

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:
Магдаев Даламбек Магомедович
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:

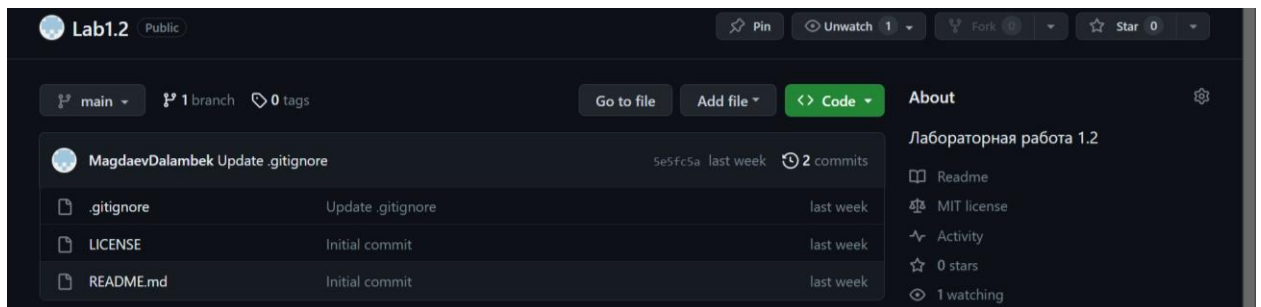


Рисунок 1. Новый репозиторий Lab1.2

2. Проработал примеры лабораторной работы:

```
PS C:\Users\dalam> 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.
PS C:\Users\dalam> |
```

Рисунок 2. Клонирование репозитория

```
PS C:\Users\dalam\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 verison 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
```

Рисунок 3. Результат работы команды git log

```

PS C:\Users\dalam\simplegit-progit> git log -p -2
commit ca82a6dffa17ec66f44342007202690a93763949 (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

```

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

```

PS C:\Users\dalam\simplegit-progit> git log --stat
commit ca82a6dffa17ec66f44342007202690a93763949 (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(+)
PS C:\Users\dalam\simplegit-progit>

```

Рисунок 5. Результат работы команды git log --stat

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

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

```
PS C:\Users\dalam\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
PS C:\Users\dalam\simplegit-progit> |
```

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

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

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

```
PS C:\Users\dalam> 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 | 400.00 KiB/s, done.
Resolving deltas: 100% (837/837), done.
PS C:\Users\dalam> |
```

Рисунок 9. Клонирование репозитория ticgit

```
PS C:\Users\dalam> cd ticgit
PS C:\Users\dalam\ticgit> git remote
origin
PS C:\Users\dalam\ticgit> |
```

Рисунок 10. Результат работы команды git remote

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

Рисунок 11. Результат работы команды git remote -v

```
PS C:\Users\dalam\ticgit> git remote add pb https://github.com/paulboone/ticgit
PS C:\Users\dalam\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)
PS C:\Users\dalam\ticgit> |
```

Рисунок 12. Результат работы команды git remote add pb

```
PS C:\Users\dalam\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 | 36.00 KiB/s, done.
From https://github.com/paulboone/ticgit
* [new branch]      master    -> pb/master
* [new branch]      ticgit    -> pb/ticgit
PS C:\Users\dalam\ticgit> |
```

Рисунок 13. Результат работы команды git fetch pb

```
PS C:\Users\dalam\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)
PS C:\Users\dalam\ticgit>
```

Рисунок 14. Результат работы команды git remote show origin

```
PS C:\Users\dalam\ticgit> git remote rename pb paul
Renaming remote references: 100% (2/2), done.
PS C:\Users\dalam\ticgit> git remote
origin
paul
PS C:\Users\dalam\ticgit> |
```

Рисунок 15. Результат работы команды git remote rename pb paul

```
PS C:\Users\dalam\ticgit> git remote remove paul
PS C:\Users\dalam\ticgit> git remote
origin
PS C:\Users\dalam\ticgit>
```

Рисунок 16. Результат работы команды git remote remove paul

```
PS C:\Users\dalam\ticgit> git tag
PS C:\Users\dalam\ticgit> git tag -l "v1.8.5*"
PS C:\Users\dalam\ticgit> git tag -a v1.4 -m "my version 1.4"
PS C:\Users\dalam\ticgit> git tag
v1.4
PS C:\Users\dalam\ticgit> |
```

Рисунок 17. Создание аннотированного тега

```
PS C:\Users\dalam\ticgit> git show v1.4
tag v1.4
Tagger: MagdaevDalambek <dalambekmagdaev11125@gmail.com>
Date: Thu Oct 19 16:19:08 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 TicG
it 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
PS C:\Users\dalam\ticgit> |
```

Рисунок 18. Результат работы команды git show v1.4

```
PS C:\Users\dalam\ticgit> git tag -d v1.4
Deleted tag 'v1.4' (was 9f18875)
PS C:\Users\dalam\ticgit> |
```

Рисунок 19. Удаление ранее созданного тега

3. Клонировал ранее созданные репозиторий:

```
PS C:\Users\dalam> git clone https://github.com/MagdaevDalambek/Lab1.2.git
Cloning into 'Lab1.2'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
PS C:\Users\dalam> |
```

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:

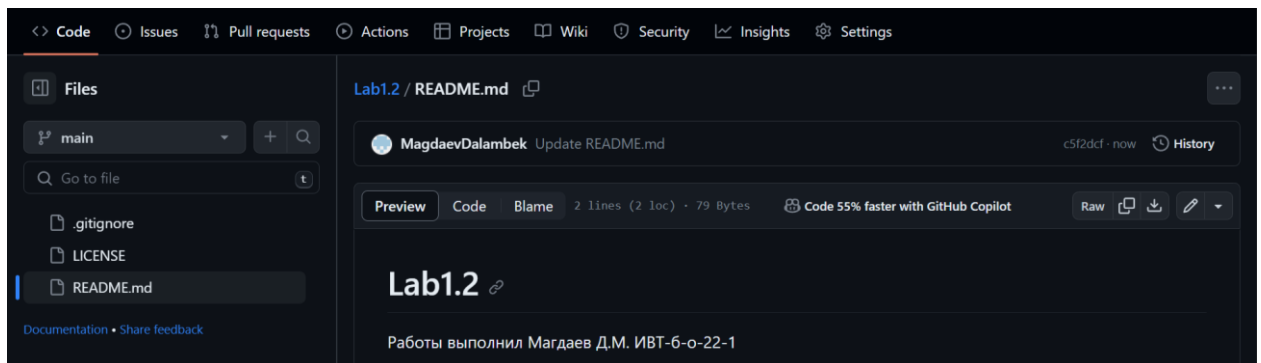


Рисунок 22. Добавление информации в файл README.md

6. Написал небольшую программу на языке Python:

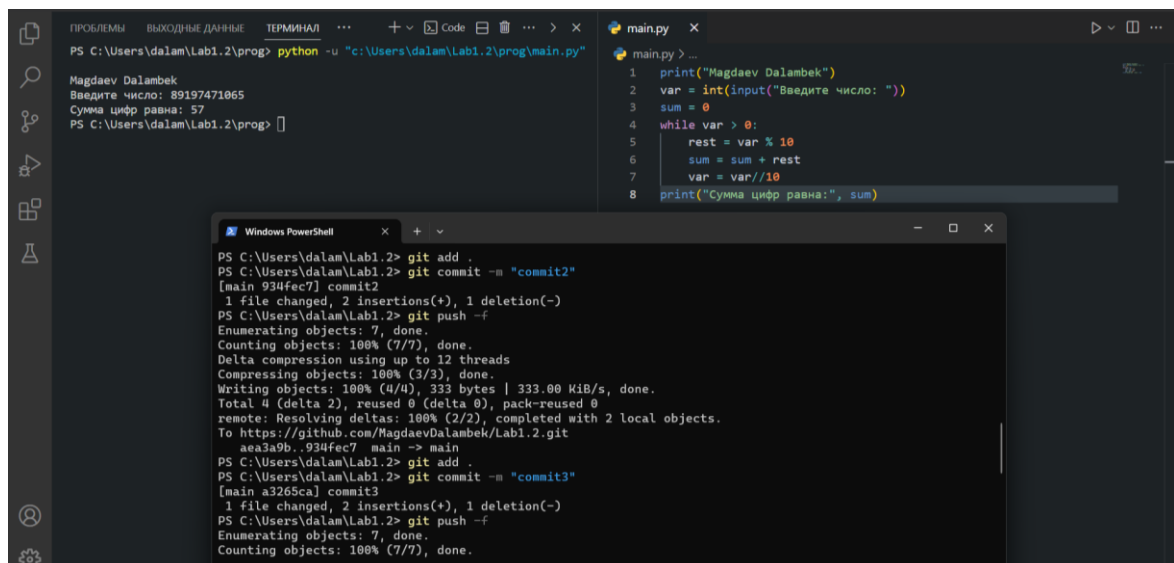


Рисунок 23. Первые 4 коммита и код программы

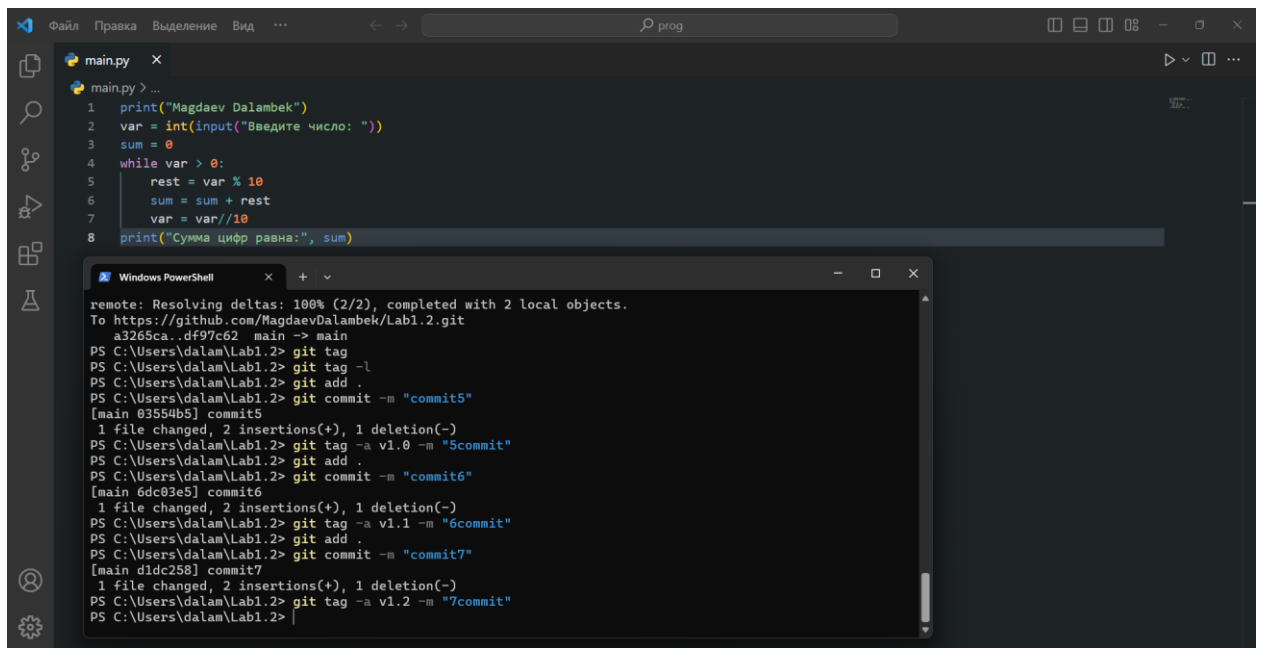


Рисунок 24. Последние 3 коммита с тегами

7. Просмотрел историю хранилища:

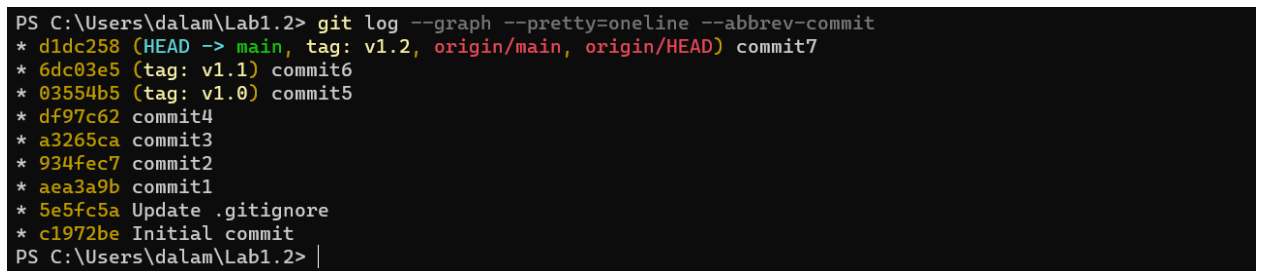


Рисунок 25. Результат работы команды `git log --graph --pretty=oneline --abbrev-commit`

8. Посмотрел содержимое коммитов:


```

PS C:\Users\dalam\Lab1.2> git show HEAD
commit d1dc25801e73ea6ca872c3f57674f2610afff6c7 (HEAD -> main, tag: v1.2, origin/main, origin/HEAD)
Author: MagdaevDalambek <dalambekmagdaev11125@gmail.com>
Date: Mon Nov 27 20:05:38 2023 +0300

    commit7

diff --git a/prog/main.py b/prog/main.py
index e31f30c..718c1d7 100644
--- a/prog/main.py
+++ b/prog/main.py
@@ -4,4 +4,5 @@ sum = 0
     while var > 0:
         rest = var % 10
         sum = sum + rest
-        var = var//10
\ No newline at end of file
+        var = var//10
+print("Сумма цифр равна:", sum)
\ No newline at end of file
PS C:\Users\dalam\Lab1.2> git show HEAD~1
commit 6dc03e58322037214f84bdf3345ff114831f01c0 (tag: v1.1)
Author: MagdaevDalambek <dalambekmagdaev11125@gmail.com>
Date: Mon Nov 27 20:04:27 2023 +0300

    commit6

diff --git a/prog/main.py b/prog/main.py
index 919e4d3..e31f30c 100644
--- a/prog/main.py
+++ b/prog/main.py
@@ -3,4 +3,5 @@ var = int(input("Введите число: "))
sum = 0
while var > 0:
    rest = var % 10
-    sum = sum + rest
\ No newline at end of file
+    sum = sum + rest
+    var = var//10

```

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

```

PS C:\Users\dalam\Lab1.2> git show a3265ca
commit a3265ca8dccfec3d0c9572e8e265d1544964d21e
Author: MagdaevDalambek <dalambekmagdaev11125@gmail.com>
Date: Mon Nov 27 12:32:13 2023 +0300

    commit3

diff --git a/prog/main.py b/prog/main.py
index 7ecb41f..0ee0cc1 100644
--- a/prog/main.py
+++ b/prog/main.py
@@ -1,3 +1,4 @@
print("Magdaev Dalambek")
var = int(input("Введите число: "))
- sum = 0
\ No newline at end of file
+ sum = 0
+ while var > 0:
\ No newline at end of file
PS C:\Users\dalam\Lab1.2> |

```

Рисунок 27. Результат работы команды git show a3265ca

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

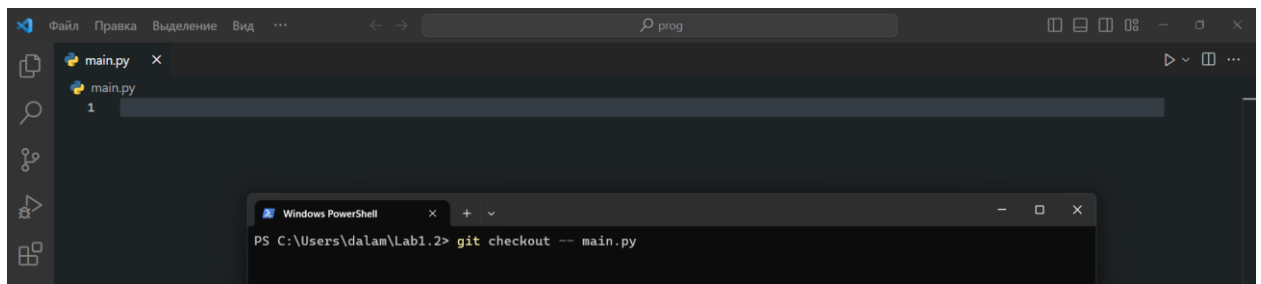


Рисунок 28. Пустой main.py

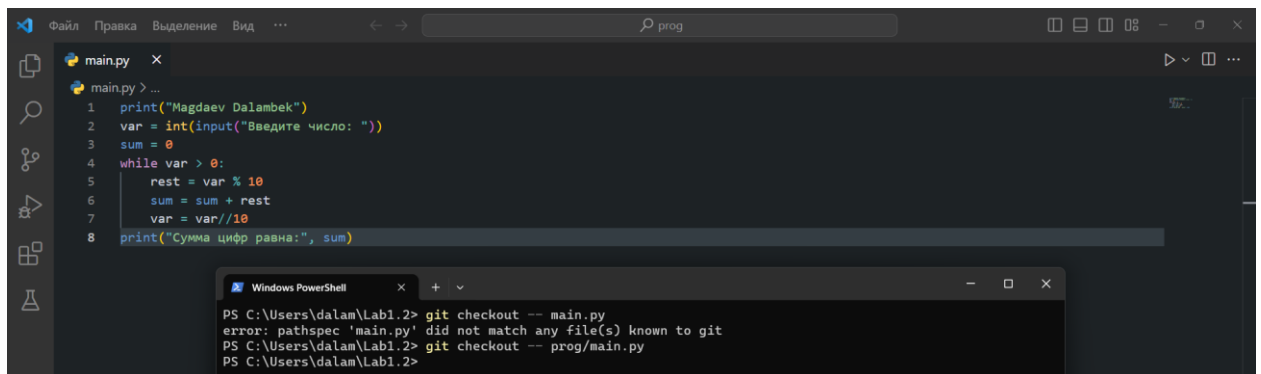


Рисунок 29. Код вернулся после команды git checkout -- prog/main.py

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

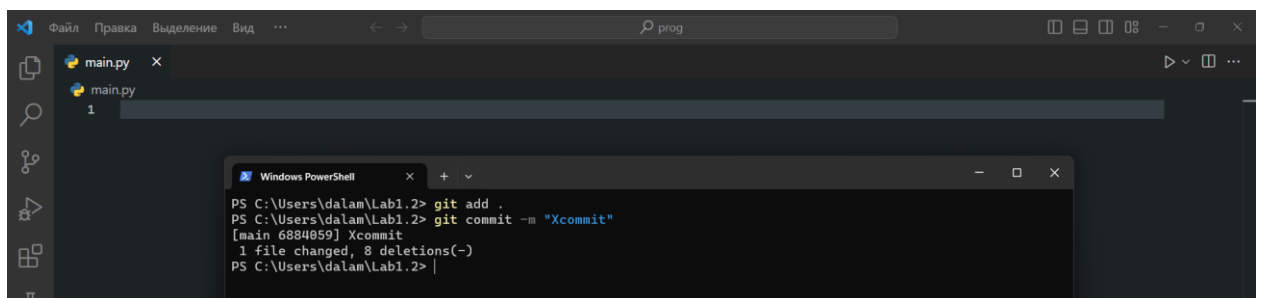


Рисунок 30. Коммит после удаления кода

11. Откатил состояние хранилище к предыдущей версии коммита:

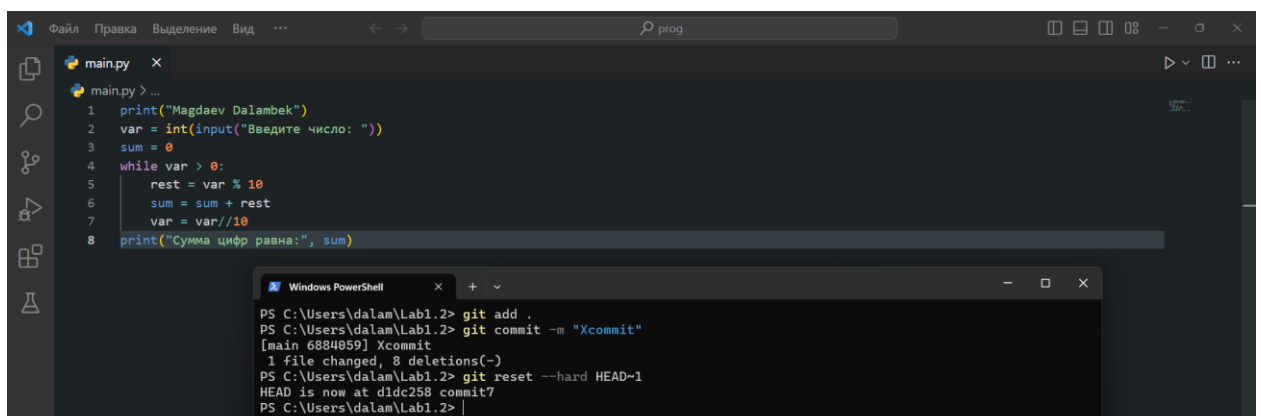


Рисунок 31. Код вернулся после команды git reset --hard HEAD~1

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 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 репозиториях данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозиториях, вы можете запустить команду `git remote`. Она выведет названия доступных удалённых репозиториях. 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 репозиториях, 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 - а 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 были исследованы возможности Git для работы с локальными репозиториями.