МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 2.12
по дисциплине «Основы программной инженерии»

Выполнил студент группы

ПИЖ-б-о-21-1

Трушева В. О. .« » 2022г.

Подпись студента_____

Работа защищена «

»_____20__г.

Проверила Воронкин Р.А.

_____

(подпись)

Ставрополь 2022

Методика и порядок выполнения работы

1. Изучить теоретический материал работы.

2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.

3. Выполните клонирование созданного репозитория.

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.

5. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

6. Создайте проект PyCharm в папке репозитория.

7. Проработать примеры лабораторной работы.

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

def decorator_function(func):
    def wrapper():
        print('Функция-обёртка!')
        print('Оборачиваемая функция: {}'.format(func))
        print('Выполняем обёрнутую функцию...')
        func()
        print('Выходим из обёртки')
    return wrapper


@decorator_function
def hello_world():
    print('Hello world!')


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

```
if __name__ == '__main__'
1_Task_(primer_1)
D:\fgit\15_LR\PyCharm\Scripts\python.exe "D:\fgit\15_LR\Tasks\1_Task_(
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x0000018ABF0DFF60>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки

Process finished with exit code 0
```

Рисунок 1 – Результат выполнения программы

```
2_Task_(primer_2)
D:\fgit\15_LR\PyCharm\Scripts\python.exe "D:\fgit\15_LR\Tasks\2_Task_(primer
[*] Время выполнения: 1.1928489208221436 секунд.
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang=
var f=this||self;var h,k=[];function l(a){for(var b;a&&(!a.getAttribute||!(b
function n(a,b,c,d,g){var e="";c||-1!==b.search("&ei=")||(e="&ei="+l(d),-1==
document.documentElement.addEventListener("submit",function(b){var a;if(a=b.
```

Рисунок 2 – Результат выполнения программы

8. Выполнить индивидуальное задание.

Вариант – 10.

Условие. Объявите функцию, которая принимает строку, удаляет из нее все подряд идущие пробелы и переводит ее в нижний регистр – малые буквы. Результат (строка) возвращается функцией. Определите декоратор, который строку, возвращенную функцией, переводит в азбуку Морзе, используя следующий словарь для замены русских букв и символа пробела на соответствующие последовательности из точек и тире:

```
morze = {'а': '.-', 'б': '-...', 'в': '.--', 'г': '--.', 'д': '-..', 'е':
'.', 'ё': '.', 'ж': '...-', 'з': '--..', 'и': '..', 'й':'.---', 'к': '-.-',
'л': '.-..', 'м': '--', 'н': '-.', 'о': '---', 'п': '.--.', 'р': '.-.', 'с':
'...', 'т': '-', 'у': '..-', 'ф': '..-.', 'х': '....', 'ц': '-.-.', 'ч': '--
-.', 'ш': '----', 'щ': '--.-', 'ъ': '--.--', 'ы': '-.--', 'ь': '-..-', 'э':
'..-..', 'ю': '..--', 'я': '.-.-', ' ': '-...-'}
```

Преобразованная строка возвращается декоратором. Примените декоратор к функции и вызовите декорированную функцию. Результат работы отобразите на экране.

9. Зафиксируйте изменения в репозитории.

10. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

11. Выполните слияние ветки для разработки с веткой master/main.

12. Отправьте сделанные изменения на сервер GitHub.

13. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Вопросы для защиты работы

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 программирования называется 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 в качестве 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.

5. Какова структура декоратора функций?

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

def decorator_function(func):
    def wrapper():
        print('Функция-обёртка!')
        print('Оборачиваемая функция: {}'.format(func))
        print('Выполняем обёрнутую функцию...')
        func()
        print('Выходим из обёртки')
    return wrapper


@decorator_function
def hello_world():
    print('Hello world!')


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

```
orator_function()
пример_1 (1)
"C:\Users\ynakh\OneDrive\Рабочий стол\Git\Lab-15-OPJ\p15\venv\Scripts\
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x000002C0B4797AC0>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки

Process finished with exit code 0
```

6. Самостоятельно изучить как можно передать параметры декоратору, а не 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')


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

```
гшрж
"C:\Users\ynakh\OneDrive\Рабочий стол\
Look
This is args

Process finished with exit code 0
```