Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №5 «Модульное тестирование в Python»

Выполнил:
Студент группы ИУ5-32Б:
Секретов Кирилл
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2022 г.

Описание задания

1. Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
2. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
3. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
    - TDD – фреймворк (не менее 3 тестов).
    - BDD – фреймворк (не менее 3 тестов).
    - Создание Mock-объектов.

Текст программы

Программа для тестирования взятая из первой лабораторной работы BiSquareRoot.py

```
import sys
import math

def get_coef(index, prompt):
    '''
    Читаем коэффициент из командной строки или вводим с клавиатуры
    Args:
        index (int): Номер параметра в командной строке
        prompt (str): Приглашение для ввода коэффицента
    Returns:
        float: Коэффициент квадратного уравнения
    '''
    try:
        coef = float(sys.argv[index])
    except:
        while True:
            try:
                coef = float(input(prompt))
                break
            except ValueError:
                print("Ошибка! Попробуйте ещё раз...")
    return coef

def check(result, root):
    if root == 0.0:
        result.append(root)
    elif root > 0.0:
        result.append(-round(math.sqrt(root), 3))
        result.append(round(math.sqrt(root), 3))

def get_roots(a, b, c):
    result = []
    if a == 0:
        if b != 0:
            root = -c / b
            check(result, root)
        return result
    D = b*b - 4*a*c
    if D == 0.0:
        root = -b / (2.0*a)
        check(result, root)
    elif D > 0.0:
        sqrtD = math.sqrt(D)
        root1 = (-b + sqrtD) / (2.0*a)
        root2 = (-b - sqrtD) / (2.0*a)
        check(result, root1)
        check(result, root2)
    return result


def main():
    a = get_coef(1, 'Введите коэффициент A:')
    b = get_coef(2, 'Введите коэффициент B:')
    c = get_coef(3, 'Введите коэффициент C:')

    if a == b == c == 0.0:
        print('Бесконечное число корней')
    else:
        roots = get_roots(a,b,c)
        len_roots = len(roots)
        if len_roots == 0:
            print('Нет корней')
        elif len_roots == 1:
```

```
            print('Один корень: {}'.format(roots[0]))
        elif len_roots == 2:
            print('Два корня: {} и {}'.format(roots[0], roots[1]))
        elif len_roots == 3:
            print('Три корня: {}, {} и {}'.format(roots[0], roots[1], roots[2]))
        elif len_roots == 4:
            print('Четыре корня: {}, {}, {} и {}'.format(roots[0], roots[1],
roots[2], roots[3]))

if __name__ == "__main__":
    main()
```

TDD

```
import unittest
from unittest.mock import patch, Mock
from main import get_roots, check
import math

class TestRoots(unittest.TestCase):

    def test_roots_is_equal(self):
        self.assertEqual(get_roots(1,-7,4), [-2.524, 2.524, -0.792, 0.792])
        self.assertEqual(get_roots(1,-4,4), [-1.414, 1.414])
        self.assertEqual(get_roots(-4,40,0), [-0.0, -3.162, 3.162])
        self.assertEqual(get_roots(1,0,-16), [-2.0, 2.0])

    def test_string_root(self):
        self.assertRaises(TypeError, get_roots, '0000')
        self.assertRaises(TypeError, get_roots, 'True')
        self.assertRaises(TypeError, get_roots, [1,1])

if __name__ == '__main__':
    unittest.main()
```

BDD

Feature файл

```
Feature: Test Biquadratic equation Functionality

Scenario:Biquadratic equation solver tester
    Given Biquadratic equation solver is running
    When a, b, c are "1", "-7", and "4"
    Then Result is "-2.524, 2.524, -0.792, 0.792"

Scenario: Biquadratic equation solver tester
    Given Biquadratic equation solver is running
    When a, b, c are "1", "-4", and "4"
    Then Result is "-1.414, 1.414"

Scenario: Biquadratic equation solver tester
    Given Biquadratic equation solver is running
    When a, b, c are "1", "0", and "-16"
    Then Result is "-2.0, 2.0"
```

Bdd_main.py

```
from behave import given, when, then
from main import get_roots, check


@given(u"Biquadratic equation solver is running")
def step_impl(context):
    print(u"Step:Biquadratic equation solver is running")
```

```
@when(u'a, b, c are "{a}", "{b}", and "{c}"')
def step_impl(context, a, b, c):
    print(u'Step: a, b, c are "{}", "{}", and "{}"'. format(a, b, c))
    b = str(get_roots(int(a), int(b), int(c))).rpartition(']')[0]
    c = b.partition('[')[2]
    context.result = c
    print(u'Stored result "{}" in context'. format(context.result))

@then(u'Result is "{out}"')
def step_impl(context, out):
    if (context.result == str(out)):
        print(u'Step: Result is right: "{}", "{}"'.format(context.result, out))
        pass
    else :
        raise Exception ("Invalid root is returned.")
```

Примеры выполнения программ

```
..
----------------------------------------------------------------------
Ran 2 tests in 0.003s

OK
>>>
```

```
C:\Users\MainUser\Desktop\lab_4\code>behave
Feature: Test Biquadratic equation Functionality # features/bdd_main.feature:1

  Scenario: Biquadratic equation solver tester    # features/bdd_main.feature:3
    Given Biquadratic equation solver is running  # steps/bdd_main.py:5
    When a, b, c are "1", "-7", and "4"           # steps/bdd_main.py:9
    Then Result is "-2.524, 2.524, -0.792, 0.792" # steps/bdd_main.py:17

  Scenario: Biquadratic equation solver tester   # features/bdd_main.feature:8
    Given Biquadratic equation solver is running # steps/bdd_main.py:5
    When a, b, c are "1", "-4", and "4"          # steps/bdd_main.py:9
    Then Result is "-1.414, 1.414"               # steps/bdd_main.py:17

  Scenario: Biquadratic equation solver tester   # features/bdd_main.feature:13
    Given Biquadratic equation solver is running # steps/bdd_main.py:5
    When a, b, c are "1", "0", and "-16"         # steps/bdd_main.py:9
    Then Result is "-2.0, 2.0"                   # steps/bdd_main.py:17

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.002s
```