

**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: «Программирование на Python»

Тема: «Декораторы функций в языке Python»

Выполнил: студент 2 курса

группы ИВТ-б-о-21-1

Назаров Никита Назаров

Выполнение работы.

```
H:\cross\git\nazarov>git clone https://github.com/NikitaNazarov179/2.12.git
Cloning into '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.
```

Рисунок 1 – клонирование репозитория

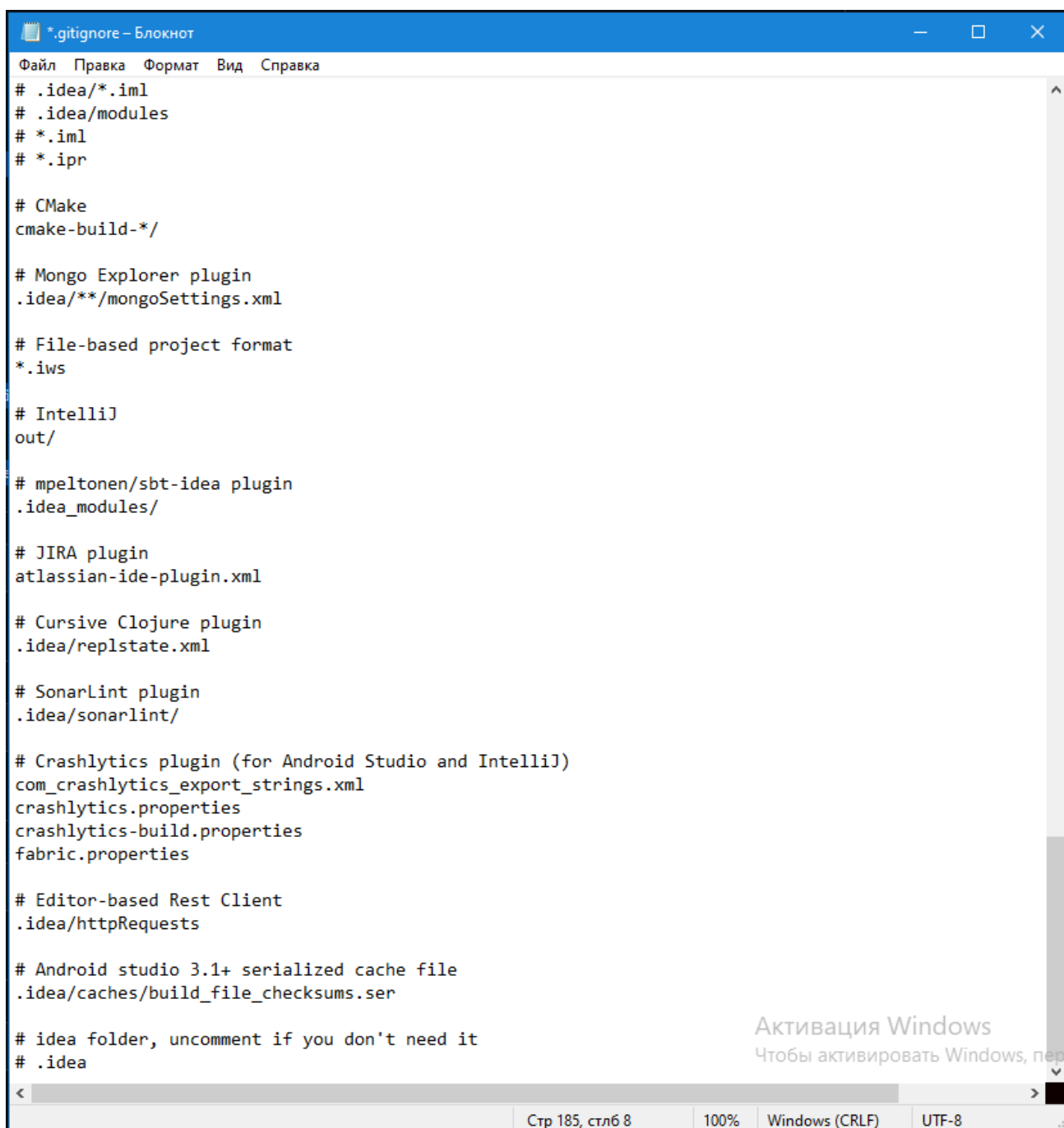
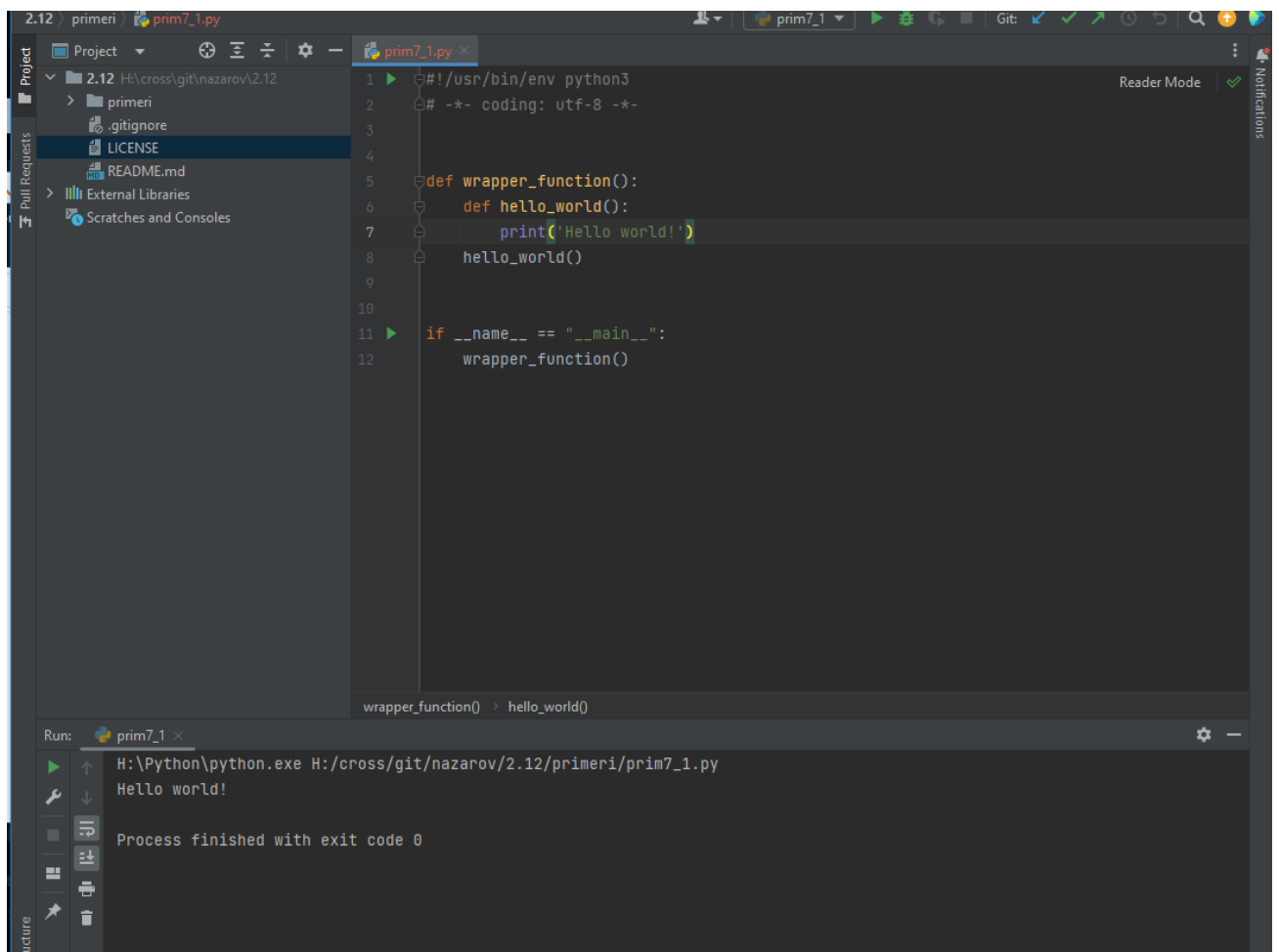


Рисунок 2 – редактирование gitignore

```
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? [H:/cross/git/nazarov/2.12/.git/hooks]
```

Рисунок 3 – организация репозитория в соответствии с git flow



The screenshot shows an IDE window with a project named 'prim7_1'. The left sidebar displays the project structure, including files like '2.12', 'prim7_1.py', '.gitignore', 'LICENSE', and 'README.md'. The main editor area shows the content of 'prim7_1.py', which is a Python script. The script defines a 'wrapper_function()' that calls 'hello_world()', which prints 'Hello world!'. The script is executed, and the output 'Hello world!' is shown in the 'Run' console at the bottom. The console also indicates that the process finished with exit code 0.

```
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: prim7_1

```
H:\Python\python.exe H:/cross/git/nazarov/2.12/prim7_1.py
Hello world!

Process finished with exit code 0
```

Рисунок 4 – проработал 1 пример

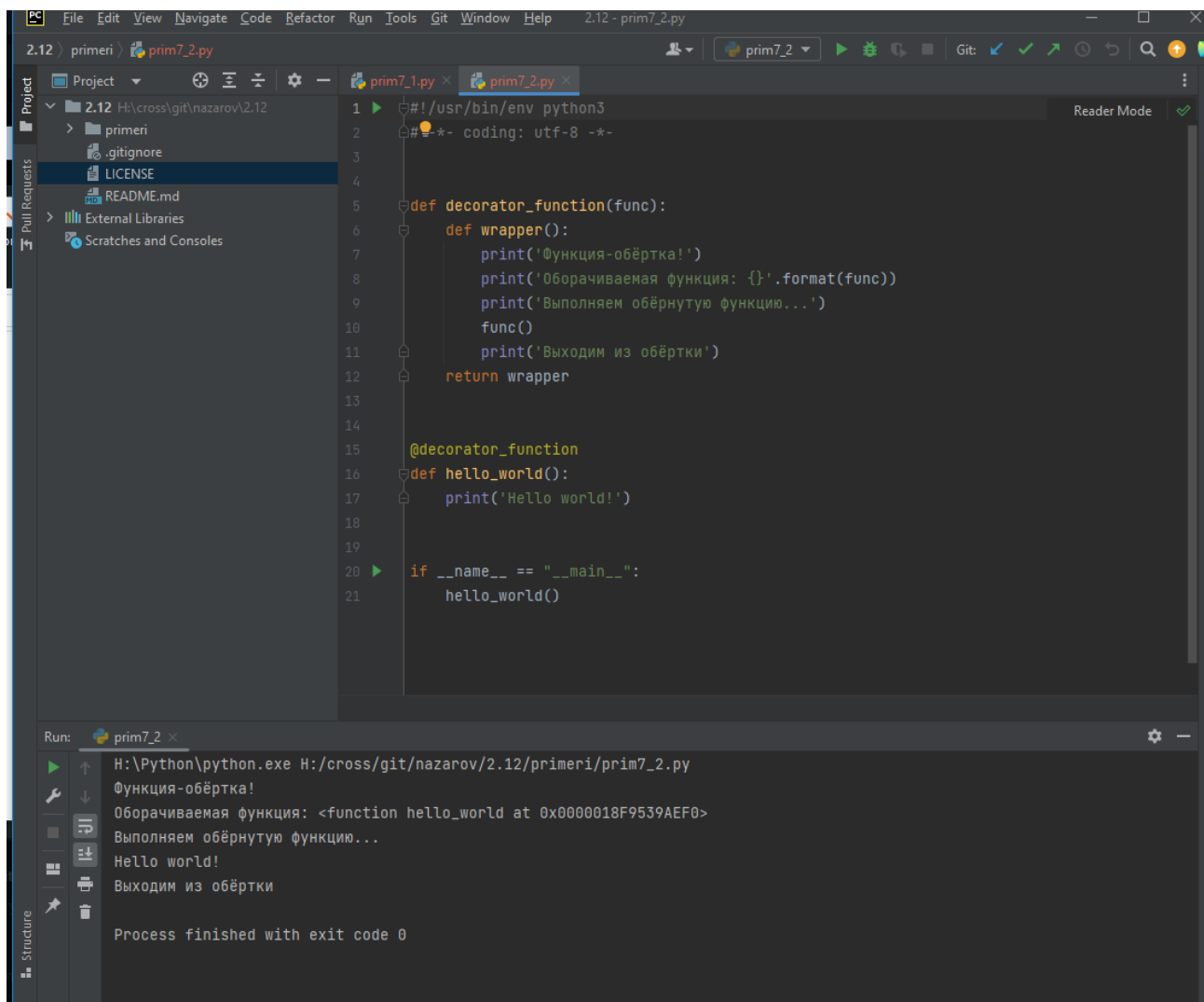


Рисунок 5 – проработал 2 пример

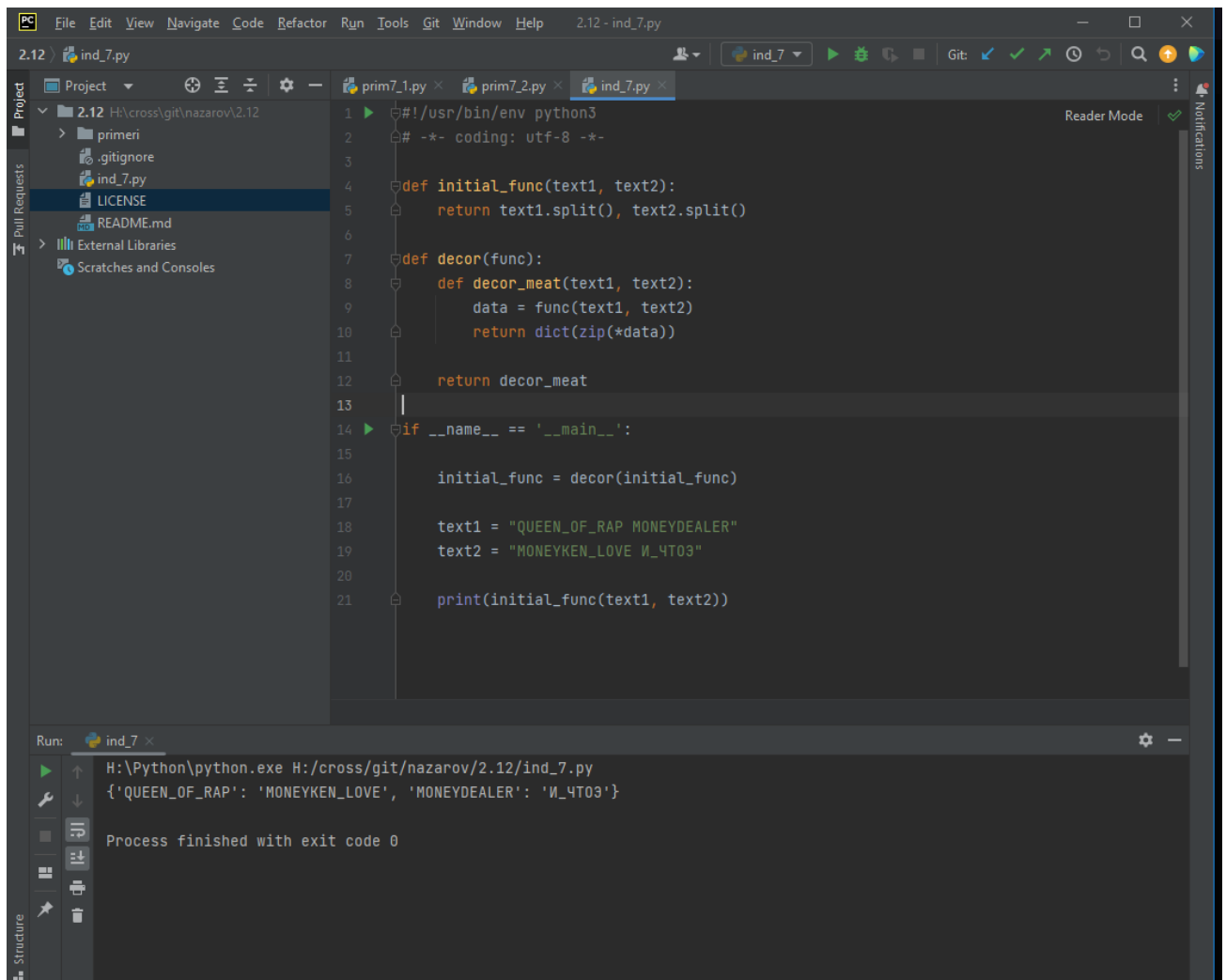


Рисунок 6 – выполнил индивидуальное задание

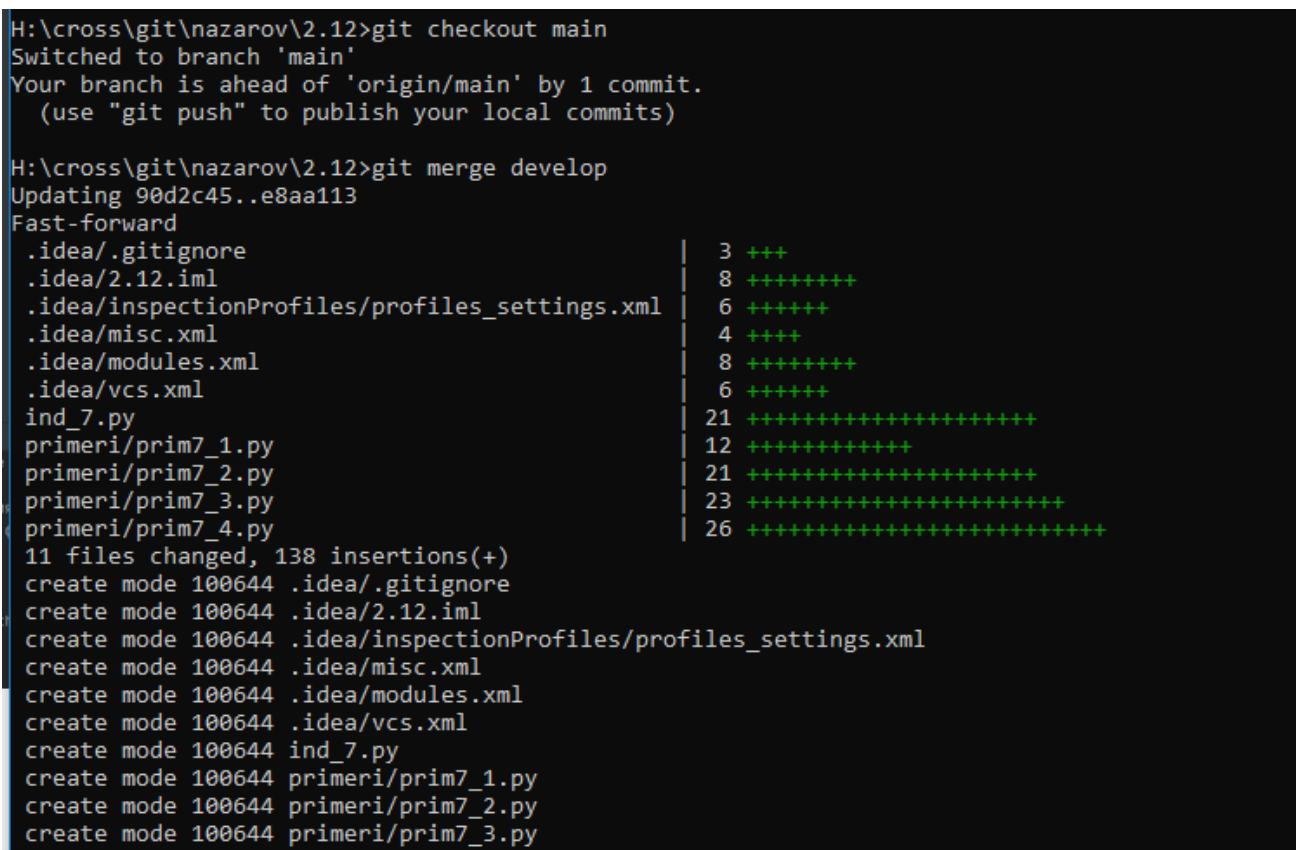


Рисунок 7 – слияние веток и переход на основную

```
H:\cross\git\nazarov\2.12>git push
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 12 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (21/21), 4.58 KiB | 1.53 MiB/s, done.
Total 21 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/NikitaNazarov179/2.12.git
   fc9c77a..e8aa113  main -> main
```

Рисунок 8 – слияние веток

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 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 декоратору, а не декорируемой функции?

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

