**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: |  | Проверил: |
| студент группы ИУ5-35Б |  | преподаватель каф. ИУ5 |
| Самойлов Константин |  | Нардид А.Н. |
| Подпись и дата: |  | 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**

**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()  
 *# Обрабатываем неправильный ввод* while True:  
 try:  
 coef = float(coef\_str)  
 except:  
 print("Введены неправильные данные.", prompt)  
 coef\_str = input()  
 else:  
 return coef  
  
  
def get\_roots(a, b, c):  
 result = []  
 D = b \* b - 4 \* a \* c  
 if D == 0.0:  
 root = -b / (2.0 \* a)  
 if root >= 0:  
 result.append(math.sqrt(root))  
 result.append(-math.sqrt(root))  
 elif D > 0.0:  
 sqD = math.sqrt(D)  
 root1 = (-b + sqD) / (2.0 \* a)  
 if root1 >= 0:  
 if root1 == 0:  
 result.append(root1)  
 else:  
 result.append(math.sqrt(root1))  
 result.append(-math.sqrt(root1))  
 root2 = (-b - sqD) / (2.0 \* a)  
 if root2 >= 0:  
 if root2 == 0.0:  
 result.append(root2)  
 else:  
 result.append(math.sqrt(root2))  
 result.append(-math.sqrt(root2))  
 result = set(result)  
 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:')  
 *# Вычисление корней* roots = get\_roots(a, b, c)  
 *# Вывод корней* len\_roots = len(roots)  
 if len\_roots == 0:  
 print('Нет корней', end=" ")  
 return  
 elif len\_roots == 1:  
 print('Один корень:', end=" ")  
 elif len\_roots == 2:  
 print('Два корня:', end=" ")  
 elif len\_roots == 3:  
 print('Три корня:', end=" ")  
 else:  
 print('Четыре корня:')  
 print(\*roots, sep=", ")  
  
  
*# Если сценарий запущен из командной строки*if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**Файлы для BDD-тестирования**

**test\_BDD.py**

from main import get\_roots  
from pytest\_bdd import scenarios, given, when, then, parsers  
  
scenarios("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):  
 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

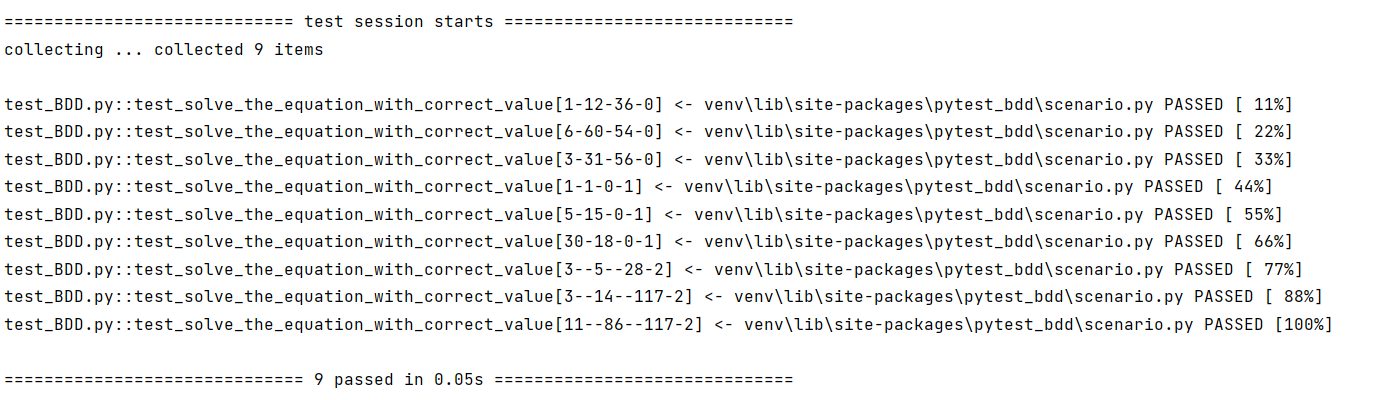
**Файлы для TDD тестирования:**

**test\_TDD.py**

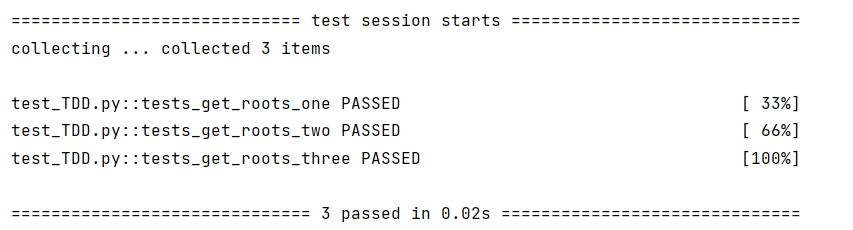
import pytest  
from main import get\_roots  
  
def tests\_get\_roots\_one():  
 temp = get\_roots(1, 1, 0)  
 assert temp == {0}  
 temp = get\_roots(5, 15, 0)  
 assert temp == {0}  
 temp = get\_roots(30, 18, 0)  
 assert temp == {0}  
  
def tests\_get\_roots\_two():  
 temp = get\_roots(3, -5, -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}

**Экранные формы**

**BDD тестирование:**

****

**TDD тестирование:**

****