CSE4006: Software Engineering

Lab 1: Git

Software Engineering Lab

Install

OS X

- http://sourceforge.net/projects/git-osx-installer/

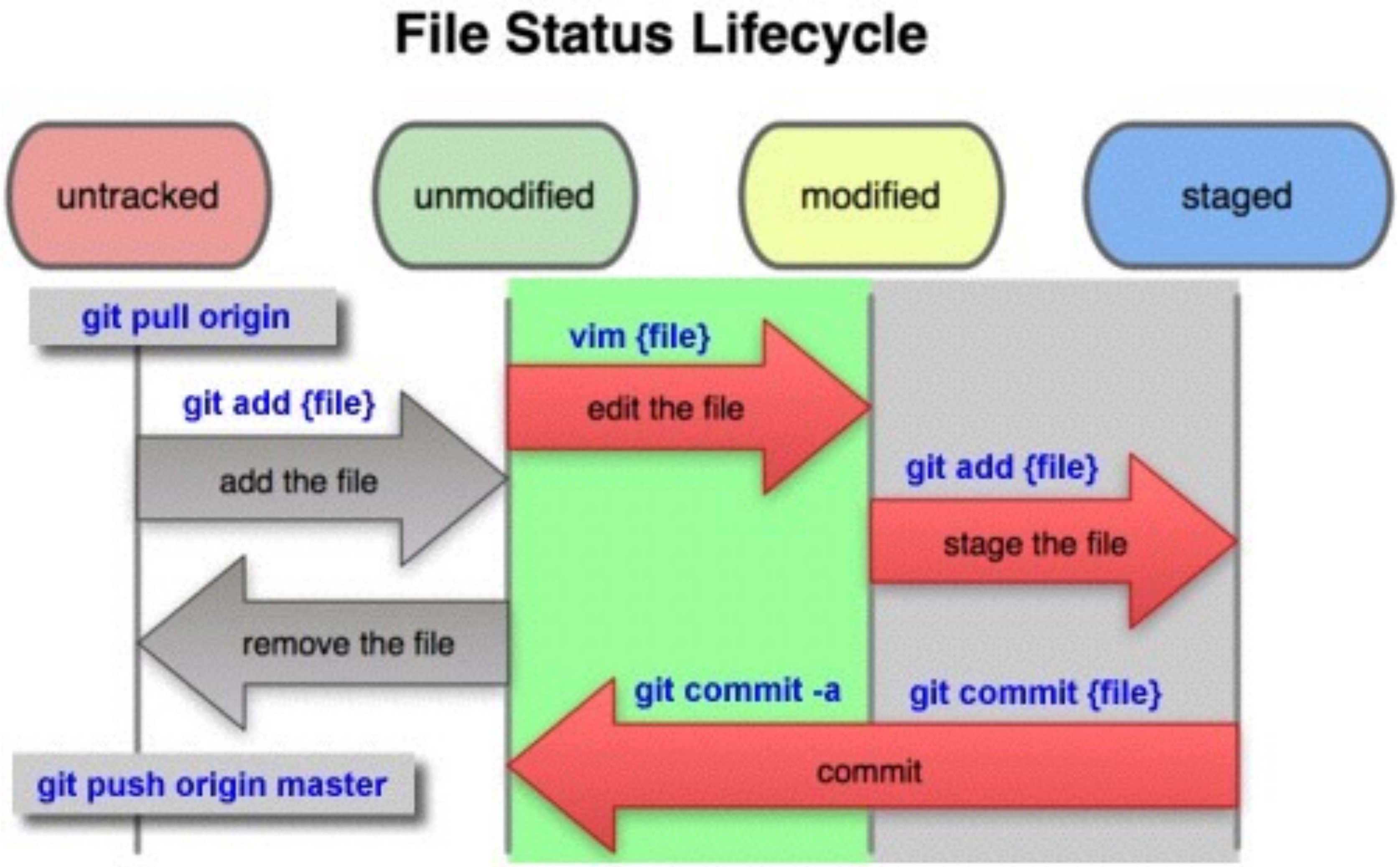
Windows

- http://msysgit.github.com/

Linux

- $ apt-get install git

File Status Lifecycle of Git



User Information

Configure user name and email address

* You cannot change information after commit

```
  $ git config –global user.name “My Name”
$ git config –global user.email “my@email.com”
```

Configure default editor

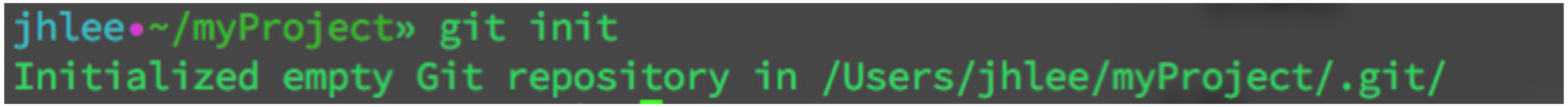
* recommend configuring the most comfortable text editor for you

```
$ git config –global core.editor vi
```

New Repository

Create a new folder and repository

```
$ mkdir myProject
$ cd myProject
$ git init
```

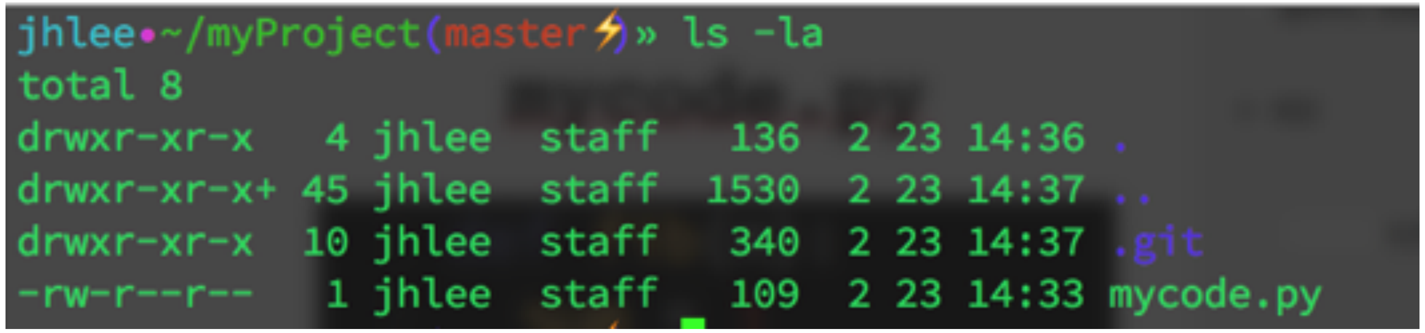


```
jhlee•~/myProject» git init
Initialized empty Git repository in /Users/jhlee/myProject/.git/
```

Create a New File

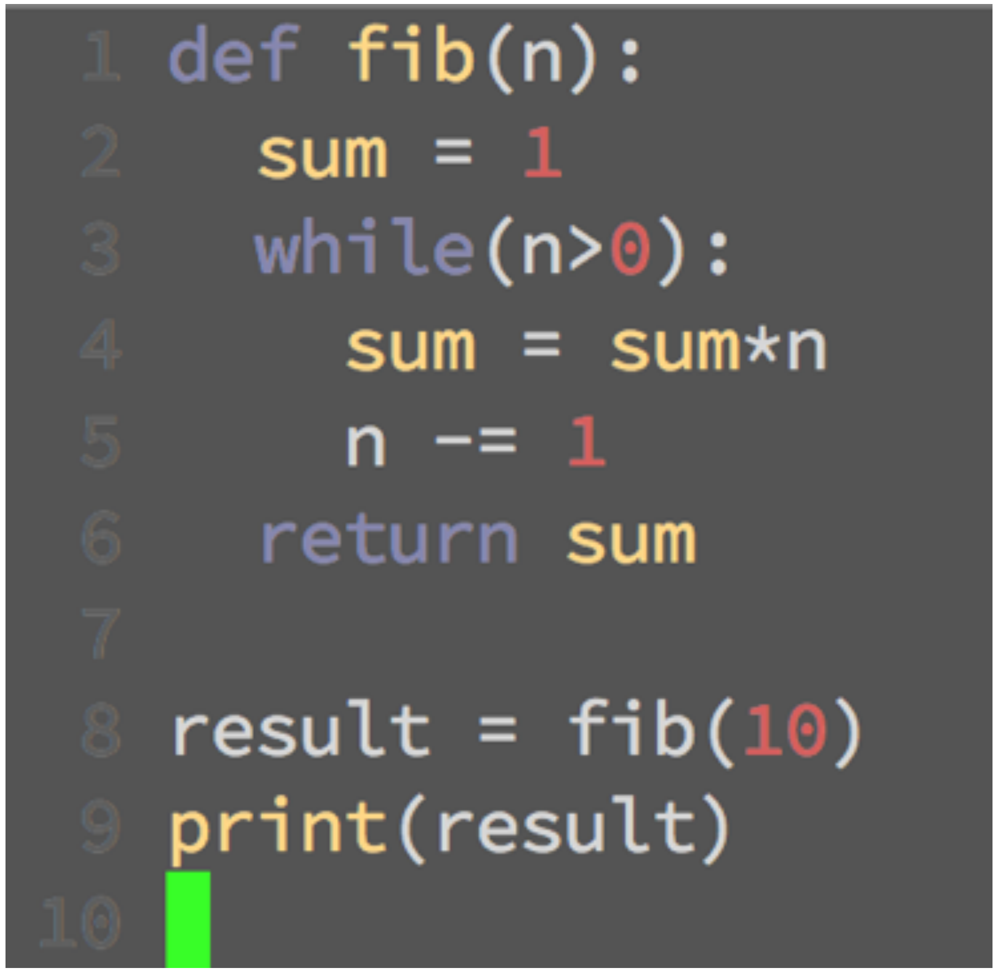
Create a new file, ‘mycode.py’, which is a Fibonacci program

```
$ vi mycode.py
```



```
jhlee•~/myProject(master⚡)» ls -la
total 8
drwxr-xr-x   4 jhlee  staff   136  2 23 14:36 .
drwxr-xr-x+ 45 jhlee  staff  1530  2 23 14:37 ..
drwxr-xr-x  10 jhlee  staff   340  2 23 14:37 .git
-rw-r--r--   1 jhlee  staff   109  2 23 14:33 mycode.py
```

mycode.py



```
def fib(n):
  sum = 1
  while(n>0):
    sum = sum*n
    n -= 1
  return sum

result = fib(10)
print(result)
```

Add and Commit Files

Add the **created** file,
‘mycode.py’, and **track** it

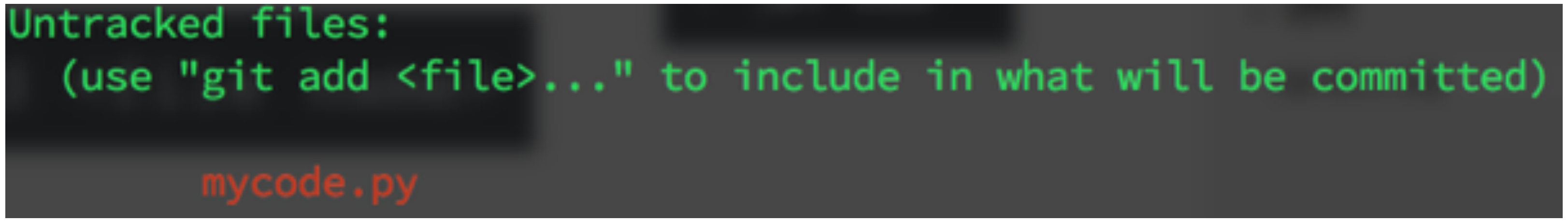
```
$ git add <file name>
```

Add all files

```
$ git add *
```

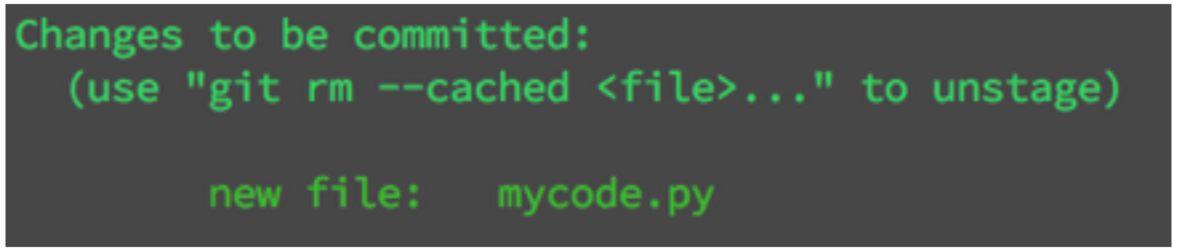
Check git repository status

```
$ git status
```



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

        mycode.py
```

↓



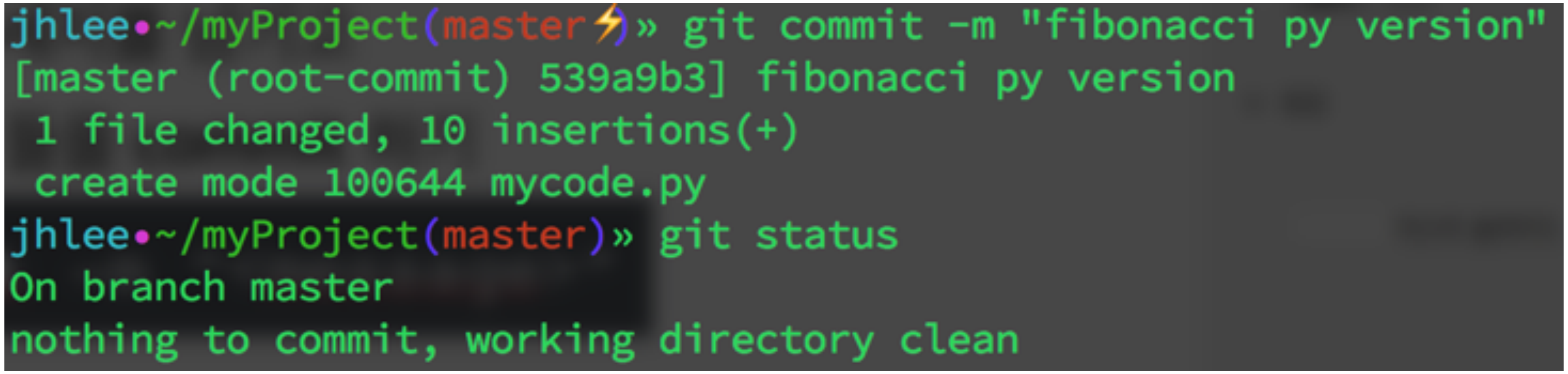
```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   mycode.py
```

Add and Commit Files

Write commit message

and commit added files

```
$ git commit -m “<message>”
```

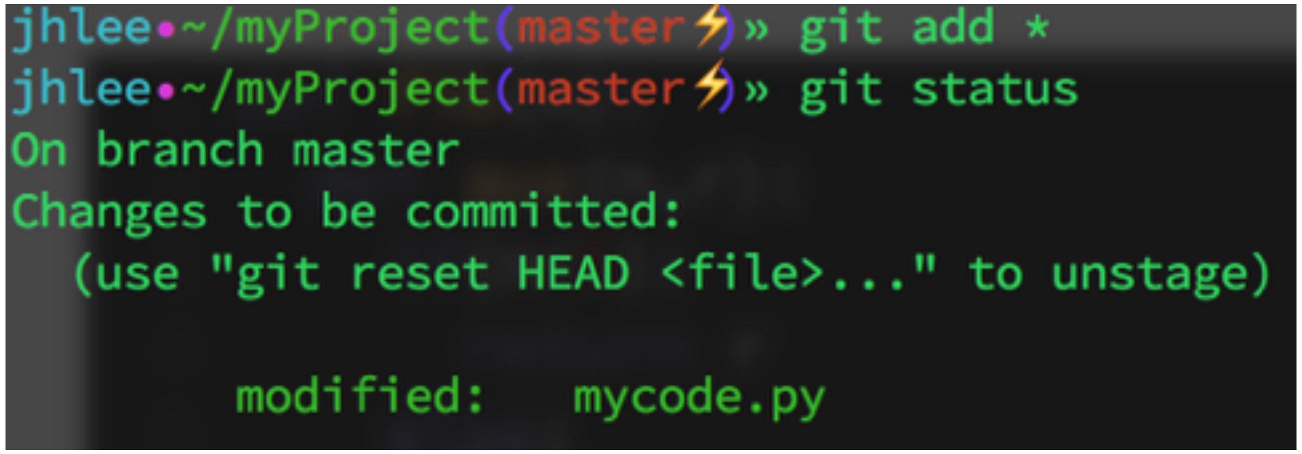


```
jhlee•~/myProject(master⚡)» git commit -m "fibonacci py version"
[master (root-commit) 539a9b3] fibonacci py version
 1 file changed, 10 insertions(+)
 create mode 100644 mycode.py
jhlee•~/myProject(master)» git status
On branch master
nothing to commit, working directory clean
```

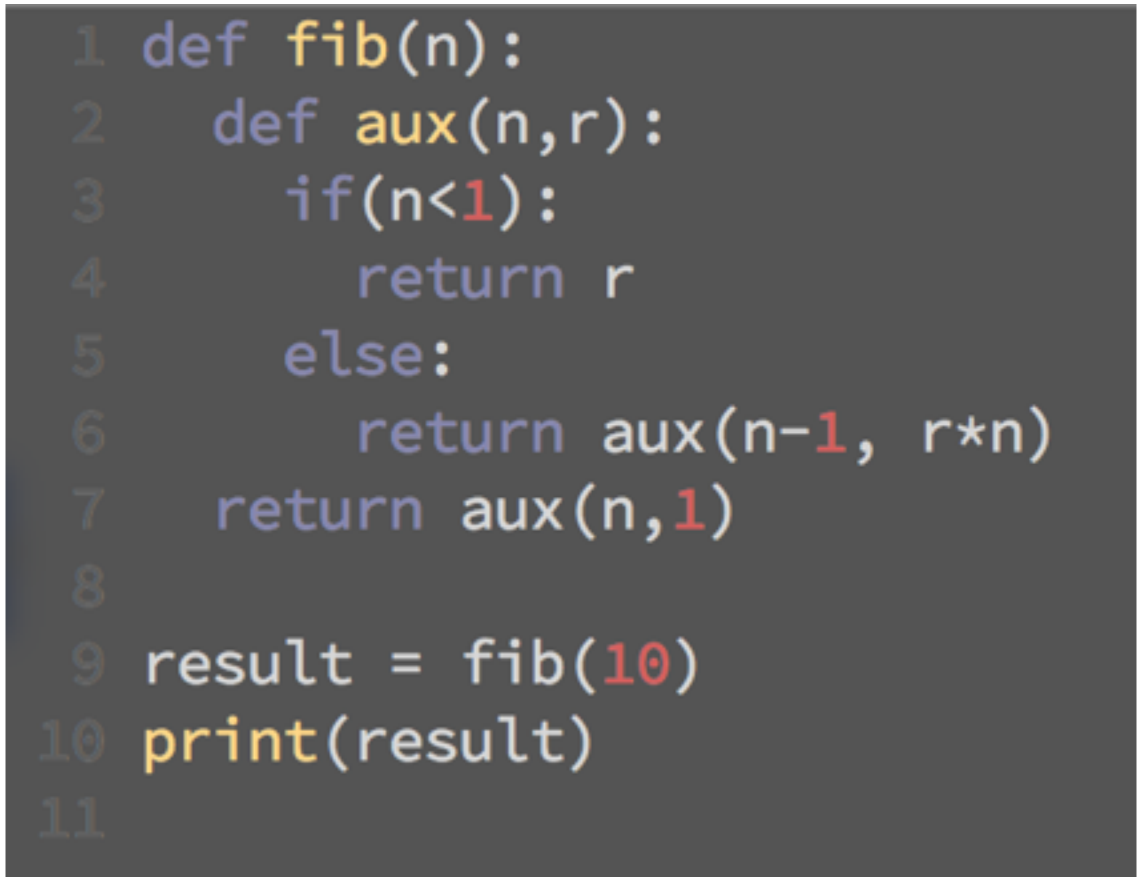
Edit Files

Change the ‘mycode.py’
(Fibonacci program to tail recursion)
and add it to git

```
$ vi mycode.py
$ git add *
```



```
jhlee•~/myProject(master⚡)» git add *
jhlee•~/myProject(master⚡)» git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   mycode.py
```

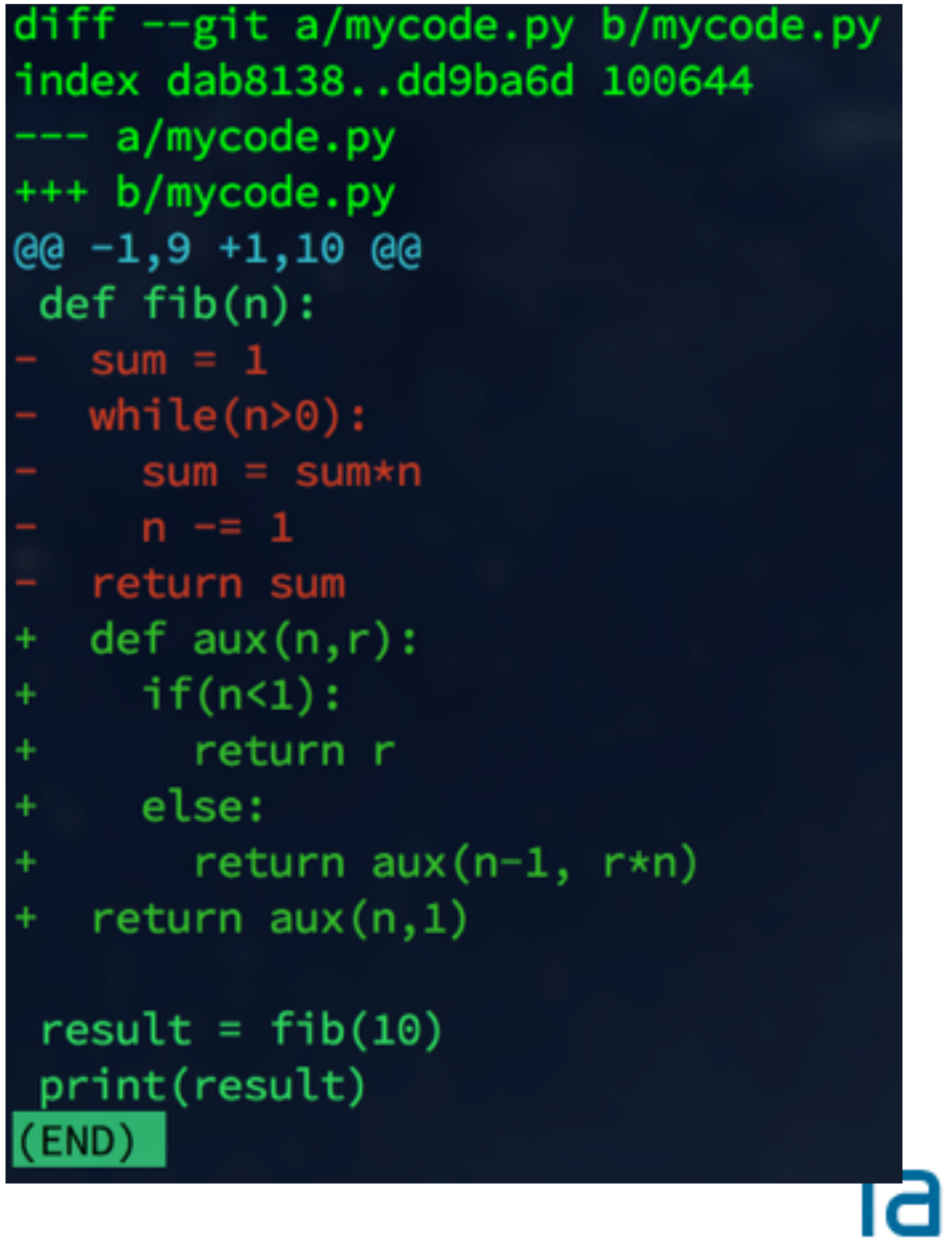


```
def fib(n):
  def aux(n,r):
    if(n<1):
      return r
    else:
      return aux(n-1, r*n)
  return aux(n,1)

result = fib(10)
print(result)

```

View Changes

Compare the added file with the committed file

```
$ git diff --staged
```



```
diff --git a/mycode.py b/mycode.py
index dab8138..dd9ba6d 100644
--- a/mycode.py
+++ b/mycode.py
@@ -1,9 +1,10 @@
 def fib(n):
-  sum = 1
-  while(n>0):
-    sum = sum*n
-    n -= 1
-  return sum
+  def aux(n,r):
+    if(n<1):
+      return r
+    else:
+      return aux(n-1, r*n)
+  return aux(n,1)

 result = fib(10)
 print(result)
(END)
```

Cancel Added Files

Create and add a text file, ‘todo.txt’

```
$ vi todo.txt
$ git add *
```

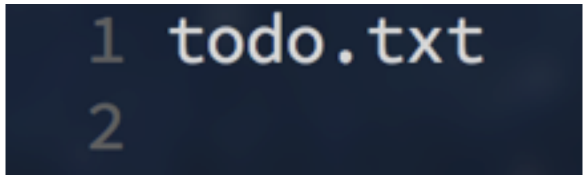
Remove a specific file in the staged state

```
$ git rm --cached todo.txt
```

Ignore Files

Create ‘.gitignore’ and
add the name rules for files to be ignored

```
$ vi .gitignore
```



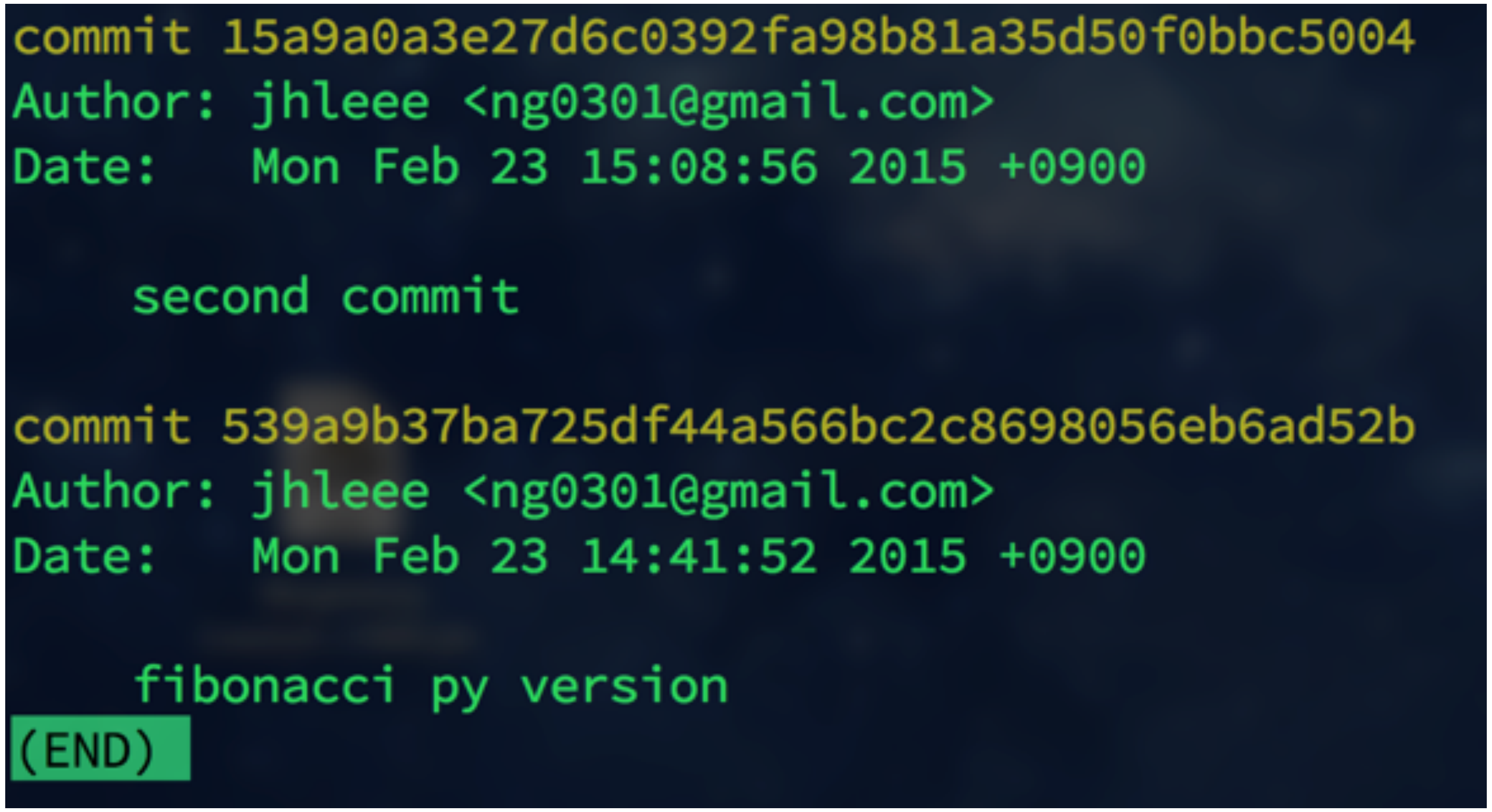
```
todo.txt

```

Check that it works

```
$ git add *
$ git status
```

View commit logs

```
$ git log
```



```
commit 15a9a0a3e27d6c0392fa98b81a35d50f0bbc5004
Author: jhleee <ng0301@gmail.com>
Date:   Mon Feb 23 15:08:56 2015 +0900

    second commit

commit 539a9b37ba725df44a566bc2c8698056eb6ad52b
Author: jhleee <ng0301@gmail.com>
Date:   Mon Feb 23 14:41:52 2015 +0900

    fibonacci py version
(END)
```

Commit Missing Files

If you want to add missing file
or change the commit message

```
$ git commit --amend
```

These three lines are considered as one commit

```
$ git commit -m “asdf”
$ git add forgotten_file
$ git commit -m “I forgot”
```

Turn Back to the Recent Commit

If you want to turn a specific file
or all of files back to the recent commit

```
$ git checkout <filename>
$ git checkout
```

Change Files and Remove

Remove a file

```
$ git rm <filename>
```

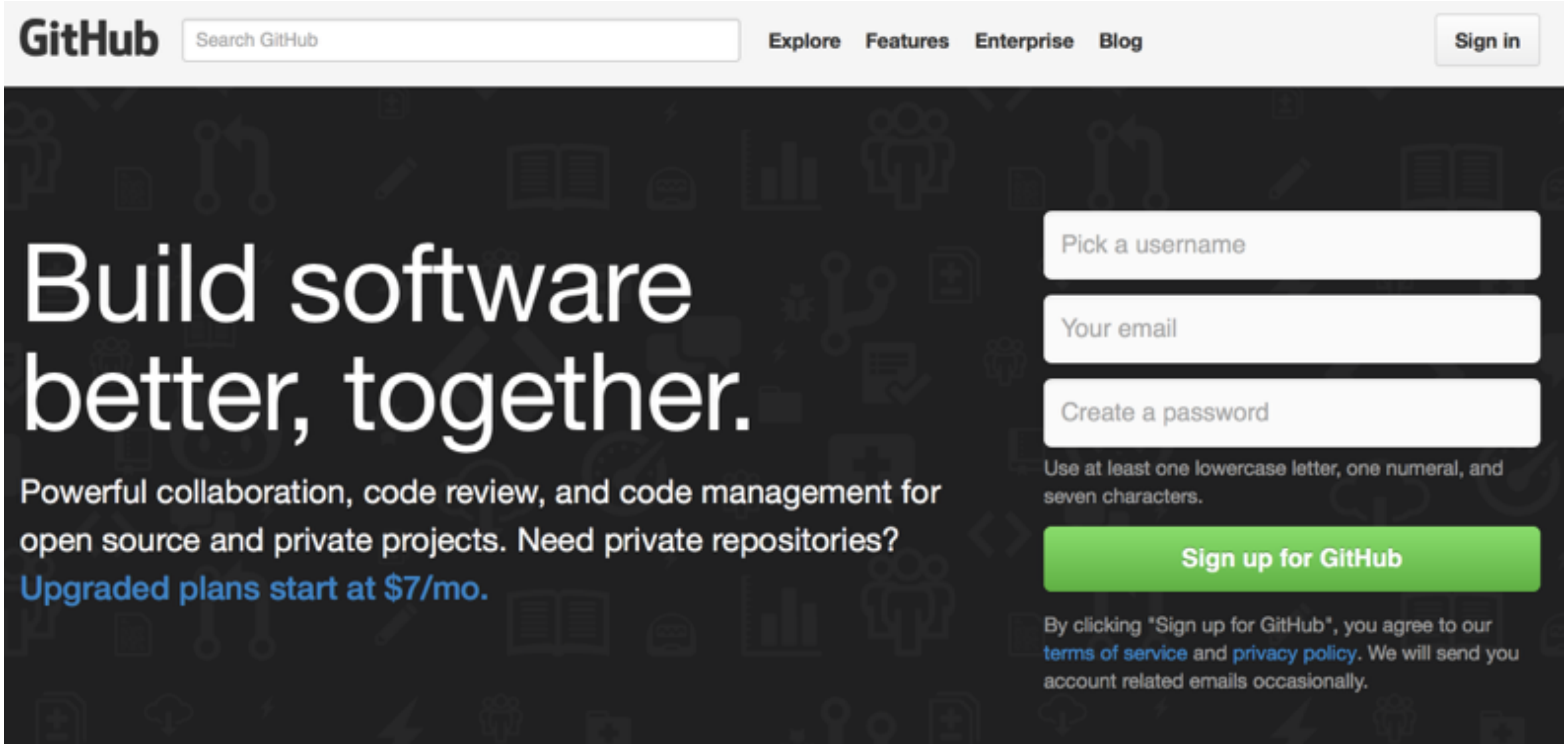
Change file name

```
$ git mv <filename> <new_name>
```

*   If you change file name without using this command,
git considers the existing file is removed and the new file is created

Create a Remote Repository

Sign in https://github.com



Build software better, together.

Powerful collaboration, code review, and code management for open source and private projects. Need private repositories? Upgraded plans start at $7/mo.

Pick a username

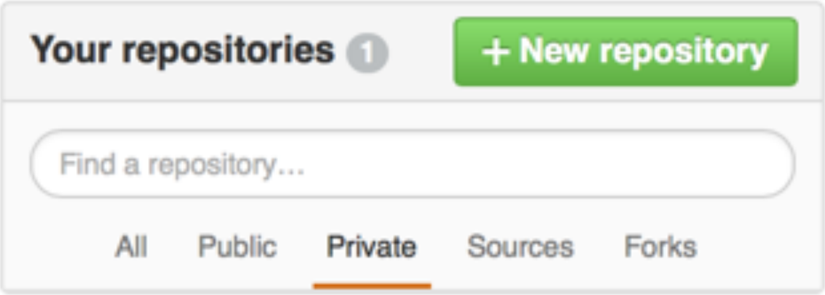
Your email

Create a password

Use at least one lowercase letter, one numeral, and seven characters.

By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We will send you account related emails occasionally.

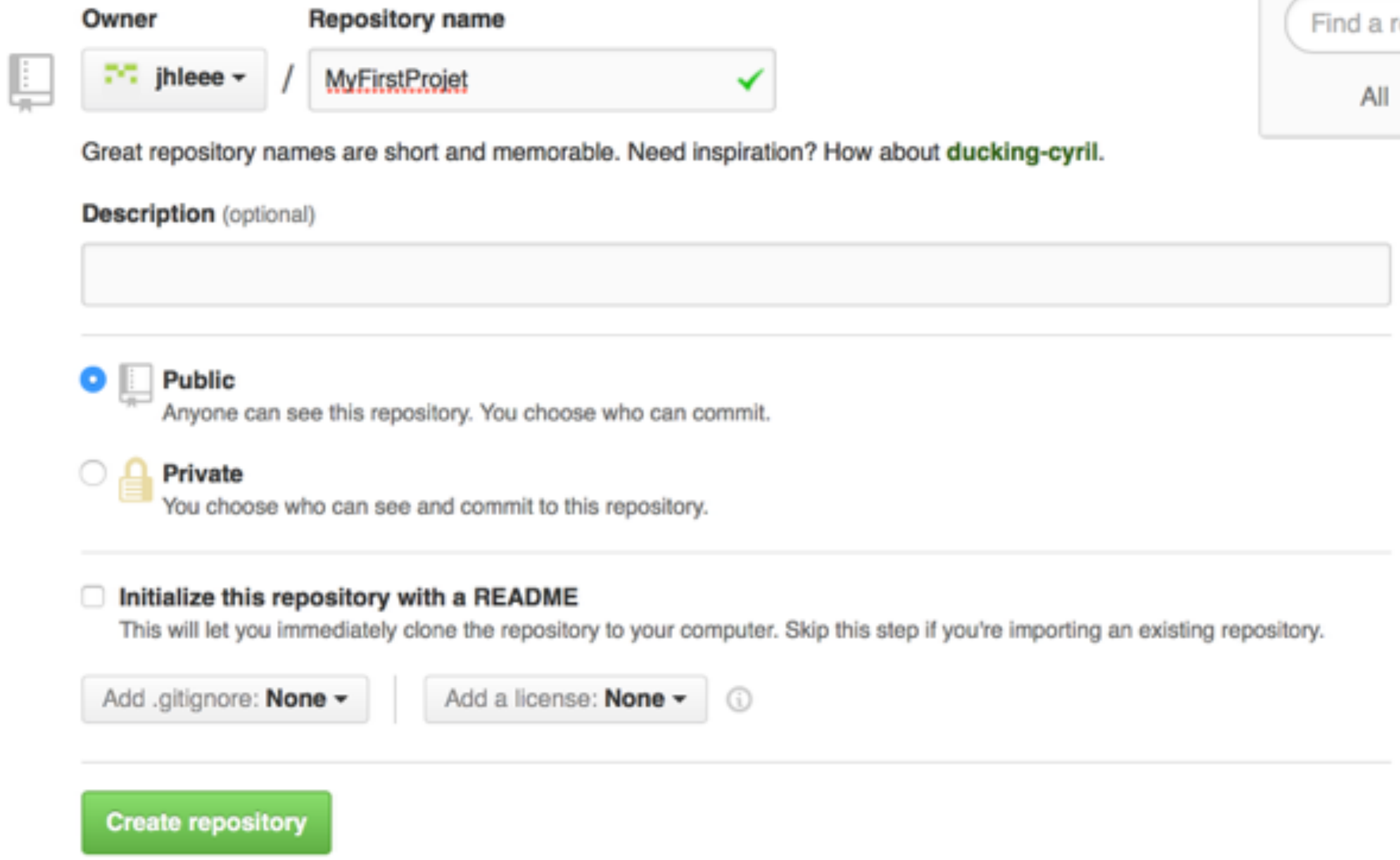
Create a New Repository



Your repositories 1 + New repository

Find a repository…

All Public Private Sources Forks



Owner: jhleee / Repository name: MyFirstProjet ✓

Great repository names are short and memorable. Need inspiration? How about **ducking-cyril**.

Description (optional)

● **Public**
Anyone can see this repository. You choose who can commit.

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

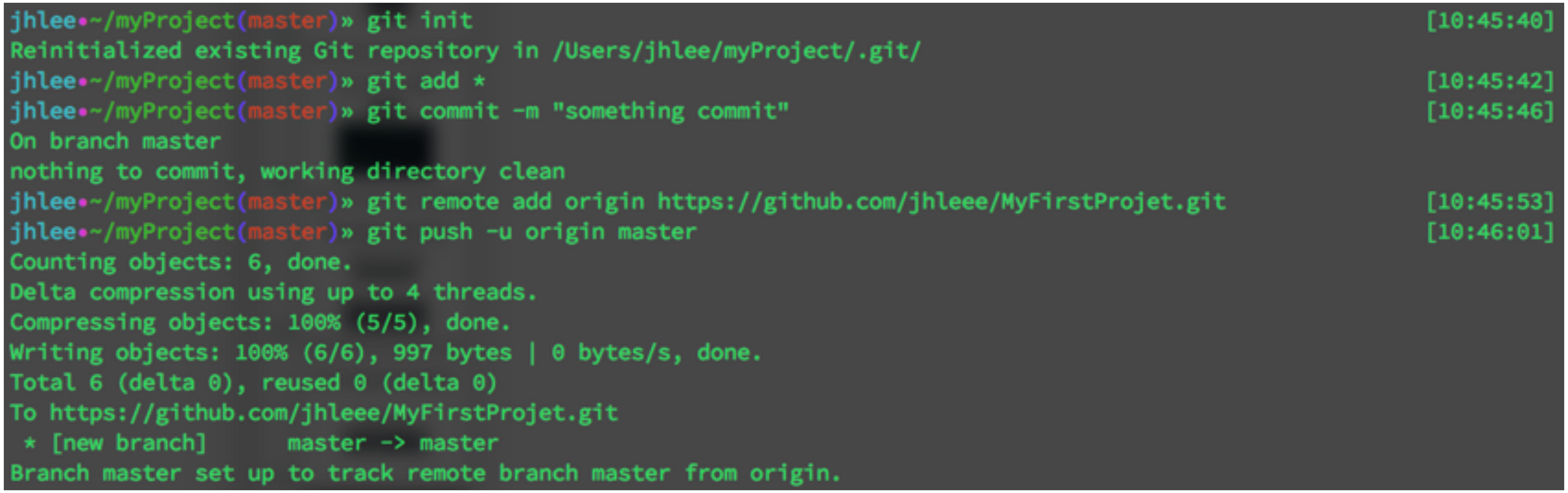
□ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Create repository

Connect to Remote Repository

```
$ git init
$ git add <files>
$ git commit -m “message”
$ git remote add origin <URL>
$ git push -u origin master
```



```
jhlee•~/myProject(master)» git init                                                                      [10:45:40]
Reinitialized existing Git repository in /Users/jhlee/myProject/.git/
jhlee•~/myProject(master)» git add *                                                                     [10:45:42]
jhlee•~/myProject(master)» git commit -m "something commit"                                              [10:45:46]
On branch master
nothing to commit, working directory clean
jhlee•~/myProject(master)» git remote add origin https://github.com/jhleee/MyFirstProjet.git             [10:45:53]
jhlee•~/myProject(master)» git push -u origin master                                                     [10:46:01]
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 997 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/jhleee/MyFirstProjet.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Get Files From Remote Repository

Clone files from repository of the remote server

```
$ git clone <URL>
```

Push and Pull at Remote Repository

Synchronize local repository with remote repository

```
$ git pull
```

Push updates in local to remote server

*If you want to push to other branch input the branch name instead master

```
$ git push origin master
```

Create a New Branch and Remove

Create a new branch

```
$ git checkout -b <branch_name>
```

Publish a branch

*Other people can’t access before it’s pushed

```
$ git push origin <branch_name>
```

branch ?

It’s used if you want to work isolatedly. Basically, it’s used after master branch is created. You can work in the other branch and merge it with master branch when it finishes.

Create a New Branch and Remove(cont)

Turn back to master branch

```
$ git checkout master
```

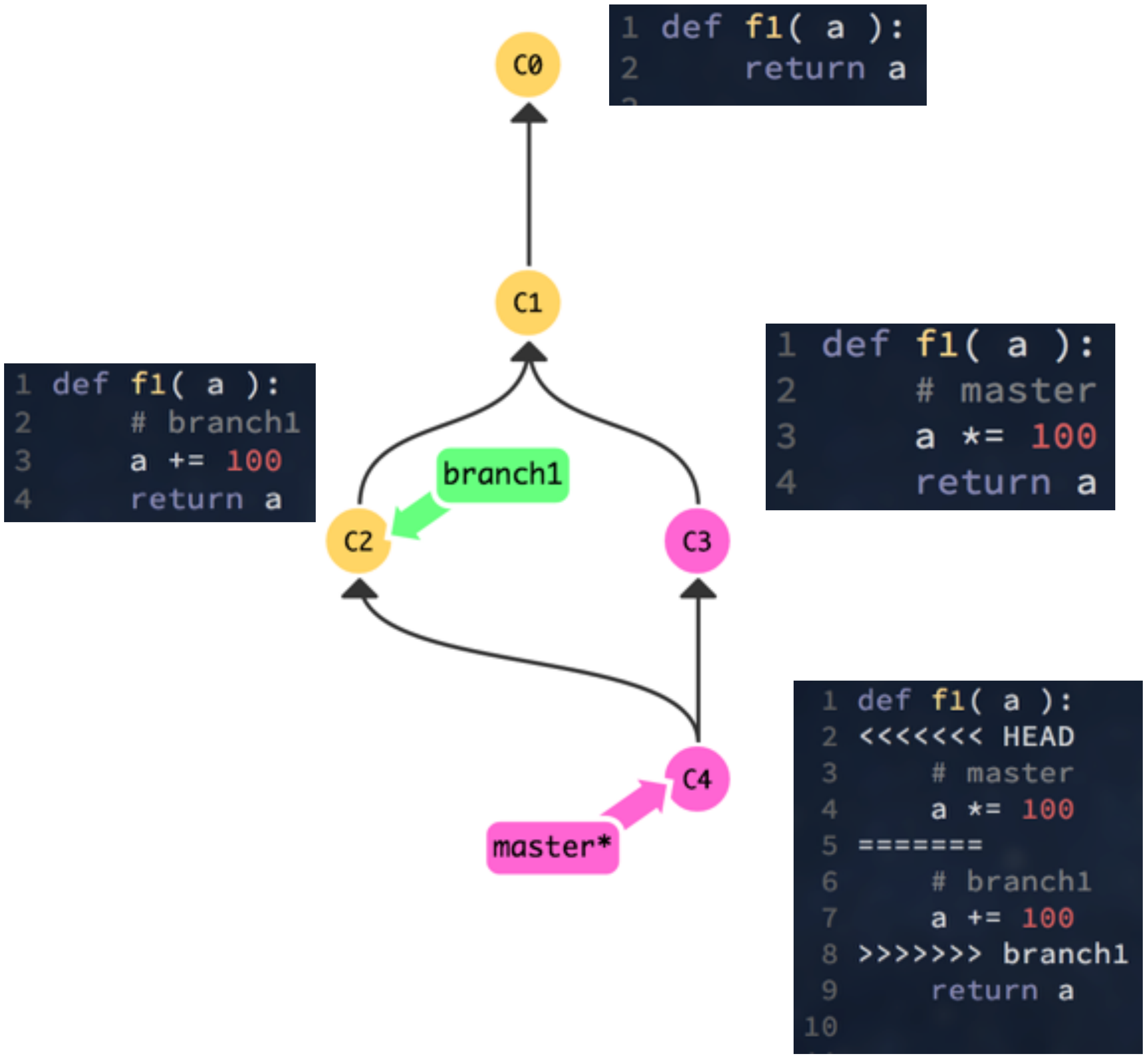
Merge a branch to master branch

```
$ git merge <branch_name>
```

Remove a branch

```
$ git branch -d <branch_name>
```

Resolving a Merge Conflict



```
def f1( a ):
    return a
```

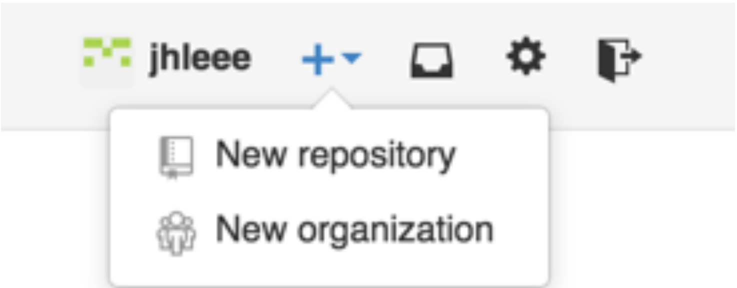
```
def f1( a ):
    # branch1
    a += 100
    return a
```

```
def f1( a ):
    # master
    a *= 100
    return a
```

```
def f1( a ):
<<<<<<< HEAD
    # master
    a *= 100
=======
    # branch1
    a += 100
>>>>>>> branch1
    return a
```

Exercise 1
- Create a New Organization

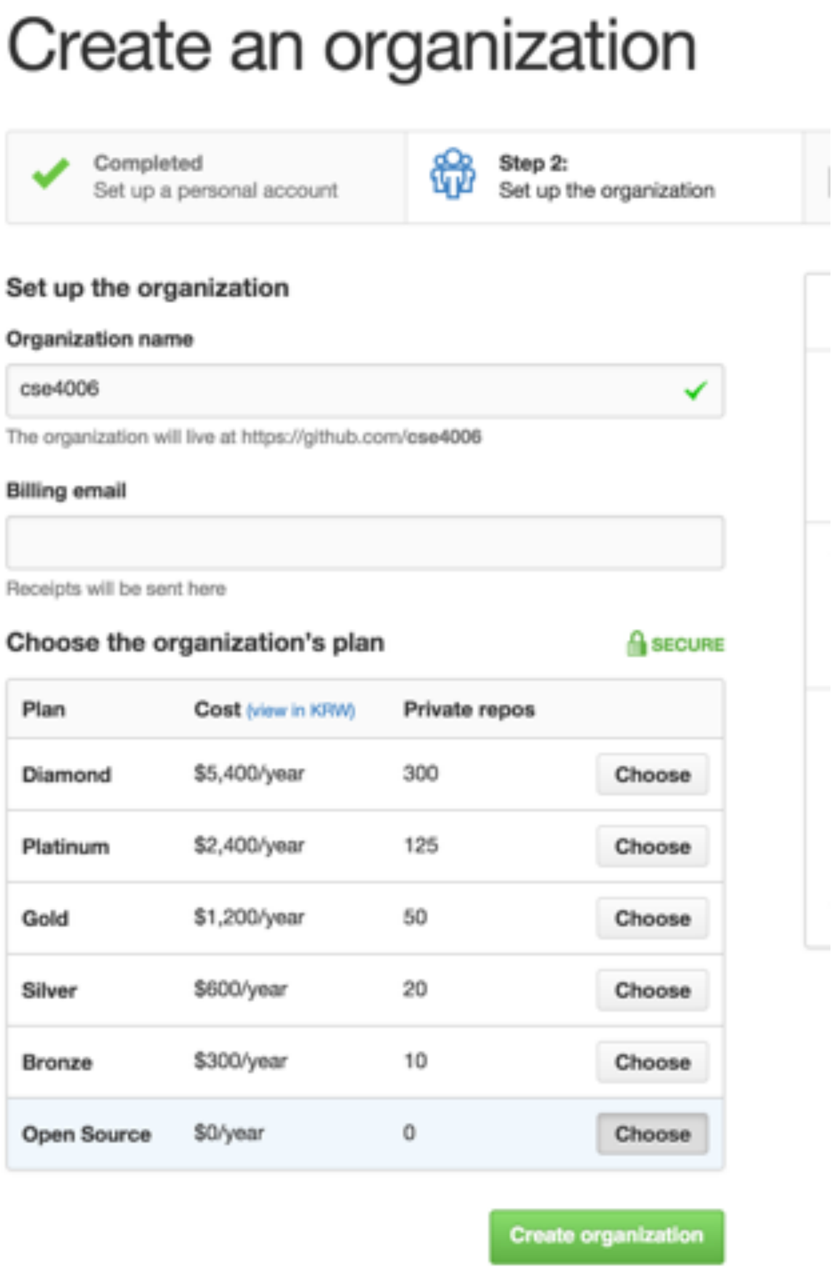
1.



jhleee + ▾

- New repository
- New organization

2.



Create an an organization

✓ Completed
Set up a personal account

Step 2:
Set up the organization

Set up the organization

Organization name

cse4006 ✓

The organization will live at https://github.com/**cse4006**

Billing email

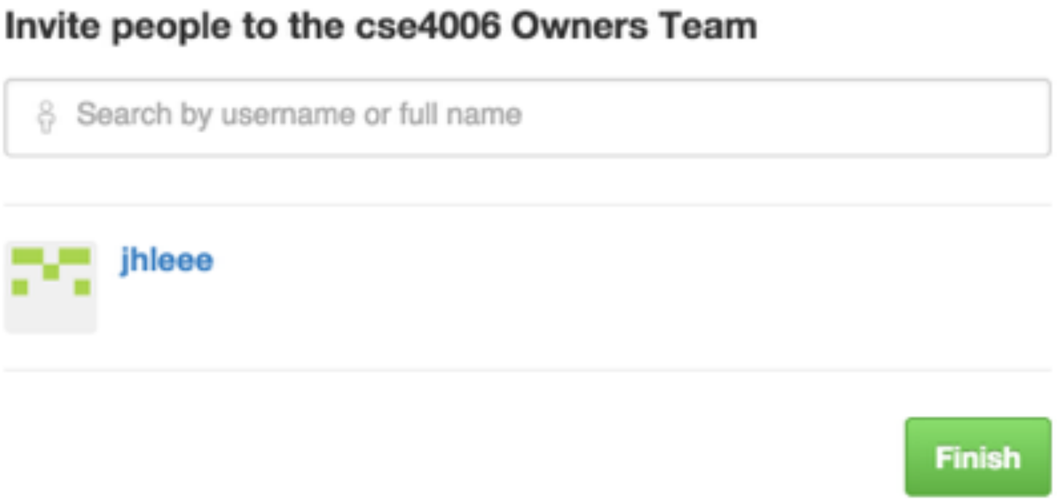
Receipts will be sent here

Choose the organization's plan SECURE

Plan	Cost (view in KRW)	Private repos	
Diamond	$5,400/year	300	Choose
Platinum	$2,400/year	125	Choose
Gold	$1,200/year	50	Choose
Silver	$600/year	20	Choose
Bronze	$300/year	10	Choose
Open Source	$0/year	0	Choose

Create organization

3.



Invite people to the cse4006 Owners Team

Search by username or full name

jhleee

Finish

Exercise 1

1. Create an Organization.

2. Create a file that has at least 3 meaningful function.

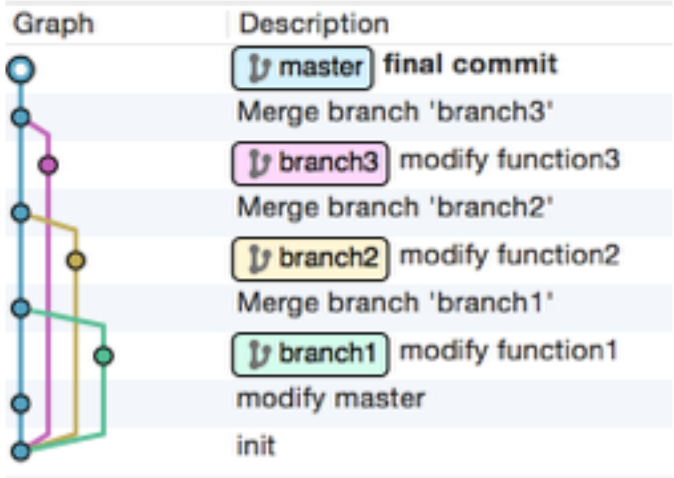
3. Each member create a **branch** from **master**.

4. Modify a function for each **branch**.

5. Add any text to file in master

6. Merge all branch into master.

GUI log viewer



```
Graph    Description
         master  final commit
         Merge branch 'branch3'
         branch3  modify function3
         Merge branch 'branch2'
         branch2  modify function2
         Merge branch 'branch1'
         branch1  modify function1
         modify master
         init
```

Check the log :

```
$ git log --graph --oneline
```

Additional

A cheat sheet of common Git commands

- https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf