Welcome to this video which will explore some of the features of the Python standard library.
Python has numerous functions that are available to us
without having to rely on anything external, like a library we'd have to import.
Before we delve into those however, let's
take a look at what I mean when I sat that Python has functions available to us.
A function consists of three components; a name,
input arguments, of which there could be 0 or many, and some output.
You can see that the Python statement here look quite like
how we created a variable in the video 'introduction to programming'.
That's because it is. We have our name, which you'll remember could be anything we like,
we have the assignment operator, the equals character,
and the value that we're assigning to our variable 'b' is the
output, created by the abs function we're using.
We say that the abs function returns a value.
In fact, abs is short for absolute.
In this example we've given the abs function negative 3 as input.
If we print 'b' and run the program, 3 will be printed to the screen.
The abs function calculated the absolute value of negative 3,
returned 3, and this value was assigned to the variable b.
We saw that the abs function required only one input.
Sometimes functions may require multiple inputs.
Take for example, the power function.
This one calculates exponents by taking two numbers as input.
The first number, X in this example, is raised to the power of the second number, Y.
So if 2 and 3 are input in that order and separated by a comma,
We could store that in a variable that we might decide to name result.
There are a huge number of built-in functions, so we cannot possibly go through them all.
But let's have a look at some really useful ones.
Before we do, it's worth taking a look at how the python documentation works.
Let's use the 'round' function as an example.
The documentation will first give the name of the function

followed by the parameters in parentheses.

You'll often see some parameters in square brackets

These are written in square brackets to indicate that they are optional.

To find out what they are for, you just need to read the description below.

Here it says that ndigits (the optional argument) is to specify the precision to which we want to round.

It also says that if it's omitted, which means left out entirely when we use the function,

The round function will return to the nearest whole number.

Don't be afraid to read documentation if you don't know what something does.

It's your best friend and you can't possibly expect to remember everything.

The round function takes as input a floating point number

and rounds to however many digits you specify.

If you only give one argument,

the number to be rounded, it will round to the nearest integer, or whole number.

If you give it a second input, for example 2, it will round to two decimal places.

The int function will convert, or pass, a string input into an integer.

If we give it the string '5' in single quotation marks, it will return the integer 5.

This is useful if we have some numerical string,

that we want to apply a mathematical operation to.

We first need it to be a numeral, so we convert with int.

The float function is basically the same as int, but it takes a string and converts it to a floating point number.

The str function is basically the int and float functions, but in reverse.

It takes an input and converts it to a string representation.

So, if the integer 5 was input str would return the string '5'.

Notice the apostrophes that indicate that it's a string.

Perhaps the most common function that you'll see

and that we've, in fact, used in other videos is the print function.

As you've seen, the print function most commonly outputs its input to the shell,

or more technically, the systems standard output.

Print can take many different types and numbers of inputs.

It could take a single string, or a string in an integer.

The print function outputs the string and the integer as one joint string, separated by a space.

That's pretty neat. But why does it happen?

Print can take three optional inputs; sep, end and file.

Here it's the sep optional input that's separating our string and integer with a space.

Even though we didn't explicitly give print a sep input, by default, the separator is a space.

We could easily change it do a dash, for example,

by adding sep equals the dash, as an input.

If we execute the line again, the two inputs will now be separated by the dash.

You should experiment with sep by changing

the separating character and adding more than two inputs.

See what happens.

Also, the default end of line character is the newline character.

Try changing this in the same way as we changed sep and see what happens.

Experimenting with functions is the best way to learn how they work.

The last in-built function that we'll look at is input.

This function allows you to take input for a user of your program.

Take this line of code as an example.

We've got a variable that we've named guess.

It has been assigned the value of the output of the input function.

As input to the input function

we've entered a prompt, asking the user for their guess.

When this is run the input function will wait

for an input from the user, waiting for the newline character,

which we'll get when the user presses return.

The guess is converted to a string and stored in our guess variable.

One last example shows you how you can combine, or nest, functions.

Let's say we want to take in a number from a user of our program and add it to 5.

If you remember, the input function converts all input into a string.

We can't use a string for mathematical operations,

we need a numeral. So, let's use the int function to convert the string to an int.

We would wrap our call to the input function with a call to the int function,

everything inside the int function will be converted to an int.

This is very powerful and useful to remember.

Sometimes there are functions that we really want to use

but they are not part of the built-in functions that Python provides.

The function might still be part of the standard library

or it might be part of a library written by somebody else.

Usually in these cases we need to first import the library before we can use the function.

Let's say that we want to generate a random number between 1 and 10.

The function that we want to use is called randint and it lives in the 'random' library.

So at the top of our file we use the 'import' keyword and then name the library that we want to import.

Then we can use the function by calling first the name of the library, followed by the dot notation,

followed by the name of the function, randint in this case.

This essentially just reads as "call the randint function from within the 'random' library.

Now our variable will hold a number between 1 and 10.

And that's it for our exploration of the Python standard library.

Please practice those that have been shown here

and look at the Python 3.6 documentation for other in-built functions that you can try.

See you in the next video.