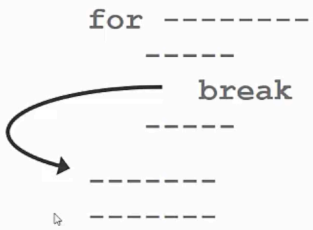
Overview: `break` and `continue`

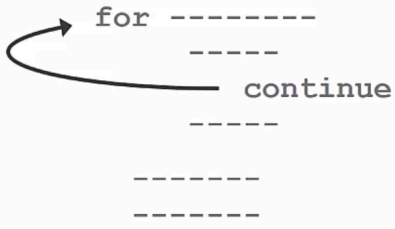
`break`

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



`continue`

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



`break` statement: `while` loop

Loop forever until user decides to exit:

```
while True:
    value = input("Enter value, 'exit' to exit")
    if value == 'exit':
        break
    # code to handle user input


print "User entered 'exit'"
```


`break` statement: `for` loop

Searching for *one* instance of a specific device:

```
for device in devices_list:
    if device.ip == search_ip_addr:
        print 'Found device:', device
        break
    # do other stuff

# rest of the program
```

`continue` statement: `for` loop

Searching for *multiple* instances of a specific device:

```
for device in devices_list:
    if device.ip != search_ip_addr:
        continue
    # do stuff with matching device
```

This could be handy in finding duplicate information, such as but not limited to

You may wish to either exit your loop, or continue on to the next iteration without continuing to process the code block. These actions are possible with the `break` and `continue` statements

The `break` statement tells Python that you wish to exit the loop immediately. Control then proceeds at the first line following the code block for your `for` or `while` statement.

The `continue` statement tells Python that you wish to proceed to the next iteration of the loop immediately. Control then proceeds at the top of the code block for the loop. For a `for` loop, iteration takes place as it normally would, which means that the iteration item (for example, line in a file, or item in a list) is set to the next value.

Break Example: while Loop

The following code shows an example of breaking out of the middle of a `while` loop.

```
while True:
   value = input("Enter value, 'exit' to exit")
   if value == 'exit':
      break
   # code to handle user input

print "User entered 'exit'"
```

In the example, the loop will continue until the user types in 'exit'. When this occurs, `value` with match the exit string, and `break` is called, causing control to pass to the line after the end of the code block.

Break Example: for Loop

Breaking out of a `for` loop works the same as `while` loops.

The following code shows an example of breaking out in the middle of a `for` loop.

```
for device in devices_list:
   if device.ip == search_ip_addr:
      print 'Found device:', device
      break
   # do other stuff
# rest of the program
```

In the example, the devices list is examined, one device at a time, looking for a device with an IP address that matches the given search_ip_address. Once a match is found, it is printed, and the loop breaks, continuing at the line of code immediately following the `for` loop code block.

```
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.strip().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 '-------  ---------  -------------------  --------------------'

ip_addresses = set()

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

    print '{0:8} {1:10} {2:20} {3:20}'.format(device[0],device[1],device[2],device[3]),

    # Print 'duplicate' if IP address exists for another device
    if device[2] in ip_addresses:
        print '*Duplicate IP!*'
        continue

    ip_addresses.add(device[2])
    print ''

print ''
```

Notice set()

Also notice the print formatting
{0:8} = first item will occupy 8 characters
{1:10} = second item will occupy 10 characters

If device[2] in ip_addresses → store the value of device[2] into the set() of ip_addresses, if value seen print duplicate else continue and add that value to the set

** This logic will print the first duplicate; it wont highlight BOTH ip's just one of the duplicates.

Lastly note the "," at the end of the print statement, this allots for the potential to print "duplicate" if required. In other words, don't just automatically go to the next line, parse through the logic and continue

The empty print statement; print'', allows us to go to new line after the duplicate ip logic

The following example will look at the while logic. Here our output will print the index:

```
cisco@cisco-python:/var/local/PyNE/labs/sections/section10$ python S05-4-find-dev.py

Idx Name     OS-type  IP address           Software
--- -------- -------- -------------------- --------------------
 1: d01-is   ios      Mgmt:10.3.21.5       Version 5.3.1
 2: d02-is   ios      Mgmt:10.3.21.6       Version 4.22.18
 3: d03-nx   nx-os    Mgmt:10.3.21.7       Version 5.3.1
 4: d04-nx   nx-os    Mgmt:10.3.21.8       Version 5.3.1
 5: d05-xr   ios-xr   Mgmt:10.3.21.8       Version 4.16.9
 6: d06-xr   ios-xr   Mgmt:10.3.21.10      Version 5.3.0
 7: d07-xe   ios-xe   Mgmt:10.3.21.19      Version 4.16.0
 8: d08-xe   ios-xe   Mgmt:10.3.21.22      Version 5.3.0

Enter device IP address to find (Ctrl-C to exit):
```

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

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

index = 0

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

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

    print '{0:2}: {1:8} {2:8} {3:20} {4:20}'.format(index+1,
                                                    device_info[0],device_info[1],
                                                    device_info[2],device_info[3])


    index += 1 # increment our index

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

while True: # Loop forever, until user terminates program

    # Request user to input the IP address we will search for
    try:
        ip_address = raw_input('Enter device IP address to find (Ctrl-C to exit):')
    except KeyboardInterrupt:
        break;

    # Loop through our devices looking for a match on IP address
    for index in range(0, len(devices_list)):

        device_info = devices_list[index] # Get information for this device in the list
        if device_info[2][5:] == ip_address:  # Check to see if device IP is a match

            # If a match, print results and stop looking
            print '{0:2}: {1:8} {2:8} {3:20} {4:20}'.format(index+1,
                                                            device_info[0],device_info[1],
                                                            device_info[2],device_info[3])

            break
        else:
            continue

    else:  # We get here if we exhausted the device list, IP not found
        print '--- Given IP address not found ---'

print '\n'
print 'Device search terminated.\n'
```

Notice the try/except logic for the interrupt which signals the end of the application

Also in the for loop we use formatting to parse the input file; **if device_info[2][5:]**

This tells use to look at the 3rd element and start to read the substring of a particular string; read 5 characters in.