**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 **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

# Отчет по лабораторной работе №5
# по дисциплине Базовые компоненты интернет технологии

## Тема работы: "Модульное тестирование в Python"

**Выполнил:**
Студент группы ИУ5Ц-54Б
Перевощиков Н.Д. 23.11.22 г.
(дата, подпись)

**Проверил:**
Преподаватель
Канев А.И.
(дата, подпись)

Москва, 2022

# СОДЕРЖАНИЕ ОТЧЕТА

## 1. Цель лабораторной работы

Изучение возможностей модульного тестирования в языке Python.

## 2. 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 (необязательное дополнительное задание).

## 3. Текст программы

### Для Unittest

**Main.py**

```
from function.get_roots import get_roots
# Основная программа
def main():
    while True:
        try:
            a = get_coef(1, 'Введите коэффициент A:')
            b = get_coef(2, 'Введите коэффициент B:')
            c = get_coef(3, 'Введите коэффициент C:')

            if (a == None or b == None or c == None):
                print()
                print('Ошибка заполнения!')
                break

            # Вычисление корней
            roots = get_roots(a, b, c)

            # Вывод корней
            len_roots = len(roots)
            if len_roots == 0:
                print('Нет корней')
            elif len_roots == 1:
                print('Один корень {}'.format(round(roots[0], 2)))
            elif len_roots == 2:
                print('Два корня: {} и {}'.format(round(roots[0], 2),
round(roots[1], 2)))
            elif len_roots == 3 and roots[0] == 0.0:
                print('Три корня: {} и {} и {}'.format(round(roots[0], 2),
round(roots[1], 2), round(roots[2], 2)))
            elif len_roots == 3:
                print('Два корня: {} и {}'.format(round(roots[1], 2),
round(roots[2], 2)))
            elif len_roots == 4:
                print(
                    'Четыре корня: {} и {} и {} и {}'.format(round(roots[0],
2), round(roots[1], 2), round(roots[2], 2),
                                                             round(roots[3],
2)))
            else:
                print('Ошибка! Корней нет!')
            break

        except ArithmeticError:
            print('Ошибка! Коэффициент a должен быть натуральным числом!')
            break

# Если сценарий запущен из командной строки
if __name__ == "__main__":
    main()
```

В папке function:

**Init.py**

```
# print('Подключение функционального файла')
```

**Get_coef.py**

```
import sys
```

```python
def get_coef(index, prompt):
    if (index > 3 or index < 1):
        print('Индекс вне диапазона. Индексом должен быть 1, 2 и 3')
        return None

    try:
        # Пробуем прочитать коэффициент из командной строки
        coef_str = sys.argv[index]

        if(coef_str[0] == '-'):
            coef_str = sys.argv[index].replace('-','')
            # print('coef_str_if', coef_str)
        else:
            coef_str = sys.argv[index]
            # print('coef_str_else', coef_str)

        if(coef_str.isdigit() == True):
            coef_str = sys.argv[index]
            # print(f'{coef_str} явл-ется числом', )
        else:
            print('Ошибка! Введите натуральное число!')
            return None

    except:
        while True:
            # Вводим с клавиатуры
            print(prompt)
            coef_str = input()

            # Проверка, есть ли минус числа и нулевой коэффициент?
            if (coef_str[0] == '-'):
                coef_str_buff = coef_str.replace('-', '')
                if (coef_str_buff.isdigit()):
                    break
            if (coef_str.isdigit()):
                break
            else:
                print("Ошибка! Введите натуральное число!")
                return None

    # Переводим строку в действительное число
    coef = float(coef_str)
    return coef
```

## Get_goef_test.py

```python
import sys

def get_coef_test_no_cmd(index, value):
    if (index > 3 or index < 1):
        print('Индекс вне диапазона. Индексом должен быть 1, 2 и 3')
        return None

    while True:
        # Вводим с клавиатуры
        coef_str = str(value)
        # Проверка, есть ли минус числа и нулевой коэффициент?

        if (coef_str[0] == '-'):
            coef_str_buff = coef_str.replace('-', '')
            if (coef_str_buff.isdigit()):
                break
        if (coef_str.isdigit()):
            break
```

```
        else:
            print("Ошибка! Введите натуральное число!")
            return None

    # Переводим строку в действительное число
    coef = float(coef_str)
    return coef


def get_coef_test_with_cmd(index, value):
    if(index > 3 or index < 1):
        print('Индекс вне диапазона. Индексом должен быть 1, 2 и 3')
        return None

    # Пробуем прочитать коэффициент из командной строки
    coef_str = str(value)

    if (coef_str[0] == '-'):
        coef_str = coef_str.replace('-', '')
        # print('coef_str_if', coef_str)
    else:
        coef_str = str(value)
        # print('coef_str_else', coef_str)

    if (coef_str.isdigit() == True):
        coef_str = str(value)
        # print(f'{coef_str} явл-ется числом', )
    else:
        print('Ошибка! Введите натуральное число!')
        return None

    coef = float(coef_str)
    return coef
```

**Get_roots.py**

```
import math
# Функция высчисления дискримината
# Получение корней
def get_roots(a, b, c):
    result = []
    D = b * b - 4 * a * c

    # Если дискриминат равен нулю, то корень может быть только один
    if D == 0.0:
        root = -b / (2.0 * a)
        # result.append(root)
        if (root > 0.0):
            root1 = math.sqrt(root)
            result.append(root1)
            result.append(-root1)

    # Если дискриминат больше нуля, то корнем может быть четыре
    elif D > 0.0:
        sqD = math.sqrt(D)
        root1 = (-b + sqD) / (2.0 * a)
        root2 = (-b - sqD) / (2.0 * a)

        if (root1 == 0):
            result.append(abs(root1))
        elif (root2 == 0):
            result.append(abs(root2))

        if (root1 > 0.0):
```

```
            root3 = math.sqrt(root1)
            result.append(root3)
            result.append(-root3)

        if (root2 > 0.0):
            root4 = math.sqrt(root2)
            result.append(root4)
            result.append(-root4)

    return result
```

После установки в Python Packages: pytest_bbd и behave. Должно работать основная программа test.py, в папке feature программа check_root.feature, а в папке feature есть папка steps: roots.py.

**Test.py**

```
# Подключаем библиотеку unitetest для тестирования
import unittest
import math

'''
assertEqual(self, first, second)
first - передаваемое значение
second - полученное значение (в тело функции должен быть return, если вы там
не оставили, тогда прописать здесь как None)
если передаваемое значение совпадает с полученным значением, то тест пройден
успешно
'''

# Вычисление корня
from function.get_roots import get_roots

# Тест на сумму
class test_get_roots(unittest.TestCase):
    # https://tutomath.ru/baza-znanij/bikvadratnye-uravneniya.html
    # Пример №1
    # Дискриминат больше 0 и 4 корней
    def test_example_1(self):
        self.assertEqual(
            get_roots(1, -5, 6), [math.sqrt(3), -math.sqrt(3), math.sqrt(2),
-math.sqrt(2)]
        )

    # Пример №2
    # Дискриминат равен 0 и 2 корня
    def test_example_2(self):
        self.assertEqual(
            get_roots(1, -4, 4), [math.sqrt(2), -math.sqrt(2)]
        )

    # Пример №3
    # Дискриминат больше 0 и 3 корней
    def test_example_3(self):
        self.assertEqual(
            get_roots(-4, 16, 0), [0, 2, -2]
        )

    # Пример №4
    # Дискриминат равен 0 и 2 корней
    def test_example_4(self):
        self.assertEqual(
            get_roots(1, 0, -16), [2.0, -2.0]
        )
```

```
    # Пример №5
    # Дискриминат равен 0 и нет корней
    def test_example_5(self):
        self.assertEqual(
            get_roots(1, 0, 10), []
        )

    # Пример №6
    # Дискриминат больше 0 и 2 корня
    def test_example_6(self):
        self.assertEqual(
            get_roots(1, -5, -36), [3, -3]
        )

    # Пример №7
    # Дискриминат больше 0 и 4 корней
    def test_example_7(self):
        self.assertEqual(
            get_roots(1, -5, 4), [2.0, -2.0, 1.0, -1.0]
        )


# Получение коэффициента с комадной строки или ввода
from function.get_coef_test import get_coef_test_no_cmd,
get_coef_test_with_cmd

class test_get_coef_no_cmd(unittest.TestCase):
    # Без командной строки

    # Тест на обычное числа
    def test_value_index_1(self):
        self.assertEqual(
            get_coef_test_no_cmd(1, 1), 1.0
        )

    # Тест на нулевое числа
    def test_value_index_2(self):
        self.assertEqual(
            get_coef_test_no_cmd(2, 0), 0.0
        )

    # Тест на обычное числа
    def test_value_index_3(self):
        self.assertEqual(
            get_coef_test_no_cmd(3, 5), 5.0
        )

    # Тест на индекс
    def test_value_index_4(self):
        self.assertEqual(
            get_coef_test_no_cmd(4, 7), None
        )

    # Тест на индекс
    def test_value_index_0(self):
        self.assertEqual(
            get_coef_test_no_cmd(0, 7), None
        )

    # Тест на отрицательное число
    def test_value_index_negative_sign(self):
        self.assertEqual(
            get_coef_test_no_cmd(3, -5), -5.0
```

```
        )

    # Тест на другие символ
    def test_value_other_char(self):
        self.assertEqual(
            get_coef_test_no_cmd(3, 'a'), None
        )

    # Тест на другие символ
    def test_value_other_char_and_negative_sign(self):
        self.assertEqual(
            get_coef_test_no_cmd(3, '-a'), None
        )

class test_get_coef_with_cmd(unittest.TestCase):
    # С командной строки

    # Тест на обычное числа
    def test_value_index_1(self):
        self.assertEqual(
            get_coef_test_with_cmd(1, 1), 1.0
        )

    # Тест на нулевое числа
    def test_value_index_2(self):
        self.assertEqual(
            get_coef_test_with_cmd(2, 0), 0.0
        )

    # Тест на обычное числа
    def test_value_index_3(self):
        self.assertEqual(
            get_coef_test_with_cmd(3, 5), 5.0
        )

    # Тест на индекс
    def test_value_index_4(self):
        self.assertEqual(
            get_coef_test_with_cmd(4, 7), None
        )

    # Тест на индекс
    def test_value_index_0(self):
        self.assertEqual(
            get_coef_test_with_cmd(0, 7), None
        )

    # Тест на отрицательное число
    def test_value_negative_sign(self):
        self.assertEqual(
            get_coef_test_with_cmd(3, -5), -5.0
        )

    # Тест на другие символ
    def test_value_other_char(self):
        self.assertEqual(
            get_coef_test_with_cmd(3, 'a'), None
        )

    # Тест на другие символ
    def test_value_other_char_and_negative_sign(self):
        self.assertEqual(
            get_coef_test_with_cmd(3, '-a'), None
        )
```

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

## Для Behave

## Check_root.feature

```
Feature: The biquadrate equation

   # Нет корней
  Scenario Outline: Checking the roots of biquadrate equations: NO ROOT
    Given I have a function calculation root
    And I get coefficient: <A>, <B>, <C>
    When Calculating
    Then Watch roots: <root1>, <root2>, <root3>, <root4>

    Examples:
      | A | B | C  | root1 | root2 | root3 | root4 |
      | 1 | 0 | 10 | 0     | 0     | 0     | 0     |


    # Два корня
  Scenario Outline: Checking the roots of biquadrate equations: TWO ROOTS
    Given I have a function calculation root
    And I get coefficient: <A>, <B>, <C>
    When Calculating
    Then Watch roots: <root1>, <root2>, <root3>, <root4>

    Examples:
      | A | B  | C   | root1 | root2 | root3 | root4 |
      | 1 | -4 | 4   | 1.41  | -1.41 | 0     | 0     |
      | 1 | -5 | -36 | 3     | -3    | 0     | 0     |


    # Три корня
  Scenario Outline: Checking the roots of biquadrate equations: TWO ROOTS
    Given I have a function calculation root
    And I get coefficient: <A>, <B>, <C>
    When Calculating
    Then Watch roots: <root1>, <root2>, <root3>, <root4>

    Examples:
      | A  | B  | C | root1 | root2 | root3 | root4 |
      | -4 | 16 | 0 | 0     | 2     | -2    | 0     |


    # Четыре корня
  Scenario Outline: Checking the roots of biquadrate equations: FOUR ROOTS
    Given I have a function calculation root
    And I get coefficient: <A>, <B>, <C>
    When Calculating
    Then Watch roots: <root1>, <root2>, <root3>, <root4>

    Examples:
      | A | B  | C | root1 | root2 | root3 | root4 |
      | 1 | -5 | 6 | 1.73  | -1.73 | 1.41  | -1.41 |
      | 1 | -5 | 4 | 2.0   | -2.0  | 1.0   | -1.0  |
```

## Root.py

```
from behave import Given, When, Then
from function.get_roots import get_roots
```

```python
@Given('I have a function calculation root')
def step_impl(context):
    pass


@Given("I get coefficient: {A}, {B}, {C}")
def given_increment(context, A, B, C):
    context.A = int(A)
    context.B = int(B)
    context.C = int(C)
    print(f'Коэффициенты: {A}, {B}, {C}')


@When("Calculating")
def given_increment(context):
    roots = get_roots(context.A, context.B, context.C)
    context.results = roots
    # print(f'Корни: {roots}')


@Then("Watch roots: {root1}, {root2}, {root3}, {root4}")
def then_results(context, root1, root2, root3, root4):
    len_roots = len(context.results)

    # Вывод корней
    if len_roots == 0:
        # assert (context.results == 0)
        print('Нет корней')
    elif len_roots == 1:
        assert context.results == float(root1)
        # print('Один корень {}'.format(round(root1, 2)))
    elif len_roots == 2:
        assert round(context.results[0], 2) == float(root1)
        assert round(context.results[1], 2) == float(root2)
        print('Два корня: {} и {}'.format(root1, root2))
    elif len_roots == 3 and int(root1) == 0.0:
        assert round(context.results[0], 2) == float(root1)
        assert round(context.results[1], 2) == float(root2)
        assert round(context.results[2], 2) == float(root3)
        print('Три корня: {} и {} и {}'.format(root1, root2, root3))
    elif len_roots == 3:
        assert round(context.results[0], 2) == float(root1)
        assert round(context.results[1], 2) == float(root2)
        print('Два корня: {} и {}'.format(root1, root2))
    elif len_roots == 4:
        assert round(context.results[0], 2) == float(root1)
        assert round(context.results[1], 2) == float(root2)
        assert round(context.results[2], 2) == float(root3)
        assert round(context.results[3], 2) == float(root4)
        print('Четыре корня: {} и {} и {} и {}'.format(root1, root2, root3,
root4))
    else:
        print('Ошибка! Корней нет!')
```

## 4. Результат

**PyCharm:**

```
Testing started at 22:33 ...
Коэффициенты: 1, 0, 10
Нет корней
Коэффициенты: 1, -4, 4
Два корня: 1.41 и -1.41
Коэффициенты: 1, -5, -36
Два корня: 3 и -3
Коэффициенты: -4, 16, 0
Три корня: 0 и 2 и -2
Коэффициенты: 1, -5, 6
Четыре корня: 1.73 и -1.73 и 1.41 и -1.41
Коэффициенты: 1, -5, 4
Четыре корня: 2.0 и -2.0 и 1.0 и -1.0

Process finished with exit code 0
```

Нет корней:

```
Testing started at 22:33 ...
Коэффициенты: 1, 0, 10
Нет корней

Process finished with exit code 0
```

Два корня:

```
Testing started at 22:33 ...
Коэффициенты: 1, -4, 4
Два корня: 1.41 и -1.41
Коэффициенты: 1, -5, -36
Два корня: 3 и -3
Коэффициенты: -4, 16, 0
Три корня: 0 и 2 и -2

Process finished with exit code 0
```

Три корня:

```
Testing started at 22:33 ...
Коэффициенты: 1, -4, 4
Два корня: 1.41 и -1.41
Коэффициенты: 1, -5, -36
Два корня: 3 и -3
Коэффициенты: -4, 16, 0
Три корня: 0 и 2 и -2

Process finished with exit code 0
```

Четыре корня:

```
Testing started at 22:34 ...
Коэффициенты: 1, -5, 6
Четыре корня: 1.73 и -1.73 и 1.41 и -1.41
Коэффициенты: 1, -5, 4
Четыре корня: 2.0 и -2.0 и 1.0 и -1.0

Process finished with exit code 0
```

**Командная строка:**

**Для Behave**

Путь файла для работы программы main.py:

..\LAB_05>python main.py

```
D:\Работа\МГТУ им. Н.Э.Баумана\Программирование\Программы\Программы за 5 семестр\LAB_05>python main.py
Введите коэффициент A:
1
Введите коэффициент B:
0
Введите коэффициент C:
10
Нет корней
```

```
D:\Работа\МГТУ им. Н.Э.Баумана\Программирование\Программы\Программы за 5 семестр\LAB_05>python main.py
Введите коэффициент A:
1
Введите коэффициент B:
-4
Введите коэффициент C:
4
Два корня: 1.41 и -1.41
```

```
D:\Работа\МГТУ им. Н.Э.Баумана\Программирование\Программы\Программы за 5 семестр\LAB_05>python main.py
Введите коэффициент A:
1
Введите коэффициент B:
-5
Введите коэффициент C:
-36
Два корня: 3.0 и -3.0
```

```
D:\Работа\МГТУ им. Н.Э.Баумана\Программирование\Программы\Программы за 5 семестр\LAB_05>python main.py
Введите коэффициент A:
-4
Введите коэффициент B:
16
Введите коэффициент C:
0
Три корня: 0.0 и 2.0 и -2.0
```

```
D:\Работа\МГТУ им. Н.Э.Баумана\Программирование\Программы\Программы за 5 семестр\LAB_05>python main.py
Введите коэффициент A:
1
Введите коэффициент B:
-5
Введите коэффициент C:
6
Четыре корня: 1.73 и -1.73 и 1.41 и -1.41
```

```
D:\Работа\МГТУ им. Н.Э.Баумана\Программирование\Программы\Программы за 5 семестр\LAB_05>python main.py
Введите коэффициент A:
1
Введите коэффициент B:
-5
Введите коэффициент C:
4
Четыре корня: 2.0 и -2.0 и 1.0 и -1.0
```

**Для Unittest**

Путь файла для работы программы test.py:

..\LAB_05>python test.py

```
D:\Работа\МГТУ им. Н.Э.Баумана\Программирование\Программы\Программы за 5 семестр\LAB_05>python test.py
Индекс вне диапазона. Индексом должен быть 1, 2 и 3
....Индекс вне диапазона. Индексом должен быть 1, 2 и 3
..Ошибка! Введите натуральное число!
.Ошибка! Введите натуральное число!
.Индекс вне диапазона. Индексом должен быть 1, 2 и 3
....Индекс вне диапазона. Индексом должен быть 1, 2 и 3
..Ошибка! Введите натуральное число!
.Ошибка! Введите натуральное число!
........
----------------------------------------------------------------------
Ran 23 tests in 0.003s

OK
```