



**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»**

Лабораторная работа №5 по БКИТ

“Модульное тестирование Python.”

Выполнил:  
студент группы ИУ5-31Б  
Ашуров Г. В.

Проверил:  
Гапанюк Юрий Евгеньевич

2022г.

Задание:

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).

Для тестирования была выбрана лабораторная работа №1

Код лабораторной работы №1 (файл eq.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 коэффициента
    Returns:
        float: Коэффициент квадратного уравнения
    """
    try:
        # Пробуем прочитать коэффициент из командной строки
        coef_str = sys.argv[index]
    except:
        # Вводим с клавиатуры
        print(prompt)
        coef_str = input()
    # Переводим строку в действительное число
    while True:
        try:
            float(coef_str)
            break
        except:
            print('Ошибка, введите число')
            coef_str = input()
    coef = float(coef_str)
    return coef

def get_roots(a, b, c):
```

```

'''
Вычисление корней квадратного уравнения
Args:
    a (float): коэффициент A
    b (float): коэффициент B
    c (float): коэффициент C
Returns:
    list[float]: Список корней
'''

result = []
D = b * b - 4 * a * c
print(D)
if D == 0.0:
    root = -b / (2.0 * a)
    if root > 0:
        result.append(math.sqrt(root))
        result.append(-math.sqrt(root))
    elif root == 0:
        result.append(0)
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(math.sqrt(root1))
        result.append(-math.sqrt(root1))
    elif root1 == 0:
        result.append(root1)
    if root2 > 0:
        result.append(math.sqrt(root2))
        result.append(-math.sqrt(root2))
    elif root2 == 0:
        result.append(math.fabs(root2))
result = sorted(result)
return result

def main():
    '''
    Основная функция
    '''
    a = get_coef(1, 'Введите коэффициент A:')
    b = get_coef(2, 'Введите коэффициент B:')
    c = get_coef(3, 'Введите коэффициент C:')
    # Вычисление корней
    roots = get_roots(a, b, c)
    # Вывод корней
    roots = sorted(roots)
    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: {}, {}'.format(roots[0], roots[1]))
elif len_roots == 3:
    print('Три корня: {}, {}, {}'.format(roots[0], roots[1], roots[2]))
elif len_roots == 4:
    print('Четыре корня: {}, {}, {}, {}'.format(
        roots[0], roots[1], roots[2], roots[3]))

```

TDD-тестирование (файл tdd.py):

```

from eq import get_roots
import unittest

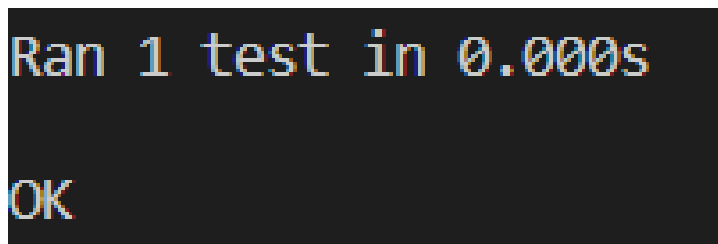
class Testing(unittest.TestCase):
    def __init__(self, methodName: str = ...) -> None:
        super().__init__(methodName)
        self.equation1 = get_roots(5, -4, 1)
        self.equation2 = get_roots(1, -5, -36)
        self.equation3 = get_roots(1, -5, 4)
        self.equation4 = get_roots(1, 0, 0)

    def test_area(self):
        self.assertEqual(self.equation1, [])
        self.assertEqual(self.equation2, [-3, 3])
        self.assertEqual(self.equation3, [-2, -1, 1, 2])
        self.assertEqual(self.equation4, [0])

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

```

Результат выполнения программы:



```

Ran 1 test in 0.000s

OK

```

BDD-тестирование:

Файл test.feature:

Feature: behave testing

Scenario Outline: check roots

Given <a> is a, b is <b>, c is <c>

When start

Then expected<result>

Examples:

	a	b	c	result
	1	-10	9	"[-3, 1, 1, 3]"
	4	16	0	"[-2, 0, 2]"
	1	1	-2	"[-1, 1]"
	2	5	0	"[0]"
	5	-4	1	"[]"

Файл test.py:

```
from eq import get_roots
from behave import *

@given('{A} is a, b is {B}, c is {C}')
def step_impl(context, A, B, C):
    global a
    global b
    global c
    a = int(A)
    b = int(B)
    c1 = int(C)
    return True

@When("start")
def step_impl(context):
    global res
    res = get_roots(a, b, c1)
    if type(a) == float:
        return True
    else:
        return False

@Then("expected{result}")
def step_impl(context, result):
    try:
        assert (res == result)
        return True
    except:
        return False
```

Результат работы программы:

```
1 feature passed, 0 failed, 0 skipped  
5 scenarios passed, 0 failed, 0 skipped  
15 steps passed, 0 failed, 0 skipped, 0 undefined  
Took 0m0.007s
```