`else` Statements

```
if condition:
   # code for when condition is true

else:
   # code for when condition is false
```

Examples:

```
if version == current_version:
   print(dev_info.name + ': version is up-to-date')

else:
   print(dev_info.name + ': ' + dev_info.version)
   print(dev_info.name + ': version needs updating')
```


`elif` Statements

- Simplifies code, less verbose, avoids nested '`else ... if`'; like '`switch`' statements in other languages (C, Java)

```
if condition:
   # code for when if condition is true
elif condition:
   # code for when elif condition is true
elif condition:
   # code for when elif condition is true
else:
   # code for everything else
```

It is quite possible to write an `if` statement which is complete in and of itself. For example, "if this condition is met, perform this operation." However, much of the time you will want to have an `else` clause associated with the `if.`

The general form of the `else` statement is:

```
   if condition:
      # code for when condition is true
   else:
      # code for when condition is false
```

Like `for` loops, the beginning and end of the code block for the `else` statement is identified by indentation; when the indentation goes back to the original level, the `else` code block is complete. The following example shows a simple string comparison of the version, with code that prints whether the code is up to date, or whether it needs to be updated.

```
   if version == current_version:
      print(dev_info.name + ': version is up-to-date')
   else:
      print(dev_info.name + ': ' + dev_info.version)
      print(dev_info.name + ': version needs updating')
```

elif

Sometimes your logic will call for a sequence of comparisons, each requiring a different action. Some languages have `switch` or `case` statements; Python uses a condensed version of an 'else if' statement, called `elif`.

An example of the use of `elif` would be testing for the OS type of the device to which you are talking, and taking different actions based on that information:

```
if device.os_type == 'ios':
    # perform actions appropriate for ios type devices
elif device.os_type == 'xr':
    # perform actions appropriate for xr type devices
elif device.os_type == 'nx':
    # perform actions appropriate for nx type devices
elif device.os_type == 'xe':
    # perform actions appropriate for xe type devices
else:
    # Unknown device type, take appropriate action
```

Terminal - cisco@cisco-python: /var/local/PyNE/labs/sections/section09

```
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:10.3.21.9,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
```

Terminal - cisco@cisco-python: /var/local/PyNE/labs/sections/section09

```
from pprint import pprint

# Print heading
print ''
print 'Counts of different OS-types for all devices'
print '============================================'

os_types = { 'Cisco IOS':    {'count':0, 'devs':[] },
             'Cisco Nexus':  {'count':0, 'devs':[] },
             'Cisco IOS-XR': {'count':0, 'devs':[] },
             'Cisco IOS-XE': {'count':0, 'devs':[] } }

# Read all lines of device information from file
file = open('devices','r')
for line in file:

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

    # Put device information into dictionary for this one device
    device_info = {} # Create a dictionary of device info
    device_info['name'] = device_info_list[0]
    device_info['os-type'] = device_info_list[1]

    name = device_info['name'] # get the device name
    os = device_info['os-type'] # get the OS-type for comparisons

"S04-4-os-types.py" 50 lines, 1624 characters written
```

Continued on next page

```
    # Based on the OS-type, increment the count and add name to list
    if os == 'ios':
        os_types['Cisco IOS']['count'] += 1
        os_types['Cisco IOS']['devs'].append(name)

    elif os == 'nx-os':
        os_types['Cisco Nexus']['count'] += 1
        os_types['Cisco Nexus']['devs'].append(name)

    elif os == 'ios-xr':
        os_types['Cisco IOS-XR']['count'] += 1
        os_types['Cisco IOS-XR']['devs'].append(name)

    elif os == 'ios-xe':
        os_types['Cisco IOS-XE']['count'] += 1
        os_types['Cisco IOS-XE']['devs'].append(name)

    else:
        print "   Warning: unknown device type:", os

pprint(os_types)
print '' # Print final blank line

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

Note the use of dictionary and embedding another dictionary.

```
Terminal - cisco@cisco-python: /var/local/PyNE/labs/sections/section09
cisco@cisco-python:/var/local/PyNE/labs/sections/section09$ python S04-4-os-types.py

Counts of different OS-types for all devices
============================================
{'Cisco IOS': {'count': 2, 'devs': ['d01-is', 'd02-is']},
 'Cisco IOS-XE': {'count': 2, 'devs': ['d07-xe', 'd08-xe']},
 'Cisco IOS-XR': {'count': 2, 'devs': ['d05-xr', 'd06-xr']},
 'Cisco Nexus': {'count': 2, 'devs': ['d03-nx', 'd04-nx']}}
```