NYC DATA SCIENCE ACADEMY

Github Homework

Data Science Bootcamp

Note:

Keep track of your commands in a text file that you can submit.

NYC DATA SCIENCE
ACADEMY

Question 1

1. Create a directory named test.
2. Make test be a git repository.
3. To see the current state of our project check into your project and see what is in test.

NYC DATA SCIENCE ACADEMY

Question 2

1. Create a file named a.txt in test.
2. Add a.txt to staging area.
3. Look at the output of the status command
4. Now commit your file and look at the output of the status command.

NYC DATA SCIENCE ACADEMY

Question 3

1. Create a new directory named subtest in test and add a couple of files such asb.txt,c.txt,d.txt to it.
2. Add files such as a1.txt,a2.txt to test.
3. Add all the files in test to the staging area.

NYC DATA SCIENCE ACADEMY

Question 4

1. Run git status to see what you are about to commit,then commit them to repository.
2. Now let's view all the commits you made so far.
3. Now find out how to make Git display just the 2 most recent commits.

NYC DATA SCIENCE ACADEMY

Question 5

1. Now we create a new empty Github repository test.
2. Connect your own local repository and remote repository.
3. Push your data in test to remote repository.
4. We assume that someone else has pulled your changes and pushed their own commits. You need pull down changes in remote repository.

NYC DATA SCIENCE ACADEMY

Question 6

1. Make some changes for `a.txt` in test.
2. Look at the changes in the working directory that are not yet staged for the next commit.
3. Add the file to the staging area; now see the differences you just staged.
4. Try to show differences between two versions specified.

NYC DATA SCIENCE ACADEMY

Question 7

1.Edit the file `a1.txt` and undo the changes in the working directory.

2.Again edit the file `a1.txt` and add it into the stage,trying to undo the changes in the stage.

3.Edit the file `a1.txt` and commit it to the repository,trying to undo the changes in the repository.

NYC DATA SCIENCE ACADEMY

Question 8

1. You will need to use the documentation for this question. Use the help or look for resources from the online documentation.
2. Create a branch `test1` and switch to the branch.
3. Try to remove all the commits you made so far,and commit your changes.
4. Switch back to the master branch and merge your changes from the `test1` branch into the master branch.

NYC DATA SCIENCE ACADEMY

Question 9

1. Now you can delete your `test1` branch.
2. Push everything you've been working on the local repository to your remote repository.

NYC DATA SCIENCE ACADEMY

Question 10

Work in a small group of 2 to 4 people.

1. First, one person in the group should create a public repository using their GitHub account.
2. The same person should then follow the instructions from GitHub to add a remote repository, and then push data to their repository.
3. All of the other members of the group should then be added as collaborators, so they can commit to the repository as well.
4. Next, everyone else in the group should clone the repository from GitHub. Verify that the content of the repository is what is expected.

NYC DATA SCIENCE ACADEMY

Question 10 cont.

5. One of the group members who just cloned should now make a local commit, then push it. Everyone should verify that when they pull, that commit is added to their local repository (use git log to check for it).
6. Look at each other's git log output. Notice how the SHA-1 is the same for a given commit across every copy of the repository.
7. Two members of the group should now make a commit locally, and race to push it. To keep things simple, be sure to edit different files. What happens to the runner-up?

Question 10 cont.

8. The runner-up should now pull. As a group, look at the output of the command.
9. Look at the git log, and notice that there is a merge commit.
10. You may also wish to view the DAG in gitk.
11. Repeat the last two steps a couple of times, to practice.
12. Now create a situation where two group members both edit the same line in the same file and commit it locally. Race to push.
13. When the runner-up does a pull, they should get a merge conflict.
14. Look as a group at the file in conflict, create a branch to resolve it .
15. Stage the fix,commit it,and then merge it into the master branch.
16. Notice how this procedure is exactly the one you got used to when resolving conflicts in branches.

Bonus

Try to play around with and post your Github.io blog.

NYC DATA SCIENCE
ACADEMY