An important skill to have while programming is being able to apply example code/concepts to different situations. Like many other subjects, if all you can do is use an exact example in its exact situation, it may not do you any good elsewhere. However, if you not only understand the basic example, but also its underlying principles, you can apply it to different situations.

**General process:**
When tasked with a daunting program, a general process we learned is to break it down into chunks. Think about the different parts that need to fit together to get to the end goal before you begin writing any code. In this way, you can focus on little pieces, and each one feels like a tiny victory when trying to complete the whole.

There are always multiple ways and better ways to write your program. When we first started writing programs, we learned some of the basic functions and elements that go into programming. As we got into harder and longer programs, we learned selection statement, loops, and how to create functions that would allow us to get more out of our code (Less code to have your program run). This was all necessary, otherwise we could be writing 100 lines of code to make a program run, when in reality that same program could run with 25 lines of code written. So helping us become more knowledgeable and efficient coders, I think helped us learn more about the general process of programming.

There are also many skills that come to mind that are not necessarily computer science specific skills, but are very helpful to have in programming.

- Organizational skills - knowing how to keep track of your work and also keeping your work organized for others.
- Debugging/Troubleshooting skills - What can you try? What isn’t working? What am I missing?
- Perseverance skills, but also knowing when to step away for a bit.

**Our Process:**
Things that I learned about my process of programming, were that I felt like I tended to write my code based on what I was comfortable with. Looking back at some of the projects that we did, I probably could have and should have written my code to be more efficient. I really felt as though when I would sit down to work on the practice programs and the individual programs, that I would try to visualize the code before I would even start typing. I would try to think to myself how it should look and what was I trying to accomplish. I found this worked out sometimes and other times I don’t think I got anywhere with it. What I probably should have done was write what I thought it should look like, test it, and then try to make corrections from there. I also learned that my process was probably a lot slower than most people’s and I spent a lot of time working on the programs to get them to work. I remember Ben saying that if you can’t figure something out within 15-30 minutes to take a break and walk away from it, otherwise you will just be wasting your time. It was hard for me to walk away from the code
once I started, so I probably wasted way too much time throughout the semester looking at a screen hoping something would click in my mind. These are things that I think will become easier for me to do as I code more and teach coding, but as a beginner it was something that was very difficult for me.

One thing I learned about my process is to run many trials throughout. Most of the time, this is by executing print statements to test each “chunk” of my code. Once problems became more difficult, it was very hard to pinpoint where I was going wrong without looking at each piece individually. Something else I learned was when things get so difficult it feels like I’ll never finish, I need to step away and/or ask for advice from my peers/instructor. On numerous occasions, an idea has popped into existence after I have stepped away from constantly staring at my computer screen.

I didn’t realize how systematic I am when it comes to programming, until I worked on the paired programs. I try to visualize and figure out all of the necessary math formulas before diving into the program. Once I had a pretty good idea of what needed to happen then I could start at the beginning and work my way through. I didn’t really like to test programs out until the end, probably because I am still very much a beginner and need to keep track of where I am in my program.