

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего
образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»**

Кафедра

инфокоммуникаций

**Институт цифрового
развития**

ОТЧЁТ

по лабораторной работе №15

Дисциплина: «Основы программной инженерии»

Тема: «Декораторы функций в языке Python»

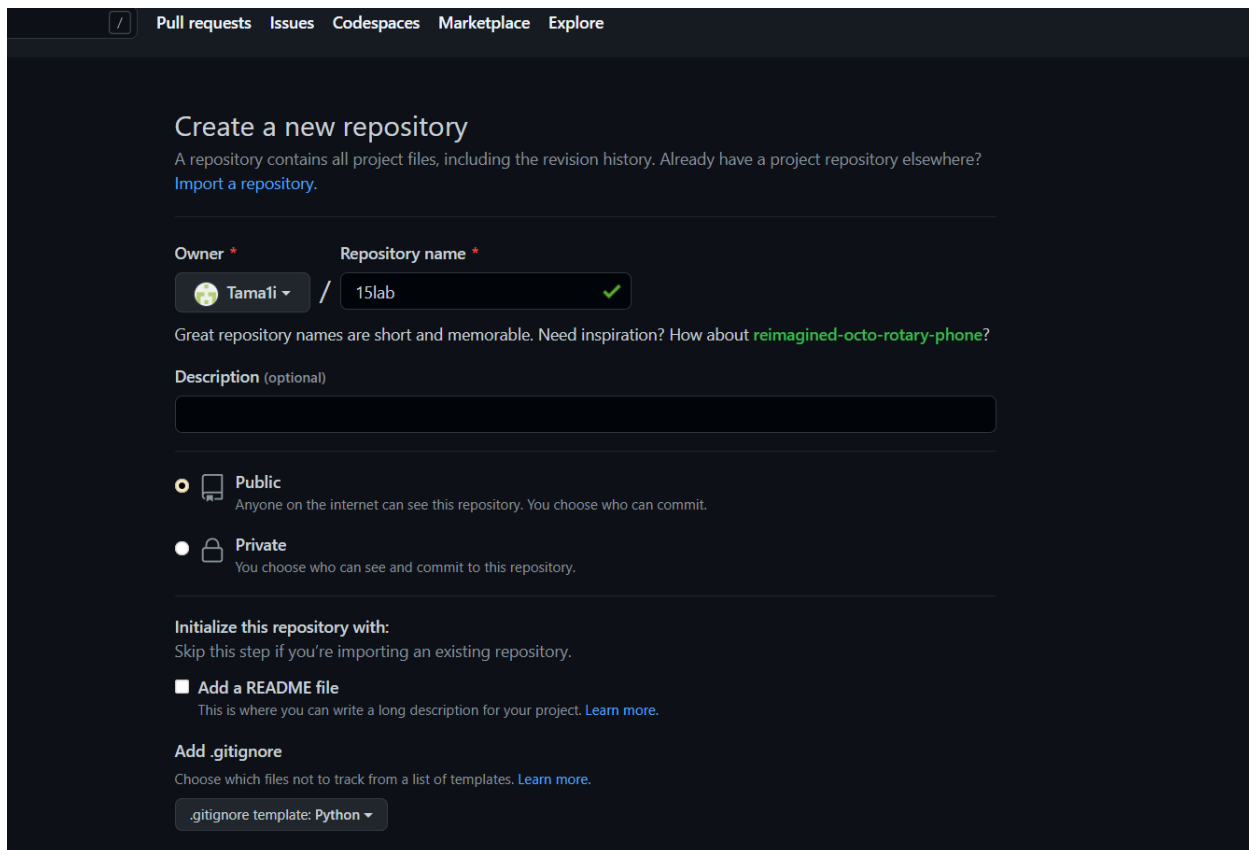
Выполнил: студент 2
курса группы Пиж-б-о-
21-1

Рязанцев Матвей
Денисович

Ставрополь 2022

Цель работы: приобретение навыков по работе с функциями с переменным числом

Выполнение работы



GitHub interface showing the 'Create a new repository' page. The page includes fields for 'Owner' (TamaLi) and 'Repository name' (15lab). It also shows options for 'Description' (optional), 'Public' (selected) or 'Private' repository, and 'Initialize this repository with:' (Add a README file, .gitignore template: Python).

Рисунок 1 -создание репозитория

```
D:\gite\15lab>git flow init

which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/gite/15lab/.git/hooks]
```

Рисунок 2 - Организация репозитория по модели ветвления git flow

Код общего задания:

1)

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

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

if __name__ == "__main__":
    wrapper_function()
```

2)

```
#!/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()
```

3)

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

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
    requests.get('https://google.com')

if __name__ == "__main__":
    fetch_webpage()
```

```
fetch_webpage()
```

4)

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

```
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

if __name__ == "__main__":
    webpage = fetch_webpage('https://google.com')
    print(webpage)
```

Индивидуальное задание

Объявите функцию с именем `get_sq`, которая вычисляет площадь прямоугольника по двум параметрам: `width` и `height` – ширина и высота прямоугольника и возвращает результат. Определите декоратор для этой функции с именем (внешней функции) `func_show`, который отображает результат на экране в виде строки (без кавычек): "Площадь прямоугольника: ". Вызовите декорированную функцию `get_sq`.

Код программы:

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

import sys

def func_show(func):
    def get_sq(**wh):
        x = 1
        for v in wh.values():
            x *= int(v)
        func(x)
    return get_sq

@func_show
def result(wh):
    print(f"Площадь прямоугольника: {wh}")
```

```
if __name__ == '__main__':  
    result(width=4, height=6)
```

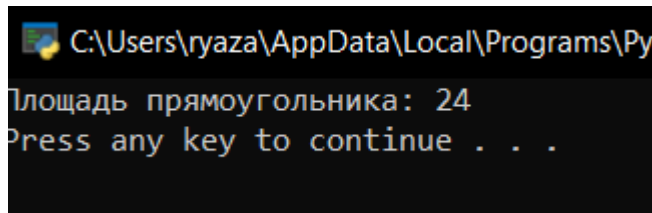


Рисунок 4 – результат работы программы

Контрольные вопросы

Контр. вопросы и ответы на них:

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?

Потому что с ними можно работать как с переменными, могут быть переданы как аргумент процедуры, могут быть возвращены как результат выполнения процедуры, могут быть включены в другие структуры данных.

3. Каково назначение функций высших порядков?

Основной задачей функций высших порядков является возможность принимать в качестве аргументов и возвращать другие функции.

4. Как работают декораторы?

Они берут декорируемую функцию в качестве аргумента и позволяют совершать с ней какие-либо действия до и после того, что сделает эта функция, не изменяя её.

5. Какова структура декоратора функций?

Функция decorator принимает в качестве аргумента функцию func, внутри функции decorator другая функция wrapper. В конце декоратора

происходит возвращение функции `wrapper`.

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

1. Достаточно обернуть функцию декоратор в другую функцию, которая будет принимать аргументы. И сделать