**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №5

«Модульное тестирование в Python»

|  |  |  |
| --- | --- | --- |
| Выполнил:  Каженец Д.Н.  ИУ5-31Б |  | Проверил:  Гапанюк Е.Ю. |
| Подпись и дата: |  | Подпись и дата: |

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).

TDD - фреймворк

import unittest

from main import Rectangle, Circle, Square

from math import pi

class TestGeometricShapes(unittest.TestCase):

def test\_rectangle\_area(self):

rect = Rectangle(8, 4, 'синий')

self.assertEqual(rect.area(), 32)

def test\_rectangle\_repr(self):

rect = Rectangle(8, 4, 'синий')

expected = "Название: Прямоугольник, Ширина: 8, Высота: 4, Цвет: синий, Площадь: 32"

self.assertEqual(repr(rect), expected)

def test\_circle\_area(self):

circ = Circle(3, 'зеленый')

self.assertAlmostEqual(circ.area(), pi \* 3\*\*2, places=2)

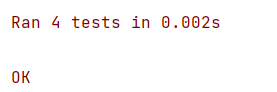
def test\_square\_name(self):

sqr = Square(5, 'красный')

self.assertEqual(sqr.name(), "Квадрат")

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()



BDD - фреймворк

from behave import given, when, then

from main import Rectangle, Circle, Square

import math

@given('a rectangle with width {width:d} and height {height:d}')

def step\_given\_rectangle(context, width, height):

context.rectangle = Rectangle(width, height, 'синий')

@when('I calculate the area')

def step\_when\_calculate\_area(context):

context.area = context.rectangle.area()

@then('the area should be {expected\_area:d}')

def step\_then\_area\_should\_be(context, expected\_area):

assert context.area == expected\_area, f"Expected {expected\_area}, but got {context.area}"

@given('a square with side {side:d}')

def step\_given\_square(context, side):

context.square = Square(side, 'красный')

@then('the name should be "{expected\_name}"')

def step\_then\_square\_name\_should\_be(context, expected\_name):

assert context.square.name() == expected\_name, f"Expected {expected\_name}, but got {context.square.name()}"

@given('I have a circle with radius {radius:d} and color "{color}"')

def step\_create\_circle(context, radius, color):

context.circle = Circle(radius, color)

@when('I calculate the area of the circle')

def step\_calculate\_area(context):

context.area = context.circle.area()

@then('the area should be approximately {expected:f}')

def step\_check\_area(context, expected):

assert math.isclose(context.area, expected, rel\_tol=1e-2)

