**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**Московский государственный технический университет**

**им. Н.Э. Баумана**

**(МГТУ им. Н.Э. Баумана)**

**Кафедра «Системы обработки информации и управления» (ИУ5)**

Отчёт по лабораторной работе № 5

По курсу: «Базовые компоненты интернет-технологий»

Выполнил: Никулин Данила Дмитриевич

студент группы ИУ5-31Б.

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_.\_\_\_.2022г.

Подпись: \_\_\_\_\_\_\_\_\_\_\_.

г. Москва 2022 г.

**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).

**Приложение 1. Текст программы:**

**test\_TDD.py**

import pytest

from main import get\_roots

def tests\_get\_roots\_zero():

temp = get\_roots(1, 12, 36)

assert len(temp) == 0

temp = get\_roots(6, 60, 54)

assert len(temp) == 0

temp = get\_roots(3, 31, 56)

assert len(temp) == 0

def tests\_get\_roots\_one():

temp = get\_roots(1, 1, 0)

assert temp == {0}

temp = get\_roots(5, 15, 0)

assert temp == {0}

temp = get\_roots(30, 18, 0)

assert temp == {0}

def tests\_get\_roots\_two():

temp = get\_roots(3, -5, -28)

assert temp == {2, -2}

temp = get\_roots(3, -14, -117)

assert temp == {3, -3}

temp = get\_roots(11, -86, -117)

assert temp == {3, -3}

def tests\_get\_roots\_three():

temp = get\_roots(1, -9, 0)

assert temp == {-3, 0, 3}

temp = get\_roots(3, -75, 0)

assert temp == {-5, 0, 5}

temp = get\_roots(7, -112, 0)

assert temp == {-4, 0, 4}

def tests\_get\_roots\_four():

temp = get\_roots(7, -287, 2800)

assert temp == {-5, -4, 4, 5}

temp = get\_roots(13, -689, 2548)

assert temp == {-7, -2, 2, 7}

temp = get\_roots(1, -73, 576)

assert temp == {-8, -3, 3, 8}

**test\_BDD.py**

from main import get\_roots

from pytest\_bdd import scenarios, given, when, then, parsers

scenarios("test\_equation.feature")

@given(parsers.parse("The A coefficient {A:d}"), target\_fixture = "coefA")

def t\_root\_input\_1(A):

return A

@given(parsers.parse('The B coefficient {B:d}'), target\_fixture = "coefB")

def t\_root\_input\_2(B):

print("=================")

print(B)

return B

@given(parsers.parse('The C coefficient {C:d}'), target\_fixture = "coefC")

def t\_root\_input\_3(C):

return C

@when(parsers.parse('Solve the equation'), target\_fixture = "equ")

def t\_root\_solve(coefA, coefB, coefC):

return get\_roots(coefA, coefB, coefC)

@then(parsers.parse("I get {zero:d} roots"))

def t\_then(equ, zero):

assert len(equ) == zero

**test\_equation.feature**

Feature:

Scenario Outline

This app solve biquatratic equation

Scenario Outline: Solve the equation with correct value

Given The A coefficient <A>

And The B coefficient <B>

And The C coefficient <C>

When Solve the equation

Then I get <D> roots

Examples:

|A | B| C|D|

| 1| 12| 36|0|

| 6| 60| 54|0|

| 3| 31| 56|0|

| 1| 1| 0|1|

| 5| 15| 0|1|

|30| 18| 0|1|

| 3| -5| -28|2|

| 3| -14|-117|2|

|11| -86|-117|2|

| 1| -9| 0|3|

| 3| -75| 0|3|

| 7|-112| 0|3|

| 7|-287|2800|4|

|13|-689|2548|4|

| 1| -73| 576|4|

**main.py**

import sys

import math

def get\_coef(index, prompt):

try:

# Пробуем прочитать коэффициент из командной строки

coef\_str = sys.argv[index]

except:

# Вводим с клавиатуры

print(prompt)

coef\_str = input()

flag = False

# Проверка на число

while (flag == False):

try:

# Пробуем перевести строку в действительное число

coef = float(coef\_str)

except:

# При ошибке просим повторить ввод коэффициента

print(prompt)

coef\_str = input()

else:

flag = True

return coef

def get\_roots(a, b, c):

result = []

D = b \* b - 4 \* a \* c

if D == 0.0:

root = -b / (2.0 \* a)

result.append(root)

elif D > 0.0:

sqD = math.sqrt(D)

quadratic1 = (-b + sqD) / (2.0 \* a)

quadratic2 = (-b - sqD) / (2.0 \* a)

# y = x^2 проверяем игрек на положительность

if (quadratic1 >= 0):

root1 = math.sqrt(quadratic1)

root2 = -root1

result.append(root1)

if (root1 != root2):

result.append(root2)

if (quadratic2 >= 0):

root3 = math.sqrt(quadratic2)

root4 = -root3

result.append(root3)

if (root3 != root4):

result.append(root4)

return result

def main():

a = get\_coef(1, 'Введите коэффициент А:')

b = get\_coef(2, 'Введите коэффициент B:')

c = get\_coef(3, 'Введите коэффициент C:')

roots = get\_roots(a, b, c)

len\_roots = len(roots)

if len\_roots == 0:

print('Действительных корней нет')

elif len\_roots == 1:

print(f'Один корень: {roots[0]}')

elif len\_roots == 2:

print('Два корня: {} и {}'.format(roots[0], roots[1]))

elif len\_roots == 3:

print('Три корня: {} и {} и {}'.format(roots[0], roots[1], roots[2]))

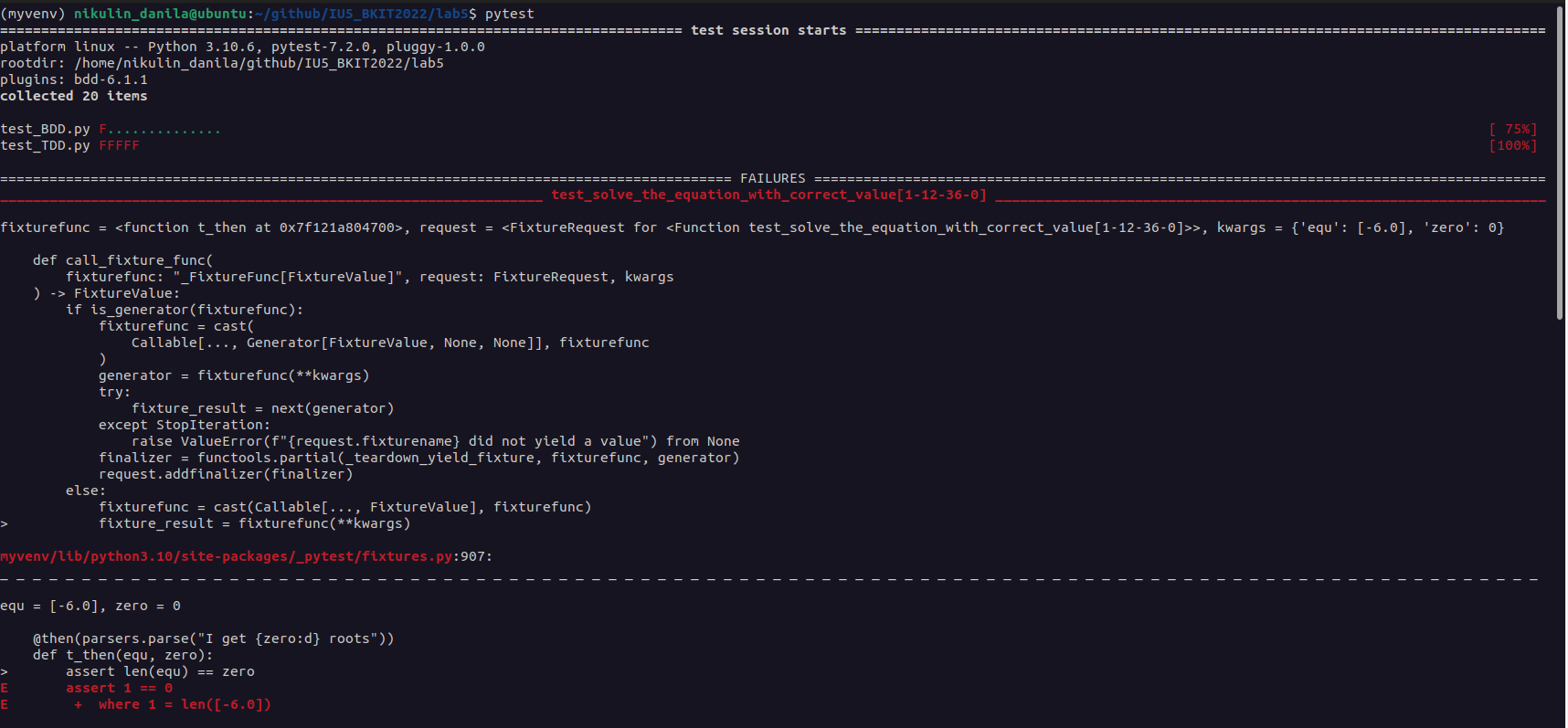
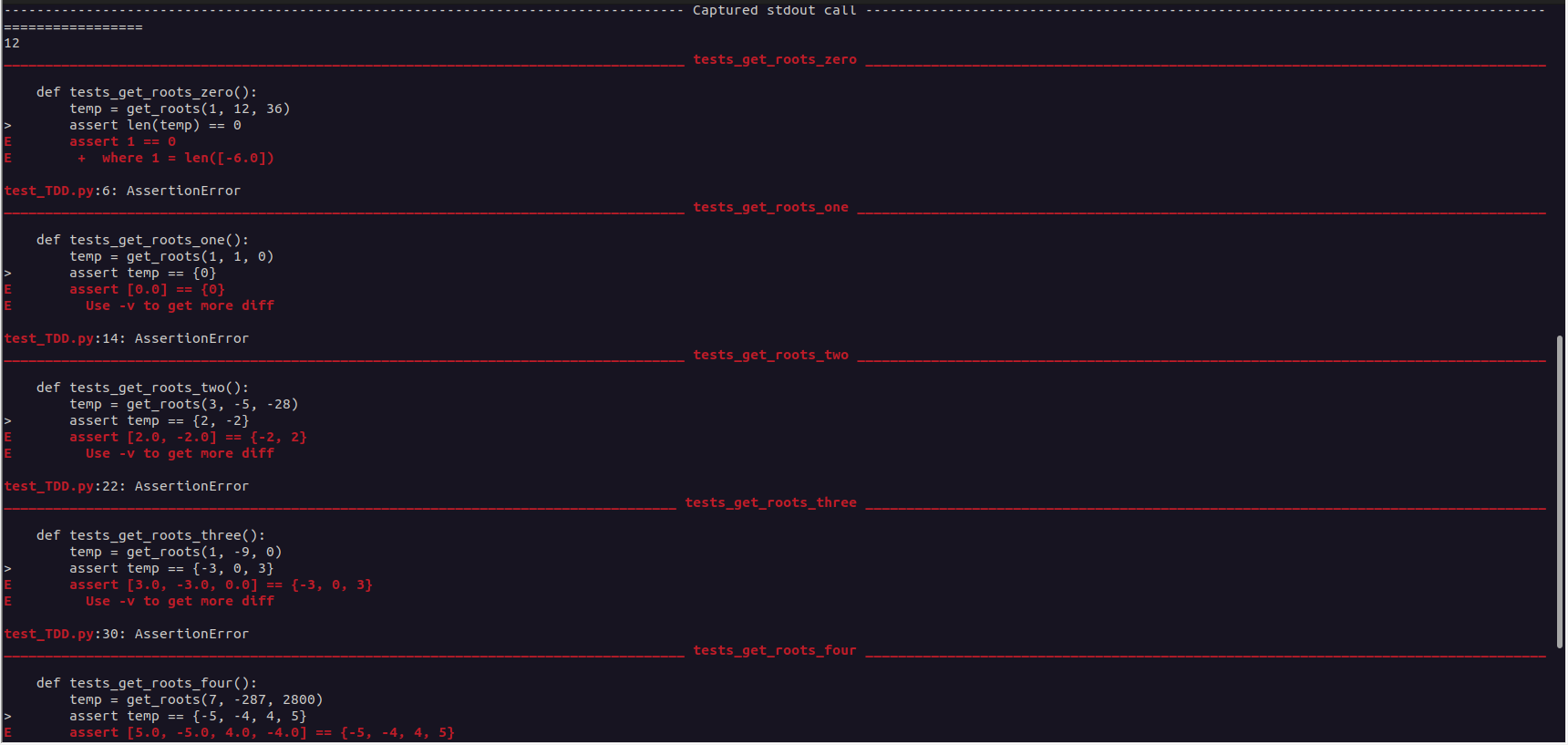
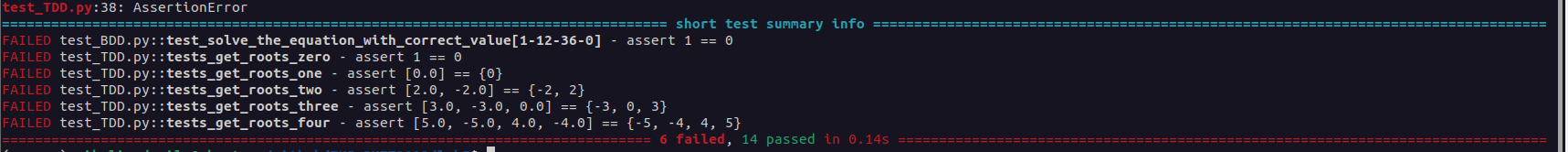
else:

print('Четыре корня: {} и {} и {} и {}'.format(roots[0], roots[1], roots[2],

roots[3]))

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Приложение 2. Результаты тестирования:**