Лабораторная работа №4

Задания на списковые включения

- 3. Используя списковое включение сформируйте список, элементы которого нацело делятся на 3 и 5 в диапазоне от -43 до 57 и выведите полученный результат в терминал.
- 9. Дан список my_list1 = [1, 2, 3, 4, 9, 7, 4, 6, 22, 3, 84, 21, 45, 76]. Используя списковое включение прибавьте к каждому значению его элементов единицу, если оно больше 15 и выведите полученный результат в терминал.
- 10. Используя списковое включение сформируйте список в диапазоне от 0 до 100 из чисел, в состав которых входит цифра один и выведите полученный результат в терминал.

```
'''
created by bulat 07.02.2023
'''
from typing import List


def task_3(start: int, end: int) -> List[int]:
    '''
    Return an array where each element is divisible by both 3 and 5
    '''

    array = [el for el in range(start, end) if (el % 3 == 0  and el % 5 ==
0)]
    return array

def task_9(array: List[int]) -> List[int]:
    """
    Increment each array element by one if it is greater than 15
    """

    new_array = [el+ 1 if el > 15 else el for el in array]
    return new_array

def task_10(start: int, end: int) -> List[int]:
    """
    return an array whose entry contains the number 1
    """

    array = [el for el in range(start, end) if '1' in str(el)]
```

```
    return array
```

```
import unittest
import list_comprehensions as lc


class TestListComprehensions(unittest.TestCase):

    def test_task_3(self):
        self.assertEqual(lc.task_3(5, 20), [15])

        right_array = [-30, -15, 0, 15, 30, 45]
        self.assertEqual(lc.task_3(-43, 57), right_array)

    def test_task_9(self):
        answer = lc.task_9([16, 15, 14])
        self.assertEqual(answer, [17, 15, 14])

        test_case = [1, 2, 3, 4, 9, 7, 4, 6, 22, 3, 84, 21, 45, 76]
        right_answer = [1, 2, 3, 4, 9, 7, 4, 6, 23, 3, 85, 22, 46, 77]
        self.assertEqual(lc.task_9(test_case), right_answer)

    def test_task_10(self):
        start, end = 0, 15
        right_answer = [1, 10, 11, 12, 13, 14]
        self.assertEqual(lc.task_10(start, end), right_answer)
```

```
➜  4_lab git:(main) ✗ python3 -m unittest -v tests/test_functions.py
test_add (tests.test_functions.TestFunctions) ... ok
test_gcd (tests.test_functions.TestFunctions) ... ok
test_is_bit_set (tests.test_functions.TestFunctions) ... ok
test_list_create (tests.test_functions.TestFunctions) ... ok
test_volume_of_box (tests.test_functions.TestFunctions) ... ok

----------------------------------------------------------------------
Ran 5 tests in 0.001s

OK
```

Задания на функции

- 3. Напишите функцию add, принимающую на вход два списка и возвращающую сумму их элементов.
- 4. Напишите функцию list_create, на вход которой подается 3 значения любого типа и возвращающую сформированный из них список.

- 12. Напишите функцию is_bit_set, на вход которой подается два значения. Первое значение представляет собой целочисленное число, у которого будет проверяться факт того, установлен ли заданный бит (второй аргумент функции) в единицу или нет. Результат проверки необходимо вернуть в виде булевского значения (True – да, нет – False).
- 20. Напишите функцию gcd, на вход которой подается два целочисленных значения. Функция должна быть реализована без использования рекурсии и возвращать их наибольший общий делитель.
- 21. Напишите функцию volume_of_box на вход которой подается 3 значения (ширина, длина, высота), где 2 значения (ширина = 10, высота = 7) заданы по умолчанию. Функция должна рассчитывать и возвращать объем коробки с заданными параметрами.

```
'created by bulat 08.03.2023'
from typing import List, Union, Any


ArrayType = List[Union[int, float]]


def add(array1: ArrayType, array2: ArrayType) -> ArrayType:
    """
    take two arrays and returns an array with the sums of thier elements
    """
    new_array = []
    index = 0
    while index < len(array1) and index < len(array2):
        new_array.append(array1[index] + array2[index])
        index += 1
    while index < len(array1):
        new_array.append(array1[index])
        index += 1
    while index < len(array2):
        new_array.append(array2[index])
        index += 1
    return new_array

def list_create(a: Any, b: Any, c: Any) -> List[Any]:
    """
    take 3 random elements and return array with this elements
    """
    return [a, b, c]

def is_bit_set(number: int, bit_id: int) -> bool:
    """
    number: the integer whose bits are checking
    bit_id: bit sequence number
```

```
    return: is the bit_id bit set to one
    """
    check_number = 1 << bit_id
    return number & check_number != 0

def gcd(a: int, b: int) -> int:
    """
    a, b: numbers whose common divisor is sought
    return greatest common divisor
    """
    while b != 0:
        temp = b
        b = a % b
        a = temp
    return a

def volume_of_box(l: int, h: int=7, w: int=10) -> int:
    """
    h: box height
    w: box width
    l: box length

    return: box volume
    """
    return l*h*w
```

```
from unittest import TestCase
import functions as func


class TestFunctions(TestCase):
    def test_add(self):
        test_case = [1, 2, 3], [4, 5, 6]
        right_answer = [5, 7, 9]
        self.assertEqual(func.add(*test_case), right_answer)

    def test_list_create(self):
        test_case = ['123', 123, None]
        self.assertEqual(func.list_create(*test_case), test_case)

    def test_is_bit_set(self):
        test_cases = [
            (1, 0, True),
            (1, 1, False),
            (0, 0, False),
            (7, 0, True),
            (7, 1, True),
            (7, 2, True),
```

```
            (7, 3, False)
        ]
        for test in test_cases:
            number, bit_id, right_answer = test
            self.assertEqual(func.is_bit_set(number, bit_id),
right_answer)

    def test_gcd(self):
        test_cases = [
            (6, 3, 3),
            (6, 9, 3),
            (12, 16, 4),
            (11, 17, 1),
        ]
        for test in test_cases:
            a, b, right_answer = test
            self.assertEqual(func.gcd(a, b), right_answer)

    def test_volume_of_box(self):
        test_cases = [
            ([1], 70),
            ([1, 2], 20),
            ([1, 1, 1], 1),
        ]
        for test in test_cases:
            params, right_answer = test
            self.assertEqual(func.volume_of_box(*params), right_answer)
```

```
➜  4_lab git:(main) ✗ python3 -m unittest -v
tests/test_list_comprehensions.py
test_task_10 (tests.test_list_comprehensions.TestListComprehensions) ...
ok
test_task_3 (tests.test_list_comprehensions.TestListComprehensions) ... ok
test_task_9 (tests.test_list_comprehensions.TestListComprehensions) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```