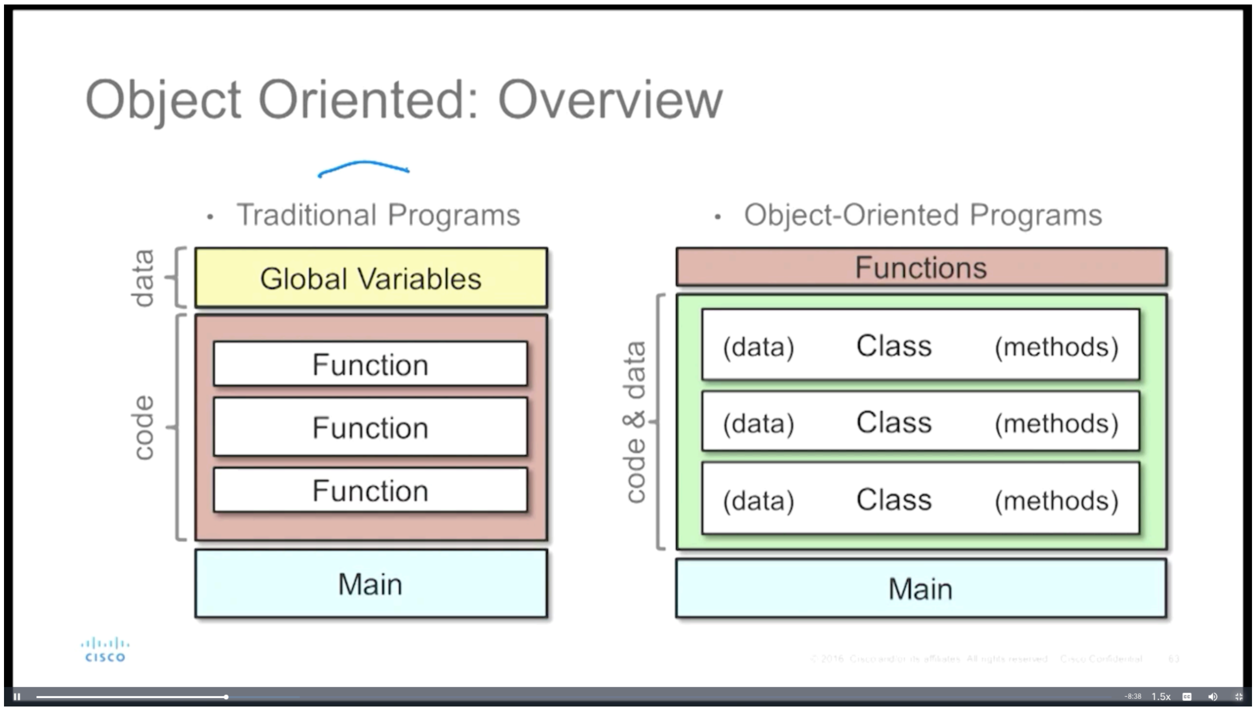
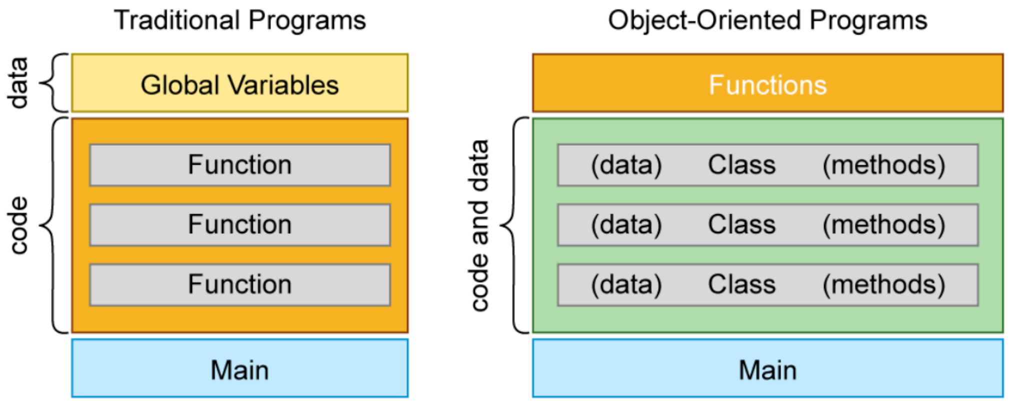


Object Oriented: Overview

Object-oriented programming (OOP) has been used regularly by software developers for a couple decades. Network management systems have used object-oriented programming languages such as C++ and Java for just as long. This lesson provides an overview of the principles of object-oriented development, and the ways that it can be applied to networks and network programmability.



The figure above compares software that is developed using traditional, structured programming techniques, versus those created using object-oriented technology.

Tradtional Programs = Structured Programming
Objects are defined in classes, classes will have data and methods
Methods/functions – terminology can be used interchangably

In a traditional, non-OO application, the following components are visible:

- **Global variables:** Typically structured applications will have a fair amount of data stored in 'global' storage, accessible by most, if not all, parts of the application. The danger with global variables is that they can be changed from anywhere, possibly causing unintended side-effects, and adding an element of complexity to applications.
- **Code:** Separating code into functions that are based on purpose is a good thing, and helps to isolate functionality and provide for code re-use. However, functions must have data passed to them, which can sometimes cause unnecessary complexity and overhead.
- **Main:** Every standalone application will have a 'main' set of code, that which gets executed from the start to the finish. Having code that is located in functions can help to keep the 'main' part of the application from getting too long and messy, but functions can only go so far in this endeavor.

In comparison, object-oriented software tends to be composed as follows:

- **Functions:** There will still be some independent, external functions, usually performing operations on smaller, isolated bits of information. Examples might be translation or MAC or IP addresses from strings to binary format, parsing of route strings, extracting version information, and so on
- **Code & data:** The unique aspect of object-oriented software development is that data and code will reside within 'objects'. The data (called attributes) will represent information and state regarding the object; the code will be object-based functions (called methods) which will perform operations on the object's data. Thus, all related data and code are organized in this manner, and held together in the object to which they belong.
- **Main:** Since there are less global variables (data is stored in objects), and fewer functions to call, object-oriented applications can have a simpler and smaller 'main' section of code.

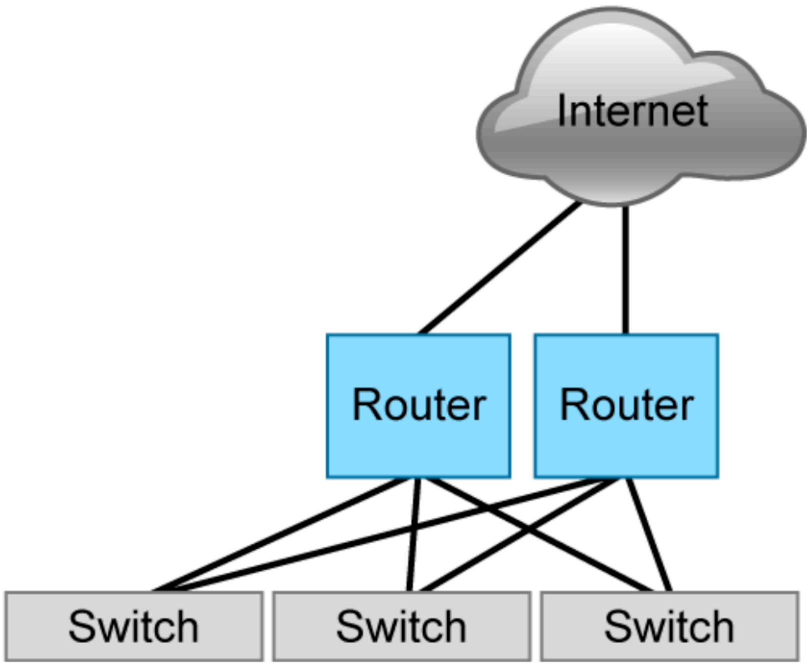
These abstract ideas begin to paint the picture of what object-oriented programming is all about, but the question may be asked, "What is an object, and how would I use it?" The following example will help solidify the theory into a tangible, real-world environment.

Object Oriented and Networks

Object-oriented programming languages and techniques have been used by network management systems for more than two decades. Why is OO so popular among network control and monitoring systems? Basically because networks lend themselves directly to the use of OO technology and modeling.

Consider the network picture below:

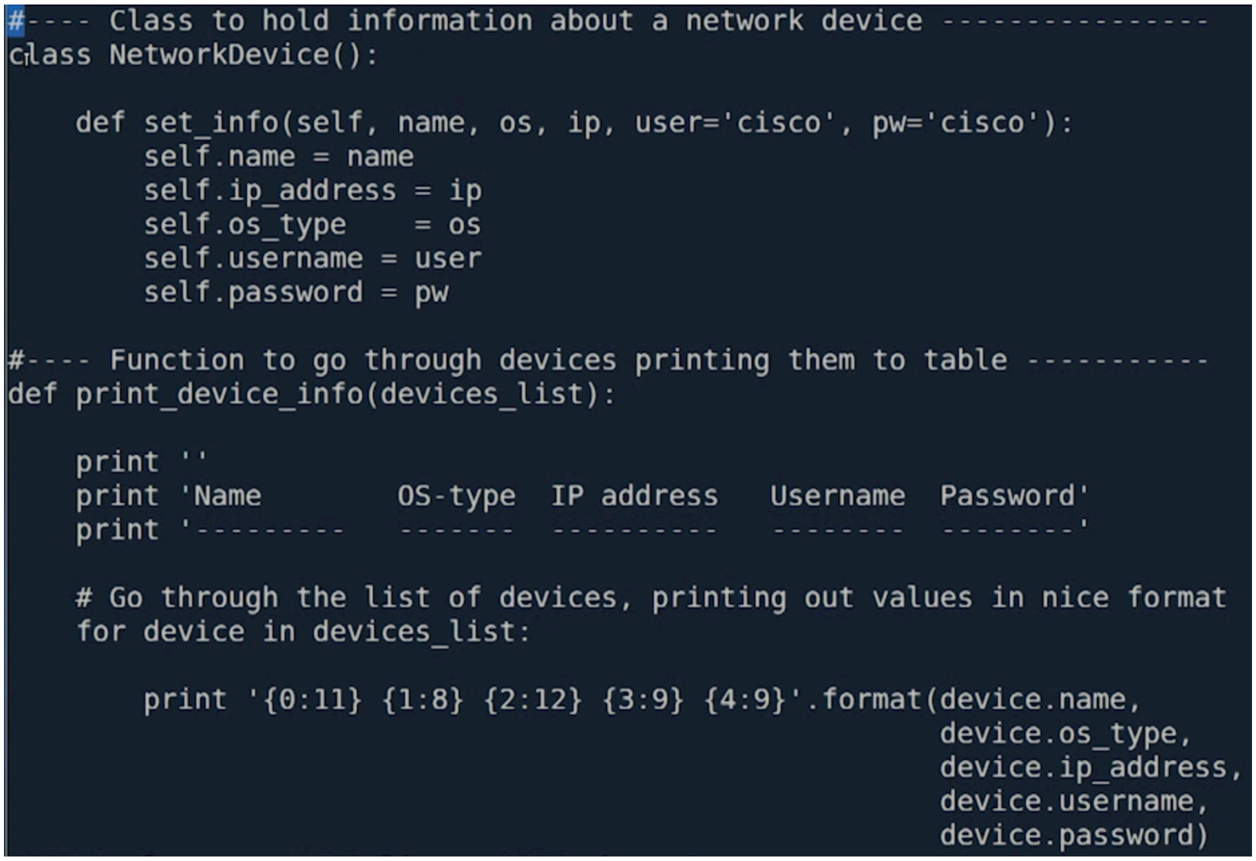
- Networks: A collection of objects
  - Routers
  - Switches
  - Links, Interfaces, and Ports
- Each object:
  - Data (Attributes)
  - Operations (methods)
- Model-driven



Every item in the picture can be thought of as an 'object' from an object-oriented programming point of view. Specifically:

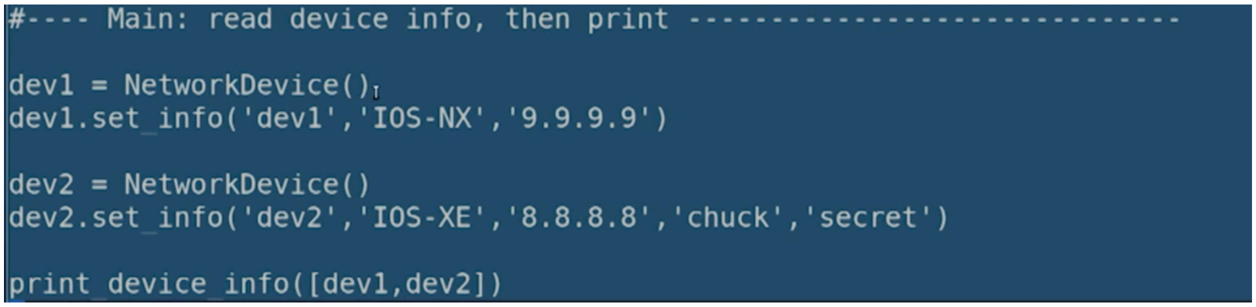
- Every switch has data that is associated with it, such as switch name, IP address, ports, spanning tree information, software version, and so on. Every switch also has certain operations that can be done to it: enable a port, disable a port, turn on edge security, set up ACLs, even download new software. Switch objects should be able to store much of this information, and provide methods for performing operations on that data.
- Every router has data that is associated with it such as router name, IP addresses, subnets and VLANs, interfaces, policy-based routing state and information, OSPF or IS-IS topology. And every router also has certain operations that can be done to it: enable an interface, disable an interface, set static routes, routing protocols, set up ACLs or policy-based routes, and so on.
- The internet itself can be thought of an object with attributes (access data, bandwidth, firewalling information), with operations (enable or disable firewall policies, set bandwidth levels for specific types of traffic, configuration of access parameters).

It is true that the actual devices themselves have the actual data, and have actual operations which can be performed. However, what was described above is the idea of model-driven control and management of systems. In this environment, software models of the actual devices are used to simplify the job of the network programmer. And through using these models (objects), the capabilities of a network programmability application can be increased significantly.



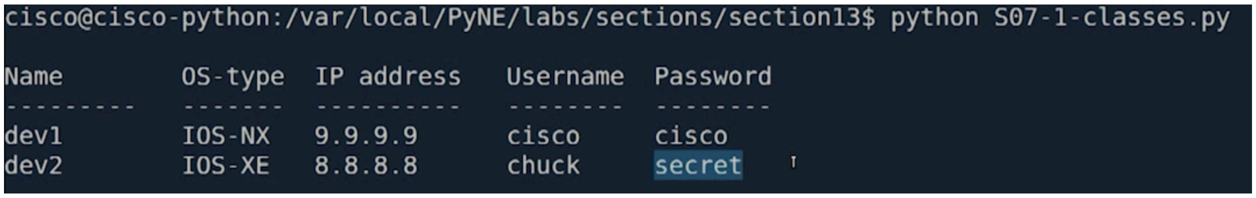
```
#---- Class to hold information about a network device ---------------
class NetworkDevice():

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

#---- 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 '{0:11} {1:8} {2:12} {3:9} {4:9}'.format(device.name,
                                                       device.os_type,
                                                       device.ip_address,
                                                       device.username,
                                                       device.password)
```

Notice the “self” variable



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

dev1 = NetworkDevice()
dev1.set_info('dev1','IOS-NX','9.9.9.9')

dev2 = NetworkDevice()
dev2.set_info('dev2','IOS-XE','8.8.8.8','chuck','secret')

print_device_info([dev1,dev2])
```

Create an object of the class “NetworkDevice” called dev1 and dev2 and call the method/function of set_info and pass in the a set of parameters



```
cisco@cisco-python:/var/local/PyNE/labs/sections/section13$ python S07-1-classes.py

Name        OS-type  IP address   Username  Password
---------   -------  ----------   --------  --------
dev1        IOS-NX   9.9.9.9      cisco     cisco
dev2        IOS-XE   8.8.8.8      chuck     secret
```