МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет по лабораторной работе №15 по дисциплине:основы программной инженерии

Выполнил:

студент группы ПИЖ-б-о-21-1

Турклиев Владимир Назирович

Проверил:

доцент кафедры инфокоммуникаций

Романкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ

Пример 1

```
def benchmark(func):
    import time

    def wrapper():
        start = time.time()
        func()
        end = time.time()
        print('[*] Время выполнения: {} секунд.'.format(end-start))
    return wrapper

@benchmark
def fetch_webpage():
    import requests
    webpage = requests.get('https://google.com')

fetch_webpage()
```

Пример 2

```
def benchmark(func):
    import time

    def wrapper(*args, **kwargs):
        start = time.time()
        return_value = func(*args, **kwargs)
        end = time.time()
        print('[*] Время выполнения: {} секунд.'.format(end-start))
        return return_value
    return wrapper

@benchmark
def fetch_webpage(url):
    import requests
    webpage = requests.get(url)
    return webpage.text

webpage = fetch_webpage('https://google.com')

print(webpage)
```

Индивидуальное задание

Вводятся два списка (каждый с новой строки) из слов, записанных через пробел. Имеется функция, которая преобразовывает эти две строки в два списка слов и возвращает эти списки. Определите декоратор для этой функции, который из этих двух списков формирует словарь, в котором ключами являются слова из первого списка, а значениями – соответствующие элементы из второго списка. Полученный словарь должен возвращаться при вызове декоратора. Примените декоратор к первой функции и вызовите ее. Результат (словарь) отобразите на экране.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def decorator(func):
    def decorator_inside(A, B):
        data = func(A, B)
        return dict(zip(*data))

    return decorator_inside

@decorator
def listing(A, B):
    return A.split(), B.split()


if __name__ == '__main__':
    First = input("Введите первую строку: ")
    Second = input("Введите вторую строку: ")
    print(listing(First, Second))
```

```
if __name__ == '__main__'
ind ×
C:\Users\Vova\AppData\Local\Programs\Python\Python39\python.exe "
Введите первую строку: 1 2 3
Введите вторую строку: -один -два -три
{'1': '-один', '2': '-два', '3': '-три'}
```

GitHub - https://github.com/vegas007gof/lab15

Ответы на контрольные вопросы:

1. Что такое декоратор?

Декоратор – это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода.

2. Почему функции являются объектами первого класса?

Объектами первого класса в контексте конкретного языка программированияназывается

элементы, с которыми можно делать всё то же, что и с любым другим объектом: передавать, как параметр, возвращать из функции и
присваивать переменной.

3. Каково назначение функций высших порядков?

Функции высших порядков – это такие функции, которые могут принимать вкачестве аргументов и возвращать другие функции.

4. Как работают декораторы?

Декоратор – это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода.Внутри декораторы мы определяем другую функцию, обёртку, так сказать,
которая обёртывает функцию-аргумент и затем изменяет её поведение. Мысоздаём декоратор, замеряющий время выполнения функции. Далее мы
используем его функции, которая делает GET- запрос к главной странице. Чтобы измерить скорость, мы сначала сохраняем время перед выполнениемобёрнутой функции, выполняем её снова сохраняем текущее время и вычитаем из него начальное. Выражение @decorator_function вызывает decorator_function() с hello_world вкачестве аргумента и присваивает имени hello_world возвращаемую
функцию.

```
def benchmark(func):
    import time

    def wrapper():
        start = time.time()
        func()
        end = time.time()
        print('[*] Время выполнения: {} секунд.'.format(end-start))
    return wrapper

@benchmark
def fetch_webpage():
    import requests
    webpage = requests.get('https://google.com')

fetch_webpage()
```

5. Какова структура декоратора функций?

```python
def benchmark(func):
    import time

    def wrapper(*args, **kwargs):
        start = time.time()
        return_value = func(*args, **kwargs)
        end = time.time()
        print('[*] Время выполнения: {} секунд.'.format(end-start))
        return return_value
    return wrapper

@benchmark
def fetch_webpage(url):
    import requests
    webpage = requests.get(url)
    return webpage.text

webpage = fetch_webpage('https://google.com')
print(webpage)
```

6. Самостоятельно изучить как можно передать параметры декоратору, ане декорируемой функции?

```python
import functools


def decoration(*args):
    def dec(func):
        @functools.wraps(func)
        def decor():
            func()
            print(*args)
        return decor
    return dec


@decoration('This is *args')
def func_ex():
    print('Look at that')

if __name__ == '__main__':
    func_ex()
```

```
Look at that
This is *args

Process finished with exit code 0
```