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 № 2

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

Проверил: Баталов Д.И.

Санкт-Петербург

2023

Оценочный лист результатов ЛР № 2

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

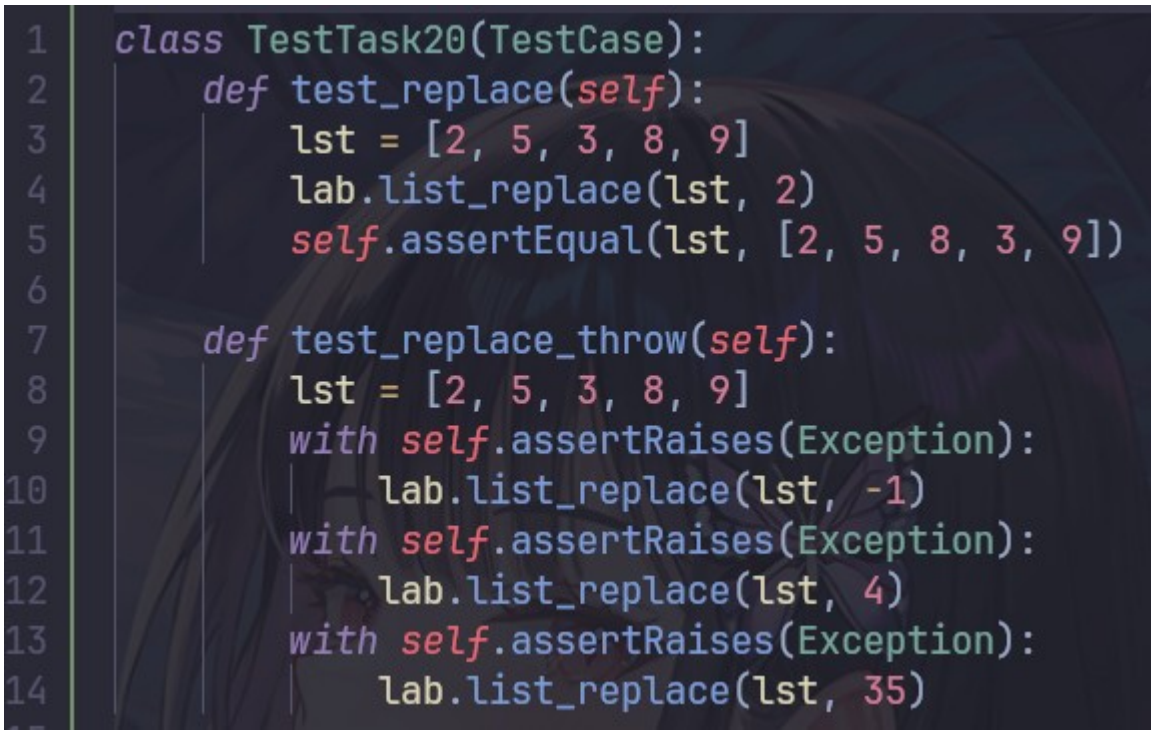
«Информационные и вычислительные

системы» Баталов Д.И. «__»

_____2023 г.

Тестирование

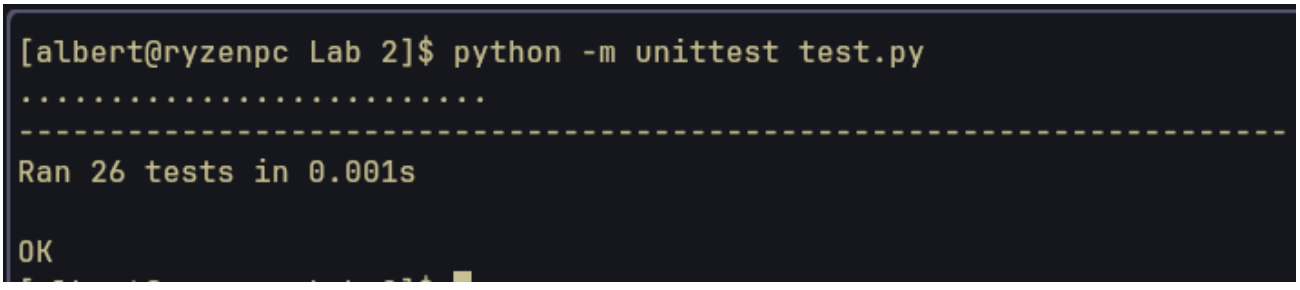
Для тестов используется модуль встроенной библиотеки unittest. Формат тестов везде одинаковы: сначала вызов функции или процедуры, потом сверка её результата с ожидаемым с помощью assert. Если assert имеет два параметра, тестируемое значение всегда слева, ожидаемое всегда справа. Вот пример одного из 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 2]$ python -m unittest test.py
..........................
----------------------------------------------------------------------
Ran 26 tests in 0.001s

OK
```

Более подробные результаты:

```
[albert@ryzenpc Lab 2]$ python -m unittest test.py --verbose
test_task_1 (test.LabTest.test_task_1) ... ok
test_task_10 (test.LabTest.test_task_10) ... ok
test_task_11_for (test.LabTest.test_task_11_for) ... ok
test_task_11_sum (test.LabTest.test_task_11_sum) ... ok
test_task_11_while (test.LabTest.test_task_11_while) ... ok
test_task_12 (test.LabTest.test_task_12) ... ok
test_task_13 (test.LabTest.test_task_13) ... ok
test_task_14 (test.LabTest.test_task_14) ... ok
test_task_16 (test.LabTest.test_task_16) ... ok
test_task_17 (test.LabTest.test_task_17) ... ok
test_task_18 (test.LabTest.test_task_18) ... ok
test_task_19 (test.LabTest.test_task_19) ... ok
test_task_2 (test.LabTest.test_task_2) ... ok
test_task_20 (test.LabTest.test_task_20) ... ok
test_task_3 (test.LabTest.test_task_3) ... ok
test_task_4_for (test.LabTest.test_task_4_for) ... ok
test_task_4_while (test.LabTest.test_task_4_while) ... ok
test_task_5_for (test.LabTest.test_task_5_for) ... ok
test_task_5_while (test.LabTest.test_task_5_while) ... ok
test_task_6_count (test.LabTest.test_task_6_count) ... ok
test_task_6_for (test.LabTest.test_task_6_for) ... ok
test_task_6_while (test.LabTest.test_task_6_while) ... ok
test_task_7_comp (test.LabTest.test_task_7_comp) ... ok
test_task_7_for (test.LabTest.test_task_7_for) ... ok
test_task_8 (test.LabTest.test_task_8) ... ok
test_task_9 (test.LabTest.test_task_9) ... ok

----------------------------------------------------------------------
Ran 26 tests in 0.001s

OK
```

1. Выведите все символы из строки «Данная часть была посвящена больше синтаксисуpython и вопросам документации кода», значения индексов которых делятся на 2.

Решение:

```python
def task_1(s: str) -> str:
    return s[::2]
```

Тесты:

```python
    def test_task_1(self):
        s = "Данная часть была посвящена больше синтаксису python и вопросам документации кода"
        result = lab.task_1(s)
        self.assertEqual(result, "Дна ат ыапсяеаблш иткиурто  орсмдкмнаи оа")
```

2. Выведите все символы из строки «Данная часть была посвящена больше синтаксису Python и вопросам документации кода», значения индексов которых без остатка

Решение:

```python
def task_2(s: str) -> str:
    result = [s[i] if i % 3 == 0 and i % 4 != 0 else "" for i in range(len(s))]
    result = "".join(result)
    return result
```

Тесты:

```python
    def test_task_2(self):
        s = "Данная часть была посвящена больше синтаксису python и вопросам документации кода"
        result = lab.task_2(s)
        self.assertEqual(result, "н слпв леаи п п кеио")
```

3. Выведите все символы из строки «Данная часть была посвящена больше синтаксису Python и вопросам документации кода», значения индексов которых при делении на 6 дают остаток 2, 4, и 5.

Решение:

```python
def task_3(s: str) -> str:
    result = [s[i] if i % 6 in [2, 4, 5] else "" for i in range(len(s))]
    result = "".join(result)
    return result
```

Тесты:

```
    def test_task_3(self):
        s = "Данная часть была посвящена больше синтаксису python и вопросам документации кода"
        result = lab.task_3(s)
        self.assertEqual(result, "наяатьыа сящабош стксуруо иоромдомнти ка")
```

4. Выведите числа из диапазона от 1 до 10, используя цикл for и while.

Решение:

```
def task_4_for(count: int) -> None:
    for i in range(1, count + 1):
        print(f"{i} ", end="")
    print()


def task_4_while(count: int) -> None:
    counter = 1
    while counter ⩽ count:
        print(f"{counter} ", end="")
        counter += 1
    print()
```

Тесты:

```
    @mock.patch("sys.stdout", new_callable=io.StringIO)
    def test_task_4_for(self, stdout):
        lab.task_4_for(10)
        self.assertEqual(stdout.getvalue(), "1 2 3 4 5 6 7 8 9 10 \n")

    @mock.patch("sys.stdout", new_callable=io.StringIO)
    def test_task_4_while(self, stdout):
        lab.task_4_while(10)
        self.assertEqual(stdout.getvalue(), "1 2 3 4 5 6 7 8 9 10 \n")
```

5. Выведите числа из диапазона от –20 до 20 с шагом 3, используя цикл for и while.

Решение:

```python
def task_5_for(start: int, end: int, step: int) -> None:
    for i in range(start, end + 1, step):
        print(f"{i} ", end="")
    print()


def task_5_while(start: int, end: int, step: int) -> None:
    counter = start
    while counter <= end:
        print(f"{counter} ", end="")
        counter += step
    print()
```

Тесты:

```python
    @mock.patch("sys.stdout", new_callable=io.StringIO)
    def test_task_5_for(self, stdout):
        lab.task_5_for(-20, 20, 3)
        self.assertEqual(
            stdout.getvalue(), "-20 -17 -14 -11 -8 -5 -2 1 4 7 10 13 16 19 \n"
        )

    @mock.patch("sys.stdout", new_callable=io.StringIO)
    def test_task_5_while(self, stdout):
        lab.task_5_while(-20, 20, 3)
        self.assertEqual(
            stdout.getvalue(), "-20 -17 -14 -11 -8 -5 -2 1 4 7 10 13 16 19 \n"
        )
```

6. Посчитайте количество вхождений элемента со значением «3» в следующем списке: [3 0 1 3 0 4 3 3 4 56 6 1 3], используя цикл for, while и метод count.

Решение:

```python
def task_6_for(lst: list, elem: Any) -> int:
    count = 0
    for e in lst:
        count += 1 if e == elem else 0
    return count
```

```python
def task_6_while(lst: list, elem: Any) -> int:
    count = 0
    counter = len(lst)
    while counter:
        counter -= 1
        if lst[counter] == elem:
            count += 1
    return count


def task_6_count(lst: list, elem: Any) -> int:
    return lst.count(elem)

```

Тесты:

```python
    def test_task_6_for(self):
        lst = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
        result = lab.task_6_for(lst, 3)
        self.assertEqual(result, 5)

    def test_task_6_while(self):
        lst = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
        result = lab.task_6_while(lst, 3)
        self.assertEqual(result, 5)

    def test_task_6_count(self):
        lst = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
        result = lab.task_6_count(lst, 3)
        self.assertEqual(result, 5)
```

7. Сформируйте список из элементов строки «список доступных атрибутов», используя механизм списковых включений и цикл for.

Решение:

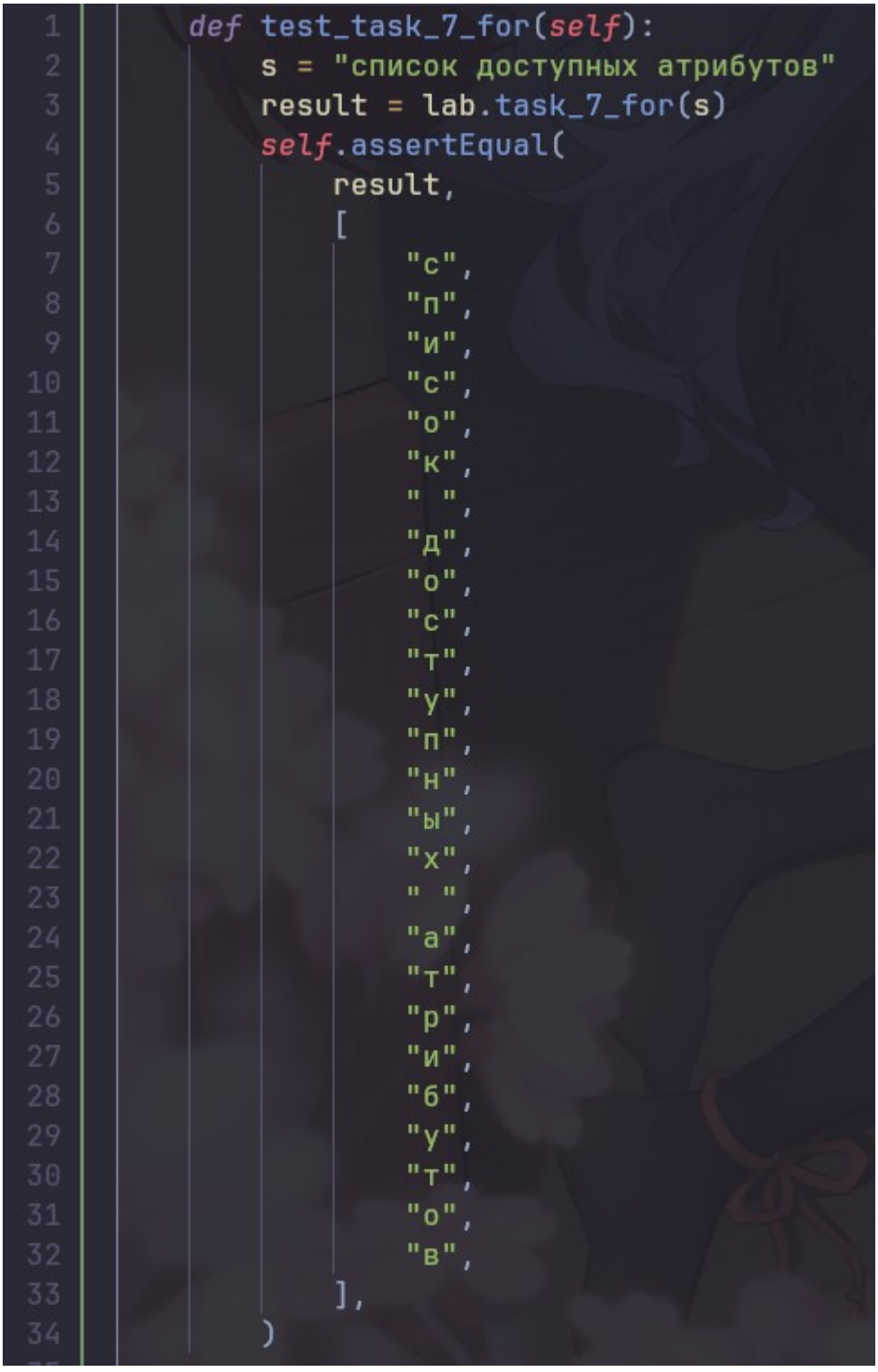
```python
def task_7_comp(s: str) -> list:
    return [c for c in s]
```

```python
def task_7_for(s: str) -> list:
    result = []
    for c in s:
        result.append(c)
    return result
```

Тесты:

```python
def test_task_7_comp(self):
    s = "список доступных атрибутов"
    result = lab.task_7_comp(s)
    self.assertEqual(
        result,
        [
            "с",
            "п",
            "и",
            "с",
            "о",
            "к",
            " ",
            "д",
            "о",
            "с",
            "т",
            "у",
            "п",
            "н",
            "ы",
            "х",
            " ",
            "а",
            "т",
            "р",
            "и",
            "б",
            "у",
            "т",
            "о",
            "в",
        ],
    )
```

2.



```
    def test_task_7_for(self):
        s = "список доступных атрибутов"
        result = lab.task_7_for(s)
        self.assertEqual(
            result,
            [
                "с",
                "п",
                "и",
                "с",
                "о",
                "к",
                " ",
                "д",
                "о",
                "с",
                "т",
                "у",
                "п",
                "н",
                "ы",
                "х",
                " ",
                "а",
                "т",
                "р",
                "и",
                "б",
                "у",
                "т",
                "о",
                "в",
            ],
        )
```

8. Сформируйте единичную матрицу N × N, используя механизм списковых включений.

Решение:

```python
def task_8(size: int) -> List[List]:
    return [[int(col == row) for col in range(size)] for row in range(size)]
```

Тесты:

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

9. Напишите программу, выводящую элементы списка [3 0 1 3 0 4 3 3 4 56 6 1 3] в обратной последовательности.

Решение:

```python
def task_9(lst: list) -> list:
    return lst[::-1]
```

Тесты:

```python
    def test_task_9(self):
        lst = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
        result = lab.task_9(lst)
        self.assertEqual(result, [3, 1, 6, 56, 4, 3, 3, 4, 0, 3, 1, 0, 3])
```

10. Напишите программу, которая выводит числе в диапазоне от 1 до 9, кроме 5 и 7.

Решение:

```python
def task_10(start: int, end: int, exclude: List[int]) -> None:
    numbers = [i for i in range(start, end + 1) if i not in exclude]
    strs = map(str, numbers)
    print(", ".join(strs))
```

Тесты:

```python
    @mock.patch("sys.stdout", new_callable=io.StringIO)
    def test_task_10(self, stdout):
        lab.task_10(1, 9, [5, 7])
        self.assertEqual(stdout.getvalue(), "1, 2, 3, 4, 6, 8, 9\n")
```

11. Напишите программу, выводящую сумму элементов списка [3 0 1 3 0 4 3 3 4 56 6 1 3], используя цикл for, while и метод sum.

Решение:

```python
def task_11_for(lst: list[float]) -> float:
    res = 0
    for n in lst:
        res += n
    return res


def task_11_while(lst: list[float]) -> float:
    res = 0
    counter = len(lst) - 1
    while counter >= 0:
        res += lst[counter]
        counter -= 1
    return res


task_11_sum = sum
```

Тесты:

```python
def test_task_11_for(self):
    lst = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
    result = lab.task_11_for(lst)  # type: ignore
    self.assertEqual(result, 87)

def test_task_11_while(self):
    lst = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
    result = lab.task_11_while(lst)  # type: ignore
    self.assertEqual(result, 87)

def test_task_11_sum(self):
    lst = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
    result = lab.task_11_sum(lst)  # type: ignore
    self.assertEqual(result, 87)
```

12. Напишите программу, выводящую сумму элементов списка [3 0 1 3 0 4 3 3 4 56 6 1 3], значения индексов которых делятся на без остатка на 3, используя цикл for и while.

Решение:

```python
def compose2(f, g):
    return lambda *args, **kwargs: f(g(*args, *kwargs))


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


task_12 = compose(sum, partial(filter, lambda n: n % 3 == 0))
```

Тесты:

```python
    def test_task_12(self):
        lst = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
        result = lab.task_12(lst)
        self.assertEqual(result, 21)
```

13. Сформируйте список, значения элементов которого находятся в диапазоне от 23 до 35.

Решение:

```python
def task_13(start: int, end: int) -> list[int]:
    return list(range(start, end + 1))
```

Тесты:

```python
def test_task_13(self):
    lst = lab.task_13(23, 35)
    self.assertEqual(lst, [23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35])
```

14. Сформируйте список, значения элементов которого находятся в диапазонеот 3 до 15 с шагом 4.

Решение:

```python
def task_14(start: int, end: int, step: int) -> list[int]:
    return list(range(start, end + 1, step))
```

Тесты:

```python
    def test_task_14(self):
        lst = lab.task_14(3, 15, 4)
        self.assertEqual(lst, [3, 7, 11, 15])
```

15. Сформируйте список, значения элементов которого находятся в диапазонеот 3 до 25 и без остатка делятся на 3.

Решение:

```python
def task_15(start: int, end: int, predicate: Callable[[int], bool]) -> list[int]:
    return list(filter(predicate, range(start, end + 1)))
```

Тесты:

```python
    def tesk_task_15(self):
        def predicate(n):
            return n % 3 == 0

        lst = lab.task_15(3, 25, predicate)
        self.assertEqual(lst, [3, 6, 9, 12, 15, 18, 21, 24])
```

16. Сформируйте словарь из двух списков [3 0 1 3 0 4 3 3 4 56 6 1 3] и [2, 4, 7, 26, 33], используя встроенную функцию zip. Выведите словарь в консоль и объясните, почему он получился такого вида.

Решение:

```python
task_16 = compose(dict, zip)
```

Тесты:

```python
    def test_task_16(self):
        lst1 = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
        lst2 = [2, 4, 7, 26, 33]
        result = lab.task_16(lst1, lst2)
        self.assertEqual(result, {3: 26, 0: 33, 1: 7})
```

17. Выведите различными способами в консоль элементы списка [3 0 1 3 0 4 3 3 4 56 6 1 3] с их индексами.

Решение:

```
task_17 = compose(
    print,
    "\n".join,
    partial(map, (lambda i_n: f"{i_n[0]+1}: {i_n[1]}")),
    enumerate,
)
```

Тесты:

```
@mock.patch("sys.stdout", new_callable=io.StringIO)
def test_task_17(self, stdout):
    lst = [3, 0, 1, 3, 0, 4, 3, 3, 4, 56, 6, 1, 3]
    lab.task_17(lst)
    self.assertEqual(
        stdout.getvalue(),
        "1: 3\n2: 0\n3: 1\n4: 3\n5: 0\n6: 4\n7: 3\n8: 3\n9: 4\n10: 56\n11: 6\n12: 1\n13: 3\n",
    )
```

18. Напишите программу, которая считывает целое число (месяц), после чего выводит сезон к которому этот месяц относится.

Решение:

```
def task_18(month: int) -> str:
    match month:
        case 1 | 2 | 12:
            return "Winter"
        case 3 | 4 | 5:
            return "Spring"
        case 6 | 7 | 8:
            return "Summer"
        case 9 | 10 | 11:
            return "Autumn"
        case _:
            return "Unknown"
```

Тесты:

```
def test_task_18(self):
    self.assertEqual(lab.task_18(1), "Winter")
    self.assertEqual(lab.task_18(5), "Spring")
    self.assertEqual(lab.task_18(6), "Summer")
    self.assertEqual(lab.task_18(10), "Autumn")
```

19. Напишите программу, выводящую среднее из трех значений.

Решение:

```python
def task_19(numbers: list[float]):
    numbers_len = len(numbers)
    match numbers:
        case []:
            return None
        case [n]:
            return n
        case [a, b]:
            return (a + b) / 2
        case _ if numbers_len % 2 == 1:
            return sorted(numbers)[numbers_len // 2]
        case _ if numbers_len % 2 == 0:
            second_index = len(numbers) // 2
            first_index = second_index - 1
            numbers = sorted(numbers)
            return (numbers[first_index] + numbers[second_index]) / 2
```

Тесты:

```python
    def test_task_19(self):
        nums = [9, 2, 7]
        result = lab.task_19(nums)
        self.assertEqual(result, 7)
```

2. Напишите программу, выводящую таблицу умножения для задаваемого пользователем числа от 1 до 9 (включительно).

Решение:

```python
def task_20(num: int) -> None:
    strs = map(lambda i: f"{num} * {i} = {num * i}", range(1, 10))
    print("\n".join(strs))
```

Тесты:

```python
    @mock.patch("sys.stdout", new_callable=io.StringIO)
    def test_task_20(self, stdout):
        lab.task_20(3)
        self.assertEqual(
            stdout.getvalue(),
            "3 * 1 = 3\n3 * 2 = 6\n3 * 3 = 9\n3 * 4 = 12\n3 * 5 = 15"
            + "\n3 * 6 = 18\n3 * 7 = 21\n3 * 8 = 24\n3 * 9 = 27\n",
        )
```