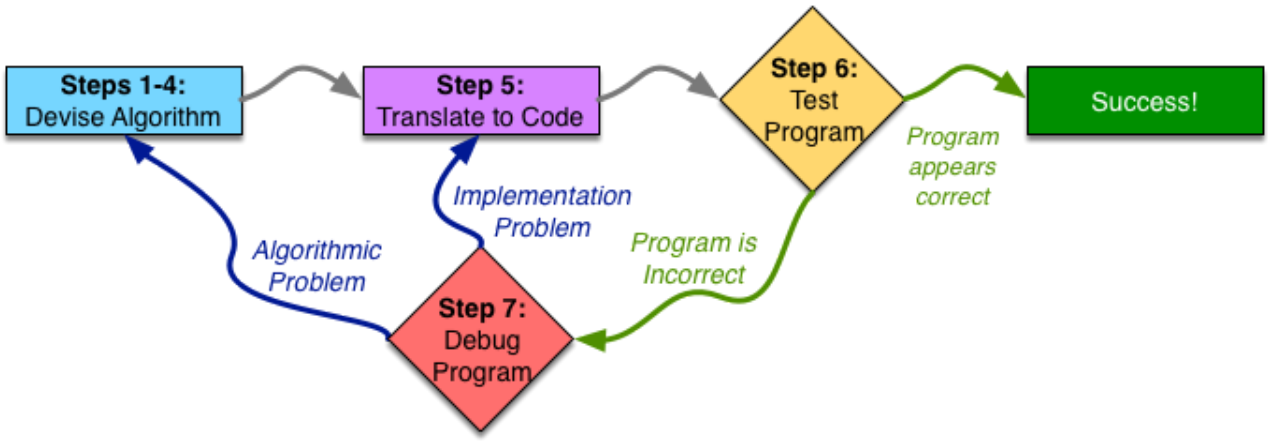
Overview of the Seven Steps

coursera.org/learn/programming-fundamentals/supplement/p8XS3/overview-of-the-seven-steps

The Seven Steps



This figure shows a high-level overview of the programming process. A programmer starts by devising the algorithm for the task she is trying to solve. We will split this planning phase into four steps in the process of writing a program, which we will discuss in more detail shortly. At the end of these four steps, the programmer should have a complete plan for the task at hand—and be convinced that the plan is a good one.

After devising a proper algorithm, she is ready for Step 5 of the programming process: translating her plan into code in the programming language she is using for her current project. Initially, translation to code will go slowly, as you will be unfamiliar with the syntax, likely needing to look up the specific details often. However, even if slow, it should be fairly straightforward. You already devised the plan, so you should have done all the actual problem-solving tasks already. Your algorithm may have some complex steps, but that is fine. As we will see later, whenever your algorithm calls for a step that is too complicated to be simply translated into a few lines of code, you should turn that step into its own separate programming task and repeat the programming process on it. In the next course, we will discuss translation to code in much more detail, as well as how to turn the code into something that the computer can run.

Once the algorithm is implemented in code, the programmer must test her code, which is the 6th Step of the programming process. By testing the program, the programmer tries to uncover errors in her algorithm or implementation. If the programmer finds errors in her program, she debugs the program (Step 7): finding out the cause of the error, and fixing it. The programmer may need to return to the algorithm design steps (if the error lies in the algorithm) or to translation to code (if the error lies in the implementation) to correct the error. The programmer then repeats all of the later steps.

At some point, the programmer completes enough test cases with no errors to become convinced that her program is correct. Note that we said the programmer becomes *convinced* that her program is correct. No amount of testing can guarantee that the

program is correct. Instead, more testing increases the programmer's confidence that the code is correct. When the programmer is convinced her code is correct, she has successfully completed the task at hand. We will discuss testing and debugging in much more detail in the next course.

Completed