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

Отчет о лабораторной работе №9

по дисциплине основы программной инженерии

Выполнил: Духно Михаил Александрович,

2 курс, группа ПИЖ-б-о-20-1,

Проверил: Доцент кафедры инфокоммуникаций, Воронкин Р.А.

Ставрополь, 2021 г

```python
def hello_world():
    print('hello wordl!')

type(hello_world)
#<class 'function'>
class Hello:
    pass
type(Hello)
#<class 'type'>
type(10)
#<class 'int'>

def wrapper_function():
    def hello_world():
        print('hello world!')
    hello_world()

wrapper_function()
#hello world!
```

Рисунок 15.1 – Примеры

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

@decorator_function
def hello_world():
    print('hello_world')

hello_world()
#hello_world
```

Рисунок 15.2 – Как работают декораторы

```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)
```

Рисунок 15.3 – Декораторы с аргументами

```python
import math


def console_message(func):
    def print_message(*args):
        square = func(*args)
        print('Площадь круга равна = %.2f' %square)
    return print_message


@console_message
def calculate_square(radius):
    return math.pi * radius**2


if __name__ == '__main__':
    radius = int(input('Введите значение радуиса: '))
    calculate_square(radius)
```

Рисунок 15.4 – Код программы индивидуального задания

```
Введите значение радуиса: 5
Площадь круга равна = 78.54

Process finished with exit code 0
```

Рисунок 15.5 – Результат работы программы

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?

«Функции первого класса» (FCF) – это функции, которые рассматриваются как так называемые «первоклассные граждане» (FCC). FCC на языке программирования – это объекты (здесь очень свободно используется термин «объекты»), которые:

- Может использоваться как параметры
- Может использоваться как возвращаемое значение
- Может быть присвоен переменным
- Может храниться в структурах данных, таких как хеш-таблицы, списки, …

3. Каково назначение функций высших порядков?

В языках, где функции можно принимать и передавать в качестве значений, функции называются *гражданами первого сорта (first-class citizen)*. А функции, которые принимают в качестве аргументов другие функции и/или возвращают функции в качестве результата, принято называть *функциями высшего порядка (или же функциями высших порядков, ФВП, high-order functions)*.

Таким образов функции высших порядков нужны для расширения функционала других функций

4. Как работают декораторы?

выражение @decorator_function вызывает decorator_function() с

hello_world в качестве аргумента и присваивает имени hello_world возвращаемую функцию.

5. Какова структура декоратора функций?

```
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, а не декорируемой функции?

```
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
```