Jennifer Mondragon
A#: 04117836
COSC 6374
Instructor: Carlos Rubio-Medrano
March 7th, 2023

Assignment One

For this assignment, I tried a couple of different implementations. Firstly, the enron_to_mbox.py file that was intended to convert the files did not work on my system, due to the file not being able to recognize files correctly. It would say that there was no file called '1.', when reading from the first folder, therefore I had to stop there. Next, I tried to use the files given from the uploaded materials, but that only provided another issue, saying that there was no "permission", which I was not able to resolve. Since I did not want to struggle anymore with the files itself, I opted to use the .cvs file listed in the materials. This did not cause any problems and I was able to start implementing right after installing the necessary packages. Once installed, I was able to split the .csv file into different 'headers' such as date, subject, the folder, who the email was from and who it was sent to, and the body of the message.

Running the Program

As per the assignment instructions, everything is runnable through the command line. In order to run, at least within PyCharm, the syntax is as follows:

```
python main.py <function> <parameters>
```

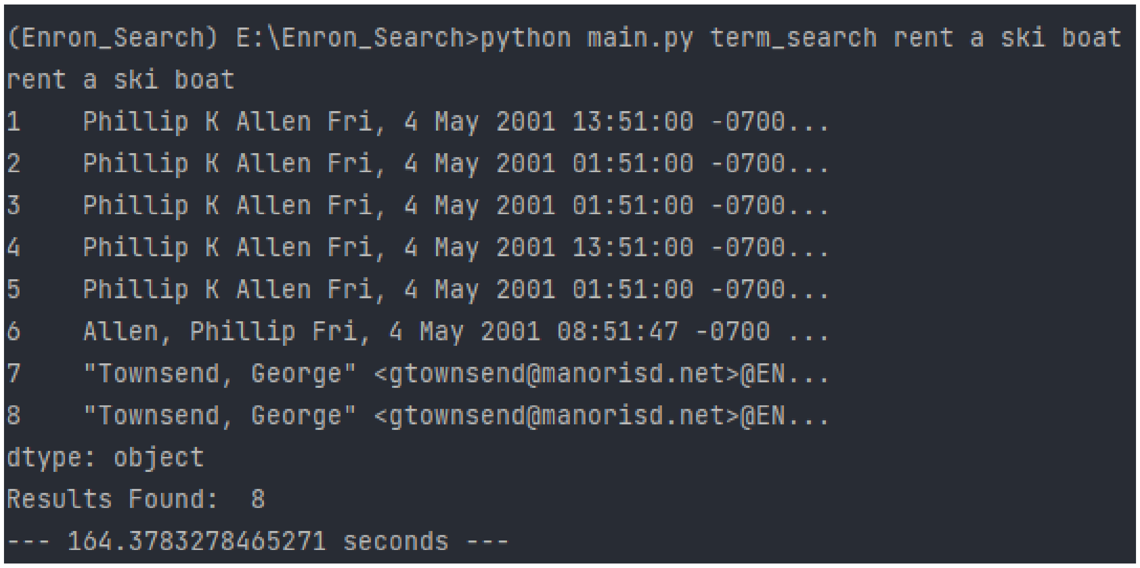
For example:

```
python main.py term_search rent a ski boat
```

Term_Search Function

The term_search function uses the arguments passed from the end of the function call argument, until the end of the command line statement. This means that everything after the function call is considered the search term. With that, the user can enter in a statement, and the function will print the resulting information, that being the 'To' information, as well as the date information from the email. In this example, the user can enter in the phrase "rent a ski boat", and the function returns the results, that being eight instances of the phrase being found, as well as where the phrases were being found.

Jennifer Mondragon
A#: 04117836
COSC 6374
Instructor: Carlos Rubio-Medrano
March 7th, 2023

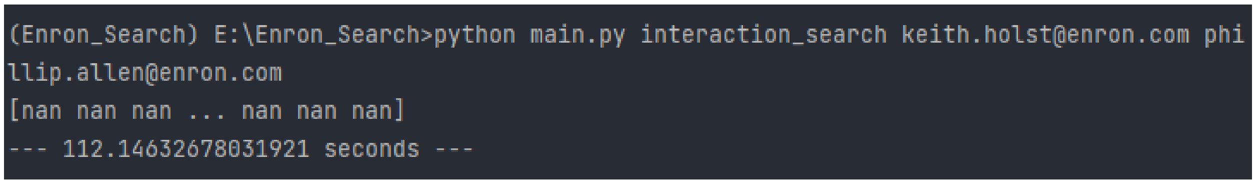


```
(Enron_Search) E:\Enron_Search>python main.py term_search rent a ski boat
rent a ski boat
1    Phillip K Allen Fri, 4 May 2001 13:51:00 -0700...
2    Phillip K Allen Fri, 4 May 2001 01:51:00 -0700...
3    Phillip K Allen Fri, 4 May 2001 01:51:00 -0700...
4    Phillip K Allen Fri, 4 May 2001 13:51:00 -0700...
5    Phillip K Allen Fri, 4 May 2001 01:51:00 -0700...
6    Allen, Phillip Fri, 4 May 2001 08:51:47 -0700 ...
7    "Townsend, George" <gtownsend@manorisd.net>@EN...
8    "Townsend, George" <gtownsend@manorisd.net>@EN...
dtype: object
Results Found:  8
--- 164.3783278465271 seconds ---
```

Figure 1: Term_Search Example

Interaction_Search Function

The interaction_search function uses the arguments of two addresses, and searches for an interaction between these two addresses. This can be either the sent or received email, as long as an interaction is completed between these two addresses. Once gathered, the information regarding the subject of the email, and the date are printed alongside the information regarding who sent and who received the emails. As for this function, I was not able to get it fully functional, as it only prints nan through all the emails, instead of only pulling the ones that contain the passed addresses. I tried many different methods of trying to join, merge, and compare the addresses to the column data, but was not able to implement this fully.



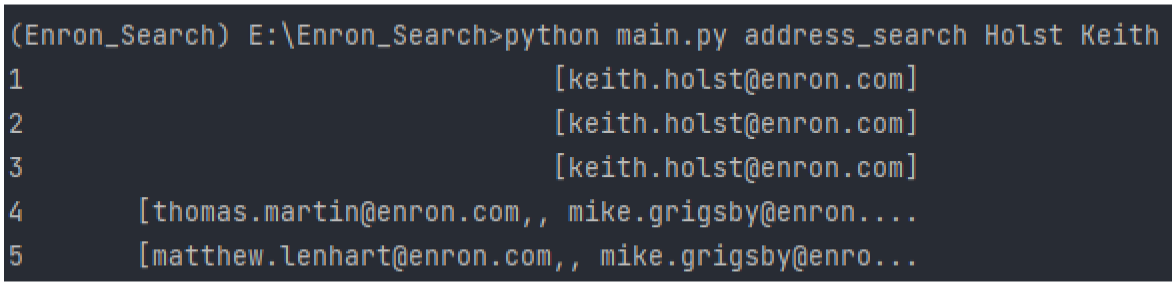
```
(Enron_Search) E:\Enron_Search>python main.py interaction_search keith.holst@enron.com phi
llip.allen@enron.com
[nan nan nan ... nan nan nan]
--- 112.14632678031921 seconds ---
```

Figure 1: Interaction_Search Example

Jennifer Mondragon
A#: 04117836
COSC 6374
Instructor: Carlos Rubio-Medrano
March 7th, 2023

Address_Search Function

The address_search function uses the arguments of the employee's first name and last name. That is then sent to the function, address_search, where the employee name is used to find the email address associated with it. Given there are users with the same names, multiple addresses will be printed, as the function searches for containing names, not email addresses. Once the program has determined the name, it is able to compare the names associated with the data frame 'X-To', which uses the names of the employees. Following that, the program prints out the email associated with that employee name. I was not able to find out a way to drop the emails from the list when there are multiple emails listed, such as large emails. Remove duplicates also did not remove repeated emails, which makes the list longer than it should be.



```
(Enron_Search) E:\Enron_Search>python main.py address_search Holst Keith
1                                  [keith.holst@enron.com]
2                                  [keith.holst@enron.com]
3                                  [keith.holst@enron.com]
4        [thomas.martin@enron.com,, mike.grigsby@enron....
5        [matthew.lenhart@enron.com,, mike.grigsby@enro...
```

Figure 3: Address_Search Example