|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

**Отчет по лабораторной работе №6**

**по дисциплине Базовые компоненты интернет технологии**

#### Тема работы: "Разработка бота на основе конечного автомата для Telegram с использованием языка Python"

**Выполнил:**

Студент группы ИУ5Ц-54Б

Перевощиков Н.Д. 09.12.22 г.

(дата, подпись)

**Проверил:**

Преподаватель

Канев А.И.

(дата, подпись)

Москва, 2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ ОТЧЕТА**

[1. Цель лабораторной работы 3](#_Toc122114329)

[2. Описание задания 3](#_Toc122114330)

[3. Текст программы 4](#_Toc122114331)

[Config.py 4](#_Toc122114332)

[\_\_init\_\_.py 4](#_Toc122114333)

[Arithmetic\_calculate.py 4](#_Toc122114334)

[Bot\_telegram.py 6](#_Toc122114335)

[Json\_function.py 9](#_Toc122114336)

[Work\_calculate.py 10](#_Toc122114337)

[Moscow.jpg 11](#_Toc122114338)

[4. Результаты работы программы 12](#_Toc122114339)

## Цель лабораторной работы

Изучение разработки ботов в Telegram.

## Описание задания

Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.

1. Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

## Текст программы

## Config.py

Здесь должен быть токен для доступа к HTTP API

token = ''

## \_\_init\_\_.py

print('Directory calculate')

## Arithmetic\_calculate.py

class mathematical\_calculator(object):  
 def \_\_init\_\_(self, math\_calculate):  
 self.math\_calculation = math\_calculate  
 self.math\_calculation\_list = self.convert\_string\_in\_list(math\_calculate)  
 self.list\_enumeration\_of\_sign = self.enumeration\_of\_sign(self.math\_calculation\_list)  
  
 self.type\_error = None  
  
 for sign in self.list\_enumeration\_of\_sign:  
 self.arithmetic\_operations(sign, self.math\_calculation\_list)  
  
 if(self.type\_error == None):  
 self.result = float(self.math\_calculation\_list[0])  
  
 # Преобразование тип строки в list  
 def convert\_string\_in\_list(self, str\_calculate):  
 str\_1 = []  
 str\_meaning = ''  
 for i in str\_calculate:  
 if(i != ' '):  
 str\_meaning += i  
 else:  
 str\_1.append(str\_meaning)  
 str\_meaning = ''  
 #append () добавляет в конец списка элемент, переданный ему в качестве аргумента.  
 str\_1.append(str\_meaning)  
  
 return str\_1  
  
 #Растановка приоритета операции  
 def enumeration\_of\_sign(self, list\_str):  
 count\_list = []  
 for i in list\_str:  
 if ('\*' == i): count\_list.append(i)  
 if ('/' == i): count\_list.append(i)  
 if ('+' == i): count\_list.append(i)  
 if ('-' == i): count\_list.append(i)  
  
 count\_list = self.priority(count\_list)  
  
 return count\_list  
  
 #Подержка функции по расстановку приоритета операции  
 def priority(self, list\_str):  
 list\_1 = []  
 size = len(list\_str)  
 count = 0  
 while (size != 0):  
 if ('\*' in list\_str or '/' in list\_str):  
 for i in list\_str:  
 if (i == '\*' or i == '/'):  
 list\_1.append(i)  
 size -= 1  
 if ('+' in list\_str or '-' in list\_str):  
 for i in list\_str:  
 if (i == '+' or i == '-'):  
 list\_1.append(i)  
 size -= 1  
  
 return list\_1  
  
 #Арифметические операции  
 def arithmetic\_operations(self, sign, list):  
 result = None  
 if (sign in list):  
 for i in range(1, len(list)-1):  
 try:  
 if (list[i] == sign):  
 if (sign == '\*'):  
 result = float(list[i - 1]) \* float(list[i + 1])  
 elif (sign == '/'):  
 result = float(list[i - 1]) / float(list[i + 1])  
 elif (sign == '+'):  
 result = float(list[i - 1]) + float(list[i + 1])  
 elif (sign == '-'):  
 result = float(list[i - 1]) - float(list[i + 1])  
  
 list[i] = result  
 del list[i - 1: i]  
 del list[i: i + 1]  
  
 #Деление на 0  
 except ZeroDivisionError:  
 self.type\_error = 'Division by 0'  
 self.result = 'infinity'  
  
 #Граница вне диапазона  
 except:  
 return result  
  
 def calculate(self, math\_calculate):  
 self.math\_calculation = math\_calculate  
 self.math\_calculation\_list = self.convert\_string\_in\_list(math\_calculate)  
 self.list\_enumeration\_of\_sign = self.enumeration\_of\_sign(self.math\_calculation\_list)  
  
 self.type\_erorr = None  
  
 for sign in self.list\_enumeration\_of\_sign:  
 self.arithmetic\_operations(sign, self.math\_calculation\_list)  
  
 if(self.type\_erorr == None):  
 self.result = float(self.math\_calculation\_list[0])  
  
 return self

## Bot\_telegram.py

import config  
import telebot  
from telebot import types  
import random  
  
#SageMathCloud (сокращённо SMC) — это онлайновый сервис, в котором можно написать математический или любой другой расчёт.  
from arithmetic\_calculate import mathematical\_calculator as smc  
from json\_function import combined\_data, delete\_data\_for\_id\_user, load\_data\_for\_id\_user  
from work\_calculate import generator\_of\_meaning  
  
# Создание бота  
bot = telebot.TeleBot(config.token)  
  
SPISOK = '''  
/menu - Меню (кнопки переключателя)  
/calculate - Бот-калькулятор (посчет арифметических операций)  
/story - Просмотр история вычисления  
/cleaner - Очистка истории вычисления  
/generation - Генерация случайных вычислений  
/image - Просмотр изображения  
'''  
  
# Список меню  
@bot.message\_handler(commands=['spisok'])  
def menu(message):  
 bot.send\_message(message.chat.id, SPISOK)  
  
# При нажатии на /menu  
@bot.message\_handler(commands=['menu'])  
def menu(message):  
 markup = types.InlineKeyboardMarkup(row\_width=1)  
 btn\_1 = types.InlineKeyboardButton(text="Решить пример", callback\_data='btn\_1')  
 btn\_2 = types.InlineKeyboardButton(text="Посмотреть историю вычисления", callback\_data='btn\_2')  
 btn\_3 = types.InlineKeyboardButton(text="Очистить истории вычисления", callback\_data='btn\_3')  
 btn\_4 = types.InlineKeyboardButton(text="Генерировать вычисления", callback\_data='btn\_4')  
 btn\_5 = types.InlineKeyboardButton(text="Посмотреть изображение", callback\_data='btn\_5')  
 markup.add(btn\_1, btn\_2, btn\_3, btn\_4, btn\_5)  
 bot.send\_message(message.chat.id, text=f"Привет, {message.from\_user.first\_name}! Выберите то, что Вам нужно",  
 reply\_markup=markup)  
  
# Кнопки переключателя при нажатии на /menu  
@bot.callback\_query\_handler(func=lambda callback: callback.data)  
def check\_callback\_data(callback):  
 #Пользовательский индентификатор  
 user\_id = str(callback.from\_user.id)  
  
 if (callback.data == "btn\_1"):  
 bot.send\_message(callback.message.chat.id, 'Напишите пример вычсления')  
  
 # Пользовательский индентификатор  
 user\_id = str(callback.from\_user.id)  
  
 @bot.message\_handler(content\_types=["text"])  
 def info(message):  
 meaning = smc(message.text)  
 bot.send\_message(message.chat.id, f'Решение: {meaning.result}')  
 data = {  
 user\_id: [{"id": random.randint(0, 10000),  
 "meaning": str(message.text),  
 "result": str(meaning.result)}]  
 }  
 combined\_data(data, str(message.from\_user.id))  
  
 elif (callback.data == "btn\_2"):  
 bot.send\_message(callback.message.chat.id, 'История вычисления')  
 data = load\_data\_for\_id\_user(str(user\_id))  
 if(data == 'Ошибка! Такого идентификатора не существует.'):  
 bot.send\_message(callback.message.chat.id, 'Нет базы данных')  
 else:  
 for j in range(len(data) - 1):  
 id = data[j]['id']  
 meaning = data[j]['meaning']  
 result = data[j]['result']  
 print\_info = f'id:{id}\n{meaning} = {result}\n\n'  
 bot.send\_message(callback.message.chat.id, print\_info)  
  
 elif(callback.data == "btn\_3"):  
 bot.send\_message(callback.message.chat.id, 'Очистка истории вычисления')  
  
 check\_error = delete\_data\_for\_id\_user(user\_id)  
  
 if(check\_error != 'Ошибка! Такого идентификатора не существует.'):  
 bot.send\_message(callback.message.chat.id, 'Операция прошла успешно')  
 else:  
 bot.send\_message(callback.message.chat.id, check\_error)  
  
 elif (callback.data == "btn\_4"):  
 bot.send\_message(callback.message.chat.id, 'Генерация случайных вычислений')  
 generator\_of\_meaning(user\_id)  
 bot.send\_message(callback.message.chat.id, 'Операция прошла успешно')  
  
 elif (callback.data == "btn\_5"):  
 img = open('moscow.jpg', 'rb')  
 bot.send\_photo(callback.message.chat.id, img)  
  
 else:  
 bot.send\_message(callback.chat.id, 'Нет такой команды. Введите /spisok')  
  
# Вычисления  
@bot.message\_handler(commands=['calculate'])  
def start\_calculate(message):  
 bot.send\_message(message.chat.id, 'Напишите пример вычисления')  
  
 # Пользовательский идентификатор  
 user\_id = str(message.from\_user.id)  
  
 @bot.message\_handler(content\_types=["text"])  
 def info(message):  
 meaning = smc(message.text)  
 bot.send\_message(message.chat.id, f'Результат решения: {meaning.result}')  
 data = {  
 user\_id: [{"id": random.randint(0, 10000),  
 "meaning": str(message.text),  
 "result": str(meaning.result)}]  
 }  
 combined\_data(data, str(message.from\_user.id))  
  
# Просмотр история вычисления  
@bot.message\_handler(commands=['story'])  
def start\_story(message):  
 bot.send\_message(message.chat.id, 'История вычисления')  
  
 # Пользовательский идентификатор  
 user\_id = str(message.from\_user.id)  
  
 data = load\_data\_for\_id\_user(str(user\_id))  
  
 if (data == 'Ошибка! Такого идентификатора не существует.'):  
 bot.send\_message(message.chat.id, 'Нет базы данных')  
 else:  
 for j in range(len(data) - 1):  
 id = data[j]['id']  
 meaning = data[j]['meaning']  
 result = data[j]['result']  
 print\_info = f'id: {id}\n{meaning} = {result}\n\n'  
 bot.send\_message(message.chat.id, print\_info)  
  
#Просмотр изображения  
@bot.message\_handler(commands=['image'])  
def start\_image(message):  
 img = open('moscow.jpg', 'rb')  
 bot.send\_photo(message.chat.id, img)  
  
#Генерация случайных вычислений  
@bot.message\_handler(commands=['generation'])  
def start\_generation(message):  
 bot.send\_message(message.chat.id, 'Генерация случайных вычислений')  
  
 #Пользовательский идентификатор  
 user\_id = str(message.from\_user.id)  
  
 generator\_of\_meaning(user\_id)  
  
 bot.send\_message(message.chat.id, 'Операция прошла успешно')  
  
#Очистка истории вычисления  
@bot.message\_handler(commands=['cleaner'])  
def start\_cleaner(message):  
 bot.send\_message(message.chat.id, 'Очистка истории вычислений')  
  
 # Пользовательский идентификатор  
 user\_id = str(message.from\_user.id)  
  
 check\_error = delete\_data\_for\_id\_user(user\_id)  
  
 if (check\_error != 'Ошибка! Такого идентификатора не существует.'):  
 bot.send\_message(message.chat.id, 'Операция прошла успешно')  
 else:  
 bot.send\_message(message.chat.id, check\_error)  
  
#Работа программы в телеграме без остановки  
bot.polling(none\_stop=True)

## Json\_function.py

import json  
  
file\_locator = 'D:\Работа\МГТУ им. Н.Э.Баумана\Программирование\Программы\Программы за 5 семестр\LAB\_06\calculate\data'  
  
#Запись данных  
def data\_recording(data, title=file\_locator):  
 with open(f"{title}.json", "w", encoding="utf-8") as file:  
 json.dump(data, file, indent=2, ensure\_ascii=False)  
  
#Чтение данных  
def load\_data(title=file\_locator):  
 with open(f"{title}.json", "r") as file:  
 data = json.load(file)  
 return data  
  
#Добавление данных  
def combined\_data(data\_json, id\_user='id\_user', title=file\_locator):  
 #Если есть файл и не пустой  
 try:  
 with open(f"{title}.json", encoding="utf-8") as file:  
 data = json.load(file)  
 temp = data[id\_user]  
 for info\_data in data\_json[id\_user]:  
 n = {  
 'id': info\_data['id'],  
 'meaning': info\_data['meaning'],  
 'result': info\_data['result']  
 }  
 temp.append(n)  
 data\_recording(data)  
 #Если нет файла  
 except:  
 data\_recording(data\_json)  
  
#Загрузка данных для идентификатора пользователя  
def load\_data\_for\_id\_user(id\_user, title=file\_locator):  
 try:  
 with open(f"{title}.json", "r", encoding="utf-8") as file:  
 data = json.load(file)  
 temp = data[id\_user]  
 for info\_data in data[id\_user]:  
 n = {  
 'id': info\_data['id'],  
 'meaning': info\_data['meaning'],  
 'result': info\_data['result']  
 }  
 temp.append(n)  
 return temp  
 except:  
 return 'Ошибка! Такого идентификатора не существует.'  
  
#Удаление данных для индентификатора пользователя  
def delete\_data\_for\_id\_user(id\_user, title=file\_locator):  
 try:  
 with open(f"{title}.json", encoding="utf-8") as file:  
 data = json.load(file)  
 data\_1 = {}  
 for id\_user\_data in data:  
 if (id\_user != id\_user\_data):  
 temp = data[id\_user\_data]  
 data\_1 = {id\_user\_data: []}  
 for j in temp:  
 n = {  
 'id': j['id'],  
 'meaning': j['meaning'],  
 'reault': j['result']  
 }  
 data\_1[id\_user\_data].append(n)  
 temp.append(data\_1)  
 data\_recording(data\_1)  
 except:  
 return 'Ошибка! Такого идентификатора не существует.'

## Work\_calculate.py

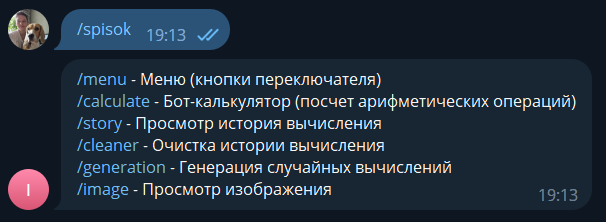
import random  
  
from calculate.json\_function import data\_recording, load\_data, combined\_data, load\_data\_for\_id\_user  
from calculate.arithmetic\_calculate import mathematical\_calculator as smc  
  
#Генерация значений  
def generator\_of\_meaning(id\_user='id\_user'):  
 arithmetic = ['+', '-', '/', '\*']  
  
 arith = arithmetic[random.randint(0, 3)]  
 gen\_id = random.randint(0, 100000)  
 m\_1 = random.randint(0, 1000)  
 m\_2 = random.randint(0, 1000)  
 class\_calculate = smc(str(m\_1) + ' ' + str(arith) + ' ' + str(m\_2))  
  
 data = {  
 str(id\_user): [  
 {"id": gen\_id,  
 "meaning": (str(m\_1) + ' ' + str(arith) + ' ' + str(m\_2)),  
 "result": class\_calculate.result}  
 ]  
 }  
 combined\_data(data, id\_user)  
  
#Получение информации  
def get\_info():  
 try:  
 data = load\_data()  
 return data  
 except:  
 return 'Нет файла'  
  
#Получение информации с идентификатором пользователя  
def get\_info\_id\_user(id\_user):  
 try:  
 data = load\_data\_for\_id\_user(id\_user)  
 return data  
 except:  
 return 'Нет файла'

## Moscow.jpg

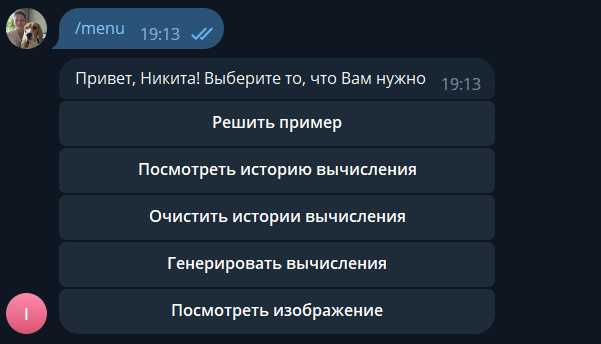


## Результаты работы программы

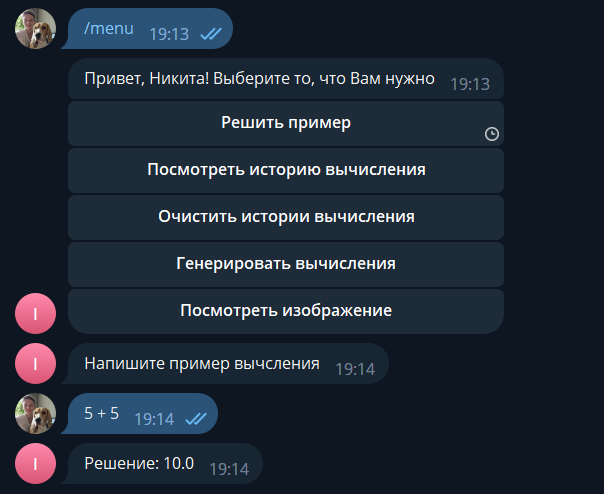
**Список меню (при вводе ссылки «/spisok»)**



**Меню переключателя (при нажатии на ссылку «/menu»)**

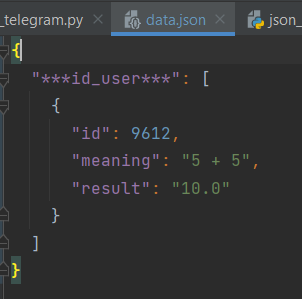


**Калькулятор вычисления (при нажатии на кнопку «Решить пример»)**

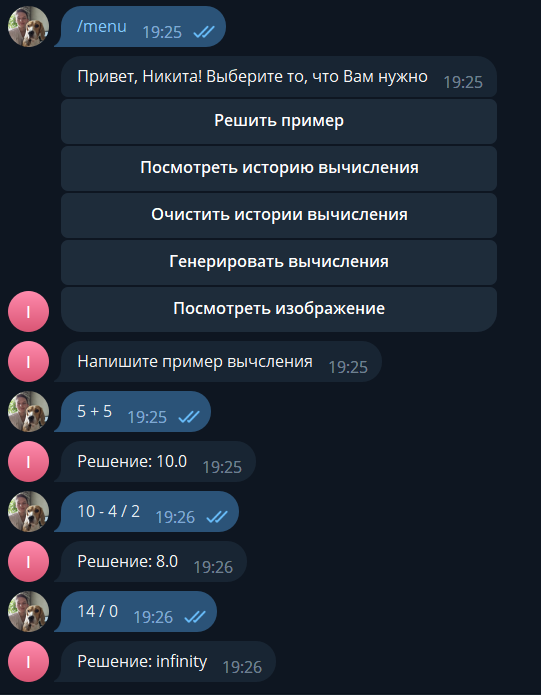


**Данные хранятся в data.json**

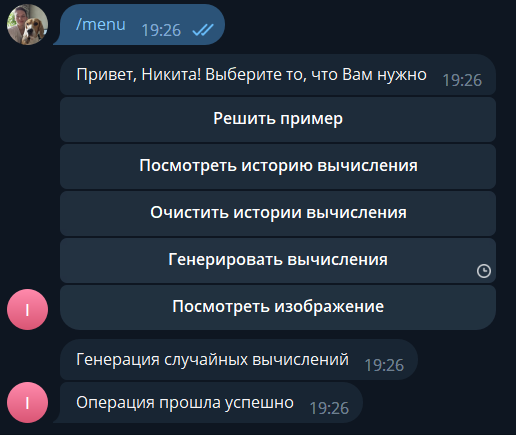
Вместо “\*\*\*id\_user\*\*\*” должны быть цифры индекса пользователя из телеграмма.



**Несколько вычислений**

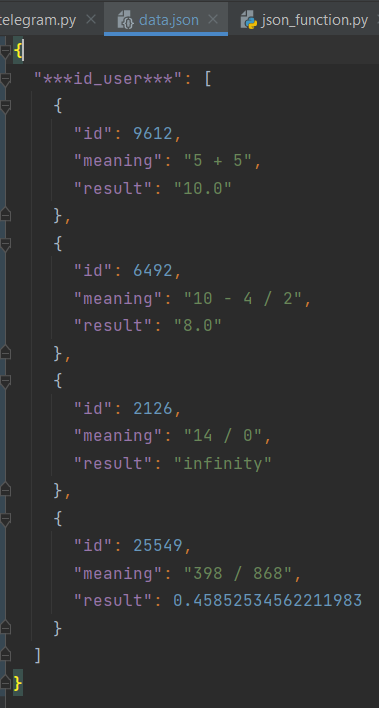


**Генерация случайных вычислений (при нажатии на кнопку «Генерировать вычисления»)**

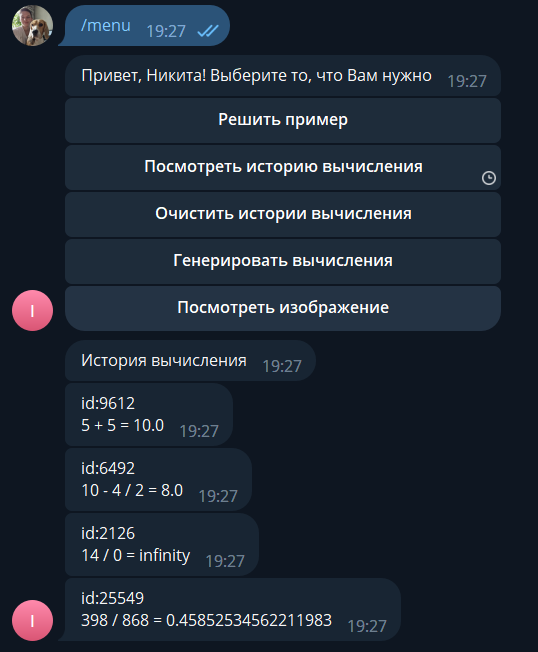


**Обновленные данные хранятся в data.json**

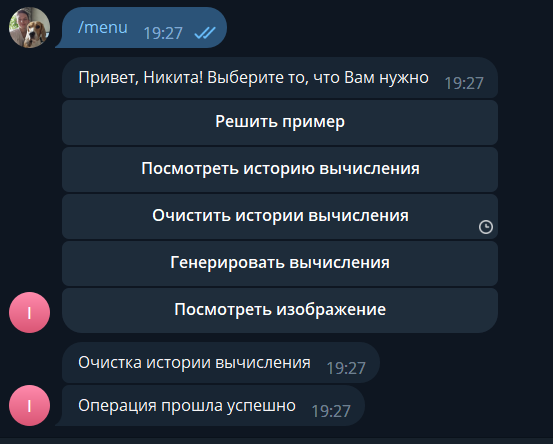
Вместо “\*\*\*id\_user\*\*\*” должны быть цифры индекса пользователя из телеграмма.



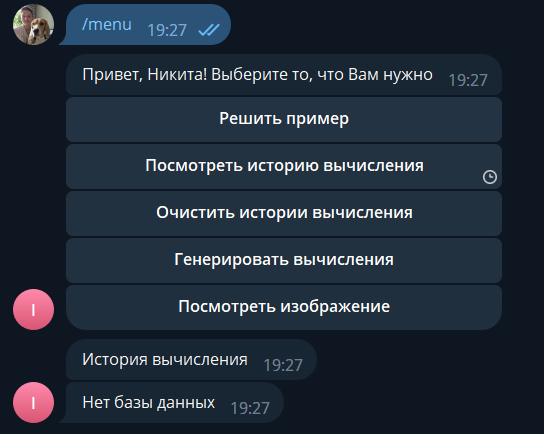
**Чтение и просмотр данные из data.json (при нажатии на кнопку «Посмотреть историю вычисления»)**



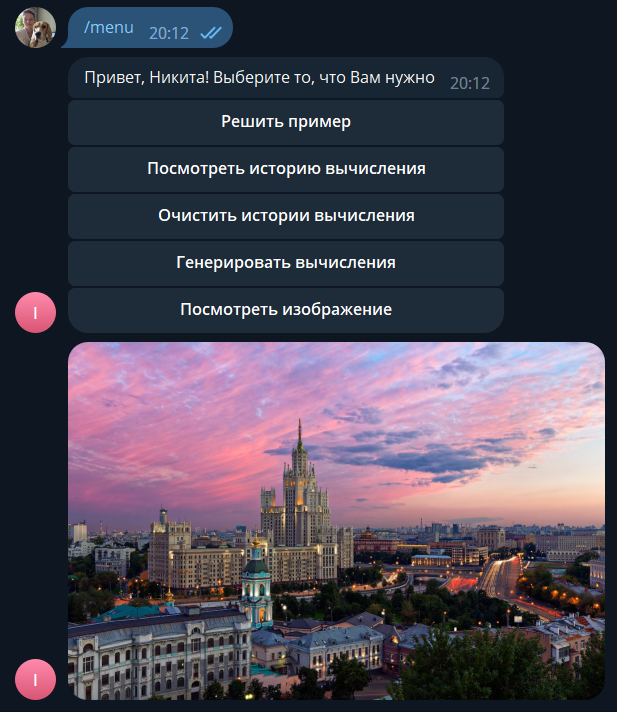
**Очистка истории вычисления (при нажатии на кнопку «Очистить истории вычисления»)**



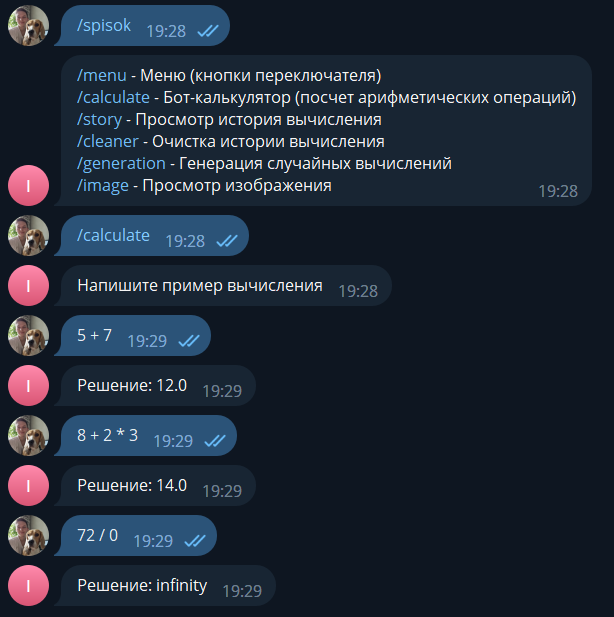
**Проверка, очистила ли история вычисления (при нажатии на кнопку «Посмотреть историю вычисления»)**



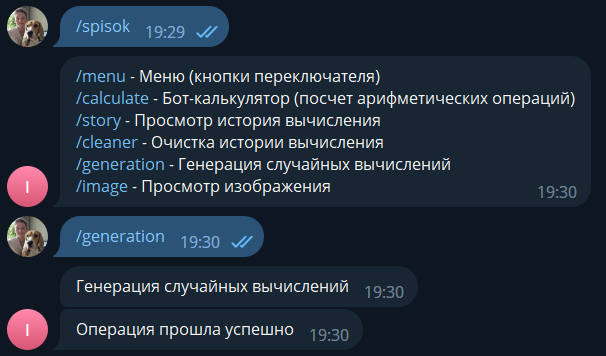
**Просмотр изображения (при нажатии на кнопку «Посмотреть изображение»)**



**Калькулятор вычисления (при нажатии на ссылку «/calculate»)**

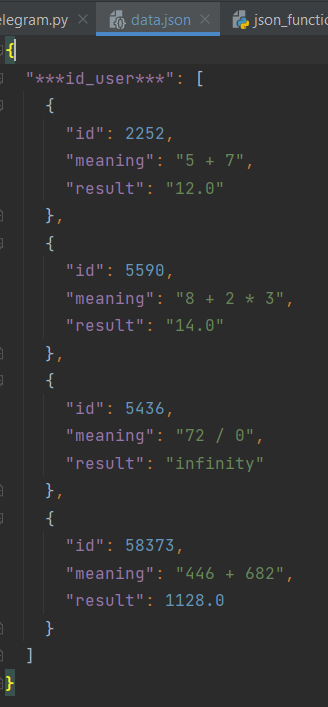


**Генерация случайных вычислений (при нажатии на ссылку «/generation»)**

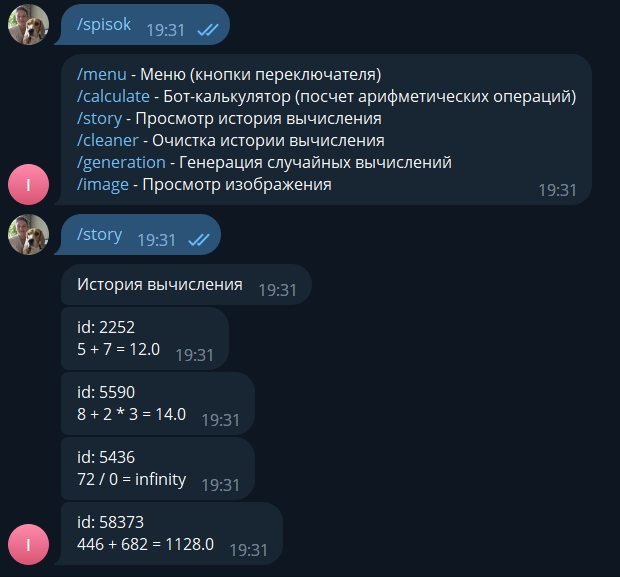


**Данные хранятся в data.json**

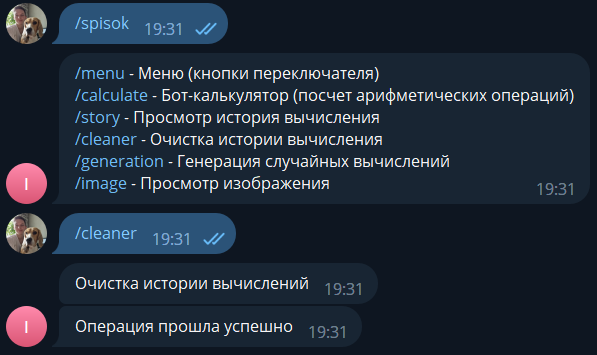
Вместо “\*\*\*id\_user\*\*\*” должны быть цифры индекса пользователя из телеграмма.



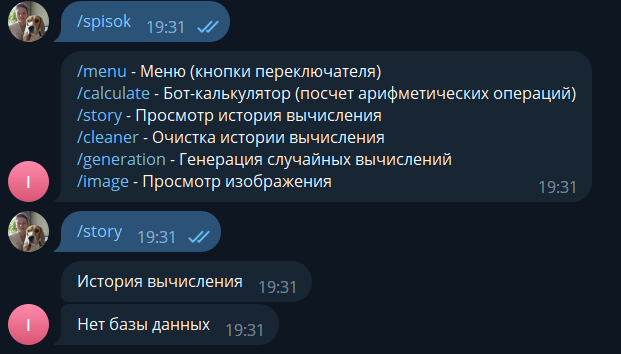
**Чтение и просмотр данные из data.json (при нажатии на ссылку «story»)**



**Очистка истории вычисления (при нажатии на ссылку «/cleaner»)**



**Проверка, очистила ли история вычисления (при нажатии на ссылку «/story»)**



**Просмотр фотографии (при нажатии на ссылку «/image»)**

