Numerical data types

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

- There are many different ways to represent numbers in code with numerical data types. Numerical data types are simply different ways we classify our numbers. In several programming languages, we group our numbers into two types, whole numbers and decimal numbers. However, the exact data types and groupings often depend on the language. Most languages represent a whole number with something called an int or an int data type. Whole numbers and integers can also be represented with something called a short or a long, depending on the language. As for the decimals, the data types that often represent them are floats and double. Now I just threw a ton of terminology at you. What's the difference between all of these? The difference between each of these data types is the precision with which they store their values. In other words, the difference is the range of numerical values that the data type can store. The actual sizes and ranges of these data types depend on the programming language, but here's what they look like in Java. This idea of bits and bytes might seem scary. But don't worry, we'll get to it. For now, just know that a single one or zero is called a bit, and a group of 16 one and zeros is called 16 bits, just like a group of 32 ones, zeros would be called 32 bits. It's just a way to show how much space each data type takes up and what values each data type can hold. Notice that the data types with the higher amount of bits can store a higher range of values. Now let's break down these data types. For decimal values in Java, double is generally considered the default choice. But if you need to save memory, you should consider using a float. However, if your numbers are too big or too small, then you might be forced to use a double because a float supports a smaller range of values. If you are using whole numbers and want to save memory, the short data type is a great option. But again, you might be forced to use an int depending on how large your value is. If you need a range of values wider than those provided by an int, then the long data type is a great choice. Now it's important to remember that not all languages have the same data types. I've introduced the Java data types here because you'll see int and double in most languages and float, short, and long in a few others. How many bits they take up may differ depending on the language, but the general concept is the same. You can also have signed and unsigned numerical data types. Whether something is signed or unsigned affects the specific range of values that the given data type contains. Everything we've looked at so far is signed, meaning it contains both positive and negative values. We can also have unsigned data types, where they only contain positive values, and it doubles the positive value range. This means if we had an unsigned short, it could store the values from zero to 65,000 instead of negative 32,000 to positive 32,000. Switching to and from signed or unsigned essentially shifts the range of values that a given numerical data type can cover. Looking at other languages, JavaScript only has numbers, no ints or doubles of floats, just numbers. Others, like Java and Visual Basic, give you more flexibility in how you categorize your data and how much memory it takes up. In Python 2, we have int, long, float, and complex numerical data types. But Python 3 makes all of your integers longs, but still calls them ints, which ultimately gives them higher precision and makes them take up more space. This makes sense though, since Python is often used for many scientific and mathematical computations. The way we store and access data will look different in different languages, but the theory and overall picture is the same. Here are a few examples of what storing various numbers looks like Java, JavaScript, and Swift. Don't worry too much about the syntax here. Just know that we have a way of capturing values with code in various languages.