This video will discuss compound statements and for loops in Python.
Loops allow certain lines of a script to be repeated until a specified condition is met.

- **A loop** is a set of statements that repeats until some condition is met.
- **Loops repeat until…**
  - all items in a list or dictionary have been run through the loop *(for loop)*
  - a certain condition is met *(while loop)*
- **Loops are compound statements…**

**Loops** allow certain lines of a script to be repeated until a specified condition is met.

A **for loop** will repeat until it has processed all items in a Python data collection – these data collections include lists and dictionaries as well as arrays and sets (which will be discussed later in the course).

A **while loop** will repeat until a specific condition is met.

Loops are **compound statements**.
Compound statements contain multiple lines of code that are linked to a single operation.

Compound statements begin with a header line – the header line ends with a colon.

The lines below the header line are indented to indicate their association with the header line.

All lines associated with the header line must be indented by the same amount.
A for loop requires a list, dictionary, or other Python data collection as its input.

The for loop will run one time for each item in the data collection.

In each iteration, an item from the data collection is assigned to a variable and the indented lines below the for loop header are run.
Here is an example of a for loop.

The loop starts with a header line

Which ends in a colon.

The header line begins with **for**

Followed by a variable.

The variable is followed by **in**

Which is followed by the data collection (typically a list or dictionary).

Below the header will be one or more indented lines that are inside the for loop.
Let's look at an example of a for loop in operation.

In the 1\textsuperscript{st} iteration of the loop, the integer 1 is assigned to the variable \textit{item}. The print statement prints the value 1.

In the 2\textsuperscript{nd} iteration of the loop, the integer 2 is assigned to the variable \textit{item}. The print statement prints the value 2.

The process repeats for the 3\textsuperscript{rd} and 4\textsuperscript{th} iterations until all items in the list have been processed.
Sequential lists can often be useful when using for loops. The `range` function can be used to easily create sequential lists.

For the `range` function, you can specify a single integer value to create a list in which the sequence starts with zero and stopping at the specified value (note that the ending value is not included in the result).

To start the list with a non-zero value, specify both a starting and ending value for the list.

To create a sequence with an increment other than 1, specify a third integer for the increment amount.

Note that sequential lists can count down from the start value to the end value if you specify a negative increment.
A sequential list created by the range function can be convenient if you want to create a for loop that runs a specific number of times.

The range function is equivalent to a list and can be included directly in the for loop header line.

The range function will create the list before the for loop runs.

The loop in this example will simply print the values in the sequential list which range from 0 to 5.