



> Конспект > 7 урок > PYTHON

> Оглавление 7 урока

1. Замена элементов в зависимости от их значений
2. Проверка на непропущенные значения
3. Работа с ошибками
4. Булиновские результаты логических серий
5. Случайные числа
6. API
7. vk
8. Доступ к Google API
9. Работа с гугл докуми
10. json
11. Интернет-запросы
12. Конструирование ссылок
13. Яндекс.Метрика
14. Телеграм

> Замена элементов в зависимости от их значений

Функция `np.where()` позволяет задать новые значения, основываясь на старых. Она принимает 3 аргумента:

- `condition` — условие, то есть серия со списком `True` и `False`
- `x` — на что заменить `True`
- `y` — на что заменить `False`

```
a = pd.Series([0, 1, 2, 3, 1, 2, 3, 4, 5, 6])
np.where(a > 2, 'Higher than 2', 'Lesser than 2')
array(['Lesser than 2', 'Lesser than 2', 'Lesser than 2', 'Higher than 2',
      'Lesser than 2', 'Lesser than 2', 'Higher than 2', 'Higher than 2',
      'Higher than 2', 'Higher than 2'], dtype='<U13')
```

[Документация](#)

> Проверка на непропущенные значения

`notna` — это метод-антоним `isna`, возвращает True, если значение не NA. Альтернативный способ получить такой результат — инвертировать результат применения `isna` с помощью `~` (так во многих языках обозначают not, в логических сериях *pandas* также).

```
df.notna()

# Same as previous
~df.isna()
```

[Документация](#)

> Работа с ошибками

Помните ошибки, которые возникают при работе в питоне? Хорошая новость — их можно обрабатывать, но нужно делать это разумно.

Мы можем сделать так, чтобы программа продолжила работать дальше после ошибки. Это делается с помощью конструкции try-except:

```
expenditures = 0
income = 100
try:
    ratio = income / expenditures
except:
    print('Something went wrong, mb expenditures are 0?')
```

`try` и `except` должны быть вместе вплотную. Как это работает: после `try` ставится `:` и идёт блок кода, который пытается выполниться. Если ему это удаётся, то блок `except` (тоже с `:`) пропускается. Если же в блоке `try` произошла ошибка, то вместо прекращения ошибки идёт переход в блок `except` и выполняется код, содержащийся там. Далее следует выход из `except` и программа работает с кодом в скрипте дальше.

Теперь о том, зачем это нужно. Не нужно вставлять `try`, чтобы ваш код не падал с ошибками, и радоваться. Это специальный инструмент для работы с чувствительными местами, где может произойти ошибка, и где вам нужно действовать разными способами в случаях успешной работы блока или ошибки.

Во многих случаях try-except можно заменить полотно предварительных проверок, то есть проверками выполнимости перед выполнением рискованной операции (в примере выше — проверкой на равенство expenditures 0).

[Больше информации](#)

> Булиновские результаты логических серий

Мы уже много работали с логическими сериями, например:

```
predicate
Unnamed: 0      True
company41     False
company51     False
company50     False
company76     False
company47     False
company27     False
company48     False
dtype: bool
```

Иногда необходимо выяснить агрегированное значение серии — все ли там значения True, или есть ли хотя бы один True среди них. Для этого используются специальные методы.

all

Все ли значения в серии True? Аналогично использованию `and` между всеми значениями:

```
predicate.all()
```

False

[Документация](#)

any

Есть ли в серии хотя бы одно значение True? Аналогично использованию `or` между всеми значениями:

```
predicate.any()
```

True

[Документация](#)

> random

Служит для генерации случайных чисел. Есть аналог в numpy:

```
# returns number from [a, b]
random.randint(a, b)
```

[Документация](#)

> API (Application Programming Interface)

Вещь, которая значительно облегчает выполнение задач. По сути, это библиотека от создателей веб-сервиса (сайта, где можно что-то сделать), позволяющая быстро выполнить действия с этим сервисом. Как правило для работы с api необходимо получить токен.

[Больше информации](#)

Токен

Токен — это уникальная последовательность символов, позволяющая авторизоваться на сайте и работать с API. Пример

токена: d9b70b356593da15f73083d7a0e0554586ca5f743fc0f30dabb993f9917b4317725d4db40a3d5e3729607

> vk

API для ВКонтакте, позволяет программно выполнять действия, например, писать сообщения, выбирать друзей и так далее. [Документация по API VK](#)

Подготовка автоматизации:

- Создать группу
- В управлении группой зайти в Работу с API
- Создать ключ
- Зайти в раздел *Сообщения*
- Выбрать *Сообщения Сообщества: Включены*
- Зайти в группу
- Выбрать *разрешить сообщения* (находится в *Ещё*; если там стоит *Запретить сообщения*, то всё в порядке)
- В меню пригласить группу в чат, нажав *Добавить в беседу*

Если вам не помогли предыдущие пункты, то вот альтернативный способ:

- Создать группу
- В управлении группой зайти в *Работу с API*
- Создать ключ
- Зайти в раздел *Сообщения*
- Выбрать *Сообщения Сообщества: Включены*
- Зайти в группу
- Выбрать *разрешить сообщения* (находится в *Ещё*, если там стоит *Запретить сообщения*, то всё в порядке)
- Зайти в управление
- Выбрать *беседы*
- Создать беседу
- Нажать на неё
- Нажать *присоединиться*

Питоновская часть

Подготовка

```
import vk_api

# Token which you obtained via vk
app_token = 'd9b70b356593da15f73083d7a0e0578f86c545743fc0f30dabb993f9917b4317725d4db40a3d5e3729607' # id of the 1st chat
chat_id = 1 # id of my user-receiver
my_id = 148915653 # Initialize session
vk_session = vk_api.VkApi(token=app_token)

# Make it possible to use vk api methods as python methods
vk = vk_session.get_api()
```

Отправка сообщений

```
vk.messages.send(
    chat_id=chat_id,
    random_id=random.randint(1, 2 ** 31),
    message='Это я, Почтальон Печкин!')
```

Отправка документов

```
# Specify path to the file and its future name in the message
path_to_file = '/home/arleg/Downloads/Telegram Desktop/corr_plot.pdf'
file_name = 'plot.pdf'

upload_url = vk.docs.getMessagesUploadServer(peer_id=my_id)["upload_url"]
file = {'file': (file_name, open(path_to_file, 'rb'))}

# Send request to post this doc on vk.com
response = requests.post(upload_url, files=file)

json_data = json.loads(response.text)

saved_file = vk.docs.save(file=json_data['file'], title=file_name)
attachment = 'doc_{_}'.format(saved_file['doc']['owner_id'], saved_file['doc']['id'])

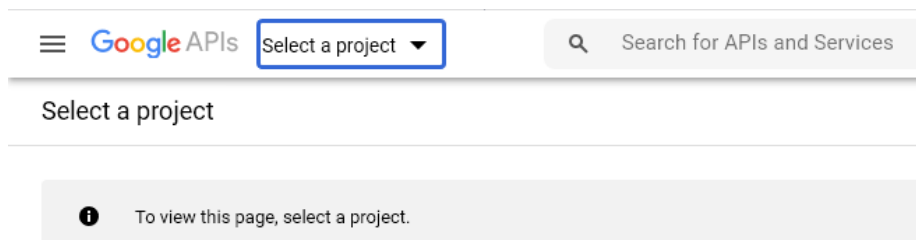
vk.messages.send(
    chat_id=chat_id, # id of chat where to send
    random_id=random.randint(1, 2 ** 31), # random number for message identification
    message='Привёз посылку для вашего мальчика!', # message text, optional here
    attachment=attachment) # attachment name
```

Заменяя `chat_id=chat_id` на `user_id=my_id`, можно отправлять сообщения себе. Только поставьте в `my_id` свой id)

[Документация](#)

> Получение доступа к API Google

1. Перейдите на [Google Developer Console](#)
2. Нажмите Select a project



3. Нажмите NEW PROJECTВведите имя и нажмите Create


Select a project

 **NEW PROJECT**

 Search projects and folders

Google APIs Search for APIs and Services


New Project

 You have 11 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name *
test ?

Project ID: utopian-pen-297211. It cannot be changed later. [EDIT](#)

Location *
 No organization [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

4. Нажмите Enable Apps and Services

Google APIs test Search for APIs and Services

API APIs & Services APIs & Services [+ ENABLE APIS AND SERVICES](#)


5. Выберите из предложенных Google Drive API и Google Sheets API (они находятся ниже в списке) и для каждого из них..... нажмите ENABLE (нужно для их подключения) Для выбора следующего API, выберите Google APIs слева, а затем перейти в Library (или просто нажмите в браузере на стрелку назад)

Welcome to the API Library


The API Library has documentation, links, and a smart search experience.

Search for APIs & Services


Maps [VIEW ALL \(15\)](#)




Maps SDK for Android
Google
Maps for your native Android app.



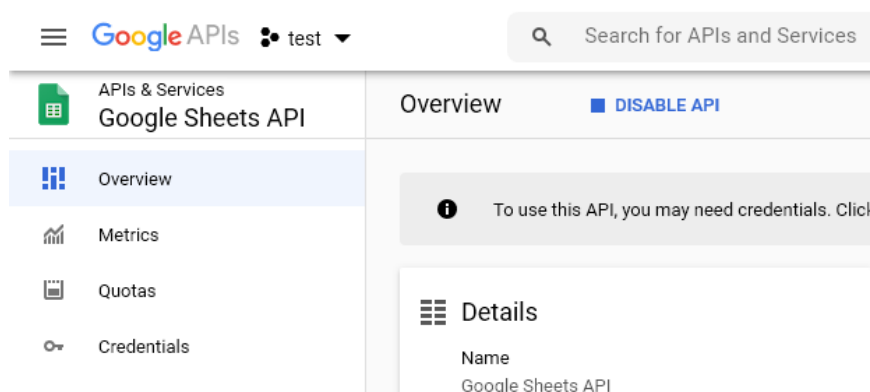
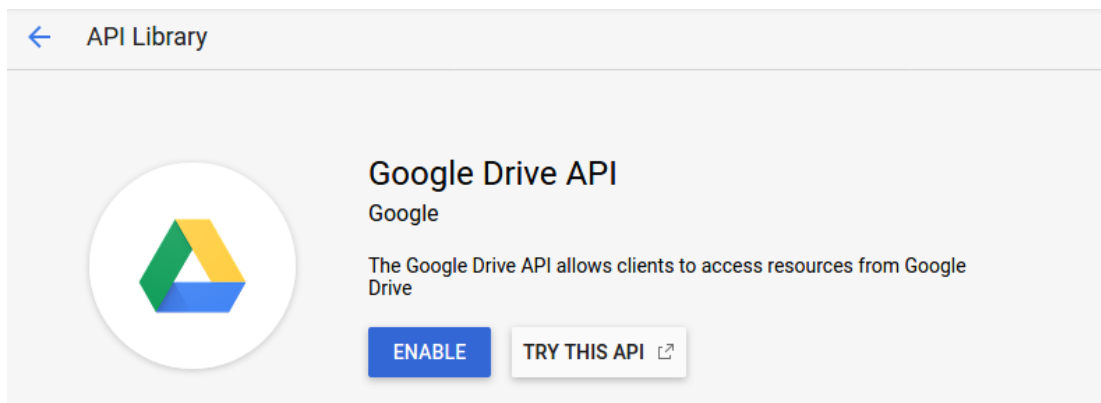
Maps SDK for iOS
Google
Maps for your native iOS app.



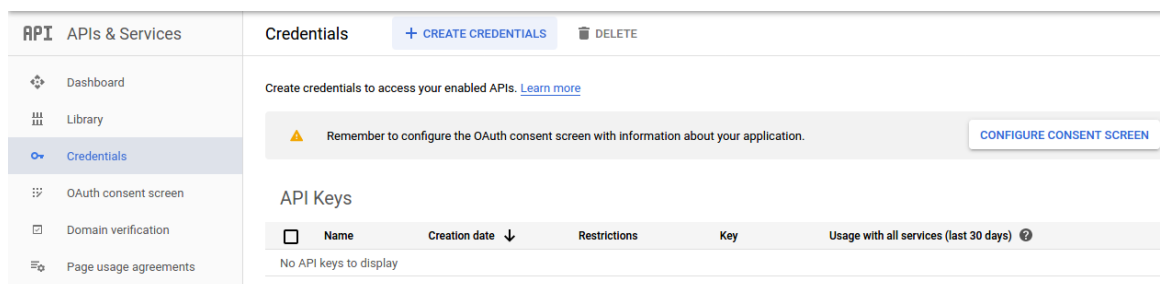
Maps JavaScript API
Google
Maps for your website



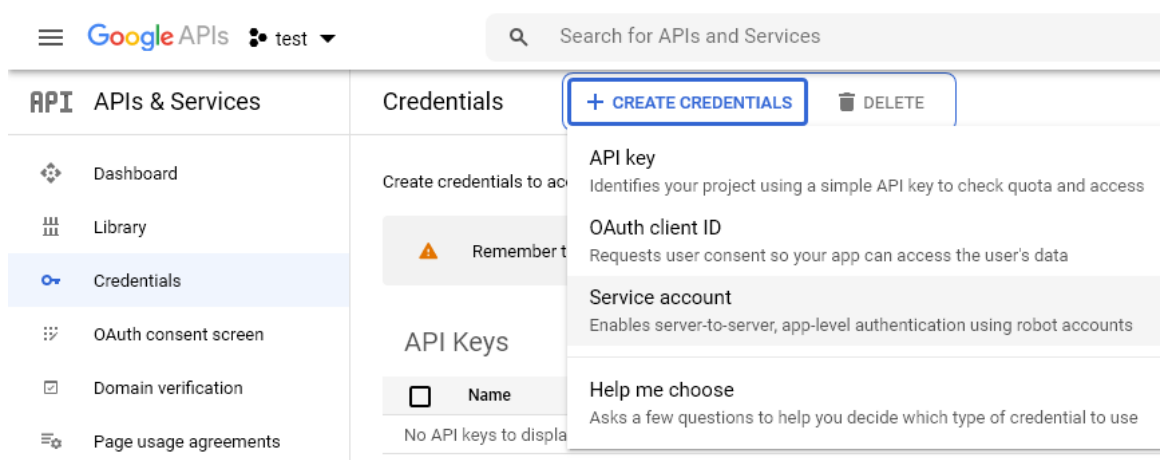
Places API
Google
Get detailed information about 100 million places



6. Снова нажмите Google APIs слева и перейдите во вкладку Credentials



7. Нажмите Create credentials и выберите Service account



8. Введите имя и нажмите CREATE

Create service account

1

Service account details

2

Grant this service account access to project (optional)

3

Grant users access to this service account (optional)

Service account details

Service account name

testname

Display name for this service account

Service account ID

testname @mimetic-obelisk-268109.iam.gserviceaccount.com

X

↺

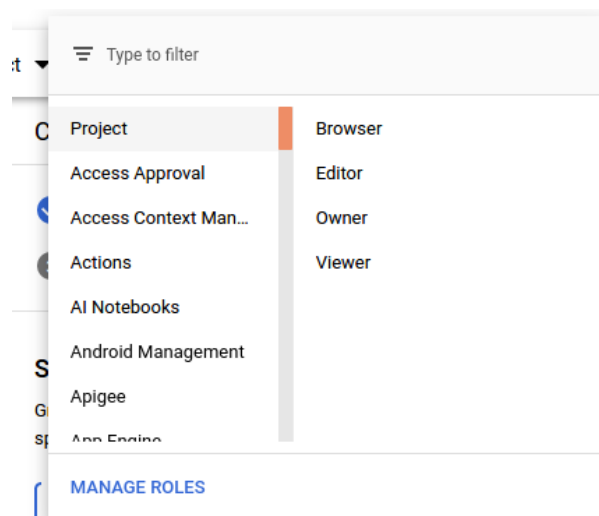
Service account description

Describe what this service account will do

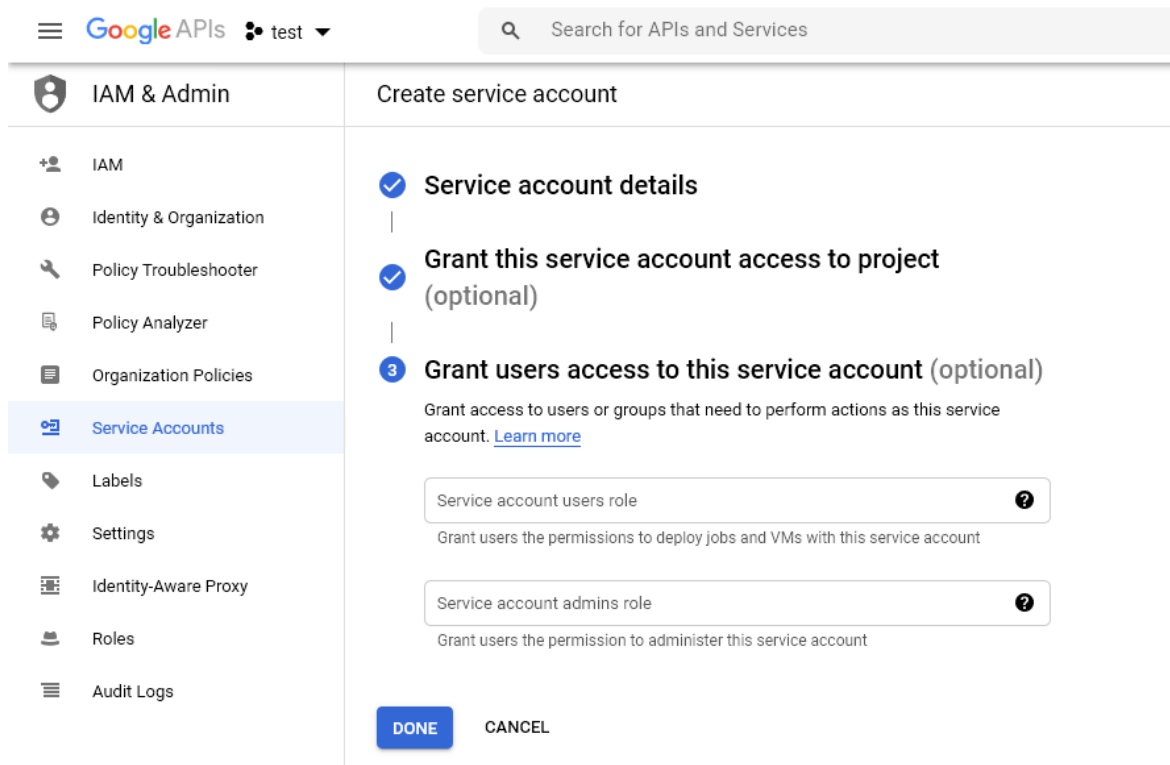
CREATE

CANCEL

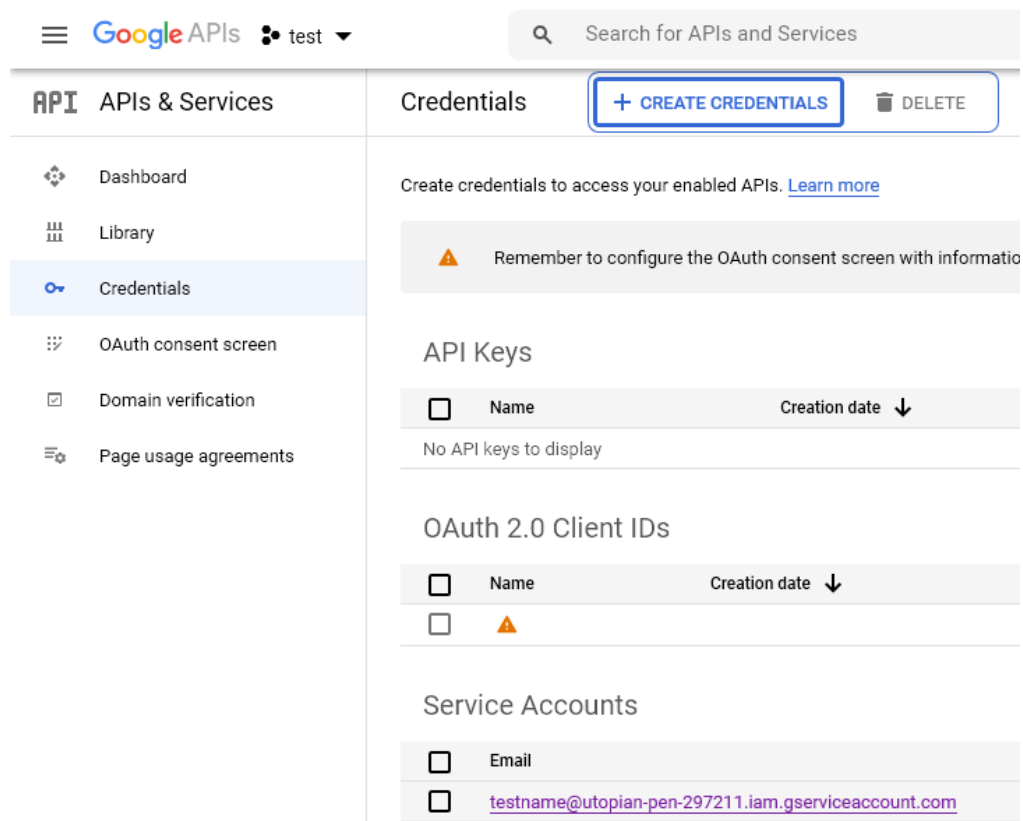
9. Выберите роль Owner и нажмите Continue



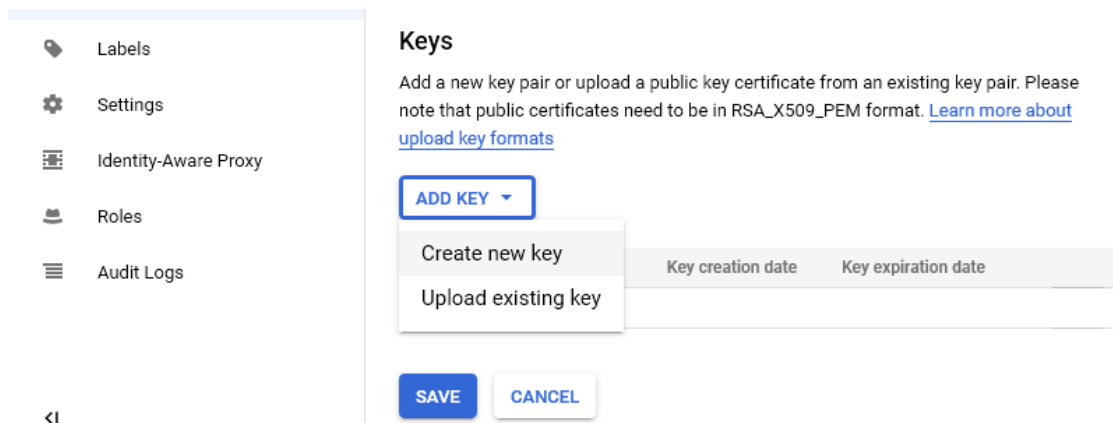
10. Нажмите DONE



11. Нажмите на почту credential (в Service Accounts)



12. Прокрутите ниже и нажмите ADD KEY > Create new key



13. Выберите json и нажмите CREATE

Create private key for "testname"

Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.

Key type

☒ JSON

Recommended

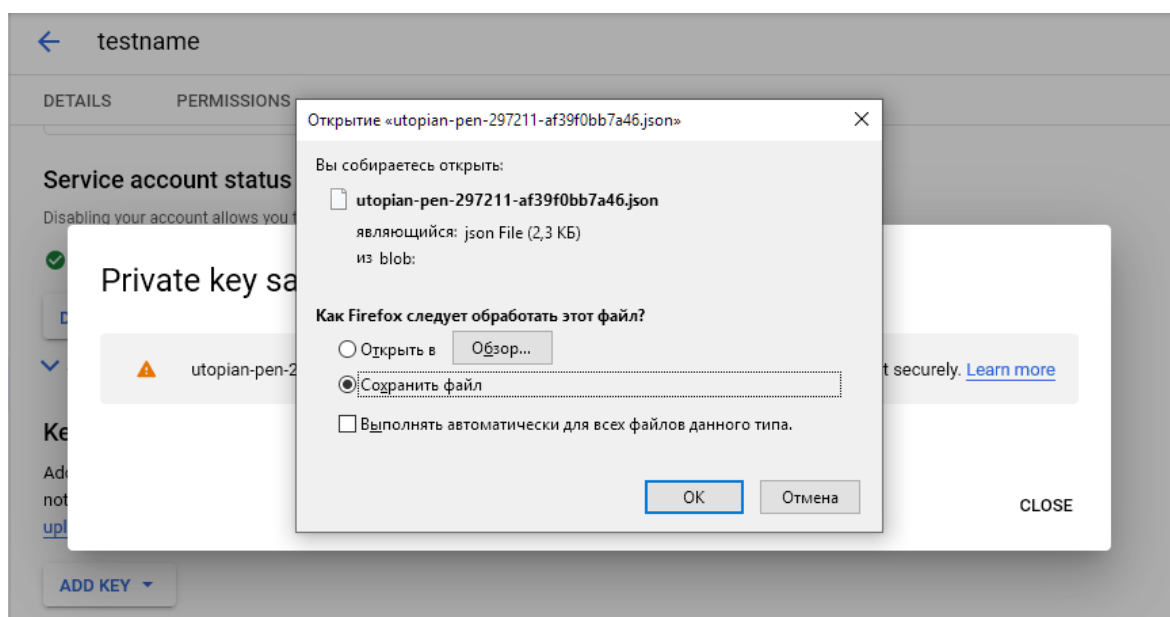
☐ P12

For backward compatibility with code using the P12 format

CANCEL

CREATE

14. Сохраните созданный json файл



15. Нажмите SAVE

Labels

Settings

Identity-Aware Proxy

Roles

Audit Logs

Keys

Add a new key pair or upload a public key certificate from an existing key pair. Please note that public certificates need to be in RSA_X509_PEM format. [Learn more about upload key formats](#)

ADD KEY ▾

Type	Status	Key	Key creation date
	Active	af39f0bb7a46f7a9249c08cea8b3114c7a827aab	Nov 30, 2020

SAVE

CANCEL

16. Откройте файл ключа в текстовом просмотрщике и скопируйте почту в разделе client_email (это действие можно выполнить через библиотеку json из питона)

```
"client_email": "googleapi@***-*****-*****.iam.gserviceaccount.com",
```

17. Для проверки, что всё работает, откройте какую-нибудь таблицу в google docs и выберите там настройки доступа, введите скопированную почту в поле и нажмите *готово*
18. Затем в питоне выполните следующее:

```
import gspread
from oauth2client.service_account import ServiceAccountCredentials

# Specify path to your file with credentials
path_to_credential = 'credentials.json' # Specify name of table in google sheets
table_name = 'name of your table'

scope = ['https://spreadsheets.google.com/feeds',
         'https://www.googleapis.com/auth/drive']

credentials = ServiceAccountCredentials.from_json_keyfile_name(path_to_credential, scope)

gs = gspread.authorize(credentials)
work_sheet = gs.open(table_name)

# Select 1st sheet
sheet1 = work_sheet.sheet1

# Get data in python lists format
data = sheet1.get_all_values()

# Get header from data
headers = data.pop(0)

# Create df
df = pd.DataFrame(data, columns=headers)
df.head()
```

После этого в `df` у вас должно быть содержимое таблицы.

[Инструкция](#) на английском и [ещё одна](#)

> Работа с гугл документами

Подготовка

```
import pandas as pd
import gspread
from df2gspread import df2gspread as d2g
from oauth2client.service_account import ServiceAccountCredentials
```

```
scope = ['https://spreadsheets.google.com/feeds',
        'https://www.googleapis.com/auth/drive']

my_mail = 'your@mail'
path_to_credentials = 'credentials.json' # Authorization
credentials = ServiceAccountCredentials.from_json_keyfile_name(path_to_credentials, scope)
gs = gspread.authorize(credentials)
```

Загрузка таблицы из гугл доков

```
# Name of the table in google sheets, # can be url for open_by_url# or id (key) part for open_by_key
table_name = 'table name' # Your table# Get this table
work_sheet = gs.open(table_name)

# Select 1st sheet
sheet1 = work_sheet.sheet1

# Get data in python lists format
data = sheet1.get_all_values()

# Get header from data
headers = data.pop(0)

# Create df
df = pd.DataFrame(data, columns=headers)
df.head()
```

При считывании таблицы должен быть доступ к ней, например, при считывании по ссылке (url) с помощью функции `open_by_url` нужно настроить права доступа на данную таблицу.

Создание своей таблицы

```
# Create empty table
table_name = 'A new spreadsheet'

sheet = gs.create(table_name)

# Make it visible to other guys
sheet.share(my_mail, perm_type='user', role='writer')
```

Документация

Экспорт датафрейма в гугл доки

Для переноса датафрейма в таблицу гуглдока, нужно, чтобы вы сделали эту таблицу из питона. Поэтому прогоните перед этой частью предыдущий раздел с желаемым названием таблицы:

```
# Create your df
df = ...

# Looks like spreadsheet should be already present at the dist (so, run code in create table section)
sheet = 'Master'
d2g.upload(df, table_name, sheet, credentials=credentials, row_names=True)
```

При этом необходимо, чтобы в `df` не было повторяющихся индексов (`reset_index`)

Документация

Чтобы получить ссылку на гугл документ, воспользуйтесь следующим кодом

```
spreadsheet_url = "https://docs.google.com/spreadsheets/d/%s" % sheet.id
```

где `sheet` — результат `gs.create(table_name)`

> json

Библиотека для работы с json'ом — одним из самых распространённых форматов данных в вебе. Он используется для пересылки данных между веб-сервисами, и очень похож на словари в python. Одноимённая библиотека `json` используется для его преобразования в действительно питоновский словарь.

```
import json

# Convert json to python dict
json_data = json.loads(some_json)
```

[Формат](#)

[Документация](#)

> Интернет-запросы

Библиотека `requests` позволяет взаимодействовать с сайтами. Метод `get()` принимает ссылку на сайт в виде строки, и возвращает объект, содержащий ответ с сайта в виде строки:

```
import requests

query = requests.get(url)
```

[Документация](#)

> Конструирование ссылок

Если вам не нравится вручную делать ссылки-запросы для Яндекс.Метрики, есть решение:

```
from urllib.parse import urlencode

# Base path to service
base_url = 'https://api-metrika.yandex.net/stat/v1/data?#' # Parameters of query
params = {'metrics': 'ym:s:visits',
          'dimensions': 'ym:s:date',
          'id': '44147844'}

visits_url = base_url + urlencode(params)
visits_url
'https://api-metrika.yandex.net/stat/v1/data?metrics=ym%3As%3Avisits&dimensions=ym%3As%3Adate&id=44147844'
```

Получившаяся ссылка в нескольких местах не совпадает с полученной ранее, но она такая же — просто в ней произведено дополнительное экранирование символов. Далее её можно точно так же использовать в `requests.get`

Чтобы закодировать несколько значений параметра с одним именем, просто передайте в словарь вместо значения к нужному параметру список из них, и укажите параметр `doseq=True`

```
# Base path to service
base_url = 'https://api-metrika.yandex.net/stat/v1/data?#' # Parameters of query
params = {'metrics': 'ym:s:visits',
          'dimensions': ['ym:s:date', 'ym:s:isRobot'],
          'id': '44147844'}

visits_url = base_url + urlencode(params, doseq=True)
visits_url
'https://api-metrika.yandex.net/stat/v1/data?metrics=ym%3As%3Avisits&dimensions=ym%3As%3Adate&dimensions=ym%3As%3AisRobot&id=44147844'
```

Параметры requests.get

Всё можно сделать еще короче — достаточно записать словарь с параметрами в `params` внутри `requests.get`

```
requests.get(base_url,
              params={
                  'metrics': 'ym:s:visits',
                  'dimensions': ['ym:s:date', 'ym:s:isRobot'],
                  'id': 44147844
              })
```

Этот код делает то же самое, что предварительное конструирование ссылки и `requests.get` на ней.

[Больше информации](#)

> Yandex metrika

Список доступных параметров для запросов в Яндекс.Метрике.

```
import pandas as pd
import requests
import json

# Base url to service
url = 'https://api-metrika.yandex.net/stat/v1/data?'# &-separated parameters of query in a form of name=value, taken from the
metrika site
visits = 'metrics=ym:s:visits&dimensions=ym:s:date&id=44147844'
url = url + visits

# Get json of response
query = requests.get(url)
json_data = json.loads(query.text)

# Conversion of obtained json to dataframe
visits_data = pd.DataFrame([(
    i['dimensions'][0]['name'],
    i['metrics'][0]) for i in json_data['data']],
    columns=['date', 'visits'])
```

visits_data

	date	visits
0	2020-02-13	1085.0
1	2020-02-17	1067.0
2	2020-02-12	1040.0
3	2020-02-14	808.0
4	2020-02-15	544.0
5	2020-02-16	496.0
6	2020-02-18	71.0

> Телеграм(м)

Маленькое напоминание: Telegram забанен, поэтому для открытия ссылок, связанных с ним, необходим VPNUpdate: телегу разблокировали, но кто знает, надолго ли

Получение токена

Чтобы автоматизировать работу в телеграме необходимо создать бота и получить токен. Для этого:

1. В телеграме найдите @Botfather
2. Нажмите start (или напишите /start) в диалоговом окне — появится сообщение с информацией о создании ботов
3. Отправьте ему сообщение /newbot
4. Введите имя для бота
5. Введите username бота, он должен заканчиваться на bot
6. Появится сообщение, содержащее токен

Перечень команд доступных для общения с Отцом всех ботов (используйте VPN, если ссылка не открывается)

Диалог с ботом

Воспользуйтесь username бота (или ссылкой на него), полученным от Botfather, начните диалог и отправьте ему что-нибудь. Затем введите в браузере ссылку вида

```
https://api.telegram.org/bot<token>/getUpdates
```

Где вместо `<token>` будет ваш токен.

В открывшемся окне вы увидите содержание json файла, где будет содержаться id чата (result > 0 > chat > id). Сохраните его.

После этого через бота можно посылать сообщения вам с помощью модуля `request`. Чтобы отправлять сообщения кому-нибудь другому, попросите его начать диалог с ботом, и повторите операцию с просмотром страницы, чтобы выяснить id чата.

Прокси

Для обхода блокировки в коде используются прокси. Помните, что их работа, к сожалению, нестабильна – может быть долгий отклик и программа будет долго выполняться, или они могут отрубать доступ из-за большого числа запросов. В таких случаях нужно брать другое прокси. Для Telegram нужны https прокси. К счастью, пока телеграм разбанен, всё работает без прокси:

Отправка сообщений

```
import requests
import json
from urllib.parse import urlencode

token = 'your_token'
chat_id = 123 # your chat id

message = 'test' # text which you want to send

params = {'chat_id': chat_id, 'text': message}

base_url = f'https://api.telegram.org/bot{token}/'
url = base_url + 'sendMessage?' + urlencode(params)
# Only if you need it# proxy = {'https': 'https://77.48.23.199:57842'}# To send request via proxy# resp = requests.get(url, proxies=proxy)
resp = requests.get(url)
```

После этого бот должен прислать вам сообщение test

Отправка документов

```
# Path to necessary file
filepath = 'your_path'

url = base_url + 'sendDocument?' + urlencode(params)

files = {'document': open(filepath, 'rb')}

# If you need proxy# resp = requests.get(url, files=files, proxies=proxy)
resp = requests.get(url, files=files)
```

В результате ваш документ отправится в чат.

Другой вариант работы с телеграмом

Существуют библиотеки для создания ботов в телеге. Лучше пользоваться ими, но будьте осторожны, может слететь юпитер (старых версий — сейчас, скорее всего, всё будет нормально)

Список проксиАльтернативные инструкции