Unit 1 Lab 2: Gossip, Page 1

August 7, 2021

## Pair Programming

**On this page,** you will learn about collaborating by using pair programming.

In the early days of computers, programming was something you did alone. You planned the program in your head, and then you sat down and wrote the code.

The programs people write these days are far too complicated for that. Graphical user interfaces, interacting with servers on the Internet, internationalization, security against malware attacks, and many more requirements are part of a serious software development effort, and a program is likely to have *hundreds* of authors.

This means that *collaboration skills* are as important as coding skills today. Even if you don't end up working as a programmer, collaboration is part of most jobs.

There are several ways to organize a collaboration. In this course, you'll use a collaboration technique called [*pair programming*](http://en.wikipedia.org/wiki/Pair_programming)—writing computer programs in teams of two. One partner, the *driver,* controls the computer. The other, the *navigator*, helps catch errors and keeps track of how the code being written fits into the overall plan. The navigator is also in charge of reading the lab pages out loud. Aim for equal time in each role, switching about every five minutes.

#### **How *not* to do pair programming:**

The biggest mistake in pair programming is for the navigator to tune out and let the driver do all the work. The navigator has to be paying attention; otherwise you're just taking turns doing the entire work instead of making use of the benefits of having two people.

#### How to collaborate in pair programming

* **Keep talking.** The driver should constantly explain the reason for using each block. The navigator should interrupt with questions: What about doing it this other way? How does this program avoid such-and-such bug? Don't we have to deal with this requirement and that requirement at the same time? We're pretty much repeating this other piece of code over there; could we hide that code in a block and use the same block in both places? And so on.
* **Listen.** Try to understand each other's thinking before you disagree.
* **Work as a team.** If you find yourself being defensive about feedback, remember that nobody's code is perfect right away. It's the navigator's *job* to find problems in the driver's code, and when you switch places, the new navigator will repay the favor. You're a team, not a competition.
* **Work toward consensus.** Don't bargain; if you have disagreements about how something should be done, don't say "well, you can do this your way if I can have this other thing my way." Both partners should listen to each other's *reasons* and work together to find a solution that meets both needs.
* **Speak your mind and work out your differences.** That doesn't mean you should *try* to disagree, but if you do, you can still be friends. Work out the conflict by listening and understanding each other, rather than by giving in just to end the argument.
* **Regardless of experience, take turns.** Don't let the more experienced programmer overwhelm the other partner. Make sure both partners are comfortable contributing, in either role. A group is almost always better at problem-solving than any of the people in the group is alone.

In the first unit, we'll use these symbols to remind you to switch roles at specific points in the lab. After this unit, you're on your own to be sure to switch about every five minutes.  
Pair Programming Swap Pair Programming Swap

1. Your teacher will hand out instructions for the Lost on the Moon game.