Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждениевысшего образования «Московский государственный технический университетимени Н.Э. Баумана (национальный исследовательский университет МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления» Кафедра

«Системы обработки информации и управления»

Лабораторная работа №5

«Модульное тестирование в Python»

по предмету

«Базовые компоненты интернет-технологий»

Выполнил:

студент группы № ИУ5-31Б

Михалёв Ярослав

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Юрий

2022 г.

Постановка задачи

- Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
- Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
- 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).

Задание

В качестве объекта для тестирования была выбрана программа, находящая корни биквадратного уравнения

Текст программы

equation.py

```
import math
import sys


def calculate(A, B, C):
  if type(A) not in [int, float]:
    raise TypeError("Коэффициент A должен быть положительным вещественным числом!")
  if type(B) not in [int, float]:
    raise TypeError("Коэффициент B должен быть неотрицательным вещественным числом!")
  if type(C) not in [int, float]:
    raise TypeError("Коэффициент C должен быть неотрицательным вещественным числом!")

  if A == 0:
    raise ValueError("Коэффициент A должен быть положительным вещественным числом!")

  D = B * B - 4 * A * C

  if D > 0:
    t = (-B - math.sqrt(D)) / (2 * A)
    if t > 0:
      x1 = math.sqrt((-B + math.sqrt(D)) / (2 * A))
      x2 = -x1
      x3 = math.sqrt((-B - math.sqrt(D)) / (2 * A))
      x4 = -x3
      return tuple(sorted(set([x1, x2, x3, x4])))
  elif D == 0:
    x1 = math.sqrt(-B / 2 * A)
    x2 = -x1
    return tuple(sorted(set([x1, x2])))

  return ()
```

```
def main():
  A = 1
  B = 1
  C = 1
  try:
    A = float(sys.argv[1])
    B = float(sys.argv[2])
    C = float(sys.argv[3])
  except Exception as e:
    print("Не удалось прочитать коэффициенты!")
    while True:
      try:
        A = float(input("Введите коэффициент A\n> "))
        if A != 0:
          break
        else:
          print("Коэффициент A не может равняться нулю")
      except Exception as e:
        print("Коэффициент A введен некорректно!")
        pass
    while True:
      try:
        B = float(input("Введите коэффициент B\n> "))
        break
      except Exception as e:
        print("Коэффициент B введен некорректно!")
        pass
    while True:
      try:
        C = float(input("Введите коэффициент C\n> "))
        break
      except Exception as e:
        print("Коэффициент C введен некорректно!")
        pass

  roots = calculate(A, B, C)
  if (len(roots)) > 0:
    print("Корни:")
    for root in roots:
      print(root + ", ")
  else:
    print("Корней нет!")

if __name__ == "__main__":
  main()
```

test_equation.py

```
import unittest
from equation import calculate

class TestEquation(unittest.TestCase):

  def test_calculate(self):
    self.assertEqual(calculate(1, -10, 9), (-3, -1, 1, 3))
    self.assertEqual(calculate(-4, 16, 0), (-2, 0, 2))
    self.assertEqual(calculate(431, -123, 665), ())

  def test_value(self):
    with self.assertRaises(ValueError) as e:
      calculate(0, 33, 9)

  def test_type(self):
    with (self.assertRaises(TypeError)) as e:
      calculate(12, "B", 4)

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

tests.py

```
from behave import  Given, When, Then
from equation import  calculate

@Given("equation with coef A {A} B {B} C {C}")
def given_increment(context, A: {str}, B: {str}, C: {str}):
  context.A = int(A)
  context.B = int(B)
  context.C = int(C)

@When("we calculate {object}")
def given_increment(context, object: {str}):
  context.results = calculate(context.A, context.B, context.C)

@Then("we should see root1 {root1} root2 {root2} root3 {root3} root4 {root4}")
def then_results(context, root1: {str}, root2: {str}, root3: {str}, root4: {str}):
  if (root1 == "empty"):
    assert (context.results == ())
  elif (root2 == "empty"):
    assert(context.results == (int(root1)))
  elif (root3 == "empty"):
    assert (context.results == (int(root1), int(root2)))
  elif (root4 == "empty"):
    assert (context.results == (int(root1), int(root2), int(root3)))
  else:
    assert(context.results == (int(root1), int(root2), int(root3), int(root4)))
```

tests.feature

```
Feature: Testing the Equation
 Scenario: Test calculate 4 roots
  Given equation with coef A 1 B -10 C 9
  When we calculate roots
  Then we should see root1 -3 root2 -1 root3 1 root4 3

 Scenario: Test calculate 3 roots
  Given equation with coef A -4 B 16 C 0
  When we calculate roots
  Then we should see root1 -2 root2 0 root3 2 root4 empty

 Scenario: Test calculate 0 roots
  Given equation with coef A 431 B -123 C 665
  When we calculate roots
  Then we should see root1 empty root2 empty root3 empty root4 empty
```

Анализ результатов

В качестве TDD – фреймворка был использован пакет unittest

```
D:\GitHub\IU5_BKIT2022\Лабораторные работы\5. Модульное тестирование в Python>python test_equation.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

В качестве BDD – фреймворка был использован пакет behave

```
D:\GitHub\IU5_BKIT2022\Лабораторные работы\5. Модульное тестирование в Python>behave
Feature: Testing the Equation # features/steps/tests.feature:1

  Scenario: Test calculate 4 roots                      # features/steps/tests.feature:2
    Given equation with coef A 1 B -10 C 9              # features/steps/tests.py:4
    When we calculate roots                             # features/steps/tests.py:10
    Then we should see root1 -3 root2 -1 root3 1 root4 3 # features/steps/tests.py:14

  Scenario: Test calculate 3 roots                         # features/steps/tests.feature:7
    Given equation with coef A -4 B 16 C 0                 # features/steps/tests.py:4
    When we calculate roots                                # features/steps/tests.py:10
    Then we should see root1 -2 root2 0 root3 2 root4 empty # features/steps/tests.py:14

  Scenario: Test calculate 0 roots                                      # features/steps/tests.feature:12
    Given equation with coef A 431 B -123 C 665                         # features/steps/tests.py:4
    When we calculate roots                                             # features/steps/tests.py:10
    Then we should see root1 empty root2 empty root3 empty root4 empty # features/steps/tests.py:14

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.000s
```