

**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 №2.12

Дисциплина: «Декораторы функций в языке Python»

Тема: «Замыкания в языке Python»

Выполнил: студент 2 курса

группы ИВТ-б-о-21-1

Кочкаров Умар Ахматович

Ставрополь 2022


Цель работы: приобретение навыков по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.

Ход работы:

1. Создать общедоступный репозиторий с лицензией MIT и языком Python.

Owner *

Repository name *


 umarkochkarov ▾

 /


lb2..12 ✓

Great repository names are short and memorable. Need inspiration? How about [verbose-couscous?](#)

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore


Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▾

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

 You are creating a public repository in your personal account.

Create repository

Рисунок 1. Создание репозитория

2. Клонировать репозиторий на ПК:

```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/python/Ла62.12
$ git clone https://github.com/umarkochkarov/lb2.12.git
Cloning into 'lb2.12'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 2. Клонирование репозитория

3. Организовать репозиторий в соответствии с моделью ветвления git-flow.

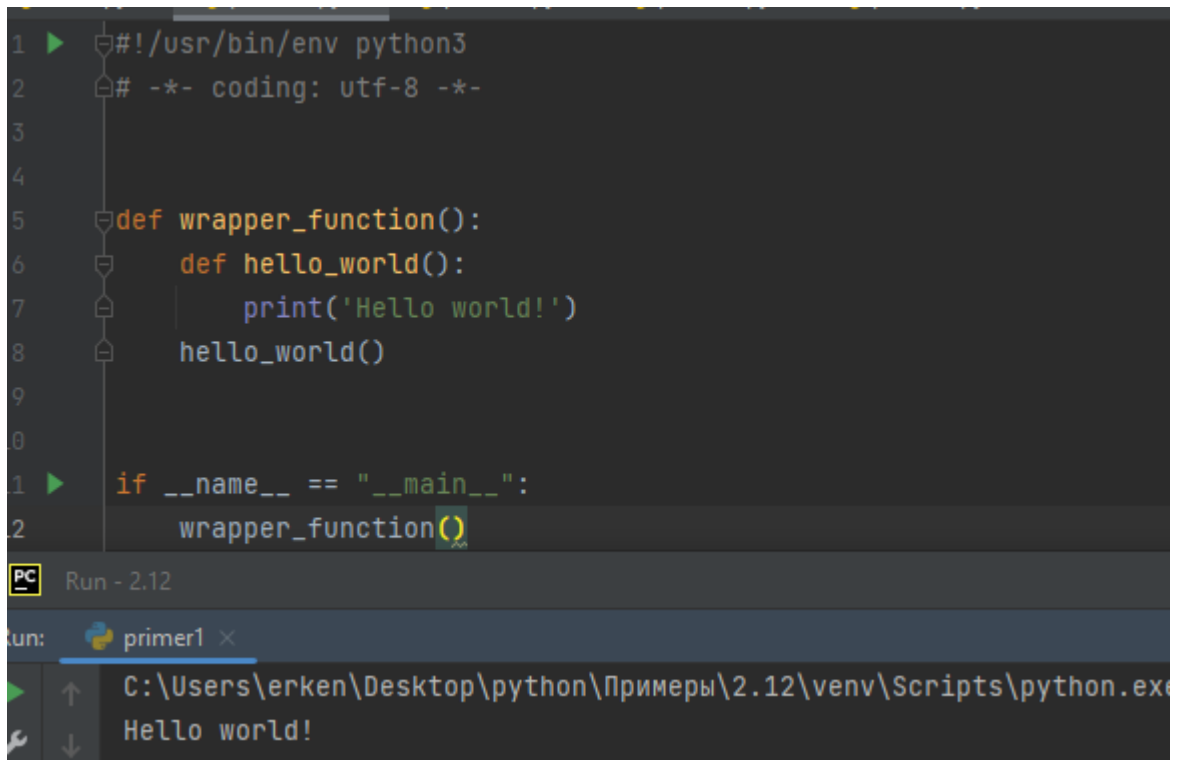
```
erken@LAPTOP-ESTC60GF MINGW64 ~/Desktop/python/Ла62.12/lb2.12 (main)
$ git flow init

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? [C:/Users/erken/Desktop/python/Ла62.12/lb2.12/.git/hooks]
```

Рисунок 3. Организация репозитория в соответствии с моделью git-flow

4. Проработка примеров из лабораторной работы:



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def wrapper_function():
6      def hello_world():
7          print('Hello world!')
8      hello_world()
9
10
11 if __name__ == "__main__":
12     wrapper_function()
```

Run - 2.12

Run: primer1 x

C:\Users\erken\Desktop\python\Примеры\2.12\venv\Scripts\python.exe

Hello world!

Рисунок 4. Результат выполнения программы задания 1

```
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def wrapper_function():
6      def hello_world():
7          print('Hello world!')
8      hello_world()
9
10
11  ▶  if __name__ == "__main__":
12      wrapper_function()
```

Run - 2.12

Run: primer2 x

▶ ↑ C:\Users\erken\Desktop\python\Примеры\2.12\venv\Scripts\python.exe C:/Users
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x0000029D58615480>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки

Рисунок 5. Результат выполнения программы задания 2

```
#!/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()
```

Рисунок 6. Задание 3

```

1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      def benchmark(func):
6          import time
7
8          def wrapper(*args, **kwargs):
9              start = time.time()
10             return_value = func(*args, **kwargs)
11             end = time.time()
12             print('[*] Время выполнения: {} секунд.'.format(end - start))
13             return return_value
14
15         return wrapper
16
17
18     @benchmark
19     def fetch_webpage(url):
20         import requests
21         webpage = requests.get(url)
22         return webpage.text
23
24
25  ▶  if __name__ == "__main__":
26         webpage = fetch_webpage('https://google.com')
27         print(webpage)

```

Рисунок 7. Задание 4

5. Индивидуальное задание (вариант 8)

Объявите функцию, которая вычисляет площадь круга и возвращает вычисленное значение. В качестве аргумента ей передается значение радиуса. Определите декоратор для этой функции, который выводит на экран сообщение: «Площадь круга равна = ». В строке выведите числовое значение с точностью до сотых. Примените декоратор к функции и вызовите декорированную функцию.

```
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  from math import pi
6
7
8  def func_show(func):
9      def wrapper(r):
10         res = float("{:.2f}".format(func(r)))
11         print(f"Площадь круга: {res}")
12         return res
13
14     return wrapper
15
16
17     @func_show
18     def get_sq(r):
19         return pi * pow(r, 2)
20
21
22  ▶  if __name__ == "__main__":
23     get_sq(r=5)
```

PC Run - 2.12

Run: main ×

▶ ↑ C:\Users\erken\Desktop\python\Примеры\2.12\venv\Scripts\python.exe C:/Users/
Площадь круга: 78.54
⚙ ↓

Рисунок 8. Индивидуальное задание

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?

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 позволяет 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 функций `wrapper`. В конце декоратора происходит возвращение функции `wrapper`.

6. Самостоятельно изучить как можно передать параметры.

Достаточно обернуть функцию декоратор в другую функцию, которая будет принимать аргументы. И сделать вывод функций `wrapper` и `decorator` будет принимать аргументы. И сделать вывод функций `wrapper` и `decorator`.