Работаем с API Google Drive с помощью Python

Решил написать достаточно подробную инструкцию о том как работать с API Google Drive v3 с помощью клиентской библиотеки Google API для Python. Статья будет полезна тем, кому приходится часто работать с документами в Google Drive: скачивать и загружать новые документы, удалять файлы, создавать папки.

Также я покажу пример того как можно с помощью API скачивать файлы Google Sheets в формате Excel, или наоборот: заливать в Google Drive файл Excel в виде документа Google Sheets.

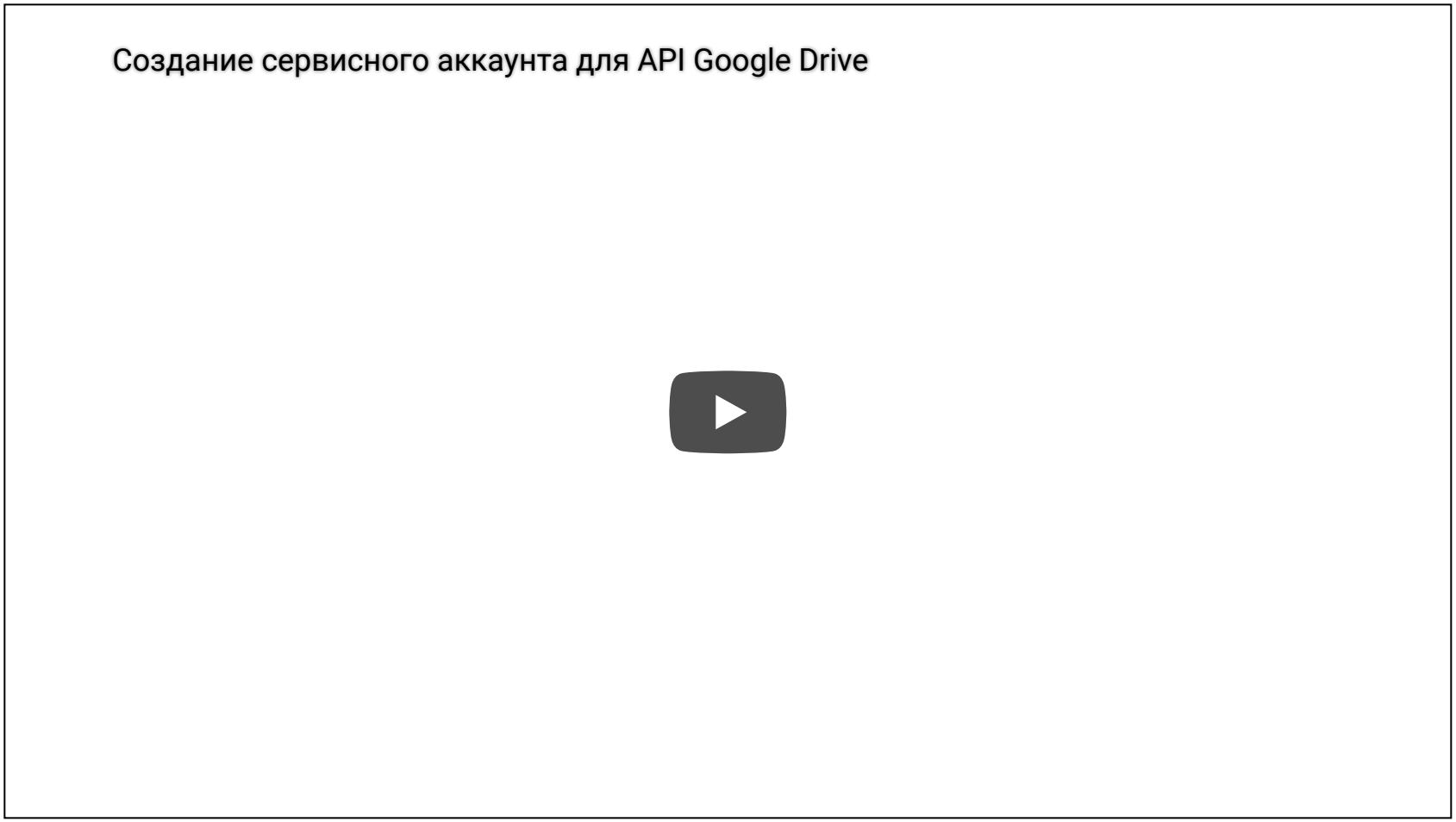
Использование API Google Drive может быть полезным для автоматизации различной рутины, связанной с отчетностью. Например, я использую его для того, чтобы по расписанию загружать заранее подготовленные отчеты в папку Google Drive, к которой есть доступ у конечных потребителей отчетов.

Все примеры на Python 3.

Создание сервисного аккаунта и получение ключа

Прежде всего создаем сервисный аккаунт в консоли Google Cloud и для email сервисного аккаунта открываем доступ на редактирование необходимых папок. Не забудьте добавить в папку файлы, если их там нет, потому что файл нам понадобится, когда мы будем выполнять первый пример — скачивание файлов из Google Drive.

Я записал небольшой скринкаст, чтобы показать как получить ключ для сервисного аккаунта в формате JSON.



Установка клиентской библиотеки Google API и получение доступа к API

Сначала устанавливаем клиентскую библиотеку Google API для Python

```
pip install --upgrade google-api-python-client
```

Дальше импортируем нужные модули или отдельные функции из библиотек.

Ниже будет небольшое описание импортируемых модулей. Это для тех кто хочет понимать, что импортирует, но большинство просто может скопировать импорты и вставить в ноутбук :)

Модуль service_account из google.oauth2 понадобится нам для авторизации с помощью сервисного аккаунта.

Классы MediaIoBaseDownload и MediaFileUpload, как ясно из названий, пригодятся, чтобы скачать или загрузить файлы. Эти классы импортируются из googleapiclient.http

Функция build из googleapiclient.discovery позволяет создать ресурс для обращения к API, то есть это некая абстракция над REST API Drive, чтобы удобнее обращаться к методам API.

```
from google.oauth2 import service_account
from googleapiclient.http import MediaIoBaseDownload,MediaFileUpload
from googleapiclient.discovery import build
import pprint
import io

pp = pprint.PrettyPrinter(indent=4)
```

Указываем Scopes. Scopes — это перечень возможностей, которыми будет обладать сервис, созданный в скрипте. Ниже приведены Scopes, которые относятся к API Google Drive (из официальной документации):

Scopes	
https://www.googleapis.com/auth/drive	See, edit, create, and delete all of your Google Drive files
https://www.googleapis.com/auth/drive.appdata	View and manage its own configuration data in your Google Drive
https://www.googleapis.com/auth/drive.file	View and manage Google Drive files and folders that you have opened or created with this app
https://www.googleapis.com/auth/drive.metadata	View and manage metadata of files in your Google Drive
https://www.googleapis.com/auth/drive.metadata.readonly	View metadata for files in your Google Drive
https://www.googleapis.com/auth/drive.photos.readonly	View the photos, videos and albums in your Google Photos
https://www.googleapis.com/auth/drive.readonly	See and download all your Google Drive files
https://www.googleapis.com/auth/drive.scripts	Modify your Google Apps Script scripts' behavior

Как видно, разные Scope предоставляют разный уровень доступа к данным. Нас интересует Scope «https://www.googleapis.com/auth/drive», который позволяет просматривать, редактировать, удалять или создавать файлы на Google Диске.

Также указываем в переменной SERVICE_ACCOUNT_FILE путь к файлу с ключами сервисного аккаунта.

```
SCOPES = ['https://www.googleapis.com/auth/drive']
SERVICE_ACCOUNT_FILE = '/home/makarov/Google Drive Test-fc4f3aea4d98.json'
```

Создаем Credentials (учетные данные), указав путь к сервисному аккаунту, а также заданные Scopes. А затем создаем сервис, который будет использовать 3ю версию REST API Google Drive, отправляя запросы из-под учетных данных credentials.

```
credentials = service_account.Credentials.from_service_account_file(
        SERVICE_ACCOUNT_FILE, scopes=SCOPES)
service = build('drive', 'v3', credentials=credentials)
```

Получение списка файлов

Теперь можно получить список файлов и папок, к которым имеет доступ сервис. Для этого выполним запрос list, выдающий список файлов, со следующими параметрами:

pageSize — количество результатов выдачи. Можете смело ставить максимальное значение 1000. У меня стоит 10 результатов, чтобы показать как быть, когда нужно получить результаты по следующей страницы результатов

параметр files() в fields — параметр, указывающий, что нужно возвращать список файлов, где в скобках указан список полей для файлов, которые нужно показывать в результатах выдачи. Со всеми возможными полями можно познакомиться в документации (https://developers.google.com/drive/api/v3/reference/files) в разделе «Valid fields for files.list». У меня указаны поля для файлов: id (идентификатор файла в Drive), name (имя) и mimeType (тип файла). Чуть дальше мы рассмотрим пример запроса с большим количеством полей

nextPageToken в fields — это токен следующей страницы, если все результаты не помещаются в один ответ

```
results = service.files().list(pageSize=10,
                                fields="nextPageToken, files(id, name,
mimeType)").execute()
```

Получили вот такие результаты:

```
pp.pprint(results)
```


```
{   'files': [   {   'id': '0B7TyWvrAtxvgc3RhcnRlcl9maWxl',
                     'mimeType': 'application/pdf',
                     'name': 'Getting started'},
                 {   'id': '1uuecd6ndiZlj3d9dSVeZeKyEmEkC7qyr',
                     'mimeType': 'application/vnd.google-apps.folder',
                     'name': 'Folder'},
                 {   'id': '1HKC4U1BMJTsonlYJhUKzM-ygrIVGzdBr',
                     'mimeType': 'text/csv',
                     'name': 'Data.csv'},
                 {   'id': '10MM2f3V98wTu7GsoZSxzr9hkTGYvq_Jfb2HACvB9KjE',
                     'mimeType': 'application/vnd.google-apps.spreadsheet',
                     'name': 'Sheet'},
                 {   'id': '1fWhi4Jigrwl5oY_iJw9KJ-Eqr7D71UkY',
                     'mimeType': 'application/octet-stream',
                     'name': 'Dashboard.pbix'},
                 {   'id': '1NL1V_32OtQfB7zqe6t-xvTbbLQYXv6WYn5oPCpbPlPM',
                     'mimeType': 'application/vnd.google-apps.presentation',
                     'name': 'Great data'},
                 {   'id': '1RCTL5RyvTO2N5YYZHvB6xVqOgmpMVi5Z',
                     'mimeType': 'text/plain',
                     'name': 'Query_2.sql'},
                 {   'id': '1zV9hyUJpCKiz2KNksyLH4JPMeX0H3e93',
                     'mimeType': 'text/csv',
                     'name': 'data (4).csv'},
                 {   'id': '1oVSIGdm37p73MRb6ZFu5b0Nu2WsNallG',
                     'mimeType': 'text/csv',
                     'name': 'data.csv'},
                 {   'id': '1dLFxGYn198eIpdrPoD47b0BhHT9umWeJ',
                     'mimeType': 'text/csv',
                     'name': 'data (2).csv'}],
    'nextPageToken': '~!!~AI9FV7TeUokGsREbeTF26g2i2s7iQLD3phGpP-R1uTPIKLMTborvzj2sMiYYIkLRq8gV6ddiGSB8F635Lib8dAAeWnkTCQ_ydfDK
Ha6hhVAsy_DEfn5nXrOo1jmN2YIy-mnA0oY3-aJV9v0oSVwyp9neGoI978U4MjhCQOdDJBf8jg7exgGNJw5QIwnDn2fRb1XFMhODdTpEGuUa3Qi41MPaQNKjlGD654
zoyh300KWdAftoi97BcbgVtF7cWvEJFrZQizuRGQIo6qDovWKei8Ypf5ga9jc2a_-zPZBoVZpHL-sx13PAiiC98QQjT1uXmDRmKmupwdbluxA8ZkEH882uBZ0Yyiby
U5zdqdWaeOix4eAJDWTmF6F8Tnjv0jeOXIfa3J75eUry'}
```


```
print(len(results.get('files')))
```

Получив из результатов nextPageToken мы можем передать его в следущий запрос в параметре pageToken, чтобы получить результаты следующей страницы. Если в результатах будет nextPageToken, это значит, что есть ещё одна или несколько страниц с результатами

```
nextPageToken = results.get('nextPageToken')
results_for_next_page = service.files().list(pageSize=10,
                                fields="nextPageToken, files(id, name, mimeType)",
                                pageToken=nextPageToken).execute()
print (results_for_next_page.get('nextPageToken'))
```

```
~!!~AI9FV7RPyvBQTsFlBLme5kNuwQf-NvPKI5eHMrzY8z1yPkaA__eRgPuPmm0DspQ9QZeHt7N8W8dy9tpHmJ50kKEytVStL7H6DfXhusq9nasayr2nHKxLOR1RL6
_Qgy5n5IMvk4zMRWWV7Y29mRPX4uBblepRgh2Zav5J9NBkhT-9hamneNkIqrzUy4IAwP31S1J3CLcGVdshFrVROQa3cUn2VdVHPhlRyklT5G08r8eihv4qiMINbFVy
J78oTDJiYNRVsCHl6Lf7hKle3AOhtQRCFne7UqJQ2gcOthmwdX72j6XfpFjnTWN9eIqKaSj-mMT4AiCVJ4XGIjxlerat-xkqYuV2cxOQWswlCj_5UNEeASQH0jbCCi
3_j_oC0F4MbBkcNqrqqwI3
```

Таким образом, мы можем сделать цикл, который будет выполняться до тех пор, пока в результатах ответа есть nextPageToken. Внутри цикла будем выполнять запрос для получения результатов страницы и сохранять результаты к первым полученным результатам

```
results = service.files().list(pageSize=10,
                                fields="nextPageToken, files(id, name,
mimeType)").execute()
nextPageToken = results.get('nextPageToken')
while nextPageToken:
        nextPage = service.files().list(pageSize=10,
                                        fields="nextPageToken, files(id, name,
mimeType, parents)",
                                        pageToken=nextPageToken).execute()
        nextPageToken = nextPage.get('nextPageToken')
        results['files'] = results['files'] + nextPage['files']
print(len(results.get('files')))
```

Дальше давайте рассмотрим какие ещё поля можно использовать для списка возвращаемых файлов. Как я уже писал выше, со всеми полями можно ознакомиться по ссылке. Давайте рассмотрим самые полезные из них:

parents — ID папки, в которой расположен файл/подпапка

createdTime — дата создания файла/папки

permissions — перечень прав доступа к файлу

quotaBytesUsed — сколько места от квоты хранилища занимает файл (в байтах)

```
results = service.files().list(
        pageSize=10, fields="nextPageToken, files(id, name, mimeType, parents,
createdTime, permissions, quotaBytesUsed)").execute()
```

Отобразим один файл из результатов с расширенным списком полей. Как видно permissions содержит информацию о двух юзерах, один из которых имеет role = owner, то есть владелец файла, а другой с role = writer, то есть имеет право записи.

```
pp.pprint(results.get('files')[0])
```

```
{   'createdTime': '2019-02-19T15:16:27.052Z',
    'id': '1RCTL5RyvTO2N5YYZHvB6xVqOgmpMVi5Z',
    'mimeType': 'text/plain',
    'name': 'Query_2.sql',
    'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ'],
    'permissions': [   {   'deleted': False,
                           'displayName': 'namby-pamby@tensile-verve-232214.iam.gserviceaccount.com',
                           'emailAddress': 'namby-pamby@tensile-verve-232214.iam.gserviceaccount.com',
                           'id': '13173508647601803619',
                           'kind': 'drive#permission',
                           'role': 'writer',
                           'type': 'user'},
                       {   'deleted': False,
                           'displayName': 'Alexey Makarov',
                           'emailAddress': 'ax.makarov@gmail.com',
                           'id': '15194547887078870598',
                           'kind': 'drive#permission',
                           'photoLink': 'https://lh3.googleusercontent.com/a-/AAuE7mC0AQU5BwnLOBCG2aXOFeYMZLn4DzvfXC3NFTLp7g=s
64',
                           'role': 'owner',
                           'type': 'user'}],
    'quotaBytesUsed': '0'}
```

Очень удобная штука, позволяющая сократить количество результатов в запросе, чтобы получать только то, что действительно нужно — это возможность задать параметры поиска для файлов. Например, мы можем задать в какой папке искать файлы, зная её id:

```
results = service.files().list(
    pageSize=5,
    fields="nextPageToken, files(id, name, mimeType, parents, createdTime)",
    q="'1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ' in parents").execute()
pp.pprint(results['files'])
```

```
[   {   'createdTime': '2019-02-19T15:16:27.052Z',
        'id': '1RCTL5RyvTO2N5YYZHvB6xVqOgmpMVi5Z',
        'mimeType': 'text/plain',
        'name': 'Query_2.sql',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']},
    {   'createdTime': '2019-02-19T15:16:08.602Z',
        'id': '1zV9hyUJpCKiz2KNksyLH4JPMeX0H3e93',
        'mimeType': 'text/csv',
        'name': 'data (4).csv',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']},
    {   'createdTime': '2019-02-19T15:16:08.602Z',
        'id': '1oVSIGdm37p73MRb6ZFu5b0Nu2WsNallG',
        'mimeType': 'text/csv',
        'name': 'data.csv',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']},
    {   'createdTime': '2019-02-19T15:16:08.602Z',
        'id': '1dLFxGYn198eIpdrPoD47b0BhHT9umWeJ',
        'mimeType': 'text/csv',
        'name': 'data (2).csv',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']},
    {   'createdTime': '2019-02-19T15:16:08.602Z',
        'id': '1bYubUxVCNXm0l2R8bckITKCeymxaBBX8',
        'mimeType': 'text/csv',
        'name': 'data (5).csv',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']}]
```

С синтаксисом поисковых запросов можно ознакомиться в документации. Ещё один удобный способ поиска нужных файлов — по имени. Вот пример запроса, где мы ищем все файлы, содержащие в названии «data»:

```
results = service.files().list(
    pageSize=10,
    fields="nextPageToken, files(id, name, mimeType, parents, createdTime)",
```

```
        q="name contains 'data'").execute()
    pp.pprint(results['files'])
```

```
[   {   'createdTime': '2019-02-19T15:22:16.418Z',
        'id': '1NL1V_32OtQfB7zqe6t-xvTbbLQYXv6WYn5oPCpbPlPM',
        'mimeType': 'application/vnd.google-apps.presentation',
        'name': 'Great data',
        'parents': ['1uuecd6ndiZlj3d9dSVeZeKyEmEkC7qyr']},
    {   'createdTime': '2019-02-19T15:16:08.602Z',
        'id': '1zV9hyUJpCKiz2KNksyLH4JPMeX0H3e93',
        'mimeType': 'text/csv',
        'name': 'data (4).csv',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']},
    {   'createdTime': '2019-02-19T15:16:08.602Z',
        'id': '1oVSIGdm37p73MRb6ZFu5b0Nu2WsNallG',
        'mimeType': 'text/csv',
        'name': 'data.csv',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']},
    {   'createdTime': '2019-02-19T15:16:08.602Z',
        'id': '1dLFxGYn198eIpdrPoD47b0BhHT9umWeJ',
        'mimeType': 'text/csv',
        'name': 'data (2).csv',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']},
    {   'createdTime': '2019-02-19T15:16:08.602Z',
        'id': '1bYubUxVCNXm0l2R8bckITKCeymxaBBX8',
        'mimeType': 'text/csv',
        'name': 'data (5).csv',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']},
    {   'createdTime': '2019-02-19T15:16:08.602Z',
        'id': '1udviwdMiKrzTAXb4SulJQ10-mczy5m3V',
        'mimeType': 'text/csv',
        'name': 'data (3).csv',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']},
    {   'createdTime': '2019-02-19T15:16:07.316Z',
        'id': '1HLIzBEdsb3uozK6OTuX1Mexhy0kSKnVv',
        'mimeType': 'text/csv',
        'name': 'data (1).csv',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']}]
```

Условия поиска можно комбинировать. Возьмем условие поиска в папке и совместим с условием поиска по названию:

```
    results = service.files().list(
        pageSize=10,
        fields="nextPageToken, files(id, name, mimeType, parents, createdTime)",
        q="'1uuecd6ndiZlj3d9dSVeZeKyEmEkC7qyr' in parents and name contains
```

```
    'data'").execute()
    pp.pprint(results['files'])
```

```
[   {   'createdTime': '2019-02-19T15:22:16.418Z',
        'id': '1NL1V_32OtQfB7zqe6t-xvTbbLQYXv6WYn5oPCpbPlPM',
        'mimeType': 'application/vnd.google-apps.presentation',
        'name': 'Great data',
        'parents': ['1uuecd6ndiZlj3d9dSVeZeKyEmEkC7qyr']}]
```

Скачивание файлов из Google Drive

Теперь рассмотрим как скачивать файлы из Google Drive. Для этого нам понадобится создать запрос request для получения файла. После этого задаем интерфейс fh для записи в файл с помощью библиотеки io, указав в filename название файла (таким образом, можно сохранять файлы из Google Drive сразу с другим названием). Затем создаем экземпляр класса MediaIoBaseDownload, передав наш интерфейс для записи файла fh и запрос для скачивания файла request. Следующим шагом скачиваем файл по небольшим кусочкам (чанкам) с помощью метода next_chunk.

Если из предыдущего описания вам мало что понятно, не запаривайтесь, просто укажите свой file_id и filename, и всё у вас будет в порядке.

```
file_id = '1HKC4U1BMJTsonlYJhUKzM-ygrIVGzdBr'
request = service.files().get_media(fileId=file_id)
filename = '/home/makarov/File.csv'
fh = io.FileIO(filename, 'wb')
downloader = MediaIoBaseDownload(fh, request)
done = False
while done is False:
    status, done = downloader.next_chunk()
    print ("Download %d%%." % int(status.progress() * 100))
```

Файлы Google Sheets или Google Docs можно конвертировать в другие форматы, указав параметр mimeType в функции export_media (обратите внимание, что в предыдущем примере

скачивания файла мы использоали другую функцию get_media). Например, файл Google Sheets можно конвертировать и скачать в виде файла Excel.

```
file_id = '10MM2f3V98wTu7GsoZSxzr9hkTGYvq_Jfb2HACvB9KjE'
request = service.files().export_media(fileId=file_id,
                                                    mimeType='application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet')
filename = '/home/makarov/Sheet.xlsx'
fh = io.FileIO(filename, 'wb')
downloader = MediaIoBaseDownload(fh, request)
done = False
while done is False:
    status, done = downloader.next_chunk()
    print ("Download %d%%." % int(status.progress() * 100))
```

Затем скачанный файл можно загнать в датафрейм. Это достаточно простой способ получить данные из Google Sheet в pandas-dataframe, но есть и другие способы, например, воспользоваться библиотекой gspread.

```
import pandas as pd
df = pd.read_excel('/home/makarov/Sheet.xlsx')
df.head(5)
```

	id	count
0	.p43DqAlRavBbg1oDFSFzu	98
1	5NAEhw4KlcYpCdNzA/6xcu	24
2	HGnqYBm9w1egayK9eDHgBe	12
3	h52zj8PC1XJ1naMxhUyqE.	11
4	JbPx9urrduW6335HK4ljiu	10

Загрузка файлов и удаление в Google Drive

Рассмотрим простой пример загрузки файла в папку. Во-первых, нужно указать folder_id — id папки (его можно получить в адресной строке браузера, зайдя в папку, либо получив все файлы

и папки методом list). Также нужно указать название name, с которым файл загрузится на Google Drive. Это название может быть отличным от исходного названия файла. Параметры folder_id и name передаем в словарь file_metadata, в котором задаются метаданные загружаемого файла. В переменной file_path указываем путь к файлу. Создаем объект media, в котором будет указание по какому пути находится загружаемый файл, а также указание, что мы будем использовать возобновляемую загрузку, что позволит нам загружать большие файлы. Google рекомендует использовать этот тип загрузки для файлов больше 5 мегабайт. Затем выполняем функцию create, которая позволит загрузить файл на Google Drive.

```
folder_id = '1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ'
name = 'Script_2.py'
file_path = '/home/makarov/Script.py'
file_metadata = {
                'name': name,
                'parents': [folder_id]
            }
media = MediaFileUpload(file_path, resumable=True)
r = service.files().create(body=file_metadata, media_body=media,
fields='id').execute()
pp.pprint(r)
```

```
{'id': '18Wwvuye8dOjCZfJzGf45yQvB87Lazbzu'}
```

Как видно выше, при вызове функции create возвращается id созданного файла. Можно удалить файл, вызвав функцию delete. Но мы этого делать не будет так как файл понадобится в следующем примере

```
service.files().delete(fileId='18Wwvuye8dOjCZfJzGf45yQvB87Lazbzu').execute()
```

Сервисный аккаунт может удалить ли те файлы, которые были с помощью него созданы. Таким образом, даже если у сервисного аккаунта есть доступ на редактирование папки, то он не может удалить файлы, созданные другими пользователями. Понять что файл был создан помощью сервисного аккаунта можно задав поисковое условие с указанием email нашего

сервисного аккаунта. Узнать email сервисного аккаунта можно вызвав атрибут signer_email у объекта credentials

```
print (credentials.signer_email)
```

```
mamby-pamby@local-grove-232309.iam.gserviceaccount.com
```

```
results = service.files().list(
    pageSize=10,
    fields="nextPageToken, files(id, name, mimeType, parents, createdTime)",
    q="'namby-pamby@tensile-verve-232214.iam.gserviceaccount.com' in
owners").execute()
pp.pprint(results['files'][0:3])
```

```
[   {   'createdTime': '2019-02-19T15:55:04.089Z',
        'id': '18Wwvuye8dOjCZfJzGf45yQvB87Lazbzu',
        'mimeType': 'text/x-python',
        'name': 'Script_2.py',
        'parents': ['1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ']},
    {   'createdTime': '2019-02-19T14:59:32.028Z',
        'id': '0BxnMldK8lKquc3RhcnRlcl9maWxl',
        'mimeType': 'application/pdf',
        'name': 'Getting started',
        'parents': ['0ABnMldK8lKquUk9PVA']}]
```

Дальше — больше. С помощью API Google Drive мы можем загрузить файл с определенным mimeType, чтобы Drive понял к какому типу относится файл и предложил соответствующее приложение для его открытия.

```
folder_id = '1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ'
name = 'Sample data.csv'
file_path = '/home/makarov/sample_data_1.csv'
file_metadata = {
                'name': name,
                'mimeType': 'text/csv',
                'parents': [folder_id]
            }
```

```
    media = MediaFileUpload(file_path, mimetype='text/csv', resumable=True)
    r = service.files().create(body=file_metadata, media_body=media,
    fields='id').execute()
    pp.pprint(r)
```

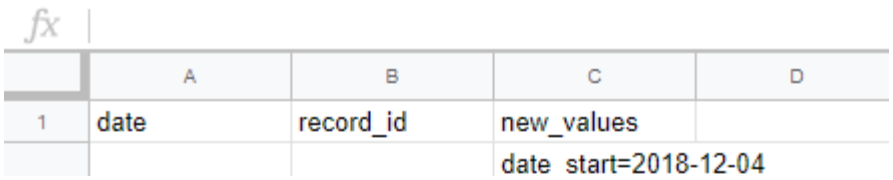
```
{'id': '1Nv-kHov1clPBbRcl26IxLSpvTeSqaSk9'}
```

Но ещё более классная возможность — это загрузить файл одного типа с конвертацией в другой тип. Таким образом, мы можем залить csv файл из примера выше, указав для него тип Google Sheets. Это позволит сразу же конвертировать файл для открытия в Гугл Таблицах. Для этого надо в словаре file_metadata указать mimeType «application/vnd.google-apps.spreadsheet».

```
    folder_id = '1mCCK9QGQxLDED8_pgq2dyvkmGRXhWEtJ'
    name = 'Sheet from csv'
    file_path = '/home/makarov/notebooks/sample_data_1.csv'
    file_metadata = {
                    'name': name,
                    'mimeType': 'application/vnd.google-apps.spreadsheet',
                    'parents': [folder_id]
                }
    media = MediaFileUpload(file_path, mimetype='text/csv', resumable=True)
    r = service.files().create(body=file_metadata, media_body=media,
    fields='id').execute()
    pp.pprint(r)
```

```
{'id': '1xq6Ler0ypPimEFMxer_wtYlklNB7MCBoM6aSZft5dAc'}
```

Таким образом, загруженный нами CSV-файл будет доступен как Гугл Таблица:



	A	B	C	D
1	date	record_id	new_values	
			date start=2018-12-04	

2	2018-12-11 8:47:	2187055	component=abc customer=4735
3	2018-12-19 12:4	2199969	date_start=2018-12-18 component=abc customer=19799
4	2018-12-19 12:4	2199971	date_start=2018-12-18 component=abc customer=19978
5	2018-12-19 12:4	2199973	date_start=2018-12-18 component=abc customer=15017
6	2018-12-19 12:4	2199975	date_start=2018-12-18 component=abc customer=20566
7	2018-12-19 12:4	2199977	date_start=2018-12-18 component=abc customer=9810
8	2018-12-19 12:4	2199979	date_start=2018-12-18 component=abc customer=27516
9	2018-12-19 12:4	2199981	date_start=2018-12-18 component=abc customer=29566
10	2018-12-19 12:4	2199983	date_start=2018-12-18 component=abc customer=27024
11	2018-12-19 12:4	2199985	date_start=2018-12-18 component=abc customer=24219

Ещё одна часто необходимая функция — это создание папок. Тут всё просто, создание папки также делается с помощью метода create, надо только в file_metadata указать mimeType «application/vnd.google-apps.folder»

```
folder_id = '1uuecd6ndiZlj3d9dSVeZeKyEmEkC7qyr'
name = 'New Folder'
file_metadata = {
    'name': name,
    'mimeType': 'application/vnd.google-apps.folder',
```

```
        'parents': [folder_id]
    }
    r = service.files().create(body=file_metadata,
                                        fields='id').execute()
    pp.pprint(r)
```

```
{'id': '1FvcJo5XV-IRLcGq97hN8v5Z6zWBTH-XI'}
```

Заключение

Все содержимое этой статьи также представлено в виде ноутбука для Jupyter Notebook.

В этой статье мы рассмотрели лишь немногие возможности API Google Drive, но одни из самых необходимых:

- Просмотр списка файлов
- Скачивание документов из Google Drive (в том числе, скачивание с конвертацией, например, документов Google Sheets в формате Excel)
- Загрузка документов в Google Drive (также как и в случае со скачиванием, с возможностью конвертации в нативные форматы Google Drive)
- Удаление файлов
- Создание папок

Вступайте в группу на Facebook и подписывайтесь на мой канал в Telegram, там публикуются интересные статьи про анализ данных и не только.

Успехов!

2019 api python автоматизация

Большая подборка полезных ссылок про Pandas ← Ctrl → Головоломка про рандомный сэмпл

6 комментариев

Yaroslav	Добрый день. Очень крутая статья, спасибо огромное. Можете подсказать как мне быть если я закидываю в spreadsheet csv файл и при этом мне необходимо указывать определённое имя листа в файле. Где можно указать этот параметр и как?
Алексей Макаров •	Здравствуйте! Насчет того как задать имя листа через API не подскажу. Но можно экспортировать фрейм в Excel с помощью to_excel (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_excel.html) и указать название листа в параметре sheet_name
Alexey S	Добрый день, возможно у вас есть пример того, как перенести файл из одной папки или корня google drive в другую?
Aratta Dev	Отличная статья! Спасибо
Антон Кизернис	Большое спасибо.
Константин Задворнов	Алексей, здравствуйте) Читал Вашу статью, она помогла мне создать сервисный аккаунт. Но у меня возникла новая проблема: создавая папку на Google

Drive, я не могу увидеть её на своём Google диске. Скажите, пожалуйста, что может быть не так. Заранее спасибо

Саня

Выскакивает ошибка Daily Limit for Unauthenticated Use Exceeded. Continued use requires signup

Ваш комментарий

Имя
и фамилия

Эл. почта

адрес не будет опубликован

Текст

☐ Получать комментарии других по почте

Отправить Ctrl + Enter

Популярное

Кого читать по теме аналитики данных

Становясь гуру API Яндекс.Метрики

С чего аналитику начать изучение Python

Как читать свои сообщения ВКонтакте через API

Список стоп-слов Яндекс.Директа

Как получить список станций Московского метрополитена по API

Как в Pandas разбить одну колонку на несколько

Как конвертировать JSON в CSV

Как использовать Google BigQuery с помощью Python

Движок — Эгея