МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С. П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

Вывод информации в документ GoogleDoc из консольной утилиты через API Google.

Выполнил:
Садриев Б. А.
гр.6213

Проверил:
Востокин С. В.

Самара 2022

Создание приложения на Python, выполняющего запросы по API Google Sheets:

В первую очередь, вам необходимо соблюсти следующие предварительные условия:

- Python 2.6 или выше;
- Инструмент управления пакетами pip;
- Google Cloud Platform с включенным API. Чтобы создать проект и включить API, необходимо перейти к разделу «Создание проекта» и подключить API(Google Sheets API);
- Учетные данные для авторизации(credentials);
- Учетная запись Google.

Первым шагом устанавливаем клиентскую библиотеку Google для Python, с помощью команды в терминале:

pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib

Далее переходим на сайт Google Cloud Platform, входим с помощью учетной записи Google и создаем проект:

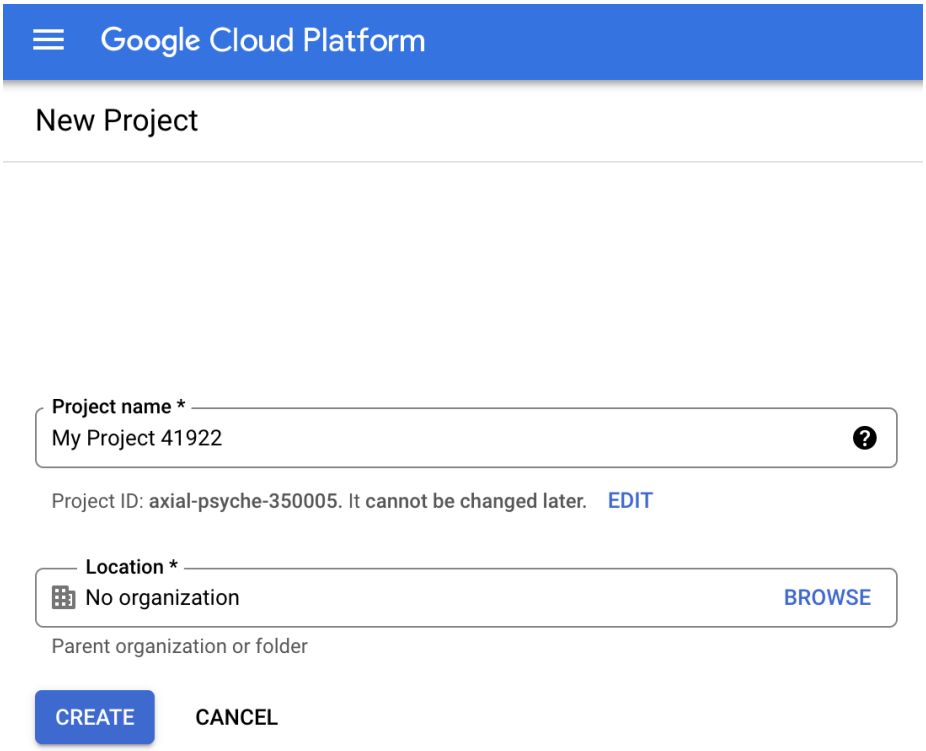

Рисунок 1 – Создание проекта.

Далее переходим в раздел «APIs & Services», где создаем учетные данные(credentials) типа OAuth client ID(Запрашивает согласие пользователя,

чтобы ваше приложение могло получить доступ к данным пользователя). При создании, указываем тип приложения, как настольное приложение. После создания, загружаем файл с нашими данными, переименовав его в «credentials.json».

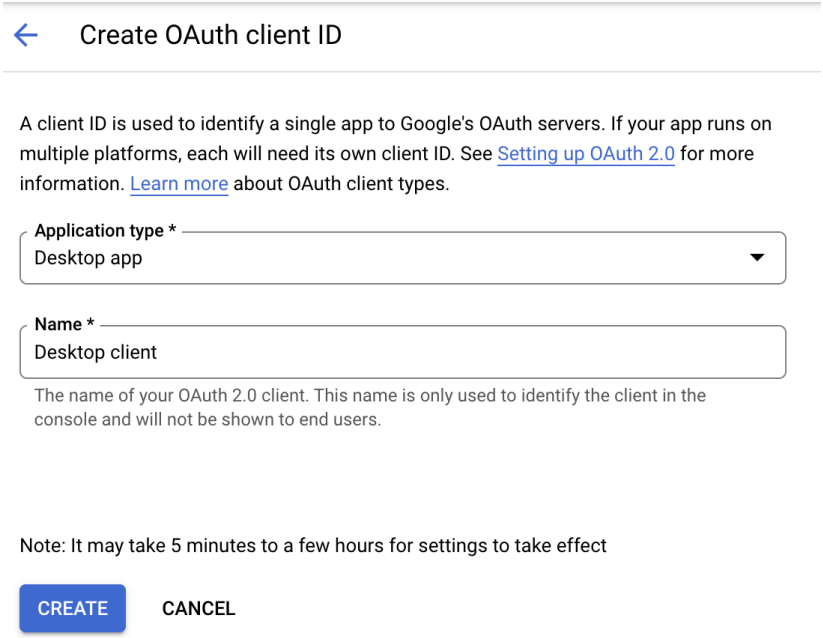

Рисунок 2 – Создание учетных данных.

Далее создаем в вашей рабочей директории файл main.py и вставляем следующий код:

```
from __future__ import print_function

import os.path

from google.auth.transport.requests import Request
from google.oauth2.credentials import Credentials
from google_auth_oauthlib.flow import InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError

# При изменении SCOPES, удалите файл token.json.
SCOPES = ['https://www.googleapis.com/auth/spreadsheets.readonly']

# Идентификатор и диапазон образца таблицы.
SAMPLE_SPREADSHEET_ID = '1BxiMVs0XRA5nFMdKvBdBZjgmUUqptlbs74OgvE2upms'
SAMPLE_RANGE_NAME = 'Class Data!A2:E'

def main():
    """ Показывает базовое использование API Sheets.
   Выводит значения из образца таблицы.
    """
    creds = None
    # Токен файла.json хранит токены доступа и обновления пользователя и создается автоматически, когда
#поток авторизации завершается в первый раз.
    if os.path.exists('token.json'):
        creds = Credentials.from_authorized_user_file('token.json', SCOPES)
    # Если нет доступных (действительных) учетных данных, позволяем пользователю войти в систему.
    if not creds or not creds.valid:
```

```
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file(
                'credentials.json', SCOPES)
            creds = flow.run_local_server(port=0)
        # Сохраняем the credentials
        with open('token.json', 'w') as token:
            token.write(creds.to_json())

    try:
        service = build('sheets', 'v4', credentials=creds)

        sheet = service.spreadsheets()
        result = sheet.values().get(spreadsheetId=SAMPLE_SPREADSHEET_ID,
                                    range=SAMPLE_RANGE_NAME).execute()
        values = result.get('values', [])

        if not values:
            print('No data found.')
            return

        print('Name, Major:')
        for row in values:
            # Выводим столбцы A and E, которые идут от 0 до 4.
            print('%s, %s' % (row[0], row[4]))
    except HttpError as err:
        print(err)

if __name__ == '__main__':
    main()
```

Замечание: важно отметить, что token может быть не только в формате .json, но и в формате .pickle.

Пример рабочей программы:

Файл Google.py:

```
import pickle
import os
from google_auth_oauthlib.flow import Flow, InstalledAppFlow
from googleapiclient.discovery import build
from googleapiclient.http import MediaFileUpload, MediaIoBaseDownload
from google.auth.transport.requests import Request

def Create_Service(client_secret_file, api_name, api_version, *scopes):
    # print(client_secret_file, api_name, api_version, scopes, sep='-')
    CLIENT_SECRET_FILE = client_secret_file
    API_SERVICE_NAME = api_name
    API_VERSION = api_version
    SCOPES = [scope for scope in scopes[0]]
    #print(SCOPES)

    cred = None

    pickle_file = f'token_{API_SERVICE_NAME}_{API_VERSION}.pickle'
    # print(pickle_file)

    if os.path.exists(pickle_file):
        with open(pickle_file, 'rb') as token:
            cred = pickle.load(token)
```

```
    if not cred or not cred.valid:
        if cred and cred.expired and cred.refresh_token:
            cred.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file(CLIENT_SECRET_FILE, SCOPES)
            cred = flow.run_local_server()

        with open(pickle_file, 'wb') as token:
            pickle.dump(cred, token)

    try:
        service = build(API_SERVICE_NAME, API_VERSION, credentials=cred)
        #print(API_SERVICE_NAME, 'сервис успешно создан')
        print('сервис успешно создан')
        return service
    except Exception as e:
        print('Unable to connect.')
        print(e)
        return None
```

Файл Helper.py:

```
def print Hello():

    print('''

    Дополнительная лабораторная работа

    Выполнил Садриев Богдан 6213

    Дополнительная лабораторная работа

    Выполнил студент Садриев Богдан из группы 6213

    Нажмите 1 для чтения данных

    Нажмите 2 для добавления записи данных о продукте

    Нажмите 3 для удаления записи

    Нажмите 4 для копирования записи

    Нажмите 0 чтобы очистить все данные в таблице



    Нажмите 121 для выхода из программы''')
```

Файл Main.py:

```
import sys

import time

from os import system

from Google import Create_Service

from Helper import printHello

import pandas as pd

import os

import pyautogui
```

```
CLIENT_SECRET_FILE = 'credentials.json'
API_SERVICE_NAME = 'sheets'
API_VERSION = 'v4'
SCOPES = ['https://www.googleapis.com/auth/spreadsheets']

service = Create_Service(CLIENT_SECRET_FILE, API_SERVICE_NAME, API_VERSION, SCOPES)
spreadsheet_id = '1SvEhB0jtDOZ9KqZJF7xu7r1tR54h9wtQ7k2yOD-3Rxc'
mySpreadsheets = service.spreadsheets().get(spreadsheetId=spreadsheet_id).execute()
def case1():
    response = service.spreadsheets().values().get(
        spreadsheetId=spreadsheet_id,
        majorDimension='ROWS',
        range='ListOne'
    ).execute()
    #print(response['values'])
    columns = response['values'][0]
    data = response['values'][1:]
    df = pd.DataFrame(data, columns=columns)
    df2 = df.set_index('Счетчик')
    print(df2)
def case2():
    name = input("Введите название продукта: ")
    categoty = input("Введите категорию товара: ")
    weight = input("Введите вес товара: ")
    estimation = input("Введите оценку качества: ")
    price = input("Введите цену товара: ")
    range1 = 'A'
    lastRow = 1
    response = service.spreadsheets().values().get(
        spreadsheetId=spreadsheet_id,
        range='ListOne!A1:A'
    ).execute()
    lastRow += len(response['values'])  # lastRow выше на 1 значение
    range3 = range1 + str(lastRow)
    worksheet_name = 'ListOne!'
    cell_range_insert = range3
    values = (
```

```
        (lastRow - 1, name, categoty, weight, estimation, price),
    )
    value_range_body = {
        'majorDimension': 'ROWS',  #COLUMNS
        'values': values
    }
    service.spreadsheets().values().update(
        spreadsheetId=spreadsheet_id,
        valueInputOption='USER_ENTERED',
        range=worksheet_name + cell_range_insert,
        body=value_range_body
    ).execute()
def case3():
    case1()
    row_number = int(input("Введите номер удаляемой записи: "))
    row_number_next = row_number - 1
    lastRow = 1
    response = service.spreadsheets().values().get(
        spreadsheetId=spreadsheet_id,
        range='ListOne!A1:A'
    ).execute()
    lastRow += len(response['values'])  # lastRow выше на 1 значение
    request_body_delete = {
        'requests': [
            {
                'deleteDimension': {
                    'range': {
                        'dimension': 'ROWS',
                        'startIndex': row_number_next + 1,
                        'endIndex': row_number + 1  ###
                    }
                }
            }
            ]
    }
    response = service.spreadsheets().batchUpdate(
        spreadsheetId=spreadsheet_id,
```

```
        body=request_body_delete
    ).execute()
    request_body3 = {
        'requests': [
            {
                'autoFill': {
                    'useAlternateSeries': False,
                    'sourceAndDestination': {
                        'source': {
                            'startRowIndex': 2,
                            'endRowIndex': 4,
                            'startColumnIndex': 0,
                            'endColumnIndex': 1
                        },
                        'dimension': 'ROWS',
                        'fillLength': lastRow - 6
                    }
                }
            }
        ]
    }
    response = service.spreadsheets().batchUpdate(
        spreadsheetId=spreadsheet_id,
        body=request_body3
    ).execute()
def case4():
    case1()
    row_number_start = int(input("Введите номер копируемой строки: "))
    lastRow = 1
    response = service.spreadsheets().values().get(
        spreadsheetId=spreadsheet_id,
        range='ListOne!A1:A'
    ).execute()
    lastRow += len(response['values'])  # lastRow выше на 1 значение
    request_body2 = {
        'requests': [
            {
```

```
            'copyPaste': {
                'source': {
                    'startRowIndex': row_number_start,
                    'endRowIndex': row_number_start + 1,  ##
                    'startColumnIndex': 1,
                    'endColumnIndex': 6
                },
                'destination': {
                    'startRowIndex': lastRow - 1,
                    'endRowIndex': lastRow,  ##
                    'startColumnIndex': 1,
                    'endColumnIndex': 6
                },
                'pasteType': 'PASTE_FORMULA'
            }
        }
    ]
}
response = service.spreadsheets().batchUpdate(
    spreadsheetId=spreadsheet_id,
    body=request_body2
).execute()
request_body3 = {
    'requests': [
        {
            'autoFill': {
                'useAlternateSeries': False,
                'sourceAndDestination': {
                    'source':{
                                'startRowIndex': 2,
                                    'endRowIndex': 4,
                                    'startColumnIndex': 0,
                                    'endColumnIndex': 1
                    },
                    'dimension': 'ROWS',
                    'fillLength': lastRow - 4
                }
```

```
            }
          }
        ]
    }
    response = service.spreadsheets().batchUpdate(
        spreadsheetId=spreadsheet_id,
        body=request_body3
    ).execute()
def default():
    print('Произведен выход...')
def case0():
    service.spreadsheets().values().clear(
        spreadsheetId=spreadsheet_id,
        range='ListOne'
    ).execute()
def switch(menu):
    dict={
        1: case1,
        2: case2,
        3: case3,
        4: case4,
        0: case0
    }
    return dict.get(menu, default)()
files = ['data/ascii10.txt']
frames = []
clear = lambda: system('clear')
for name in files:
    with open(name, 'r', encoding='utf8') as f:
        frames.append(f.readlines())
for i in range(1):
    for frame in frames:
        print(''.join(frame))
        time.sleep(0.5)
        clear = lambda: system('clear')
menu = 77
while menu != 121:
```

```
printHello()
menu = int(input("Введите номер действия: "))
switch(menu)
```

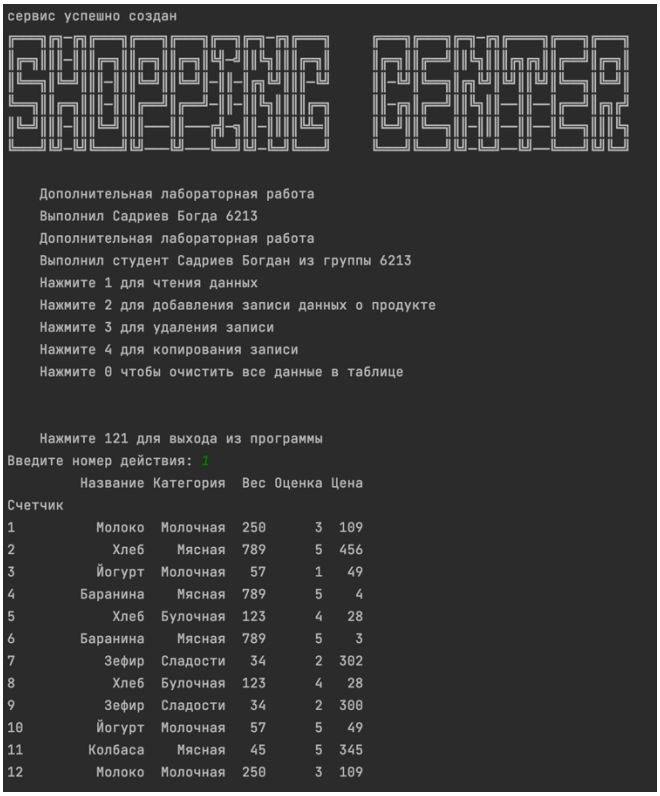

Рисунок 3 – Чтение данных из таблицы.