**ABSTRACT**

The foremost purpose of our project is to deliver the latest and relevant news articles to the user’s provided E-mail. The News Aggregator application is a simple yet technical app that uses various available and popular libraries to provide relevant news articles according to the user’s interest and thus efficiently delivers latest news articles from the internet through automation of the process of web scraping for the news content as well as emailing it to the user so that he / she can stay updated in their topics of interest on-the-go.

**INTRODUCTION**

As we have discussed that the aim of our project is to eliminate the everyday hassle for the people of searching latest and relevant news articles on the web by completely automating this process to efficiently deliver news to the users with the help of our robust backend and makes the overall process hassle – free and user friendly. We also maintain a csv file for storing user credentials like E-mail and their topics of interest.

**PROBLEM STATEMENT**

In the fast paced, busy life of ours, it is difficult to be updated with the news and the happenings. People, rather than searching for news on Google, want someone to send them news on their E-mail so that they can avoid the hassles for searching for it. This is one worldly problem that needs a solution.

**OBJECTIVES**

1. **No wastage of time searching:** By automating the process of extraction and delivering the latest news from the internet, the user won’t have to waste their time searching for relevant articles.
2. **Formatted news according to user interest:** News only according to the user’s interest will be selected and delivered with proper formatting.
3. **Available on-the-go:** Daily news of interest will be sent to the user provided E-mail regularly so that they can be up-to-date from wherever they are.

**FEATURES**

1. **Automated News Extraction:** We use several web scraping technologies along with some backend programming to extract the latest and the most relevant news article.
2. **Distinct and User-Friendly GUI:** We have a distinct and simple GUI which asks the user for their E-mail and their topics of interest of news of which they want to read. We also support alternate method of data collection through Google Forms.
3. **Hassle-Free Experience:** Our robust backend ensures the user has a hassle-free and smooth experience. Our simple backend simplifies the work even further and maximizes efficiency.

**LIBRARIES USED**

The Libraries used in our project are:

1. Tkinter
2. Pandas
3. GoogleNews
4. Newspaper
5. Datetime
6. re
7. smtplib
8. config
9. time
10. csv

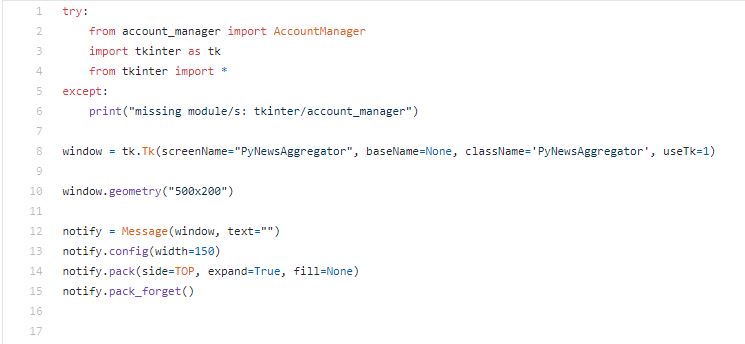
**CODE SNIPPETS**

**GUI.py**

This is the front-end of our project.

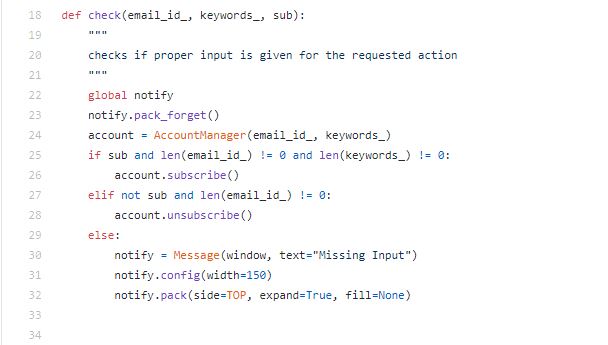
* It uses the tkinter library.
* Allows the user to subscribe to / unsubscribe from the service.

Here are some snapshots of the file:



The code in the above snapshot performs the following operations:

* Importing the account\_manager module and the tkinter library.
* Creating a window and setting its geometry(dimensions).
* Allocating space for messages at the bottom of the window (since the messages need to be displayed at function calls).



The above function:

* Checks if the required input is provided for the requested action (email and keywords to subscribe, just email to unsubscribe).
* It creates an AccountManager object and initializes it with the given input.
* If all the required input is provided for the requested action, the function calls the member function corresponding to the action through the object (object.subcsribe() / object.unsubscribe()).
* Else, it displays “Missing Input”.



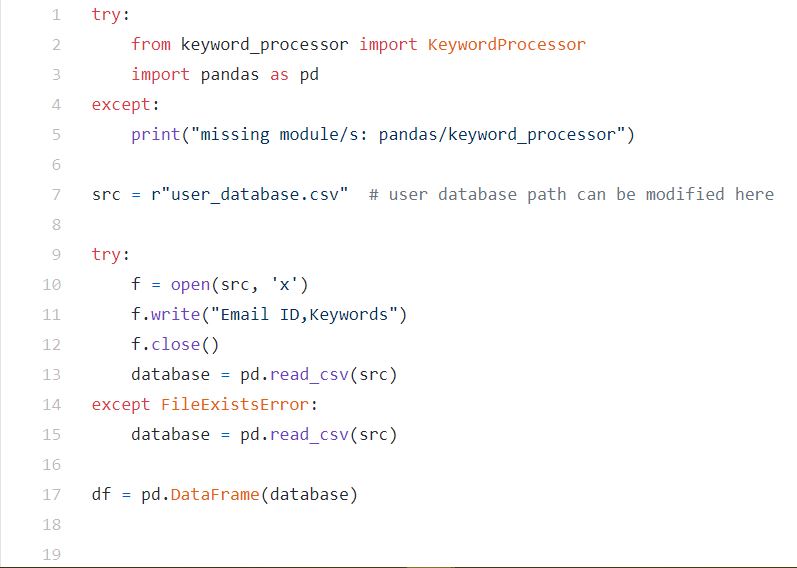
The code in the above snapshot:

* Creates input fields with labels and buttons.
* Maps the buttons to the check function with parameters corresponding to the button (sub = True to subscribe, False to unsubscribe).
* Binds the buttons to the “ENTER” key.
* Runs the window in an infinite loop.

**account\_manager.py**

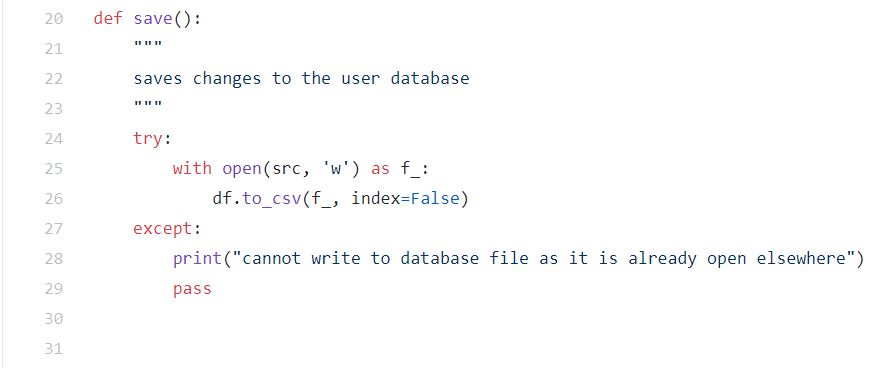
* This module manages the user database.
* It uses the pandas library to do so.
* The database is a csv file.

Here are some snapshots of the module:

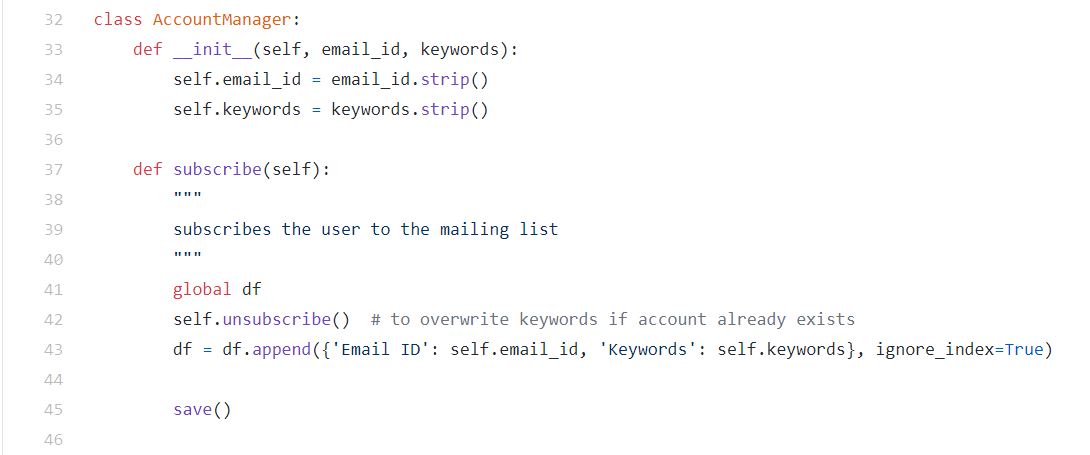


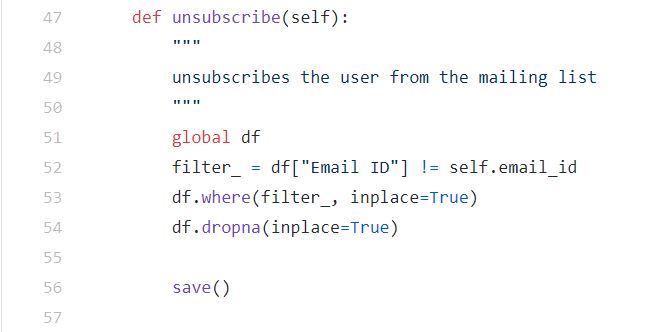
The code in the above snapshot performs the following operations:

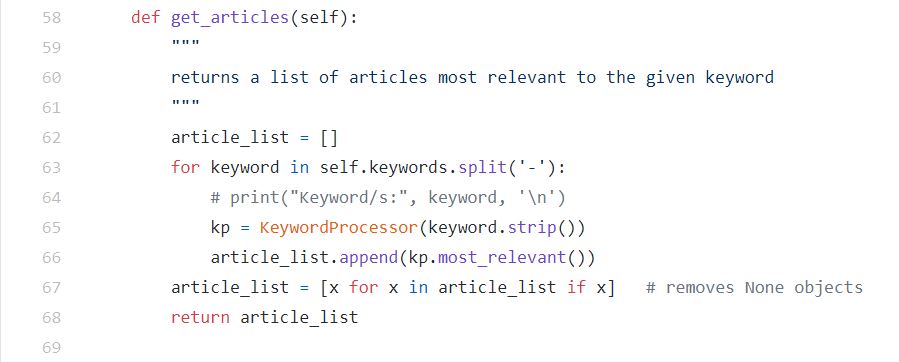
* Importing the keyword\_processor module and the pandas library.
* Assigning the path to the user database to a variable.
* Storing the contents of the csv file in a pandas dataframe.



The above function saves the changes made to the user database.







Here’s the AccountManager class.

Its data members are: “email\_id” and “keywords”.

Its member functions are:

* subscribe(): This function appends the email ID and preferred keywords of the user to the database. If the email ID already exists in the database, it first unsubscribes the user and then appends the details to the database, effectively overwriting the keywords.
* unsubscribe(): This function removes the user from the database by filtering the dataframe and replacing rows matching the user details with NULL or NaN values.
* get\_articles(): This function creates objects of the KeywordProcessor class and uses its member function “most\_relevant” to create a list of Article objects most relevant to the given keywords.

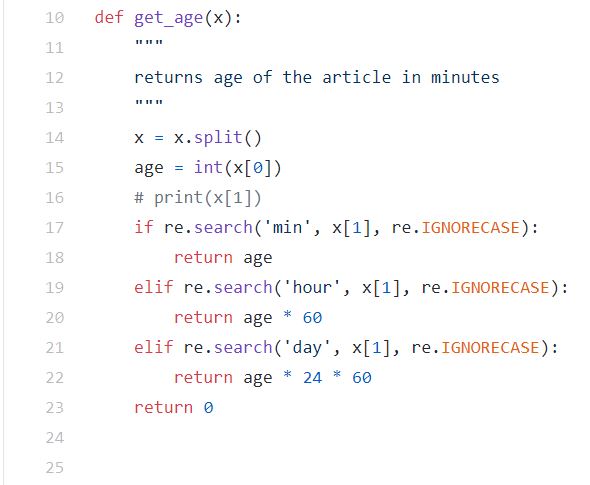
**keyword\_processor.py**

* This module processes a keyword and provides the news article most relevant to it.
* It uses the GoogleNews, newspaper, datetime and re libraries to achieve this.

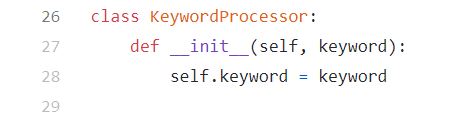
Here are some snapshots of the module:



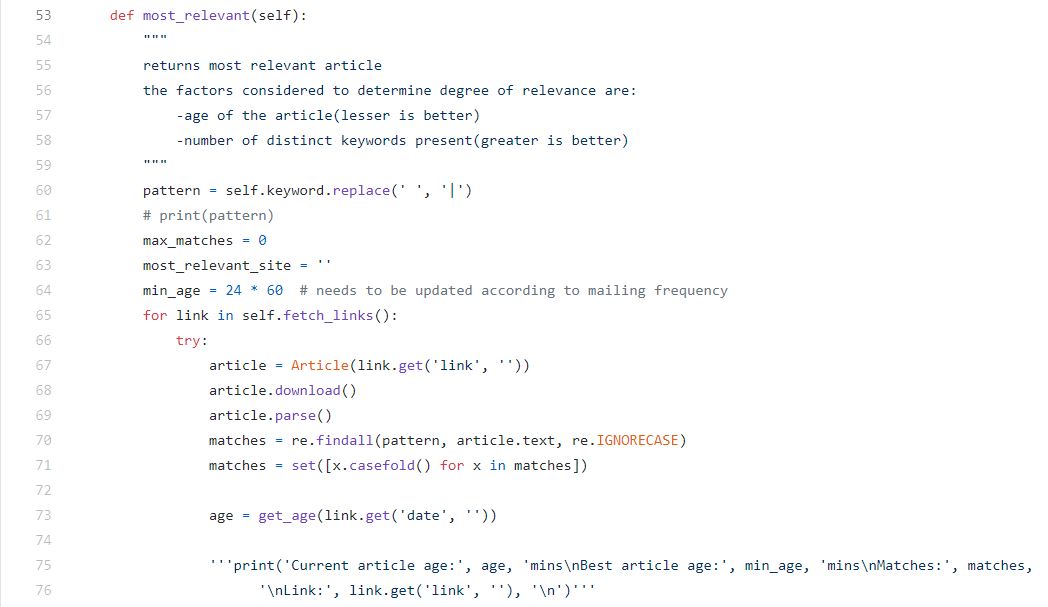
The above code imports the above-mentioned libraries.



This function calculates the age of the article in minutes. Its argument is a string of the form “x minutes/hours/days”. It uses the search function of the re library to look for substrings like “min/hour/day” and calculates the age accordingly.









Here’s the KeywordProcessor class.

Its data member is: “keyword”.

Its member functions are:

* fetch\_links(): This function creates an object of the GoogleNews class, uses it to search for the keyword on “Google News” and returns links to articles from at most a day ago.
* most\_relevant(): This function goes through the links returned by the “fetch\_links” function and determines the most relevant article based on the following criteria:
  + Age of the article processed by the “get\_age” function (lesser is better)
  + Number of distinct keywords (determined using regular expressions) present in the article (greater is better)

**email\_automation.py**

The file named as email\_automation.py is the fourth and the last file of our project. It is the file that basically interacts with all of our other three python modules/files directly or indirectly. Basically, it is the driving wheel of our project. But again, we know that this file is of zero importance if our other three files are non-functional.

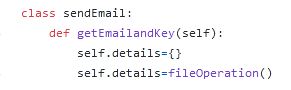
The email automation file exercises control over the Keyword Processor and Account Manager through a simple python-ic import statement. The other key imports in this file comprise of smtplib, datetime, config, csv.

We have used a famous Python Data Structure which was very helpful in this file. The Data Structure was a Dictionary (unordered, non-default). The email id’s were fed as the keys to the dictionary while the topics of interest of the user served as the values of it.



Here is a snapshot from our Github repository that highlights key imports along with the use of file handling and dictionary.

Our file comprises of a SendEmail Class which has necessary objects and attributes along with methods that efficiently help us in making the email service automated. The key functions in the Class SendEmail is send(self), calls Account Manager, and several other email functionalities from the smptp library. The timedate module checks for the timestamp in which the email needs to be sent. The usage of config library helps keep our password and email discreet.

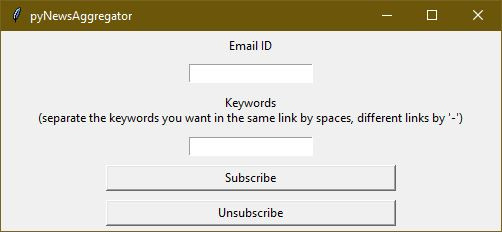


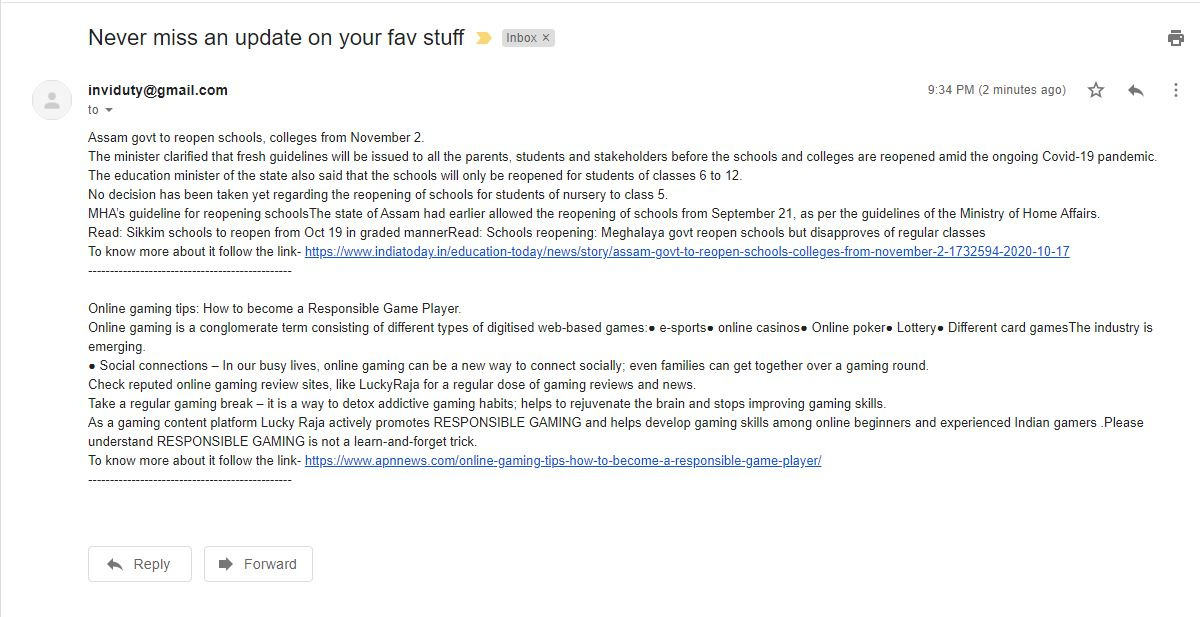
Try and except blocks at every possible juncture ensures that our program is not terminated abruptly at Run Time. We always have a control on the potential bug which would terminate our program in any worst case.

A snapshot of the very important ‘send’ function which acts as the heart of the module ‘emai\_automation’.



**RESULT**





**CONCLUSION**

Hence, by completing this Python project we understand how to use various libraries in our project and also helps us to expand our horizons to use and implement various libraries and understand how they can be used to increase the efficiency of our project.