ГУАП

КАФЕДРА № 41

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ассистент		Е.К. Григорьев
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 6

Декораторы и генераторные функции

по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4116		Четвергов В.Ю.
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

Цель работы: познакомиться с основными способами использования декораторов и генераторов в Python. Вариант 7

Часть 1. Задания на декораторы:

Напишите декоратор pow_list_result, который будет возводить в квадрат результирующее значение (элементы списка) функции sum_list, на вход которой подается целочисленный список и возвращается сумма его элементов.

```
def pow_list_result(func):
    def wrapper(lst):
        result = func(lst)
        return [x**2 for x in result]
    return wrapper
@pow_list_result
def sum_list(lst):
    return sum(lst)
lst = list(map(int, input("Введите целочисленный список через пробел: ").split()))
print(sum_list(lst))
```

Напишите декоратор sqrt_list_result, который будет вычислять корень результирующего значения функции max_number, на вход которой подается целочисленный список и возвращается его найденное максимальное значение

```
import math
def sqrt_list_result(func):
    def wrapper(lst):
        result = func(lst)
        return math.sqrt(result)
    return wrapper
```

```
@sqrt_list_result
def max_number(lst):
    return max(lst)
lst = list(map(int, input("Введите целочисленный список через пробел: ").split()))
print(max_number(lst))
```

Напишите декоратор upcase_result, который будет переводить все символы результирующего значения функции reverse_str в верхний регистр. На вход функции reverse_str подается строка и возвращается ее инвертированное представление

```
def upcase_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        return result.upper()
    return wrapper
@upcase_result
def reverse_str(string):
    return string[::-1]
```

Часть 2. Задания на генераторы:

Напишите генераторную функцию list_mod, на вход которой подается список целочисленных значений и значение k. Функция должна возвращать только те элементы списка, значение которых удовлетворяют следующему условию: list_mode[i] mod k == 0

```
def list_mod(lst, k):
    for x in lst:
        if x % k == 0:
            yield x
input_str = input("Введите список элементов через пробел: ")
# преобразуем строку в список чисел
```

```
my_list = [int(num) for num in input_str.split()]
k = int(input("k= "))
result = list_mod(my_list, k)
print(list(result))
```

Напишите генераторную функцию list_pow, на вход которой подается список целочисленных значений, и она возвращает на каждой итерации квадрат значения элемента списка

```
def list_pow(lst):
    for num in lst:
        yield num ** 2
input_str = input("Введите список элементов через пробел: ")
# преобразуем строку в список чисел
my_list = [int(num) for num in input_str.split()]
for square in list_pow(my_list):
    print(square)
```

Напишите генераторную функцию list_odd, на вход которой подается список целочисленных значений. Функция должна возвращать только те четные значения из поданного на ее вход списка

```
def list_odd(lst):
    return [x for x in lst if x % 2 == 0]
input_str = input("Введите список элементов через пробел: ")
# преобразуем строку в список чисел
my_list = [int(num) for num in input_str.split()]
for square in list_odd(my_list):
    print(square)
```

Вывод: я ознакомлен с основными способами работы с декораторами Python