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 №1.2
дисциплины «Программирование на Python»

Выполнил:
Омонкулов Исомиддин
Волижон угли
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 практики:
Воронкин Р.А., доцент кафедры инфокоммуникаций

_____
(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.

Порядок выполнения работы:

1. Создал новый репозиторий:

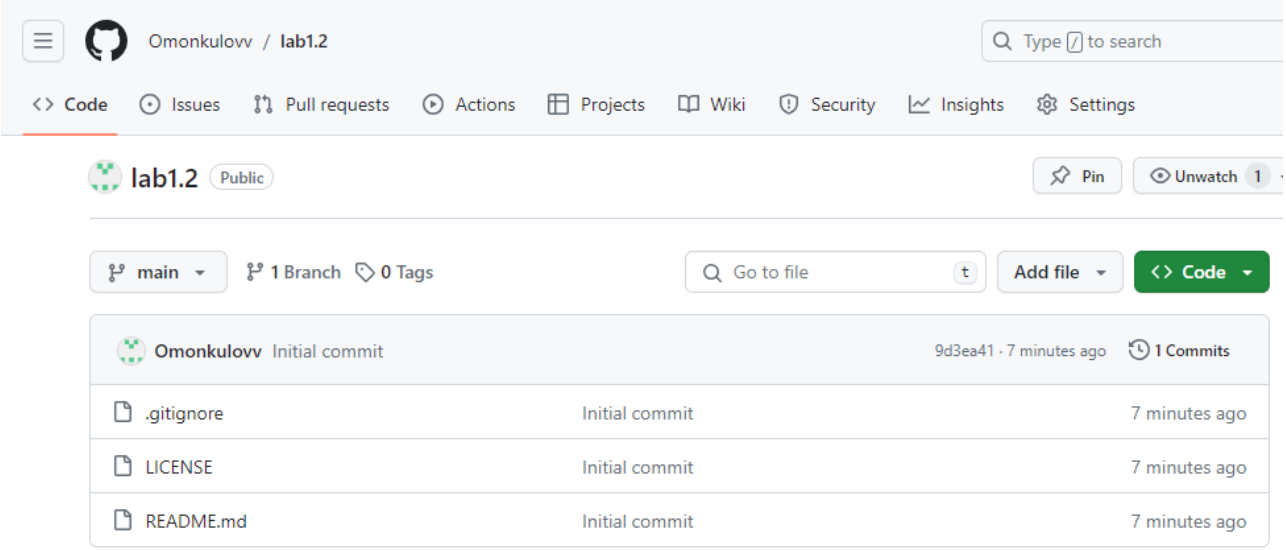

Рисунок 1. Новый репозиторий Lab1.2

2. Проработал примеры лабораторной работы:

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc>git clone https://github.com/schacon/simplegit-progit
Cloning into 'simplegit-progit'...
remote: Enumerating objects: 13, done.
remote: Total 13 (delta 0), reused 0 (delta 0), pack-reused 13
Receiving objects: 100% (13/13), done.
Resolving deltas: 100% (3/3), done.

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc>
```

Рисунок 2. Клонирование репозитория

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc>git log
commit 9d3ea41e8b2c88376af40dd51ebbf1774d15c6a7 (HEAD -> main, origin/main, origin/HEAD)
Author: Omonkulovv <159341332+Omonkulovv@users.noreply.github.com>
Date:   Thu Feb 8 00:18:55 2024 +0300

    Initial commit
```

Рисунок 3. Результат работы команды git log

```
Author: Omonkulovv <159341332+Omonkulovv@users.noreply.github.com>
Date:   Thu Feb 8 00:18:55 2024 +0300

    Initial commit

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..68bc17f
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1,160 @@
+# Byte-compiled / optimized / DLL files
+__pycache__/
+*.py[cod]
+*$py.class
+
+# C extensions
+*.so
+
+# Distribution / packaging
+.Python
+build/
+develop-eggs/
+dist/
+downloads/
+eggs/
+.eggs/
+lib/
:
```

Рисунок 4. Результат работы команды git log -p -2

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc>git log --stat
commit 9d3ea41e8b2c88376af40dd51ebbf1774d15c6a7 (HEAD -> main, origin/main, origin/HEAD)
Author: Omonkulovv <159341332+Omonkulovv@users.noreply.github.com>
Date:   Thu Feb 8 00:18:55 2024 +0300

    Initial commit

 .gitignore | 160 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 LICENSE    |  21 ++++++++
 README.md  |   1 +
 3 files changed, 182 insertions(+)

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc>
```

Рисунок 5. Результат работы команды git log --stat

Рисунок 6. Результат работы команды git log --pretty=online

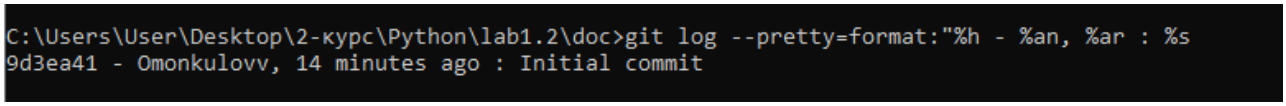

Рисунок 7. Результат команды git log --pretty=format:"%h - %an, %ar : %s"

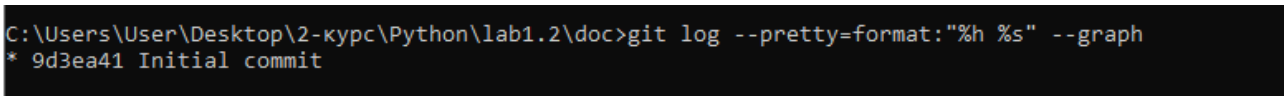

Рисунок 8. Результат работы команды git log --pretty=format:"%h %s" --graph

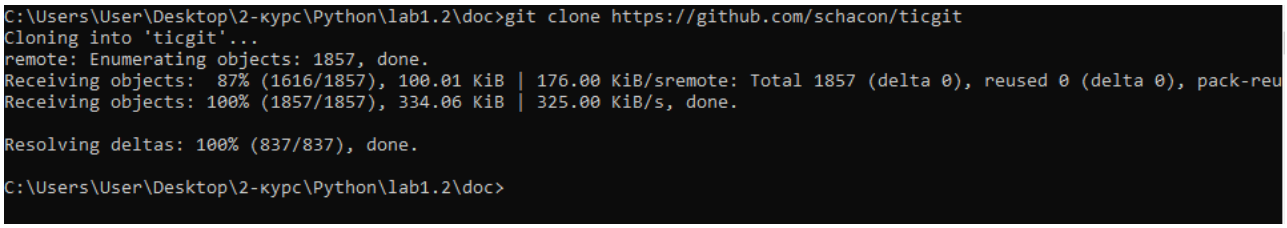

Рисунок 9. Клонирование репозитория ticgit

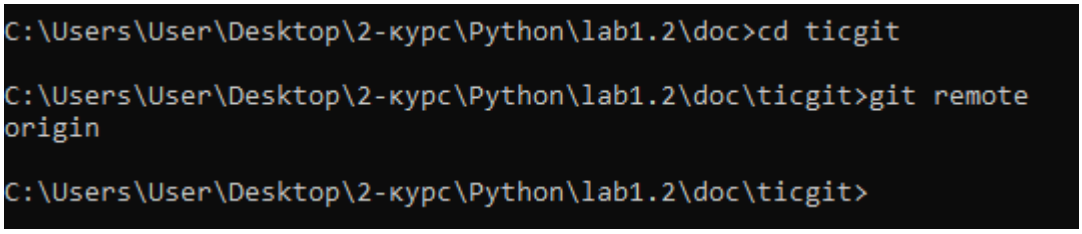

Рисунок 10. Результат работы команды git remote

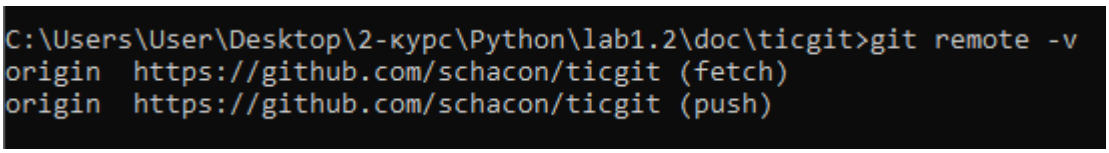

Рисунок 11. Результат работы команды git remote -v

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git remote add pb https:github.com/paulboone/tic

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git remote -v
origin  https://github.com/schacon/ticgit (fetch)
origin  https://github.com/schacon/ticgit (push)
pb      https:github.com/paulboone/tic (fetch)
pb      https:github.com/paulboone/tic (push)
```

Рисунок 12. Результат работы команды git remote add pb

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git fetch pb
ssh: Could not resolve hostname https: Name or service not known
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

Рисунок 13. Результат работы команды git fetch pb

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git remote show origin
* remote origin
  Fetch URL: https://github.com/schacon/ticgit
  Push  URL: https://github.com/schacon/ticgit
  HEAD branch: master
  Remote branches:
    master tracked
    ticgit tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)
```

Рисунок 14. Результат работы команды git remote show origin

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git remote rename pb paul

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git remote
origin
paul

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>
```

Рисунок 15. Результат работы команды git remote rename pb paul

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git remote remove paul

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git remote
origin
```

Рисунок 16. Результат работы команды git remote remove paul

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git tag

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git tag -l "v1.8.5*"

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git tag -a v1.4 -m "my version 1.4"

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git tag
v1.4
```

Рисунок 17. Создание аннотированного тега

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git show v1.4
tag v1.4
Tagger: Isomiddin <isomiddinomonqulov2004@gmail.com>
Date:   Thu Feb 8 00:41:29 2024 +0300

my version 1.4

commit 847256809a3d518cd36b8f81859401416fe8d945 (HEAD -> master, tag: v1.4, origin/master, origin/HEAD)
Author: Jeff Welling <Jeff.Welling@Gmail.com>
Date:   Tue Apr 26 17:29:17 2011 -0700

    Added note to clarify which is the canonical TicGit-ng repo

diff --git a/README.mkd b/README.mkd
index ab92035..9ea9ff9 100644
--- a/README.mkd
+++ b/README.mkd
@@ -1,3 +1,6 @@
+Note: the original TicGit author has pulled all the TicGit-ng changes into his repository, creating a potentially confu
sing situation. The schacon TicGit repo, this one, is not consistently maintained. For up to date TicGit-ng info and cod
e, check the canonical TicGit-ng repository at
+https://github.com/jeffWelling/ticgit
+
 ## TicGit-ng ##

 This project provides a ticketing system built on Git that is kept in a

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>
```

Рисунок 18. Результат работы команды git show v1.4

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>git tag -d v1.4
Deleted tag 'v1.4' (was a51eb1c)

C:\Users\User\Desktop\2-курс\Python\lab1.2\doc\ticgit>
```

Рисунок 19. Удаление ранее созданного тега

3. Клонировал ранее созданные репозиторий:

```
C:\Users\User\Desktop\2-курс\Python\lab1.2\doc>git clone https://github.com/Omonkulovv/lab1.2.git
Cloning into 'lab1.2'...
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 20. Клонирование репозитория Lab1.2

4. Дополнил .gitignore:

```
160    #.idea/
161    .vscode
162
```

Рисунок 21. Добавленная строка в .gitignore

5. Добавил в файл README.md информацию о группе и ФИО:

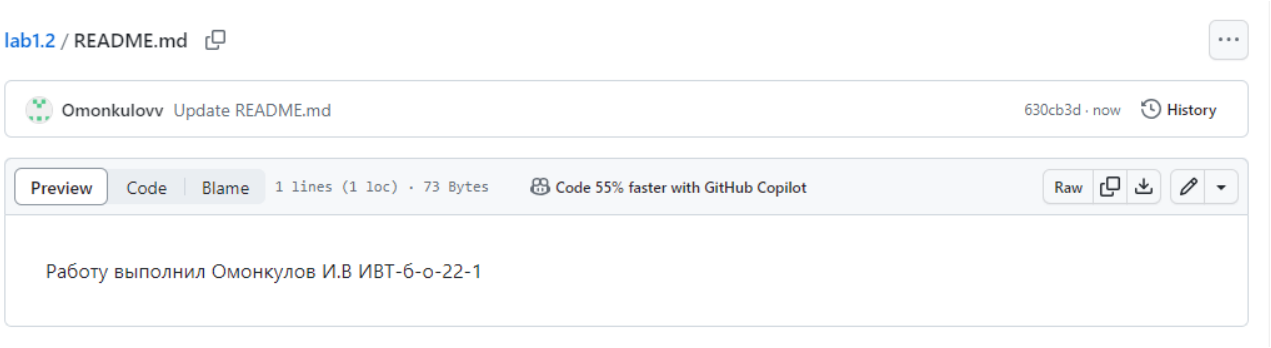

Рисунок 22. Добавление информации в файл README.md

6. Написал небольшую программу на языке Python:

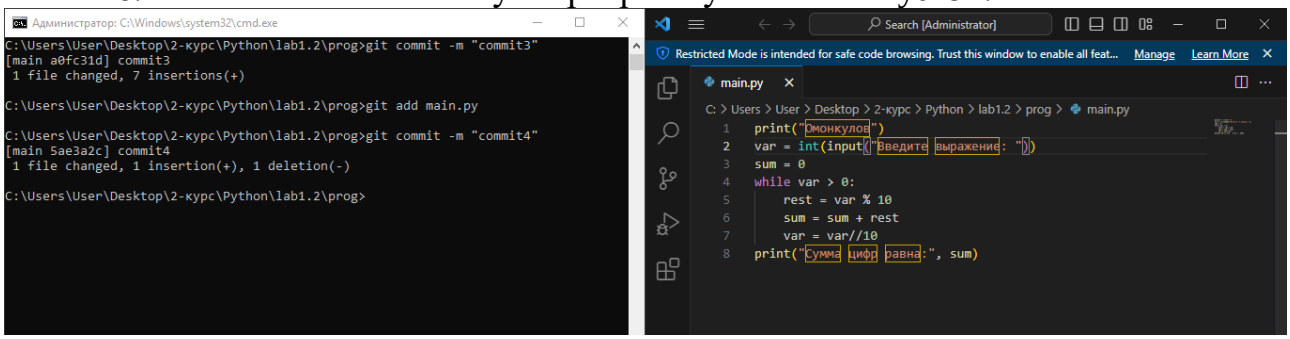

Рисунок 23. Первые 4 коммита и код программы

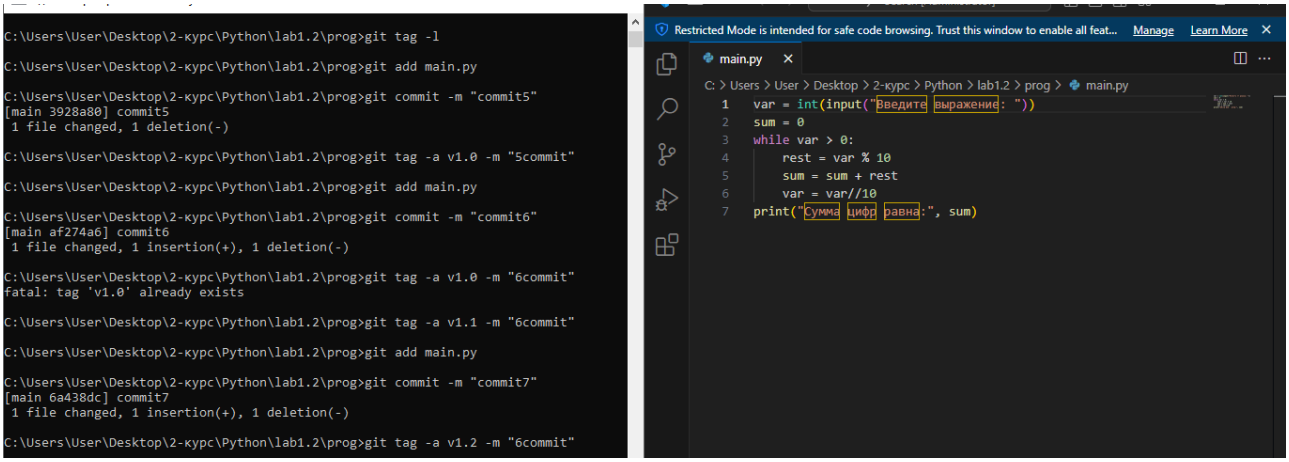

Рисунок 24. Последние 3 коммита с тегами

7. Просмотрел историю хранилища:

```
PS C:\Users\dalam\Lab1.2> git log --graph --pretty=oneline --abbrev-commit
* d1dc258 (HEAD -> main, tag: v1.2, origin/main, origin/HEAD) commit7
* 6dc03e5 (tag: v1.1) commit6
* 03554b5 (tag: v1.0) commit5
* df97c62 commit4
* a3265ca commit3
* 934fec7 commit2
* aea3a9b commit1
* 5e5fc5a Update .gitignore
* c1972be Initial commit
PS C:\Users\dalam\Lab1.2>
```

Рисунок 25. Результат работы команды git log --graph --pretty=oneline --abbrev-commit

8. Посмотрел содержимое коммитов:

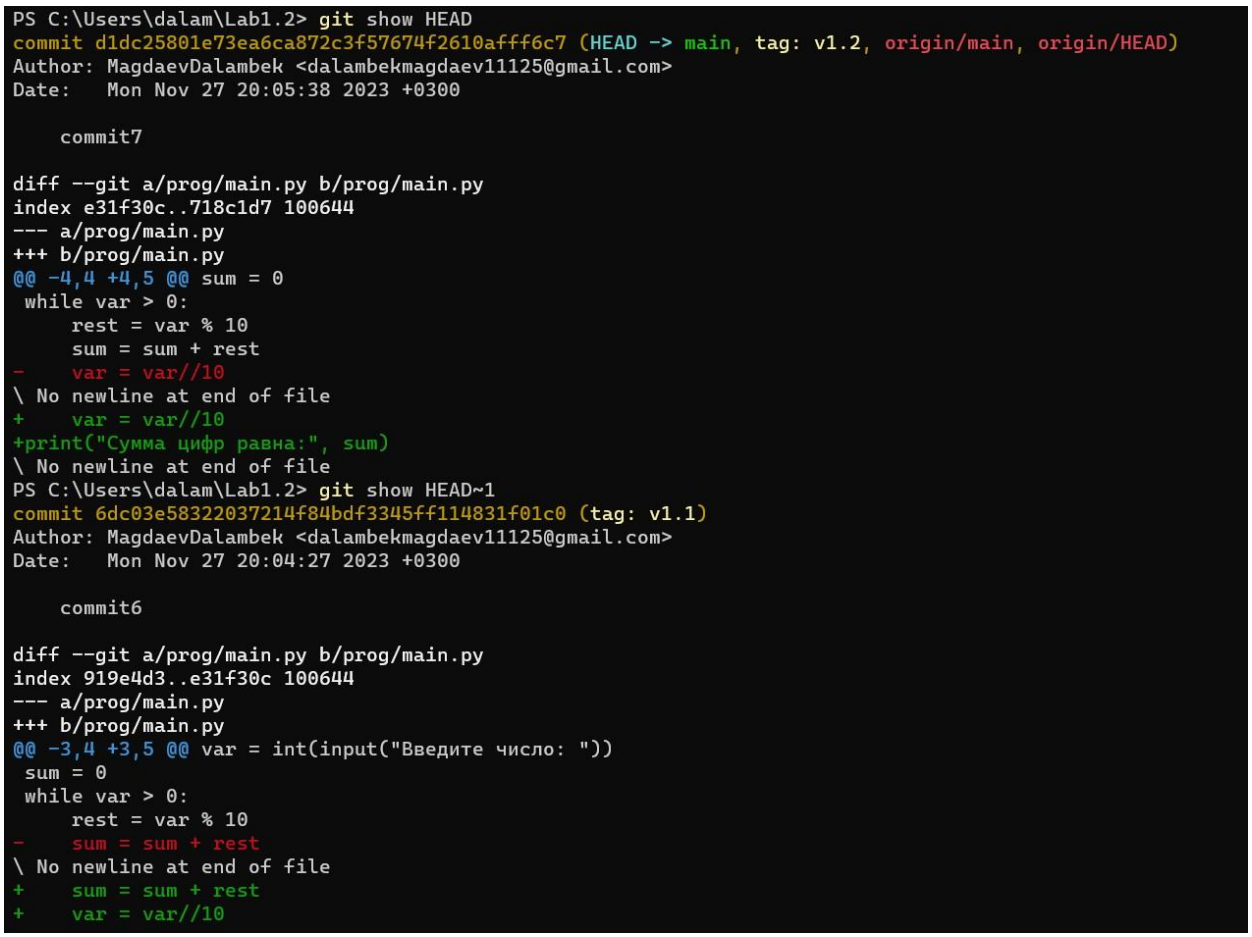

Рисунок 26. Результат работы команд git show HEAD и git show HEAD~1

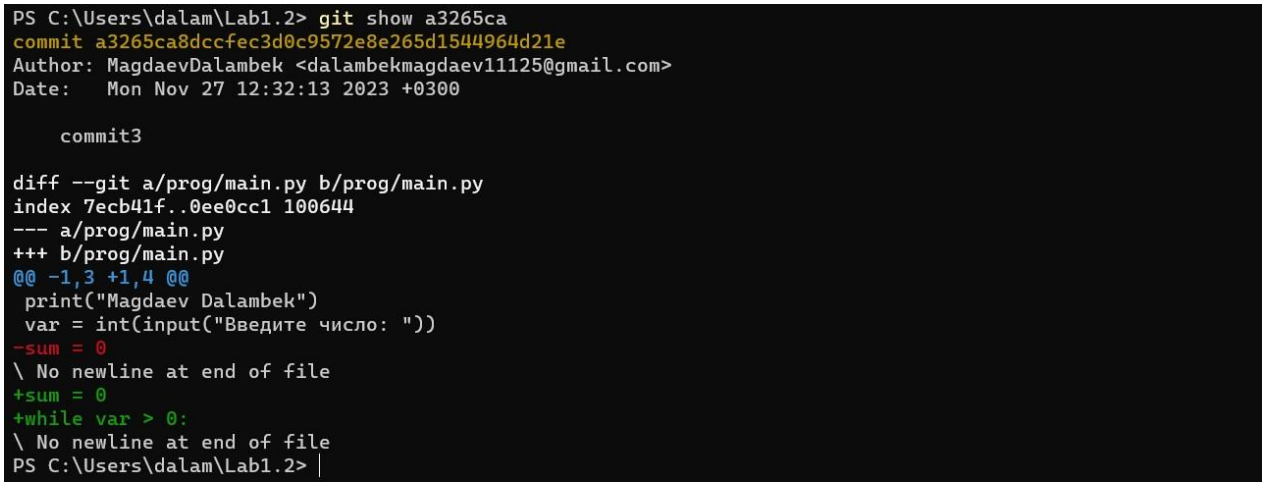

Рисунок 27. Результат работы команды git show a3265ca

9. Удалил код из файла main.py, а затем удалил все несохраненные изменения:

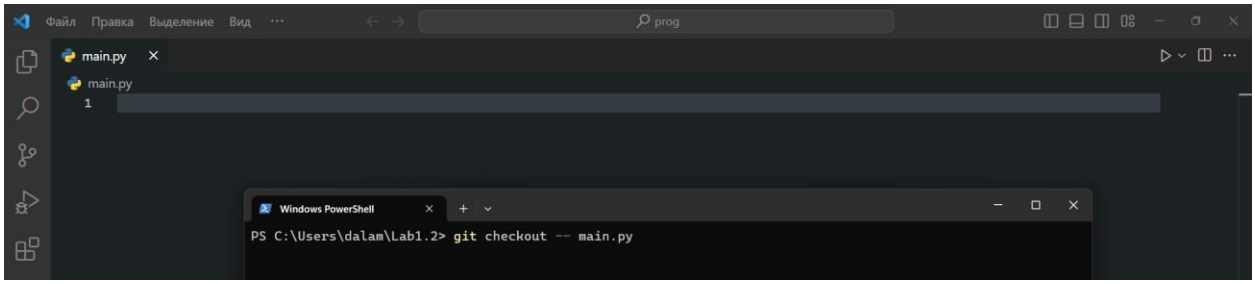

Рисунок 28. Пустой main.py

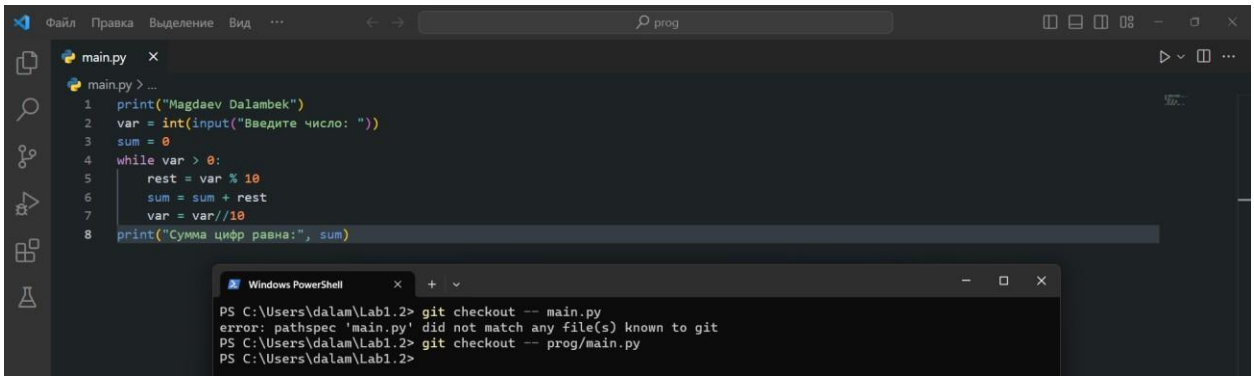

Рисунок 29. Код вернулся после команды git checkout -- prog/main.py

10. Удалил весь код из файла main.py, а затем сделал коммит:

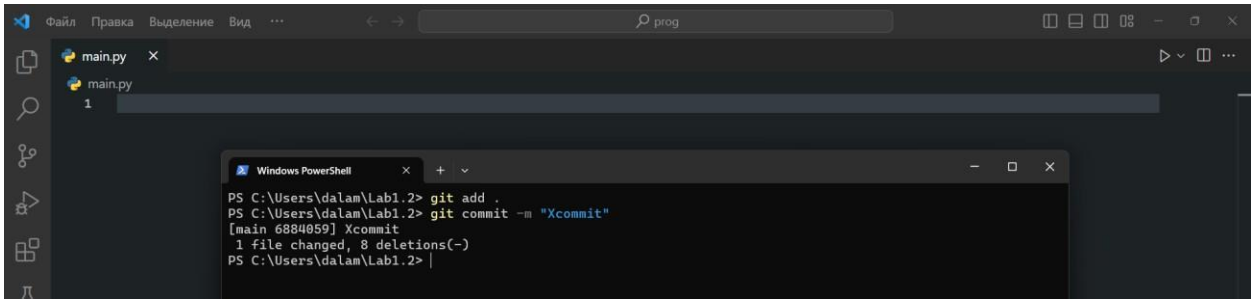

Рисунок 30. Коммит после удаления кода

11. Откатил состояние хранилище к предыдущей версии коммита:

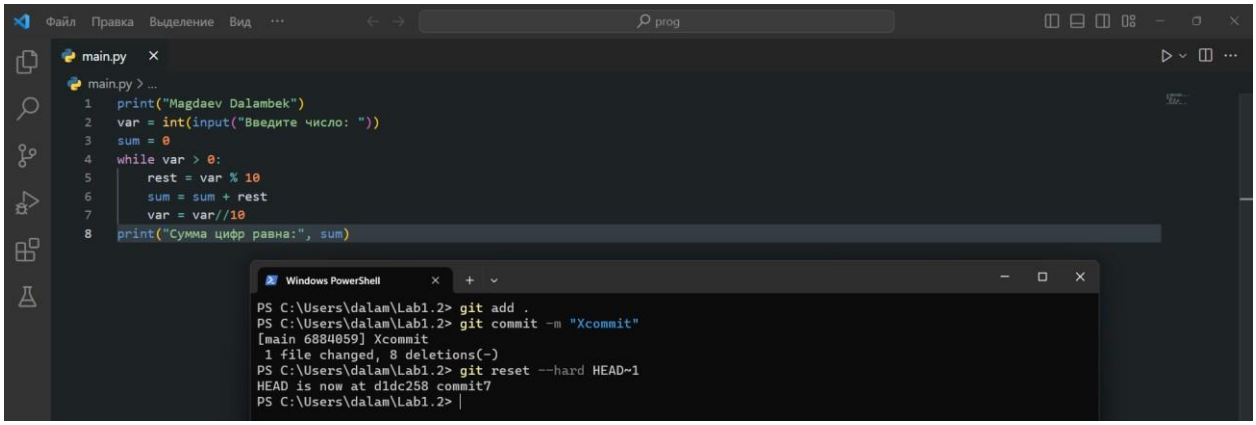

Рисунок 31. Код вернулся после команды git reset --hard HEAD~1

Вывод: чтобы удалить не сохраненные коммитом изменения, можно выполнить команду git checkout -- <имя файла>, это действие удалит все несохраненные изменения, а чтобы удалить сохраненные коммитом изменения, нужно откатить состояние хранилища к предыдущей версии коммита командой git reset --hard HEAD~1, это действие вернет все хранилище к состоянию, которое было зафиксировано в предыдущем коммите. Все изменения, внесенные после этого коммита, будут потеряны.

Ответы на контрольные вопросы:

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

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 – этолюбое 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, скоторого 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, оторые 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 были исследованы возможности Git для работы с локальными репозиториями.