

**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  
Декораторы функций в языке Python  
по дисциплине «Технологии программирования»**

Выполнил студент группы ИВТ-б-о-20-1

Ищенко Т.С. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

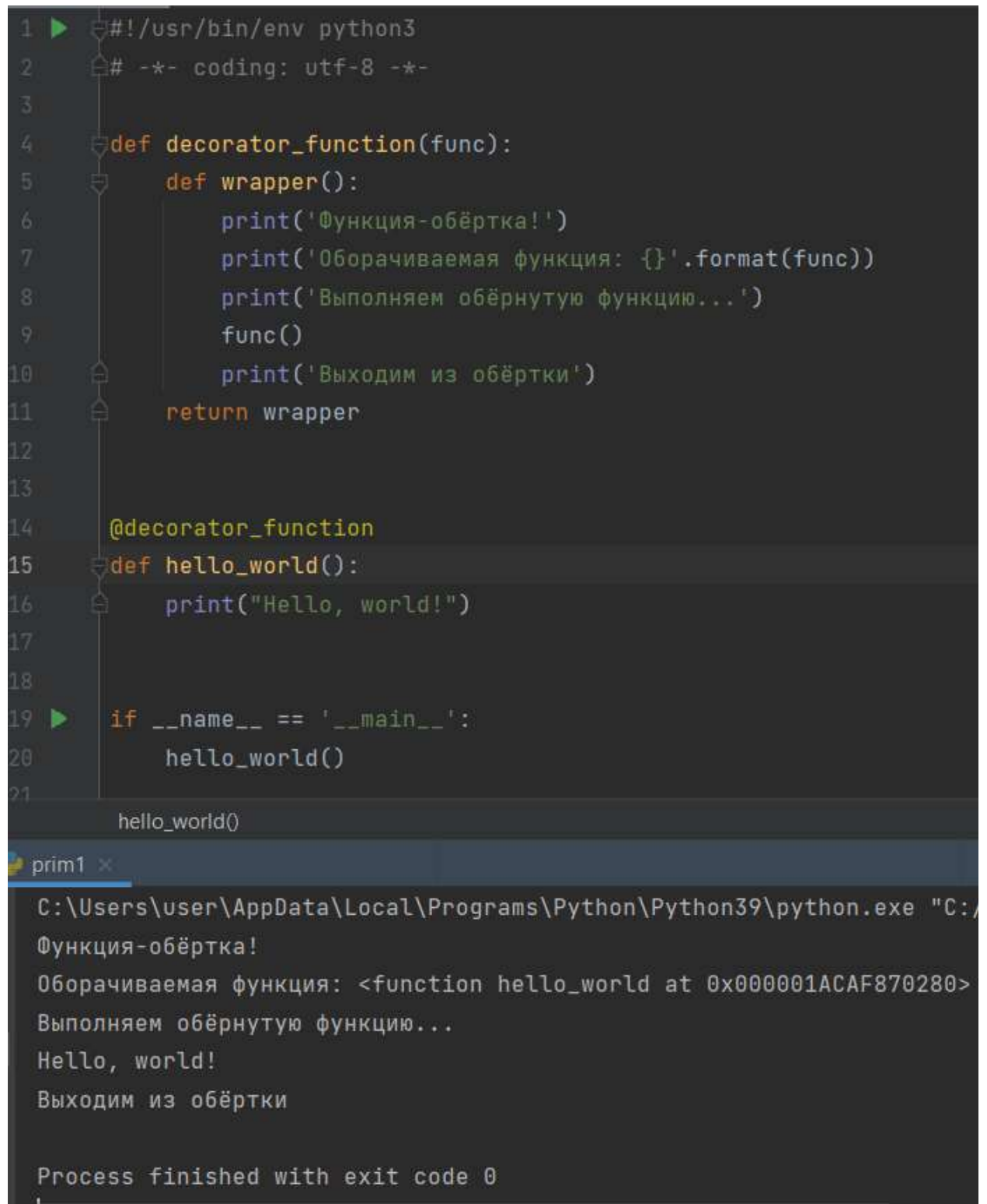
Проверила Воронкин Р.А. \_\_\_\_\_

(подпись)

Ставрополь 2021

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

### 1. Произвёл выполнение примера



```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      def decorator_function(func):
5          def wrapper():
6              print('Функция-обёртка!')
7              print('Оборачиваемая функция: {}'.format(func))
8              print('Выполняем обёрнутую функцию...')
9              func()
10             print('Выходим из обёртки')
11         return wrapper
12
13
14     @decorator_function
15     def hello_world():
16         print("Hello, world!")
17
18
19     if __name__ == '__main__':
20         hello_world()
21
```

hello\_world()

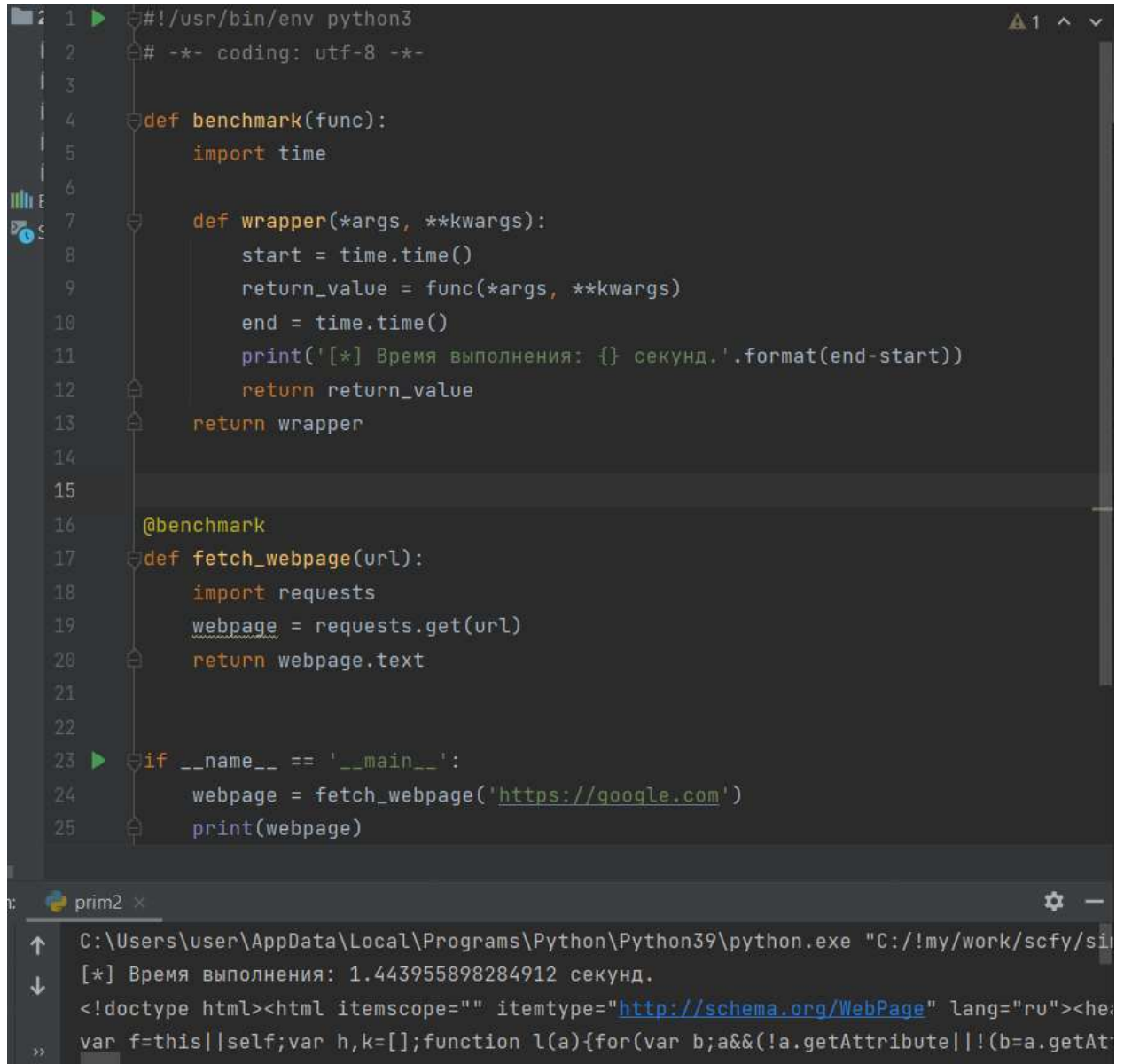
prim1 x

C:\Users\user\AppData\Local\Programs\Python\Python39\python.exe "C:\  
Функция-обёртка!  
Оборачиваемая функция: <function hello\_world at 0x000001ACAF870280>  
Выполняем обёрнутую функцию...  
Hello, world!  
Выходим из обёртки

Process finished with exit code 0

Рисунок 1 – Результат выполнения примера

## 2. Произвёл выполнение примера



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

The screenshot shows a Python IDE with a dark theme. The code defines a `benchmark` decorator that uses `time.time()` to measure the execution time of a function. The decorator is applied to the `fetch_webpage` function, which uses `requests.get` to fetch the content of `https://google.com`. The main block of the script calls `fetch_webpage` and prints the result. The output window at the bottom shows the execution time as 1.443955898284912 seconds and the beginning of the HTML response from Google.

Рисунок 2 – Результат выполнения второго примера

3. Выполнил индивидуальное задание, согласно методическим рекомендациям

```
13
14 def decorator(func):
15     def culc(*args):
16         return func(sum(args))
17     return culc
18
19
20 @decorator
21 def otvet(storoni):
22     print(f'Периметр фигуры равен: {storoni}')
23
24
25 if __name__ == '__main__':
26     otvet(*list(map(float, input('Введите стороны многоугольник: ').split()))))
27
```

otvet()

idz x

C:\Users\user\AppData\Local\Programs\Python\Python39\python.exe "C:/!my/work/scfy/si  
Введите стороны многоугольник: 2 2 2 2  
Периметр фигуры равен: 8.0

Рисунок 3 – Результат выполнения индивидуального задания

#### 4. Произвёл проверку созданных файлов на соответствие PEP8

```
(tools) C:\!my\work\scfy\sim_3\tech prog\2.12>flake8
.\idz.py:6:80: E501 line too long (85 > 79 characters)
.\prim1.py:20:18: W292 no newline at end of file

(tools) C:\!my\work\scfy\sim_3\tech prog\2.12>flake8
.\idz.py:5:80: E501 line too long (94 > 79 characters)

(tools) C:\!my\work\scfy\sim_3\tech prog\2.12>flake8

(tools) C:\!my\work\scfy\sim_3\tech prog\2.12>
```

Рисунок 4 – Результат выполнения проверки на PEP8

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 декоратору, а не декорируемой функции? – Достаточно обернуть функцию декоратор в другую функцию, которая будет принимать аргументы. И сделать вывод функций `wrapper` и `decorator`. =

Вывод: в ходе выполнения лабораторной работы приобрел навыки по работе с декораторами функций при написании программ с помощью языка программирования Python версии 3.x.