Common algorithms in programming

Now that we've seen what an algorithm is, we can investigate some of the more common ones that appear in a variety of programming scenarios. One of the most common algorithm types you'll run into are searching algorithms and these pop up in cases where you need to find a specific piece of data inside a larger data structure. For example, searching for a substring inside of a larger string, or maybe finding a file within a set of nested folders on a file system. Sorting algorithms are another very common type used when working with ordered sets of data. And, as you might have guessed, they take a set of data, and place it into a particular order. Computational algorithms are used to take one set of data and derive another set of data from it. And a simple example might be calculating whether a given number is a prime number or maybe computing a temperature in one scale when you already have it in another scale. And finally, there are collection algorithms, which involve manipulating or navigating among sets of data that are stored within a particular structure. And again, it's not hard to imagine some examples here, counting specific items, filtering out unwanted data, and so on. So, let's take a moment and try implementing a simple algorithm. And forgive me for a moment while I momentarily transport us to a math class where we are going to build a program that uses Euclid's Algorithm to calculate the greatest common denominator of two integer numbers. So the greatest common denominator of two numbers is the largest integer that divides both numbers without leaving a remainder. And to take a specific example, the greatest common denominator of eight and 20 is four because four divides into eight two times with no remainder and four divides 20 by five times, and again with no remainder. So, the way the algorithm works is this, we have two numbers, A and B, such that A is greater than B. So we divide A by B and then look at the remainder. Now, if the remainder is zero, and remainder is R in this case, then we stop because now we found the greatest common denominator. Otherwise, we set A to B, and then B to the remainder, and repeat at step one until the remainder is zero. So, let's go look at this using our example of 20 and eight, and we can see that A starts as 20, and B starts as eight, and the remainder of dividing 20 by eight is four, because eight times two is 16, there's four left over. So then A becomes B, which is eight, and then B becomes R, which is four, and then we do the division again, and in this case, the remainder is now zero because eight is divided by four exactly. So that gives us the greatest common denominator of four, so let's go ahead and build this in code. So here I am in Visual Studio Code, and in my introduction folder, I'm going to open up gcd\_start.py, and I'm going to use Python for this, but if you're feeling intrepid and you want to use another language like JavaScript, that's fine too. So here in my gcd\_start file, I'm going to write the code for the gcd function, and what I'm going to do is use a while loop for this and detect when B is equal to zero as the stopping point. So I'll say while B is not equal to zero. So what I'm going to do is inside the loop, I will store aside the value of A in a temporary variable named T. So I'll have T is equal to A. And then I'm going to go ahead and set A equal to B ahead of time. And now I'm going to recalculate B to the value of A, which is now in this temporary variable right here, and I'm going to divide that by B using the module load divisor operator, which gives me the remainder value. So I'm going to say B is equal to T percent B. So, if B is now zero, then this loop will stop, and I can just go ahead and return A because we found the greatest common denominator. And remember, I set A equal to B ahead of time. So if we found the greatest common denominator then we stop. And you can see that I've got two lines here to test this function out, so here's our 20 and eight example that we've already seen, and we know that that should be four. And then here's another example, finding the GCD of 60 and 96, and that should be 12. Now, I have the Python extension here installed in Visual Studio Code, so if you're using Visual Studio code and you have the Python extension installed, you can just go ahead and run this right within the editor by executing the run python file in terminal command, and you can see that I've got 12 and four is the answer, and if you don't have Visual Studio Code installed, or you're not using the Python extension, don't worry, you can just run this from the terminal. So I'll go ahead and bring the terminal up. And here in my terminal, let me just make sure the Python three is installed. And you can see that it is. So I'm going to go into my exercise files folder, which is on the desktop, and it's in exercise files. And there's my introduction chapter, so I'll cd into my introduction chapter, and I'm going to write python3 gcd\_start. And there you can see the answers are 12 and four.