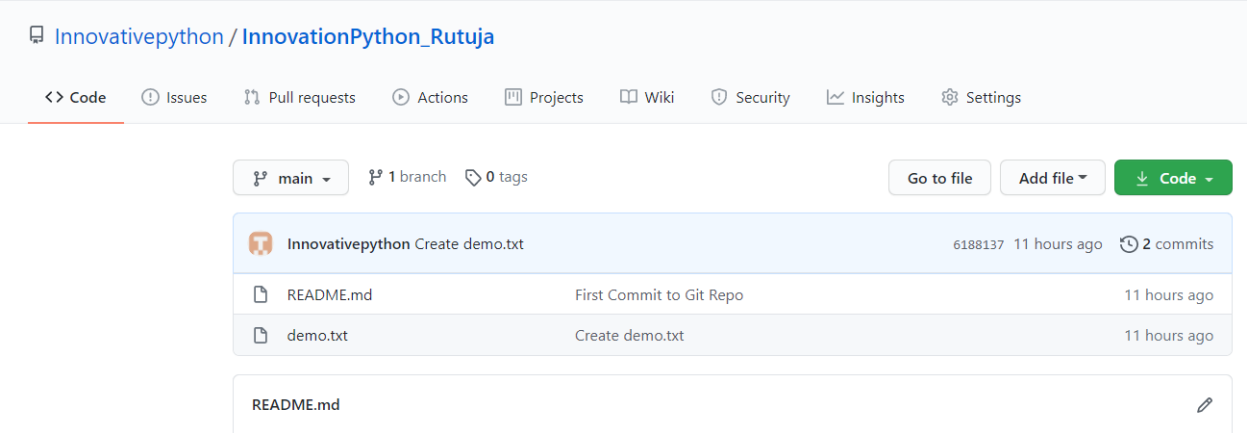
Assignment 1

Git Tasks

Submitted by: Rutuja Dilip Shivgan

1. Make a repository on GitHub with the name **"InnovationPython_yourname"** eg: "InnovationPython_Ankush".



a. Practice on following commands:

■ Git Clone

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (developer)
$ git clone https://github.com/Innovativepython/InnovationPython_Rutuja.git
Cloning into 'InnovationPython_Rutuja'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```

■ Git Diff

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (developer)
$ git Diff
fatal: cannot handle Diff as a builtin

rutuj@MSI MINGW64 ~/Desktop/gitlocal (developer)
$ git diff
```

■ Git Add

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (master)
$ echo "# InnovationPython_Rutuja" >> README.md

rutuj@MSI MINGW64 ~/Desktop/gitlocal (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory
```

■ Git Status

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (master)
$ git status
On branch master

No commits yet

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

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md
```

■ Git Log

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (developer)
$ git log
commit c9275a32cfb94f8b9230fde3a812ee7ee88d3600 (HEAD -> developer, origin/developer)
Author: rutuja shivgan <rutujadshivgan@gmail.com>
Date:   Wed Oct 28 00:50:27 2020 -0400

    Commit from developer branch

commit 6188137e4877ac62fb48dd3e0e7d79f48694d3a8 (origin/main, main)
Author: Innovativepython <73563583+Innovativepython@users.noreply.github.com>
Date:   Wed Oct 28 00:22:27 2020 -0400

    Create demo.txt

commit b143a9795682e58670d603c6b586cf0e9aa2ac79
Author: rutuja shivgan <rutujadshivgan@gmail.com>
Date:   Tue Oct 27 23:46:02 2020 -0400

    First Commit to Git Repo

rutuj@MSI MINGW64 ~/Desktop/gitlocal (developer)
$
```

■ Git Commit

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (developer)
$ git commit -m "Commit from developer branch"
On branch developer
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

■ Git Push

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (developer)
$ git push -u origin developer
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 324.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'developer' on GitHub by visiting:
remote:      https://github.com/Innovativepython/InnovationPython_Rutuja/pull/new/developer
remote:
To https://github.com/Innovativepython/InnovationPython_Rutuja.git
 * [new branch]      developer -> developer
Branch 'developer' set up to track remote branch 'developer' from 'origin'.
```

■ Git Rest

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (developer)
$ git Rest
git: 'Rest' is not a git command. See 'git --help'.

The most similar command is
        reset

rutuj@MSI MINGW64 ~/Desktop/gitlocal (developer)
$
```

■ Git Pull

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (main)
$ git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 697 bytes | 58.00 KiB/s, done.
From https://github.com/Innovativepython/InnovationPython_Rutuja
 * branch            main       -> FETCH_HEAD
   b143a97..6188137  main       -> origin/main
Updating b143a97..6188137
Fast-forward
 demo.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 demo.txt
```

■ Git Branch

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (master)
$ git branch -M main

rutuj@MSI MINGW64 ~/Desktop/gitlocal (main)
$ git remote add origin https://github.com/Innovativepython/InnovationPython_Rutuja.git
```

■ Git Checkout

```
rutuj@MSI MINGW64 ~/Desktop/gitlocal (main)
$ git checkout -b developer
Switched to a new branch 'developer'

rutuj@MSI MINGW64 ~/Desktop/gitlocal (developer)
$ vim demo.txt
```

2. Read about the difference between Git and GitHub

Ans: Git is a version control system which allows user to manage and keep track of your repository history. Github is a cloud based hosting service that let users to manage git repository.

3. Read about Git Workflow:

Ans: Git workflow is a process that makes user accomplish his or her work in a productive and efficient manner. Git has a lot of flexibility hence user can manage the changes.

There are different types of workflows:

1) Centralized workflow:
2) Feature branch workflow
3) Gitflow workflow
4) Forking workflow

1) Centralized workflow: In this workflow, it gives every developer local copy of their project. Hence developers can work independently and they can commit changes to their local repository. This workflow also gives access to merging and branching model. Developer don't have to upstream until they are convinced.
2) Feature branch workflow: The main advantage of this workflow is that all the feature development take place in dedicated branch instead of the master branch. It makes easy for multiple developers to work on different features without disturbing the main code. Developers create new branch every time they work on new feature. Feature branches can be pushed to central repository. It makes develops to share their code with other developers without making any changes in official code.
3) Gitflow workflow: It has 2 branches instead of single master branch to record the history of the project. This workflow provides robust framework for managing large projects. This workflow is suitable for schedule release cycle.
4) Forking workflow: It gives every developer their own server side of repository. Hence each developer has two repositories i.e. private local one and public server side one. This workflow is often found in open source projects. The contributions of every developer can be integrated without the need for every developer to push their work to single central repository.

4. How many types of version control systems are there?

Ans: There are three types of version control systems

1) Local version control system
2) Centralized version control system
3) Distributed version control system

5. Explain Branching concept in Git.

Ans: Branch is a lightweight movable pointer to one of the commits. Branches serves as abstraction for edit or commit process. The default branch name in git is master. The git branch lets you create, list, rename and change the branch.

6. Explain Forking Workflow in Git.

Ans: It gives every developer their own server side of repository. Hence each developer has two repositories i.e. private local one and public server side one. This workflow is often found in open source projects. The contributions of every developer can be integrated without the need for every developer to push their work to single central repository.