What are algorithms?

Selecting transcript lines in this section will navigate to timestamp in the video

- [Instructor] Let's get started by understanding what an algorithm is by working through a hypothetical example. So suppose you had a collection of shapes and you needed to group all of the same shapes together in individual sets. Now there's a variety of ways you could do this, but probably the simplest way would be to write some kind of loop function to iterate over each of the shapes, determine what kind of shape each one is, and then add that shape to the correct group. At the end of this process, you would have separate groups of each shape type. So this complete set of steps taken together that solve a specific problem, forms an algorithm. Algorithms have several characteristics that could be used to describe them. For example, any given algorithm has an associated complexity and usually there's more than one. There's space complexity, which describes how much memory and storage space the algorithm needs to do its work. There's also time complexity, which describes how efficient the algorithm is relative to the size of the input it is given to work on. And we'll learn more about time complexity later in the chapter. Algorithms also typically have a defined set of inputs and output. In other words, any given algorithm has a set of input values it can work on in order to produce a result. Sorting algorithms, for example, take a set of data values and attempt to put them into a specific order. It's also possible to talk about algorithm classification using a variety of criteria, just a few of which I've listed here. Some algorithms work on their data sets in sequential fashion, which means that they are serial in nature. Whereas a parallel algorithm can break up a data set into smaller pieces and then work on each simultaneously. An algorithm can be exact, in which case it produces a known predictable value, or it can be approximate, in which case, it tries to find an answer that might or might not be exact. For example, a facial recognition algorithm might not give the same answer every single time. Algorithms can be deterministic, in which case it executes each step with an exact decision, or it could be non-deterministic, in which it attempts to produce a solution using successive guesses, which become more accurate over time. Of course, there's much more to the field of algorithms than I can capture in just a few slides. And we won't have time to get into all of these different intricacies in this course since it's just an introduction to the subject. But this should give you a basic sense of what algorithms are and how they work. Next, we'll learn a little bit about some of the more common algorithms that you'll encounter when writing your own programs.