MLOps Blog

How to Deal With Files in Google Colab: Everything You Need to Know

Table of contents

6 min

21st April, 2023

Machine Learning Tools

Google Colaboratory is a free Jupyter notebook environment that runs on Google's cloud servers, letting the user leverage backend hardware like GPUs and TPUs. This lets you do everything you can in a Jupyter notebook hosted in your local machine, without requiring the installations and setup for hosting a notebook in your local machine.

Colab comes with (almost) all the setup you need to start coding, but what it doesn't have out of the box is your datasets! How do you access your data from within Colab?

In this article we will talk about:

- How to load data to Colab from a multitude of data sources
- How to write back to those data sources from within Colab
- Limitations of Google Colab while working with external files

Directory and file operations in Google Colab

Since Colab lets you do everything which you can in a locally hosted Jupyter notebook, you can also use shell commands like `ls, dir, pwd, cd, cat, echo`, et cetera using line-magic (%) or bash (!).

To browse the directory structure, you can use the file-explorer pane on the left.

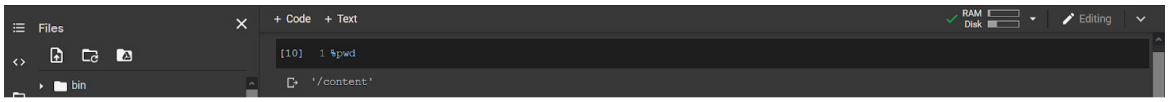
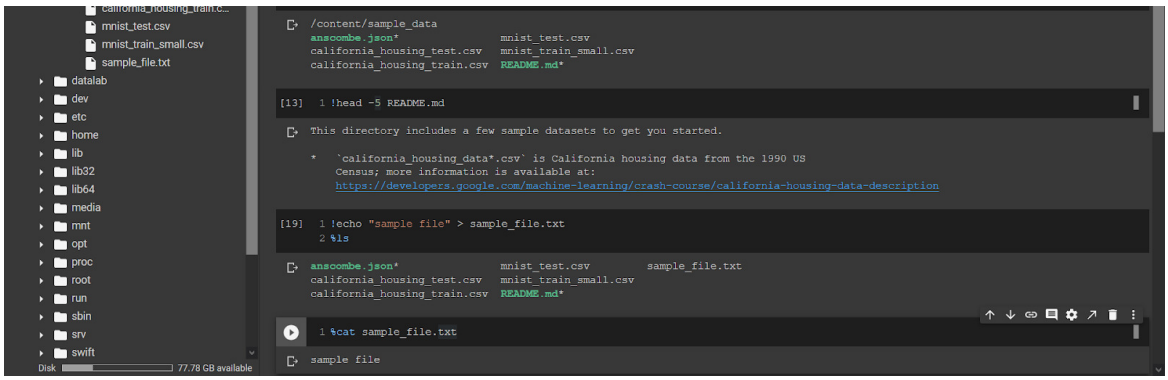


Table of contents



How to upload files to and download files from Google Colab

Since a Colab notebook is hosted on Google's cloud servers, there's no direct access to files on your local drive (unlike a notebook hosted on your machine) or any other environment by default.

However, Colab provides various options to connect to almost any data source you can imagine. Let us see how.

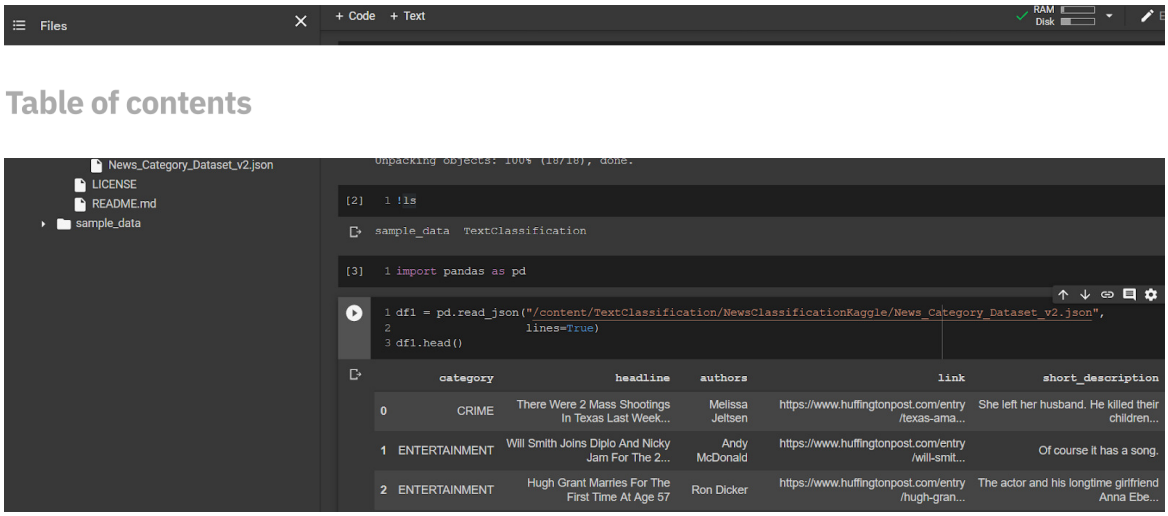
Accessing GitHub from Google Colab

You can either clone an entire GitHub repository to your Colab environment or access individual files from their raw link.

Clone a GitHub repository

You can clone a GitHub repository into your Colab environment in the same way as you would in your local machine, using git `clone`. Once the repository is cloned, refresh the file-explorer to browse through its contents.

Then you can simply read the files as you would in your local machine.



Load individual files directly from GitHub

In case you just have to work with a few files rather than the entire repository, you can load them directly from GitHub without needing to clone the repository to Colab.

To do this:

1. click on the file in the repository,
2. click on **View Raw**,
3. copy the URL of the raw file,
4. use this URL as the location of your file.

```
[4]  1 df3 = pd.read_json("News_Category_Dataset_v2.json", lines=True)
     2 df3.head()
```

	category	headline	authors	link
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...	Melissa Jeltsen	https://www.huffingtonpost.com/entry/texas-ama...
1	ENTERTAINMENT	Will Smith Joins Diplo And Nicky Jam For The 2...	Andy McDonald	https://www.huffingtonpost.com/entry/will-smit...

Accessing Local File System to Google Colab

Table of contents

Access local files through the file-explorer

Uploading files from local file system through file-explorer

You can either use the upload option at the top of the file-explorer pane to upload any file(s) from your local file system to Colab in the present working directory.

To upload files directly to a subdirectory you need to:

1. Click on the three dots visible when you hover above the directory

2. Select the “upload” option.

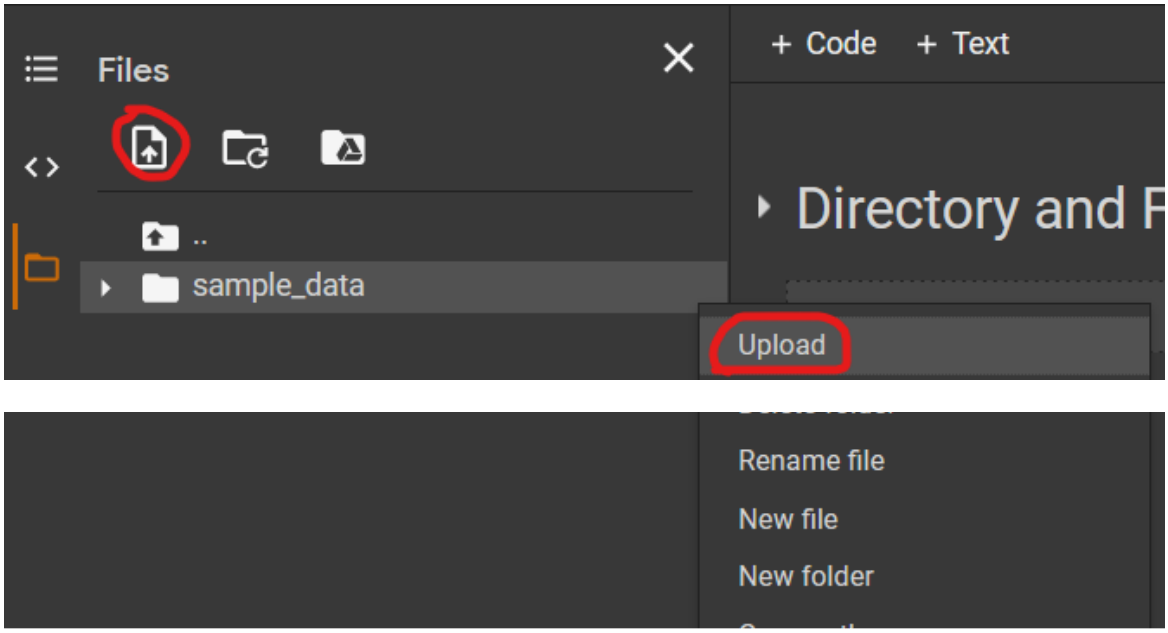
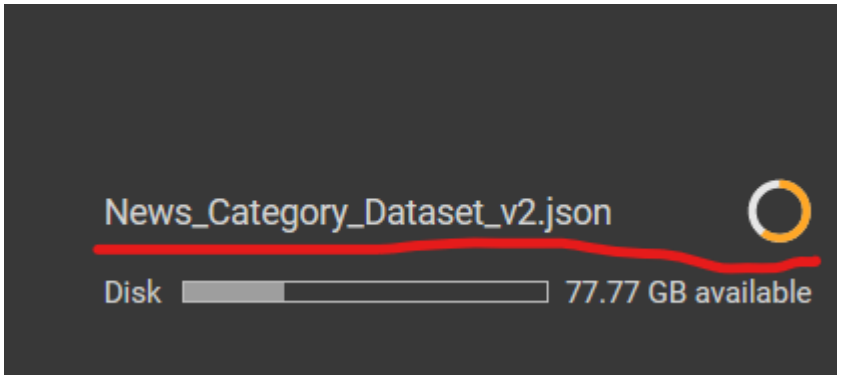


Table of contents

---

3. Select the file(s) you wish to upload from the “File Upload” dialog window.

4. Wait for the upload to complete. The upload progress is shown at the bottom of the file-explorer pane.



Once the upload is complete, you can read from the file as you would normally.

```
[4]  1 df3 = pd.read_json("News_Category_Dataset_v2.json", lines=True)
     2 df3.head()
```

	category	headline	authors	link
0	CRIME	There Were 2 Mass Shootings In Texas Last Week...	Melissa Jeltsen	https://www.huffingtonpost.com/entry/texas-ama...
1	ENTERTAINMENT	Will Smith Joins Diplo And Nicky Jam For The 2...	Andy McDonald	https://www.huffingtonpost.com/entry/will-smit...

Downloading files to local file system through file-explorer

Click on the three dots which are visible while hovering above the filename, and select the “download” option.

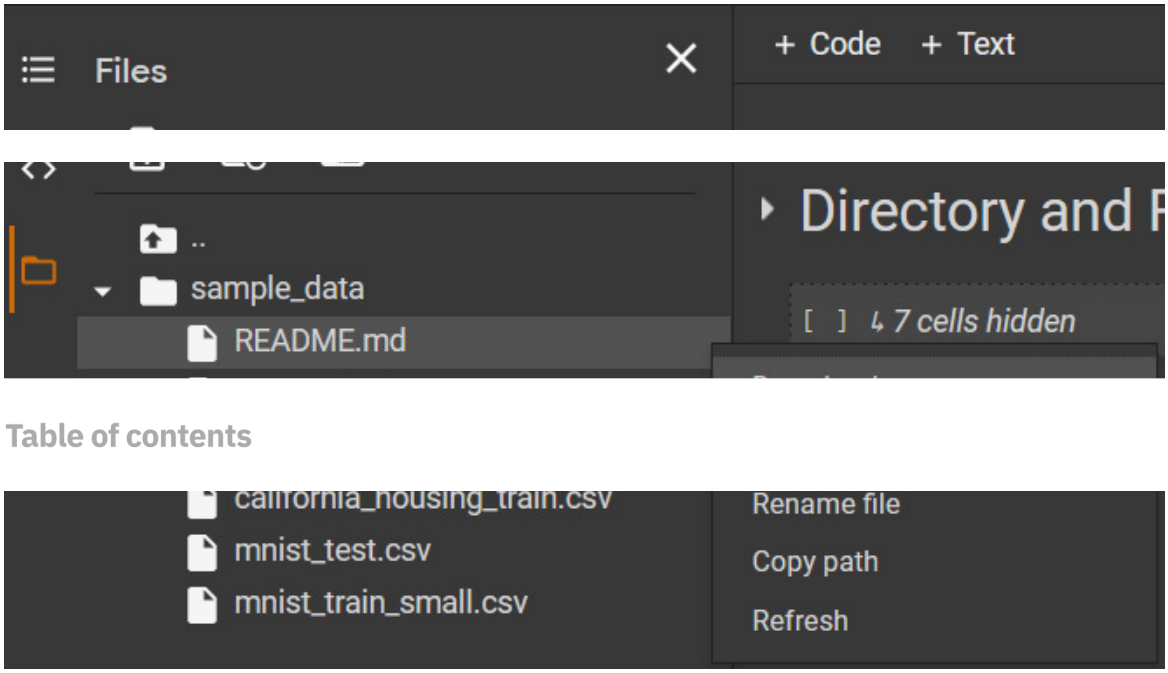


Table of contents

Accessing local file system using Python code

This step requires you to first import the `files` module from the `google.colab library`:

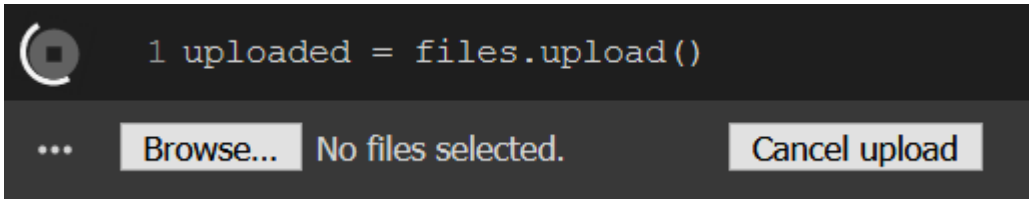
```
from google.colab import files
```

Uploading files from local file system using Python code

You use the upload method of the `files` object:

```
uploaded = files.upload()
```

Running this opens the File Upload dialog window:



Select the file(s) you wish to upload, and then wait for the upload to complete. The upload progress is displayed:

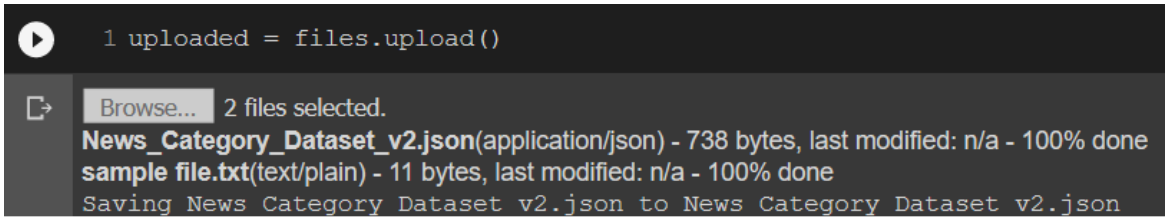


Table of contents

The `uploaded` object is a dictionary having the filename and content as it's key-value pairs:

```
1 uploaded
```

```
{'News_Category_Dataset_v2.json': b'{"category": "CRIME", "headline": "There Were
 'sample file.txt': b'sample text'}
```

Once the upload is complete, you can either read it as any other file from colab:

```
df4 = pd.read_json("News_Category_Dataset_v2.json",
lines=True)
```

Or read it directly from the `uploaded` dict using the `io` library:

```
import io
df5 =
pd.read_json(io.BytesIO(uploaded['News_Category_Dataset_v2.js
on']), lines=True)
```

Make sure that the filename matches the name of the file you wish to load.

Downloading files from Colab to local file system using Python code:

The `download` method of the files object can be used to download any file from colab to your local drive. The download progress is displayed, and once the download completes, you can choose where to save it in your local machine.

```
files.download('/content/sample_data/california_housing_train.csv')
```

Accessing Google Drive from Google

Table of contents

You can use the `drive` module from `google.colab` to mount your entire Google Drive to Colab by:

1. Executing the below code which will provide you with an authentication link

```
from google.colab import drive
drive.mount('/content/gdrive')
```

2. Open the link

3. Choose the Google account whose Drive you want to mount

4. Allow Google Drive Stream access to your Google Account

5. Copy the code displayed, paste it in the text box as shown below, and press Enter

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=

Enter your authorization code:
```

Once the Drive is mounted, you'll get the message "`Mounted at /content/gdrive`", and you'll be able to browse through the contents of your Drive from the file-explorer pane.

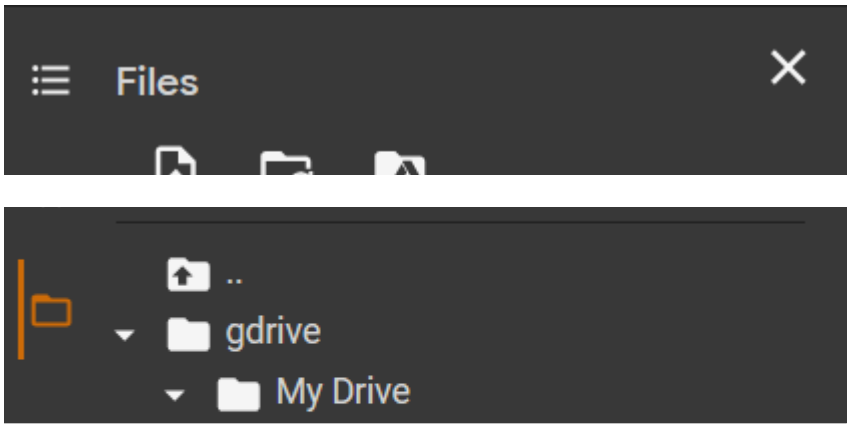
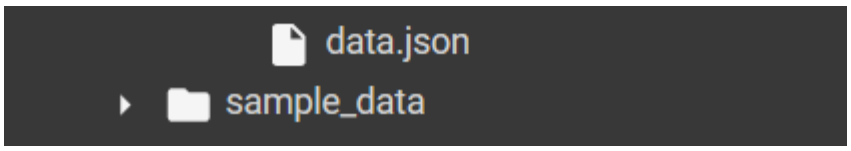


Table of contents

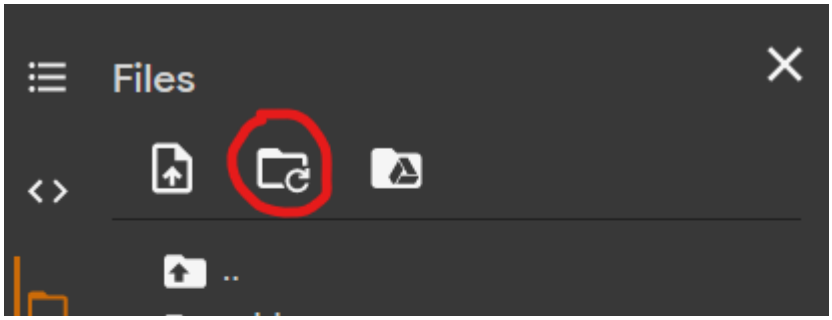


Now you can interact with your Google Drive as if it was a folder in your Colab environment. Any changes to this folder will reflect directly in your Google Drive. You can read the files in your Google Drive as any other file.

You can even write directly to Google Drive from Colab using the usual file/directory operations.

```
!touch "/content/gdrive/My Drive/sample_file.txt"
```

This will create a file in your Google Drive, and will be visible in the file-explorer pane once you refresh it:



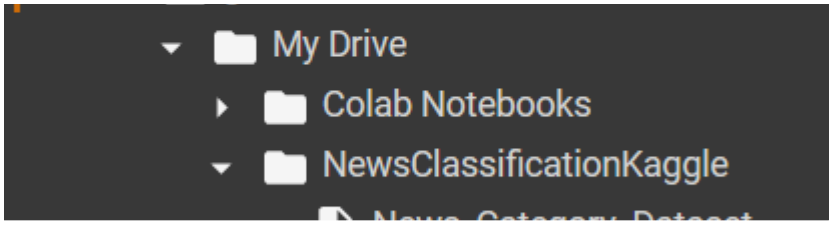
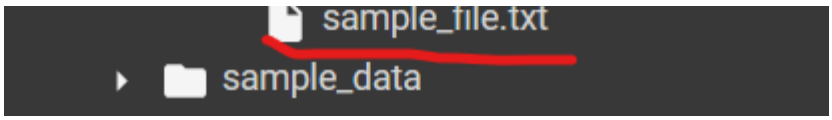
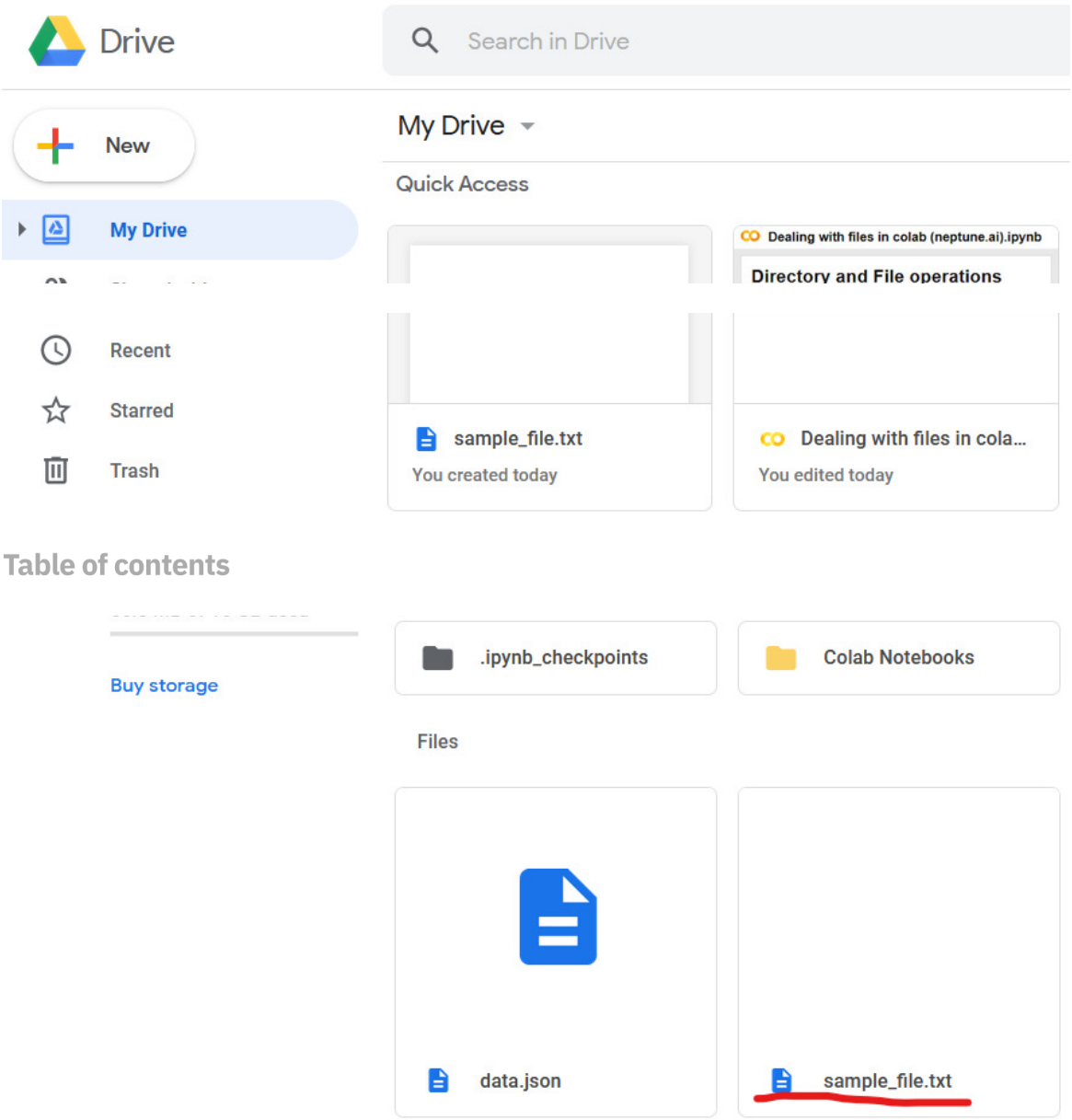


Table of contents





Accessing Google Sheets from Google Colab

To access Google Sheets:

1. You need to first authenticate the Google account to be linked with Colab by running the code below:

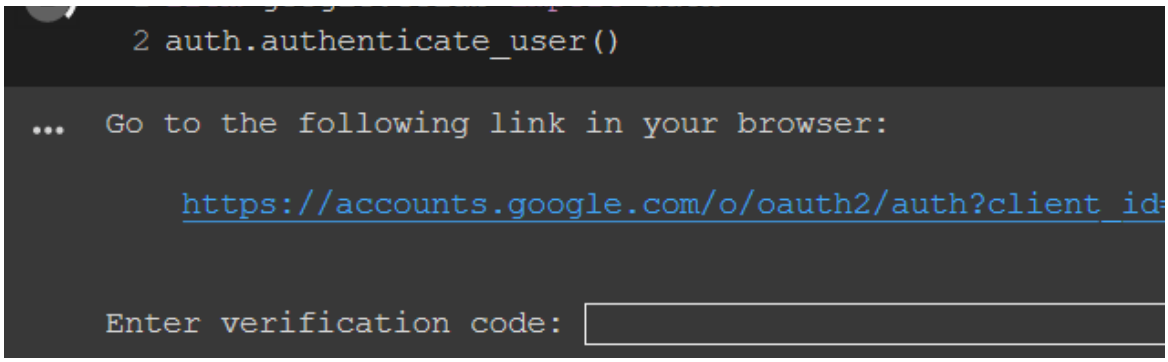
```
from google.colab import auth
auth.authenticate_user()
```

2. Executing the above code will provide you with an authentication link. Open the link,

4. Allow Google Cloud SDK to access your Google Account,

5. Finally copy the code displayed and paste it in the text box shown, and hit Enter.

Table of contents

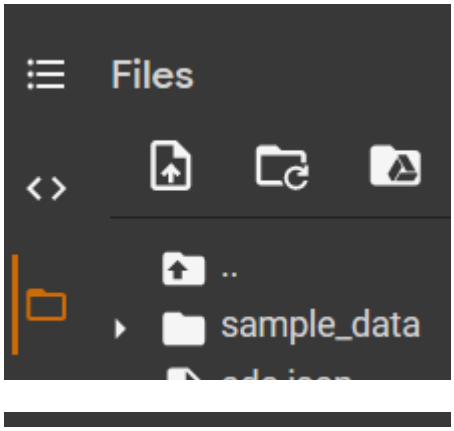


To interact with Google Sheets, you need to import the preinstalled `gspread` library. And to authorize `gspread` access to your Google account, you need the `GoogleCredentials` method from the preinstalled `oauth2client.client` library:

```
import gspread
from oauth2client.client import GoogleCredentials

gc =
gspread.authorize(GoogleCredentials.get_application_default()
)
```

Once the above code is run, an Application Default Credentials (ADC) JSON file will be created in the present working directory. This contains the credentials used by `gspread` to access your Google account.



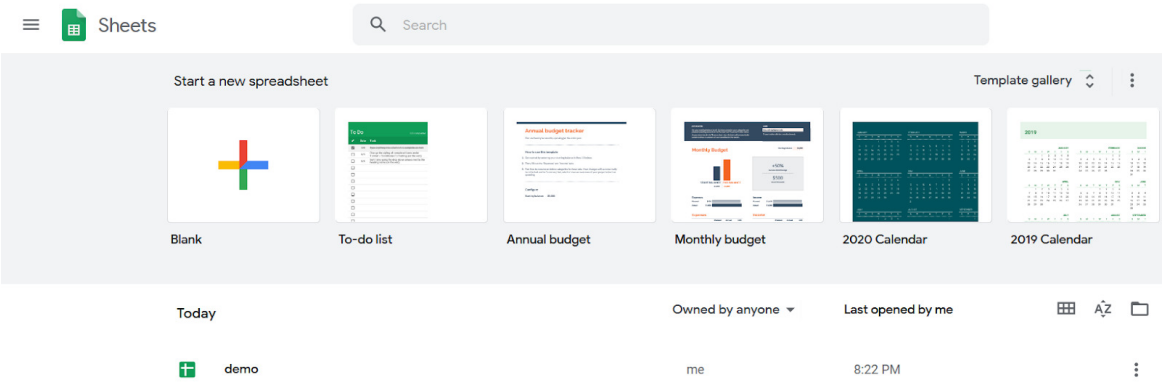
Once this is done, you can now create or load Google sheets directly from your Colab environment.

Table of contents

1. Use the `gc` object's create method to `create` a workbook:

```
wb = gc.create('demo')
```

2. Once the workbook is created, you can view it in sheets.google.com.



3. To write values to the workbook, first open a worksheet:

```
ws = gc.open('demo').sheet1
```

4. Then select the cell(s) you want to write to:

```
cells = ws.range('A1:B2')
cells
```

```
[<Cell R1C1 ''>, <Cell R1C2 ''>, <Cell R2C1 ''>, <Cell R2C2 ''>]
```

5. This creates a list of cells with their index (R1C1) and value (currently blank)

You can modify the individual cells by updating their value attribute:

```
cells[0].value = 'as'
cells[1].value = 'easy'
cells[2].value = 'as'
```

Table of contents

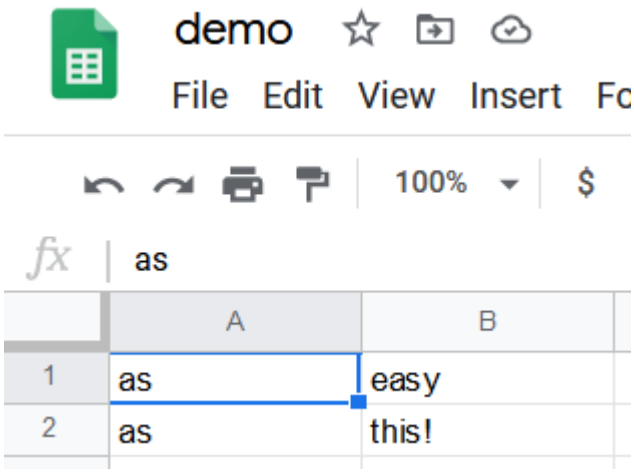
```
[<Cell R1C1 'as'>, <Cell R1C2 'easy'>, <Cell R2C1 'as'>, <Cell R2C2 'this!'>]
```

6. To update these cells in the worksheet, use the update_cells method:

```
ws.update_cells(cells)
```

```
{'spreadsheetId': '10nlyR8oClsBYhfyMOeR8Hgc3E6CVnm5C2hoPvlR9IL0',
 'updatedCells': 4,
 'updatedColumns': 2,
 'updatedRange': 'Sheet1!A1:B2',
 'updatedRows': 2}
```

7. The changes will now be reflected in your Google Sheet.



Downloading data from a Google Sheet

1. Use the `gc` object's open method to open a workbook:

```
wb = gc.open('demo')
```

2. Then read all the rows of a specific worksheet by using the `get_all_values` method:

```
[['as', 'easy'], ['as', 'this!']]
```

Table of contents

3. To load these to a dataframe, you can use the DataFrame object's `from_record` method:

```
1 pd.DataFrame.from_records(rows)
```

	0	1
0	as	easy
1	as	this!

Accessing Google Cloud Storage (GCS) from Google Colab

You need to have a Google Cloud Project (GCP) to use GCS. You can create and access your GCS buckets in Colab via the preinstalled `gsutil` command-line utility.

1. First specify your project ID:

```
project_id = '<project_ID>'
```

2. To access GCS, you've to authenticate your Google account:

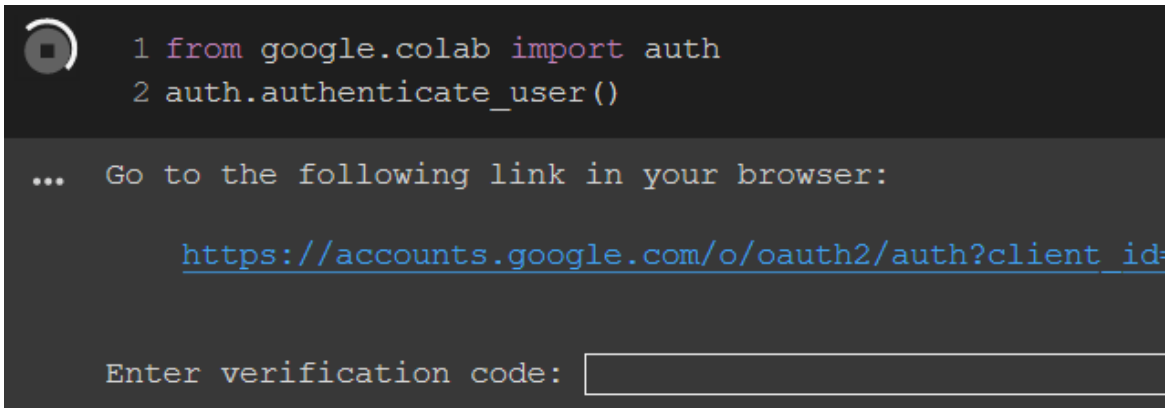
```
from google.colab import auth
```

3. Executing the above code will provide you with an authentication link. Open the link,

Table of contents

5. Allow Google Cloud SDK to access your Google Account,

6. Finally copy the code displayed and paste it in the text box shown, and hit Enter.



7. Then you configure `gsutil` to use your project:

```
!gcloud config set project {project_id}
```

8. You can make a bucket using the make bucket (`mb`) command. GCP buckets must have a universally unique name, so use the preinstalled `uuid` library to generate a Universally Unique ID:

```
import uuid
bucket_name = f'sample-bucket-{uuid.uuid1()}'
!gsutil mb gs://{bucket_name}
```

9. Once the bucket is created, you can upload a file from your colab environment to it:

```
!gsutil cp /tmp/to_upload.txt gs://{bucket_name}/
```

Table of contents


```
!gsutil cp gs://{bucket_name}/{filename} {download_location}
```

Once the download has finished, the file will be visible in the Colab file-explorer pane in the download location specified.

Accessing AWS S3 from Google Colab

You need to have an AWS account, configure IAM, and generate your access key and secret access key to be able to access S3 from Colab. You also need to install the `awscli` library to your colab environment:

1. Install the `awscli` library

```
!pip install awscli
```

2. Once installed, configure AWS by running `aws configure`:

```
!aws configure

AWS Access Key ID [None]: <your_access_key>
AWS Secret Access Key [None]: <your_secret_access_key>
Default region name [None]:
Default output format [None]:
```

enter.

Then you can download any file from S3:

Table of contents

`filepath_on_s3` can point to a single file, or match multiple files using a pattern.

You will be notified once the download is complete, and the downloaded file(s) will be available in the location you specified to be used as you wish.

To upload a file, just reverse the source and destination arguments:

```
!aws s3 cp ./{upload_from} s3://{bucket_name} --recursive --
exclude "*" --include {file_to_upload}
```

`file_to_upload` can point to a single file, or match multiple files using a pattern.

You will be notified once the upload is complete, and the uploaded file(s) will be available in your S3 bucket in the folder specified: https://s3.console.aws.amazon.com/s3/buckets/**{bucket_name}/{folder}**/?region=**{region}**

Accessing Kaggle datasets from Google Colab

To download datasets from Kaggle, you first need a Kaggle account and an API token.

Token".

2. Open the kaggle.json file, and copy its contents. It should be in the form of {"username":"########", "key":"################################"}.

Table of contents

```
!mkdir ~/.kaggle #create the .kaggle folder in your root
directory
!echo '<PASTE_CONTENTS_OF_KAGGLE_API_JSON>' >
~/.kaggle/kaggle.json #write kaggle API credentials to
kaggle.json
!chmod 600 ~/.kaggle/kaggle.json  # set permissions
!pip install kaggle #install the kaggle library
```

4. Once the kaggle.json file has been created in Colab, and the Kaggle library has been installed, you can search for a dataset using

```
!kaggle datasets list -s {KEYWORD}
```

5. And then download the dataset using

```
!kaggle datasets download -d {DATASET NAME} -p
/content/kaggle/
```

The dataset will be downloaded and will be available in the path specified (`/content/kaggle/` in this case).

Accessing MySQL databases from Google Colab

1. You need to import the preinstalled `sqlalchemy` library to work with relational databases:

Table of contents

2. Enter the connection details and create the engine:

```
HOSTNAME = 'ENTER_HOSTNAME'
USER = 'ENTER_USERNAME'
PASSWORD = 'ENTER_PASSWORD'
DATABASE = 'ENTER_DATABASE_NAME'

connection_string = f'mysql+pymysql://{MYSQL_USER}:
{MYSQL_PASSWORD}@{MYSQL_HOSTNAME}/{MYSQL_DATABASE}'

engine = sqlalchemy.create_engine(connection_string)
```

3. Finally, just create the SQL query, and load the query results to a dataframe using `pd.read_sql_query()`:

```
query = f"SELECT * FROM {DATABASE}.{TABLE}"

import pandas as pd
df = pd.read_sql_query(query, engine)
```

Limitations of Google Colab while working with Files

One important caveat to remember while using Colab is that the files you upload to it won't be available forever. Colab is a temporary environment with an idle timeout of 90 minutes and an absolute timeout of 12 hours. This means that the runtime will disconnect if it has remained idle for 90 minutes, or if it has been in use for 12 hours. On disconnection, you lose all your variables, states, installed packages, and files and will be connected to an entirely new and clean environment on reconnecting.

Also, Colab has a disk space limitation of 108 GB, of which only 77 GB is

Table of contents

Conclusion

Google Colab is a great tool for individuals who want to harness the power of high-end computing resources like GPUs, without being restricted by their price.

In this article, we have gone through most of the ways you can supercharge your Google Colab experience by reading external files or data in Google Colab and writing from Google Colab to those external data sources.

Depending on your use-case, or how your data architecture is set-up, you can easily apply the above-mentioned methods to connect your data source directly to Colab, and start coding!

Other resources

- Getting Started with Google CoLab | How to use Google Colab
- External data: Local Files, Drive, Sheets and Cloud Storage
- Importing Data to Google Colab — the CLEAN Way
- Get Started: 3 Ways to Load CSV files into Colab | by A Apte
- Downloading Datasets into Google Drive via Google Colab | by Kevin Luk

More about How to Deal With Files in Google Colab: Everything You Need to Know

Check out our **product resources** and **related articles** below:

How to Use SHAP Values to Optimize and Debug ML Models

Read more

Related article

Table of contents

Related article

How to Build ML Model Training Pipeline

Read more

Related article

What Does GPT-3 Mean For the Future of MLOps? With David Hershey

Read more

Explore more content topics:

Computer Vision | General | Machine Learning Tools | ML Experiment Tracking | ML Model Development | MLOps | Natural Language Processing | Reinforcement Learning | Tabular Data | Time Series

Table of contents

Newsletter

Top MLOps articles, case studies, events (and more) in your inbox every month.

Your e-mail

Get Newsletter

PRODUCT

SOLUTIONS

DOCUMENTATION

COMPARE

COMMUNITY

COMPANY

The Best MLOps Tools MLOps at a Reasonable Scale ML Metadata Store
MLOps: What, Why, and How Experiment Tracking in Machine Learning

Terms of service Privacy policy

Table of contents