Defining Child Classes

In this lab, you will define a base class for a generic network device, and will define child classes for specific device types: IOS and IOS-XR. The main functionality for this lab will be to implement `connect` and `get_interfaces` methods for your child classes.

You will read in the **challenge-devices** file that is located in the **~/Desktop/PRNE/section13** folder. This file has the following information:

```
ios-01,ios,10.30.30.1,admin,cisco
ios-02,ios,10.30.30.2,admin,cisco
xr-01,ios-xr,10.0.0.1,cisco,cisco
base-01,unknown,10.20.20.1,unknown,unknown
```

Step 1

Define a base network device class with the usual attributes for name, IP, username, and password. Your base class will have an empty connect method that sets the session attribute to `None`, and a `get_interfaces` method that returns a string such as "Base device, cannot get interfaces".

```
import pexpect

#---- 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.interfaces = ''

    def connect(self):
        self.session = None

    def get_interfaces(self):
        self.interfaces = '--- Base Device, does not know how to get interfa
```

Step 2

Define child device-specific classes for IOS and IOS-XR device types.

For your IOS class, your connect method will actually connect to the device, storing the session as an attribute. Your `get_interfaces` method will do run the `show int brief` command on the device, and will store the output.

For your IOS-XR class, you will create 'stub' methods for both `connect` and `get_interfaces`. The `connect` method will set the session to None, and the `get_interfaces` method will return a string saying "Getting interface information a different way", to simulate the behavior of different object types satisfying requests in a device-specific manner.

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

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

    #---- Connect to device ------------------------------------------
    def connect(self):

        print '--- connecting IOS: telnet '+self.ip_address

        self.session = pexpect.spawn('telnet '+self.ip_address, timeout=20)
        result = self.session.expect(['Username:', pexpect.TIMEOUT])

        # Check for failure
        if result != 0:
            print '--- Timeout or unexpected reply from device'
            return None

        result = self.session.expect(['Password:', pexpect.TIMEOUT])
        # Check for failure
        if result != 0:
            print '--- Timeout or unexpected reply from device'
            return None
```

```
        # Successfully got password prompt, logging in with password
        self.session.sendline(self.password)
        self.session.expect('>')

    #---- Get interfaces from device ---------------------------------
    def get_interfaces(self):

        self.session.sendline('show interfaces summary')
        result = self.session.expect('>')

        self.interfaces = self.session.before


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

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

    #---- Connect to device ------------------------------------------
    def connect(self):

        print '--- connecting XR: ssh '+self.username+'@'+self.ip_address


    #---- Get interfaces from device ---------------------------------
    def get_interfaces(self):

        self.interfaces = '--- XR Device interface info ---'
```

Step 3

Create a function to read in the 'challenge-devices' file, creating and returning the list of device objects, created as appropriate for each IOS, IOS-XR, and generic device in the file.

```
#============================================================================
def read_devices_info(devices_file):

    devices_list = []

    file = open(devices_file,'r')
    for line in file:

        device_info = line.strip().split(',')

        # 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)

    return devices_list
```

Step 4

Create a function to print the device information (name, IP, username, password), and then display the interface information.

Note

Your 'interfaces' for this exercise is only the complete string returned by the device via the `show interface summary` command. You do not need to parse that output, since this challenge is about OOP and not concerned with details of parsing CLI output.

```
#===================================================================
def print_device_info(device):

    print '-------------------------------------------------------'
    print '    Device Name:      ',device.name
    print '    Device IP:        ',device.ip_address
    print '    Device username:  ',device.username,
    print '    Device password:  ',device.password

    print ''
    print '    Interfaces'
    print ''

    print device.interfaces
    print '-------------------------------------------------------\n\n'
```

Step 5

Your main code should read the devices file using your 'read' function, and for each device object, perform a connect, then perform a get_interfaces, then print the device data using your 'print' function.

Answer

```
#===================================================================
# Main program: connect to device, show interface, display

devices_list = read_devices_info('challenge-devices')

for device in devices_list:

    print '==== Device ============================================================

    session =  device.connect()     # Connect to device

    device.get_interfaces()         # Get interface info for this device
    print_device_info(device)       # Print interface info for this device
```