`for` loop example: items from list

- Iterate through devices that have been stored in a list:

```
# (code to build list of devices)
for device in devices_list: # iterate over list
    # do something with 'device'
    ...
```

- The `for` statement iterates over a sequence of devices in the list
- At every iteration, `'device'` is set to the next device in the list
- Completes when the sequence has been iterated through

`for` loop example: items from dictionaries

- It is also possible to iterate through the items in a dictionary, but instead of one item, you have two (the key and the value):

```
for key,value in device_dict.items():
    ip_addr = key
    device_info = value
```

- The iteration takes the next item from the directionary, putting key and value into your item variables (`'key'` and `'value'` above)
- In the code example above, the key is the IP address, and the value is the actual device information, and we set these appropriately

`for` loop example: items from dictionaries

- It is also possible to iterate through the items in a dictionary, but instead of one item, you have two (the key and the value):

```
for key,value in device_dict.items():

    ip_addr = key
    device_info = value
```

- In the code example above, the key is the IP address, and the value is the actual device information, and we set these appropriately

In the same way that text files are iterable, data structures such as lists, tuples, and dictionaries are iterable as well.

For Loop: Items from a List

The process of iterating through a list is as simple as reading a line from a file. Assuming you have a Python list of devices that is called devices_list, the code is as follows:

```
for device in devices_list: # iterate over list
   # do something with 'device'
```

Python will start with the first item in the list (remember that lists are sequences, so it will be in that order), and in the example above, device is assigned to reference the first item in the devices_list. The code block below is executed, then control returns back to the top, with device now referencing the second item in the list.

For Loop: Items from a Dictionary

With Python, you can also iterate through items in a dictionary, but the syntax is a bit different and may be somewhat confusing at first.

Rather than having just one variable which gets assigned to reference the item in the iterated sequence, the for loop for dictionary items returns two references, one for the key, and one for the value:

```
for key,value in device_dict.items():
   ip_addr = key
   device_info = value
```

In the example above, both the key and the value are retrieved from the dictionary, using the items() method. With both the key and value, your application can take the appropriate action, in this case setting the IP address and device info variables.

For Loop: Keys from a Dictionary

With Python, you can also iterate through just the keys in a dictionary; the syntax is a bit more like iterating through a list.

The `for` loop returns keys from the dictionary, and your code can make use of the keys in whatever manner is appropriate:

```
for key in device_dict:
   ip_addr = key
   device_info = device_dict[ip_addr]
```

In the example above, the `for` loop returns the key from the device dictionary, and that key is then used to access the value in the dictionary. This example performs the same function as the previous example, but does it by iterating through the keys only.