**Discovery 29: Explore JSON Objects and Scripts in Python**

**Introduction**

In this lab exercise, you will first learn to work with basic JSON objects using the JSON module in Python. It will become important to understand and know as you start working with REST APIs that are returning data in JSON.

You will then be running Python scripts on a Cisco Nexus Operating System (Cisco NX-OS) virtual machine (Cisco NX-OSv) from the CLI. Because there is no text editor available at the CLI, you will create an EEM applet that will write the Python script for you. You do not need to dwell on the details of how EEM works, because that is beyond the scope of this training. However, the examples that are used here will give you some idea of what EEM can do.

Note that a more typical way to get Python code onto a device is to use the **copy** CLI command to copy files from a TFTP, FTP, or web server. Using EEM has certain advantages however, including the fact that all processing is on the device.

**Note**

If a page is unresponsive, it is probably caused by the **Unlock Login Keyring** prompt. You may need to scroll down to find the prompt. Click **Cancel** on that prompt and your ability to type on the page will be restored.

**Note**

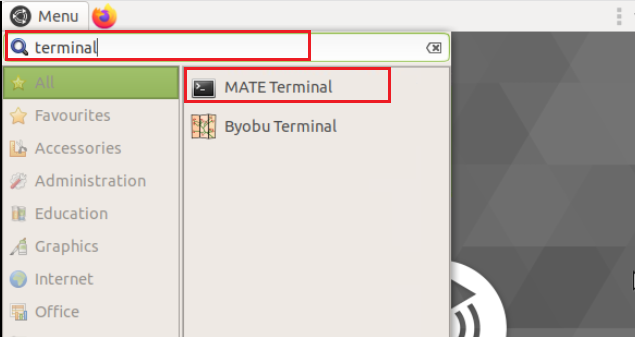
Actual lab devices are not used in this activity. This activity is a simulation based on a series of tasks. Often, it is not always possible to provide real lab equipment because of the nature and type of technology. These lab simulations are based on real equipment and actual lab tasks. The labs are performed using a simulation of real equipment. There are no setup or initialization time requirements, and the simulation is available immediately.

Your configuration task is as follows:

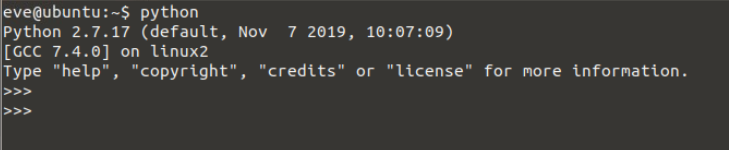
* Work with JSON objects in Python

**Task 1: Work with JSON Objects in Python**

**Step 1:** Search for terminal and select MATE Terminal from the Student Workstation.



**Step 2:** Enter the Python Shell using the **python** command.

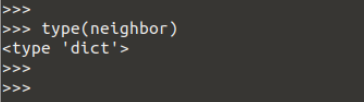
Enter the following information:****

**Step 3:** Create the following dictionary. You will see how a dictionary object with key-value pairs natively maps to JSON objects of name-value pairs. They are, for all intents and purposes the same thing.



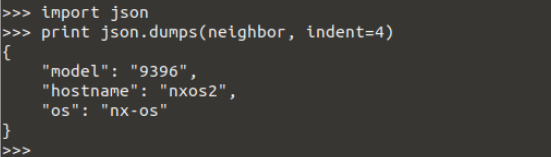
**Step 4:** Verify the data type of variable that is called neighbor using the type function. It is always important to see the differences of objects that are dictionaries and JSON strings that look like dictionaries. You will see the difference in the next few steps.

Enter the following information:



**Step 5:** Import the JSON module and dump the dictionary that is called neighbors as a JSON string.

Enter the following information:



**Note**

The function that is called dumps means dump as a string. It is also a helpful way to pretty print dictionaries.

**Step 6:** Save the object being dumped as a new variable called data. Print data and then check its data type using the type function.

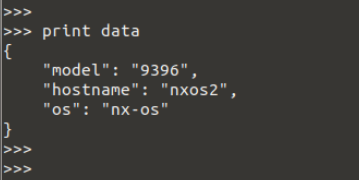
Enter the following information:



First print it as a string literal.

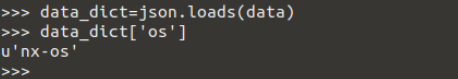


Now use the print statement.

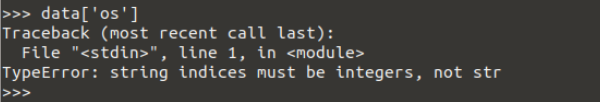
****

**Step 7:** Use the **json.loads()** functions to load a JSON string and make it a dictionary.

Enter the following information:



If you tried access os from data, you would see this error:



**Note**

APIs are going to return data as JSON strings. This means you will need to use json.loads to work with API responses. The reverse is true as well. To make an API request, you need to send a JSON string maning you’ll need to do a json.dumps(<your-dictionary>) to send your object over the wire

**Step 8:** Close the terminal window by selecting the **"x"** at the upper left portion of the window. Then, select the **Close Terminal** button on the pop-up window.

Make the following selections:

