**Python script Google Calendar to Google Sheet.**

**Introduction**

The "Calendar to Sheets" script is a Python program that reads events from your Google Calendar and adds them to a Google Sheet. The program uses the Google Calendar API and the Google Sheets API to access your calendar and sheet.

When the program is run, it first prompts you to authenticate with your Google account and grant permission for the script to access your Google Calendar and Google Sheets. Once authenticated, the program reads the events happening today from your primary calendar and adds them to the Google Sheet.

If an event is already in the sheet, the program increments the event's count. If an event is not found in the sheet, the program displays a pop-up listing the unmatched events, and then displays another pop-up when script is successfully run. The program uses the PyAutoGUI library to display a notification when the script is done running.

**Libraries Used:**

- google-auth

- google-auth-oauthlib

- google-auth-httplib2

- google-api-python-client

- gspread

- oauth2client

- datetime

- os

- tkinter (only used for the message box)

**Global Variables:**

1. `SCOPES`: A list of strings that define the permissions required by the application to access the user's Google Calendar data. In this script, the scope is set to read and write events to the primary calendar.

2. `CREDENTIALS\_FILE`: A string representing the name of the JSON file that contains the client ID and secret for the Google API. The file must be stored in the same directory as the script.

3. `TOKEN\_FILE`: A string representing the name of the JSON file that contains the user's access and refresh tokens. This file is generated automatically by the script when the user logs in for the first time.

4. `SPREADSHEET\_ID`: A string representing the unique identifier of the Google Sheet where the event data is to be written. This ID can be found in the URL of the spreadsheet or use sheet title.

5. `SERVICE\_NAME`: A string representing the name of the Google API service that is being used. In this script, the service name is "calendar".

**Functions used in the script are:**

1. `main()`: The main function of the script that orchestrates the flow of the script. It sets up the Google Sheets API and the Google Calendar API, gets the events happening today from the calendar, and writes them to the Google Sheet.

2. `get\_gsheet\_credentials()`: A helper function that sets up the credentials for the Google Sheets API. It prompts the user to log in if credentials are not available, and saves the credentials in a file for later use.

3. `get\_calendar\_credentials()`: A helper function that sets up the credentials for the Google Calendar API. It prompts the user to log in if credentials are not available, and saves the credentials in a file for later use.

4. `create\_google\_sheet()`: A function that creates a new Google Sheet and returns its ID.

5. `get\_worksheet\_by\_name(sheet name)`: A function that returns a worksheet object given its name.

6. `write\_to\_worksheet(wks values)`: A function that writes a list of lists to a worksheet.

7. `write\_events\_to\_sheet(wks events)`: A function that writes a list of events to a worksheet.

8. `get\_events\_today(service)`: A function that gets the events happening today from the Google Calendar API.

9. `update\_event\_count\_in\_sheet(wks events\_today)`: A function that updates the count of events in the Google Sheet.

10. `get\_all\_events\_from\_sheet(wks)`: A function that returns a list of all events in the Google Sheet.

**Error handling mechanisms:**

1. Try-Except blocks are used to catch exceptions in different parts of the code, such as when attempting to refresh the access token or when writing to the Google Sheet.

2. `RefreshError` exceptions are handled specifically by displaying an error message to the user and prompting them to re-authorize the application.

3. `HttpError` exceptions are handled by displaying an error message with the error details returned by the Google Calendar API.

4. The script checks if the necessary credentials are available and prompts the user to log in if they are not. If the user does not log in successfully, an error message is displayed.

5. The script also checks if the required scopes are authorized and prompts the user to grant the necessary permissions if they are not.

**To implement this code, you will need to follow these steps:**

1. Install the required packages:

**pip install:**

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

pip install gspread <https://docs.gspread.org/en/latest/>

pip install tk (Used to create the pop ups)

2. Set up a Google Cloud Platform project, enable the Google Calendar API and Google Sheets API, and create credentials for a "Desktop app". Below are the links with guides to do this.

<https://console.cloud.google.com/>

<https://developers.google.com/calendar/api/quickstart/python>

<https://developers.google.com/sheets/api/quickstart/python>

3. Download the client\_secret.json file and save it to the same directory as this script.

4. Create a Google Sheet with the name "INSERT SHEET NAME" and add the headers "Event Name" and "Count" to the first row. **Note:** You will need to add the email address of the project to the permission to edit by clicking on the share button in Google sheets and adding the address which allows it to edit the sheet. Also calendar event names MUST be the same as the Google sheet.

5. Run this script, which will prompt you to log in to your Google account and authorize the script to access your Google Calendar and Google Sheets.

6. The script will then retrieve the events happening today from the Google Calendar API and compare them with the data in the Google Sheet.

7. If an event name is found in the Google Sheet, the script will update the count for that event.

8. If an event name is not found in the Google Sheet, the script will add the event name to a list of not found events and display a pop-up message indicating that the event was not found.

9. The script will then display a pop-up message indicating that it has been run.

10. You can schedule the script to run automatically at a specific time every day using Windows scheduler. Warning - Make sure the time you use in the script does not go over to the next day from your time zone and run the wrong day. **Note: I added a pop up when I get an error.**

Now let's look at the code itself:

1. **The script imports several Python libraries needed to run the code**

import datetime

import gspread

import schedule

import json

import time

import os

import tkinter as tk

from google.auth.transport.requests import Request

from google.oauth2.credentials import Credentials

from google\_auth\_oauthlib.flow import InstalledAppFlow

from googleapiclient.discovery import build

from googleapiclient.errors import HttpError

from tkinter import messagebox

1. **This code initializes the Google APIs' scopes that the script will access. In this example, the scopes are for Google Calendar and Google Sheets.**

SCOPES = ['https://www.googleapis.com/auth/calendar',

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

1. **Define the main function which will handle the authorization and access to the Google Sheets API and Google Calendar API.**

def main():

    """Shows basic usage of the Google Calendar API.

    Prints the name of the events happening today to a Google Sheet.

    """

1. **Check for previously saved credentials, and create credentials object**

if os.path.exists('token.json'):

        creds = Credentials.from\_authorized\_user\_file('token.json', SCOPES)

1. **If no valid credentials are found, uses installed app flow to authorize the user. If there is a problem with authorisation a pop up will display a message.**

if not creds or not creds.valid:

        if creds and creds.expired and creds.refresh\_token:

            try:

                creds.refresh(Request())

            except google.auth.exceptions.RefreshError:

                tk.messagebox.showerror(

                    "Error", "Your authorization token has expired or been revoked. Please re-authorize the application.")

                return

        else:

            flow = InstalledAppFlow.from\_client\_secrets\_file(

                'credentials.json', SCOPES)

            creds = flow.run\_local\_server(port=0)

*# Save the credentials for the next run*

        with open('token.json', 'w') as token:

            token.write(creds.to\_json())

1. **Build the Google Sheets API and Google Calendar API client objects.**

try:

        service = build('calendar', 'v3', credentials=creds)

1. **Get the current date in the correct format from Calendar API and extracts the list of events from the events\_result object.**

now = datetime.datetime.utcnow()

        start\_of\_day = datetime.datetime(

            now.year, now.month, now.day, 0, 0, 0, 0, datetime.timezone.utc)

        end\_of\_day = datetime.datetime(

            now.year, now.month, now.day, 23, 59, 59, 999999,

datetime.timezone.utc)

        print(f'Getting the events happening today')

        events\_result = service.events().list(calendarId='primary',

                                             timeMin=start\_of\_day.isoformat(),

timeMax=end\_of\_day.isoformat(),

singleEvents=True,

                                             orderBy='startTime').execute()

        events = events\_result.get('items', [])

1. **If no events are found a pop-up will appear with a message.**

if not events:

            root = tk.Tk()

            root.withdraw()

            messagebox.showinfo("No upcoming events",

                                "There are no events happening today.")

            return

1. **Creates and stores a list of events.**

*# Create a list of event names*

        events\_today = [event['summary'] for event in events]

1. **Authenticate and open the Google Sheet. Note: If you want to select a different tap you can call the following method and select the number of the tab:** sh.get\_worksheet(0)

sa = gspread.service\_account()

        sh = sa.open("Google Sheet name here")

        wks = sh.sheet1

1. **Get the data from the Google Sheet and initialise unmatched events list**

data = wks.get\_all\_values()

        not\_found\_events = []

1. **Code is checking if each event happening today is already in the Google Sheet and updating the count of the event if it is, or adding it to a list of not found events if it's not. Note: you can modify the second number to add data to a different cell.** wks.update\_cell(data.index(row) + 1, 3, count)

for event in events\_today:

            event\_found = False

            for row in data:

                if event in row:

*# print(f"{event} is already in the Google Sheet")*

                    count = int(row[2]) + 1

                    wks.update\_cell(data.index(row) + 1, 3, count)

                    event\_found = True

                    break

            if not event\_found:

                print(f"{event} is not in the Google Sheet")

                not\_found\_events.append(event)

1. **Code initializes the root window of a Tkinter application and hides it from view. This handles pop-up windows.**

root = tk.Tk()

        root.withdraw()

1. **Code returns a pop-up warning if there are unmatched events, and then displays a pop-up indicating script was run successfully.**

if not\_found\_events:

            messagebox.showinfo(

                "Error", f"The following events were not found:\n\n{',

'.join(not\_found\_events)}")

*# Show a pop-up message indicating that the script has been run*

        root = tk.Tk()

        root.withdraw()

        tk.messagebox.showinfo("Success", "The script has been run!")

**Conclusion:**

The "Calendar to Sheets" Python script is a useful tool for automatically retrieving and updating event information from a user's Google Calendar and recording it in a Google Sheet. The script makes use of the Google Calendar API and Google Sheets API to access and manipulate data. The script can be customized to meet specific needs by adjusting the parameters used to retrieve data and the format of the output in the Google Sheet. The script also includes error handling to ensure that the user is informed of any issues with authorization or connectivity. Overall, the "Calendar to Sheets" script can save time and streamline the process of managing event data in Google Calendar and Google Sheets.

**Main function**

This code imports the required packages and defines the scopes for the Google APIs. The SCOPES variable specifies the permissions that the script requires to access the user's Google Calendar and Google Sheets.

Overall, the `main()` function is designed to retrieve information about events happening today from the Google Calendar API, update the count of each event in a Google Sheet, and provide the user with a pop-up message indicating which events were not found and that the script has run successfully.

**Below is a step-by-step breakdown of what the `main()` function does:**

1. Checks for existing user credentials stored in a local `token.json` file. If it finds valid credentials, it uses them to authenticate the user. Otherwise, it prompts the user to log in and save the new credentials to the `token.json` file for future use.

2. Uses the authenticated credentials to call the Google Calendar API and retrieve a list of all events happening today.

3. Filters out events that are not happening today and creates a list of the names of events that are happening today.

4. Authenticates and opens a Google Sheet named "INSERT SHEET NAME" using the `gspread` library.

5. Retrieves all data from the first sheet of the opened Google Sheet and checks if each event name from the list created in step 3 is already in the data.

6. If an event name is found in the data, it updates the count of that event in the Google Sheet.

7. If an event name is not found in the data, it adds the event name to a list of not-found events.

8. Displays a pop-up message indicating which events were not found in the Google Sheet, if any.

9. Displays a pop-up message indicating that the script has been run successfully.