Planning

Writing code is (or at least, should be), 90% planning. Investing an extra 10 minutes in carefully planning out a piece of code can save hours of debugging a snarled mess later on. Many novice programmers mistakenly jump right into writing code without a plan, only to end up pouring hours into what should be a relatively short task.

Planning first is not only the best approach for novices, but also for skilled programmers. However, if you see a highly experienced programmer in action, you may not see her planning when working on a relatively easy problem. Not seeing her planning does not mean that she is not doing it, but just that she is capable of doing all of the planning in her head. As you advance in programming skill, this will eventually happen for you as well—there will be certain problems that you can just solve in your head and write down the solution. Of course, having practiced the skills required to solve harder problems will be key, as your skills will be put to better use if you work on problems at the difficult end of your capabilities.

As we discussed in the previous course, planning for programming primarily consists of developing the algorithm to solve the relevant problem. Once the algorithm is devised (and tested), translating it to code becomes relatively straightforward. Once you have implemented your program in code, you will need to test—and likely debug—that implementation. Having a clear plan of what the program *should* do at each step makes the debugging process significantly easier.

Even though the previous course explained Steps 1–4, and worked some examples, we are going to revisit them now. One reason for revisiting these steps is that they are crucial to programming, and it is likely that they have somewhat faded from your mind, as we last saw them last course. However, we are also going to revisit them now as you have been introduced to the material in the modules on Reading Code and Types which you did not know in the first week. Accordingly, we can now talk about types, and representing everything as numbers. After we revisit Steps 1–4, we will continue on to Step 5, translating our algorithms to code. At the end of this module, the video Writing isPrime will work an entire problem from Step 1 to Step 5.