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 ИУ5 «Системы обработки информации и управления»

Домашнее задание

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:
студент группы ИУ5-33Б
Зорькин А.В.

Проверил:
Канев А.И.

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).

Текст программы:

Пункт 1:

Текст программы:

main.py

```
import telebot
import config
import db
import math
import numpy

bot = telebot.TeleBot(config.token)


@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id, 'Решим биквадратное уравнение, с тебя
только коэффициенты!')
    db.set(db.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_FIRST_NUM.value)
    bot.send_message(message.chat.id, 'Введите первое число')


# Обработка первого числа
@bot.message_handler(func=lambda message: db.get(
    db.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_FIRST_NUM.value)
def first_num(message):
    text = message.text
    try:
        float(text)
        if text == '0':
            bot.send_message(message.chat.id, 'Первый коэффициент не может
быть ноль!')
            bot.send_message(message.chat.id, 'Введите первое число')
            return
        else:
            # Состояние не изменяется, выводится сообщение об ошибке
            # Меняем текущее состояние
            db.set(db.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_SECOND_NUM.value)
            # Сохраняем первое число
            db.set(db.make_key(message.chat.id,
config.States.STATE_FIRST_NUM.value), text)
            bot.send_message(message.chat.id, 'Введите второе число')
    except ValueError:
        # Состояние не изменяется, выводится сообщение об ошибке
        bot.send_message(message.chat.id, 'Пожалуйста введите число!')
```

```
        return


# Обработка второго числа
@bot.message_handler(func=lambda message: db.get(
    db.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_SECOND_NUM.value)
def second_num(message):
    text = message.text
    try:
        float(text)
        # Меняем текущее состояние
        db.set(db.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_THIRD_NUM.value)
        # Сохраняем второе число
        db.set(db.make_key(message.chat.id,
config.States.STATE_SECOND_NUM.value), text)
        bot.send_message(message.chat.id, 'Введите третье число')

    except ValueError:
        bot.send_message(message.chat.id, 'Пожалуйста введите число!')
        return


# Обработка третьего числа
@bot.message_handler(func=lambda message: db.get(
    db.make_key(message.chat.id, config.CURRENT_STATE)) ==
config.States.STATE_THIRD_NUM.value)
def third_num(message):
    text = message.text
    try:
        float(text)
        # Сохраняем третье число
        db.set(db.make_key(message.chat.id,
config.States.STATE_THIRD_NUM.value), text)
        # Нахождение корней
        v1 = db.get(db.make_key(message.chat.id,
config.States.STATE_FIRST_NUM.value))
        v2 = db.get(db.make_key(message.chat.id,
config.States.STATE_SECOND_NUM.value))
        v3 = db.get(db.make_key(message.chat.id,
config.States.STATE_THIRD_NUM.value))
        a = float(v1)
        b = float(v2)
        c = float(v3)
        resultfinal = bikvadrat(a, b, c)
        bot.send_message(message.chat.id, "Имеем корни:
{}".format(resultfinal))
        # Меняем текущее состояние
        db.set(db.make_key(message.chat.id, config.CURRENT_STATE),
config.States.STATE_FIRST_NUM.value)
        # Выводим сообщение
        bot.send_message(message.chat.id, 'Введите первое число')
    except ValueError:
        bot.send_message(message.chat.id, 'Введите число!')
        return


def bikvadrat(a, b, c):
    result = []
    D = b * b - 4 * a * c
    if D == 0.0:
        root = -b / (2.0 * a)
        result.append(root)
```

```
    elif D > 0.0:
        sqD = math.sqrt(D)
        root1 = (-b + sqD) / (2.0 * a)
        root2 = (-b - sqD) / (2.0 * a)
        result.append(root1)
        result.append(root2)
    resultfinal = []
    for x in result:
        if x > 0:
            resultfinal.append(numpy.sqrt(x))
            resultfinal.append(-numpy.sqrt(x))
        elif x == 0:
            resultfinal.append(0)
    return resultfinal


if __name__ == '__main__':  # делает нам бесконечный цикл получения данных
    bot.infinity_polling()
```

config.py

```
from enum import Enum

# Токент бота
token = '2114680737:AAE5w5v9UDxLv51rBFUjPA12IEBraDaT5qY'

# Файл базы данных Vedis
db_file = "db.vdb"

# Ключ записи в БД для текущего состояния
CURRENT_STATE = "CURRENT_STATE"


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

db.py

```
from vedis import Vedis
import config


# Чтение значения
def get(key):
    with Vedis(config.db_file) as db:
        try:
            return db[key].decode()
        except KeyError:
            # в случае ошибки значение по умолчанию - начало диалога
            return config.States.S_START.value

        # Запись значения
```

```
def set(key, value):
    with Vedis(config.db_file) as db:
        try:
            db[key] = value
            return True
        except:
            # тут желательно как-то обработать ситуацию
            return False


# Создание ключа для записи и чтения
def make_key(chatid, keyid):
    res = str(chatid) + '__' + str(keyid)
    return res
```

TDD:

```
import unittest
import sys, os

sys.path.append(os.getcwd())
from main import *


class Test_bikvadrat(unittest.TestCase):
    def test_bikvadrat(self):
        self.assertEqual(bikvadrat(1, -2, 1), [1, -1])
        self.assertEqual(bikvadrat(1, -6, 9), [1.7320508075688772, -
1.7320508075688772])
```

Пункт 2:

TDD:

```
import unittest
import sys, os

sys.path.append(os.getcwd())
from shoe_store import *


class Test_Shoe_store(unittest.TestCase):
    def test_receiving_sneakers(self):
        self.assertEqual(Nikefactory().createsneakers().receiving_sneakers(),
'Create Nikesneakers')

self.assertEqual(Adidasfactory().createsneakers().receiving_sneakers(),
'Create Adidassneakers')

    def test_style_slates(self):
        self.assertEqual(Nikefactory().createslates().receiving_slates(),
'Create Nikeslates')
        self.assertEqual(Adidasfactory().createslates().receiving_slates(),
'Create Adidasslates')
```

BDD:

```
# -- FILE: features/example.feature
Feature: Showing off behave

  Scenario: Function return message about creation
    Given Factory
    When test_receiving_sneakers return OK
    And test_style_slates return OK
    Then good job
```

```
from behave import *

from tests.test import *


@given('Factory')
def first_step(context):
    context.a = Test_Shoe_store()


@when('test_receiving_sneakers return OK')
def check_sneakers(context):
    context.a.test_receiving_sneakers()


@when('test_style_slates return OK')
def check_slates(context):
    context.a.test_style_slates()


@then('good job')
def last_step(context):
    pass
```

Результаты выполнения программы:

1 пункт:

```
Launching unittests with arguments python -m unittest test.Test_bikvadrat in


Ran 1 test in 0.006s

OK

Process finished with exit code 0
```

2 пункт:

TDD:

```
C:\Users\LexaC\PycharmProjects\pattern\venv\Scripts\python.exe "C:\Program File
Testing started at 19:32 ...
Launching unittests with arguments python -m unittest C:/Users/LexaC/PycharmPro



Ran 2 tests in 0.019s

OK

Process finished with exit code 0
```

BDD:

```
Microsoft Windows [Version 10.0.19042.1288]
(c) Microsoft Corporation. All rights reserved.
(venv) C:\Users\LexaC\PycharmProjects\pattern>behave features\myfeature.feature
Feature: Showing off behave # features/myfeature.feature:2

  Scenario: Function return message about creation  # features/myfeature.feature:4
    Given Factory                                   # features/steps/myfeaturesteps.py:6
    When test_receiving_sneakers return OK          # features/steps/myfeaturesteps.py:11
    And test_style_slates return OK                 # features/steps/myfeaturesteps.py:16
    Then good job                                   # features/steps/myfeaturesteps.py:21

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
4 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.006s
```