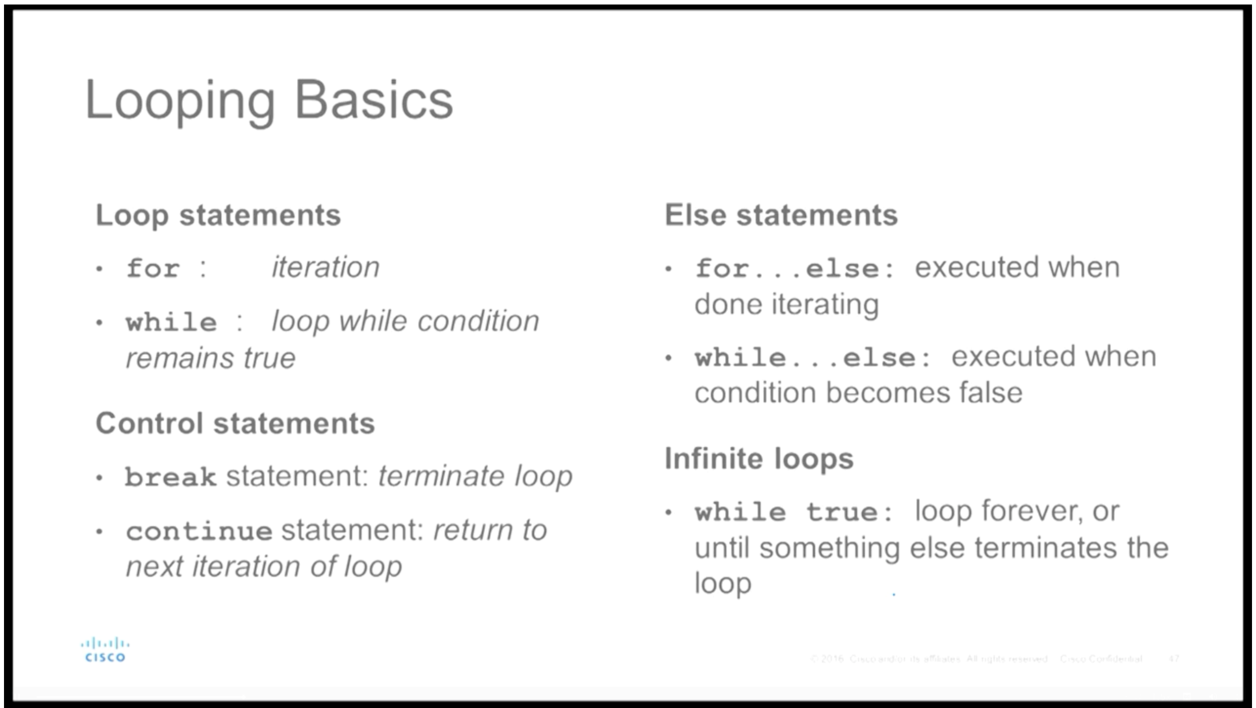


Looping Basics

Loop statements

- `for` : *iteration*
- `while` : *loop while condition remains true*

Control statements

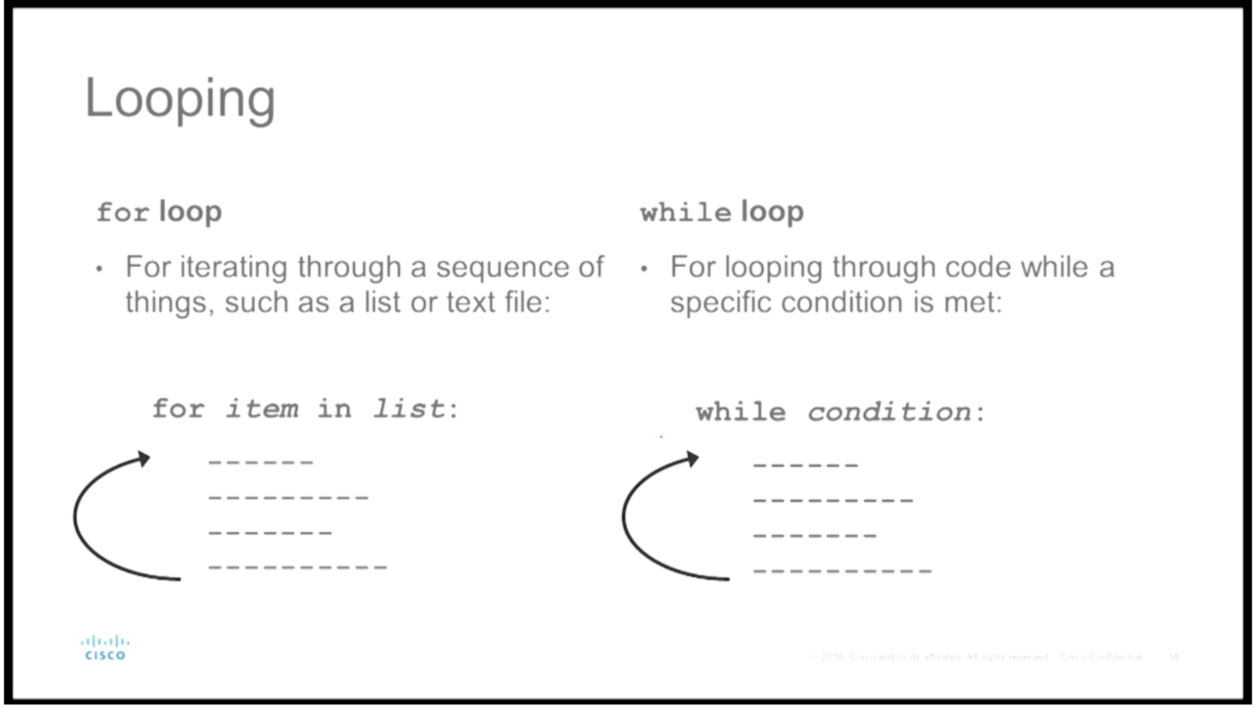
- `break` statement: *terminate loop*
- `continue` statement: *return to next iteration of loop*

Else statements

- `for...else:` executed when done iterating
- `while...else:` executed when condition becomes false

Infinite loops

- `while true:` loop forever, or until something else terminates the loop



Looping

`for` loop

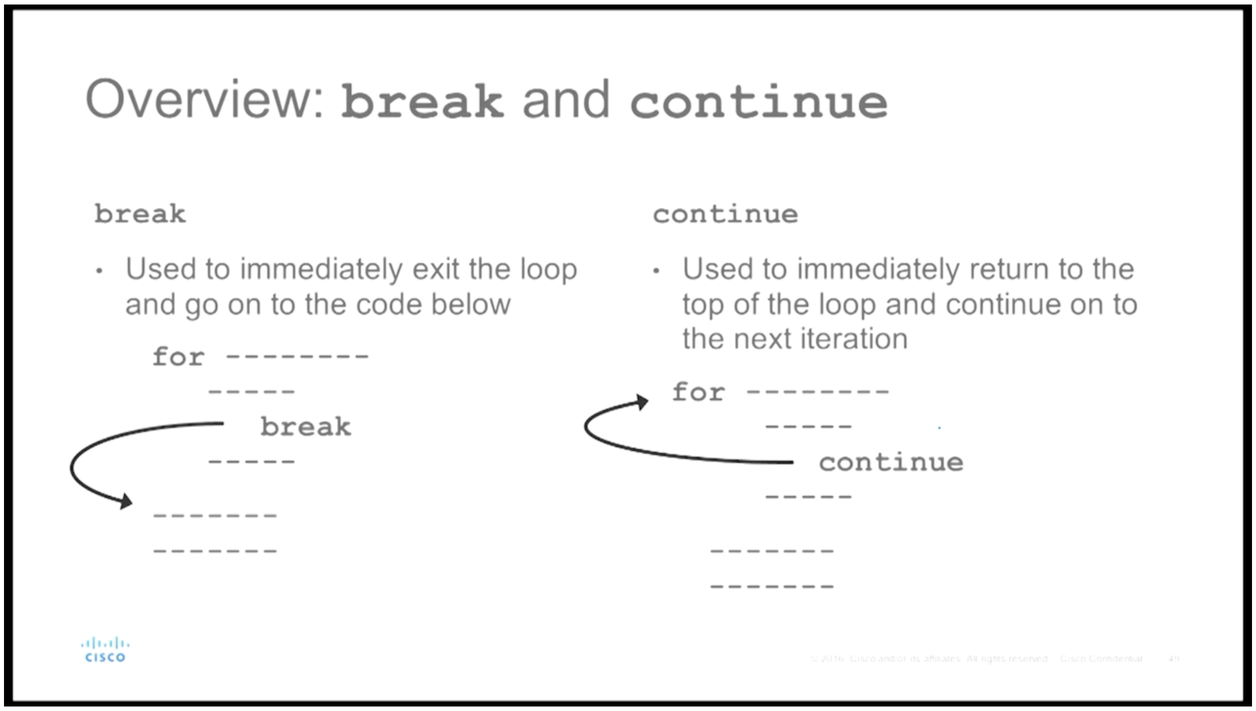
- For iterating through a sequence of things, such as a list or text file:

```
for item in list:
    ------
    ----------
    -------
    -----------
```

`while` loop

- For looping through code while a specific condition is met:

```
while condition:
    ------
    ----------
    -------
    -----------
```



Overview: **`break`** and **`continue`**

`break`

- Used to immediately exit the loop and go on to the code below

```
for --------
   -----
      break
   -----
-------
-------
```

`continue`

- Used to immediately return to the top of the loop and continue on to the next iteration

```
for --------
     -----
        continue
     -----
  -------
  -------
```

A primary goal of network programmability is to take a task that a person could do manually, such as upgrading IOS on a switch, and automate the process. For example, you can upgrade 1,000 switches using the same process with no mistakes in a small fraction of the time it would take a person to do it manually.

If you want to perform the same function over and over, the programming mechanism you will use to do that is called a loop. A loop is a way to perform the same set of programming instructions over and over, such as connecting to a switch and issuing an upgrade command, and stop at some point once you have accomplished whatever task you are performing. There are different types of loop statements such as for, while and range statements that are useful in different scenarios.

If you are creating a script to upgrade your top-of-rack switches in a data center, your script may start by opening a file that contains a list of all your switches. Opening a file and process every line in the file is a common place to use a while loop:

```
file = open('devices','r')
line = file.readline()
while line:
     # store the switch name and IP address in a dictionary
     # read the next line
     line = file.readline()
```

Once your script has read the list of top-of-rack switches into a dictionary data structure, you could use a “for” loop to process each switch in the dictionary.

```
for key,value in switch_dict.items():
      switch_name = key
      ip_addr = value
      # connect to switch at ip_addr and perform an upgrade
```

If you want to limit the upgrade to the first 10 racks in the data center, you might use a `range` statement instead of a `for` loop to limit processing to a subset of the switches. If you have a switch naming convention of sw-1, sw-2, sw-3, you could accomplish this with the following `range` statement:

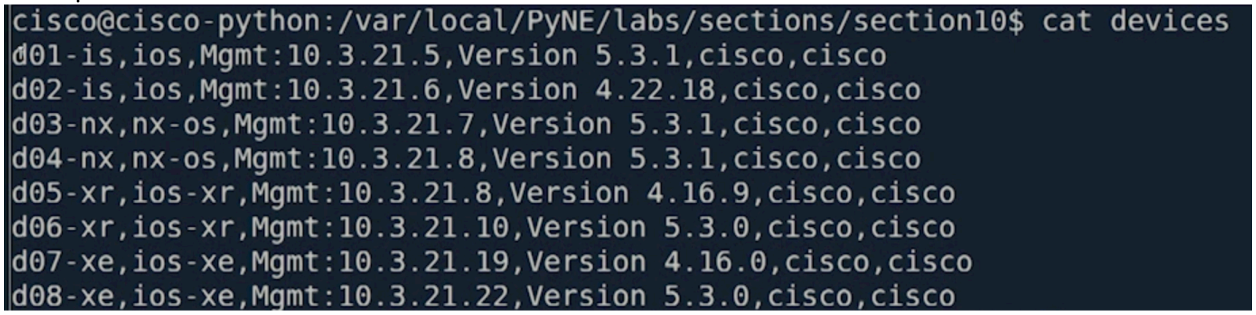
```
for index in range(1,10):
     switch_name = “sw-“ + index
     ip_addr = switch_dict[switch_name]
             # connect to switch at ip_addr and perform an upgrade
```

There are three major types of looping statements: `for`, `while`, and `range`.

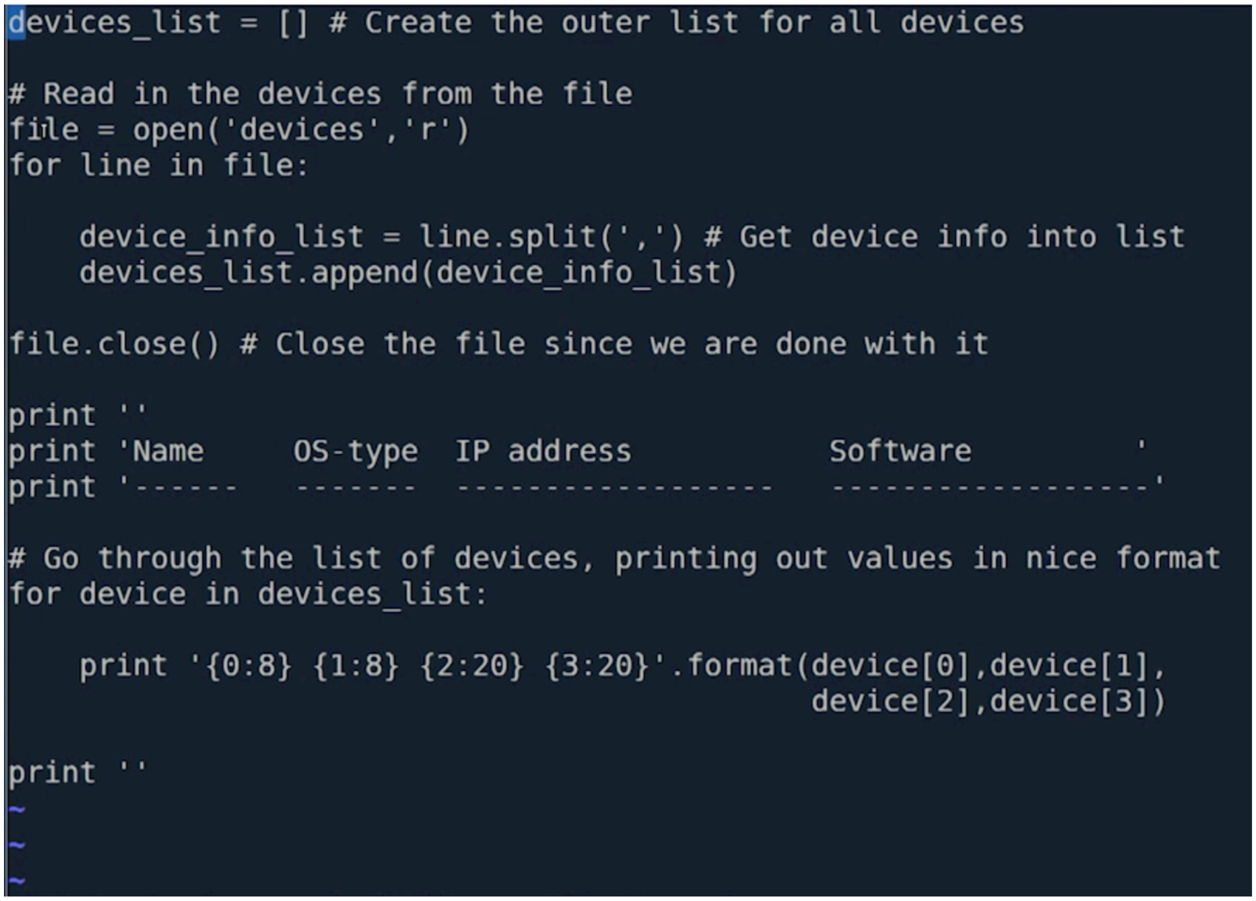
These looping mechanisms are used as follows:

- `for` **loop:** `for` loops are used to iterate through a sequence of items, such as a list, or a dictionary, or through a set of lines in a file.
- `while` **loop:** `while` loops provide similar functionality as `for` loops, continuing to repeat until a condition is met.
- `range()`**:** In a `for` loop, a range can be used to iterate over a numerical sequence.

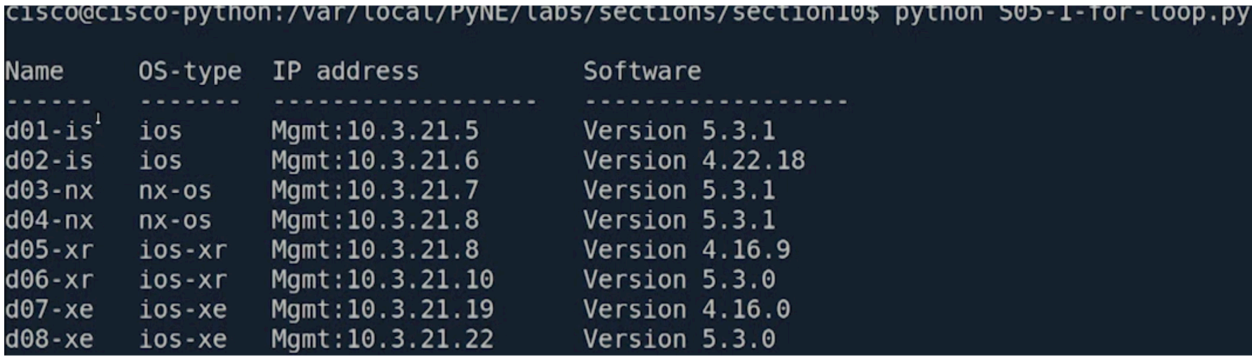
Example:



```
cisco@cisco-python:/var/local/PyNE/labs/sections/section10$ cat devices
d01-is,ios,Mgmt:10.3.21.5,Version 5.3.1,cisco,cisco
d02-is,ios,Mgmt:10.3.21.6,Version 4.22.18,cisco,cisco
d03-nx,nx-os,Mgmt:10.3.21.7,Version 5.3.1,cisco,cisco
d04-nx,nx-os,Mgmt:10.3.21.8,Version 5.3.1,cisco,cisco
d05-xr,ios-xr,Mgmt:10.3.21.8,Version 4.16.9,cisco,cisco
d06-xr,ios-xr,Mgmt:10.3.21.10,Version 5.3.0,cisco,cisco
d07-xe,ios-xe,Mgmt:10.3.21.19,Version 4.16.0,cisco,cisco
d08-xe,ios-xe,Mgmt:10.3.21.22,Version 5.3.0,cisco,cisco
```



```
devices_list = [] # Create the outer list for all devices

# Read in the devices from the file
file = open('devices','r')
for line in file:

    device_info_list = line.split(',') # Get device info into list
    devices_list.append(device_info_list)

file.close() # Close the file since we are done with it

print ''
print 'Name     OS-type  IP address          Software          '
print '------   -------  ------------------  ------------------'

# Go through the list of devices, printing out values in nice format
for device in devices_list:

    print '{0:8} {1:8} {2:20} {3:20}'.format(device[0],device[1],
                                             device[2],device[3])

print ''
~
~
~
```



```
cisco@cisco-python:/var/local/PyNE/labs/sections/section10$ python S05-1-for-loop.py

Name      OS-type  IP address           Software
------    -------  ------------------   ------------------
d01-is    ios      Mgmt:10.3.21.5       Version 5.3.1
d02-is    ios      Mgmt:10.3.21.6       Version 4.22.18
d03-nx    nx-os    Mgmt:10.3.21.7       Version 5.3.1
d04-nx    nx-os    Mgmt:10.3.21.8       Version 5.3.1
d05-xr    ios-xr   Mgmt:10.3.21.8       Version 4.16.9
d06-xr    ios-xr   Mgmt:10.3.21.10      Version 5.3.0
d07-xe    ios-xe   Mgmt:10.3.21.19      Version 4.16.0
d08-xe    ios-xe   Mgmt:10.3.21.22      Version 5.3.0
```