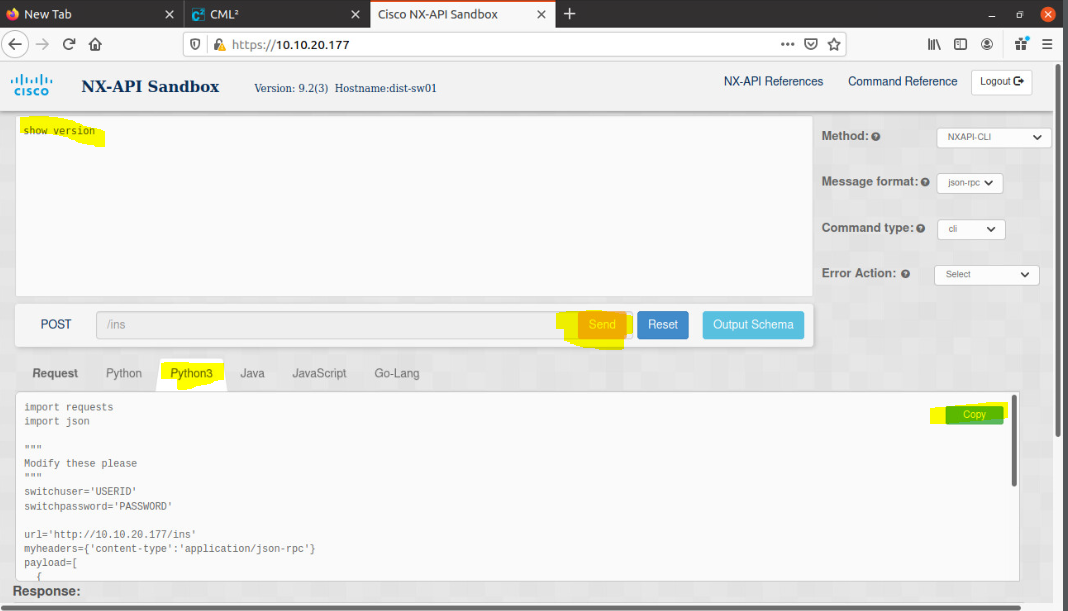
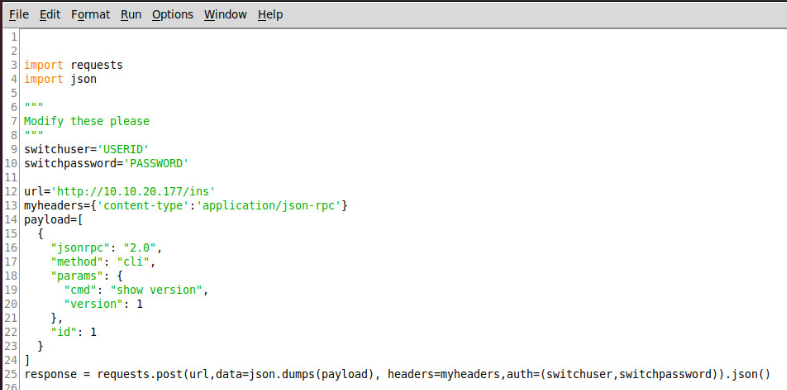
**NX-API SANDBOX**

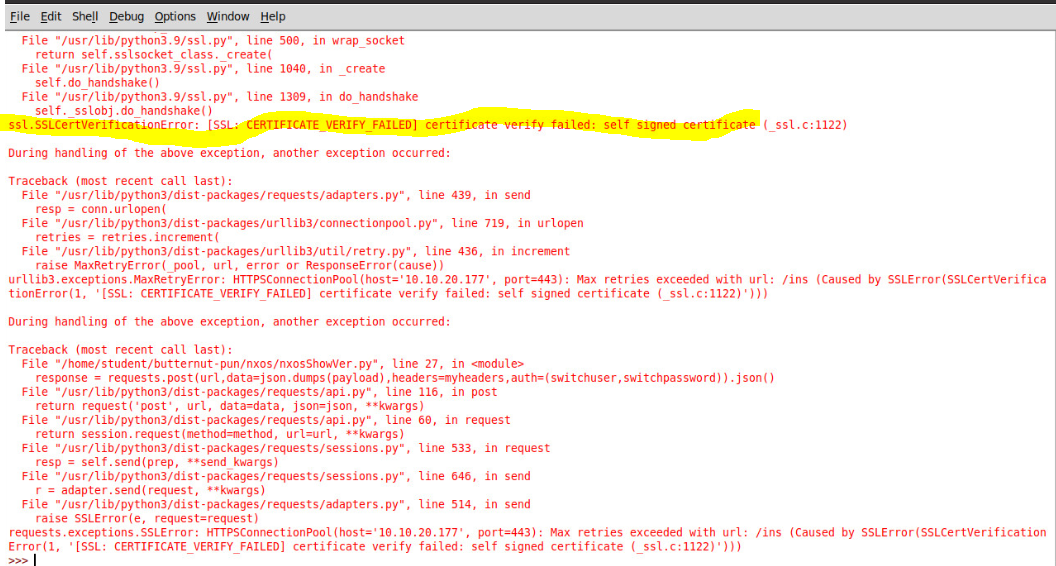
Instructions: In this lab you will set up and explore the NX-API Sandbox to facilitate writing API requests to NXOS Switches. Complete the script below and submit it before the beginning of class on the date specified in the assignment sheet. Submit copies of your code and screenshots of the code running 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. **20 points**

1. Make a Cisco Modeling Labs (CML) reservation
2. Once in, go to the console for the dist-sw01 switch
3. In config mode, type *feature nxapi* to enable the API interface on the switch
4. We want to do automated tasks and we want to use the NX-API Sandbox to help us build our Python code. The NX-API Sandbox accepts any NX-OS command and converts it to formatted data to send to the switch.
5. To access the sandbox, in your browser (on your Linux workstation) enter the management IP address of dist-sw01 (10.10.20.177). Accept the certificate and use cisco/cisco as your credentials.
6. We want to send a show version to our switch. Type *show version* in the command window and click **Send**. Then click on the **Python3** tab in the Request window below. After the code is generated, click the **Copy** button and paste the code to an IDLE window.



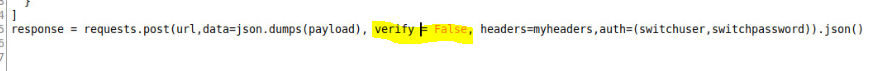


1. While the code above, displays nothing yet, we can attempt to run it. Can you see what needs to be modified for sure before we try to run it (hint: see lines 9 and 10)? We also need to send our request to the SSL interface of the Switch (line 12 needs url modified to https). Once you fix our authentication and url, run it so that we can explore the output in a Python interpreter shell. Don’t panic. It will fail miserably. See below:

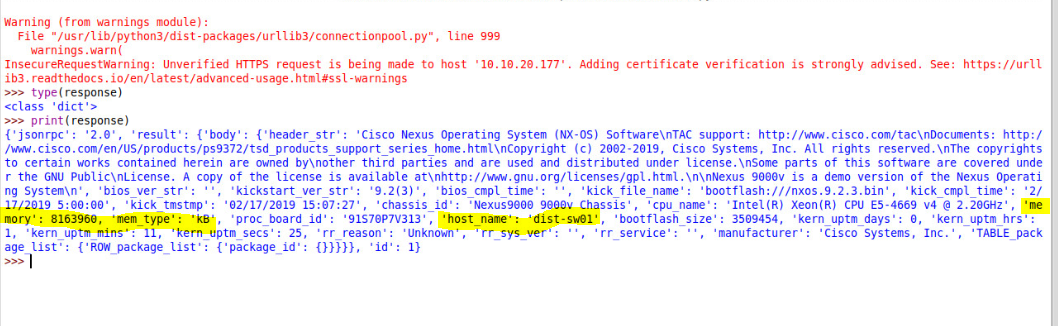


Note that our connection was ultimately failed. It failed by the switch because we didn’t encrypt it. We couldn’t encrypt our https request because our client rejected the untrusted certificate just like when your browser gave you a certificate warning when you logged into the NX\_API Sandbox. In your browser, you told it to trust the untrusted certificate. We will need to modify our request to accept an untrusted certificate. When you run it again after making the change, you will only receive a warning, but your script will run.

In our code, add the following and then re-run your script. Note nothing will happen yet because we have not added any output. You will receive a warning, however:



1. When you run it now, nothing happens except that get a warning. This is expected because we haven’t scripted any output. From the interpreter, verify the type and contents of the response variable:

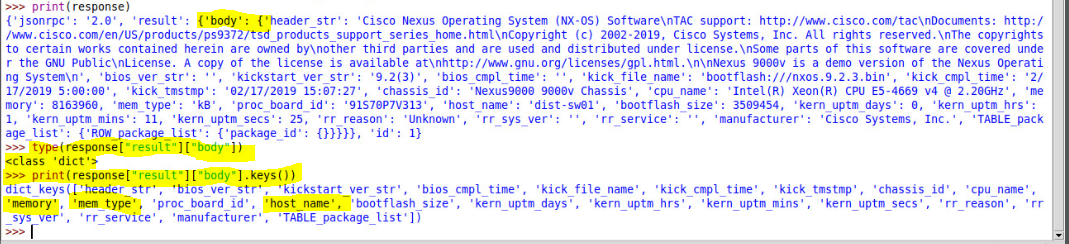


Now comes the beauty of the NX-API Sandbox. What if our task was to run a show version on our switch and then display our host name and memory? We would simply:

1. Build our API request in the sandbox
2. Make any changes (I recommend that you take good notes of the changes we make) to the code to ensure it works in our environment
3. Add code to customize the actions desired by your script. For example, change a value on a device or display a value on a device.

To do number 3, we need to determine the format that our response from the device is in. The information for our show version request is contained in our dictionary, highlighted above. The key/value pairs, host\_name/dist-sw01, memory/8163960, and mem\_type/kB are clearly dictionaries (curly braces “key”: “value” format). They are nested too. It can be a challenge to read them, but we can parse the object, one nested object at a time, to determine our structure.

We can see that response is a dictionary, with the first key/value pair being ‘jsonrpc: “2.0”. The second is pair is “result” : “body” which is a nested dictionary.



After printing response, I observed the curly braces to determine where my nested dictionaries are. I confirmed my observation with a type() method. Since response[“result”][“body”] is a dictionary, I won’t need to iterate through it like a list. I can use the keys() method to confirm I have the correct information and I can access individual key values as in:



or using a variable for the key:



1. **Modify your code in IDLE to display your hostname, memory and memory type with something like:**

Hostname = dist-sw01 Memory = 8163960 kB