George Dixon

12/5/18

Final Project Report

1. I had three main goals for this project. The first goal was to follow the project instructions to the best of my ability, demonstrating my knowledge of class content and receiving full credit for the assignment. My second goal was to become comfortable using an API I had no previous experience using. I hoped to use online resources, including youtube videos and stackoverflow, in order to leverage the API in the ways I wanted. My third goal was to complete the project without going to office hours or asking my peers for help. Throughout the course of my coding experiences so far, I’ve become reliant on my peers and office hours, which isn’t a bad thing. I do however think I was becoming too accustomed to going to others at the first sign of trouble. To counter this, I wanted to complete the project completely on my own, using only online resources.
2. I am very happy to say that I believe I achieved all of my goals. It’s impossible to say that I achieved my first goal at this point, but I am planning on taking my finished project to my GSI and seeing if he would give me full credit based on what I have. I was also able to achieve my second and third goal, becoming comfortable with the Gmail API and completing the project on my own.
3. One of the most significant problems I faced was understanding the Gmail API. A lot of the online documentation didn’t interact with the API directly, instead most chose to interact with the domain itself to retrieve pertinent information. The only other really big problem I encountered was the fact that the data returned for each email varied significantly in their structure, so the location of the information I wanted was often in different places. This led me to have to write 5 different functions for retrieving the date for emails. Other minor problems include not knowing how to create visualizations and my limited experience writing to and retrieving information from sql databases.
4. For the social media report, please run final\_project\_206.py and look for a print statement in your terminal window. It will only show up after you exit the visualization. This is what I got my last time running the program:

The day with the most emails is Mon which has 49 emails.

Tue had 23 emails.

Wed had 17 emails.

Sun had 16 emails.

Thu had 14 emails.

Fri had 10 emails.

Sat had 7 emails.

1. First, run the following command to install a gmail API library using pip: pip install --upgrade google-api-python-client oauth2client . Then, download the credentials.json file that should be within the zipped folder. Then, run the final\_project\_206.py file. It’ll ask you to sign into your gmail account so it can access the data, make sure that the gmail account has at least 100 emails. It’ll output a matplotlib file and will print some info in your terminal window or wherever you’re running the file (it’s a print statement).
2. **Get\_day\_of\_week\_with\_semicolon**-takes an email id as an input and outputs the day of the week the email was sent and the email id in a list

**Get\_date\_from\_edge\_case\_1**- takes an email id as an input and outputs the day of the week the email was sent and the email id in a list

**get\_date\_from\_edge\_case\_2-**takes an email id as an input and outputs the day of the week the email was sent and the email id in a list

**get\_date\_from\_edge\_case\_3-**takes an email id as an input and outputs the day of the week the email was sent and the email id in a list

**get\_date\_from\_edge\_case\_4-**takes an email id as an input and outputs the day of the week the email was sent and the email id in a list

**Process\_api\_results-**takes a dictionary that has a list of message id’s within it. It goes through each id and makes a request to the gmail API for said id via the above functions. It outputs a list of lists. It also pauses briefly every 50 retrievals as to not overload the api

**write\_to\_database**-This takes a list of lists and writes to a database called ‘emails.sqlite’. If the database doesn’t exist, it creates a table with a column for the day of the week and a column for the email id. If the database does exist, the function retrieves all the email ids currently in the database and only adds new ones. There are no set inputs/outputs, just the database creation/change

**Process\_data**- doesn’t have any set inputs or outputs. It instead takes information from the emails.sqlite file and prints the day that has the most emails. It also prints each day and how many emails they have

**Show\_data**- doesn’t have any set inputs. It instead takes information from the emails.sqlite database and creates a bar chart with the days of the week on the x axis and the number of emails on the y axis.

7.

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Issue Description | Location of Resource | Result |
| 12/1 | Needed to learn how to interact with the google API and get the necessary credentials to do so. | https://developers.google.com/gmail/api/quickstart/python | I was able to access the google API and get a list of labels. |
| 12/3 | Needed to learn how to get a list of email IDs from the google API. | https://www.codediesel.com/api/how-to-access-gmail-using-python/ | I was able to get a list of email ID’s. |
| 12/4 | Needed to know how to change the colors of the bars in my matplotlib bar chart. | https://stackoverflow.com/questions/18973404/setting-different-bar-color-in-matplotlib-python | I was able to change the colors of the bars in my matplotlib bar chart. |
| 12/4 | Needed to know how to create a legend in my matplotlib bar chart. | https://stackoverflow.com/questions/18974928/how-to-create-custom-legend-in-matplotlib-based-on-the-value-of-the-barplot | I was able to create a legend for my matplotlib bar chart. |
| 12/4 | Needed to be able to select a certain column from an sqlite table. | https://www.tutorialspoint.com/sqlite/sqlite\_select\_query.htm | I was able to return a certain column from an sqlite file. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |