

**федеральное государственное автономное образовательное  
учреждение высшего образования**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет информационных технологий**

*Кафедра Информатики и информационных технологий*

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 6**

**Дисциплина:** Тестирование программного обеспечения

**Тема:** Написание модульных тестов (Unit Tests)

**Выполнил(а): студент(ка) группы 221-3711**

Морозов К.А.  
(Фамилия И.О.)

**Дата, подпись** \_\_\_\_\_  
(Дата) \_\_\_\_\_  
(Подпись) \_\_\_\_\_

**Проверил:** \_\_\_\_\_  
(Фамилия И.О., степень, звание) \_\_\_\_\_  
(Оценка) \_\_\_\_\_

**Дата, подпись** \_\_\_\_\_  
(Дата) \_\_\_\_\_  
(Подпись) \_\_\_\_\_

**Замечания:** \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

**Москва 2025**

# **Лабораторная работа 6 – Написание модульных тестов (Unit Tests)**

## **Цель работы**

Цель: Понять принципы unit-тестирования и научиться тестировать изолированные функции и методы.

## **Задание**

Объект тестирования: Класс или набор функций с бизнес-логикой (например, класс `Calculator` с методами `add`, `divide`, `is\_prime\_number`).

Задание (Автоматическое тестирование):

1. Используя фреймворк `unittest` или `pytest`, напишите модульные тесты для всех методов.
2. Используйте технику параметризации для тестирования с разными наборами данных.
3. Для метода `divide` убедитесь, что тест проверяет возникновение исключения (exception) при делении на ноль.

## Практическая часть

Как и было предложено в задании лабораторной работы, я сделал калькулятор с пятью функциями (плюс, минус, умножить, разделить и проверка на простое число):

```
class Calculator:
    @staticmethod
    def add(a, b):
        return a + b

    @staticmethod
    def subtract(a, b):
        return a - b

    @staticmethod
    def multiply(a, b):
        return a * b

    @staticmethod
    def divide(a, b):
        if b == 0:
            raise ZeroDivisionError("division by zero")
        return a / b

    @staticmethod
    def is_prime_number(n: int) -> bool:
        if n <= 1:
            return False
        if n == 2 or n == 3:
            return True
        if n % 2 == 0 or n % 3 == 0:
            return False
        i = 5
        while i * i <= n:
            if n % i == 0 or n % (i + 2) == 0:
                return False
            i += 6
        return True
```

Далее с помощью pytest я написал к нему несколько unit-тестов с разными ожидаемыми и подставляемыми значениями:

```
class TestCalculator:  
    @pytest.mark.parametrize(  
        "a,b,expected",  
        [  
            (2, 3, 5),  
            (-2, 5, 3),  
            (0, 0, 0),  
            (1.5, 2.5, 4.0)  
        ]  
    )  
    def test_add(self, a, b, expected):  
        assert Calculator.add(a, b) == expected  
  
    @pytest.mark.parametrize(  
        "a,b,expected",  
        [  
            (10, 4, 6),  
            (-1, -1, 0),  
            (3.5, 1.2, 2.3)  
        ]  
    )  
    def test_subtract(self, a, b, expected):  
        result = Calculator.subtract(a, b)  
        if any(isinstance(x, float) for x in (a, b, expected)):  
            assert result == pytest.approx(expected)  
        else:  
            assert result == expected  
  
    @pytest.mark.parametrize(  
        "a,b,expected",  
        [  
            (7, 6, 42),  
            (0, 100, 0),  
            (-3, 5, -15),  
            (2.5, 4, 10.0)  
        ]  
    )  
    def test_multiply(self, a, b, expected):  
        result = Calculator.multiply(a, b)  
        if any(isinstance(x, float) for x in (a, b, expected)):  
            assert result == pytest.approx(expected)  
        else:  
            assert result == expected
```

```

@pytest.mark.parametrize(
    "a,b,expected",
    [
        (9, 3, 3),
        (7, 2, 3.5),
        (-10, 2, -5),
        (5.0, 2.0, 2.5)
    ]
)
def test_divide_ok(self, a, b, expected):
    result = Calculator.divide(a, b)
    assert result == pytest.approx(expected)

def test_divide_by_zero_raises(self):
    with pytest.raises(ZeroDivisionError):
        Calculator.divide(10, 0)

@pytest.mark.parametrize("n", [2, 3, 5, 7, 11, 97, 9973])
def test_is_prime_true(self, n):
    assert Calculator.is_prime_number(n) is True

@pytest.mark.parametrize("n", [-10, -1, 0, 1, 4, 6, 9, 100, 10000])
def test_is_prime_false(self, n):
    assert Calculator.is_prime_number(n) is False

```

Все юнит-тесты выполнены успешно

```

PS C:\Users\mrjes\Desktop\ТПО\tro\Тестирование\б\Testing_6_TPO_Morozov\test6> pytest -q
.....
32 passed in 0.03s
PS C:\Users\mrjes\Desktop\ТПО\tro\Тестирование\б\Testing_6_TPO_Morozov\test6>

```

Ссылка на github с кодом лабораторной:

Ссылка на яндекс диск с видео-защитой:

[https://disk.yandex.ru/d/Jg3tB\\_oH4RDkOA](https://disk.yandex.ru/d/Jg3tB_oH4RDkOA)