

**Московский государственный технический  
университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

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

Выполнил:

студент группы ИУ5-32Б  
Лялько Никита

Подпись и дата:

Проверил:

преподаватель каф. ИУ5  
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2022 г.

## Описание задания

### 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).

# Текст программы

## Файлы для BDD-тестирования

### bdd/features/lab\_1.feature

Feature: Program should be able to solve biquadratic equation

In order to make sure the program  
solves equations correctly I have the following  
test scenarios:

Scenario Outline: Test my function

Given I have the numbers <A>, <B> and <C>

When I solve the equation with those numbers

Then I expect to get <N> roots

Examples:

A	B	C	N
1	2	3	0
2	3	4	0
5	0	0	1
1	4	-5	2
2	-5	3	4
1	-25	0	3

### bdd/lab\_1/equation\_solver.py

```
import math

def getEquationRoots(A, B, C):
    result = []
    D = B*B - 4*A*C
    if A == 0:
        if B == 0:
            return []
        else:
            result = [-math.sqrt(math.abs(-C/B)), math.sqrt(math.abs(-C/B))]
    if D < 0:
        return []
    elif D == 0:
        try:
            X1 = -math.sqrt( (-B)/(2 * A) )
            X2 = math.sqrt( (-B)/(2*A) )
        except:
            return []
        result = list(set([X1, X2]))
    else:
```

```

firstPair = True
secondPair = True
try:
    X1 = -math.sqrt((-B + math.sqrt(D))/(2 * A))
    X2 = math.sqrt((-B + math.sqrt(D))/(2 * A))
except:
    firstPair = False
try:
    X3 = -math.sqrt((-B - math.sqrt(D))/(2 * A))
    X4 = math.sqrt((-B - math.sqrt(D))/(2 * A))
except:
    secondPair = False

if firstPair:
    result.append(X1)
    result.append(X2)
if secondPair:
    result.append(X3)
    result.append(X4)
result = list(set(result))
return sorted(result)

```

### bdd/radish/steps.py

```
# -*- coding: utf-8 -*-
```

```

from radish import given, when, then
from sys import path

```

```

path.append('.')
from lab_1.equation_solver import getEquationRoots

```

```
@given("I have the numbers {A:g}, {B:g} and {C:g}")
```

```

def have_numbers(step, A, B, C):
    step.context.A = A
    step.context.B = B
    step.context.C = C

```

```
@when("I solve the equation with those numbers")
```

```

def sum_numbers(step):
    step.context.N = len(getEquationRoots( \
        step.context.A, step.context.B, step.context.C))

```

```
@then("I expect to get {N:g} roots")
```

```

def expect_result(step, N):
    assert step.context.N == N

```

## Файлы для TDD тестирования:

**tdd/lab\_1.py**

(Идентично bdd/lab\_1/equation\_solver.py)

**tdd/tdd.py**

```
from lab_1 import getEquationRoots

def tests_get_roots_zero():
    temp = getEquationRoots(1, 2, 3)
    assert len(temp) == 0
    temp = getEquationRoots(2, 3, 4)
    assert len(temp) == 0

def tests_get_roots_one():
    temp = getEquationRoots(5, 0, 0)
    assert temp == [0]

def tests_get_roots_two():
    temp = getEquationRoots(1, 4, -5)
    assert temp == [-1, 1]

def tests_get_roots_three():
    temp = getEquationRoots(1, -25, 0)
    assert temp == [-5, 0, 5]
    temp = getEquationRoots(1, -9, 0)
    assert temp == [-3, 0, 3]

def tests_get_roots_four():
    temp = getEquationRoots(1, -10, 9)
    assert temp == [-3, -1, 1, 3]
```

## Экранные формы

### BDD тестирование:

```
ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    ТЕРМИНАЛ    JUPYTER    КОНСОЛЬ ОТЛАДКИ

PS C:\Users\Dmitriy\Documents\study\BKIT\lab\5\bdd> radish -b radish features/
Feature: Program should be able to solve biquadratic equation # features/lab_1.feature
  In order to make sure the program
  solves equations correctly I have the following
  test scenarios:

  Scenario Outline: Test my function
    Given I have the numbers <A>, <B> and <C>
    When I solve the equation with those numbers
    Then I expect to get <N> roots

  Examples:
  | A | B | C | N |
  | 1 | 2 | 3 | 0 |
  | 2 | 3 | 4 | 0 |
  | 5 | 0 | 0 | 1 |
  | 1 | 4 | -5 | 2 |
  | 2 | -5 | 3 | 4 |
  | 1 | -25 | 0 | 3 |

  1 features (1 passed)
  6 scenarios (6 passed)
  18 steps (18 passed)
  Run 1670276272 finished within a moment
PS C:\Users\Dmitriy\Documents\study\BKIT\lab\5\bdd>
```

### TDD тестирование:

```
ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    ТЕРМИНАЛ    JUPYTER    КОНСОЛЬ ОТЛАДКИ    + v powershell

PS C:\Users\Dmitriy\Documents\study\BKIT\lab\5\tdd> pytest -v tdd.py
===== test session starts =====
platform win32 -- Python 3.9.13, pytest-7.2.0, pluggy-1.0.0 -- C:\Users\Dmitriy\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundat
ion.Python.3.9_qbz5n2kfra8p0\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Dmitriy\Documents\study\BKIT\lab\5\tdd
plugins: bdd-6.1.1
collected 5 items

tdd.py::tests_get_roots_zero PASSED [ 20%]
tdd.py::tests_get_roots_one PASSED [ 40%]
tdd.py::tests_get_roots_two PASSED [ 60%]
tdd.py::tests_get_roots_three PASSED [ 80%]
tdd.py::tests_get_roots_four PASSED [100%]

===== 5 passed in 0.05s =====
PS C:\Users\Dmitriy\Documents\study\BKIT\lab\5\tdd>
```