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»
Отчет по домашнему заданию

Выполнил:
студент группы ИУ5-31Б
Егошин Дмитрий Павлович

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Подпись:_____

Дата:_____

Москва, 2021 г.

Домашнее задание
Описание задания

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

1.tests.py

```
from main import minus_output, divide_output
import unittest


class Tests(unittest.TestCase):

    def test_minus(self):
        msg = minus_output(['3', '2', 'minus', 'Посчитать'])
        self.assertEqual('3.0 - 2.0 = 1.0', msg)

    def test_divide(self):
        msg = divide_output(['4', '2', 'divide', 'Посчитать'])
        self.assertEqual('4.0 / 2.0 = 2.0', msg)
```

Результат выполнения

```
"C:\Users\Дмитрий Павлович\AppData\Local\Programs\Python\Python38-32\python.exe" "C:\
Testing started at 22:50 ...
Launching unittests with arguments python -m unittest tests.Tests in F:\BKIT\DZ



Ran 2 tests in 0.002s

OK

Process finished with exit code 0
```

2.StepsM, StepsP

```
# -*- coding: utf-8 -*-
import string
from main import *
from behave import given, when, then


@given(u'I have context for multiplying: [\'{first}\', \'{second}\',
\'{action}\', \'{result}\']')
def step_mult(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 multiply_output')
def step_mult(context):
    context.msg = multiply_output([context.first, context.second,
context.action, context.result])


@then(u'I expect to get message with multiplying result: \'{msg}\'')
def step_mult(context, msg: string):
    assert context.msg == msg
```

```
# -*- 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_pow(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 sum_output')
def step_pow(context):
    context.msg = sum_output([context.first, context.second, context.action,
context.result])


@then(u'I expect to get message with sum result: \'{msg}\'')
def step_pow(context, msg: string):
    assert context.msg == msg
```

Результат выполнения

```
PS F:\BKIT\DZ> behave
Feature: multuply # Features/bddM.feature:1

  Scenario: multiply 3 and 4                                          # Features/bddM.feature:2
    Given I have context for multiplying: ['3', '4', 'multiply', 'Посчитать'] # steps/stepsM.py:7
    When I call multiply_output                                       # steps/stepsM.py:15
    Then I expect to get message with multiplying result: '3.0 * 4.0 = 12.0'  # steps/stepsM.py:20

Feature: plus # Features/bddP.feature:1

  Scenario: 3 plus 4                                                # Features/bddP.feature:2
    Given I have context for summing: ['3', '4', 'plus', 'Посчитать'] # steps/stepsP.py:7
    When I call sum_output                                          # steps/stepsP.py:15
    Then I expect to get message with sum result: '3.0 + 4.0 = 7.0'   # steps/stepsP.py:20

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