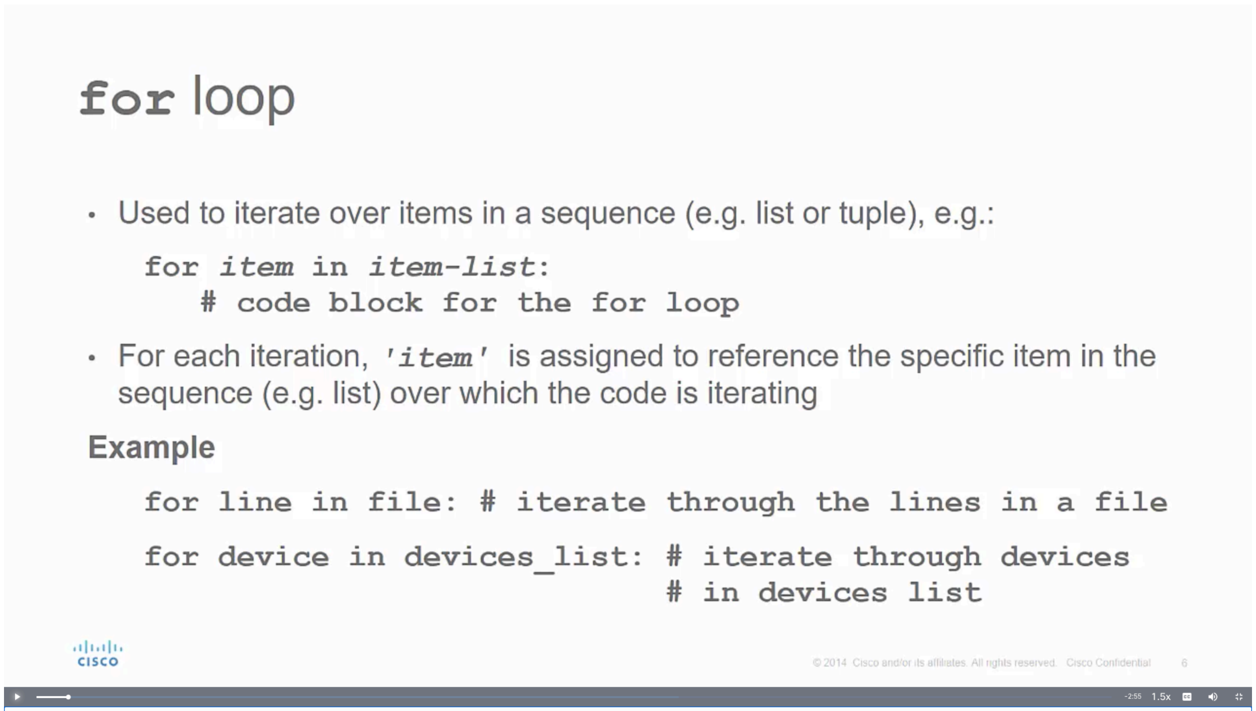


for loop

- Used to iterate over items in a sequence (e.g. list or tuple), e.g.:

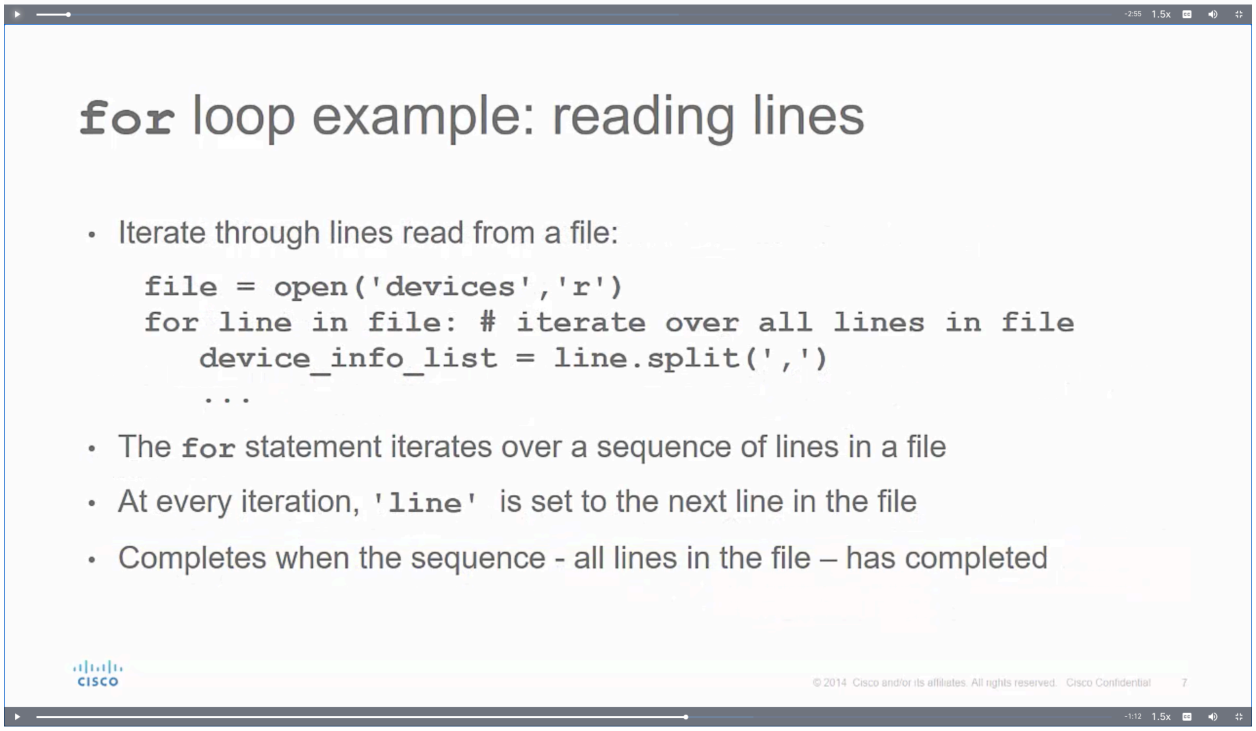
```
for item in item-list:
   # code block for the for loop
```

- For each iteration, *'item'* is assigned to reference the specific item in the sequence (e.g. list) over which the code is iterating

Example

```
for line in file: # iterate through the lines in a file

for device in devices_list: # iterate through devices
                            # in devices list
```



for loop example: reading lines

- Iterate through lines read from a file:

```
file = open('devices','r')
for line in file: # iterate over all lines in file
   device_info_list = line.split(',')
   ...
```

- The `for` statement iterates over a sequence of lines in a file
- At every iteration, `'line'` is set to the next line in the file
- Completes when the sequence - all lines in the file – has completed

`for` loops are used to iterate over items in a sequence. Data structures such as lists and tuples are sequenced which means they can be iterated or stepped through.

The general structure of a `for` loop, in simplified form, can be thought of as follows:

```
for item in item-list:
    # code block to execute for every iteration of the for loop
```

The process that takes place is as follows:

- The variable `item` is set to reference the first value in the `item_list`.
- The code block is executed.
- The program returns to the top of the `for` loop, where `item` is set to reference the next value in the sequence.
- The code block is executed, control returns to the top of the `for` loop, where `item` is set to the next value.
- The process continues until the `for` loop reaches the last `item` in `item_list`.

Some examples of simple `for` loops:

```
for line in file:
```

The example is opening a file and reading it line-by-line. The `for` statement starts with the first line in the file, executes the code block, then returns and executes the code block for the second line in the file, then the third, and so on.

The first time through, `line` references the first line in the file; the second time through, it references the second line, and so on.

The next example takes a list of devices, and one by one, iterates through the list. The first time through, `device` references the first item in the list of devices, the second time through it references the second item in the list, and so on.

```
for device in devices_list:
```

Example for Loop: Reading Lines

Python provides several built-in objects that can be iterated across, using very simple syntax. One example is reading the lines of a text file.

The process of reading lines from a file in Python because certain files are iterable. For a text file, the process consists of:

- Opening the file
- Iterating through the file line-by-line using a `for` statement

The following example shows this process:

```
file = open('devices','r')
for line in file: # iterate over all lines in file
   device_info_list = line.split(',')
```

The `for` statement above causes the code to loop through the process of reading a file one line at a time, taking the specified action in the code block below the `for` statement.

The `for` statement effectively says the following:

- Start with the file 'file'
- Take the first line and assign it to 'line'
- Perform the statements in the code block (in this case, taking the text line, splitting it into individual items based on commas (','), and creating a list from those items)
- Repeat until all lines of the file have been processed.

Other languages require some type of index for iteration, and require explicitly reading a line of the file.