

**Project 1.3.9a Tools for Collaboration**

Introduction

|  |  |
| --- | --- |
| Version control is a powerful tool for developing any project or product. If you are collaborating with one or more other people, it is even more important. Have you ever found it difficult to keep papers well organized? What happens when 20 people all work on the same document? | Github.com/about  Figure 1: The GitHub Team |

Materials

* Computer with Enthought Canopy distribution of Python® programming language, Internet browser, and GitHub client software
* GitHub individual account and membership in organizational account's team

Procedure

**Background Information**

Frequent changes and testing are important during development of a digital product. A **version control** system allows developers to keep track of those changes. A **distributed version control** system allows multiple developers to work on the same project and keep track of everyone’s individual changes, even if they are not on the same network.



There are many different distributed version control systems, including BitKeeper, Mercurial and Git. Git was developed by Linus Torvalds for use during development of the Linux operating system kernel, and has since become the most widely used distributed version control software.

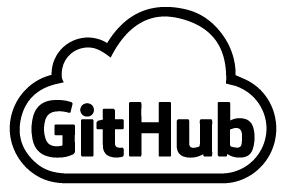
Figure 2: Git Logo (git-scm.com)

Git works by tracking **repositories**, which are folders that contain code and related files and documents for a project. When developers make changes they record them with a **commit**. A commit is a checkpoint in the development of a project.

**Github** is an online Git repository hosting service that makes it easy to collaborate on a project with multiple developers. Github consists of an online web portal and a local client program with a GUI (instead of running all the commands at the command line).

**Workflow**

There are three distinct parts to the software you will be using: The Git shell (the command line for Git), Github for Windows Client (the GUI for Git on your computer) and the Github web interface (online at [github.com](http://github.com/)). This document will indicate what you need for each step with the icons shown below:



This symbol means use the Git shell:This symbol means use the Github For Windows Client:This symbol means use the Github online interface:

Github hosts many different repositories for projects, many of them **open source**. Open source projects make their source code available to anyone, usually free of charge, and many people often contribute modifications and enhancements to the project. Most repositiories on Github are shared as read-only, meaning others can view and copy the project, but cannot modify it without the approval of the owner or manager.

When a developer makes a copy of a shared read-only repository, they create a **fork**. (Think of a tree: the main trunk is the original repository, and it can split off into another direction at a fork.) A developer can then make changes to their copy, and if they change something that they think should be incorporated in the main project, they can make a **pull request**, asking the project owner to “pull” their changes and **merge** them into the original code.

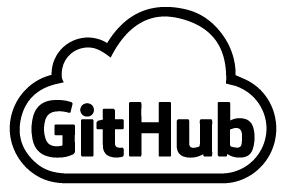
A repository is always kept both on the developer’s computer (**local**) and online on the Github servers (**remote**). Once a repository is forked, it is then cloned to the local computer so the code can be worked on. As changes and commits are made, the local repository is synced with the remote repository to keep the two copies “in sync” with each other.

A repository can also have multiple strands of development going on simultaneously. These are called **branches** . (Think of the tree again: a trunk or fork of a trunk can split off into multiple branches.) Every repository has a master branch, and can have any number of other branches. Often times there are branches for things like bug fixes, testing, and enhancements. At anytime these branches can be merged back into the master branch, just like a fork can be merged back into the master.

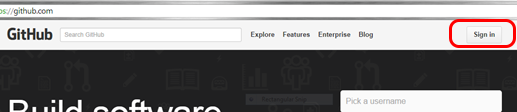
Now it’s time to put all this to work!

**Part I. Setup**

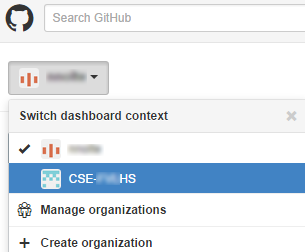
Your teacher has created a repository and shared it with you through your school’s organizational account on Github. You need to **fork** it, **clone** it, and make a **branch** for testing.



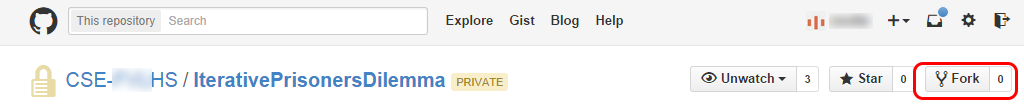
1. Create a fork of the repository.
   1. Go to [github.com](http://github.com/) and login to your account.

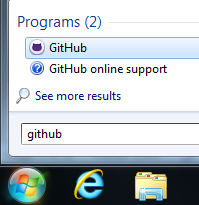


* 1. Switch to your High School organizational account.



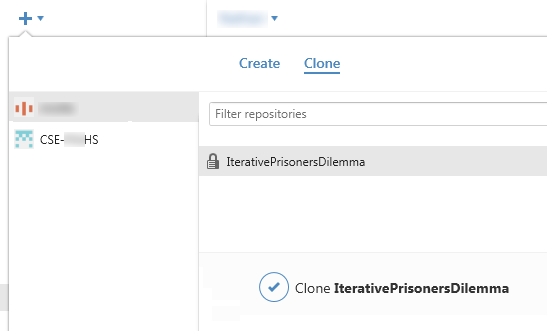
* 1. Click on the IterativePrisonersDilemma repository and click “Fork” to make a copy of it in your account.



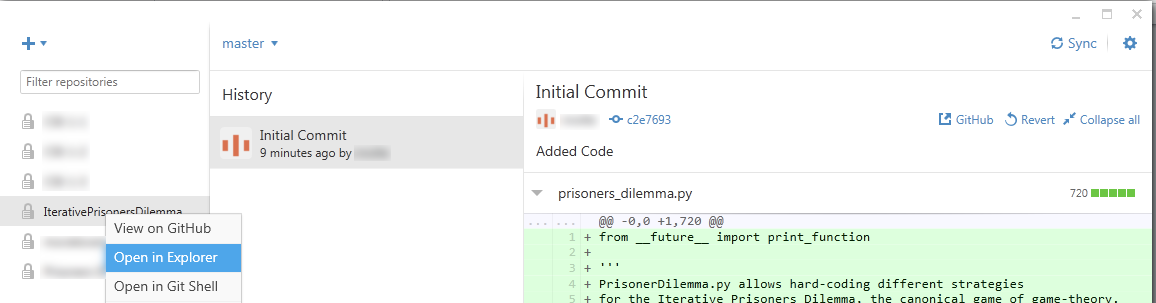




1. Clone the repository to your computer.
   1. Open the Github For Windows Client on your computer.
   2. Go to Tools 🡪 Options and log in to your Github account.
   3. Click on the “+” symbol, choose “Clone”, click on the IterativePrisonersDilemma repository and click the checkmark at the bottom. Choose a location to save the repository as directed by your teacher.

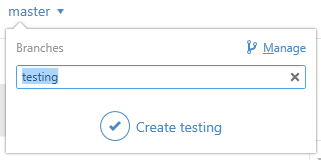


* 1. Once the repository is cloned, you will see it in the list on the left of the window under the “+”. Left-clicking on it opens it in the client. Right-clicking on it gives you several options, including opening it in Explorer. The history probably only has one commit, because that is when your teacher published the original repository. Click to open the repository in Explorer.

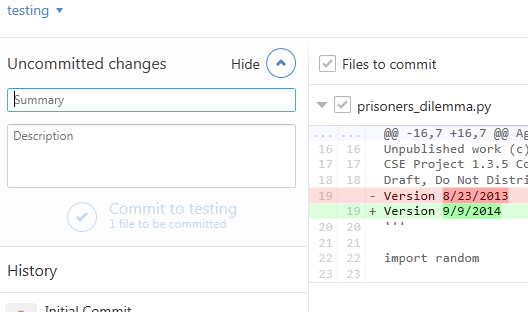


* 1. You should see the prisoners\_dilemma.py file in the Windows Explorer window. You may also see a hidden .git folder. This is where important information is kept that Git uses to keep track of changes. Don’t make any changes to the files in this folder.

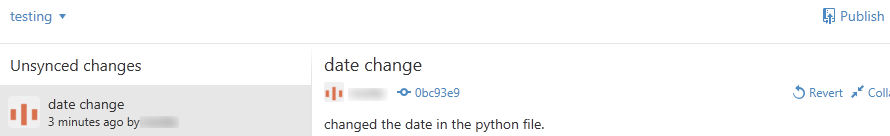
1. Make a “Testing” branch and a commit.
   1. Back in the Github For Windows Client window, note the word “master” with an arrow next to it. That indicates you are working on the master branch. The master branch is what we will eventually make our changes to in order to submit a pull request to merge our code with the rest of the class. For now, we want a “testing” branch that allows us to explore and make changes to the code without affecting the master branch. Click on the arrow, type “testing” in the box, and click the checkmark to create a new branch.



* 1. Note that it nows says “testing” where before it said “master.” That means you are on the testing branch, not the master branch. Go back to the Windows Explorer window that has the prisoners\_dilemma.py file in it and open the Python file with Canopy. Change the date in the Python file at the end of the multiline comment. Save the Python file.
  2. Go back to Github For Windows. It should now show that there are uncommitted changes to the repository. Click on the arrow next to uncommitted changes, and it will show you a preview of the file with the date change you made.



* 1. In order for Git to make a checkpoint of this change, you need to make a commit. Enter a summary (“Date Change”) and description (“Changed the date in the Python file.”) and click “Commit to testing.”
  2. You now have a record of that change in the testing branch. However, the master branch is still unchanged. Click on the arrow next to testing and switch back to the master branch. You will see that there are no changes to that branch. Also, when you go back to Python, you will see that the change you made has disappeared! Git actually changed the file on the computer to reflect what is current in the master branch. The change still exists – try changing back to the testing branch. You can see how you can make changes to code without affecting the master version of the code and easily switch between the two.
  3. One last step – you need to sync your local changes to the remote repository on Github. Since you have not synced since you created your new branch, it says “Publish” in the upper right hand corner. Click on this to send your new branch and commit up to the remote repository.

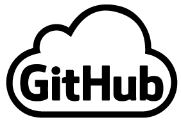


* 1. Open the document 1.3.9.b IterativePrisonersDilemma.docx and explore the code and make changes. Don’t forget to ensure you are working in the testing branch and make commits and sync as you work.

**Part II. Contribute Your Strategy to the Class Collaboration**



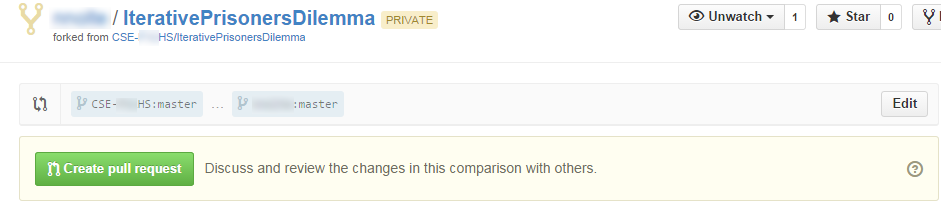
1. You should have your IPD strategy in the Python file within the master branch. Once you save the Python file you will need to make a commit in the Github for Windows Client. Sync your repository so the remote and client are current.



1. Open the repository on Github by either right-clicking on it in the list or going to [github.com](http://github.com/).
2. Create a **Pull Request** so your changes can be merged into the original code.
   1. Click on the green symbol as shown to enter the compare and pull request screen:



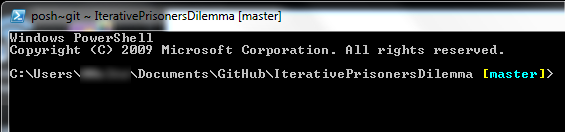
* 1. The top line shows what two things you are comparing. In this case it should show the master branch of the original repository in your school’s organization account (the base fork) and your master branch in your forked copy of the repository (the head fork). If it does not, click on the “Edit” button to change it.



* 1. Click on “Create pull request” and add in a title a comment (much like a commit message). In the pull request type the names of your group members. Finish by once again clicking “Create pull request.”

Your teacher will now get a message that someone has code they would like to add into the original repository. If everything was done correctly, your team’s code will be able to be merged into the original along with the code of every other team. If someone edited the wrong location in the file, there may be a merge conflict that needs to be fixed. Merge conflicts and how to fix them are beyond the scope this activity.

1. The last thing to do before you can run the tournament is to get all the changes made to the code into your copy of the code. To do this, you will need to resync the original fork you made, and you will need to use the Git shell.
   1. Open the Git Shell either by opening the program indicated by the icon or by right-clicking on the repository in the Github for Windows Client and choosing “Open in Git Shell.” A PowerShell window should open, and the path should be the location of your local repository followed by [master].



* 1. Run the command:

**git fetch upstream**

This gets the code from the original master branch, which should now have every teams’ code incorporated. Next, run the command:

**git merge upstream/master**

This command updates your fork to be the same as the original repository.

* 1. You are all set! You can return to part V of 1.3.9.b IterativePrisonersDilemma.docx to run the tournament.