Whether you are programming with VS Code, writing some new cutting-edge Apex code for your Salesforce org, or using Flow Builder to automate tasks to save your company time and money, you are programming. As a programmer, you need to think like a programmer.

What does it mean to think like a programmer? What does a Programmer do, and how does one think like a programmer? In the most basic terms, a programmer solves problems. I saw this quote about a programmer's work, and I think it fits here.

> Make it work, make it right, make it fast.
>
> *-Kent Beck*

In Kent's quote, his first step is to make the program or app work, and I think there is a step or two before this. To make the program work, you must thoroughly understand the problem that you are solving and the acceptance criteria to determine if the problem is solved. In other words, you must be able to define what solved means. What is the acceptable endpoint to the problem? Is this app going to update a record or create a new record? What is the expected outcome?

Once you understand the problem, you must develop a solution. To do this, you must consider all the pieces involved in the problem. In Salesforce, you need to know what Objects and Fields are needed and how the app will interact with the Objects and Fields. If you are writing a Flow to update records, how will you get the new information from the user? A Screen Flow is one way to do this. Once we gather the information, what do we do with it? In this example, we must Update or Create a record. Every piece of the puzzle should be thought about before you start writing code or creating a Flow. By thinking through each puzzle piece, you will be more efficient when you write the code or start building the Flow.

Now that we understand the problem and know how to solve it, we write the code or design the Flow in Flow Builder. Great, problem solved, I can move on to all the other tasks I have to do and know that the problem is solved, and I never have to think about it again. Wrong. Now that we have a program or Flow created, we must test it. Testing our code or Flow helps us find issues in the code before the Users do. Great, we tested everything multiple times, and everything works. Done. We solved the problem and tested our solution repeatedly, and it worked; and we can stop worrying about it and move on.

Wrong. Once we are solved the problem, written the code, and built the Flow, we need to

refactor it. Refactor, what does that mean? Well, according to Wikipedia:

> "Refactoring is intended to improve the software's design, structure, or implementation while preserving its functionality."
>
> -*Wikipedia*

In Kent Beck's quote above, this is the make it right part. Just because our current solution solves the problem does not mean it is the right or best solution. The best practice for a programmer is to go back over the problem and the current solution and develops ways to improve it. Can the solution be done better or more efficiently? If so, you need to rewrite your code or redesign your Flow to make it the best possible solution you can. Great, we went over everything again and improved the efficiency, speed, and readability of our Flow or code, and we are officially done. Wrong again.

As our knowledge grows and Salesforce releases updates to the platform, and technology improves in general, we must revisit our Flows and code and determine if we can improve them further. Flows and apps are living documents that need to be revisited and see if the solution can be improved over time. A good Salesforce Administrator or Developer looks back over previously created solutions to see if they can improve upon them. Why do they do this? To be a better programmer, improve their organization. Taking the time to look over all aspects of your org keeps your org running more efficiently.

Thinking like a programmer means that you must think, constantly striving to understand the problem, solve it, make it better, and constantly revisit your previous solutions to see if they can be improved as technology and your knowledge improves. It sounds like a lot of work, and it is, but if you do this, you will continually be improving your org, and you will continue to be the rockstar that you are.