**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая Кибернетика и Информационные технологии»

Лабораторная работа №1

**Основы работы с системой контроля версий**

Выполнил: Студент группы

БВТ2203

Георгян Сергей

Москва

2023

**Цель работы:** изучение базовых функций и подходов к командной разработке в системе контроля версий Git

**Задание:** определить проект (состоящий из нескольких файлов), который будет добавлен под версионный контроль, и произвести ряд операций в СКВ, включающих:

1. Создание репозитория или клонирование существующего
2. Внесение ряда изменений в файл(ы) и выполнение коммитов (кол-во не меньше 2)
3. Откат изменений до одного из предыдущих коммитов двумя разными способами
4. Создание, переименование и удаление ветки
5. Слияние ветки с веткой мастер с решением конфликта содержания файлов
6. Настройка игнорирования git’ом файлов определенного расширения с использованием glob-шаблонов
7. Вывод лога коммитов в измененном пользовательском формате
8. Настройка удалённого репозитория: получение и отправка изменений в репозиторий
9. Организация совместной работы над проектом: клонирование чужого репозитория, выполнение коммитов, отправка изменений

Ход работы:

Создаем проект, инициализируем новый репозиторий при помощи команды ‘git init’, добавляем нужные файлы при помощи команды ‘git add’ и проверяем их при помощи команды ‘git status’

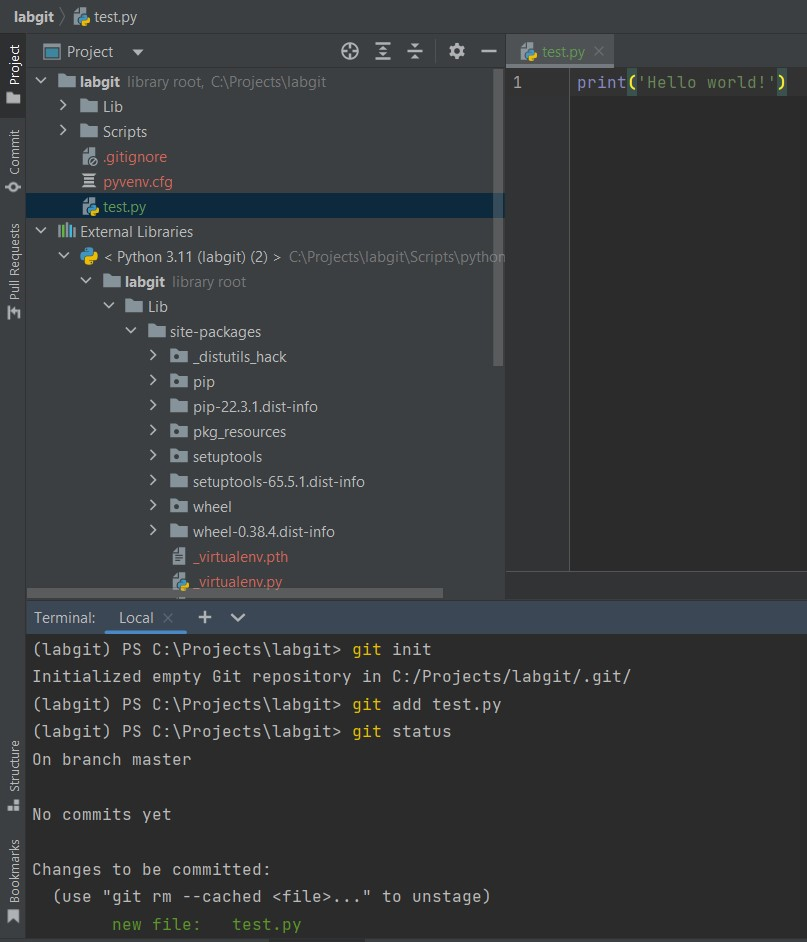


Рис.1 – инициализация репозитория

Далее делаем первый коммит ‘git commit -m (название коммита)’



Рис.2 – первый коммит

Далее вносим изменения в файл test.py посредством изменения текста с ‘Hello World’ на ‘Goodbye World’ добавляем файл в репозиторий git и делаем второй коммит

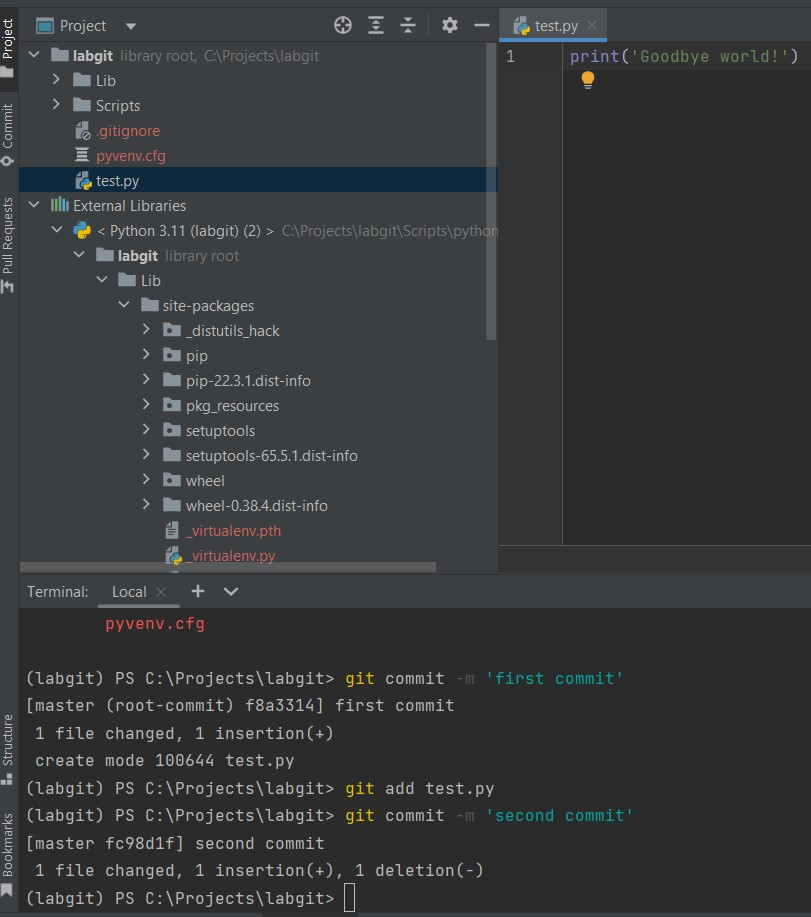


Рис.3 – второй коммит

Также делаем и третий коммит изменив текст с ‘Goodbye world’ на ‘Good morning world’

Посмотрим на наши коммиты используя команду ‘git log’

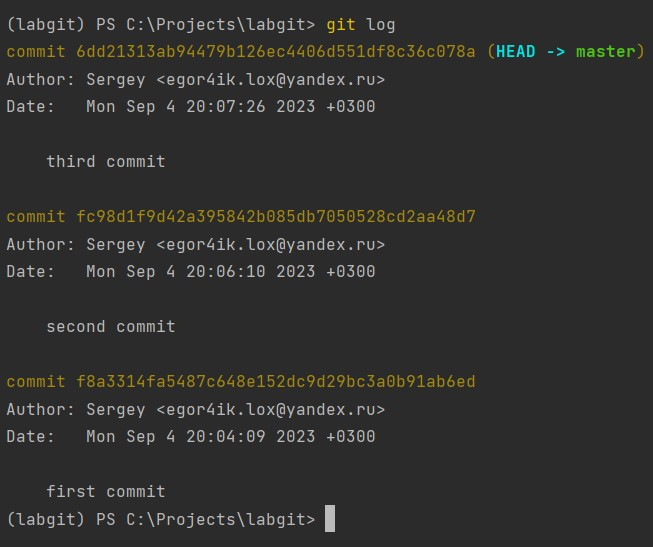


Рис.4 – список коммитов

Теперь откатимся на один коммит назад при помощи команды ‘git reset --hard HEAD~1’

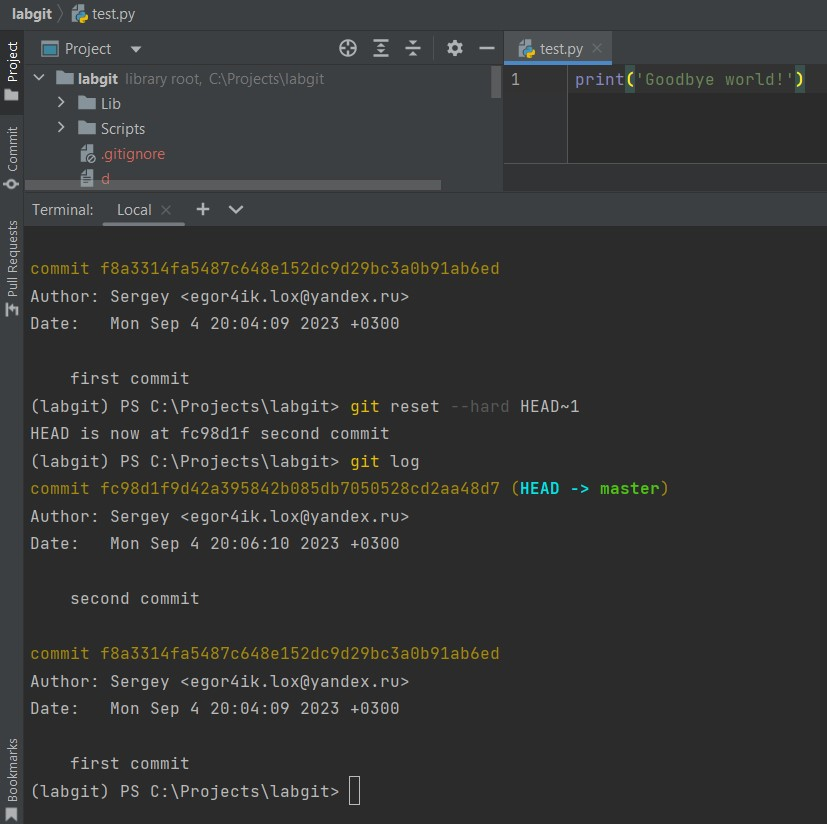


Рис.5 – откат коммита командой reset

Выведя список коммитов мы видим что эта команда полностью удалила текущий коммит. Теперь откатимся на еще один коммит назад при помощи команды ‘git revert HEAD –no-edit’

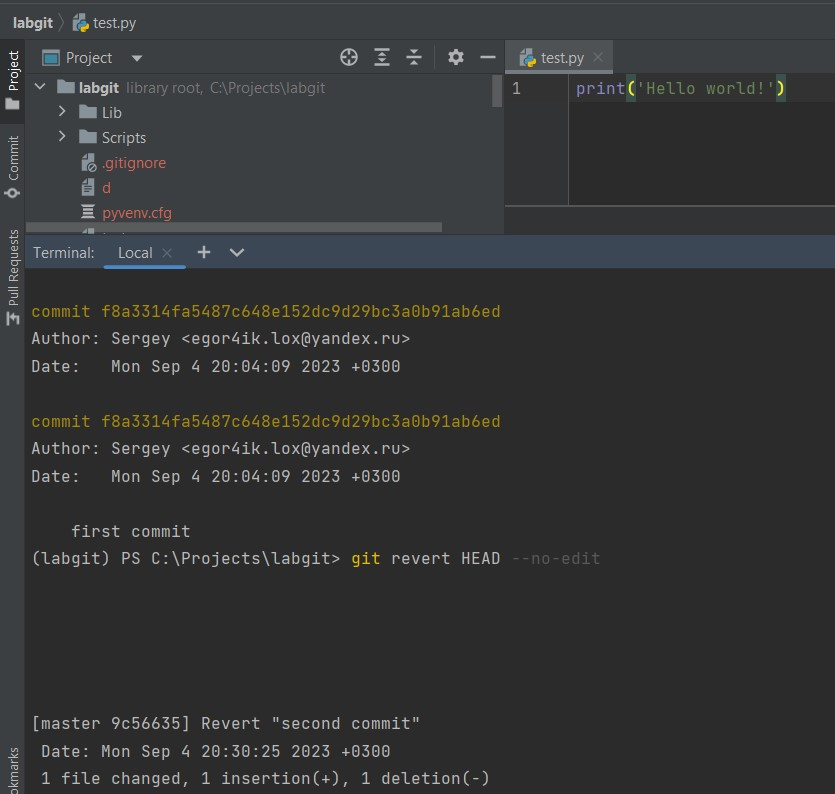


Рис.6 – откат коммита командой revert

У нас было два коммита, при использовании этой команды мы получили новый коммит, который копирует тот, до которого мы хотели откатиться.

Создаем новую ветку и называем ее ‘sergo’ командой git branch sergo’, переходим на нее командой ‘git checkout sergo’ и переименовываем командой ‘git branch -m sergey’. Также выводим список всех веток командой ‘git branch’

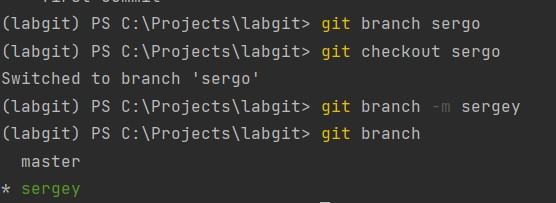


Рис.7 – создание и переименование ветки

Теперь удалим эту ветку при помощи команды ‘git branch -d sergey’

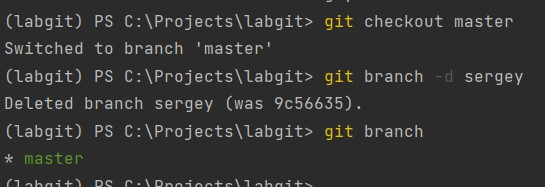


Рис.8 – удаление ветки

Еще раз создаем новую ветку ‘sergey’, изменяем файл test.py и делаем коммит, также переходим на ветку ‘master’ изменяем уже тут файл test.py и снова делаем коммит, теперь при попытки слияния двух веток будет возникать конфликт.

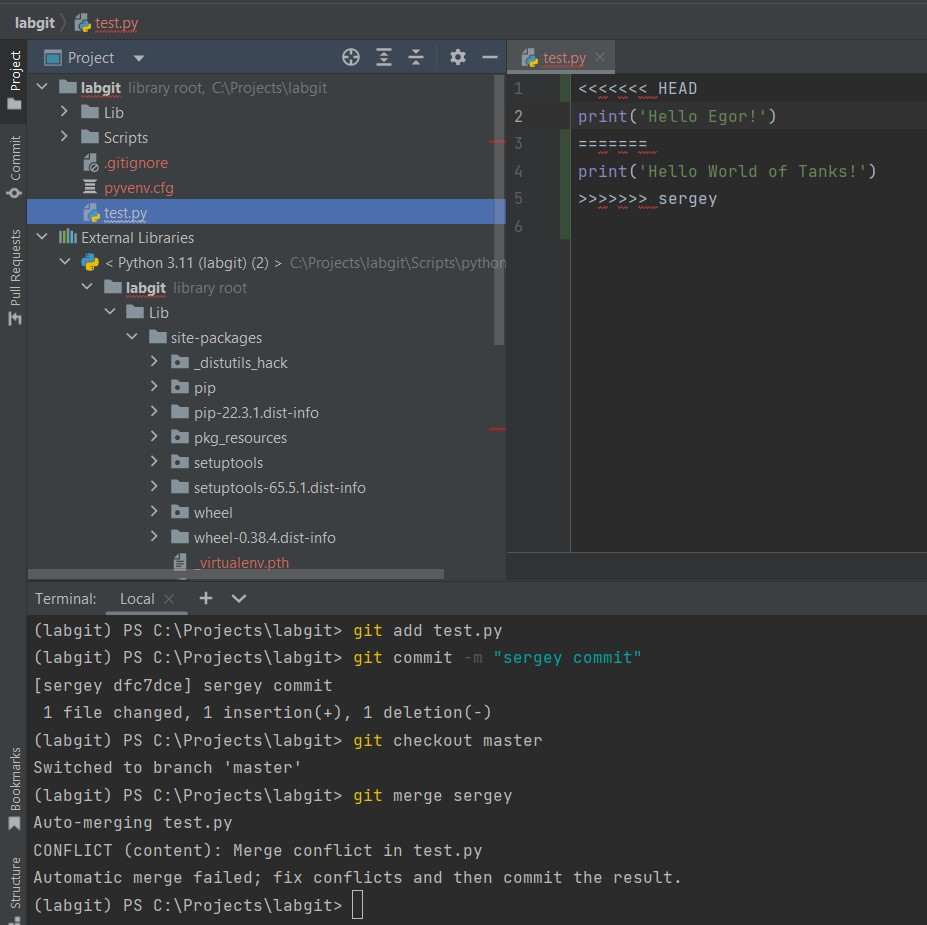


Рис.9 – конфликт слияния

Для решения этого конфликта изменяем файл test.py посредством изменения текста на один из данных и делаем коммит, после выходим из текстового редактора и получаем сообщение об успешном слиянии веток.

 Рис.10 – слияние веток

В папке проекта создаем файл .gitignore и в нем прописываем шаблоны для игнорирования определенных файлов при попытке добавить их в git.

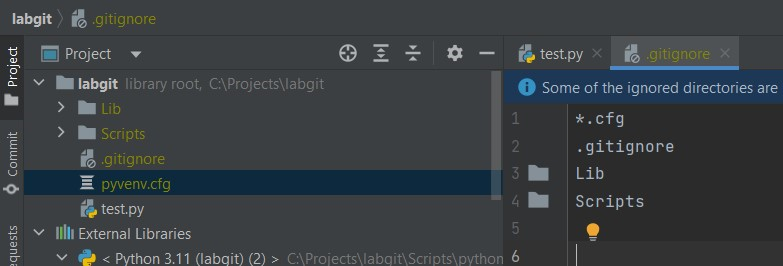


Рис.11 – содержимое ‘.gitignore’

Теперь в моем случае при использовании команды ‘git add .’ не будут добавляться директории ‘Lib’ и ‘Scripts’, файл ‘.gitignore’ и все файлы с форматом ‘cfg’

Далее выводим список коммитов в пользовательском формате при помощи команды (git log –prety=format:”%h - %an, %ad : %s”)

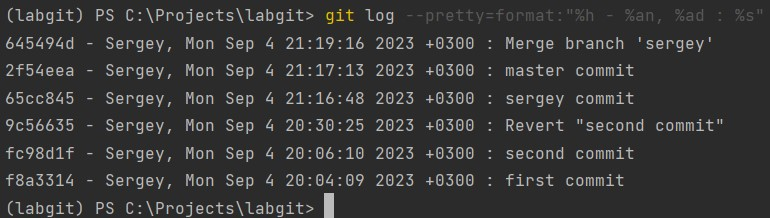


Рис.12 – список коммитов в польз. формате

Теперь создадим удаленный репозиторий на платформе Github

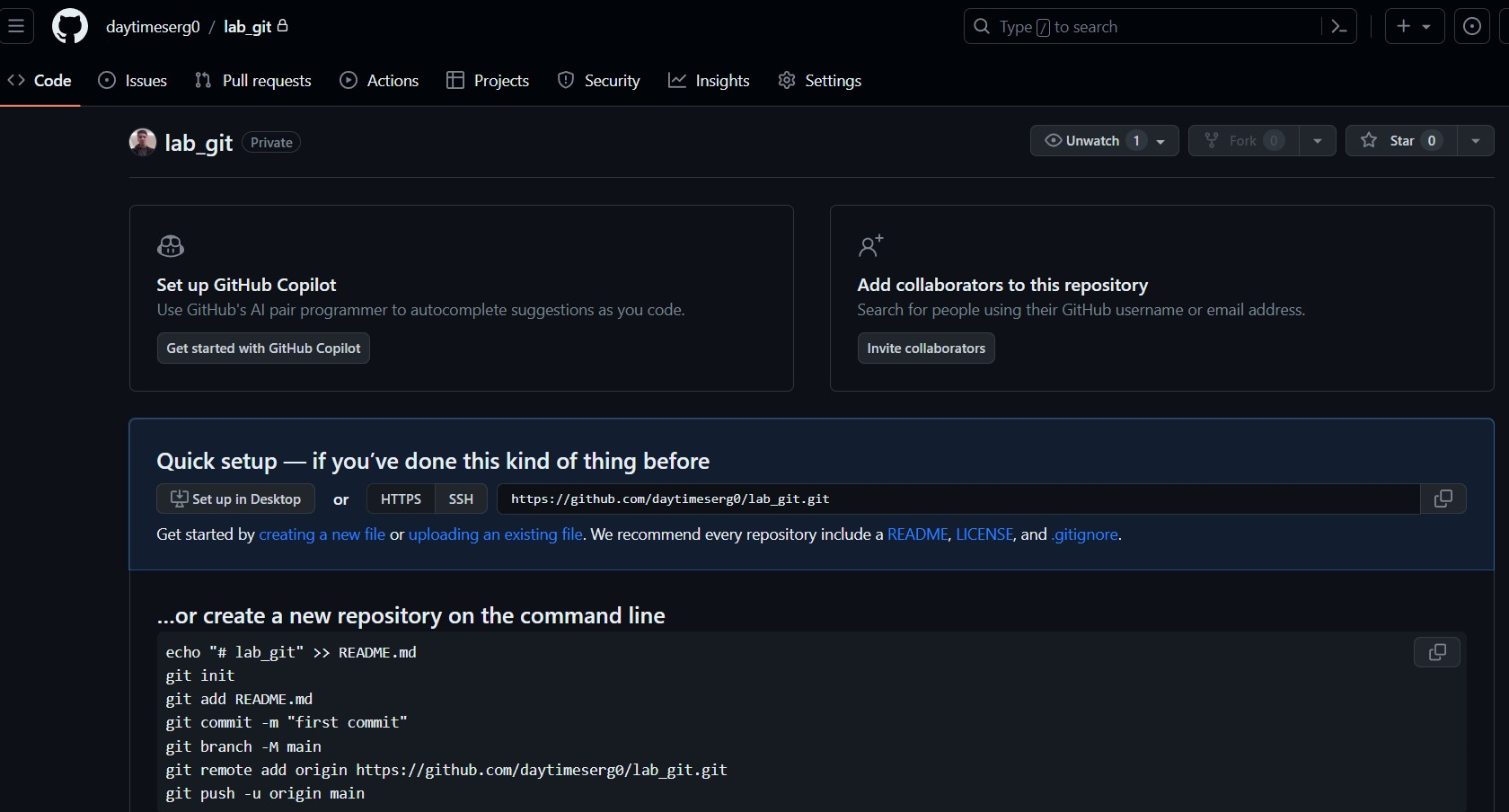


Рис.13 – создание удаленного репозитория Github

Далее связываем удаленный репозиторий с локальным командой ‘git remote add origin URL\_удаленного\_репозитория’. И отправляем изменения из локального в удаленный при помощи команды ‘git push origin master’

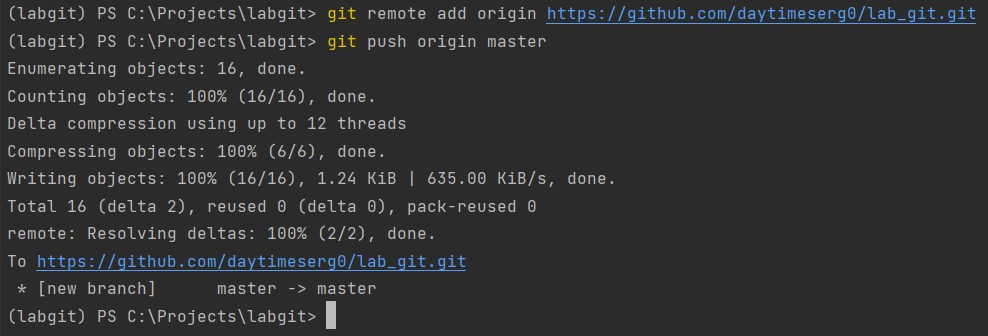


Рис.14 – отправка данных на удаленный репозиторий

Заходим на Github и видим изменения репозитория. Теперь провернем тоже самое но в обратном порядке. Создадим новый файл в удаленном репозитории и назовем его ‘hi.txt’



Рис.15 изменение удаленного репозитория

Для получения изменений из удаленного репозитория воспользуемся командой ‘git pull origin master’ и увидим появления файла ‘hi.txt’ в нашем локальном репозитории

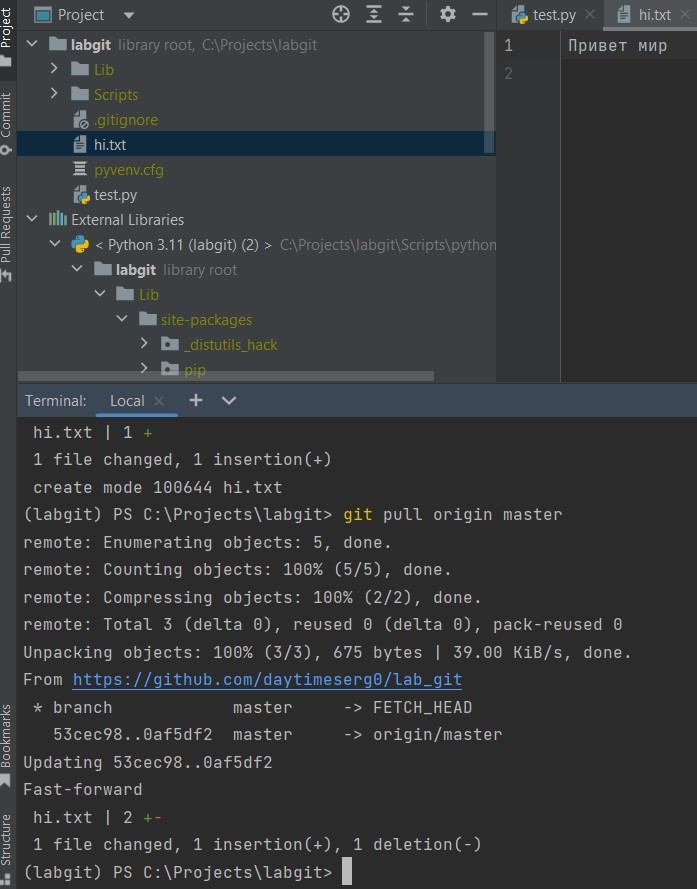


Рис.16 – получение данных из удаленного репозитория

Для выполнения следующего задания я предоставил доступ к своему удаленному репозиторию другу, а он соответственно мне.

Копируем ссылку на его репозиторий создаем новую директорию и прописываем в консоли ‘git clone URL\_репозитория’ и видим новую папку со всеми файлами удаленного репозитория друга.

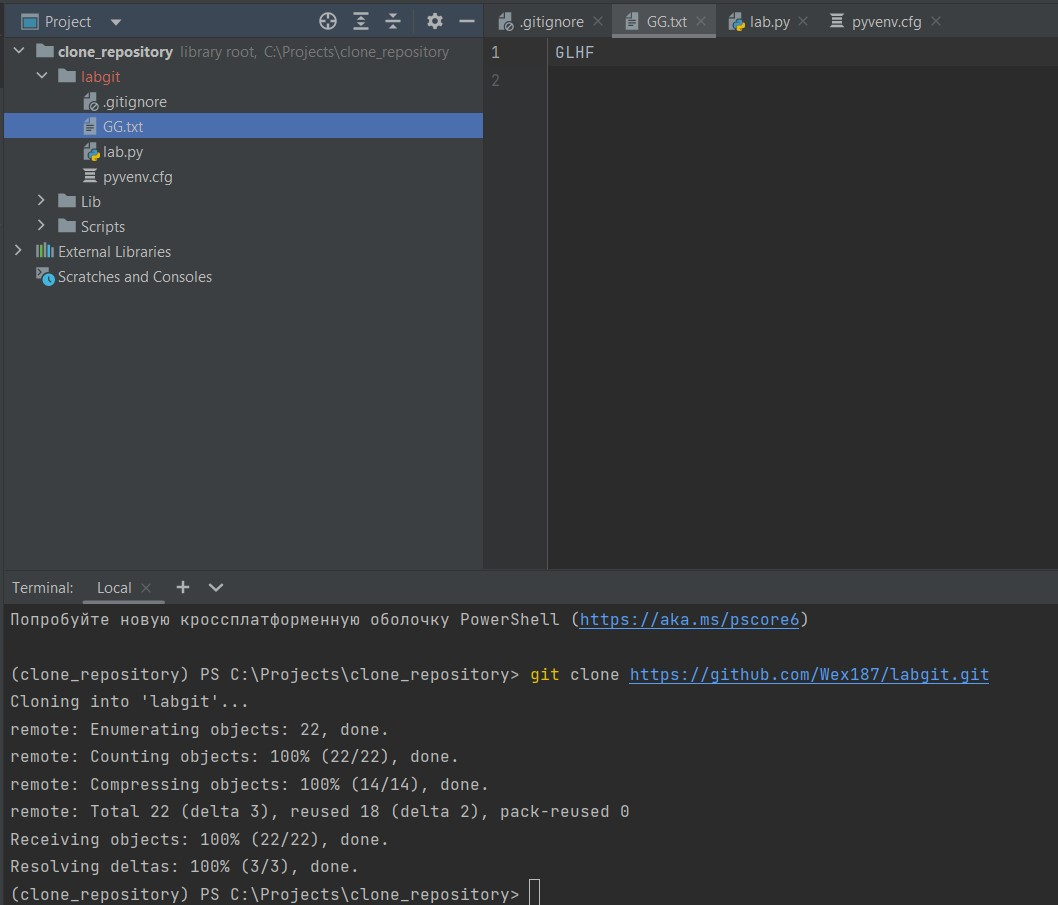


Рис.17 – клонирование удаленного репозитория

Вносим изменения в файл ‘lab.py’ делаем коммит, и пушим изменения

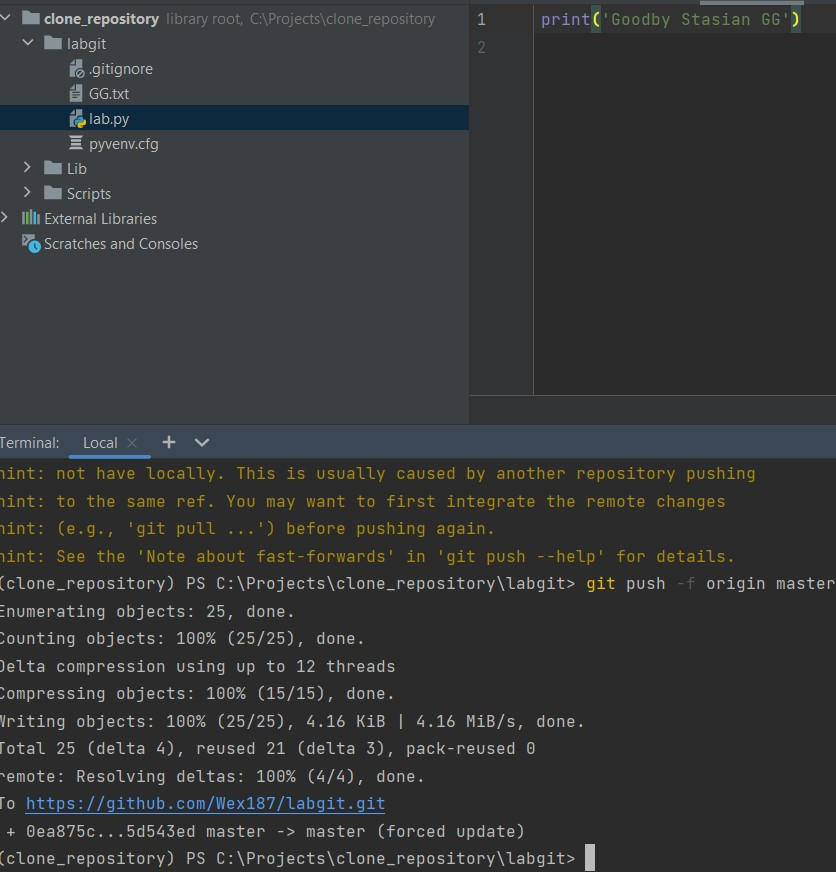
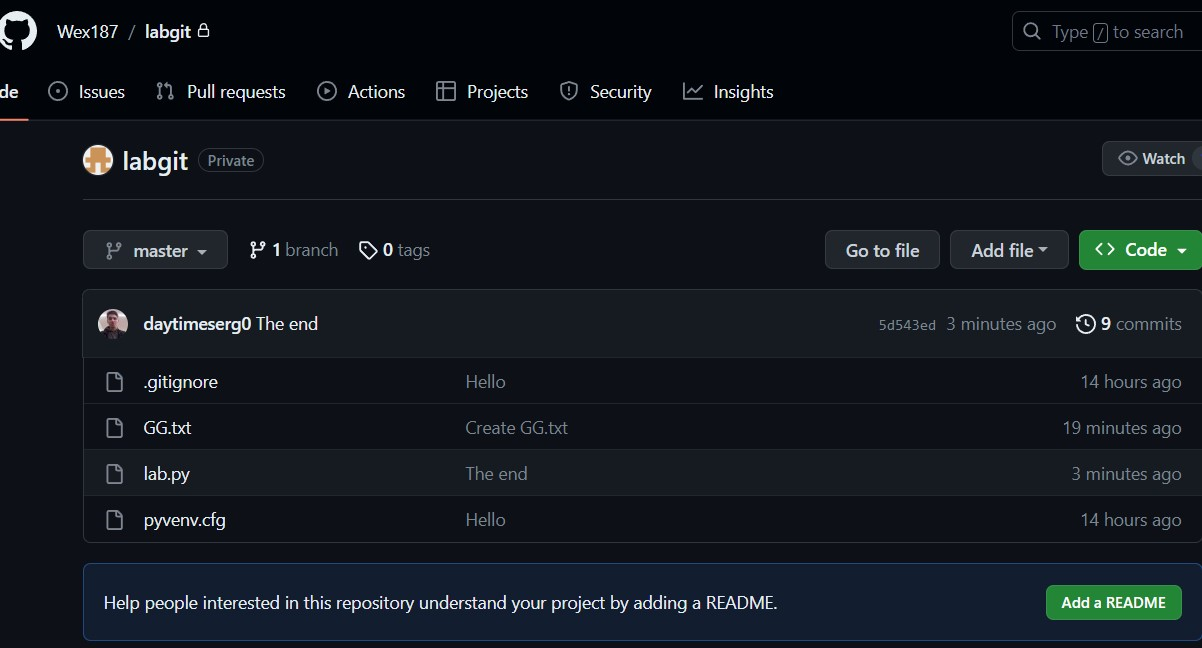


Рис.18 – изменение и пуш в удаленный репозиторий

Заходим на github и видим изменения в репозитории друга



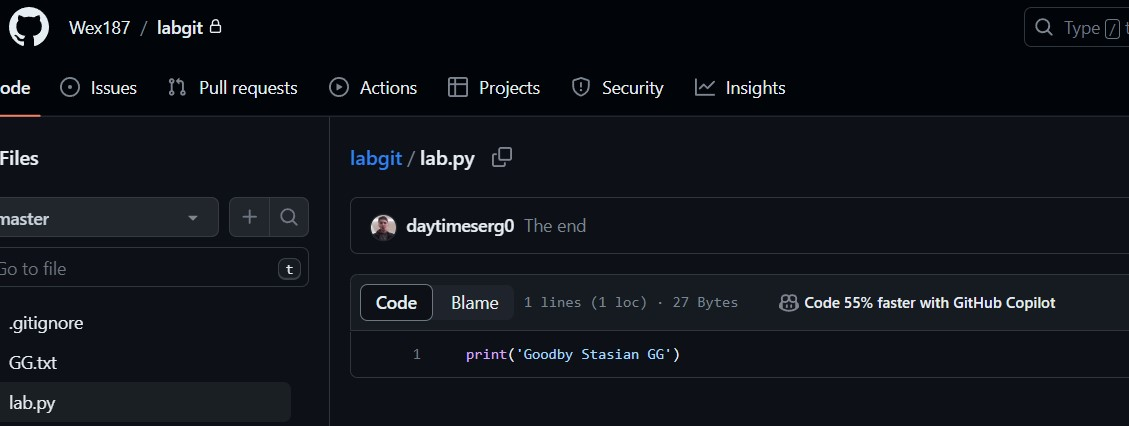


Рис.19-20 – Изменения в удаленном репозитории друга

**Вывод:**

Я научился использовать перечень основных команд git.

**Список источников:**

1. [**https://git-scm.com/book/ru/v2/Приложение-C%3A-Команды-Git-Основные-команды**](https://git-scm.com/book/ru/v2/Приложение-C%3A-Команды-Git-Основные-команды)
2. [**https://www.atlassian.com/ru/git/tutorials/using-branches/merge-conflicts#:~:text=Команду%20git%20reset%20можно%20использовать,заведомо%20удовлетворительное%20состояние%20конфликтующих%20файлов**](https://www.atlassian.com/ru/git/tutorials/using-branches/merge-conflicts#:~:text=Команду%20git%20reset%20можно%20использовать,заведомо%20удовлетворительное%20состояние%20конфликтующих%20файлов)**.**