Assignment questions:

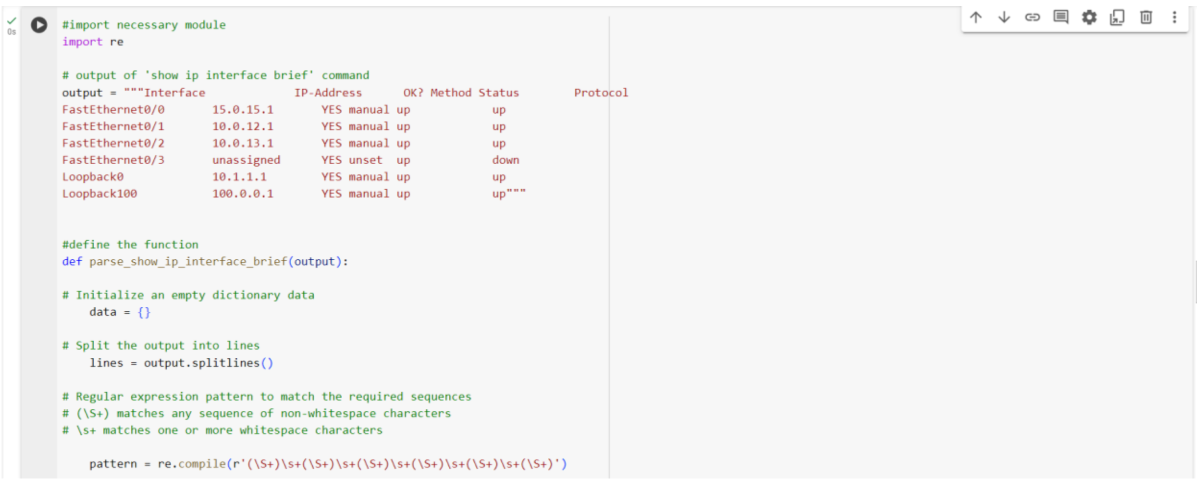
1) Consider the following output of “Show ip interface brief” of router R1. Try to create a parser for the below output without using any predefined functions, try to use regular expressions and yield the output as given below in dictionary format.

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.

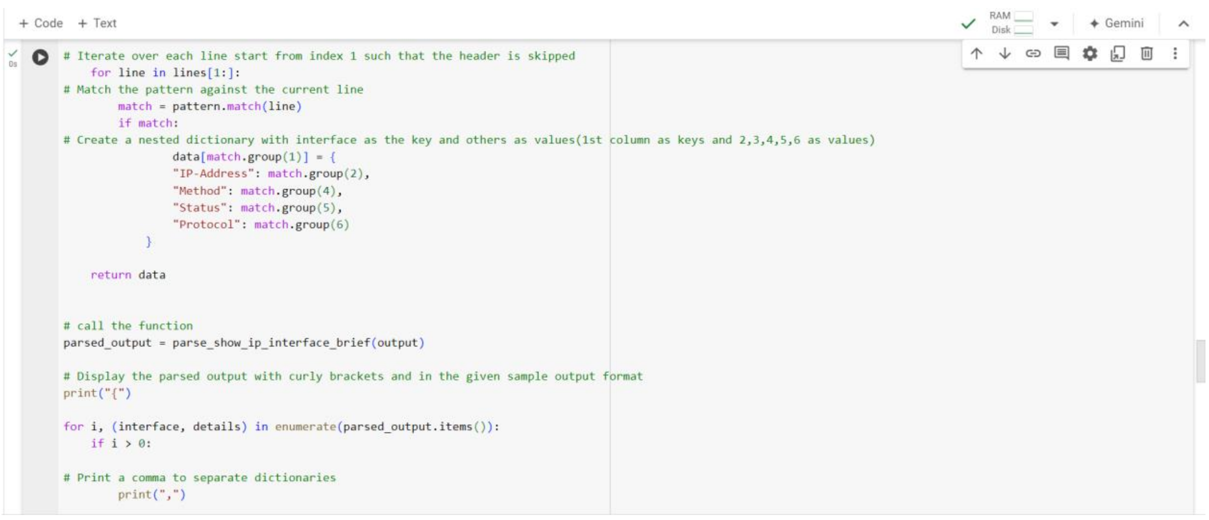
```
R1#show ip interface brief
Interface             IP-Address       OK? Method Status          Protocol
FastEthernet0/0       15.0.15.1        YES manual up              up
FastEthernet0/1       10.0.12.1        YES manual up              up
FastEthernet0/2       10.0.13.1        YES manual up              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:

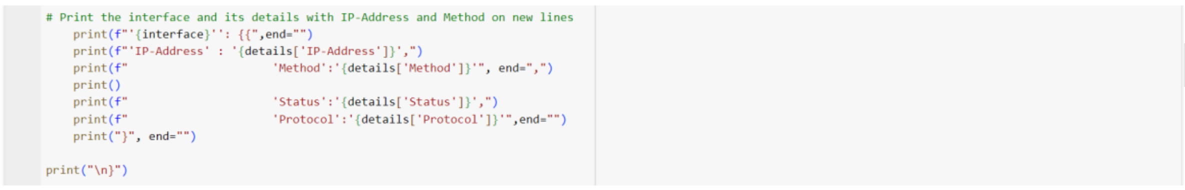
```
{
“FastEthernet0/0”:{“IP-Address”:“15.0.15.1”,
                  “Method”:“manual”,
                  “Status”:“up”,
                  “Protocol”:“up”},
“FastEthernet0/1”:{“IP-Address”:“10.0.12.1”,
                  “Method”:“manual”,
                  “Status”:“up”,
                  “Protocol”:“up”},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



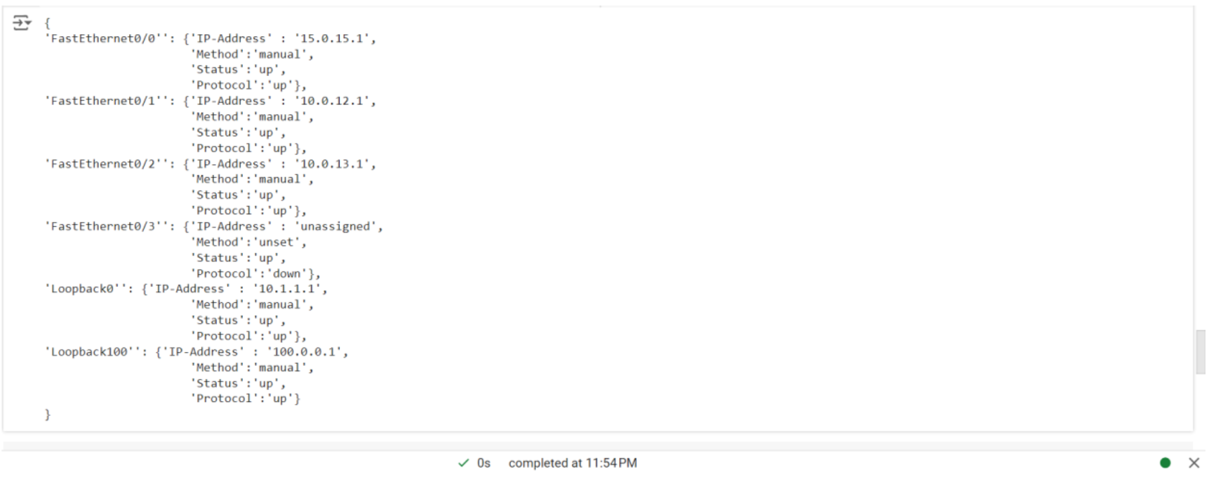
```
#import necessary module
import re

# output of 'show ip interface brief' command
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"""


#define the function
def parse_show_ip_interface_brief(output):

# Initialize an empty dictionary data
    data = {}

# Split the output into lines
    lines = output.splitlines()

# Regular expression pattern to match the required sequences
# (\S+) matches any sequence of non-whitespace characters
# \s+ matches one or more whitespace characters

    pattern = re.compile(r'(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)')
```



```
# Iterate over each line start from index 1 such that the header is skipped
    for line in lines[1:]:
# Match the pattern against the current line
        match = pattern.match(line)
        if match:
# Create a nested dictionary with interface as the key and others as values(1st column as keys and 2,3,4,5,6 as values)
            data[match.group(1)] = {
                "IP-Address": match.group(2),
                "Method": match.group(4),
                "Status": match.group(5),
                "Protocol": match.group(6)
            }

    return data


# call the function
parsed_output = parse_show_ip_interface_brief(output)

# Display the parsed output with curly brackets and in the given sample output format
print("{")

for i, (interface, details) in enumerate(parsed_output.items()):
    if i > 0:

# Print a comma to separate dictionaries
        print(",")
```



```
# Print the interface and its details with IP-Address and Method on new lines
    print(f"'{interface}'': {{",end="")
    print(f"'IP-Address' : '{details['IP-Address']}',")
    print(f"                      'Method':'{details['Method']}'", end=",")
    print()
    print(f"                      'Status':'{details['Status']}',")
    print(f"                      'Protocol':'{details['Protocol']}'",end="")
    print("}", end="")

print("\n}")
```



```
{
'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'},
'FastEthernet0/2'': {'IP-Address' : '10.0.13.1',
                     'Method':'manual',
                     'Status':'up',
                     'Protocol':'up'},
'FastEthernet0/3'': {'IP-Address' : 'unassigned',
                     'Method':'unset',
                     'Status':'up',
                     'Protocol':'down'},
'Loopback0'': {'IP-Address' : '10.1.1.1',
                     'Method':'manual',
                     'Status':'up',
                     'Protocol':'up'},
'Loopback100'': {'IP-Address' : '100.0.0.1',
                     'Method':'manual',
                     'Status':'up',
                     'Protocol':'up'}
}
```

PSEUDO CODE:

```
Step 1: Start
Step 2: Import the necessary module “re”-regular expression
Step 3: Store the given sample “show ip interface brief” command into output variable
Step 4: Define the function “parse_show_ip_interface_brief
Step 5: Initialize an empty dictionary “data”
Step 6: Split the output into array lines
Step 7: Create a nested dictionary with interface as the keys and others as values
Step 8: return the dictionary data
Step 9: call the function
Step 10: Display the dictionary with the respective output format
Step 11: Stop
```

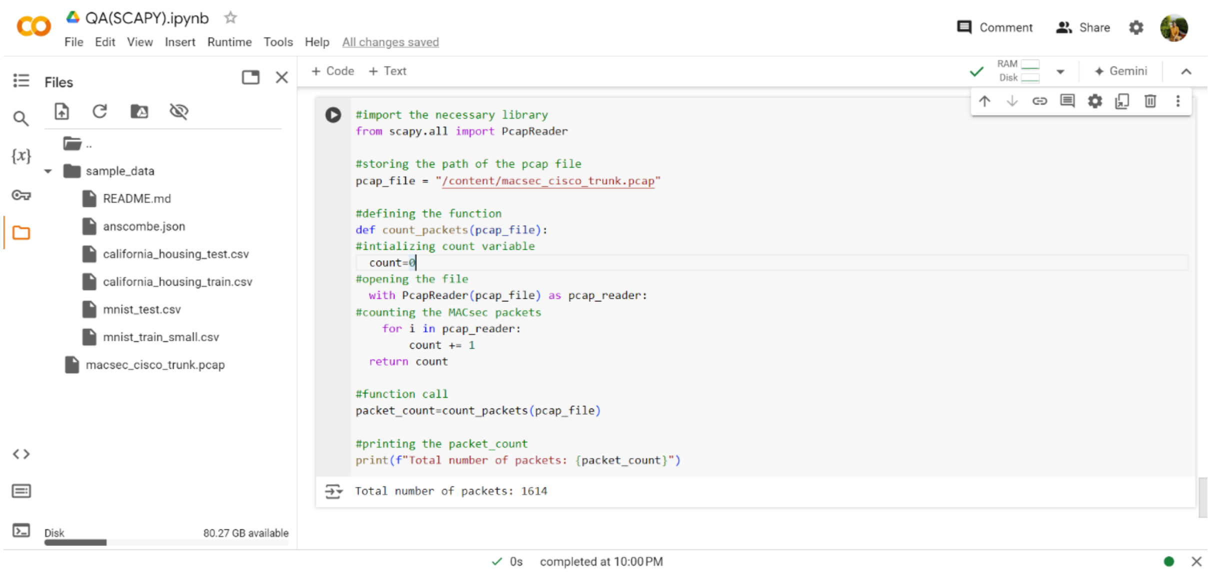
2)Consider the below PCAP:

macsec_cisco_trunk.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.



```
#import the necessary library
from scapy.all import PcapReader

#storing the path of the pcap file
pcap_file = "/content/macsec_cisco_trunk.pcap"

#defining the function
def count_packets(pcap_file):
#intializing count variable
  count=0
#opening the file
  with PcapReader(pcap_file) as pcap_reader:
#counting the MACsec packets
    for i in pcap_reader:
        count += 1
  return count

#function call
packet_count=count_packets(pcap_file)

#printing the packet_count
print(f"Total number of packets: {packet_count}")
```

```
Total number of packets: 1614
```

PSEUDO CODE:

```
Step 1: Start
Step 2: Import the scapy network library
Step 3: Assign the pcap file path to the variable "pcap_file"
Step 4: Define the function "count_packets"
Step 5: Initialize the count to 0
Step 6: Create an object "pcap_reader" for "PcapReader module"
Step 7: Iterate and increment the count value till the end of the file
Step 8: Return the count value
Step 9: Call the function "count_packets"
```

Step 10: Display the count value using the variable “packet_count”

Step 11: Stop