# Summary

My code is scraping data from a specific Goodreads URL containing 100 books. It takes the result (Authors, Rating) and loads in a DataFrame and loads it to Google Spreadsheets. The code also sends this data in an email to a Gmail account.

# Information which the user should know

## Authentication

To access spreadsheets via Google Sheets API you need to authenticate and authorize your application.

If you plan to access spreadsheets on behalf of a bot account use Service Account.

If you’d like to access spreadsheets on behalf of end users (including yourself) use OAuth Client ID.

## Enable API Access for a Project

Head to Google Developers Console and create a new project (or select the one you already have).

In the box labeled “Search for APIs and Services”, search for “Google Drive API” and enable it.

In the box labeled “Search for APIs and Services”, search for “Google Sheets API” and enable it.

For Bots: Using Service Account

A service account is a special type of Google account intended to represent a non-human user that needs to authenticate and be authorized to access data in Google APIs [sic].

Since it’s a separate account, by default it does not have access to any spreadsheet until you share it with this account. Just like any other Google account.

Enable API Access for a Project if you haven’t done it yet.

Go to “APIs & Services > Credentials” and choose “Create credentials > Service account key”.

Fill out the form

Click “Create” and “Done”.

Press “Manage service accounts” above Service Accounts.

Press on ⋮ near recently created service account and select “Manage keys” and then click on “ADD KEY > Create new key”.

Select JSON key type and press “Create”.

Very important! Go to your spreadsheet and share it with a client\_email from the step above. Just like you do with any other Google account. If you don’t do this, you’ll get a gspread.exceptions.SpreadsheetNotFound exception when trying to access this spreadsheet from your application or a script.

Move the downloaded file to ~/.config/gspread/service\_account.json. Windows users should put this file to %APPDATA%\gspread\service\_account.json.

Create a new Python file with this code:

## For End Users: Using OAuth Client ID

Go to “APIs & Services > OAuth Consent Screen.” Click the button for “Configure Consent Screen”.

In the “1 OAuth consent screen” tab, give your app a name and fill the “User support email” and “Developer contact information”. Click “SAVE AND CONTINUE”.

There is no need to fill in anything in the tab “2 Scopes”, just click “SAVE AND CONTINUE”.

In the tab “3 Test users”, add the Google account email of the end user, typically your own Google email. Click “SAVE AND CONTINUE”.

Double check the “4 Summary” presented and click “BACK TO DASHBOARD”.

Go to “APIs & Services > Credentials”

Click “+ Create credentials” at the top, then select “OAuth client ID”.

Select “Desktop app”, name the credentials and click “Create”. Click “Ok” in the “OAuth client created” popup.

Download the credentials by clicking the Download JSON button in “OAuth 2.0 Client IDs” section.

Move the downloaded file to ~/.config/gspread/credentials.json. Windows users should put this file to %APPDATA%\gspread\credentials.json.

# The code:

## Web scarping:

import gspread

import pandas as pd #for creating dataframes

import regex as re #create regular expressions

import requests #accessing websites

from bs4 import BeautifulSoup # creating objects from html

from df2gspread import df2gspread as d2g

from oauth2client.service\_account import ServiceAccountCredentials

#accessing a Goodreads top list URL and creating and object

url = "https://www.goodreads.com/shelf/show/100-books-to-read-before-you-die"

r = requests.get(url)

soup = BeautifulSoup(r.content, 'lxml')

# Find top 5 rated authors spreadsheet

data\_frame = {}

#finding the data inside the HTML using regular expression

for rating, author in zip(soup.findAll("div", {"class": "left"}),

soup.findAll("div", {"class": "authorName\_\_container"})):

data\_frame[author.text[1:][:-2]] = str(re.findall(pattern=r'avg rating \d{1}.\d{2}',

string=str(rating.find("span", {"class": "greyText smallText"}))))[-6:-2]

f = sorted(data\_frame.items(), key=lambda item: item[1], reverse=True)[:5]

df = pd.DataFrame(f, columns=['Author', 'Average Rating'])

#accessing Google Spreadsheets using a JSON file with credentials

scope = ['https://spreadsheets.google.com/feeds',

'https://www.googleapis.com/auth/drive']

credentials = ServiceAccountCredentials.from\_json\_keyfile\_name(

'jsonFileFromGoogle.json', scope)

gc = gspread.authorize(credentials)

#Create a Google Form and uploading it to Google Spreadsheet

spreadsheet\_key = '17DeaxJHYT3yews6f9UsmvIZzPYoCXVrrbxkPI4I4BO8'

wks\_name = 'Automation'

d2g.upload(df, spreadsheet\_key, wks\_name, credentials=credentials, row\_names=True)

## Email sending

import smtplib

import ssl

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

import yaml

from Web\_Scrabing import df, data\_frame

#putting credentials in a yaml file

credentials = yaml.load(open('./credentials.yml'), Loader=yaml.FullLoader)

#printing the data

print(data\_frame)

# Create a secure SSL context

try:

smtp\_server = "smtp.gmail.com"

port = credentials['host\_port'] # For starttls

sender\_email = credentials['my\_id'] # sender's mail id

receiver\_email = credentials['recipant\_id'] # list of reciever's mail ids

password = credentials['my\_password']

context = ssl.create\_default\_context()

print('SSL run succesfully')

except Exception as e:

print(e)

# Try to log in to server and send email

try:

server = smtplib.SMTP(host=credentials['host\_add'], port=credentials['host\_port'])

server.ehlo()

server.starttls()

server.login(credentials['my\_id'], credentials['my\_password'])

print('Succesfully logged in')

except Exception as e:

print(e)

# Creation of the MIMEMultipart Object

try:

recipients = credentials['recipant\_id']

emaillist = [elem.strip().split(',') for elem in recipients]

msg = MIMEMultipart()

msg['Subject'] = "Top 5 Authors and Ratings"

msg['From'] = credentials['my\_id']

msg['To'] = credentials['recipant\_id']

html = """\

<html>

<head></head>

<body>

{0}

</body>

</html>

""".format(df.to\_html())

part1 = MIMEText(html, 'html')

msg.attach(part1)

print('Object created')

except Exception as e:

print(e)

# Sending email

try:

server.send\_message(msg)

server.quit()

print('Email sent')

except Exception as e:

print(e)