To complete this tutorial, you will need:

- To have a Google Account.
- Python 2.6 or higher installed on your development system.
- To have `pip` installed.

Step 1 - Create a New Google Cloud Platform Project

Google Cloud Platform projects allow developers to work with Google Workspace APIs, including the Google Sheets API. Keep in mind that Google sets a quota on the number of projects a given user can create. If you are using the Google Cloud Platform for the first time, then this obviously shouldn't be a problem. However, if you need a higher project quota in the future, you can always request one from Google.

- To begin, open the Google Cloud Console. Enter your Google account credentials to login.
- Once you are logged in, you will land at the **Dashboard** page where you will be able to view a summary of your existing projects. Of course, if this is your first time using the Google Cloud Console, you will not have any existing projects.
- Click on the main menu (hamburger) icon in the upper-left corner of the screen and select `IAM & Admin > Create a Project` from the drop-down menu.
- Enter a name for the new project . You will also see a **Project ID** in a small font directly underneath the project **Name** input field. The **Project ID** is created automatically based on the project name that you select. You can click on `Edit` to change the **Project ID**. However, keep in mind that once the new project is created, the **Project ID** cannot be changed.
- Optionally, enter a location for the project. The location specifies the organization or folder to which the project will belong. As a new user, the default value of this field may display `No organization` and does not need to be changed. If you used a Google account that is managed, by your employer for example, when logging in, the **Location** field may display a different value.
- Click on **Create** to create the project.

- You will be re-directed back to the **Dashboard** page. In a few minutes, you should see the newly created project listed under the **Project Info** window.

Step 2 - Enable the Google Sheets API

Now that you have created a new Google Cloud Platform project, you need to enable the Google Sheets API for the project.

- From the **Dashboard** page, click on the main menu icon in the upper-left corner again and select `APIs & Services > Library`.
- From the **Library** page, scroll down to the Google Workspace section and click on `Google Sheets API`.
- Once on the **Google Sheets API** page, click on `Enable`.
- The Google Sheets API is now enabled for the new Google Cloud Platform project created in Step 1.

Step 3 - Create a Service Account

A Python application that wants to interact with the Google Sheets API must be authenticated and authorized to access the resources that the API supports. Google provides different options for the authentication and authorization mechanisms depending on the needs of the developer. In this step, we will choose to configure a new **Service Account** for authentication and authorization purposes.

Step 3a - Create a New Service Account

- From the **Dashboard** page, click on the main menu icon in the upper-left corner again and select `APIs & Services > Credentials`.
- On the **Credentials** page, click on `Manage Service Accounts` under **Service Accounts**.

- You will be directed to the **Service Accounts** page where you need to click on `Create Service Account`.
- Enter a new name for the service account. You can choose any name that you would like. Notice that as you enter your chosen service account name, an associated `Service account ID` will be automatically generated in the **Service account ID** field. Additionally, an e-mail address matching the service account ID will also be automatically generated. For example, if you entered `some new service account` in the name field, you might see a strange looking e-mail address ending in `.iam.gserviceaccount.com` like `some-new-service-account@young-home-460928.iam.gserviceaccount.com` as the service account e-mail. Copy this e-mail address since you will use it later when connecting a Python application to a Google Sheets spreadsheet via the Google Sheets API.
- Once you have entered a name for the service account, click on `Create And Continue`.
- Click on the `Select a role` dropdown menu from the **Grant this service account access to project** section. Look for the `Basic` option on the left side of the menu and hover over it. Then, click on `Editor` from the right-side of the menu. The `Editor` role will allow a Python application using this service account to read and write to a Google Sheets spreadsheet that it is interacting with via the Google Sheets API.
- Click on `Done` to create the new service account.
- You will be redirected back to the **Service Accounts** page.

Step 3b - Create a New Key for the Service Account

- From the **Service Accounts** page, click on the options menu icon (i.e. the icon with three vertical dots) under the **Actions** label for the new service account. When the menu opens, click on `Manage Keys`.
- On the **Keys** page, click on `Add Key` and then select `Create new key`.
- The key format should already be set to JSON when the **Create private key** modal opens. Click on `Create`. A new private key will be generated and a JSON file download with the new key should automatically start. Note the location of the JSON file with the private key value as it will be needed later when connecting to the Google Sheets API from a Python application.

Step 4 - Install Required Google Libraries for Python

A Python application needs to import certain Google libraries for Python to successfully authenticate with and interact with the Google Sheets API.

Step 4a - Install the Google API Python Client Library

The Google Client Libraries provide access to Google Cloud APIs for a variety of languages. In this tutorial, we will install the Google API Python Client Library.

- You can install the Google API Python Client Library using the following command:

```
pip install google-cloud
```

Step 4b - Install the Google Authentication Libraries

The Google Authentication libraries are used for authentication with Google Workspace APIs, and in this particular case, the Google Sheets API.

- First, install the Google Authentication Library for Python using the following command:

```
pip install google-auth
```

- Second, install the Google Authentication OAuth Library for Python using the following command:

```
pip install google-auth-oauthlib
```

The required Python libraries have now been installed.

Step 5 - Configuring a Test Application in Python

With everything setup in Steps 1 - 4, you can now start programmatically interacting with Google Sheets spreadsheets via the Google Sheets API. This steps builds a simple test application to retrieve the title of a Google Sheets spreadsheet.

Step 5a - Create a New Google Sheets Spreadsheet

- In **Google Sheets**, create a new spreadsheet and give it a title such as `My New Google Sheets Spreadsheet`.
- For this simple test, the new spreadsheet does not need to have any data in it.

Step 5b - Share the New Google Sheets Spreadsheet With Your Service Account

- Click on the green **Share** button in the upper right corner of the new Google Sheets spreadsheet.
- When the dialog opens, copy the Google Cloud Service Account e-mail address from Step 3a into the **Add people and groups** field.
- Uncheck the **Notify people** option and click **Send**.

Step 5c - Record the URL Identifier for the New Google Sheets Spreadsheet

- Your new Google Sheets spreadsheet will have a URL like:

```
https://docs.google.com/spreadsheets/d/8VaaiCuZ2q09IVndzU54s1RtxQreAxgFNaUPf9su5hK0/edit#gid=0
```

- Record the portion of the URL string starting after `d/` and ending before `/edit`.

- In other words, you would write down `8VaaiCuZ2q09IVndzU54s1RtxQreAxgFNaUPf9su5hK0` using this example.

Step 5d - Configure the Test Application in Python

- To keep things simple, put the `key.json` file generated in Step 3b in the same directory where you will create the test application in Python.
- Build the test application as follows:

```
from google.oauth2 import service_account
from googleapiclient.discovery import build

spreadsheet_id = ENTER_YOUR_SPREADSHEET_ID FROM_STEP_5c_HERE_WITH_QUOTES
# For example:
# spreadsheet_id = "8VaaiCuZ2q09IVndzU54s1RtxQreAxgFNaUPf9su5hK0"

credentials = service_account.Credentials.from_service_account_file("key.json",
scopes=["https://www.googleapis.com/auth/spreadsheets"])
service = build("sheets", "v4", credentials=credentials)

request = service.spreadsheets().get(spreadsheetId=spreadsheet_id, ranges=[],
includeGridData=False)
sheet_props = request.execute()

print(sheet_props["properties"]["title"])

# Output:
# My New Google Sheets Spreadsheet
```

Conclusion

In this tutorial, you created a new Google Cloud Platform project, enabled the Google Sheets API for that project, created a new service account to authenticate with the Google Sheets API, installed all required Google libraries for Python, and built a simple test application. As mentioned in the introduction, the possibilities are almost endless when programmatically interfacing with Google Sheets. See Google Sheets for Developers documentation for more information.