

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

**«Исследование основных возможностей Git и GitHub»
Отчет по лабораторной работе № 1.2
по дисциплине «Основы программной инженерии»**

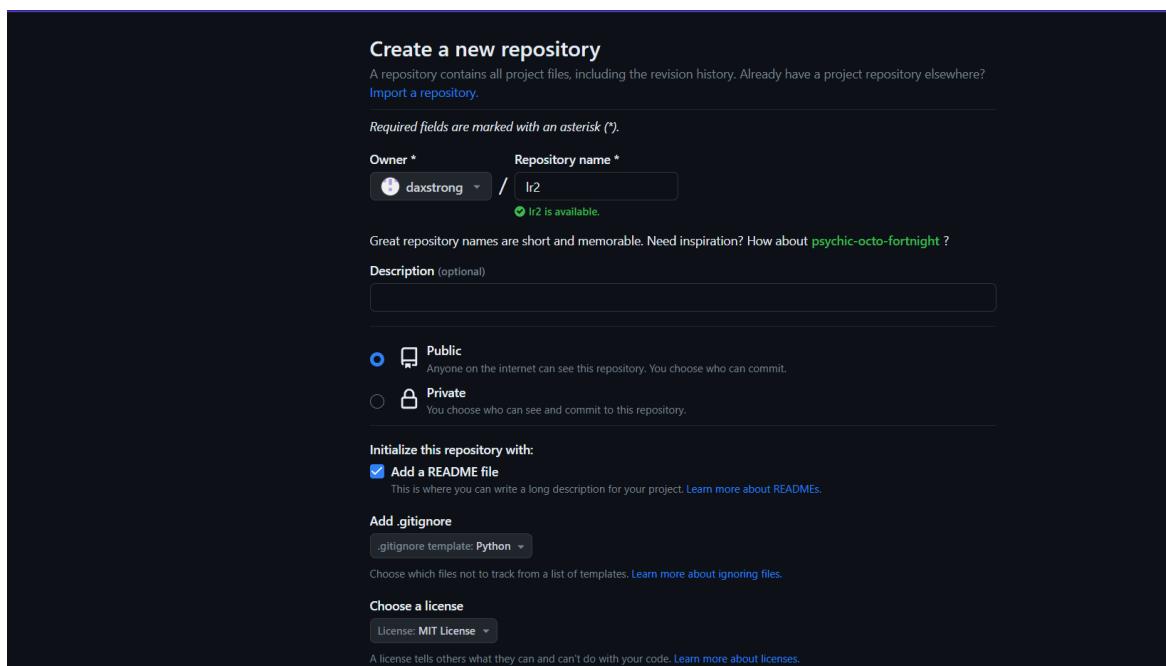
Выполнил студент группы ПИЖ-б-о-22-1
Юрьев Илья Евгеньевич.
Подпись студента _____
Работа защищена « » _____ 20__ г.
Проверил Богданов С.С. _____
(подпись)

Ставрополь 2023

Цель работы: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

Ход работы:

1. Создадим общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный язык программирования:



Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * 👤 daxstrong / Repository name * lr2
✔ lr2 is available.

Great repository names are short and memorable. Need inspiration? How about [psychic-octo-fortnight](#) ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

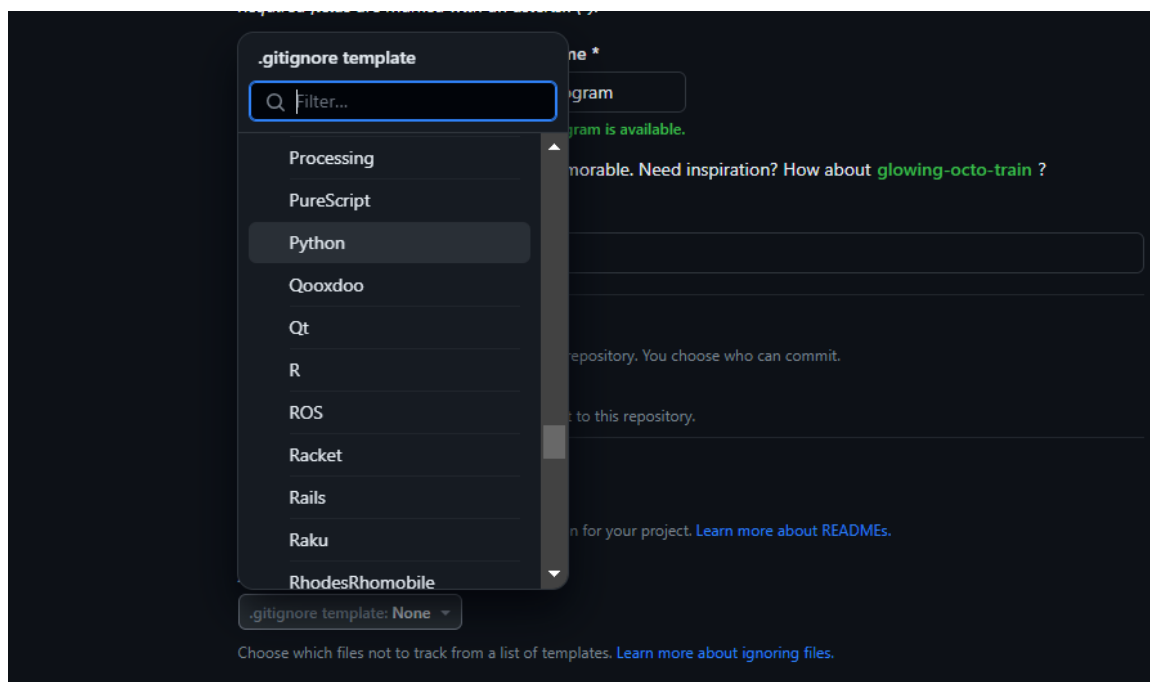
Add .gitignore
.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: MIT License

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

Рисунок 1 – Создание нового репозитория



.gitignore template

Filter...

- Processing
- PureScript
- Python**
- Qooxdoo
- Qt
- R
- ROS
- Racket
- Rails
- Raku
- RhodesRhomobile

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Рисунок 2 – Выбор языка программирования

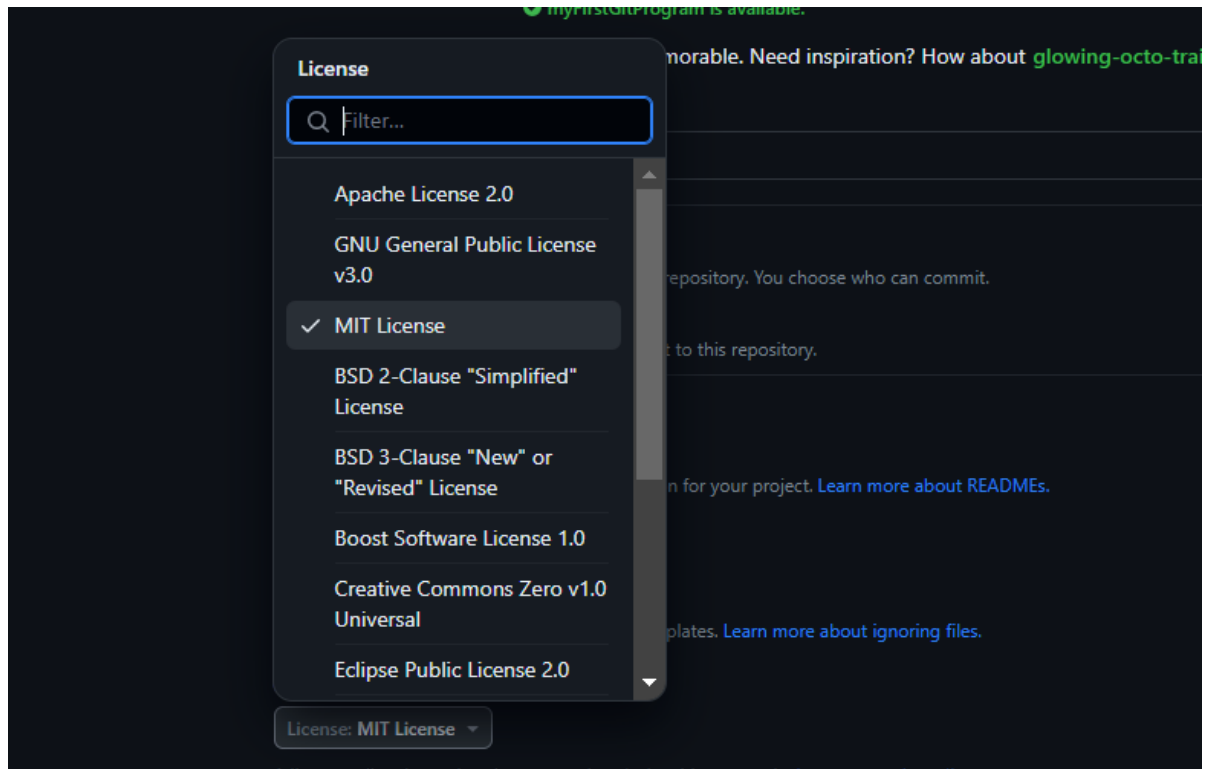


Рисунок 3 – Выбор MIT лицензии

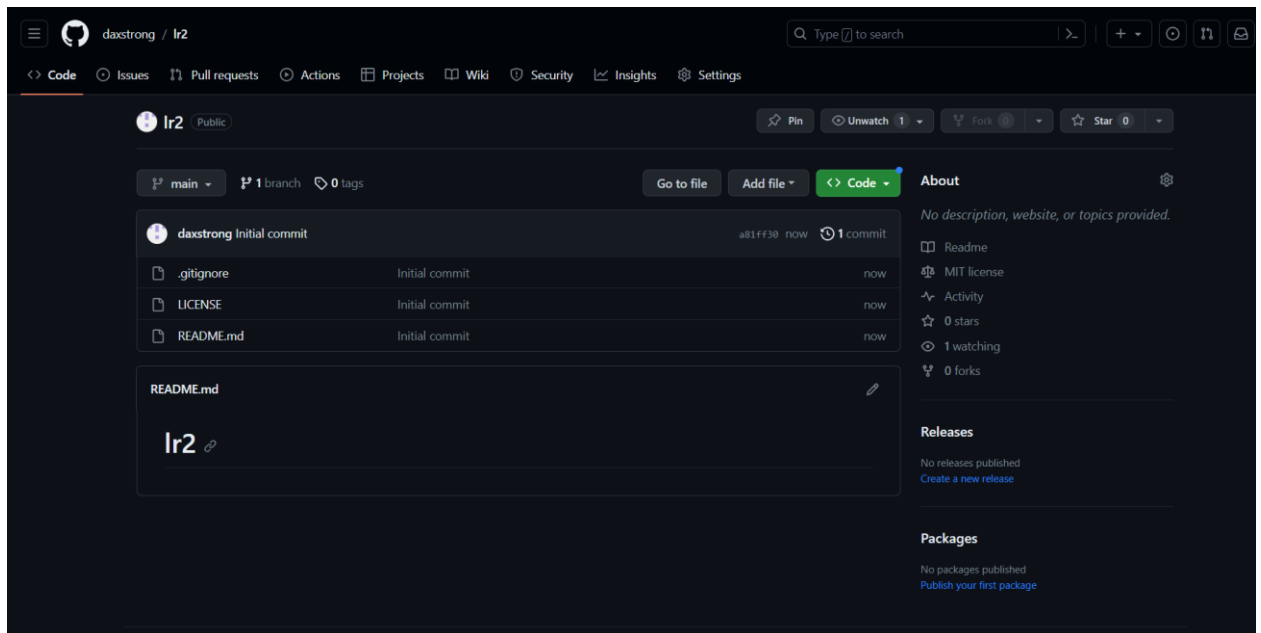
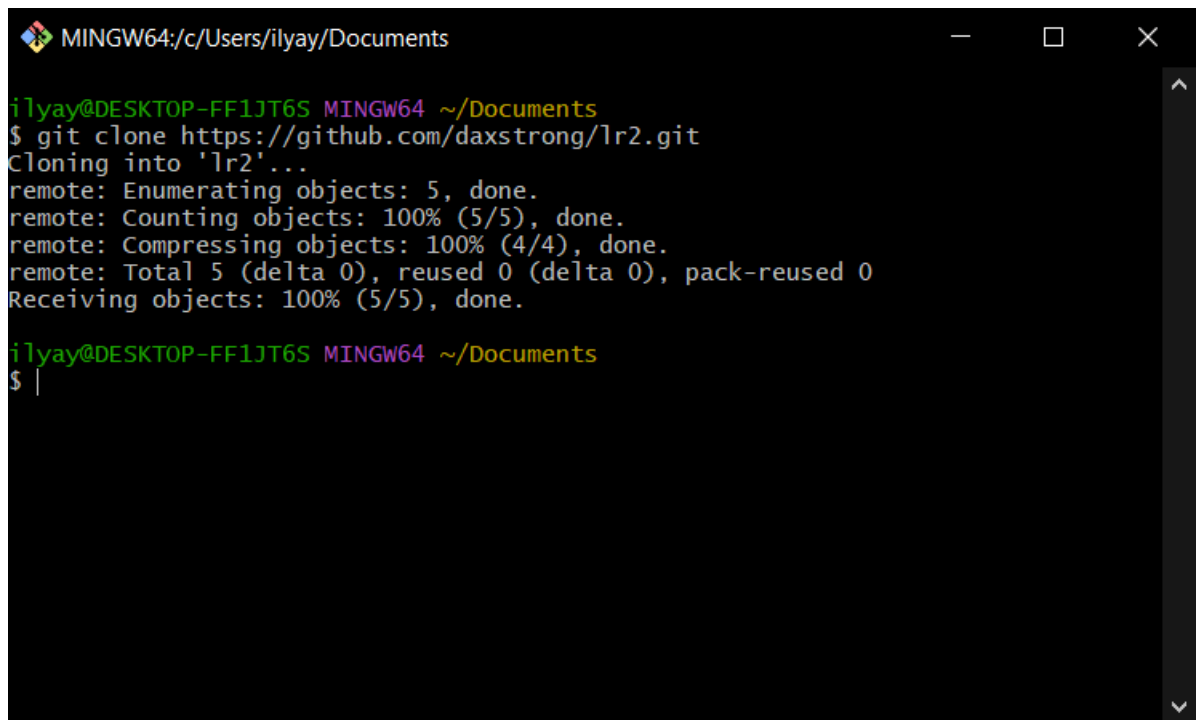


Рисунок 4 – Созданный репозиторий

2. Выполним клонирование созданного репозитория на рабочий компьютер:

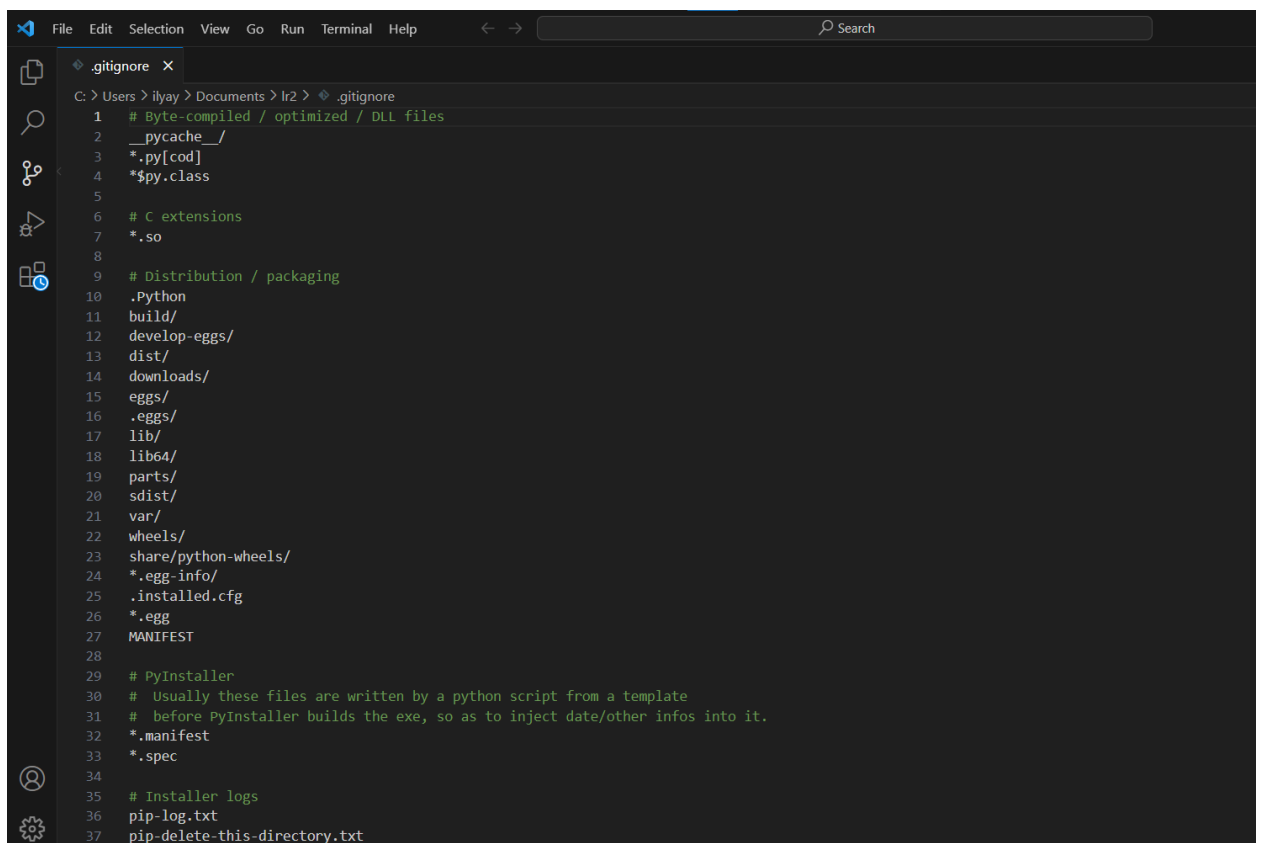


```
MINGW64:/c/Users/ilyay/Documents
ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents
$ git clone https://github.com/daxstrong/lr2.git
Cloning into 'lr2'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents
$ |
```

Рисунок 6 – Клонирование репозитория с помощью команды «git clone»

3. Дополним файл .gitignore необходимыми правилами для языка программирования Python:



```
.gitignore
1 # Byte-compiled / optimized / DLL files
2 __pycache__ /
3 *.py[cod]
4 *$py.class
5
6 # C extensions
7 *.so
8
9 # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28
29 # PyInstaller
30 # Usually these files are written by a python script from a template
31 # before PyInstaller builds the exe, so as to inject date/other infos into it.
32 *.manifest
33 *.spec
34
35 # Installer logs
36 pip-log.txt
37 pip-delete-this-directory.txt
```

Рисунок 7 – Содержимое файла «.gitignore»

4. Добавим в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу:

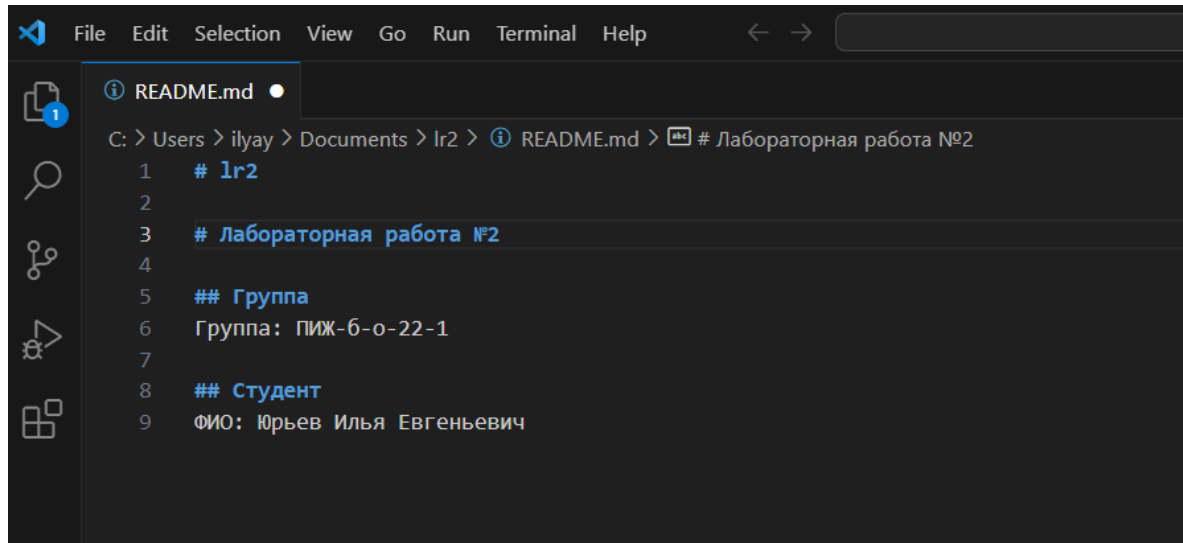


Рисунок 8 – Добавление информации в файл README.md

5. Напишем небольшую программу на выбранном языке программирования. Зафиксируем изменения в локальном репозитории:

```
commit 75005fbd3530df3eafd689a07fc720b4c0228f2f (HEAD -> main)
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 25 23:06:53 2023 +0300

    Завершение программы и окончание работы

commit 2fbf15cb669a96457f9aee990aec09d174049cf3 (tag: v1.2)
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 25 23:03:01 2023 +0300

    Вывод результата на экран

commit 70193d6661e97686814c202d3b2913f945d02a36 (tag: v1.1)
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 25 22:50:24 2023 +0300

    Вычисление среднего арифметического

commit 483c42710ff4ad1016c4df2a3ded072d35d4a43f
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 25 22:49:22 2023 +0300

    Вычисление суммы чисел

commit a30bdfeadac832c5d8f06ce341a772143aeae5f0
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 25 22:48:01 2023 +0300

    Запрос чисел и добавление их в список

commit 0569d9275601fb84a3768c8a46d9b33c2aa7da3b (tag: v1.0)
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 25 22:46:36 2023 +0300

    Добавление запроса количества чисел

commit eb364de0ce94e0c779ebfb9f6fe2bf08cef4e1a8
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 25 22:34:24 2023 +0300

    Инициализация репозитория и создание программы
```

Рисунок 9 – История коммитов и тегов

```
MINGW64:/c/Users/ilyay/Documents/lr2

ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)
$ git log --graph --pretty=oneline --abbrev-commit
* 75005fb (HEAD -> main) Завершение программы и окончание работы
* 2fbf15c (tag: v1.2) Вывод результата на экран
* 70193d6 (tag: v1.1) Вычисление среднего арифметического
* 483c427 Вычисление суммы чисел
* a30bdfe Запрос чисел и добавление их в список
* 0569d92 (tag: v1.0) Добавление запроса количества чисел
* eb364de Инициализация репозитория и создание программы
* a81ff30 (origin/main, origin/HEAD) Initial commit

ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)
$ |
```

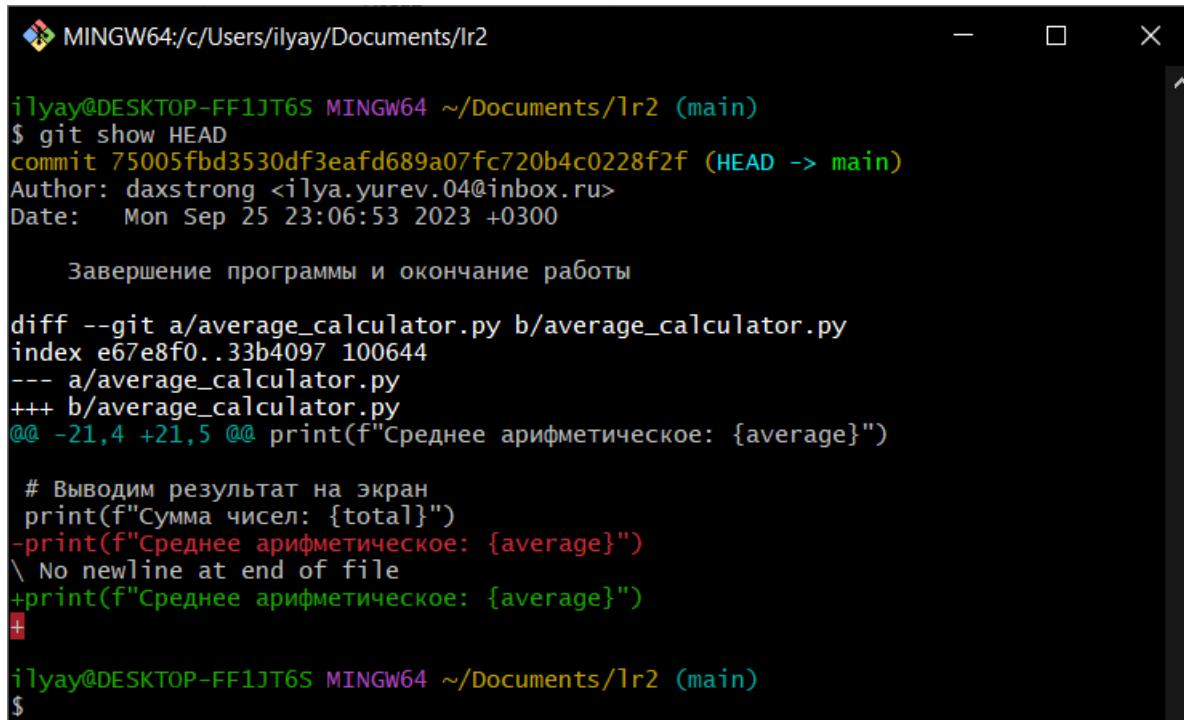
Рисунок 10 – История коммитов

```
File Edit Selection View Go Run Terminal Help
average_calculator.py X

C: > Users > ilyay > Documents > lr2 > average_calculator.py > ...
1 # Инициализация пустого списка для чисел
2 numbers = []
3
4 # Запрашиваем у пользователя количество чисел, которые он хочет ввести
5 count = int(input("Введите количество чисел: "))
6
7 # Запрашиваем числа у пользователя и добавляем их в список
8 for i in range(count):
9     num = float(input(f"Введите число {i+1}: "))
10    numbers.append(num)
11
12 # Вычисляем сумму чисел
13 total = sum(numbers)
14
15 # Вычисляем среднее арифметическое
16 average = total / count
17
18 # Выводим результат на экран
19 print(f"Сумма чисел: {total}")
20 print(f"Среднее арифметическое: {average}")
21
22 # Выводим результат на экран
23 print(f"Сумма чисел: {total}")
24 print(f"Среднее арифметическое: {average}")
25
26 |
```

Рисунок 11 – Готовая программа

6. Просмотреть содержимое коммитов командой `git show <ref>`, где `<ref>`:



```
MINGW64:/c/Users/ilyay/Documents/lr2

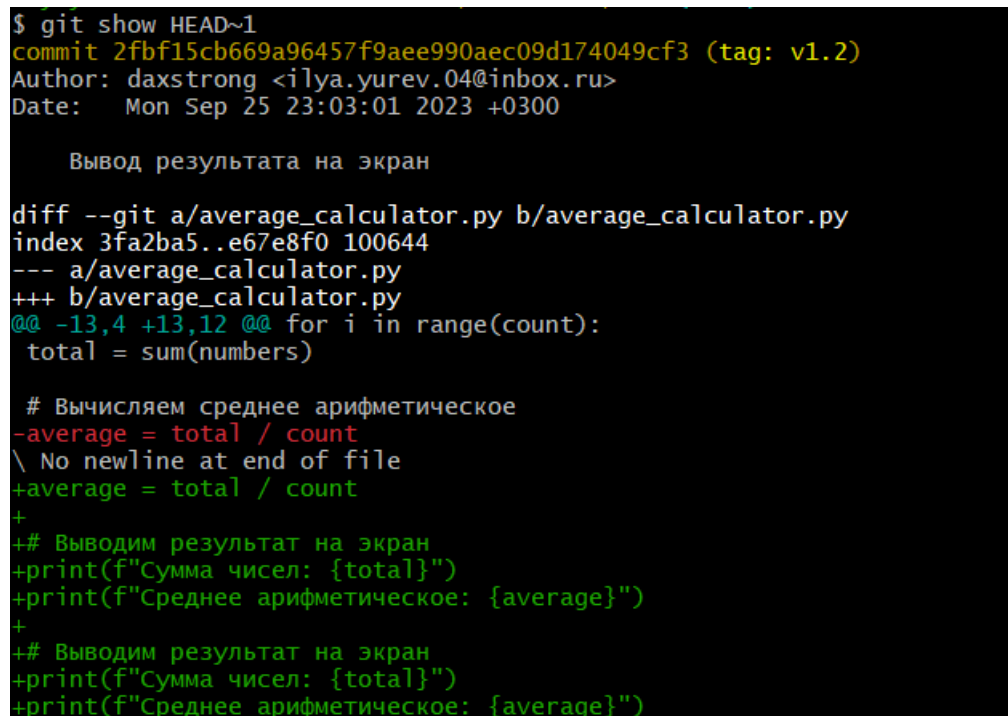
ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)
$ git show HEAD
commit 75005fbd3530df3eafd689a07fc720b4c0228f2f (HEAD -> main)
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 25 23:06:53 2023 +0300

    Завершение программы и окончание работы

diff --git a/average_calculator.py b/average_calculator.py
index e67e8f0..33b4097 100644
--- a/average_calculator.py
+++ b/average_calculator.py
@@ -21,4 +21,5 @@ print(f"Среднее арифметическое: {average}")

 # Выводим результат на экран
 print(f"Сумма чисел: {total}")
-print(f"Среднее арифметическое: {average}")
\ No newline at end of file
+print(f"Среднее арифметическое: {average}")
+
ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)
$
```

Рисунок 12 – `git show HEAD`



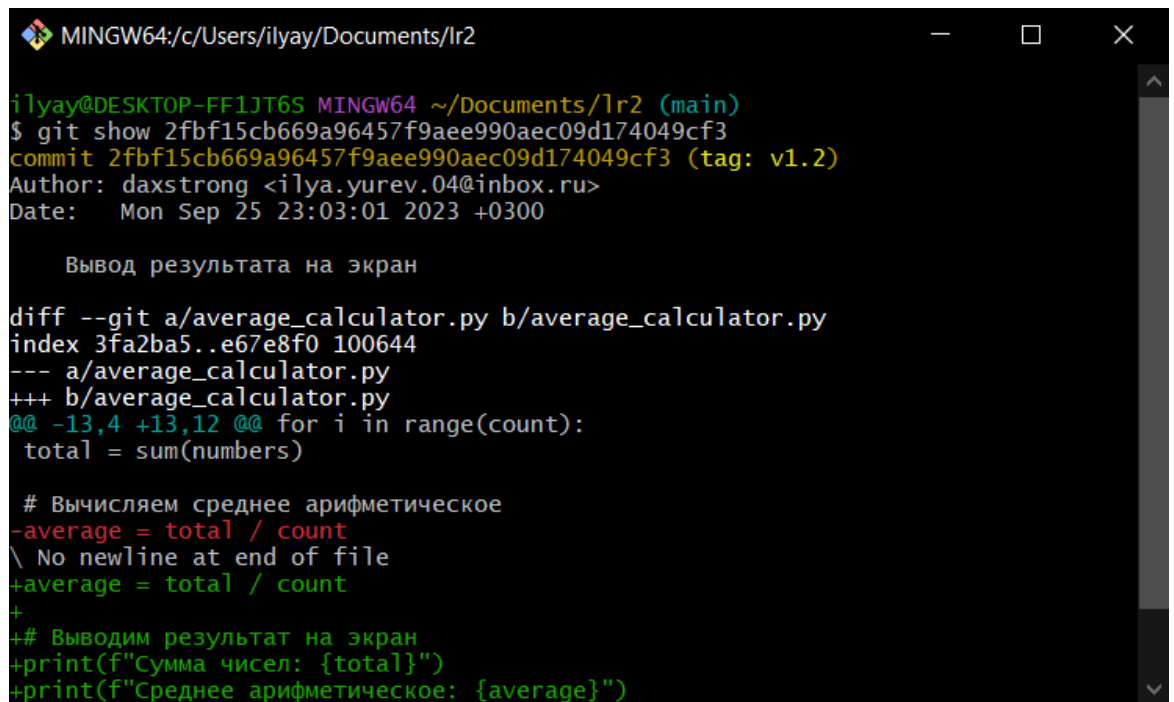
```
$ git show HEAD~1
commit 2fbf15cb669a96457f9aee990aec09d174049cf3 (tag: v1.2)
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 25 23:03:01 2023 +0300

    Вывод результата на экран

diff --git a/average_calculator.py b/average_calculator.py
index 3fa2ba5..e67e8f0 100644
--- a/average_calculator.py
+++ b/average_calculator.py
@@ -13,4 +13,12 @@ for i in range(count):
     total = sum(numbers)

 # Вычисляем среднее арифметическое
-average = total / count
\ No newline at end of file
+average = total / count
+
+# Выводим результат на экран
+print(f"Сумма чисел: {total}")
+print(f"Среднее арифметическое: {average}")
+
+# Выводим результат на экран
+print(f"Сумма чисел: {total}")
+print(f"Среднее арифметическое: {average}")
```

Рисунок 13 – `git show HEAD~1`



```
MINGW64:/c/Users/ilyay/Documents/lr2
ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)
$ git show 2fbf15cb669a96457f9aee990aec09d174049cf3
commit 2fbf15cb669a96457f9aee990aec09d174049cf3 (tag: v1.2)
Author: daxstrong <ilya.yurev.04@inbox.ru>
Date: Mon Sep 25 23:03:01 2023 +0300

    Вывод результата на экран

diff --git a/average_calculator.py b/average_calculator.py
index 3fa2ba5..e67e8f0 100644
--- a/average_calculator.py
+++ b/average_calculator.py
@@ -13,4 +13,12 @@ for i in range(count):
     total = sum(numbers)

    # Вычисляем среднее арифметическое
-    average = total / count
\ No newline at end of file
+    average = total / count
+
+# Выводим результат на экран
+print(f"Сумма чисел: {total}")
+print(f"Среднее арифметическое: {average}")
```

Рисунок 14 – git show <хеш>

7. Освойте возможность отката к заданной версии:

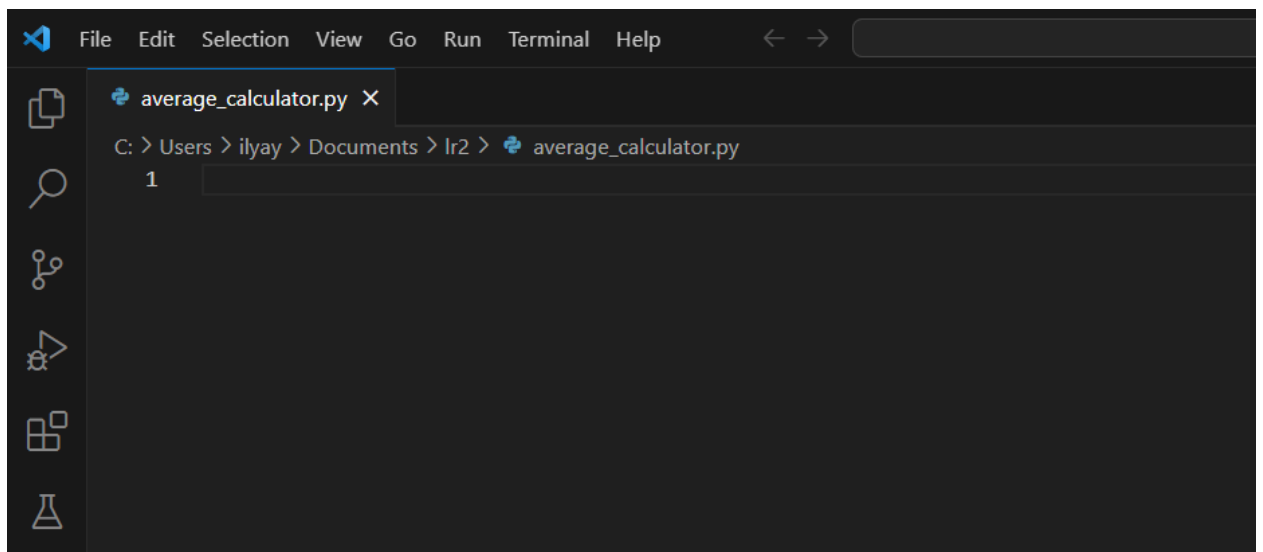
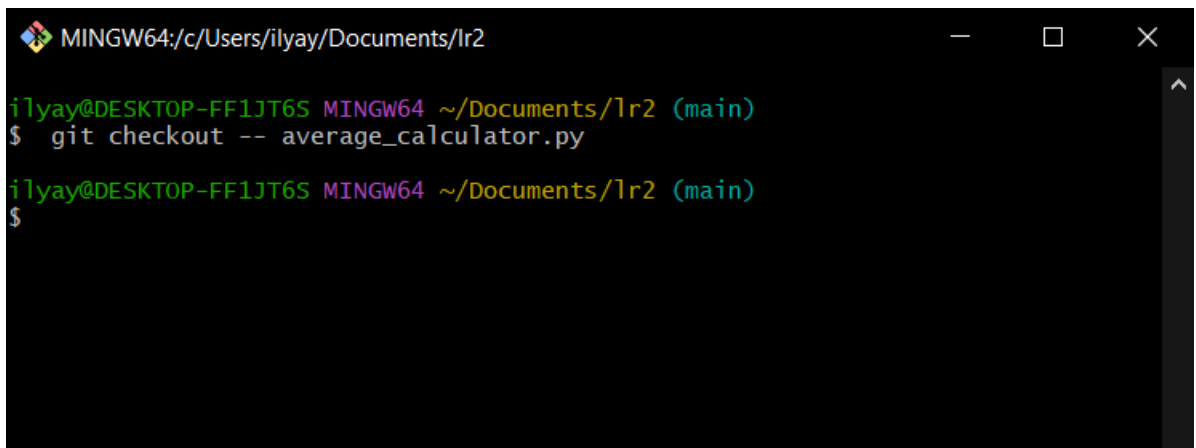
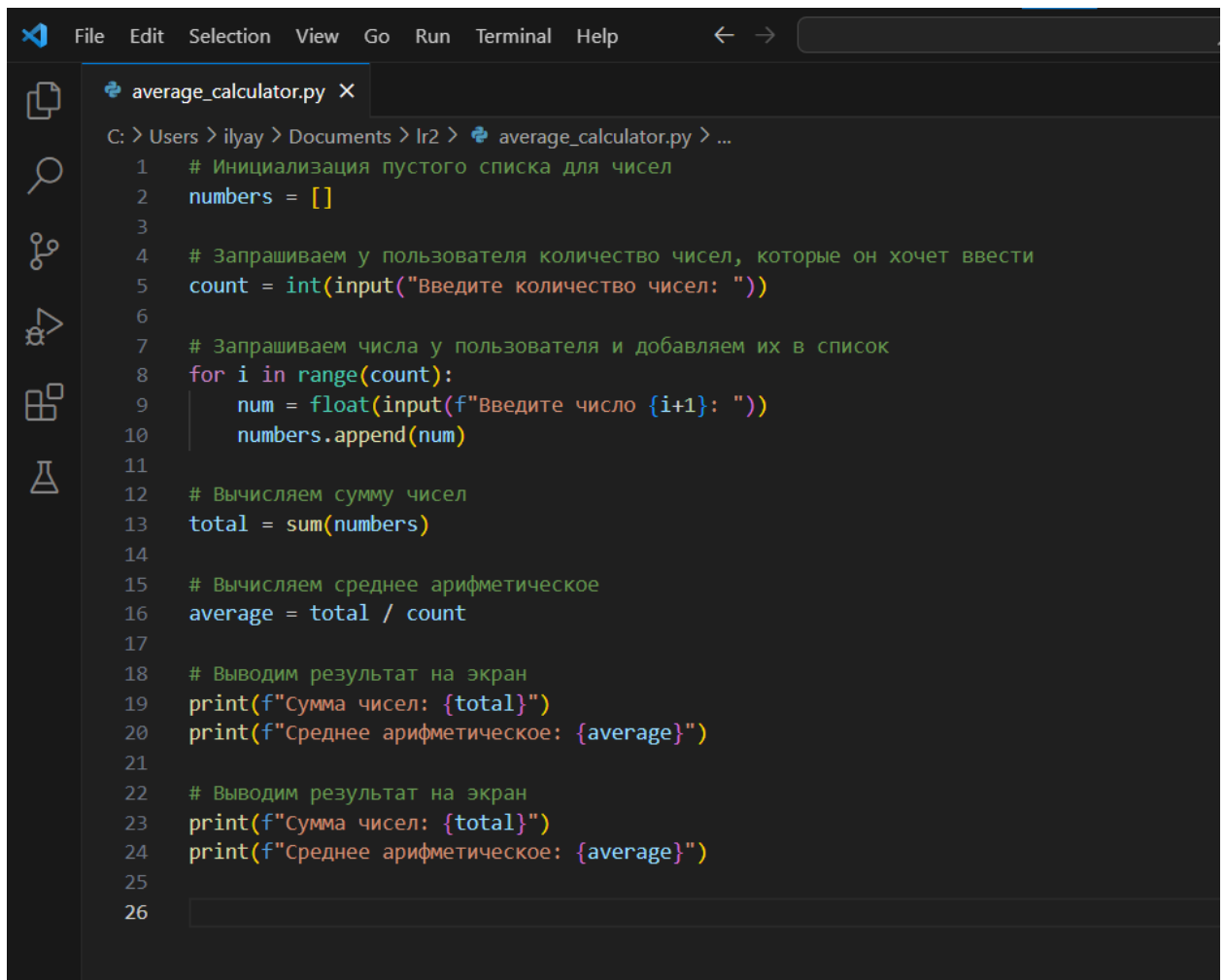


Рисунок 15 – Удаление кода

A terminal window with a black background and green text. The title bar shows the path 'MINGW64:/c/Users/ilyay/Documents/lr2'. The prompt is 'ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)'. The command 'git checkout -- average_calculator.py' has been entered and executed. The prompt is now '\$'.

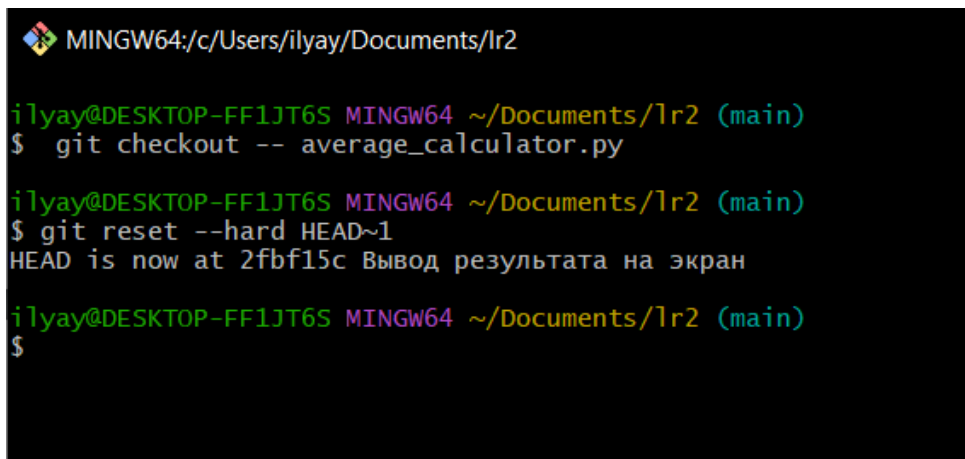
```
MINGW64:/c/Users/ilyay/Documents/lr2
ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)
$ git checkout -- average_calculator.py
ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)
$
```

Рисунок 16 – Удаление всех несохраненных изменений

A screenshot of the Visual Studio Code editor. The file 'average_calculator.py' is open. The code is in Python and calculates the average of a list of numbers. The editor has a dark theme. The left sidebar shows icons for Explorer, Search, Source Control, Run and Debug, Extensions, and Testing. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help.

```
File Edit Selection View Go Run Terminal Help
average_calculator.py X
C: > Users > ilyay > Documents > lr2 > average_calculator.py > ...
1  # Инициализация пустого списка для чисел
2  numbers = []
3
4  # Запрашиваем у пользователя количество чисел, которые он хочет ввести
5  count = int(input("Введите количество чисел: "))
6
7  # Запрашиваем числа у пользователя и добавляем их в список
8  for i in range(count):
9      num = float(input(f"Введите число {i+1}: "))
10     numbers.append(num)
11
12  # Вычисляем сумму чисел
13  total = sum(numbers)
14
15  # Вычисляем среднее арифметическое
16  average = total / count
17
18  # Выводим результат на экран
19  print(f"Сумма чисел: {total}")
20  print(f"Среднее арифметическое: {average}")
21
22  # Выводим результат на экран
23  print(f"Сумма чисел: {total}")
24  print(f"Среднее арифметическое: {average}")
25
26
```

Рисунок 17 – Результат выполнения команды



```
MINGW64:/c/Users/ilyay/Documents/lr2

ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)
$ git checkout -- average_calculator.py

ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)
$ git reset --hard HEAD~1
HEAD is now at 2fbf15c Вывод результата на экран

ilyay@DESKTOP-FF1JT6S MINGW64 ~/Documents/lr2 (main)
$
```

Рисунок 18 – Откат к предыдущей версии

Ответы на контрольные вопросы:

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

Для просмотра истории коммитов в Git используется команда `'git log'`. Дополнительные опции позволяют настроить вывод истории коммитов. Некоторые из них:

Для просмотра истории коммитов в Git используется команда **git log**. Дополнительные опции позволяют настроить вывод истории коммитов. Некоторые из них:

- oneline: Вывести краткую информацию о коммитах на одной строке.
- graph: Вывести историю коммитов в виде графа (если есть ветвления).
- author=<имя_автора>: Показать коммиты только указанного автора.
- since=<дата> и --until=<дата>: Показать коммиты, сделанные между заданными датами.

- grep=<подстрока>: Показать коммиты, содержащие указанную подстроку в сообщении коммита, и много других.

2. Как ограничить вывод при просмотре истории коммитов?

Вы можете ограничить вывод при просмотре истории коммитов, например, указав количество коммитов с флагом **-n**, например: **git log -n 5**, чтобы отобразить только 5 последних коммитов.

3. Как внести изменения в уже сделанный коммит?

Если вы хотите внести изменения в последний коммит, вы можете использовать команду `git commit --amend`. Она позволяет добавить новые изменения к последнему коммиту.

4. Как отменить индексацию файла в Git?

Для отмены индексации файла (убрать файл из "staging area") используйте команду `git reset HEAD <файл>`. Например, `git reset HEAD myfile.txt`.

5. Как отменить изменения в файле?

Для отмены локальных изменений в файле, которые еще не были закоммичены, используйте команду `git checkout -- <файл>`. Это удалит несохраненные изменения в файле.

6. Что такое удаленный репозиторий Git?

Удаленный репозиторий Git (Remote repository) — это репозиторий, который находится на удаленном сервере и используется для хранения и обмена кодом с другими участниками проекта. Он может быть общедоступным или приватным.

7. Как выполнить просмотр удаленных репозиториях данного локального репозитория?

Вы можете просмотреть список удаленных репозиториях с помощью команды `git remote -v`. Она выведет список удаленных репозиториях и их URL.

8. Как добавить удаленный репозиторий для данного локального репозитория?

Чтобы добавить удаленный репозиторий, используйте команду `git remote add <имя> <URL>`, где <имя> - это имя удаленного репозитория, а <URL> - URL удаленного репозитория.

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

Для отправки изменений на удаленный репозиторий используйте `git push`, а для получения изменений из удаленного репозитория используйте `git pull`.

10. Как выполнить просмотр удаленного репозитория?

Чтобы просмотреть содержимое удаленного репозитория без его клонирования, вы можете воспользоваться командой `git ls-remote <URL>`, где `<URL>` - URL удаленного репозитория. Эта команда выведет список удаленных веток и хэшей коммитов.

11. Каково назначение тэгов Git?

Тэги Git используются для пометки определенных коммитов как важных точек в истории репозитория. Они часто используются для обозначения версий вашего проекта или для выделения конкретных релизов.

12. Как осуществляется работа с тэгами Git?

Для создания тэга используйте команду `git tag <имя_тэга>`, например, `git tag v1.0`. Чтобы ассоциировать тэг с определенным коммитом, укажите его хэш, например, `git tag -a v1.0 -m "Первый релиз" <хэш_коммита>`. Для отправки тэгов на удаленный репозиторий используйте `git push origin <имя_тэга>`. Для просмотра доступных тэгов используйте `git tag`.

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

Флаг `--prune` в командах `git fetch` и `git push` используется для удаления удаленных веток или тэгов, которые больше не существуют на удаленном репозитории. Это помогает поддерживать локальное и удаленное хранилище в синхронизированном состоянии, удаляя устаревшие ссылки на ветки и тэги.