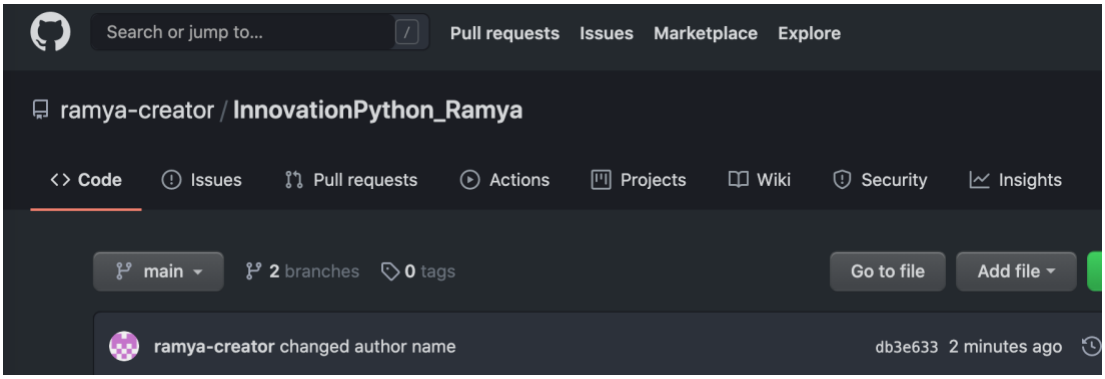
# Ramya Vemuru
# Git Theory Assignment

**(1) Make a repository on GitHub with the name "InnovationPython_yourname" eg: "InnovationPython_Ankush".**
**Practice on following commands:**




**Git Clone:**

```
Ramya $
Ramya $ls
Ramya $git clone https://github.com/ramya-creator/InnovationPython_Ramya.git
Cloning into 'InnovationPython_Ramya'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
Ramya $
```

**Git Diff:**

```
InnovationPython_Ramya $git diff README.md
diff --git a/README.md b/README.md
index 7cbd0fc..8d63661 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,2 @@
-# innovation_python
\ No newline at end of file
+# innovation_python
+This is a sample change for diff.
InnovationPython_Ramya $
```

**Git Add, Git Status:**

```
InnovationPython_Ramya $git add README.md
InnovationPython_Ramya $git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   README.md

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

        newfile

InnovationPython_Ramya $
```

**Git Commit:**

```
InnovationPython_Ramya $git commit -m "Modified git readme file"
[main 316c774] Modified git readme file
 1 file changed, 2 insertions(+), 1 deletion(-)
InnovationPython_Ramya $git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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

        newfile

nothing added to commit but untracked files present (use "git add" to track)
InnovationPython_Ramya $8
```

**t:**

**Git Log:**

```
InnovationPython_Ramya $git log
commit 316c77487983228e2110cdea0e28b2d5fe7bf2d4 (HEAD -> main)
Author: Varun Thalluru <vthalluru2@apple.com>
Date:   Sat Oct 31 21:13:43 2020 -0500

    Modified git readme file

commit 460e072fe324c4c6af818f7f33e0317feadf8ce3 (origin/main, origin/HEAD)
Author: ramya-creator <73764994+ramya-creator@users.noreply.github.com>
Date:   Sat Oct 31 20:38:50 2020 -0500

    Initial commit
```

**Git Branch:**

```
InnovationPython_Ramya $git branch
* main
InnovationPython_Ramya $
```

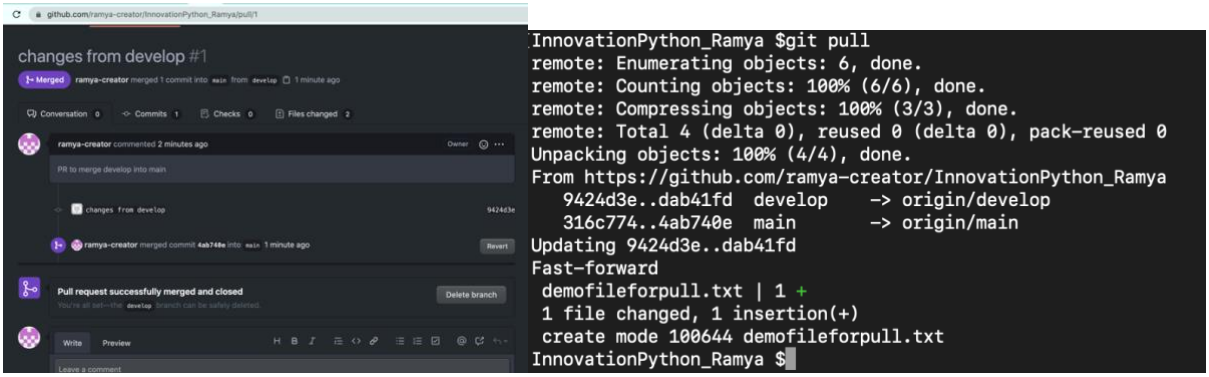
**Git Push:**

```
InnovationPython_Ramya $git push -u origin main
Username for 'https://github.com': ramya-creator
Password for 'https://ramya-creator@github.com':
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 308 bytes | 308.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/ramya-creator/InnovationPython_Ramya.git
   460e072..316c774  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
InnovationPython_Ramya $
```

**Git Checkout:**

```
InnovationPython_Ramya $git branch
* main
InnovationPython_Ramya $git checkout -b develop
Switched to a new branch 'develop'
InnovationPython_Ramya $git branch
* develop
  main
```

**Git Pull:**




## (2) Read about difference between Git and Github.

**Git** is a revision control system, a tool to manage your source code history. **GitHub** is a hosting service for Git repositories.

## (3) Read about Git Workflow.

Git Workflow is a recipe or recommendation for how to use Git to accomplish work in a consistent and productive manner. Git workflows encourage users to leverage Git effectively and consistently. Git can be used effectively and consistently by the developers because of the Git Workflows.

## (4) How many types of version control systems are there?

-Local version control system
-Centralized version control system
-Distributed version control system

## (5) Explain Branching concept in Git.

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master. As you start making commits, you're given a master branch that points to the last commit you made.

## (6) Explain Forking Workflow in Git.

The Forking Workflow is fundamentally different than other popular Git workflows. Instead of using a single server-side repository to act as the “central” codebase, it gives every developer their own server-side repository. This means that each contributor has not one, but two Git repositories: a private local one and a public server-side one. The Forking Workflow is most often seen in public open source projects.