There's one other thing we can do with loops, and that's to include an ***else*** clause. Experienced programmers ***often avoid using else*** in this way, and that's because *less experienced programmers can struggle to understand their code*. But it's something you might come across, so I want to cover it here.

We've seen ***else*** when we looked at ***testing conditions*** using ***if***, but it can also be ***used at the end of loops***.

It's possibly an ***unfortunate word to use for what it does with loops though***, because its use ***doesn't imply "otherwise" in any sense***.

What it does is cause a block of code to be executed if the loop was allowed to continue to the end, in other words, if the loop wasn't broken out of.

→ To summarize, ***if-else*** is used for making ***conditional decisions*** based on a ***single condition***, while ***for-else*** is used in ***loop constructs to specify what should happen when the loop completes without a break statement***. They serve different purposes and are used in different contexts.

With what we've covered so far, any example we use will either be contrived or could be written better some other way. I don't like showing you code that could be written better, so the first example will be a bit contrived.

Let's assume we've noticed that lottery numbers divisible by 8 are less likely to appear. Now I know that's not realistic - I did say this example was contrived.

We want to write some code that will ***check*** a ***list of numbers***, and let us ***know*** if the ***list contains any values that are divisible by 8***.

So let's start with a new file. We'll call this one contrived.py

```
01    # Create a list of numbers
02    numbers = [1, 45, 32, 12, 60]
03
04    # Iterate through the list of numbers
05    for number in numbers:
06        # Check if the number is divisible by 8
07        if number % 8 == 0:
08            # Print a message and exit the loop if the condition is met
09            print("The numbers are unacceptable")
10            break
11
12
```

We'll start with a new list:

```
numbers = [1, 45, 32, 12, 60]
```

We're going to look at ***lists*** in the next section. All we need to know here is that they're ***enclosed*** in ***square brackets*** and are a ***sequence type***. We know that we can ***iterate over sequences*** - we did that a lot with the string type. Let's try out some code.

So here's what we're doing, for each number in the list, we're ***checking*** if it's ***exactly divisible by 8***, using a ***modulo or remainder operator***. If it is, we *print a message rejecting* the list and ***break out of the loop***. Now these numbers should be rejected because we've got the value ***32*** *in the list on* ***line 2***.

```
The numbers are unacceptable
```

we run the program and the numbers are rejected, so that works.

Let's see what happens with the ***list of numbers that aren't divisible by 8***.

So I'm going to change the ***32*** to a ***31*** on ***line 2***. Run the code again:

```
01    # Create a list of numbers
02    numbers = [1, 45, 31, 12, 60]
03
04    # Iterate through the list of numbers
05    for number in numbers:
06        # Check if the number is divisible by 8
07        if number % 8 == 0:
08            # Print a message and exit the loop if the condition is met
09            print("The numbers are unacceptable")
10            break
```

You see this time; we ***don't get the message:*** `The numbers are unacceptable`. This is an *expected behavior* since we changed ***32*** back to ***31***

The ***code works but it's not very friendly***. It ***hasn't given us any output***.

So what would be a lot friendlier is if we've ***got a message*** *saying* that *these numbers were okay*. This is where Python's ***else*** can be ***useful when used in a loop.***

When our loop finds a value that's not acceptable, our code ***breaks out of the loop***, as we saw the first time, we ran the program. That means the loop will only terminate normally if all the values are acceptable.

So what we want to do is ***run some code when the loop Terminates Normally***.

We ***don't want that code to execute if we break out of the loop***, only when it goes all the way around, and that's exactly what ***else*** does. So let's see that working.

So I'm going to put an ***else*** here on ***line 11***

```
01    # Create a list of numbers
02    numbers = [1, 45, 31, 12, 60]
03
04    # Iterate through the list of numbers
05    for number in numbers:
06        # Check if the number is divisible by 8
07        if number % 8 == 0:
08            # Print a message and exit the loop if the condition is met
09            print("The numbers are unacceptable")
10            break
11    else:
12        # This block is executed if the loop completes without a 'break'
13        print("All those numbers are fine")
14
15
```

This code creates a list of numbers and iterates through the list. For each number in the list, it checks if the number is divisible by 8 using the modulo operator. If the condition is met, it prints a message and ***exits*** the loop using the ***break*** statement. If the loop completes without encountering a ***break***, the code in the ***else*** block is executed, indicating that all the numbers are fine.

So if I run the program, this time we get the message:

```
All those numbers are fine
```

We ***didn't break out of the loop***, in other words, and the ***code*** *in the* ***else block is executed***.

Let's see what happens when I change the list on ***line 2*** to include an ***unacceptable number***.

We'll change the value on Index position 3 from ***12*** to ***16*** in the ***numbers*** list

```
01    # Create a list of numbers
02    numbers = [1, 45, 31, 16, 60]
03
04    # Iterate through the list of numbers
05    for number in numbers:
06        # Check if the number is divisible by 8
07        if number % 8 == 0:
08            # Print a message and exit the loop if the condition is met
09            print("The numbers are unacceptable")
10            break
11    else:
12        # This block is executed if the loop completes without a 'break'
13        print("All those numbers are fine")
14
15
```

This time we get the output:

```
The numbers are unacceptable
```

In other words, we've ***broken out of the loop***, and because the ***loop didn't terminate normally***, the code in the ***else block*** - this is the code on ***line 11*** - ***didn't get executed.***

Now once again, take careful notice of the ***indentation***. This is particularly tricky, here as an ***else*** can follow both the ***if*** or the ***for*** statements.

It's only the ***indentation level*** that *ties* the ***else*** to the ***for***. Here, the ***else*** on ***line 11*** is at the ***same level*** as the ***for***, on ***line 5***. That means ***else*** is ***associated*** with the ***for loop*** and *not with the if statement*.

This is a contrived example to show how ***else*** works when it's ***used with a loop***. You may want to *step through the code* in the ***debugger*** to see exactly what's happening.

If you do, remember to change ***16*** back to ***12*** and debug the code again, and that'll let you see the else block executing when the loop terminates normally.

In the next video, we'll see a more realistic use of else. See you in the next video."