

## Lab – CLI Automation with Python using netmiko

Nombre:	Montserrat Orduña Basaldua
Num Control:	1222100895

### 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.

- a. Start a new Windows command prompt (`cmd`).
- b. 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))
```

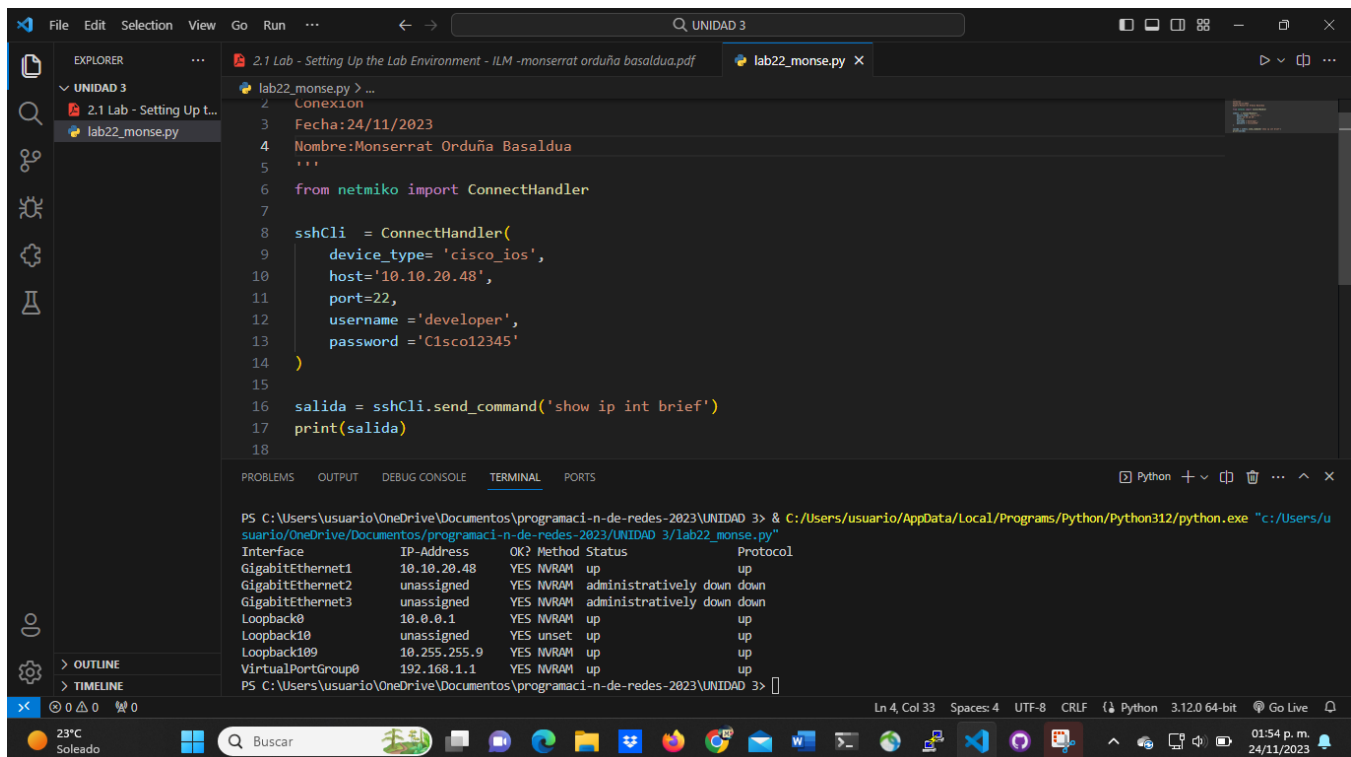
- b. Execute the Python script file to see the results.

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?



The screenshot shows a Python script named `lab22_monse.py` in Visual Studio Code. The script uses `netmiko` to connect to a Cisco device and run the `show ip int brief` command. The terminal output shows the command execution and the resulting interface status table.

```
2.1 Lab - Setting Up the Lab Environment - ILM - monserrat orduña basaldua.pdf
lab22_monse.py x
lab22_monse.py > ...
1 Conexión
2 Fecha: 24/11/2023
3 Nombre: Monserrat Orduña Basaldua
4
5
6 from netmiko import ConnectHandler
7
8 sshCli = ConnectHandler(
9     device_type='cisco_ios',
10    host='10.10.20.48',
11    port=22,
12    username='developer',
13    password='Cisco12345'
14)
15
16 salida = sshCli.send_command('show ip int brief')
17 print(salida)
18
```

Terminal Output:

```
PS C:\Users\usuario\OneDrive\Documentos\programaci-n-de-redes-2023\UNIDAD 3> & C:\Users\usuario\AppData\Local\Programs\Python\Python312\python.exe "c:/Users/usuario/OneDrive/Documentos/programaci-n-de-redes-2023/UNIDAD 3/lab22_monse.py"
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
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\usuario\OneDrive\Documentos\programaci-n-de-redes-2023\UNIDAD 3>
```

## 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.

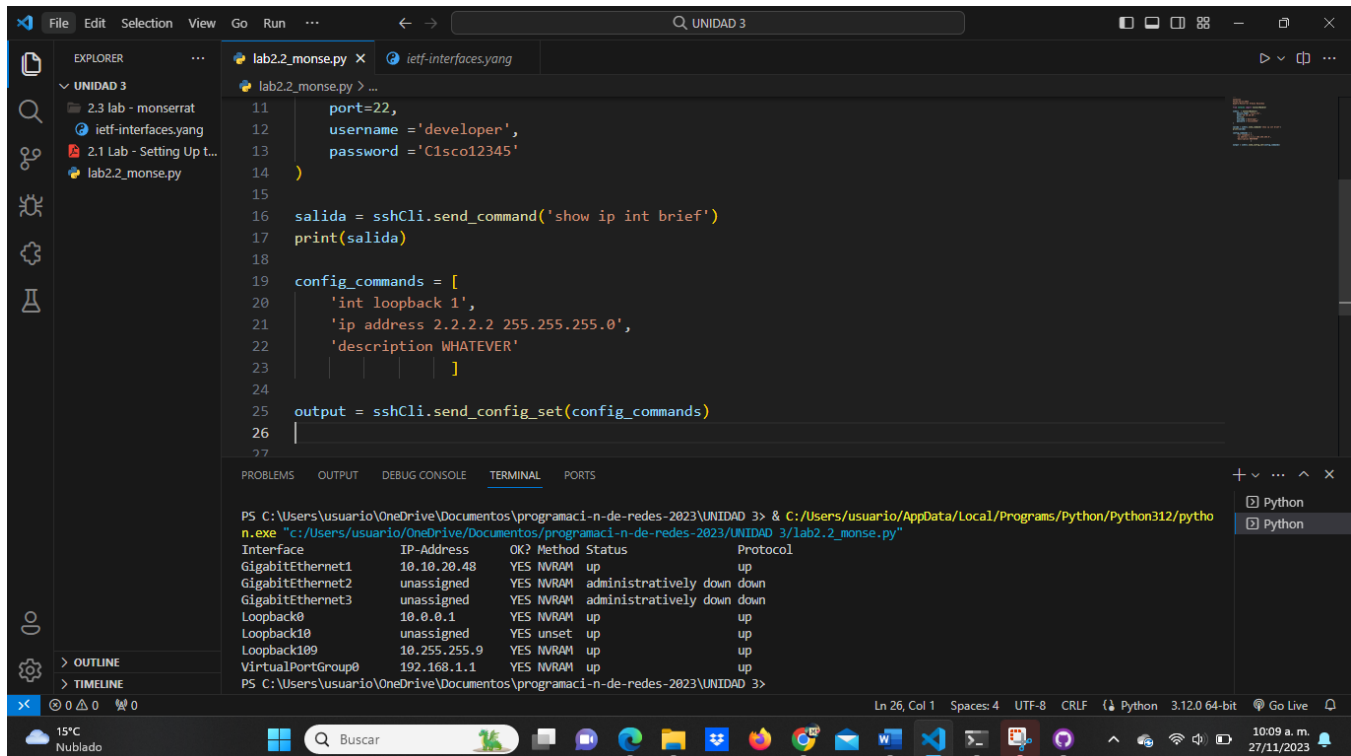
### Create a new loopback interface

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

## Lab – CLI Automation with Python using netmiko

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)
```

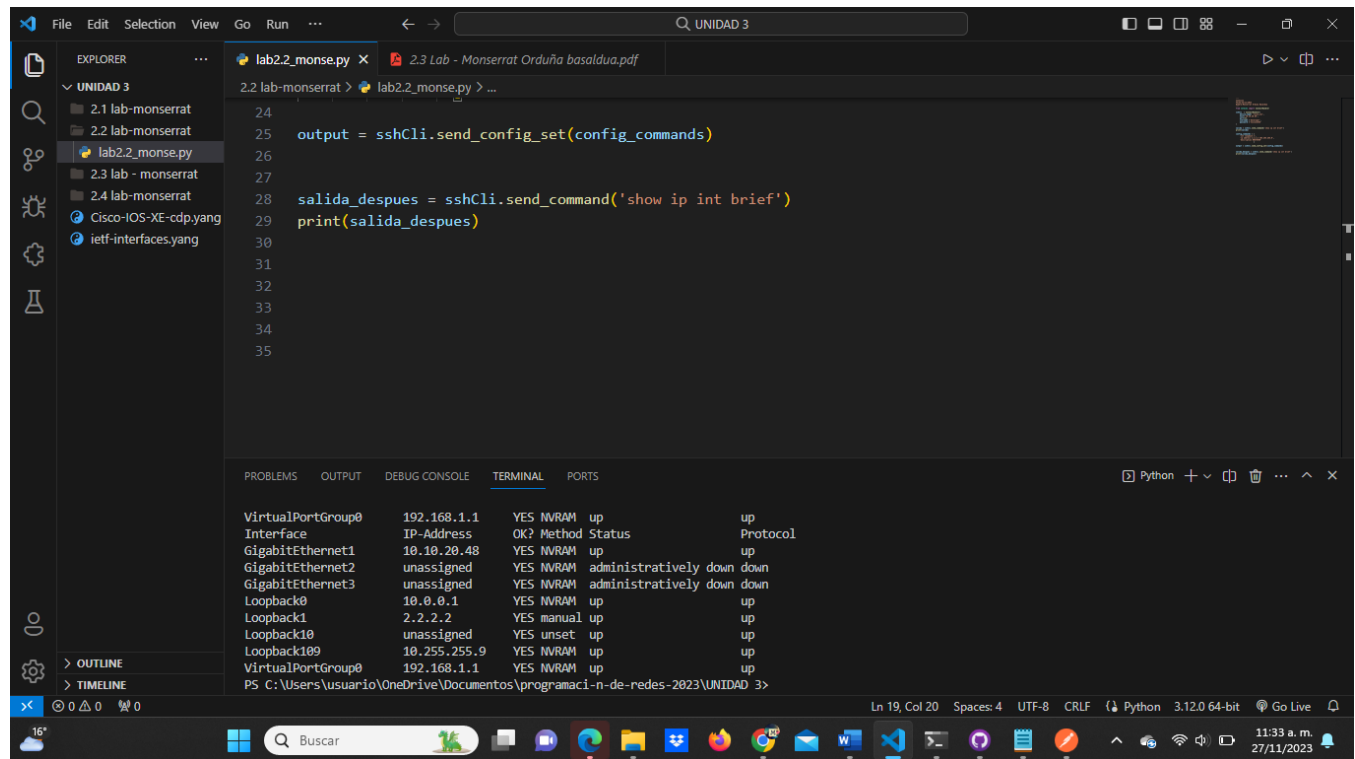


Execute the Python script file and verify the results

Why does the output from “show ip int brief” not include the “loopback1” interface? no se incluye en la salida de "show ip int brief" porque el código Python solo ejecuta el comando show ip int brief antes de crear la interfaz de loopback. En el momento de ejecutar ese comando, la interfaz "loopback1" aún no ha sido creada.

How to execute and display the output from the “show ip int brief” command after the loopback interfaces was created? Con el siguiente commando se puede mostrar la salida de la loopback1

## Lab – CLI Automation with Python using netmiko



The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'UNIDAD 3' with several files, including 'lab2.2\_monse.py'. The main editor window displays the contents of 'lab2.2\_monse.py', which is a Python script using Netmiko to connect to a Cisco device and execute commands. The terminal window shows the output of the script, which is a table of interface information.

```
24
25 output = sshCli.send_config_set(config_commands)
26
27
28 salida_despues = sshCli.send_command('show ip int brief')
29 print(salida_despues)
30
31
32
33
34
35
```

Interface	IP-Address	Method	Status	Protocol
VirtualPortGroup0	192.168.1.1	YES	NVRAM	up
Interface		OK?	Method	Status
GigabitEthernet1	10.10.20.48	YES	NVRAM	up
GigabitEthernet2	unassigned	YES	NVRAM	administratively down
GigabitEthernet3	unassigned	YES	NVRAM	administratively down
Loopback0	10.0.0.1	YES	NVRAM	up
Loopback1	2.2.2.2	YES	manual	up
Loopback10	unassigned	YES	unset	up
Loopback100	10.255.255.9	YES	NVRAM	up
VirtualPortGroup0	192.168.1.1	YES	NVRAM	up

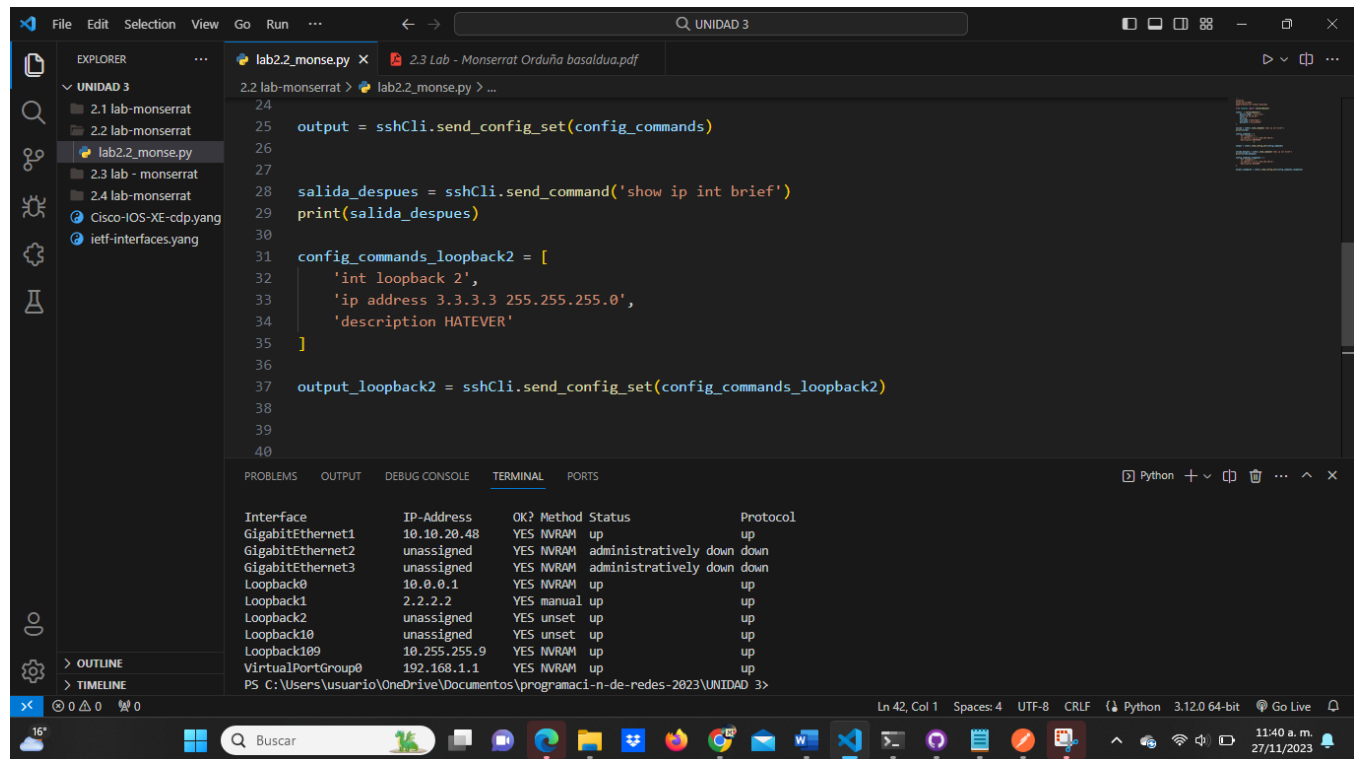
PS C:\Users\usuario\OneDrive\Documentos\programaci-n-de-redes-2023\UNIDAD 3>

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.

Con el siguiente comando se crea la loopback2

## Lab – CLI Automation with Python using netmiko



The screenshot shows a VS Code editor with a Python script named `lab2.2_monse.py` and its output in the terminal. The script uses `netmiko` to connect to a Cisco switch and configure a new loopback interface.

```
24
25 output = sshCli.send_config_set(config_commands)
26
27
28 salida_despues = sshCli.send_command('show ip int brief')
29 print(salida_despues)
30
31 config_commands_loopback2 = [
32     'int loopback 2',
33     'ip address 3.3.3.3 255.255.255.0',
34     'description HATEVER'
35 ]
36
37 output_loopback2 = sshCli.send_config_set(config_commands_loopback2)
38
39
40
```

The terminal output shows the result of the `show ip int brief` command:

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
Loopback2	unassigned	YES	unset	up	up
Loopback10	unassigned	YES	unset	up	up
Loopback100	10.255.255.9	YES	NVRAM	up	up
VirtualPortGroup0	192.168.1.1	YES	NVRAM	up	up

Was the new loopback2 interface successfully created?

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