

Python quickstart

Quickstarts explain how to set up and run an app that calls a Google Workspace API.

Google Workspace quickstarts use the API client libraries to handle some details of the authentication and authorization flow. We recommend that you use the client libraries for your own apps. This quickstart uses a simplified authentication approach that is appropriate for a testing environment. For a production environment, we recommend learning about [authentication and authorization](/workspace/guides/auth-overview) (/workspace/guides/auth-overview) before [choosing the access credentials](/workspace/guides/create-credentials#choose_the_access_credential_that_is_right_for_you)

(/workspace/guides/create-credentials#choose_the_access_credential_that_is_right_for_you) that are appropriate for your app.

Create a Python command-line application that makes requests to the Google Sheets API.

Objectives

- Set up your environment.
- Install the client library.
- Set up the sample.
- Run the sample.

Prerequisites

To run this quickstart, you need the following prerequisites:

- Python 3.10.7 or greater
- The [pip](https://pypi.python.org/pypi/pip) (https://pypi.python.org/pypi/pip) package management tool
- [A Google Cloud project](/workspace/guides/create-project) (/workspace/guides/create-project).
- A Google Account.

Set up your environment

To complete this quickstart, set up your environment.

Enable the API


Before using Google APIs, you need to turn them on in a Google Cloud project. You can turn on one or more APIs in a single Google Cloud project.

- In the Google Cloud console, enable the Google Sheets API.

[Enable the API](https://console.cloud.google.com/flows/enableapi?apiid=sheets.googleapis.com) (https://console.cloud.google.com/flows/enableapi?apiid=sheets.googleapis.com)

Configure the OAuth consent screen

If you're using a new Google Cloud project to complete this quickstart, configure the OAuth consent screen and add yourself as a test user. If you've already completed this step for your Cloud project, skip to the next section.

1. In the Google Cloud console, go to Menu  > **APIs & Services** > **OAuth consent screen**.

[Go to OAuth consent screen](https://console.cloud.google.com/apis/credentials/consent) (https://console.cloud.google.com/apis/credentials/consent)

2. For **User type** select **Internal**, then click **Create**.
3. Complete the app registration form, then click **Save and Continue**.
4. For now, you can skip adding scopes and click **Save and Continue**. In the future, when you create an app for use outside of your Google Workspace organization, you must change the **User type** to **External**, and then, add the authorization scopes that your app requires.
5. Review your app registration summary. To make changes, click **Edit**. If the app registration looks OK, click **Back to Dashboard**.

Authorize credentials for a desktop application

To authenticate end users and access user data in your app, you need to create one or more OAuth 2.0 Client IDs. A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, you must create a separate client ID for each platform.

1. In the Google Cloud console, go to Menu  > **APIs & Services** > **Credentials**.

[Go to Credentials](https://console.cloud.google.com/apis/credentials) (https://console.cloud.google.com/apis/credentials)

2. Click **Create Credentials** > **OAuth client ID**.
3. Click **Application type** > **Desktop app**.
4. In the **Name** field, type a name for the credential. This name is only shown in the Google Cloud console.
5. Click **Create**. The OAuth client created screen appears, showing your new Client ID and Client secret.
6. Click **OK**. The newly created credential appears under **OAuth 2.0 Client IDs**.
7. Save the downloaded JSON file as `credentials.json`, and move the file to your working directory.

Install the Google client library

- Install the Google client library for Python:

```
pip install --upgrade google-api-python-client google-auth-httpplib2 googl
```

Configure the sample

1. In your working directory, create a file named `quickstart.py`.
2. Include the following code in `quickstart.py`:

`sheets/quickstart/quickstart.py`

[Hub](https://github.com/googleworkspace/python-samples/blob/main/sheets/quickstart/quickstart.py) (https://github.com/googleworkspace/python-samples/blob/main/sheets/quickstart/quickstart.py)

```
import os.path

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

# If modifying these scopes, delete the file token.json.
```

```

SCOPES = ["https://www.googleapis.com/auth/spreadsheets.readonly"]

# The ID and range of a sample spreadsheet.
SAMPLE_SPREADSHEET_ID = "1BxiMVs0XRA5nFMdKvBdBZjgmUUqptlbs740gvE2upms"
SAMPLE_RANGE_NAME = "Class Data!A2:E"

def main():
    """Shows basic usage of the Sheets API.
    Prints values from a sample spreadsheet.
    """
    creds = None
    # The file token.json stores the user's access and refresh tokens, and
    # created automatically when the authorization flow completes for the f
    # time.
    if os.path.exists("token.json"):
        creds = Credentials.from_authorized_user_file("token.json", SCOPES)
    # If there are no (valid) credentials available, let the user log in.
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        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())

    try:
        service = build("sheets", "v4", credentials=creds)

        # Call the Sheets API
        sheet = service.spreadsheets()
        result = (
            sheet.values()
            .get(spreadsheetId=SAMPLE_SPREADSHEET_ID, range=SAMPLE_RANGE_NAME)
            .execute()
        )
        values = result.get("values", [])

        if not values:
            print("No data found.")
            return

        print("Name, Major:")
        for row in values:
            # Print columns A and E, which correspond to indices 0 and 4.

```

```
        print(f"{row[0]}, {row[4]}")
    except HttpError as err:
        print(err)

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

Run the sample

1. In your working directory, build and run the sample:

```
python3 quickstart.py
```

2. The first time you run the sample, it prompts you to authorize access:

- a. If you're not already signed in to your Google Account, sign in when prompted. If you're signed in to multiple accounts, select one account to use for authorization.
- b. Click **Accept**.

Your Python application runs and calls the Google Sheets API.

Authorization information is stored in the file system, so the next time you run the sample code, you aren't prompted for authorization.

Next steps

- [Troubleshoot authentication and authorization issues](/sheets/api/troubleshoot-authentication-authorization)
(/sheets/api/troubleshoot-authentication-authorization)
- [Sheets API reference documentation](/sheets/api/reference/rest) (/sheets/api/reference/rest)
- [Google APIs Client for Python documentation](/api-client-library/python) (/api-client-library/python)
- [Google Sheets API PyDoc documentation](https://developers.google.com/resources/api-libraries/documentation/sheets/v4/python/latest/index%2Ehtml)
(https://developers.google.com/resources/api-libraries/documentation/sheets/v4/python/latest/index%2Ehtml)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2024-03-08 UTC.