

## RINCON ULLOA YAZMIN ELIZABETH

### Lab – CLI Automation with Python using netmiko

#### Objectives

- Part 1: Install the netmiko Python module
- Part 2: Connect to IOS XE's SSH service using netmiko
- Part 3: Use netmiko to gather information from the device
- Part 4: Use netmiko to alter configuration on the device

#### Background / Scenario

For simple network automation using a remote telnet or ssh based command line, network administrators have been using various screen scraping techniques for a long period of time. Initially the “expect” based scripts we utilized to automate entering commands when a specific expected string appeared on the command line. With the evolution of the Python language, the netmiko Python module has emerged as an open source project hosted and maintained on GitHub.com that provides a simple network automation interface using similar techniques like the “expect” based scripts.

In this lab activity, you will identify the potential but also the limitations of using netmiko to transport CLI commands for network automation.

#### Required Resources

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

#### Instructions

##### Part 1: Install the netmiko Python module

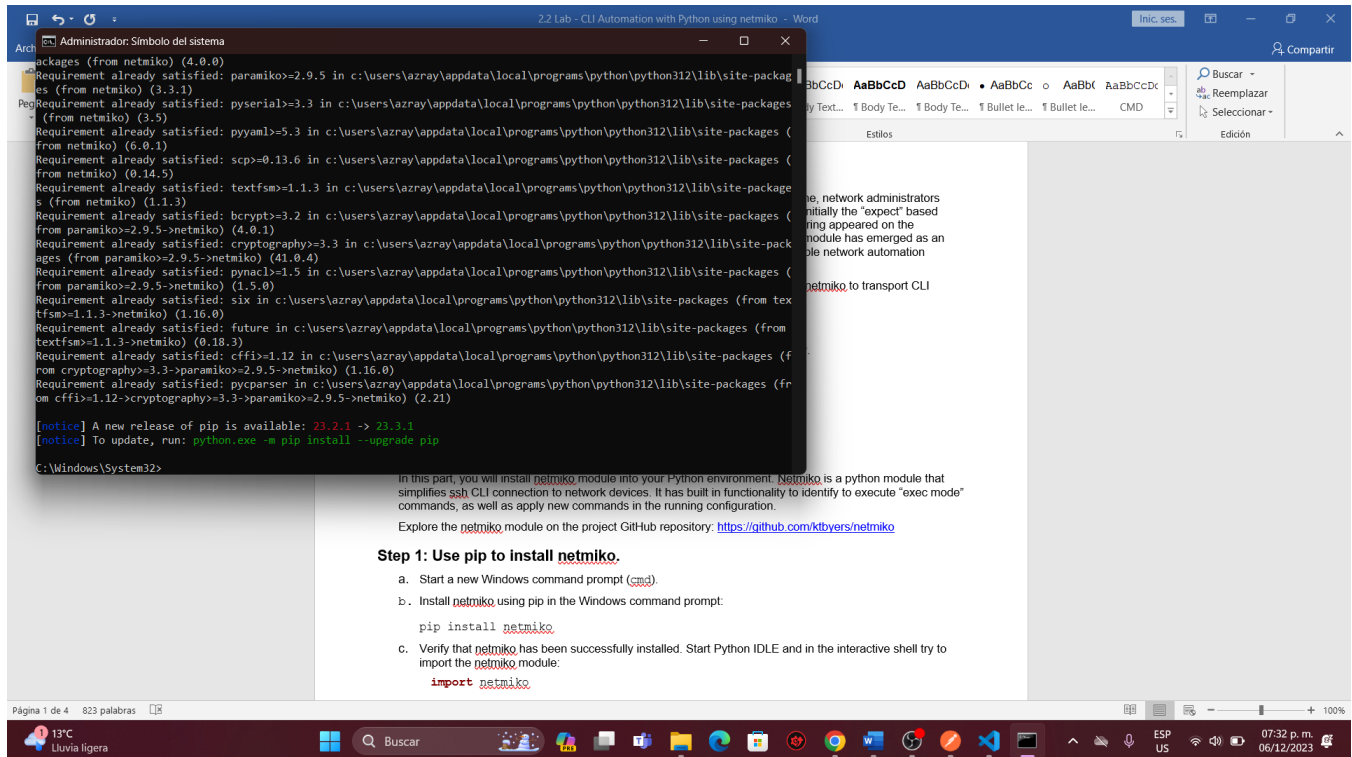
In this part, you will install netmiko module into your Python environment. Netmiko is a python module that simplifies ssh CLI connection to network devices. It has built in functionality to identify to execute “exec mode” commands, as well as apply new commands in the running configuration.

Explore the netmiko module on the project GitHub repository: <https://github.com/ktbyers/netmiko>

##### Step 1: Use pip to install netmiko.

- Start a new Windows command prompt (`cmd`).
- Install netmiko using pip in the Windows command prompt:

```
pip install netmiko
```



- c. Verify that netmiko has been successfully installed. Start Python IDLE and in the interactive shell try to import the netmiko module:

```
import netmiko
```

## Part 2: Connect to IOS XE's SSH service using netmiko

### Connect to IOS XE's SSH service using netmiko.

The netmiko module provides a “ConnectHandler()” function to setup the remote ssh connection. After a successful connection, the returned object represents the ssh cli connection to the remote device.

- a. In Python IDLE, create a new Python script file:
- b. In the new Python script file editor, import the “ConnectHandler()” function from the netmiko module:

```
from netmiko import ConnectHandler
```

- c. Setup a sshCli connection object using the ConnectHandler() function to the IOS XE device.

```
sshCli = ConnectHandler(  
    device_type='cisco_ios',  
    host='192.168.56.101',  
    port=22,  
    username='cisco',  
    password='cisco123!'  
)
```

The parameters of the `ConnectHandler()` function are:

- `device_type` – identifies the remote device type
- `host` – the address (host or IP) of the remote device (adjust the IP address “192.168.56.101” to match your router’s current address)
- `port` – the remote port of the ssh service
- `username` – remote ssh username (in this lab “cisco” for that was setup in the IOS XE VM)
- `password` – remote ssh password (in this lab “cisco123!” for that was setup in the IOS XE VM)

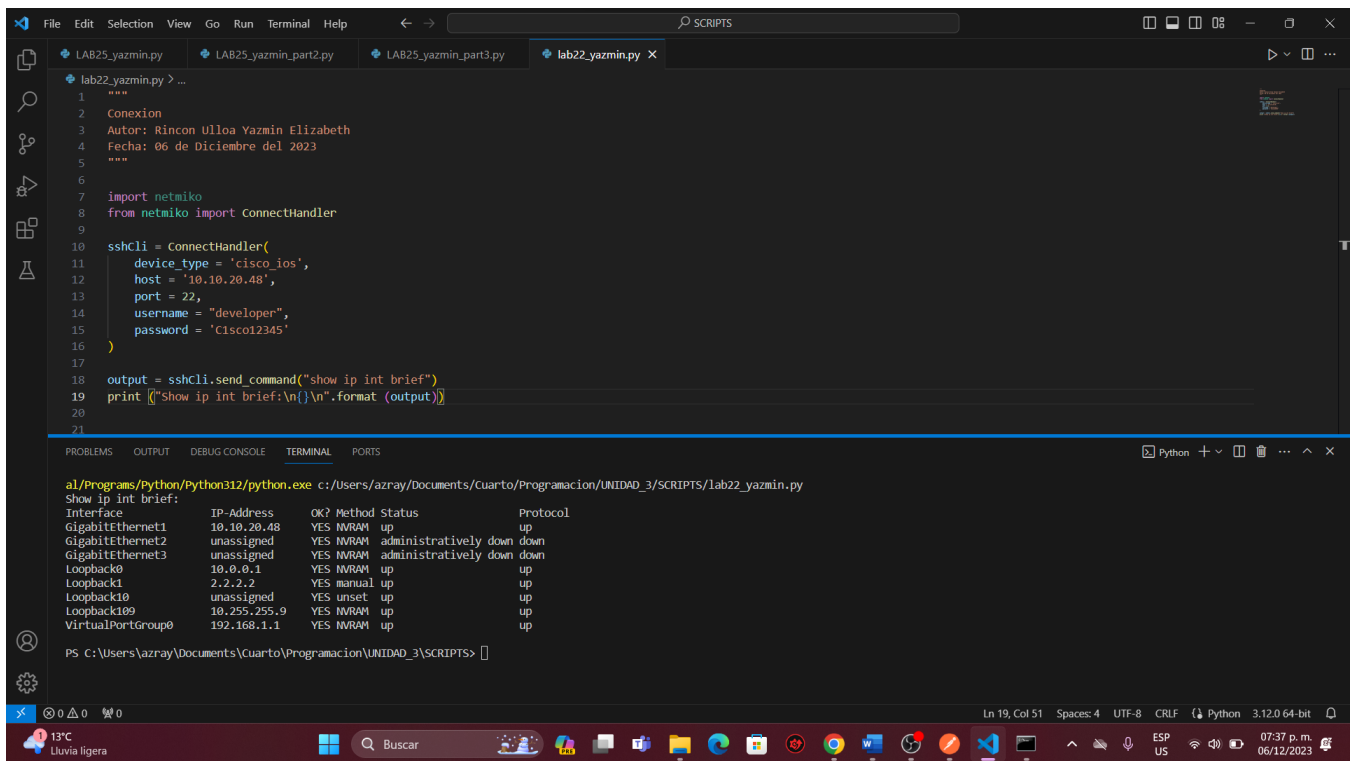
### Part 3: Use netmiko to gather information from the device

#### Send show commands and display the output

- a. Using the `sshCli` object, returned by the `ConnectHandler()` function that represents the ssh cli remote session, send some “show” command and print the output. Use the `send_command()` function of the `sshCli` object with a string parameter that represents the command you wish to execute in the exec mode:

```
output = sshCli.send_command("show ip int brief")
print("show ip int brief:\n{}\n".format(output))
```

- b. Execute the Python script file to see the results.



```
1  """
2  Conexion
3  Autor: Rincon Ulloa Yazmin Elizabeth
4  Fecha: 06 de Diciembre del 2023
5  """
6
7  import netmiko
8  from netmiko import ConnectHandler
9
10 sshCli = ConnectHandler(
11     device_type = 'cisco_ios',
12     host = '10.10.20.48',
13     port = 22,
14     username = "developer",
15     password = 'C1sc012345'
16 )
17
18 output = sshCli.send_command("show ip int brief")
19 print(f"Show ip int brief:\n{}\n".format(output))
20
21
```

```
al/Programs/Python/Python312/python.exe c:/Users/azray/Documents/Cuarto/Programacion/UNIDAD_3/SCRIPTS/lab22_yazmin.py
Show ip int brief:
Interface      IP-Address      OK? Method Status      Protocol
GigabitEthernet1  10.10.20.48    YES NVRAM  up          up
GigabitEthernet2  unassigned     YES NVRAM  administratively down down
GigabitEthernet3  unassigned     YES NVRAM  administratively down down
Loopback0       10.0.0.1       YES NVRAM  up          up
Loopback1       2.2.2.2       YES manual up          up
Loopback10      unassigned     YES unset  up          up
Loopback109     10.255.255.9   YES NVRAM  up          up
VirtualPortGroup0 192.168.1.1    YES NVRAM  up          up

PS c:\Users\azray\Documents\Cuarto\Programacion\UNIDAD_3\SCRIPTS>
```

If you have not saved the script file yet, you will be prompted to save it before it is executed.

- c. Verify the results:

```
>>>
RESTART: O:\tmp\Ch2_Files\Python Files with Solutions\lab 2.2 - CLI Automation
with Python using netmiko - sol.py
Sending 'sh ip int brief'.
IP interface status and configuration:
Interface      IP-Address      OK? Method Status      Protocol
GigabitEthernet1  192.168.56.101 YES DHCP    up          up
>>>
```

- d. Verify the data type of the “output” variable. How would you extract the IP address and the Interface Name into variables? What if there were multiple interfaces?

## Part 4: Use netmiko to alter configuration on the device

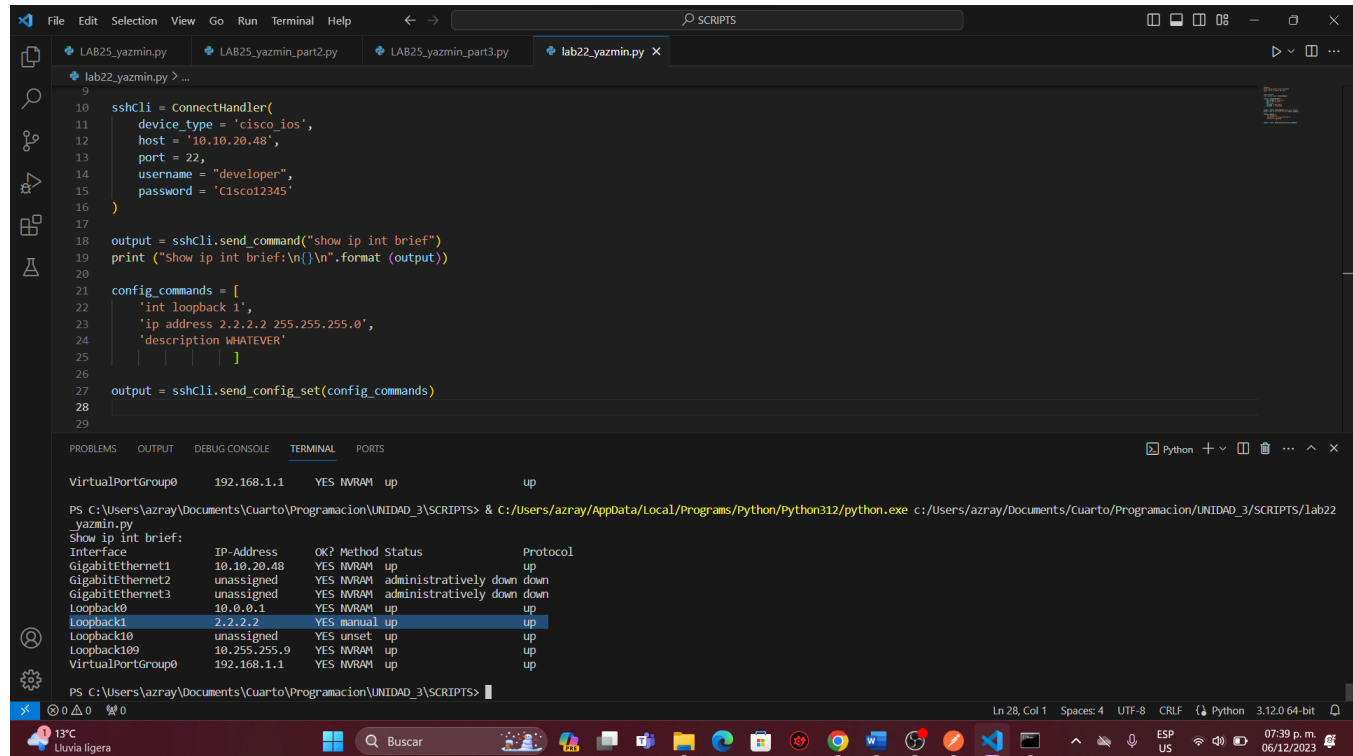
In the following steps, you will alter the configuration of the device by creating new loopback interfaces.

### Create a new loopback interface

Using the `sshCli` object, returned by the `ConnectHandler()` function that represents the ssh cli remote session, send some configuration command and print the output. Use the `send_config_set()` function of the `sshCli` object with a list parameter including the configuration commands as strings you wish to execute in the exec mode:

```
config_commands = [  
    'int loopback 1',  
    'ip address 2.2.2.2 255.255.255.0',  
    'description WHATEVER'  
]  
  
output = sshCli.send_config_set(config_commands)
```

Execute the Python script file and verify the results



```
9  
10 sshcli = ConnectHandler(  
11     device_type = 'cisco_ios',  
12     host = '10.10.20.48',  
13     port = 22,  
14     username = "developer",  
15     password = 'Cisco12345'  
16 )  
17  
18 output = sshcli.send_command("show ip int brief")  
19 print ("Show ip int brief:\n{}\n".format (output))  
20  
21 config_commands = [  
22     'int loopback 1',  
23     'ip address 2.2.2.2 255.255.255.0',  
24     'description WHATEVER'  
25 ]  
26  
27 output = sshcli.send_config_set(config_commands)  
28  
29
```

VirtualPortGroup0 192.168.1.1 YES NVRAM up up

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/lab22\_yazmin.py

Show ip int brief:

Interface	IP-Address	OK?	Method	Status	Protocol
GigabitEthernet1	10.10.20.48	YES	NVRAM	up	up
GigabitEthernet2	unassigned	YES	NVRAM	administratively down	down
GigabitEthernet3	unassigned	YES	NVRAM	administratively down	down
Loopback0	10.0.0.1	YES	NVRAM	up	up
Loopback1	2.2.2.2	YES	manual	up	up
Loopback10	unassigned	YES	unset	up	up
Loopback109	10.255.255.9	YES	NVRAM	up	up
VirtualPortGroup0	192.168.1.1	YES	NVRAM	up	up

PS C:\Users\azray\Documents\Cuarto\Programacion\UNIDAD\_3\SCRIPTS>

Why does the output from “show ip int brief” not include the “loopback1” interface?

Aparece la loopback1 creada

How to execute and display the output from the “show ip int brief” command after the loopback interfaces was created?

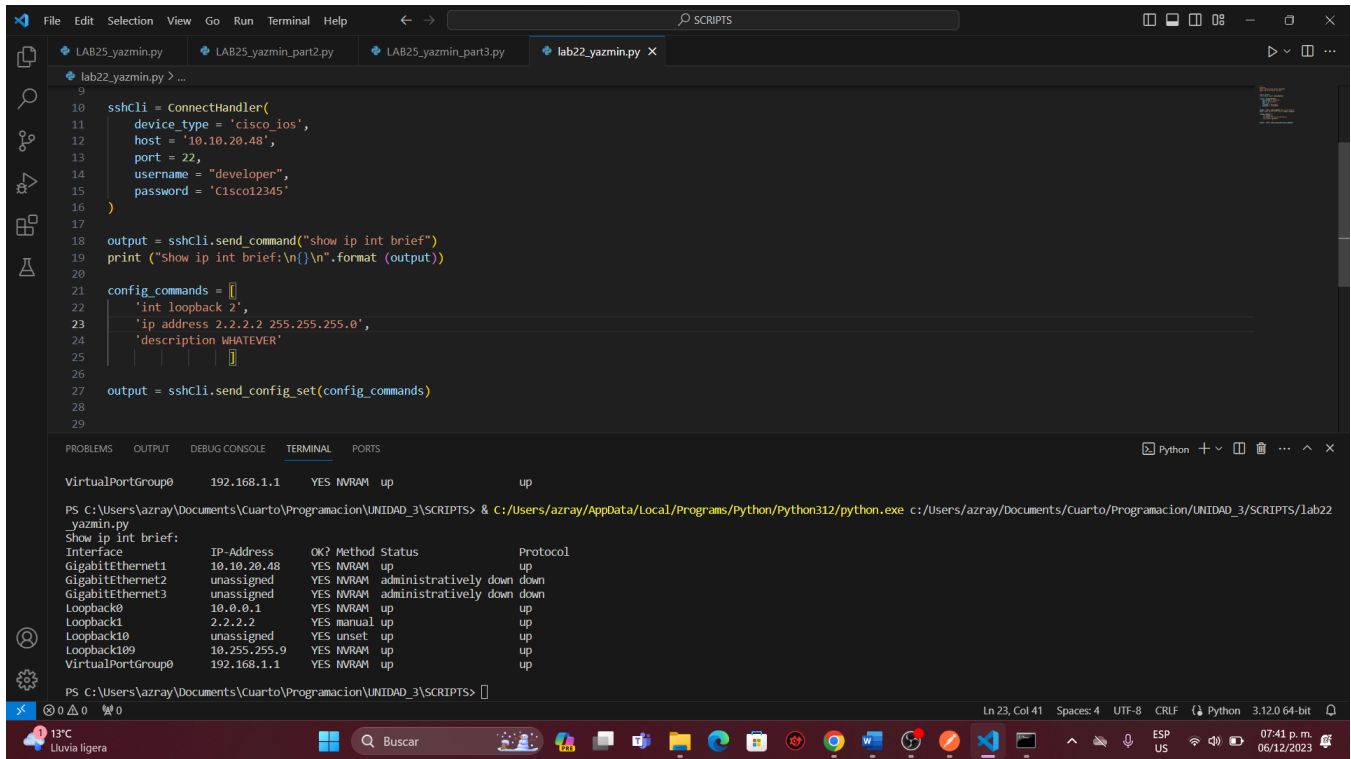
Podria pasar porque la interfaz loopback no existen debido a que ejecutar el commando show ip int brief muestra todas las interfaces existentes.

Add code to create a new loopback interface (loopback2) with the same IP address as on the existing loopback interface, only with a different description.

Execute the Python script file and verify the results.

Was the new loopback2 interface successfully created?

## Lab – CLI Automation with Python using netmiko



The screenshot shows a Visual Studio Code editor with a Python script named `lab22_yazmin.py` and its terminal output. The script uses the `netmiko` library to connect to a Cisco device and send commands.

```
9
10 sshcli = ConnectHandler(
11     device_type = 'cisco_ios',
12     host = '10.10.20.48',
13     port = 22,
14     username = "developer",
15     password = 'Cisc012345'
16 )
17
18 output = sshcli.send_command("show ip int brief")
19 print ("show ip int brief:\n{}\n".format (output))
20
21 config_commands = [
22     'int loopback 2',
23     'ip address 2.2.2.2 255.255.255.0',
24     'description WHATEVER'
25 ]
26
27 output = sshcli.send_config_set(config_commands)
28
29
```

The terminal output shows the command `show ip int brief` being executed on a Cisco device. The output is as follows:

```
VirtualPortGroup0 192.168.1.1 YES NVRAM up up
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/lab22_yazmin.py
Show ip int brief:
Interface IP-Address OK? Method Status Protocol
GigabitEthernet1 10.10.20.48 YES NVRAM up up
GigabitEthernet2 unassigned YES NVRAM administratively down down
GigabitEthernet3 unassigned YES NVRAM administratively down down
Loopback0 10.0.0.1 YES NVRAM up up
Loopback1 2.2.2.2 YES manual up up
Loopback100 unassigned YES unset up up
Loopback109 10.255.255.9 YES NVRAM up up
VirtualPortGroup0 192.168.1.1 YES NVRAM up up
PS C:\Users\azray\Documents\cuarto\Programacion\UNIDAD_3\SCRIPTS>
```

No debido a que las 2 loopback tendrian la misma direccion

Was the new configuration change accepted, partially accepted or rejected?

Se rechaza