**Exercises 1 - 1 - 3**

1. Working in pairs, modify your script so that the sprite wraps around when it reaches the left side of the screen.

a. One partner in your pair should stop working at their computer and become the “navigator.” The navigator will look at the screen while their partner “drives.” The driver controls the mouse and keyboard. The navigator thinks about how the program will work, advising and coaching the driver. The navigator can point at their partner’s screen but should not touch the keyboard or mouse. The driver should stop and listen to the navigator. You can flip a coin to decide who gets to navigate first. Switch every 10-15 minutes. Together, modify both of your programs.

b. Once you’ve changed your code a little bit, test it. Usually, it won’t do what you think, and you have to figure out why it is doing what it is doing. That is a **bug**. Fixing the parts of a program that don’t work right is **debugging**. Even professional programmers write programs with a lot of bugs and spend a lot of their time debugging. It can be fun! The more frustrating it is to find and understand the bug, the more satisfying it is when you figure it out - like a puzzle!

c.

**Conclusion Questions**

1. Why is it a good idea to write programs in small pieces and check how well they work after each piece is written?

2. Professional programmers usually have to write a piece of a program several times before they get it right. Think of other tasks in life that require you to figure things out. How is programming similar or different?

3. Reflect on how well you worked together when pair programming. What could you do next time to make a partnership function even better?

4. The introduction to this activity said, “Even the most complicated software or cell phone app is made out of very simple steps that happen one at a time.” That is not quite true. Each **central processing unit core** **(CPU core)** can process one instruction at a time. A quad-core computer processes four instructions at once. But any computer can run more programs “at once” than the number of processor cores it has. How do you think a dual-core computer, for instance, can make it look and feel like it is running ten or more programs at once?