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 № 3

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

Проверил: Баталов Д.И.

Санкт-Петербург

2023

Оценочный лист результатов ЛР № 3

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

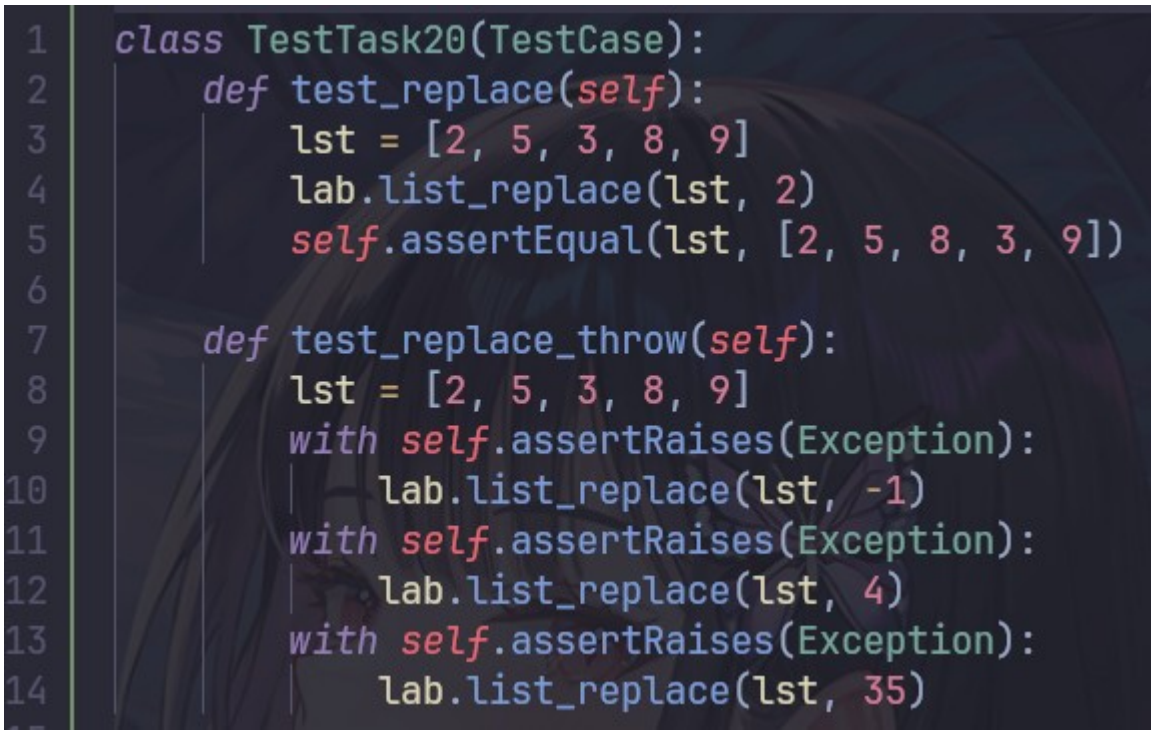
«Информационные и вычислительные

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@ryzenpc Lab 3]$ python -m unittest test.py
..................
----------------------------------------------------------------------
Ran 18 tests in 0.001s

OK
```

Более подробные результаты:

```
[albert@ryzenpc Lab 3]$ python -m unittest test.py --verbose
test_count_letters (test.LabTest.test_count_letters) ... ok
test_distinct (test.LabTest.test_distinct) ... ok
test_even_indices (test.LabTest.test_even_indices) ... ok
test_factorial (test.LabTest.test_factorial) ... ok
test_grange (test.LabTest.test_grange) ... ok
test_grange_func (test.LabTest.test_grange_func) ... ok
test_grange_step (test.LabTest.test_grange_step) ... ok
test_in_range (test.LabTest.test_in_range) ... ok
test_is_palindrome (test.LabTest.test_is_palindrome) ... ok
test_log_args (test.LabTest.test_log_args) ... ok
test_max_num (test.LabTest.test_max_num) ... ok
test_product (test.LabTest.test_product) ... ok
test_square (test.LabTest.test_square) ... ok
test_square_dec (test.LabTest.test_square_dec) ... ok
test_str_reverse (test.LabTest.test_str_reverse) ... ok
test_sum_nums (test.LabTest.test_sum_nums) ... ok
test_upper (test.LabTest.test_upper) ... ok
test_wrap_in_b_tag (test.LabTest.test_wrap_in_b_tag) ... ok

----------------------------------------------------------------------
Ran 18 tests in 0.001s

OK
```

0. Подготовка

```python
from collections.abc import Callable, Iterable
from functools import partial, reduce
from typing import OrderedDict


def compose2(f, g):
    return lambda *a, **kw: f(g(*a, **kw))


def compose(*functions):
    return reduce(compose2, functions)

```

1. Напишите функцию, вычисляющую максимальное из трех чисел.

Решение:

```python
max_num = max
```

Тесты:

```python
    def test_max_num(self):
        a, b, c = 5, 9, 2
        result = lab.max_num(a, b, c)
        self.assertEqual(result, 9)
```

2. Напишите функцию, которая возвращает сумму элементов списка.

Решение:

```python
sum_nums = sum
```

Тесты:

```python
    def test_sum_nums(self):
        lst = [4, 6, 2, 9]
        result = lab.sum_nums(lst)
        self.assertEqual(result, 21)
```

3. Напишите функцию, которая возвращает произведение элементов списка.

Решение:

```
def product2(a: float, b: float) -> float: return a * b

product = partial(reduce, product2)
```

Тесты:

```
    def test_product(self):
        lst = [4, 6, 2, 9]
        result = lab.product(lst)
        self.assertEqual(result, 432)
```

4. Напишите функцию, которая возвращает инвертированную строку, подаваемую ей на вход.

Решение:

```
def str_reverse(s: str) -> str:
    return s[::-1]
```

Тесты:

```
    def test_str_reverse(self):
        s = "Привет всем!"
        result = lab.str_reverse(s)
        self.assertEqual(result, "!месв тевирП")
```

5. Напишите функцию для вычисления факториала задаваемого числа.

Решение:

```
factorial = compose(
    product,
    partial(range, 1),
    lambda n: n + 1 if n > 0 else 2,
)
```

Тесты:

```python
    def test_factorial(self):
        result1 = lab.factorial(6)
        self.assertEqual(result1, 720)

        result2 = lab.factorial(0)
        self.assertEqual(result2, 1)
```

6. Напишите функцию, которая проверяет, входит ли задаваемое значение в определенный диапазон или нет.

Решение:

```python
def in_range(num: float, start: float, end: float) -> bool:
    return start <= num <= end
```

Тесты:

```python
    def test_in_range(self):
        num1, start1, end1 = 40, 35, 69
        self.assertTrue(lab.in_range(num1, start1, end1))

        num2, start2, end2 = 70, 35, 69
        self.assertFalse(lab.in_range(num2, start2, end2))
```

7. Напишите функцию, которая подсчитывает количество элементов в нижнем и верхнем регистрах у входной строки.

Решение:

```python
count_letters = compose(
    sum,
    lambda s: (1 if c.islower() or c.isupper() else 0 for c in s)
)
```

Тесты:

```python
    def test_count_letters(self):
        s = "Где-то буквы, где-то знаки препинания, где-то цифры 44234."
        result = lab.count_letters(s)
        self.assertEqual(result, 40)
```

8. Напишите функцию, удаляющую повторяющиеся элементы в списке.

Решение:

```
distinct = compose(list, OrderedDict.fromkeys)
```

Тесты:

```
    def test_task_8(self):
        matrix = lab.task_8(3)
        self.assertEqual(matrix, [[1, 0, 0], [0, 1, 0], [0, 0, 1]])
```

9. Напишите функцию, выводящую элементы списка с четным индексом.

Решение:

```
def even_indices(iterable: list) -> list:
    return iterable[::2]
```

Тесты:

```
    def test_even_indices(self):
        lst = [1, 0, 1, 4, 5, 0, 1, 2, 4, 9]
        result = lab.even_indices(lst)
        self.assertEqual(result, [1, 1, 5, 1, 4])
```

10. Напишите функцию, которая проверяет, является ли строка палиндромом (читается одинаково как слева направо, так и наоборот).

Решение:

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

Тесты:

```
    def test_is_palindrome(self):
        s1 = "шалаш"
        self.assertTrue(lab.is_palindrome(s1))

        s2 = "шашал"
        self.assertFalse(lab.is_palindrome(s2))
```

11. Напишите декоратор, обертывающий возвращаемое строковое значение функции в тег « ».

Решение:

```python
def wrap_in_b_tag(func: Callable) -> Callable:
    return lambda s: f"<b>{func(s)}</b>"
```

Тесты:

```python
    def test_wrap_in_b_tag(self):
        @lab.wrap_in_b_tag
        def str_f(n):
            return 'xd' * n

        result = str_f(4)
        self.assertEqual(result, "<b>xdxdxdxd</b>")
```

12. Декорируйте функцию из первого упражнения таким образом, чтобы возвращаемое ей значение возводилось в квадрат.

Решение:

```python
def square_dec(func: Callable) -> Callable:
    return compose(lambda n: n * n, func)
```

Тесты:

```python
    def test_square_dec(self):
        decorated = lab.square_dec(lab.max_num)
        a, b, c = 9, 2, 5
        result = decorated(a, b, c)
        self.assertEqual(result, 81)
```

13. Декорируйте функцию из четвертого упражнения таким образом, чтобы возвращалась строка в верхнем регистре.

Решение:

```python
def upper(func: Callable) -> Callable:
    return lambda s: func(s).upper()
```

Тесты:

```
    def test_upper(self):
        decorated = lab.upper(lab.str_reverse)
        s = "Всем привет!"
        result = decorated(s)
        self.assertEqual(result, "!ТЕВИРП МЕСВ")
```

14. Напишите декоратор, выводящий значения аргументов, подаваемых на вход декорируемой функции.

Решение:

```
def log_args(func: Callable) -> Callable:
    def logged(*args, **kwargs):
        print("Arguments:", args)
        print("Keyword arguments:", kwargs)
        return func(*args, **kwargs)

    return logged
```

Тесты:

```
@mock.patch("sys.stdout", new_callable=io.StringIO)
def test_log_args(self, stdout):
    @lab.log_args
    def test_f(a, b, inv=False):
        prod = a * b
        return prod if not inv else 1 / prod

    args = (4, 5)
    kwargs = { "inv": False }
    n = test_f(*args, **kwargs)     "n" is not accessed

    self.assertEqual(
        stdout.getvalue(),
        f'Arguments: {args}\nKeyword arguments: {kwargs}\n'
    )
```

15. Напишите генераторную функцию, позволяющую проводить итерацию по значениям в диапазоне от 23 до 37.

Решение:

```python
def grange(start: int, end: int) -> Iterable:
    for i in range(start, end + 1):
        yield i
```

Тесты:

```python
def test_grange(self):
    lst = list(lab.grange(23, 37))
    self.assertEqual(
        lst,
        [23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]
    )
```

16. Напишите генераторную функцию, позволяющую проводить итерацию по значениям в диапазоне от 5 до 37 с шагом 4.

Решение:

```python
def grange_step(start: int, end: int, step: int) -> Iterable:
    for i in range(start, end + 1, step):
        yield i
```

Тесты:

```python
def test_grange_step(self):
    lst = list(lab.grange_step(5, 37, 4))
    self.assertEqual(
        lst,
        [5, 9, 13, 17, 21, 25, 29, 33, 37]
    )
```

18. Напишите генераторное выражение, позволяющее итерироваться элементам списка [3 0 1 3 0 4 3 3 4 56 6 1 3] возводя их при этом в квадрат.

Решение:

```python
def square(lst: list) -> Iterable:
    for n in lst:
        yield n * n
```

Тесты:

```
    def test_square(self):
        lst = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
        result = list(lab.square(lst))
        self.assertEqual(
            result,
            [9, 0, 1, 9, 0, 16, 9, 9, 16, 3136, 36, 1, 9]
        )
```

19. Напишите генераторную функцию, позволяющую проводить итерацию по значениям в диапазоне от 0 до 100 с шагом, регулируемым в процессе ее работы.

Решение:

```
def grange_func(start: int, end: int, step_f: Callable) -> Iterable:
    def get_steps():
        step_sum = 0
        index = 0
        while True:
            step = step_f(index)
            step_sum += step
            if step_sum <= end - start:
                yield step
            else:
                break
            index += 1

    current = start
    for step in get_steps():
        yield current
        current += step
```

Тесты:

```
    def test_grange_func(self):
        def step_f(n):
            return 2 * n

        result = list(lab.grange_func(0, 100, step_f))
        self.assertEqual(
            result,
            [0, 0, 2, 6, 12, 20, 30, 42, 56, 72]
        )
```