

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: «Программирования на Python»
Тема: « Декораторы
функций в языке Python»

Выполнил: студент 1 курса
группы ИВТ-б-о-21-1
Толубаев Рамиль Ахметович

Ставрополь 2022

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

Ход работы:

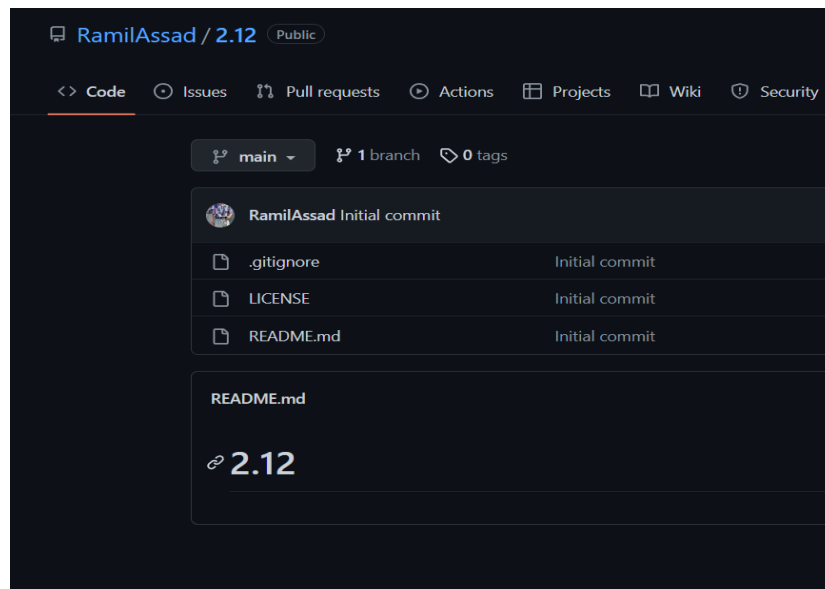


Рисунок 1.Создал репозиторий

Пример 1

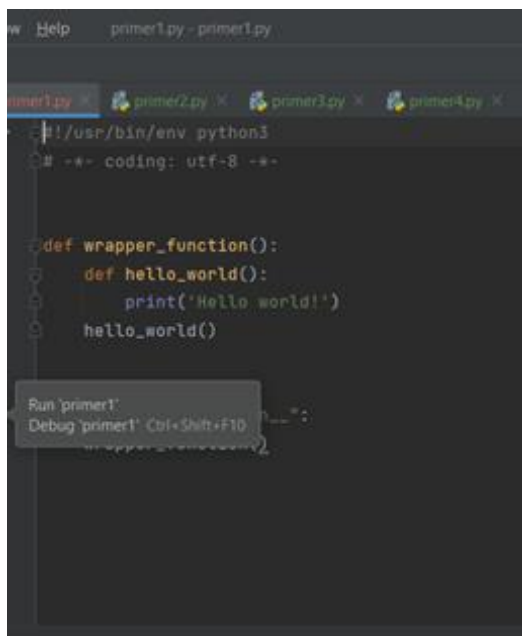


Рисунок 2.Код первого примера

Пример 2

```
Help  primer1.py - primer2.py

primer1.py x primer2.py x primer3.py x primer4.py x ind.py x
#!/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()
```

Рисунок 3.Код второго примера

Пример 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()
```

Рисунок 4.Код третьего примера

Пример 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)
```

Рисунок 5. Код четвертого примера

Индивидуальное задание

Объявите функцию, которая принимает строку на кириллице и преобразовывает ее в латиницу, используя следующий словарь для замены русских букв на соответствующее латинское написание.

Функция должна возвращать преобразованную строку. Замены делать без учета регистра (исходную строку перевести в нижний регистр – малые буквы). Определите декоратор с параметром `chars` и начальным значением " !? " , который данные символы преобразует в символ "-" и, кроме того, все подряд идущие дефисы (например, "--" или "---") приводит к одному дефису. Полученный результат должен возвращаться в виде строки. Примените декоратор со значением `chars="?!:;,."` к функции и вызовите декорированную функцию. Результат отобразите на экране.

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

def pink(func):

    def pnk(text, chars=" !?"):

        print(text)

        h = ''.join(map(lambda x: x if x not in chars else '-', func(text)))
        print(h)
        while '---' in h:
            h = h.replace('---', '-')
            print(h)
        return h

    return pnk

@pink
def wrapper(text):
    p = text.lower()
    t = {'ё': 'yo', 'а': 'a', 'б': 'b', 'в': 'v', 'г': 'g', 'д': 'd', 'е': 'e',
        'ж': 'zh', 'з': 'z', 'и': 'i', 'й': 'y', 'к': 'k', 'л': 'l',
        'м': 'm', 'н': 'n', 'о': 'o', 'п': 'p', 'р': 'r', 'с': 's', 'т': 't',
        'у': 'u', 'ф': 'f', 'х': 'h', 'ц': 'c', 'ч': 'ch', 'ш': 'sh',
        'щ': 'shch', 'ь': '', 'ы': 'y', 'ё': '', 'а': 'e', 'ё': 'yu',
        'я': 'ya'}
    return p.translate({ord(key): t[key] for key in t})
```

Рисунок 6. Код индивидуального задания

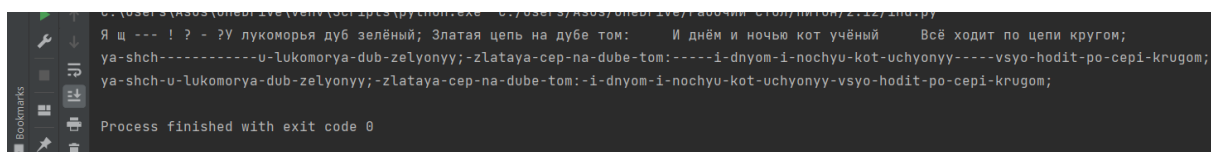


Рисунок 7. Результат индивидуального задания

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 функция `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`.