

Факультет Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

**Отчет по лабораторной работе №4 по курсу
Базовые компоненты интернет-технологий**

" Шаблоны проектирования и модульное тестирование в Python "

5

(количество листов)

Вариант № 16

Исполнитель

студент группы РТ5-316

Нижаметдинов М.Ш.

“ 16 ” сентября 2021 г.

Проверил

Доцент кафедры ИУ5

Гапанюк Ю.Е.

“ ” 2021 г.

Москва, 2021 г.

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. 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 Моск-объектов.

Текст программы

Файл «main.py»

```
import sys
import math

# Функция проверки ввода на число
def is_number(str):
    try:
        # Если удастся преобразовать строку в число, функция возвращает значение
        [Истина]
        float(str)
        return True
    except ValueError:
        # Иначе - [Ложь]
        return False

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 коэффициента
    Returns:
        float: Коэффициент квадратного уравнения
    """
    try:
        # Пробуем прочитать коэффициент из командной строки
        coef_str = sys.argv[index]
    except:
        # Вводим с клавиатуры
        print(prompt)
        coef_str = input()
        # Программа будет запрашивать коэффициенты,
        # пока не будет введено число
    while not is_number(coef_str):
        print(prompt)
        coef_str = input()
```

```

# Переводим строку в действительное число
coef = float(coef_str)
return coef

def get_roots(a, b, c):
    """
    Вычисление корней квадратного уравнения
    Args:
        a (float): коэффициент А
        b (float): коэффициент В
        c (float): коэффициент С
    Returns:
        list[float]: Список корней
    """
    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)
        root1 = (-b + sqD) / (2.0 * a)
        root2 = (-b - sqD) / (2.0 * a)
        result.append(root1)
        result.append(root2)
    return result

def main():
    """
    Основная функция
    """
    a = get_coef(1, 'Введите коэффициент А:')
    b = get_coef(2, 'Введите коэффициент В:')
    c = get_coef(3, 'Введите коэффициент С:')
    # Вычисление корней
    roots = 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]))

# Если сценарий запущен из командной строки
if __name__ == "__main__":
    main()

# Пример запуска
# qr.py 1 0 -4

```

Файл «TEST_TDD.py»

```

import unittest
from main import get_roots

class TEST_TDD(unittest.TestCase):
    def test(self):
        self.assertEqual(get_roots(-4, 16, 0), [-0.0, 4.0])
        self.assertEqual(get_roots(1, 1, -2), [1.0, -2.0])
        self.assertEqual(get_roots(1, 1, 1), [])

```

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

Файл «FEATURE_BDD.feature»

```
Feature: Testing the function get_roots
  Scenario: Get roots of biquadratic equation for coefficients [-4, 16, 0]
    Given I put coefficients [-4, 16, 0] into the function
    Then I get roots [-0.0, 4.0]

  Scenario: Get roots of biquadratic equation for coefficients [1, 1, -2]
    Given I put coefficients [1, 1, -2] into the function
    Then I get roots [1.0, -2.0]

  Scenario: Get roots of biquadratic equation for coefficients [1, 1, 1]
    Given I put coefficients [1, 1, 1] into the function
    Then I get roots []
```

Файл «TEST_BDD.py»

```
from behave import given, then
from main import get_roots

@given('I put coefficients {coefficients} into the function')
def step_impl(context, coefficients: str):
    coefficients = list(map(int, coefficients.replace("[", "").replace("]", "",
    "").split(", ")))
    context.result = get_roots(coefficients[0], coefficients[1], coefficients[2])

@then('I get roots {result}')
def step_impl(context, result: str):
    if result != '[]':
        result = list(map(float, result.replace("[", "").replace("]", "",
        "").split(", ")))
        assert context.result == result
    else:
        assert context.result == []
```

Файл «TEST MOCK.py»

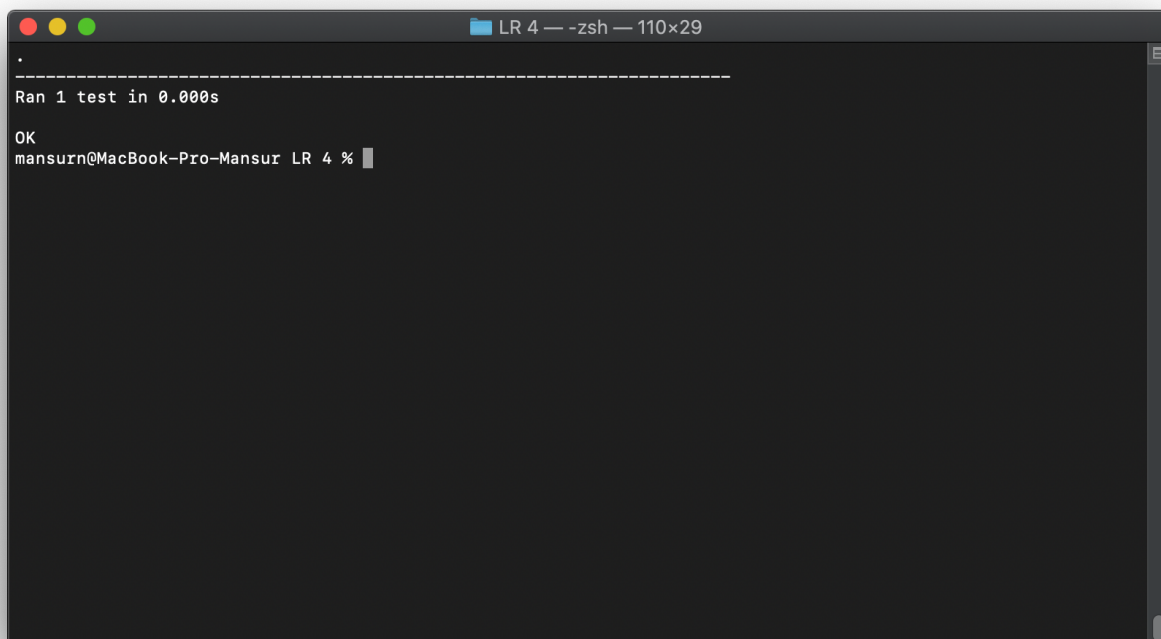
```
import unittest
from unittest.mock import Mock

from main import get_roots

class TEST MOCK(unittest.TestCase):
    def test(self):
        root_mock = Mock(return_value=5)
        get_roots(root_mock(), 5, 5)

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

Экранные формы с примерами выполнения программы



```
LR 4 — -zsh — 110x29
.  
-----  
Ran 1 test in 0.000s  
OK  
mansurn@MacBook-Pro-Mansur LR 4 %
```

The image shows a terminal window with a dark background. The title bar at the top indicates the window is titled "LR 4" and is running a "zsh" shell, with dimensions of "110x29". The terminal content shows a prompt ".", followed by a dashed line separator, then the output "Ran 1 test in 0.000s", then "OK", and finally the shell prompt "mansurn@MacBook-Pro-Mansur LR 4 %" with a cursor.