Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»
Отчёт по домашней работе

Выполнил:
студент группы ИУ5-31Б
Шимолина Полина
Кирилловна

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Подпись:_____

Дата:_____

Москва, 2021 г.

Задание:

1. Модифицируйте код лабораторной работы №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).

main.py

```
import telebot
from telebot import types
from enum import Enum
from vedis import Vedis


def degree(a, b):
    return a ** b


def summ(a, b):
    return a + b


def multiply(a, b):
    return float(a * b)


def istrue(a):
    if a:
        return True
    else:
        return False


bot = telebot.TeleBot('5015036432:AAEUYmBb9Pej2po-zdrB07GnVkqx8cJ0trs')

db_file = "db.vdb"
CURRENT_STATE = "CURRENT_STATE"


class States(Enum):
    STATE_START = "STATE_START"  # Начало нового диалога
    STATE_FIRST_NUM = "STATE_FIRST_NUM"
    STATE_SECOND_NUM = "STATE_SECOND_NUM"
    STATE_OPERATION = "STATE_OPERATION"


# получить
def get(key):
    with Vedis(db_file) as db:
        try:
            return db[key].decode()
        except KeyError:
            return States.S_START.value
```

```
# записать
def set(key, value):
    with Vedis(db_file) as db:
        try:
            db[key] = value
            return True
        except:
            return False


# ключ
def make_key(chatid, keyid):
    res = str(chatid) + '__' + str(keyid)
    return res


@bot.message_handler(commands=['start'])
def cmd_start(message):
    bot.send_message(message.chat.id, 'Я - бот-калькулятор! _-_')
    bot.send_message(message.chat.id, 'Сначала введите 2 числа, потом
действие')
    set(make_key(message.chat.id, CURRENT_STATE),
States.STATE_FIRST_NUM.value)
    bot.send_message(message.chat.id, 'Число:')


@bot.message_handler(commands=['reset'])
def cmd_reset(message):
    bot.send_message(message.chat.id, 'Сброс')
    set(make_key(message.chat.id, CURRENT_STATE),
States.STATE_FIRST_NUM.value)


@bot.message_handler(func=lambda message: get(make_key(message.chat.id,
CURRENT_STATE)) == States.STATE_FIRST_NUM.value)
def first_num(message):
    text = message.text
    if not text.replace('.', "", 1).isdigit():
        bot.send_message(message.chat.id, '-_-')
        return
    else:
        set(make_key(message.chat.id, CURRENT_STATE),
States.STATE_SECOND_NUM.value)
        set(make_key(message.chat.id, States.STATE_FIRST_NUM.value), text)
        bot.send_message(message.chat.id, 'Число:')


@bot.message_handler(
    func=lambda message: get(make_key(message.chat.id, CURRENT_STATE)) ==
States.STATE_SECOND_NUM.value)
def second_num(message):
    text = message.text
    if not text.replace('.', "", 1).isdigit():
        bot.send_message(message.chat.id, '-_-')
        return
    else:
        set(make_key(message.chat.id, CURRENT_STATE),
States.STATE_OPERATION.value)
        set(make_key(message.chat.id, States.STATE_SECOND_NUM.value), text)
        markup = types.ReplyKeyboardMarkup(row_width=2)
        b1 = types.KeyboardButton('^')
        b2 = types.KeyboardButton('√')
        b3 = types.KeyboardButton('+')
        b4 = types.KeyboardButton('-')
```

```
        b5 = types.KeyboardButton('*')
        b6 = types.KeyboardButton('/')
        markup.add(b1, b2, b3, b4, b5, b6)
        bot.send_message(message.chat.id, 'Действие:', reply_markup=markup)


@bot.message_handler(func=lambda message: get(make_key(message.chat.id,
CURRENT_STATE)) == States.STATE_OPERATION.value)
def operation(message):
    # Текущее действие
    op = message.text
    fv1 = float(get(make_key(message.chat.id, States.STATE_FIRST_NUM.value)))
    fv2 = float(get(make_key(message.chat.id,
States.STATE_SECOND_NUM.value)))
    res = 0
    if op == '^':
        res = degree(fv1, fv2)
    elif op == '√':
        res = degree(fv1, 1 / fv2)
    elif op == '+':
        res = summ(fv1, fv2)
    elif op == '-':
        res = summ(fv1, -fv2)
    elif op == '*':
        res = multiply(fv1, fv2)
    elif op == '/':
        res = multiply(fv1, 1 / fv2)
    markup = types.ReplyKeyboardRemove(selective=False)
    if op == '√':
        bot.send_message(message.chat.id,
                         f'{get(make_key(message.chat.id,
States.STATE_SECOND_NUM.value))}{op}{get(make_key(message.chat.id,
States.STATE_FIRST_NUM.value))}={str(res)}',
                         reply_markup=markup)
    elif op == '^' or op == '+' or op == '-' or op == '*' or op == '/':
        bot.send_message(message.chat.id,
                         f'{get(make_key(message.chat.id,
States.STATE_FIRST_NUM.value))}{op}{get(make_key(message.chat.id,
States.STATE_SECOND_NUM.value))}={str(res)}',
                         reply_markup=markup)

    set(make_key(message.chat.id, CURRENT_STATE),
States.STATE_FIRST_NUM.value)
    bot.send_message(message.chat.id, 'Число:')


if __name__ == '__main__':
    bot.infinity_polling()
```

steps.py

```
from main import *
from behave import *



@given('I have {a} * {b}')
def step_impl(context, a, b):
    context.a = float(a)
    context.b = float(b)


@when('I solve this math problem')
def step_impl(context):
```

```
    context.answer = multiply(context.a, context.b)


@then('I expect to get the answer: {answer}')
def step_impl(context, answer):
    res = float(answer)
    assert context.answer == res, f'{context.answer} is not {res}'

@given('I have {a} + {b}')
def step_impl(context, a, b):
    context.a = float(a)
    context.b = float(b)


@when('I solve the math problem')
def step_impl(context):
    context.answer = summ(context.a, context.b)


@then('I expect the answer: {answer}')
def step_impl(context, answer):
    res = float(answer)
    assert context.answer == res, f'{context.answer} is not {res}'
```

test.feature

```
Feature: check the summ function
  Scenario: solve 8 * 2
    Given I have 8 * 3
    When I solve this math problem
    Then I expect to get the answer: 24.0

  Scenario: solve 6 + 23
    Given I have 6 + 23
    When I solve the math problem
    Then I expect the answer: 29.0
```

TDD.py

```
import lab6
import unittest
from unittest.mock import patch


class Tests(unittest.TestCase):
    def test_1(self):
        self.assertEqual(3.0, lab6.degree(9, 1 / 2))

    def test_2(self):
        self.assertEqual(-229.0, lab6.summ(6, -235))
```

Экранные формы:
-m behave

```
Feature: check the summ function # test.feature:1

  Scenario: solve 8 * 2                    # test.feature:2
    Given I have 8 * 3                     # steps/steps.py:5
    When I solve this math problem         # steps/steps.py:11
    Then I expect to get the answer: 24.0 # steps/steps.py:16

  Scenario: solve 6 + 23                 # test.feature:7
    Given I have 6 + 23                  # steps/steps.py:21
    When I solve the math problem  # steps/steps.py:27
    Then I expect the answer: 29.0 # steps/steps.py:32

1 feature passed, 0 failed, 0 skipped
2 scenarios passed, 0 failed, 0 skipped
6 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.003s
```

TDD