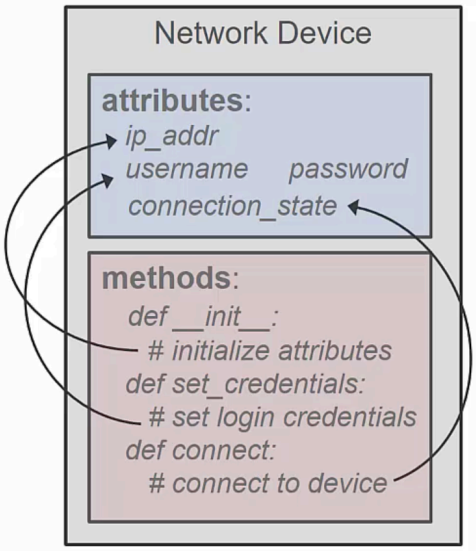
Methods (Functions)

Methods:

- Modify ('set') attributes of object
- Retrieve ('get') attributes of object
- Perform operations (e.g. 'connect') with object



'self'

`self`: 'this object instance'

- Every function takes `self` as the first parameter
- Every instance variable must be preceded with `self`.

```
class NetworkDevice():

    def __init__(self, ... ):
        self.name = ...
        self.ip_address = ...

    def set_creds(self, ... ):
        self.user = ...
        self.pw = ...
```

'__init__'

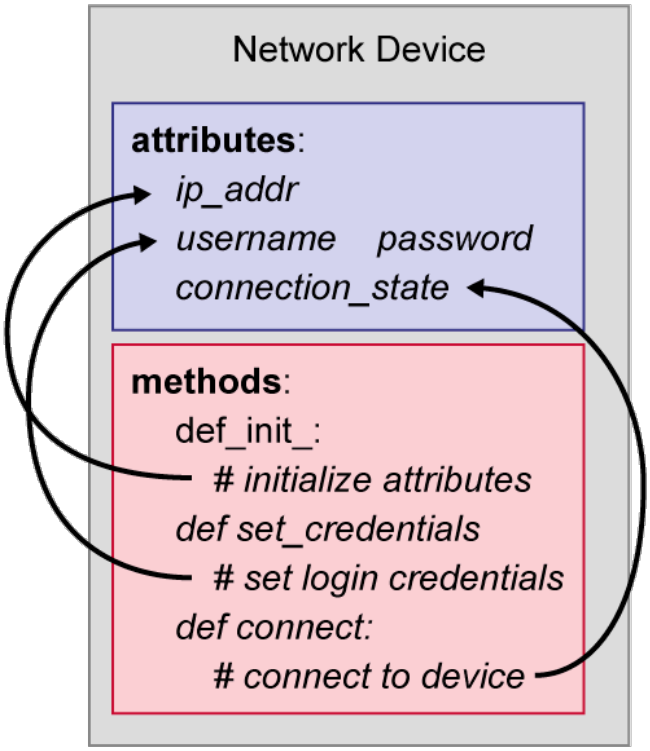
__init__: called at object initialization

- Great place to initialize your instance variables
- Parameters specified when your object is created are passed to your __init__ function

```
class NetworkDevice():

    def __init__(self, ... ):

        self.name = ...
        self.ip_address = ...

        self.interfaces = []
        self.routes = {}

        self.version = ''

    def connect(...)
    def retrieve_interfaces(...)
    def retrieve_routes(...)
```


Init is a function for every class that is created. This is called when object is instantiated. Any parameters specified in init will be passed to init routine.

Methods are used to perform operations, typically on the instance data for the object, sometimes to execute some external operation based on attributes of the object.



In the following code, observe the two aspects of methods that: (a) Each parameter list must have 'self' as the first parameter; and (b) Referencing instance attributes requires the use of 'self.' before the attribute name.

```
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.session = None

    def get_name(self):
        return self.name

    def get_ip_address(self):
        return self.ip_address

    def connect(self):
        self.session = pexpect.spawn('ssh '+self.username+
                                     '@'+self.ip_address,
                                      timeout=20)
        result = self.session.expect(['password:', pexpect.TIMEOUT])
        ...
        self.session.sendline(self.password)
        ...
```

Observe the following from the code above:

- Initialization function (__init__) takes parameters include two with default values.
- 'self.' is used for referencing attributes of the object (instance attributes). Note that you will mostly be using either local variables (like 'result' in the example above) and instance variables (denoted with the 'self.' prefix
- The session object is local within the object – the calling code does not need to maintain this detail; it is the responsibility of the NetworkDevice object to maintain that information.

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

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

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

    devices = [] # Create a list for all devices

    # 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
        device = NetworkDevice(device_info[0],device_info[2],
                               device_info[1],device_info[3],device_info[4])

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

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

    return devices # return a reference to the list we created
```

```
#---- 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} {2:16} {3:9} {4:9}'.format(device.name,
                                                      device.os_type,
                                                      device.ip_address,
                                                      device.username,
                                                      device.password)

    print ''

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

devices_list = read_device_info('devices')
print_device_info(devices_list)

devices_list = read_device_info('real-devices')
print_device_info(devices_list)
```

```
cisco@cisco-python:/var/local/PyNE/labs/sections/section13$ python S07-2-init.py

Name        OS-type  IP address       Username  Password
---------   -------  --------------   --------  --------
d01-ios     ios      10.3.21.5        cisco     cisco
d02-ios     ios      10.3.21.6        cisco     cisco
d03-nx      nx-os    10.3.21.7        cisco     cisco
d04-nx      nx-os    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      ios-xe   10.3.21.19       cisco     cisco
d08-xe      ios-xe   10.3.21.22       cisco     cisco


Name        OS-type  IP address       Username  Password
---------   -------  --------------   --------  --------
ios-01      ios      10.30.30.1       admin     cisco
ios-02      ios      10.30.30.2       admin     cisco
ios-03      ios      10.30.30.3       admin     cisco

cisco@cisco-python:/var/local/PyNE/labs/sections/section13$
```