SI 206 Final Project Report

Zeyao Wang

Include: In this report, I will introduce my goal, which goals I achieved, the problems I faced, my social media “report”, the instructions for running my code, documentation for each function I wrote and all the resources,I used.

1. My Goals

In this Final Project, my goals are to calculate the number of emails on each day of the week that you received based on the email address of the sender you entered in.
Then to create a **.csv** report and use a bar chart to clearly show when I receive emails from Canvas the most frequently.

2. Which Goals I Achieved

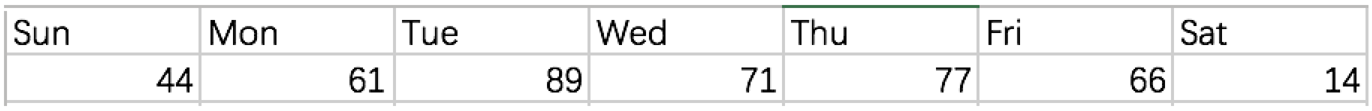
I created functions to extracted data from GMAIL API. Then, I calculated the number of emails received from specific sender (input) each day of the week and the number of emails received from specific sender (input) each time of the day.
Next, store those data to **JSON** file and **SQLite** database and exported to a **.csv** report and a bar chart that allows audiences to intuitively accept the visualization of data. Therefore, I achieved all my goals.

3. What Problems I faced

At first, I had no idea how to use the API, so I searched and followed the GMAIL API instructions to figure out authorization. But the result shew that it was failed to authorize Gmail account. Then I turned on the IMAP in the Gmail setting by search how to use IMAP (Internet Mail Access Protocol) to access the data on Google.

4. Your Social Media “Report”

I use *notifications@instructure.com*(Canvas) to be my specific sender.
(Because each person gets different number of emails from *notifications@instructure.com*(Canvas), the result will be different, which is the reason I can't use `Unittest`)
My social Media “Report” get 422 emails totally. There are 44 emails sent on Sunday, 61 emails on Monday, 89 emails on Tuesday, 71 emails on Wednesday, 77 emails on Thursday, 66 emails on Friday and 14 emails on Saturday. Most emailed sent on the weekday especially on Tuesday and Thursday and less emailed sent on weekends.



Sun	Mon	Tue	Wed	Thu	Fri	Sat
44	61	89	71	77	66	14

Table 1: The number of Emails Received Each Day of the Week

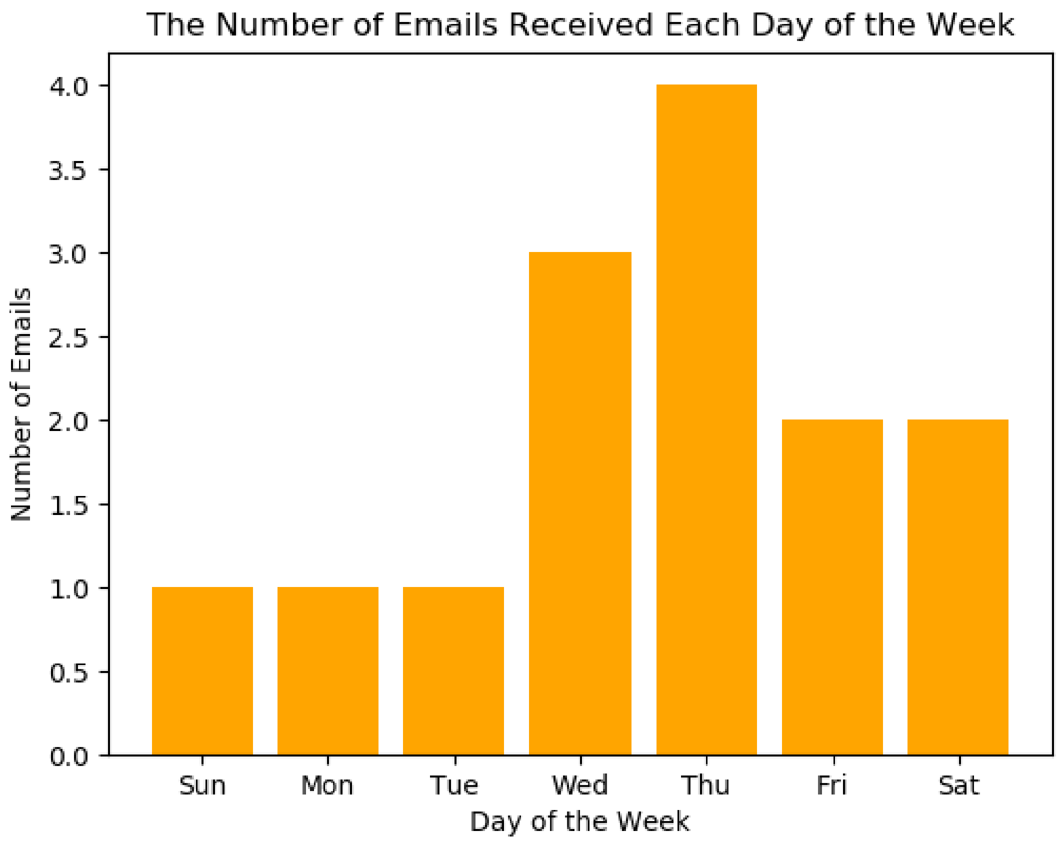


Table 2: The number of Emails Received Each Day of the Week

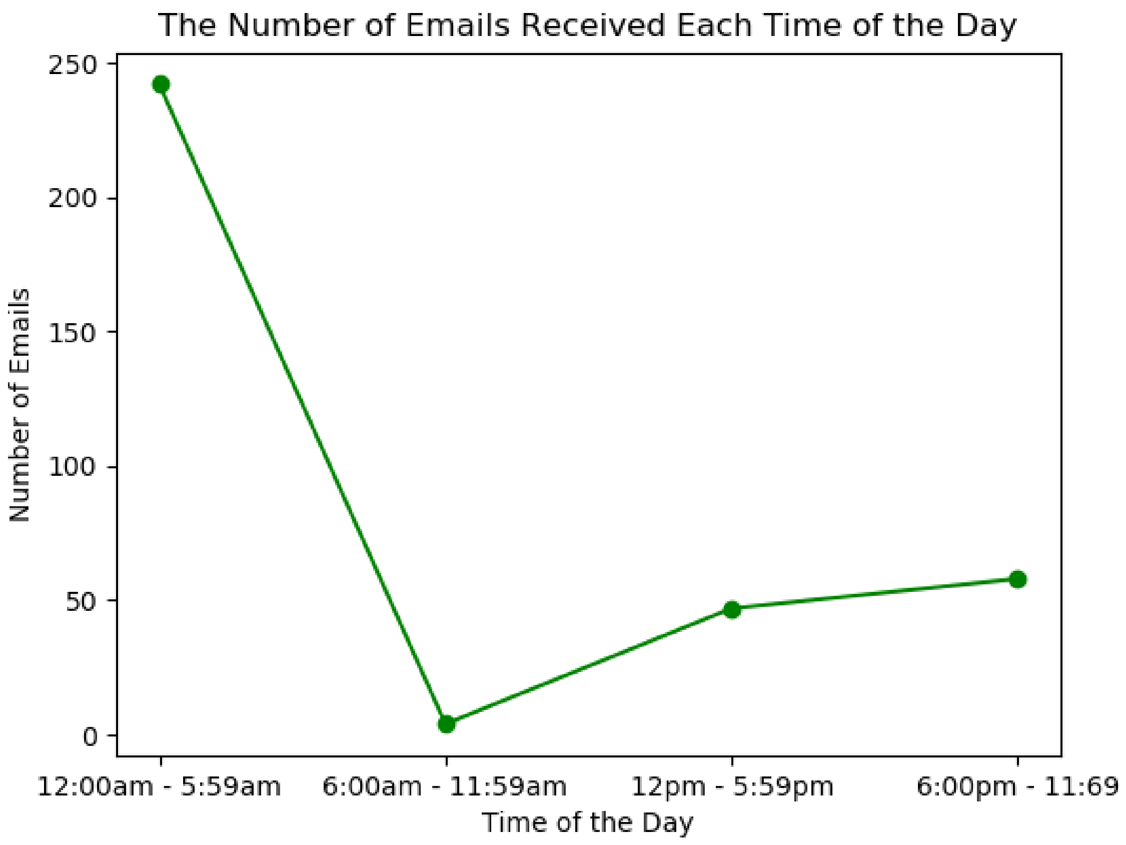


Table 3: The number of Emails Received Each Time of the Day

5. Instructions for running your code

- Set up Gmail API
  - Use this website https://developers.google.com/gmail/api/quickstart/python to turn on the Gmail API
  - Copy this command to install the library using pip

```
pip install --upgrade google-api-python-client oauth2client
```

  - Create a file called `quickstart.py` to set up the sample

```
from __future__ import print_function
from googleapiclient.discovery import build
from httplib2 import Http
from oauth2client import file, client, tools

# If modifying these scopes, delete the file token.json.
SCOPES = 'https://www.googleapis.com/auth/gmail.readonly'

def main():
    """Shows basic usage of the Gmail API.
    Lists the user's Gmail labels.
    """
    # The file token.json stores the user's access and refresh tokens, and
is
    # created automatically when the authorization flow completes for the
first
    # time.
    store = file.Storage('token.json')
    creds = store.get()
    if not creds or creds.invalid:
        flow = client.flow_from_clientsecrets('credentials.json', SCOPES)
        creds = tools.run_flow(flow, store)
    service = build('gmail', 'v1', http=creds.authorize(Http()))

    # Call the Gmail API
    results = service.users().labels().list(userId='me').execute()
    labels = results.get('labels', [])

    if not labels:
        print('No labels found.')
    else:
        print('Labels:')
        for label in labels:
            print(label['name'])

if __name__ == '__main__':
    main()
```

  - Run the Sample

```
python quickstart.py
```

  - Then login into your Google Account, and you will be asked to select one account to use for the authorization.
  - Click the **Accept** button.
- Enter email address and password
- Type in the specific sender you want
  (I used *notifications@instructure.com*(Canvas))
  (It may take a few seconds to run the code)
- Get **ZeyaoWang-Part1_Email_Data.json** file and **ZeyaoWang-Part1_Email_Data_SQL.sqlite3** database which store the data you received from the specific sender you type in
- Get **ZeyaoWang-Part2_Report.csv** about the number of emails received from each day of the week which is EXCEL file reading the report easily
- Get a bar chart **ZeyaoWang-Part3_Bar.png** show the number of emails received each day of the week intuitively
- Get a point plot **ZeyaoWang-Part3_Point.png** show the number of emails received each time of the day intuitively

6. Documentation for each function I wrote

1. Set up the Authorization: `auth()`
   This function let you enter your email address and password. If you enter the wrong password, it will show "Login Failed!".

```
def auth(user, pwd, imap_url):
    con = imaplib.IMAP4_SSL(imap_url)
    try:
        con.login(user, pwd)
    except imaplib.IMAP4.error:
        print("Login Failed!")
        sys.exit()
    return con
```

2. Extracted Emails from Different Sender: `get_specificEmail()`
   The input of this function is the specific sender. The output will be the number of emails from that specific sender. For example, I put "*notifications@instructure.com*" as the specific sender and I got 422 emails.

```
def get_specificEmail(emailAddress):
    temp_command = '(FROM "' + emailAddress + '")'
    resp, items = con.search(None, temp_command)
    items = items[0].split()
    print(len(items))
    break_ = False

    final_list = []

    for emailid in items:
        if ( break_ ):
            break
        temp_dict = {}
        resp, data = con.fetch(emailid, "(RFC822)")
        email_body = data[0][1]
        msg = email.message_from_bytes(email_body)
        temp_dict['Sender'] = msg["from"]
        temp_dict['Receiver'] = msg["to"]
        temp_dict['Subject'] = msg["subject"]
        temp_dict['DateOfSend'] = msg["date"]
        final_list.append(temp_dict)
    print(final_list)
    return final_list
```

3. Calculate the number of emails received from each day of the week : `getDayDict()`
   The input is the SQLite database which stored your database before. The output is the number of emails received each day of the week.

```
def getDayDict(cur):
    cur.execute('SELECT DataOfSend FROM Email')
    result = cur.fetchall()
    result2 = {'Sun':0, 'Mon':0, 'Tue':0, 'Wed':0, 'Thu':0, 'Fri':0, 'Sat':0, }
    for i in result:
        weekday = i[0][0:3]
        if weekday == 'Sun':
            result2['Sun'] += 1
        elif weekday == 'Mon':
            result2['Mon'] += 1
        elif weekday == 'Tue':
            result2['Tue'] += 1
        elif weekday == 'Wed':
            result2['Wed'] += 1
        elif weekday == 'Thu':
            result2['Thu'] += 1
        elif weekday == 'Fri':
            result2['Fri'] += 1
        elif weekday == 'Sat':
            result2['Sat'] += 1
    return result2
```

4. Calculate the number of emails received from each time of the day `getTimeDict()`
   The input is the SQLite database which stored your database before. The output is the number of emails received each time of the day (use 12:00am - 5:59am, 6:00am - 11:59pm, 12pm - 5:59 pm, and 6:00pm - 11:59pm).

```
def getTimeDict(cur):
    cur.execute('SELECT DataOfSend FROM Email')
    result = cur.fetchall()
    result3 = {'12:00am - 5:59am':0, '6:00am - 11:59am':0, '12pm - 5:59pm':0,
'6:00pm - 11:69':0}
    for i in result:
        weekday = int(i[0][16:18])
        if 1 <= weekday <= 5:
            result3['12:00am - 5:59am'] += 1
        elif 6<= weekday <= 11:
            result3['6:00am - 11:59am'] += 1
        elif 12 <= weekday <= 17:
            result3['12pm - 5:59pm'] += 1
        elif 18 <= weekday <= 24:
            result3['6:00pm - 11:69'] += 1
    return result3
```

5. Export data to .csv file : `dictTocsv()`
   The input is the dictionary you get from `getDayDict()` function. The output is a .csv file to show the number of emails received from each day of the week.

```
def dictTocsv(diction):
    with open('ZeyaoWang-Report.csv','w') as f:
        w=csv.writer(f)
        w.writerow(diction.keys())
        w.writerow(diction.values())
```

6. Draw a Bar Chart to show the result: `drawBarChart()`
   The input is the dictionary you get from `getDayDict()` function. The output is to a visualization of the data - a bar chart to show the result intuitively.

```
def drawBarChart(dayDict):
    names = dayDict.keys()
    values = dayDict.values()

    plt.bar(names, values)
    plt.title('The Number of Emails Received Each Day of the Week')
    plt.xlabel('Day of the Week')
    plt.ylabel('Number of Emails')
    plt.savefig("ZeyaoWang-Bar.png")
    plt.show()
```

7. Draw a Point Plot to show the result: `drawPointPlot()`
   The input is the dictionary you get from `getTimeDict()` function. The output is to a visualization of the data - a point plot to show the result intuitively.

```
def drawPointPlot(dayDict):
    names = dayDict.keys()
    values = dayDict.values()

    plt.plot(names, values, 'go-')
    plt.title('The Number of Emails Received Each Time of the Day')
    plt.xlabel('Time of the Day')
    plt.ylabel('Number of Emails')
    plt.savefig("ZeyaoWang-Part3_Point.png")
    plt.show()
```

7. document all the resources that I used

Date	Issue Description	Location of Resource	Result(did it solve the issue?)
2018/12/9	Set up Gmail API	Python Quickstart-Google Developer: https://developers.google.com/gmail/api/quickstart/python	Solved
2018/12/9	Get Emails from Lables	StackOverFlow: https://stackoverflow.com/questions/11736842/imaplib-select-on-big-inbox-too-many-arguments-for-command	Solved
2018/12/12	Get Sender From Gmail API	StackOverFlow: https://stackoverflow.com/questions/24781501/get-sender-email-from-gmail-api	Solved

Date	Issue Description	Location of Resource	Result(did it solve the issue?)
2018/12/12	Insert Data to SQLite	SQLITE Tutorial: http://www.sqlitetutorial.net/sqlite-python/insert/	Solved