МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа №4. “Написание Unit-тестов”

По дисциплине «Инструментальные средства разработки ПО»

Выполнил студент группы №
M6666
Гений, миллиардер, плэйбой

Проверил:
Филантроп и просто хороший

САНКТ-ПЕТЕРБУРГ

2023

1. Цели и задачи тестирования

Цель тестирования:

- Проверить функциональность программы «Математические формулы»

Задачи тестирования:

- Проверить основные функции
- Идентифицировать и зарегистрировать дефекты, если таковые обнаружатся.
- Удостовериться, что продукт проходить все написанные тесты

2. Описание тестируемого продукта

«Математические формулы» - продукт вычисляет периметр и площадь трех фигур: круга, прямоугольника, квадрата и треугольника

Требования к продукту

- Корректный вывод результата
- При подстановке заведомо неверных значений – выбросить исключение

3. Область тестирования

Тестирование будет охватывать следующие компоненты:

- Модуль circle
- Модуль triangle
- Модуль rectangle
- Модуль square

4. Стратегия тестирования

Стратегия тестирования включает в себя следующие методы и техники:

- Функциональное тестирование

5. Критерии приемки

Для успешного завершения тестирования и приемки продукта должны быть выполнены следующие критерии:

- Все тесты прошли успешно.

- Зарегистрированные дефекты были исправлены или утверждены как неприменимые.

- Продукт соответствует установленным требованиям.

6. Ожидаемые результаты

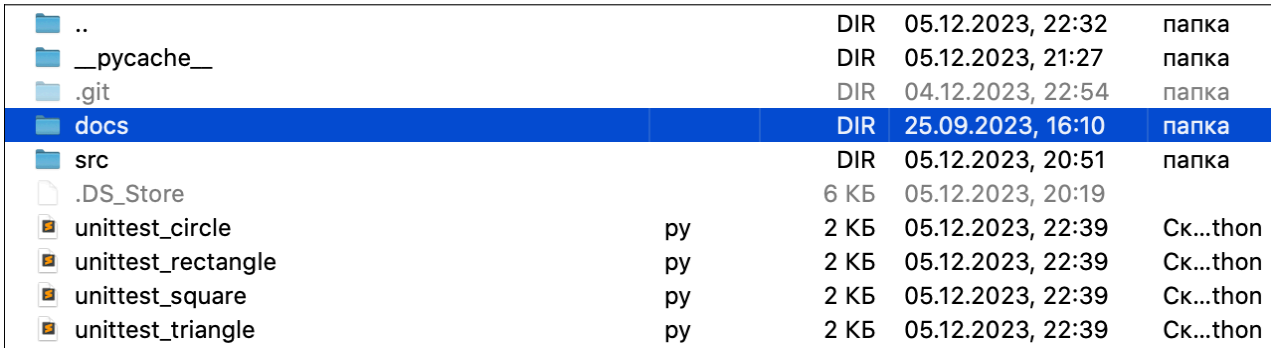
Ожидается предоставление следующей информации:

- Отчет о прохождении тестов.

- Статусы тестирования для каждого компонента.

Отчет

1. Создал 4 .py файла с unittest’ами для каждой фигуры:



..		DIR	05.12.2023, 22:32	папка
__pycache__		DIR	05.12.2023, 21:27	папка
.git		DIR	04.12.2023, 22:54	папка
docs		DIR	25.09.2023, 16:10	папка
src		DIR	05.12.2023, 20:51	папка
.DS_Store		6 КБ	05.12.2023, 20:19	
unittest_circle	py	2 КБ	05.12.2023, 22:39	Ск...thon
unittest_rectangle	py	2 КБ	05.12.2023, 22:39	Ск...thon
unittest_square	py	2 КБ	05.12.2023, 22:39	Ск...thon
unittest_triangle	py	2 КБ	05.12.2023, 22:39	Ск...thon

```python
import unittest
import math

from src.circle import area, perimeter

class CircleAreaTestCases(unittest.TestCase):
    def test_circle_int_first(self):
        self.assertEqual(area(5), math.pi * 5 * 5)

    def test_circle_int_second(self):
        self.assertEqual(area(57285), math.pi * 57285 * 57285)

    def test_circle_string_first(self):
        with self.assertRaises(TypeError):
            area("85")

    def test_circle_string_second(self):
        with self.assertRaises(TypeError):
            area("ITMO")

    def test_circle_bool_first(self):
        with self.assertRaises(TypeError):
            area(True)

    def test_circle_bool_second(self):
        with self.assertRaises(TypeError):
            area(False)

    def test_circle_negative_int_first(self):
        with self.assertRaises(ValueError):
            area(-5)

    def test_circle_negative_int_second(self):
        with self.assertRaises(ValueError):
            area(-59853)

    def test_circle_zero_int(self):
        with self.assertRaises(ValueError):
            area(0)

class CirclePerimeterTestCases(unittest.TestCase):
```

```python
class CirclePerimeterTestCases(unittest.TestCase):
    def test_circle_int_first(self):
        self.assertEqual(perimeter(7), 2 * math.pi * 7)

    def test_circle_int_second(self):
        self.assertEqual(perimeter(57285), 2 * math.pi * 57285)

    def test_circle_string_first(self):
        with self.assertRaises(TypeError):
            perimeter("85")

    def test_circle_string_second(self):
        with self.assertRaises(TypeError):
            perimeter("ITMO")

    def test_circle_bool_first(self):
        with self.assertRaises(TypeError):
            perimeter(True)

    def test_circle_bool_second(self):
        with self.assertRaises(TypeError):
            perimeter(False)

    def test_circle_negative_int_first(self):
        with self.assertRaises(ValueError):
            perimeter(-5)

    def test_circle_negative_int_second(self):
        with self.assertRaises(ValueError):
            perimeter(-59853)

    def test_circle_zero_int(self):
        with self.assertRaises(ValueError):
            perimeter(0)


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

```python
import unittest
import math

from src.rectangle import area, perimeter

class RectangleAreaTestCases(unittest.TestCase):
    def test_rectangle_int_first(self):
        self.assertEqual(area(5, 19), 5 * 19)

    def test_rectangle_int_second(self):
        self.assertEqual(area(57285, 187533), 57285 * 187533)

    def test_rectangle_string_first(self):
        with self.assertRaises(TypeError):
            area("85", "92")

    def test_rectangle_string_second(self):
        with self.assertRaises(TypeError):
            area("85", 19)

    def test_rectangle_bool_first(self):
        with self.assertRaises(TypeError):
            area(True, True)

    def test_rectangle_bool_second(self):
        with self.assertRaises(TypeError):
            area(False, False)

    def test_rectangle_negative_int_first(self):
        with self.assertRaises(ValueError):
            area(-5, 8)

    def test_rectangle_negative_int_second(self):
        with self.assertRaises(ValueError):
            area(-59853, -15)

    def test_rectangle_zero_int(self):
        with self.assertRaises(ValueError):
            area(0, 8)

    def test_rectangle_zero_int(self):
        with self.assertRaises(ValueError):
            area(0, 0)
```

```python
class RectanglePerimeterTestCases(unittest.TestCase):
    def test_rectangle_int_first(self):
        self.assertEqual(perimeter(5, 9), 2 * (5 + 9))

    def test_rectangle_int_second(self):
        self.assertEqual(perimeter(57285, 394732), 2 * (57285 + 394732))

    def test_rectangle_string_first(self):
        with self.assertRaises(TypeError):
            perimeter("85", "92")

    def test_rectangle_string_second(self):
        with self.assertRaises(TypeError):
            perimeter("85", 19)

    def test_rectangle_bool_first(self):
        with self.assertRaises(TypeError):
            perimeter(True, True)

    def test_rectangle_bool_second(self):
        with self.assertRaises(TypeError):
            perimeter(False, False)

    def test_rectangle_negative_int_first(self):
        with self.assertRaises(ValueError):
            perimeter(-5, 8)

    def test_rectangle_negative_int_second(self):
        with self.assertRaises(ValueError):
            perimeter(-59853, -15)

    def test_rectangle_zero_int(self):
        with self.assertRaises(ValueError):
            perimeter(0, 8)

    def test_rectangle_zero_int(self):
        with self.assertRaises(ValueError):
            perimeter(0, 0)


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

```python
import unittest
import math

from src.square import area, perimeter

class SquareAreaTestCases(unittest.TestCase):
    def test_square_int_first(self):
        self.assertEqual(area(5), 5 * 5)

    def test_square_int_second(self):
        self.assertEqual(area(57285), 57285 * 57285)

    def test_square_string_first(self):
        with self.assertRaises(TypeError):
            area("85")

    def test_square_string_second(self):
        with self.assertRaises(TypeError):
            area("ITMO")

    def test_square_bool_first(self):
        with self.assertRaises(TypeError):
            area(True)

    def test_square_bool_second(self):
        with self.assertRaises(TypeError):
            area(False)

    def test_square_negative_int_first(self):
        with self.assertRaises(ValueError):
            area(-5)

    def test_square_negative_int_second(self):
        with self.assertRaises(ValueError):
            area(-59853)

    def test_square_zero_int(self):
        with self.assertRaises(ValueError):
            area(0)
```

```python
class SquarePerimeterTestCases(unittest.TestCase):
    def test_square_int_first(self):
        self.assertEqual(perimeter(9), 4 * 9)

    def test_square_int_second(self):
        self.assertEqual(perimeter(394732), 4 * 394732)

    def test_square_string_first(self):
        with self.assertRaises(TypeError):
            perimeter("85")

    def test_square_string_second(self):
        with self.assertRaises(TypeError):
            perimeter("ITMO")

    def test_square_bool_first(self):
        with self.assertRaises(TypeError):
            perimeter(True)

    def test_square_bool_second(self):
        with self.assertRaises(TypeError):
            perimeter(False)

    def test_square_negative_int_first(self):
        with self.assertRaises(ValueError):
            perimeter(-5)

    def test_square_negative_int_second(self):
        with self.assertRaises(ValueError):
            perimeter(-59853)

    def test_square_zero_int(self):
        with self.assertRaises(ValueError):
            perimeter(0)

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

```python
import unittest
import math

from src.triangle import area, perimeter

class TriangleAreaTestCases(unittest.TestCase):
    def test_triangle_int_first(self):
        self.assertEqual(area(5, 8), 5 * 8 / 2)

    def test_triangle_int_second(self):
        self.assertEqual(area(57285, 38752), 57285 * 38752 / 2)

    def test_triangle_string_first(self):
        with self.assertRaises(TypeError):
            area("85", 91)

    def test_triangle_string_second(self):
        with self.assertRaises(TypeError):
            area("ITMO", "kdjf")

    def test_triangle_bool_first(self):
        with self.assertRaises(TypeError):
            area(True, False)

    def test_triangle_bool_second(self):
        with self.assertRaises(TypeError):
            area(False)

    def test_triangle_negative_int_first(self):
        with self.assertRaises(ValueError):
            area(-5, 85)

    def test_triangle_negative_int_second(self):
        with self.assertRaises(ValueError):
            area(-59853, -158)

    def test_triangle_zero_int(self):
        with self.assertRaises(ValueError):
            area(0, 0)
```

```python
class TrianglePerimeterTestCases(unittest.TestCase):
    def test_triangle_int_first(self):
        self.assertEqual(perimeter(9, 8, 1), 9 + 8 + 1)

    def test_triangle_int_second(self):
        self.assertEqual(perimeter(394732, 1985321, 38529999), 394732 + 38529999 + 1985321)

    def test_triangle_string_first(self):
        with self.assertRaises(TypeError):
            perimeter("85", "35", "15")

    def test_triangle_string_second(self):
        with self.assertRaises(TypeError):
            perimeter("ITMO", "IS", "M3108")

    def test_triangle_bool_first(self):
        with self.assertRaises(TypeError):
            perimeter(True)

    def test_triangle_bool_second(self):
        with self.assertRaises(TypeError):
            perimeter(False)

    def test_triangle_negative_int_first(self):
        with self.assertRaises(ValueError):
            perimeter(-5, 158, 29)

    def test_triangle_negative_int_second(self):
        with self.assertRaises(ValueError):
            perimeter(-59853, -15, -25)

    def test_triangle_zero_int(self):
        with self.assertRaises(ValueError):
            perimeter(0, 19, 10)

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

2. Изменения в основных файлах:

```python
import math

def area(r):
    '''
    Возвращает площадь круга

        Параметры:
            r (int): радиус круга

        Возвращаемое значение:
            math.pi * r * r: искомая площадь круга
    '''

    if (type(r) is not int):
        raise TypeError("Аргумент должен быть int")

    if (r <= 0):
        raise ValueError("Аргумент должны быть больше ноля")

    return math.pi * r * r

def perimeter(r):
    '''
    Возвращает периметр круга

        Параметры:
            r (int): радиус круга

        Возвращаемое значение:
            2 * math.pi * r: искомый периметр круга
    '''

    if (type(r) is not int):
        raise TypeError("Аргумент должен быть int")

    if (r <= 0):
        raise ValueError("Аргумент должны быть больше ноля")

    return 2 * math.pi * r
```

```python
def area(a, b):
    '''
    Возвращает площадь прямоугольника

        Параметры:
            a (int): длина прямоугольника
            b (int): ширина прямоугольника

        Возвращаемое значение:
            a * b: искомая площадь прямоугольника
    '''

    if (type(a) is not int or type(b) is not int):
        raise TypeError("Аргументы должен быть int")

    if (a ⩽ 0 or b ⩽ 0):
        raise ValueError("Аргументы должны быть больше ноля")



    return a * b

def perimeter(a, b):
    '''
    Возвращает периметр прямоугольника

        Параметры:
            a (int): длина прямоугольника
            b (int): ширина прямоугольника

        Возвращаемое значение:
            a * b: искомый периметр прямоугольника
    '''

    if (type(a) is not int or type(b) is not int):
        raise TypeError("Аргументы должен быть int")

    if (a ⩽ 0 or b ⩽ 0):
        raise ValueError("Аргументы должны быть больше ноля")

    return 2*(a + b)
```

```python
def area(a):
    '''
    Возвращает площадь квадрата

        Параметры:
            a (int): длина квадрата
            b (int): ширина квадрата

        Возвращаемое значение:
            a * b: искомая площадь квадрата
    '''

    if (type(a) is not int):
        raise TypeError("Аргумент должен быть int")

    if (a ⩽ 0):
        raise ValueError("Аргумент должны быть больше ноля")

    return a * a


def perimeter(a):
    '''
    Возвращает периметр квадрата

        Параметры:
            a (int): длина квадрата
            b (int): ширина квадрата

        Возвращаемое значение:
            4 * a: искомый периметр квадрата
    '''

    if (type(a) is not int):
        raise TypeError("Аргумент должен быть int")

    if (a ⩽ 0):
        raise ValueError("Аргумент должны быть больше ноля")

    return 4 * a
```

```python
def area(a, h):
    '''
    Возвращает площадь треугольника

        Параметры:
            a (int): основание треугольника
            h (int): высота треугольника

        Возвращаемое значение:
            a * h / 2: искомая площадь треугольника
    '''

    if (type(a) is not int or type(h) is not int):
        raise TypeError("Аргументы должен быть int")

    if (a ⩽ 0 or h ⩽ 0):
        raise ValueError("Аргументы должны быть больше ноля")

    return a * h / 2

def perimeter(a, b, c):
    '''
    Возвращает периметр треугольника

        Параметры:
            a (int): одна сторона треугольника
            b (int): вторая сторона треугольника
            c (int): третья сторона треугольника

        Возвращаемое значение:
            a + b + c: искомый периметр треугольника
    '''

    if (type(a) is not int or type(b) is not int or type(c) is not int):
        raise TypeError("Аргументы должен быть int")

    if (a ⩽ 0 or b ⩽ 0 or c ⩽ 0):
        raise ValueError("Аргументы должны быть больше ноля")

    return a + b + c
```

3. Оформил таблицу с результатами unittest’ов:

Unit Tests

circle.py

Test name	Function	Input	Expected	Status
test_circle_int_first	area	5	78.53981633974483	OK
	perimeter	7	43.982297150257104	OK
test_circle_string_first	area	"ITMO"	"Аргумент должен быть int"	FAILED
	perimeter	"M3108"	"Аргумент должен быть int"	FAILED
test_circle_negative_int_first	area	-5	"Аргумент должен быть больше ноля"	FAILED
	perimeter	-7	"Аргумент должен быть больше ноля"	FAILED
test_circle_zero	area	0	"Аргумент должен быть больше ноля"	FAILED
	perimeter	0	"Аргумент должен быть больше ноля"	FAILED

rectangle.py

Test name	Function	Input	Expected	Status
test_rectangle_int_first	area	5, 19	95	OK
	perimeter	5, 9	28	OK
test_rectangle_string_first	area	"ITMO"	"Аргументы должен быть int"	FAILED
	perimeter	"M3108"	"Аргументы должен быть int"	FAILED
test_rectangle_negative_int_first	area	-5	"Аргументы должны быть больше ноля"	FAILED
	perimeter	-7	"Аргументы должны быть больше ноля"	FAILED
test_rectangle_zero	area	0	"Аргументы должны быть больше ноля"	FAILED
	perimeter	0	"Аргументы должны быть больше ноля"	FAILED

square.py

Test name	Function	Input	Expected	Status
test_square_int_first	area	5	25	OK
	perimeter	7	28	OK
test_square_string_first	area	"ITMO"	"Аргумент должен быть int"	FAILED
	perimeter	"M3108"	"Аргумент должен быть int"	FAILED
test_square_negative_int_first	area	-5	"Аргумент должен быть больше ноля"	FAILED
	perimeter	-7	"Аргумент должен быть больше ноля"	FAILED
test_square_zero	area	0	"Аргумент должен быть больше ноля"	FAILED
	perimeter	0	"Аргумент должен быть больше ноля"	FAILED

triangle.py

Test name	Function	Input	Expected	Status
test_triangle_int_first	area	5, 8	20	OK
	perimeter	5, 8, 2	15	OK
test_triangle_string_first	area	"ITMO"	"Аргументы должен быть int"	FAILED
	perimeter	"M3108"	"Аргументы должен быть int"	FAILED
test_triangle_negative_int_first	area	-5	"Аргументы должны быть больше ноля"	FAILED
	perimeter	-7	"Аргументы должны быть больше ноля"	FAILED
test_triangle_zero	area	0	"Аргументы должны быть больше ноля"	FAILED
	perimeter	0	"Аргументы должны быть больше ноля"	FAILED

4. Запушил в репозиторий:

```
MacBook-Pro-Dmitri:geometric_lib deus$ git add .
MacBook-Pro-Dmitri:geometric_lib deus$ git commit -m "Labwork 4 done"
[main c40038b] Labwork 4 done
 19 files changed, 451 insertions(+), 4 deletions(-)
 create mode 100644 .DS_Store
 create mode 100644 .gitignore
 create mode 100644 src/.DS_Store
 create mode 100644 src/__init__.py
 create mode 100644 src/__init__.pyc
 create mode 100644 src/__pycache__/__init__.cpython-37.pyc
 create mode 100644 src/__pycache__/circle.cpython-37.pyc
 create mode 100644 src/__pycache__/rectangle.cpython-37.pyc
 create mode 100644 src/__pycache__/square.cpython-37.pyc
 create mode 100644 src/__pycache__/triangle.cpython-37.pyc
 rename circle.py => src/circle.py (54%)
 rename rectangle.py => src/rectangle.py (59%)
 rename square.py => src/square.py (58%)
 rename triangle.py => src/triangle.py (60%)
 create mode 100644 unittest_circle.py
 create mode 100644 unittest_rectangle.py
 create mode 100644 unittest_square.py
 create mode 100644 unittest_triangle.py
MacBook-Pro-Dmitri:geometric_lib deus$ git push
Enter passphrase for key '/Users/deus/.ssh/id_rsa':
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 4 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (24/24), 6.80 KiB | 2.27 MiB/s, done.
Total 24 (delta 11), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (11/11), done.
To github.com:soilow/geometric_lib.git
   f0c441e..c40038b  main -> main
MacBook-Pro-Dmitri:geometric_lib deus$
```