|  |  |
| --- | --- |
|  | **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»

Курс «Базовые компоненты интернет технологий»

Отчет по Домашнему заданию

Выполнил:

студент группы ИУ5-34Б

Григорян Александр Проверил:

Гапанюк Ю.Е.

2021 г*.*

**Описание**

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**

**Файл Features**

**bddminus.feature**

Feature: minus  
 Scenario: minus 5 and 4  
 Given I have context for minusing: ['5', '4', 'minus', 'Посчитать']  
 When I call minusoutput  
 Then I expect to get message with minusing result: '5 - 4 = 1'

**bddplus.feature**

Feature: plus  
 Scenario: plus 3 and 4  
 Given I have context for summing: ['3', '4', 'plus', 'Посчитать']  
 When I call sumoutput  
 Then I expect to get message with summing result: '3 + 4 = 7'

**Файл Steps**

**stepsminus.py**

# -\*- coding: utf-8 -\*-  
*import* string  
*from* main *import* \*  
*from* behave *import* given, when, then  
  
@given(u'I have context for minusing: [\'{first}\', \'{second}\', \'{action}\', \'{result}\']')  
*def* step\_minus(*context*, *first*: string, *second*: string, *action*: string, *result*: string):  
 *context*.first = *first  
 context*.second = *second  
 context*.action = *action  
 context*.result = *result*@when(u'I call minusoutput')  
*def* step\_minus(*context*):  
 *context*.msg = minusoutput([*context*.first, *context*.second, *context*.action, *context*.result])  
  
@then(u'I expect to get message with minusing result: \'{msg}\'')  
*def* step\_minus(*context*, *msg*: string):  
 *assert context*.msg == *msg*

**stepsplus.py**

# -\*- coding: utf-8 -\*-  
*import* string  
*from* main *import* \*  
*from* behave *import* given, when, then  
  
  
@given(u'I have context for summing: [\'{first}\', \'{second}\', \'{action}\', \'{result}\']')  
*def* step\_plus(*context*, *first*: string, *second*: string, *action*: string, *result*: string):  
 *context*.first = *first  
 context*.second = *second  
 context*.action = *action  
 context*.result = *result*@when(u'I call sumoutput')  
*def* step\_plus(*context*):  
 *context*.msg = sumoutput([*context*.first, *context*.second, *context*.action, *context*.result])  
  
  
@then(u'I expect to get message with summing result: \'{msg}\'')  
*def* step\_plus(*context*, *msg*: string):  
 *assert context*.msg == *msg*

**main.py**

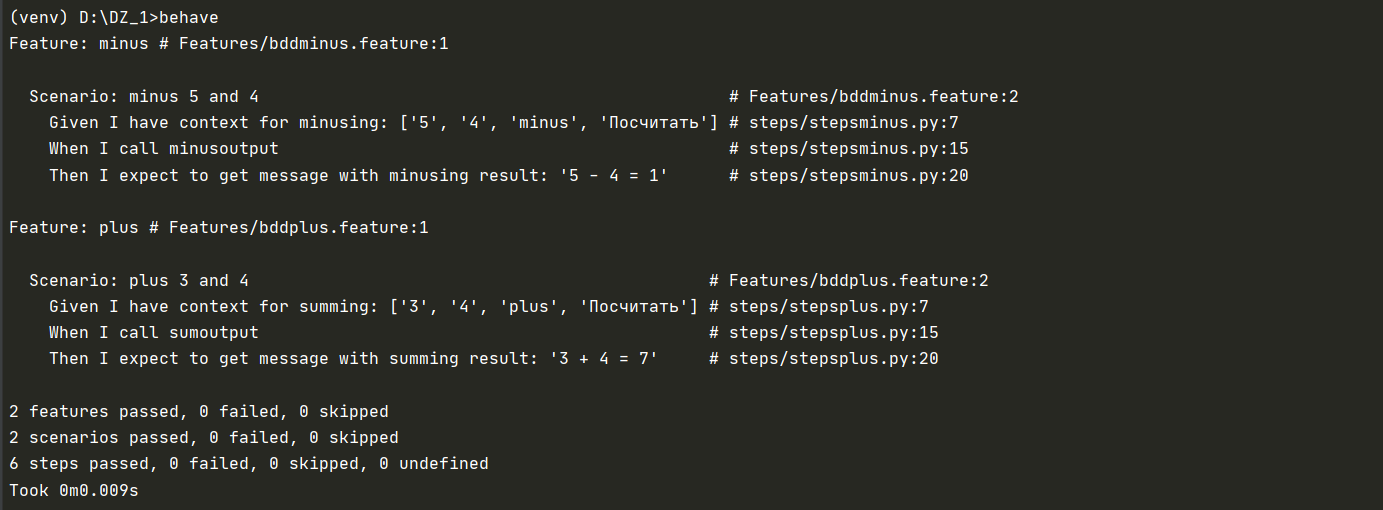
*import* telebot  
*from* telebot *import* types  
  
TOKEN = '\*\*\*'  
bot = telebot.TeleBot(TOKEN)  
*global* config  
config = ['Первое число', 'Второе число', 'Действие', 'Посчитать']  
*global* cases  
cases = ['first', 'second', 'action', 'res']  
*global* call  
call = ''  
*global* actions  
actions = ['plus', 'minus']  
  
  
*def* sumoutput(*config*):  
 *config*[3] = int(*config*[0]) + int(*config*[1])  
 *return* f'{int(*config*[0])} + {int(*config*[1])} = {*config*[3]}'  
  
  
*def* minusoutput(*config*):  
 *config*[3] = int(*config*[0]) - int(*config*[1])  
 *return* f'{int(*config*[0])} - {int(*config*[1])} = {*config*[3]}'  
  
  
@bot.message\_handler(commands='start')  
*def* start(*message*):  
 msg = 'Добро пожаловать.'  
 markup = types.InlineKeyboardMarkup()  
 btn = types.InlineKeyboardButton('Начать работу', callback\_data='work')  
 markup.add(btn)  
 bot.send\_message(*message*.chat.id, msg, reply\_markup=markup)  
  
  
@bot.callback\_query\_handler(*lambda message*: *message*.data == 'work')  
*def* work(*message*):  
 msg = 'Введите данные'  
 markup = types.InlineKeyboardMarkup(row\_width=1)  
 *for* i *in* range(4):  
 btn = types.InlineKeyboardButton(f'{config[i]}', callback\_data=f'{cases[i]}')  
 markup.add(btn)  
 btn = types.InlineKeyboardButton('Сбросить', callback\_data='reset')  
 markup.add(btn)  
 bot.send\_message(*message*.message.chat.id, msg, reply\_markup=markup)  
  
  
@bot.callback\_query\_handler(*lambda message*: *message*.data == cases[0] *or message*.data == cases[1])  
*def* nums(*message*):  
 *global* call  
 call = cases[cases.index(*message*.data)]  
 msg = 'Введите число'  
 bot.send\_message(*message*.message.chat.id, msg)  
  
  
@bot.callback\_query\_handler(*lambda message*: *message*.data == cases[2])  
*def* action(*message*):  
 msg = 'Выберите действие'  
 markup = types.InlineKeyboardMarkup(row\_width=2)  
 btn = types.InlineKeyboardButton('+', callback\_data='plus')  
 btn1 = types.InlineKeyboardButton('-', callback\_data='minus')  
 markup.add(btn, btn1)  
 bot.send\_message(*message*.message.chat.id, msg, reply\_markup=markup)  
  
  
@bot.callback\_query\_handler(*lambda message*: *message*.data *in* actions)  
*def* act(*message*):  
 *if message*.data == actions[0]:  
 config[2] = actions[0]  
 *elif message*.data == actions[1]:  
 config[2] = actions[1]  
 markup = types.InlineKeyboardMarkup(row\_width=1)  
 msg = 'Введите данные'  
 *for* i *in* range(4):  
 *if not* config[i].isdigit() *and not* config[i] *in* actions:  
 btn = types.InlineKeyboardButton(f'{config[i]}', callback\_data=f'{cases[i]}')  
 markup.add(btn)  
 btn = types.InlineKeyboardButton('Сбросить', callback\_data='reset')  
 markup.add(btn)  
 bot.send\_message(*message*.message.chat.id, msg, reply\_markup=markup)  
  
  
@bot.callback\_query\_handler(*lambda message*: *message*.data == cases[3])  
*def* res(*message*):  
 *if* config[0].isdigit() *and* config[1].isdigit() *and* (config[2] *in* actions):  
 *if* config[2] == 'plus':  
 msg = sumoutput(config)  
 *else*:  
 msg = minusoutput(config)  
 markup = types.InlineKeyboardMarkup()  
 btn = types.InlineKeyboardButton('Сбросить', callback\_data='reset')  
 markup.add(btn)  
 bot.send\_message(*message*.message.chat.id, msg, reply\_markup=markup)  
 *else*:  
 msg = 'Недостаточно данных'  
 markup = types.InlineKeyboardMarkup()  
 *for* i *in* range(4):  
 *if not* config[i].isdigit() *and not* config[i] *in* actions:  
 btn = types.InlineKeyboardButton(f'{config[i]}', callback\_data=f'{cases[i]}')  
 markup.add(btn)  
 btn = types.InlineKeyboardButton('Сбросить', callback\_data='reset')  
 markup.add(btn)  
 bot.send\_message(*message*.message.chat.id, msg, reply\_markup=markup)  
  
  
@bot.callback\_query\_handler(*lambda message*: *message*.data == 'reset')  
*def* reset(*message*):  
 msg = 'Данные сброшены.'  
 *global* config  
 config = ['Первое число', 'Второе число', 'Действие', 'Посчитать']  
 markup = types.InlineKeyboardMarkup()  
 btn = types.InlineKeyboardButton('Продолжить', callback\_data='work')  
 markup.add(btn)  
 bot.send\_message(*message*.message.chat.id, msg, reply\_markup=markup)  
  
  
@bot.message\_handler(content\_types='text')  
*def* text(*message*):  
 *if* (call == cases[0] *or* call == cases[1]) *and message*.text.isdigit():  
 *if* call == cases[0]:  
 config[0] = *message*.text  
 bot.send\_message(*message*.chat.id, f'Вы ввели первое число {int(*message*.text)}')  
 *elif* call == cases[1]:  
 config[1] = *message*.text  
 bot.send\_message(*message*.chat.id, f'Вы ввели второе число {int(*message*.text)}')  
 markup = types.InlineKeyboardMarkup(row\_width=1)  
 msg = 'Введите данные'  
 *for* i *in* range(4):  
 *if not* config[i].isdigit() *and not* config[i] *in* actions:  
 btn = types.InlineKeyboardButton(f'{config[i]}', callback\_data=f'{cases[i]}')  
 markup.add(btn)  
 btn = types.InlineKeyboardButton('Сбросить', callback\_data='reset')  
 markup.add(btn)  
 bot.send\_message(*message*.chat.id, msg, reply\_markup=markup)  
 *if* (call == cases[0] *or* call == cases[1]) *and not message*.text.isdigit():  
 msg = 'Ошибка, введите число'  
 bot.send\_message(*message*.chat.id, msg)  
  
# bot.polling(none\_stop=True)

**tests.py**

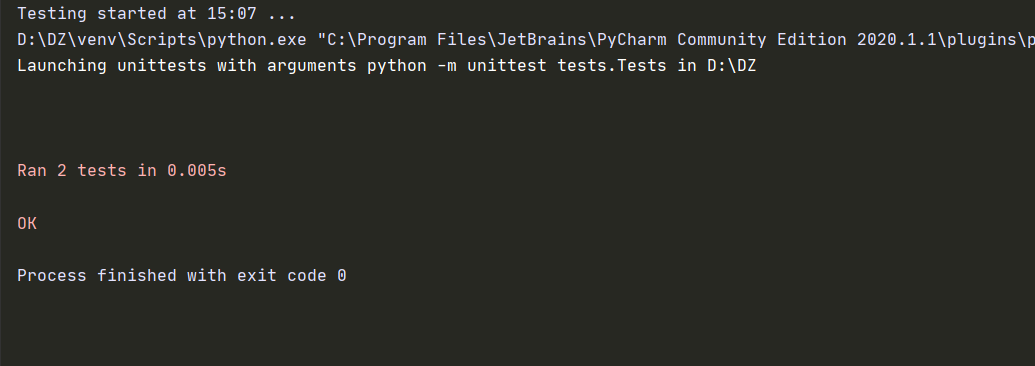
*from* main *import* sumoutput, minusoutput  
*import* unittest  
  
  
*class* Tests(unittest.TestCase):  
  
 *def* test\_sum(self):  
 msg = sumoutput(['1', '2', 'plus', 'Посчитать'])  
 self.assertEqual('1 + 2 = 3', msg)  
  
 *def* test\_minus(self):  
 msg = minusoutput(['3', '2', 'minus', 'Посчитать'])  
 self.assertEqual('3 - 2 = 1', msg)

**Экранные формы с примерами выполнения программы**

**BDD-фреймворк, 2 теста**

****

**tests.py (TDD-фреймворк, 2 теста)**

****