# 

# **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»**

# **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ»**



##### **Факультет управления и информатики в технологических системах**

**Направление подготовки (бакалавриат) 09.03.02 Информационные системы и технологии**

**Отчет**

**по учебной практике(ознакомительная)**

наименование (вид) практики

Выполнил студент гр. У-223

Извеков Е.А.\_\_\_\_\_\_\_\_\_

###### (ф.и.о.)

###### \_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

###### Проверили:

\_\_\_\_\_\_\_Доцент Денисенко В.В.\_\_\_\_\_\_\_

*(должность руководителя от организации) (ф.и.о.)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

**МП**  *(дата)*

\_\_\_Доцент Денисенко В.В.\_\_\_\_\_\_\_\_

*(должность руководителя от университета) (ф.и.о)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

Воронеж – 2023

Содержание

1. Введение ………………………………………………………………………3

2. Индивидуальное задание №53……………………………………………….5

3. Реализация индивидуального задания………………………………………6

4. Пример реализации задания………………………………………………….8

5. Отчет по программе…………………………………………………………..11

6. Заключение……………………………………………………………………18

7. Список используемых источников…………………………………………..19

Введение

Python — это скриптовый язык программирования. Он универсален, поэтому подходит для решения разнообразных задач и многих платформ, начиная с iOS и Android и заканчивая серверными ОС.

Задумка по реализации языка появилась в конце 1980-х годов, а разработка его реализации началась в 1989 году сотрудником голландского института CWI Гвидо ван Россумом. Для распределённой операционной системы Amoeba требовался расширяемый скриптовый язык, и Гвидо начал разрабатывать Python на досуге, позаимствовав некоторые наработки для языка ABC (Гвидо участвовал в разработке этого языка, ориентированного на обучение программированию). В феврале 1991 года Гвидо опубликовал исходный текст в группе новостей alt.sources. С самого начала Python проектировался как объектно-ориентированный язык.

Гвидо ван Россум назвал язык в честь популярного британского комедийного телешоу 1970-х «Летающий цирк Монти Пайтона», поскольку автор был поклонником этого телешоу, как и многие другие разработчики того времени, а в самом шоу прослеживалась некая параллель с миром компьютерной техники.

Это интерпретируемый язык — он не компилируется, то есть до запуска

представляет из себя обычный текстовый файл. Программировать можно

практически на всех платформах, язык хорошо спроектирован и логичен.

Разработка идёт в разы быстрее, потому что кода здесь куда меньше, чем

на других языках. И ещё Python отлично подходит новичкам.

Его можно встретить в вебе и на мобильных устройствах, в приложениях

и решениях, связанных с машинным обучением (нейросети и искусственный

интеллект), а также в качестве встроенной системы.

Чаще всего Python используется в веб-разработке. Для работы с ним

подключают фреймворки: Pyramid, Pylons, TurboGears, Flask, CherryPy и — самый популярный — Django.

Мобильная разработка на Python менее популярна. Для Android чаще пишут на Java, C#, C++ или Kotlin, а для iOS — на Swift или Objective-C. На Python обычно программируют серверную часть приложения. Например, клиент Instagram для iOS написан на Objective-C, а сервер — на Python.

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

Индивидуальное задание №53

Добавление данных в гугл-таблицу через чат-бот. Есть таблица с датами на месяц, в сообщении отправляем дату и описание события, если для даты ячейка с событием уже записана, записываем в следующую свободную.

Реализация индивидуального задания №53

Цель работы: написать программу(чат-бот), котороя будет производить добавление введенных пользователем даты и описания в гугл-таблицы.

Решение: Программа выполнена в среде PyCharm Community Edition (v 2022.2.1) на macOs Mojave версии 10.14.6.

Перед тем как начать процесс написания кода, я выбираю мессенджер для реализации чат-бота. Мой выбор остановился на telegram, потому что я уже работал в этой среде, знаю нужные мне библиотеки.

Создание своего чат-бота.

Пишем Вotfather, там я указываю нужные мне параметры, такие как, название, имя. После этого я получаю его токен и бот готов к написанию кода.

Изучение гугл-библиотек и их документаций.

Я перешел на официальную страницу гугл-таблицы и начал изучать ее. Там я выяснил, что мне нужно будет создать проект к которому необходимо будет подключить 2 необходимых API: Google Drive – для работы с документами и Google Sheets – непосредственно для работы в гугл-таблицах. После этого обязательно нужно создать сервисный аккаунт, который будет выполнять роль внесения изменений в гугл-таблицах, предоставить ему доступ к проекту на уровне «редактор». Файлом скачаются все необходимые данные для этого сервисного аккаунта, а именно его api, электронная почта, ключ для авторизации в сервисах гугл и необходимый токен, все это будет в формате json. Далее в документации нахожу готовый образец по считыванию и добавлению данных в гугл-таблицу. Можно приступать к написанию кода.

Установка библиотек.

Я решил написать программу в 2-ух файлах. В одной будет бот, который будет работать непосредственно с пользователем, а во 2-ой будет производиться редактирование или внесение данных в гугл-таблицу.

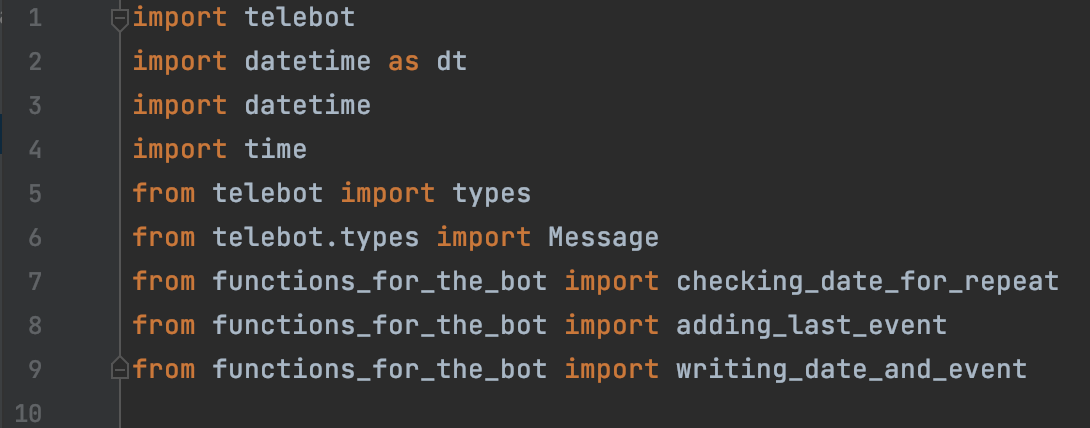


Рисунок 1. Необходимые библиотеки для работы 1-го файла

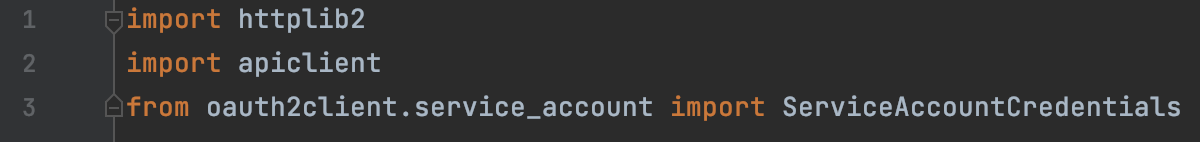


Рисунок 2. Необходимые библиотеки для *работы* 2-го файла

Работа кода.

Далее я создаю переменную bot в которую вставляю ранее сохраненный api бота. Реализую приветственное окно, бот кратко знакомит пользователя со своими функциям. Одна из возможностей бота это «Добавлять даты и описания событий». Нажимая на эту кнопку, пользователю предоставляется возможность ввести дату. Если введенная дата меньше той, что отправлена в чат, бот отправит сообщение, «Извини, тобой введенная дата не соответствует образцу или она меньше той, которая сейчас, обратите внимание на год. Повторите попытку.». И пользователь сразу может повторить попытку без необходимости перезапуска программы. Далее эту дату бот отправляет во 2-ой файл, где в свою очередь она должна провериться в таблице на повторяемость. Но перед этим необходимо авторизоваться в гугл сервисах и это происходит автоматически, без участия пользователя. Если введенная дата есть в таблице, то бот предлагает просто добавить описание или завершить процесс. Если этой даты в таблице не обнаружено, то он находит последнюю заполненную ячейку и на следующей добавляет дату, после предлагает ввести описание и оно вносится напротив даты уже в новую ячейку. После проделанных операций, он выводит ссылку измененной гугл-таблицы.

Пример реализации задания:

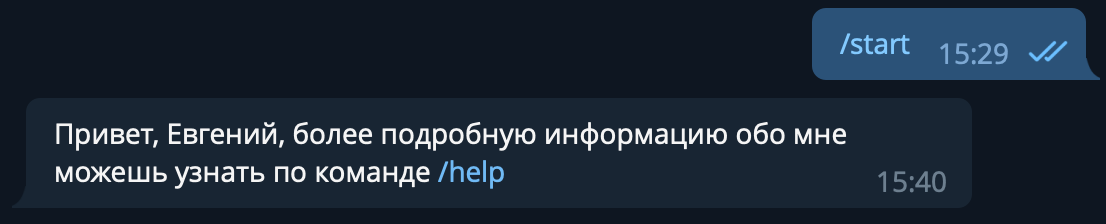


Рисунок 3. Приветственное окно

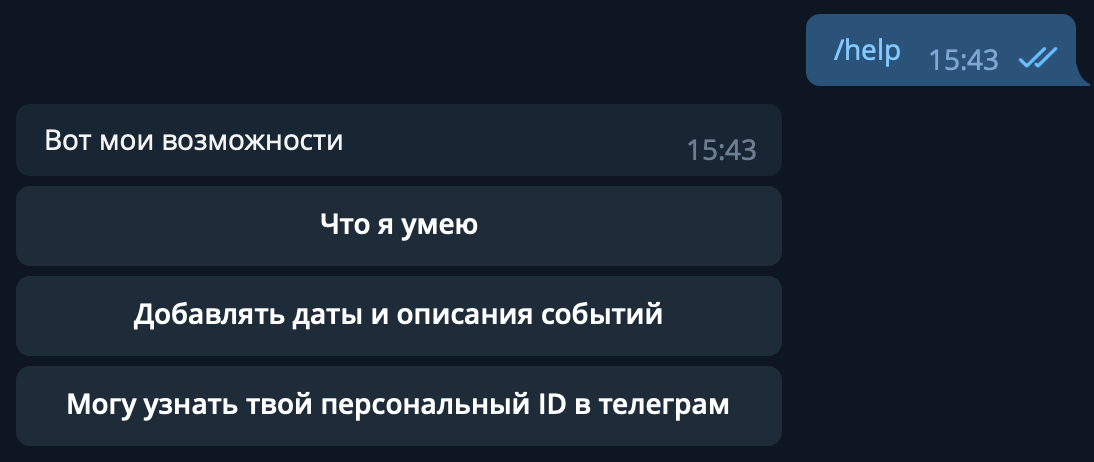


Рисунок 4. Функции бота

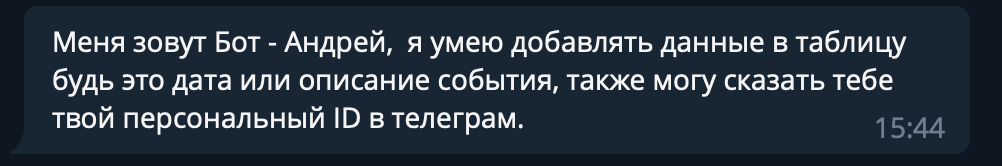


Рисунок 5. Описание функций бота

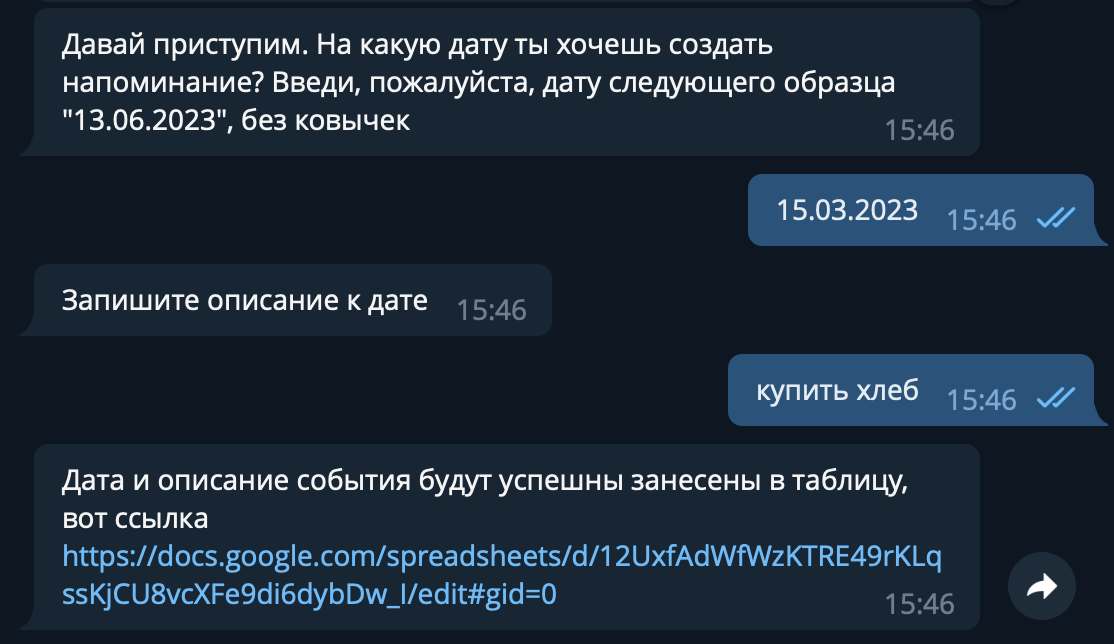


Рисунок 6. Добавление неповторяющийся даты и описания

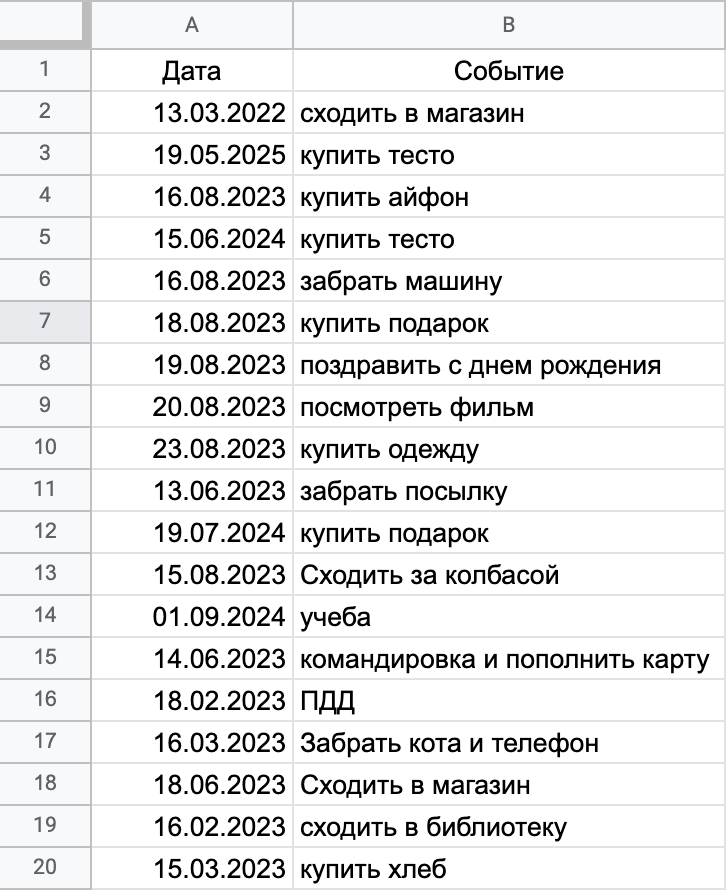


Рисунок 7. Результат добавления

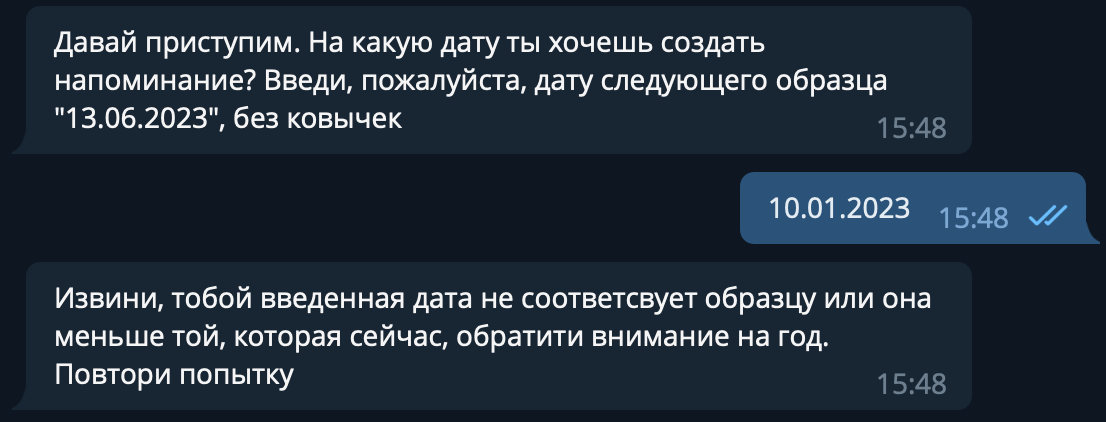


Рисунок 8. Добавление ошибочной даты

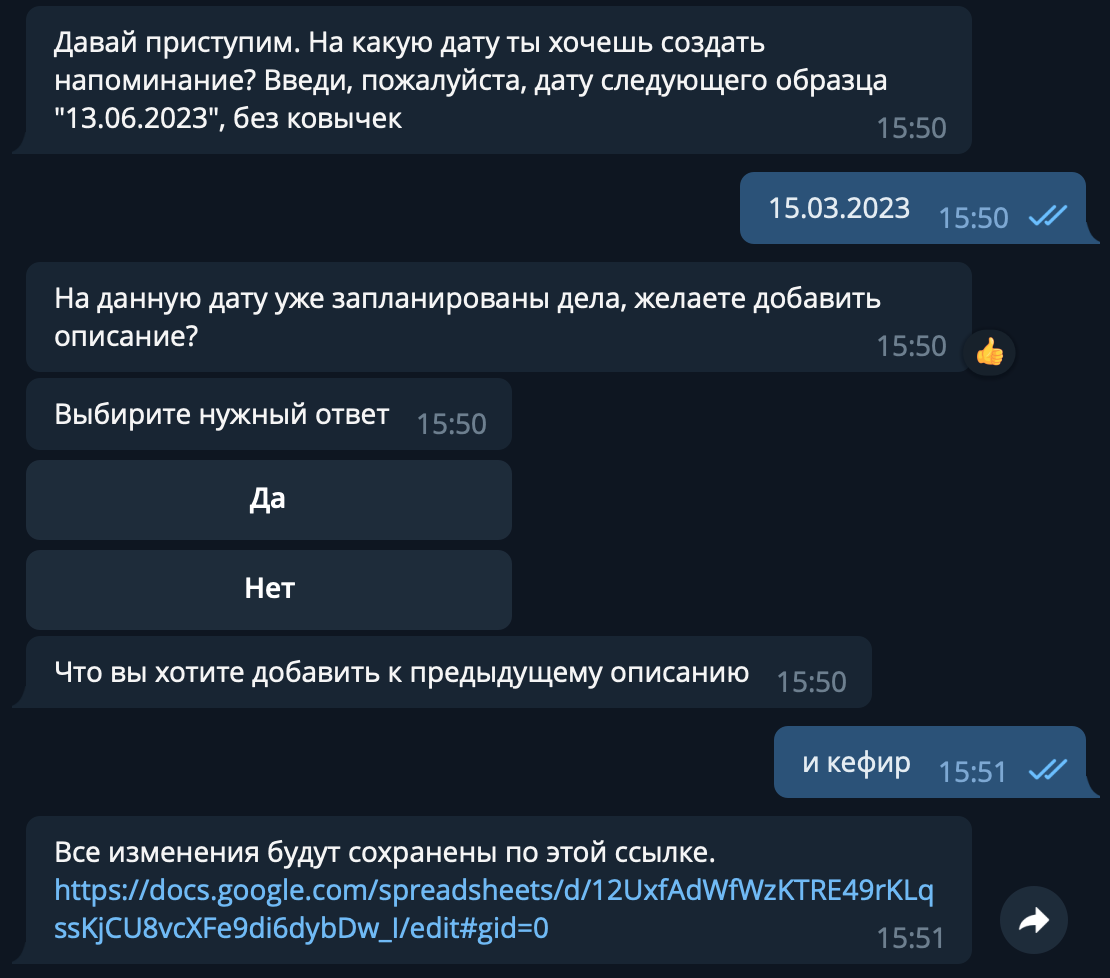


Рисунок 9. Добавление повторяющийся даты



Рисунок 10. Добавление повторяющийся даты

Отсчет по программе.

Задание: реализовать чат-бот, который получает дату и описание от пользователя и вносит их в гугл-таблицу.

Решение:

Файл с ботом:

import telebot

import datetime as dt

import datetime

import time

from telebot import types

from telebot.types import Message

from google.draf\_google\_sheets import checking\_date\_for\_repeat

from google.draf\_google\_sheets import adding\_last\_event

from google.draf\_google\_sheets import writing\_date\_and\_event

bot = telebot.TeleBot('5860576074:AAEgRzZjXV6xDYZzkvEASR2iCeK1bHaRAKA')

def date\_check(date):

try:

dates = dt.datetime.strptime(date, '%d.%m.%Y')

if dt.datetime.now() < dates:

if len(date.split('.')) == 3:

try:

datetime.datetime.strptime(date, '%d.%m.%Y')

date = True

return date

except Exception:

date = False

return date

else:

date = False

return date

else:

date = False

return date

except:

return False

users = {}

@bot.message\_handler(commands=['start'])

def start(message: Message):

bot.send\_message(message.chat.id, f'Привет, {message.from\_user.first\_name}, более подробную информацию обо мне можешь узнать по команде /help ')

@bot.message\_handler(commands=['help'])

def help(message: Message):

markup = types.InlineKeyboardMarkup()

markup.add(types.InlineKeyboardButton(text='Что я умею', callback\_data='get\_info'))

markup.add(types.InlineKeyboardButton(text='Добавлять даты и описания событий', callback\_data='create\_an\_even'))

markup.add(types.InlineKeyboardButton(text='Могу узнать твой персональный ID в телеграм', callback\_data='get\_my\_tgId'))

bot.send\_message(message.chat.id, 'Вот мои возможности', reply\_markup=markup)

@bot.callback\_query\_handler(func=lambda call: True)

def query\_handler(call):

if call.data == 'get\_info':

bot.send\_message(call.message.chat.id, 'Меня зовут Бот - Андрей, я умею добавлять данные в таблицу будь это дата или описание события, также могу сказать тебе твой персональный ID в телеграм.')

if call.data == 'get\_my\_tgId':

bot.send\_message(call.message.chat.id, f'Твой телеграм ID: {call.message.from\_user.id}')

if call.data == 'create\_an\_even':

msg = bot.send\_message(call.message.chat.id, 'Давай приступим. На какую дату ты хочешь создать напоминание? Введи, пожалуйста, дату следующего образца "13.06.2023", без ковычек')

bot.register\_next\_step\_handler(msg, user\_answer)

if call.data == 'YES':

msg = bot.send\_message(call.message.chat.id, 'Что вы хотите добавить к предыдущему описанию')

bot.register\_next\_step\_handler(msg, adding\_last\_event)

time.sleep(15)

bot.send\_message(call.message.chat.id, 'Все изменения будут сохранены по этой ссылке. https://docs.google.com/spreadsheets/d/12UxfAdWfWzKTRE49rKLqssKjCU8vcXFe9di6dybDw\_I/edit#gid=0')

if call.data == 'NO':

msg = bot.send\_message(call.message.chat.id, 'Хорошо, на этом тогда закончим')

def user\_answer(message):

if date\_check(message.text) == False:

msg = bot.send\_message(message.chat.id, 'Извини, тобой введенная дата не соответсвует образцу или она меньше той, которая сейчас, обратити внимание на год. Повтори попытку')

bot.register\_next\_step\_handler(msg, user\_answer)

time.sleep(3)

else:

answer\_date = message.text

if checking\_date\_for\_repeat(answer\_date) == True:

bot.send\_message(message.chat.id, 'На данную дату уже заплaнированы дела, желаете добавить описание?')

markup = types.InlineKeyboardMarkup()

markup.add(types.InlineKeyboardButton(text='Да', callback\_data='YES'))

markup.add(types.InlineKeyboardButton(text='Нет', callback\_data='NO'))

bot.send\_message(message.chat.id, 'Выбирите нужный ответ', reply\_markup=markup)

else:

answer\_description = bot.send\_message(message.chat.id, 'Запишите описание к дате')

abs = bot.register\_next\_step\_handler(answer\_description, writing\_date\_and\_event, answer\_date)

time.sleep(15)

bot.send\_message(message.chat.id, 'Дата и описание события будут успешны занесены в таблицу, вот ссылка https://docs.google.com/spreadsheets/d/12UxfAdWfWzKTRE49rKLqssKjCU8vcXFe9di6dybDw\_I/edit#gid=0')

bot.polling(none\_stop=True)

2-ой файл с внесением изменений в гугл таблицу:

import httplib2

import apiclient

from oauth2client.service\_account import ServiceAccountCredentials

creds = 'creds.json' #my\_APIs

table\_ID = '12UxfAdWfWzKTRE49rKLqssKjCU8vcXFe9di6dybDw\_I' #google sheets ID

read\_from\_file = ServiceAccountCredentials.from\_json\_keyfile\_name(creds, ['https://www.googleapis.com/auth/spreadsheets', 'https://www.googleapis.com/auth/drive'])

identification = read\_from\_file.authorize(httplib2.Http()) #log\_in

choose\_for\_working = apiclient.discovery.build('sheets', 'v4', http=identification)

#working with google libraries for reading

def checking\_date\_for\_repeat(date):

global checkpoint\_for\_repeat , last\_checkpoint

values = choose\_for\_working.spreadsheets().values().get(

spreadsheetId=table\_ID,

range='A2:A50',

majorDimension='ROWS').execute()

for i in range (len(values['values'])):

if date == values['values'][i][0]:

checkpoint\_for\_repeat = i

last\_checkpoint = len(values['values'])

return True

elif date != values['values'][i][0]:

checkpoint\_for\_repeat = i

last\_checkpoint = len(values['values'])

if i == len(values['values']) + 1:

return False

def writing\_date\_and\_event(event, date):

check = last\_checkpoint + 2

checkA = 'A' + str(check)

checkB = 'B' + str(check)

values = choose\_for\_working.spreadsheets().values().batchUpdate(

spreadsheetId=table\_ID,

body={

"valueInputOption": "USER\_ENTERED",

"data": [

{'range': f'{checkA}:{checkB}',

"majorDimension": "COLUMNS",

"values": [[f'{date}'], [f'{event.text}']]

}

]

}).execute()

def adding\_last\_event(event):

check = checkpoint\_for\_repeat

values = choose\_for\_working.spreadsheets().values().get(

spreadsheetId=table\_ID,

range='B2:B50',

majorDimension='ROWS').execute()

old\_description = values['values'][check][0]

check += 2

checkB = 'B' + str(check)

event = str(old\_description) + ' ' + str(event.text)

values = choose\_for\_working.spreadsheets().values().batchUpdate(

spreadsheetId=table\_ID,

body={

"valueInputOption": "USER\_ENTERED",

"data": [

{'range': f'{checkB}',

"majorDimension": "COLUMNS",

"values": [[f'{event}']]

}

]

}).execute()

Заключение.

В ходе учебной практики, я научился искать нужную информацию в интернете, анализировать чужие написанные приложения и извлекать из них полезные данные для написания своей программы. Также приобрел навыки создания примерной структуры и функций кода. Познакомился с новыми библиотеками google. Проведены тестовые испытания программы, которые показали, что код работает правильно, ошибок не выдает.

Список используемых источников

1. Google sheets [Электронный ресурс] - URL. Режим доступа: https://developers.google.com/sheets/api/reference/rest/v4/spreadsheets.values/append
2. Начинаем работу с Google Sheets на Python. От регистрации до чтения данных [Электронный ресурс] – URL. Режим доступа: https://habr.com/ru/post/483302/
3. Google Sheets API + Python. Чтение и запись с Сервисным Аккаунтом Google Cloud [Электронный ресурс] – URL. Режим доступа: https://habr.com/ru/post/575160/
4. Языки программирования [Электронный ресурс] – URL. Режим доступа: https://dvsemenov.ru/google-tablicy-i-python-podrobnoe-rukovodstvo-s-primerami/
5. Google API Sheets in Python Чтение и Запись Данных с Помощью Сервисного Аккаунта [Электронный ресурс] – URL. Режим доступа: https://it4each.com/blog/google-api-sheets-in-python-chtenie-i-zapis-dannykh-s-pomoshchiu-servisnogo-akkaunta/