Python Google API Challenge

By Hesam Nikzad Jamnani

In this brief document, I will explain each step of this mini-project.

1- Setup a test Gmail account:
   a. *username: hesam.jamnani@gmail.com*
   b. *password: Harchi123@*

2- Install the Google client library as it is mentioned in the developers' guide with Windows Command Prompt for Python:

   pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib

3- I used this YouTube video and inspired its script for a better understanding and coding.

4- In the Google Cloud console create a new project named "Python Google API Challenge"

5- Select ***APIs & Services > Library*** from the menu and search "Gmail API". Choose the *first* item and ***ENABLE*** it.

6- Select ***APIs & Services > Credentials*** from the menu and push the ***+CREATE CREDENTIALS*** button. Select ***OAuth client ID*** and ***Desktop app*** as the ***application type*** in the options menu. Name the credential file "challenge" and push the ***CREATE*** button.

7- Select ***APIs & Services > OAuth consent screen*** from the menu. Push ***+ ADD USERS*** button to add test users who are allowed to work with this application. There is just one user (hesam.jamnani@gmail.com) in this list.

8- Select ***APIs & Services > Credentials*** from the menu again. Download the ***Challenge*** JSON file in the ***OAuth 2.0 Client IDs***. Rename it to ***credentials.json*** and move to the local workspace directory.

9- In the workspace directory, I made two Python scripts. One for coding a class named Gmail which is the base of the code and name it Google.py. Another script is Test.py which uses Google.py as a library in it and a Gmail object can be defined with it.

10- The idea is to create an object/class which needs just a credential.json file. Therefore, in the __init__ function I used concepts mentioned in No. 3 to define a service with the credential file. By seeing the "Successfully connected to hesam.jamnani@gmail.com" notification you can be sure the connection is done and the service is ready to use. After the first time signing in with users in the list of No. 7, a token will be saved as a pickle file in the ***token files*** folder which helps the user to avoid other sign-ins.

11- I defined a method/function named ***Send*** for this object which needs three elements (receiver, subject, body) to send an email via the hesam.jamnani@gmail.com email address. As it is mentioned in the comments of the code a *MIMEMultipart* object had to be defined and after attaching all three variables, convert to a raw string to be ready to send.

12- A function is created in the class to read all the emails (Sent and Inbox). With ***Read_emails*** all emails after being listed in a variable called *messages* are extracted with their ***id***s. Each email message has details, including ***id, subject, and receiver*** which are concluded to be in the ***headers*** after many messages from Inbox and sent have been checked in an online JSON viewer website. Also, it is inferred the ***body*** is stored in the ***snippet***. Since the letters of the words are both upper and lower case in the headers ***name*** key in different situations (sent or inbox) I decided to convert them to lower case before any comparison within *if* conditions.

After extraction and gathering the desired data, they are stored as a DataFrame in a variable named ***mails*** of the object.

13- The search function idea is based on finding the number of keywords on the subject or body texts (it assumes the subject is 5 times as important as the body). First, I defined a function to make texts sign less (there could be many other signs in the ***extras*** list). Then, convert the keyword to a lowercase one to assure searching is not going to be case-sensitive. After calling the emails internally, we can work on a Pandas DataFrame which contains all essential materials. As it is mentioned in the code's comments normalization, converting strings to lists of words, counting the keywords, and calculating and sorting the points of emails happened on the DataFrame. Finally, the results appear by iteration on the DataFrame and if there is no result, a notice will demonstrate.

14- In the Test.py script the class object from Google.py is imported like a standard library. Also, there are examples of how to run the service and use the sending and searching features.