1. Install git bash
2. Create a new folder in local drive
   1. **Mkdir QApython**
   2. **cd QApython**
3. initialize a git repository in the root of the folder, run the [git init](http://git-scm.com/docs/git-init) command
4. add a new file to the project, using any text editor you like or running a [**touch**](http://linux.die.net/man/1/touch)command.
   1. **touch newFile.py**
5. After creating the new file, use the [**git status**](http://git-scm.com/docs/git-status)command to see which files git knows exist.
6. To add a file to a commit, you first need to add it to the staging environment. To do this, you can use the [**git add**](http://git-scm.com/docs/git-add) **<filename>** command
7. tell git to package files into a commit using the [**git commit**](http://git-scm.com/docs/git-commit) command.

Run the command **git commit -m "Your message about the commit"**

1. Branches allow you to move back and forth between 'states' of a project. For instance, if you want to add a new page to your website you can create a new branch just for that page without affecting the main part of the project. Once you're done with the page, you can [merge](http://git-scm.com/docs/git-merge) your changes from your branch into the master branch
2. Let's say you are on the master branch and want to create a new branch to develop your web page. Here's what you'll do: Run [**git checkout -b <my branch name>**](http://git-scm.com/docs/git-checkout)
3. use the [**git branch**](http://git-scm.com/docs/git-branch) command to confirm that your branch was created
4. The branch name with the asterisk next to it indicates which branch you're pointed to at that given time.
5. if you switch back to the master branch and make some more commits, your new branch won't see any of those changes until you [merge](http://git-scm.com/docs/git-merge) those changes onto your new branch.
6. Create a repository in Github manually
   1. **New repository QAnew created**
7. GitHub will ask if you want to create a new repo from scratch or if you want to add a repo you have created locally. In this case, since we've already created a new repo locally, we want to push that onto GitHub so follow the '.or **push an existing repository from the command line**' section
   1. **Git remote add origin** [**https://github.com/lasyabheemendra/QAnew.git**](https://github.com/lasyabheemendra/QAnew.git)
   2. **Git push –u origin master**
8. To push changes onto a new branch on GitHub, you'll want to run[**git push**](http://git-scm.com/docs/git-push) **origin yourbranchname**
9. what that "origin" word means in the command above. What happens is that when you clone a remote repository to your local machine, git creates an alias for you. In nearly all cases this alias is called "[origin](https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes)."
10. If you refresh the GitHub page, you'll see note saying a branch with your name has just been pushed into the repository. You can also click the 'branches' link to see your branch listed there.
11. A pull request (or PR) is a way to alert a repo's owners that you want to make some changes to their code. It allows them to review the code and make sure it looks good before putting your changes on the master branch.
12. Create pull request
13. Click on merge pull request
14. Confirm merge In order to get the most recent changes that you or others have merged on GitHub, use the **git pull origin master** command
15. we can use the [**git log**](http://git-scm.com/docs/git-log) command again to see all new commits
16. If we want to switch from branch to master run **git checkout master**
17. To know changes made to file w.r.t saved file run **git diff filename**
18. To compare the state of your files with those in the staging area, you can use git diff -r HEAD. The -r flag means "compare to a particular revision", and HEAD is a shortcut meaning "the most recent commit".

**To clone repository from Github to local computer**

1. Create a respository in github
2. Navigate to desired directory in local computer using terminal command
3. Git clone <url>