# `while` loop

- Used to repeat certain code until a condition is met

```
while condition:
    # code block for the while loop
```

- Continues to loop until *`condition`* is no longer true
- No automatic iterators, must be done manually if required
- Commonly seen is looping forever until event:

```
while True:
    # some event that breaks out of loop
```




# `while` loop: read file example

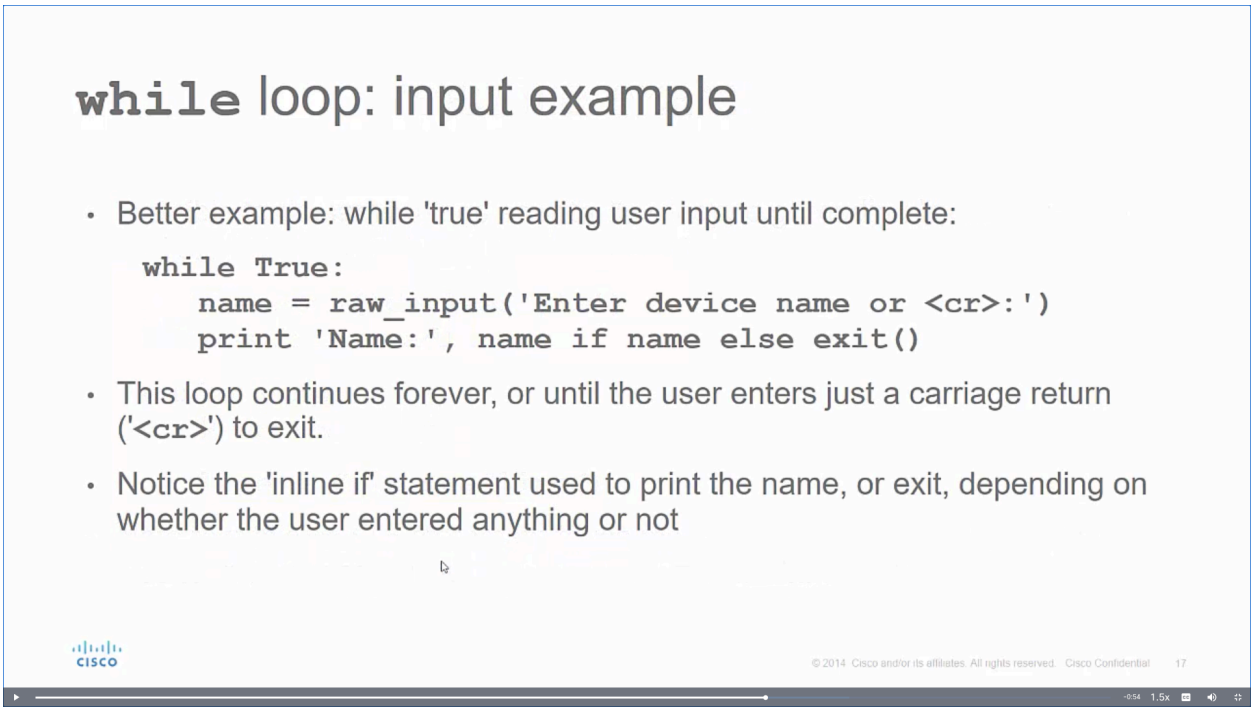
- Possible to do same things as with `for` loop, e.g. reading lines from a file and exiting when all lines have been read

```
file = open('devices','r')
line = file.readline()

while line:
    # do something

    line = file.readline()
```

- Note that this **`while`** statement takes more lines of code than the equivalent **`for`** loop functionality, but operates in a similar manner




The `while` loop is useful for repeating code over and over again, as long as a certain condition is met.

The `while` statement specifies the condition to be evaluated every time through the loop. As long as the condition is met, the looping continues.

```
while condition:
    # code block for the while loop
```

Note that `while` loops do not have automatic iterators, meaning that if you are wishing to go through a sequence of items of some sort, the indexing and selection of items for each iteration must be done manually.

One of the more common while statements that you will see is a `while true` loop.

```
while True:
    # code block for the while loop,
    # including checking for some event that breaks out of loop
```

In this type of `while` loop, the program will repeat the code block over and over, until some event occurs. Some examples of this type of `while` loop are:

- Looping until the user decides to terminate the program by entering appropriate quitting value.
- Looping, reading other types of I/O such as messages or events, until the program is externally terminated

## while Loop: Read File Example

The following example shows what reading a file line-by-line would look like, done with a `while` loop.

The code for reading devices from a file is as follows:

```
file = open('devices','r')
line = file.readline()
while line:
   # do something
   line = file.readline()
```

## while Loop: Input Example

The following example shows an example of having a 'while true' loop to repeat forever, until the user decides to terminate the program.

The code for reading input from the user is as follows:

```
while True:
   name = raw_input('Enter device name or <cr>:')
   print 'Name:',name if name else exit()
```

There are a few things to note in the above example:

- 'while true': the code is looping forever using `while true`, at least until some event causes the code to terminate the loop.
- The example is using an input function that is called `raw_input`, which prints the text that is provided, and awaits input from the user, followed by a carriage return.
- The code quits the `while` loop when the user presses carriage return without entering any data.
- The last statement is an inline `if` statement, which reads as follows: "Print the value of 'name', if the value of 'name' is not empty; if it is empty, then exit()"