Git and Pair programming

Problem ID: p01pairprogrammingandgit

In this assignment, you work in groups of 2-3 students each and apply pair programing. Moreover, you need to use git during the development of your solution.

We introduced git, and presented you with step-by-step instructions, in Assignment 5. As a refresher we will present them here below again, updated to reflect today's work, and a little bit improved. But first, here is a quick overview of what to do to start off this assignment.

What to do:

1. One of you should create a private GitHub repository called "`TileTraveller`". Make sure the repo is private.
2. On GitHub in the TileTraveller repo, go to Settings → Collaborators and add the other students as collaborators. This is done because the other students need to be able to clone the repo and push changes to it.
3. Everybody on the team can now clone the repo to their computer with:
   - `git clone remote-repository-URL`
4. You will take turns writing the code, so work should be done on one person's computer first, let us refer to it as computer1. The other team members should take a supporting role, actively aiding the programmer by offering helpful suggestions, and pointing out errors as they arise. As the wikipedia article states:

   > *Pair programming is a [. . .] technique in which two programmers work together at one workstation. One, the* driver*, writes code while the other, the* observer *or* navigator*, reviews each line of code as it is typed in. The two programmers switch roles frequently. While reviewing, the observer also considers the "strategic" direction of the work, coming up with ideas for improvements and likely future problems to address. This is intended to free the driver to focus all of their attention on the "tactical" aspects of completing the current task, using the observer as a safety net and guide.*

5. The team should now design an algorithm as a description for solving the game. Your algorithm should be detailed enough such that the original problem is divided into subproblems that are easier to solve (divide-and-conquer). Put your algorithm description at the top of the program file, your .py file, as a comment.

   Use `git add` and `git commit` to commit your changes locally.

6. Now push your changes to the GitHub repo, and the next person pulls them, `git pull`, to their computer (computer2), switching roles. Then start implementing your algorithm on computer2.

   Start by implementing a skeleton of your program, i.e. the main program that mainly calls functions, but the functions are currently just stubs that do not have any functionality yet. Define these functions so they can be called, but do not write any code in them yet. You can just type `pass` or `return` to satisfy the syntax for your editor, or better yet, write a `"""docstring"""` describing what the function is supposed to do. Make sure that your program runs, but it does not really do anything at this point. Commit and push the changes.

7. Switch roles again, now working on computer1 or computer3. Use `git pull` to get the latest changes. Implement one of the functions. Make sure that the program runs after the implementation. Commit and push changes.
8. Go back to step 7 until all functionality has been implemented. Finally, put the path to your GitHub repo as a comment in your program file.
9. Thoroughly test your program in VSCode. When you are convinced that it works as intended, submit your solution on Gradescope.

Further explanation

Now, for more detailed instructions, if needed: One of you should create a private repository on Github. Follow the instructions that appear on Github.

1. Pick a name for the repo, in this case "`TileTraveller`".
2. Make the repo private.
3. Check the box for making a `README` file.
4. Check the box for making a `.gitignore` file. Select the python template, you can type "*py*" in the search field.
5. You do not really need to protect this homework with a license, so skip that for now.
6. Click "Create repository".

Once you have created the repo, go to Settings → Collaborators and add the other students as collaborators.
This is done because the other students need to be able to clone the repo and push changes to it. They will get an email with an invitation. Accept the invitation.

All team members should clone the repo to their local machines:

1. In the repo home page, you should see a green button near the top, labelled "Code".
2. Click it, and copy the url, https, not ssh.
3. Open a terminal, you can open it inside VSCode if you prefer. For Windows, we recommend git bash, but any terminal should do.
4. Navigate using `cd` to a folder where you want to store the repo on your computer.

   If you are having trouble accessing the correct drive, you can also open the folder in the GUI, right click, and click open "Git Bash Here" or "Open with Code" or similar.
5. Then "`git clone url-to-repository`".

If/When you are prompted to log in, you can use a Personal Access Token (PKA) as the password. You probably already set this up in Assignment 5, but if not, you can generate a new PKA:

1. Go to GitHub.
2. Click on your profile image in the top-right corner and click settings.
3. Under "Developer Settings", bottom left, click "Personal access tokens".
4. Create a new token. Make sure to check the checkbox for "repo", which will also check all the subselections.
5. Set the expiry date after the course is done, 90 days is enough.
6. Give it a description, so you will know why you created this token when you take a look at your tokens at a later time, to help you judge whether you need to revoke them. This can be something like "For my laptop during Programming class 2023" for example.
7. Scroll all the way down, and click "Generate token".
8. Store the generated token somewhere safe. If you lose this token you can never reclaim it, although if you do, you can revoke it and generate a new one.
9. Use this token as your password when cloning, pulling and pushing commits to the GitHub repo from your local machine.

If this does not work, make sure you have the latest version of git installed and try again.
`cd` into the folder git creates for you. Open this folder in VSCode.

1. Now, one of you should write the text of your algorithm in a file called `tile_traveller.py`, as a comment at the top of the file.
2. Inspect the result of "`git status`".
3. Use "`git add tile_traveller.py`" to move changes to the staging area. You can hit Tab for autocompletion.
4. You can also use "`git add .`", notice the dot, to stage all changes to any files in the folder.
5. Commit your changes with "`git commit -m "Describe the algorithm"`".
6. You can see a log of all your commits with git log. Hit "`q`" to escape the log interface.
7. You can use "`git log -oneline`" to make the history more compact.
8. Push your changes to the Github repo with "`git push`". Inspect your commits on Github.
9. Have the next person pull the changes from Github with "`git pull`".
10. Then start implementing your algorithm, refer to the instructions above on how to apply pair programming.
11. During your implementation, make sure you do "`git status`", "`git add`", and "`git commit`" regularly, "`git log`" is useful too.
12. Take care to write good commit messages, in English, that describe what the change involves.
13. Use imperative mood in commit messages. For example, "Fix bug", and not "Fixed bug" or "Fixes bug".
14. Also inspect your commits on Github, and get familiar with the UI.