Defining Classes

In this lab, you will define a base class for a generic networking device and sub-classes for IOS and IOS-XR types of network devices. Specifically, you will read the list of devices from a file, and will create specific types of device objects depending on the OS-type of the device.

Define Base and Child Classes

In this exercise, you will define a base class and child classes for different types of network devices.

Step 1

Define a base class for a generic network device, with an initialization function to set device name, IP, username, and password. Because the device type is unknown for a generic device, set os_type to 'unknown'.

Answer

```
#---- Class to hold information about a generic network device --------
class NetworkDevice():

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        self.name = name
        self.ip_address = ip
        self.username = user
        self.password = pw
        self.os_type = 'unknown'
```

Step 2

Define two child classes which derive from the base NetworkDevice class. One class will be for IOS devices, the other will be for IOS-XR devices. Each specific device class will have an initialization function which takes as input the device name, IP, username, and password. It will set its os_type to the appropriate value.

Answer

```
#---- Class to hold information about an IOS-XE network device --------
class NetworkDeviceIOS(NetworkDevice):

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)
        self.os_type = 'ios'

#---- Class to hold information about an IOS-XR network device --------
class NetworkDeviceXR(NetworkDevice):

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)
        self.os_type = 'ios-xr'
```

Step 3

Create a function which reads the devices file located in the PRNE/section13 folder, creates the device objects, and adds them to the list. Your function will create different device objects depending on whether the os-type of the device that is read from the file is an IOS device, an IOS-XR device, or neither. If neither, your function will ignore the device and continue reading the file.

Your function should return a list of your created devices objects (note that some will be IOS objects, some will be IOS-XR objects.

```
#---- Function to read device information from file ------------------
def read_device_info(devices_file):

    devices_list = []

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

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

        # Create a device object with this data
        if device_info[1] == 'ios':

            device = NetworkDeviceIOS(device_info[0],device_info[2],
                                      device_info[3],device_info[4])

        elif device_info[1] == 'ios-xr':

            device = NetworkDeviceXR(device_info[0],device_info[2],
                                     device_info[3],device_info[4])

        else:
            continue  # go to the next device in the file

        devices_list.append(device) # add this device object to list

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

Step 4

Create a function which prints the devices list that was created.

Answer

```
#---- Function to go through devices printing them to table -----------
def print_device_info(devices_list):

    print ''
    print 'Name     OS-type  IP address       Username  Password'
    print '------   -------  --------------   --------  --------'

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

        print '{0:8} {1:8} {2:16} {3:8} {4:8}'.format(device.name,
                                                      device.os_type,
                                                      device.ip_address,
                                                      device.username,
                                                      device.password)

    print ''
```

Step 5

Your main code should read the devices file, and print the device object information.

Your output should look something like the following:

```
Name     OS-type  IP address       Username  Password
------   -------  --------------   --------  --------
d01-is   ios      10.3.21.5        cisco    cisco
d02-is   ios      10.3.21.6        cisco    cisco
d05-xr   ios-xr   10.3.21.9        cisco    cisco
d06-xr   ios-xr   10.3.21.10       cisco    cisco
```

Answer

```
#---- Main: read device info, then print -----------------------------

devices = read_device_info('devices')
print_device_info(devices)
```

Define Classes and Overriding Methods

In this exercise, you will define a base class for a generic networking device and sub-classes for IOS and IOS-XR devices. You will also override a method to have specific implementations of the method for each child class.

Step 6

Define a base class for a generic network device, with an initialization function to set device name, IP, username, and password.

Note: for this exercise you will not be storing os_type. You will be creating methods which return the OS type specifically for the base or child classes.

There will be a method for the base class called `get_type` which returns `base`.

Answer

```
#---- Class to hold information about a generic network device --------
class NetworkDevice():

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        self.name = name
        self.ip_address = ip
        self.username = user
        self.password = pw

    def get_type(self):
        return 'base'
```

Step 7

Define two child classes which derive from the base `NetworkDevice` class. One class will be for IOS devices, the other will be for IOS-XR devices. Each specific device class will have an initialization function which takes as input the device name, IP, username, and password.

Create methods for each child class called `get_type`. For the IOS class, this method will return 'IOS', for the XR class, this method will return 'IOS-XR'.

In your initialization method, rather than setting your attributes directly, you will call into the initialization method for your base class to set the attributes for name, IP, username, and password.

Answer

```
#---- Class to hold information about an IOS-XE network device --------
class NetworkDeviceIOS(NetworkDevice):

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)

    def get_type(self):
        return 'IOS'

#---- Class to hold information about an IOS-XR network device --------
class NetworkDeviceXR(NetworkDevice):

    def __init__(self, name, ip, user='cisco', pw='cisco'):
        NetworkDevice.__init__(self, name, ip, user, pw)

    def get_type(self):
        return 'IOS-XR'
```

Step 8

Create a function which takes as input the name of the devices file located in the PRNE/section13 folder and creates device objects appropriate to the device os-type. Do not skip any devices; if it is an IOS-XR device, create an XR object; if it is an IOS device, create an IOS object; if it is neither, then create the generic, base type of device.

Your function will return a list of your created devices objects; some will be IOS objects, some will be XR objects, and some will be base objects.

```
#---- Function to read device information from file -------------------
def read_device_info(devices_file):

    devices_list = []

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

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

        # Create a device object with this data
        if device_info[1] == 'ios':

            device = NetworkDeviceIOS(device_info[0],device_info[2],
                                      device_info[3],device_info[4])

        elif device_info[1] == 'ios-xr':

            device = NetworkDeviceXR(device_info[0],device_info[2],
                                     device_info[3],device_info[4])

        else:
            device = NetworkDevice(device_info[0],device_info[2],
                                   device_info[3],device_info[4])

        devices_list.append(device) # add this device object to list

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

    return devices_list
```

Step 9

Create a function which prints the devices list that was created.

Answer

```
#---- Function to go through devices printing them to table -----------
def print_device_info(devices_list):

    print ''
    print 'Name     OS-type  IP address       Username  Password'
    print '------   -------  --------------   --------  --------'

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

        print '{0:8} {1:8} {2:16} {3:9} {4:9}'.format(device.name,
                                                      device.get_type(),
                                                      device.ip_address,
                                                      device.username,
                                                      device.password)

    print ''
```

Step 10

Your main code should read the devices file, and print the device object information.

Your output should look similar to:

```
Name      OS-type  IP address        Username  Password
------    -------  --------------    --------  --------
d01-is    IOS      10.3.21.5         cisco     cisco
d02-is    IOS      10.3.21.6         cisco     cisco
d03-nx    base     10.3.21.7         cisco     cisco
d04-nx    base     10.3.21.8         cisco     cisco
d05-xr    IOS-XR   10.3.21.9         cisco     cisco
d06-xr    IOS-XR   10.3.21.10        cisco     cisco
d07-xe    base     10.3.21.19        cisco     cisco
d08-xe    base     10.3.21.22        cisco     cisco
```

Answer

```
#---- Main: read device info, then print -----------------------------

devices = read_device_info('devices')
print_device_info(devices)
```