

# Github Notes

## Commonly faced problems when we didn't use Git

- Hardisk failure & computer crash
  - Hard for multiple people to work on same code
  - Difficult to track changes revert back to previous version
- 
- git clone <https://github.com/milindzuge01/test.git>
  - check git status --> git status
  - To push in staging area --> git add hungry.py

**git add hungry.py** will add hungry.py file to staging area. Later on staging area will be used to commit files when you run **git commit**

- Commit the changes locally --> git commit -m 'first version of hungry code'

**git commit** adds your changes to local version control database. You have still not pushed these changes to remote server (github in this case)

- To check log of your code --> git log
- To push code on main github server --> git push

**git push** will now push committed changes to remote server (github in this case)

- To see the changes between current code & previous code --> git difftool HEAD

**git difftool** shows difference between your local changes and previous version of the file

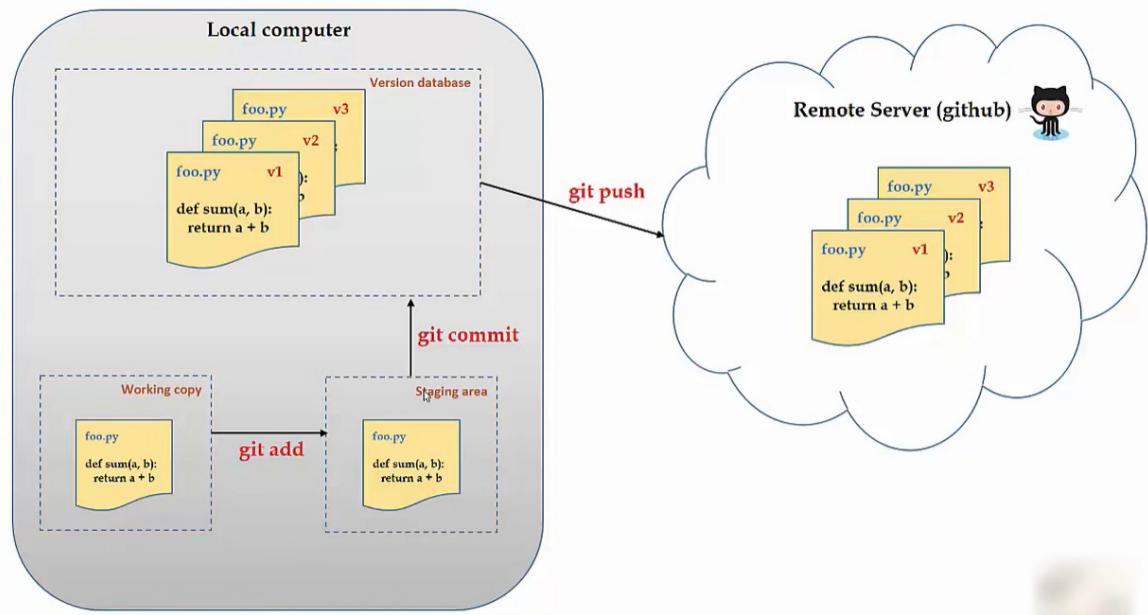
```

C:\Users\...\hungry.py
hungry=input("are you hungry?")
if hungry=="yes":
    print("eat samosa")
else:
    print("do your homework")

hungry.py
hungry=input("are you hungry?")
if hungry=="yes":
    print("eat samosa")
else:
    print("do your homework")

```

- step 1: git add hungry.py
- step 2: git commit -m 'first version of hungry code'
- STEP 3: git push



## Git Tutorial 5: Undoing/Reverting/Resetting code changes

We will see how to,

**1) Undo uncommitted changes**

**git checkout --**

**2) Undo committed changes**

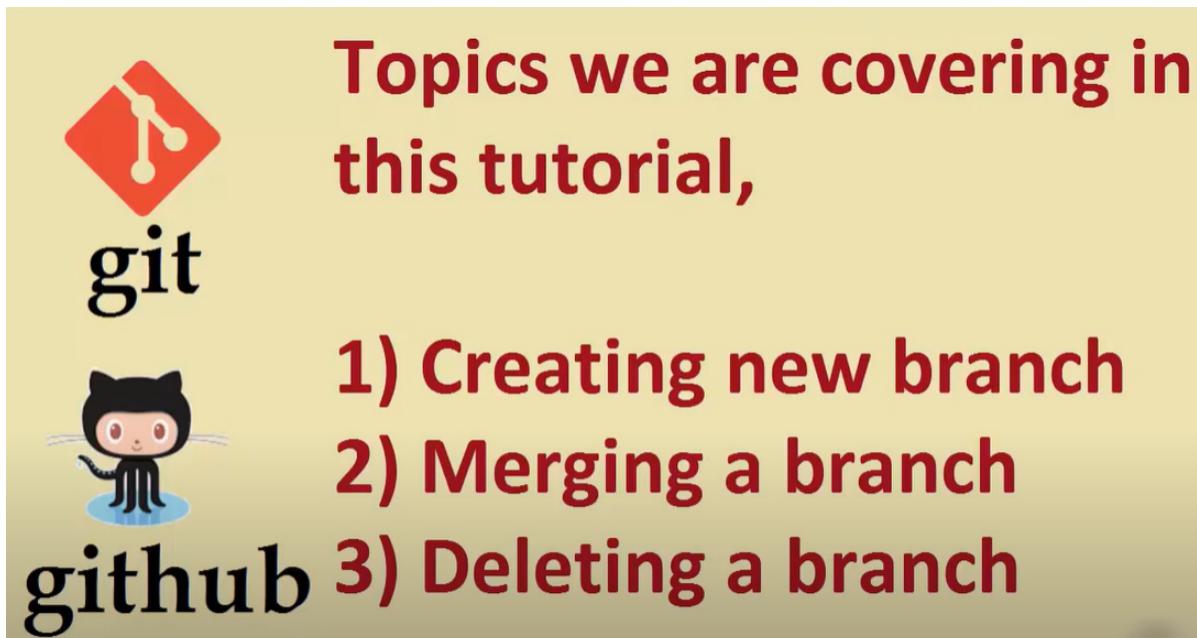
**git revert**

**3) Resetting changes**

**git reset**

- undo uncommitted changes in working directory --> git restore file\_name or git checkout -- file\_name
  - undo uncommit changes for all file --> git checkout -- .
- undo commited changes in staging directory --> git revert commit\_id
  - explicitly commit revert changes in the staging dirctory --> git revert -n commit\_id
    - to explicitly commit after reverting --> git commit -m 'revert change name'

## Git Tutorial 6: Branches (Create, Merge, Delete a branch)



master is a default branch  
in git

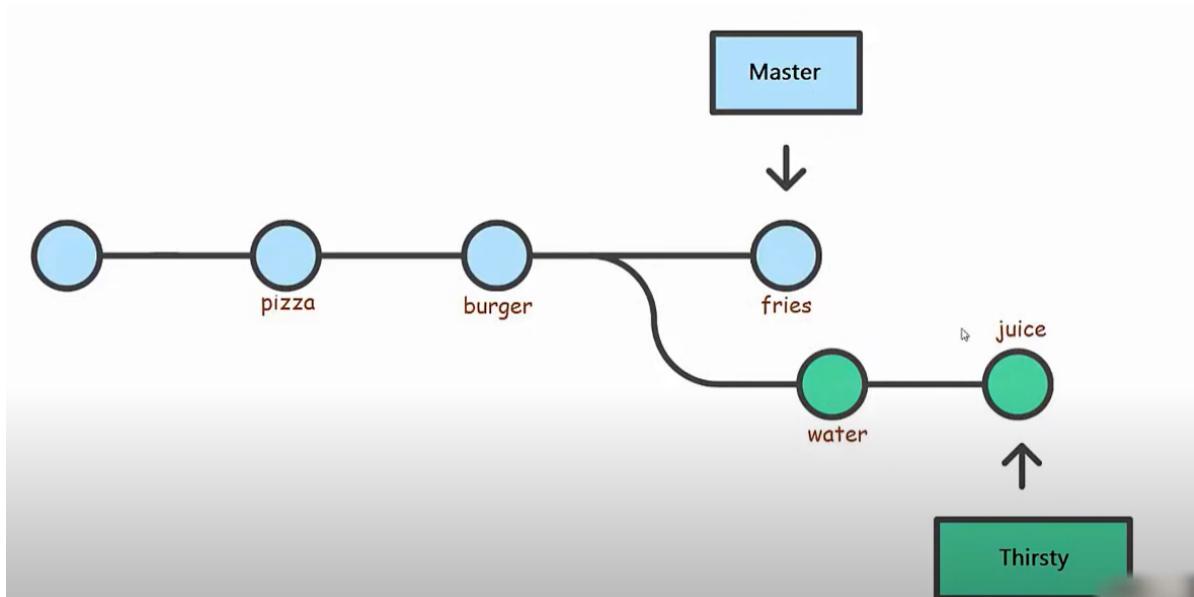
```
MINGW64:/c/Code/test
learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (master)
$ git branch
* master

learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (master)
$ git branch thirsty ← This creates a new branch called thirsty

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

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

- shows branches in a prompt --> git branch
- star denotes the active branch
- create new branch --> git branch thirsty
- change the branch to using --> git checkout branch\_name

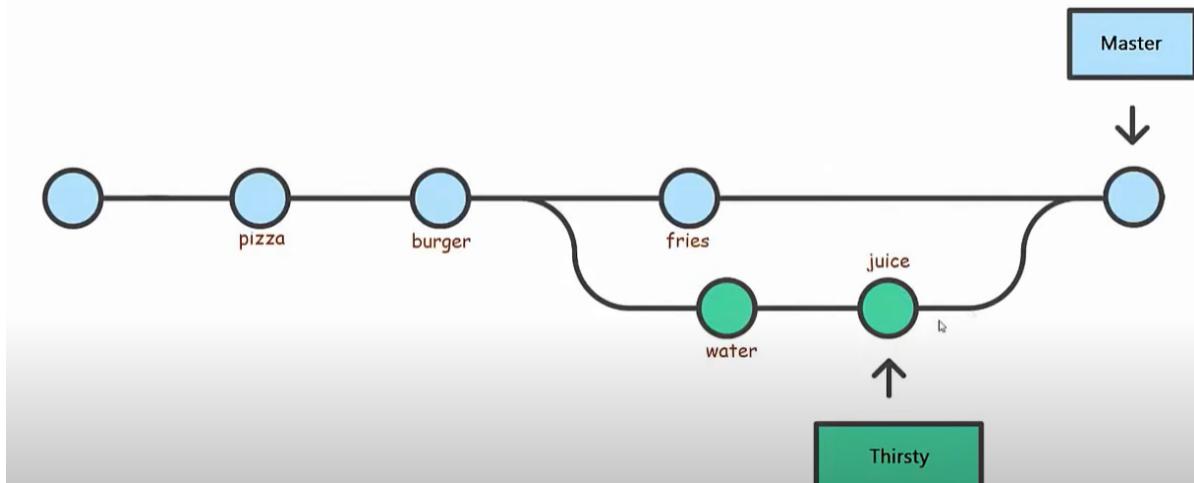


# merge two branches

```

learnr@DESKTOP-VNHTVVB MINGW64 /c/code/test (master)
$ git checkout thirsty
Switched to branch 'thirsty'
learnr@DESKTOP-VNHTVVB MINGW64 /c/code/test (thirsty)
$ git checkout master
Switched to branch 'master'
Your branch and 'origin/master' have diverged,
and have 1 and 4 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)
learnr@DESKTOP-VNHTVVB MINGW64 /c/code/test (master)
$ git merge thirsty
  This merges thirsty branch into active branch (master in this case)

```



# deleting a branch

- creating new branch --> git checkout -b dummy

```
learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/test (thirsty)
$ git checkout -b dummy
Switched to a new branch 'dummy'

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

- delete a branch --> git branch -d dummy

## Git Tutorial 7: What is HEAD?

HEAD is a reference to  
the most recent commit in  
current branch (in most of  
the cases)

- to see most recent commit --> git show HEAD

```

MINGW64:/c/Code/hungry
$ git log
commit 841ff3157000c41a3c18b2034b765f2b39fc366
Author: learnp <learnpythonlanguage@gmail.com>
Date: Tue Aug 16 07:58:03 2016 -0400

    burger

commit 4bc4ed9d2d6fe4e6ef2a553f9c3df5106c641b49
Author: learnp <learnpythonlanguage@gmail.com>
Date: Mon Aug 15 19:15:01 2016 -0400

    pizza

commit 23ea3b3b27805cd9cf6683713dd1ced676e32e7b
Author: codebasics <learnpythonlanguage@gmail.com>
Date: Sat Aug 13 07:51:09 2016 -0400

    First version of hungry.py

commit e2e8fd5e0e8e3a792275baf1da91f6bda4c457cf
Author: codebasics <learnpythonlanguage@gmail.com>
Date: Sat Aug 13 07:49:22 2016 -0400

    Initial commit

Learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/hungry (master)
$ git show HEAD
commit 841ff3157000c41a3c18b2034b765f2b39fc366
Author: learnp <learnpythonlanguage@gmail.com>
Date: Tue Aug 16 07:58:03 2016 -0400

    burger

diff --git a/hungry.py b/hungry.py
index 6646c0c..669b1b2 100644
--- a/hungry.py
+++ b/hungry.py
@@ -2,5 +2,6 @@ hungry=input("are you hungry?")
if hungry=="yes":
    print("eat samosa")
    print("eat pizza")
+   print("eat burger")
else:
    print("do homework")

```

- to check changes between any two commit --> git difftool old\_commit\_id new\_commit\_id
- above also achieved by using below :

```

MINGW64:/c/Code/hungry
Learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/hungry (master)
$ git log
commit 841ff3157000c41a3c18b2034b765f2b39fc366
Author: learnp <learnpythonlanguage@gmail.com>
Date: Tue Aug 16 07:58:03 2016 -0400

    burger

commit 4bc4ed9d2d6fe4e6ef2a553f9c3df5106c641b49
Author: learnp <learnpythonlanguage@gmail.com>
Date: Mon Aug 15 19:15:01 2016 -0400

    pizza

commit 23ea3b3b27805cd9cf6683713dd1ced676e32e7b
Author: codebasics <learnpythonlanguage@gmail.com>
Date: Sat Aug 13 07:51:09 2016 -0400

    First version of hungry.py

commit e2e8fd5e0e8e3a792275baf1da91f6bda4c457cf
Author: codebasics <learnpythonlanguage@gmail.com>
Date: Sat Aug 13 07:49:22 2016 -0400

    Initial commit

Learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/hungry (master)
$ git difftool 23ea3b3b27805cd9cf6683713dd1ced676e32e7b 4bc4ed9d2d6fe4e6ef2a553f9c3df5106c641b49

Learnp@DESKTOP-VNHTVVB MINGW64 /c/Code/hungry (master)
$ git difftool HEAD~2 HEAD~1

```

# When HEAD doesn't point to most recent commit, you go into detached HEAD state.

- move head to any commit --> git checkout commit\_id

## Git Tutorial 8 - .gitignore file



- create file in main local directory using --> touch .gitignore
- add file name which we want to hide from git status in it & saved it.

here hided yellow highlighted files & it all included in .gitignore file

Name	Date modified	Type	Size
.git	10-05-2022 19:27	File folder	
.gitignore	10-05-2022 19:27	Text Document	1 KB
.a.exe	10-05-2022 19:21	Text Document	0 KB
.b.exe	10-05-2022 19:25	Text Document	0 KB
.hello	10-05-2022 19:25	Text Document	0 KB
hungry.ipynb	10-05-2022 14:50	IPYNB File	2 KB
hungry1.py	10-05-2022 19:21	PY File	1 KB
README.md	10-05-2022 15:39	MD File	1 KB

```

Untracked files:
(use "git add <file>..." to include in what will be committed)
.gitignore
.a.exe.txt
.b.exe.txt
.hello.txt

nothing added to commit but untracked files present (use "git add" to track)

DELL@DESKTOP-D28TC6J MINGW64 /d/python/test (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

nothing added to commit but untracked files present (use "git add" to track)

DELL@DESKTOP-D28TC6J MINGW64 /d/python/test (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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)

DELL@DESKTOP-D28TC6J MINGW64 /d/python/test (main)
$ 

```

## Git Tutorial 9: Diff and Merge using meld



1) Install meld

2) Diff (git difftool)

3) Merge (git mergetool)

- install meld --> <https://meldmerge.org/>
- configure .gitconfig file -->

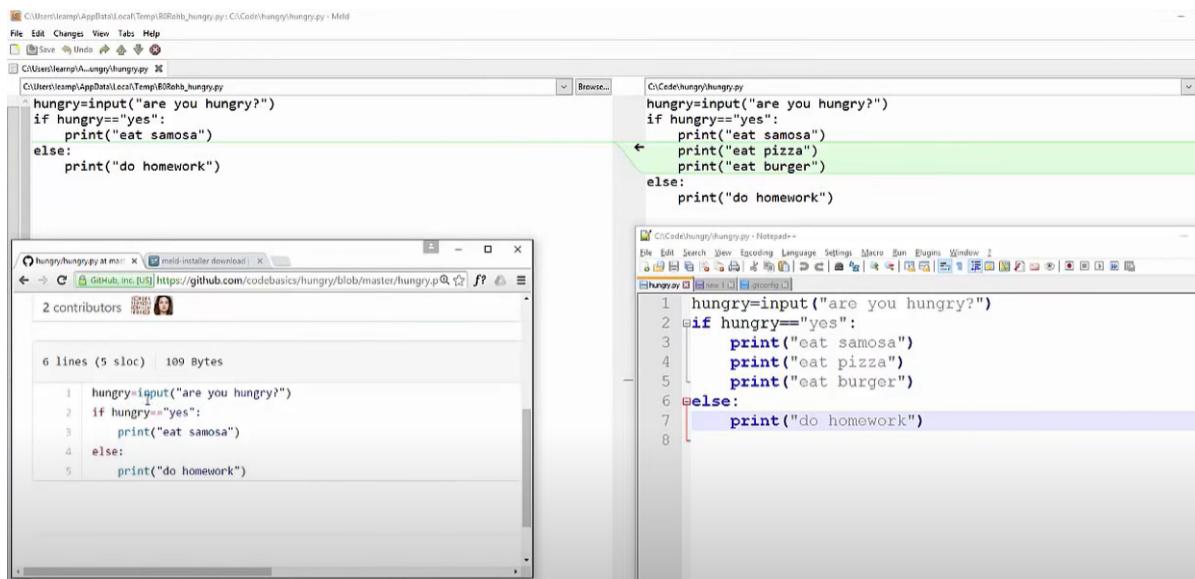
```
[diff] tool = meld [difftool "meld"] path = C:/Program Files (x86)/Meld/Meld.exe [difftool]  
prompt = false
```



```
*.gitconfig - Notepad
File Edit Format View Help
[user]
    email = milindzuge01@gmail.com
    name = Milind

[diff]
    tool = meld
[difftool "meld"]
    path = C:/Program Files (x86)/Meld/Meld.exe
[difftool]
    prompt = false
```

- command to see differences between saved local code & git repository --> git difftool origin/main



- launch specific tool from multiple diff tool using -->
  - git difftool -t meld
  - git difftool -t vimdiff

**when git repository's & local code is out of sync then we have a merge conflict using --> git pull**

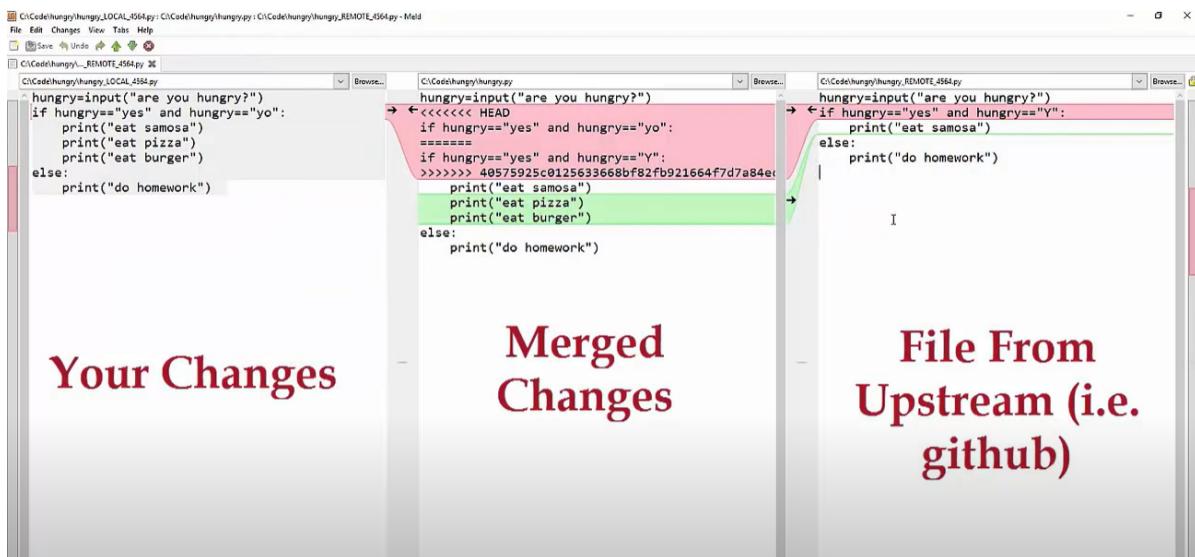
```
DELL@DESKTOP-D28TC6J MINGW64 /d/python/test (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 676 bytes | 56.00 KiB/s, done.
From https://github.com/milindzuge01/test
  01c77e1..b34f075 main      -> origin/main
Auto-merging hungry2.py
CONFLICT (content): Merge conflict in hungry2.py
Automatic merge failed; fix conflicts and then commit the result.
```

- configure .gitconfig file -->

```
[merge] tool = meld [mergetool "meld"] path = C:/Program Files (x86)/Meld/Meld.exe
[mergetool] keepBackup = false
```

```
[merge]
    tool = meld
[mergetool "meld"]
    path = C:/Program Files (x86)/Meld/Meld.exe
[mergetool]
    keepBackup = false
```

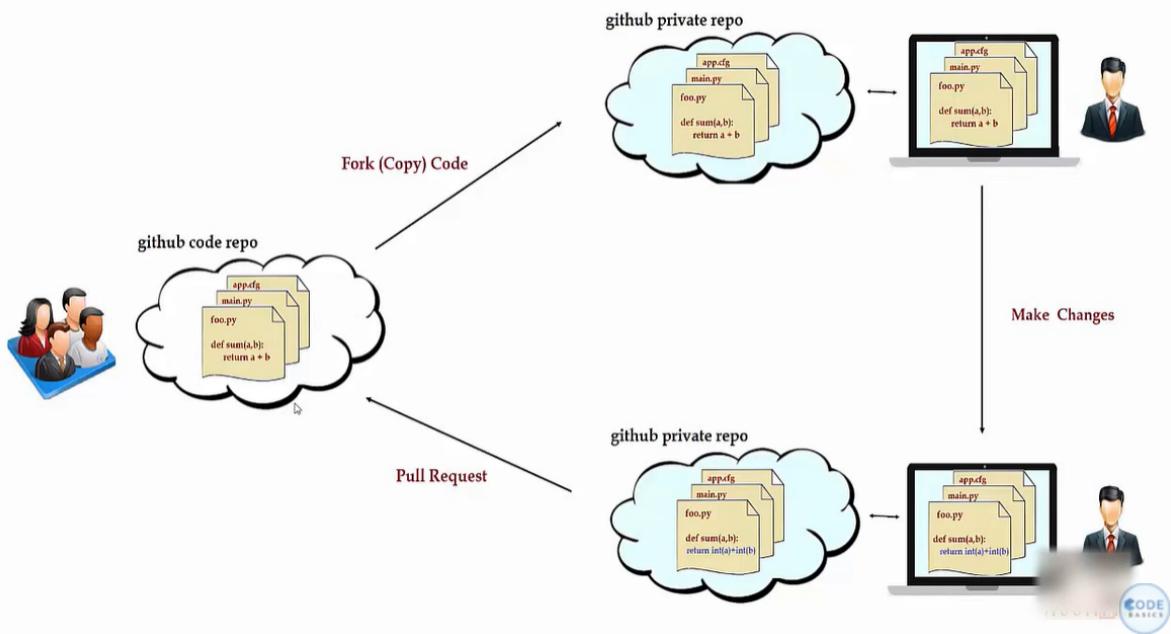
- command to see differences between saved local code,git repository with merging --> git mergetool



## Git Github Tutorial 10: What is Pull Request?

# What is Pull Request

- On github, owner can share his code with others
- Other person can make code changes and send a request to owner to pull/merge his code changes into owner's repository



In [ ]: