ФЕДЕРАЛЬНОЕ АГЕНСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ Императора Александра I»

Кафедра «Информационные и вычислительные системы»

Дисциплина «Программирование на языках высокого уровня (Python)»

ОТЧЁТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

Выполнил студент Шефнер А.
Факультет: АИТ
Группа: ИВБ-211

Проверил: Баталов Д.И.

Санкт-Петербург

2023

Оценочный лист результатов ЛР № 4

Ф.И.О. студента _____Шефнер Альберт_____

Группа _____ИВБ-211_____

№ п/п	Материалы необходимые для оценки знаний, умений и навыков	Показатель оценивания	Критерии Оценивания	Шкала оценивания	Оценка
1	Лабораторная работа№	Соответствие методике выполнения	Соответствует	7	
			Не соответствует	0	
		Срок выполнения	Выполнена в срок	2	
			Выполнена с опозданием на 2 недели	0	
		оформление	Соответствует требованиям	1 0	
			Не соответствует		
	ИТОГО количество баллов			10	

Доцент кафедры

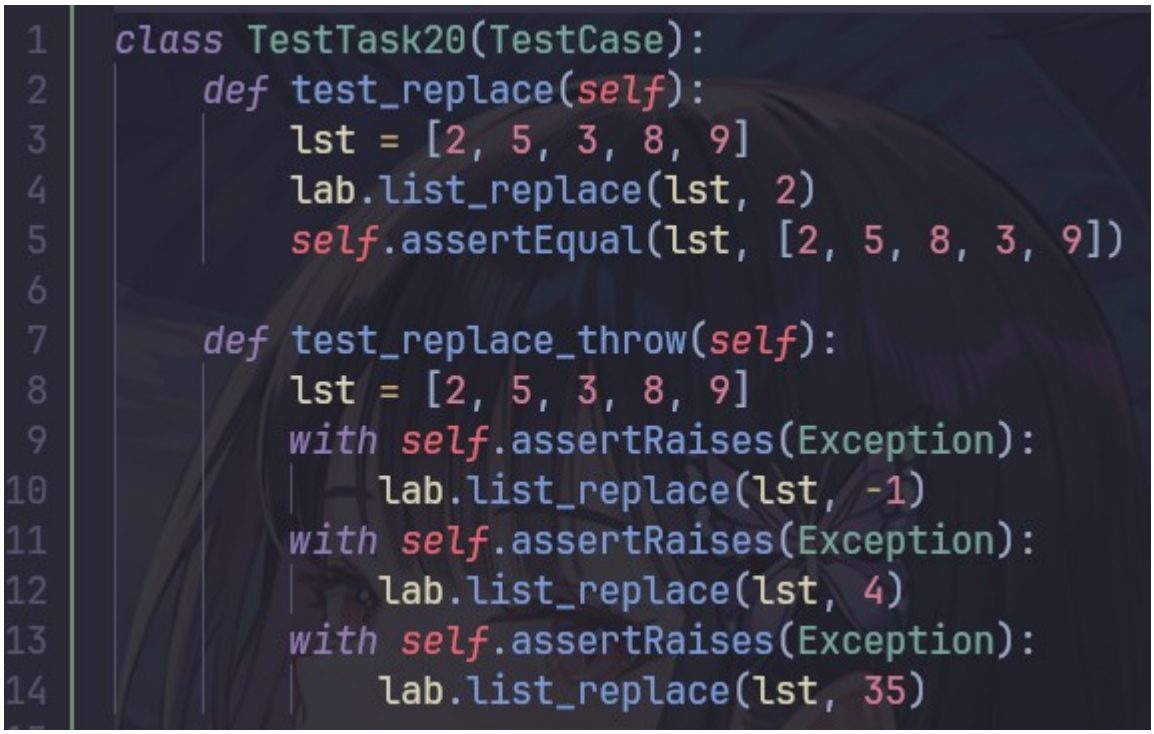
«Информационные и вычислительные

системы» Баталов Д.И. «__»

_____2023 г.

Тестирование

Для тестов используется модуль встроенной библиотеки unittest. Формат тестов везде одинаковы: сначала вызов функции или процедуры, потом сверка её результата с ожидаемым с помощью assert. Если assert имеет два параметра, тестируемое значение всегда слева, ожидаемое всегда справа. Вот пример одного из тестов:



```
class TestTask20(TestCase):
    def test_replace(self):
        lst = [2, 5, 3, 8, 9]
        lab.list_replace(lst, 2)
        self.assertEqual(lst, [2, 5, 8, 3, 9])

    def test_replace_throw(self):
        lst = [2, 5, 3, 8, 9]
        with self.assertRaises(Exception):
            lab.list_replace(lst, -1)
        with self.assertRaises(Exception):
            lab.list_replace(lst, 4)
        with self.assertRaises(Exception):
            lab.list_replace(lst, 35)
```

Программа успешно проходит все приведённые ниже тесты.

```
[albert@fedora Lab 4]$ python -m unittest test.py
...............
----------------------------------------------------------------------
Ran 15 tests in 0.001s

OK
```

Более подробные результаты:

```
[albert@fedora Lab 4]$ python -m unittest test.py --verbose
test_area_of_circle (test.TestArea.test_area_of_circle) ... ok
test_area_of_rectangle (test.TestArea.test_area_of_rectangle) ... ok
test_area_of_triangle (test.TestArea.test_area_of_triangle) ... ok
test_mul (test.TestArithmetic.test_mul) ... ok
test_sub (test.TestArithmetic.test_sub) ... ok
test_sum (test.TestArithmetic.test_sum) ... ok
test_dict_len (test.TestDicts.test_dict_len) ... ok
test_key_in (test.TestDicts.test_key_in) ... ok
test_get_set (test.TestEncapsulation.test_get_set) ... ok
test_set_raises (test.TestEncapsulation.test_set_raises) ... ok
test_in_list (test.TestLists.test_in_list) ... ok
test_list_count (test.TestLists.test_list_count) ... ok
test_is_palindrome (test.TestStrings.test_is_palindrome) ... ok
test_str_len (test.TestStrings.test_str_len) ... ok
test_str_lower (test.TestStrings.test_str_lower) ... ok

----------------------------------------------------------------------
Ran 15 tests in 0.001s

OK
```

1. Напишите модуль, содержащий функции, которые выполняют следующие арифметические операции: сложение, вычитание, умножение.

Решение:

test.py × arithmetic.py ×

```python
# 1. Напишите модуль, содержащий функции, которые выполняют следующие
# арифметические операции: сложение, вычитание, умножение.


def sum(a: float, b: float) -> float:
    return a + b

def sub(a: float, b: float) -> float:
    return a - b

def mul(a: float, b: float) -> float:
    return a * b
```

Тесты:

```python
class TestArithmetic(TestCase):
    def test_sum(self):
        a, b = 4, 7
        self.assertEqual(arithmetic.sum(a, b), a + b)

    def test_sub(self):
        a, b = 4, 7
        self.assertEqual(arithmetic.sub(a, b), a - b)

    def test_mul(self):
        a, b = 4, 7
        self.assertEqual(arithmetic.mul(a, b), a * b)
```

2. Напишите модуль, содержащий функции, которые выполняют следующие операции: проверка наличия элемента в списке, подсчет частоты вхождения элемента в список.

Решение:

test.py × arithmetic.py × lists.py ×

```python
# 2. Напишите модуль, содержащий функции, которые выполняют следующие
# операции: проверка наличия элемента в списке, подсчет частоты вхождения
# элемента в список.


def in_list(lst: list, elem) -> bool:
    return elem in lst

def list_count(lst: list, elem) -> int:
    return lst.count(elem)
```

Тесты:

```python
class TestLists(TestCase):
    def test_in_list(self):
        lst = [4, 6, 1, 2]
        self.assertTrue(lists.in_list(lst, 4))
        self.assertFalse(lists.in_list(lst, 9))

    def test_list_count(self):
        lst = [3, 2, 4, 0, 3, 1, 2, 3]
        result = lists.list_count(lst, 3)
        self.assertEqual(result, 3)
```

3. Напишите модуль, содержащий функции, которые выполняют следующие операции: проверку, является ли строка палиндромом, подсчет длины строки, перевод всех символов в нижний регистр.

Решение:

test.py × arithmetic.py × lists.py × strings.py ×

```python
# 3. Напишите модуль, содержащий функции, которые выполняют следующие
# операции: проверку, является ли строка палиндромом, подсчет длины строки,
# перевод всех символов в нижний регистр.

def is_palindrome(s: str) -> bool:
    return s == s[::-1]

def str_len(s: str) -> int:
    return len(s)

def str_lower(s: str) -> str:
    return s.lower()
```

Тесты:

```python
class TestStrings(TestCase):
    def test_is_palindrome(self):
        s1 = "шалаш"
        self.assertTrue(strings.is_palindrome(s1))

        s2 = "шашал"
        self.assertFalse(strings.is_palindrome(s2))

    def test_str_len(self):
        s = "Privet"
        result = strings.str_len(s)
        self.assertEqual(result, 6)

    def test_str_lower(self):
        s = "Privet"
        result = strings.str_lower(s)
        self.assertEqual(result, "privet")
```

4. Напишите модуль, содержащий функции, которые выполняют следующие операции: подсчет площади круга, прямоугольника и треугольника.

Решение:

test.py × area.py ×

```python
# 4. Напишите модуль, содержащий функции, которые выполняют следующие
# операции: подсчет площади круга, прямоугольника и треугольника.

import math

def area_of_circle(radius: float) -> float:
    return math.pi * radius * radius

def area_of_rectangle(side_a: float, side_b: float) -> float:
    return side_a * side_b

def area_of_triangle(side_a: float, side_b: float, side_c: float) -> float:
    halfperim = (side_a + side_b + side_c) / 2
    area = math.sqrt(
        halfperim * (halfperim - side_a) * (halfperim - side_b) * (halfperim - side_c)
    )

    return area
```

Тесты:

```python
class TestArea(TestCase):
    epsilon: float = 0.0001

    def test_area_of_circle(self):
        radius = 5
        result = area.area_of_circle(radius)
        self.assertLess(abs(78.53981634 - result), self.epsilon)

    def test_area_of_rectangle(self):
        a, b = 4, 6
        result = area.area_of_rectangle(a, b)
        self.assertEqual(result, 24)

    def test_area_of_triangle(self):
        a, b, c = 3, 4, 5
        result = area.area_of_triangle(a, b, c)
        self.assertLess(abs(a * b / 2 - result), self.epsilon)
```

5. Напишите модуль, содержащий функции, которые выполняют следующие операции: подсчет количества элементов в словаре, проверку на наличие ключа в словаре.

Решение:

```python
# 5. Напишите модуль, содержащий функции, которые выполняют следующие
# операции: подсчет количества элементов в словаре, проверку на наличие ключа в
# словаре.

def dict_len(dct: dict) -> int:
    return len(dct)

def key_in(dct: dict, key) -> bool:
    return key in dct.keys()
```

Тесты:

```python
class TestDicts(TestCase):
    def test_dict_len(self):
        dct = {1: 4, 2: 5, 7: 5}
        result = dicts.dict_len(dct)
        self.assertEqual(result, 3)

    def test_key_in(self):
        dct = {1: 4, 2: 5, 7: 5}
        self.assertTrue(dicts.key_in(dct, 1))
        self.assertFalse(dicts.key_in(dct, 3))
```

6. Напишите модуль, содержащий внутренние имена, значения которых можно получить через функции верхнего уровня модуля.

Решение:

test.py × encapsulation.py ×

```python
# 6. Напишите модуль, содержащий внутренние имена, значения которых можно
# получить через функции верхнего уровня модуля.

__secret__: str = "X452lu"


def get_secret() -> str:
    global __secret__
    return __secret__


def set_secret(new: str) -> None:
    if not isinstance(new, str):
        raise ValueError("New secret must be 'str'")

    global __secret__
    __secret__ = new
```

Тесты:

```python
class TestEncapsulation(TestCase):
    def test_get_set(self):
        new = "xd"
        encapsulation.set_secret(new)
        self.assertEqual(encapsulation.get_secret(), new)

    def test_set_raises(self):
        new = 5
        with self.assertRaises(ValueError):
            encapsulation.set_secret(new) # type: ignore
```