

# Tools for Collaboration

## Introduction

Controlling many different versions of a program can be difficult and confusing. Using a tool to help with version control can be a powerful addition to any project development. 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?



The GitHub Team

## Materials

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

## Resources

[1.3.9-10 sourceFiles.zip](#)

# Procedure

## Part I: Set up a GitHub repository

1. Compare tools for collaboration. GitHub is especially useful as a tool for collaboration when a large group of people are working on a development project where files depend on each other. Describe the circumstances when another tool for collaboration is ideal, such as DropBox, Google Drive, Google docs or sheets, or some other tool you have experienced.
2. Learn vocabulary. Alternating with your partner, read the following background information aloud and then answer the questions in a and b.

Frequent changes and testing are important during development of a digital product. A version control system allows a person to create software or any other product and keep track of those

changes. A distributed version control system allows a group of people to work on the same project and keep track of everyone's individual changes.




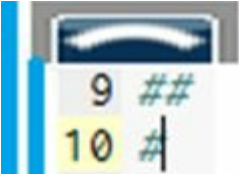


There are many different distributed version control systems, including BitKeeper, Mercurial and git. Git was developed by Linus Torvalds for use during development of the Linux operating system kernel, and has since become the most widely used distributed version control software.

Git works by tracking a **repository**, which is a folder containing subfolders with files for a project. The files can be code, images, sound, or anything else. A developer can change the files and save them using the operating system or an application, just as a user usually does. When the developer wants to permanently record the current version of one or more files, she records it with a **commit**. A commit is a checkpoint in the development of a project.

GitHub is an online service that hosts git repositories, layering on additional features such as issue tracking, project web pages, and visualizations of who is doing what. These tools make it to collaborate. GitHub consists of a web portal to your online repos and a local client program that can access both your online repos and your local repos. The local client lets you use git with either a graphical user interface or a command line interface.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

There are three distinct parts to the software you will be using: The git shell (the command line for git), GitHub Client for Windows or Mac (the GUI for git on your computer) and the GitHub web interface (online at [github.com](https://github.com)). You will also work with three more familiar tools. This document will indicate what tool you need for each step with the icons shown below:

| Shell   | Client  | Web   |
|---|---|---|
|  |  |  |
| This symbol means use the git shell   | This symbol means use the GitHub Client for Windows or Mac                          | This symbol means use the GitHub web browser interface                                |
| Code Editor   | IPython   | OS Explorer   |
|  |  |  |

|   |   |  |
|---|---|--|
| This symbol means use the code editor in Canopy | This symbol means use the IPython session in Canopy | This symbol means find the file in folders with the operating system |
|---|---|--|

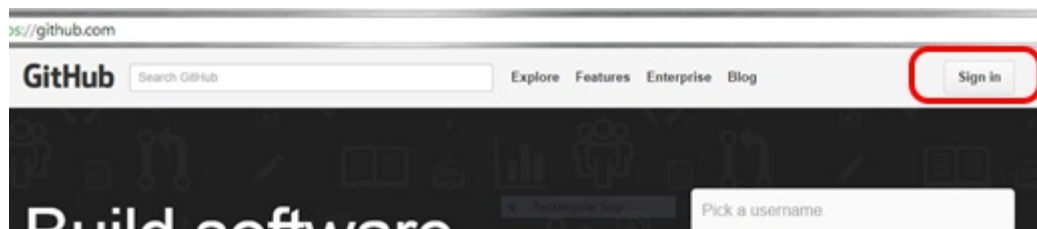
- Git tracks a collection files. What is the collection called?
- Git keeps track of old versions of files. Not every version that is saved is remembered, however. Git only remembers versions that are saved and \_\_\_\_\_. [Version Control: Part 1: Editing and Publishing Files reader\\_4.html](#)

3.

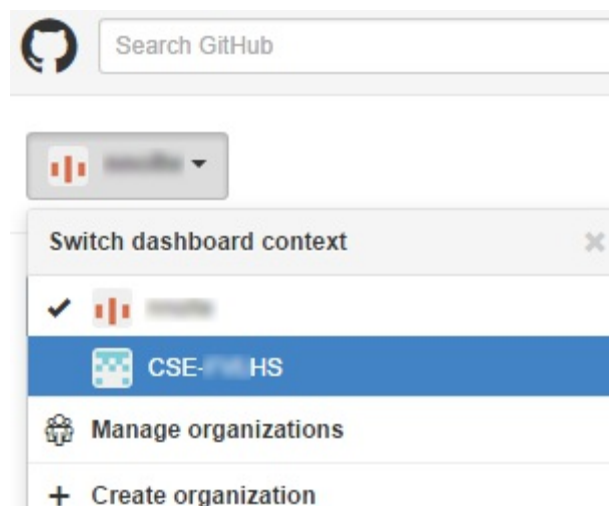


Fork a repo. Your teacher has created a repository for your class and shared it with you through the teacher's organizational account on GitHub. You need to **fork** it, clone it, and make a **branch** for testing. Create a fork of the repository.

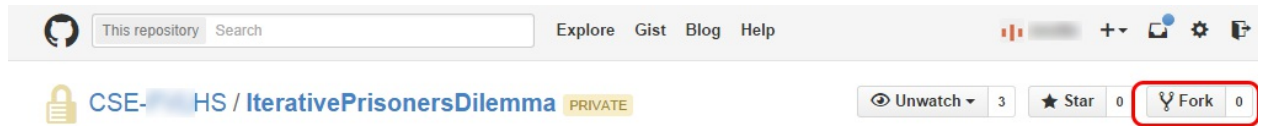
- Go to [github.com](https://github.com) and login to your account.



- Switch to your teacher's organizational account.



- Select the IterativePrisonersDilemma repository and select **Fork** to make a copy of it in your account.

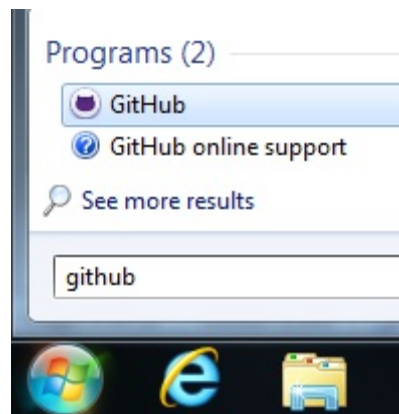


#### 4. Clone a repo

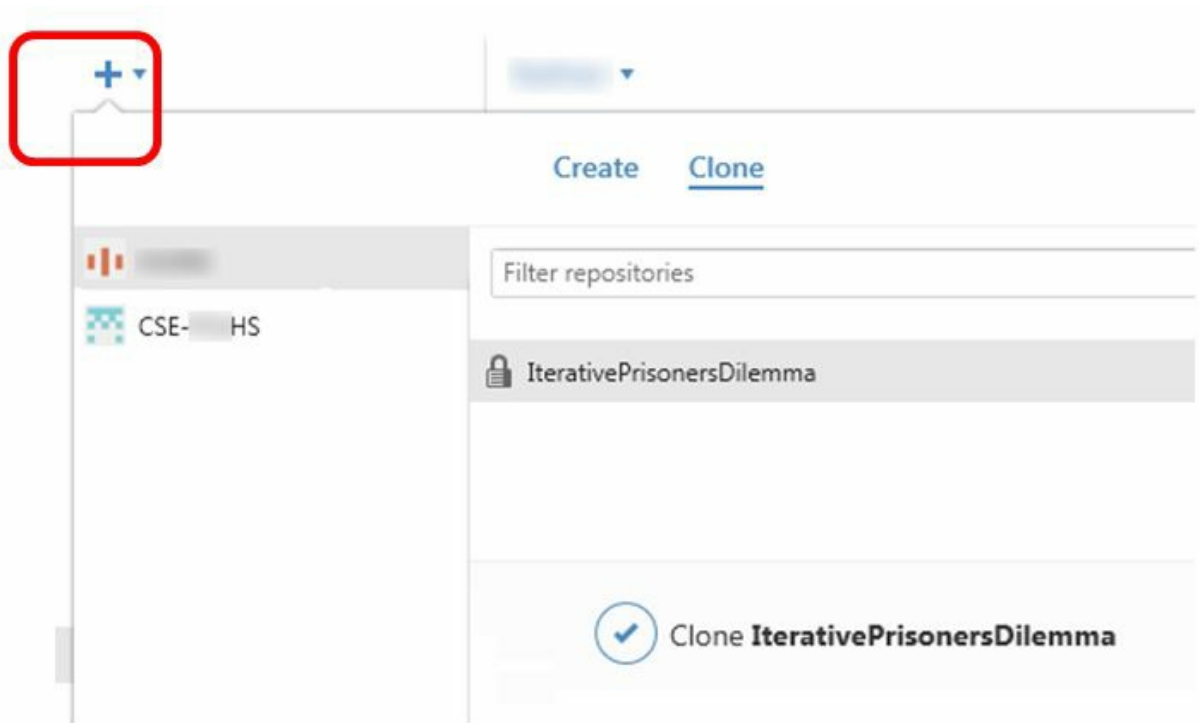


Clone the repository to your computer as follows.

- Open the GitHub Client on your computer.

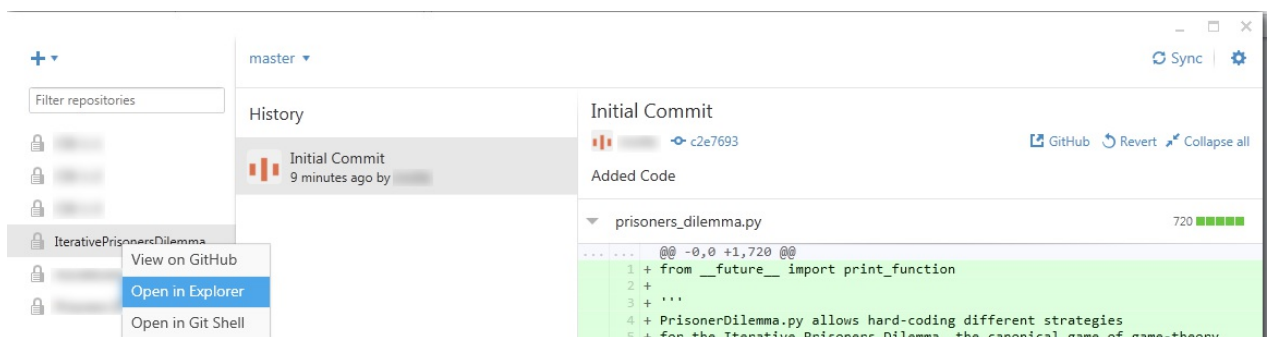


- Select **Tools > Options** and log in to your GitHub account.
- Select the + symbol, choose **Clone**, select the IterativePrisonersDilemma repository and select the checkmark at the bottom. Choose a location to save the repository as directed by your teacher.



o

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. Right-click to open the repository in Explorer.



## Part II: Make some changes

5. You will be making changes in a branch on the local repository. Alternating with your partner, read the following aloud and then answer questions a and b.

GitHub hosts many different repositories for projects, many of them open source. Open source

projects are public and make their source code available to anyone, usually free of charge, and many people often contribute modifications and enhancements to the project. Most repositories on GitHub are shared as read-only, meaning that anyone can view and copy the project, but these people with read-only privileges will not be able to modify the project 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 in the park: 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 GitHub 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 files can be modified. 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. Similarly, the remote repository can be caught up to the original, upstream repository.

A repository can also have multiple strands of development going on simultaneously. These are called branches. (Think of a tree in the park again: a trunk or fork of a trunk can split off into multiple branches.) A repository can be forked from another repository, and each 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 any time these branches can be merged back into the master branch, just like a fork can be merged back into the upstream repo.

- Use one word in each blank:

The local, remote, and upstream \_\_\_\_\_ can each have multiple \_\_\_\_\_.

When a participant in a collaborative group on GitHub is ready to have their work used by the group, the participant makes a \_\_\_\_\_.

- When working with a repository, a developer typically works with three copies. Match each word or phrase on the left with one of the words or phrases on the right.

Original

Fork (also Remote)

Online copy

Upstream

Local copy

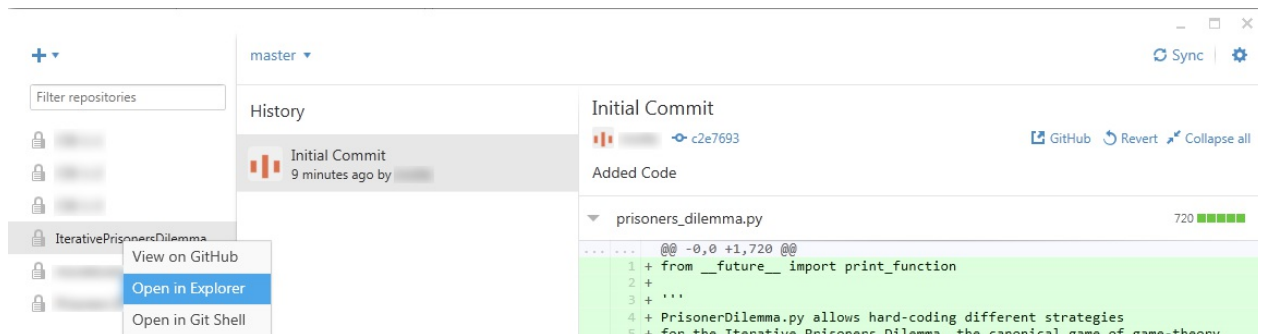
Clone

6.



**Explore a repository.** Once the repository is cloned, you will see it in the list on the left of the window under the “+”. Left-clicking on the repo name opens it in the client. Right-clicking on the repo name provides several options in the context menu, including opening the repo folder in the operating system's Explorer. The middle panel, History, has one or more commits that include your teacher initially creating the repository. Selecting a commit shows the files that had new versions saved as part of the commit.

- Select one of the commits. What are some of the file names that had versions stored as part of that commit?
- Right-click the repo name and open the repository in Explorer.



You should be able to find the `prisoners_dilemma.py` file in an OS Explorer window.

- You may also see a `.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.

## 7. Save and commit a change.



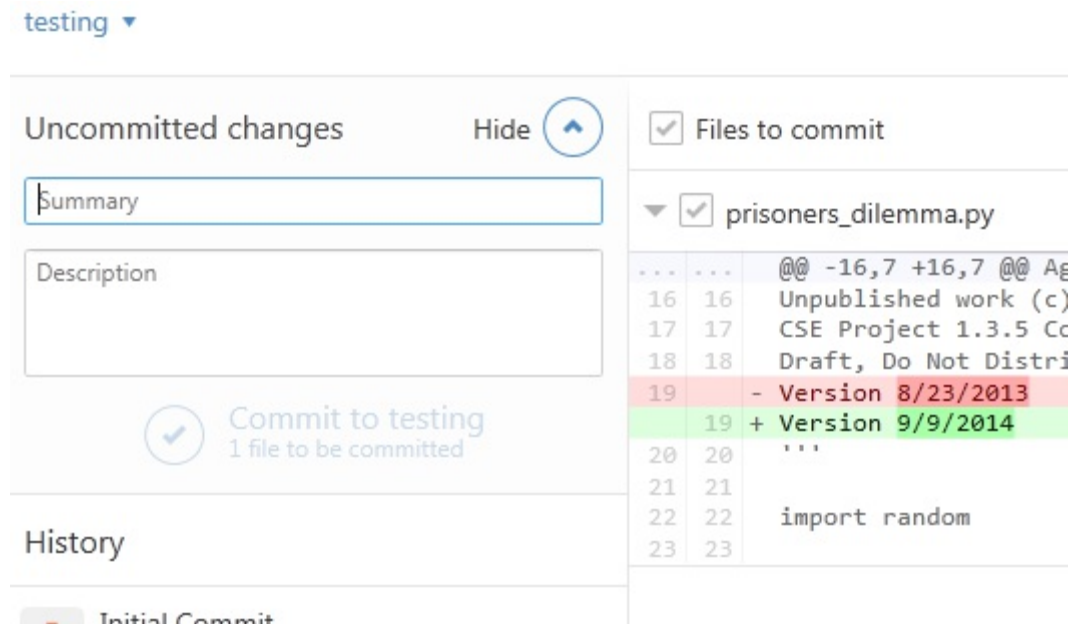
In Canopy, open the team file assigned to you by your teacher. The `teamNN.py` file is in the folder for your local repository.

- In that file, assign a different string to the variable `team_name`. The string can be any team name you choose.

```
team_name = 'The name the team gives to itself'
```

- Save the file.
- 

Go back to the GitHub Client. That program should now show that there are uncommitted changes to the repository. Select the arrow next to uncommitted changes and you should see an excerpt of the file showing changes you made.



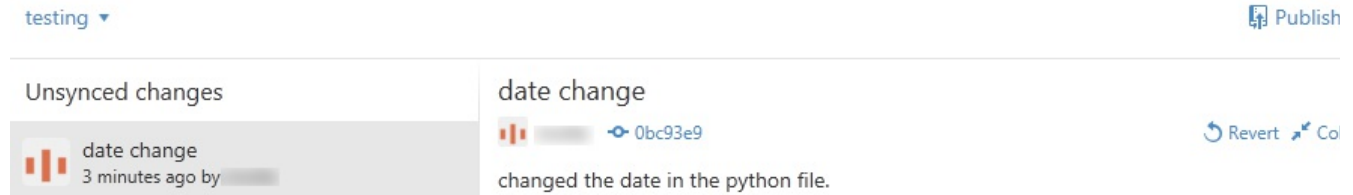
- Although your changes are saved on your local computer, they are not stored permanently by git. To create a permanent checkpoint of this change, you need to make a commit. Enter a summary ("Team 4 names") and description ("Individual and team name") and select **Commit to testingxx**.
- You now have a record of that change in the testingxx branch. However, the master branch is still unchanged. Click on the arrow next to testingxx and switch back to the master branch. You will see that there are no changes to that branch. Also, Canopy may warn you that the file has changed on disk; 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 testingxx 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.

## Part III: Share your changes

8. 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.



9. Highlight the code for the strategy that you are going to use in the tournament and copy it to the clipboard (**Edit** → **Copy** or **<Ctrl>+C**).
10. Switch to the “master” branch of your fork of the IPD Repository:
  - In the GitHub for Windows Client, Click on the arrow next to “testing” and choose “master.”
  - Canopy will automatically update the code file you have open, showing the original version of the code.
  - Find the location for your assigned team number, and paste in the copied code. Be careful not to make any changes anywhere else in the code.

11.



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.

12.



Open the repository on GitHub by either right-clicking on it in the list or directing a Web browser to [github.com](https://github.com).

13. Create a **Pull Request** so your changes can be merged into the original code.

- Click on the green symbol as shown to enter the compare and pull request screen:



- 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.



- Click on “Create pull request” and add a title and comment, similar to what you did with a commit by adding a commit message and comment. In the pull request type the names of your group members. Finish making the pull requests by again selecting “Create pull request.”

14. Your teacher will now get a message that someone has code they would like to merge into the original repository. If everything was done correctly, your class's code will be easily merged into the class repo's current version (the head commit) on the master branch. The current commit in that master branch will have the original code modified with the code of every other team. If two different developers both make changes on the same line of code, merging the commits requires work in the git shell—skill beyond what you will learn in this course.

Was the teacher able to easily merge your code with the code from others merged before you?

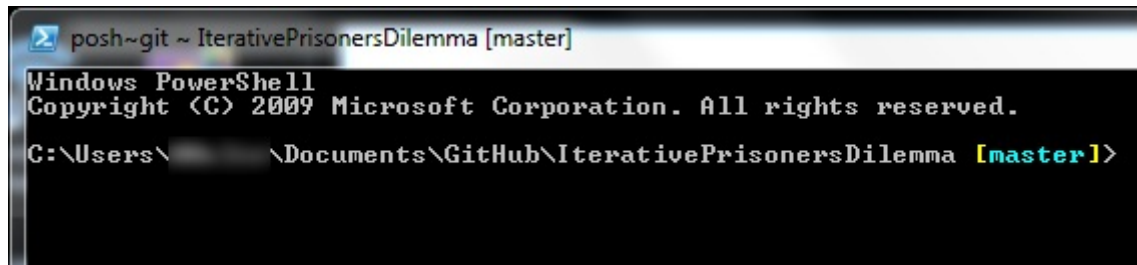
- 15.



Before you use the repo for the tournament in the next project, you will want to get all the

changes made to the class' repo into your copy of the code. To do this, you will need to resync the original fork you made. You will need to use the git shell.

- Open the git shell by right-clicking the repository in the GitHub for Windows/Mac 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].



- Run the command:

```
git fetch upstream
```

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

```
git merge upstream/master
```

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

You are all set for the upcoming tournament!

## Part IV: Contrast git with other collaboration tools

16. Many tools for collaboration allow multiple people to work together on a set of documents. For programming, it is not sufficient for a tool to record each version of file A and each version of file B. Any collaboration tool for programming together with a large group must which version of file B existed at the same time as each version of file A. Explain why that would be even more important with code than with other types of collaboratively produced documents.
17. GitHub public repos are commonly used for free open source software (**FOSS**) projects. Open source means that software code can be inspected by anyone, and is the opposite of closed source. Open source code does not have to be free, which means available to use without having to pay any money, the opposite of the traditional commercial rights-reserved copyright licensing. FOSS projects are public community projects, often controlled with democratic processes; the *Python* programming language is an excellent example of that. FOSS projects are licensed under Creative Commons licenses or other similar copyright licenses that place contributors' work in the Commons without charge. The following four statements represent both sides of two separate debates. Use class discussion or the Web to identify a reasoning and an anecdotal argument in support of each of the four positions.

| Position  | Supporting argument by reasoning | Supporting argument by anecdote |
|---|----------------------------------|---------------------------------|
| 1. Open source software is more secure than closed source software.                                 |                                  |                                 |
| 2. Open source software is less secure than closed source software.                                 |                                  |                                 |
| 3. Commons licensing motivates higher quality and quantity for peoples' creative work.              |                                  |                                 |
| 4. Commercial copyright licensing motivates higher quality and quantity for peoples' creative work. |                                  |                                 |

18. Even with all the Internet-capable ways of collaborating, it is still recommended that a team of software developers be located in the same place to collaborate. Why do you think face-to-face collaboration is still considered superior?
19. From the second column in the table below, identify a tool for collaboration that is familiar to you and describe it to your partner. From a classmate, a teacher or parent, or the Web, learn about at least one tool that you didn't previously know about.

| Typical Purpose of Tool                 | Examples   |
|---|--|
| Shared storage                          | Google Drive, Dropbox, Hightail                          |
| Simultaneous editing of prose documents | Google docs, Microsoft 360                               |
| Code version control and issue tracking | GitHub, Mercurial, BitBucket                             |
| Schedule/event planning                 | SignUpGenius, Facebook events, Doodle Poll, Evite        |
| Crowdfunding                            | Kickstarter, Indiegogo, GoFundMe                         |
| Crowd sourcing of work                  | SurveyMonkey, Amazon Mechanical Turk, Zooniverse         |
| Crowd sourcing of writing               | Wikipedia, Stack Overflow                                |
| Videoconferencing                       | Skype, Zoom, GoToMeeting, Adobe Connect, Google Hangouts |

For the tool that you learned about, describe the tool in writing.

## Conclusion

1. What is version control?
2. Why is version control so helpful when collaborating with people?
3. How has the Internet changed the way people collaborate?