

**Московский государственный технический  
университет им. Н.Э. Баумана**

Факультет «Радиотехнический»  
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчёт по лабораторной работе №4  
«Шаблоны проектирования и модульное тестирование в Python»

Выполнил:

студент группы РТ5-31Б  
Петров Егор

Подпись и дата:

Проверил:

преподаватель каф. ИУ5  
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2021 г

## Описание задания

1. Необходимо для произвольной предметной области реализовать от одного до трех шаблонов проектирования: один порождающий, один структурный и один поведенческий. 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

## main.py:

```
import sys
import math

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()
    # Переводим строку в действительное число
    # Проводим проверку на корректность ввода

    try:
        coef = float(coef_str)
    except ValueError:
        coef = input_validation(prompt)
    return coef

def get_roots(a, b, c):
    """
    Вычисление корней квадратного уравнения
    Args:
        a (float): коэффициент A
        b (float): коэффициент B
        c (float): коэффициент C
    Returns:
        list[float]: Список корней
    """
    result = []
    D = b * b - 4 * a * c
    if D == 0.0:
        root = -b / (2.0 * a)
        if root > 0.0:
            roota = -(math.sqrt(root))
            rootb = math.sqrt(root)
            result.append(roota)
            result.append(rootb)
        elif root == 0.0:
            result.append(root)
    elif D > 0.0:
        sqD = math.sqrt(D)
        root1 = (-b + sqD) / (2.0 * a)
        root2 = (-b - sqD) / (2.0 * a)
        if root1 > 0.0:
            root1a = -(math.sqrt(root1))
            root1b = math.sqrt(root1)
            result.append(root1a)
            result.append(root1b)
        elif root1 == 0.0:
            result.append(root1)
```

```

        if root2 > 0.0:
            root2a = -(math.sqrt(root2))
            root2b = math.sqrt(root2)
            result.append(root2a)
            result.append(root2b)
        elif root2 == 0.0:
            result.append(root2)
    result.sort()
    return result

def main():
    """
    Основная функция
    """
    print('Программа для вычисления корней биквадратного уравнения')
    a = get_coef(1, 'Введите коэффициент A:')
    b = get_coef(2, 'Введите коэффициент B:')
    c = get_coef(3, 'Введите коэффициент C:')
    # Вычисление корней
    roots = get_roots(a, b, c)
    roots.sort()
    # 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 {}'.format(roots[0], roots[1], roots[2]))
    elif len_roots == 4:
        print('Четыре корня: {}, {}, {} и {}'.format(roots[0], roots[1], roots[2], roots[3]))

def input_validation(prompt):
    """
    Функция для обработки исключения
    """
    while True:
        print("Ошибка ввода!")
        print(prompt)
        new_coef_str = input()
        try:
            new_coef = float(new_coef_str)
            break
        except ValueError:
            continue
    return new_coef

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

# Пример запуска
# qr.py 1 0 -4

```

## TDDtest.py:

```
import unittest

from main import get_roots

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

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

## BDDfeat.feature:

```
Feature: testing the function get_roots
    Scenario: get roots of biquadratic equation for coef. [-4, 16, 0]
        Given I put coefficients [-4, 16, 0] into the function
        Then I get roots [-2.0, 0.0, 2.0]

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

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

## BDDtest.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 == []
```

## **MOCKtest.py:**

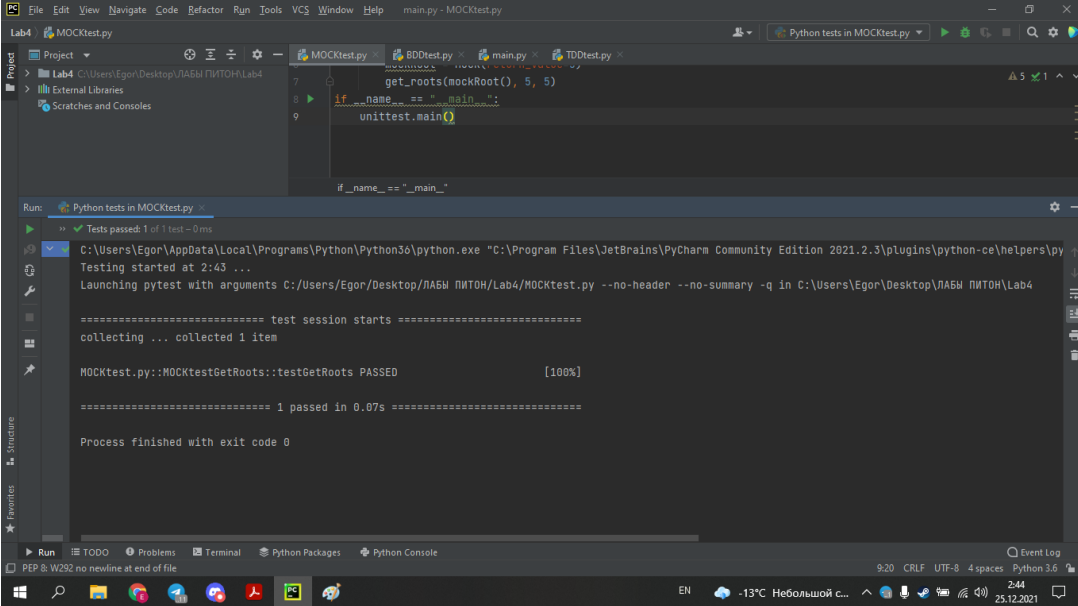
```
import unittest
from unittest.mock import Mock

from main import get_roots

class MOCKtestGetRoots(unittest.TestCase):
    def testGetRoots(self):
        mockRoot = Mock(return_value=5)
        get_roots(mockRoot(), 5, 5)

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

# Примеры работы программы



This screenshot shows the PyCharm IDE with the file `main.py - MOCKtest.py` open. The code defines a `get_roots` function and a `testGetRoots` method. The Run window at the bottom shows the test execution results:

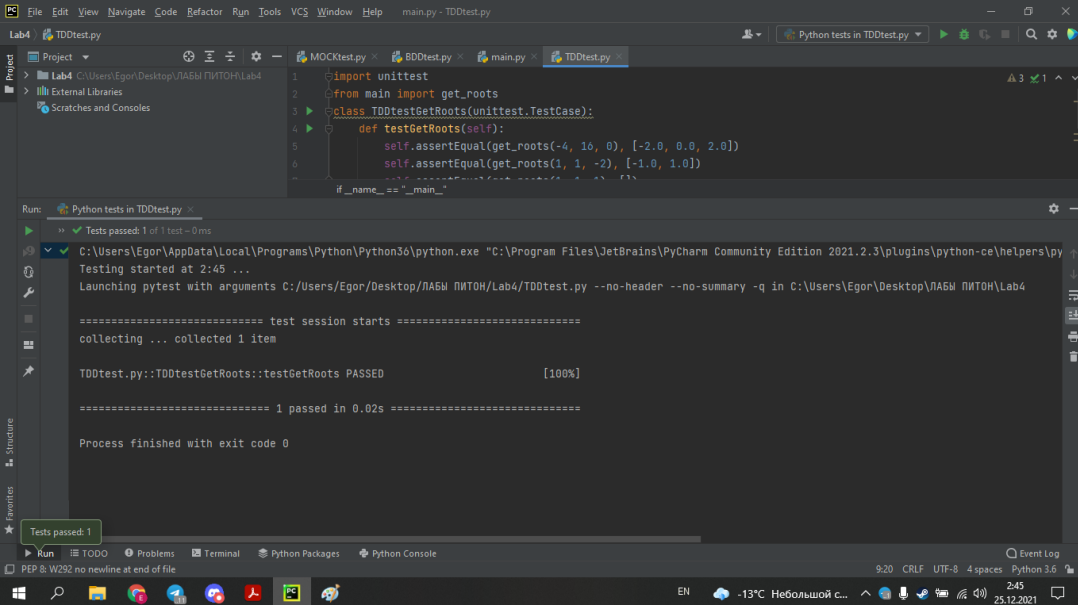
```
Run: Python tests in MOCKtest.py
Tests passed: 1 of 1 test - 0 ms
C:\Users\Egor\AppData\Local\Programs\Python\Python36\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2021.2.3\plugins\python-ce\helpers\py
Testing started at 2:43 ...
Launching pytest with arguments C:\Users\Egor\Desktop\ЛАБЫ ПИТОН\Lab4\MOCKtest.py --no-header --no-summary -q in C:\Users\Egor\Desktop\ЛАБЫ ПИТОН\Lab4

===== test session starts =====
collecting ... collected 1 item

MOCKtest.py::MOCKtestGetRoots::testGetRoots PASSED [100%]

===== 1 passed in 0.07s =====

Process finished with exit code 0
```



This screenshot shows the PyCharm IDE with the file `main.py - TDDtest.py` open. The code defines a `TDDtestGetRoots` class with a `testGetRoots` method. The Run window at the bottom shows the test execution results:

```
Run: Python tests in TDDtest.py
Tests passed: 1 of 1 test - 0 ms
C:\Users\Egor\AppData\Local\Programs\Python\Python36\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2021.2.3\plugins\python-ce\helpers\py
Testing started at 2:45 ...
Launching pytest with arguments C:\Users\Egor\Desktop\ЛАБЫ ПИТОН\Lab4\TDDtest.py --no-header --no-summary -q in C:\Users\Egor\Desktop\ЛАБЫ ПИТОН\Lab4

===== test session starts =====
collecting ... collected 1 item

TDDtest.py::TDDtestGetRoots::testGetRoots PASSED [100%]

===== 1 passed in 0.02s =====

Process finished with exit code 0
```