Published in Python in Plain English

Abhinav Anand Follow
Jun 1, 2021 · 5 min read · Listen

Save

Uploading and Downloading Files from Google Drive using Python

Transfer files from Google Drive in just 5 lines of Python code.

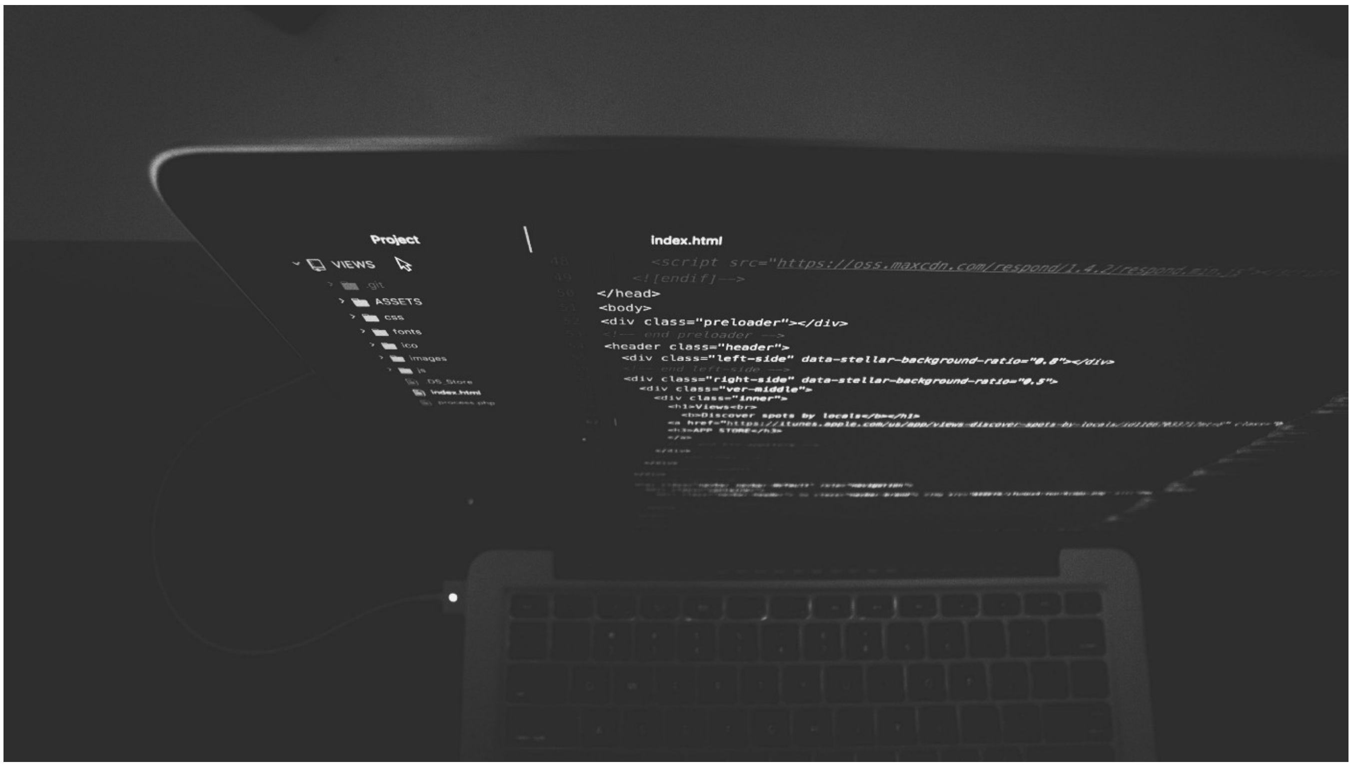

Photo by Nate Grant on Unsplash

Intro

Google Drive is the most used cloud storage platform. A need for some minimal yet effective tool to transfer contents to and from Drive is important!

There is a number of such libraries already present with Pydrive being the most amazing. Pydrive has a lot of functionalities but there are certain major functionalities that it lacks. Especially, when it comes to the most used feature i.e. Bulk uploading/downloading.

Thus with a goal to address the bulk uploading/downloading functionality in a way that is easy-to-use and yet covers all the users' requirements. I built this library Zdrive.

Zdrive

ZDrive is a lightweight and easy-to-use Python library to upload and download content from Google Drive.

Let's explore the feature it provides:

- Upload folders in a specified folder in the Drive maintaining the **same directory structure** as present locally.
- Download folders from a specified folder in the Drive to a specified folder in the PC maintaining the **same directory structure** as present in the Drive.
- Can sustain minor network interruptions.
- Download or Upload the whole directory in less than 5 lines of code.
- Support: * **OS**: Linux, Windows, Mac * **Language**: Python 2.x, 3.x

> *Note: Pydrive doesn't enable you to download contents to a specified location. Also while downloading/uploading you can't maintain the same directory structure as in the input folder.*

Maintaining the same directory structure:

Let's take the case of uploading a folder to Drive!

Suppose the input folder specified by the user that needs to be uploaded to Drive is: `/User/abhinavanand/Documents/parking_lot/` (let's call it `input_folder`). Also, suppose currently the structure of `input_folder` is as shown below.

```
input_folder/
|-- README.md
|-- bin
|   |-- parking_lot
|   |-- parking_lot.sh
|   |-- run_functional_tests
|   |-- setup
|   `-- setup.sh
|-- file_inputs.txt
|-- requirements.txt
|-- src
|   |-- __init__.py
|   |-- lib
|   |   `-- utils.py
|   |
|   |-- main.py
|   `-- models
|       |-- __init__.py
|       |-- car.py
|       |-- lot.py
|       `-- parking_lot.py
`-- tests
    |-- test_libs.py
    |-- test_main.py
    `-- test_models.py
```

Now suppose the user wants this `input_folder` to be uploaded to a specific location(folder) in the drive, let's call it `output_folder` . So below is the final directory structure of the `output_folder` once the upload completes.

```
output_folder/
|-- README.md
|-- bin
|   |-- parking_lot
|   |-- parking_lot.sh
|   |-- run_functional_tests
|   |-- setup
|   `-- setup.sh
|-- file_inputs.txt
|-- requirements.txt
|-- src
|   |-- __init__.py
|   |-- lib
|   |   `-- utils.py
|   |
|   |-- main.py
|   `-- models
|       |-- __init__.py
|       |-- car.py
```

```
    |       |-- lot.py
    |       `-- parking_lot.py
    `-- tests
        |-- test_libs.py
        |-- test_main.py
        `-- test_models.py
```

So **Zdrive** makes sure, that both directory structure is the same so that it is easy for the user to find the files in Drive and not being overwhelmed by the number of contents. Awesome, isn't it?

> *Vice-versa is done while downloading contents from Drive.*

Now, that we know the features of Zdrive, let's jump into coding and see how easy it is to download and upload content using Zdrive.

Initial Setup

Follow the article to get your drive-API credentials. Once you have the `clients-secret.json` file, rename it to `credentials.json` and place it in the same folder where you'll be running the script.

Downloading

Google Drive is semantic (also called `tag-based`) file system meaning it stores files not based on their location, but based on an ID. Thus for performing any action related to a file/folder in Drive, we would need the IDs of the file/folder. Zdrive allows you to retrieve a list of files/folders present inside the Drive along with their IDs.

The `Downloader` class in **Zdrive** allows you to download folder/files from the Drive as shown below.

```python
from zdrive import Downloader

output_directory = "/home/abhinav/Documents"
d = Downloader()

# folder which want to download from Drive
folder_id = 'XXXX-YYYY-ZZZZ'
d.downloadFolder(folder_id, destinationFolder=output_directory)
```

downloader.py hosted with ❤ by GitHub view raw

> *Note: The* `destinationFolder` *is an optional parameter which specifies where the contents downloaded from Drive should go, if the parameter is not provided then Zdrive will create a folder named* `drive_content` *in the folder where the script is run and download the contents there.*

location inside the local PC. Data can be uploaded at the ROOT level of the Drive or inside any specific folder in the Drive.

Since the folder to be uploaded can contain multiple levels of child folders, one inside another which can significantly slow down the process thus the level of child directories to be uploaded can be decided by `max_depth` parameter as shown in the example below. Although, this is an *optional parameter* with a default value of 5.

```python
from zdrive import Uploader

input_directory = "/home/abhinav/Downloads"
u = Uploader()

# creating a folder in Drive where we want to upload the folder
parent_folder_id = u.createFolder(name="Data")
result = u.uploadFolder(input_directory, max_depth=3,
                        parentId=parent_folder_id)
print(result)
```

uploader.py hosted with ❤ by GitHub view raw

Output:

```
'{
    "files":
        {
            "/Users/abhinavanand/Downloads/test/def.pdf": "1pJNIu-
0oyzaUgjLvnf6-3mk81iwLBXyS"
        },
    "folders":
        {
        "/Users/abhinavanand/Downloads/test/test-level-1":
            {
                "files":
```

```
                {
                    "/Users/abhinavanand/Downloads/test/test-level-
1/abc.pdf": "1YwZs__92yzWdM2e7Nc2atF5lzLnyYV9i"
                },
                "folders": {},
                "id": "1zzh hGImg94SnzrMC8LdH1vgbO3LMksD"
            }
        }
}'
```

182

> *Note: The `parentId` is also an optional parameter, if no `parentId` is specified, then `Uploader()` would upload the contents from local PC to the ROOT level in Drive. Also, In case of a minor internet interruption(~10–15 secs) the upload would be paused and once the internet connection is stable. The uploading will get resumed.*

Conclusion

Since at the time of writing this blog Zdrive is just the initial phase where the goal was to address the major reasons for using a script to automate Google Drive operations, I'm sure there are a lot of improvements that can be done.

If you guys have found it useful then do drop a ⭐ (star) to the GitHub Repository. Do raise a PR in case you want to add some more amazing functionalities. Also, if you have an idea then you can just file an enhancement in the issue tracker.

GitHub Link:

ab-anand/ZDrive

A lightweight and easy to use Python library to upload and download contents from Google Drive. Google Drive is the…

github.com

PyPi Link:

ZDrive

A lightweight and easy to use Python library to upload and download contents from Google Drive. Google Drive is the…

pypi.org

More content at plainenglish.io

Python Programming Coding Google Drive Software Development