Welcome to this introductory video on programming in the Python language. Learning to write programs provides you with practice and skills, and logical thinking and problem-solving. Both of which are hugely valuable. To program you first need to learn a programming language so that you can communicate with a computer. Think of a computer as an electronic machine that doesn't understand what you're saying but it does respond. At the heart of every computer is the central processing unit or CPU. This is the part that executes programs. The CPU however only understands instructions written in machine language, or binary. If we want to give a computer instructions that it can follow we could do it in binary but that's much too hard, so we need a programming language that makes sense to people but can be translated to binary for the CPU. There are many programming languages that we could choose from but in this course we use Python. When learning to program, Python can minimize the obstacles in front of your learning progress. It allows you to focus on how to think instead of how to construct instructions. Importantly you can focus on how to solve the problem rather than focusing on the language itself. It's also useful in the real world, both in industry and in research.

The 2017 IEEE ranking of programming language usage ranked Python as number one. Also, it's free. Let's take a look at how Python works, starting with comments. The comment operator is the hash character. A comment will span the entire line and they are ignored by Python, they're for the readers benefit. There should be a space between the hash and the comment text. They should start with an uppercase letter and end with a full stop. You should use comments to describe each variable, unpack logic that might not be self-evident and highlight major parts of a program. Variables - a variable is a name that represents a value stored in the computer's memory. As seen in the video titled introduction to idle you can use Python like a calculator. However, it's usually more convenient to give names to the values that you are using. Any calculation that we conduct, or output that we produce, is stored in memory. To make it easy to refer to these memory locations we can use a name.

These named locations we call variables. To create a variable and give it a value

we can use an assignment statement. In this example we create a variable named 'num'. We use the assignment operator, the equal sign, to assign it a value of 3. From this point onward if we refer to num in our program we will get that stored value, 3. We say that the num variable references the value 3. Looking at the statement we read it as num is assigned a value of 3. Now we can use this variable in other calculations. Speaking of naming variables we could have named our num variable almost anything that we wanted, it needn't have been num. I say almost, because there are some naming rules that we must follow in Python and indeed many other languages. All variable names must start with a letter or an underscore character. It cannot contain spaces. They may contain letters, numbers, and the underscore character. They are also case sensitive so the variables num with a lowercase n and num with an uppercase n would be different. You cannot use Pythons reserved keywords as variable names. As well as these rules there are some conventions that good programmers should follow. This consistency makes sharing and understanding code much easier. For variable names use lowercase letters. The only exception is for constants which are special types of variables whose values will not change throughout the execution of the program. These should be named all in uppercase letters. Names should be as short as possible, while still meaningful to a reader and self-descriptive. You should always choose names for your variables that give an indication of their purpose otherwise you'll find that when you read your code later you may not understand what all the variables are for. When you need to use a variable name that contains more than one word choose one of two approaches. An underscore to separate the words, or camelcase where every word after the first starts with an uppercase letter. Either choice is fine but be consistent. Decide which you like better and stick with it. Okay, so the real power of using variables is that their value can change as the program progresses along its execution. Our variable num that we assigned a value of instead. What we've done in this example is to shift the reference from 3 to 7 the 3 would still exist somewhere in memory but we cannot access it as it is

no longer referenced. Consider how powerful this ability to change a variable is. Imagine that we use the variable num 150 times in our program. When we shift the reference from 3 to 7 that new value of 7 is automatically accessed by all those uses of the variable and we only had to change it once. Let's go through some examples of how to use variables in simple arithmetic. To do this we need to understand some basic types of numeric variables available to us. We have the int type which is short for integer, or a whole number with no decimal point like
boolean which can be one of two values, true or false. Lastly, there's float. This is a floating-point number. A number that contains a decimal, like 3.14. There are other types as well but let's work with these for now.
The basic arithmetic operations in Python use a plus symbol for addition, a dash symbol for subtraction, the asterisk for multiplication, and forward slash for division. A couple of other operations you may use are the double forward slash, which is the floor operator. This gives us the largest integer value less than the result of a division. For example 15 divided by 4 would give 3.75, while the floor of 15 divided by 4, which you would write as 15 double forward slash 4 gives
which is the percentage symbol gives us
the remainder of a division. Using that same example of 15 divided by 4, 15 modulo 4 gives 3, as 4 goes evenly into 12 with 3 remainder. Try a couple of other examples like 11 modulo 3 and 10 modulo 5. See if you can guess what the result will be before you try it in the idle shell. Lastly a double asterisk is used to
raise to a power. 5 double asterisk 2 is
would give 27. Let's put this all into
practice. Let's make another variable named num and assign it the result of an addition operation, 1 plus 2. We'd start by typing the name of our variable, num in this case, but remember that we could call it anything meaningful. Then use the assignment operator, equals, and then our operation, 1 plus 2. Then let's use the print function that we learnt in introduction to idle to display the
result, and we get 3. So num is assigned the value of 1 plus 2. Let's make two new variables now, num1, with a value of 3, and num2, with a value of 5. Then let's make a

new variable that we'll name total, that is assigned the value of num1 plus num2. See how we can use both of these first variables to assign a new value to a new variable. Consider this example, we create a variable num and assign it a value of 6. Then we assign num again. This time the value num plus 1. If we were to print num at this point what would be the displayed value? Let's make this just a little more complicated. If we make a variable, let's call it result this time, that references the result of the operation 6 plus 2 times 5, what value will result store? The correct answer is the order of operations. Python follows standard algebraic rules of precedence and associativity, so some operations are calculated before others. In our example the multiplication has a higher precedence than addition, so 2 times 5 is calculated first then added to 6 to get parentheses. Anything inside parentheses is calculated first. In cases where there are nested sets of parentheses they are calculated innermost to outermost. The next level down is exponents. Things like multiplication, division, floor and remainder. Which are all the same precedence so are calculated left-to-right. Addition and subtraction operations are at the lowest level and are also performed left-to-right. And that's the end of this introduction to programming in Python. You've now learned the basics of commenting and variables, so you should practice making some simple Python programs that create some variables, assign and change their values, and learn how each of the operators we learnt works. See you in the next video.