Module 3: Branching and Merging Git

Demo3: Demo on git merge and git rebase

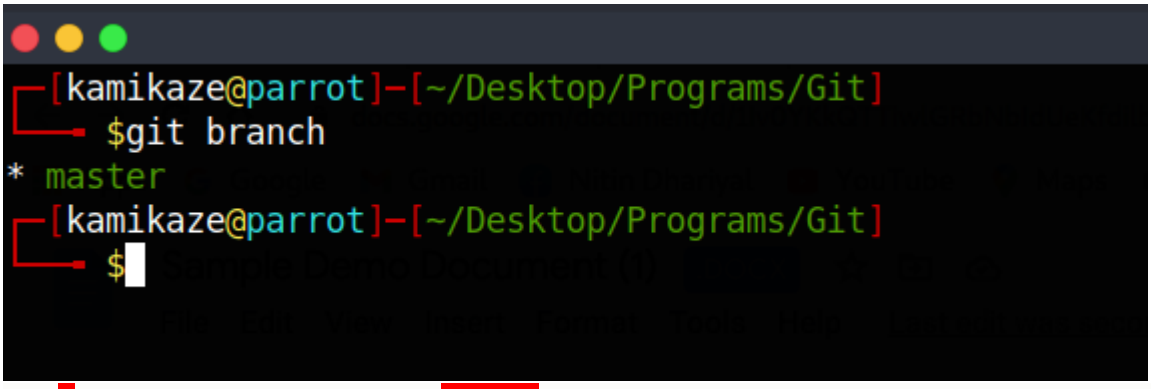
Problem Statement:

How we perform merging of two branches by git and how to rebase our git using git-rebase.
Solution:

Step 1: In This demo, we will go through GitHub Branch Creation then we merged them with the help of the Git Command Line Interface and then we rebase our git with the help of Git Command Line Interface. Let us First perform merging.

Command Used: git branch

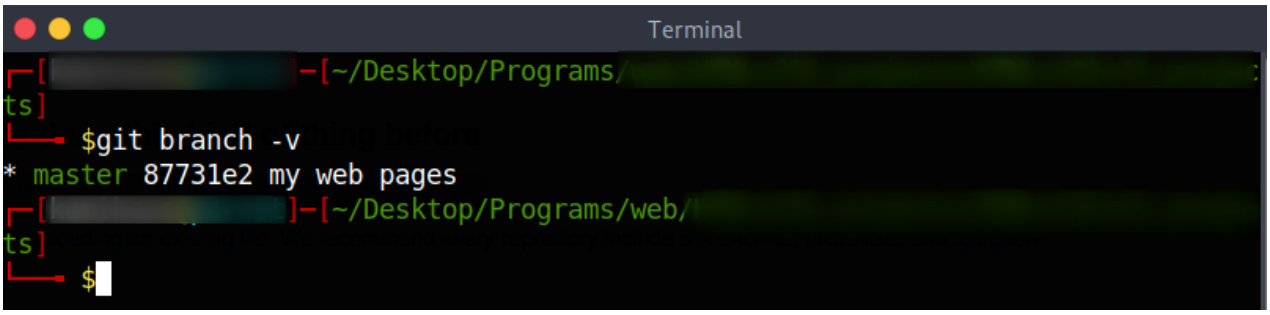
This command will do more than just create and delete branches. If you run it with no arguments, you will get a simple listing of your current branches:



The * character here prefixes the master branch: indicates the branch that you currently have checked out (i.e., the branch that HEAD points to). It means that if you commit at this point, the master branch will be moved forward with your new work.

Step 2: Let us the last commit on each branch,

Command Used: git branch -v



Step 3: Let us Create a new branch,
Command Used: git branch branchname

Step 4: Let us Checkout into our branch.
Command Used: git checkout branchname

Step 5: Let us create a file in the “thirsty” branch. I created a thirsty.py file, you can create your own file.

```
learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (master)
$ git branch thirsty

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (master)
$ git branch
* master
  thirsty

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (master)
$ git checkout thirsty
Switched to branch 'thirsty'

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (thirsty)
$ git branch
  master
* thirsty

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (thirsty)
$
```

```
hungry=input("are you hungry?")
if hungry=="yes":
    print("eat samosa")
    print("eat pizza")
    print("eat burger")
else:
    thirsty=input("ae you")
    print("do your homework")
```

Step 6: Let us add this file into our branch and make a commit for it:

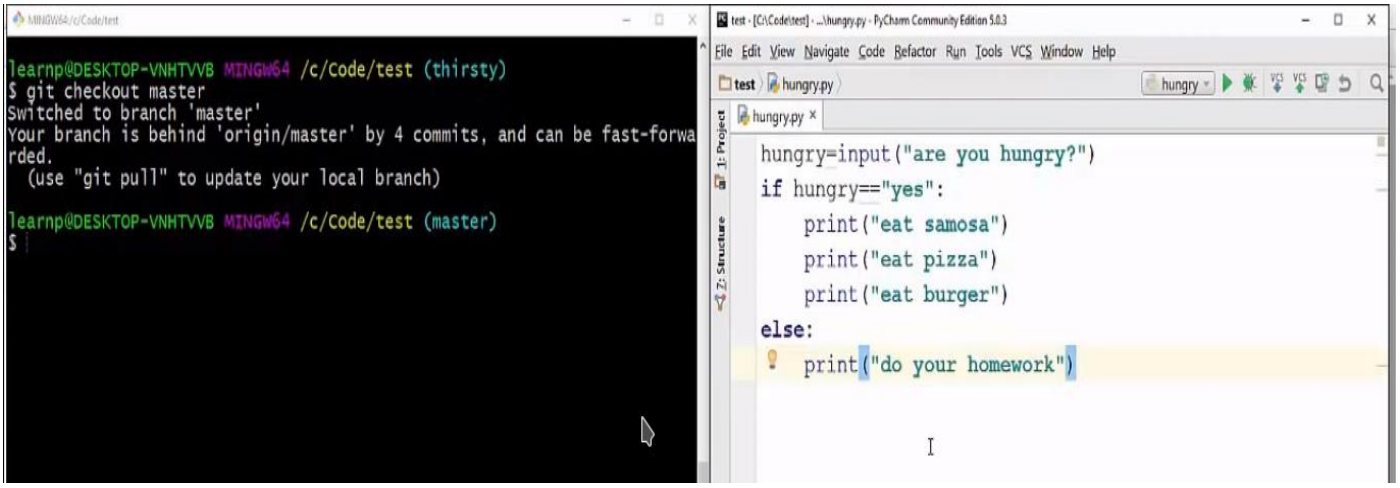
Command Used: git add filename

Command Used: git commit -m “your commit”

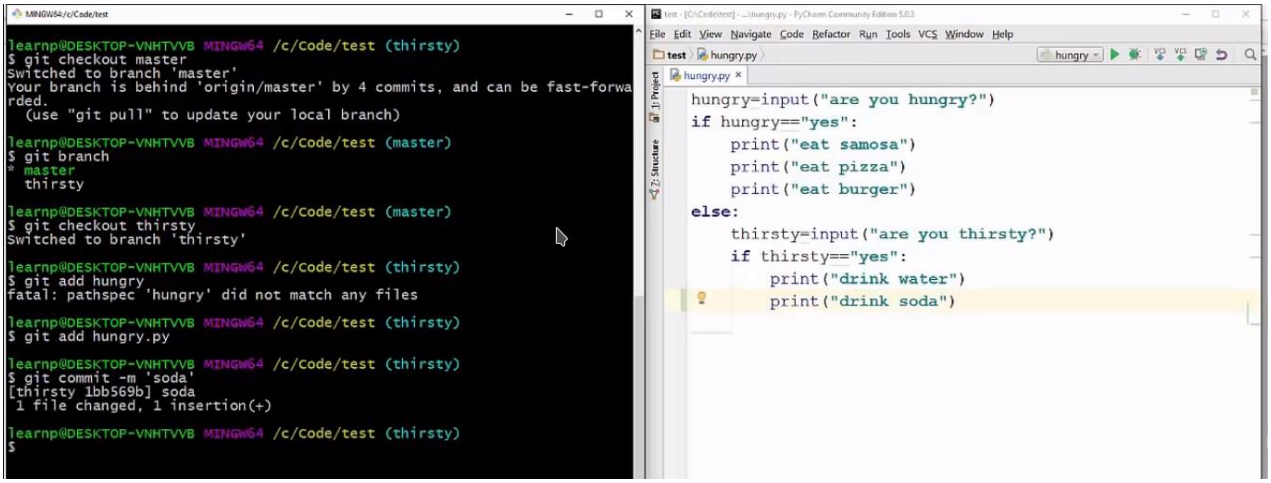
```
learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (master)
$ git branch
* master

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (master)
$ git branch thirsty

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (master)
$ git branch
* master
  thirsty

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (master)
$ git checkout thirsty
Switched to branch 'thirsty'

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (thirsty)
$ git branch
  master
* thirsty

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (thirsty)
$ git add hungry.py

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (thirsty)
$ git commit -m 'water'
[thirsty d051ec2] water
 1 file changed, 4 insertions(+), 1 deletion(-)

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (thirsty)
$
```

```
hungry=input("are you hungry?")
if hungry=="yes":
    print("eat samosa")
    print("eat pizza")
    print("eat burger")
else:
    thirsty=input("are you thirsty?")
    if thirsty=="yes":
        print("drink water")
```

Step 7: Let us check our master branch, you see the code here got changed because the changes we did are only on the “thirsty” branch.



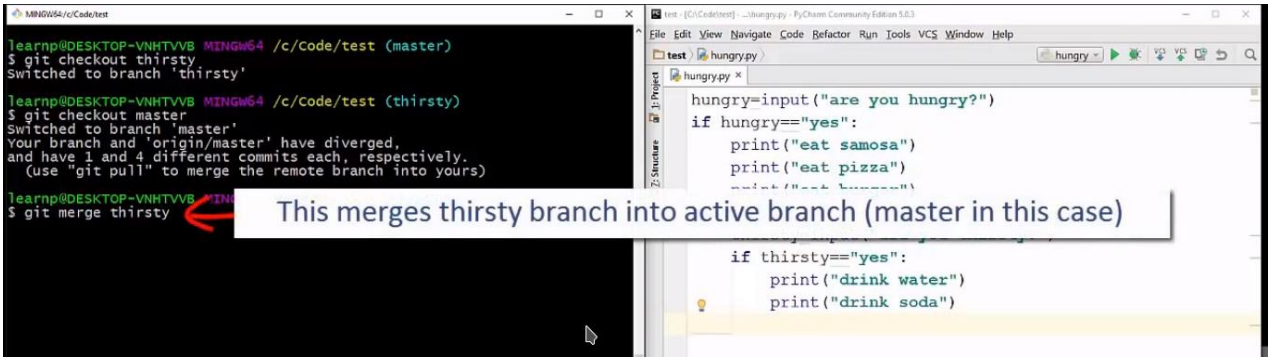
Step 8: Let us make a new branch over here, I make a branch name “hungry”, let us add a file into it(hungry.py) and make commit into it.



Step 9: let us merge branch, I am merging the thirsty branch into my master branch for that we use,

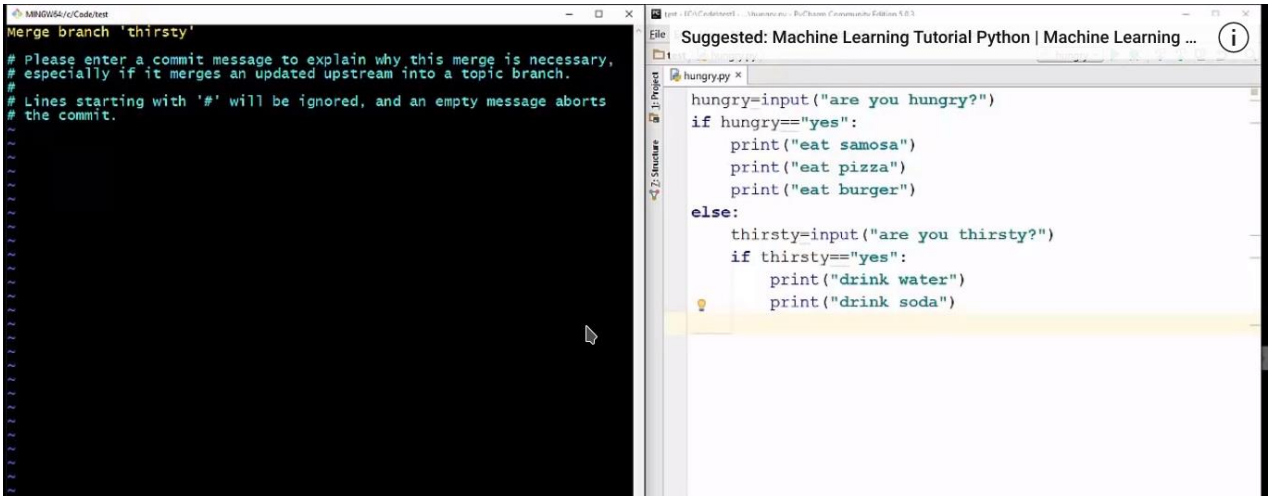
Command Used: git checkout master

Command Used: git merge branchname



Step 10: It will ask you to commit the message. after committing the message, you must use:

Command Used: !wq



Step 11: You can check the log using:

Command: git log

Finally, you can check, you have performed merging into git.

Step 12: Let us rebase our git, rebasing is changing the base of your branch from one commit to another making it appear as if you had created your branch from a different commit. Internally, Git accomplishes this by creating and applying new commitments to the specified base. It is important to realise that although the branch looks the same, it is made up of completely new commits.

Command Used: git rebase branchname

Step 13: The -I flag begins an interactive rebasing session to run git rebase. An interactive rebasing is the opportunity to change the individual commits instead of blindly moving all commits to the new base. This enables you to clear the history by removing, splitting, and altering a few commits that exist today. It is like Git commit --amend on steroids.

```
git rebase --interactive <base>
```

Step 13: Hence we finally rebase our git repository locally.