**Assignment questions:**

1. Consider the following output of “Show ip interface brief” of router R1. Try to create a parser for the below output without using any predefined functions, try to use regular expressions and yield the output as given below in dictionary format.

**Note:** Try to follow good code practice by writing it as a function and add comments explaining each line of code and write pseudo code as well.

|  |
| --- |
| R1#show ip interface brief  Interface IP-Address OK? Method Status Protocol  FastEthernet0/0 15.0.15.1 YES manual up up  FastEthernet0/1 10.0.12.1 YES manual up up  FastEthernet0/2 10.0.13.1 YES manual up up  FastEthernet0/3 unassigned YES unset up down  Loopback0 10.1.1.1 YES manual up up  Loopback100 100.0.0.1 YES manual up up |

OUTPUT:

|  |
| --- |
| {  “FastEthernet0/0”: { “IP-Address”: “15.0.15.1”,  “Method”: “manual”,  “Status”: “up”,  “Protocol”: “up”},  “FastEthernet0/1”: { “IP-Address”: “10.0.12.1”,  “Method”: “manual”,  “Status”: “up”,  “Protocol”: “up”},  .  .  .  .  .  .  .  .  .  .  .  .  .  .  “Loopback100”: { “IP-Address”: “100.0.0.1”,  “Method”: “manual”,  “Status”: “up”,  “Protocol”: “up”}  } |

**Answer:**

### **Pseudo Code**

1. Import re package to use regular expressions
2. Define function parser to accept multi line string as input
3. Create a re pattern to accept whiteline and non-whiteline charecters.

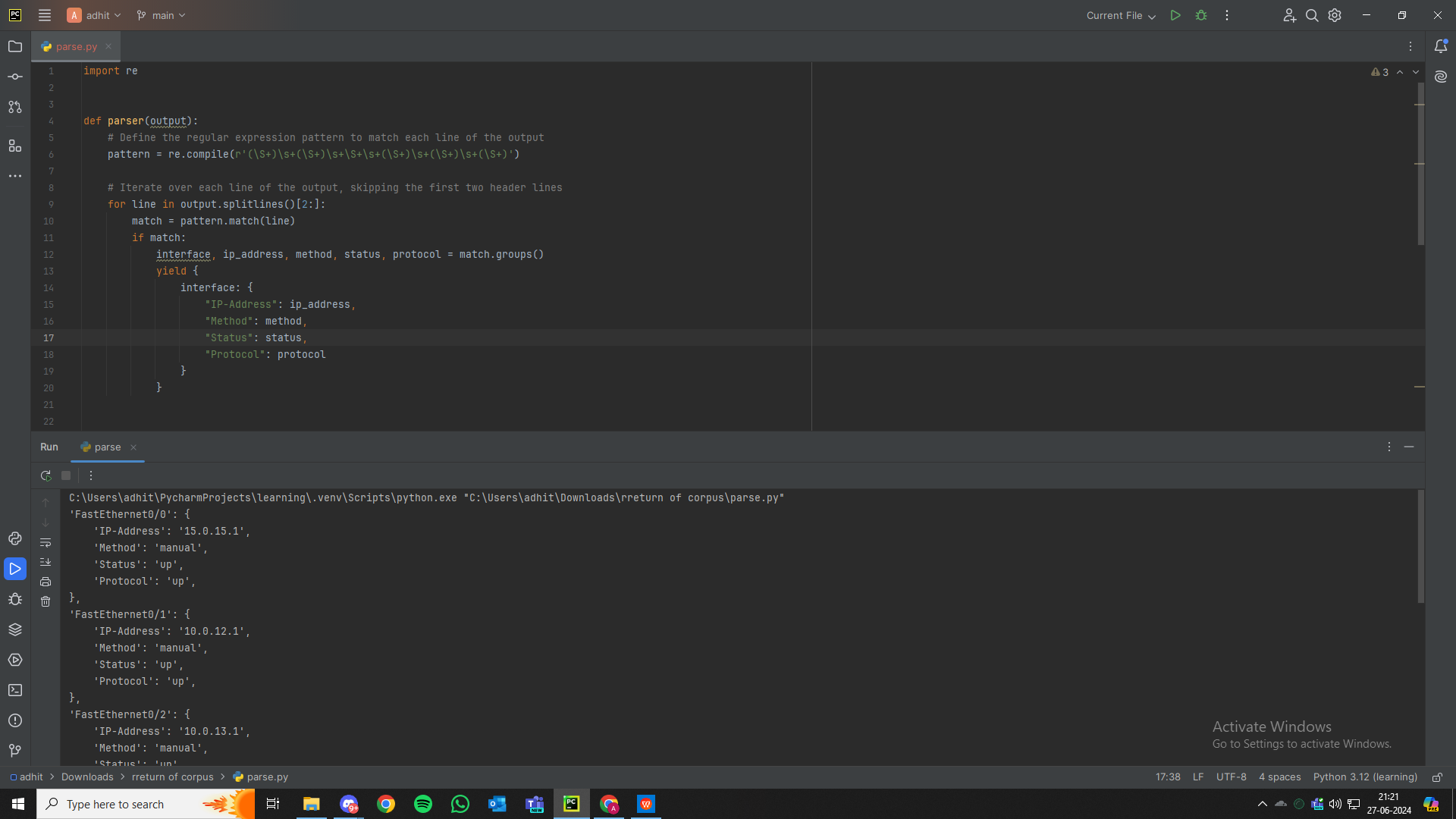
The regular expression pattern

(\S+)\s+(\S+)\s+\S+\s+(\S+)\s+(\S+)\s+(\S+)

(\S+)-pick up non-whiteline character

\s+ - ignore whiteline charecters

1. Iterate each line of output



**Code**

import re  
  
def parser(output):  
 *# Define the regular expression pattern to match each line of the output* pattern = re.compile(r'(\S+)\s+(\S+)\s+\S+\s+(\S+)\s+(\S+)\s+(\S+)')  
   
 *# Iterate over each line of the output, skipping the first two header lines* for line in output.splitlines()[2:]:  
 match = pattern.match(line)  
 if match:  
 interface, ip\_address, method, status, protocol = match.groups()  
 yield {  
 interface: {  
 "IP-Address": ip\_address,  
 "Method": method,  
 "Status": status,  
 "Protocol": protocol  
 }  
 }  
  
output = """  
Interface IP-Address OK? Method Status Protocol  
FastEthernet0/0 15.0.15.1 YES manual up up  
FastEthernet0/1 10.0.12.1 YES manual up up  
FastEthernet0/2 10.0.13.1 YES manual up up  
FastEthernet0/3 unassigned YES unset up down  
Loopback0 10.1.1.1 YES manual up up  
Loopback100 100.0.0.1 YES manual up up  
"""  
  
*# call parser function*ethernet\_interface = {}  
for dicts in parser(output):  
 ethernet\_interface.update(dicts)  
  
for interface, details in ethernet\_interface.items():  
 print(f"'{interface}': {{")  
 for key, value in details.items():  
 print(f" '{key}': '{value}',")  
 print("},")

2)Consider the below PCAP:  


MACsec (Media Access Control Security) is a security protocol that provides encryption for wired LANs. It encrypts the entire frame payload, regardless of its type—whether it’s an IP packet, voice data, or even a spanning tree protocol (STP) BPDU.

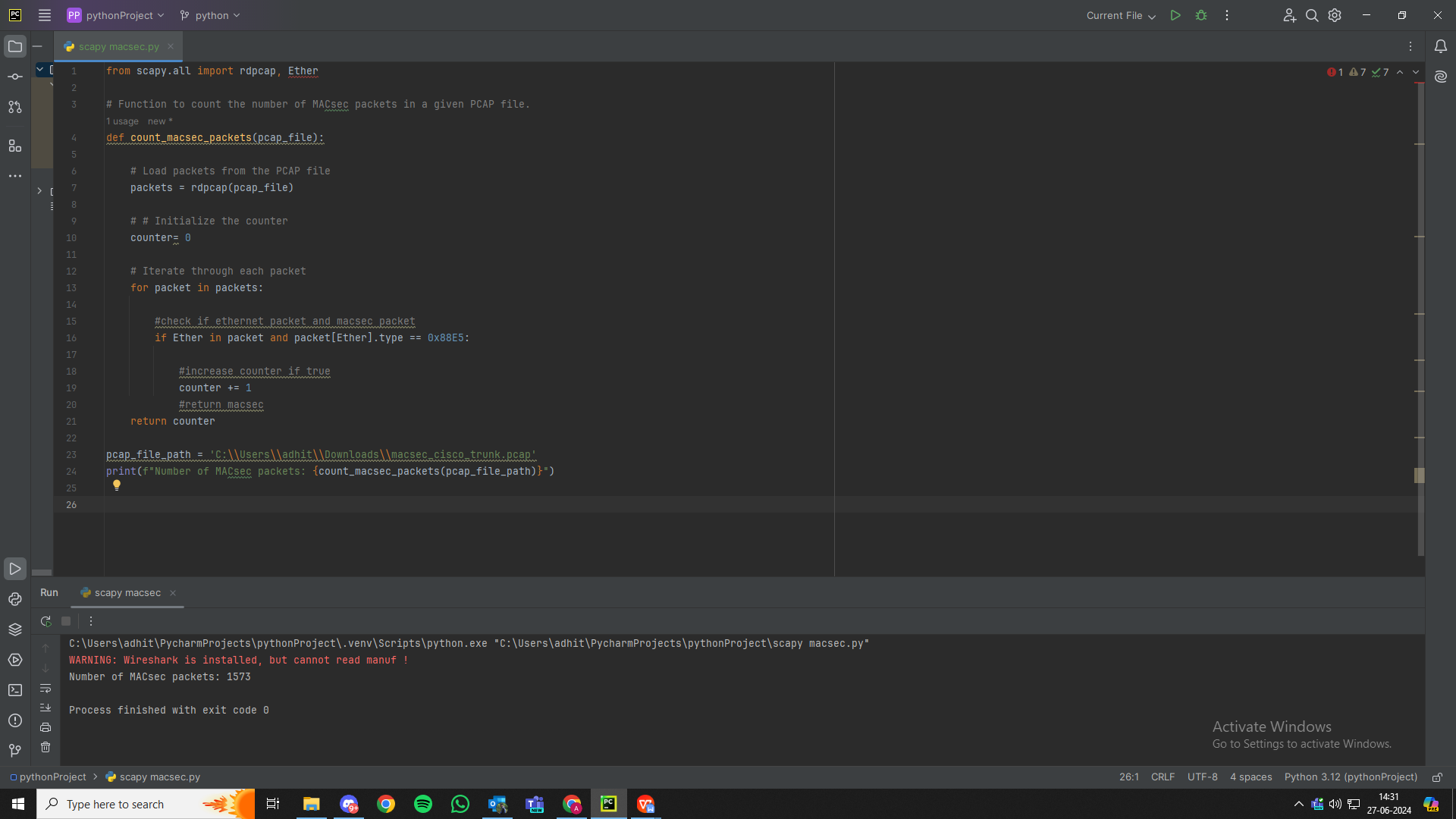
Use Python and Scapy and identify the presence and number of MACsec Packets in the PCAP given.

**Note:** Try to follow good code practice by writing it as a function and add comments explaining each line of code and write pseudo code as well.

**Answer:**

### **Pseudo Code**

1. Import scapy modules rdpcap(to read pcap file) and ether(to use ether filters).
2. Define a function to count macsec packets that uses a file path as argument.
3. Load the pcap file using scapy.
4. Intialize a counter.
5. Iterate through each packet of the pcap file and check if it is a macsec packet using ethertype filter (packet[Ether].type == 0x88E5) and increase counter if packet is identified as a macsec packet.
6. Return counter



**Code**

from scapy.all import rdpcap, Ether  
  
*# Function to count the number of MACsec packets in a given PCAP file.*def count\_macsec\_packets(pcap\_file):  
  
 *# Load packets from the PCAP file* packets = rdpcap(pcap\_file)  
  
 *# # Initialize the counter* counter= 0  
  
 *# Iterate through each packet* for packet in packets:  
  
 *#check if ethernet packet and macsec packet* if Ether in packet and packet[Ether].type == 0x88E5:  
  
 *#increase counter if true* counter += 1  
 *#return macsec* return counter  
  
pcap\_file\_path = 'C:\\Users\\adhit\\Downloads\\macsec\_cisco\_trunk.pcap'  
print(f"Number of MACsec packets: {count\_macsec\_packets(pcap\_file\_path)}")