

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 (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 \_\_\_\_\_

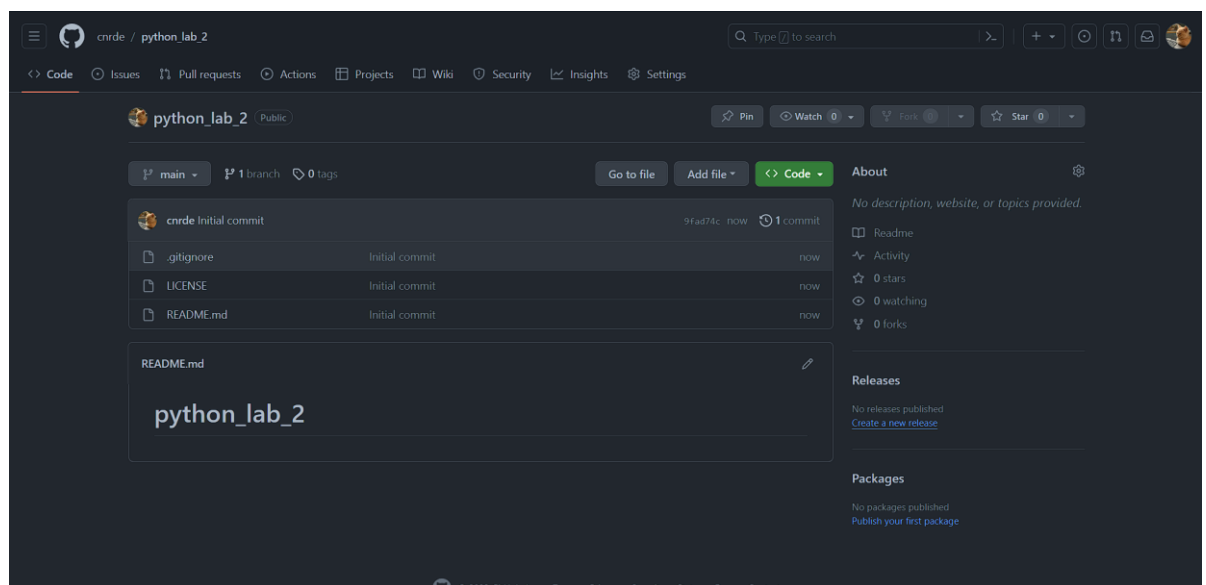
Ставрополь, 2023 г.

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.



3. 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.

```
Администратор: Командная строка

C:\Users\User\Desktop>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>
```

Рисунок 1. Клонирование репозитория.

```
Администратор: Командная строка

C:\Users\User\Desktop\simplegit-progit>git log
commit ca82a6dff817ec66f44342007202690a93763949 (HEAD -> master, origin/master, origin/HEAD)
Author: Scott Chacon <schacon@gmail.com>
Date: Mon Mar 17 21:52:11 2008 -0700

    changed the version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gmail.com>
Date: Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gmail.com>
Date: Sat Mar 15 10:31:28 2008 -0700

    first commit

C:\Users\User\Desktop\simplegit-progit>
```

Рисунок 2. Результат работы команды git log.

```
Администратор: Командная строка

C:\Users\User\Desktop\simplegit-progit>git log -p -2
commit ca82a6dff817ec66f44342007202690a93763949 (HEAD -> master, origin/master, origin/HEAD)
Author: Scott Chacon <schacon@gmail.com>
Date: Mon Mar 17 21:52:11 2008 -0700

    changed the verison number

diff --git a/Rakefile b/Rakefile
index a874b73..8f94139 100644
--- a/Rakefile
+++ b/Rakefile
@@ -5,7 +5,7 @@ require 'rake/gempackagetask'
 spec = Gem::Specification.new do |s|
   s.platform = Gem::Platform::RUBY
   s.name = "simplegit"
-  s.version = "0.1.0"
+  s.version = "0.1.1"
   s.author = "Scott Chacon"
   s.email = "schacon@gmail.com"
   s.summary = "A simple gem for using Git in Ruby code."

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gmail.com>
Date: Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code

diff --git a/lib/simplegit.rb b/lib/simplegit.rb
index a0a60ae..47c6340 100644
--- a/lib/simplegit.rb
+++ b/lib/simplegit.rb
@@ -18,8 +18,3 @@ class SimpleGit
   end

end

- if $0 == __FILE__
-   git = SimpleGit.new
-   puts git.show
- end
\ No newline at end of file

C:\Users\User\Desktop\simplegit-progit>
```

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

```
Администратор: Командная строка

C:\Users\User\Desktop\simplegit-progit>git log -stat
error: did you mean '--stat' (with two dashes)?

C:\Users\User\Desktop\simplegit-progit>git log --stat
commit ca82a6dff817ec66f44342007202690a93763949 (HEAD -> master, origin/master, origin/HEAD)
Author: Scott Chacon <schacon@gmail.com>
Date: Mon Mar 17 21:52:11 2008 -0700

    changed the verison number

Rakefile | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gmail.com>
Date: Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code

lib/simplegit.rb | 5 -----
1 file changed, 5 deletions(-)

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gmail.com>
Date: Sat Mar 15 10:31:28 2008 -0700

    first commit

README      | 6 +++++
Rakefile    | 23 ++++++
lib/simplegit.rb | 25 ++++++
3 files changed, 54 insertions(+)

C:\Users\User\Desktop\simplegit-progit>
```

Рисунок 4. Результат работы команды `git log --stat`.

```
Администратор: Командная строка

C:\Users\User\Desktop\simplegit-progit>git log --pretty=oneline
ca82a6dff817ec66f44342007202690a93763949 (HEAD -> master, origin/master, origin/HEAD) changed the veris
on number
085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7 removed unnecessary test code
a11bef06a3f659402fe7563abf99ad00de2209e6 first commit
```

Рисунок 5. Результат работы команды `git log --pretty=oneline`.

```
Администратор: Командная строка

C:\Users\User\Desktop\simplegit-progit>git log --pretty=format:"%h - %an, %ar : %s"
ca82a6d - Scott Chacon, 16 years ago : changed the verison number
085bb3b - Scott Chacon, 16 years ago : removed unnecessary test code
a11bef0 - Scott Chacon, 16 years ago : first commit
```

Рисунок 6. Результат работы команды  
`git log --pretty=format:"%h - %an, %ar : %s"`.

```
Администратор: Командная строка

C:\Users\User\Desktop\simplegit-progit>git log --pretty=format:"%h %s" --graph
* ca82a6d changed the verison number
* 085bb3b removed unnecessary test code
* a11bef0 first commit

C:\Users\User\Desktop\simplegit-progit>
```

Рисунок 7. Результат работы команды  
git log --pretty=format:"%h %s" --graph.

```
Администратор: Командная строка

C:\Users\User\Desktop\simplegit-progit>cd C:\Users\User\Desktop

C:\Users\User\Desktop>git clone https://github.com/schacon/ticgit
Cloning into 'ticgit'...
remote: Enumerating objects: 1857, done.
remote: Total 1857 (delta 0), reused 0 (delta 0), pack-reused 1857
Receiving objects: 100% (1857/1857), 334.06 KiB | 952.00 KiB/s, done.
Resolving deltas: 100% (837/837), done.
```

Рисунок 8. Клонирование репозитория ticgit.

```
Администратор: Командная строка

C:\Users\User\Desktop\ticgit>git remote
origin
```

Рисунок 9. Результат работы команды git remote.

```
Администратор: Командная строка

C:\Users\User\Desktop\ticgit>git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)
```

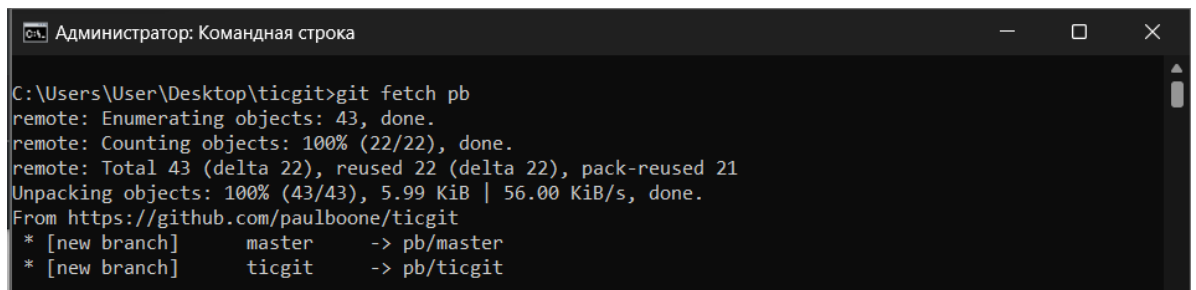
Рисунок 10. Результат работы команды git remote -v.

```
Администратор: Командная строка

C:\Users\User\Desktop\ticgit>git remote add pb https://github.com/paulboone/ticgit

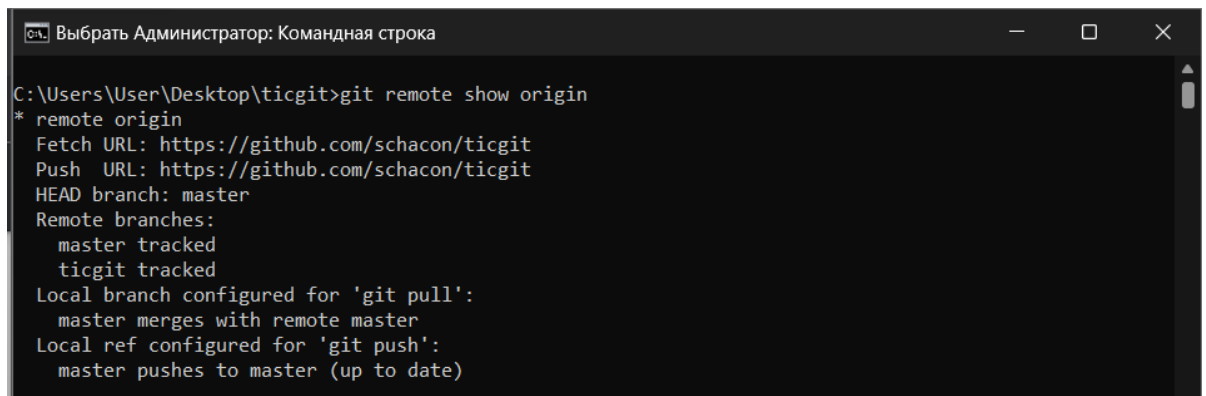
C:\Users\User\Desktop\ticgit>git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)
pb      https://github.com/paulboone/ticgit (fetch)
pb      https://github.com/paulboone/ticgit (push)
```

Рисунок 11. Результат работы команды  
git remote add pb <https://github.com/paulboone/ticgit>



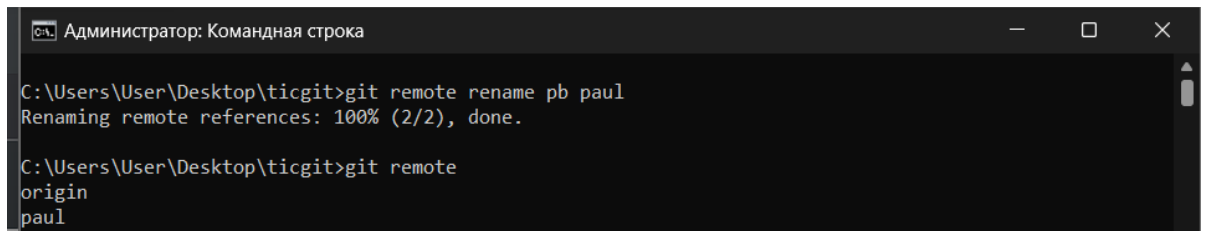
```
C:\Users\User\Desktop\ticgit>git fetch pb
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (22/22), done.
remote: Total 43 (delta 22), reused 22 (delta 22), pack-reused 21
Unpacking objects: 100% (43/43), 5.99 KiB | 56.00 KiB/s, done.
From https://github.com/paulboone/ticgit
* [new branch]      master    -> pb/master
* [new branch]      ticgit    -> pb/ticgit
```

Рисунок 12. Результат работы команды git fetch pb.



```
C:\Users\User\Desktop\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)
```

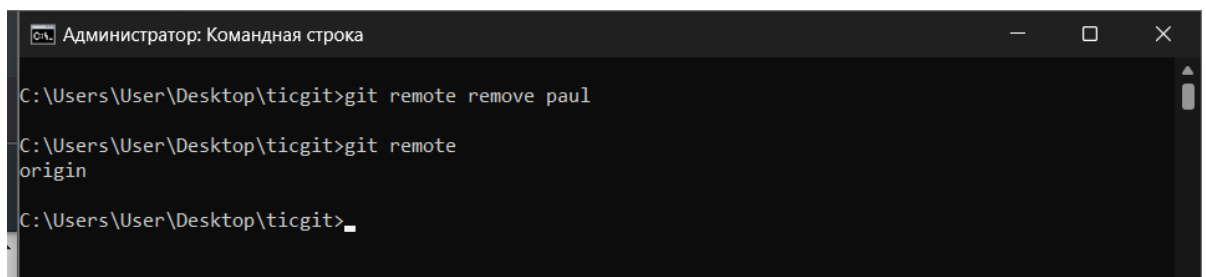
Рисунок 13. Результат работы команды git remote show origin.



```
C:\Users\User\Desktop\ticgit>git remote rename pb paul
Renaming remote references: 100% (2/2), done.

C:\Users\User\Desktop\ticgit>git remote
origin
paul
```

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



```
C:\Users\User\Desktop\ticgit>git remote remove paul

C:\Users\User\Desktop\ticgit>git remote
origin

C:\Users\User\Desktop\ticgit>
```

Рисунок 15. Результат работы команды git remote remove paul.

```
Администратор: Командная строка

C:\Users\User\Desktop\ticgit>git tag

C:\Users\User\Desktop\ticgit>git tag -l "v1.8.5*"

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

C:\Users\User\Desktop\ticgit>git tag
v1.4
```

Рисунок 16. Создал аннотированный тег.

```
Администратор: Командная строка

C:\Users\User\Desktop\ticgit>git show v1.4
tag v1.4
Tagger: cnrde <conrade.tube@gmail.com>
Date: Mon Dec 4 20:43:30 2023 +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 confusing situation. The schacon TicGit repo, this one, is not consistently maintained. For
+up to date TicGit-ng info and code, 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
```

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

```
Администратор: Командная строка

C:\Users\User\Desktop\ticgit>git tag -d v1.4
Deleted tag 'v1.4' (was 55fcfbcb)
```

Рисунок 18. Удаление ранее созданного тега.



#### 4. Выполнил клонирование созданного репозитория.

```
Выбрать Администратор: Командная строка

C:\Users\User\Desktop\Учеба\СКФУ\3 семестр\Воронкин\Питон\Лаб2>git clone https://github.com/cnrde/python_lab_2.git
Cloning into 'python_lab_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.

C:\Users\User\Desktop\Учеба\СКФУ\3 семестр\Воронкин\Питон\Лаб2>
```

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

```
python_lab_2 / .gitignore

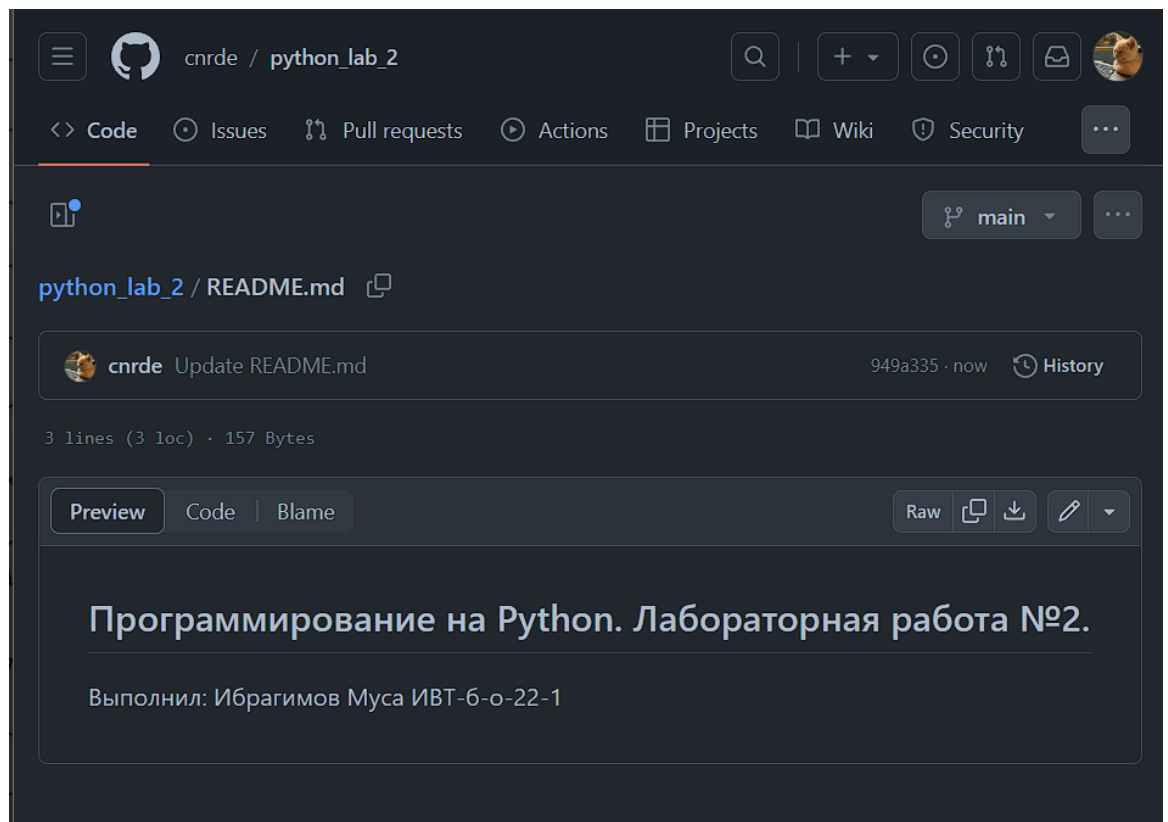
cnrde Update .gitignore 7e84d7b · 2 minutes ago History

164 lines (134 loc) · 3.04 KB

Code Blame Raw Copy Download Edit View Source

1  #.idea/
2  **/.DS_Store
3  .vscode
4
5  # Byte-compiled / optimized / DLL files
6  __pycache__/
7  *.py[.cod]
8  *$py.class
9
10 # C extensions
11 *.so
12
13 # Distribution / packaging
14 .Python
15 build/
16 develop-eggs/
17 dist/
18 downloads/
19 eggs/
20 .eggs/
21 lib/
22 lib64/
23 parts/
24 sdist/
25 var/
```

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



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

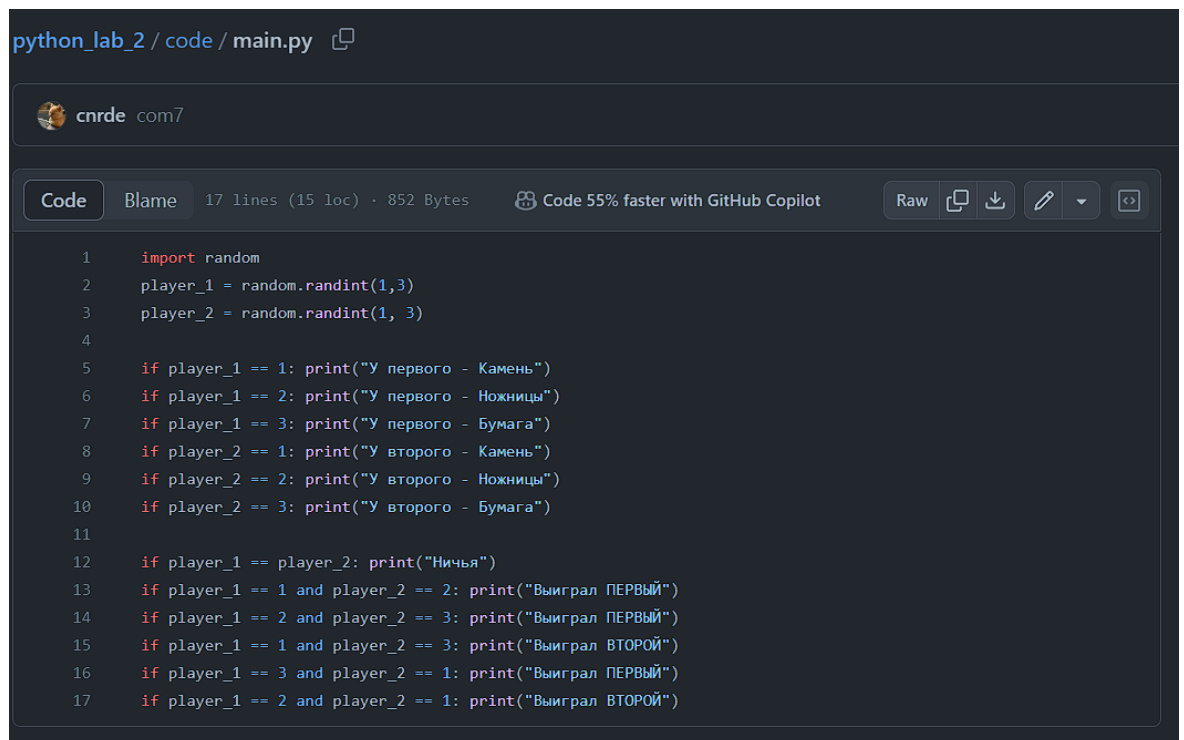


Рисунок 1. Небольшая программа.

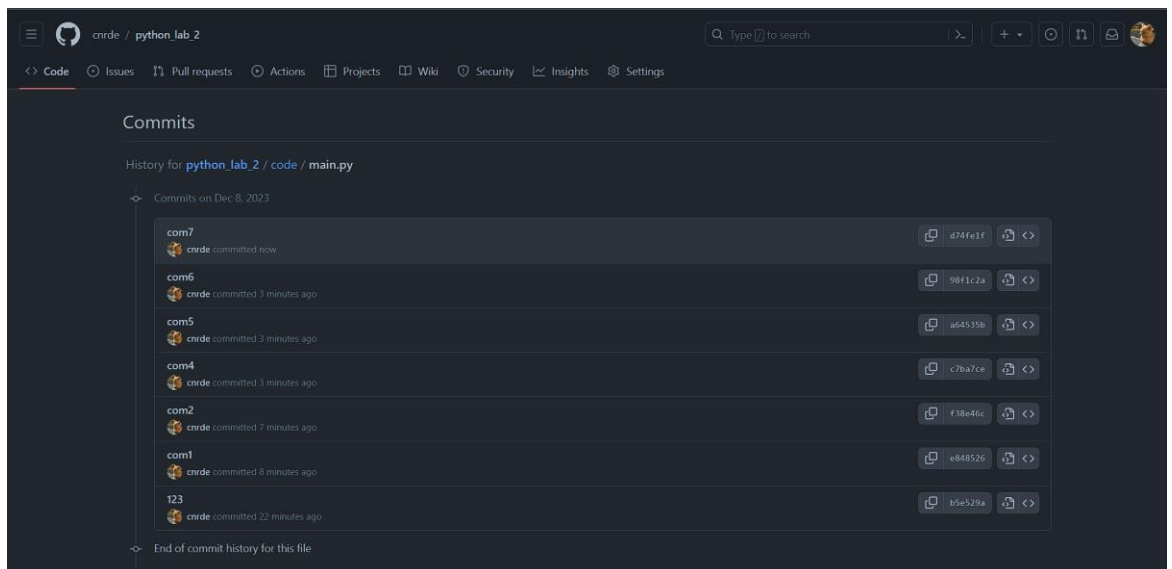
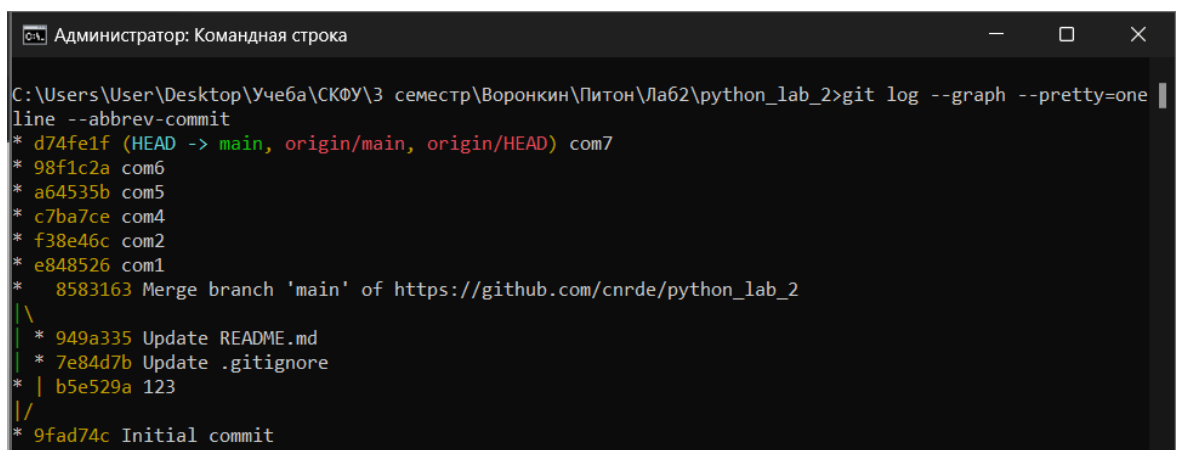
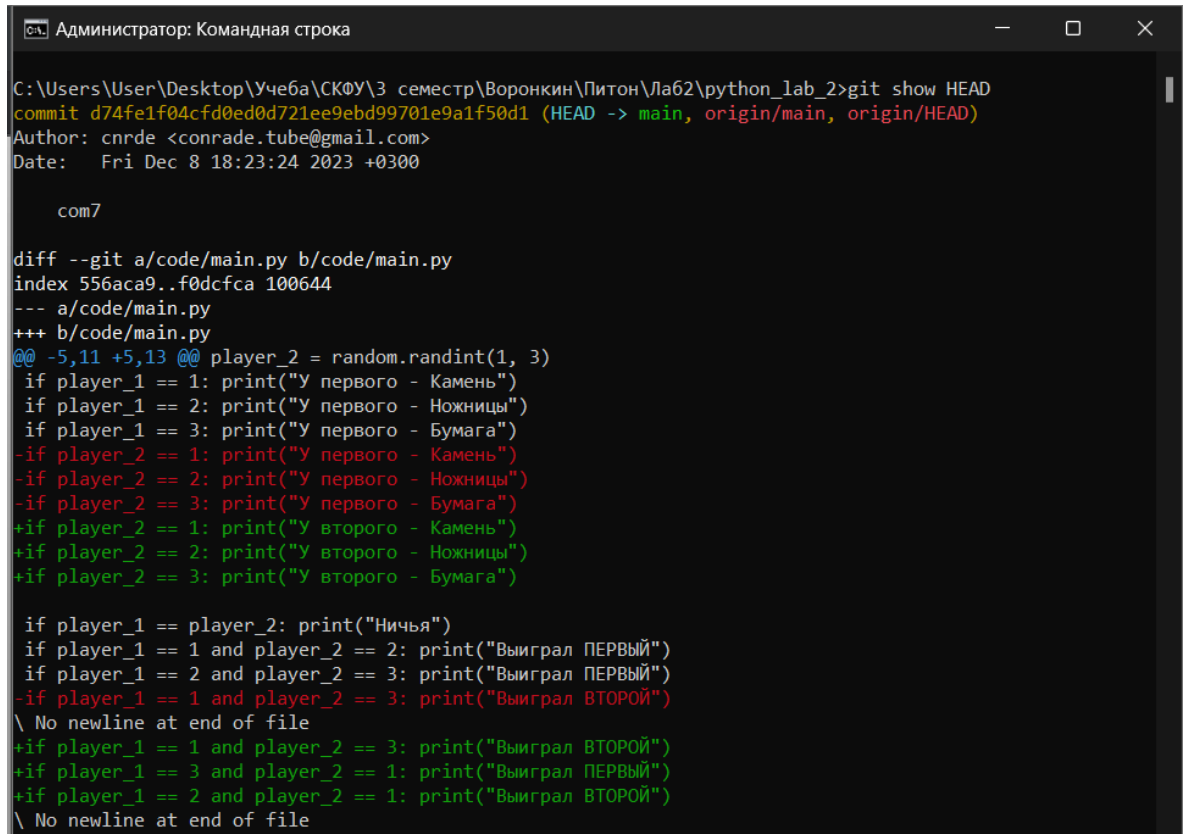


Рисунок 2. История коммитов.

8. Просмотрел историю хранилища командой `git log --graph --pretty=oneline --abbrev-commit`.



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



```
Администратор: Командная строка

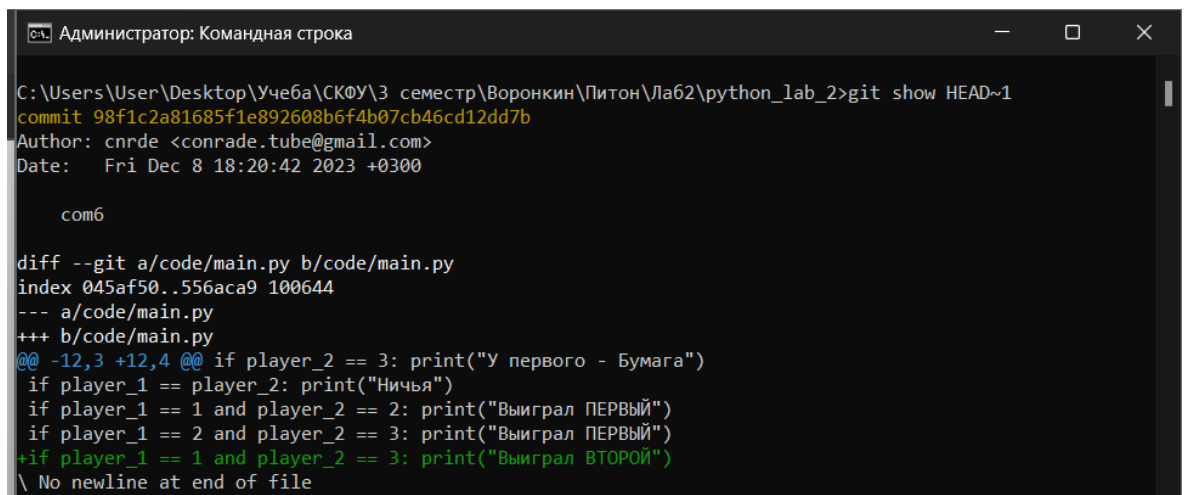
C:\Users\User\Desktop\Учеба\СКФУ\3 семестр\Воронкин\Питон\Лаб2\python_lab_2>git show HEAD
commit d74fe1f04cfd0ed0d721ee9ebd99701e9a1f50d1 (HEAD -> main, origin/main, origin/HEAD)
Author: cnrde <conrade.tube@gmail.com>
Date:   Fri Dec 8 18:23:24 2023 +0300

    com7

diff --git a/code/main.py b/code/main.py
index 556aca9..f0dcfca 100644
--- a/code/main.py
+++ b/code/main.py
@@ -5,11 +5,13 @@ player_2 = random.randint(1, 3)
 if player_1 == 1: print("У первого - Камень")
 if player_1 == 2: print("У первого - Ножницы")
 if player_1 == 3: print("У первого - Бумага")
-if player_2 == 1: print("У первого - Камень")
-if player_2 == 2: print("У первого - Ножницы")
-if player_2 == 3: print("У первого - Бумага")
+if player_2 == 1: print("У второго - Камень")
+if player_2 == 2: print("У второго - Ножницы")
+if player_2 == 3: print("У второго - Бумага")

 if player_1 == player_2: print("Ничья")
 if player_1 == 1 and player_2 == 2: print("Выиграл ПЕРВЫЙ")
 if player_1 == 2 and player_2 == 3: print("Выиграл ПЕРВЫЙ")
-if player_1 == 1 and player_2 == 3: print("Выиграл ВТОРОЙ")
\ No newline at end of file
+if player_1 == 1 and player_2 == 3: print("Выиграл ВТОРОЙ")
+if player_1 == 3 and player_2 == 1: print("Выиграл ПЕРВЫЙ")
+if player_1 == 2 and player_2 == 1: print("Выиграл ВТОРОЙ")
\ No newline at end of file
```

Рисунок 1. Ввёл команду `git show HEAD`.



```
Администратор: Командная строка

C:\Users\User\Desktop\Учеба\СКФУ\3 семестр\Воронкин\Питон\Лаб2\python_lab_2>git show HEAD~1
commit 98f1c2a81685f1e892608b6f4b07cb46cd12dd7b
Author: cnrde <conrade.tube@gmail.com>
Date:   Fri Dec 8 18:20:42 2023 +0300

    com6

diff --git a/code/main.py b/code/main.py
index 045af50..556aca9 100644
--- a/code/main.py
+++ b/code/main.py
@@ -12,3 +12,4 @@ if player_2 == 3: print("У первого - Бумага")
 if player_1 == player_2: print("Ничья")
 if player_1 == 1 and player_2 == 2: print("Выиграл ПЕРВЫЙ")
 if player_1 == 2 and player_2 == 3: print("Выиграл ПЕРВЫЙ")
+if player_1 == 1 and player_2 == 3: print("Выиграл ВТОРОЙ")
\ No newline at end of file
```

Рисунок 2. Ввёл команду `git show HEAD~1`.

```
Администратор: Командная строка

C:\Users\User\Desktop\Учеба\СКФУ\3 семестр\Воронкин\Питон\Лаб2\python_lab_2>git show 98f1c2a816
commit 98f1c2a81685f1e892608b6f4b07cb46cd12dd7b
Author: cnrde <conrade.tube@gmail.com>
Date:   Fri Dec 8 18:20:42 2023 +0300

    комб

diff --git a/code/main.py b/code/main.py
index 045af50..556aca9 100644
--- a/code/main.py
+++ b/code/main.py
@@ -12,3 +12,4 @@ if player_2 == 3: print("У первого - Бумага")
if player_1 == player_2: print("Ничья")
if player_1 == 1 and player_2 == 2: print("Выиграл ПЕРВЫЙ")
if player_1 == 2 and player_2 == 3: print("Выиграл ПЕРВЫЙ")
+if player_1 == 1 and player_2 == 3: print("Выиграл ВТОРОЙ")
\ No newline at end of file
```

Рисунок 3. Ввел команду git show 98f1c2a816.

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

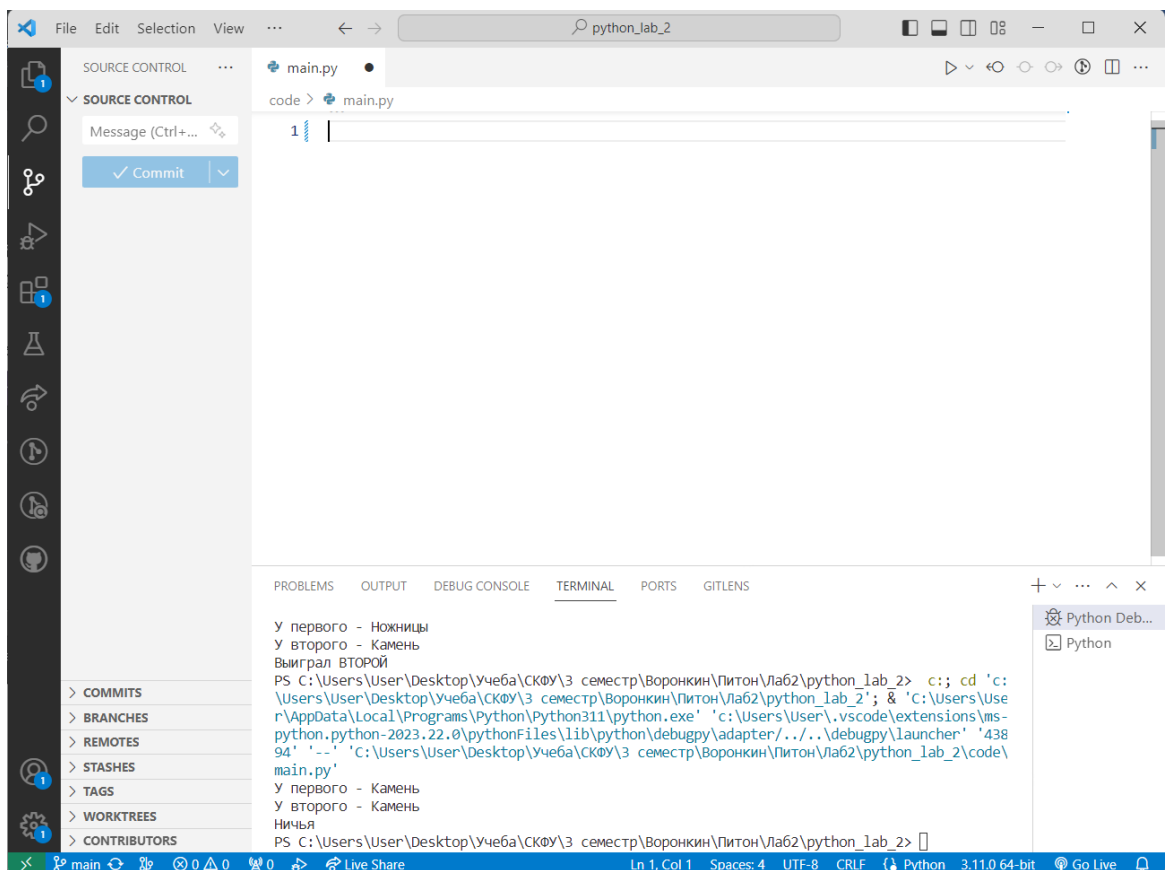


Рисунок 1. Удалил код из файла main.py.

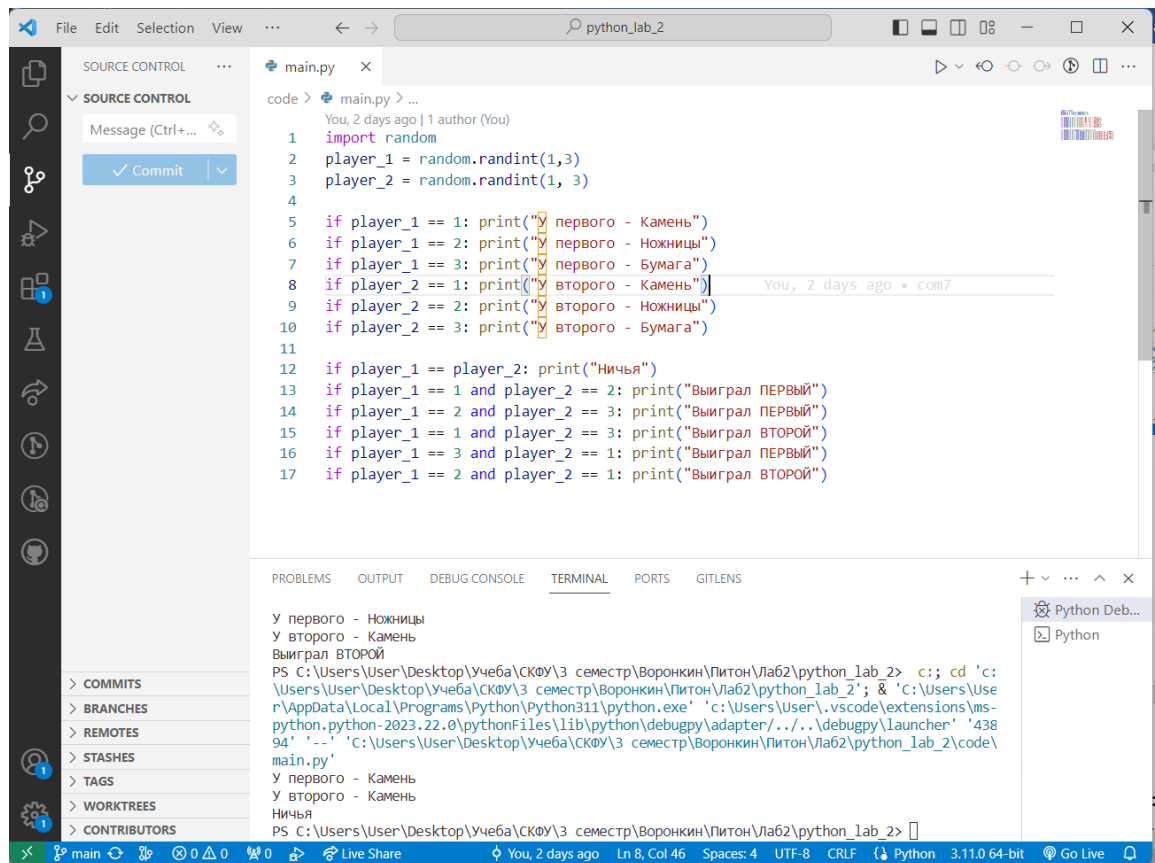


Рисунок 2. Восстановился код после команды git checkout –main.py.

### 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 репозитория данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозитория, вы можете запустить команду `git remote`. Она выведет названия доступных удалённых репозитория. 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 репозитория, вы можете использовать команду `git remote show <remote>`. Она



выдаёт URL удалённого репозитория, а также информацию об отслеживаемых ветках.

#### 11. Каково назначение тэгов Git?

Как и большинство СКВ, Git имеет возможность пометить 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 тега 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 удалённого отслеживания, которые больше не существуют в удалённом репозитории, а из описания, предоставленного `git help push: --prune` используется для удаления ветвей на удалённом репозитории, для которых нет аналога в локальном репозитории.

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