Problem Statement

Hooli Co, Has put Harry and Kane to implement a new set of features in their feature software. Both started working separately, making their own copies of the same source code. Now, it has become difficult for them to track the changes they’ve made in the original code,and they are finding it difficult to merge their code together.

Select the directory where I want to initialize Git

Initialize Git in the directory

* Syntax: git init

Create a src folder

Syntax: mkdir src

Create two files inside src folder

* Syntax: echo “app.py” >> app.py

# created app.py file

def addition(a,b):

    return a+b

addition(4,5)

def division(a,b):

    return a/b

division(4,5)

* Syntax: echo “main.py” >> main.py

# created main.py

class Hooli:

    def \_\_init\_\_(self):

        print("welcome to the python")

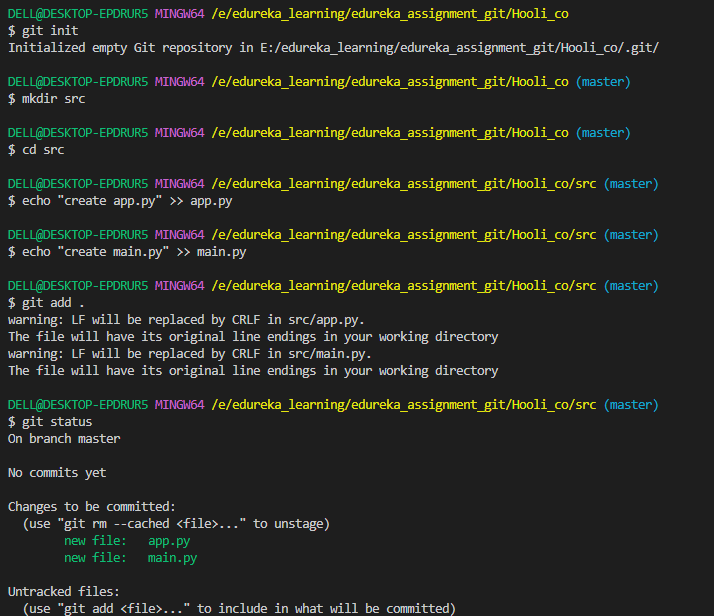
python = Hooli()

This two files to add the staging area

* Syntax: git add .

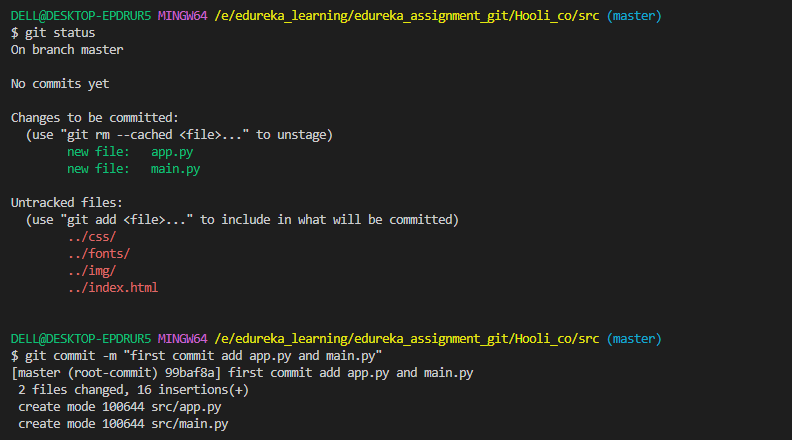
To check the working tree status

* Syntax: git status



App.py and main.py To commit the staged files to our local repository:

* Syntax: git commit –m “first commit add app.py and main.py”

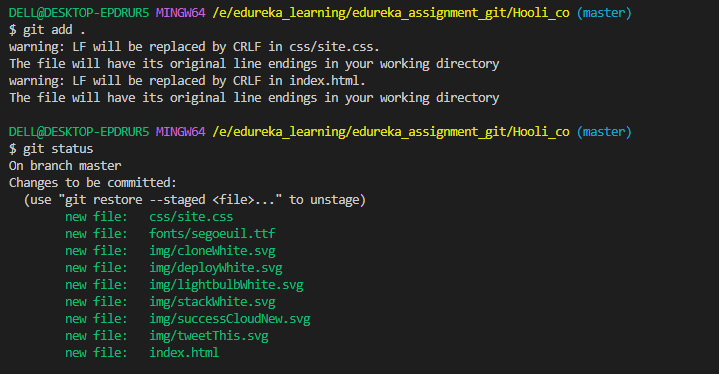


All structure files to add the staging area

* Syntax: git add .

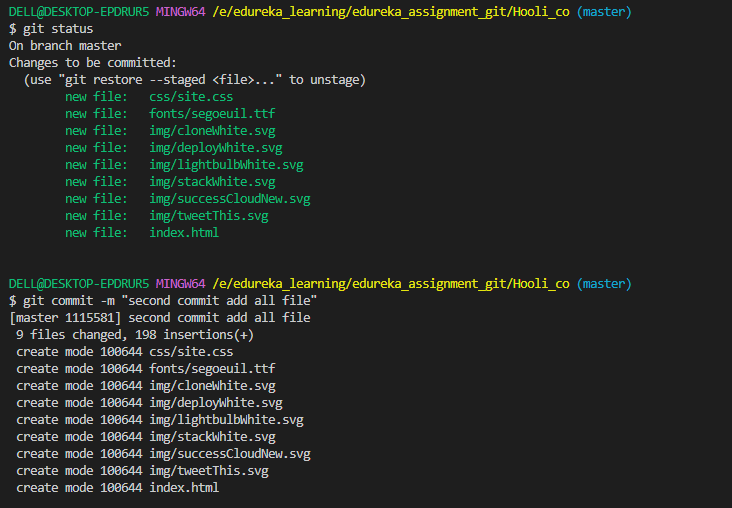
To check the working tree status

* Syntax: git status



To commit the staged files to our local repository:

* Syntax: git commit –m “first commit add app.py and main.py”

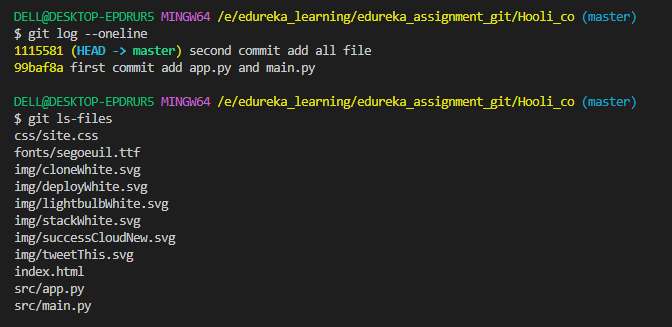


Check log

Syntax:git log –online

Check all files in local repository

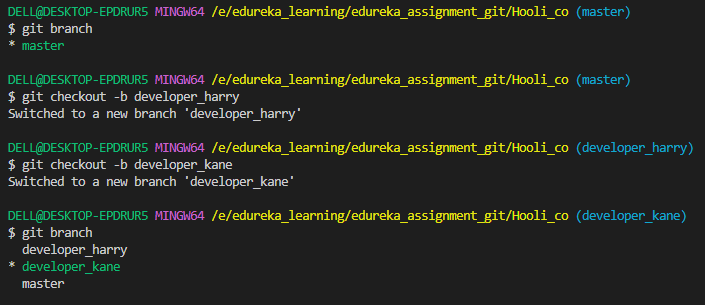
Syntax: git ls-files



1. Create two separate branches from master

Syntax: git checkout –b developer\_harry

Syntax: git checkout –b developer\_kane

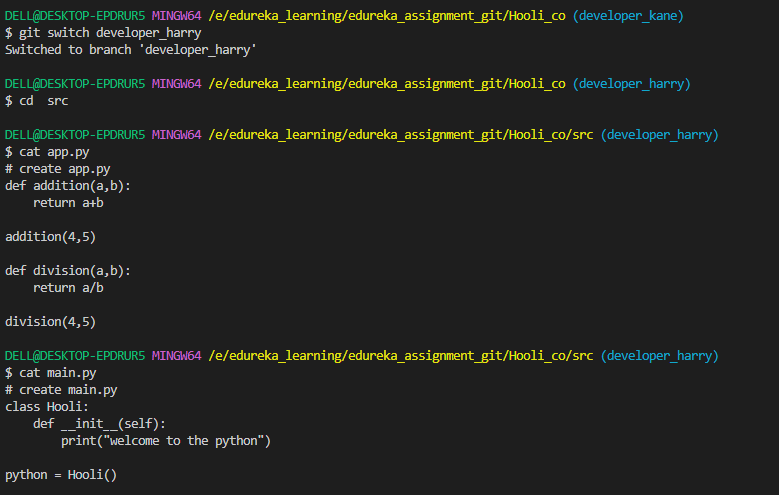


Switch to developer\_harry branch

* Syntax: git switch developer\_harry

Check app.py and main.py code in developer\_harry branch

* Syntax: cat app.py
* Syntax: cat main.py

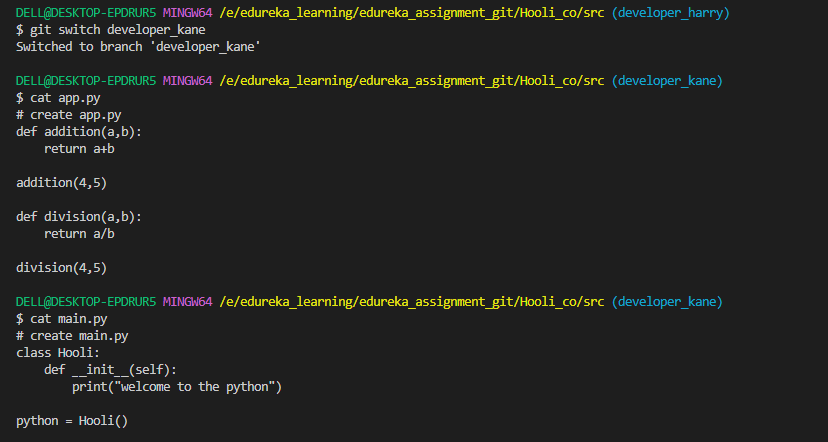


Switch to developer\_kane branch

* Syntax: git switch developer\_kane

Check app.py and main.py code in developer\_kane branch

* Syntax: cat app.py
* Syntax: cat main.py



1. Make changes in the same function of the source code in both the branches

In developer\_kane branch:

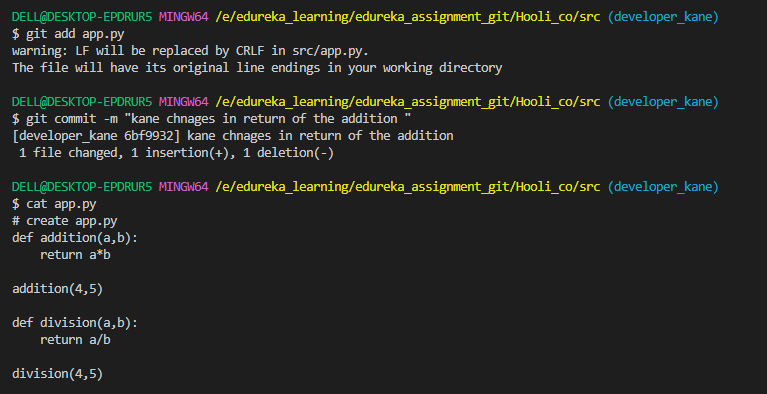
Developer kane want to change in app.py file

Kane make changes in only “ return” of the addition() function

“return a+b ” to “return a\*b

1. # create app.py
2. def addition(a,b):
3. return a\*b
4. addition(4,5)
5. def division(a,b):
6. return a/b
7. division(4,5)

* Syntax: git add .
* Syntax: git commit –m “make changes in return of the addition function”
* Syntax: cat app.py



Changes in Branch developer\_harry:

Developer harry also want to Change in app.py file

Harry change name of the addition() function to multiplication() and also make changes in return

“return a+b ” to “return a\*b

# create app.py

def multiplication(a,b):

    return a\*b

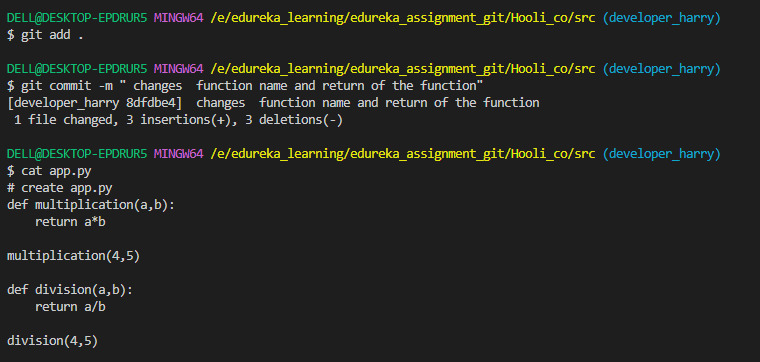
multiplication(4,5)

def division(a,b):

    return a/b

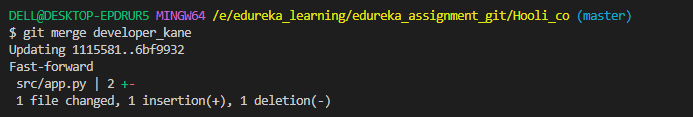
division(4,5)

* Syntax: git add .
* Syntax: git commit –m “ changes function name and return of the function”
* Syntax: cat app.py



1. Merge branch developer\_kane into the master

* Syntax: git switch master
* Syntax: git merge developer\_kane

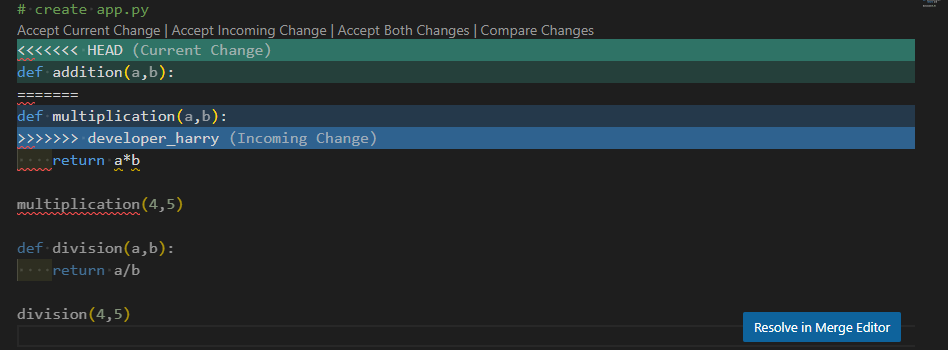


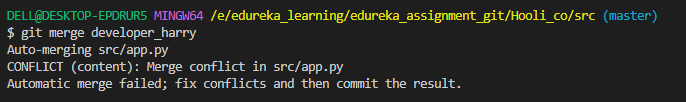
Check app.py in master branch

* Syntax: cat app.py

1. Try and merge branch2 into the master(merge conflict should arise)

* Syntax: git merge developer\_harry





Check Conflict

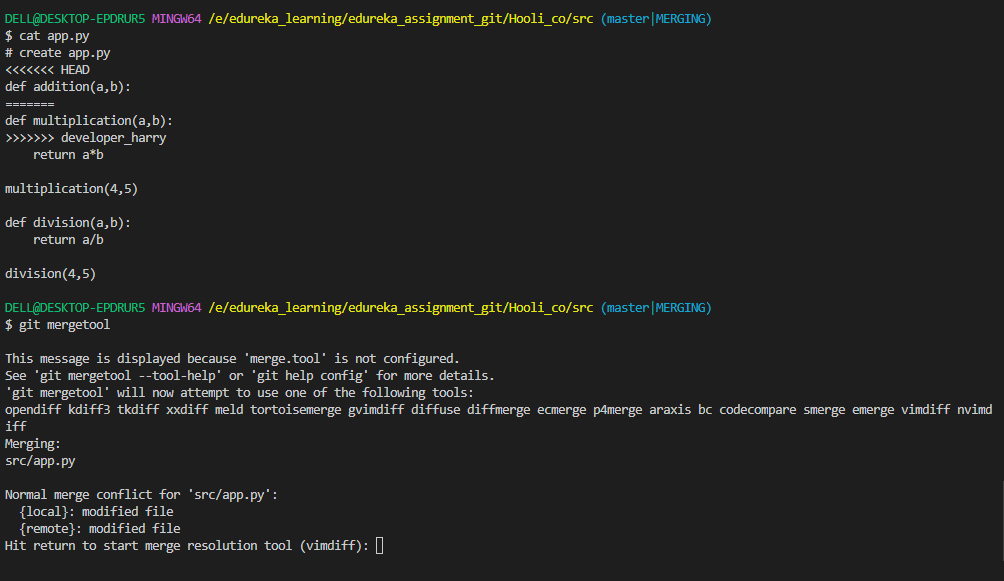
* Syntax : cat app.py

1. Install a merge tool of your choice and resolve the merge conflict using

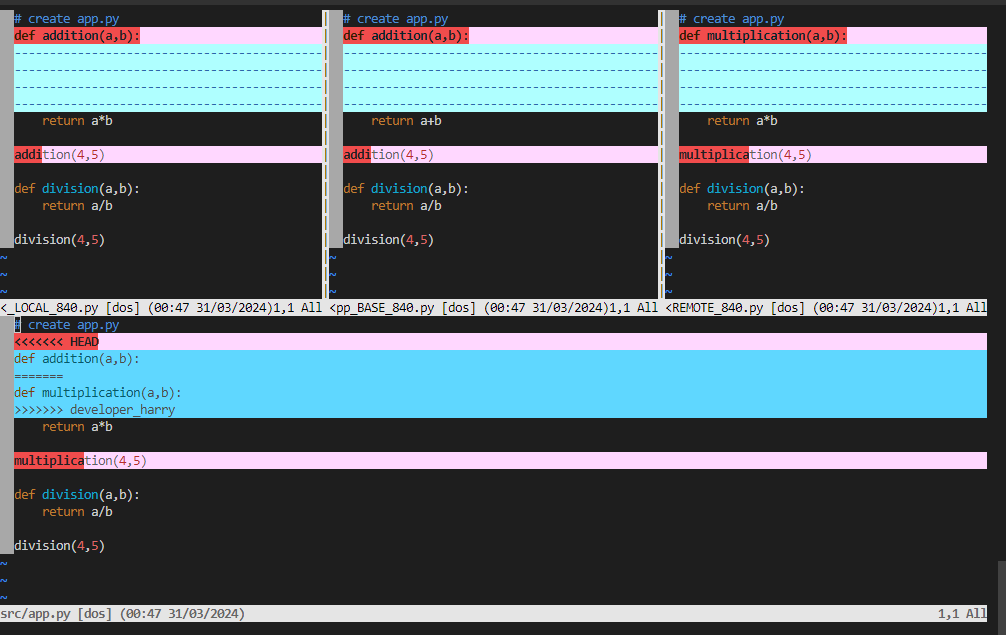
git mergetool command

Now I am using git merge conflict tool

* Syntax: git meregetool



Press Enter



Now Developer harry discuss with developer kane

So after discussion Harry and kane are decided remove to the current code of line

def addition(a,b):

return a\*b

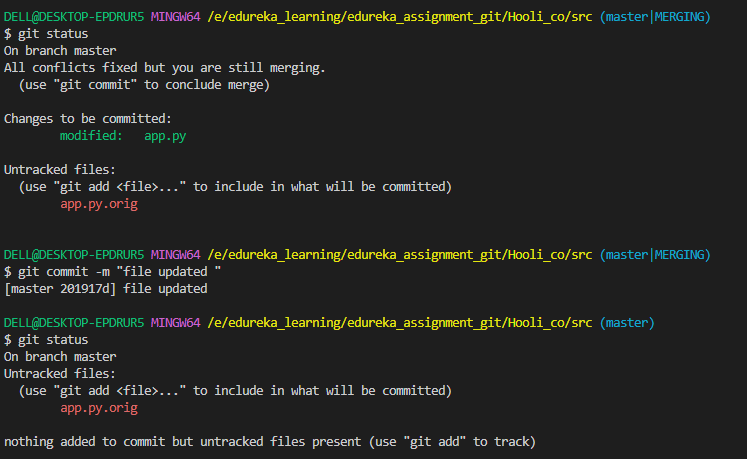


To check the working tree status

* Syntax: git status

To commit the staged files to my local repository:

* Syntax: git commit –m “file updated”



Remove temporary file app.py.orig

* Syntax: rm –rf app.py.orig

