

Защищено:  
Папин А.В..

Демонстрация:  
Папин А.В..

"\_\_" \_\_\_\_\_ 2022 г.

"\_\_" \_\_\_\_\_ 2022 г.

## **Отчет по лабораторной работе №6 по курсу базовые компоненты интернет-технологий (БКИТ)**

### **Домашнее задание**

26

(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-54Б  
Папин Алексей

Гапанюк Ю.Е.

\_\_\_\_\_  
(подпись)

"\_\_" \_\_\_\_\_ 2022 г.

## СОДЕРЖАНИЕ ОТЧЕТА

1. Цель лабораторной работы.....	2
2. Описание задания.....	2
3. Листинг программы: .....	3
3.1. config.py .....	3
3.2. calculate_arifmetic.py .....	3
4.1. calculate_bot.py.....	4
4.2. json_function.py .....	7
4.3. work_with_calculate.py .....	7
4.4. bmstu.jpg .....	8
4.5. Unittest .....	9
4.5.1.test_calculate.py .....	9
4.5.2.test_telebot.py .....	9
4.5.3.test_json.py.....	10
4.5.4.test_filed.py (Unittest к 4 лабе).....	12
4.5.5.test_unique.py (Unittest к 4 лабе) .....	13
4.6. Behave .....	14
4.6.1.check_unique.feature.....	14
4.6.2.check_filed.feature .....	15
4.7. Steps (for Behave).....	16
4.7.1.filed.py.....	16
4.7.2.unique.py .....	17
5. Результаты работы программы в Telegram.....	18
5.1. Получение справочную информацию .....	18
5.2. Основное меню переключателя .....	18
5.3. Простейший калькулятор (при нажатии на кнопку «Посчитать») .....	18
5.4. Данные хранятся в БД в формате JSON.....	19
5.5. После несколько вычислений .....	19
5.6. Обновленная БД .....	20
5.7. Чтение и просмотр БД в Телеграме (после нажатии на кнопку Посмотреть историю вычисления) .....	21
5.8. Просмотр фотографии (после нажатии на кнопку «Показать фото МГТУ им. Н.Э. Баумана») .....	22
6. Модульное тестирование.....	23
6.1. Unittest .....	23
6.1.1.test_unique.py .....	23
6.1.2.test_telebot.py .....	23
6.1.3.test_json.py.....	23
6.1.4.test_filed.py .....	24
6.1.5.test_calculate.py .....	24
6.2. Behave .....	25
6.2.1.check_unique.feature.....	25
6.2.2.check_unique.feature.....	25

## **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

```
4.
# Преобразование строкового типа в list
def delete_space_into_list(string):
    new_str = []
    str_value = ''
    for i in string:
        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(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 = prioritет(count_list)

    return count_list

# Поддержка функции по расстановку приоритета операции
def prioritет(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(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:
                return result

def calculate(value):
    new_list = delete_space_into_list(value)
    list_en = enumeration_sign(new_list)

    for sgin in list_en:
        arifmetic(sgin, new_list)

    print(float(new_list[0]))
    return float(new_list[0])

```

#### 4.1.calculate\_bot.py

```

import config
import telebot
from telebot import types
import random

from calculate_arifmetic import calculate
from calculate.work_with_calculate import get_info
from json_function import merge_data

# 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
/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="Показать фото МГТУ им. Н.Э.
Баумана", callback_data='btn3')
    markup.add(btn1, btn2, btn3)
    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):
    if (callback.data == "btn1"):
        bot.send_message(callback.message.chat.id, 'Калькулятор бот')
        bot.send_message(callback.message.chat.id, 'Напишите в чате
вычисления')

    @bot.message_handler(content_types=["text"])
    def echo(message):
        # Пользовательский идентификатор
        user_id = str(message.from_user.id)

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

    elif(callback.data == "btn2"):
        bot.send_message(callback.message.chat.id, 'История вычисления')
        data = get_info()
        for i in data:
            for j in data[i]:
                id = j['id']
                value = j['value']

```

```

        result = 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"):
        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 = calculate(message.text)
        bot.send_message(message.chat.id, f'Решение: {value}')
        data = {
            user_id: [
                {"id": random.randint(0, 10000),
                 "value": str(message.text),
                 "result": str(value)}
            ]
        }
        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')

    data = get_info()
    if (data == 'Файл отсутствует'):
        bot.send_message(message.chat.id, 'База данных отсутствует')
    else:
        for i in data:
            for j in data[i]:
                id = j['id']
                value = j['value']
                result = 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.polling(none_stop=True)

```

## 4.2.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(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)

```

## 4.3.work with calculate.py

```

import random

from calculate.json_function import write_data, load_data, merge_data
from calculate.calculate_arifmetic import calculate

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)
result = calculate(str(v1) + ' ' + str(v2))

data = {
    str(id_user): [
        {
            "id": gen_id,
            "value": v1,
            "result": result
        }
    ]
}

merge_data(data, id_user)

def get_info():
    try:
        data = load_data()
        return data
    except:
        return 'Файл отсутствует'

```

#### 4.4.bmstu.jpg



## 4.5. Unittest

### 4.5.1. test\_calculate.py

```
import unittest

from calculate.calculate_arifmetic import calculate

class test_calculate(unittest.TestCase):

    # Проверка на работу
    def test_1(self):
        self.assertEqual(calculate('10'), 10.0)

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

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

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

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

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

    def test_7(self):
        self.assertEqual(calculate('100 - 10 + 100'), 190.0)
```

### 4.5.2. test\_telebot.py

```
import unittest
import os.path

from calculate.work_with_calculate import generate_value

class test_telebot(unittest.TestCase):

    # Проверка создания файла и наличия 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
        )
```

### 4.5.3. test\_json.py

```
import unittest

from calculate.json_function import load_data, write_data, merge_data

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'
        }
    ]
}

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

    # Проверка на добавлении json дата 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(),
            {'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(),
        {'369350471': [
            { 'id': 12425, 'result': '70', 'value': '30 + 40' },
            { 'id': 78678, 'result': '110', 'value': '70 + 40' }
        ]})

```

#### 4.5.4. test\_filed.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.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'}
        ]
    )

```

#### 4.5.5. 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 числа
    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()

```

## 4.6.Behave

### 4.6.1. check\_unique.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 |
| ['a', 'A', 'b', 'B', '1', '1', '2', '2'] | ['a', 'A', 'b', 'B', '1',
'2'] | 0 |
| ['a', 'A', 'b', 'B', '1', '1', '2', '2'] | ['a', 'b', '1', '2']
| 1 |

```

#### 4.6.2. check filed.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' }] |

```

## 4.7.Steps (for Behave)

### 4.7.1. 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)

```

#### 4.7.2. 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}')

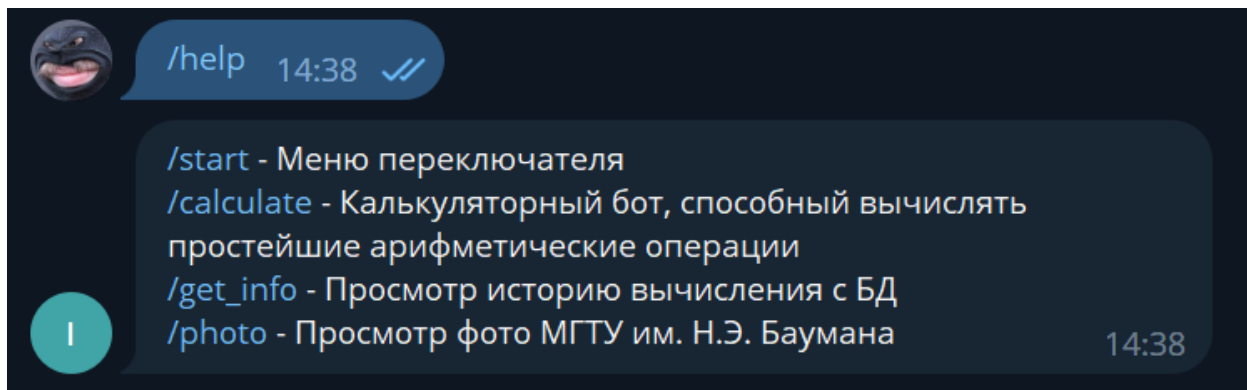
@When("Finding unique values, case: {CASE}")
def given_increment(context, CASE):
    check = bool(int(CASE))
    if (check == True):
        unique_list = Unique(context.LIST, ignore_case=check)
    else:
        unique_list = Unique(context.LIST)

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

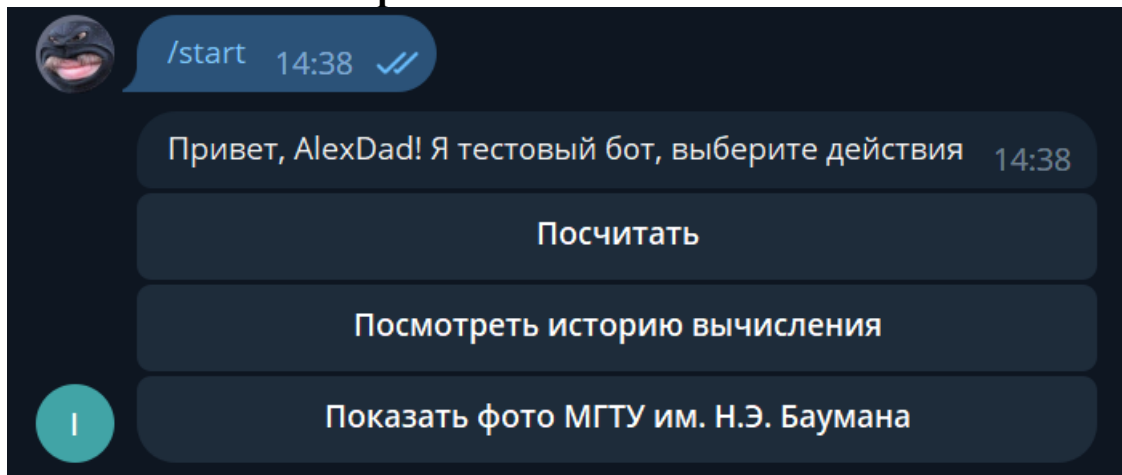
@Then("Output unique values: {UNIQUE}")
def then_results(context, UNIQUE):
    assert context.results.arr == ast.literal_eval(UNIQUE)
    print(f'Уникальные значения: {context.results.arr}')
```

## 5. Результаты работы программы в Telegram

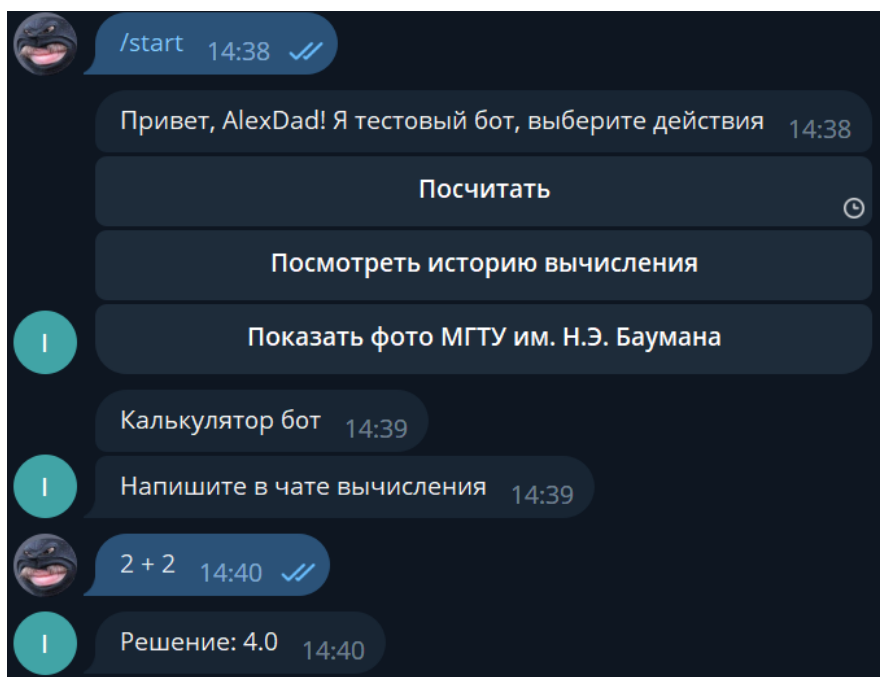
### 5.1. Получение справочную информацию



### 5.2. Основное меню переключателя



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

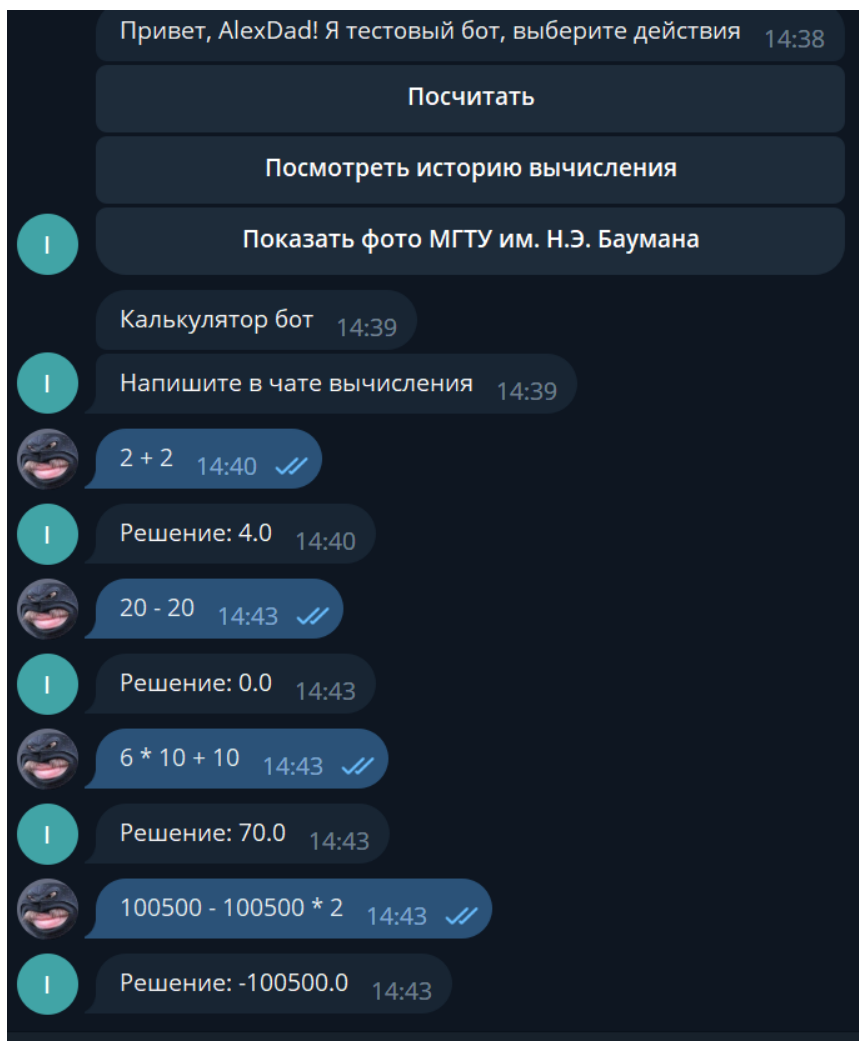


#### 5.4. Данные хранятся в БД в формате JSON

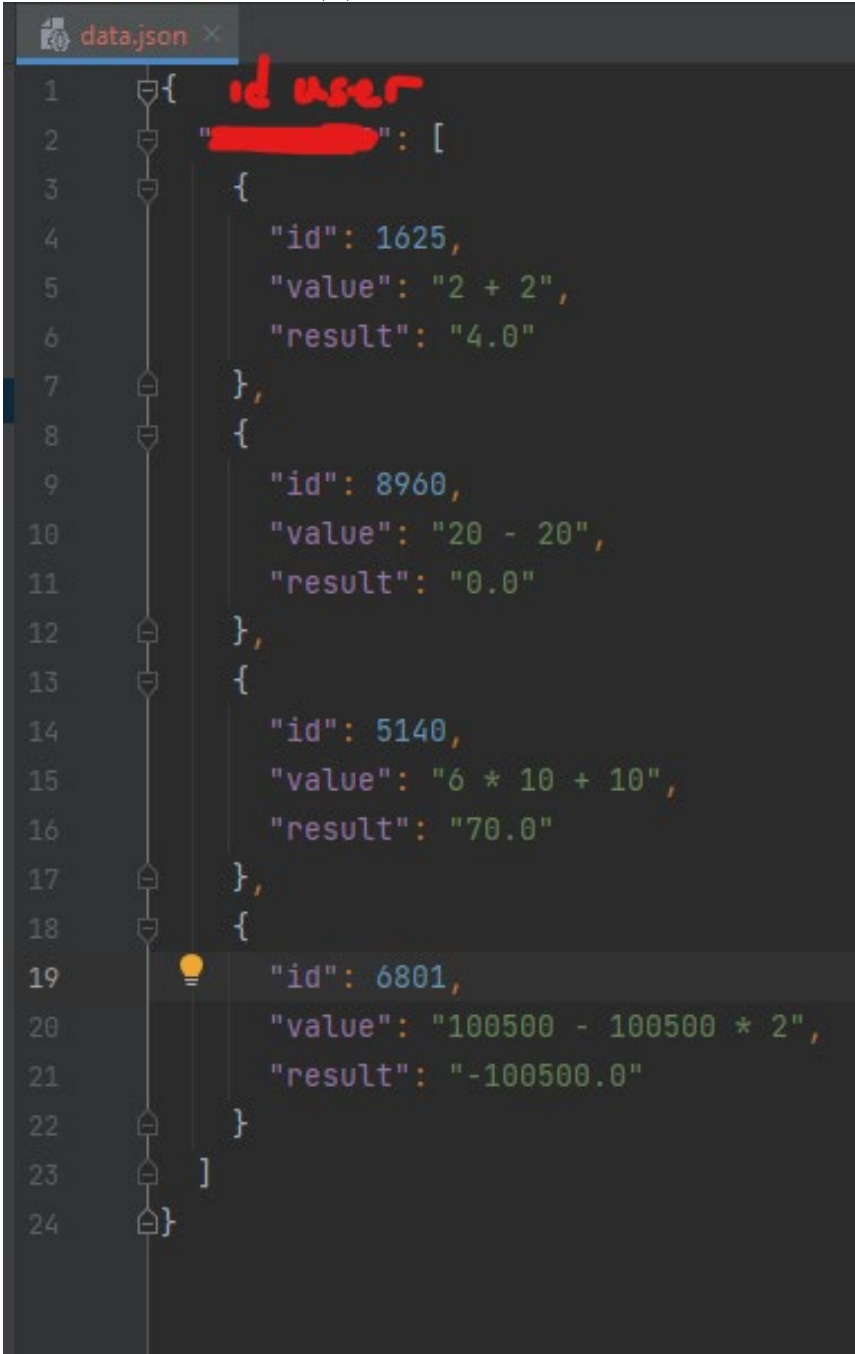


```
1 {  
2   "id": 1625,  
3   "value": "2 + 2",  
4   "result": "4.0"  
5 }  
6 ]  
7 }  
8 }  
9 }
```

#### 5.5. После несколько вычислений

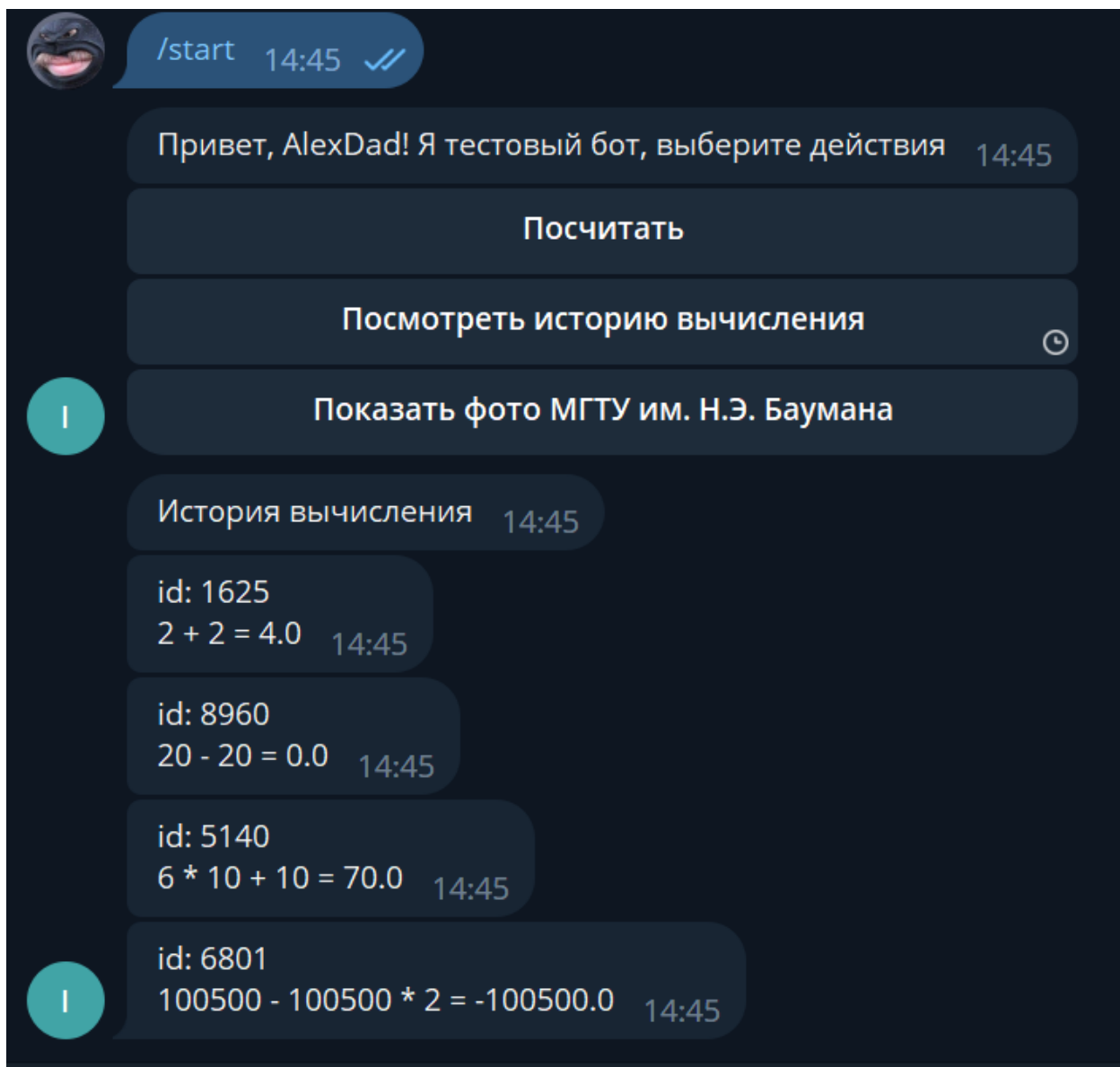


## 5.6.Обновленная БД

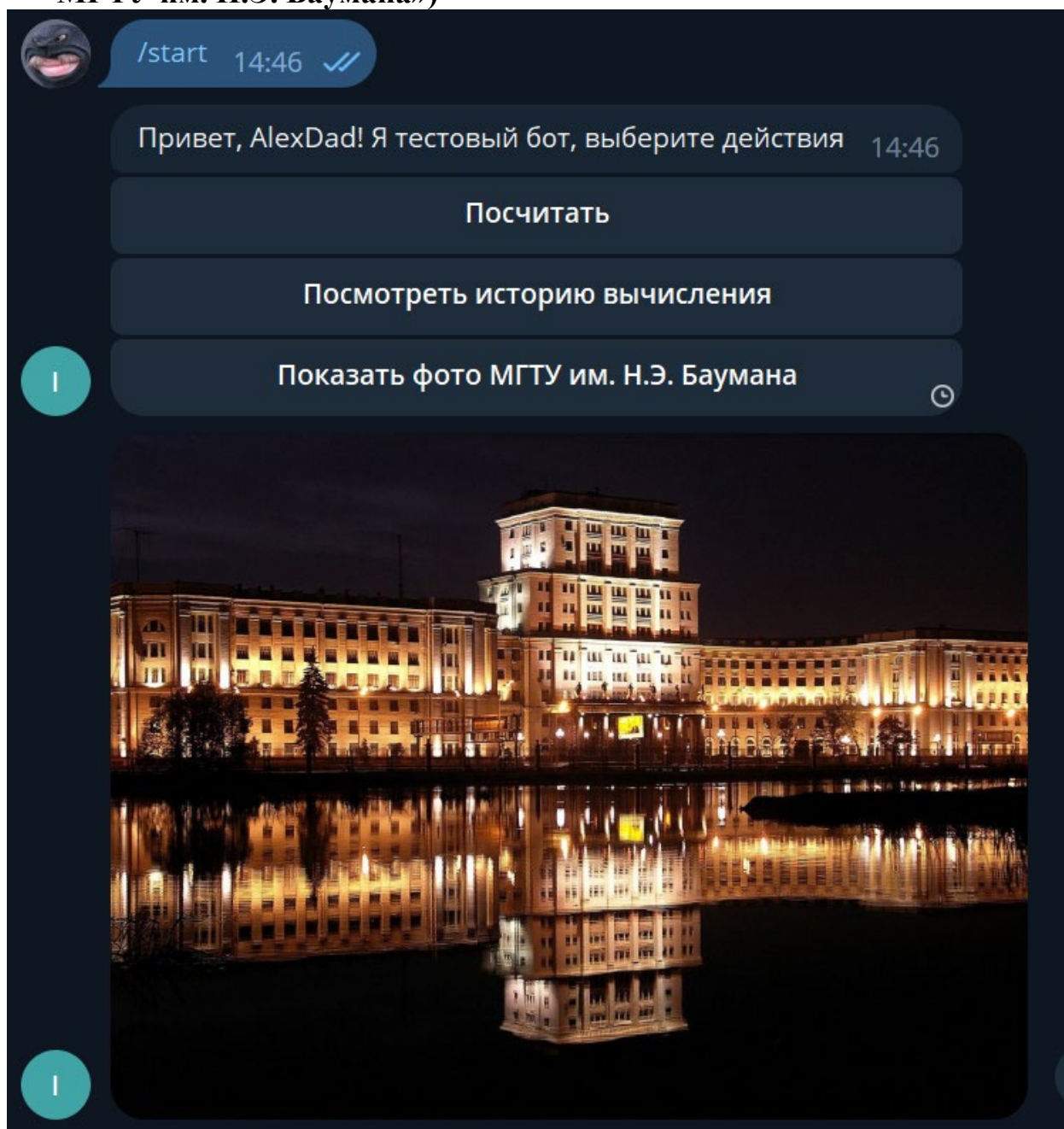


```
1  {  
2    "id user": [  
3      {  
4        "id": 1625,  
5        "value": "2 + 2",  
6        "result": "4.0"  
7      },  
8      {  
9        "id": 8960,  
10       "value": "20 - 20",  
11       "result": "0.0"  
12     },  
13     {  
14       "id": 5140,  
15       "value": "6 * 10 + 10",  
16       "result": "70.0"  
17     },  
18     {  
19       "id": 6801,  
20       "value": "100500 - 100500 * 2",  
21       "result": "-100500.0"  
22     }  
23   ]  
24 }
```

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



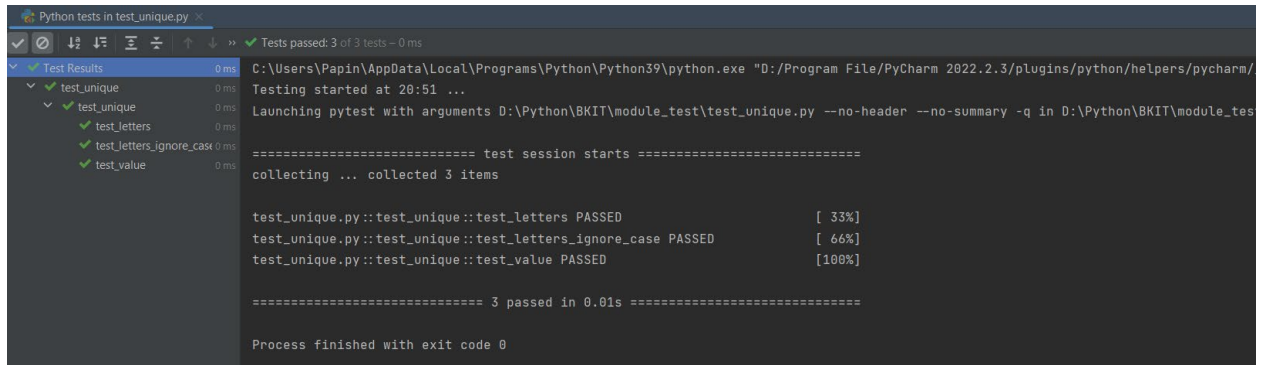
## 5.8. Просмотр фотографии (после нажатии на кнопку «Показать фото МГТУ им. Н.Э. Баумана»)



## 6. Модульное тестирование

### 6.1. unittest

#### 6.1.1. test\_unique.py



The screenshot shows the PyCharm test runner interface for the file `test_unique.py`. The left sidebar displays a tree view of test results, showing three tests: `test_unique`, `test_letters`, and `test_letters_ignore_case`, all of which are marked as passed. The main pane shows the output of the test session, including the command used to launch pytest and the results of the tests.

```
Python tests in test_unique.py
Tests passed: 3 of 3 tests - 0 ms

Test Results
test_unique 0 ms
test_letters 0 ms
test_letters_ignore_case 0 ms
test_value 0 ms

C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/helpers/pycharm/...
Testing started at 20:51 ...
Launching pytest with arguments D:\Python\BKIT\module_test\test_unique.py --no-header --no-summary -q in D:\Python\BKIT\module_test...

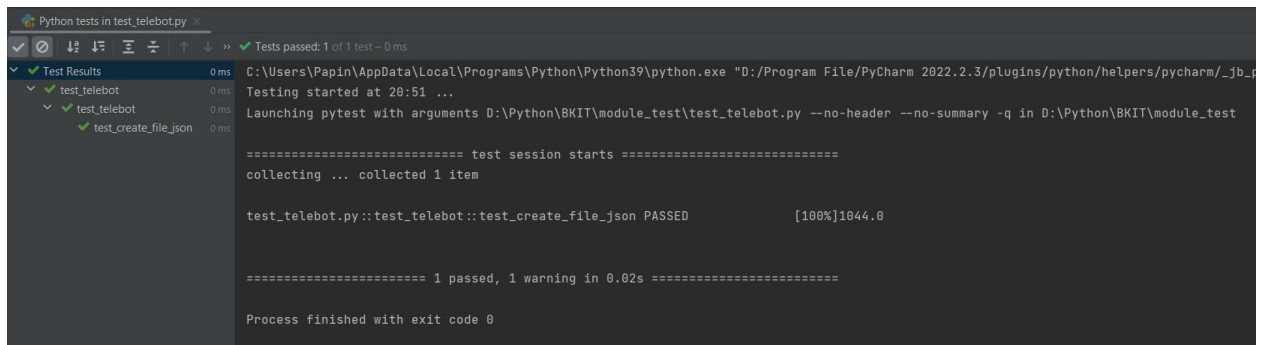
===== test session starts =====
collecting ... collected 3 items

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 0.01s =====

Process finished with exit code 0
```

#### 6.1.2. test\_telebot.py



The screenshot shows the PyCharm test runner interface for the file `test_telebot.py`. The left sidebar displays a tree view of test results, showing one test: `test_create_file_json`, which is marked as passed. The main pane shows the output of the test session, including the command used to launch pytest and the results of the tests.

```
Python tests in test_telebot.py
Tests passed: 1 of 1 test - 0 ms

Test Results
test_telebot 0 ms
test_create_file_json 0 ms

C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/helpers/pycharm/_jb.p...
Testing started at 20:51 ...
Launching pytest with arguments D:\Python\BKIT\module_test\test_telebot.py --no-header --no-summary -q in D:\Python\BKIT\module_test...

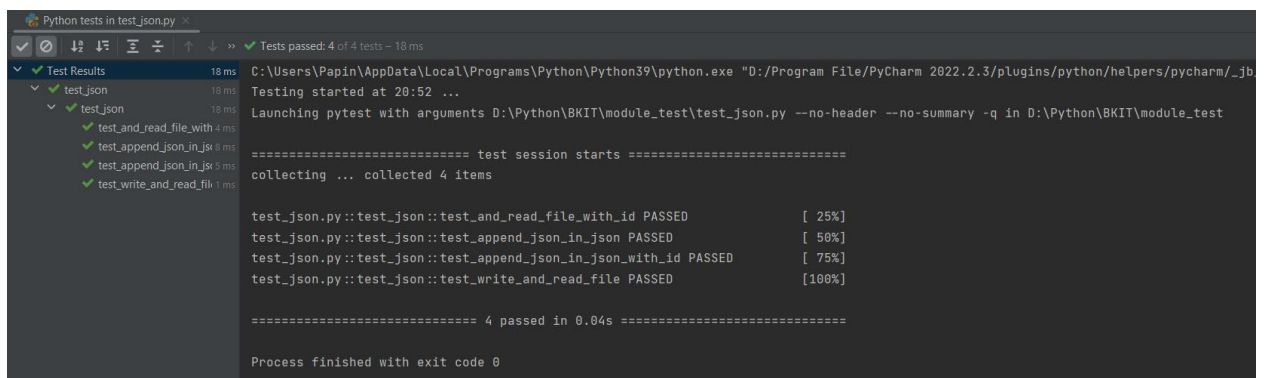
===== test session starts =====
collecting ... collected 1 item

test_telebot.py::test_telebot::test_create_file_json PASSED [100%]1044.0

===== 1 passed, 1 warning in 0.02s =====

Process finished with exit code 0
```

#### 6.1.3. test\_json.py



The screenshot shows the PyCharm test runner interface for the file `test_json.py`. The left sidebar displays a tree view of test results, showing four tests: `test_and_read_file_with_id`, `test_append_json_in_json`, `test_append_json_in_json_with_id`, and `test_write_and_read_file`, all of which are marked as passed. The main pane shows the output of the test session, including the command used to launch pytest and the results of the tests.

```
Python tests in test_json.py
Tests passed: 4 of 4 tests - 18 ms

Test Results
test_and_read_file_with_id 4 ms
test_append_json_in_json 8 ms
test_append_json_in_json_with_id 5 ms
test_write_and_read_file 1 ms

C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/helpers/pycharm/_jb.p...
Testing started at 20:52 ...
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 4 items

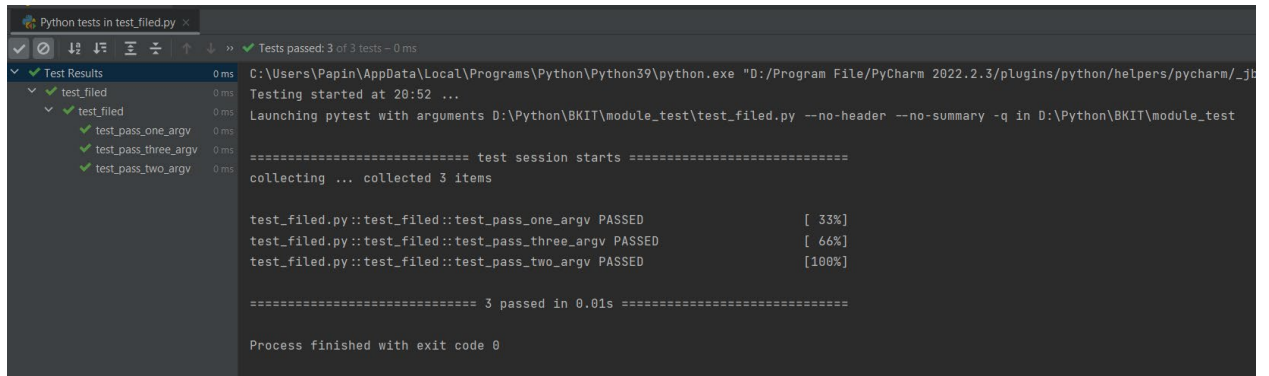
test_json.py::test_json::test_and_read_file_with_id PASSED [ 25%]
test_json.py::test_json::test_append_json_in_json PASSED [ 50%]
test_json.py::test_json::test_append_json_in_json_with_id PASSED [ 75%]
test_json.py::test_json::test_write_and_read_file PASSED [100%]

===== 4 passed in 0.04s =====

Process finished with exit code 0
```



## 6.1.4. test\_filed.py



The image shows a PyCharm test runner window for the file `test_filed.py`. The left sidebar shows a tree view with `test_filed` expanded, containing `test_pass_one_argv`, `test_pass_three_argv`, and `test_pass_two_argv`, all marked as passed. The main pane shows the test execution log. The log indicates that 3 tests passed in 0.01s. The tests are: `test_filed.py::test_filed::test_pass_one_argv PASSED [ 33%]`, `test_filed.py::test_filed::test_pass_three_argv PASSED [ 66%]`, and `test_filed.py::test_filed::test_pass_two_argv PASSED [100%]`. The process finished with exit code 0.

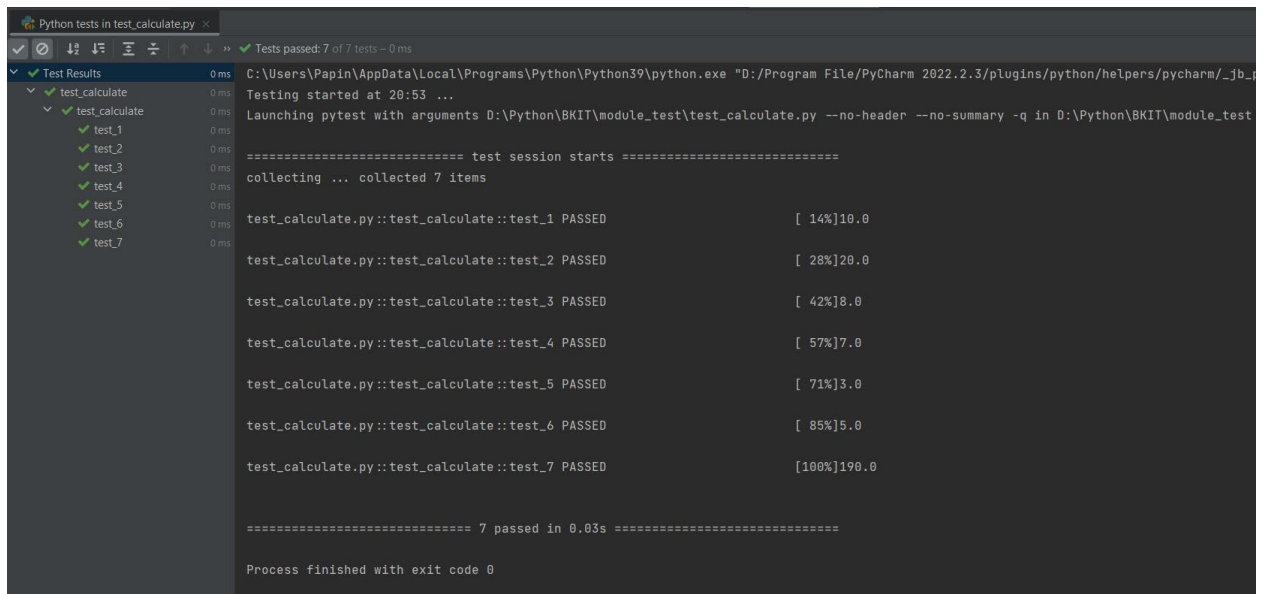
```
Python tests in test_filed.py x
✓ Tests passed: 3 of 3 tests - 0 ms
Test Results 0 ms
  ✓ test_filed 0 ms
    ✓ test_pass_one_argv 0 ms
    ✓ test_pass_three_argv 0 ms
    ✓ test_pass_two_argv 0 ms
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/helpers/pycharm/_jb_...
Testing started at 20:52 ...
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 0.01s =====

Process finished with exit code 0
```

## 6.1.5. test\_calculate.py



The image shows a PyCharm test runner window for the file `test_calculate.py`. The left sidebar shows a tree view with `test_calculate` expanded, containing `test_1` through `test_7`, all marked as passed. The main pane shows the test execution log. The log indicates that 7 tests passed in 0.03s. The tests are: `test_calculate.py::test_calculate::test_1 PASSED [ 14%]10.0`, `test_calculate.py::test_calculate::test_2 PASSED [ 28%]20.0`, `test_calculate.py::test_calculate::test_3 PASSED [ 42%]8.0`, `test_calculate.py::test_calculate::test_4 PASSED [ 57%]7.0`, `test_calculate.py::test_calculate::test_5 PASSED [ 71%]3.0`, `test_calculate.py::test_calculate::test_6 PASSED [ 85%]5.0`, and `test_calculate.py::test_calculate::test_7 PASSED [100%]190.0`. The process finished with exit code 0.

```
Python tests in test_calculate.py x
✓ Tests passed: 7 of 7 tests - 0 ms
Test Results 0 ms
  ✓ test_calculate 0 ms
    ✓ test_1 0 ms
    ✓ test_2 0 ms
    ✓ test_3 0 ms
    ✓ test_4 0 ms
    ✓ test_5 0 ms
    ✓ test_6 0 ms
    ✓ test_7 0 ms
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/helpers/pycharm/_jb_...
Testing started at 20:53 ...
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 7 items

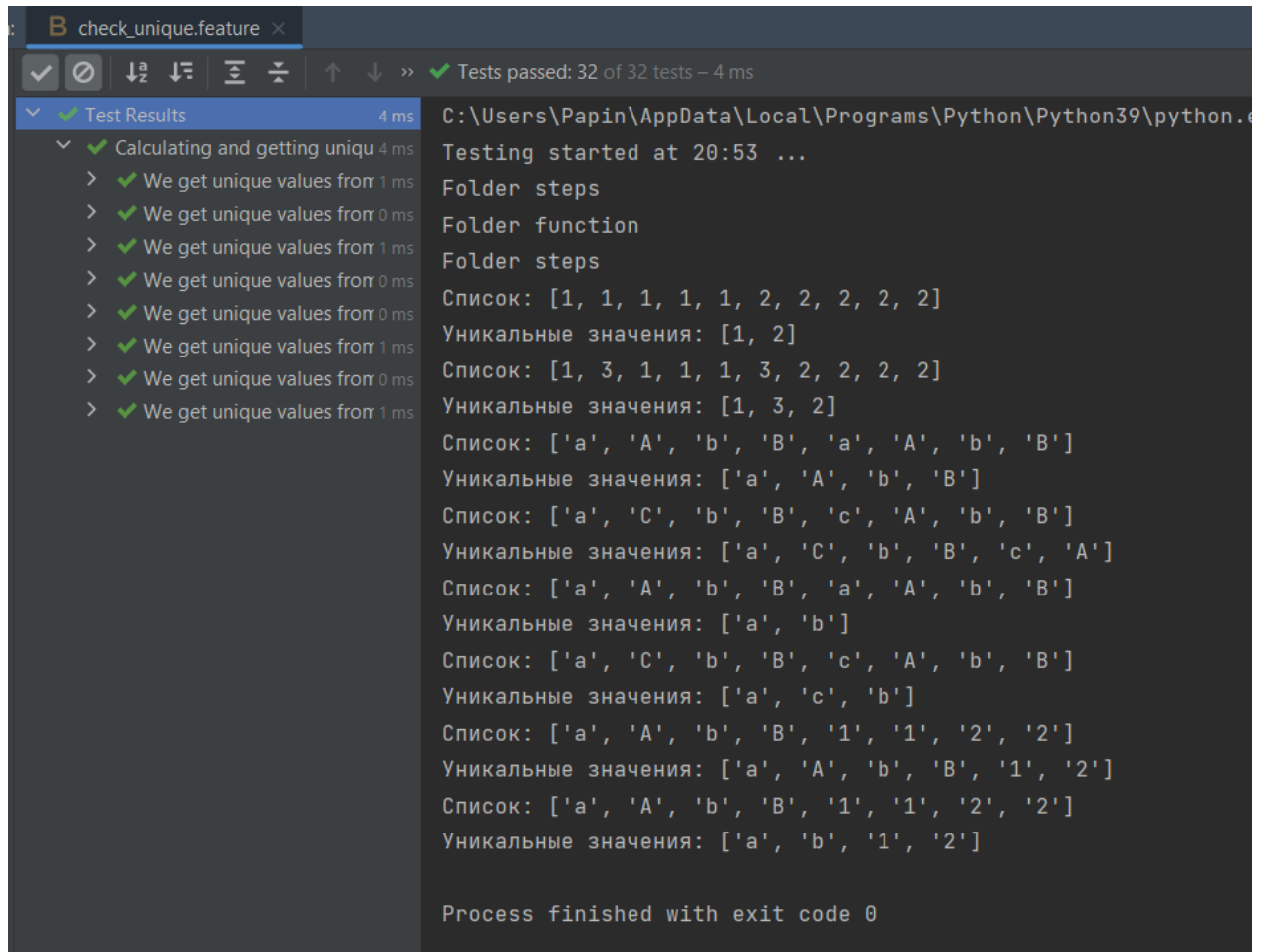
test_calculate.py::test_calculate::test_1 PASSED [ 14%]10.0
test_calculate.py::test_calculate::test_2 PASSED [ 28%]20.0
test_calculate.py::test_calculate::test_3 PASSED [ 42%]8.0
test_calculate.py::test_calculate::test_4 PASSED [ 57%]7.0
test_calculate.py::test_calculate::test_5 PASSED [ 71%]3.0
test_calculate.py::test_calculate::test_6 PASSED [ 85%]5.0
test_calculate.py::test_calculate::test_7 PASSED [100%]190.0

===== 7 passed in 0.03s =====

Process finished with exit code 0
```

## 6.2.Behave

### 6.2.1. check\_unique.feature

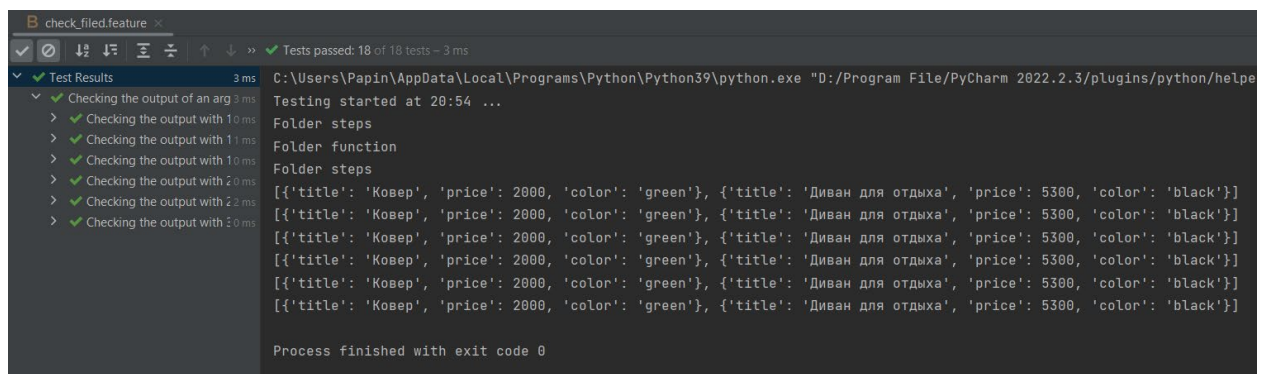


The screenshot shows the PyCharm test runner interface. The top bar indicates 'Tests passed: 32 of 32 tests - 4 ms'. The left sidebar shows a tree view of test results for 'check\_unique.feature', with all tests passing. The main panel displays the test output, which includes the following text:

```
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe
Testing started at 20:53 ...
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, 3, 2, 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']

Process finished with exit code 0
```

### 6.2.2. check\_unique.feature



The screenshot shows the PyCharm test runner interface. The top bar indicates 'Tests passed: 18 of 18 tests - 3 ms'. The left sidebar shows a tree view of test results for 'check\_unique.feature', with all tests passing. The main panel displays the test output, which includes the following text:

```
C:\Users\Papin\AppData\Local\Programs\Python\Python39\python.exe "D:/Program File/PyCharm 2022.2.3/plugins/python/help
Testing started at 20:54 ...
Folder steps
Folder function
Folder steps
[{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
[{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
[{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
[{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
[{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
[{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]

Process finished with exit code 0
```