link, pubdate):
self.guid = guid
self.title = title
self.description = description
self.link = link
self.pubdate = pubdate

def get_guid(self):
return self.guid

def get_title(self):
return self.title

def get_description(self):
return self.description
```

```
def get_link(self):
return self.link

def get_pubdate(self):
return self.pubdate


# ======================
# Triggers
# ======================

class Trigger(object):
def evaluate(self, story):
raise NotImplementedError


class PhraseTrigger(Trigger):
def __init__(self, phrase):
self.phrase = phrase.lower()

def is_phrase_in(self, text):
text = text.lower()
for char in string.punctuation:
text = text.replace(char, ' ')
text_words = text.split()
phrase_words = self.phrase.split()
for i in range(len(text_words) - len(phrase_words) + 1):
if text_words[i:i + len(phrase_words)] == phrase_words:
return True
return False


class TitleTrigger(PhraseTrigger):
def evaluate(self, story):
return self.is_phrase_in(story.get_title())


class DescriptionTrigger(PhraseTrigger):
def evaluate(self, story):
return self.is_phrase_in(story.get_description())


class TimeTrigger(Trigger):
def __init__(self, time):
self.time = datetime.strptime(time, "%Y-%m-%dT%H:%M:%SZ")
```

```
class BeforeTrigger(TimeTrigger):
def evaluate(self, story):
return story.get_pubdate() < self.time


class AfterTrigger(TimeTrigger):
def evaluate(self, story):
return story.get_pubdate() > self.time


class NotTrigger(Trigger):
def __init__(self, trigger):
self.trigger = trigger

def evaluate(self, story):
return not self.trigger.evaluate(story)


class AndTrigger(Trigger):
def __init__(self, trigger1, trigger2):
self.trigger1 = trigger1
self.trigger2 = trigger2

def evaluate(self, story):
return self.trigger1.evaluate(story) and self.trigger2.evaluate(story)


class OrTrigger(Trigger):
def __init__(self, trigger1, trigger2):
self.trigger1 = trigger1
self.trigger2 = trigger2

def evaluate(self, story):
return self.trigger1.evaluate(story) or self.trigger2.evaluate(story)


# ======================
# Filtering
# ======================

def filter_stories(stories, triggerlist):
filtered_stories = []
for story in stories:
for trigger in triggerlist:
if trigger.evaluate(story):
filtered_stories.append(story)
```

```
break
return filtered_stories


# ======================
# User-Specified Triggers
# ======================

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

triggers = {}
trigger_list = []

for line in lines:
parts = line.split(',')
if parts[0] == 'ADD':
for name in parts[1:]:
if name in triggers:
trigger_list.append(triggers[name])
else:
trigger_name = parts[0]
trigger_type = parts[1]
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':
if parts[2] in triggers:
triggers[trigger_name] = NotTrigger(triggers[parts[2]])
elif trigger_type == 'AND':
if parts[2] in triggers and parts[3] in triggers:
triggers[trigger_name] = AndTrigger(triggers[parts[2]], triggers[parts[3]])
elif trigger_type == 'OR':
if parts[2] in triggers or parts[3] in triggers:
triggers[trigger_name] = OrTrigger(triggers[parts[2]], triggers[parts[3]])
```

```
return trigger_list


# =====================
# Main Thread
# =====================

SLEEPTIME = 120  # seconds


def main_thread(master, keywords):
try:
triggerlist = []
if keywords:
for keyword in keywords:
triggerlist.append(OrTrigger(TitleTrigger(keyword), DescriptionTrigger(keyword)))

frame = Frame(master)
frame.pack(side=BOTTOM)
scrollbar = Scrollbar(master)
scrollbar.pack(side=RIGHT, fill=Y)

t = "Google & Yahoo Top News"
title = StringVar()
title.set(t)
ttl = Label(master, textvariable=title, font=("Helvetica", 18))
ttl.pack(side=TOP)
cont = Text(master, font=("Helvetica", 14), yscrollcommand=scrollbar.set)
cont.pack(side=BOTTOM)
cont.tag_config("title", justify='center')
button = Button(frame, text="Exit", command=master.destroy)
button.pack(side=BOTTOM)
guidShown = []

def get_cont(newstory):
if newstory.get_guid() not in guidShown:
cont.insert(END, newstory.get_title() + "\n", "title")
cont.insert(END, "\n------------------------------------------------------------\n", "title")
cont.insert(END, newstory.get_description())
cont.insert(END,
"\n*****************************************************************\n", "title")
guidShown.append(newstory.get_guid())

while True:
print("Polling...")
stories = process("http://news.google.com/news?output=rss")
stories.extend(process("http://news.yahoo.com/rss/topstories"))
```

```
stories = filter_stories(stories, triggerlist)

list(map(get_cont, stories))
scrollbar.config(command=cont.yview)

time.sleep(SLEEPTIME)

except Exception as e:
print(f"Error occurred: {e}")


if __name__ == '__main__':
root = Tk()
root.title("RSS Feed Filter")

keywords = input("Enter keywords (comma-separated): ").strip().split(',')
keywords = [keyword.strip() for keyword in keywords if keyword.strip()]

t = threading.Thread(target=main_thread, args=(root, keywords))
t.start()

root.mainloop()
```

➢ CODE EXPLANATION

- **Code**

```
class NewsStory:

def __init__(self, guid, title, description, link, pubdate):

self.guid = guid

self.title = title

self.description = description

self.link = link

self.pubdate = pubdate

def get_guid(self):

return self.guid
```

```
def get_title(self):

return self.title

def get_description(self):

return self.description

def get_link(self):

return self.link

def get_pubdate(self):

return self.pubdate
```

- **Explanation**

We'll create a class called NewsStory to store this information. This class will take five arguments in its constructor:

- **guid:** A globally unique identifier (string) for the news story.
- **link:** The URL of the news story (string).
- **description:** A summary of the news story (string).
- **title:** The headline of the news story (string).
- **category:** The category the news story belongs to (string).

We'll also define functions within the class to access each piece of information later in our code. These functions will likely be named get_guid(), get_link(), get_description(), get_title(), and get_category().

- **Code**

```
class Trigger(object):

def evaluate(self, story):

raise NotImplementedError

class PhraseTrigger(Trigger):

def __init__(self, phrase):

self.phrase = phrase.lower()

def is_phrase_in(self, text):

text = text.lower()

for char in string.punctuation:
```

```
text = text.replace(char, ' ')
text_words = text.split()
phrase_words = self.phrase.split()
for i in range(len(text_words) - len(phrase_words) + 1):
if text_words[i:i + len(phrase_words)] == phrase_words:
return True
return False
class TitleTrigger(PhraseTrigger):
def evaluate(self, story):
return self.is_phrase_in(story.get_title())
class DescriptionTrigger(PhraseTrigger):
def evaluate(self, story):
return self.is_phrase_in(story.get_description())
class TimeTrigger(Trigger):
def __init__(self, time):
self.time = datetime.strptime(time, "%Y-%m-%dT%H:%M:%SZ")
class BeforeTrigger(TimeTrigger):
def evaluate(self, story):
return story.get_pubdate() < self.time
class AfterTrigger(TimeTrigger):
def evaluate(self, story):
return story.get_pubdate() > self.time
class NotTrigger(Trigger):
def __init__(self, trigger):
self.trigger = trigger
```

```
def evaluate(self, story):
return not self.trigger.evaluate(story)
class AndTrigger(Trigger):
def __init__(self, trigger1, trigger2):
self.trigger1 = trigger1
self.trigger2 = trigger2
def evaluate(self, story):
return self.trigger1.evaluate(story) and self.trigger2.evaluate(story)
class OrTrigger(Trigger):
def __init__(self, trigger1, trigger2):
self.trigger1 = trigger1
self.trigger2 = trigger2
def evaluate(self, story):
return self.trigger1.evaluate(story) or self.trigger2.evaluate(story)
```

- **Explanation**
  - Now we will create another class 'Triggers' to store the various triggers that will help sort the feed. This class would be an abstract class which would serve as a basis for other sub-classes. The purpose of defining the 'evaluate' function here is to make sure that all the sub-classes take the story as input and then perform some action on it, otherwise we would receive a 'not implemented error.
  - Now using the concept of inheritance, we are going to create a sub-class 'PhraseTrigger'. This part of the code compares the words in the text received from RSS feed to the words in our chosen phrase to see if they match. But the problem here is that the text received from the RSS feed is not in string format and it may contain a lot of unrequired punctuation. So once the text has been assigned a value through a constructor and made case insensitive, we use a for loop along with an in-built python library to iterate over the text and then replace the punctuations with blank spaces. Then by using the 'split' function we break both our phrase and the RSS feed text into separate words. Next by using nested for loops we try to find the keywords in the text by comparing each word in our phrase to the words in the text. If those words are present the function returns a true value.

- Both these triggers are a sub-class of the phrase trigger which means they inherit all the properties of the parent class. Their job is to compare the titles and the description of the RSS feed and see if they contain our keywords.
- The TimeTrigger class is the time-envelope of the news feed. The first task is to convert the publishing date/time of our choice into a standard format. Now the 'BeforeTime' and 'AfterTime' functions, retrieve the publishing date of the article using the functions defined in the first part and then make sure it's in the right time zone. The 'BeforeTrigger' returns a true value if the publishing date is before the user determined one, and the same goes for the 'AfterTrigger' which is true when the publishing date of the article is after the user determined one.
- The And, OR and Not triggers are again the subclass of the class 'Triggers'. The not function filters out any story containing the phrase entered as its argument. Whereas the And and Or triggers help when we want more than phrase to present in our desired news articles.

- **Code**

```
def filter_stories(stories, triggerlist):

filtered_stories = []

for story in stories:

for trigger in triggerlist:

if trigger.evaluate(story):

filtered_stories.append(story)

break

return filtered_stories
```

- **Explanation**
  - This function takes a list of triggers defined below and with the help of a nested for loop and an if-else statement, isolates the stories that pass all of our triggers. Such stories are stored in the empty list 'filtered_stories' defined at the beginning of the function.
- **Code**

```
def read_trigger_config(filename):

trigger_file = open(filename, 'r')

lines = []

for line in trigger_file:

line = line.rstrip()
```

```
if not (len(line) == 0 or line.startswith('//')):

lines.append(line)

trigger_file.close()

triggers = {}

trigger_list = []

for line in lines:

parts = line.split(',')

if parts[0] == 'ADD':

for name in parts[1:]:

if name in triggers:

trigger_list.append(triggers[name])

else:

trigger_name = parts[0]

trigger_type = parts[1]

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':

if parts[2] in triggers:

triggers[trigger_name] = NotTrigger(triggers[parts[2]])
```

elif trigger_type == 'AND':

if parts[2] in triggers and parts[3] in triggers:

triggers[trigger_name] = AndTrigger(triggers[parts[2]], triggers[parts[3]])

elif trigger_type == 'OR':

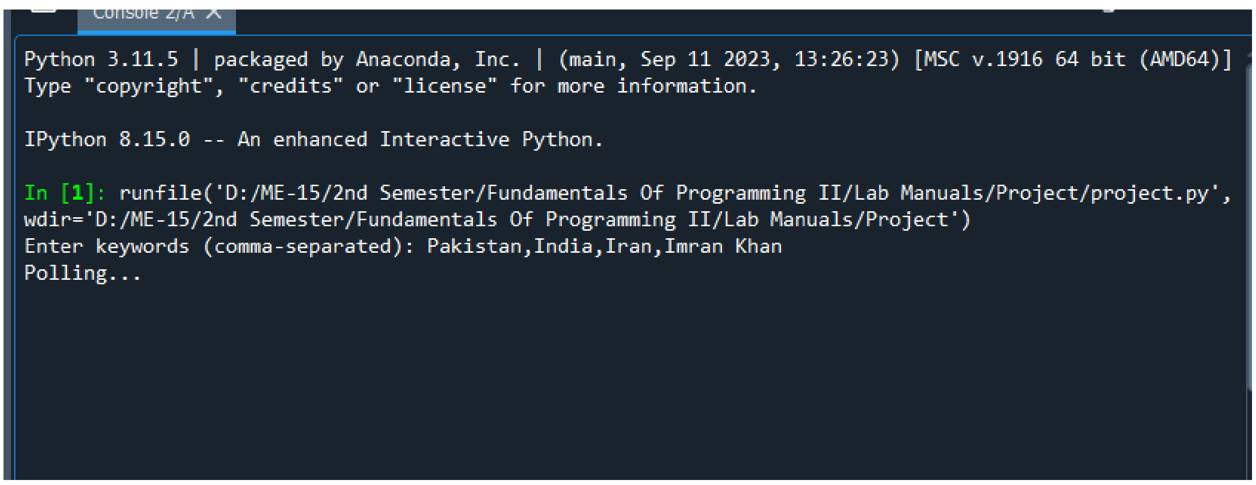
if parts[2] in triggers and parts[3] in triggers:

triggers[trigger_name] = OrTrigger(triggers[parts[2]], triggers[parts[3]])

return trigger_list

- **Explanation**
  - This last problem creates a dictionary to store all the triggers. And then stores this in a file. Most of our code depends upon taking input from the user regarding the phrases they want to see in their feed and the ones they don't. This part of the code deals with determining those phrases and then integrating them into the rest of the program.

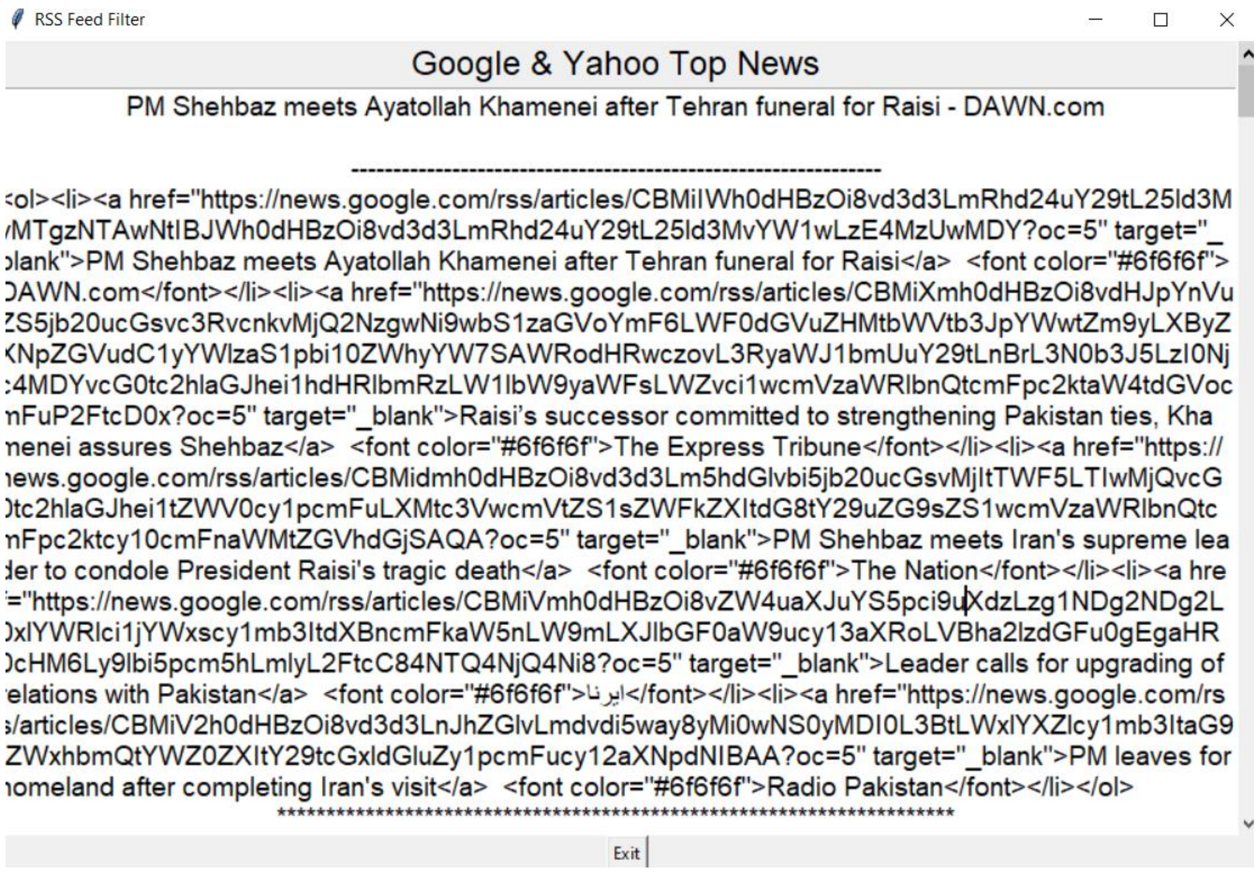
Output



```
Python 3.11.5 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:26:23) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: runfile('D:/ME-15/2nd Semester/Fundamentals Of Programming II/Lab Manuals/Project/project.py',
wdir='D:/ME-15/2nd Semester/Fundamentals Of Programming II/Lab Manuals/Project')
Enter keywords (comma-separated): Pakistan,India,Iran,Imran Khan
Polling...
```



Google & Yahoo Top News

PM Shehbaz meets Ayatollah Khamenei after Tehran funeral for Raisi - DAWN.com

```
---------------------------------------------------------------
<ol><li><a href="https://news.google.com/rss/articles/CBMiIWh0dHBzOi8vd3d3LmRhd24uY29tL25ld3M
/MTgzNTAwNtIBJWh0dHBzOi8vd3d3LmRhd24uY29tL25ld3MvYW1wLzE4MzUwMDY?oc=5" target="_
blank">PM Shehbaz meets Ayatollah Khamenei after Tehran funeral for Raisi</a>  <font color="#6f6f6f">
DAWN.com</font></li><li><a href="https://news.google.com/rss/articles/CBMiXmh0dHBzOi8vdHJpYnVu
ZS5jb20ucGsvc3RvcnkvMjQ2NzgwNi9wbS1zaGVoYmF6LWF0dGVuZHMtbWVtb3JpYWwtZm9yLXByZ
XNpZGVudC1yYWlzaS1pbi10ZWhyYW7SAWRodHRwczovL3RyaWJ1bmUuY29tLnBrL3N0b3J5LzI0Nj
c4MDYvcG0tc2hlaGJhei1hdHRlbmRzLW1lbW9yaWFsLWZvci1wcmVzaWRlbnQtcmFpc2ktaW4tdGVoc
mFuP2FtcD0x?oc=5" target="_blank">Raisi's successor committed to strengthening Pakistan ties, Kha
menei assures Shehbaz</a>  <font color="#6f6f6f">The Express Tribune</font></li><li><a href="https://
news.google.com/rss/articles/CBMidmh0dHBzOi8vd3d3Lm5hdGlvbi5jb20ucGsvMjItTWF5LTIwMjQvcG
0tc2hlaGJhei1tZWV0cy1pcmFuLXMtc3VwcmVtZS1sZWFkZXItdG8tY29uZG9sZS1wcmVzaWRlbnQtc
mFpc2ktcy10cmFnaWMtZGVhdGjSAQA?oc=5" target="_blank">PM Shehbaz meets Iran's supreme lea
der to condole President Raisi's tragic death</a>  <font color="#6f6f6f">The Nation</font></li><li><a hre
f="https://news.google.com/rss/articles/CBMiVmh0dHBzOi8vZW4uaXJuYS5pci9uXdzLzg1NDg2NDg2L
0xlYWRlci1jYWxscy1mb3ItdXBncmFkaW5nLW9mLXJlbGF0aW9ucy13aXRoLVBha2lzdGFu0gEgaHR
0cHM6Ly9lbi5pcm5hLmlyL2FtcC84NTQ4NjQ4Ni8?oc=5" target="_blank">Leader calls for upgrading of
relations with Pakistan</a>  <font color="#6f6f6f">ایرنا</font></li><li><a href="https://news.google.com/rs
s/articles/CBMiV2h0dHBzOi8vd3d3LnJhZGlvLmdvdi5way8yMi0wNS0yMDI0L3BtLWxlYXZlcy1mb3ItaG9
tZWxhbmQtYWZ0ZXItY29tcGxldGluZy1pcmFucy12aXNpdNIBAA?oc=5" target="_blank">PM leaves for
homeland after completing Iran's visit</a>  <font color="#6f6f6f">Radio Pakistan</font></li></ol>
*************************************************************************
```