LAB – DEFINING CLASSES

OBJECTIVE

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 the file **devices-12.txt**, and will create specific types of device objects depending on the OS-type of the device.

PART 1

Open a terminal and switch to the lab directory

STEP 1: OPEN A TERMINAL WINDOW

Double-click the Terminal icon on the desktop to open the terminal window for use in this lab.

STEP 2: CHANGE DIRECTORY

Change to the directory **labs/prne/** in the user home directory, which holds the files for the course labs.

```
~$ cd labs/prne/
```

PART 2

Open **Visual Studio Code**, create a new file and save it with a filename of **defining-classes-part-2.py**, ensuring to save the file in the **~/labs/prne/** directory.
This python application defines a base class and child classes for different types of network devices.

STEP 1: DEFINE BASE CLASS

Define a base class for a generic network device, with an initialisation function to set device name, IP, username, and password. As it is for a generic device, the device type is unknown, set the **os_type** to **unknown**.

```
# 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 CHILD CLASSES

Define two child classes that 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.

```
# 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 FUNCTION TO READ INFORMATION FROM FILE

Create a function that reads the devices from the file **devices-12.txt**, 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 read in 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:

        # Get device info into list
        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:
            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 FUNCTION TO DISPLAY INFORMATION

Create a function that displays the created devices list.

```
# 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, displaying 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: CREATE MAIN CODE

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

```
# Main: read device info, then print
devices = read_device_info('devices-12.txt')
print_device_info(devices)
```

STEP 6: SAVE, RUN AND VERIFY APPLICATION

Save you application and then run it from the terminal rather than from within visual studio code.

```
~/labs/prne$ python3 defining-classes-part-2.py
```

The output from your application will be displayed in your terminal window, verify that it is comparable to below.

```
devasc@labvm:~/labs/prne$ python3 defining-classes-part-2.py

Name     OS-type  IP address       Username  Password
------   -------  --------------   --------  --------
d01-is   ios      192.168.122.1    cisco     cisco
d02-is   ios      192.168.122.2    cisco     cisco
d05-xr   ios-xr   192.168.122.5    cisco     cisco
d06-xr   ios-xr   192.168.122.6    cisco     cisco
```

PART 3

Open **Visual Studio Code**, create a new file and save it with a filename of **defining-classes-part-3.py**, ensuring to save the file in the **~/labs/prne/** directory.
This python application defines 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 1: DEFINE BASE CLASS

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 that 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.

```
# 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 2: CREATE FUNCTION TO READ INFORMATION FROM FILE

Define two child classes that 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 initialisation 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 initialisation method for your base class to set the attributes for name, IP, username, and password.

```
# 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 3: CREATE FUNCTION TO READ INFORMATION FROM FILE

Create a function that reads the devices from the file **devices-12.txt** 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:

        # Get device info into list
        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)  # add this device object to list

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

    return devices_list
```

STEP 4: CREATE FUNCTION TO DISPLAY INFORMATION

Create a function that displays the created devices list.

```
# 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, displaying 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 5: CREATE MAIN CODE

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

```
# Main: read device info, then print
devices = read_device_info('devices-12.txt')
print_device_info(devices)
```

STEP 6: SAVE, RUN AND VERIFY APPLICATION

Save you application and then run it from the terminal rather than from within visual studio code.

```
~/labs/prne$ python3 defining-classes-part-3.py
```

The output from your application will be displayed in your terminal window, verify that it is comparable to below.

```
devasc@labvm:~/labs/prne$ python3 defining-classes-part-3.py

Name     OS-type  IP address       Username  Password
------   -------  --------------   --------  --------
d01-is   IOS      192.168.122.1    cisco     cisco
d02-is   IOS      192.168.122.2    cisco     cisco
d03-nx   base     192.168.122.3    cisco     cisco
d04-nx   base     192.168.122.4    cisco     cisco
d05-xr   IOS-XR   192.168.122.5    cisco     cisco
d06-xr   IOS-XR   192.168.122.6    cisco     cisco
d07-xe   base     192.168.122.7    cisco     cisco
d08-xe   base     192.168.122.8    cisco     cisco
```

PART 4 (OPTIONAL BUT HIGHLY RECOMMENDED)

As this lab is completed in NETLAB+ and your code files will be erased when the reservation ends, it is advisable to save your files in GitHub under your repository for this course.