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).

Код программы.

bisqur.py

```
import sys

def get_coef(index, prompt):
    coef_str = sys.argv[index]; g=True
    while g:
        try:
            coef = float(coef_str)
            g=False
        except:
            print(prompt)
            coef_str = input()
    print(coef)
    return coef

class BiSqUr:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def get_disc(self):
        return self.b*self.b - 4*self.a*self.c

    def get_base_solve(self):
        d = self.get_disc()
        print(d)
        if d<0: return -1,-1
        return (-self.b-d**0.5)/2, (-self.b+d**0.5)/2

    def get_solve(self):
        a, b = self.get_base_solve()
        print(a,b)
        ans = []
        if a>0: ans += [-a**0.5, a**0.5]
        if b>0: ans += [-b**0.5, b**0.5]
        if len(ans) == 0: return "no solve"
        return ans
```

bisqur_test.py (TDD)

```
import unittest
from lab_python_fp.bisqur import BiSqUr
```

```
class BiSqUrTest_1(unittest.TestCase):
    def setUp(self):
        self.bisqur = BiSqUr(1, -5, 4)

    def test_get_disc(self):
        self.assertEqual(self.bisqur.get_disc(), 9)

    def test_get_base_solve(self):
        self.assertEqual(sorted(self.bisqur.get_base_solve()), [1, 4])

    def test_get_solve(self):
        self.assertEqual(sorted(self.bisqur.get_solve()), [-2.0, -1.0, 1.0,
2.0])

class BiSqUrTest_2(unittest.TestCase):
    def setUp(self):
        self.bisqur = BiSqUr(1, 5, 4)

    def test_get_disc(self):
        self.assertEqual(self.bisqur.get_disc(), 9)

    def test_get_base_solve(self):
        self.assertEqual(sorted(self.bisqur.get_base_solve()), [-4, -1])

    def test_get_solve(self):
        self.assertEqual(self.bisqur.get_solve(), "no solution")


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

bisqur_test.feature (BDD)

```
Feature: test class BiSqUr

  Scenario: test get_disc
    Given exemple of class BiSqUr with in: 1 -5 4
    Then function get_disc called with output is 9

  Scenario: test get_base_solve
    Given exemple of class BiSqUr with in: 1 -5 4
    Then function get_base_solve called with output is 9

  Scenario: test get_solve
    Given exemple of class BiSqUr with in: 1 -5 4
    Then function get_solve called with output is 9
```

bisqur_test.py (BDD)

```
from behave import *

from lab_python_fp.bisqur import BiSqUr


@given("exemple of class BiSqUr with in: 1 -5 4")
def prepair(context):
    global ur
    ur = BiSqUr(1, -5, 4)


@then('function get_disc called with output is 9')
def get_disc_out_correct(context):
```

```
    assert ur.get_disc() == 9

@then('function get_base_solve called with output is 9')
def get_disc_out_correct(context):
    assert sorted(ur.get_base_solve()) == [1, 4]

@then('function get_solve called with output is 9')
def get_disc_out_correct(context):
    assert sorted(ur.get_solve()) == [-2.0, -1.0, 1.0, 2.0]
```

Результаты.