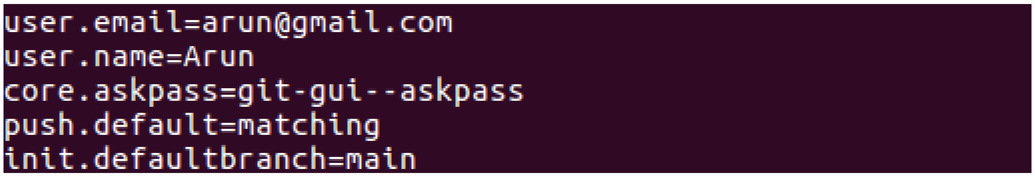
# GIT BASICS

## Set Git Configuration

### Show Present Configuration

**# git config --list**



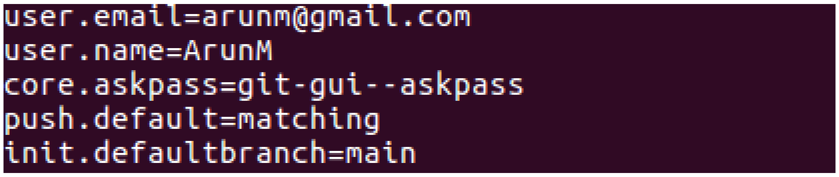
```
user.email=arun@gmail.com
user.name=Arun
core.askpass=git-gui--askpass
push.default=matching
init.defaultbranch=main
```

This will show the current user name, email and default branch that we set. By default we will be in the master branch. Currently its empty. Let us set username and email.

### Set global username and email

**# git config --global user.name  "ArunM"**
**# git config -- global user.email "[arunm@gmail.com](mailto:arunm@gmail.com)"**

Again if we check the config, we can see global user name and email is changed
**#git config --list**



```
user.email=arunm@gmail.com
user.name=ArunM
core.askpass=git-gui--askpass
push.default=matching
init.defaultbranch=main
```

*Note: optional*

### To edit global username and password as a file

**# git config --global --edit**

## Making a Git Commit

Let us create a folder first

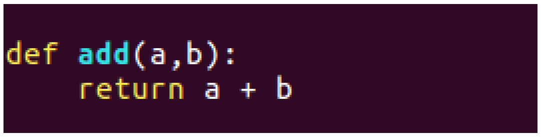
**# mkdir my_project**
**# cd my_project**

**# git init** → This will initialize the present working directory as a new project.

Once we initialized the git, if we check we can see a hidden folder **.git** in our directory. This will keep track of all actions we do in this folder.

Now start coding
**# vim calc.py**

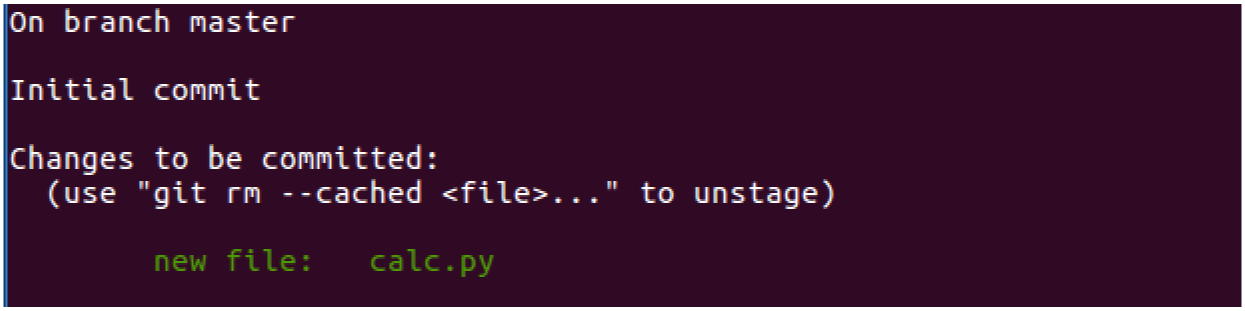
Add following line of code



```
def add(a,b):
    return a + b
```

Now add this file
**# git add calc.py**
Here we moved calc.py to stage area
(Note: To add all the files, we use **#git add .**)

Then check the status
**# git status**

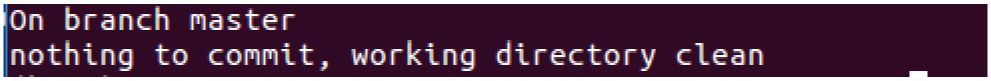


```
On branch master

Initial commit

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

        new file:   calc.py
```

Now we will commit it
**# git commit -m “calc add version”**
Now we have moved calc.py to commit area.

Again check the status

**# git status**

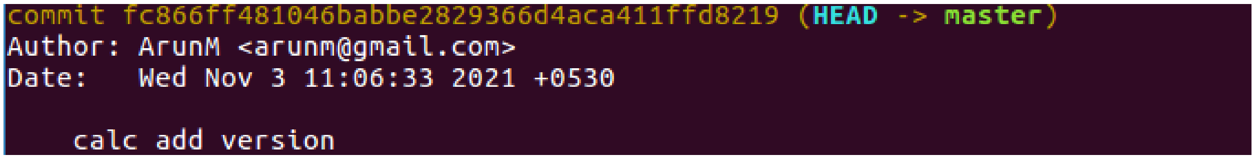


```
On branch master
nothing to commit, working directory clean
```

If we again check git status, we can see that there is no file to commit and status are up to date.

Now look for logs

**# git log --decorate**



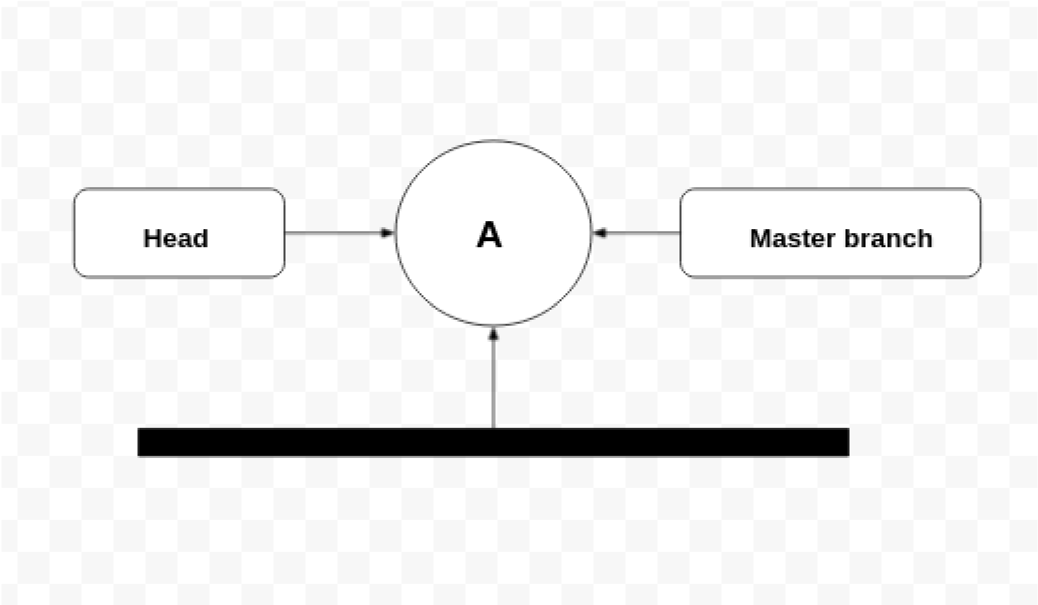
```
commit fc866ff481046babbe2829366d4aca411ffd8219 (HEAD -> master)
Author: ArunM <arunm@gmail.com>
Date:   Wed Nov 3 11:06:33 2021 +0530

    calc add version
```

We can see the commit that we just made. Actually the status of our version control looks as follows:

***Here A be the commit commit id***

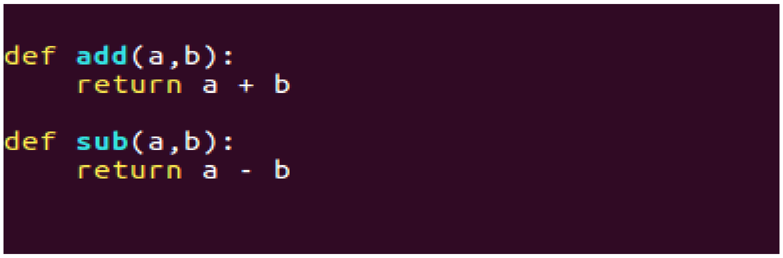
***HEAD corresponds to the active version of file***

***Master is the branch we are working on***



Now just add a subtract functionality to calc.py and commit it.

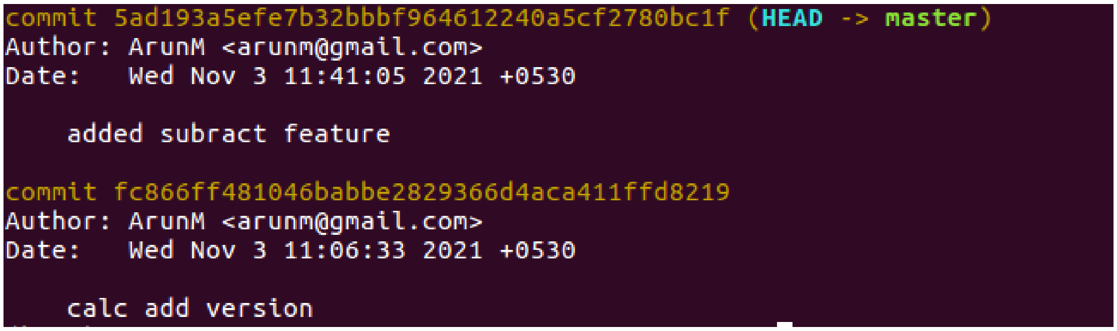
**# vim calc.py**



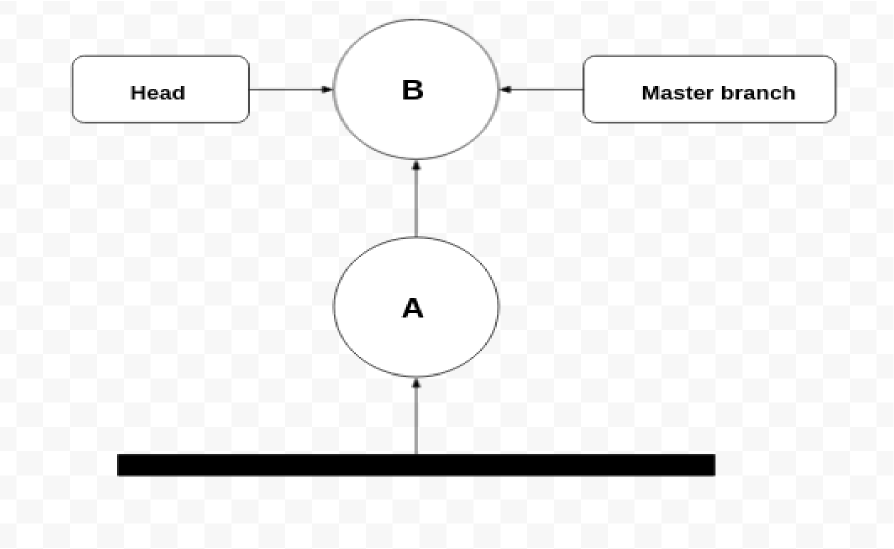
```
def add(a,b):
    return a + b

def sub(a,b):
    return a - b
```

**# git add calc.py**
**# git commit -m “added subtract feature”**

Now look for logs
**# git log --decorate**



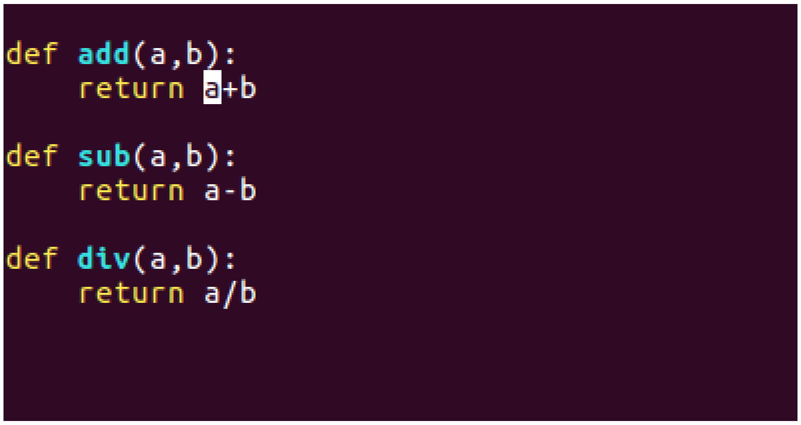
```
commit 5ad193a5efe7b32bbbf964612240a5cf2780bc1f (HEAD -> master)
Author: ArunM <arunm@gmail.com>
Date:   Wed Nov 3 11:41:05 2021 +0530

    added subract feature

commit fc866ff481046babbe2829366d4aca411ffd8219
Author: ArunM <arunm@gmail.com>
Date:   Wed Nov 3 11:06:33 2021 +0530

    calc add version
```



# Go to any previous commit

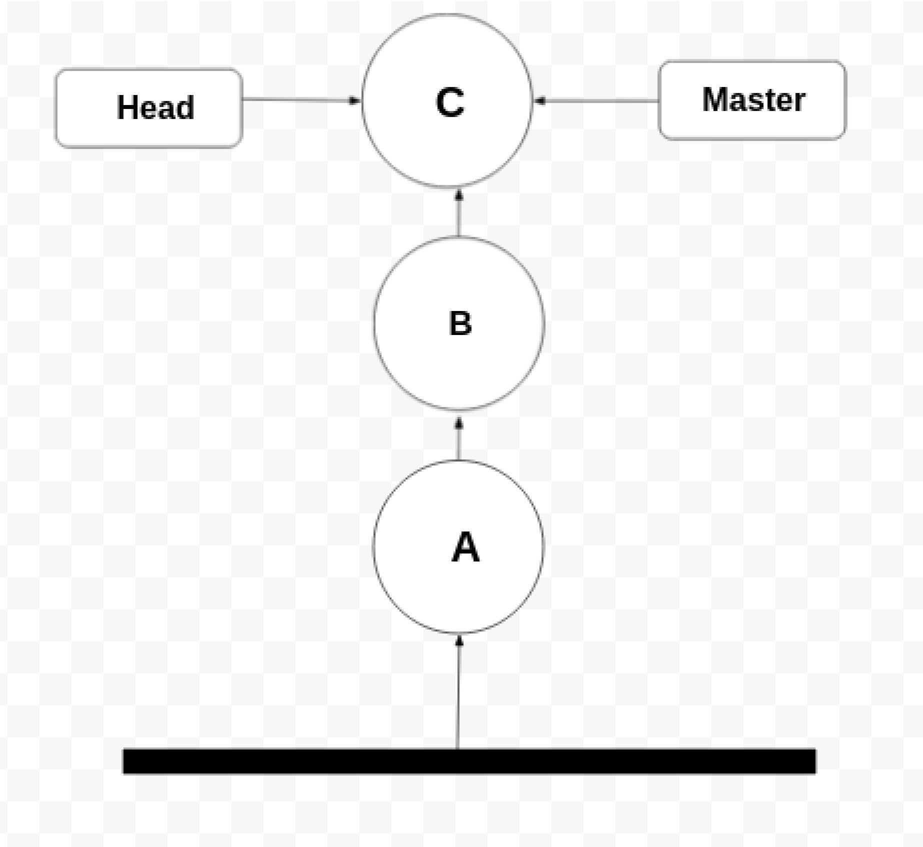
Let us make 1 more commit by including the division feature.

# vim calc.py



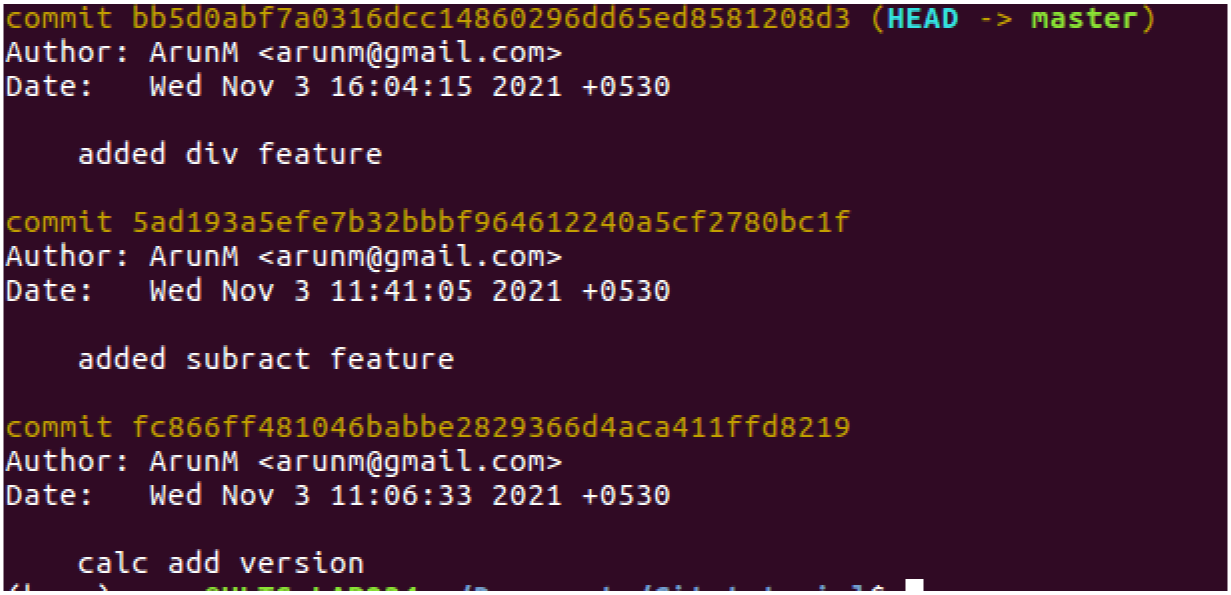
```
def add(a,b):
    return a+b

def sub(a,b):
    return a-b

def div(a,b):
    return a/b
```

**# git add .**
**# git commit -m “added div feature”**



Now git log looks as follows

**#git log --decorate**



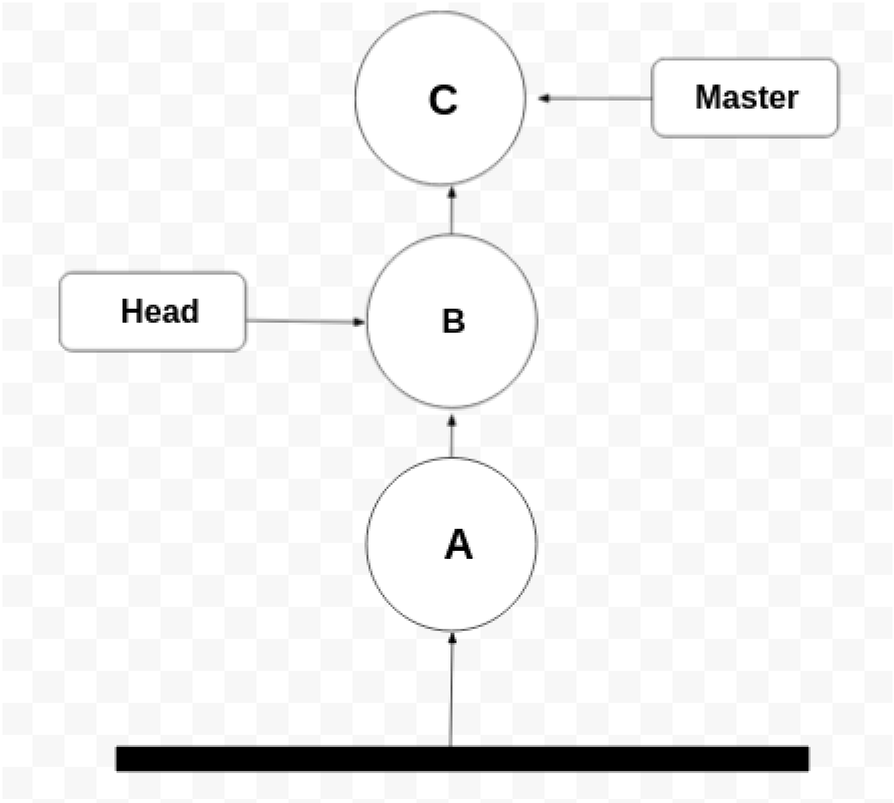
```
commit bb5d0abf7a0316dcc14860296dd65ed8581208d3 (HEAD -> master)
Author: ArunM <arunm@gmail.com>
Date:   Wed Nov 3 16:04:15 2021 +0530

    added div feature

commit 5ad193a5efe7b32bbbf964612240a5cf2780bc1f
Author: ArunM <arunm@gmail.com>
Date:   Wed Nov 3 11:41:05 2021 +0530

    added subract feature

commit fc866ff481046babbe2829366d4aca411ffd8219
Author: ArunM <arunm@gmail.com>
Date:   Wed Nov 3 11:06:33 2021 +0530

    calc add version
```

Now to go back to commit/version corresponding to “add subtract feature”

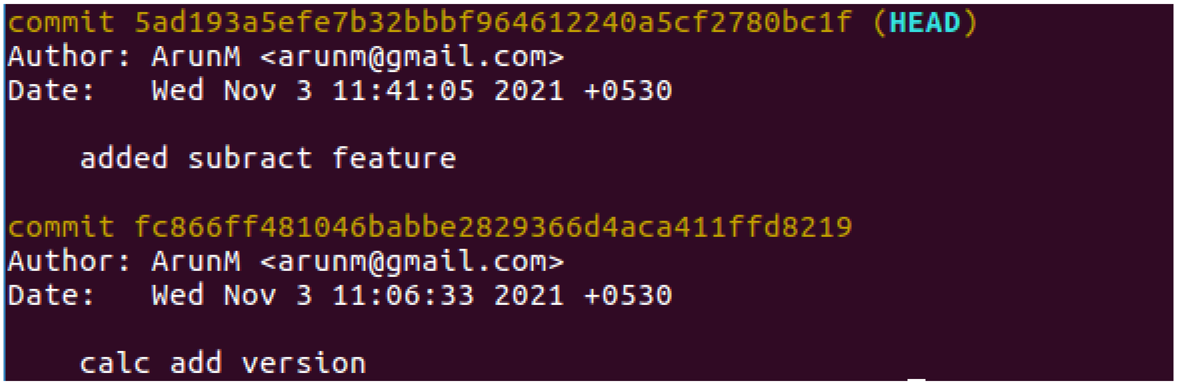
# git checkout commit id

**#git checkout 5ad193a5efe7b32bbbf964612240a5cf2780bc1f**

Now just check index.html.We can see that it will be in old version. Here what happens is HEAD moves to the checkout commit id(ie to the second version).



Now again check for git logs



```
commit 5ad193a5efe7b32bbbf964612240a5cf2780bc1f (HEAD)
Author: ArunM <arunm@gmail.com>
Date:   Wed Nov 3 11:41:05 2021 +0530

    added subract feature

commit fc866ff481046babbe2829366d4aca411ffd8219
Author: ArunM <arunm@gmail.com>
Date:   Wed Nov 3 11:06:33 2021 +0530

    calc add version
```

## How to move a file from staging to working area?

**# git reset HEAD filename**

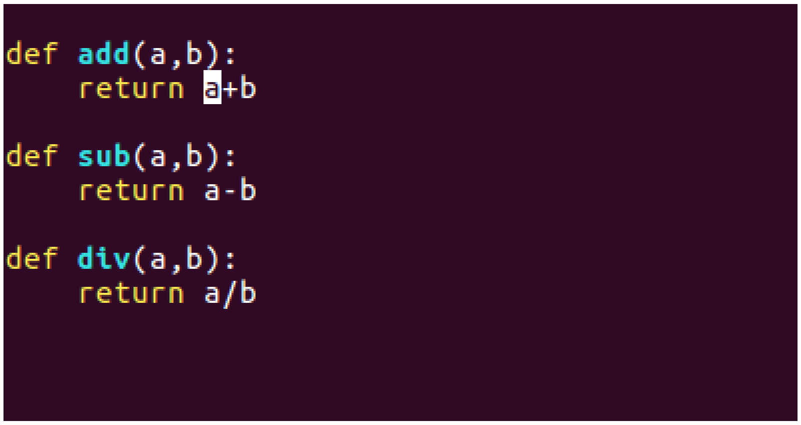
# Need of Git diff

- It helps to show the difference between the file that we are working and the same on staging

- Also it can show the difference between the file in the staging area and the commit area.

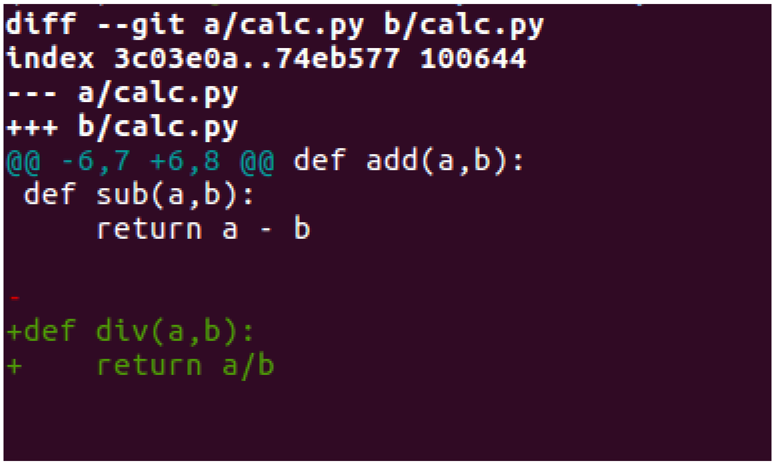
Let us explore.

Let us again add the division feature.



```
def add(a,b):
    return a+b

def sub(a,b):
    return a-b

def div(a,b):
    return a/b
```

Now just check the difference
# git diff calc.py



```
diff --git a/calc.py b/calc.py
index 3c03e0a..74eb577 100644
--- a/calc.py
+++ b/calc.py
@@ -6,7 +6,8 @@ def add(a,b):
 def sub(a,b):
     return a - b

-
+def div(a,b):
+    return a/b
```

We can see the lines that we added as output.

Note: To know difference between staging area and commit area use #git diff --staged

# Need of .gitignore

**There will be some files, folders in our working directory, we donut need to commit. Location of all those files and folders can be kept inside *.gitignore* file.**