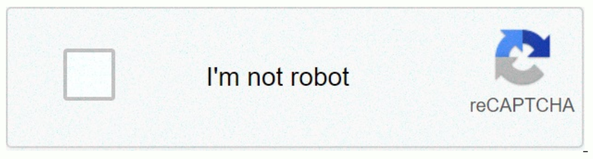

Continue

How to use google sheets api python

This post is inspired by patrick mckenzie's reminder that sometimes you don't need a database: So if you're building from a fast CRUD application for e.g. internal use, Google Docs as a backend (consumed via JSON) is *surprisingly* powerful. — Patrick McKenzie (@patio11) July 5, 2014 In this tutorial, we'll use Anton Burnashev's excellent Python package to read, write, and delete data from a Google spreadsheet with only a few lines of code. Google Drive APIs and Service Accounts At the risk of being Captain Obvious, you'll need a spreadsheet if you want to follow along with this post. If you don't have one on hand that's full of juicy data, can I suggest you get a copy of this table with contact information for all U.S. lawmakers? (Side note: Ian Webster uses this data in conjunction with Twilio to make it easy for citizens to call congress). To gain program access to the spreadsheet, you'll need to create an OAuth2 service account and credentials from the Google API Console. If you have been traumatized by the development of OAuth2 before, do not worry; service accounts are easier to use. Follow the steps and GIFs below. You'll be in and out of the console in 60 seconds (similar to Nic Cage in your favorite Nic Cage movie). Go to the Google API console. Create a new project. Click Enable API. Locate and enable the Google Drive API. Create credentials for the Web server to access the application data. Name the service account and give it the project editor role. Download the JSON file. Copy the JSON file to the code directory and rename it to client_secret.json There is the last required step to authorize your application and it's easy to miss! Find client_email inside client_secret.json. Back to the spreadsheet, click the Share button in the upper-right corner and paste the client's email into the People box to give it editing rights. Press the Submit button. If you skip this step, you'll see an error gspread.exceptions.SpreadsheetNotFound when you try to access a spreadsheet from Python. We're done with the boring part! Now to the code. Read the data from the Python table with credentials in place (you copied it to the code directory, right?) access to Google Spreadsheet in Python requires only two packages: oauth2client - enable with Google Drive API using OAuth 2.0 gspread - to interact with Google Spreadsheets Install these packages with: pip install gspread oauth 2client Then paste this code into a new file named spreadsheet.py: import gspread from oauth2client.service_account import ServiceAccountCredentials # use creds to create a client to interact with the Google Drive API scope=['] creds=ServiceAccountCredentials.from_json_keyfile_name('client_secret.json', scope) client =gspread.authorize(creds) # Find workbook by name and open first sheet # Make sure that you are using the correct name here. letter = client.open(Copy of lawmakers # Extract and print print values list_of_hashes = sheet.get_all_records() print(list_of_hashes) Run python spreadsheet.py and admire famous, well formatted data. Insert, update, and remove from the Python table We just scratched the surface of well documented and comprehensive gspreads features. For example, we extracted data to a hash list, but you can get a list of lists if you want: Or you could just pull data from one row, column or cell: sheet.row_values(1) sheet.col_values(1) sheet.cell(1, 1).value You can type in a table by changing a specific cell: sheet.update_cell(1, 1, I just wrote to a table using Python!) Or, you can insert a row into a table: row =[I am,inserting,a,row,do,a,spreadsheet,s,Python] index=1 sheet.insert_row (row, index) You can also remove a row from the table: And find out the total number of rows: Check the gspread API reference for all the details of these functions along with a few dozen others. Using Google spreadsheets with Python opens up possibilities, such as building a spreadsheet bank app as a layer of persistence, or importing data from a Google spreadsheet into jupyter notebooks and analyzing it in pandas. To start playing with Python and Twilio, check out our Python quickstarts. If you build something cool, please let me know. You can find me in gb@twilio.com or @greggyb. And if this post was helpful, please share it with someone else who might be digging. Many thanks to Devin and Sam for the reviews, on Google for the letters, and most of all to Anton for the gspread. Learn how to set up a Google Account Read and write data in Google spreadsheets using python Introduction Automation work was one of the fastest ways to achieve functional efficiency. Moreover, in this day and age, when success depends on speed, automation of countless repetitive tasks plays a key role in every industry and at the most basic level of functionality. But many of us don't understand how to automate certain tasks and end up in a loop manually doing the same things again. For example, we often spend hours a day extracting data and then copying it to tables and creating reports that lead to excessive time consumption. As a result, it would be great if we just run the script, and the data is uploaded in a table and the message is ready with just one click. There are several advantages to automating messages as if you were able to save time on data collection and removing typos and focusing more on the analysis part. This article shows you a step-by-step process for setting up your Google Account. We'll use Google APIs to read Google table data using python, and we'll update the data in the table using python. We'll read the cricket commentary data from the table and find out the number of runs scored by each batsman and then upload the results to a separate one If you're not familiar with Python, see our free Introduction to Python content tutorial Create a Google Account Read data from Google Spreadsheet update data in google tables Create a Google Account To read and update data from Google Python tables, we'll need to create a service account. This is a special type of account that is used to call authorized APIs on Google cloud services. First, make sure you have a Google Account. If you have a Google Account, follow these steps to create a Google Account. Go to the developer console. Now you're going to see something like that. Click Create Project. Then enter the project name and the name of the organization that is optional. Then click Create. Now that our project is created, we must enable the APIs that we require in this project. Click Enable APIs and Services to locate the APIs provided by Google. As a result, we will add two APIs for our project. Google Sheets API Google Drive API Then search the search bar for these APIs, and then click Enable. The Google Sheets API will look something like this. Allows you to access Google spreadsheets. You will be able to read and edit content present in spreadsheets. The Google Drive API will look something like this. Allows you to access resources from Google Drive. When you enable the required APIs in your project, it's time to create credentials for the service account. To continue, click Create Credentials. Now select the Google Drive API in the type of API you want questions. We'll call the API from a non UI based platform, so select Other non-UI (e.g. cron work, daemon). In the next question, select the app data because we don't require any user data to run our app. And we also don't use any cloud-based computing tool for our app. Finally, click the credentials I need? Button. Then share Google spreadsheets with other people and give them permission, such as editing or viewing. Similarly, we will provide access to our service account. We will give him full access so that we will be able to read as well as write tables and download json file credentials. Now the JSON file that contains the keys to access the API is downloaded. Our Google Account is ready to use. In the next section, we will read and edit the data in the table. Read the data from Google Letters We will read commentary data from the India Bangladesh Cricket Match. You can access the data here. We have ball after ball data from a complete match in the table. Now we will do a very basic task and calculate how many runs are scored each of the batsmen. We can do this by using a simple groupby in pandas. Finally, we will upload the results in a separate letter. To provide access to google sheet now, we need to access google sheet so that the API can access it. Open the JSON file that we downloaded from the developer console. Look at the client_email in the JSON file and copy it. Then click the Share button in the spreadsheet to provide access to this client email. Now we are ready to code and access the sheet using python. Below are the steps- 1. Import libraries We use gspread and oauth2client to authorize and make API calls to Google cloud services. You can install libraries using the following commands. !pip3 install gspread !pip3 install --upgrade google-api-python-client oauth2client Then we will define the scope of the application and add a JSON file that has credentials to access the API. Use a client object to open the worksheet. Just pass the letter name as an argument. To do this, you can also upload the URL of the worksheet. Access a specific worksheet: We have multiple sheets in one spreadsheet. You can access specific Google python spreadsheets by providing an index of that worksheet in get_worksheet data. For the first sheet, upload index 0 and so on. Basic API functions provide some basic functions, such as the number of columns using col_count and getting a value in a specific cell. Here are some examples of the same. 4. Get all records Then we will have all the data present in the sheet using get_all_records function. Returns the JSON string that contains the data. 5. Converting the dictionary to a data framework In data science, pandas are one of the most do not yet used libraries to perform data manipulation tasks. So first we convert the JSON string to the panda data frame. In case you are not satisfied with pandas, I highly recommend you enroll in this free course: Pandas for data analysis in Python 6. Grouping batsman Then we create groupby the number of runs scored by a batsman and upload that data frame in a separate sheet. Now we'll add this data frame to Google sheets. Update data in Google sheets Below are the steps to update data in Google sheets. First, we create a separate sheet to save the results. To do this, use the add_worksheet and upload the number of rows and columns you want and the sheet name. Then, the instance of the second sheet is made by adding an index that is 1. When you run this command, you will see a separate worksheet is created. Then convert the data frame running to a 2-D list and use the function to add values on the worksheet. Use this single line of code to update the worksheet. Then you'll see a report about the number of rows and columns updated with some additional details. End of note To summarize, in this article, we delved into understanding the different steps involved in the service account creation process. And how to read write in Google spreadsheets directly from the python console. We downloaded table data turned it into a panda data frame and created a group by table and uploaded it to the table again. This API can be very useful for automating reports. In case you want to dust off your spreadsheet concepts, I recommend the following article and of course–I hope this will help you in automating scripts and saving you a lot of your precious time. In case of any doubts, please contact us in the comments section. I'd be happy to help you. You can also read this article on our Mobile APP Related Articles