

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ  
ЛАБОРАТОРНОЙ РАБОТЕ №2**

Дисциплина: «Программирование на Python» Тема: «Исследование возможностей  
Git для работы с локальными репозиториями»

Выполнил:  
Кенесбаев Хилол Куат улы  
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)

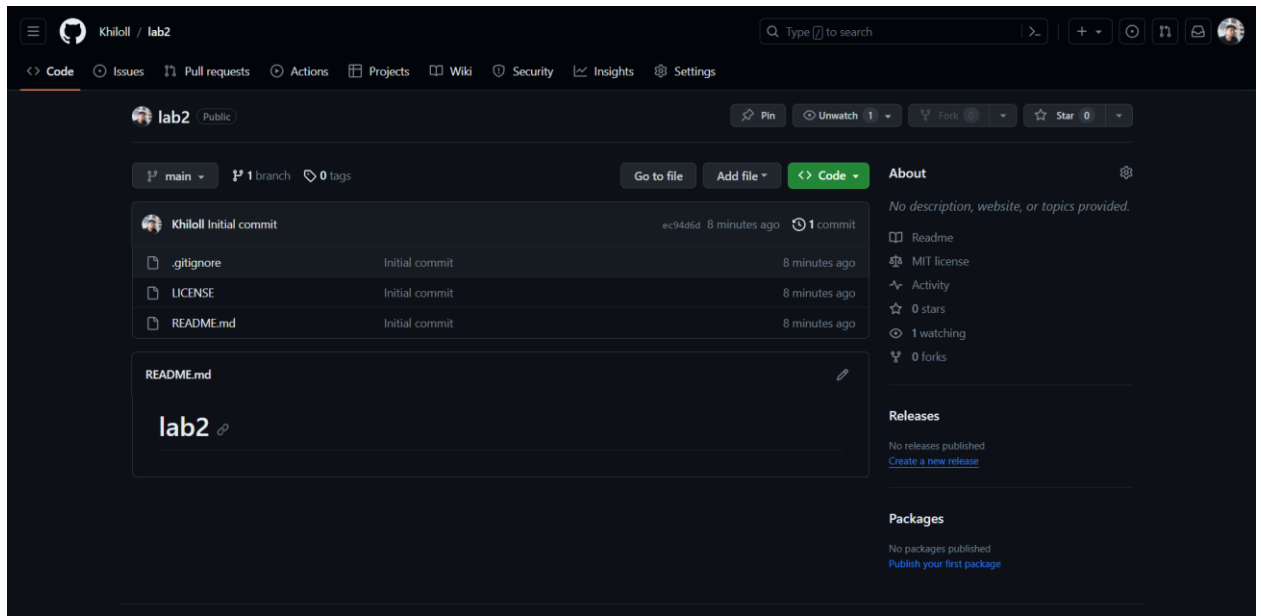
Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

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

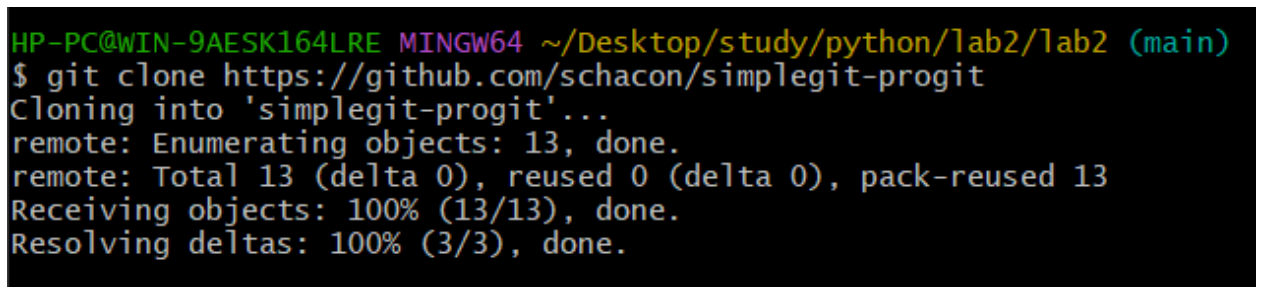
## Порядок выполнения работы:

### 1. Создал новый репозиторий



*Рисунок 1 Новый репозиторий lab2*

### 2. Проработал примеры лабораторной работы



*Рисунок 2 Клонирование репозитория*

```

(master)
$ 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*

```

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

```

*Рисунок 4 Результат работы команды 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

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(+)

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-pr
ogit (master)

```

*Рисунок 5 Результат работы команды git log --stat*

```

$ git log --pretty=oneline
ca82a6dff817ec66f44342007202690a93763949 (HEAD -> master, origin/master, or
igin/HEAD) changed the verison number
085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7 removed unnecessary test code
a11bef06a3f659402fe7563abf99ad00de2209e6 first commit

```

*Рисунок 6 Результат работы команды git log --pretty = one line*

```

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-pr
ogit (master)
$ 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 7 Результат работы команды git log --pretty=format:"%h - %an, %ar : %s"*

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit (master)
$ git log --pretty=format:"%h %s" -graph
fatal: ambiguous argument '-graph': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...']
```

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

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit (master)
$ 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 | 795.00 KiB/s, done.
Resolving deltas: 100% (837/837), done.
```

*Рисунок 9 Клонирование репозитория `ticgit`*

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/ticgit (master)
$ git remote
origin
```

*Рисунок 10 Результат работы команды `git remote`*

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/ticgit (master)
$ git remote -v
origin https://github.com/schacon/ticgit (fetch)
origin https://github.com/schacon/ticgit (push)
```

*Рисунок 11 Результат работы команды `git remote -v`*

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/ticgit (master)
$ git remote add pb https://github.com/paulboone/ticgit

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/ticgit (master)
$ 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 12 Результат работы команды `git remote add pb https://github.com/paulboone/ticgit`*

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ 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 | 62.00 KiB/s, done.
From https://github.com/paulboone/ticgit
* [new branch]      master    -> pb/master
* [new branch]      ticgit    -> pb/ticgit
```

*Рисунок 13 Результат работы команды git fetch pb*

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ 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*

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ git remote rename pb paul
Renaming remote references: 100% (2/2), done.

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ git remote
origin
paul
```

*Рисунок 15 Результат работы команды git remote rename pb paul*

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ git remote remove paul

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ git remote
origin
```

*Рисунок 16 Результат работы команды git remote remove paul*

```

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ git tag

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ git tag -l "v1.8.5*"

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ git tag -a v1.4 -m "my version 1,4"

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ git tag
v1.4

```

*Рисунок 17 Создал аннотированный тег*

```

tag v1.4
Tagger: Hilo1 <khilol2017@gmail.com>
Date: Thu Nov 2 18:29:40 2023 +0300

my version 1,4

commit 847256809a3d518cd36b8f81859401416fe8d945 (HEAD -> master, tag: v1.4, orig
in/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 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

```

*Рисунок 18 Результат работы команды git show v1.4*

```

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/simplegit-progit/
ticgit (master)
$ git tag -d v1.4
Deleted tag 'v1.4' (was 6a2bdcf)

```

*Рисунок 19 Удаление ранее созданного тега*



### 3. Клонировал ранее созданный репозиторий

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2
$ git clone https://github.com/Khilo11/lab2.git
Cloning into 'lab2'...
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 lab2

### 4. Дополнил .gitignore

```
#.idea/
**/.DS_Store
.vscode
```

Рисунок 21 Добавленные строки .gitignore

### 5. Написал небольшую программу на python

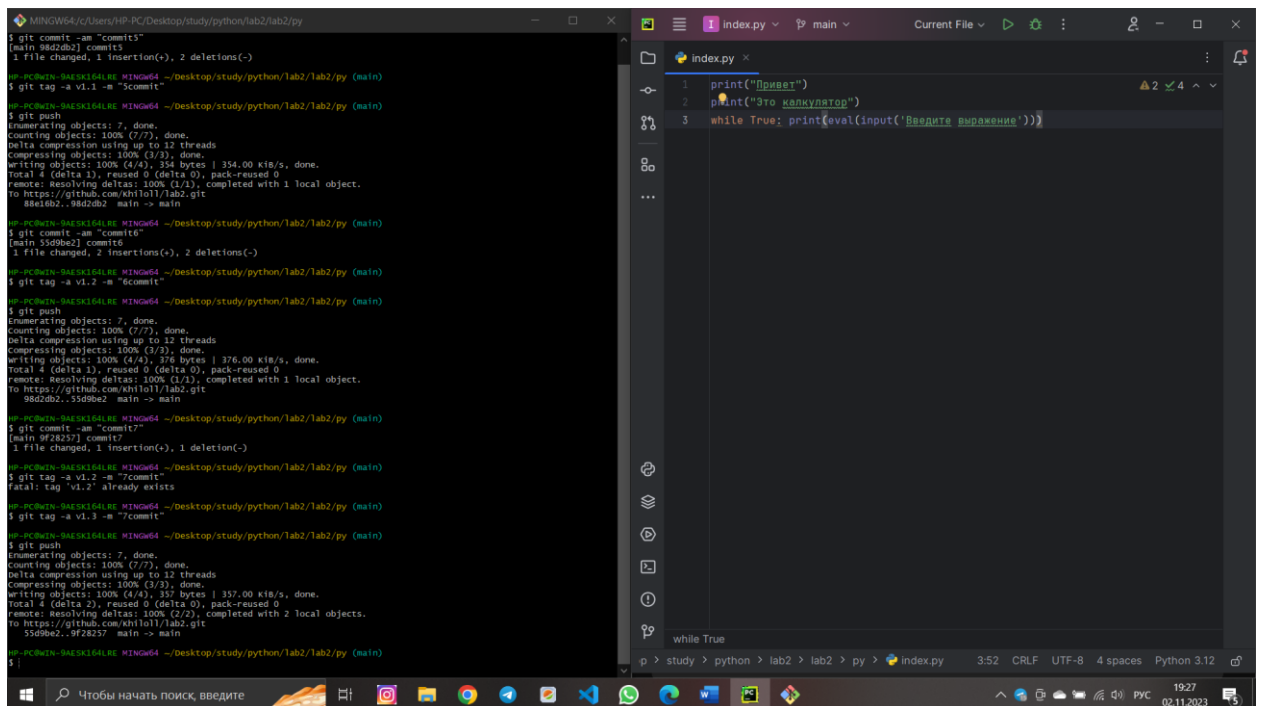
The screenshot shows a Windows terminal window on the left and a VS Code editor on the right. The terminal displays the execution of git commands to clone a repository, add files, commit, and push. The VS Code editor shows a Python file named index.py with a simple calculator program. The terminal output shows the first four commits, each adding a new file and committing it. The VS Code editor shows the code for the calculator program, which prints 'hi', 'it's', and 'calculator', and then enters a loop where it evaluates and prints the result of a user input.

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2
$ git add --all
$ git commit -m "commit2"
$ git push
$ git add py
$ git commit -m "commit3"
$ git push
$ git add py
$ git commit -m "commit4"
$ git push
```

```
1 print("hi")
2 print("it's")
3 print("calculator")
4 while True: print(eval(input('>>>')))
```

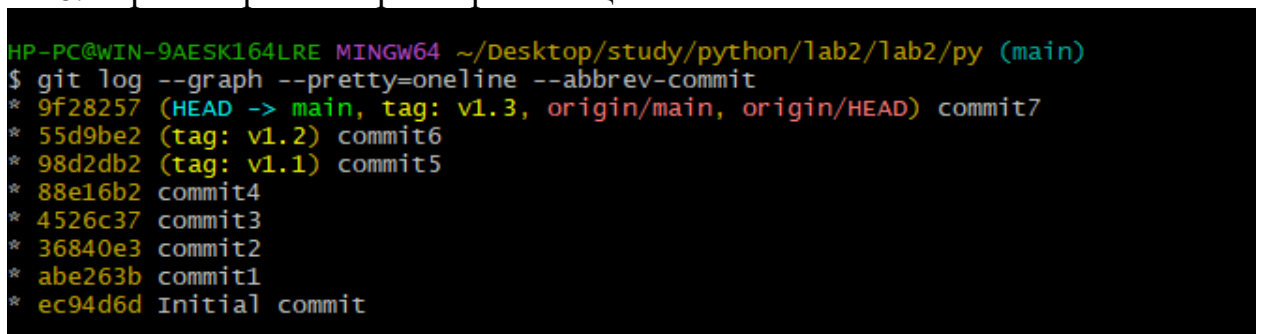
Рисунок 22 Первые 4 коммита и программа ru





*Рисунок 23 Последние 3 коммита с тегами*

## 6. Просмотрел историю хранилища



*Рисунок 24 Результат работы команды git log --graph --pretty=oneline --abbrev-commit*

## 7. Просмотрел содержимое коммитов

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/py (main)
$ git show HEAD
commit 4a94d0e1b6e45ae056480270df15478cb8485720 (HEAD -> main, origin/main, origin/HEAD)
Author: Hilo1 <khilo12017@gmail.com>
Date: Thu Nov 2 19:33:58 2023 +0300

    commit8

diff --git a/py/index.py b/py/index.py
index d0c79ad..eeff95b 100644
--- a/py/index.py
+++ b/py/index.py
@@ -1,3 +1,4 @@
 print("привет")
 print("Это калькулятор")
-while True: print(eval(input('Введите выражение'))))
\ No newline at end of file
+while True: print(eval(input('Введите выражение'))))
+print("Работу выполнил Кенесбаев Х.К")
\ No newline at end of file

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/py (main)
$ git show HEAD~1
commit 9f282575ed8abb8d45fbc22ab92bfc1aa39e0781 (tag: v1.3)
Author: Hilo1 <khilo12017@gmail.com>
Date: Thu Nov 2 19:26:25 2023 +0300

    commit7

diff --git a/py/index.py b/py/index.py
index 5dac776..d0c79ad 100644
--- a/py/index.py
+++ b/py/index.py
@@ -1,3 +1,3 @@
 print("привет")
 print("Это калькулятор")
-while True: print(eval(input('>>>')))
\ No newline at end of file
+while True: print(eval(input('Введите выражение'))))
\ No newline at end of file

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/py (main)
$ |
```

Рисунок 25 Результат работы команд `git show HEAD` и `git show HEAD~1`

8. Удалил код из файла `program.py`, а затем удалил все несохраненные изменения

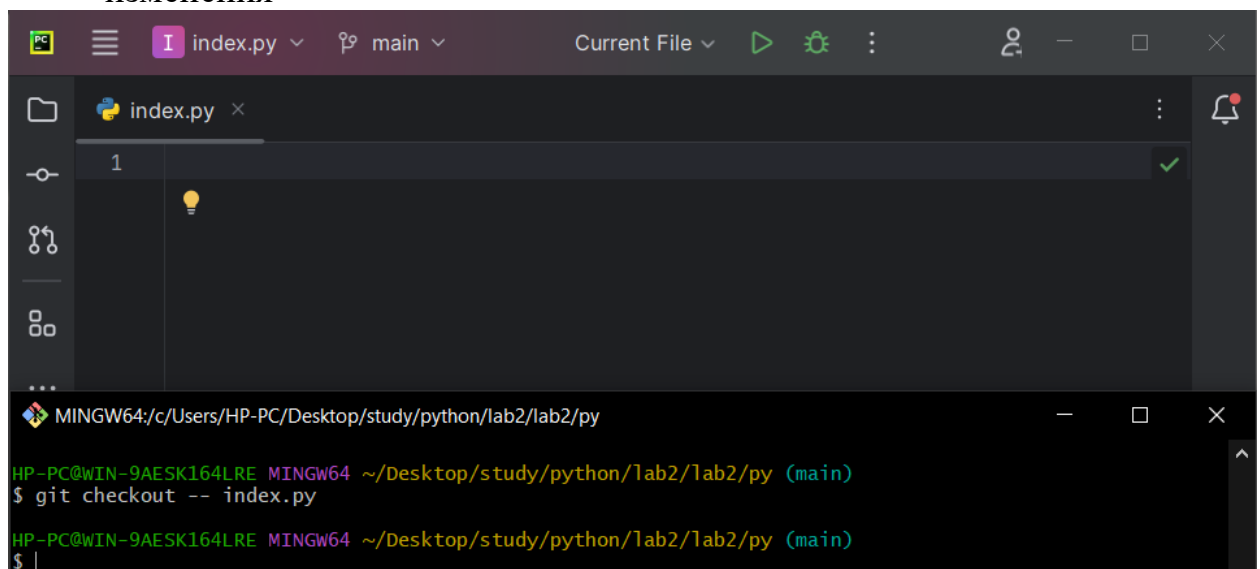


Рисунок 26 Пустой `index.py`

```
index.py x
1 print("Привет")
2 print("Это калькулятор")
3 while True: print(eval(input('Введите выражение'))))
4 print("Работу выполнил Кенесбаев Х.К")

MINGW64:/c/Users/HP-PC/Desktop/study/python/lab2/lab2/py
HP-PC@WIN-9AESH164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/py (main)
$ git checkout -- index.py
```

*Рисунок 27 Код вернулся после команды git checkout -- index.py*

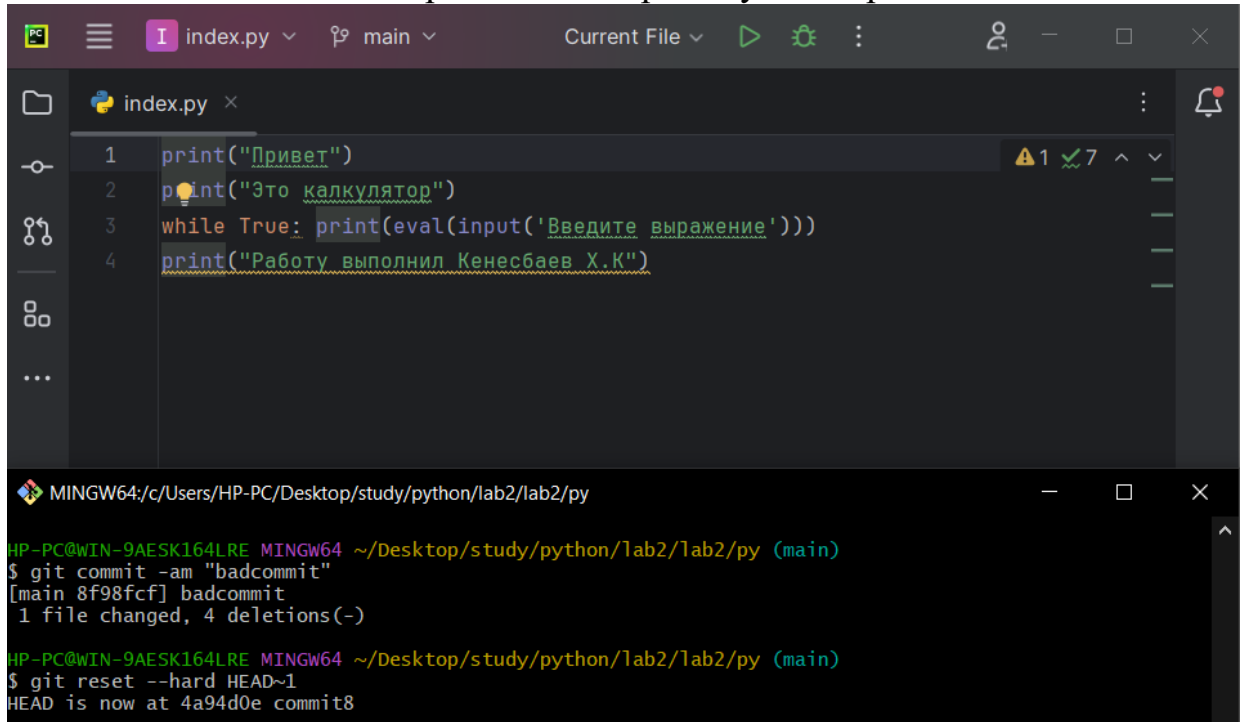
9. Удалил весь код из файла program.py, а затем сделал коммит

```
index.py x
1

MINGW64:/c/Users/HP-PC/Desktop/study/python/lab2/lab2/py
HP-PC@WIN-9AESH164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/py (main)
$ git commit -am "badcommit"
[main 8f98fcf] badcommit
1 file changed, 4 deletions(-)
HP-PC@WIN-9AESH164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/py (main)
$ |
```

*Рисунок 28 Коммит после удаления кода*

## 10. Откатил состояние хранилища к предыдущей версии коммита



The screenshot shows a code editor window with a Python file named `index.py`. The code contains four lines: `print("Привет")`, `print("Это калькулятор")`, `while True: print(eval(input('Введите выражение')))`, and `print("Работу выполнил Кенесбаев Х.К")`. Below the editor is a terminal window showing the execution of `git commit -am "badcommit"` and `git reset --hard HEAD~1`. The terminal output indicates that the reset was successful, returning the HEAD to the previous commit (4a94d0e).

```
HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/py (main)
$ git commit -am "badcommit"
[main 8f98fcf] badcommit
1 file changed, 4 deletions(-)

HP-PC@WIN-9AESK164LRE MINGW64 ~/Desktop/study/python/lab2/lab2/py (main)
$ git reset --hard HEAD~1
HEAD is now at 4a94d0e commit8
```

*Рисунок 29 Код вернулся после команды `git reset --hard HEAD~1`*

**Вывод:** чтобы удалить не сохраненные коммитом изменения, можно выполнить команду `git checkout --`, 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 `-n`, 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 страничный 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 ...` для исключения из индекса.

5. Как отменить изменения в файле? Использовать `git checkout --` 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, 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` .

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` .

10. Как выполнить просмотр удаленного репозитория? Если хотите получить побольше информации об одном из удалённых репозиториях, вы можете использовать команду `git remote show` . Она выдаёт URL удалённого репозитория, а также 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` . Для удаления тега в локальном репозитории достаточно выполнить команду `git tag -d` . 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