**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №4

«Шаблоны проектирования и модульное тестирование в Python.

»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-34Б |  |  |
| Свечникова Анна |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021г.

**Постановка задачи**

### Задание:

1. Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один структурный и один поведенческий. В качестве справочника шаблонов можно использовать [следующий каталог.](https://refactoring.guru/ru/design-patterns/catalog) Для сдачи лабораторной работы в минимальном варианте достаточно реализовать один паттерн.
2. Вместо реализации паттерна Вы можете написать тесты для своей программы решения биквадратного уравнения. В этом случае, возможно, Вам потребуется доработать программу решения биквадратного уравнения, чтобы она была пригодна для модульного тестирования.
3. В модульных тестах необходимо применить следующие технологии:
   * TDD - фреймворк.
   * BDD - фреймворк.
   * Создание Mock-объектов.

**Текст программы:**

**lab4.py**

import sys

import math

def get\_coef\_kb():

'''ввод коэффициента с клавиатуры'''

while(True):

coef\_str = input()

try:

# Переводим строку в действительное число

coef = float(coef\_str)

except:

print('коэффициент введен неверно, попробуйте ещё раз')

else:

return coef

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 коэффицента

Returns:

float: Коэффициент квадратного уравнения

'''

try:

# Пробуем прочитать коэффициент из командной строки

coef\_str = sys.argv[index]

coef = float(coef\_str)

except:

# Вводим с клавиатуры

print(prompt)

coef = get\_coef\_kb()

finally:

return coef

def discr(a, b, c):

return b\*b - 4\*a\*c

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

'''

Вычисление корней биквадратного уравнения

Args:

a (float): коэффициент А

b (float): коэффициент B

c (float): коэффициент C

Returns:

list[float]: Список корней

'''

result = set() #храним корни в множестве, чтобы не было повторяющихся

D = discr(a, b, c)

if D < 0:

#нет корней

return set()

else:

root = math.sqrt(D)

try:

res1 = (-b + root)/(2\*a)

res2 = (-b - root)/(2\*a)

except ArithmeticError:

#возможно, имеет место квадратное уравнение bx^2 + c = 0

try:

res1 = math.sqrt(-c/b)

result.add(-res1)

result.add(res1)

except ArithmeticError:

# нет корней

return set()

if(res1 >= 0):

result.add(-math.sqrt(res1))

result.add(math.sqrt(res1))

if(res2 >= 0):

result.add(-math.sqrt(res2))

result.add(math.sqrt(res2))

return result

def main():

'''

Основная функция

'''

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

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

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

# Вычисление корней

roots = list(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(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]))

elif len\_roots == 4:

print('Четыре корня: {} и {} и {} и {}'.format(roots[0], roots[1], roots[2], roots[3]))

# Если сценарий запущен из командной строки

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

main()

# Пример запуска

# lab1.py 1 0 -4

**tdd.py**

from lab4 import \*

import unittest

from unittest.mock import patch

class TddTest(unittest.TestCase):

#предположим, решение в случае, когда уравнение не имеет корней, ещё не реализовано

@patch('lab4.get\_roots', return\_value = set())

def test\_no\_roots(self, get\_roots):

self.assertEqual(get\_roots(1, 0, 1), set())

#даже если функция вернет другой результат, из-за mock тест всё равно будет считаться пройденным

"""@patch('lab4.get\_roots', return\_value = set())

def test\_no\_roots(self, get\_roots):

self.assertEqual(get\_roots(1, 0, -16), set())"""

#обычный тест для случая отсутствия корней

""" def test\_no\_roots(self):

self.assertEqual(get\_roots(1, 0, 10), set())"""

def test\_one\_root(self):

self.assertEqual(get\_roots(1, 0, 0), set([0]))

def test\_two\_roots(self):

self.assertEqual(get\_roots(1, 0, -16), set([2, -2]))

def test\_three\_roots(self):

self.assertEqual(get\_roots(-4, 16, 0), set([0, -2, 2]))

def test\_four\_roots(self):

self.assertEqual(get\_roots(1, -5, 4), set([1, -1, 2, -2]))

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

**eq\_solver.feature**

Feature: solving a biquadrate equation

Scenario: One root

Given I have the coefs 1 and 0 and 0

When I solve equation

Then I expect the result to be 0

Scenario: Two roots

Given I have the coefs 1 and 0 and -16

When I solve equation

Then I expect the result to be 2, -2

Scenario: Three roots

Given I have the coefs -4 and 16 and 0

When I solve equation

Then I expect the result to be 2, -2, 0

**step.py**

from behave import given, when, then

from lab4 import get\_roots

@given ("I have the coefs {a:g} and {b:g} and {c:g}")

def have\_coefs(context, a, b, c):

context.a = a

context.b = b

context.c = c

@when("I solve equation")

def solve\_eq(context):

context.result = get\_roots(context.a, context.b, context.c)

@then("I expect the result to be {result}")

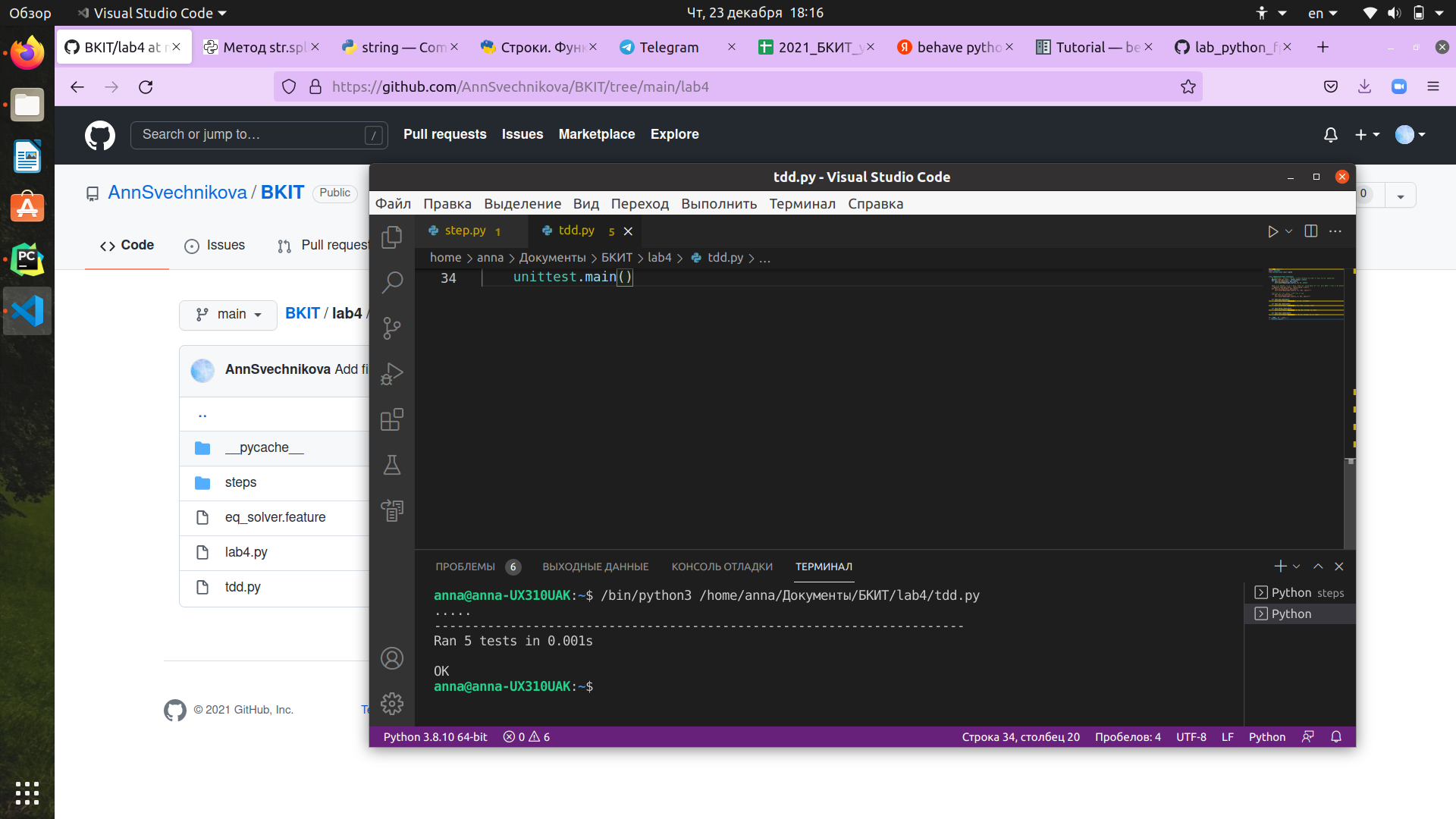
def expect\_result(context, result):

context.result == result

**Примеры выполнения**

Результаты прохождения тестов:

TDD



BDD

