

RINCON ULLOA YAZMIN ELIZABETH

Lab – RESTCONF with Python

Objectives

Part 1: RESTCONF basics in Python

Part 2: Modify interface configuration with RESTCONF in Python

Background / Scenario

Following up the previous lab activity, in this lab you will learn how to execute the RESTCONF API calls using Python scripts.

Required Resources

- Python 3.x environment
- Access to a router with the IOS XE operating system version 16.6 or higher.

Instructions

Part 1: RESTCONF in Python

In this part, you will use Python to request a RESTCONF API.

Step 1: Import modules and disable SSL warnings.

- In IDLE, click **File > New File** to open IDLE Editor.
- Save the file as **lab 2.5.py**.
- Enter the following commands to import the modules and disable SSL certificate warnings:

```
import json
import requests
requests.packages.urllib3.disable_warnings()
```

The **json** module includes methods convert JSON data to Python objects and vice versa. The **requests** module has methods that will let us send REST requests to a URI.

Step 2: Build the request components.

Create a string variable to hold the API endpoint URI and two dictionaries, one for the request header and one for the body JSON. These are the same tasks you completed in the Postman application.

- Create a variable named **api_url** and assign the URL (adjust the IP address to match the router's current address).

```
api_url = "https://192.168.56.101/restconf/data/ietf-interfaces:interfaces"
```

- Create a dictionary variable named **headers** that has keys for **Accept** and **Content-type** and assign the keys the value **application/yang-data+json**.

```
headers = { "Accept": "application/yang-data+json",
            "Content-type": "application/yang-data+json"
          }
```

- f. Create a Python tuple variable named **basicauth** that has two keys needed for authentication, **username** and **password**.

```
basicauth = ("cisco", "cisco123!")
```

Step 3: Send the request.

You will now use the variables created in the previous step as parameters for the **requests.get()** method. This method sends an HTTP GET request to the RESTCONF API. You will assign the result of the request to a variable name **resp**. That variable will hold the JSON response from the API. If the request is successful, the JSON will contain the returned YANG data model.

- g. Enter the following statement:

```
resp = requests.get(api_url, auth=basicauth, headers=headers, verify=False)
```

The various elements of this statement are:

Element	Explanation
<code>resp</code>	the variable to hold the response from the API.
<code>requests.get()</code>	the method that actually makes the GET request.
<code>api_url</code>	the variable that holds the URL address string
<code>auth</code>	the tuple variable created to hold the authentication information
<code>headers=headers</code>	a parameter that is assigned to the headers variable
<code>verify=False</code>	disables verification of the SSL certificate when the request is made

- h. Save your script and run it. There will not be any output yet but the script should run without errors. If not, review the steps and make sure your code does not contain any errors.

```

LAB25_yazmin.py
2 Rincon Ulloa Yazmin Elizabeth
3 Miercoles 06 de Diciembre del 2023
4 Laboratorio 2.5
5 Lab-RESTCONF with Python
6 """
7
8 import json
9 import requests
10 requests.packages.urllib3.disable_warnings()
11
12 api_url = "https://10.10.20.48/restconf/data/ietf-interfaces:interfaces"
13 headers = {"Accept": "application/yang-data+json",
14           "Content-type": "application/yang-data+json"}
15
16
17 basicauth = ("developer", "Cisco12345")
18 resp = requests.get(api_url, auth=basicauth, headers=headers, verify=False)

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

ions.py", line 589, in request
  resp = self.send(prepare_request_kwargs)
File "C:\Users\azray\AppData\Local\Programs\Python\Python312\Lib\site-packages\requests\sessions.py", line 703, in send
  r = adapter.send(request, **kwargs)
File "C:\Users\azray\AppData\Local\Programs\Python\Python312\Lib\site-packages\requests\adapters.py", line 507, in send
  raise ConnectTimeout(e, request=request)
requests.exceptions.ConnectTimeout: HTTPConnectionPool(host='192.168.56.101', port=443): Max retries exceeded with url: /restconf/data/ietf-interfaces:interfaces (Caused by ConnectTimeoutError(urllib3.connection.HTTPSConnection object at 0x000001FF2DF9F770, 'Connection to 192.168.56.101 timed out. (connect timeout=None)'))
PS C:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS> & C:\Users\azray\AppData\Local\Programs\Python\Python312\python.exe c:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS\LAB25_yazmin.py
PS C:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS> & C:\Users\azray\AppData\Local\Programs\Python\Python312\python.exe c:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS\LAB25_yazmin.py
PS C:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS>

```

Ln 14, Col 56 Spaces: 4 UTF-8 CRLF Python 3.12.0 64-bit

Step 4: Evaluate the response.

Now the YANG model response values can be extracted from the response JSON.

- The response JSON is not compatible with Python dictionary and list objects so it is converted to Python format. Create a new variable called **response_json** and assign the variable **resp** to it adding the **json()** method to convert the JSON. The statement is as follows:

```
response_json = resp.json()
```

- You can verify that your code returns the JSON in the IDLE Shell by temporarily adding a print statement to your script, as follows:

```
print(response_json)
```

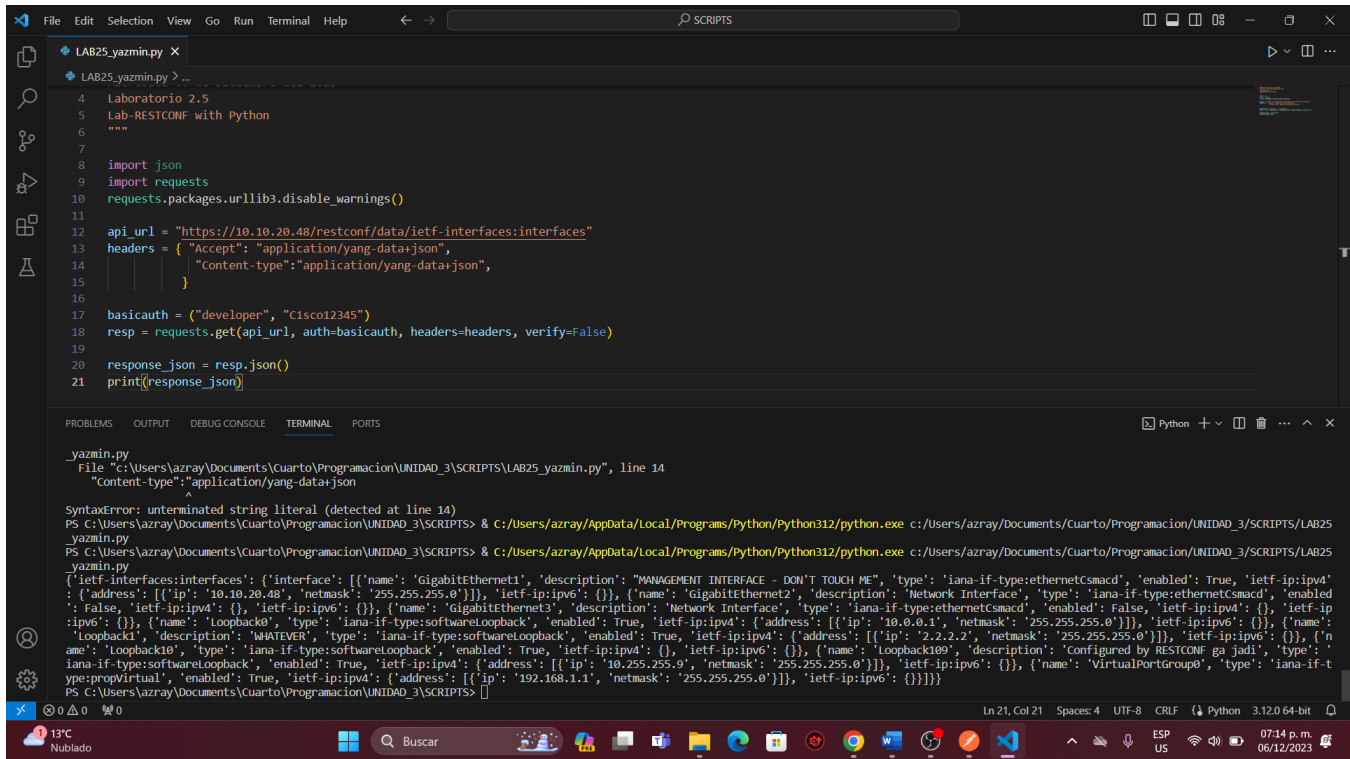
- Save and run your script. You should get output similar to the following although your service ticket number will be different:

```

>>>
RESTART: B:\DevNetAcad FS-E\Workshops\ETW3 Model Driven Programmability\Python Files with Solutions\idle\lab 2.9.py
{'ietf-interfaces:interfaces': {'interface': [{'name': 'GigabitEthernet1', 'type': 'iana-if-type:ethernetCsmacd', 'enabled': True, 'ietf-ip:ipv4': {}, 'ietf-ip:ipv6': {}}, {'name': 'Loopback1', 'description': 'WHATEVER', 'type': 'iana-if-type:softwareLoopback', 'enabled': True, 'ietf-ip:ipv4': {'address': [{'ip': '2.2.2.2', 'netmask': '255.255.255.0'}]}, 'ietf-ip:ipv6': {}}, {'name': 'Loopback2', 'description': 'NEWBUTSAME', 'type': 'iana-if-type:softwareLoopback', 'enabled': True, 'ietf-ip:ipv4': {}}, {'name': 'Loopback99', 'description': 'WHATEVER99', 'type': 'iana-if-type:softwareLoopback', 'enabled': True, 'ietf-ip:ipv4': {'address': [{'ip': '99.99.99.99', 'netmask': '255.255.255.0'}]}, 'ietf-ip:ipv6': {}}, {'name': 'Loopback100', 'description': 'TEST1', 'type': 'iana-if-type:softwareLoopback', 'enabled': True, 'ietf-ip:ipv4': {'address': [{'ip': '100.100.100.100', 'netmask': '255.255.255.0'}]}, 'ietf-ip:ipv6': {}}]}}
>>>

```

Lab – RESTCONF with Python



```
LAB25_yazmin.py
4 Laboratorio 2.5
5 Lab-RESTCONF with Python
6 """
7
8 import json
9 import requests
10 requests.packages.urllib3.disable_warnings()
11
12 api_url = "https://10.10.20.48/restconf/data/ietf-interfaces:interfaces"
13 headers = { "Accept": "application/yang-data+json",
14             "Content-type": "application/yang-data+json",
15             }
16
17 basicauth = ("developer", "Cisco12345")
18 resp = requests.get(api_url, auth=basicauth, headers=headers, verify=False)
19
20 response_json = resp.json()
21 print(response_json)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python 3.12.0 64-bit

Ln 21, Col 21 Spaces: 4 UTF-8 CRLF

13°C Nublado

PS C:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS> & C:\Users\azray\AppData\Local\Programs\Python\Python312\python.exe c:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS\LAB25_yazmin.py

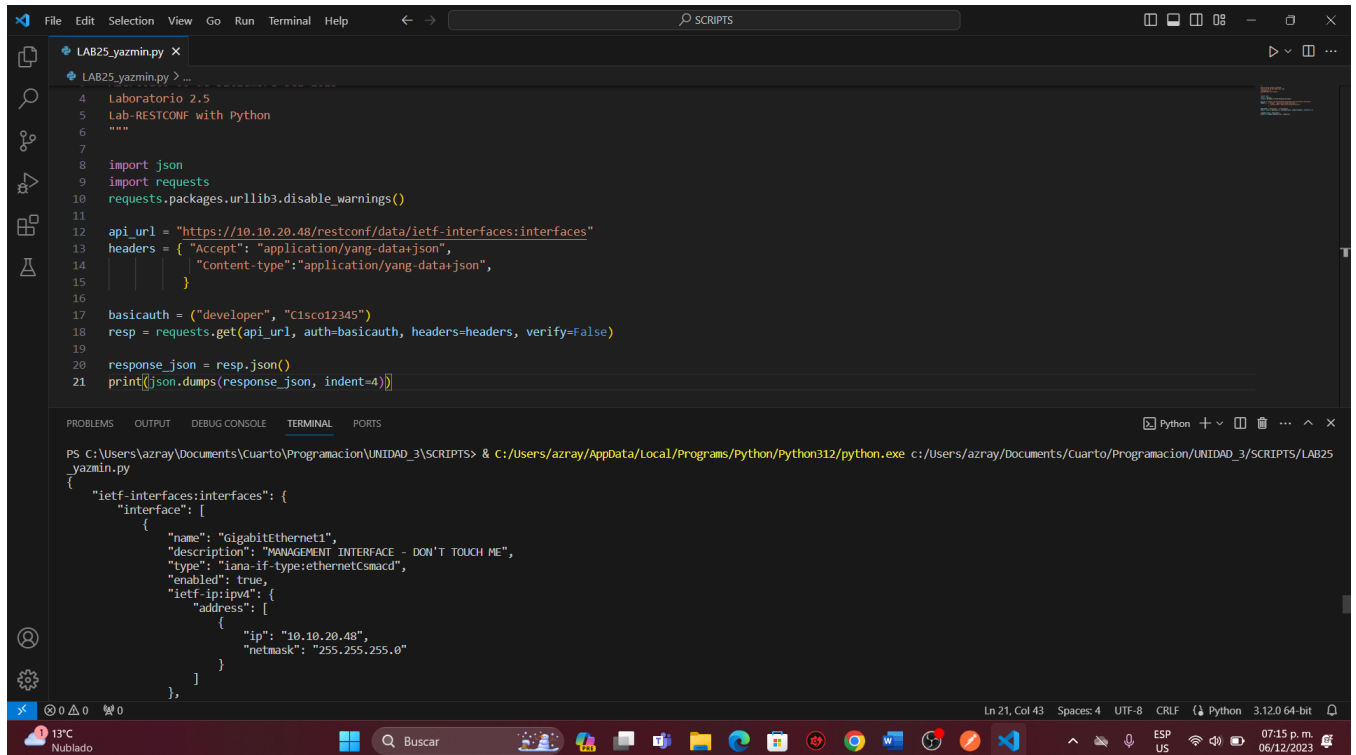
```
File "c:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS\LAB25_yazmin.py", line 14
    "Content-type": "application/yang-data+json"
    ^
SyntaxError: unterminated string literal (detected at line 14)
```

```
PS C:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS> & C:\Users\azray\AppData\Local\Programs\Python\Python312\python.exe c:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS\LAB25_yazmin.py
{"ietf-interfaces:interfaces": [{"interface": [{"name": "GigabitEthernet1", "description": "MANAGEMENT INTERFACE - DON'T TOUCH ME", "type": "iana-if-type:ethernetCsmacd", "enabled": True, "ietf-ip:ipv4": {"address": [{"ip": "10.10.20.48", "netmask": "255.255.255.0"}]}, "ietf-ip:ipv6": {}}, {"name": "GigabitEthernet2", "description": "Network Interface", "type": "iana-if-type:ethernetCsmacd", "enabled": False, "ietf-ip:ipv4": {}, "ietf-ip:ipv6": {}}, {"name": "GigabitEthernet3", "description": "Network Interface", "type": "iana-if-type:ethernetCsmacd", "enabled": False, "ietf-ip:ipv4": {}, "ietf-ip:ipv6": {}}, {"name": "Loopback0", "type": "iana-if-type:software-loopback", "enabled": True, "ietf-ip:ipv4": {"address": [{"ip": "10.0.0.1", "netmask": "255.255.255.0"}]}, "ietf-ip:ipv6": {}}, {"name": "Loopback1", "description": "WHATEVER", "type": "iana-if-type:software-loopback", "enabled": True, "ietf-ip:ipv4": {"address": [{"ip": "2.2.2.2", "netmask": "255.255.255.0"}]}, "ietf-ip:ipv6": {}}, {"name": "Loopback10", "type": "iana-if-type:software-loopback", "enabled": True, "ietf-ip:ipv4": {"address": [{"ip": "10.255.255.9", "netmask": "255.255.255.0"}]}, "ietf-ip:ipv6": {}}, {"name": "Loopback109", "description": "Configured by RESTCONF ga jadi", "type": "iana-if-type:software-loopback", "enabled": True, "ietf-ip:ipv4": {"address": [{"ip": "10.255.255.9", "netmask": "255.255.255.0"}]}, "ietf-ip:ipv6": {}}, {"name": "VirtualPortGroup0", "type": "iana-if-type:port-virtual", "enabled": True, "ietf-ip:ipv4": {"address": [{"ip": "192.168.1.1", "netmask": "255.255.255.0"}]}, "ietf-ip:ipv6": {}}]}}
```

- l. To prettify the output, use the `json.dumps()` function with the “indent” parameter:

```
print(json.dumps(response_json, indent=4))
```

- m. Save and run your script. If you experience errors, check the code again.



```
LAB25_yazmin.py
4 Laboratorio 2.5
5 Lab-RESTCONF with Python
6 """
7
8 import json
9 import requests
10 requests.packages.urllib3.disable_warnings()
11
12 api_url = "https://10.10.20.48/restconf/data/ietf-interfaces:interfaces"
13 headers = { "Accept": "application/yang-data+json",
14             "Content-type": "application/yang-data+json",
15             }
16
17 basicauth = ("developer", "Cisco12345")
18 resp = requests.get(api_url, auth=basicauth, headers=headers, verify=False)
19
20 response_json = resp.json()
21 print(json.dumps(response_json, indent=4))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python 3.12.0 64-bit

Ln 21, Col 43 Spaces: 4 UTF-8 CRLF

13°C Nublado

PS C:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS> & C:\Users\azray\AppData\Local\Programs\Python\Python312\python.exe c:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS\LAB25_yazmin.py

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "GigabitEthernet1",
        "description": "MANAGEMENT INTERFACE - DON'T TOUCH ME",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": true,
        "ietf-ip:ipv4": {
          "address": [
            {
              "ip": "10.10.20.48",
              "netmask": "255.255.255.0"
            }
          ]
        }
      }
    ]
  },
}
```

Part 2: Modify interface configuration with RESTCONF in Python

Step 5: Create the Python HTTP PUT request

In this part, you will use Python to request a RESTCONF API with a PUT method to create or modify existing configuration.

Step 6: Import modules and disable SSL warnings.

- n. In IDLE, click **File > New File** to open IDLE Editor.
- o. Save the file as **lab 2.5 part2.py**.
- p. Enter the following commands to import the modules and disable SSL certificate warnings:

```
import json
import requests
requests.packages.urllib3.disable_warnings()
```

The **json** module includes methods convert JSON data to Python objects and vice versa. The **requests** module has methods that will let us send REST requests to a URI.

Step 7: Build the request components.

Create a string variable to hold the API endpoint URI and two dictionaries, one for the request header and one for the body JSON. These are the same tasks you completed in the Postman application.

- q. Create a variable named **api_url** and assign the URL that targets the **Loopback99** interface.

```
api_url = "https://192.168.56.101/restconf/data/ietf-  
interfaces:interfaces/interface=Loopback99"
```

- r. Create a dictionary variable named **headers** that has keys for **Accept** and **Content-type** and assign the keys the value **application/yang-data+json**.

```
headers = {  
    "Accept": "application/yang-data+json",  
    "Content-type": "application/yang-data+json"  
}
```

- s. Create a Python tuple variable named **basicauth** that has two keys needed for authentication, **username** and **password**.

```
basicauth = ("cisco", "cisco123!")
```

- t. Create a Python dictionary variable **yangConfig** holding the YANG data to create new interface Loopback99 (you use here the dictionary data from the Postman lab before, be aware that the JSON's boolean **true** is in Python **True** with capital "T"):

```
yangConfig = {  
    "ietf-interfaces:interface": {  
        "name": "Loopback99",  
        "description": "WHATEVER99",  
        "type": "iana-if-type:softwareLoopback",  
        "enabled": True,  
        "ietf-ip:ipv4": {  
            "address": [  
                {  
                    "ip": "99.99.99.99",  
                    "netmask": "255.255.255.0"  
                }  
            ]  
        }  
    }  
}
```

```

    ]
},
"ietf-ip:ipv6": {}
}
}

```

Step 8: Send the PUT request.

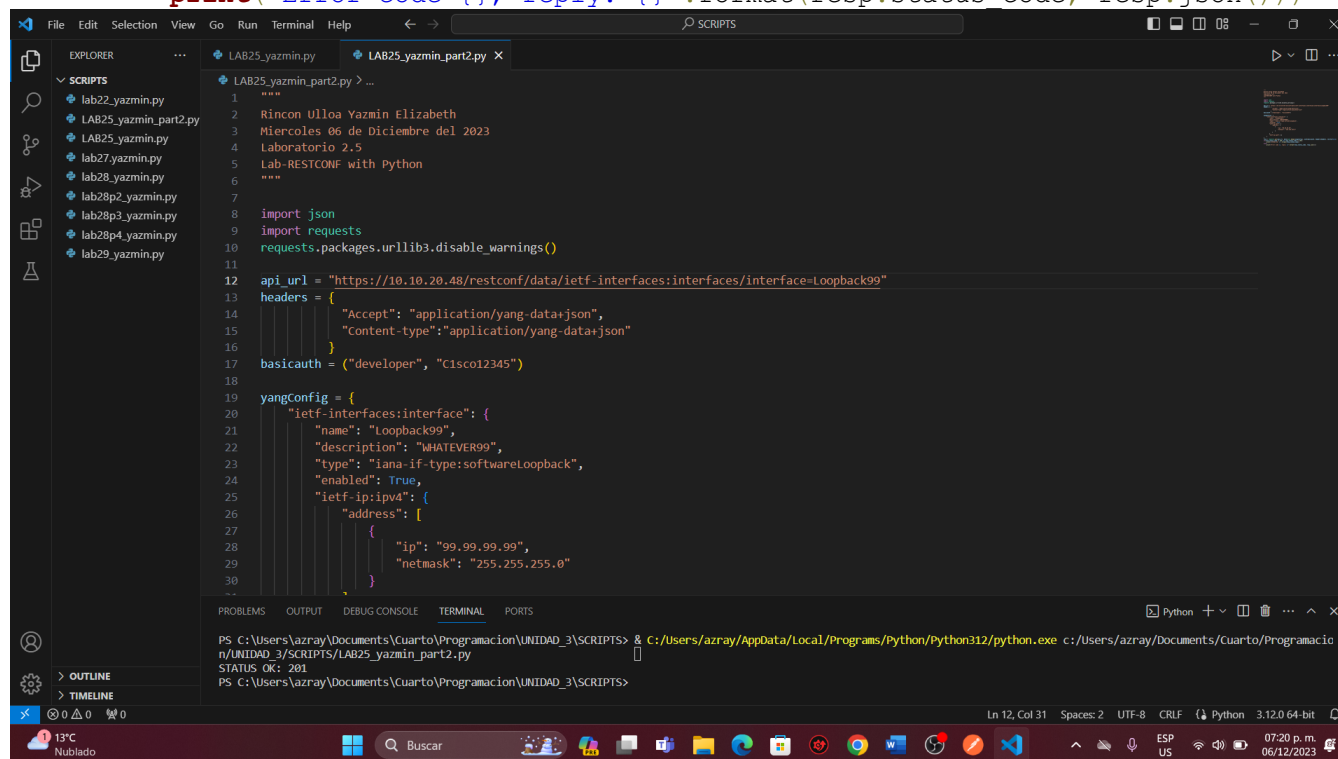
You will now use the variables created in the previous step as parameters for the **requests.put()** method. This method sends an HTTP PUT request to the RESTCONF API. You will assign the result of the request to a variable name **resp**. That variable will hold the JSON response from the API. If the request is successful, the JSON will contain the returned YANG data model.

u. Enter the following statement:

```

resp = requests.put(api_url, data=json.dumps(yangConfig), auth=basicauth,
headers=headers, verify=False)
if (resp.status_code >= 200 and resp.status_code <= 299):
    print("STATUS OK: {}".format(resp.status_code))
else:
    print("Error code {}, reply: {}".format(resp.status code, resp.json()))

```



The various elements of this statement are:

Element	Explanation
<code>resp</code>	the variable to hold the response from the API.
<code>requests.get()</code>	the method that actually makes the GET request.
<code>api_url</code>	the variable that holds the URL address string
<code>data</code>	the data to be sent to the API endpoint

auth	the tuple variable created to hold the authentication information
headers=headers	a parameter that is assigned to the headers variable
verify=False	disables verification of the SSL certificate when the request is made
resp.status_code	The HTTP status code in the API Request reply

v. Save your script and run it.

```
>>>
RESTART: B:\DevNetAcad PS-E\Workshops\ETW3 Model Driven Programmability\Python Files with Solutions\idle\lab 2.9 part2.py
STATUS OK: 201
>>> |
```

w. Verify using the IOS CLI that the new Loopback99 interface has been created (sh ip int brief).

x. Modify the code to delete the interface Loopback99.

y. What changes were applied to the code to delete the interface Loopback99?

```

11  api_url = "https://10.10.20.48/restconf/data/ietf-interfaces:interfaces/interface=Loopback99"
12  headers = {
13      "Accept": "application/yang-data+json",
14      "Content-type": "application/yang-data+json"
15  }
16  basicauth = ("developer", "C1sco12345")
17
18  yangConfig = {
19      "ietf-interfaces:interface": {
20          "name": "Loopback99",
21          "description": "WHATEVER99",
22          "type": "iana-if-type:softwareloopback",
23          "enabled": True,
24          "ietf-ip:ipv4": {
25              "address": [
26                  {
27                      "ip": "99.99.99.99",
28                      "netmask": "255.255.255.0"
29                  }
30              ]
31          },
32          "ietf-ip:ipv6": {}
33      }
34  }
35
36  resp = requests.delete(api_url, data=json.dumps(yangConfig), auth=basicauth, headers=headers, verify=False)
37  if (resp.status_code >= 200 and resp.status_code <= 299):
38      print("STATUS OK: {}".format(resp.status_code))
39  else:
40      print("Error code {}, reply: {}".format(resp.status_code, resp.json()))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

STATUS OK: 201
PS C:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS> & C:\Users\azray\AppData\Local\Programs\Python\Python312\python.exe c:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS\LAB25_yazmin_part3.py
STATUS OK: 204
PS C:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS>

```

Cambiamos el request.put por un request.delete para eliminar la loopback99