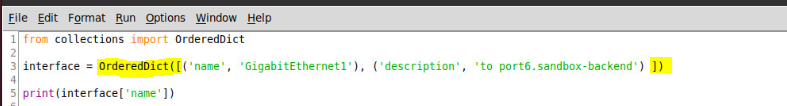
**IOS XE NETCONF YANG**

Directions: Complete each of the scripts below and submit them by the date specified in the assignment sheet and Blackboard. Submit **copies of your code** and **screenshots of the code running with each task**. Also, be sure to use the document, **Script Requirements as a guide** to writing good code. **Full credit will not be earned if you do not meet these script requirements.** **10 points**

**Ordered Dictionaries**

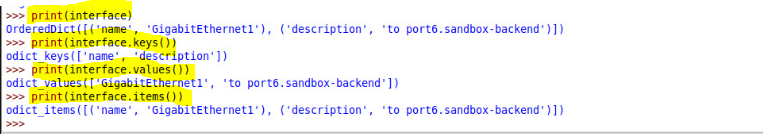
1. Ordered dictionaries are another data structure. They are a subclass of dictionary that stores data sequentially. For example, if you remove a key value pair and re-add it in an ordered dictionary, it will always be added at the end. It uses key/value pairs, but the key value pairs are enclosed in parenthesis and separated by a comma. Also, you access a value of an ordered dictionary by its key.



Note the import statement on line 1, which is required to access and construct an Ordered Dictionary. Also, the construction is different. It is constructed with the keyword, ‘OrderedDict’ followed by “ ([“ and ends with a “])”. It takes the form:

variable = OrderedDict([ (key,value),(key2,value2),(keyx,valuex)])

Accessing keys and values is the same as with a standard dictionary. Also, the .keys(), .values(), and .items() methods exist. See line 5 above for accessing a value.



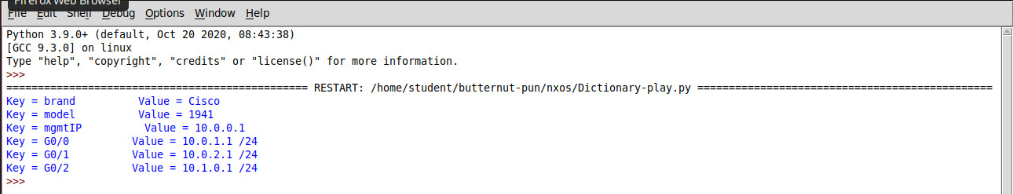
1. Convert the standard dictionary below to an Ordered Dictionary, and write a script that lists the keys and values as shown below:

router1 = {  
  "brand": "Cisco",  
  "model": "1941",  
  "mgmtIP": “10.0.0.1”,

“G0/0” : “10.0.1.1”,

“G0/1” : “10.0.2.1”,

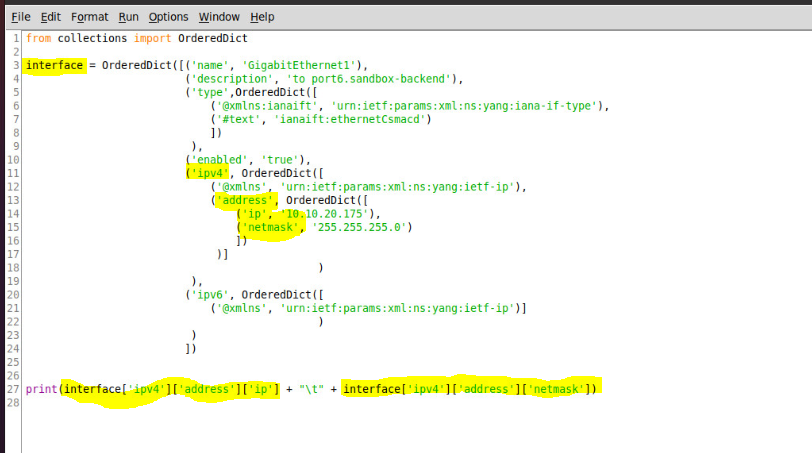
“G0/2” : “10.1.0.1”  
}



A screenshot of a computer program

Description automatically generated with medium confidence

1. You can also have nested dictionaries, as in the following:



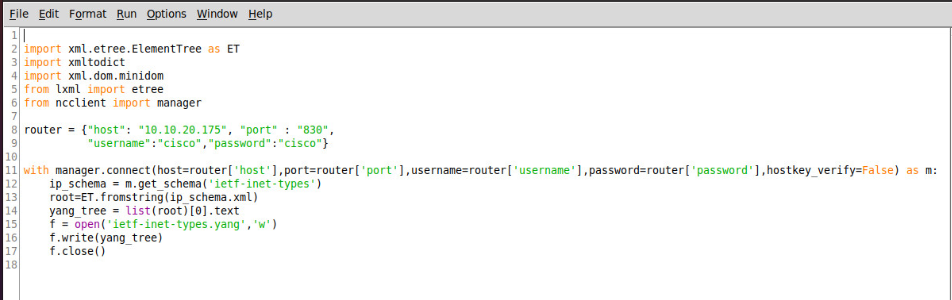


1. **Write a script that uses the above Ordered Dictionary and prints the interface name, the #text value of the interface type, the ip and the subnet mask. The code for this dictionary is located in the turnipTheBeet repository. If you don’t see it, you can pull an update down from GIT.**
2. To prepare your Workstation for Netconf and XML calls, do the following from a terminal prompt:
   1. sudo apt-get update
   2. sudo apt-get install python3-ncclient
   3. sudu apt-get install libxml2-dev
   4. sudo apt-get install libxslt1-dev
   5. pip3 install xmltodict
   6. pip3 install -U lxml
   7. pip3 install paramiko
   8. pip3 install pyang
   9. pip3 install jinja2
   10. copy the dicttoxml.py script in the turnipTheBeet repository to some place in your path.
3. Log into one of your IOSXE devices and type the following in global config:
   1. Netconf-yang

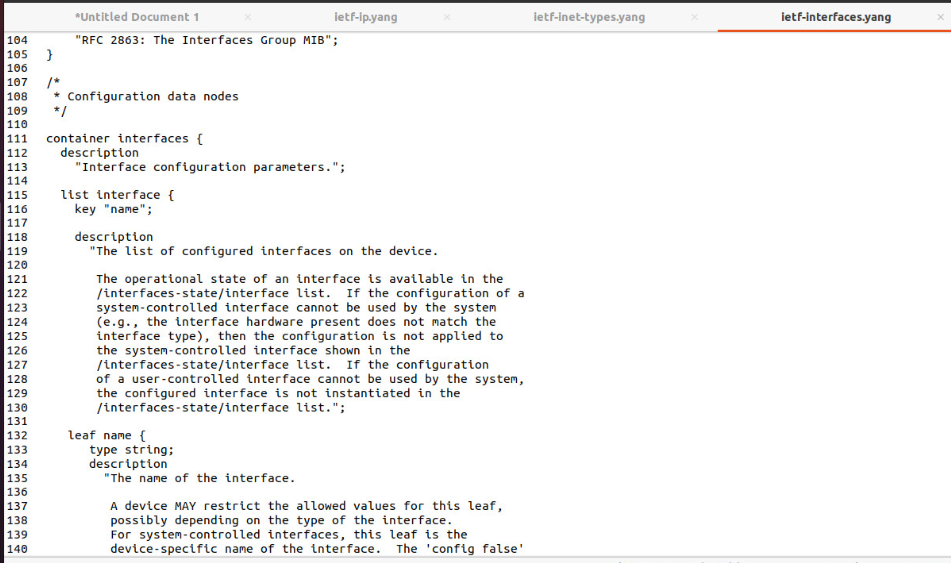
**ISO XE YANG Data Model**

YANG is a data model that we can use with RESTCONF and NETCONF to access our IOS XE devices. In this lab, we will be exploring NETCONF for configuring and monitoring our devices. We are going to configure a simple setting on an interface. To do so, we will explore one of our YANG model’s schema. NETCONF can use several transports, including SSH. We will SSH over port 830.

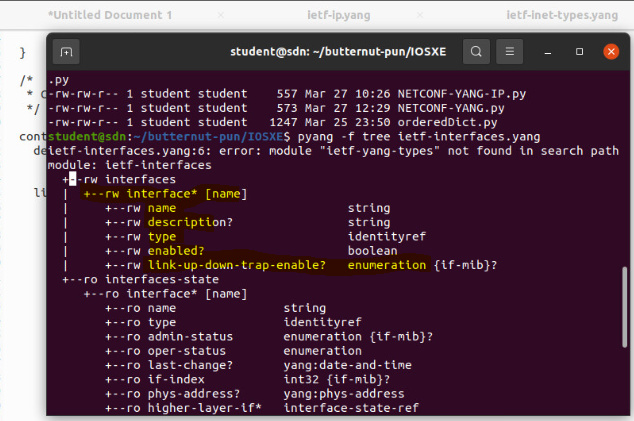
1. Update the Git, <https://github.com/dwright314/turnipTheBeet.git>
   1. In the git repository directory, type *git pull* to update the repository
2. The scripts needed to do this lab are in the repository, above. Copy all \*yang files to a directory somewhere from the turnipTheBeet repository. I ran the code below to generate the .yang files that I retrieved from the router.

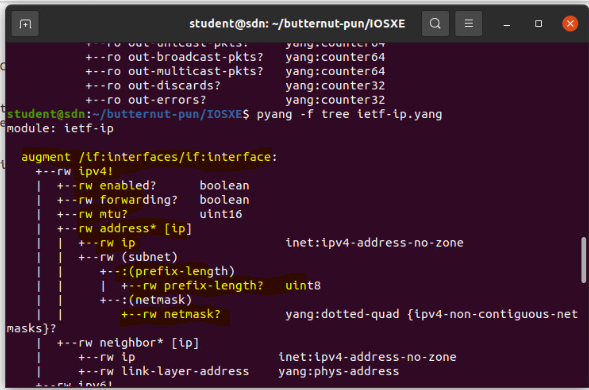


Line 11 makes our first NETCONF call on port 830. Note that I used a dictionary with values, shown on line 8, to pass parameters to the router in line 11. A connection named, m, is opened and I make a ‘get\_schema’ call for a particular schema name. I then write the schema to a file called: *schemaName*.yang. I have created these files for you. If you open the files with gedit, you will see that they are the same data we retrieved last week in RESTCONF.

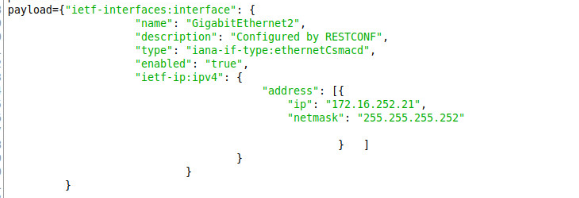


1. What we are going to do is use the pyang tool to explore the structure of the interfaces model to see if we can see how we arrived at the model we used last week. To do this, type the following two commands separately:
   1. pyang -f tree ietf-interfaces.yang
   2. pyang -f tree ietf-ip.yang
2. Below is sample output from the commands. Note the hierarchies:





1. Note that the hierarchies are defined and that the ip model augments the interface model, meaning it adds to it. See if you can spot the structure from our model below that we used with the yang models above(note we may not have used all available fields).



This is one way that we can learn which model that we want to use and the format of the model.

1. Be sure to include screenshots of any work done in this lab.