

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО РАБОТЕ №2.12
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Кондратенко Даниил Витальевич
1 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Декораторы функций в языке Python.

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

Порядок выполнения работы:

Задание 1.

Изучил теоретический материал работы, создал общедоступный репозиторий на GitHub, в котором использована лицензий MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами.

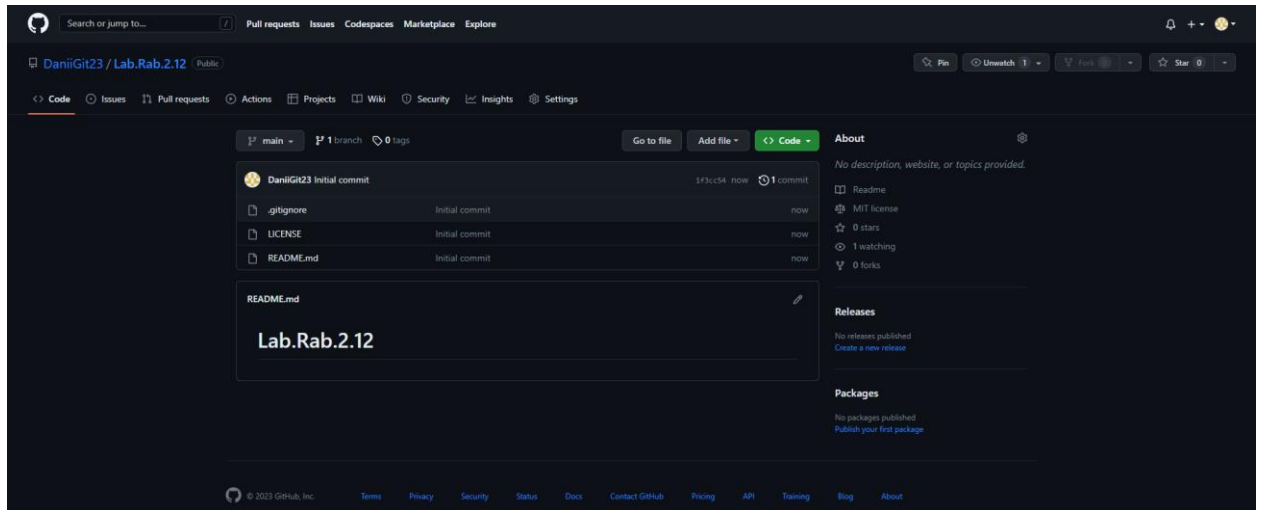


Рисунок 1 – Новый репозиторий

Задание 2.

Проклонировал свой репозиторий на свой компьютер.

Организовал свой репозиторий в соответствие с моделью ветвления git-flow, появилась новая ветка develop.

```
C:\Users\HUAWEI>git clone https://github.com/DaniiGit23/Lab.Rab.2.12.git
Cloning into 'Lab.Rab.2.12'...
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.

C:\Users\HUAWEI>cd C:\Users\HUAWEI\Lab.Rab.2.12
Синтаксическая ошибка в имени файла, имени папки или метке тома.

C:\Users\HUAWEI>cd C:\Users\HUAWEI\Lab.Rab.2.12

C:\Users\HUAWEI\Lab.Rab.2.12>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? [] t
Hooks and filters directory? [C:/Users/HUAWEI/Lab.Rab.2.12/.git/hooks]

C:\Users\HUAWEI\Lab.Rab.2.12>git status
On branch develop
nothing to commit, working tree clean
```

Рисунок 2 – Клонирование и модель ветвления git-flow

Реализовывал примеры и индивидуальные задания на основе ветки develop, без создания дополнительной ветки feature/(название ветки) по указанию преподавателя.

Задание 3.

Создал проект PyCharm в папке репозитория.

Работа с примером №1.

Добавил новый файл *primer1.py*

Работа декоратора №1.

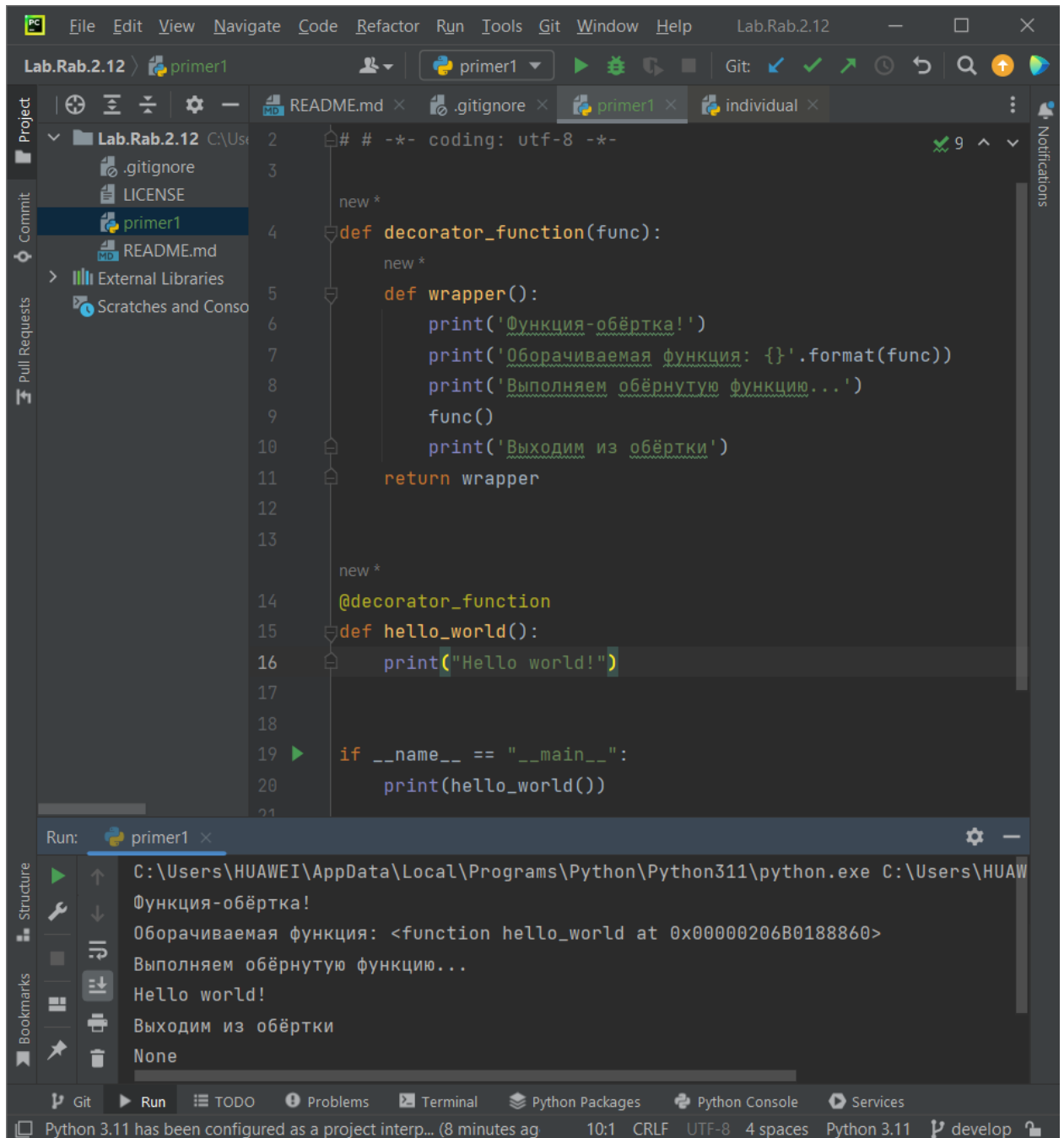


Рисунок 3 – Выполнение примера и его результат

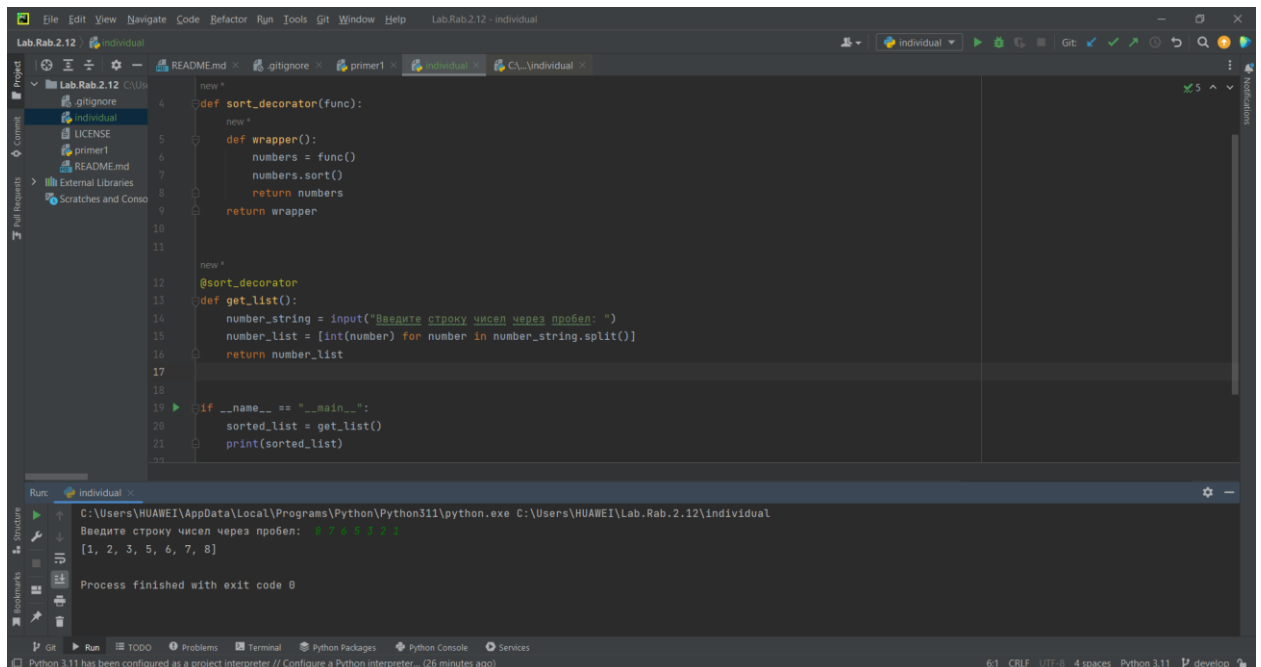
Зафиксировал данные изменения.

Задание 4.

Работа с примером индивидуальным заданием.

Добавил новый файл *individual.py*

Условие индивидуального задания: на вход программы поступает строка из целых чисел, записанных через пробел. Напишите функцию `get_list`, которая преобразовывает эту строку в список из целых чисел и возвращает его. Определите декоратор для этой функции, который сортирует список чисел, полученный из вызываемой в нем функции. Результат сортировки должен возвращаться при вызове декоратора. Вызовите декорированную функцию `get_list` и отобразите полученный отсортированный список на экране.



```
def sort_decorator(func):
    def wrapper():
        numbers = func()
        numbers.sort()
        return numbers
    return wrapper

@sort_decorator
def get_list():
    number_string = input("Введите строку чисел через пробел: ")
    number_list = [int(number) for number in number_string.split()]
    return number_list

if __name__ == "__main__":
    sorted_list = get_list()
    print(sorted_list)
```

Run: individual

C:\Users\HUAWEI\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HUAWEI\Lab.Rab.2.12\individual

Введите строку чисел через пробел: 1 2 3 5 6 7 8

[1, 2, 3, 5, 6, 7, 8]

Process finished with exit code 0

Рисунок 4 – Выполнение примера и его результат

Зафиксировал данные изменения.

Задание 5.

Слил ветку develop с веткой main, для начала перешел на ветку main (*git checkout main*), далее с помощью команды *git merge develop* слил ветку с основной и отправил на удаленный сервер.

```
C:\Users\HUAWEI\Lab.Rab.2.12>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

C:\Users\HUAWEI\Lab.Rab.2.12>git merge develop
Merge made by the 'ort' strategy.
 individual | 21 ++++++
 primer1    | 20 ++++++
 2 files changed, 41 insertions(+)
 create mode 100644 individual
 create mode 100644 primer1

C:\Users\HUAWEI\Lab.Rab.2.12>
```

Рисунок 5 – Слияние двух веток

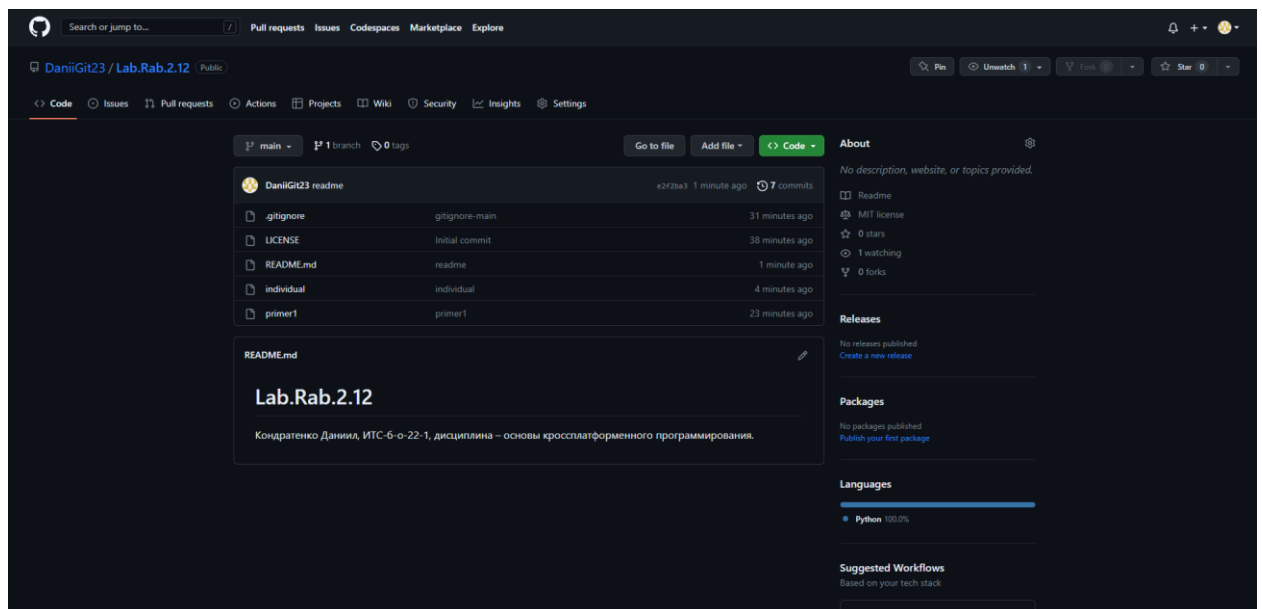


Рисунок 6 – Удаленный сервер

Ссылка на репозиторий: <https://github.com/DaniiGit23/Lab.Rab.2.12.git>

Контрольные вопросы:

1) Что такое декоратор?

Декоратор в языке программирования Python - это функция, которая принимает на вход другую функцию и расширяет ее функционал без изменения ее исходного кода.

2) Почему функции являются объектами первого класса?

В Python функции являются объектами первого класса, потому что они могут быть присвоены переменным, переданы в качестве параметров другим функциям, возвращены в качестве результатов из функций, хранены в структурах данных, например списки, словари и т.д. Это позволяет использовать функции гибко и создавать выразительный и простой код.

3) Каково назначение функций высших порядков?

Функции высших порядков - это функции, которые принимают одну или несколько функций в качестве аргументов и/или возвращают другую функцию в качестве результата. Такие функции могут использоваться для решения таких задач, как фильтрация, суммирование, упорядочивание и т.д. списка элементов, а также для создания новых функций с помощью композиции существующих (комбинирование функций).

4) Как работают декораторы?

При использовании декоратора, функция или метод класса передаются в качестве аргумента в другую функцию, которая и выполняет нужное действие, дополняя тем самым исходную функциональность.

Для создания декоратора в Python используют функцию, которая принимает в качестве аргументов другую функцию и возвращает функцию, которая представляет из себя декорированную версию первоначальной функции.

5) Какова структура декоратора функций?

1. Объявление декоратора (@decorator). Это используется для указания того, что следующая функция будет декорирована.

2. Определение функции-обертки. Эта функция принимает исходные аргументы декорируемой функции и передает их далее. Она может выполнять дополнительный код, который будет запускаться до и после вызова декорируемой функции.

3. Возврат функции-обертки. Это возвращает обернутую функцию как объект, который можно вызвать.

б) Самостоятельно изучить как можно передать параметры декоратору, а не декорируемой функции?

В Python параметры декоратору могут быть переданы с помощью вложенной функции-обертки (wrapper), которая будет вызвана после декорирования функции, но перед выполнением ее кода.

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