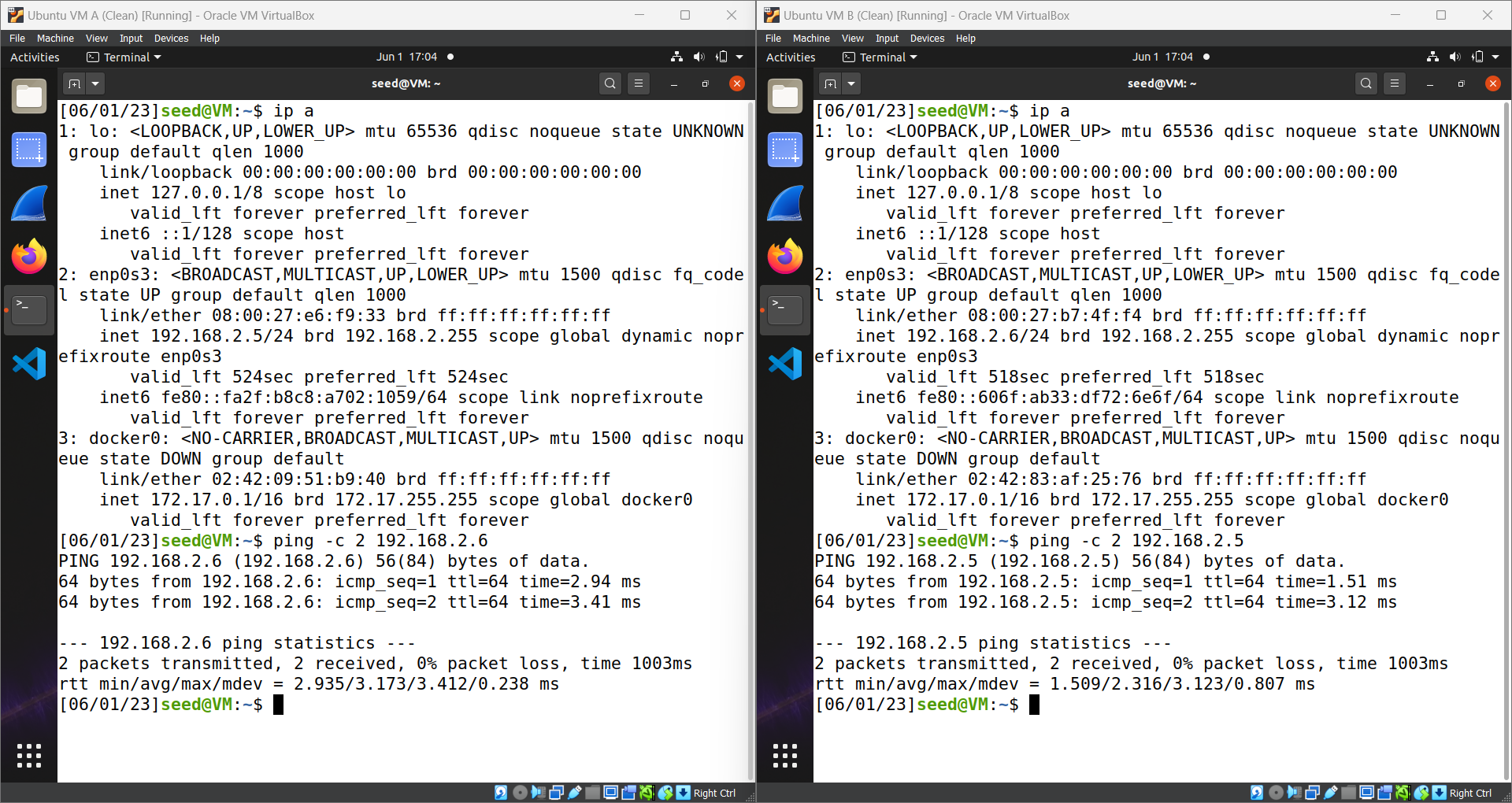
# Connection

Both VM can reach each other.



# Introduction of Scapy

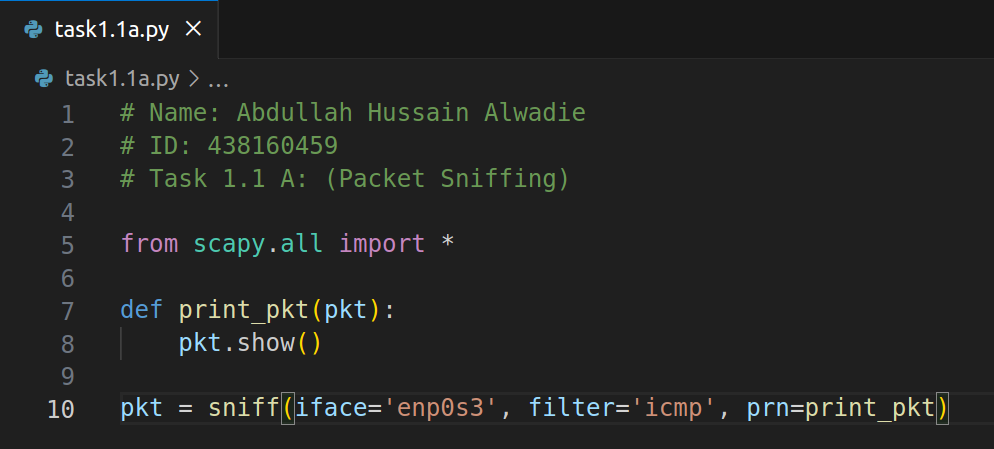
Here, in this code we imported all the content from scapy library and then created an instance of IP Object and then displayed then instance used show method (available with IP Object). We can in the result that IP Object contains all IP Protocol Headers.

A screenshot of a computer

Description automatically generated

# Task 1.1 A

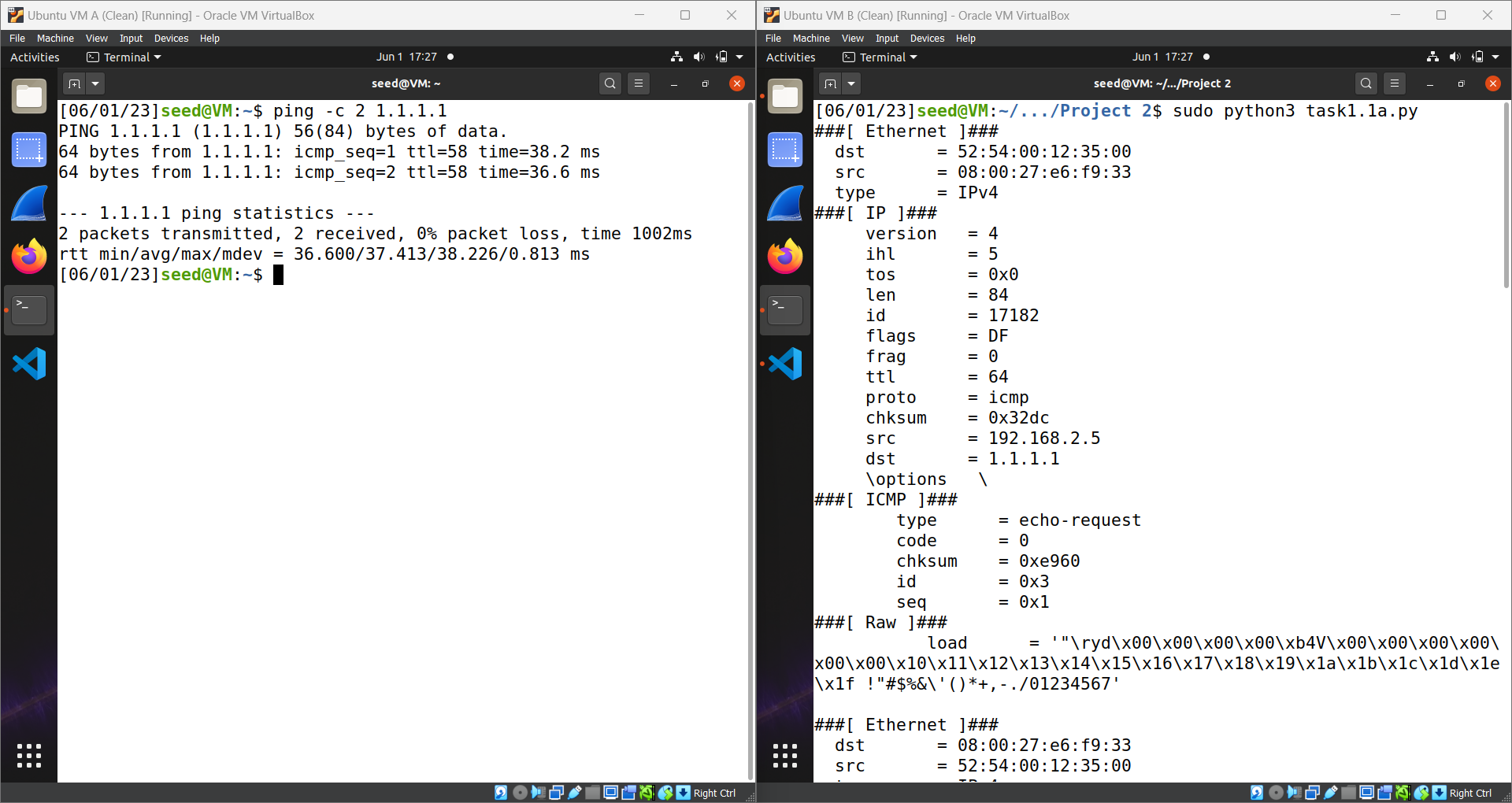
## Code



## Explanation

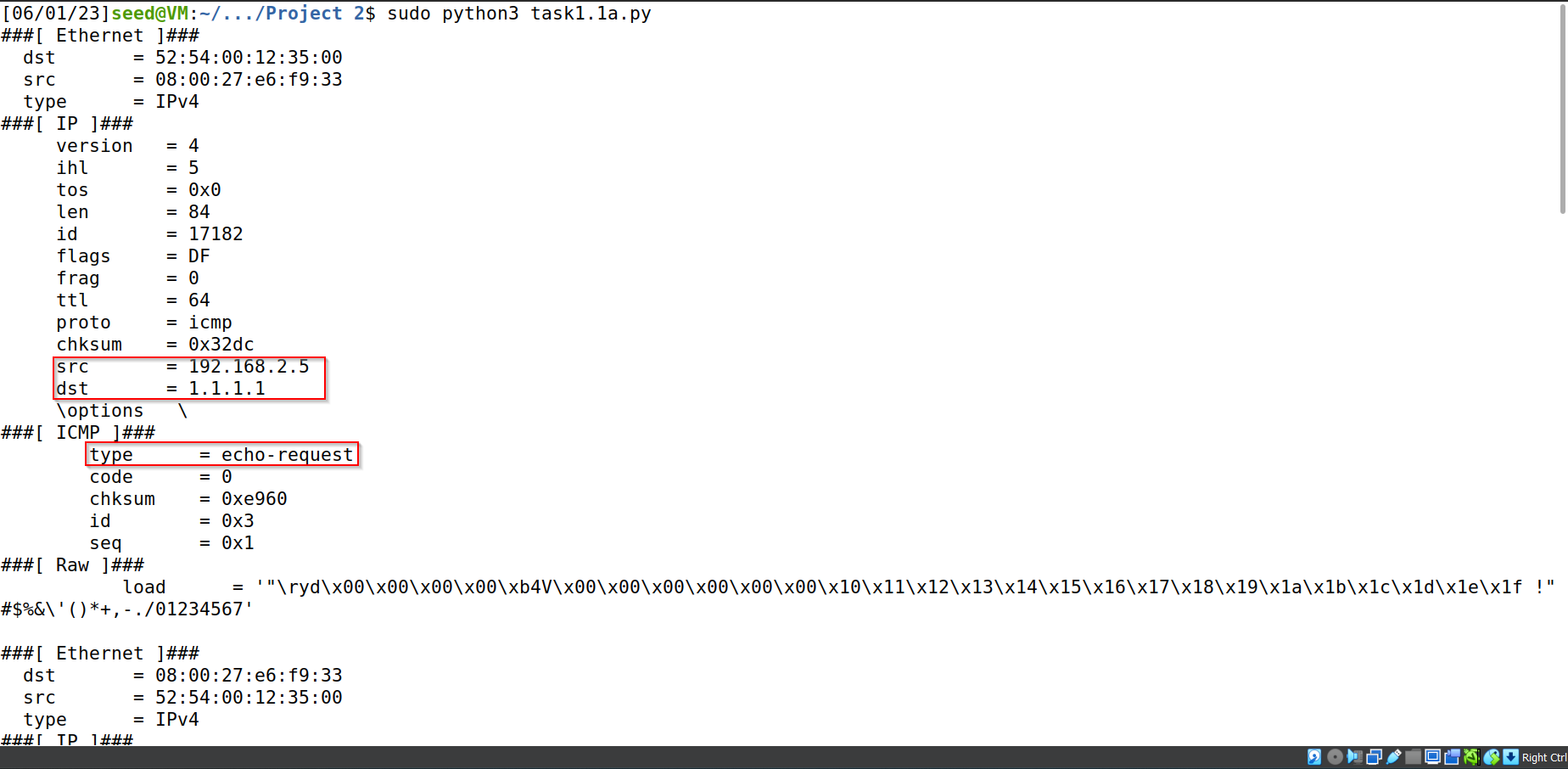
Here, in this code we are using the sniff function of scapy which does exactly what it is name is. The first argument, **iface**, selects network interface from where to sniff traffic. The second argument, **filter**, filters packet (in this case it filters only **ICMP** packets). The third argument, **prn**, receives a callback function and gets triggered when a packet is received after filtration.

## With Root Privileges



## Sniffing Results

### Echo request

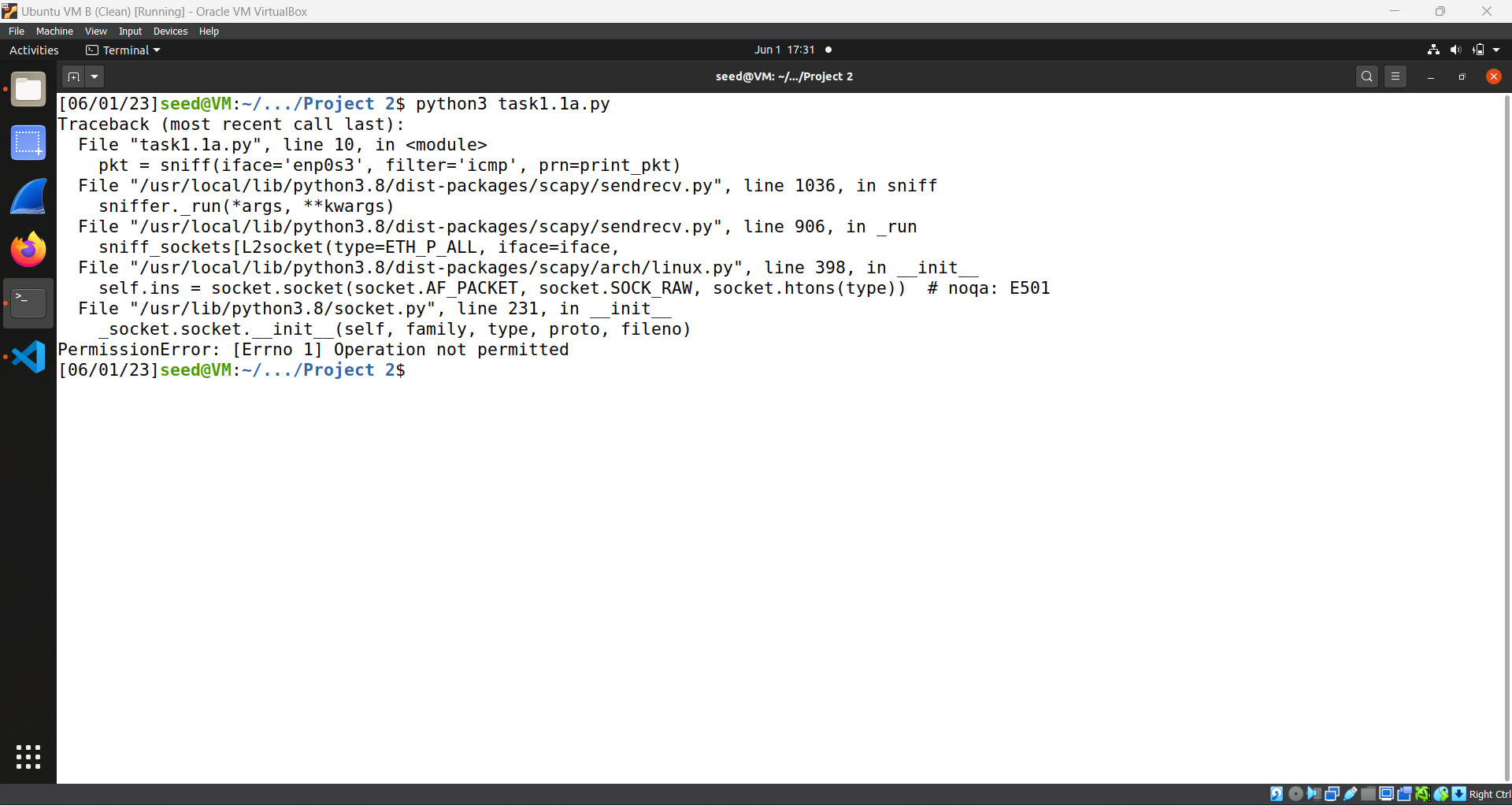


### Echo reply

A picture containing text, screenshot, font, number

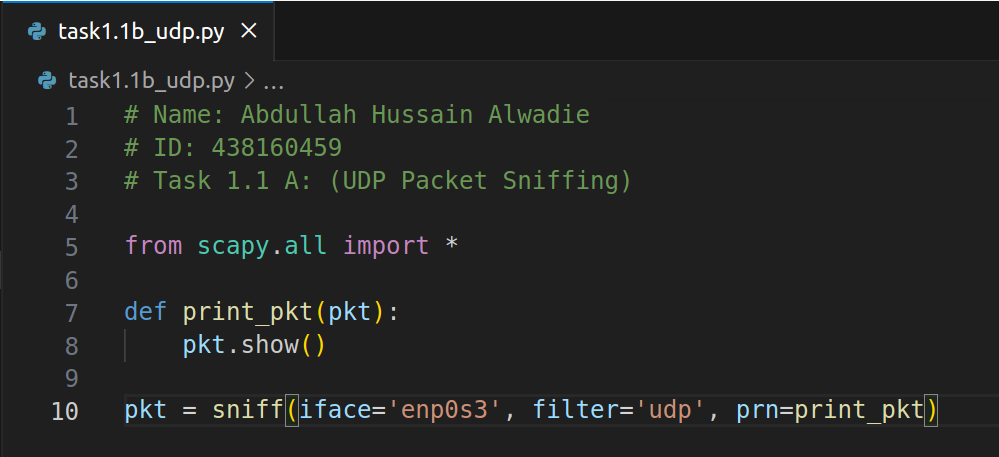
Description automatically generated

## Without Root Privileges

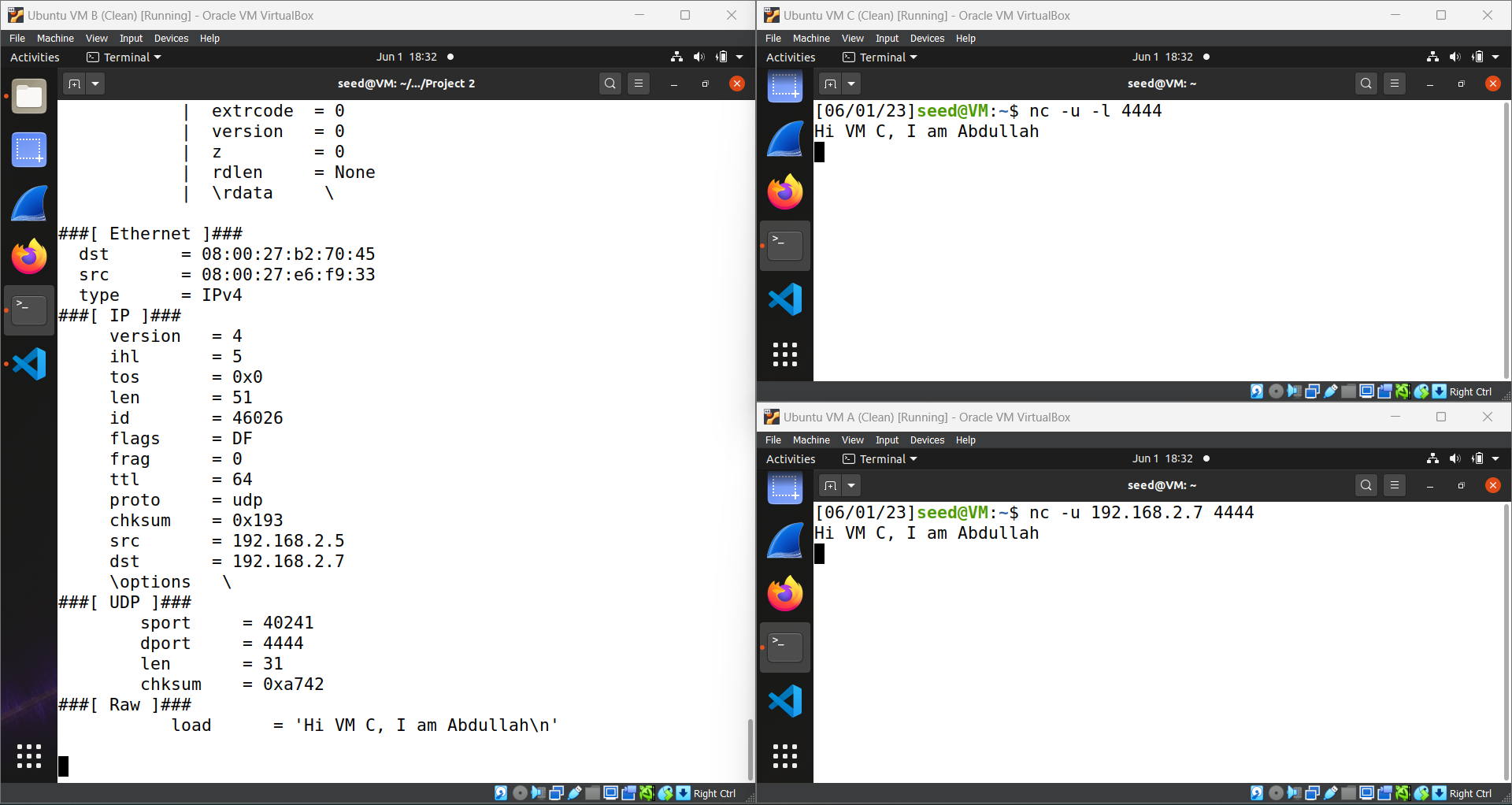


# Task 1.1 B (UDP)

## Code

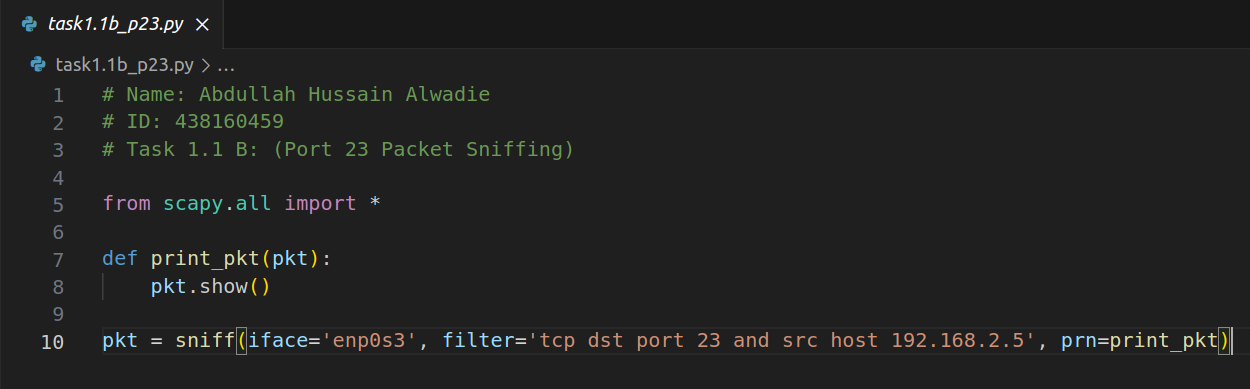


## Output

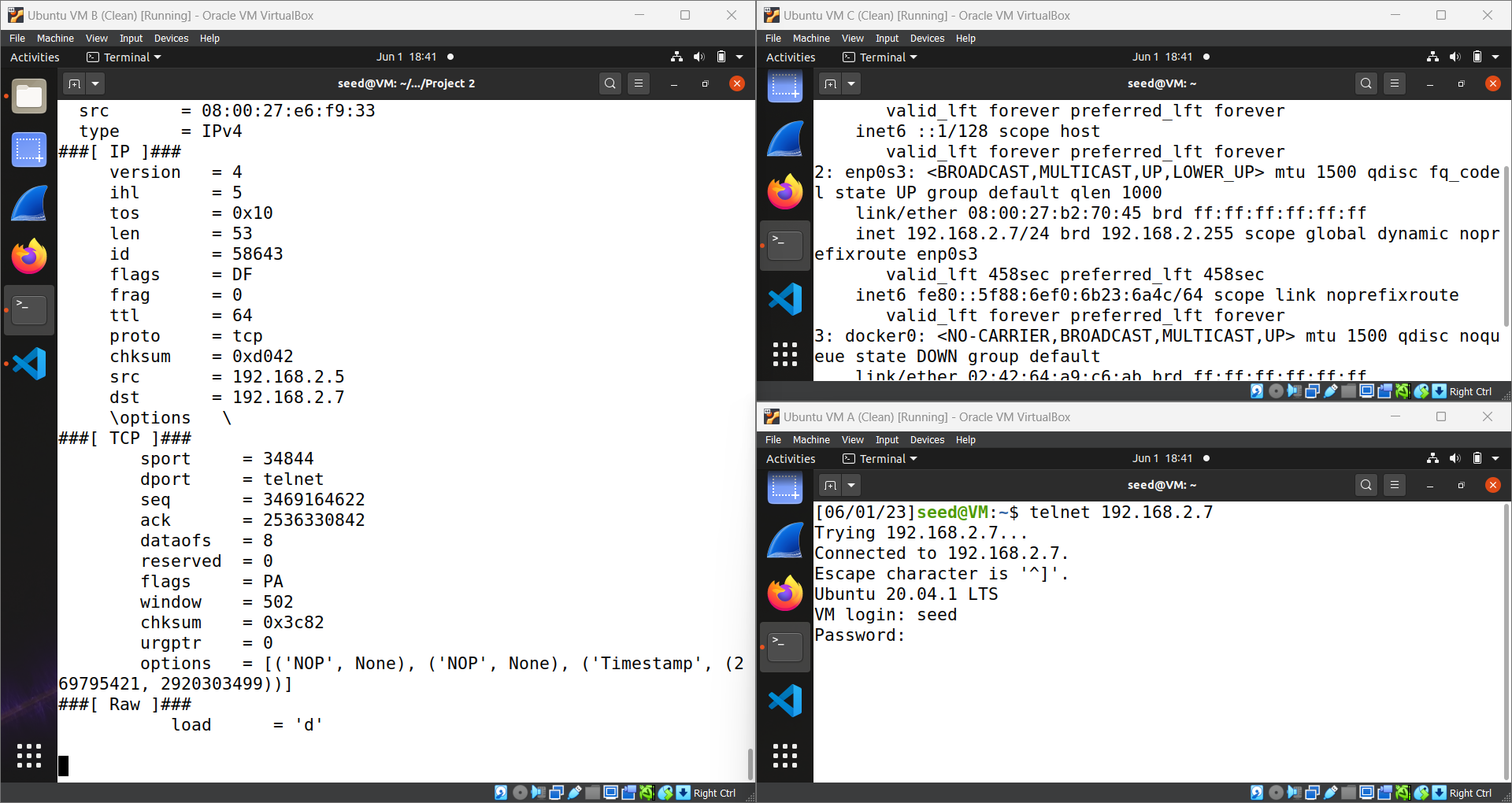


# Task 1.1 B (Port 23)

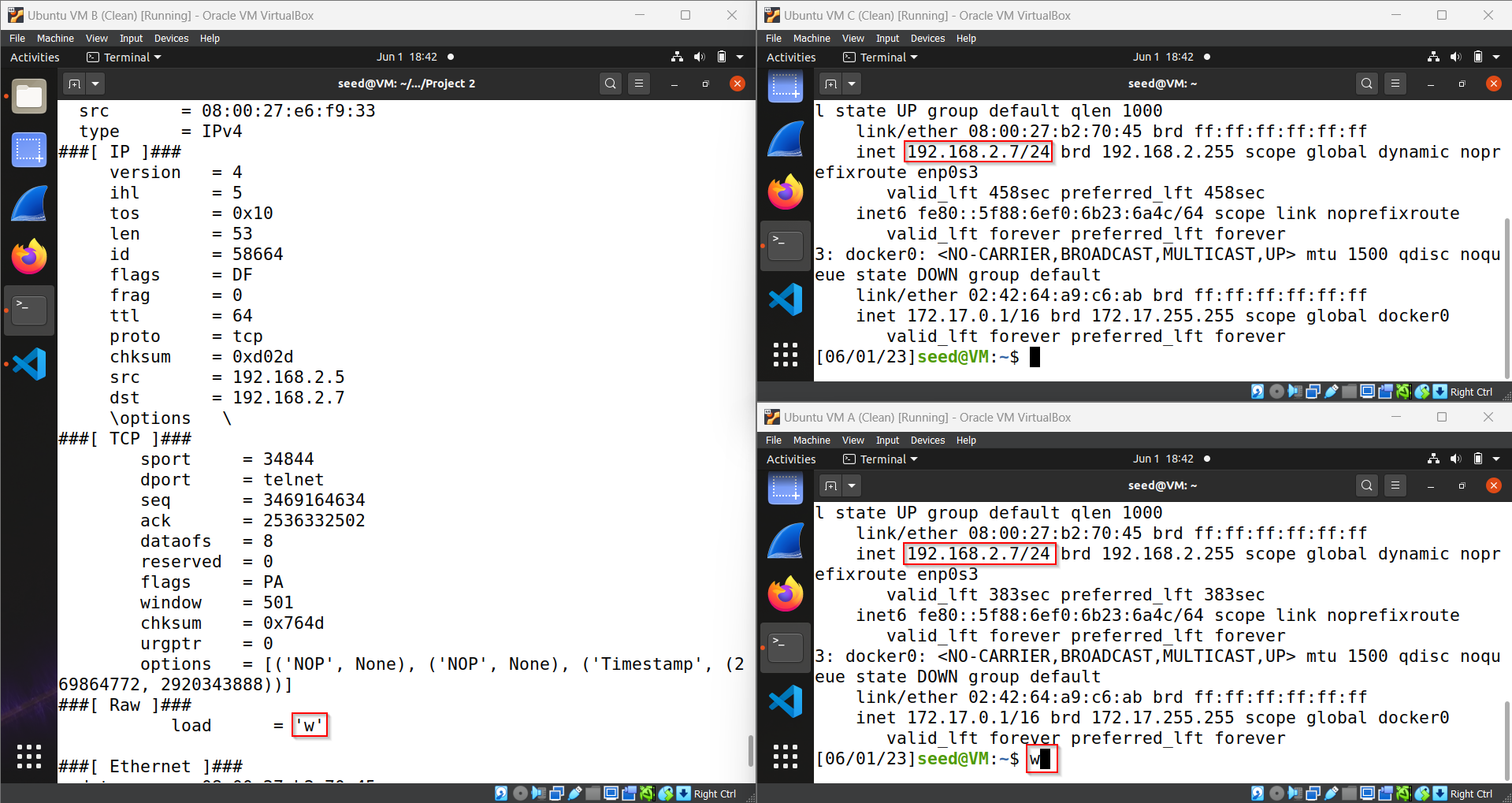
## Code



## Output



## Output After Login



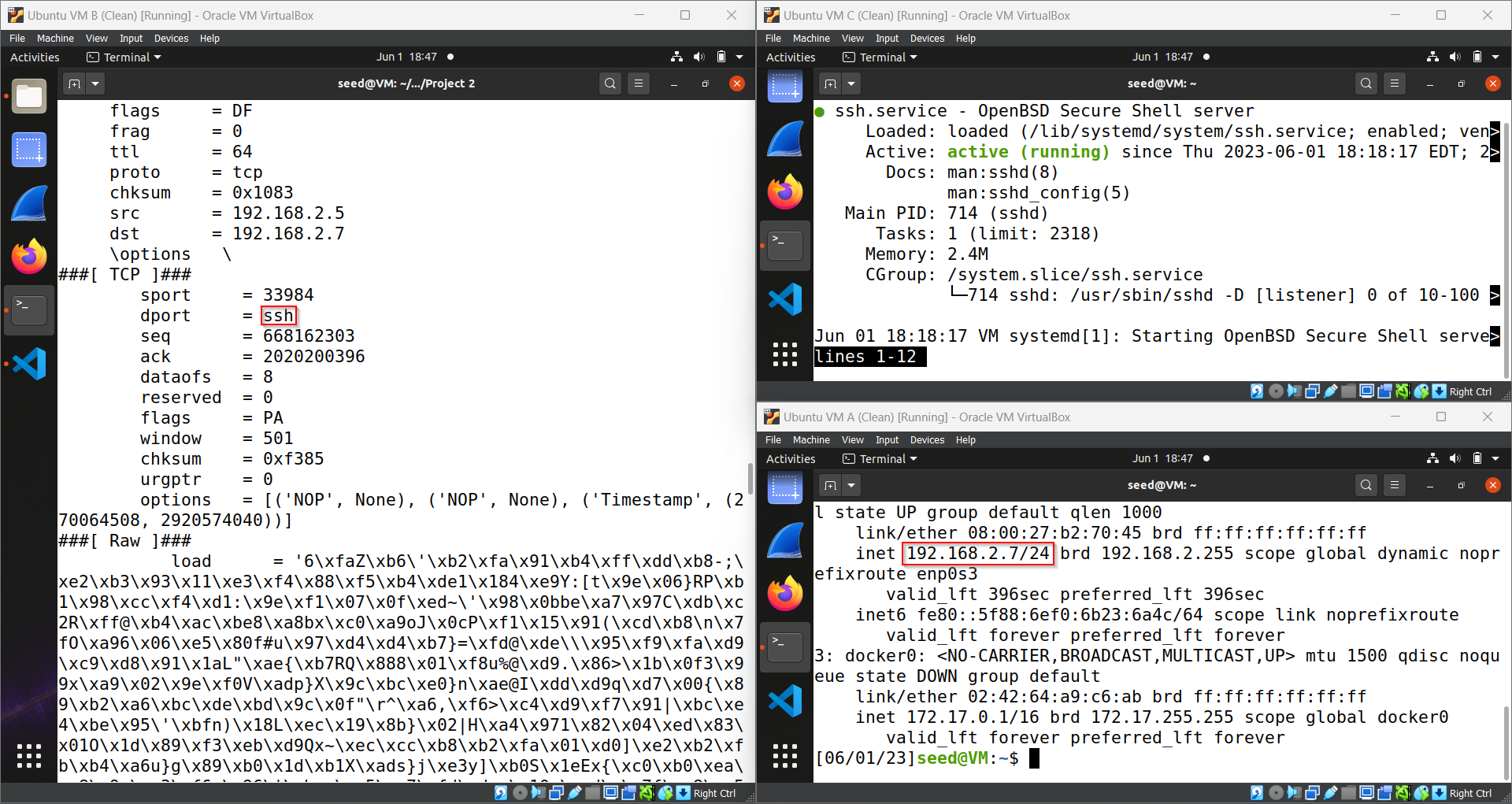
# Task 1.1 B (Port 22)

## Code

A screen shot of a computer

Description automatically generated with medium confidence

## Output



# Difference between port 22 and port 23

In port 22 payload is encrypted whereas in port 23 payload is in plaintext. In Telnet, a character (byte) is transmitted at a time whereas in multiple bytes is transmitted at a time.

# Task 1.1 B (Sending traffic Outside the Network)

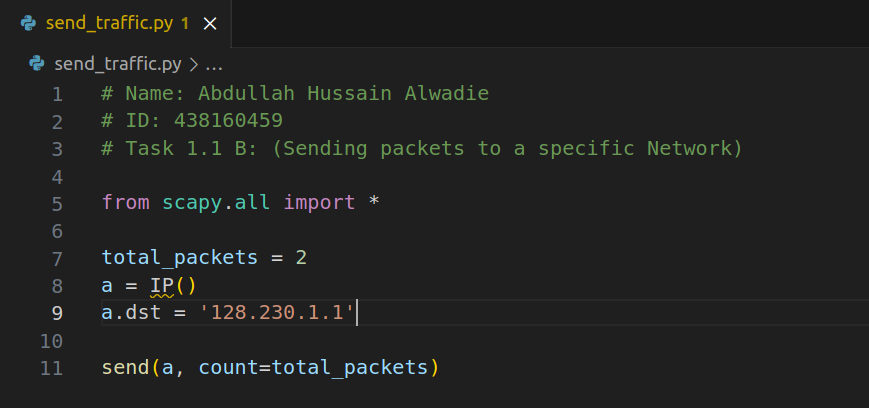
## Sniffing Code

A picture containing text, screenshot, software, multimedia software

Description automatically generated

## Triggering Code

Send two packets with no data at TCP layer, 20 bytes of 00 will be sent.



## Output

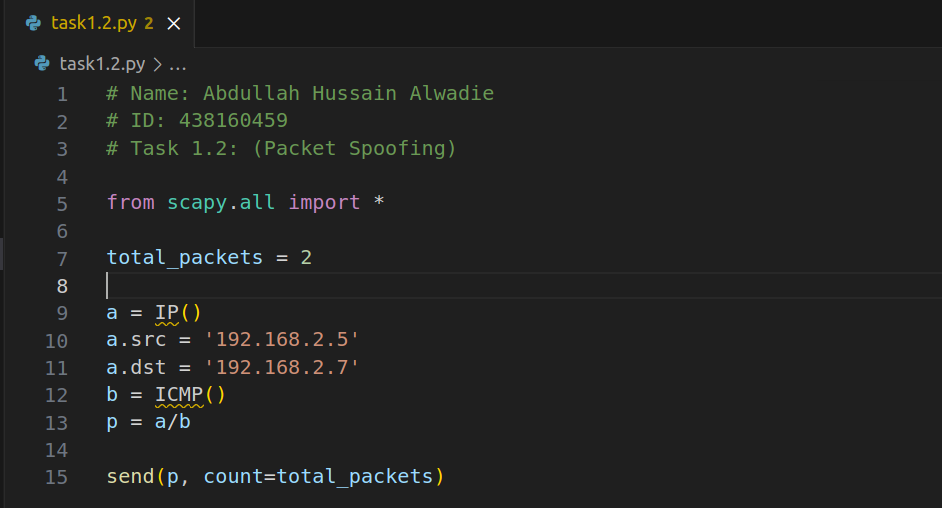
A screenshot of a computer

Description automatically generated

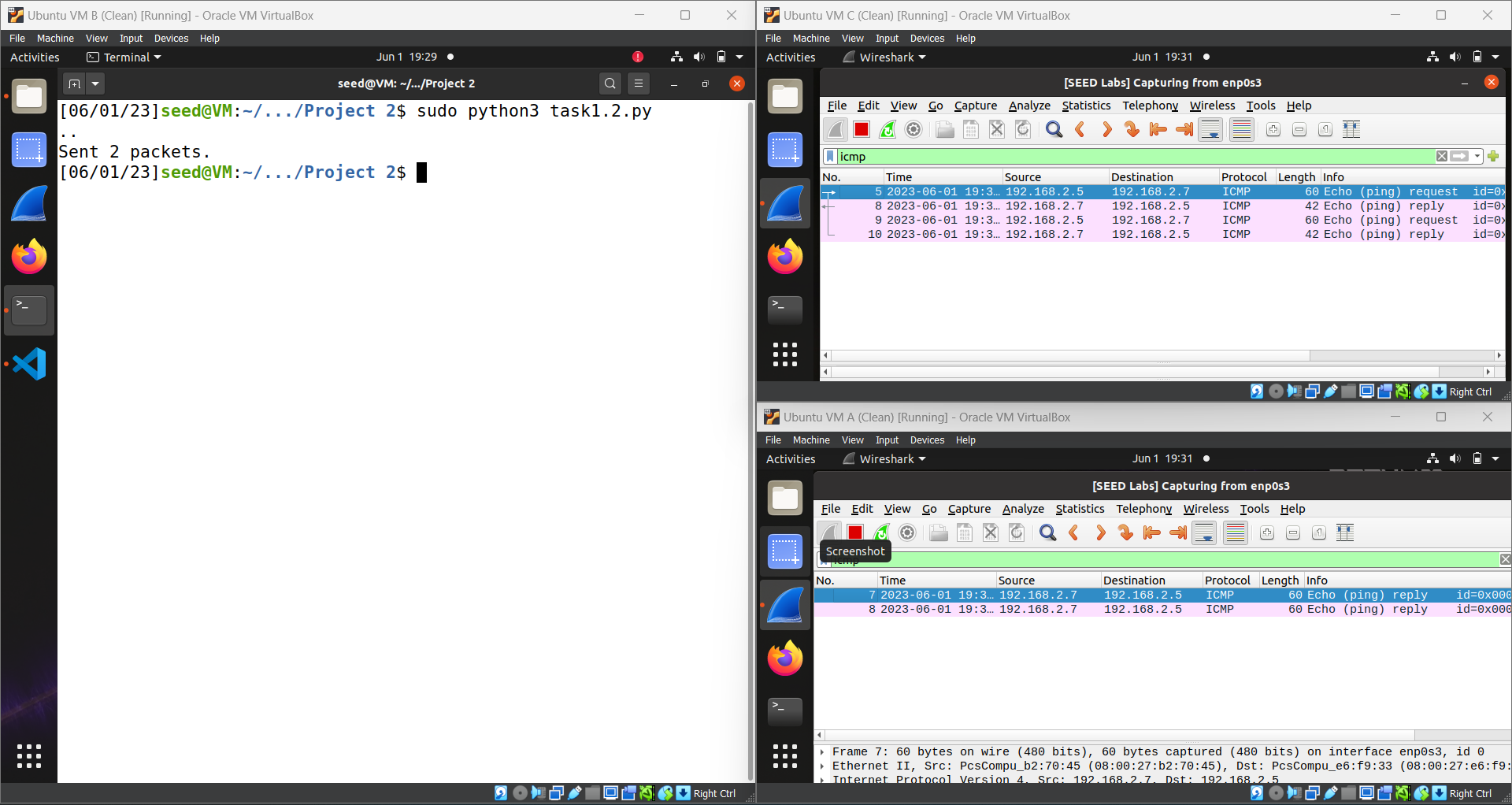
# Task 1.2

## Code

Created an instance of IP Object, set the values of source and destination IP. Created an instance of ICMP Object (by default, it is of type 8: ICMP echo request). Division operator is overloaded in IP Object. It updates it’s values and adds content in upper layer (ICMP in this case).



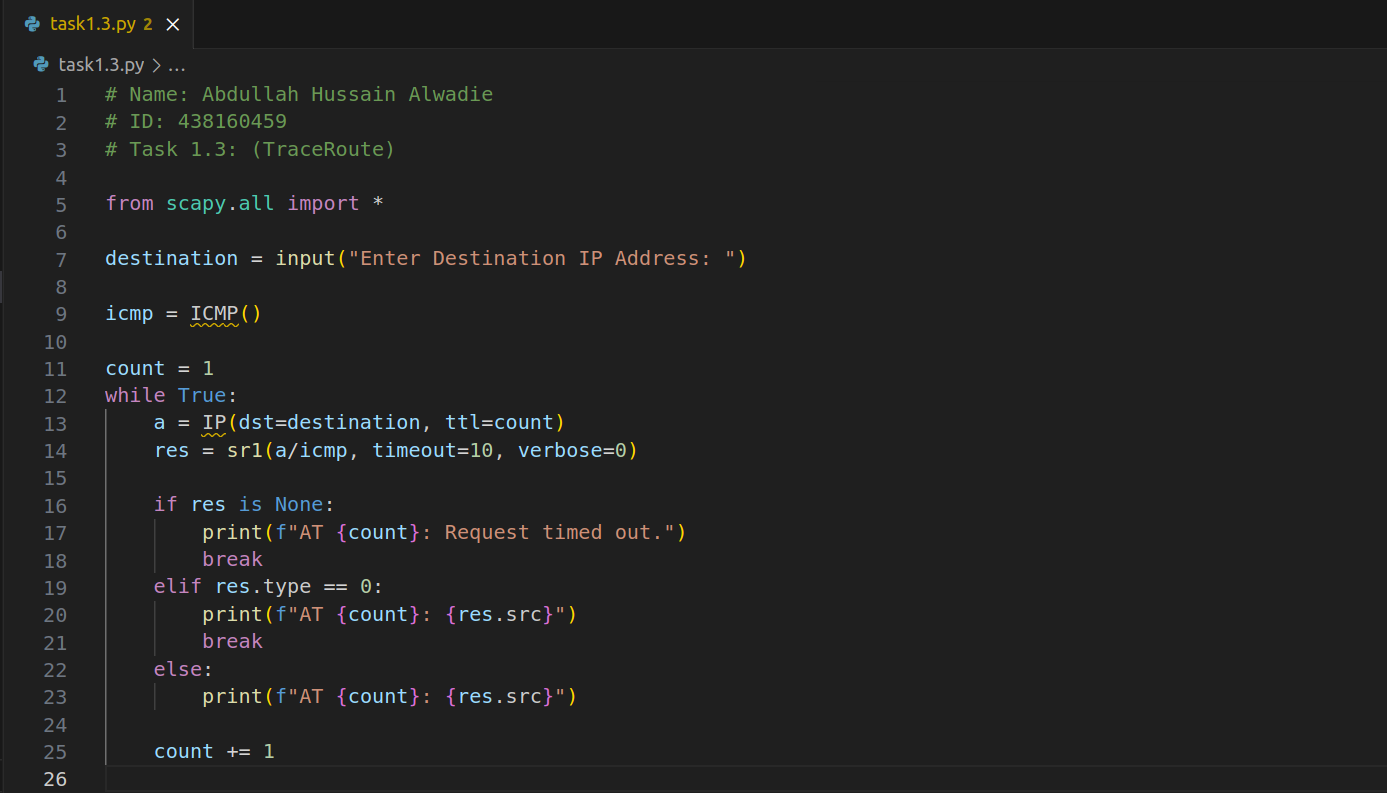
## Output



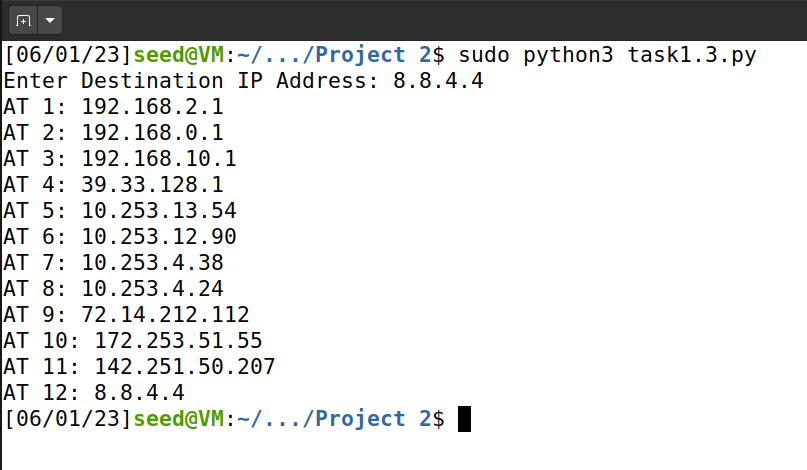
# Task 1.3

## Code

As we know, ttl basically is allowed number of hops to be covered. We start by one and keeps on increasing by one until we receive an ICMP type 0 (echo reply) instead of an ICMP type 11 (Time-to-live exceed)



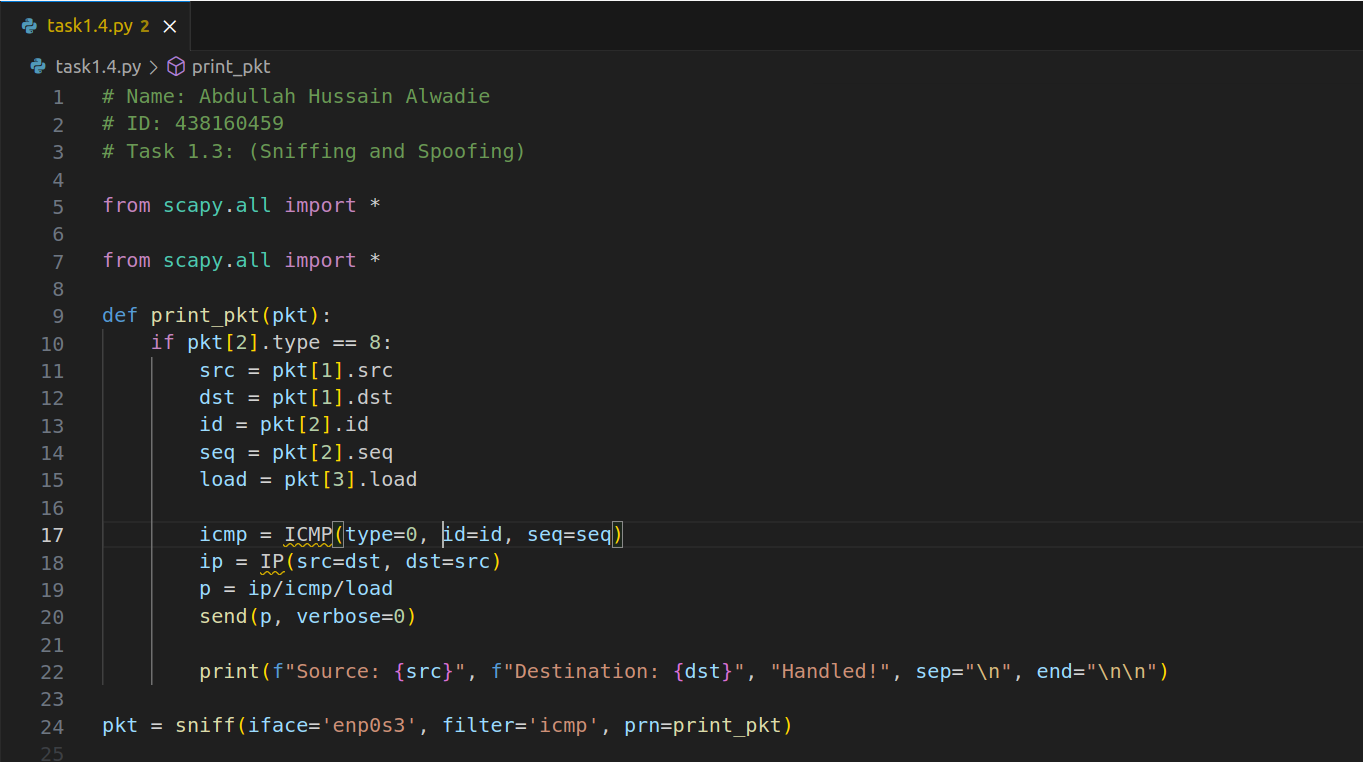
## Output



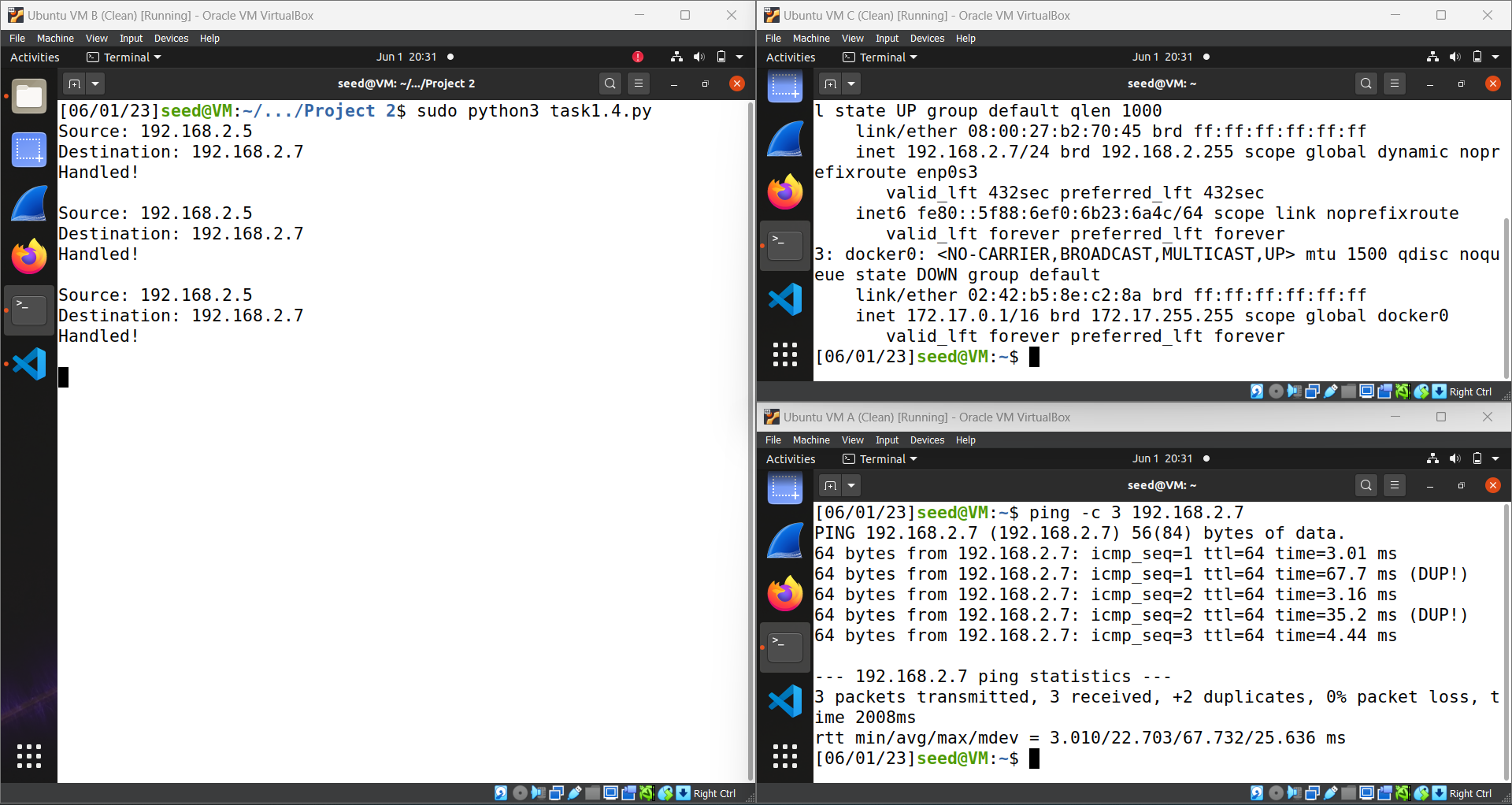
# Task 1.4

## Code

Sniffed ICMP packets and checked if they are of type 8 (echo request) then created an instance of ICMP object with type 0 (echo reply), id and seq of original ICMP header. Created an instance of IP Object with reverted source and destination Ips address and finally crafted the packet along with original payload that was received and sent it back to the source address.



## Output with Machine On



## Output with Machine Off

