Is it possible to make a script in windows with python code for automare the following actions,

1. Connect to NEs using MobaXterm software, specified NE name, port 22, username, & password,
2. Execute this command in each NE "Display version",
3. Save the logs

Yes, it is possible to automate these actions in Windows using Python. To connect to the network elements using MobaXterm, you can use a library such as Paramiko that provides an implementation of the SSHv2 protocol. To execute the command "Display version", you can use Paramiko's exec_command method and to save the logs you can write the output of the command to a file. Here is an example code to get you started:

```python
import paramiko

# Connect to the network element
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect("hostname", port=22, username="username", password="password")

# Execute the command
stdin, stdout, stderr = ssh.exec_command("Display version")

# Save the logs
with open("logs.txt", "w") as f:
    f.write(stdout.read().decode())

# Close the connection
ssh.close()
```

How to make it for multiple NE, read from NElist.txt for the NEs detail, then output different files according to it hostname?

You can modify the code to read the list of network elements from a text file, "**NElist.txt**", and create a separate log file for each network element based on its hostname. Here is an example code:

```python
import paramiko

# Read the list of network elements from the text file
with open("NElist.txt", "r") as f:
    lines = f.readlines()

# Connect to each network element
for line in lines:
    hostname, port, username, password = line.strip().split(",")
    port = int(port)

    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(hostname, port=port, username=username, password=password)

    # Execute the command
    stdin, stdout, stderr = ssh.exec_command("Display version")

    # Save the logs to a separate file based on the hostname
    with open(f"{hostname}_logs.txt", "w") as f:
        f.write(stdout.read().decode())

    # Close the connection
    ssh.close()
```

In this example, "**NElist.txt**" should contain one line for each network element, with the details separated by a comma (e.g. "**hostname,22,username,password**").

What are the requirements to use the above coding?

In order to use the code I provided above, you will need to have the following:

1. Python installed on your Windows machine. You can download it from the official Python website (https://www.python.org/downloads/).
2. The Paramiko library installed. You can install it using "**pip**" by running the following command in your terminal/command prompt:

```
pip install paramiko
```

3. MobaXterm software installed on your Windows machine and the network elements you want to connect to should be configured and accessible through MobaXterm.
4. A text file "**NElist.txt**" containing the details of the network elements you want to connect to, with one line for each network element and the details separated by a comma (e.g. "**hostname,22,username,password**").

With these requirements satisfied, you should be able to run the code and automate the actions you described in your original question.

What can cause the above coding execution stuck or running endless?

Here are some common reasons why the code might get stuck or run endlessly:

1. Incorrect credentials: If the username or password provided in "**NElist.txt**" is incorrect, the code will not be able to establish a connection to the network element and will hang indefinitely.
2. Unreachable host: If the hostname provided in "**NElist.txt**" is incorrect or the network element is not accessible, the code will hang indefinitely while trying to establish a connection.
3. Connection timeout: If the network connection is slow or unstable, the code may hang indefinitely while trying to establish a connection or receive data from the network element.
4. Incorrect command: If the command executed in each network element is incorrect or not supported by the network element, it may hang indefinitely or produce an error that the code does not handle properly.
5. Infinite loop: If the code contains an infinite loop, for example due to an incorrect condition in a loop, it will run endlessly.

To avoid these issues, you can add error handling and logging to the code to catch and diagnose problems if they occur. For example, you can catch exceptions thrown by Paramiko and log the error message to help diagnose the issue. You can also add a timeout to the connection to prevent the code from hanging indefinitely.

Can write me a new code that can solved the above issues?

Sure, here's a code that adds error handling and logging to the previous code to solve the issues mentioned:

```python
import paramiko
```