This notebook provides recipes for loading and saving data from external sources.

# Local file system

## Uploading files from your local file system

files.upload returns a dictionary of the files which were uploaded. The dictionary is keyed by the file name and values are the data which were uploaded.

from google.colab import files  
  
uploaded = files.upload()  
  
for fn in uploaded.keys():  
 print('User uploaded file "{name}" with length {length} bytes'.format(  
 name=fn, length=len(uploaded[fn])))

## Downloading files to your local file system

files.download will invoke a browser download of the file to your local computer.

from google.colab import files  
  
with open('example.txt', 'w') as f:  
 f.write('some content')  
  
files.download('example.txt')

# Google Drive

You can access files in Drive in a number of ways, including: - Mounting your Google Drive in the runtime’s virtual machine - Using a wrapper around the API such as [PyDrive](https://pythonhosted.org/PyDrive/) - Using the [native REST API](https://developers.google.com/drive/v3/web/about-sdk)

Examples of each are below.

## Mounting Google Drive locally

The example below shows how to mount your Google Drive on your runtime using an authorization code, and how to write and read files there. Once executed, you will be able to see the new file (foo.txt) at <https://drive.google.com/>.

This only supports reading, writing, and moving files; to programmatically modify sharing settings or other metadata, use one of the other options below.

**Note:** When using the ‘Mount Drive’ button in the file browser, no authentication codes are necessary for notebooks that have only been edited by the current user.

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

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client\_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect\_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response\_type=code  
Enter your authorization code:  
··········  
Mounted at /content/drive

with open('/content/drive/My Drive/foo.txt', 'w') as f:  
 f.write('Hello Google Drive!')  
!cat /content/drive/My\ Drive/foo.txt

Hello Google Drive!

drive.flush\_and\_unmount()  
print('All changes made in this colab session should now be visible in Drive.')

All changes made in this colab session should now be visible in Drive.

## PyDrive

The examples below demonstrate authentication and file upload/download using PyDrive. More examples are available in the [PyDrive documentation](https://pythonhosted.org/PyDrive/).

from pydrive.auth import GoogleAuth  
from pydrive.drive import GoogleDrive  
from google.colab import auth  
from oauth2client.client import GoogleCredentials

Authenticate and create the PyDrive client.

auth.authenticate\_user()  
gauth = GoogleAuth()  
gauth.credentials = GoogleCredentials.get\_application\_default()  
drive = GoogleDrive(gauth)

Create and upload a text file.

uploaded = drive.CreateFile({'title': 'Sample upload.txt'})  
uploaded.SetContentString('Sample upload file content')  
uploaded.Upload()  
print('Uploaded file with ID {}'.format(uploaded.get('id')))

Uploaded file with ID 14vDAdqp7BSCQnoougmgylBexIr2AQx2T

Load a file by ID and print its contents.

downloaded = drive.CreateFile({'id': uploaded.get('id')})  
print('Downloaded content "{}"'.format(downloaded.GetContentString()))

Downloaded content "Sample upload file content"

## Drive REST API

In order to use the Drive API, we must first authenticate and construct an API client.

from google.colab import auth  
auth.authenticate\_user()  
from googleapiclient.discovery import build  
drive\_service = build('drive', 'v3')

With this client, we can use any of the functions in the [Google Drive API reference](https://developers.google.com/drive/v3/reference/). Examples follow.

### Creating a new Drive file with data from Python

First, create a local file to upload.

with open('/tmp/to\_upload.txt', 'w') as f:  
 f.write('my sample file')  
  
print('/tmp/to\_upload.txt contains:')  
!cat /tmp/to\_upload.txt

/tmp/to\_upload.txt contains:  
my sample file

Upload it using the [files.create](https://developers.google.com/drive/v3/reference/files/create) method. Further details on uploading files are available in the [developer documentation](https://developers.google.com/drive/v3/web/manage-uploads).

from googleapiclient.http import MediaFileUpload  
  
file\_metadata = {  
 'name': 'Sample file',  
 'mimeType': 'text/plain'  
}  
media = MediaFileUpload('/tmp/to\_upload.txt',   
 mimetype='text/plain',  
 resumable=True)  
created = drive\_service.files().create(body=file\_metadata,  
 media\_body=media,  
 fields='id').execute()  
print('File ID: {}'.format(created.get('id')))

File ID: 1Cw9CqiyU6zbXFD9ViPZu\_3yX-sYF4W17

After executing the cell above, you will see a new file named ‘Sample file’ at <https://drive.google.com/>.

### Downloading data from a Drive file into Python

Download the file we uploaded above.

file\_id = created.get('id')  
  
import io  
from googleapiclient.http import MediaIoBaseDownload  
  
request = drive\_service.files().get\_media(fileId=file\_id)  
downloaded = io.BytesIO()  
downloader = MediaIoBaseDownload(downloaded, request)  
done = False  
while done is False:  
 # \_ is a placeholder for a progress object that we ignore.  
 # (Our file is small, so we skip reporting progress.)  
 \_, done = downloader.next\_chunk()  
  
downloaded.seek(0)  
print('Downloaded file contents are: {}'.format(downloaded.read()))

Downloaded file contents are: b'my sample file'

In order to download a different file, set file\_id above to the ID of that file, which will look like “1uBtlaggVyWshwcyP6kEI-y\_W3P8D26sz”.

# Google Sheets

Our examples below use the open-source [gspread](https://github.com/burnash/gspread) library for interacting with Google Sheets.

Import the library, authenticate, and create the interface to Sheets.

from google.colab import auth  
auth.authenticate\_user()  
  
import gspread  
from google.auth import default  
creds, \_ = default()  
  
gc = gspread.authorize(creds)

Below is a small set of gspread examples. Additional examples are available at the [gspread GitHub page](https://github.com/burnash/gspread#more-examples).

## Creating a new sheet with data from Python

sh = gc.create('My cool spreadsheet')

After executing the cell above, you will see a new spreadsheet named ‘My cool spreadsheet’ at [https://sheets.google.com](https://sheets.google.com/).

Open our new sheet and add some random data.

worksheet = gc.open('My cool spreadsheet').sheet1  
  
cell\_list = worksheet.range('A1:C2')  
  
import random  
for cell in cell\_list:  
 cell.value = random.randint(1, 10)  
  
worksheet.update\_cells(cell\_list)

{'spreadsheetId': '1dsQeN0YzXuM387l\_CuyEbsYzL2ew9TJFzR-E-RQnwxs',  
 'updatedCells': 6,  
 'updatedColumns': 3,  
 'updatedRange': 'Sheet1!A1:C2',  
 'updatedRows': 2}

## Downloading data from a sheet into Python as a Pandas DataFrame

Read back the random data that we inserted above and convert the result into a [Pandas DataFrame](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html).

worksheet = gc.open('My cool spreadsheet').sheet1  
  
# get\_all\_values gives a list of rows.  
rows = worksheet.get\_all\_values()  
print(rows)  
  
import pandas as pd  
pd.DataFrame.from\_records(rows)

[['6', '3', '4'], ['7', '2', '1']]

0

1

2

0

6

3

4

1

7

2

1

# Google Cloud Storage (GCS)

In order to use Colaboratory with GCS, you’ll need to create a [Google Cloud project](https://cloud.google.com/storage/docs/projects) or use a pre-existing one.

Specify your project ID below:

project\_id = 'Your\_project\_ID\_here'

Files in GCS are contained in [buckets](https://cloud.google.com/storage/docs/key-terms#buckets).

Buckets must have a globally-unique name, so we generate one here.

import uuid  
bucket\_name = 'colab-sample-bucket-' + str(uuid.uuid1())

In order to access GCS, we must authenticate.

from google.colab import auth  
auth.authenticate\_user()

GCS can be accessed via the gsutil command-line utility or via the native Python API.

## gsutil

First, we configure gsutil to use the project we specified above by using gcloud.

!gcloud config set project {project\_id}

Updated property [core/project].

Create a local file to upload.

with open('/tmp/to\_upload.txt', 'w') as f:  
 f.write('my sample file')  
  
print('/tmp/to\_upload.txt contains:')  
!cat /tmp/to\_upload.txt

/tmp/to\_upload.txt contains:  
my sample file

Make a bucket to which we’ll upload the file ([documentation](https://cloud.google.com/storage/docs/gsutil/commands/mb)).

!gsutil mb gs://{bucket\_name}

Creating gs://colab-sample-bucket-44971372-baaf-11e7-ae30-0242ac110002/...

Copy the file to our new bucket ([documentation](https://cloud.google.com/storage/docs/gsutil/commands/cp)).

!gsutil cp /tmp/to\_upload.txt gs://{bucket\_name}/

Copying file:///tmp/to\_upload.txt [Content-Type=text/plain]...  
/ [1 files][ 14.0 B/ 14.0 B]   
Operation completed over 1 objects/14.0 B.

Dump the contents of our newly copied file to make sure everything worked ([documentation](https://cloud.google.com/storage/docs/gsutil/commands/cat)).

!gsutil cat gs://{bucket\_name}/to\_upload.txt

my sample file

#@markdown Once the upload has finished, the data will appear in the Cloud Console storage browser for your project:  
print('https://console.cloud.google.com/storage/browser?project=' + project\_id)

https://console.cloud.google.com/storage/browser?project=Your\_project\_ID\_here

Finally, we’ll download the file we just uploaded in the example above. It’s as simple as reversing the order in the gsutil cp command.

!gsutil cp gs://{bucket\_name}/to\_upload.txt /tmp/gsutil\_download.txt  
   
# Print the result to make sure the transfer worked.  
!cat /tmp/gsutil\_download.txt

Copying gs://colab-sample-bucket483f20dc-baaf-11e7-ae30-0242ac110002/to\_upload.txt...  
/ [1 files][ 14.0 B/ 14.0 B]   
Operation completed over 1 objects/14.0 B.   
my sample file

## Python API

These snippets based on [a larger example](https://github.com/GoogleCloudPlatform/storage-file-transfer-json-python/blob/master/chunked_transfer.py) that shows additional uses of the API.

First, we create the service client.

from googleapiclient.discovery import build  
gcs\_service = build('storage', 'v1')

Create a local file to upload.

with open('/tmp/to\_upload.txt', 'w') as f:  
 f.write('my sample file')  
  
print('/tmp/to\_upload.txt contains:')  
!cat /tmp/to\_upload.txt

/tmp/to\_upload.txt contains:  
my sample file

Create a bucket in the project specified above.

# Use a different globally-unique bucket name from the gsutil example above.  
import uuid  
bucket\_name = 'colab-sample-bucket-' + str(uuid.uuid1())  
  
body = {  
 'name': bucket\_name,  
 # For a full list of locations, see:  
 # https://cloud.google.com/storage/docs/bucket-locations  
 'location': 'us',  
}  
gcs\_service.buckets().insert(project=project\_id, body=body).execute()  
print('Done')

Done

Upload the file to our newly created bucket.

from googleapiclient.http import MediaFileUpload  
  
media = MediaFileUpload('/tmp/to\_upload.txt',   
 mimetype='text/plain',  
 resumable=True)  
  
request = gcs\_service.objects().insert(bucket=bucket\_name,   
 name='to\_upload.txt',  
 media\_body=media)  
  
response = None  
while response is None:  
 # \_ is a placeholder for a progress object that we ignore.  
 # (Our file is small, so we skip reporting progress.)  
 \_, response = request.next\_chunk()  
  
print('Upload complete')

Upload complete

#@markdown Once the upload has finished, the data will appear in the Cloud Console storage browser for your project:  
print('https://console.cloud.google.com/storage/browser?project=' + project\_id)

https://console.cloud.google.com/storage/browser?project=Your\_project\_ID\_here

Download the file we just uploaded.

from apiclient.http import MediaIoBaseDownload  
  
with open('/tmp/downloaded\_from\_gcs.txt', 'wb') as f:  
 request = gcs\_service.objects().get\_media(bucket=bucket\_name,  
 object='to\_upload.txt')  
 media = MediaIoBaseDownload(f, request)  
  
 done = False  
 while not done:  
 # \_ is a placeholder for a progress object that we ignore.  
 # (Our file is small, so we skip reporting progress.)  
 \_, done = media.next\_chunk()  
  
print('Download complete')

Download complete

Inspect the downloaded file.

!cat /tmp/downloaded\_from\_gcs.txt

my sample file