Version Control and collaboration with Git and Github

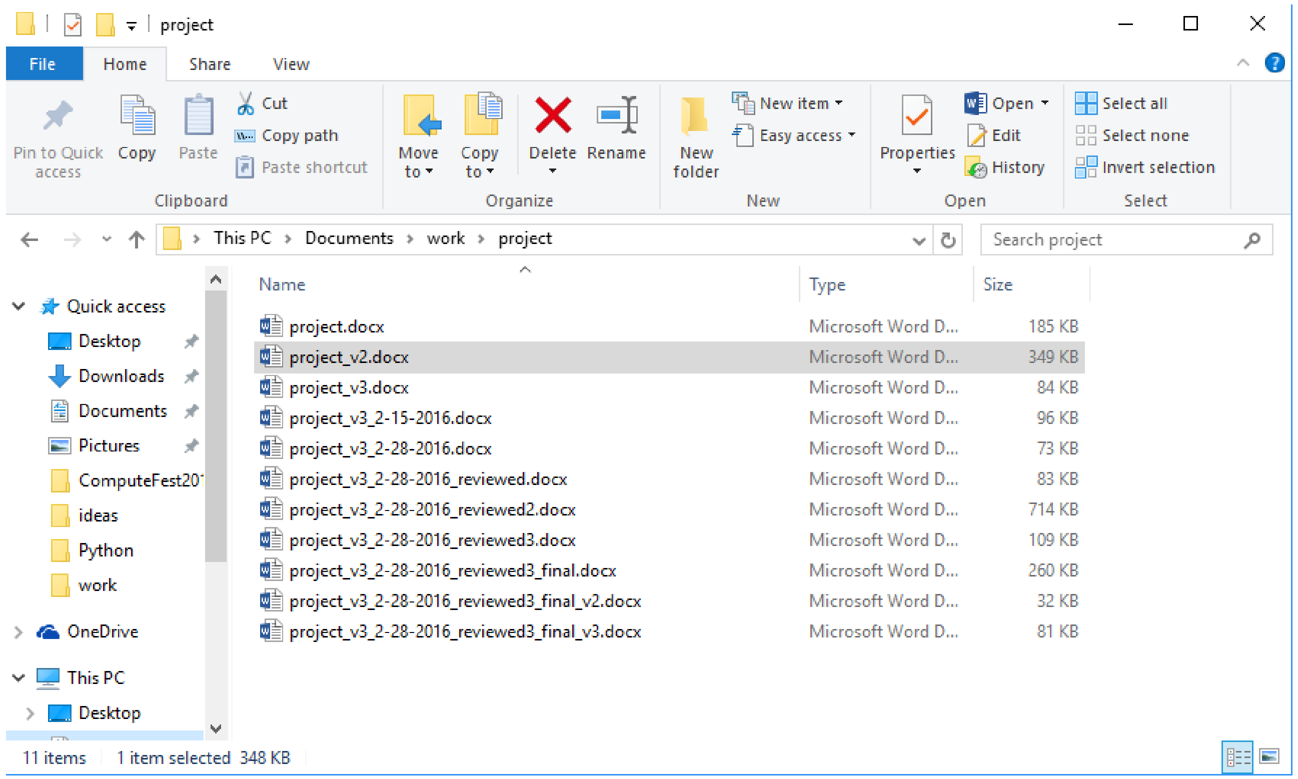
Katia Oleinik

Research Computing Services

BOSTON UNIVERSITY

Challenges of working on a project

- Undo and Redo
- Tracking changes
- Working with others
- Sharing Changes
- Overlapping work by various people



Name	Type	Size
project.docx	Microsoft Word D...	185 KB
project_v2.docx	Microsoft Word D...	349 KB
project_v3.docx	Microsoft Word D...	84 KB
project_v3_2-15-2016.docx	Microsoft Word D...	96 KB
project_v3_2-28-2016.docx	Microsoft Word D...	73 KB
project_v3_2-28-2016_reviewed.docx	Microsoft Word D...	83 KB
project_v3_2-28-2016_reviewed2.docx	Microsoft Word D...	714 KB
project_v3_2-28-2016_reviewed3.docx	Microsoft Word D...	109 KB
project_v3_2-28-2016_reviewed3_final.docx	Microsoft Word D...	260 KB
project_v3_2-28-2016_reviewed3_final_v2.docx	Microsoft Word D...	32 KB
project_v3_2-28-2016_reviewed3_final_v3.docx	Microsoft Word D...	81 KB

11 items 1 item selected 348 KB

Git history

Development began in 2005 while working on Linux Kernel
The first stable version released in December 2005

Goals set but Linus Torvalds:

- ✓ Distributed system
- ✓ Applying updates should not take longer than 3 seconds
- ✓ Take Concurrent Version System as an example of what ***not*** to do
- ✓ Support distributed system workflow
- ✓ Include strong safeguards against corruption, both accidental and malicious

Word "git" - "*unpleasant person*" in British slang

The man page describes Git as "the stupid content tracker".

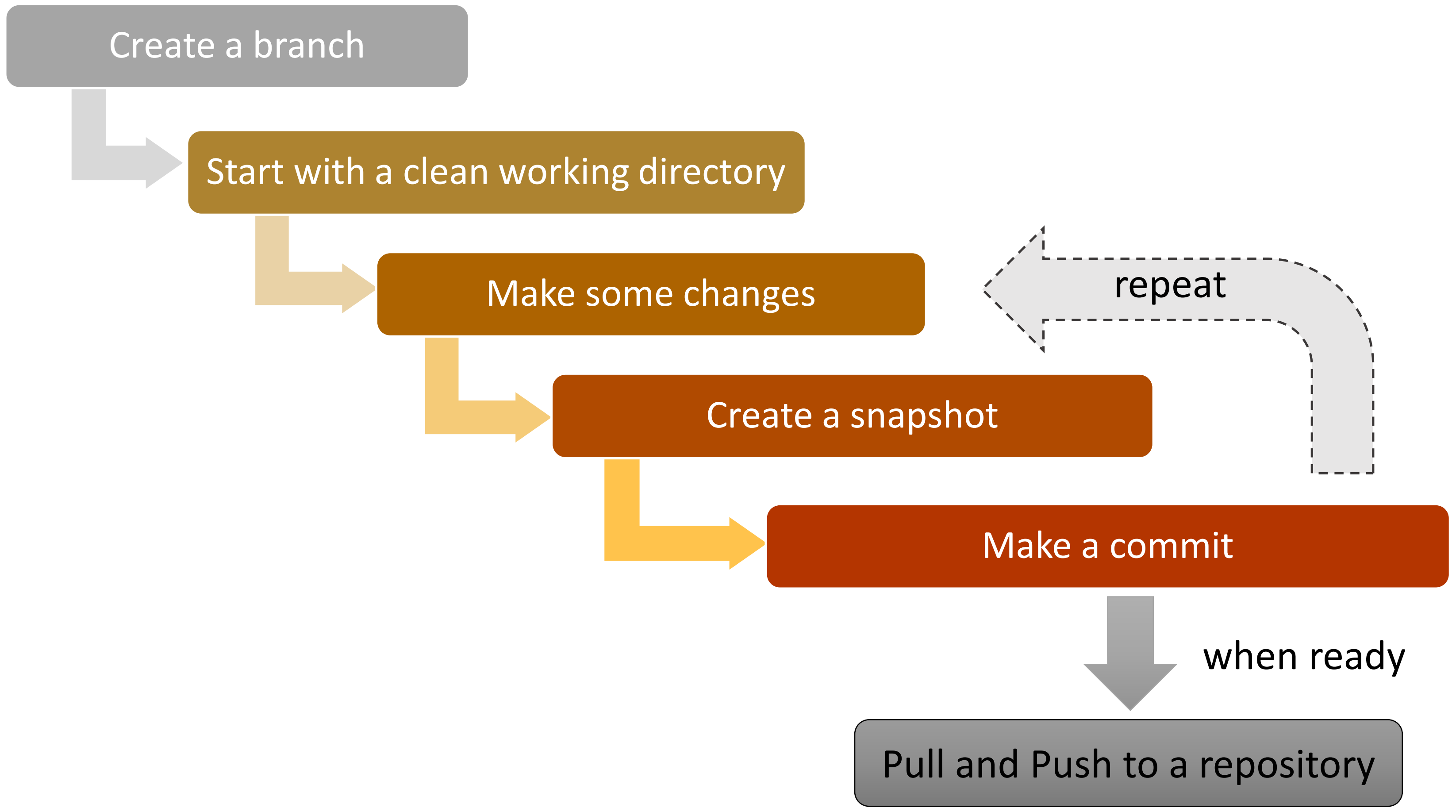
From README file of the source code:

```
"- global information tracker": you're in a good mood,
and it actually works for you. Angels sing, and a light
suddenly fills the room.
- "g*dd*mn idiotic truckload of sh*t": when it breaks
```

Git main features

- ✓ Track all your changes
- ✓ Work along with others
- ✓ Share work with others

Git Workflow



Git Terminology

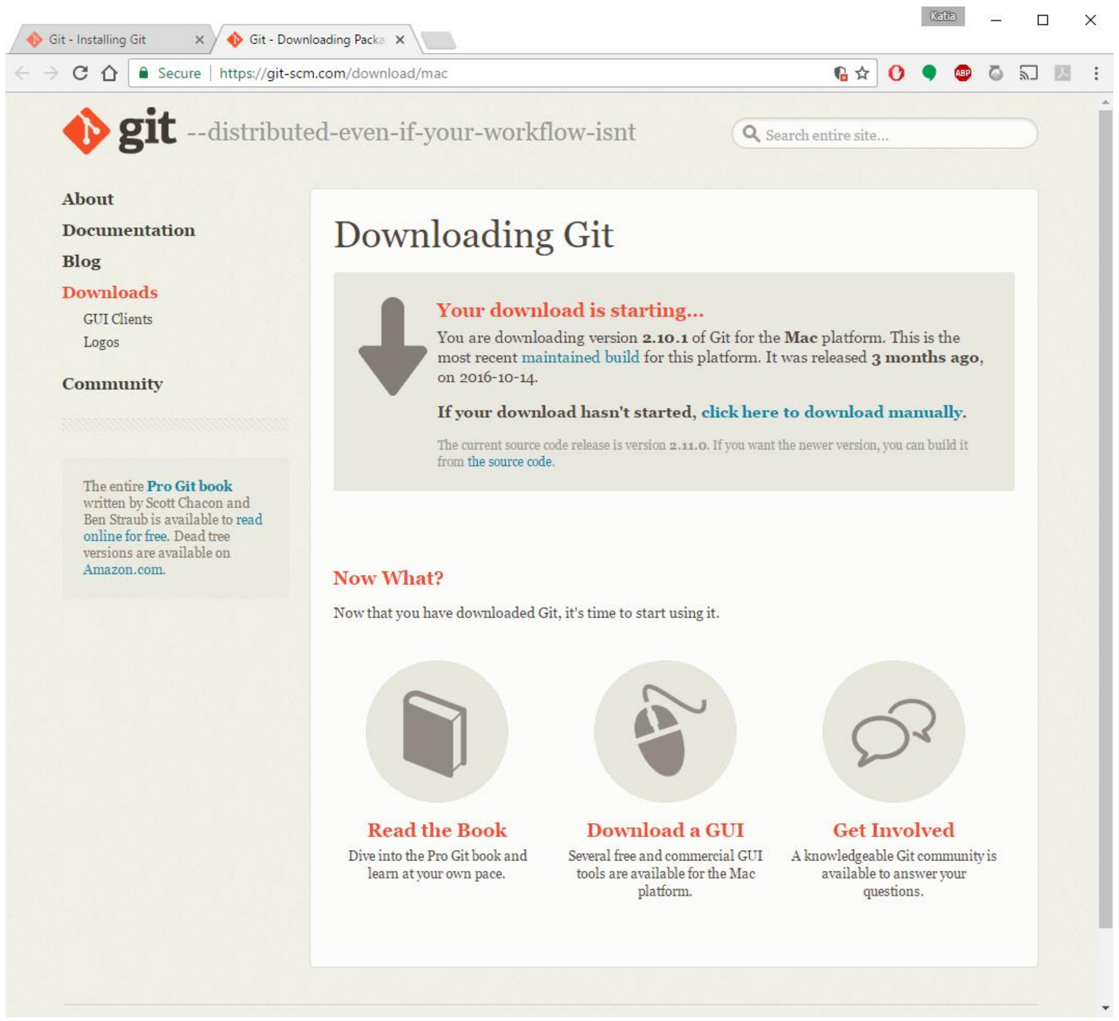
Repository - container for snapshots and history
Remote - connection to another repository for example GitHub (like URL)
Commit -

- A snapshot, basic unit of history
- Full copy of a project
- Includes author, time, comments, pointer to the parent

Reference - a pointer to commit
Branch - a separate line of workflow
Merge - a commit that combines 2 lines of history (points to 2 parents)

BOSTON UNIVERSITY

Installing Git



Git - Installing Git | Git - Downloading Packa

Secure | https://git-scm.com/download/mac

git --distributed-even-if-your-workflow-isnt

Search entire site...

About
Documentation
Blog
Downloads
GUI Clients
Logos
Community

The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Downloading Git

Your download is starting...

You are downloading version **2.10.1** of Git for the **Mac** platform. This is the most recent maintained build for this platform. It was released **3 months ago**, on 2016-10-14.

If your download hasn't started, click here to download manually.

The current source code release is version 2.11.0. If you want the newer version, you can build it from the source code.

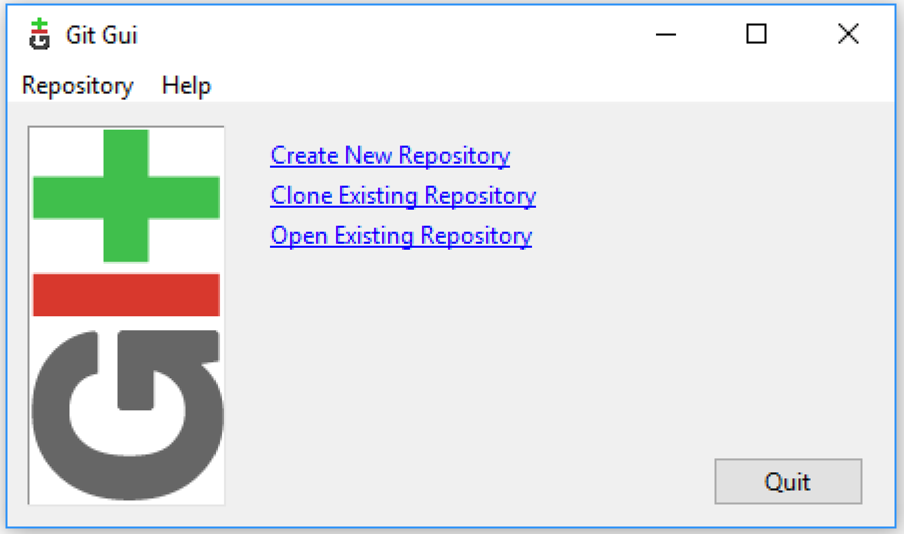
Now What?

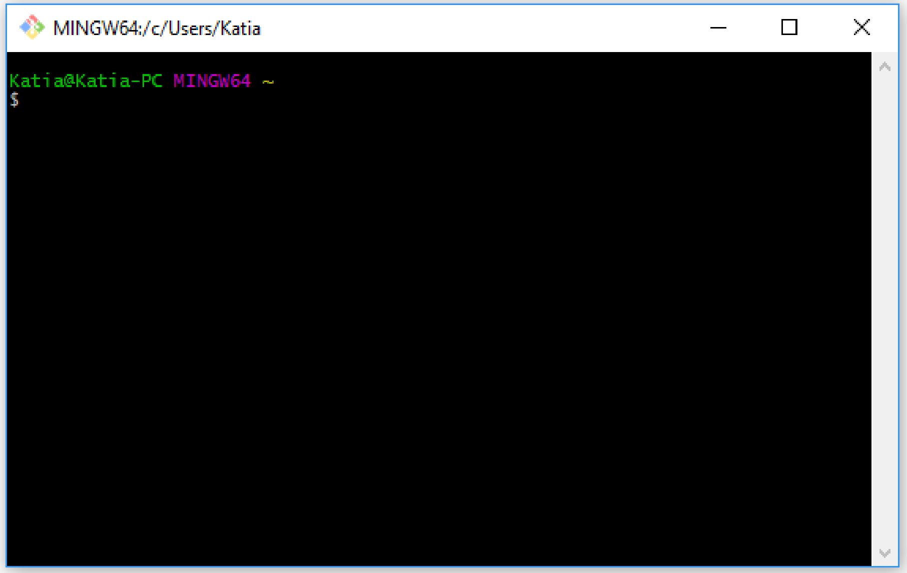
Now that you have downloaded Git, it's time to start using it.

Read the Book
Dive into the Pro Git book and learn at your own pace.

Download a GUI
Several free and commercial GUI tools are available for the Mac platform.

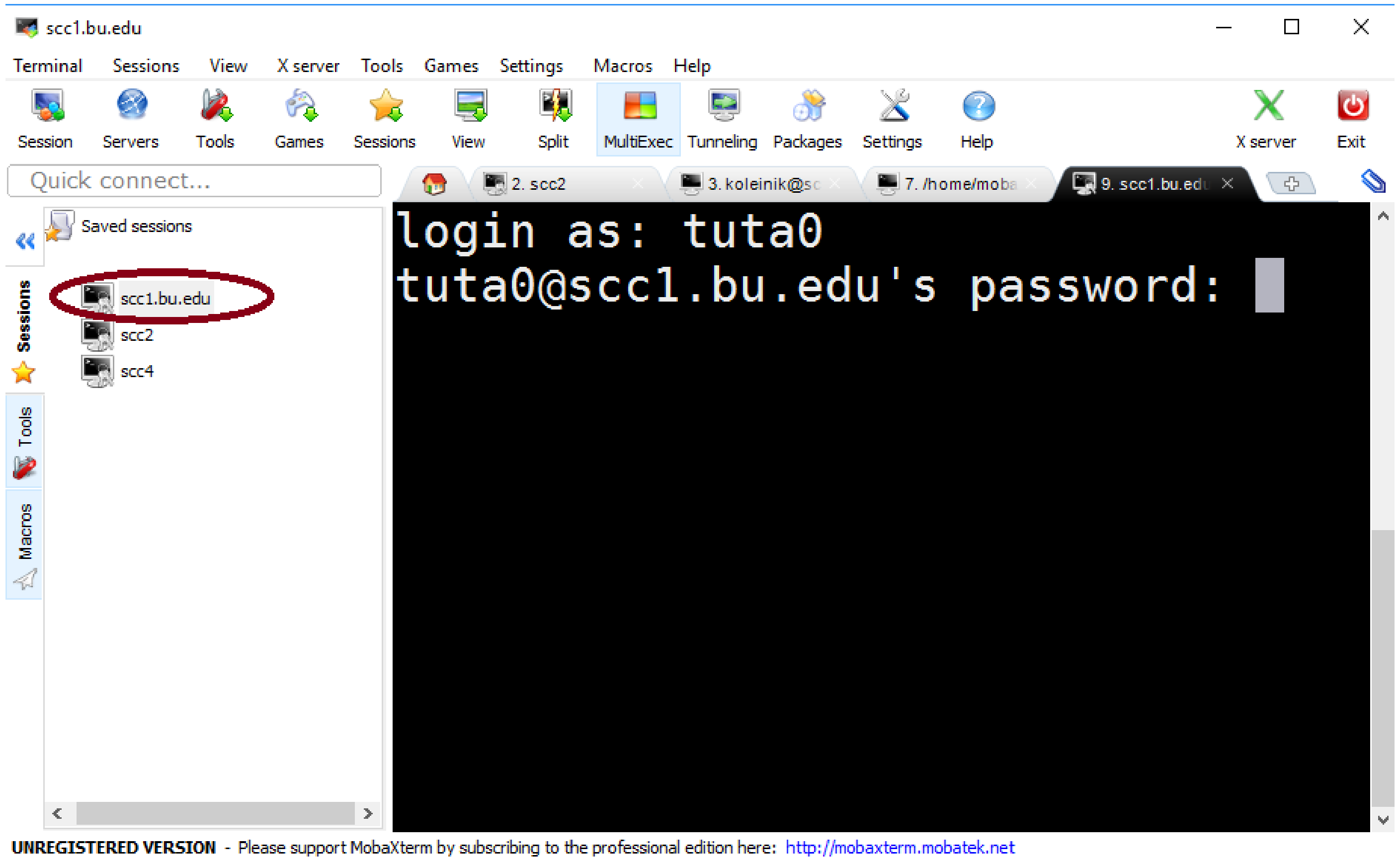
Get Involved
A knowledgeable Git community is available to answer your questions.





```
Katia@Katia-PC MINGW64 ~
$
```

Login to the SCC


scc1.bu.edu
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
2. scc2
3. koleinik@sc
7. /home/moba
9. scc1.bu.edu
Saved sessions
scc1.bu.edu
scc2
scc4
Sessions
Tools
Macros

```
login as: tuta0
tuta0@scc1.bu.edu's password:
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net

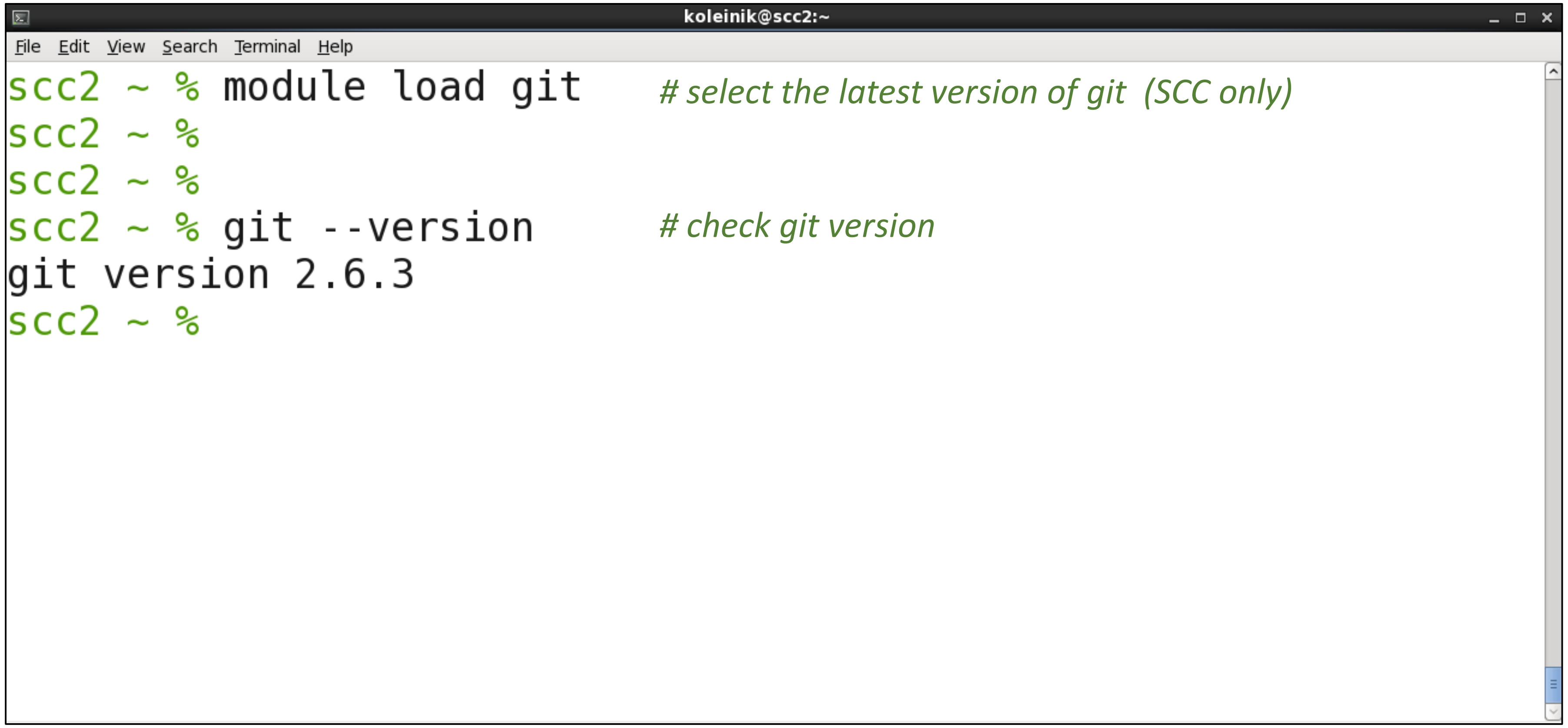
Username: tuta#
Password:

# - is the number located on your computer

Note:

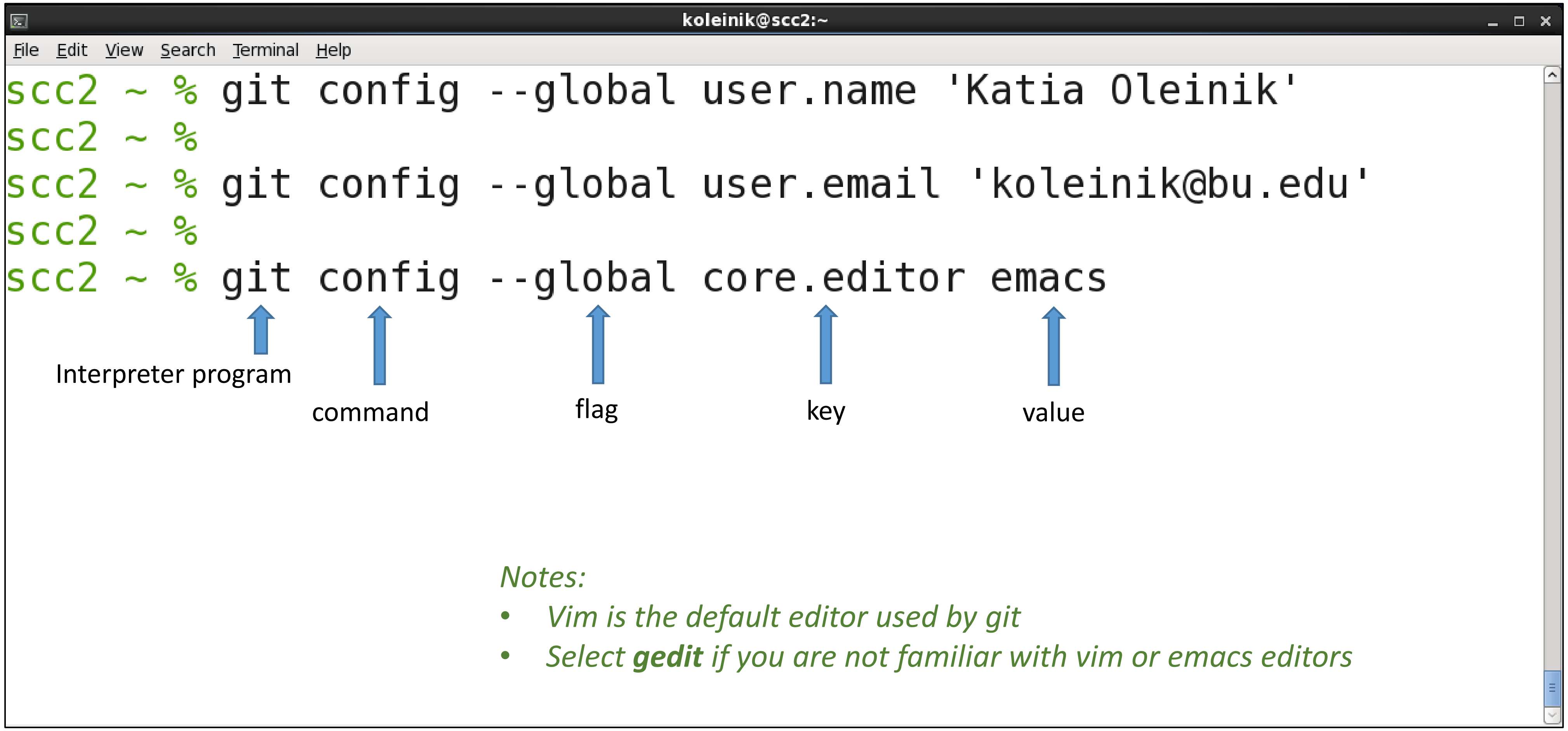
- Username and password are case-sensitive
- password will not be displayed while you are typing it

Git : basic configuration


koleinik@scc2:~

```
scc2 ~ % module load git       # select the latest version of git  (SCC only)
scc2 ~ %
scc2 ~ %
scc2 ~ % git --version         # check git version
git version 2.6.3
scc2 ~ %
```

Git : basic configuration

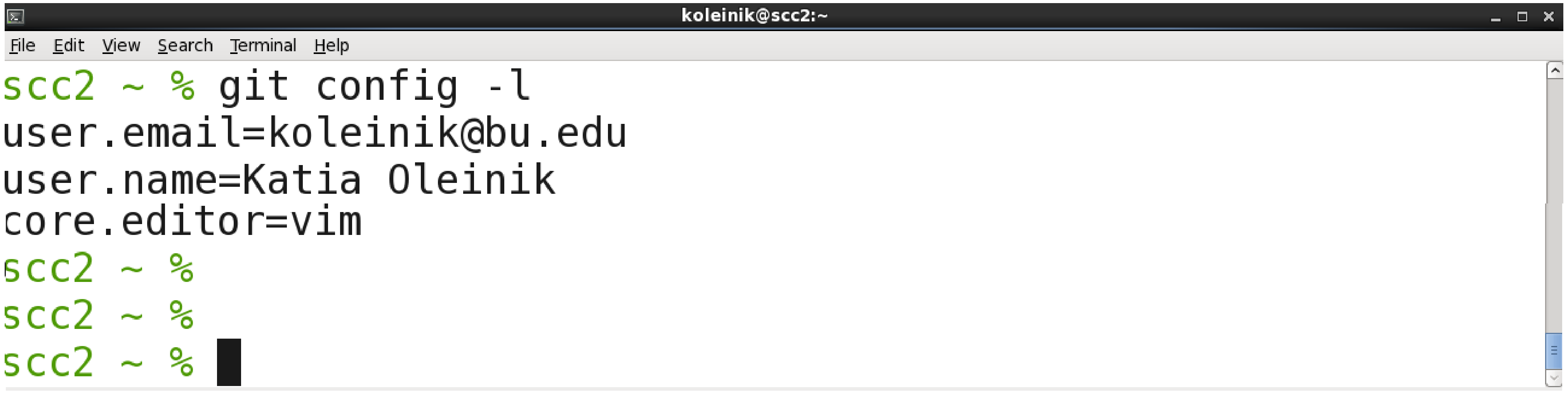


```
scc2 ~ % git config --global user.name 'Katia Oleinik'
scc2 ~ %
scc2 ~ % git config --global user.email 'koleinik@bu.edu'
scc2 ~ %
scc2 ~ % git config --global core.editor emacs
```

Notes:

- *Vim is the default editor used by git*
- *Select **gedit** if you are not familiar with vim or emacs editors*

Git : basic configuration



```
scc2 ~ % git config -l
user.email=koleinik@bu.edu
user.name=Katia Oleinik
core.editor=vim
scc2 ~ %
scc2 ~ %
scc2 ~ %
```

Git : advanced configuration

System

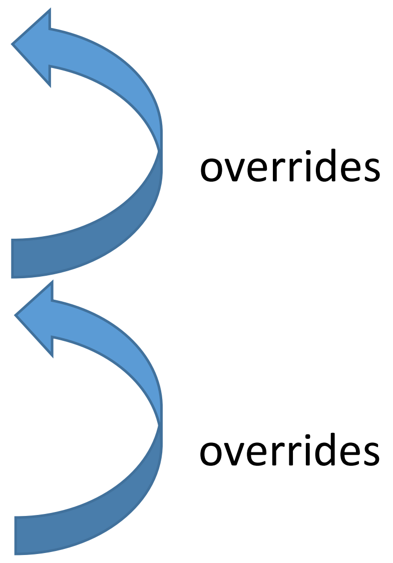
- Usually in /etc directory

Global

- ~/.gitconfig

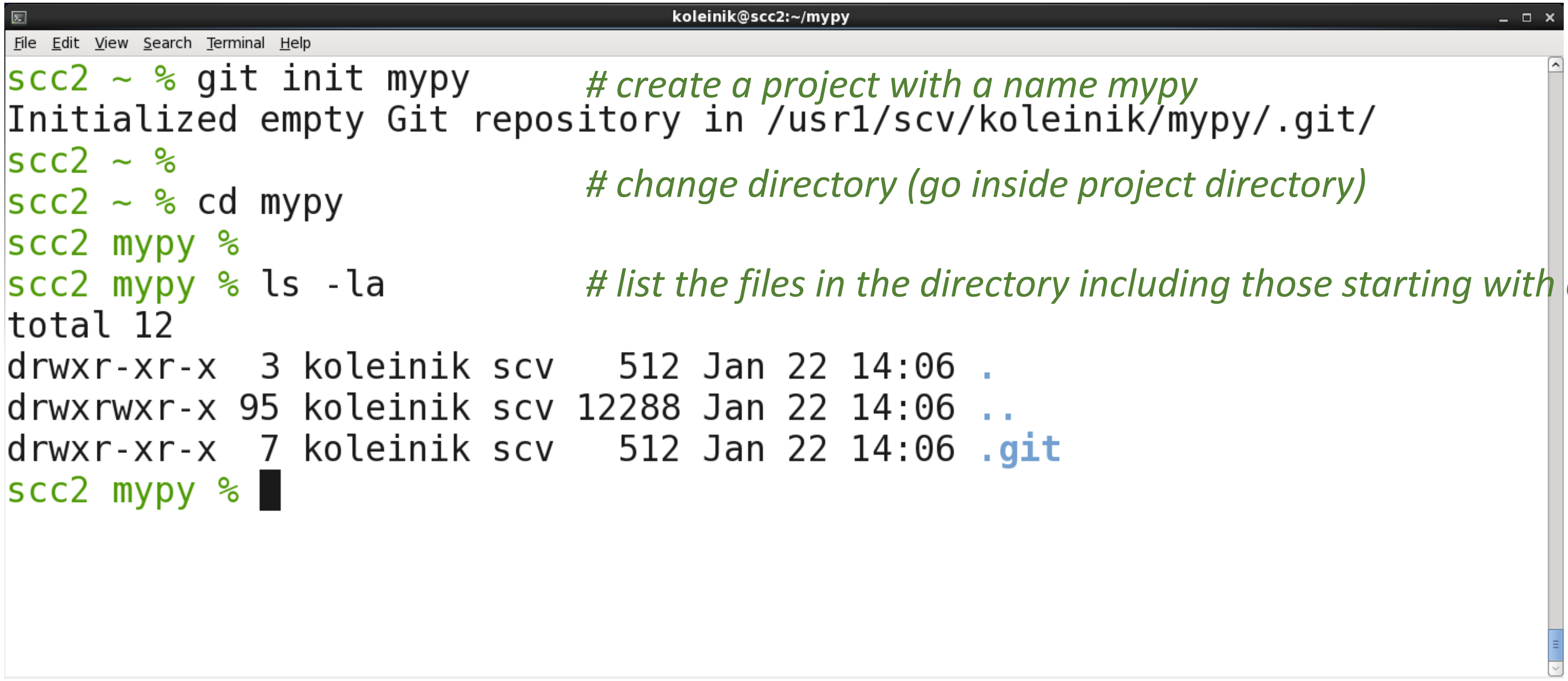
Local

- .git/config



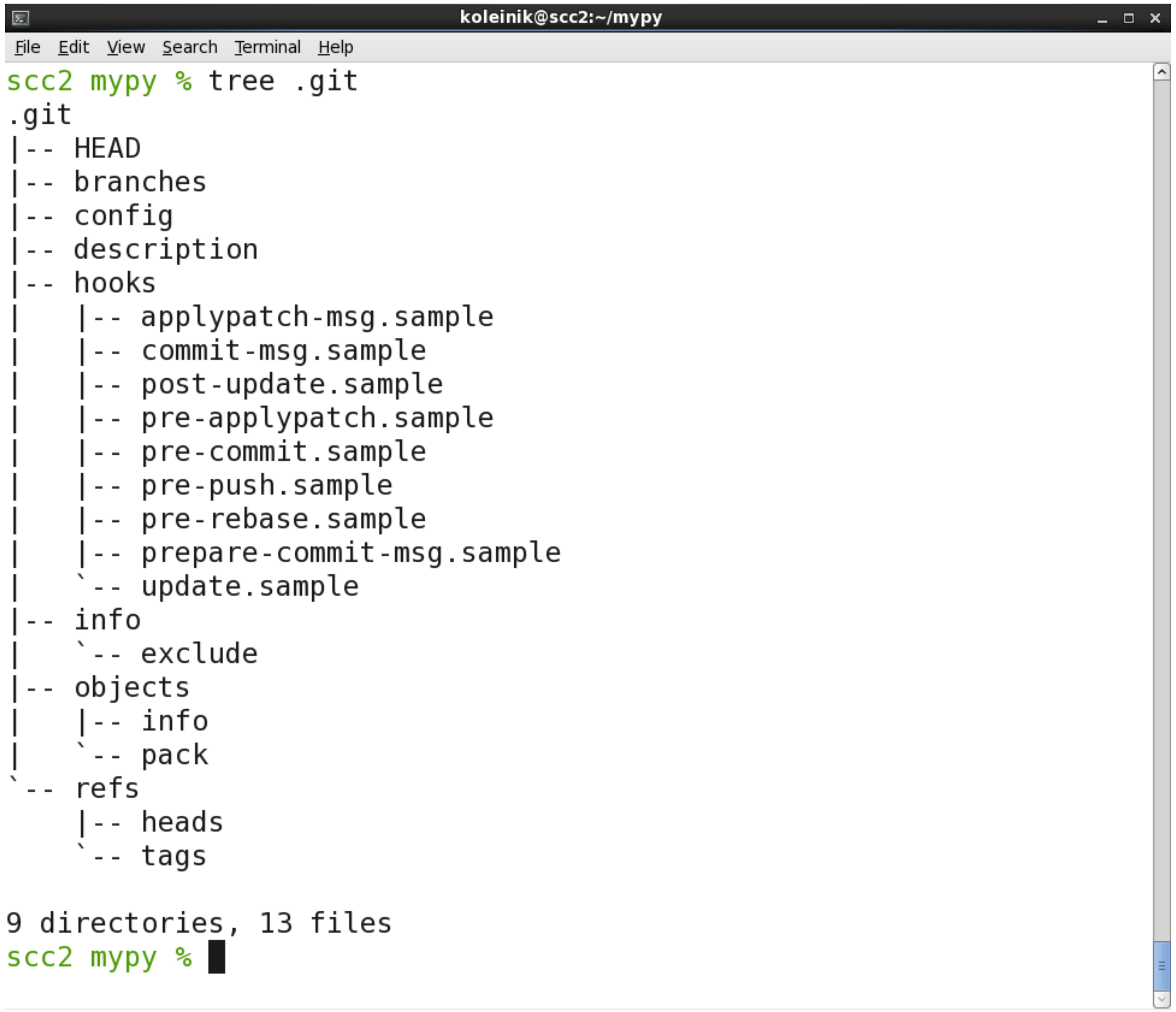
BOSTON UNIVERSITY

Git : create a repository



```
scc2 ~ % git init mypy          # create a project with a name mypy
Initialized empty Git repository in /usr1/scv/koleinik/mypy/.git/
scc2 ~ %
scc2 ~ % cd mypy                # change directory (go inside project directory)
scc2 mypy %
scc2 mypy % ls -la              # list the files in the directory including those starting with a dot
total 12
drwxr-xr-x  3 koleinik scv   512 Jan 22 14:06 .
drwxrwxr-x 95 koleinik scv 12288 Jan 22 14:06 ..
drwxr-xr-x  7 koleinik scv   512 Jan 22 14:06 .git
scc2 mypy %
```

Git : explore a repository



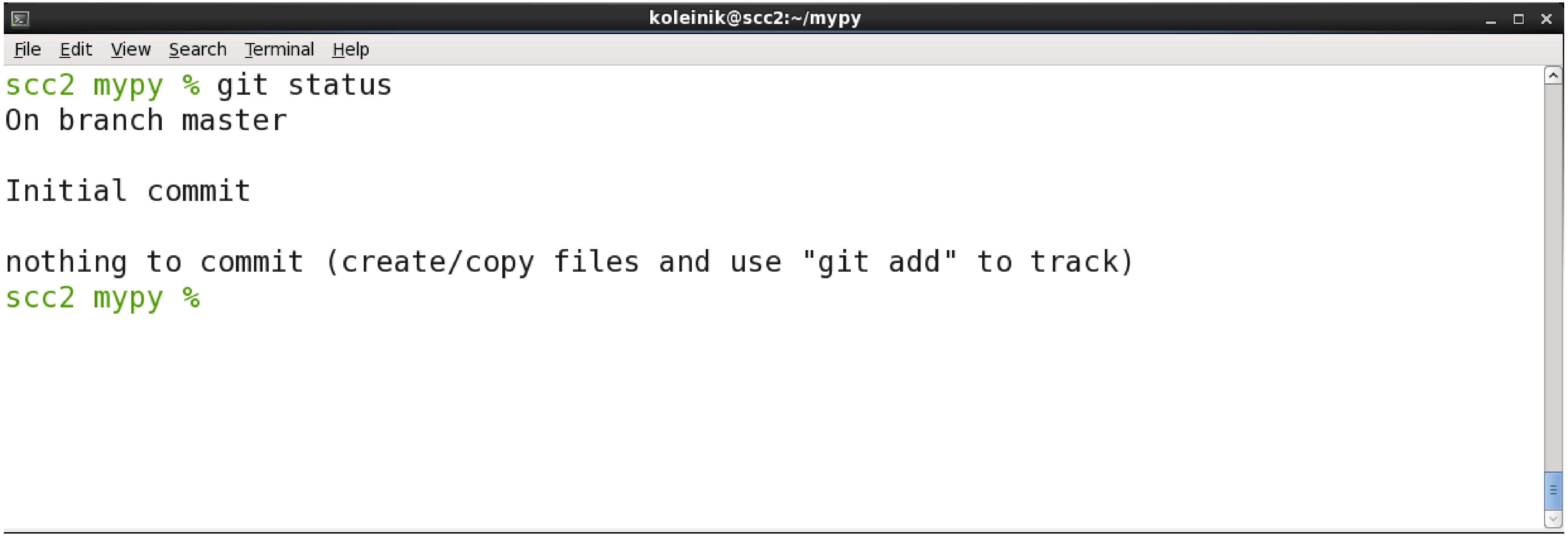
```
scc2 mypy % tree .git
.git
|-- HEAD
|-- branches
|-- config
|-- description
|-- hooks
|   |-- applypatch-msg.sample
|   |-- commit-msg.sample
|   |-- post-update.sample
|   |-- pre-applypatch.sample
|   |-- pre-commit.sample
|   |-- pre-push.sample
|   |-- pre-rebase.sample
|   |-- prepare-commit-msg.sample
|   `-- update.sample
|-- info
|   `-- exclude
|-- objects
|   |-- info
|   `-- pack
`-- refs
    |-- heads
    `-- tags

9 directories, 13 files
scc2 mypy % 
```

Git : 4 statuses

untracked	• File is not under control by git
unmodified	• Git knows about file, but it has not been modified
modified	• Git knows about the file and it has been modified
Staged	• File is ready to commit

BOSTON UNIVERSITY

Git : check the status

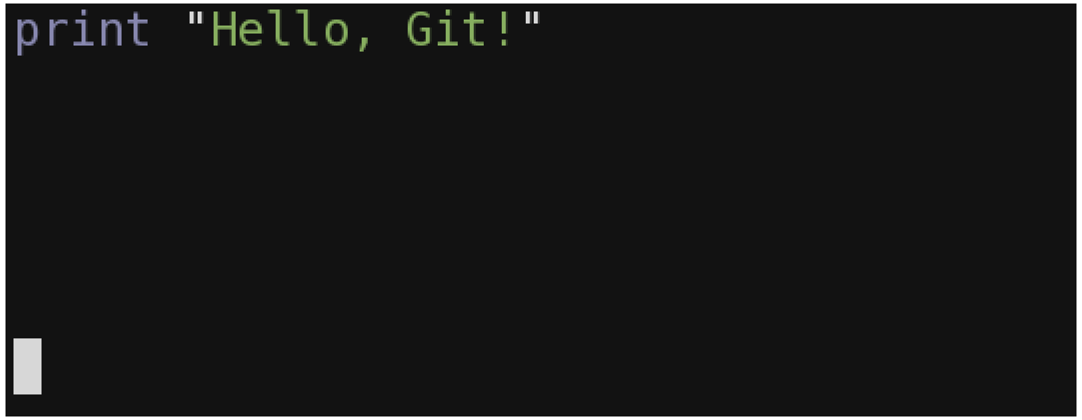


```
scc2 mypy % git status
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
scc2 mypy %
```

BOSTON UNIVERSITY

Create a new file

Using your favorite editor, open a file hello.py and enter the following content:



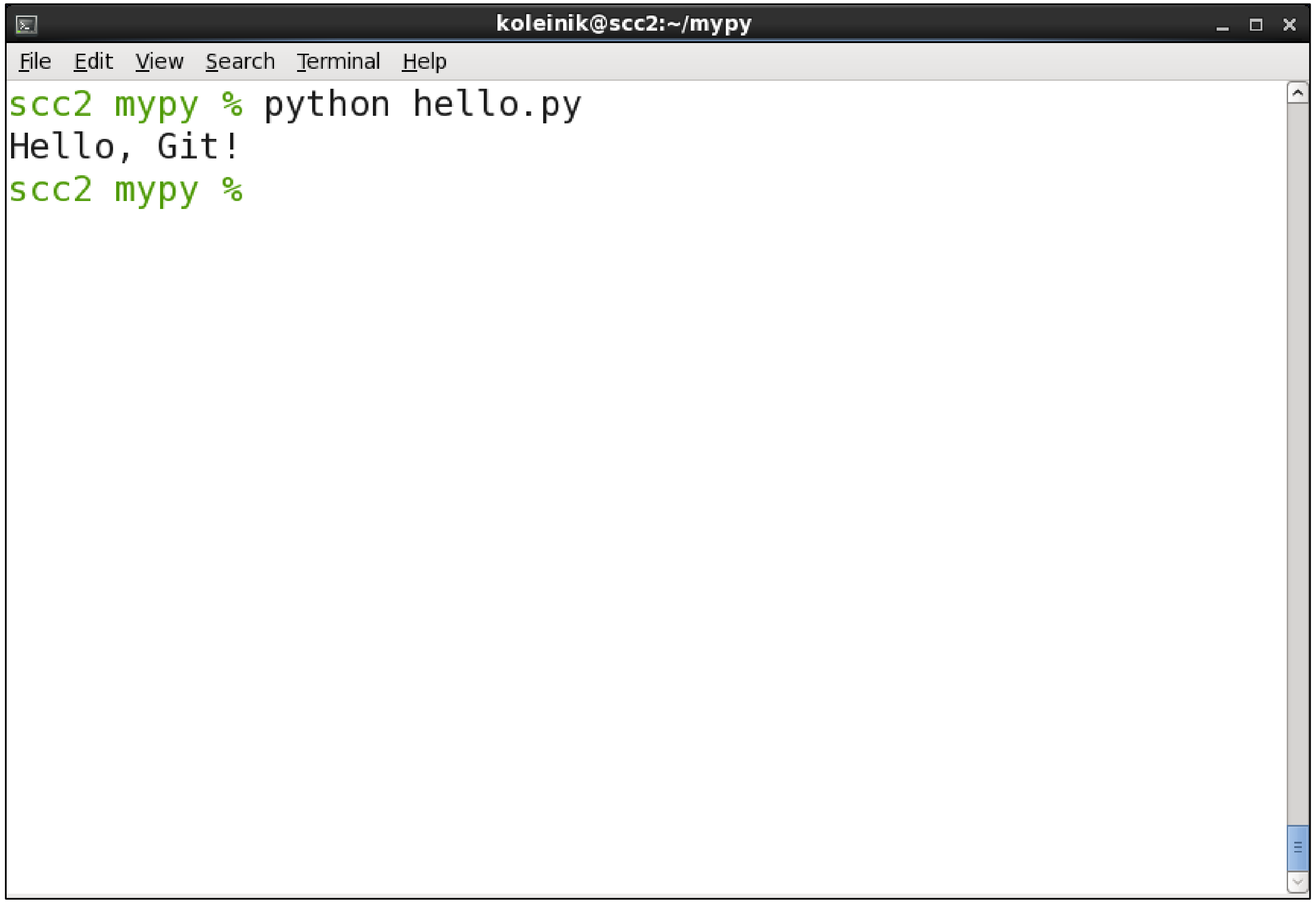
```
print "Hello, Git!"
```

Save the file with the name hello.py and exit.

Note: if you are not familiar with vim or emacs editors, used ***gedit*** to edit files:

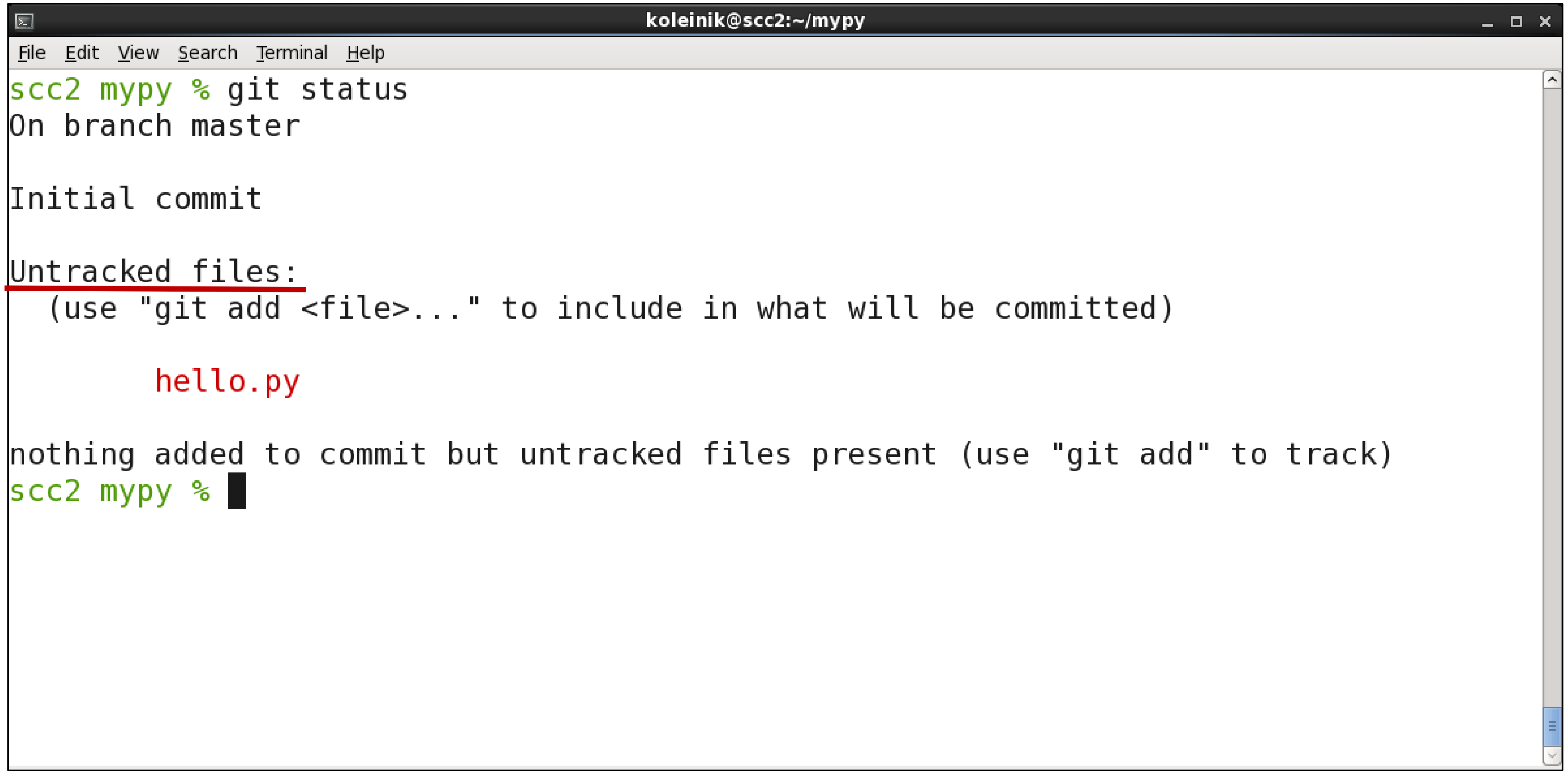
```
gedit hello.py
```

Execute python script (optional)



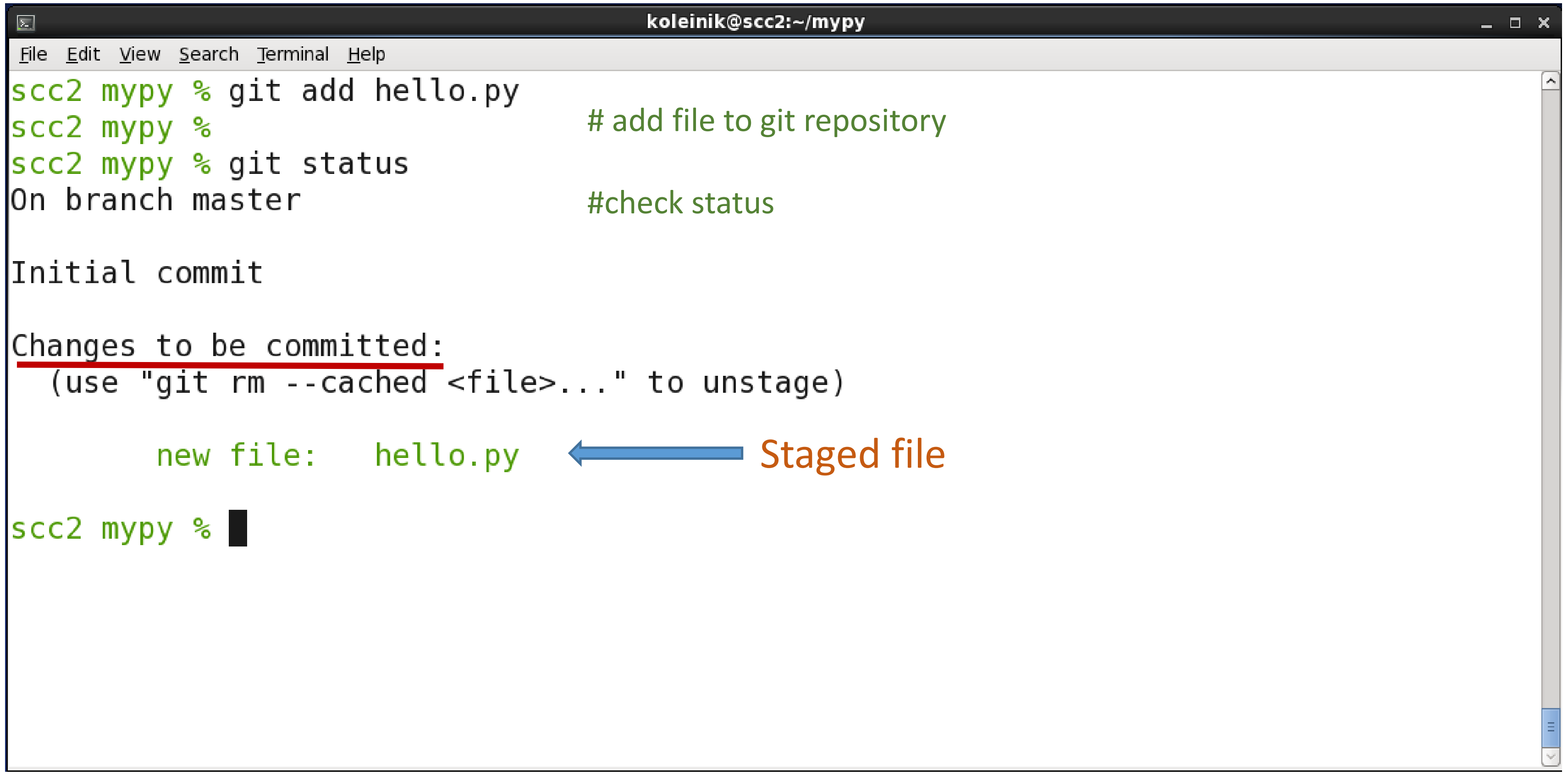
```
scc2 mypy % python hello.py
Hello, Git!
scc2 mypy %
```

Git : check the status



```
scc2 mypy % git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        hello.py

nothing added to commit but untracked files present (use "git add" to track)
scc2 mypy %
```

Git : add file to the repository



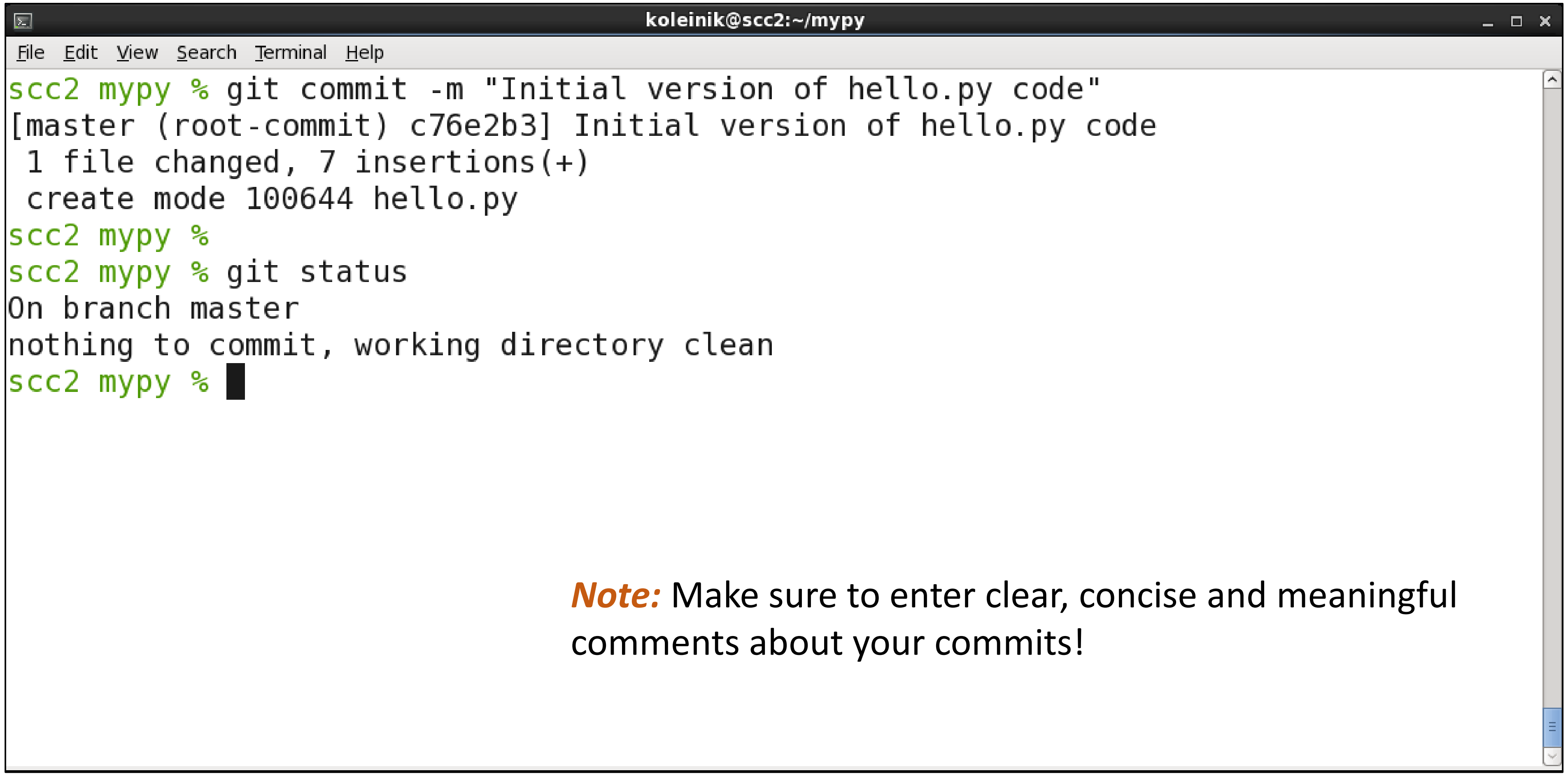
```
scc2 mypy % git add hello.py
scc2 mypy %
scc2 mypy % git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   hello.py

scc2 mypy %
```

# add file to git repository

#check status

Staged file

Git : commit

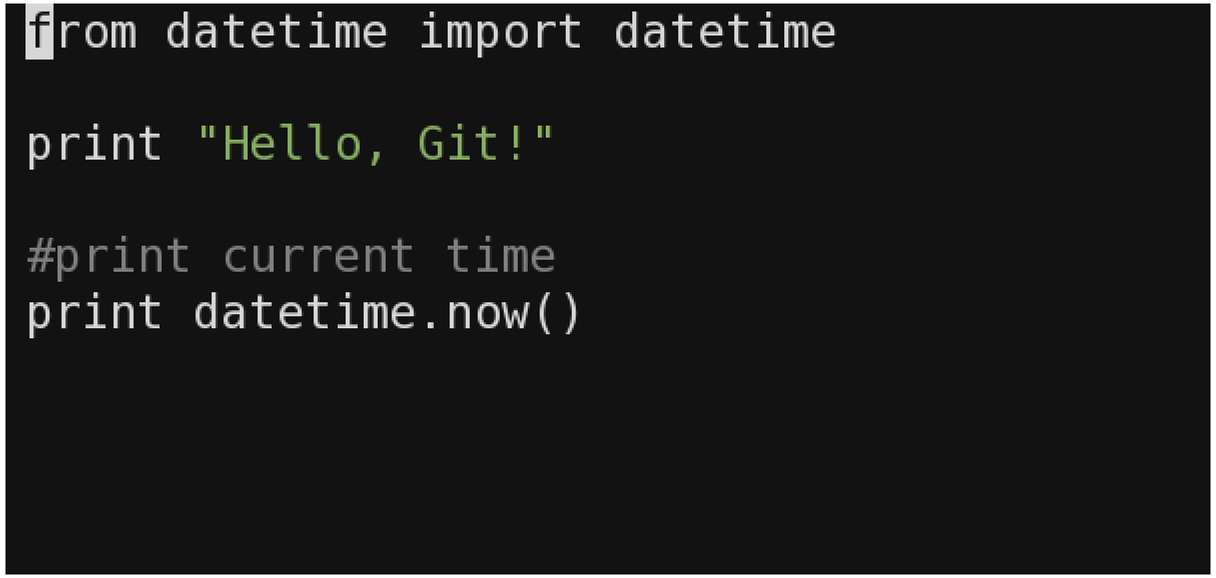


```
scc2 mypy % git commit -m "Initial version of hello.py code"
[master (root-commit) c76e2b3] Initial version of hello.py code
 1 file changed, 7 insertions(+)
 create mode 100644 hello.py
scc2 mypy %
scc2 mypy % git status
On branch master
nothing to commit, working directory clean
scc2 mypy %
```

Note: Make sure to enter clear, concise and meaningful comments about your commits!

Modify a file that is tracked by git

Using your favorite editor, modify existing python code

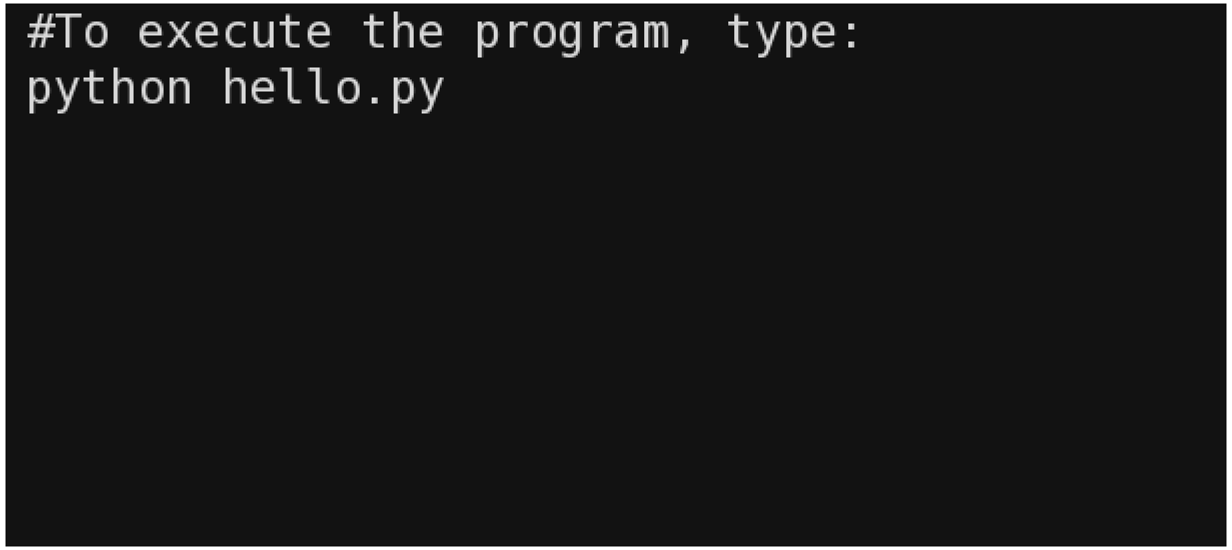


```
from datetime import datetime

print "Hello, Git!"

#print current time
print datetime.now()
```

Save the file and exit.

Create a new README file

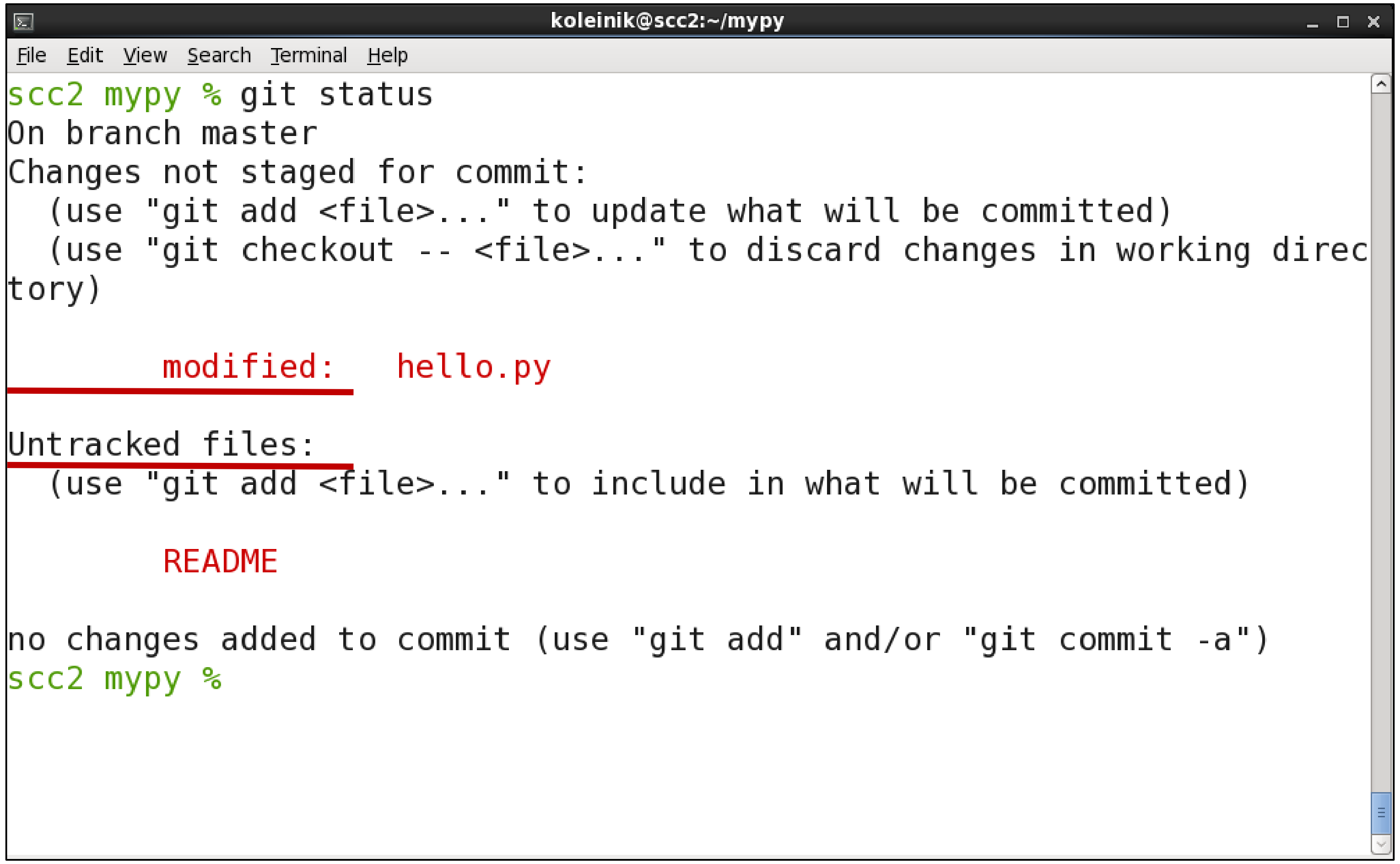
Using your favorite editor create README file and add some content:



```
#To execute the program, type:
python hello.py
```

Save the file and exit.

Git : check the status

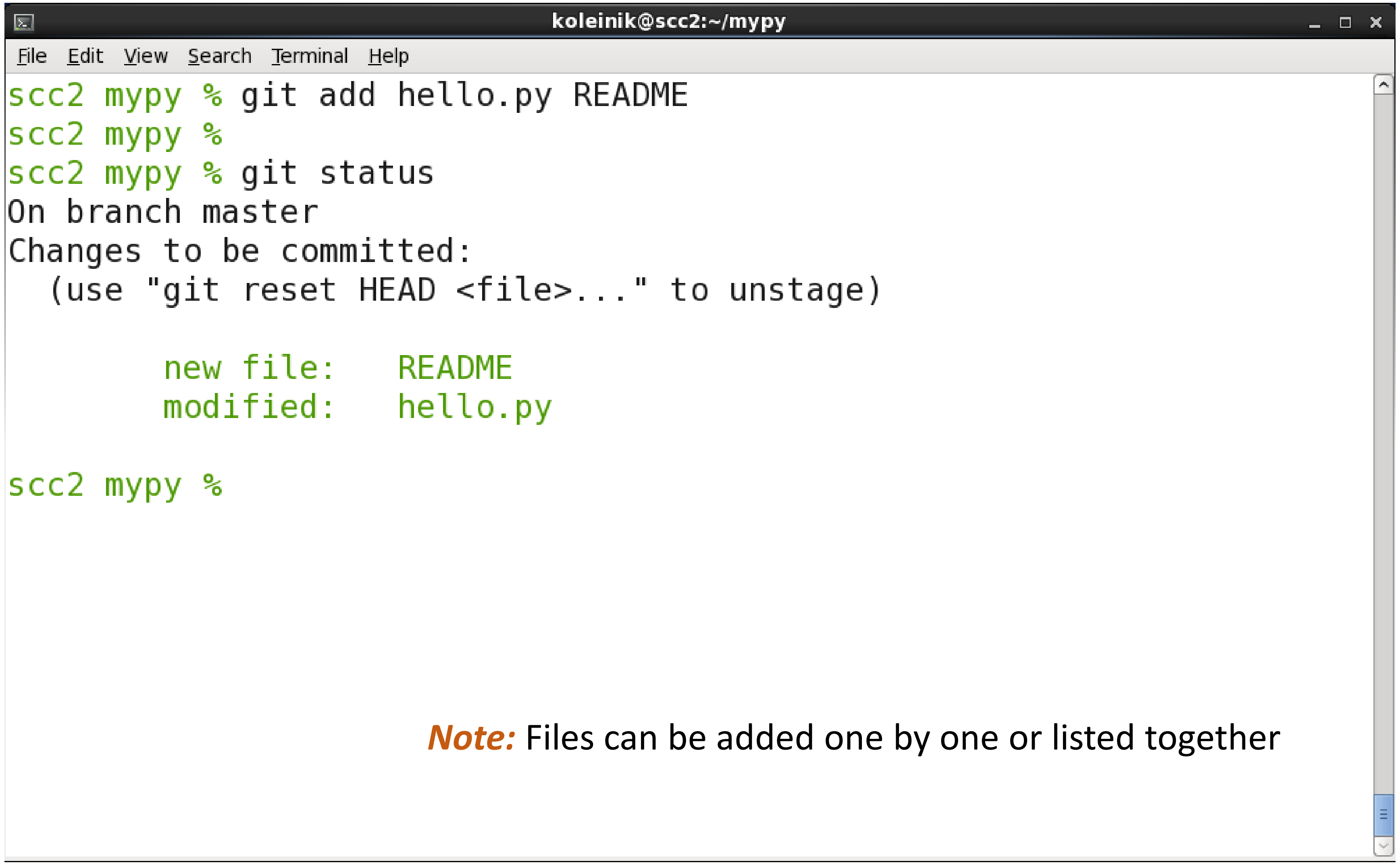


```
scc2 mypy % git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working direc
tory)

        modified:   hello.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        README

no changes added to commit (use "git add" and/or "git commit -a")
scc2 mypy %
```

hello.py has status "modified". Git knows about this file, but reminds that the file has been modified since the last commit

README has status "untracked". Git has no information about this file.

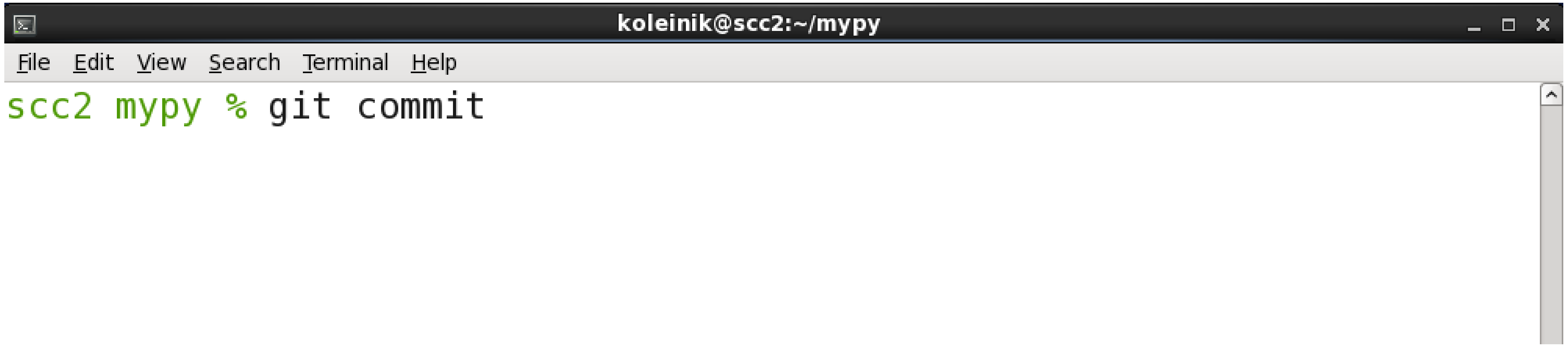
Git : add files to a staging area



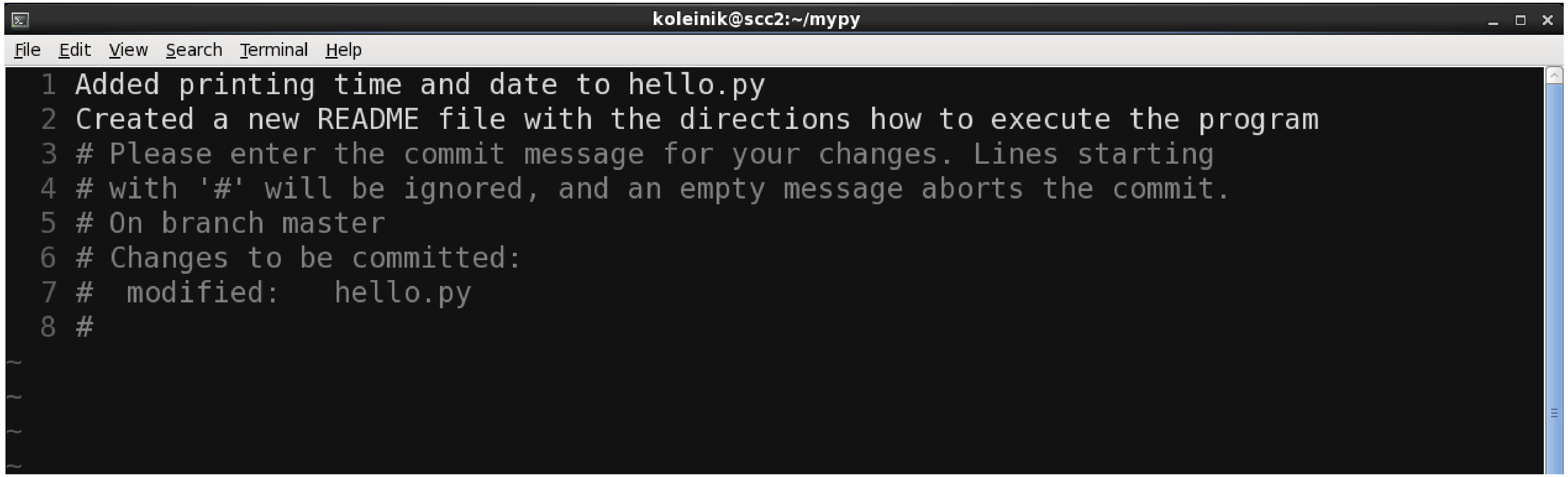
```
scc2 mypy % git add hello.py README
scc2 mypy %
scc2 mypy % git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   README
        modified:   hello.py

scc2 mypy %
```

Note: Files can be added one by one or listed together

Git : commit

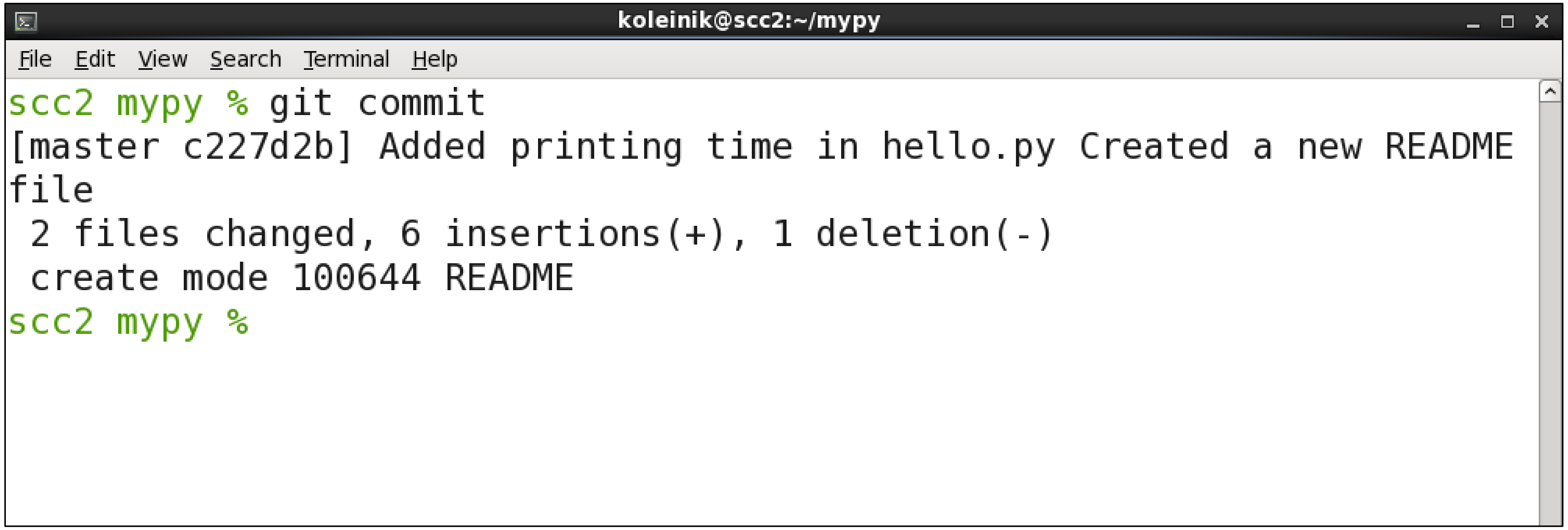


```
scc2 mypy % git commit
```



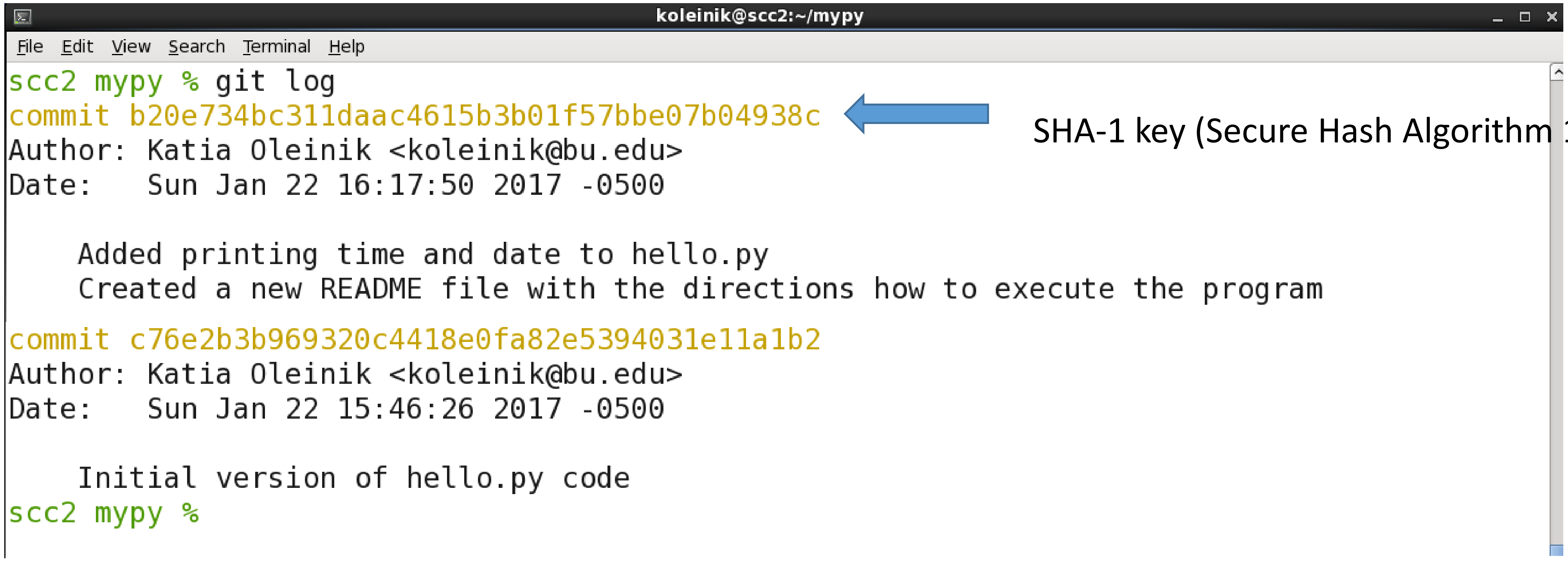
```
Added printing time and date to hello.py
Created a new README file with the directions how to execute the program
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#  modified:   hello.py
#
~
~
~
~
```

Git : commit



```
scc2 mypy % git commit
[master c227d2b] Added printing time in hello.py Created a new README
file
 2 files changed, 6 insertions(+), 1 deletion(-)
 create mode 100644 README
scc2 mypy %
```

Git : view the history of commits



```
scc2 mypy % git log
commit b20e734bc311daac4615b3b01f57bbe07b04938c
Author: Katia Oleinik <koleinik@bu.edu>
Date:   Sun Jan 22 16:17:50 2017 -0500

    Added printing time and date to hello.py
    Created a new README file with the directions how to execute the program

commit c76e2b3b969320c4418e0fa82e5394031e11a1b2
Author: Katia Oleinik <koleinik@bu.edu>
Date:   Sun Jan 22 15:46:26 2017 -0500

    Initial version of hello.py code
scc2 mypy %
```

SHA-1 key (Secure Hash Algorithm 1)

Note: Git uses SHA-1 only to produce a unique hash tag

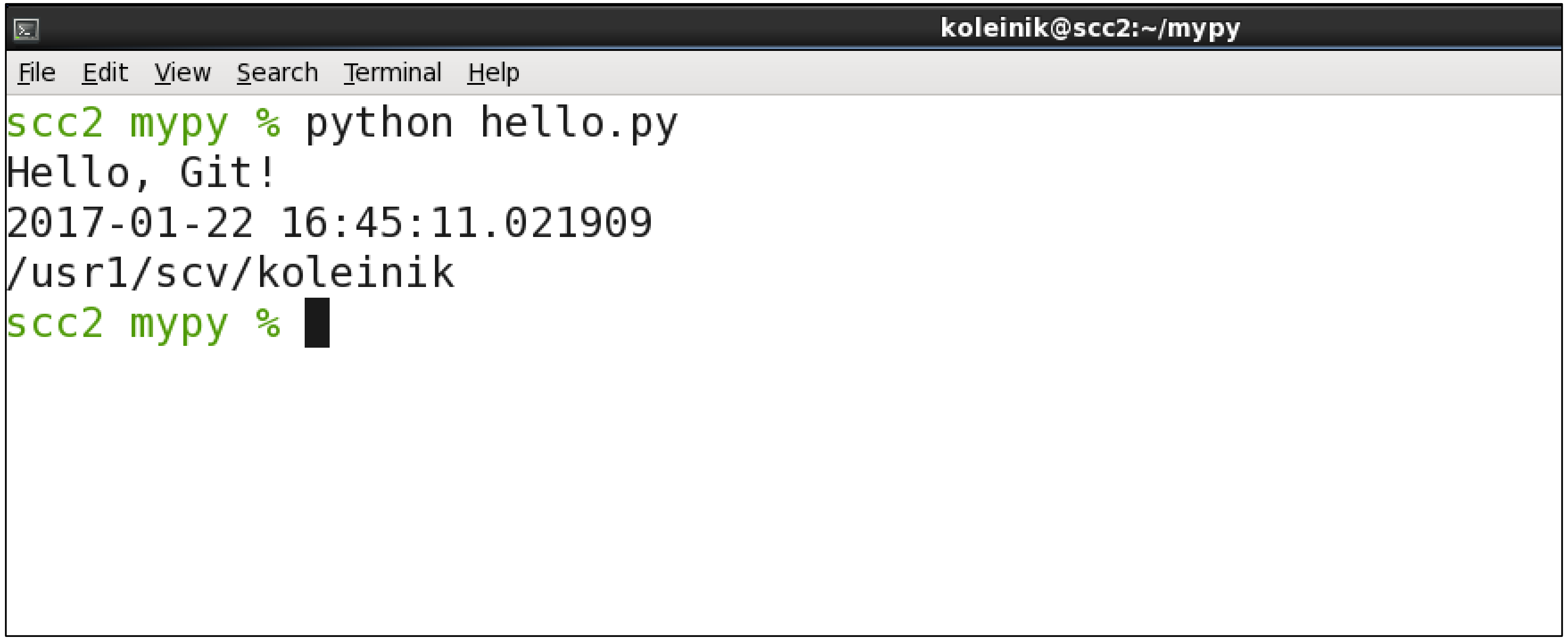
Modify a hello.py file again

Using your favorite editor, modify existing python code

```python
from datetime import datetime
import os


print "Hello, Git!"

#print current date time
print datetime.now()

#print home directory path
print os.environ['HOME']
```

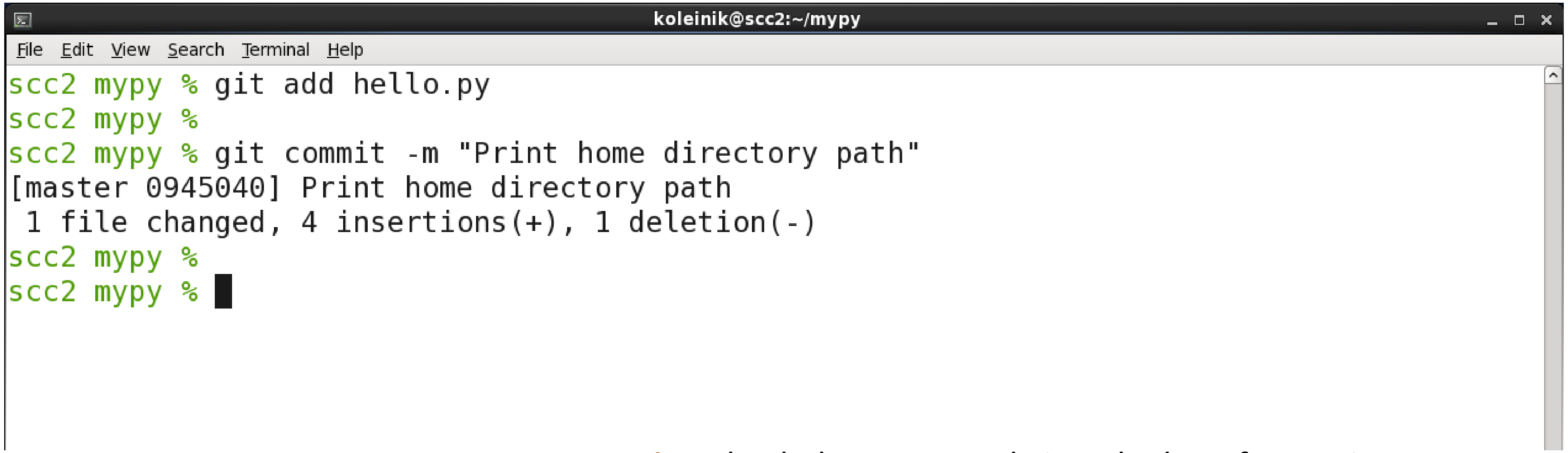
Save the file and exit.

Execute modified version of hello.py (optional)



```
scc2 mypy % python hello.py
Hello, Git!
2017-01-22 16:45:11.021909
/usr1/scv/koleinik
scc2 mypy % 
```

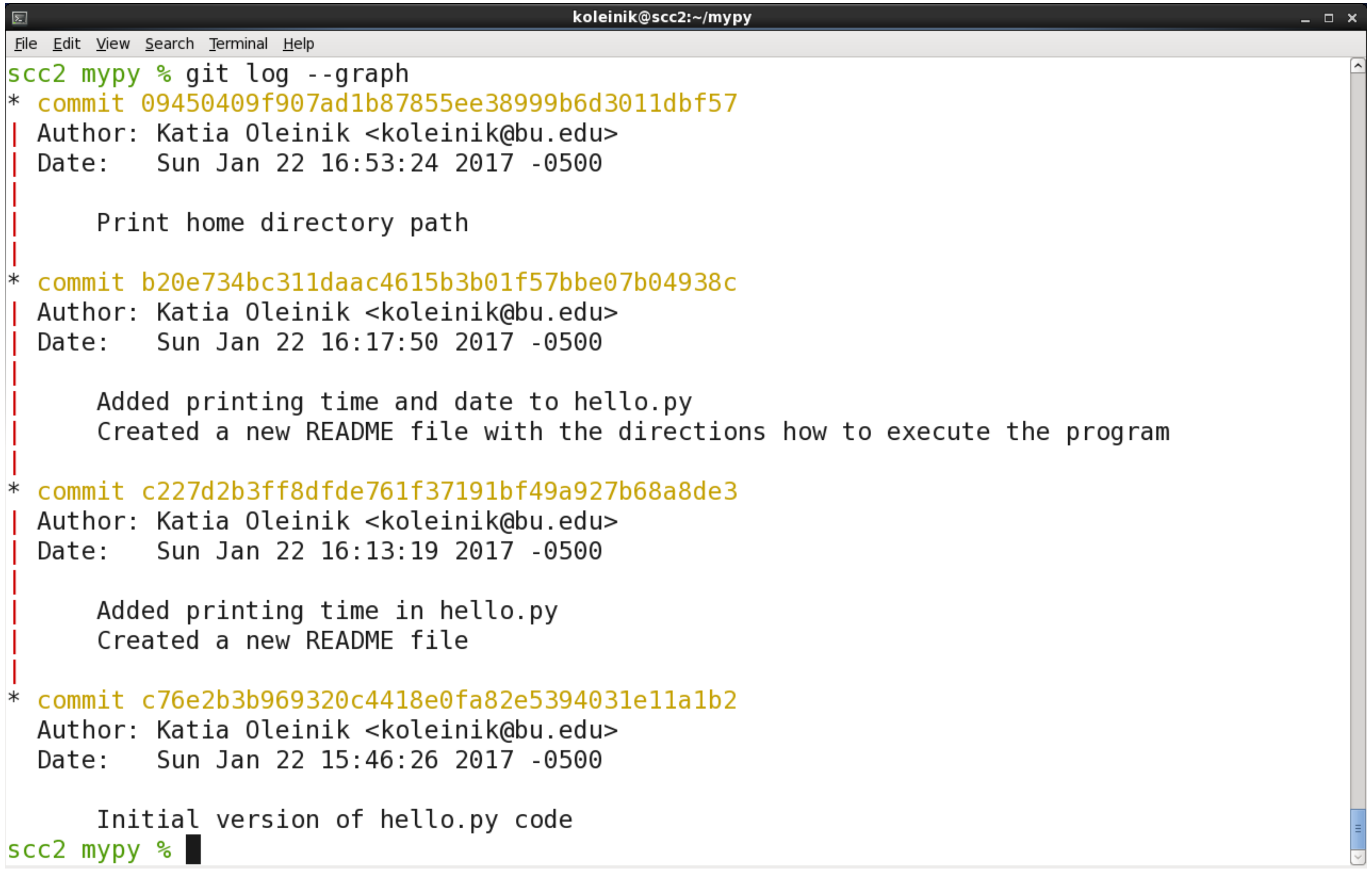
Git : add and commit file



```
scc2 mypy % git add hello.py
scc2 mypy %
scc2 mypy % git commit -m "Print home directory path"
[master 0945040] Print home directory path
 1 file changed, 4 insertions(+), 1 deletion(-)
scc2 mypy %
scc2 mypy %
```

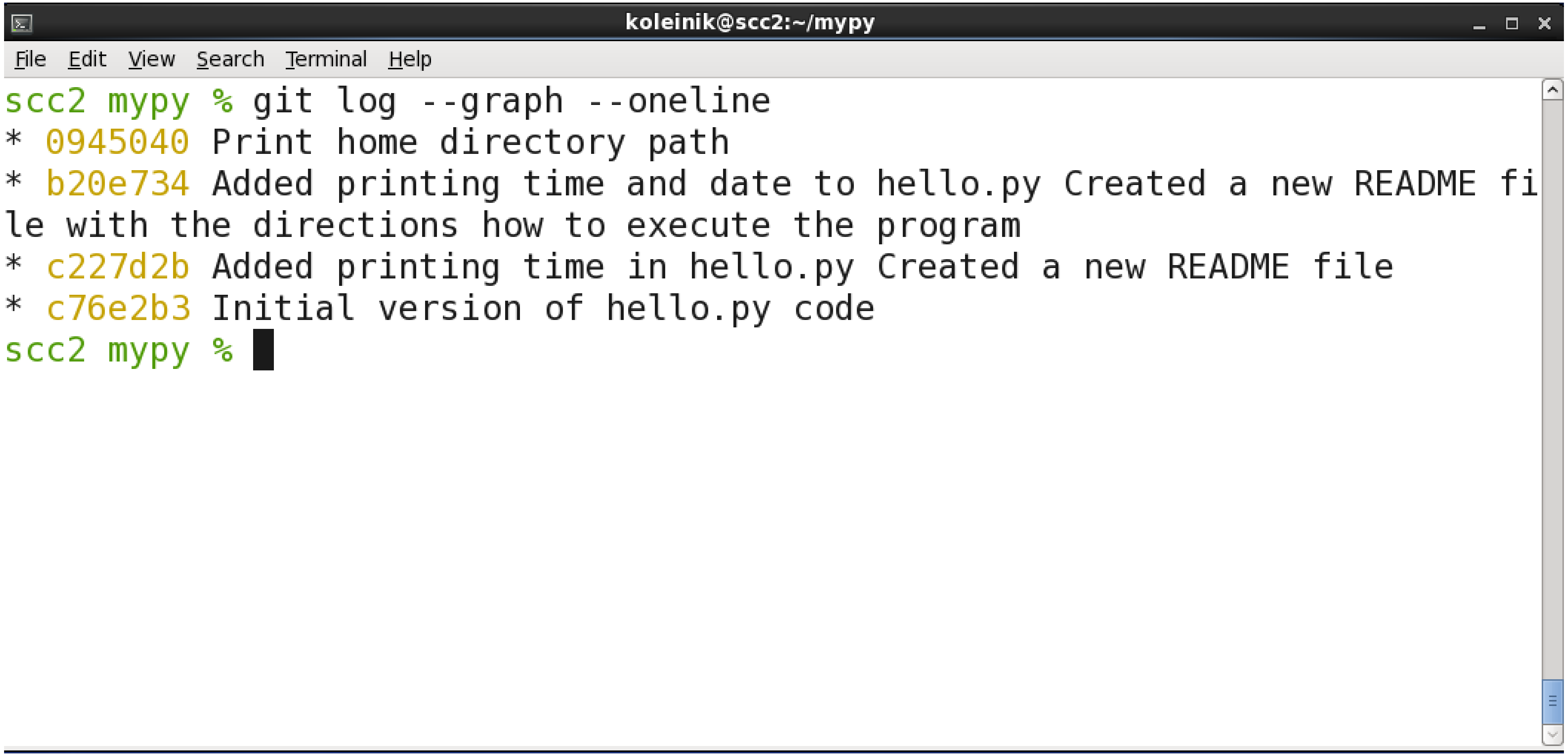
Practice: check the status and view the log of commits.

Git : view log with a graph



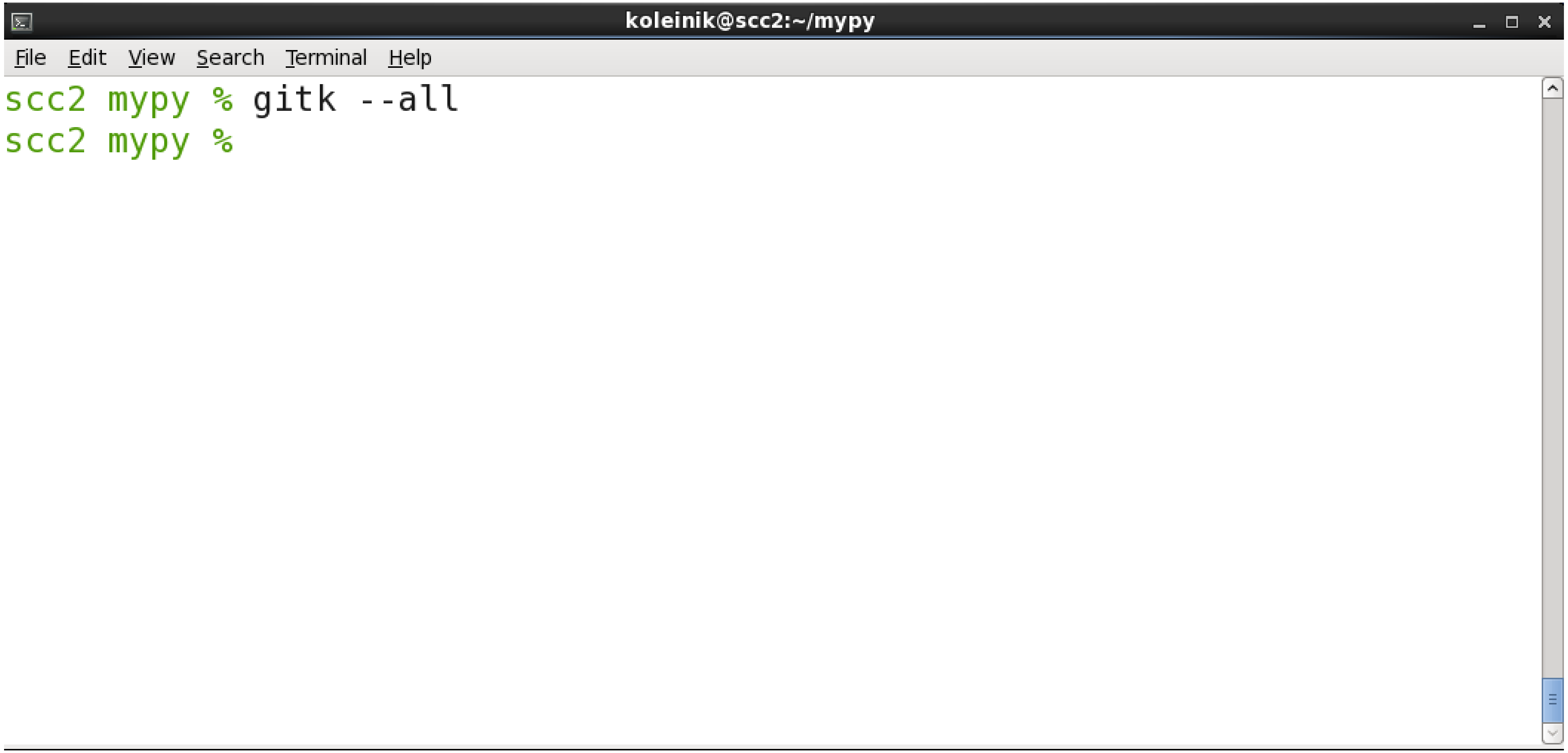
```
scc2 mypy % git log --graph
* commit 09450409f907ad1b87855ee38999b6d3011dbf57
| Author: Katia Oleinik <koleinik@bu.edu>
| Date:   Sun Jan 22 16:53:24 2017 -0500
|
|     Print home directory path
|
* commit b20e734bc311daac4615b3b01f57bbe07b04938c
| Author: Katia Oleinik <koleinik@bu.edu>
| Date:   Sun Jan 22 16:17:50 2017 -0500
|
|     Added printing time and date to hello.py
|     Created a new README file with the directions how to execute the program
|
* commit c227d2b3ff8dfde761f37191bf49a927b68a8de3
| Author: Katia Oleinik <koleinik@bu.edu>
| Date:   Sun Jan 22 16:13:19 2017 -0500
|
|     Added printing time in hello.py
|     Created a new README file
|
* commit c76e2b3b969320c4418e0fa82e5394031e11a1b2
  Author: Katia Oleinik <koleinik@bu.edu>
  Date:   Sun Jan 22 15:46:26 2017 -0500

      Initial version of hello.py code
scc2 mypy %
```

Git : one line log



```
scc2 mypy % git log --graph --oneline
* 0945040 Print home directory path
* b20e734 Added printing time and date to hello.py Created a new README fi
le with the directions how to execute the program
* c227d2b Added printing time in hello.py Created a new README file
* c76e2b3 Initial version of hello.py code
scc2 mypy %
```

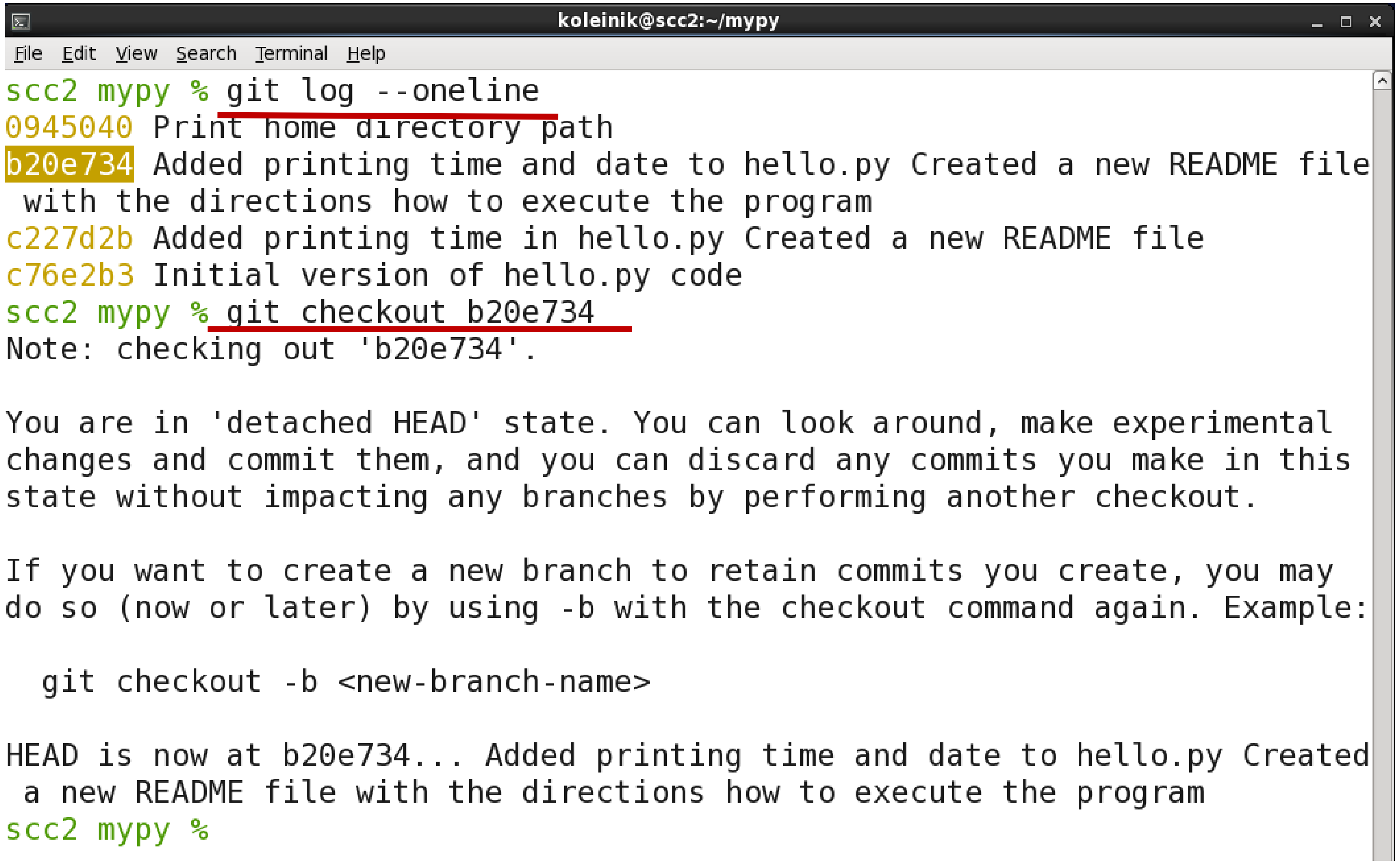
Git : graphical tool



```
scc2 mypy % gitk --all
scc2 mypy %
```

BOSTON UNIVERSITY

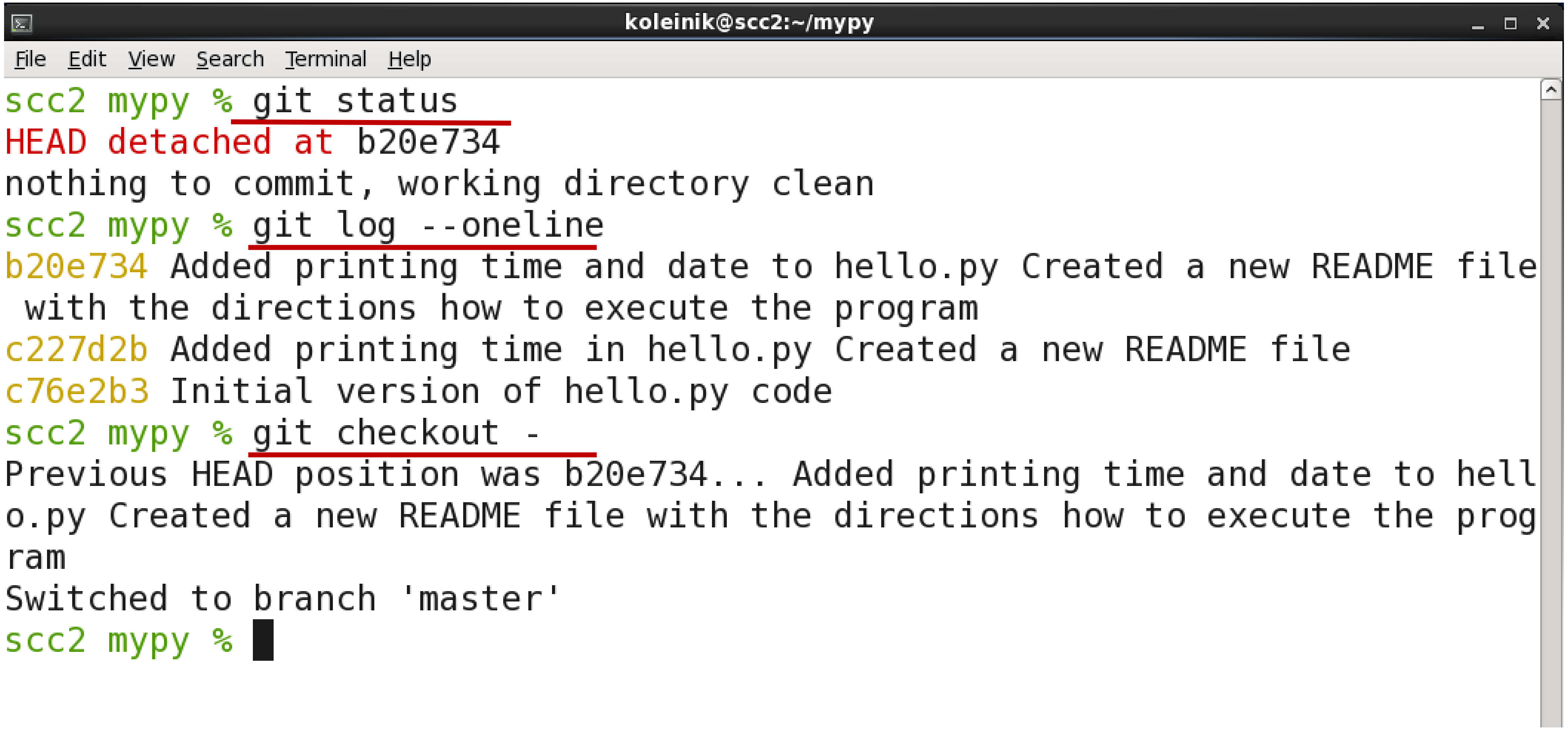
Git : reviewing previous commits



```
scc2 mypy % git log --oneline
0945040 Print home directory path
b20e734 Added printing time and date to hello.py Created a new README file
 with the directions how to execute the program
c227d2b Added printing time in hello.py Created a new README file
c76e2b3 Initial version of hello.py code
scc2 mypy % git checkout b20e734
Note: checking out 'b20e734'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at b20e734... Added printing time and date to hello.py Created
 a new README file with the directions how to execute the program
scc2 mypy %
```

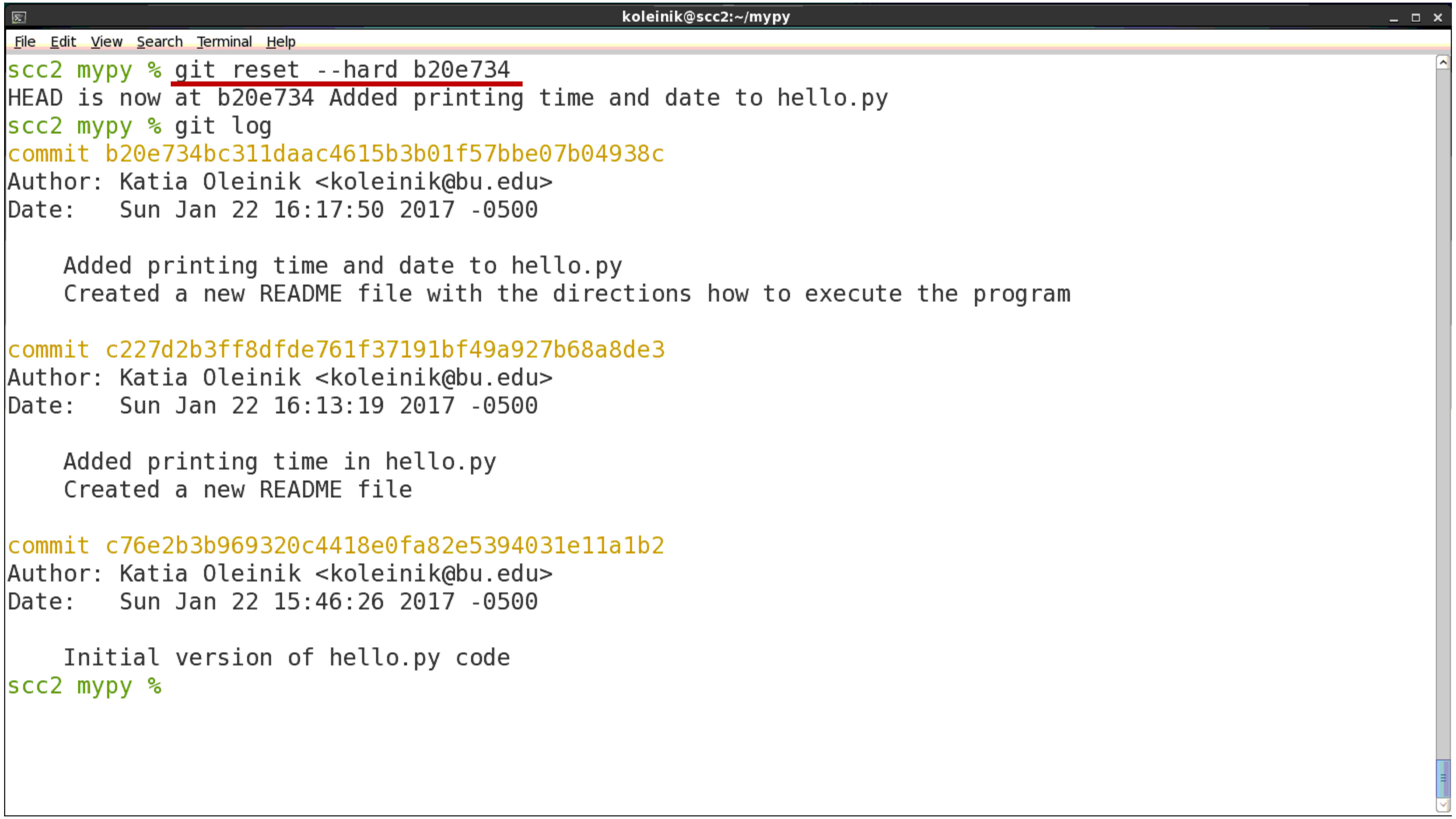
Note: Only first 7 symbols of SHA-1 key are necessary to identify the checkout

Git : returning back to the last commit



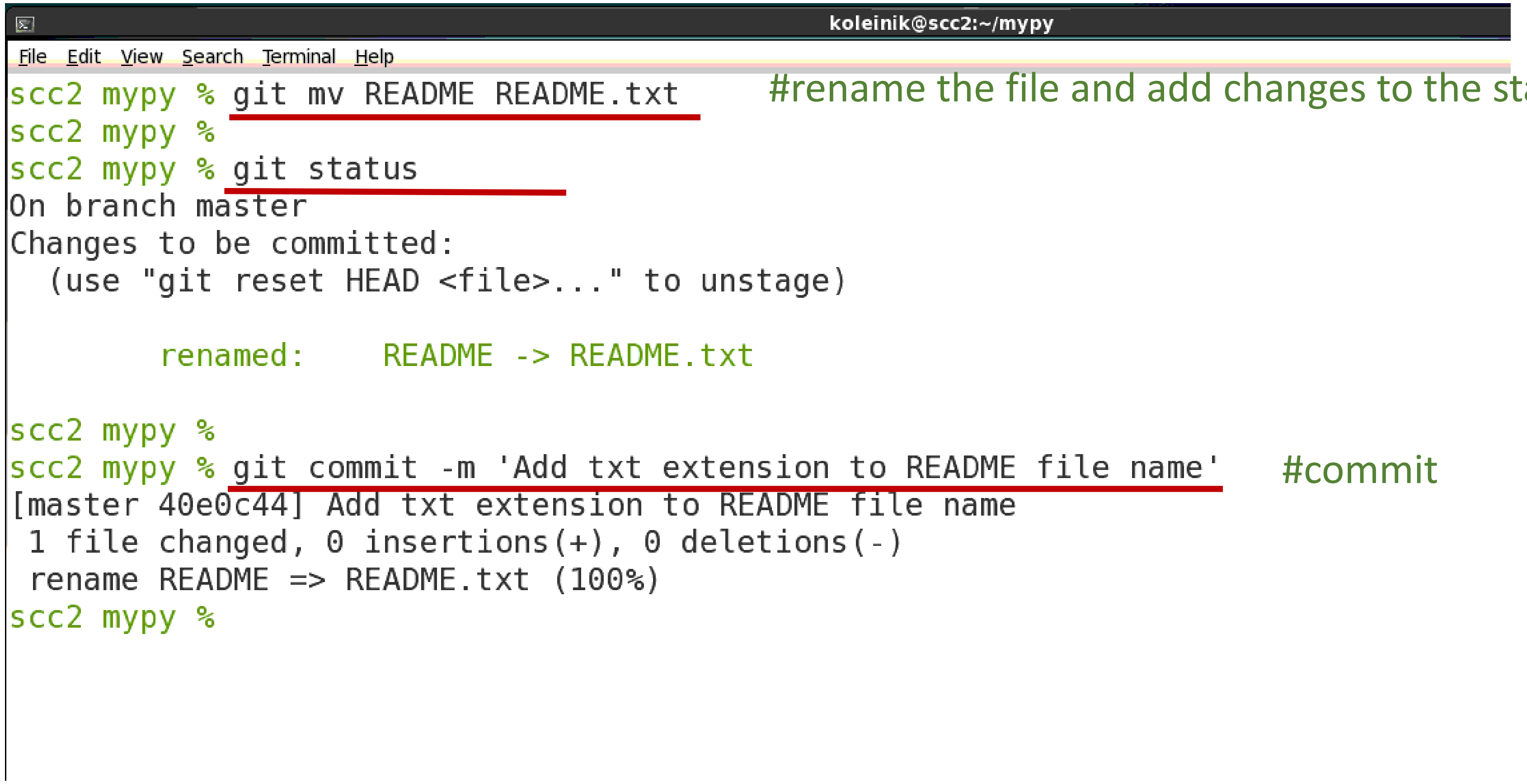
```
scc2 mypy % git status
HEAD detached at b20e734
nothing to commit, working directory clean
scc2 mypy % git log --oneline
b20e734 Added printing time and date to hello.py Created a new README file
 with the directions how to execute the program
c227d2b Added printing time in hello.py Created a new README file
c76e2b3 Initial version of hello.py code
scc2 mypy % git checkout -
Previous HEAD position was b20e734... Added printing time and date to hell
o.py Created a new README file with the directions how to execute the prog
ram
Switched to branch 'master'
scc2 mypy % 
```

Git : hard delete of the latest commits



```
scc2 mypy % git reset --hard b20e734
HEAD is now at b20e734 Added printing time and date to hello.py
scc2 mypy % git log
commit b20e734bc311daac4615b3b01f57bbe07b04938c
Author: Katia Oleinik <koleinik@bu.edu>
Date:   Sun Jan 22 16:17:50 2017 -0500

    Added printing time and date to hello.py
    Created a new README file with the directions how to execute the program

commit c227d2b3ff8dfde761f37191bf49a927b68a8de3
Author: Katia Oleinik <koleinik@bu.edu>
Date:   Sun Jan 22 16:13:19 2017 -0500

    Added printing time in hello.py
    Created a new README file

commit c76e2b3b969320c4418e0fa82e5394031e11a1b2
Author: Katia Oleinik <koleinik@bu.edu>
Date:   Sun Jan 22 15:46:26 2017 -0500

    Initial version of hello.py code
scc2 mypy %
```

Git : Renaming the files (git way)

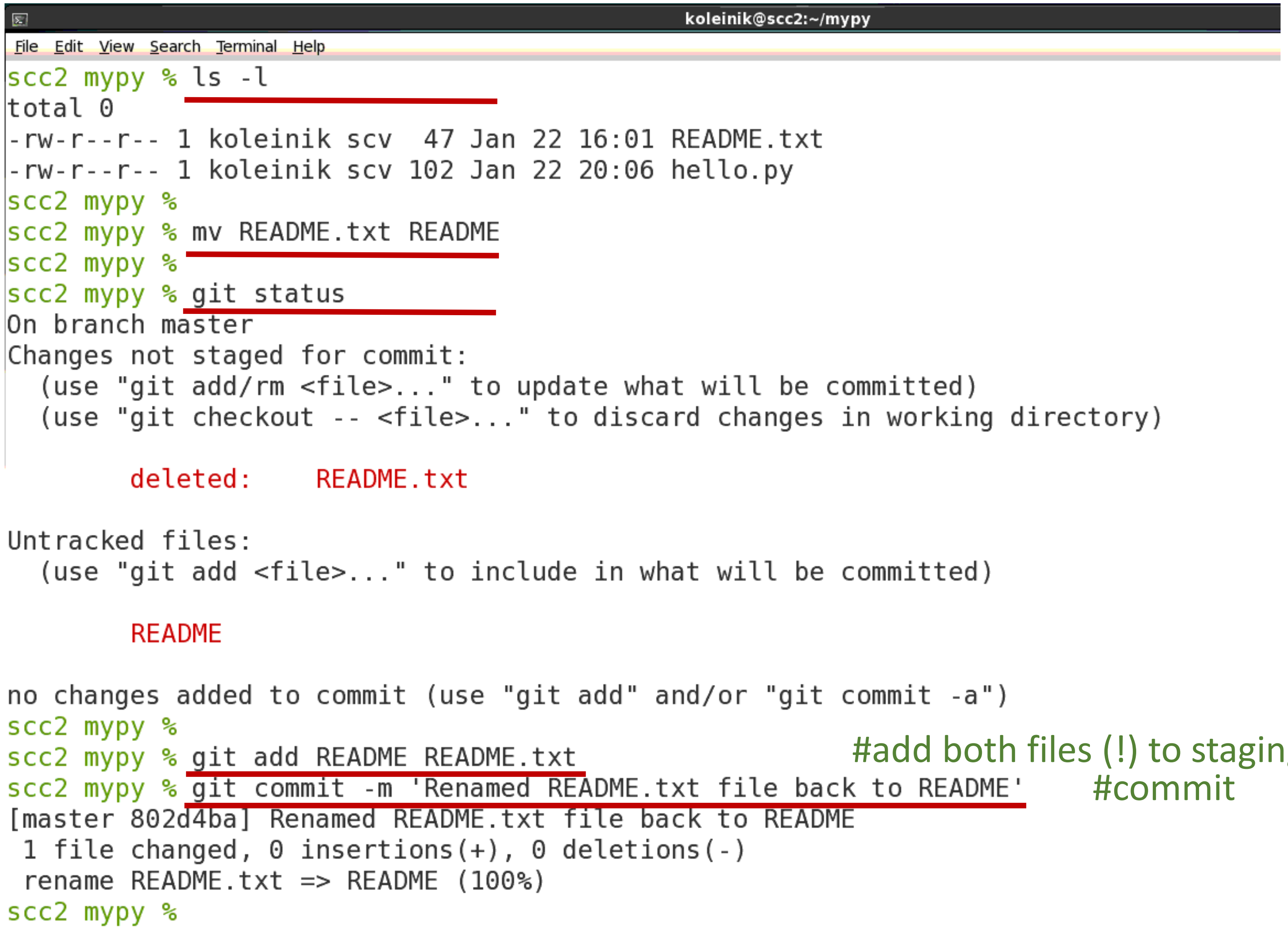


```
scc2 mypy % git mv README README.txt
scc2 mypy %
scc2 mypy % git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        renamed:    README -> README.txt

scc2 mypy %
scc2 mypy % git commit -m 'Add txt extension to README file name'
[master 40e0c44] Add txt extension to README file name
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename README => README.txt (100%)
scc2 mypy %
```

#rename the file and add changes to the staging area

#commit

Git : Renaming the files (outside git)

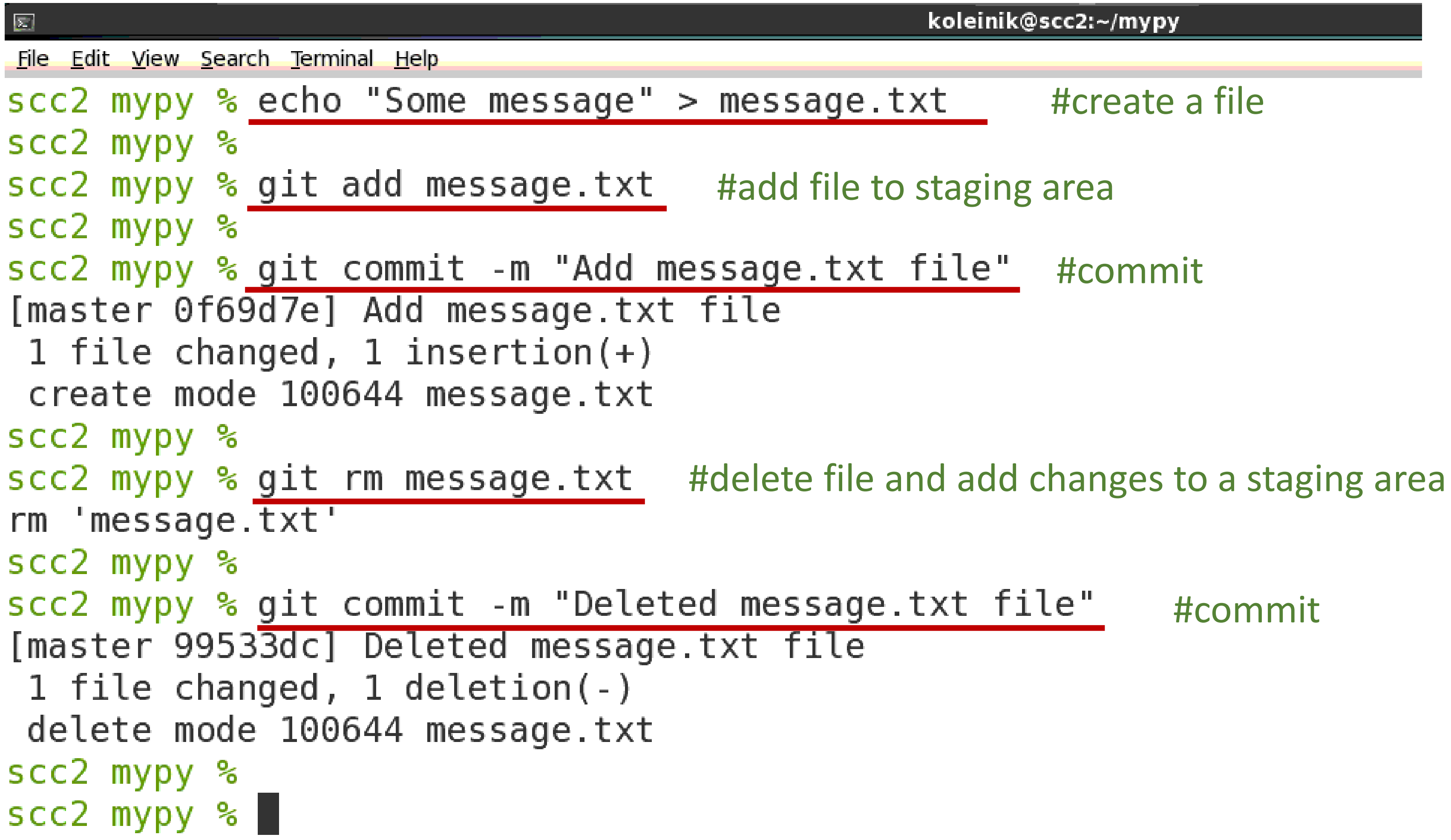


```
scc2 mypy % ls -l
total 0
-rw-r--r-- 1 koleinik scv  47 Jan 22 16:01 README.txt
-rw-r--r-- 1 koleinik scv 102 Jan 22 20:06 hello.py
scc2 mypy %
scc2 mypy % mv README.txt README
scc2 mypy %
scc2 mypy % git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    README.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        README

no changes added to commit (use "git add" and/or "git commit -a")
scc2 mypy %
scc2 mypy % git add README README.txt
scc2 mypy % git commit -m 'Renamed README.txt file back to README'
[master 802d4ba] Renamed README.txt file back to README
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename README.txt => README (100%)
scc2 mypy %
```

#add both files (!) to staging area

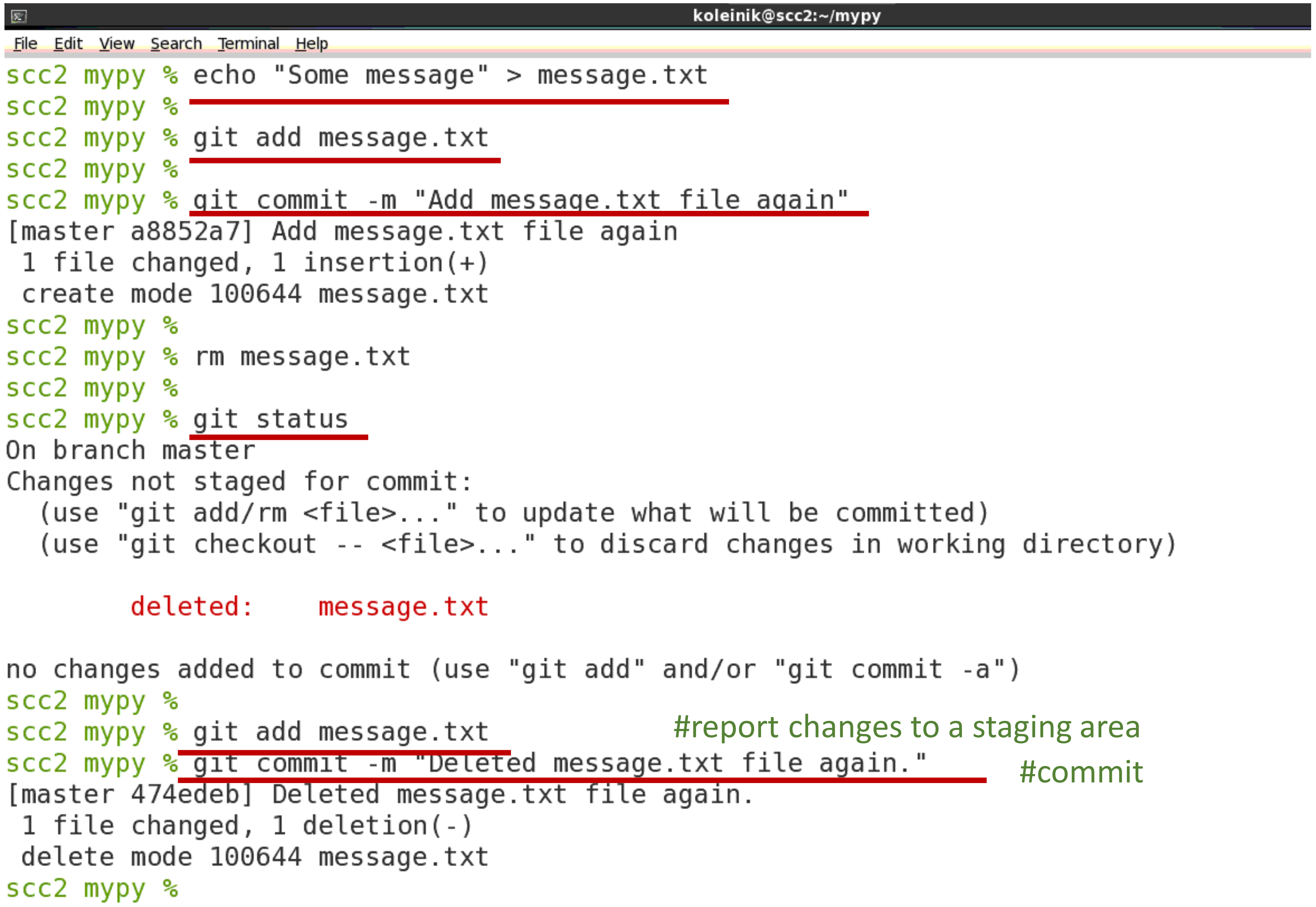
#commit

Git : Deleting the files (git way)



```
scc2 mypy % echo "Some message" > message.txt      #create a file
scc2 mypy %
scc2 mypy % git add message.txt      #add file to staging area
scc2 mypy %
scc2 mypy % git commit -m "Add message.txt file"      #commit
[master 0f69d7e] Add message.txt file
 1 file changed, 1 insertion(+)
 create mode 100644 message.txt
scc2 mypy %
scc2 mypy % git rm message.txt      #delete file and add changes to a staging area
rm 'message.txt'
scc2 mypy %
scc2 mypy % git commit -m "Deleted message.txt file"      #commit
[master 99533dc] Deleted message.txt file
 1 file changed, 1 deletion(-)
 delete mode 100644 message.txt
scc2 mypy %
scc2 mypy %
```

Git : Deleting the files (outside of git)

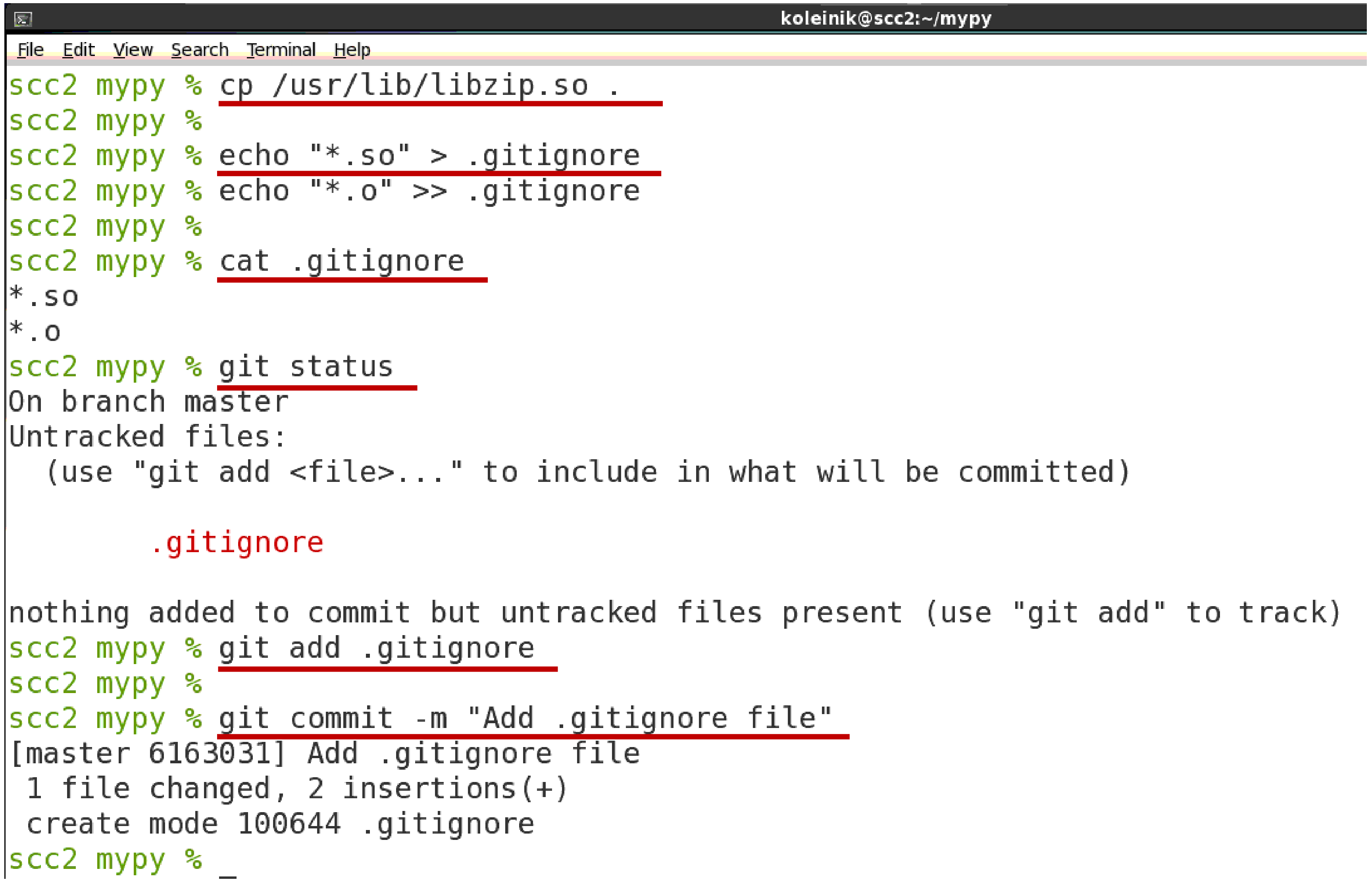


```
scc2 mypy % echo "Some message" > message.txt
scc2 mypy %
scc2 mypy % git add message.txt
scc2 mypy %
scc2 mypy % git commit -m "Add message.txt file again"
[master a8852a7] Add message.txt file again
 1 file changed, 1 insertion(+)
 create mode 100644 message.txt
scc2 mypy %
scc2 mypy % rm message.txt
scc2 mypy %
scc2 mypy % git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    message.txt

no changes added to commit (use "git add" and/or "git commit -a")
scc2 mypy %
scc2 mypy % git add message.txt
scc2 mypy % git commit -m "Deleted message.txt file again."
[master 474edeb] Deleted message.txt file again.
 1 file changed, 1 deletion(-)
 delete mode 100644 message.txt
scc2 mypy %
```

#report changes to a staging area

#commit

Git : ignore some files

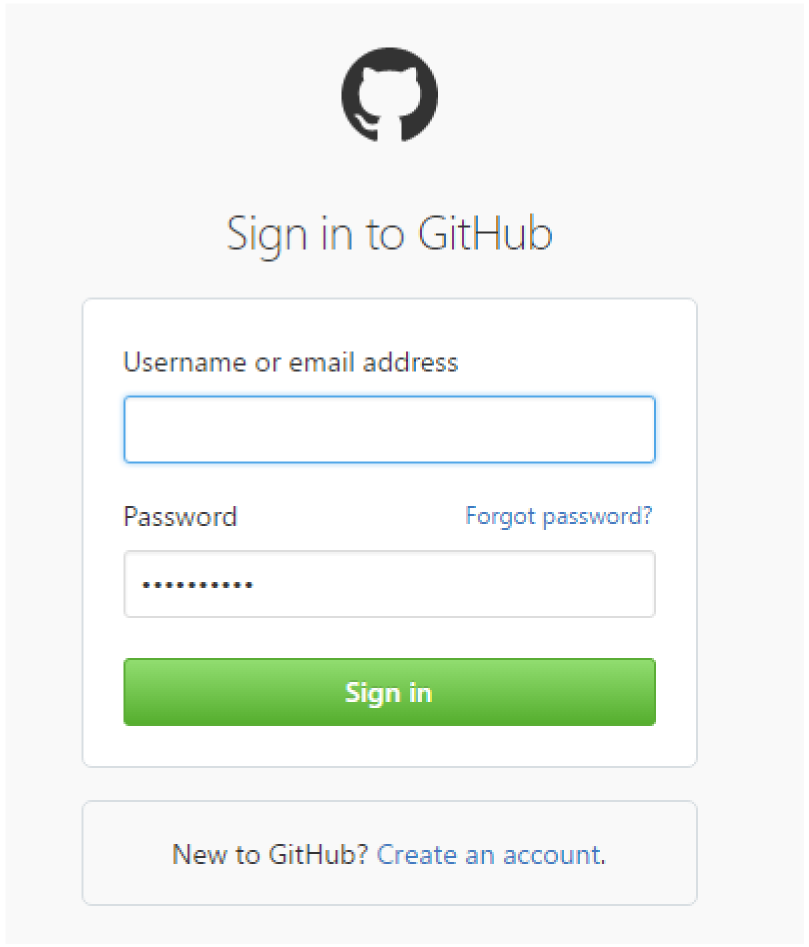


```
scc2 mypy % cp /usr/lib/libzip.so .
scc2 mypy %
scc2 mypy % echo "*.so" > .gitignore
scc2 mypy % echo "*.o" >> .gitignore
scc2 mypy %
scc2 mypy % cat .gitignore
*.so
*.o
scc2 mypy % git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
scc2 mypy % git add .gitignore
scc2 mypy %
scc2 mypy % git commit -m "Add .gitignore file"
[master 6163031] Add .gitignore file
 1 file changed, 2 insertions(+)
 create mode 100644 .gitignore
scc2 mypy % _
```

Submitting work to remote

GitHub, GitLab, Bitbucket, etc.

Login to the account



Sign in to GitHub

Username or email address

Password

Forgot password?

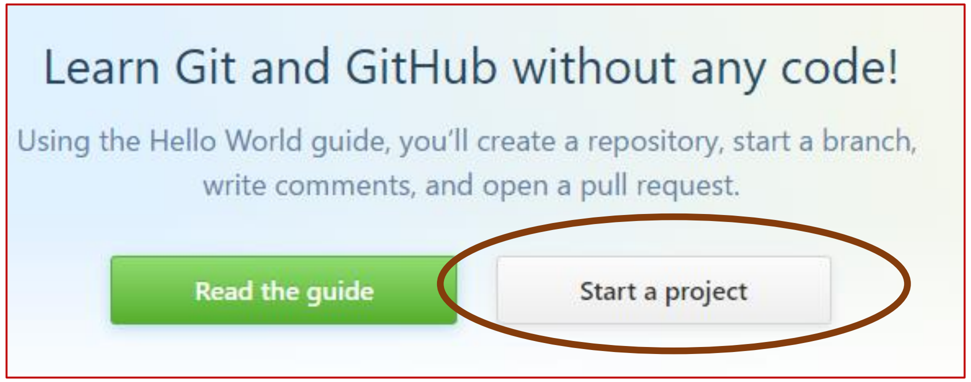
••••••••••

Sign in

New to GitHub? Create an account.

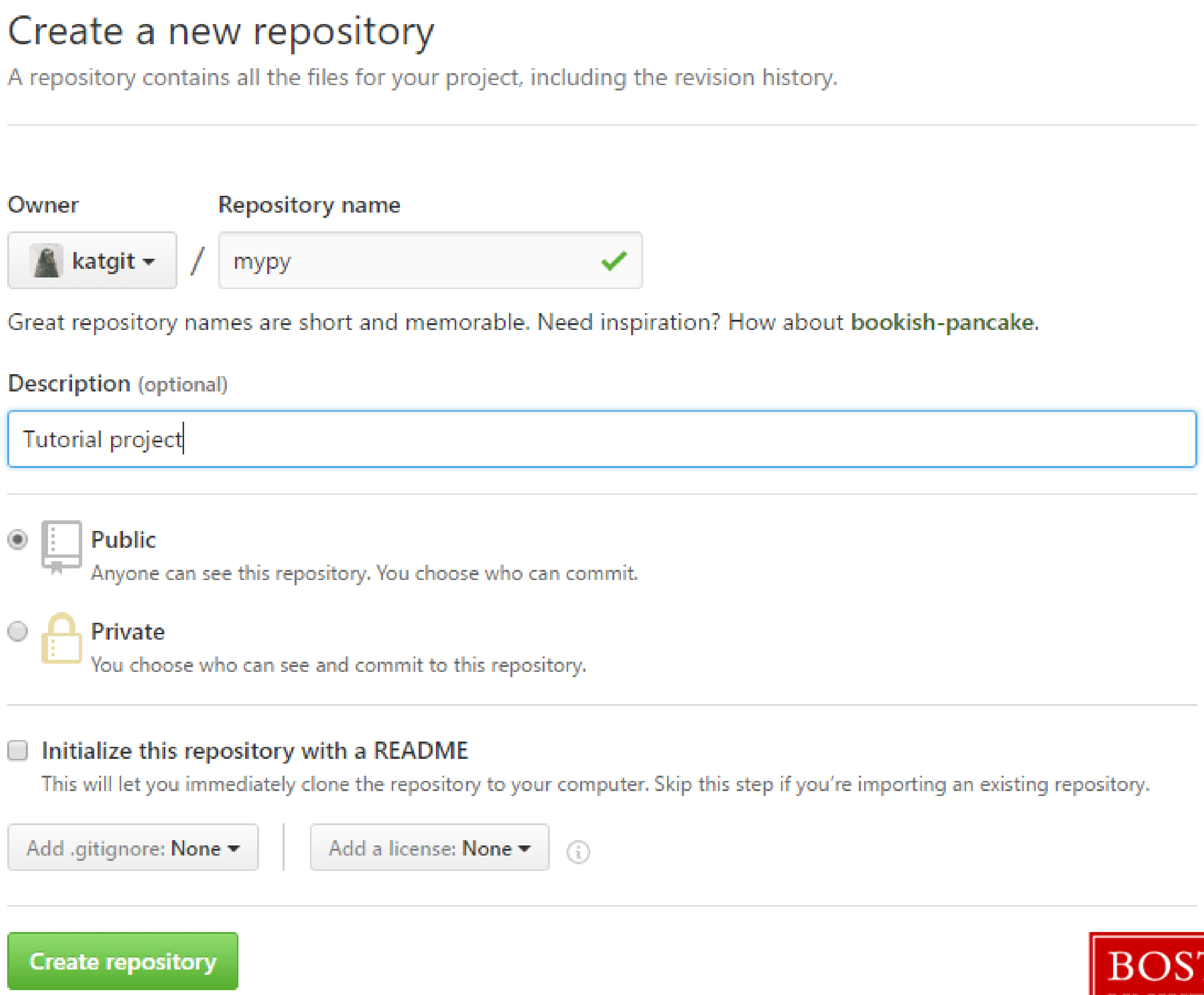
BOSTON UNIVERSITY

Start a new project



Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: katgit / Repository name: mypy

Great repository names are short and memorable. Need inspiration? How about bookish-pancake.

Description (optional): Tutorial project

[✓ Public] Anyone can see this repository. You choose who can commit.

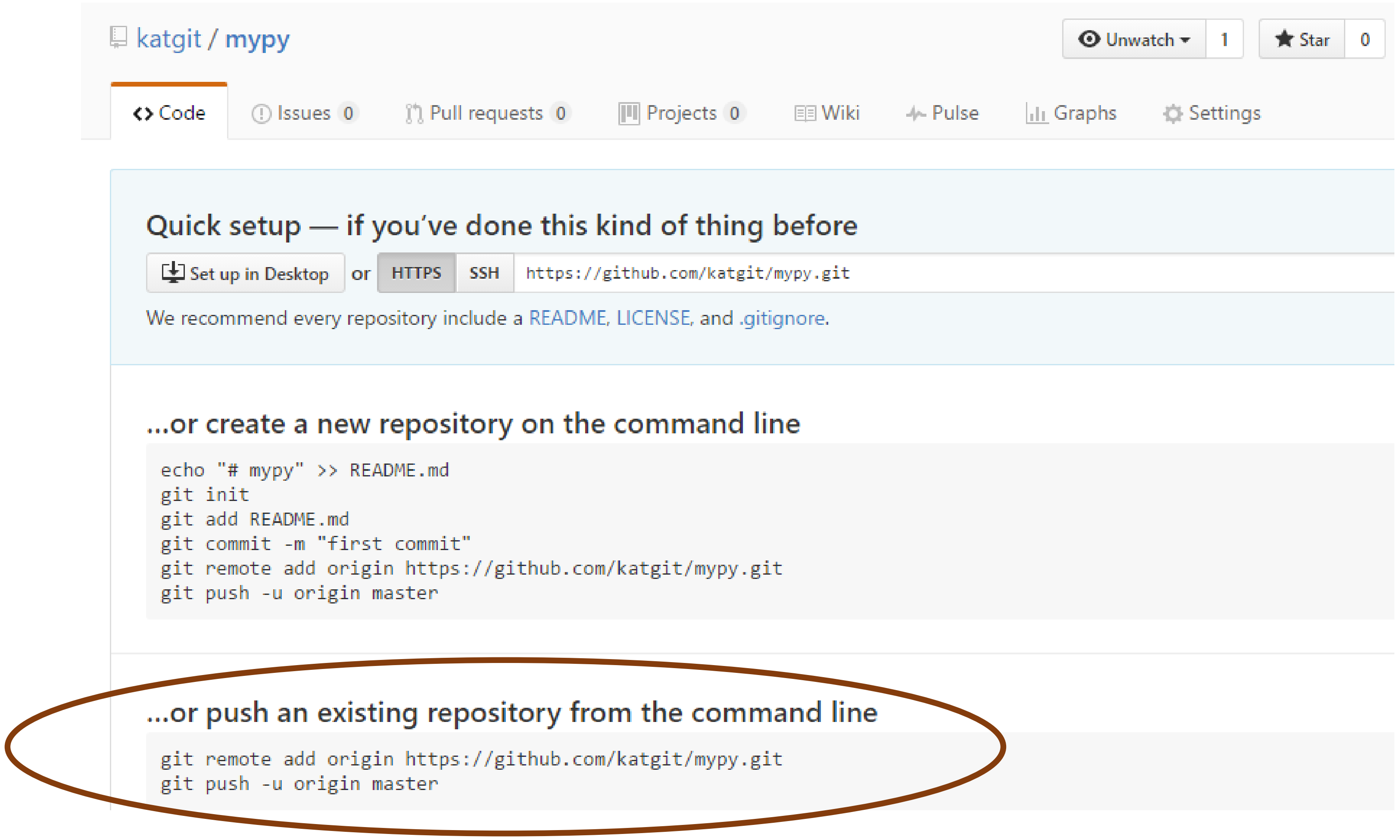
Private: You choose who can see and commit to this repository.

Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Connect your local repo to the remote



Quick setup — if you've done this kind of thing before

```
https://github.com/katgit/mypy.git
```

We recommend every repository include a README, LICENSE, and .gitignore.

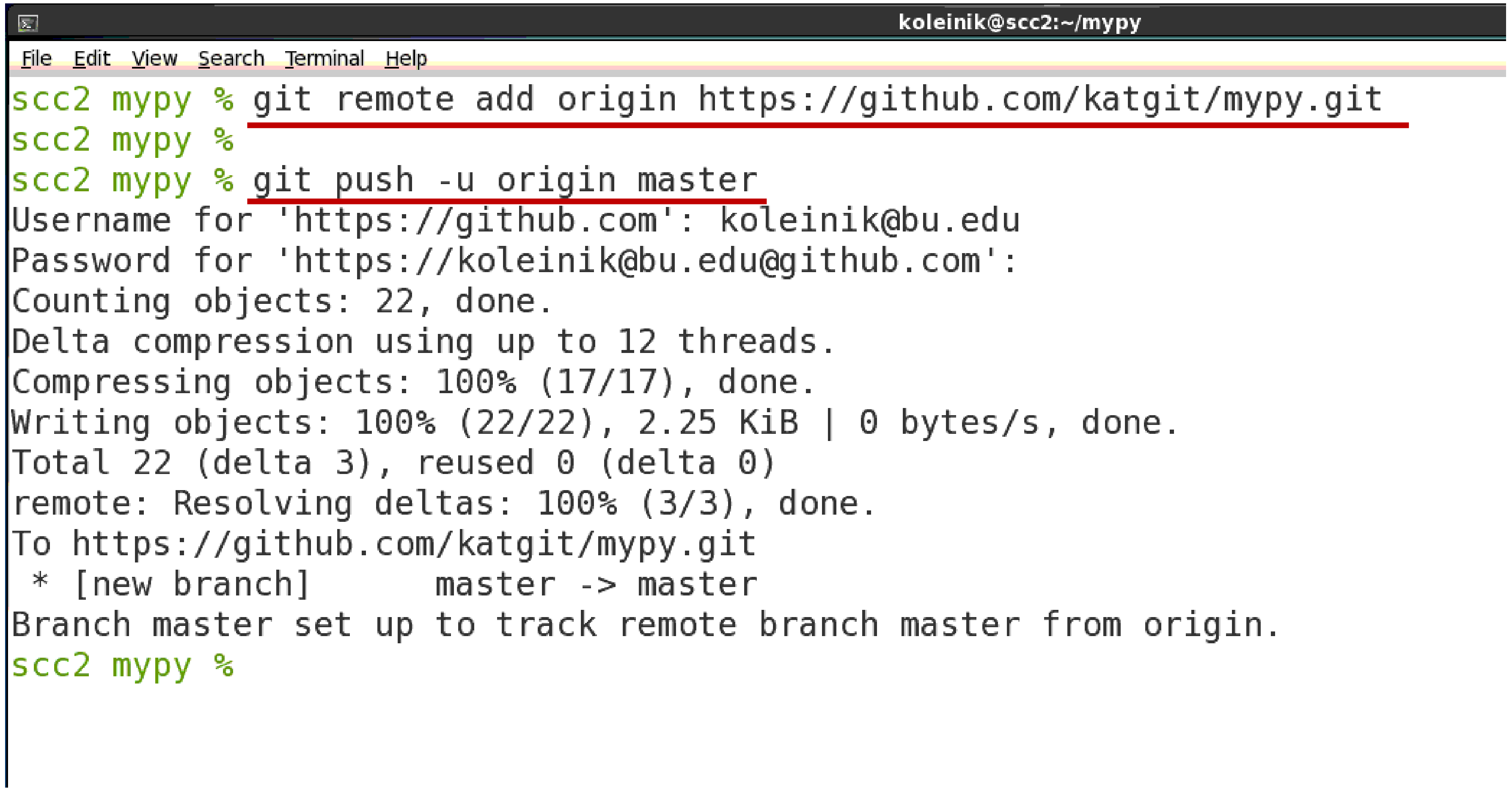
…or create a new repository on the command line

```
echo "# mypy" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/katgit/mypy.git
git push -u origin master
```

…or push an existing repository from the command line

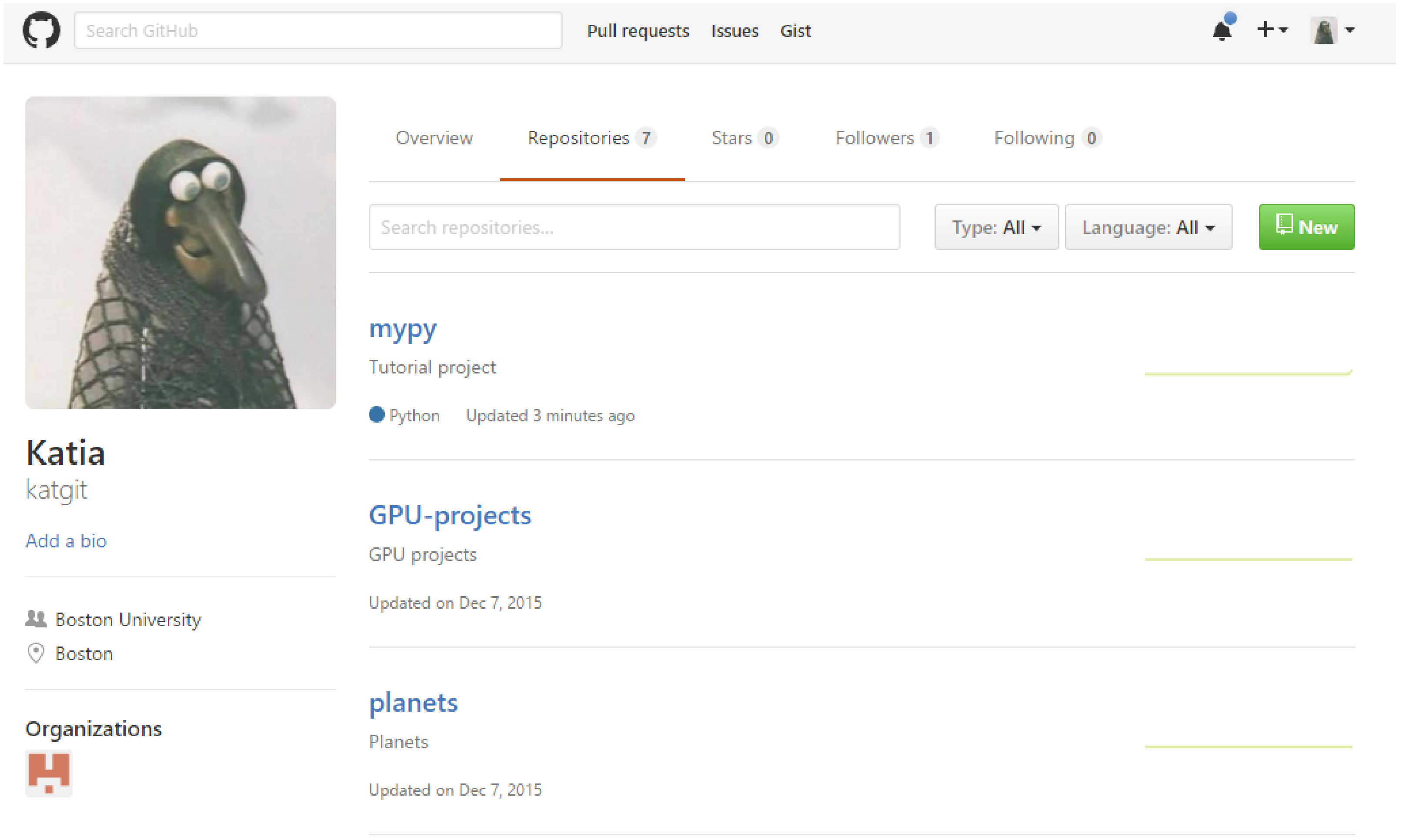
```
git remote add origin https://github.com/katgit/mypy.git
git push -u origin master
```

Connect your local repo to the remote



```
scc2 mypy % git remote add origin https://github.com/katgit/mypy.git
scc2 mypy %
scc2 mypy % git push -u origin master
Username for 'https://github.com': koleinik@bu.edu
Password for 'https://koleinik@bu.edu@github.com':
Counting objects: 22, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (17/17), done.
Writing objects: 100% (22/22), 2.25 KiB | 0 bytes/s, done.
Total 22 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/katgit/mypy.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
scc2 mypy %
```

View remote github repositories



Search GitHub

Pull requests Issues Gist

Overview Repositories 7 Stars 0 Followers 1 Following 0

Search repositories...

Type: All Language: All New

mypy
Tutorial project
Python Updated 3 minutes ago

GPU-projects
GPU projects
Updated on Dec 7, 2015

planets
Planets
Updated on Dec 7, 2015

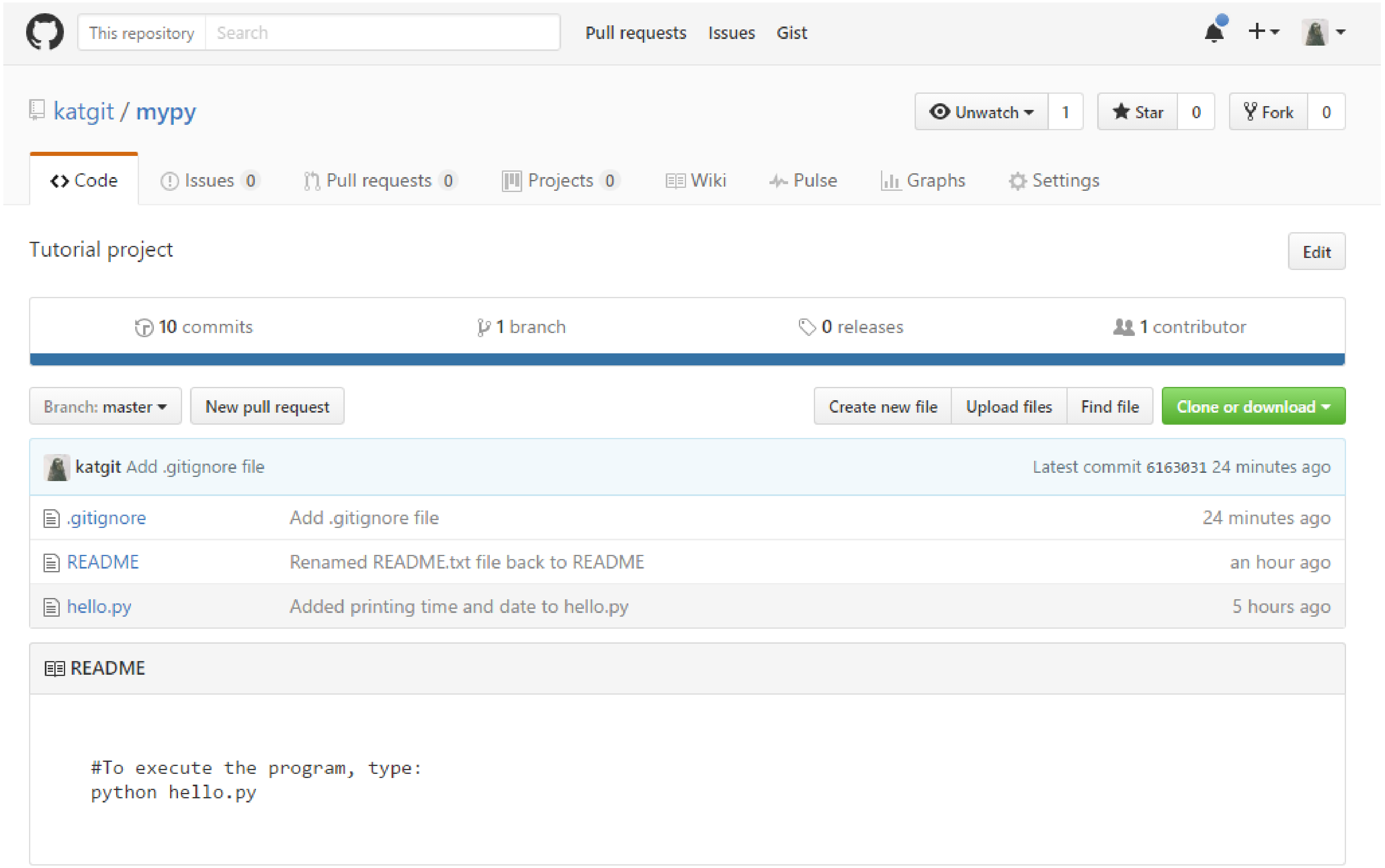
Katia
katgit
Add a bio

Boston University
Boston

Organizations

BOSTON UNIVERSITY

View remote github repositories



katgit / mypy

Tutorial project

10 commits | 1 branch | 0 releases | 1 contributor

katgit Add .gitignore file — Latest commit 6163031 24 minutes ago

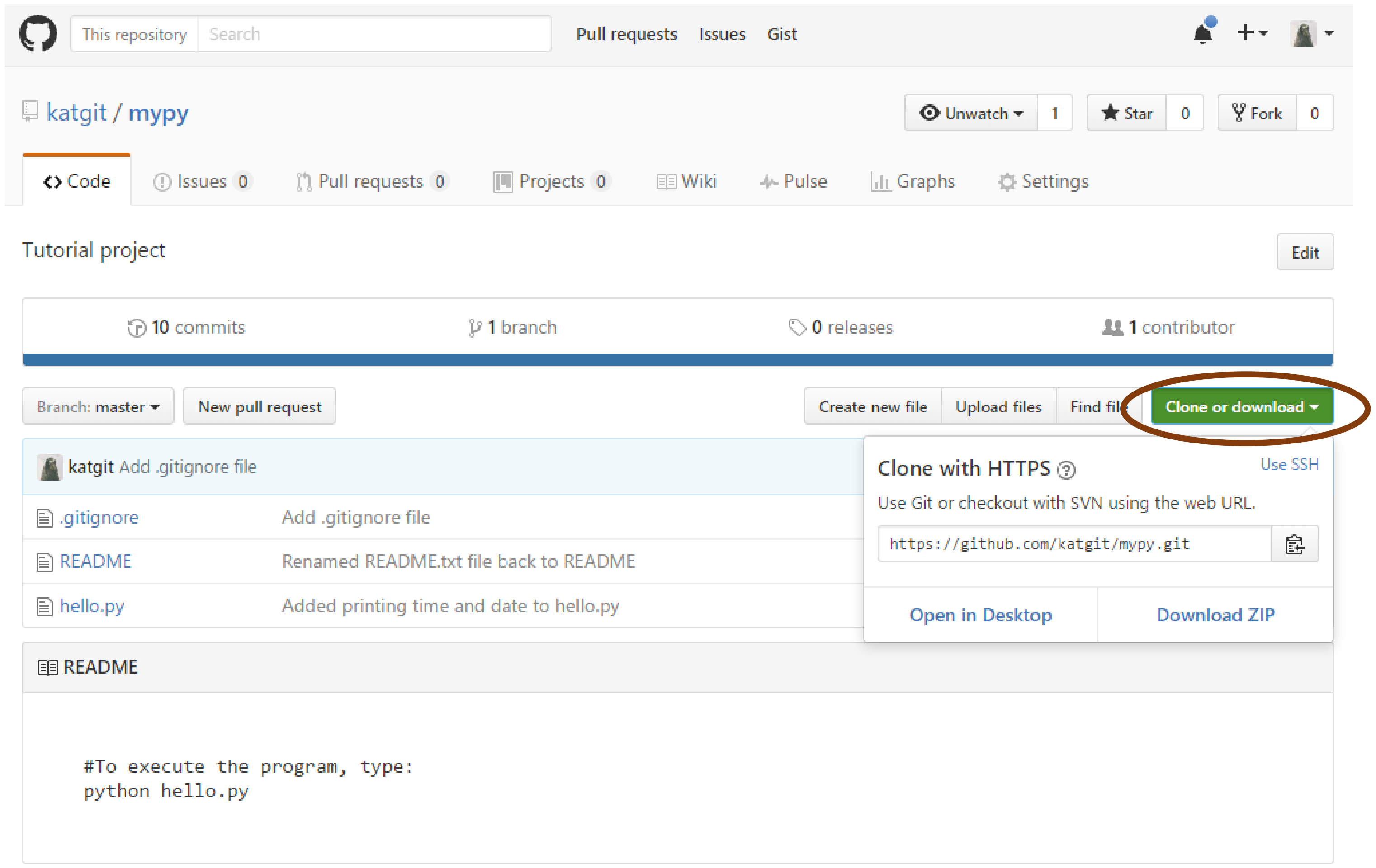
.gitignore	Add .gitignore file	24 minutes ago
README	Renamed README.txt file back to README	an hour ago
hello.py	Added printing time and date to hello.py	5 hours ago

README

```
#To execute the program, type:
python hello.py
```

Cloning remote repository

Clone Remote repository



This repository Search

Pull requests Issues Gist

katgit / mypy

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Tutorial project

Edit

10 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find fil Clone or download

katgit Add .gitignore file

.gitignore	Add .gitignore file
README	Renamed README.txt file back to README
hello.py	Added printing time and date to hello.py

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

```
https://github.com/katgit/mypy.git
```

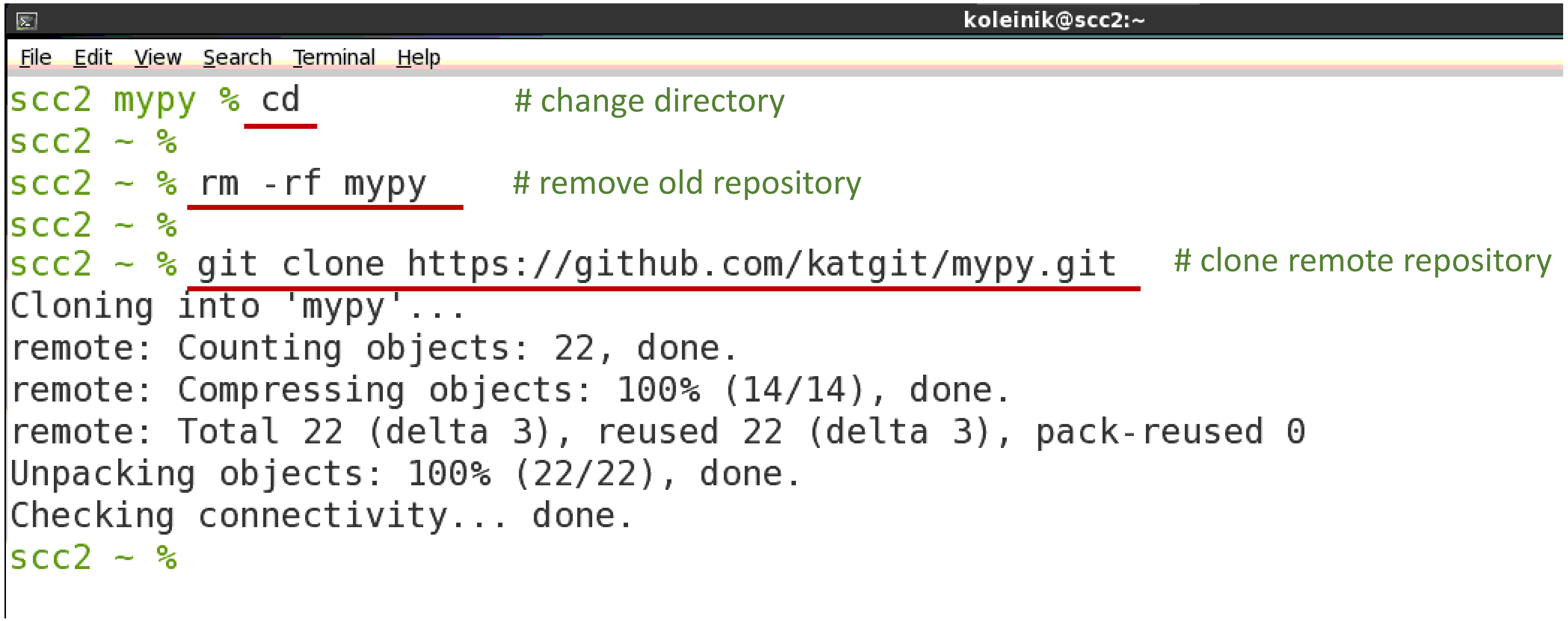
Open in Desktop Download ZIP

README

```
#To execute the program, type:
python hello.py
```

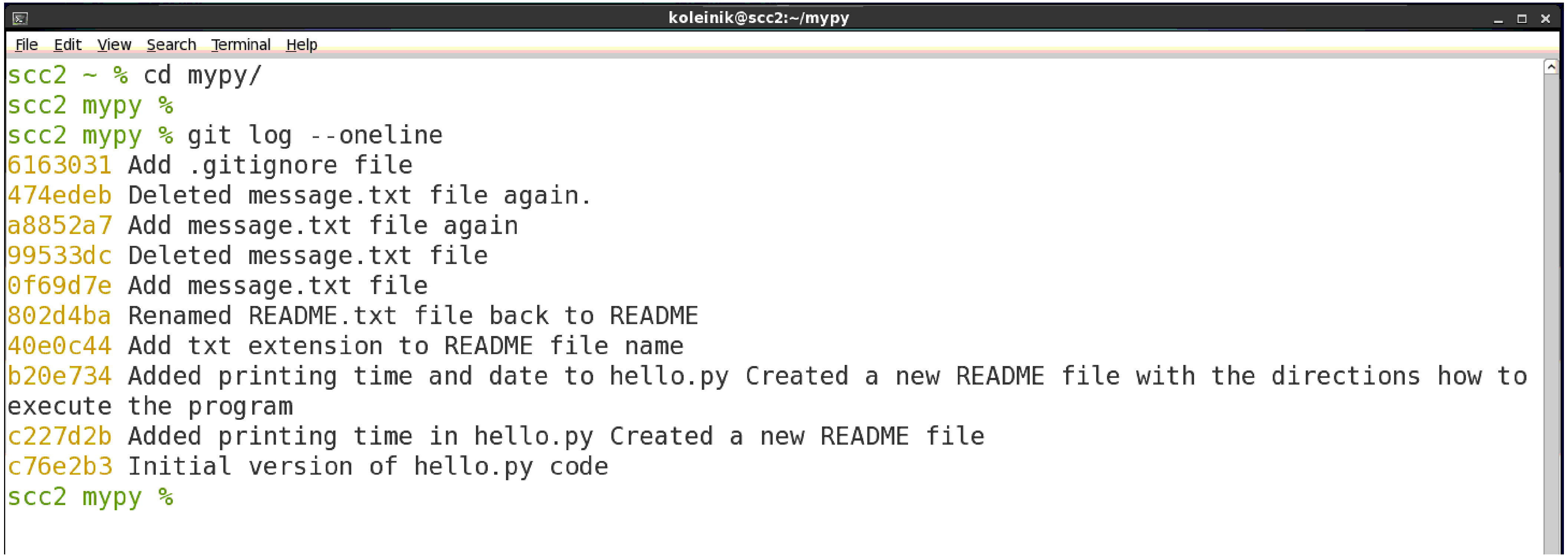
BOSTON UNIVERSITY

Clone Remote repository



```
scc2 mypy % cd                      # change directory
scc2 ~ %
scc2 ~ % rm -rf mypy                # remove old repository
scc2 ~ %
scc2 ~ % git clone https://github.com/katgit/mypy.git    # clone remote repository
Cloning into 'mypy'...
remote: Counting objects: 22, done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 22 (delta 3), reused 22 (delta 3), pack-reused 0
Unpacking objects: 100% (22/22), done.
Checking connectivity... done.
scc2 ~ %
```

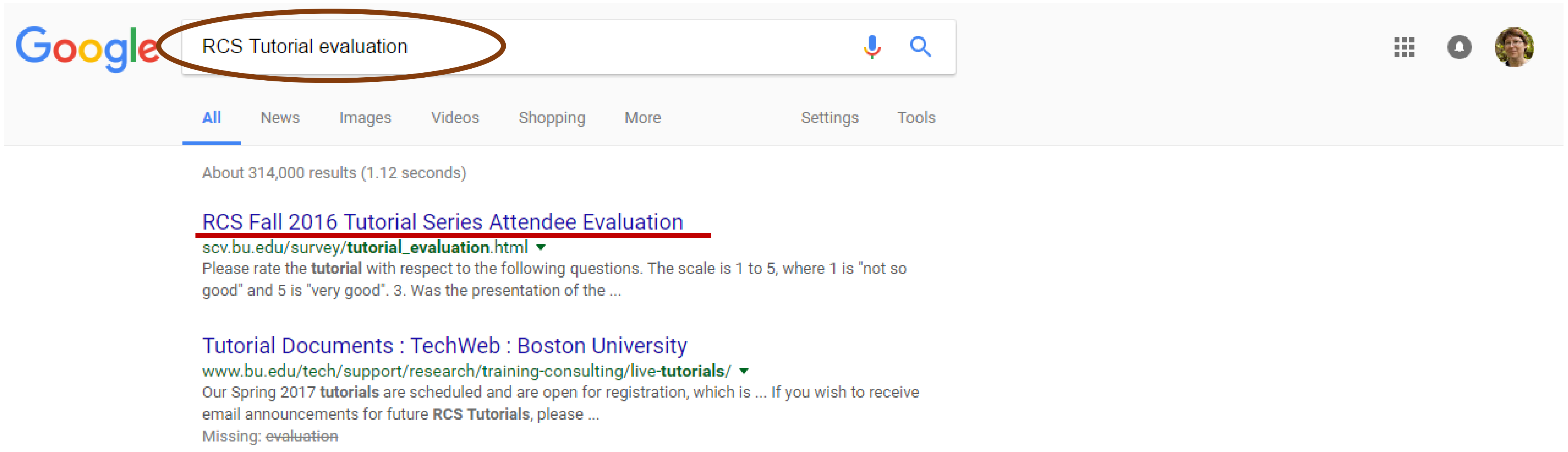
Clone Remote repository



```
scc2 ~ % cd mypy/
scc2 mypy %
scc2 mypy % git log --oneline
6163031 Add .gitignore file
474edeb Deleted message.txt file again.
a8852a7 Add message.txt file again
99533dc Deleted message.txt file
0f69d7e Add message.txt file
802d4ba Renamed README.txt file back to README
40e0c44 Add txt extension to README file name
b20e734 Added printing time and date to hello.py Created a new README file with the directions how to
execute the program
c227d2b Added printing time in hello.py Created a new README file
c76e2b3 Initial version of hello.py code
scc2 mypy %
```

Thank you!

Please, fill out evaluation:



RCS Tutorial evaluation

About 314,000 results (1.12 seconds)

RCS Fall 2016 Tutorial Series Attendee Evaluation
scv.bu.edu/survey/**tutorial_evaluation**.html
Please rate the **tutorial** with respect to the following questions. The scale is 1 to 5, where 1 is "not so good" and 5 is "very good". 3. Was the presentation of the ...

Tutorial Documents : TechWeb : Boston University
www.bu.edu/tech/support/research/training-consulting/live-**tutorials**/
Our Spring 2017 **tutorials** are scheduled and are open for registration, which is ... If you wish to receive email announcements for future **RCS Tutorials**, please ...
Missing: ~~evaluation~~

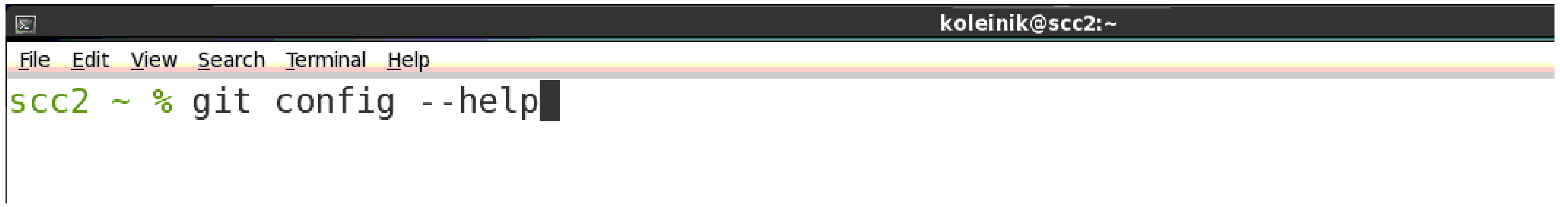
BOSTON UNIVERSITY

Apendix

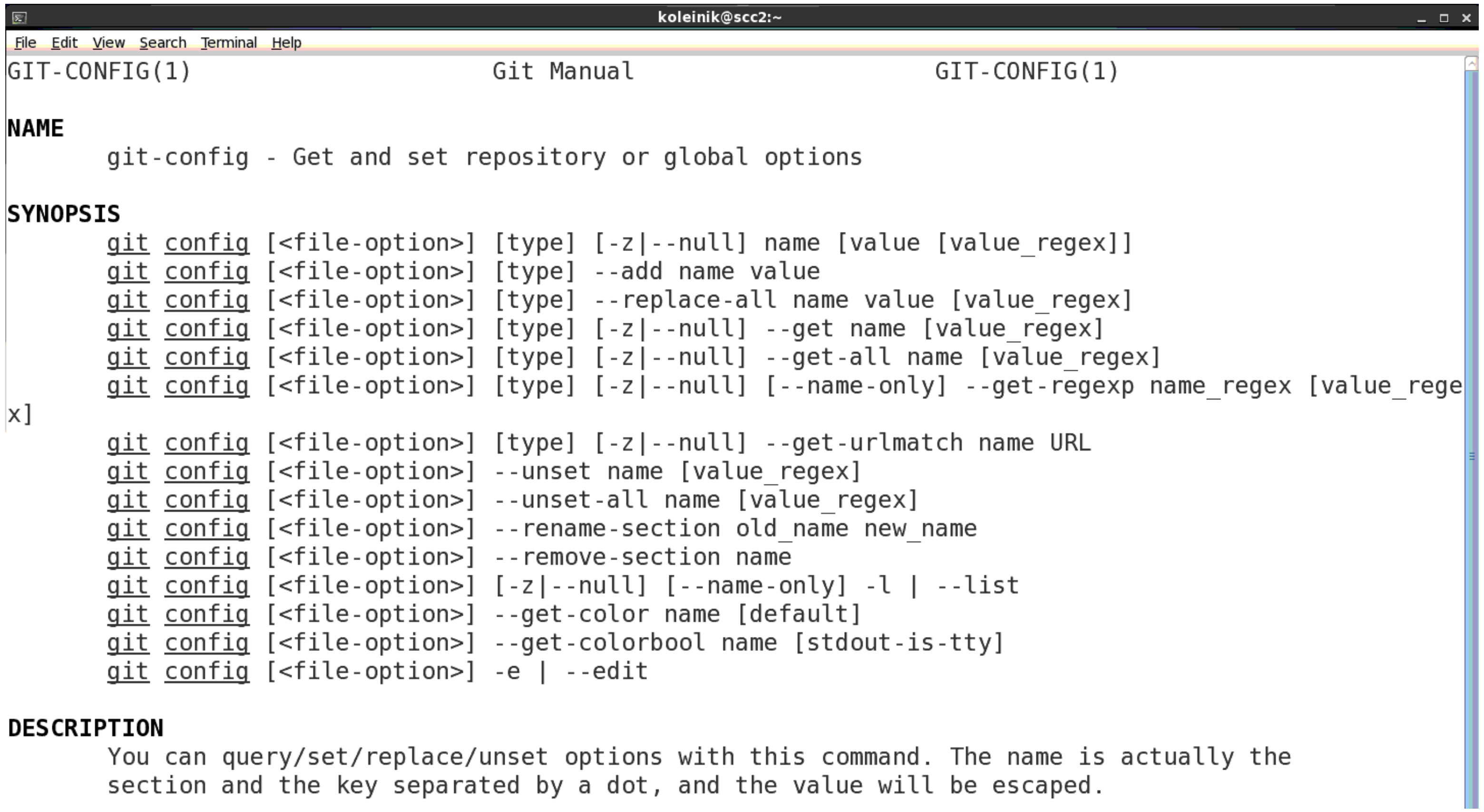
Git help

```
scc2 ~ % git help
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status
```

Git help



```
scc2 ~ % git config --help
```



```
GIT-CONFIG(1)                     Git Manual                     GIT-CONFIG(1)

NAME
       git-config - Get and set repository or global options

SYNOPSIS
       git config [<file-option>] [type] [-z|--null] name [value [value_regex]]
       git config [<file-option>] [type] --add name value
       git config [<file-option>] [type] --replace-all name value [value_regex]
       git config [<file-option>] [type] [-z|--null] --get name [value_regex]
       git config [<file-option>] [type] [-z|--null] --get-all name [value_regex]
       git config [<file-option>] [type] [-z|--null] [--name-only] --get-regexp name_regex [value_rege
x]
       git config [<file-option>] [type] [-z|--null] --get-urlmatch name URL
       git config [<file-option>] --unset name [value_regex]
       git config [<file-option>] --unset-all name [value_regex]
       git config [<file-option>] --rename-section old_name new_name
       git config [<file-option>] --remove-section name
       git config [<file-option>] [-z|--null] [--name-only] -l | --list
       git config [<file-option>] --get-color name [default]
       git config [<file-option>] --get-colorbool name [stdout-is-tty]
       git config [<file-option>] -e | --edit

DESCRIPTION
       You can query/set/replace/unset options with this command. The name is actually the
       section and the key separated by a dot, and the value will be escaped.
```

Git resources

Git official manual:
https://git-scm.com/documentation

Easy online tutorial by GitHub:
https://try.github.io

Git Immersion (popular Git tutorial):
http://gitimmersion.com/

Git docs on many languages:
http://www-cs-students.stanford.edu/~blynn/gitmagic/