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 ЛАБОРАТОРНОЙ РАБОТЕ №2**

**дисциплины**

**«Программирование на Python»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Ибрагимов Муса Айнудинович  2 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Руководитель:  Воронкин Роман Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

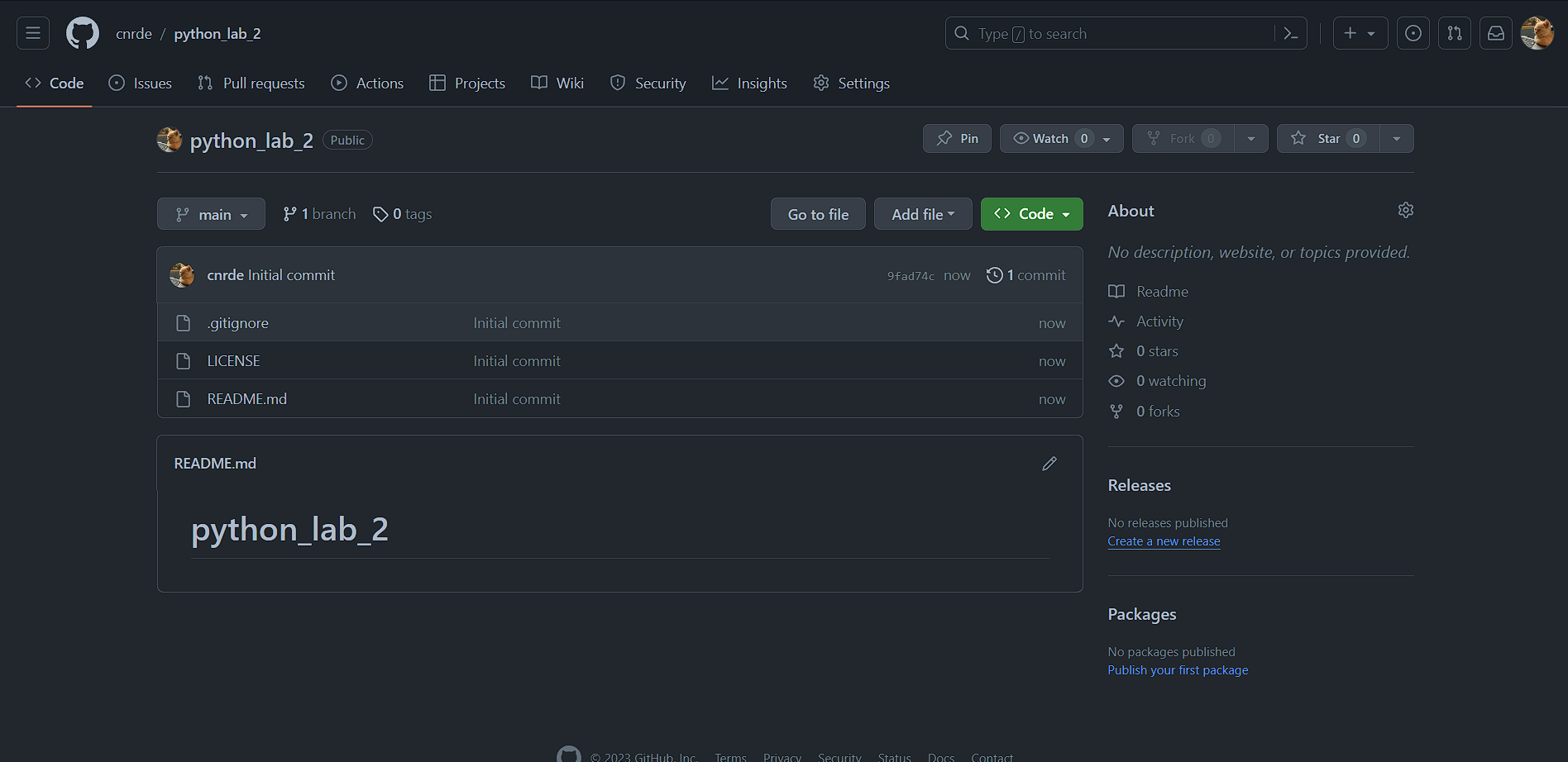
Ставрополь, 2023 г.

Тема: Исследование возможностей Git для работы с локальными репозиториями.

Цель: Исследовать базовые возможности системы контроля версий Git для работы с локальными репозиториями.

**Практическая часть:**

1. Изучил теоретический материал работы.
2. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный вами язык программирования.



1. Проработал примеры лабораторной работы. Клонировал репозиторий (Рис. 1), вписал команду git log (Рис. 2), вписал команду git log -p 2 (Рис. 3), вписал команду git log -stat (Рис. 4), вписал команду git log –pretty=oneline (Рис. 5), вписал команду git log –pretty=format:”%h - %an, %ar : %s” (Рис. 6), вписал команду git log –pretty=format:”%h %s” -graph (Рис. 7), клонировал репозиторий ticgit (Рис. 8), вписал команду git remote (Рис. 9), вписал команду git remote -v (Рис. 10), вписал команду git remote add pb <https://github.com/paulboone/ticgit> (Рис. 11), вписал команду git fetch pb (Рис. 12), вписал команду git remote show origin (Рис. 13), вписал команду git remote rename pb paul (Рис. 14), вписал команду git remote remove paul (Рис. 15), создал аннотированный тег (Рис. 16), вписал команду git show v1.4 (Рис. 17), удалил ранее созданный тег (Рис. 17) и выдохнул с облегчением.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 1. Клонирование репозитория.

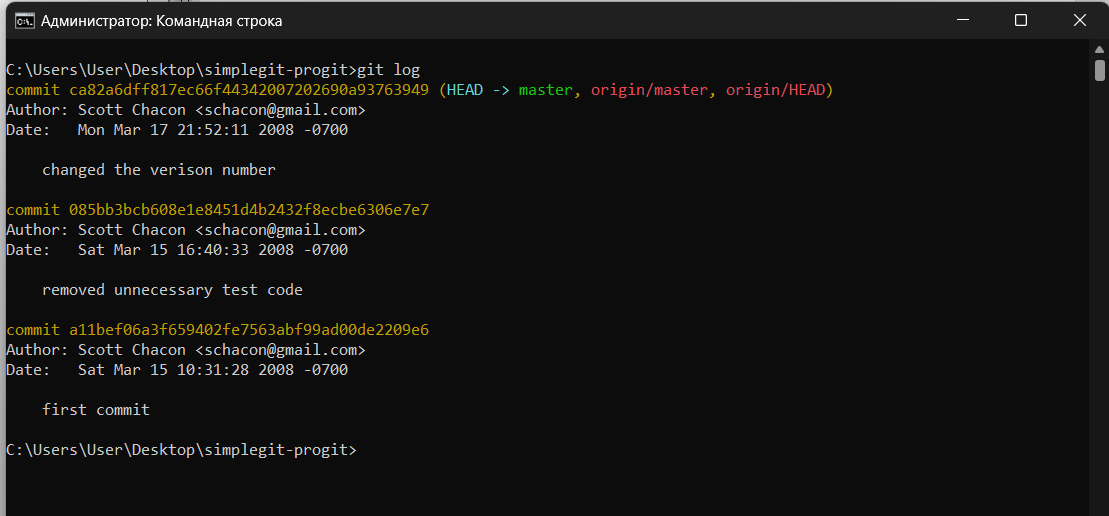


Рисунок 2. Результат работы команды git log.

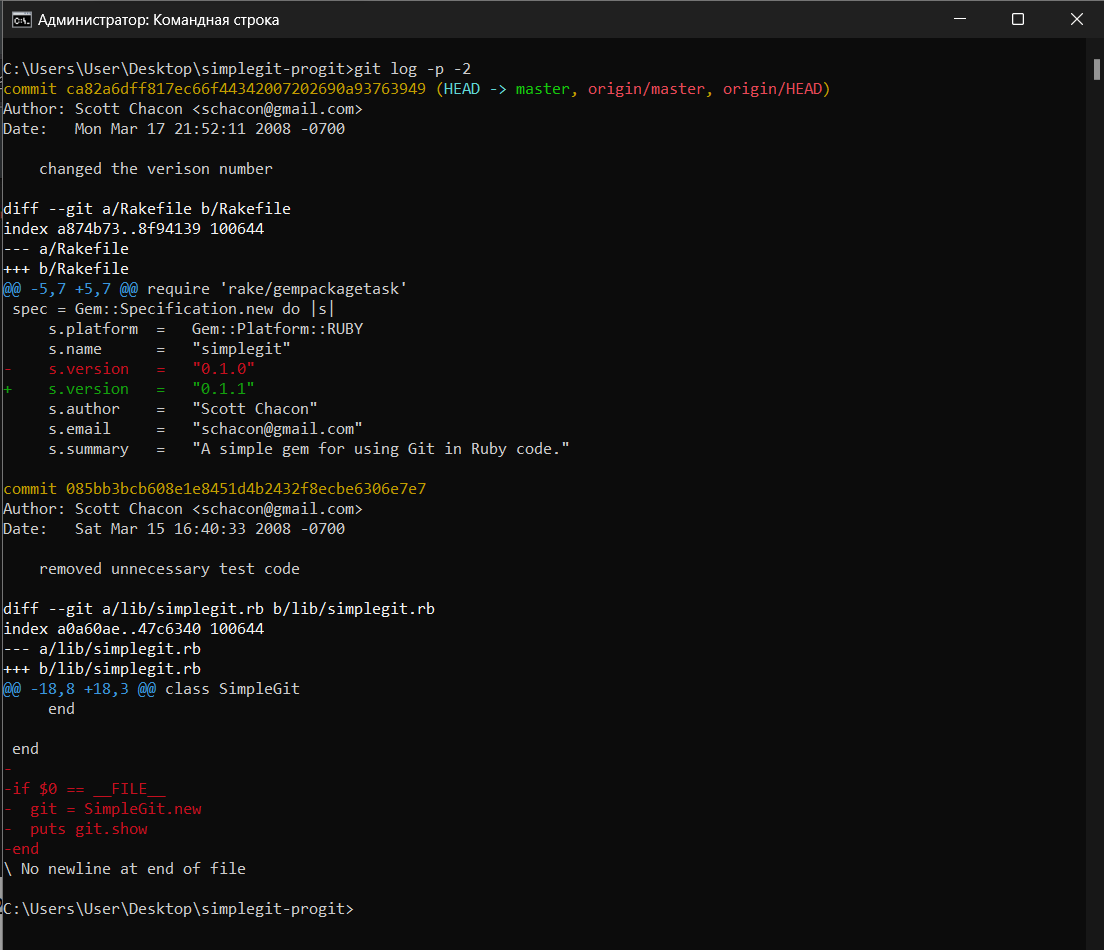


Рисунок 3. Результат работы команды git log -p -2

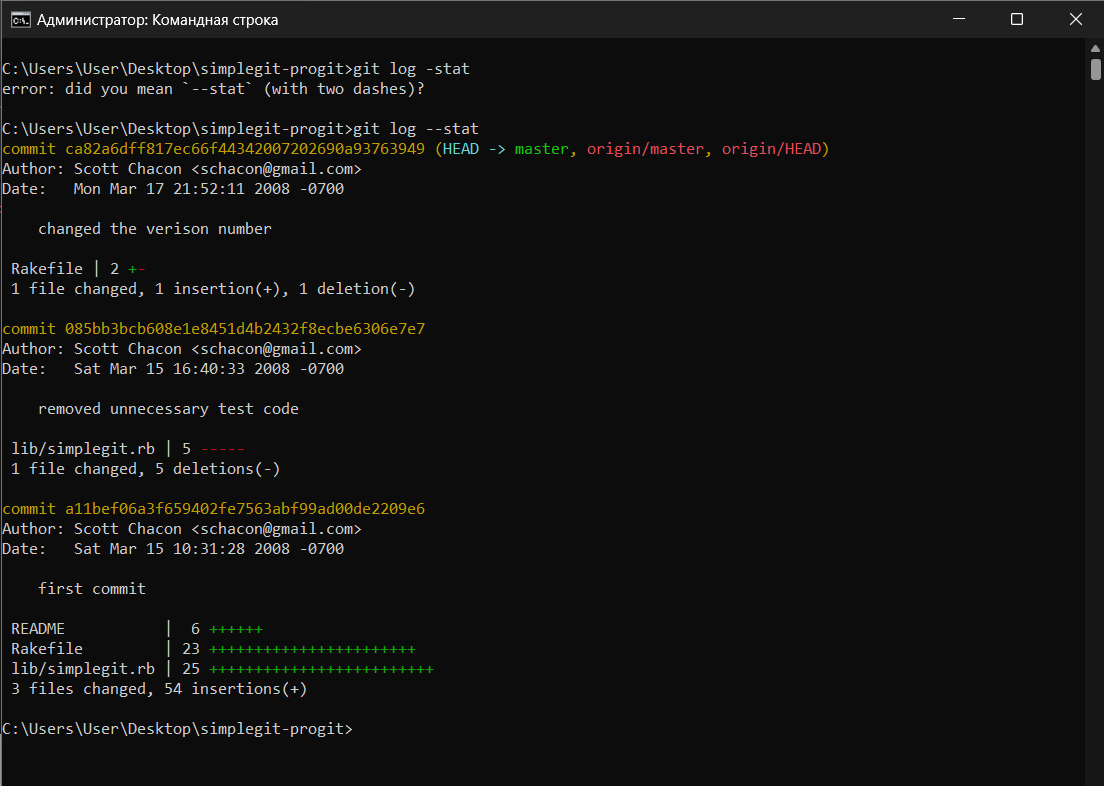


Рисунок 4. Результат работы команды git log –stat.

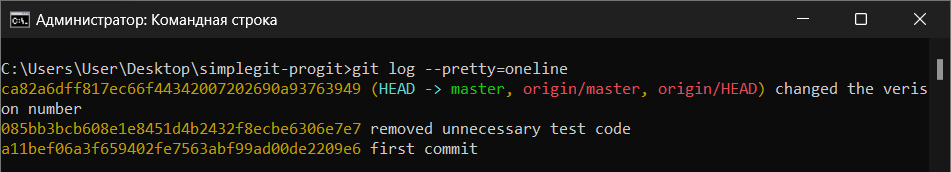


Рисунок 5. Результат работы команды git log –pretty=oneline.

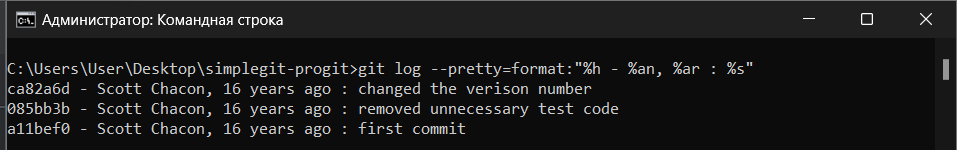


Рисунок 6. Результат работы команды

git log –pretty=format:”%h - %an, %ar : %s”.

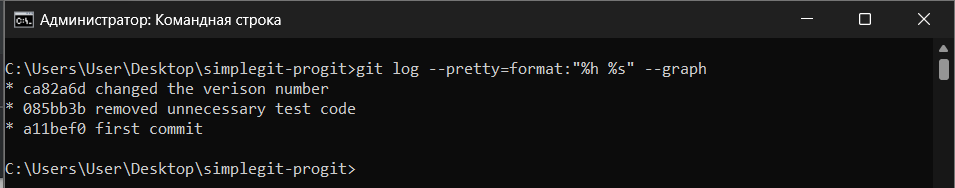


Рисунок 7. Результат работы команды

git log –pretty=format:”%h %s” –graph.

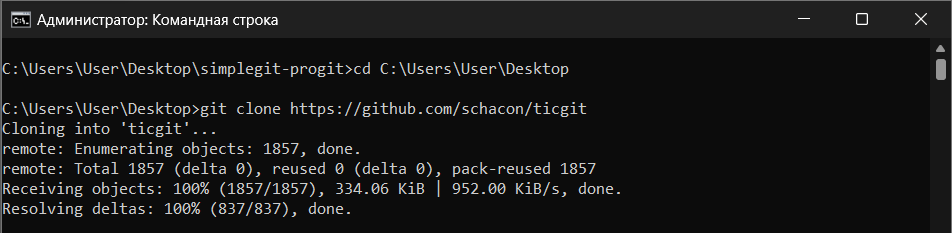


Рисунок 8. Клонирование репозитория ticgit.

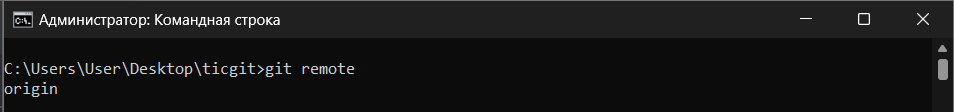


Рисунок 9. Результат работы команды git remote.

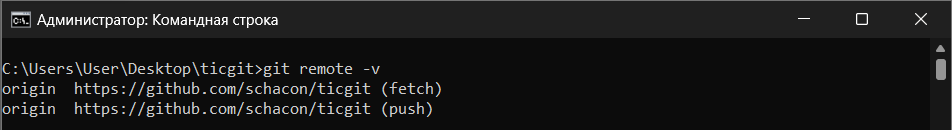


Рисунок 10. Результат работы команды git remote -v.

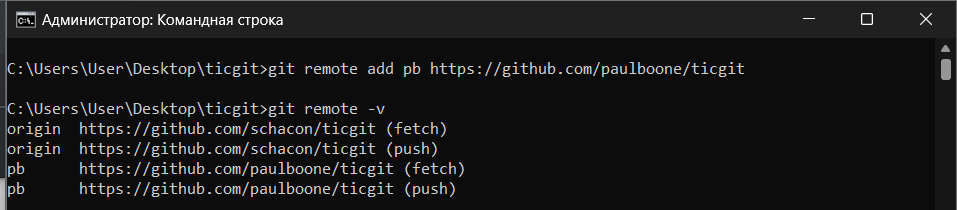


Рисунок 11. Результат работы команды

git remote add pb <https://github.com/paulboone/ticgit>

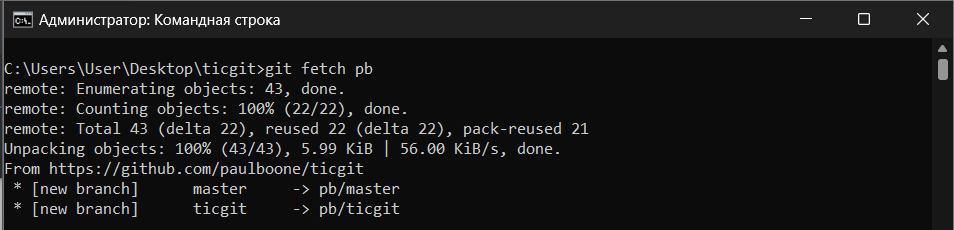


Рисунок 12. Результат работы команды git fetch pb.

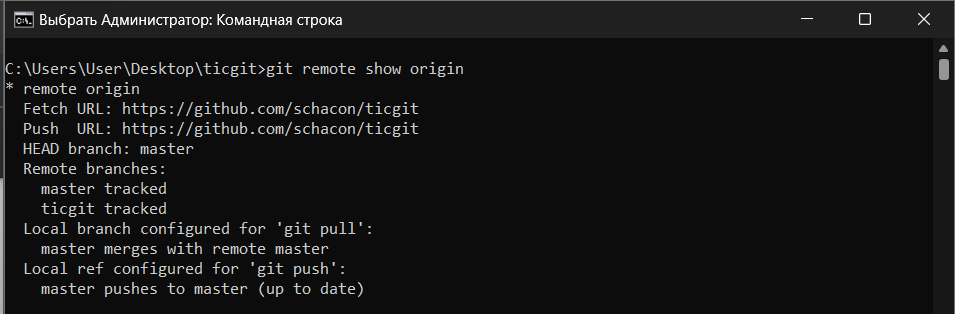


Рисунок 13. Результат работы команды git remote show origin.

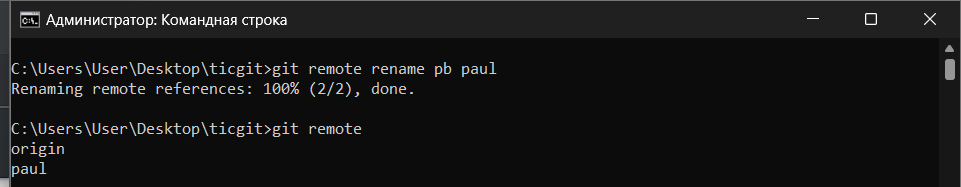


Рисунок 14. Результат работы команды git remote rename pb paul.

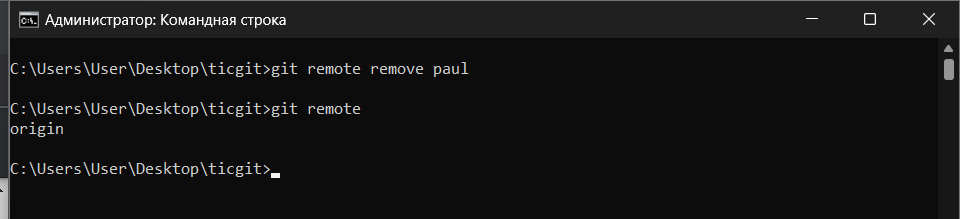


Рисунок 15. Результат работы команды git remote remove paul.

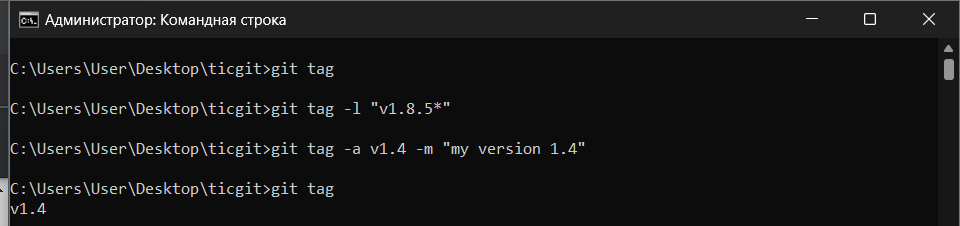


Рисунок 16. Создал аннотированный тег.

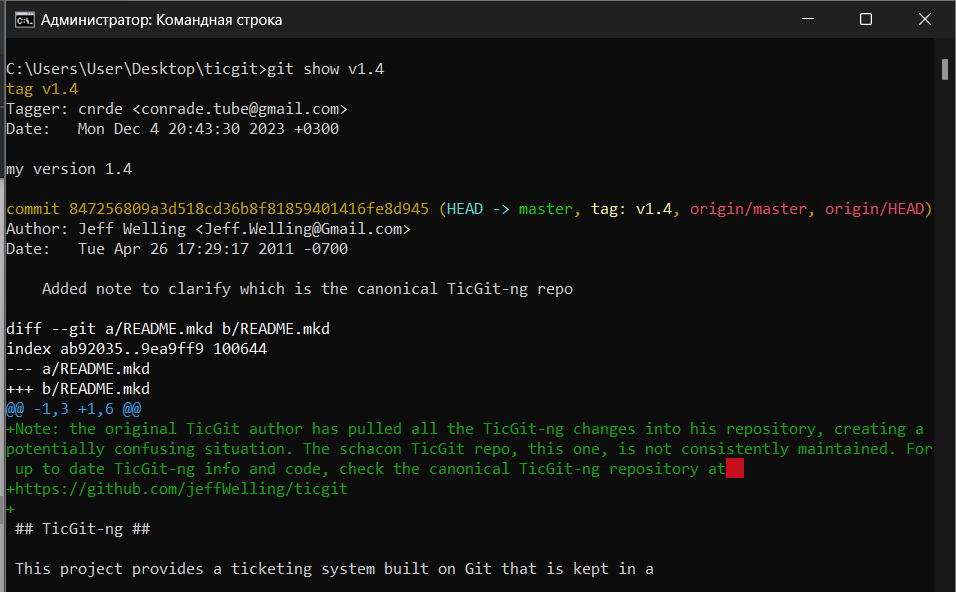
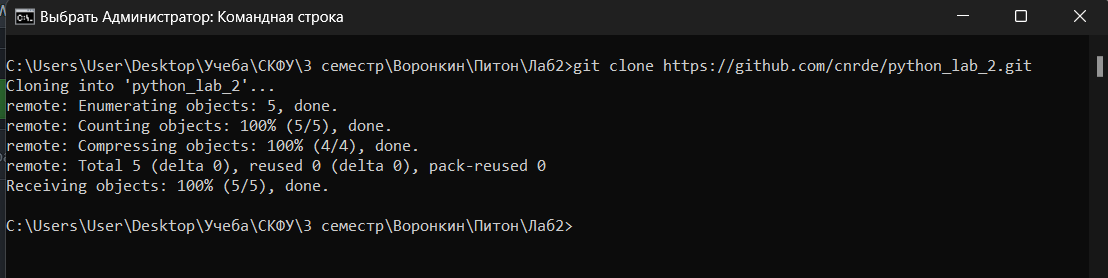


Рисунок 17. Результат работы команды git show v1.4.

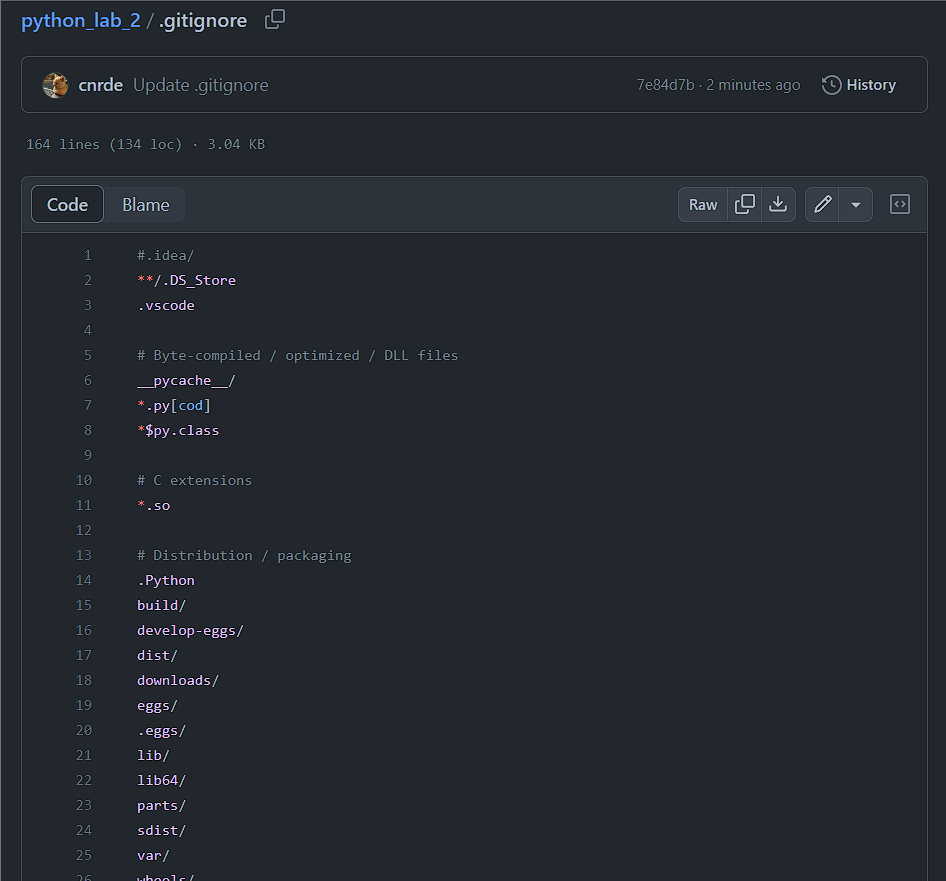


Рисунок 18. Удаление ранее созданного тега.

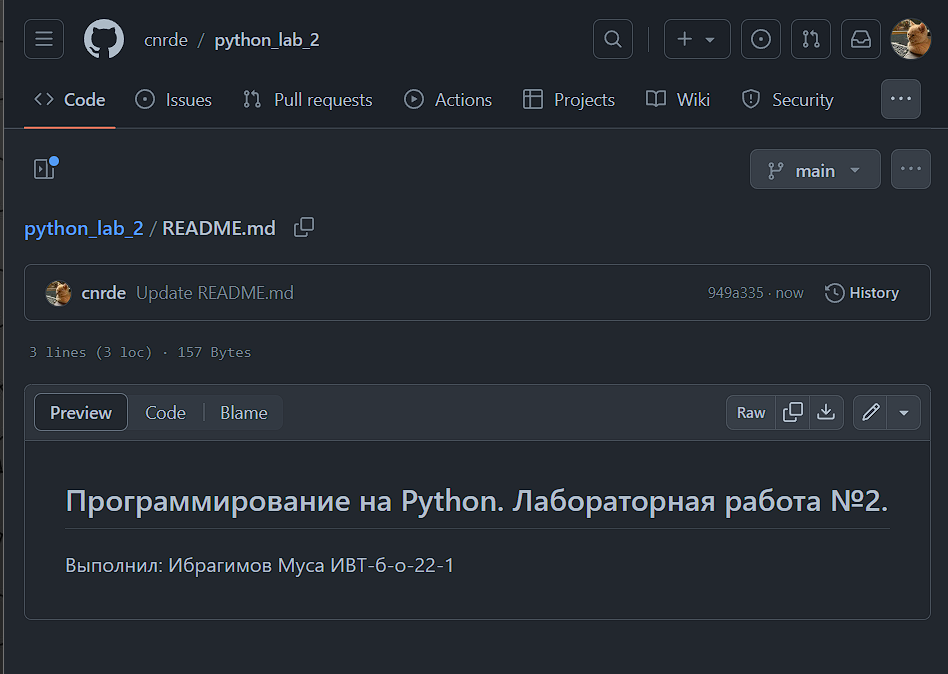
1. Выполнил клонирование созданного репозитория.



1. Дополнил .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.



1. Добавил в файл README.md информацию о дисциплине, группе и ФИО студента, выполняющего лабораторную работы (то есть меня).



1. Написал небольшую программу на Python (Рис. 1) и сделал не менее 7 коммитов (Рис. 2).

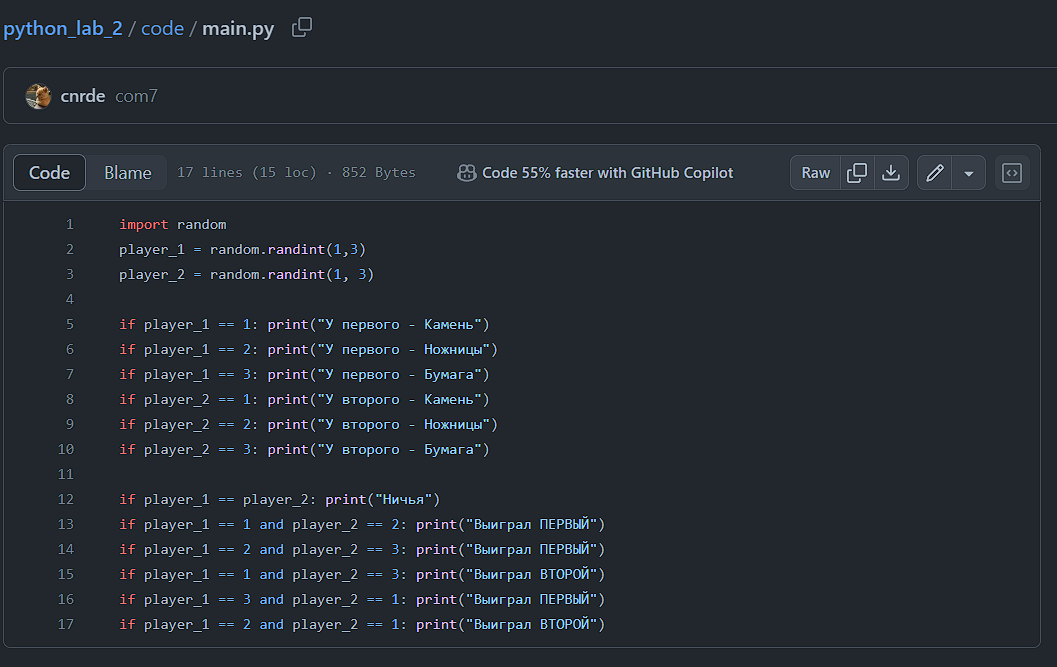


Рисунок 1. Небольшая программа.

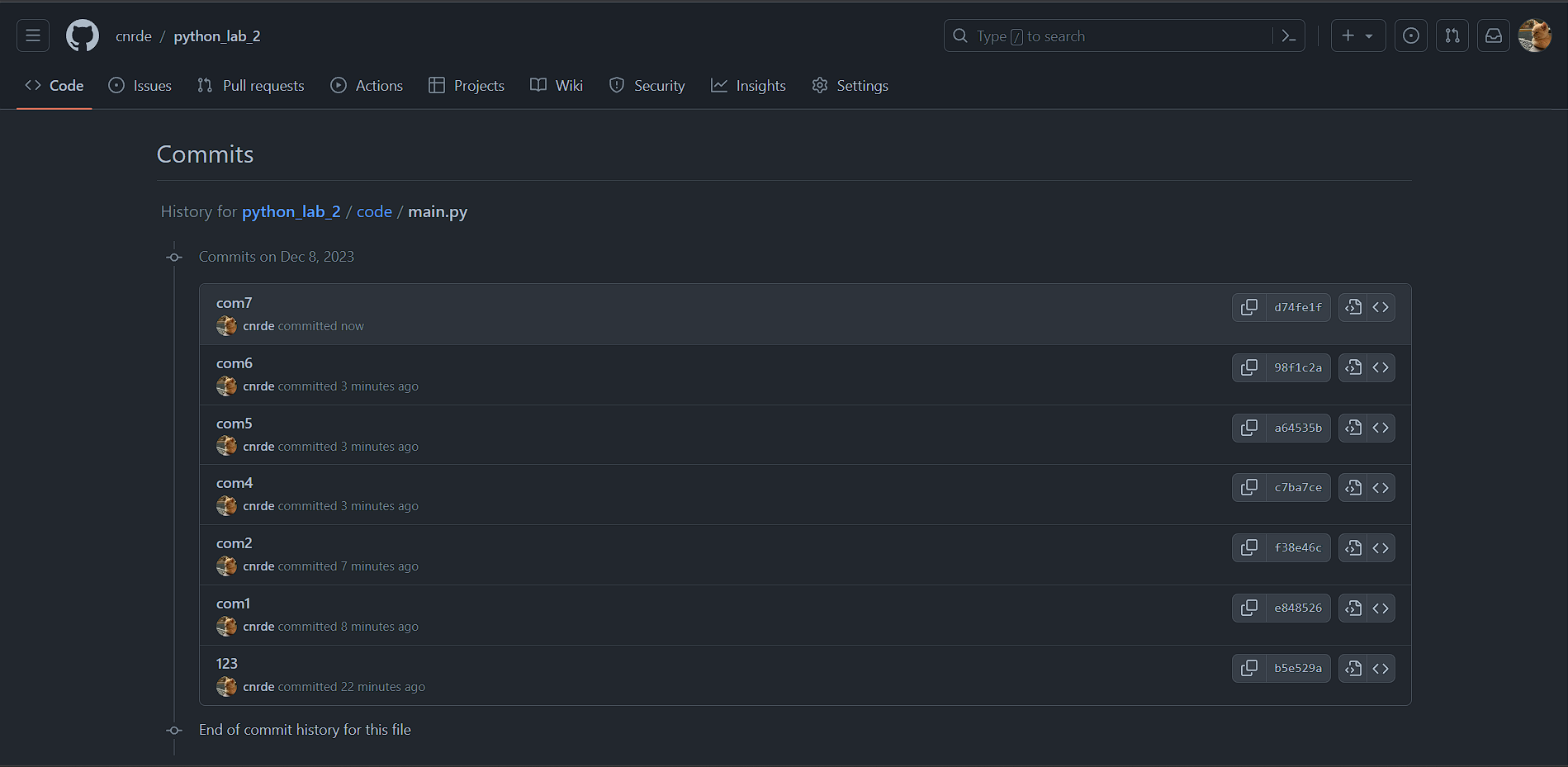
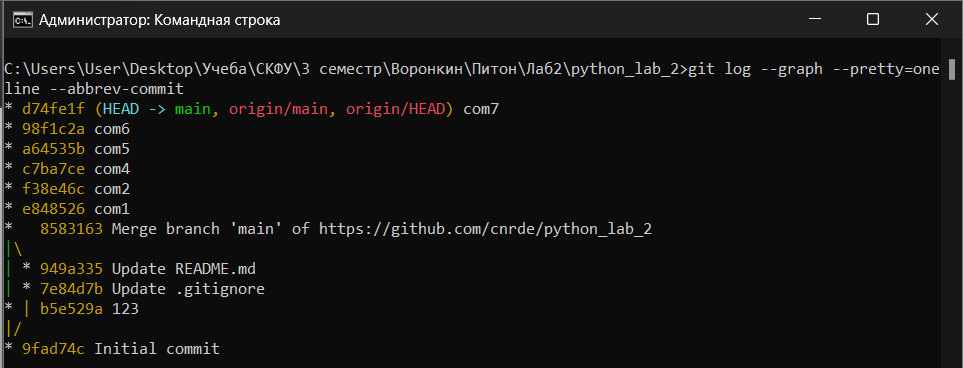


Рисунок 2. История коммитов.

1. Просмотрел историю хранилища командой git log –graph –pretty=oneline –abbrev-commit.



1. Просмотрел содержимое коммитов командой git show HEAD (Рис. 1), git show HEAD~1 (Рис. 2), git show 98f1c2a816 (Рис. 3).

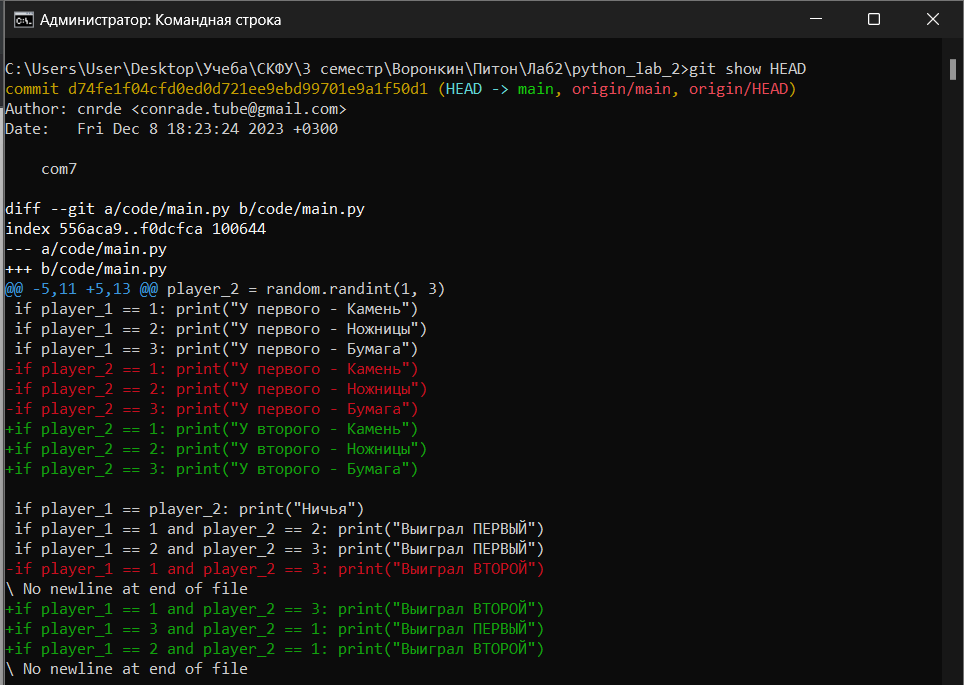


Рисунок 1. Ввёл команду git show HEAD.

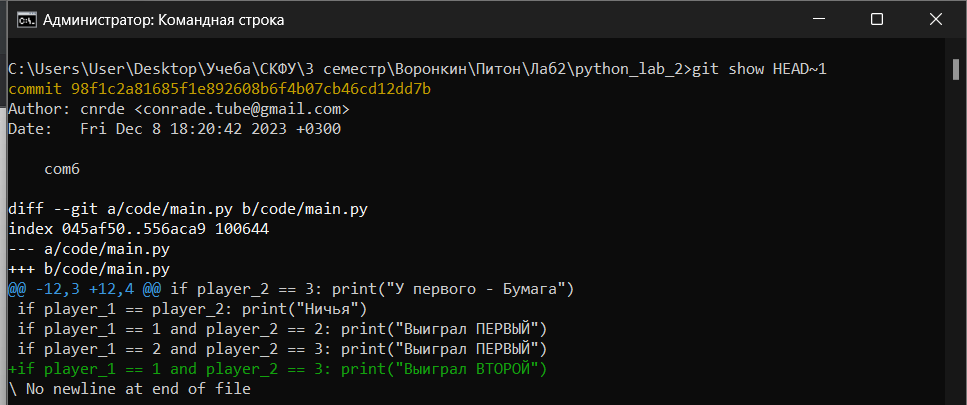


Рисунок 2. Ввёл команду git show HEAD~1.

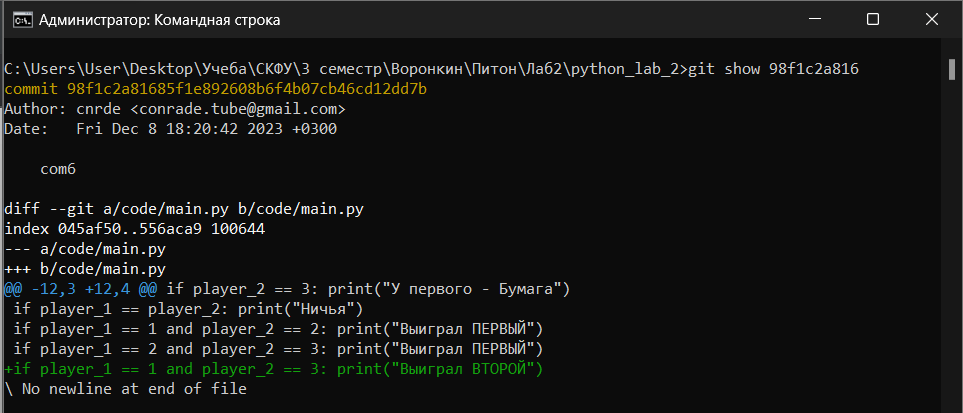


Рисунок 3. Ввел команду git show 98f1c2a816.

1. Удалил код из файла main.py, а затем удалил все несохраненные изменения (Рис. 1). Восстановил код командой git checkout –main.py (Рис. 2).

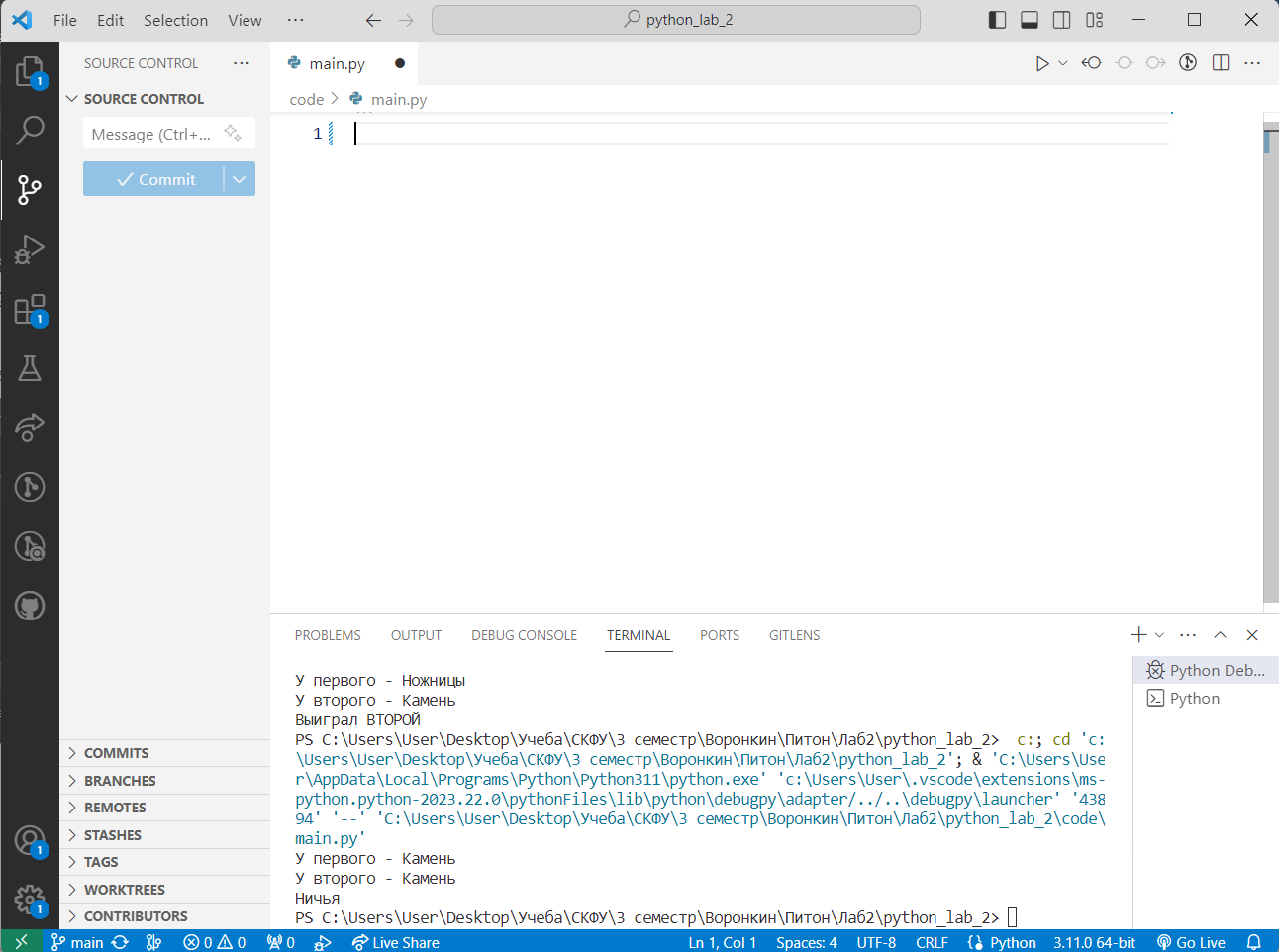


Рисунок 1. Удалил код из файла main.py.

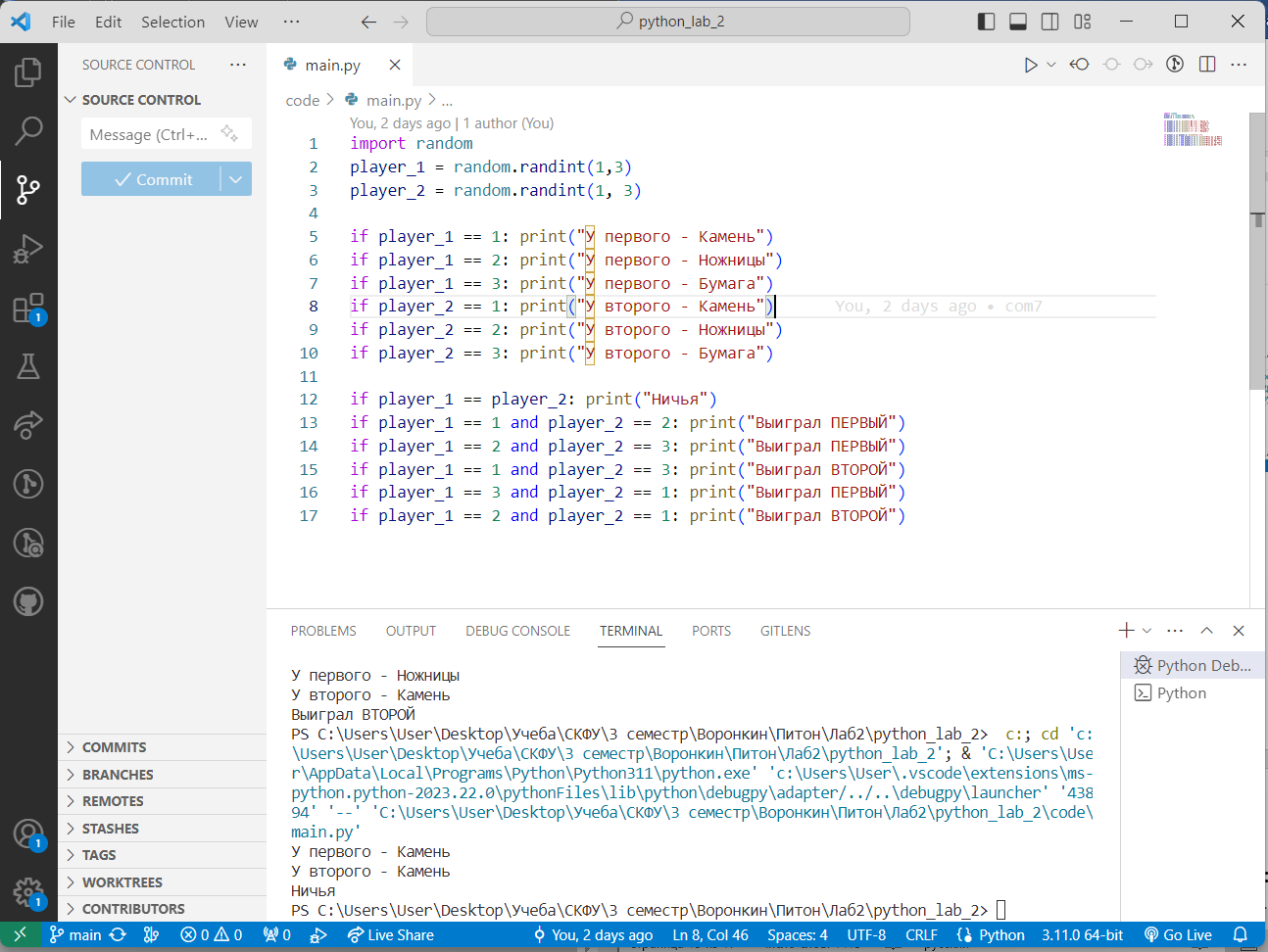


Рисунок 2. Восстановился код после команды git checkout –main.py.

**Ответы на контрольные вопросы:**

1. Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

После того, как вы создали несколько коммитов или же клонировали репозиторий с уже существующей историей коммитов, вероятно Вам понадобится возможность посмотреть, что было сделано — историю коммитов. Одним из основных и наиболее мощных инструментов для этого является команда git log. Команда git log имеет очень большое количество опций для поиска коммитов по разным критериям. Рассмотрим наиболее популярные из них. Одним из самых полезных аргументов является -p или --patch, который показывает разницу (выводит патч), внесенную в каждый коммит. Если вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию –stat. Следующей действительно полезной опцией является --pretty. Эта опция меняет формат вывода. Существует несколько встроенных вариантов отображения. Опция oneline выводит каждый коммит в одну строку, что может быть очень удобным если вы просматриваете большое количество коммитов. К тому же, опции short, full и fuller делают вывод приблизительно в том же формате, но с меньшим или большим количеством информации соответственно. Наиболее интересной опцией является format, которая позволяет указать формат для вывода информации.

2. Как ограничить вывод при просмотре истории коммитов?

В дополнение к опциям форматирования вывода, команда git log принимает несколько опций для ограничения вывода — опций, с помощью которых можно увидеть определенное подмножество коммитов. Одна из таких опций — это опция -2, которая показывает только последние два коммита. В действительности вы можете использовать -<n>, где n — это любое натуральное число и представляет собой n последних коммитов. На практике вы не будете часто использовать эту опцию, потому что Git по умолчанию использует постраничный вывод, и вы будете видеть только одну страницу за раз. Опции для ограничения вывода по времени, такие как --since и --until, являются очень удобными. Опция --author дает возможность фильтровать по автору коммита, а опция --grep искать по ключевым словам в сообщении коммита. Следующим действительно полезным фильтром является опция -S, которая принимает аргумент в виде строки и показывает только те коммиты, в которых изменение в коде повлекло за собой добавление или удаление этой строки. Последней полезной опцией, которую принимает команда git log как фильтр, является путь. Если вы укажете каталог или имя файла, вы ограничите вывод только теми коммитами, в которых были изменения этих файлов. Эта опция всегда указывается последней после двойного тире ( -- ), чтобы отделить пути от опций

3. Как внести изменения в уже сделанный коммит?

Отмена может потребоваться, если вы сделали коммит слишком рано, например, забыв добавить какие-то файлы или комментарий к коммиту. Если вы хотите переделать коммит — внесите необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав параметр –amend.

4. Как отменить индексацию файла в Git?

Использовать git reset HEAD <file>... для исключения из индекса.

5. Как отменить изменения в файле?

Использовать git checkout -- <file> для возвращения к версии из последнего коммита.

6. Что такое удаленный репозиторий Git?

Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети.

7. Как выполнить просмотр удаленных репозиториев данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду git remote. Она выведет названия доступных удалённых репозиториев. Если вы клонировали репозиторий, то увидите как минимум origin — имя по умолчанию, которое Git даёт серверу, с которого производилось клонирование.

8. Как добавить удаленный репозиторий для данного локального репозитория?

Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (shortname), просто выполните команду git remote add <shortname> <url>.

9. Как выполнить отправку/получение изменений с удаленного репозитория?

Для получения данных из удалённых проектов, следует выполнить git fetch [remote-name]. Когда вы хотите поделиться своими наработками, вам необходимо отправить их в удалённый репозиторий. Команда для этого действия простая: git push <remote-name> <branch-name>.

10. Как выполнить просмотр удаленного репозитория?

Если хотите получить побольше информации об одном из удалённых репозиториев, вы можете использовать команду git remote show <remote>. Она выдаёт URL удалённого репозитория, а такжет информацию об отслеживаемых ветках.

11. Каково назначение тэгов Git?

Как и большинство СКВ, Git имеет возможность помечать определённые моменты в истории как важные. Как правило, эта функциональность используется для отметки моментов выпуска версий (v1.0, и т. п.). Такие пометки в Git называются тегами.

12. Как осуществляется работа с тэгами Git?

Просмотреть список имеющихся тегов в Git можно очень просто. Достаточно набрать команду git tag (параметры -l и --list опциональны). Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать - a при выполнении команды tag. По умолчанию, команда git push не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду git push origin <tagname>. Для удаления тега в локальном репозитории достаточно выполнить команду git tag –d <tagname>. Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать git checkout для тега. Однако, это переведёт репозиторий в состояние «detached HEAD», которое имеет ряд неприятных побочных эффектов.

13. Самостоятельно изучите назначение флага --prune в командах git fetch и git push. Каково назначение этого флага?

Исходя из описания, предоставленного git help fetch: --prune используется для удаления ссылок удаленного отслеживания, которые больше не существуют в удаленном репозитории, а из описания, предоставленного git help push: --prune используется для удаления ветвей на удаленном репозитории, для которых нет аналога в локальном репозитории.

**Вывод:** в результате выполнения работы исследовал были исследованы возможности Git для работы с локальными репозиториями.