

Защищено:
Папин А.В..

Демонстрация:
Папин А.В..

"__" _____ 2022 г.

"__" _____ 2022 г.

Отчет по домашнему заданию по курсу базовые компоненты интернет-технологий (БКИТ)

Домашнее задание

35

(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-54Б
Папин Алексей

Гапанюк Ю.Е.

(подпись)

"__" _____ 2022 г.

СОДЕРЖАНИЕ ОТЧЕТА

1. Цель лабораторной работы.....	3
2. Описание задания.....	3
3. Листинг программы:	4
3.1. config.py.....	4
3.2. calculate_arifmetic.py	4
3.3. calculate_bot.py.....	6
3.4. json_function.py	10
3.5. work_with_calculate.py	11
3.6. bmstu.jpg	12
3.7. Unittest	13
3.7.1.test_calculate.py	13
3.7.2.test_telebot.py	13
3.7.3.test_json.py.....	15
3.7.4.test_filed.py (Unittest к 4 лабе).....	19
3.7.5.test_unique.py (Unittest к 4 лабе)	20
3.8. Behave	21
3.8.1.check_unique.feature.....	21
3.8.2.check_filed.feature	22
3.9. Steps (for Behave).....	23
3.9.1.filed.py.....	23
3.9.2.unique.py	23
4. Результаты работы программы в Telegram.....	25
4.1. Получение справочную информацию.....	25
4.2. Основное меню переключателя	25
4.3. Простейший калькулятор (при нажатии на кнопку «Посчитать»)	26
4.4. Данные хранятся в БД в формате JSON (Строчка кода 2 хранит id пользователя)	26
4.5. После несколько вычислений	27
4.6. Чтение и просмотр БД в Телеграме (после нажатии на кнопку Посмотреть историю вычисления)	28
4.7. Очистка история вычисления БД в Телеграме.....	29
4.8. Генерация случайных вычислений и просмотр вычислений (после нажатии на кнопку «Показать фото МГТУ им. Н.Э. Баумана»)	30
4.9. Просмотр фотографии (после нажатии на кнопку «Показать фото МГТУ им. Н.Э. Баумана»)	31
5. Модульное тестирование в IDE JetBrains PyCharm.....	32
5.1. Unittest	32

5.1.1.test_unique.py	32
5.1.2.test_telebot.py	32
5.1.3.test_json.py	32
5.1.4.test_filed.py	33
5.1.5.test_calculate.py	33
5.2. Behave	34
5.2.1.check_unique.feature.....	34
5.2.2.check_unique.feature.....	34

1. Цель лабораторной работы

Изучение возможностей создания ботов в Telegram и их тестирования.

2. Описание задания.

1. Модифицируйте код лабораторной работы №5 или №6 таким образом, чтобы он был пригоден для модульного тестирования.
2. Используя материалы лабораторной работы №4 создайте модульные тесты с применением TDD - фреймворка (2 теста) и BDD - фреймворка (2 теста).

3. Листинг программы:

3.1.config.py

```
token = ''
```

3.2.calculate_arifmetic.py

```
class the_simplest_mathematical_calculator(object):
    '''
    def __init__(self):
        self.math_calculation = ''
        self.math_calculation_list = []
        self.list_enumeration_sign = []
        self.type_error = ''
        self.result = None

    '''
    def __init__(self, math_calculate):
        self.math_calculation = math_calculate
        self.math_calculation_list =
self.delete_space_into_list(math_calculate)
        self.list_enumeration_sign =
self.enumeration_sign(self.math_calculation_list)

        self.type_error = None

        for sgin in self.list_enumeration_sign:
            self.arifmetic(sgin, self.math_calculation_list)

        if(self.type_error == None):
            self.result = float(self.math_calculation_list[0])

# Преобразование строкового типа в list
def delete_space_into_list(self, str_calculate):
    new_str = []
    str_value = ''
    for i in str_calculate:
        if(i != ' '):
            str_value += i
        else:
            new_str.append(str_value)
            str_value = ''

    new_str.append(str_value)

    return new_str

# Расстановка приоритета операции
def enumeration_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.prioritet(count_list)

        return count_list

# Поддержка функции по расстановку приоритета операции
def prioritet(self, list_str):
    new_list = []
    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 == '/'):
                    new_list.append(i)
            size -= 1
        if('+' in list_str or '-' in list_str):
            for i in list_str:
                if(i == '+' or i == '-'):
                    new_list.append(i)
            size -= 1

    return new_list

# Арифметические операции
def arifmetic(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]

            except ZeroDivisionError:
                self.type_error = 'Division by 0'
                self.result = 'inf'

# Деление на 0
except ZeroDivisionError:
    self.type_error = 'Division by 0'
    self.result = 'inf'

# Граница вне диапазона

```

```

        except:
            return result

    def calculate(self, math_calculate):
        self.math_calculation = math_calculate
        self.math_calculation_list =
self.delete_space_into_list(math_calculate)
        self.list_enumeration_sign =
self.enumeration_sign(self.math_calculation_list)

        self.type_error = None

        for sgin in self.list_enumeration_sign:
            self.arifmetic(sgin, self.math_calculation_list)

        if(self.type_error == None):
            self.result = float(self.math_calculation_list[0])

        return self

```

3.3.calculate_bot.py

```

import config
import telebot
from telebot import types
import random

from calculate_arifmetic import the_simplest_mathematical_calculator as smc
from json_function import merge_data, delete_data_for_id_user,
load_data_for_id_user
from work_with_calculate import generate_value

# Создание бота
bot = telebot.TeleBot(config.token)

HELP = '''
/start - Меню переключателя
/calculate - Калькуляторный бот, способный вычислять простейшие
арифметические операции
/get_info - Просмотр историю вычисления с БД
/clean - Очистка история вычисления
/random - Генерация случайных вычислений
/photo - Просмотр фото МГТУ им. Н.Э. Баумана
'''

# Справочник
@bot.message_handler(commands=['help'])
def start(message):
    bot.send_message(message.chat.id, HELP)

@bot.message_handler(commands=['start'])
def start(message):

```

```

markup = types.InlineKeyboardMarkup(row_width=1)
btn1 = types.InlineKeyboardButton(text="Посчитать", callback_data='btn1')
btn2 = types.InlineKeyboardButton(text="Посмотреть историю вычисления",
callback_data='btn2')
btn3 = types.InlineKeyboardButton(text="Очистить историю вычисления",
callback_data='btn3')
btn4 = types.InlineKeyboardButton(text="Рандомные вычисления",
callback_data='btn4')
btn5 = types.InlineKeyboardButton(text="Показать фото МГТУ им. Н.Э.
Баумана", callback_data='btn5')
markup.add(btn1, btn2, btn3, btn4, btn5)
bot.send_message(message.chat.id,
                  text=f"Привет, {message.from_user.first_name}! Я
тестовый бот, выберите действия",
                  reply_markup=markup)

# Функция переключателя
@bot.callback_query_handler(func=lambda callback: callback.data)
def check_callback_data(callback):
    # Пользовательский идентификатор
    user_id = str(callback.from_user.id)

    if (callback.data == "btn1"):
        bot.send_message(callback.message.chat.id, 'Калькулятор бот')
        bot.send_message(callback.message.chat.id, 'Напишите в чате
вычисления')

    # Пользовательский идентификатор
    user_id = str(callback.from_user.id)

    @bot.message_handler(content_types=["text"])
    def echo(message):
        value = smc(message.text)
        bot.send_message(message.chat.id, f'Решение: {value.result}')
        data = {
            user_id: [
                {"id": random.randint(0, 10000),
                 "value": str(message.text),
                 "result": str(value.result)}
            ]
        }
        merge_data(data, str(message.from_user.id))

    elif(callback.data == "btn2"):
        bot.send_message(callback.message.chat.id, 'История вычисления')

        data = load_data_for_id_user(str(user_id))
        if(data == 'Error! There is no such identifier'):
            bot.send_message(callback.message.chat.id, 'База данных
отсутствует')
        else:
            for j in range(len(data) - 1):
                id = data[j]['id']
                value = data[j]['value']

```



```

        result = data[j]['result']
        print_info = f'id: {id}\n{value} = {result}\n\n'
        bot.send_message(callback.message.chat.id, print_info)

    elif(callback.data == "btn3"):
        bot.send_message(callback.message.chat.id, 'Очистка истории
вычисления')

        check_error = delete_data_for_id_user(user_id)

        if(check_error != 'Error! There is no such identifier'):
            bot.send_message(callback.message.chat.id, 'Успешно')
        else:
            bot.send_message(callback.message.chat.id, check_error)

    elif(callback.data == "btn4"):
        bot.send_message(callback.message.chat.id, 'Генерация случайных
вычислений')
        generate_value(user_id)
        bot.send_message(callback.message.chat.id, 'Успешно')

    elif(callback.data == "btn5"):
        img = open('bmstu.jpg', 'rb')
        bot.send_photo(callback.message.chat.id, img)
    else:
        bot.send_message(callback.chat.id, 'Нет такой команды. Введите
/help')

# Вычисления
@bot.message_handler(commands=['calculate'])
def start_calculate(message):
    bot.send_message(message.chat.id, 'Калькулятор бот')
    bot.send_message(message.chat.id, 'Напишите в чате вычисления')

    # Пользовательский идентификатор
    user_id = str(message.from_user.id)

    @bot.message_handler(content_types=["text"])
    def echo(message):
        value = smc(message.text)
        bot.send_message(message.chat.id, f'Решение: {value.result}')
        data = {
            user_id: [
                {"id": random.randint(0, 10000),
                 "value": str(message.text),
                 "result": str(value.result)}
            ]
        }
        merge_data(data, str(message.from_user.id))

# Просмотри история вычисления
@bot.message_handler(commands=['get_info'])
def start_get_info(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 == 'Error! There is no such identifier'):
    bot.send_message(message.chat.id, 'База данных отсутствует')
else:
    for j in range(len(data) - 1):
        id = data[j]['id']
        value = data[j]['value']
        result = data[j]['result']
        print_info = f'id: {id}\n{value} = {result}\n\n'
        bot.send_message(message.chat.id, print_info)

@bot.message_handler(commands=['photo'])
def url(message):
    img = open('bmstu.jpg', 'rb')
    bot.send_photo(message.chat.id, img)

@bot.message_handler(commands=['random'])
def url(message):
    bot.send_message(message.chat.id, 'Генерация случайных вычислений')
    # Пользовательский идентификатор
    user_id = str(message.from_user.id)

    generate_value(user_id)

    bot.send_message(message.chat.id, 'Успешно')

@bot.message_handler(commands=['clean'])
def url(message):
    bot.send_message(message.chat.id, 'Очистка история вычисления')

    # Пользовательский идентификатор
    user_id = str(message.from_user.id)

    check_error = delete_data_for_id_user(user_id)

    if (check_error != 'Error! There is no such identifier'):
        bot.send_message(message.chat.id, 'Успешно')
    else:
        bot.send_message(message.chat.id, check_error)

bot.polling(none_stop=True)

```

3.4.json_function.py

```
import json

def write_data(data, title='D:\Python\BKIT\calculate\data'):
    with open(f"{title}.json", "w", encoding="utf-8") as file:
        json.dump(data, file, indent=2, ensure_ascii=False)

def load_data_all(title="D:\Python\BKIT\calculate\data"):
    with open(f"{title}.json", "r") as file:
        data = json.load(file)
    return data

def merge_data(data_json, id_user='id_user',
title="D:\Python\BKIT\calculate\data"):
    # Если файл существует 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]:
                y = {
                    'id': info_data['id'],
                    'value': info_data['value'],
                    'result': info_data['result']
                }
                temp.append(y)
            write_data(data)
    # Если файл не существует
    except:
        write_data(data_json)

def load_data_for_id_user(id_user, title="D:\Python\BKIT\calculate\data"):
    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]:
                y = {
                    'id': info_data['id'],
                    'value': info_data['value'],
                    'result': info_data['result']
                }
                temp.append(y)
            return temp
    except:
        return 'Error! There is no such identifier'

def delete_data_for_id_user(id_user, title="D:\Python\BKIT\calculate\data"):
    try:
```

```

with open(f"{title}.json", encoding="utf-8") as file:
    data = json.load(file)
    new_data = {}
    for id_user_data in data:
        if (id_user != id_user_data):
            temp = data[id_user_data]
            new_data = {
                id_user_data: []
            }
            for j in temp:
                y = {
                    'id': j['id'],
                    'value': j['value'],
                    'result': j['result']
                }
                new_data[id_user_data].append(y)
            temp.append(new_data)
    write_data(new_data)
except:
    return 'Error! There is no such identifier'

```

3.5.work with calculate.py

```

import random

from calculate.json_function import write_data, load_data_all, merge_data,
load_data_for_id_user
from calculate.calculate_arifmetic import
the_simplest_mathematical_calculator as smc

def generate_value(id_user='id_user'):
    arifmetic = ['+', '-', '/', '*']

    af = arifmetic[random.randint(0, 3)]
    gen_id = random.randint(0, 1000000)
    v1 = random.randint(0, 1000)
    v2 = random.randint(0, 1000)
    class_calculate = smc(str(v1) + ' ' + str(af) + ' ' + str(v2))

    data = {
        str(id_user): [
            {
                "id": gen_id,
                "value": (str(v1) + ' ' + str(af) + ' ' + str(v2)),
                "result": class_calculate.result
            }
        ]
    }

    merge_data(data, id_user)

def get_info():
    try:

```

```
        data = load_data_all()
        return data
    except:
        return 'Файл отсутствует'

def get_info_with_id_user(id_user):
    try:
        data = load_data_for_id_user(id_user)
        return data
    except:
        return 'Файл отсутствует'
```

3.6.bmstu.jpg



3.7. Unittest

3.7.1. test_calculate.py

```
import unittest

from calculate.calculate_arifmetic import
the_simplest_mathematical_calculator as smc

class test_calculate(unittest.TestCase):

    # Проверка на работу
    def test_1(self):
        self.assertEqual(smc('10.0').result, 10.0)

    def test_2(self):
        self.assertEqual(smc('10 + 10').result, 20.0)

    def test_3(self):
        self.assertEqual(smc('2 + 3 * 2').result, 8.0)

    def test_4(self):
        self.assertEqual(smc('2 + 3 + 2').result, 7.0)

    def test_5(self):
        self.assertEqual(smc('2 + 3 - 2').result, 3.0)

    def test_6(self):
        self.assertEqual(smc('5 / 2 * 2').result, 5.0)

    def test_7(self):
        self.assertEqual(smc('100 - 10 + 100').result, 190.0)

    # Деление на 0
    def test_8(self):
        self.assertEqual(smc('1 / 0').result, 'inf')

    # Умножение на 0
    def test_9(self):
        self.assertEqual(smc('1 * 0').result, 0.0)
```

3.7.2. test_telebot.py

```
import unittest
import os.path

from calculate.work_with_calculate import generate_value,
get_info_with_id_user, write_data

data_json_two_users = {
    "369350478": [
        {
            "id": 61419,
            "value": 172,
            "result": 836.0
        },
    ],
}
```

```

    {
        "id": 1158,
        "value": "10 / 0",
        "result": "inf"
    },
    {
        "id": 5936,
        "value": "10 + 10",
        "result": "20.0"
    },
    {
        "id": 8329,
        "value": "10 + 10",
        "result": "20.0"
    },
    {
        "id": 5287,
        "value": "10 + 10",
        "result": "20.0"
    }
],
"198498415": [
    {
        "id": 8034,
        "value": "15 + 81",
        "result": "96.0"
    },
    {
        "id": 947,
        "value": "988 + 4894",
        "result": "5882.0"
    },
    {
        "id": 6363,
        "value": "8 + 2 1",
        "result": "10.0"
    }
]
}

```

```

class test_telebot(unittest.TestCase):

```

```

    # Проверка создания файла

```

```

    def test_create_file_json(self):

```

```

        message_from_user_id = 369350478

```

```

        generate_value(str(message_from_user_id))

```

```

        self.assertEqual(

```

```

            os.path.exists('D:\Python\BKIT\calculate\data.json'),

```

```

            True

```

```

        )

```

```

    # Проверка на получение информации по id

```

```

def test_get_info_with_id_user(self):
    write_data(data_json_two_users)

    message_from_user_id = 369350478

    check_info = get_info_with_id_user(str(message_from_user_id))
    print(check_info)
    self.assertEqual(
        check_info, [{ 'id': 61419, 'result': 836.0, 'value': 172},
                      { 'id': 1158, 'result': 'inf', 'value': '10 / 0'},
                      { 'id': 5936, 'result': '20.0', 'value': '10 + 10'},
                      { 'id': 8329, 'result': '20.0', 'value': '10 + 10'},
                      { 'id': 5287, 'result': '20.0', 'value': '10 + 10'},
                      { 'id': 5287, 'result': '20.0', 'value': '10 + 10'}]
    )

```

3.7.3. test_json.py

```

import unittest

from calculate.json_function import load_data_all, write_data, merge_data,
load_data_for_id_user, delete_data_for_id_user

data_json = {
    "id_user": [
        {
            "id": 12425,
            "value": '30 + 40',
            "result": '70'
        }
    ]
}

data_json_big = {
    "id_user": [
        {
            "id": 52478,
            "value": '10 + 10',
            "result": '20'
        },
        {
            "id": 5437,
            "value": '10 + 10',
            "result": '20'
        },
        {
            "id": 69823,
            "value": '10 + 10',
            "result": '20'
        },
        {
            "id": 24537,
            "value": '10 + 10',
            "result": '20'
        }
    ]
}

```



```

    },
    {
        "id": 786,
        "value": '10 + 10',
        "result": '20'
    }
]
}

data_json1 = {
    "id_user": [
        {
            "id": 324,
            "value": '50 + 50',
            "result": '100'
        }
    ]
}

data_json_with_id = {
    "369350471": [
        {
            "id": 12425,
            "value": '30 + 40',
            "result": '70'
        }
    ]
}

data_json_with_id_1 = {
    "369350471": [
        {
            "id": 78678,
            "value": '70 + 40',
            "result": '110'
        }
    ]
}

data_json_two_users = {
    "369350478": [
        {
            "id": 61419,
            "value": 172,
            "result": 836.0
        },
        {
            "id": 1158,
            "value": "10 / 0",
            "result": "inf"
        },
        {
            "id": 5936,
            "value": "10 + 10",
            "result": "20.0"
        }
    ]
}

```

```

    },
    {
        "id": 8329,
        "value": "10 + 10",
        "result": "20.0"
    },
    {
        "id": 5287,
        "value": "10 + 10",
        "result": "20.0"
    }
],
"198498415": [
    {
        "id": 8034,
        "value": "15 + 81",
        "result": "96.0"
    },
    {
        "id": 947,
        "value": "988 + 4894",
        "result": "5882.0"
    },
    {
        "id": 6363,
        "value": "8 + 2 1",
        "result": "10.0"
    }
]
}

```

```

class test_json(unittest.TestCase):

    # Проверка на присутствия файла
    def test_write_and_read_file(self):
        # Создаем файл с данным
        write_data(data_json)

        # Проверяем на наличие и сходимости
        self.assertEqual(
            load_data_all(),
            {'id_user': [{'id': 12425, 'result': '70', 'value': '30 + 40'}]}
        )

    # Проверка на добавлении json дата
    def test_append_json_in_json(self):
        # Создаем файл с данным
        write_data(data_json)

        # Изменяем файл - добавление новые данных
        merge_data(data_json1)

        # Проверяем на наличие и сходимости
        self.assertEqual(
            load_data_all(),

```

```

        {'id_user': [
            {'id': 12425, 'result': '70', 'value': '30 + 40'},
            {'id': 324, 'result': '100', 'value': '50 + 50'}
        ]})

# Проверка на добавлении json data с идентификатором пользователя
def test_and_read_file_with_id(self):
    # Создаем файл с данным
    write_data(data_json_with_id)

    # Проверяем на наличие и сходимости
    self.assertEqual(
        load_data_all(),
        {'369350471': [{'id': 12425, 'result': '70', 'value': '30 +
40'}]})

    )

# Проверка на добавлении json data с идентификатором пользователя
def test_append_json_in_json_with_id(self):
    # Создаем файл с данным
    write_data(data_json_with_id)

    # Изменяем файл - добавление новые данных
    merge_data(data_json_with_id_1, str(369350471))

    # Проверяем на наличие и сходимости
    self.assertEqual(
        load_data_all(),
        {'369350471': [
            {'id': 12425, 'result': '70', 'value': '30 + 40'},
            {'id': 78678, 'result': '110', 'value': '70 + 40'}
        ]})

def test_search_id_user_and_get_info(self):
    # Создаем файл с данным
    write_data(data_json_two_users)

    # Проверяем на наличие и сходимости
    self.assertEqual(
        load_data_for_id_user('198498415'),
        [{'id': 8034, 'result': '96.0', 'value': '15 + 81'},
         {'id': 947, 'result': '5882.0', 'value': '988 + 4894'},
         {'id': 6363, 'result': '10.0', 'value': '8 + 2 1'},
         {'id': 6363, 'result': '10.0', 'value': '8 + 2 1'}])

def test_delete_data_of_id_user(self):
    # Создаем файл с данным
    write_data(data_json_two_users)

    # Удаляем данные по id пользователя
    delete_data_for_id_user('369350478')

    # Проверяем на наличие и сходимости
    self.assertEqual(

```

```
load_data_for_id_user('198498415'),
[{'id': 8034, 'result': '96.0', 'value': '15 + 81'},
 {'id': 947, 'result': '5882.0', 'value': '988 + 4894'},
 {'id': 6363, 'result': '10.0', 'value': '8 + 2 1'},
 {'id': 6363, 'result': '10.0', 'value': '8 + 2 1'}]]
```

3.7.4. test_filed.py (Unitest к 4 лабе)

```
# Подключаем библиотеку unittest для тестирования
import unittest

'''
assertEqual(self, first, second)
first - передаваемое значение
second - полученное значение (в тело функции должен быть return, если вы там
не оставили, тогда прописать здесь как None)
если передаваемое значение совпадает с полученным значением, то тест пройден
успешно
'''

from function.filed import field, goods

class test_filed(unittest.TestCase):

    # Проверка вывода с 1 аргумента
    def test_pass_one_argv(self):
        self.assertEqual(
            field(goods, 'title'),
            [
                {'title': 'Ковер'},
                {'title': 'Диван для отдыха'}
            ]
        )

    # Проверка вывода с 2 аргумента
    def test_pass_two_argv(self):
        self.assertEqual(
            field(goods, 'title color'),
            [
                {'color': 'green', 'title': 'Ковер'},
                {'color': 'black', 'title': 'Диван для отдыха'}
            ]
        )

    # Проверка вывода с 3 аргумента
    def test_pass_three_argv(self):
        self.assertEqual(
            field(goods, 'title color price'),
            [
                {'color': 'green', 'price': 2000, 'title': 'Ковер'},
                {'color': 'black', 'price': 5300, 'title': 'Диван для
отдыха'}
            ]
        )
```

```
]
)
```

3.7.5. test_unique.py (Unittest к 4 лабе)

```
# Подключаем библиотеку unittest для тестирования
import unittest

'''
assertEqual(self, first, second)
first - передаваемое значение
second - полученное значение (в тело функции должен быть return, если вы там
не оставили, тогда прописать здесь как None)
если передаваемое значение совпадает с полученным значением, то тест пройден
успешно
'''

from function.unique import Unique

class test_unique(unittest.TestCase):
    # Проверка на чисел
    def test_value(self):
        # Дан список с числами
        data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
        # Получаем уникальные значения и сохраним его в переменной
        arr_unique = Unique(data).arr
        # Проверяем
        self.assertEqual(
            arr_unique,
            [1, 2]
        )

    # Проверка на буквы
    def test_letters(self):
        # Дан список с числами
        data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        # Получаем уникальные значения и сохраним его в переменной
        arr_unique = Unique(data).arr
        # Проверяем
        self.assertEqual(
            arr_unique,
            ['a', 'A', 'b', 'B']
        )

    # Проверка на буквы без чувствительного регистра
    def test_letters_ignore_case(self):
        # Дан список с числами
        data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        # Получаем уникальные значения и сохраним его в переменной
        arr_unique = Unique(data, ignore_case = True).arr
        # Проверяем
        self.assertEqual(
            arr_unique,
            ['a', 'b']
        )
```

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

3.8.Behave

3.8.1. check_unique.feature

Feature: Checking the output of an argument from the goods dictionaries

Проверка вывода с 1 аргумента

Scenario Outline: Checking the output with 1 argument

Given I have a dictionary goods

When We enter **<arguments>** to get the desired values

Then Output to the **<check_result>**

Examples:

arguments	check_result
title	[{'title': 'Ковер'}, {'title': 'Диван для отдыха'}]
color	[{'color': 'green'}, {'color': 'black'}]
price	[{'price': 2000}, {'price': 5300}]

Проверка вывода с 2 аргумента

Scenario Outline: Checking the output with 2 argument

Given I have a dictionary goods

When We enter **<arguments>** to get the desired values

Then Output to the **<check_result>**

Examples:

arguments	check_result
title color	[{'title': 'Ковер', 'color': 'green'}, {'title': 'Диван для отдыха', 'color': 'black'}]
color price	[{'color': 'green', 'price': 2000}, {'color': 'black', 'price': 5300}]

Проверка вывода с 3 аргумента

Scenario Outline: Checking the output with 3 argument

Given I have a dictionary goods

When We enter **<arguments>** to get the desired values

Then Output to the **<check_result>**

Examples:

arguments	check_result
title color price	[{'color': 'green', 'price': 2000, 'title': 'Ковер'}, {'color': 'black', 'price': 5300, 'title': 'Диван для отдыха'}]

3.8.2. check filed.feature

Feature: Calculating and getting unique values

Уникальные значения числового типа

If <CASE> is 1 then is True

Scenario Outline: We get unique values from the list of the contained number

Given I have a class of unique values

And Getting the list: <list>

When Finding unique values, case: <CASE>

Then Output unique values: <unique>

Examples:

list	unique	CASE
[1, 1, 1, 1, 1, 2, 2, 2, 2, 2]	[1, 2]	0
[1, 3, 1, 1, 1, 3, 2, 2, 2, 2]	[1, 3, 2]	0

Уникальные значения символьного типа

Scenario Outline: We get unique values from the list of the contained char

Given I have a class of unique values

And Getting the list: <list>

When Finding unique values, case: <CASE>

Then Output unique values: <unique>

Examples:

list	unique
CASE	
['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']	['a', 'A', 'b', 'B']
0	
['a', 'C', 'b', 'B', 'c', 'A', 'b', 'B']	['a', 'C', 'b', 'B', 'c', 'A']
0	

Уникальные значения символьного типа без чувствительного регистра

Scenario Outline: We get unique values from the list of the contained char
ignore_case

Given I have a class of unique values

And Getting the list: <list>

When Finding unique values, case: <CASE>

Then Output unique values: <unique>

Examples:

list	unique	CASE
['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']	['a', 'b']	1
['a', 'C', 'b', 'B', 'c', 'A', 'b', 'B']	['a', 'c', 'b']	1

Уникальные значения смешанного типа

Scenario Outline: We get unique values from the list of the contained all type

Given I have a class of unique values

And Getting the list: <list>

When Finding unique values, case: <CASE>

Then Output unique values: <unique>

Examples:

	list	unique
CASE		
0	['a', 'A', 'b', 'B', '1', '1', '2', '2']	['a', 'A', 'b', 'B', '1', '2']
1	['a', 'A', 'b', 'B', '1', '1', '2', '2']	['a', 'b', '1', '2']

3.9.Steps (for Behave)

3.9.1. filed.py

```
from behave import Given, When, Then
from function.filed import field, goods
import ast

@Given('I have a dictionary goods')
def step_impl(context):
    context.data_dictionary = goods
    test = context.data_dictionary
    print(test)

@When("We enter {arguments} to get the desired values")
def given_increment(context, arguments):
    context.results = field(context.data_dictionary, arguments)

@Then("Output to the {check_result}")
def then_results(context, check_result):
    assert context.results == ast.literal_eval(check_result)
```

3.9.2. unique.py

```
from behave import Given, When, Then
from function.unique import Unique
import ast

@Given('I have a class of unique values')
def step_impl(context):
    pass

@Given("Getting the list: {LIST}")
def given_increment(context, LIST):
    context.LIST = list(ast.literal_eval(LIST))
    print(f'Список: {LIST}')
```



```

@When("Finding unique values, case: {CASE}")
def given_increment(context, CASE):
    check = bool(int(CASE))
    if (check == True):
        unique_list = Unique(context.LIST, ignore_case=check)
    else:
        unique_list = Unique(context.LIST)

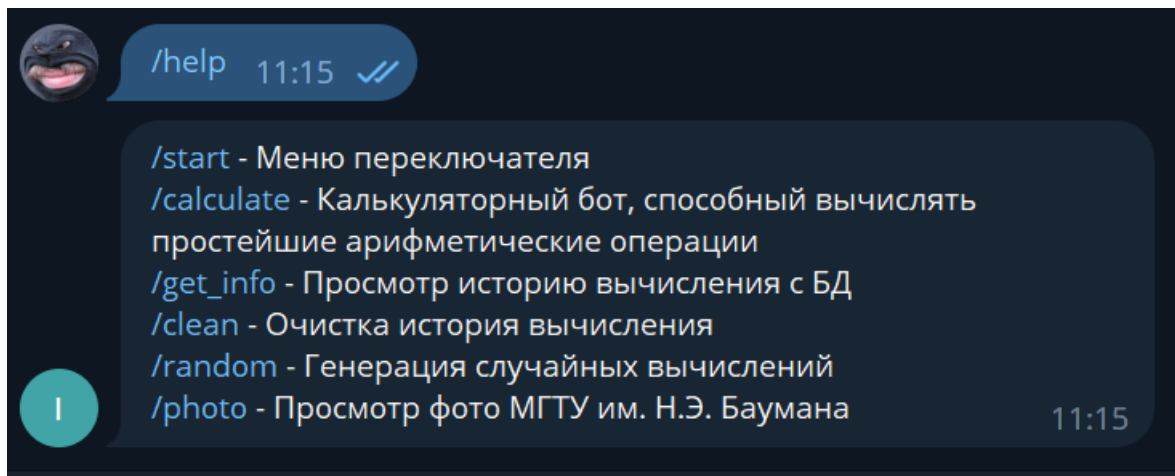
    context.results = unique_list
    # print(f'Уникальные значения: {unique_list}')

@Then("Output unique values: {UNIQUE}")
def then_results(context, UNIQUE):
    assert context.results.arr == ast.literal_eval(UNIQUE)
    print(f'Уникальные значения: {context.results.arr}')

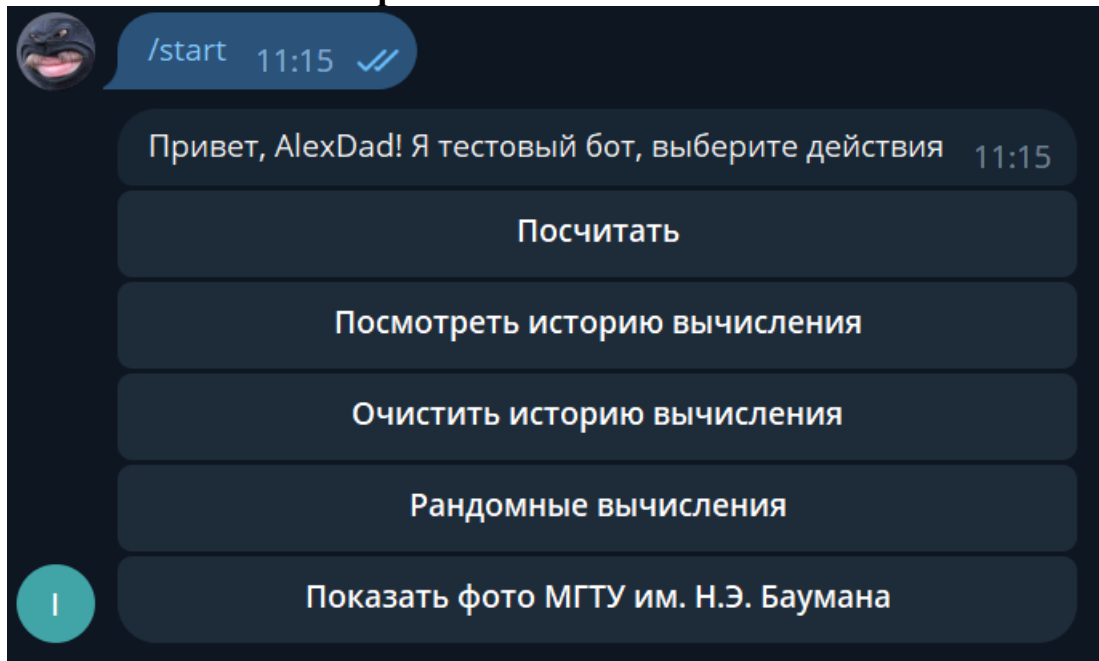
```

4. Результаты работы программы в Telegram

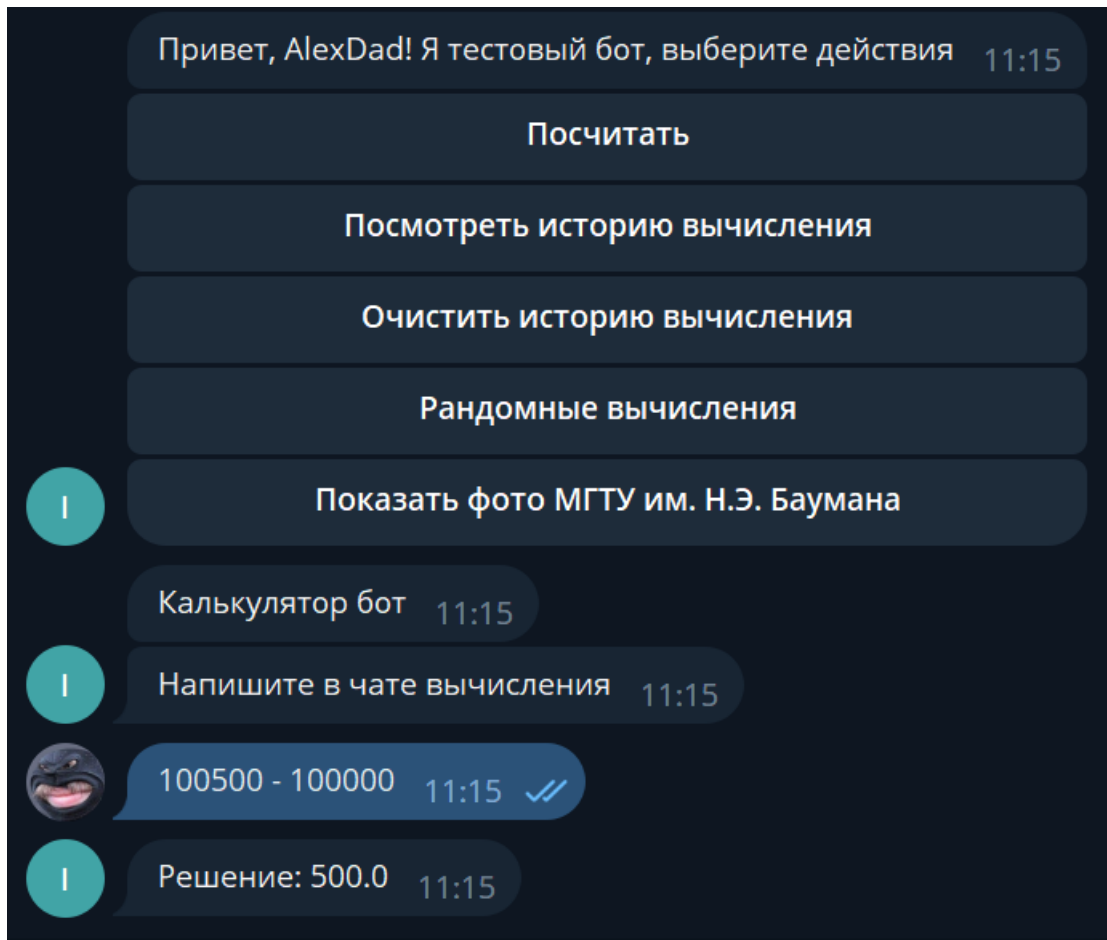
4.1. Получение справочную информацию



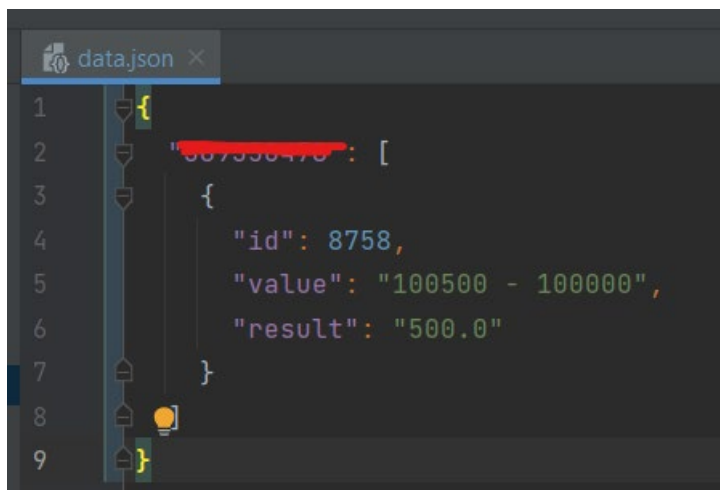
4.2. Основное меню переключателя



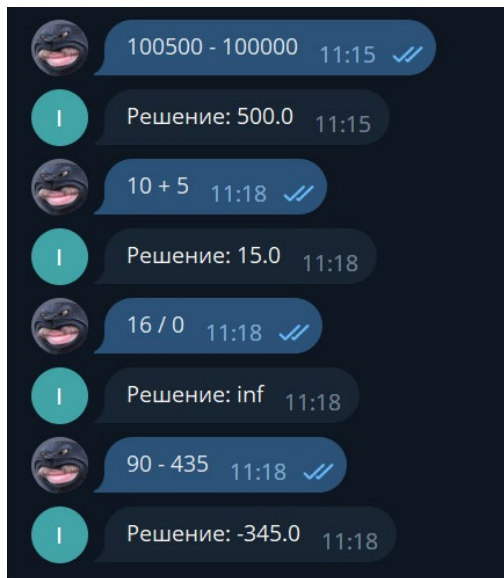
4.3. Простейший калькулятор (при нажатии на кнопку «Посчитать»)



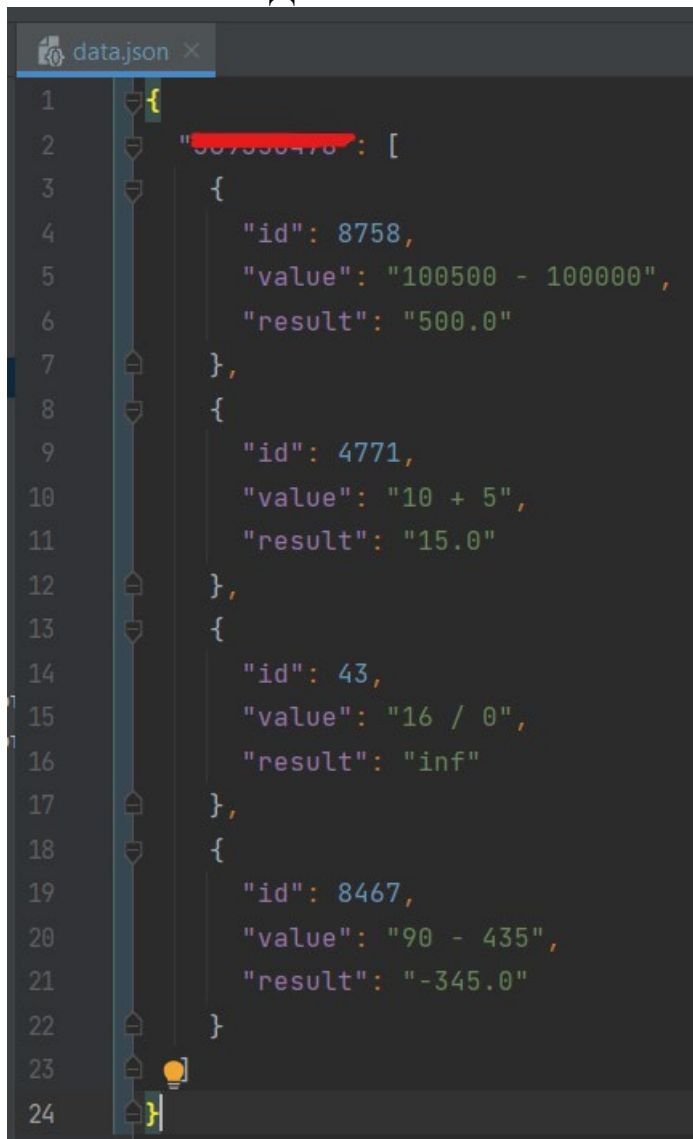
4.4. Данные хранятся в БД в формате JSON (Строчка кода 2 хранит id пользователя)



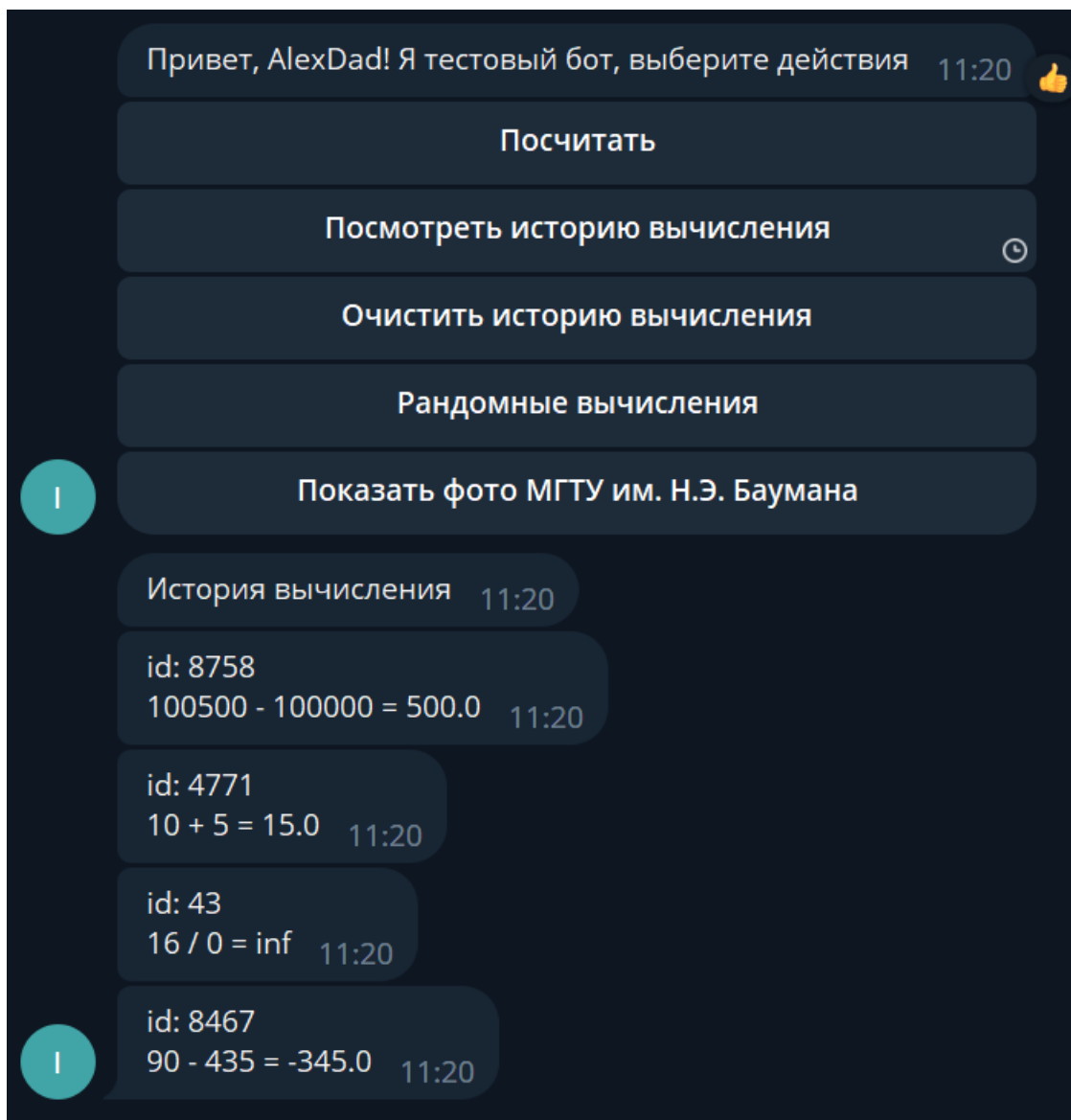
4.5. После несколько вычислений



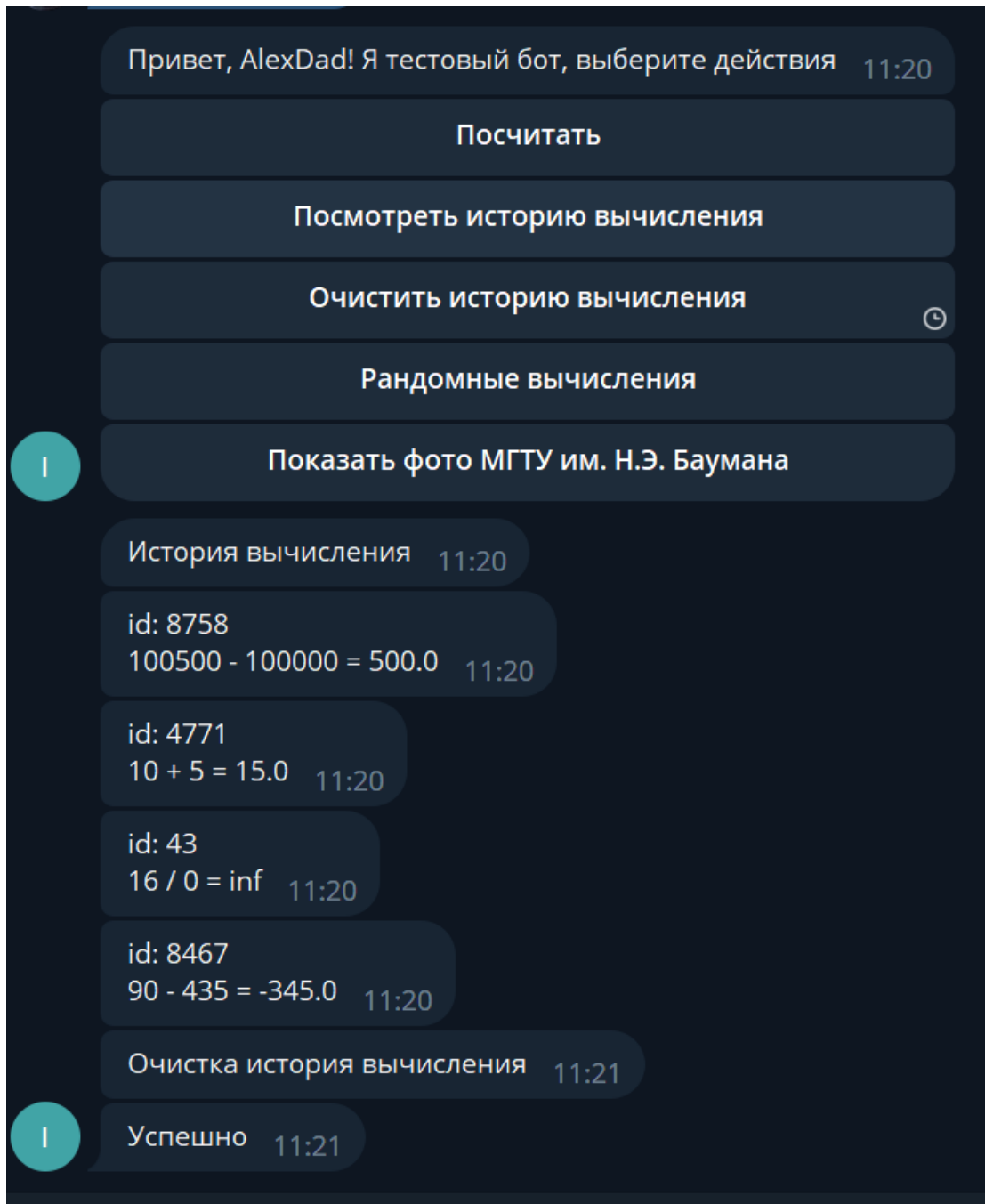
Обновленная БД



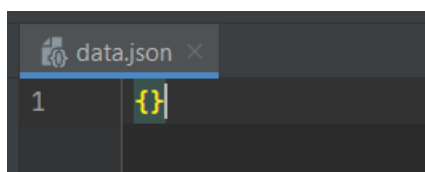
4.6. Чтение и просмотр БД в Телеграме (после нажатии на кнопку Посмотреть историю вычисления)



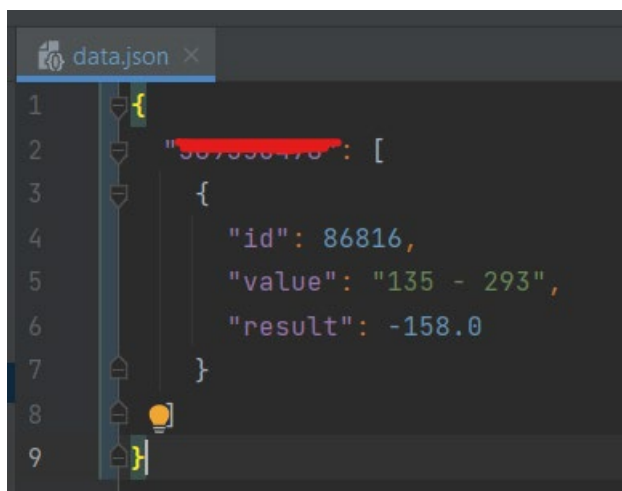
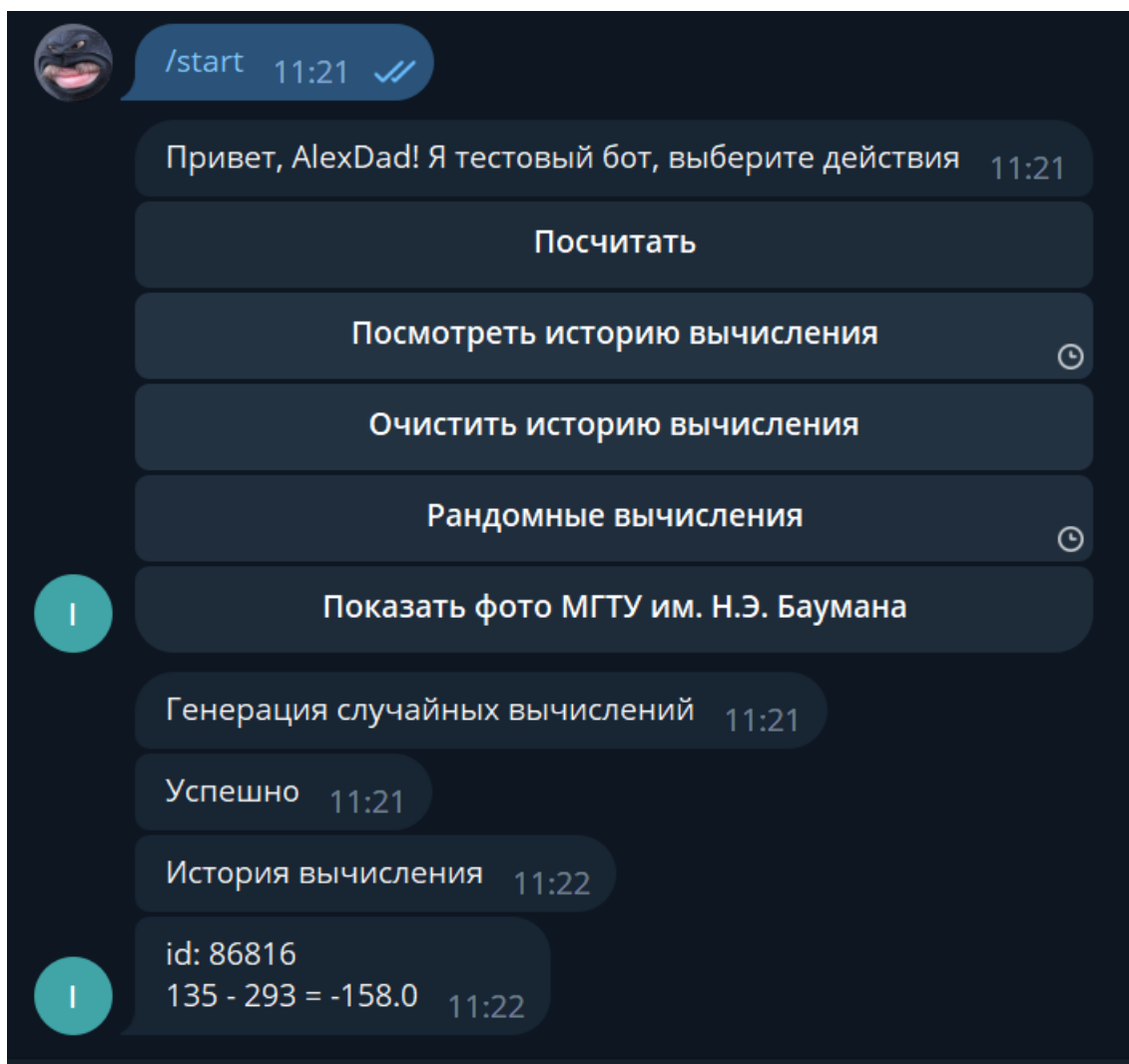
4.7. Очистка история вычисления БД в Телеграме



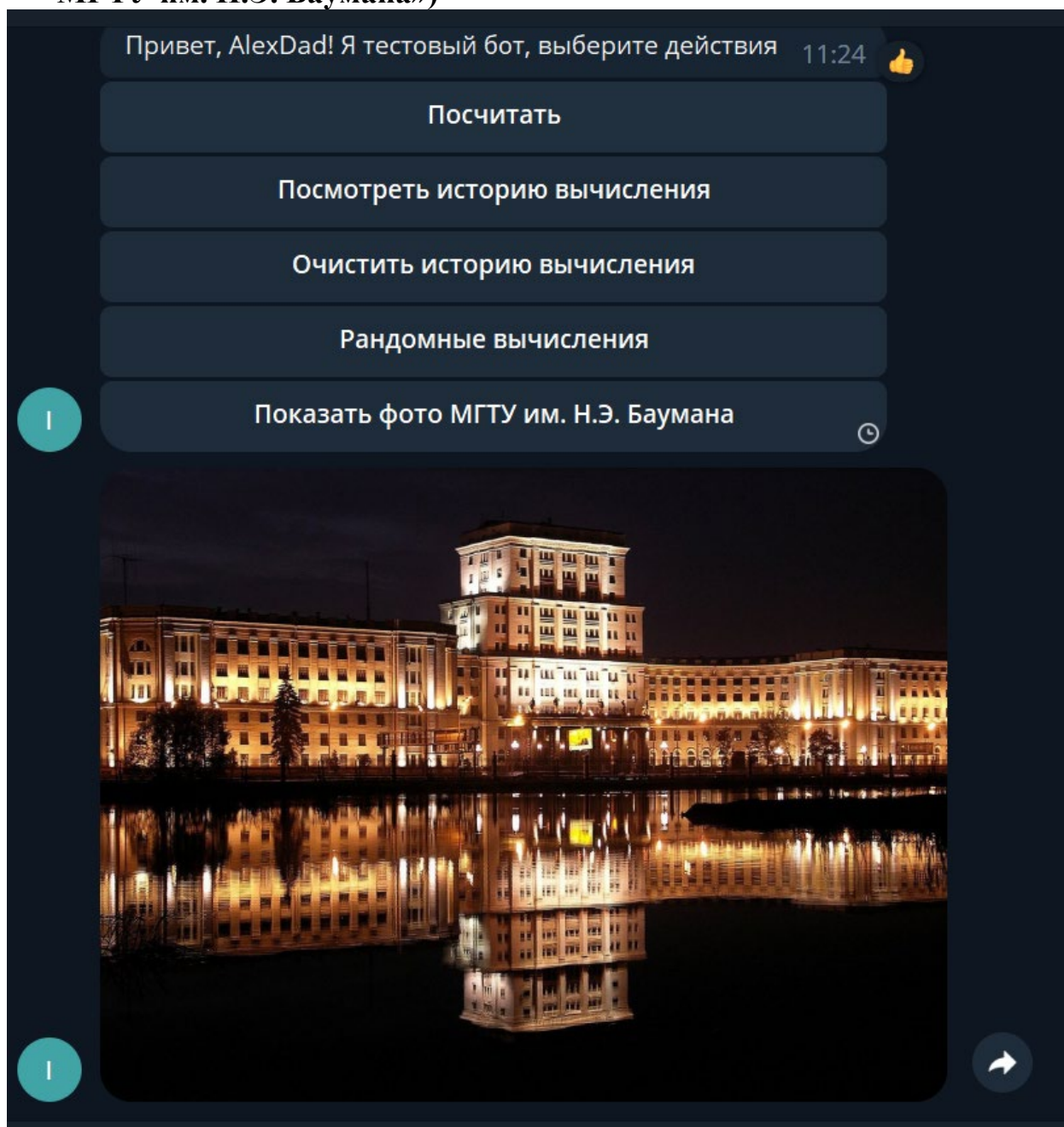
Обновленная БД



4.8. Генерация случайных вычислений и просмотр вычислений (после нажатии на кнопку «Показать фото МГТУ им. Н.Э. Баумана»)



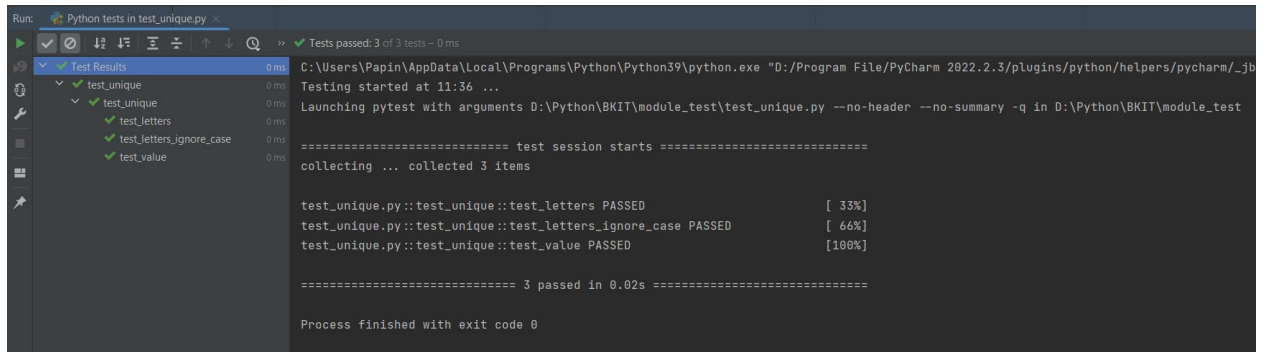
4.9. Просмотр фотографии (после нажатии на кнопку «Показать фото МГТУ им. Н.Э. Баумана»)



5. Модульное тестирование в IDE JetBrains PyCharm

5.1. unittest

5.1.1. test_unique.py



The screenshot shows the PyCharm test runner interface for the file `test_unique.py`. The left sidebar displays the 'Test Results' tree with a green checkmark next to `test_unique`. The main panel shows the test execution output, indicating that 3 tests passed in 0.02 seconds. The tests listed are `test_letters`, `test_letters_ignore_case`, and `test_value`.

```
Run: Python tests in test_unique.py
Tests passed: 3 of 3 tests - 0 ms
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/helpers/pycharm/_jb_pytest_runner.py"
Testing started at 11:36 ...
Launching pytest with arguments D:\Python\BKIT\module_test\test_unique.py --no-header --no-summary -q in D:\Python\BKIT\module_test

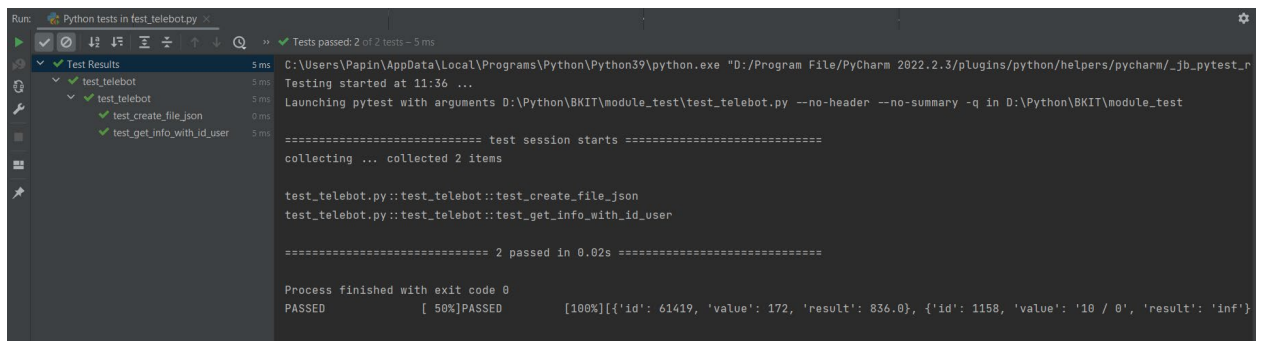
===== test session starts =====
collecting ... collected 3 items

test_unique.py::test_unique::test_letters PASSED [ 33%]
test_unique.py::test_unique::test_letters_ignore_case PASSED [ 66%]
test_unique.py::test_unique::test_value PASSED [100%]

===== 3 passed in 0.02s =====

Process finished with exit code 0
```

5.1.2. test_telebot.py



The screenshot shows the PyCharm test runner interface for the file `test_telebot.py`. The left sidebar displays the 'Test Results' tree with a green checkmark next to `test_telebot`. The main panel shows the test execution output, indicating that 2 tests passed in 0.02 seconds. The tests listed are `test_create_file_json` and `test_get_info_with_id_user`.

```
Run: Python tests in test_telebot.py
Tests passed: 2 of 2 tests - 5 ms
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/helpers/pycharm/_jb_pytest_runner.py"
Testing started at 11:36 ...
Launching pytest with arguments D:\Python\BKIT\module_test\test_telebot.py --no-header --no-summary -q in D:\Python\BKIT\module_test

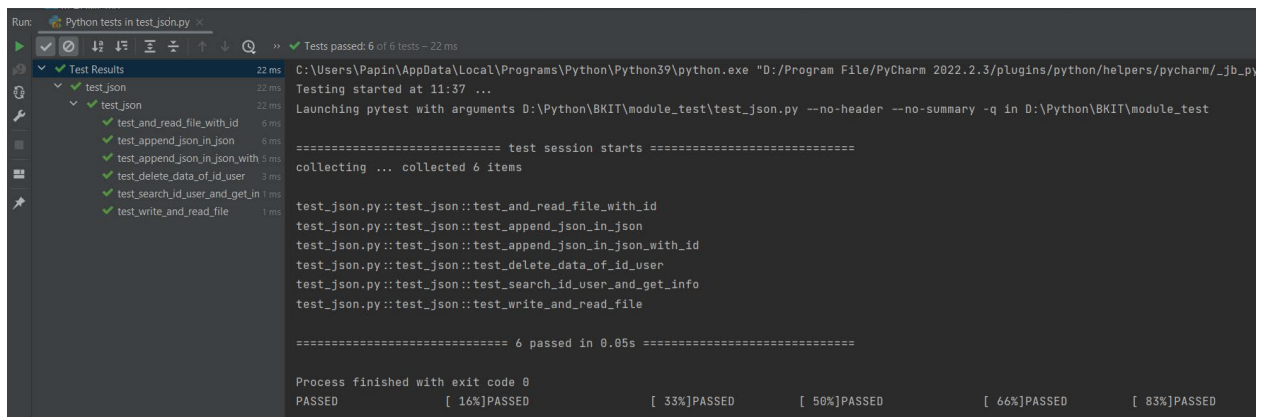
===== test session starts =====
collecting ... collected 2 items

test_telebot.py::test_telebot::test_create_file_json
test_telebot.py::test_telebot::test_get_info_with_id_user

===== 2 passed in 0.02s =====

Process finished with exit code 0
PASSED [ 50%]PASSED [100%][{'id': 61419, 'value': 172, 'result': 836.0}, {'id': 1158, 'value': '10 / 0', 'result': 'inf'}]
```

5.1.3. test_json.py



The screenshot shows the PyCharm test runner interface for the file `test_json.py`. The left sidebar displays the 'Test Results' tree with a green checkmark next to `test_json`. The main panel shows the test execution output, indicating that 6 tests passed in 0.05 seconds. The tests listed are `test_and_read_file_with_id`, `test_append_json_in_json`, `test_append_json_in_json_with_id`, `test_delete_data_of_id_user`, `test_search_id_user_and_get_info`, and `test_write_and_read_file`.

```
Run: Python tests in test_json.py
Tests passed: 6 of 6 tests - 22 ms
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/helpers/pycharm/_jb_pytest_runner.py"
Testing started at 11:37 ...
Launching pytest with arguments D:\Python\BKIT\module_test\test_json.py --no-header --no-summary -q in D:\Python\BKIT\module_test

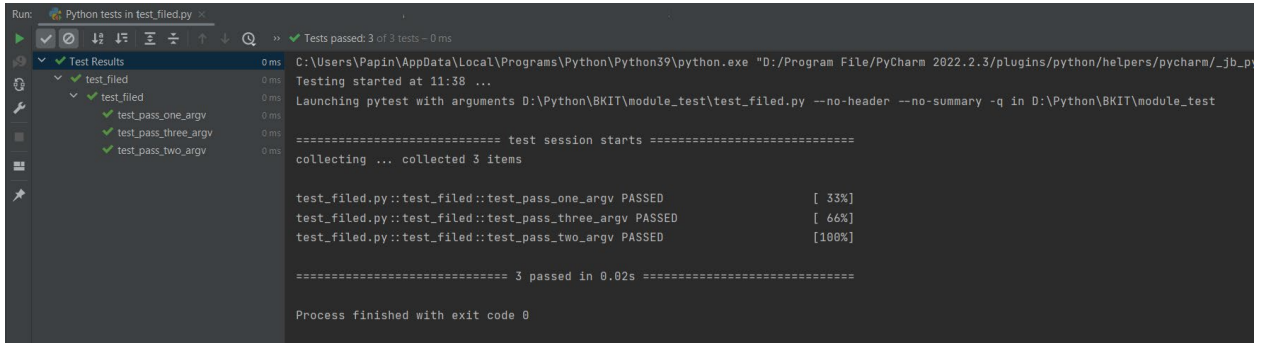
===== test session starts =====
collecting ... collected 6 items

test_json.py::test_json::test_and_read_file_with_id
test_json.py::test_json::test_append_json_in_json
test_json.py::test_json::test_append_json_in_json_with_id
test_json.py::test_json::test_delete_data_of_id_user
test_json.py::test_json::test_search_id_user_and_get_info
test_json.py::test_json::test_write_and_read_file

===== 6 passed in 0.05s =====

Process finished with exit code 0
PASSED [ 16%]PASSED [ 33%]PASSED [ 50%]PASSED [ 66%]PASSED [ 83%]PASSED [100%]
```

5.1.4. test_filed.py



The screenshot shows the PyCharm Run window for the file `test_filed.py`. The left sidebar displays the Test Results tree with a green checkmark next to `test_filed`. The main console area shows the output of the test session, including the pytest command line arguments and the results of three tests: `test_pass_one_argv`, `test_pass_three_argv`, and `test_pass_two_argv`, all of which passed.

```
Run: Python tests in test_filed.py x
Tests passed: 3 of 3 tests - 0 ms
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/helpers/pycharm/_jb_pytest.py"
Testing started at 11:38 ...
Launching pytest with arguments D:\Python\BKIT\module_test\test_filed.py --no-header --no-summary -q in D:\Python\BKIT\module_test

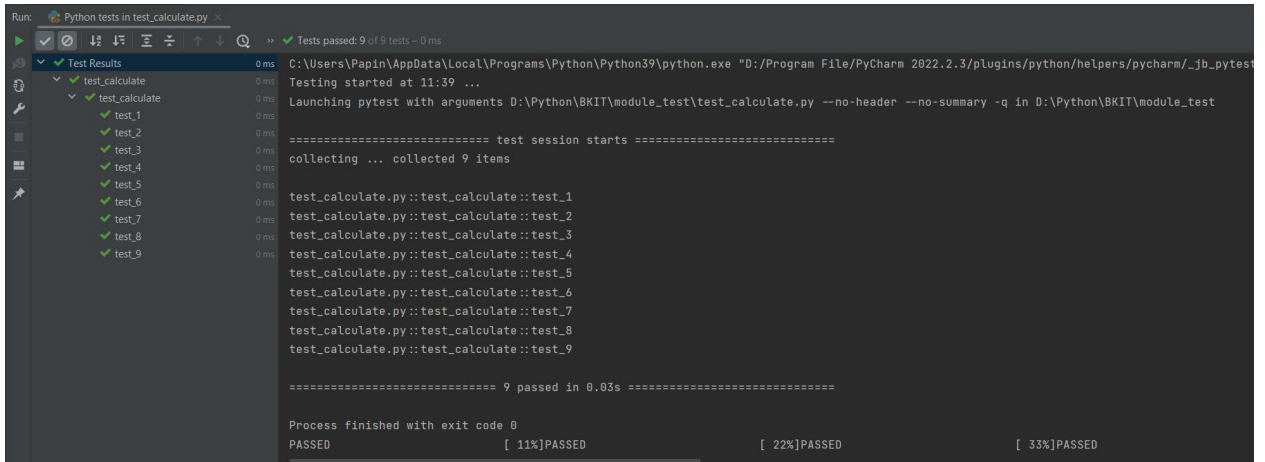
===== test session starts =====
collecting ... collected 3 items

test_filed.py::test_filed::test_pass_one_argv PASSED [ 33%]
test_filed.py::test_filed::test_pass_three_argv PASSED [ 66%]
test_filed.py::test_filed::test_pass_two_argv PASSED [100%]

===== 3 passed in 0.02s =====

Process finished with exit code 0
```

5.1.5. test_calculate.py



The screenshot shows the PyCharm Run window for the file `test_calculate.py`. The left sidebar displays the Test Results tree with a green checkmark next to `test_calculate`. The main console area shows the output of the test session, including the pytest command line arguments and the results of nine tests: `test_1` through `test_9`, all of which passed.

```
Run: Python tests in test_calculate.py x
Tests passed: 9 of 9 tests - 0 ms
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/helpers/pycharm/_jb_pytest.py"
Testing started at 11:39 ...
Launching pytest with arguments D:\Python\BKIT\module_test\test_calculate.py --no-header --no-summary -q in D:\Python\BKIT\module_test

===== test session starts =====
collecting ... collected 9 items

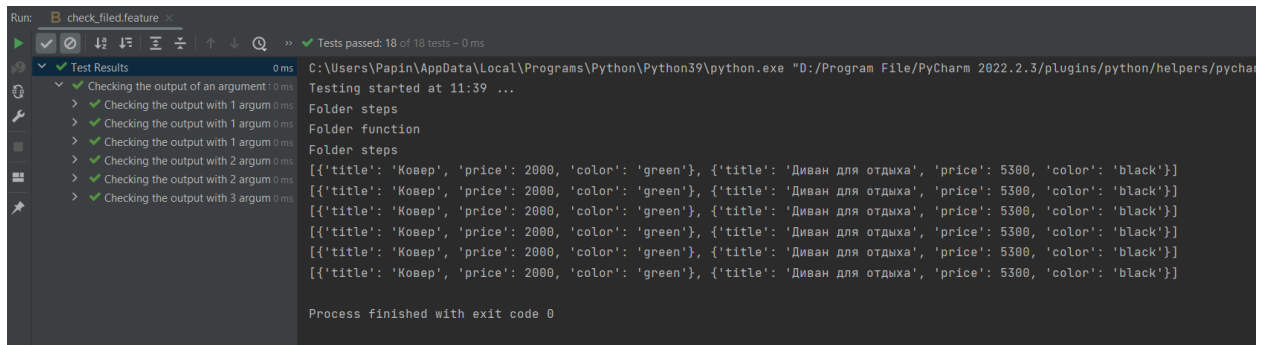
test_calculate.py::test_calculate::test_1
test_calculate.py::test_calculate::test_2
test_calculate.py::test_calculate::test_3
test_calculate.py::test_calculate::test_4
test_calculate.py::test_calculate::test_5
test_calculate.py::test_calculate::test_6
test_calculate.py::test_calculate::test_7
test_calculate.py::test_calculate::test_8
test_calculate.py::test_calculate::test_9

===== 9 passed in 0.03s =====

Process finished with exit code 0
PASSED [ 11%]PASSED [ 22%]PASSED [ 33%]PASSED
```

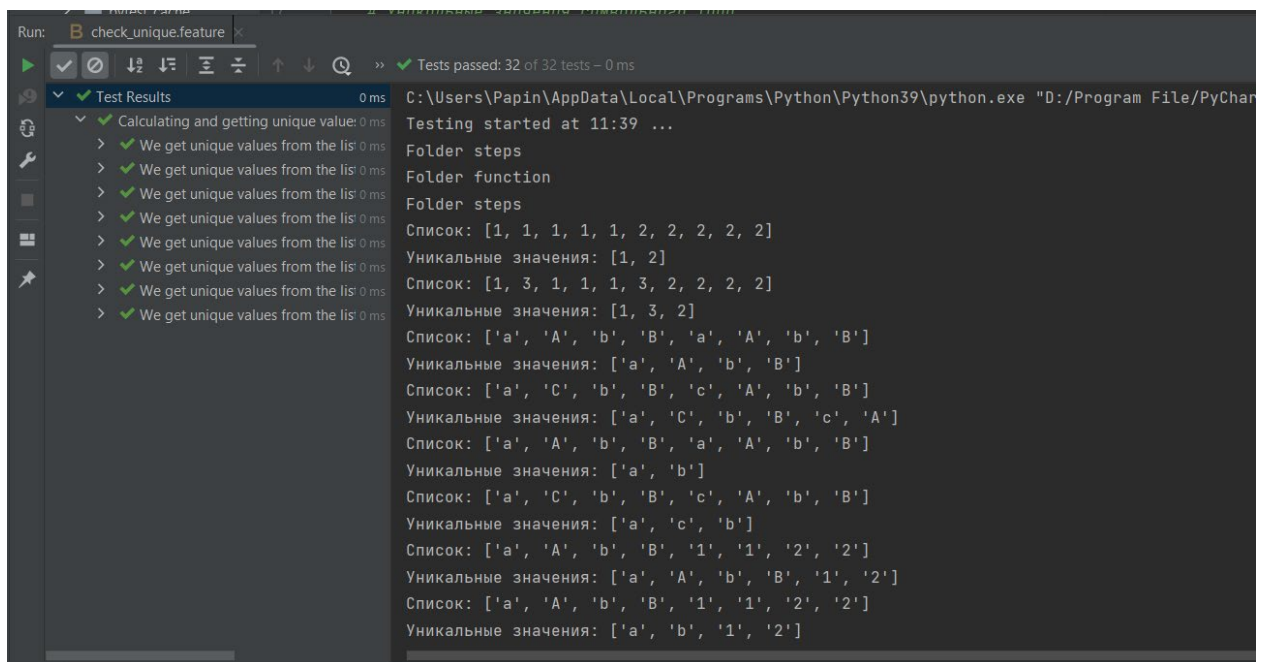
5.2.Behave

5.2.1. check_unique.feature



```
Run: B check_unique.feature x
Tests passed: 18 of 18 tests - 0 ms
Testing started at 11:39 ...
Folder steps
Folder function
Folder steps
[{'title': 'Ковсер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
[{'title': 'Ковсер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
[{'title': 'Ковсер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
[{'title': 'Ковсер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
[{'title': 'Ковсер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
[{'title': 'Ковсер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
Process finished with exit code 0
```

5.2.2. check_unique.feature



```
Run: B check_unique.feature x
Tests passed: 32 of 32 tests - 0 ms
Testing started at 11:39 ...
Folder steps
Folder function
Folder steps
Список: [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
Уникальные значения: [1, 2]
Список: [1, 3, 1, 1, 1, 1, 3, 2, 2, 2]
Уникальные значения: [1, 3, 2]
Список: ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
Уникальные значения: ['a', 'A', 'b', 'B']
Список: ['a', 'C', 'b', 'B', 'c', 'A', 'b', 'B']
Уникальные значения: ['a', 'C', 'b', 'B', 'c', 'A']
Список: ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
Уникальные значения: ['a', 'b']
Список: ['a', 'C', 'b', 'B', 'c', 'A', 'b', 'B']
Уникальные значения: ['a', 'c', 'b']
Список: ['a', 'A', 'b', 'B', '1', '1', '2', '2']
Уникальные значения: ['a', 'A', 'b', 'B', '1', '2']
Список: ['a', 'A', 'b', 'B', '1', '1', '2', '2']
Уникальные значения: ['a', 'b', '1', '2']
```