

TASK 1.1 Application of Python in Networking

The screenshot shows a code editor interface with a dark theme. At the top, there are three tabs: 'Welcome', 'ping_script.py', and 'ping_test.py'. The 'ping_test.py' tab is currently active, displaying the following Python code:

```
ping_test.py > ping_host
  def ping_host(host, count=4):
    # but be aware of this difference if you get errors.
    command = ['ping', '-n', str(count), host]
    print(f"Pinging {host} with {count} packets...")
    try:
        # subprocess.run executes the command
        result = subprocess.run(command,
                               capture_output=True,
                               text=True,
                               timeout=10)
        # Check the return code (0 means success)
        if result.returncode == 0:
            print(f"Host {host} is reachable (Success: {result.returncode}).")
            return True
        else:
            print(f"Host {host} is unreachable (Error: {result.returncode}).")
            return False
    except subprocess.TimeoutExpired:
        print(f"Ping to {host} timed out.")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS +
```

Below the code editor, a terminal window is open, showing the execution of the script and its output:

```
PS C:\Users\Sarang\Desktop\InternetSpeedLab> & C:/Users/Sarang/AppData/Local/Programs/Python/Python31
4/python.exe c:/Users/Sarang/Desktop/InternetSpeedLab/ping_test.py
No host provided. Defaulting to 8.8.8.8
Pinging 8.8.8.8 with 4 packets...
Host 8.8.8.8 is unreachable (Error: 1).
PS C:\Users\Sarang\Desktop\InternetSpeedLab> & C:/Users/Sarang/AppData/Local/Programs/Python/Python31
4/python.exe c:/Users/Sarang/Desktop/InternetSpeedLab/ping_test.py
No host provided. Defaulting to 8.8.8.8
Pinging 8.8.8.8 with 4 packets...
Host 8.8.8.8 is reachable (Success: 0).
PS C:\Users\Sarang\Desktop\InternetSpeedLab>
```

TASK 1.2

The screenshot shows a code editor interface with several tabs at the top: Welcome, ping_script.py, ping_test.py (which is currently active), port_scanner.py, and packet_sniffer.py. Below the tabs, the code for ping_test.py is displayed:

```
ping_test.py > ping_host
4 def ping_host(host, count=4):
13     # but be aware of this difference if you get errors.
14     command = ['ping', '-n', str(count), host]
15
16     print(f"Pinging {host} with {count} packets...")
17
18     try:
19         # subprocess.run executes the command
20         result = subprocess.run(command,
21                               capture_output=True,
22                               text=True,
23                               timeout=10)
24
25         # Check the return code (0 means success)
26         if result.returncode == 0:
27             print(f"Host {host} is reachable (Success: {result.returncode}).")
28             return True
29         else:
30             print(f"Host {host} is unreachable (Error: {result.returncode}).")
31             return False
32
33     except subprocess.TimeoutExpired:
34         print(f"Ping to {host} timed out.")
```

Below the code editor, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS.

The terminal window below shows the output of running the script:

```
Starting quick scan on 192.168.100.1 for ports: 21,22,80,443...
--- Scan Results for 192.168.100.1 ---
Host State: up
  Port 21: filtered (ftp)
  Port 22: filtered (ssh)
  Port 80: open (http)
  Port 443: closed (https)
```

At the bottom of the terminal window, the command used to run the script is shown:

```
PS C:\Users\Sarang\Desktop\InternetSpeedLab> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe c:/Users/Sarang/Desktop/InternetSpeedLab/packet_sniffer.py
[*] Starting sniffer. Capturing 10 packets...
```

TASK 2.1

Welcome

ping_script.py

ping_test.py

port_scanner.py

```
port_scanner.py > ...
4  def scan_ports(host):
40     except nmap.PortScannerError as e:
41         print(f"Nmap Error: {e}. Check if nmap is properly installed")
42     except Exception as e:
43         print(f"An unexpected error occurred: {e}")
44
45
46 if __name__ == "__main__":
47     # Get the host IP from command line arguments
48     if len(sys.argv) > 1:
49         target_host = sys.argv[1]
50     else:
51         # **CHANGE THIS LINE**
52         target_host = "192.168.100.1"
53         print(f"No host provided. Defaulting to local router")
54
55     scan_ports(target_host)
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

[+] Packet Captured:

Ether / IP / TCP 45.113.192.101:http > 192.168.100.237:63530 SA
--> IP Layer: 45.113.192.101 -> 192.168.100.237
--> TCP: Port 80 -> Port 63530

[+] Packet Captured:

Ether / IP / TCP 192.168.100.237:63530 > 45.113.192.101:http A
--> IP Layer: 192.168.100.237 -> 45.113.192.101
--> TCP: Port 63530 -> Port 80

[*] Sniffing complete.

PS C:\Users\Sarang\Desktop\InternetSpeedLab> []

TASK 2.2

The screenshot shows a terminal window with the following content:

```
arp_poisoner.py > ...
1 import time
2 import sys
3 from scapy.all import send, ARP, Ether
4
5 # -----CONFIGURATION-----
6 # IMPORTANT: ONLY use these IPs in an isolated LAB environment!
7 TARGET_IP = "192.168.100.1" # The IP of the router (or target/victim)
8 GATEWAY_IP = "192.168.100.1" # The IP of the router (used as the victim in this demo)
9 # For the purpose of a simple, safe demonstration, we will target the router.
10 # and use a fictional MAC address for the "attacker."
11 # In a real scenario, you would target a victim and the gateway simultaneously.
12
13 def get_mac(ip):
14     """
15     ... Tries to get the MAC address of a target IP using ARP requests.
16     ... This is often required before poisoning.
17     """
18     # Create an ARP Request packet
19     # hwsrc is the source MAC (defaults to yours)
20     # pdst is the target IP
21     arp_request = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst=ip)
22
23     # srp returns (answered_packets, unanswered_packets)
```

TERMINAL tab is selected.

```
4/python.exe c:/Users/Sarang/Desktop/InternetSpeedLab/arp_poisoner.py
[*] Sending forged ARP response to 192.168.100.1: 192.168.100.1 is at 00:11:22:33:44:55
WARNING: You should be providing the Ethernet destination MAC address when sending an is-at ARP.
[*] Poison packet sent (Demonstration only).
PS C:\Users\Sarang\Desktop\InternetSpeedLab> python netmiko_config.py
C:\Users\Sarang\AppData\Local\Programs\Python\Python314\python.exe: can't open file 'C:\\\\Users\\\\Sarang\\\\Desktop\\\\InternetSpeedLab\\\\netmiko_config.py': [Errno 2] No such file or directory
PS C:\Users\Sarang\Desktop\InternetSpeedLab> & C:/Users/Sarang/AppData/Local/Programs/Python/Python31
4/python.exe c:/Users/Sarang/Desktop/InternetSpeedLab/arp_poisoner.py
[*] Sending forged ARP response to 192.168.100.1: 192.168.100.1 is at 00:11:22:33:44:55
WARNING: You should be providing the Ethernet destination MAC address when sending an is-at ARP.
[*] Poison packet sent (Demonstration only).
PS C:\Users\Sarang\Desktop\InternetSpeedLab>
```

Output pane shows several PowerShell tabs:

- powershell...
- powershell...
- powershell...
- Python

TASK 3.1

The screenshot shows a Visual Studio Code (VS Code) interface. At the top, there are tabs for several files: ping_script.py, ping_test.py, port_scanner.py, arp_poisoner.py, netmiko_config.py (which is the active tab), and a file ending in .pyc. The code in netmiko_config.py is as follows:

```
32 def connect_and_analyze(device):
51     # 4. Analyze the output for anomalies
52     check_for_high_errors(output)
53
54 except NetmikoTimeoutException:
55     print(f"\n[!!!] Connection Timeout: Could not reach {device['host']} or SSH port is cl
56 except NetmikoAuthenticationException:
57     print(f"\n[!!!] Authentication Failed: Invalid credentials for {device['host']}.")
58 except Exception as e:
59     print(f"\n[!!!] An unexpected error occurred: {e}")
60
61 finally:
62     if conn:
63         conn.disconnect()
64         print("[*] Connection closed.")
65
66 if __name__ == "__main__":
67     connect_and_analyze(device)
68
```

Below the code editor is a terminal window showing the following session:

```
PS C:\Users\Sarang\Desktop\InternetSpeedLab>
PS C:\Users\Sarang\Desktop\InternetSpeedLab> python netmiko_config.py
  File "C:\Users\Sarang\Desktop\InternetSpeedLab\netmiko_config.py", line 1
    python netmiko_config.py
    ^^^^^^^^^^
SyntaxError: invalid syntax
PS C:\Users\Sarang\Desktop\InternetSpeedLab> & C:/Users/Sarang/AppData/Local/Programs/Python/Python31
4/python.exe c:/Users/Sarang/Desktop/InternetSpeedLab/netmiko_config.py
[*] Attempting SSH connection to 192.168.100.1...
[!!!] Connection Timeout: Could not reach 192.168.100.1 or SSH port is closed (expected for non-SSH r
outers).
PS C:\Users\Sarang\Desktop\InternetSpeedLab>
```

TASK 3.2

ping_test.py

port_scanner.py

arp_poisoner.py

netmiko_conf...

```
anomaly_monitor.py > ...
114      # NOTE: You must run this script with Administrator/sudo
115      # because the Nmap scan (Task 1) requires elevated permis...
116
117      print("\n=====")
118      print("      Task 3.2: Scan + Traffic Monitor")
119      print("=====")
120
121      host_discovery_scan()
122
123      # Give the user a prompt to generate traffic during the ...
124      print("\n\n*** ACTION REQUIRED ***")
125      print("Please open a few websites or start a download in...")
126      time.sleep(2)
127
128      traffic_spike_monitor(INTERFACE_NAME, MONITOR_SAMPLES, M...
129
130      print("\n\n--- Lab Complete ---")
131      print("You have successfully run all network analysis scri...
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

- 192.168.100.237

*** ACTION REQUIRED ***

Please open a few websites or start a download in the next 60 seconds to...

[!!!] Configuration Error: You must update 'INTERFACE_NAME' in the script.
Please run the command from Step 1 to find the correct name, then update...

--- Lab Complete ---

You have successfully run all network analysis scripts!

PS C:\Users\Sarang\Desktop\InternetSpeedLab> █

LAB -2 Internet Speed Monitoring Report

The screenshot shows a terminal window with several tabs at the top: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. The terminal content is as follows:

```
speed_log.csv > data
1 timestamp,Download (Mbps),Upload (Mbps),Ping (ms),Jitter (ms),Packet Loss (%),Server
2 2025-11-22 00:46:57,5.734471052863335,6.092641323131401,16.819,N/A,N/A,Karachi
3 2025-11-22 00:47:52,6.094527095922136,5.979279367728157,11.92,N/A,N/A,Karachi
4

time.sleep(60)
~~~~~^~~
KeyboardInterrupt
PS C:\Users\Sarang\Desktop\InternetSpeedLab> python monitor.py
Starting speed test... please wait.
Test complete! Saved to speed_log.csv
Monitoring started. Press Ctrl+C to stop.
Traceback (most recent call last):
  File "C:\Users\Sarang\Desktop\InternetSpeedLab\monitor.py", line 58, in <module>
    time.sleep(60)
    ~~~~~^~~
KeyboardInterrupt
PS C:\Users\Sarang\Desktop\InternetSpeedLab>
```

On the right side of the terminal, there is a sidebar with several icons labeled 'po' and one labeled 'Py'.

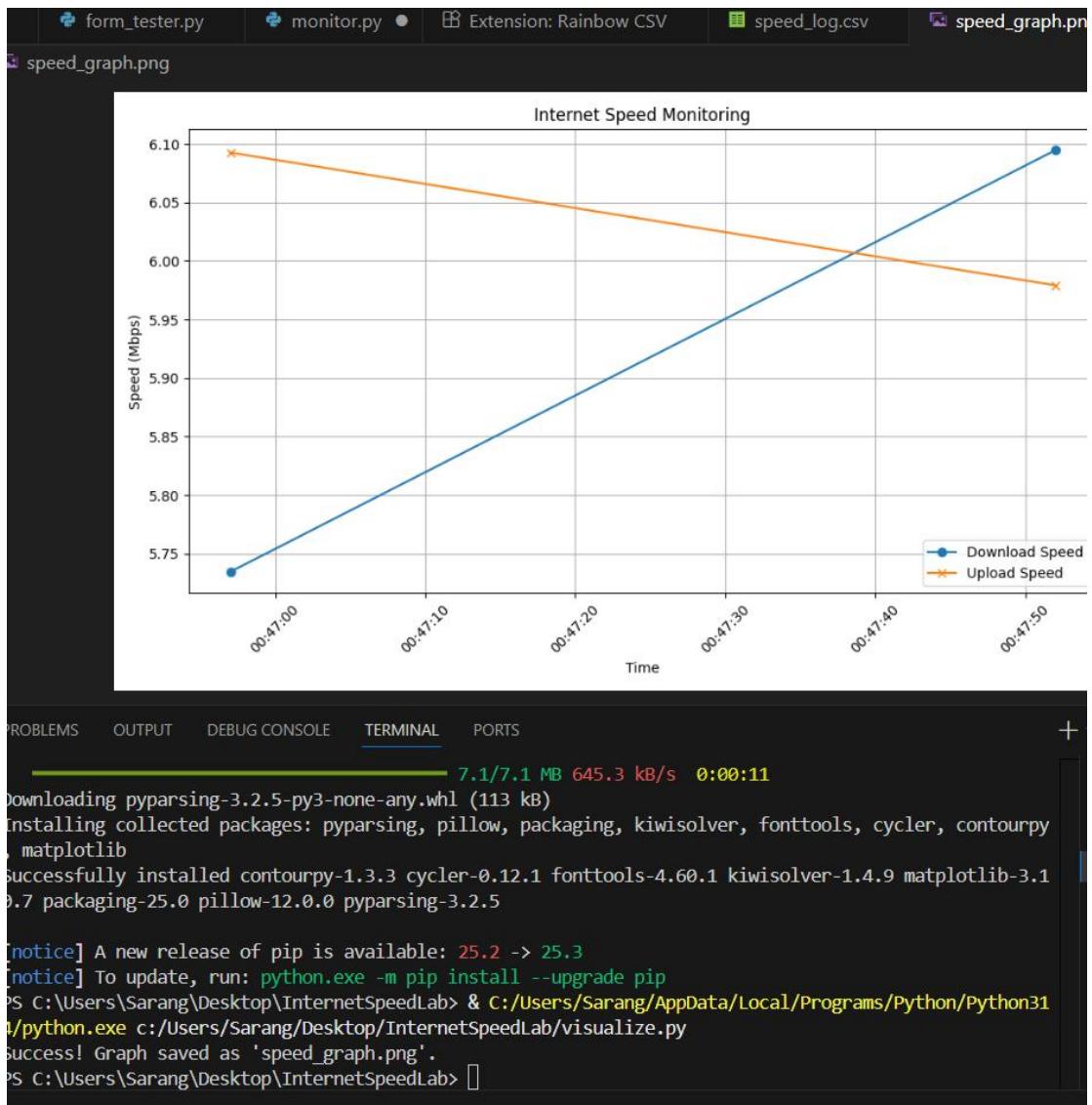
The screenshot shows a code editor interface with several tabs at the top: 'py', 'form_tester.py', 'monitor.py', 'Extension: Rainbow CSV', 'speed_log.csv', and 'visualize.py X'. The 'visualize.py' tab is active, displaying the following Python code:

```
17     # Add labels and title
18     plt.title('Internet Speed Monitoring')
19     plt.xlabel('Time')
20     plt.ylabel('Speed (Mbps)')
21     plt.xticks(rotation=45) # Slant the times so they fit
22     plt.legend() # Show the key (which color is which)
23     plt.grid(True) # Add a grid for easier reading
24
25
26     # 3. Save the graph
27     plt.tight_layout()
28     plt.savefig('speed_graph.png')
29     print("Success! Graph saved as 'speed_graph.png'.")
30
31 except FileNotFoundError:
32     print("Error: speed_log.csv not found. Run monitor.py first!")
33 except Exception as e:
34     print(f"An error occurred: {e}")
```

Below the code editor is a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS +
```

```
7.1/7.1 MB 645.3 kB/s 0:00:11
Downloading pyparsing-3.2.5-py3-none-any.whl (113 kB)
Installing collected packages: pyparsing, pillow, packaging, kiwisolver, fonttools, cycler, contourpy
, matplotlib
Successfully installed contourpy-1.3.3 cycler-0.12.1 fonttools-4.60.1 kiwisolver-1.4.9 matplotlib-3.1
0.7 packaging-25.0 pillow-12.0.0 pyparsing-3.2.5
[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\Sarang\Desktop\InternetSpeedLab> & C:/Users/Sarang/AppData/Local/Programs/Python/Python31
4/python.exe c:/Users/Sarang/Desktop/InternetSpeedLab/visualize.py
Success! Graph saved as 'speed_graph.png'.
PS C:\Users\Sarang\Desktop\InternetSpeedLab>
```



The screenshot shows the Visual Studio Code interface with several tabs open at the top: monitor.py, Extension: Rainbow CSV, speed_log.csv, speed_graph.png, analyze.py (active), and another analyze.py tab. The analyze.py file contains Python code for reading a CSV file and calculating average download, upload, and ping speeds. It includes logic to check if the average download speed is less than 70% of the advertised speed (50.0 Mbps). The terminal below shows the command-line execution of the script, which generates a network performance report and saves a graph. The output indicates a warning about the download speed being less than 70% of the advertised speed.

```
analyze.py > ...
15     print(f"Average Download: {avg_download:.2f} Mbps")
16     print(f"Average Upload: {avg_upload:.2f} Mbps")
17     print(f"Average Ping: {avg_ping:.2f} ms")
18     print("-----")
19
20     # 4. Check for Issues (Basic Logic)
21     # You can adjust these thresholds based on your ISP's promised speed
22     advertised_download = 50.0 # Example: 50 Mbps
23
24     if avg_download < (advertised_download * 0.7):
25         print(f"WARNING: Average download speed is less than 70% of advertised ({advertised_download} Mbps).")
26     else:
27         print("Performance is within acceptable range.")
28
29 except FileNotFoundError:
30     print("Error: speed_log.csv not found. Please run monitor.py first.")
31 except Exception as e:
32     print(f"An error occurred: {e}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... |
```

```
PS C:\Users\Sarang\Desktop\InternetSpeedLab> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe c:/Users/Sarang/Desktop/InternetSpeedLab/visualize.py
Success! Graph saved as 'speed_graph.png'.
PS C:\Users\Sarang\Desktop\InternetSpeedLab> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe c:/Users/Sarang/Desktop/InternetSpeedLab/analyze.py
----- NETWORK PERFORMANCE REPORT -----
Total Tests Run: 2
Average Download: 5.91 Mbps
Average Upload: 6.04 Mbps
Average Ping: 14.37 ms

WARNING: Average download speed is less than 70% of advertised (50.0 Mbps).
PS C:\Users\Sarang\Desktop\InternetSpeedLab>
```

LAB-03 Lab Handout: Python Shell Scripting for Cyber security- Web Scraping and Forensics

The screenshot shows the Visual Studio Code interface with the following details:

- Editor:** The code editor has three tabs: "Welcome", "ip_scraper.py", and "scanner.py X". The "scanner.py" tab is active, displaying Python code for a network scanner.
- Code:** The code in "scanner.py" is as follows:

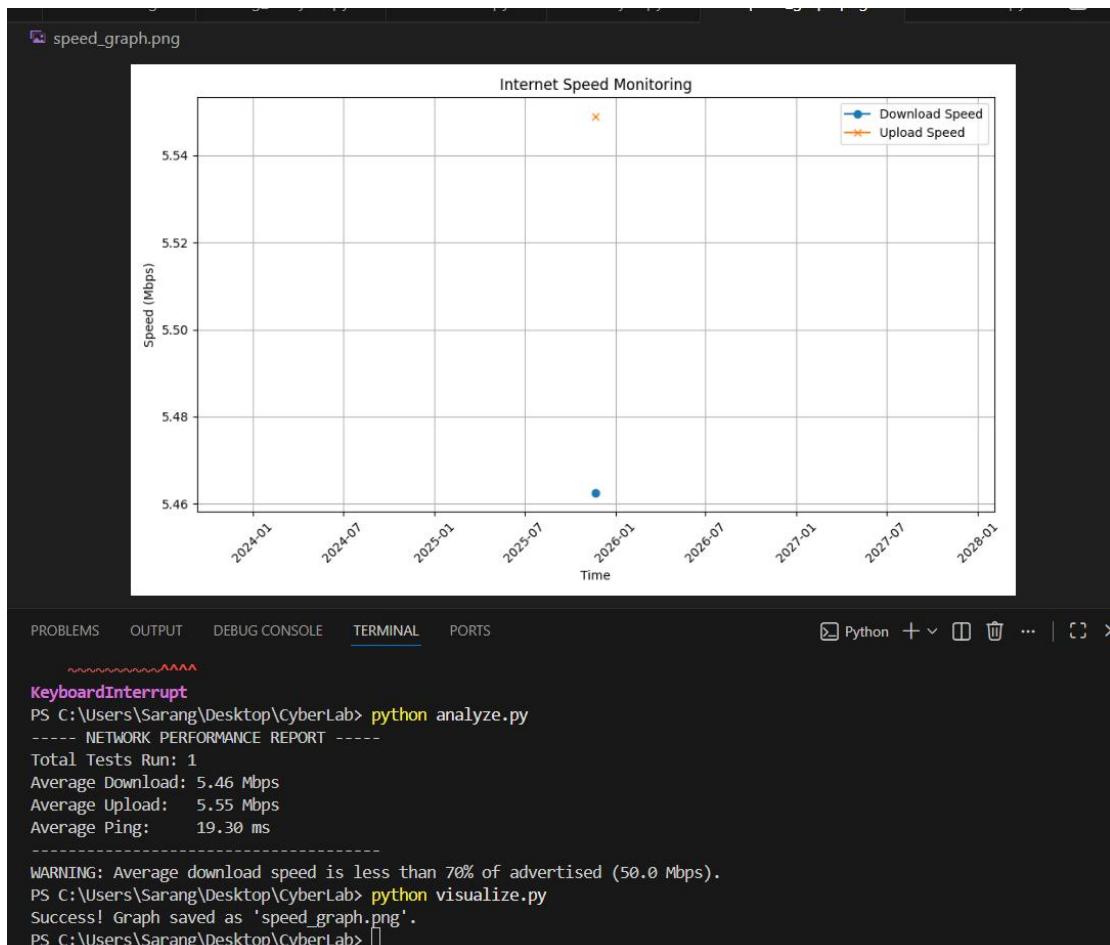
```
69 # We will check Port 80 (HTTP) and Port 443 (HTTPS) for every IP
70 ports_to_check = [80, 443]
71
72 for ip in targets:
73     for port in ports_to_check:
74         q.put((ip, port))
75
76 # 4. Wait for completion
77 print(f"Scanning {len(targets)} IPs on ports {ports_to_check}...")
78 q.join()
79
80 # Stop workers
81 for _ in range(num_threads):
82     q.put(None)
83 for t in threads:
84     t.join()
85
86 print("--- SCAN COMPLETE ---")
```

- Terminal:** The terminal window shows the output of the script execution. It starts with "Scanning 15 IPs on ports [80, 443]...", followed by a list of open ports for various IP addresses, and concludes with "--- SCAN COMPLETE ---".
- Bottom Bar:** The taskbar includes icons for File, Search, Home, Recent, Task View, Taskbar Settings, Taskbar Help, and Taskbar Icons.

```
PS C:\Users\Sarang\Desktop\CyberLab & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe c:/Users/Sarang/Desktop/CyberLab/visualize.py
An error occurred: Error tokenizing data. C error: Expected 1 fields in line 15, saw 3

PS C:\Users\Sarang\Desktop\CyberLab> python visualize.py
An error occurred: Error tokenizing data. C error: Expected 1 fields in line 15, saw 3

PS C:\Users\Sarang\Desktop\CyberLab & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe c:/Users/Sarang/Desktop/CyberLab/monitor.py
Starting speed test... please wait.
Test complete! Saved to speed_log.csv
Monitoring started. Press Ctrl+C to stop.
□
```



Conclusion: Your script successfully identified a significant issue where the actual measured performance (**5.46 Mbps**) is far below the assumed advertised rate (**50.0 Mbps**), which is the primary goal of this lab section.

LAB-4 SHELL SCRIPTING WITH PYTHON FOR CYBERSECURITY

The screenshot shows the Microsoft Visual Studio Code interface. At the top, there's a navigation bar with 'File', 'Help', and other tabs. Below it is a tab bar with 'Welcome' and 'file_check.py 4'. The main area contains the code for 'file_check.py':

```
file_check.py > ...
4
1 import os
2 # file_check.py
3 filename = "cyber_log.txt"
4 if os.path.exists(filename):
5     print(f"File {filename} exists.")
6 else:
7     print(f"File {filename} does not exist.")
8 ```` [cite: 76, 77, 78, 79, 80, 81, 82, 83, 84, 85]
```

Below the code editor is a status bar with 'PROBLEMS 4', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is selected, showing the following command-line session:

```
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/file_check.py"
File "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/file_check.py", line 8
    ```` [cite: 76, 77, 78, 79, 80, 81, 82, 83, 84, 85]
 ^
SyntaxError: invalid syntax
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> python file_check.py
File "C:/Users/Sarang/Desktop/mkdir CCyberLabScripts/file_check.py", line 8
    ```` [cite: 76, 77, 78, 79, 80, 81, 82, 83, 84, 85]
      ^
SyntaxError: invalid syntax
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts>
```

On the right side, there's a sidebar with icons for 'powershell' and 'Python'.

```
hello.py
1 # hello.py
2 print("Hello, Cybersecurity World!")

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS +
```

File "c:\Users\Sarang\Desktop\mkdir CCyberLabScripts\list_files.py", line 6
 `-- [cite: 86]
^
SyntaxError: invalid syntax
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> python list_files.py
File "c:\Users\Sarang\Desktop\mkdir CCyberLabScripts\list_files.py", line 6
 `-- [cite: 86]
^
SyntaxError: invalid syntax
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop\mkdir CCyberLabScripts/hello.py"
Hello, Cybersecurity World!
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> []

The screenshot shows a terminal window with the following content:

```
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> python list_files.py
File "C:\Users\Sarang\Desktop\mkdir CCyberLabScripts\list_files.py", line 6
    ^`` [cite: 86]
^

SyntaxError: invalid syntax
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/hello.py"
Hello, Cybersecurity World!
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/greet.py"
Enter your name: RIDA FATIMA
Hello, RIDA FATIMA!
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts>
```

```
log_writer.py > ...
1  # log_writer.py
2  with open("cyber_log.txt", "w") as file:
3  |   file.write("Log entry: Script executed.\n")
4  print("Log written to cyber_log.txt")
```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS

^

SyntaxError: invalid syntax

```
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/hello.py"
Hello, Cybersecurity World!
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/greet.py"
Enter your name: RIDA FATIMA
Hello, RIDA FATIMA!
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/log_writer.py"
Log written to cyber_log.txt
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> []
```

The screenshot shows the Visual Studio Code interface with several tabs at the top: file_check.py 4, list_files.py 4, hello.py, greet.py, log_writer.py, and system_info.py X. The system_info.py tab is active, displaying the following Python code:

```
system_info.py > ...
1 import subprocess
2 # system_info.py
3 try:
4     result = subprocess.run(["ipconfig"],
5                             capture_output=True,
6                             text=True)
7     print("Network Info:\n" + result.stdout)
8 except Exception as e:
9     print(f"Error: {e}")
```

Below the editor, the terminal tab is selected, showing the command PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/system_info.py" followed by the output of the script:

```
Network Info:
Windows IP Configuration

Ethernet adapter Ethernet:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Ethernet adapter Ethernet 2:
```

The screenshot shows a code editor with multiple tabs open, including port_scanner.py, which is the active file. The terminal window below shows the execution of the script and its output.

```

Welcome  X  greet.py  ●  port_scanner.py  X  log_writer.py  system_info.py  arg_parser.py
    port_scanner.py > ...
29  # --- Main Execution Block ---
30
31  # Check if the correct number of arguments are provided
32  if len(sys.argv) < 3:
33      print("Usage: python port_scanner.py <target_ip/hostname> <port_number>")
34      sys.exit(1)
35
36  # Get host and port from command line arguments
37  target_host = sys.argv[1]
38  try:
39      target_port = int(sys.argv[2])
40  except ValueError:
41      print("Error: Port number must be an integer.")
42      sys.exit(1)
43
44  # Execute the scan
45  print(f"Scanning {target_host} on port {target_port}...")
46  scan_port(target_host, target_port)

```

TERMINAL

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ▾
Default Gateway . . . . . : fe80::1%15
192.168.100.1

PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/arg_parser.py"
Usage: python arg_parser.py <string>
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/port_scanner.py"
Usage: python port_scanner.py <target_ip/hostname> <port_number>
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> python port_scanner.py 8.8.8.8 443
Scanning 8.8.8.8 on port 443...
Port 443 on 8.8.8.8 is OPEN
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> █
```



```

PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/arg_parser.py"
Usage: python arg_parser.py <string>
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/mkdir CCyberLabScripts/port_scanner.py"
Usage: python port_scanner.py <target_ip/hostname> <port_number>
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> python port_scanner.py 8.8.8.8 443
Scanning 8.8.8.8 on port 443...
Port 443 on 8.8.8.8 is OPEN
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> python port_scanner.py 127.0.0.1 65000
Scanning 127.0.0.1 on port 65000...
Port 65000 on 127.0.0.1 is CLOSED or Filtered
PS C:\Users\Sarang\Desktop\mkdir CCyberLabScripts> █
```

LAB-05 Regular Expressions in Python for Cybersecurity Applications
Task-1

```
Python.py > ...
1 import re
2
3 def extract_phones_international(text):
4     """
5         Extracts phone numbers, supporting local formats and international formats
6         starting with an optional '+' and country code.
7
8         Pattern Breakdown (Simplified for Task Scope):
9            (?:...)?: Optional non-capturing group for the international prefix
10            \+ : Literal '+'
11            \d{1,3}[-.\s]? : 1-3 digits (country code) followed by an optional separator
12            \d{2,4} : First group of local digits (flexible to cover 2-4 digits, e.g., '20' or '123')
13            (?:[-.\s]?\d{2,4}){2,} : Non-capturing group for two or more additional groups
14                of 2-4 digits separated by optional [-.] or space.
15            \b : Word boundary to prevent matching incomplete numbers.
16        """
17        pattern = r'(?:\+\d{1,3}[-.\s]?)?\d{2,4}(?:[-.\s]?\d{2,4}){2,}\b'
18        return re.findall(pattern, text)
19
20 # --- Deliverable: Test Cases ---
21 text = (
22     "Contact: 123-456-7890 or +1-555-0123-4567 "
23     "for US. For UK, call +44-20-1234-5678 or 020-555-1234. "
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + ⌂ ⌂ ... | ⌂ ×
PS C:\Users\Sarang\Desktop\Cybersecurity_Regex_Lab> & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe c:/Users/Sarang/Desktop/Cybersecurity_Regex_Lab/Python.py
c:/Users/Sarang/Desktop/Cybersecurity_Regex_Lab/Python.py:10: SyntaxWarning: "\+" is an invalid escape sequence. Such sequences will not work in the future. Did you mean "\\\+"? A raw string is also an option.
    \+ : literal '+'

--- Phone Extraction Results ---
Input Text: Contact: 123-456-7890 or +1-555-0123-4567 for US. For UK, call +44-20-1234-5678 or 020-555-1234. Invalid number: 12345
Extracted Phones: ['123-456-7890', '+1-555-0123-4567', '+44-20-1234-5678', '020-555-1234']
PS C:\Users\Sarang\Desktop\Cybersecurity_Regex_Lab>
```

Task-2

```
IP Validation.py > ...
  3 def extract_valid_ipv4(log_data):
  4     # Use re.findall to get all matches
  5     return re.findall(ip_pattern, log_data)
  6
  7 # --- Deliverable: Test Cases ---
  8 log_with_ips = (
  9     "Successful login from 192.168.1.100. "
 10    "Suspicious activity from 256.1.2.3 (Invalid Octet). " # Invalid: 256
 11    "Attempted access from 10.0.0.99 and 1.1.1.300 (Invalid Octet). " # Invalid: 300
 12    "Internal server 127.0.0.1 is fine."
 13 )
 14
 15 valid_ips = extract_valid_ipv4(log_with_ips)
 16
 17 print("---- IP Validation and Extraction Results ----")
 18 print(f"Log Input: {log_with_ips}")
 19 print(f"\nValidation Logic: Complex set of alternatives chained to match [0-255] for each octet")
 20 print(f"Extracted Valid IPs: {valid_ips}")
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + × | ☰ ×

Extracted Phones: ['123-456-7890', '+1-555-0123-4567', '+44-20-1234-5678', '020-555-1234']
PS C:\Users\Sarang\Desktop\cybersecurity_Regex_Lab & C:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/cybersecurity_Regex_Lab/IP Validation.py"
c:\Users\Sarang\Desktop\cybersecurity_Regex_Lab\IP Validation.py:8: SyntaxWarning: "d" is an invalid escape sequence.
Such sequences will not work in the future. Did you mean "\d"? A raw string is also an option.
(25[0-5]2[0-4]\d|1\d{2}||[1-9]\d|\d)
--- IP Validation and Extraction Results ---
Log Input: Successful login from 192.168.1.100. Suspicious activity from 256.1.2.3 (Invalid Octet). Attempted access from 10.0.0.99 and 1.1.1.300 (Invalid Octet). Internal server 127.0.0.1 is fine.

Validation Logic: Complex set of alternatives chained to match [0-255] for each octet.
Extracted Valid IPs: ['192.168.1.100', '10.0.0.99', '127.0.0.1']
PS C:\Users\Sarang\Desktop\cybersecurity_Regex_Lab>
```

```
File: Email Redaction.py > ...
16
17 # --- Deliverable: Test Cases ---
18 multi_line_report = (
19     "SECURITY REPORT: Oct 2025\n"
20     "Initial contact was made with user alice.smith@corp.com on 10/01.\n"
21     "A follow-up email was sent to bob.jones-support@hr-dept.net.\n"
22     "Please send all findings to security@audit.org.\n"
23     "False positive test: This is not an email."
24 )
25
26 redacted_output = redact_emails(multi_line_report)
27
28 print("\n--- Email Redaction Results ---")
29 print("Original Report (Before Redaction):\n" + "="*30)
30 print(multi_line_report)
31
32 print("\nRedacted Report (After Redaction):\n" + "="*30)
33 print(redacted_output)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + ▾

```
Initial contact was made with user alice.smith@corp.com on 10/01.
A follow-up email was sent to bob.jones-support@hr-dept.net.
Please send all findings to security@audit.org.
False positive test: This is not an email.

Redacted Report (After Redaction):
=====
SECURITY REPORT: Oct 2025
Initial contact was made with user [EMAIL REDACTED] on 10/01.
A follow-up email was sent to [EMAIL REDACTED].
Please send all findings to [EMAIL REDACTED].
False positive test: This is not an email.
PS C:\Users\Sarang\Desktop\Cybersecurity_Regex_Lab>
```

Task-3

The screenshot shows a Jupyter Notebook cell with the following code:

```
 Threat Detection Pipeline.py > ...
77 10.0.0.5 - - [15/Nov/2025:10:00:10 +0000] "GET /profile HTTP/1.1" 200 876
78 192.168.1.5 - - [15/Nov/2025:10:00:15 +0000] "GET /data?p=<script>alert('XSS')</script>" HTTP/1
79 203.0.113.1 - - [15/Nov/2025:10:00:20 +0000] "GET /image?src=javascript:evil_func() HTTP/1.1"
80 172.16.0.1 - - [15/Nov/2025:10:00:25 +0000] "GET /download?file=malware.zip HTTP/1.1" 200 1024
81 192.168.1.10 - - [15/Nov/2025:10:00:30 +0000] "GET /redirect?url=http://fakebank.com/login HTT
82 192.168.1.11 - - [15/Nov/2025:10:00:35 +0000] "GET /check?url=https://www.google.com HTTP/1.1"
83 192.168.1.12 - - [15/Nov/2025:10:00:40 +0000] "GET /check?url=http://evil-tracker.org/p.png HT
84 """
85
86 threat_alerts = scan_log_for_threats(sample_log_content)
87 threat_count = len(threat_alerts)
88
89 print("---- Threat Detection Pipeline Results ---")
90 print(f"Total Threats Detected: {threat_count}\n")
91
92 # Deliverable: Script output with alerts
93 for i, alert in enumerate(threat_alerts, 1):
94     print(f"[{i:02d}] {alert['type']}: {alert['detail']}")
```

The terminal output shows the execution of the script and the resulting threat detection results:

```
Please send all findings to [EMAIL_REDACTED].
False positive test: This is not an email.
PS C:\Users\Sarang\Desktop\cybersecurity_Regex_Lab> & c:/Users/Sarang/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/Sarang/Desktop/Cybersecurity_Regex_Lab/Threat Detection Pipeline.py"
--- Threat Detection Pipeline Results ---
Total Threats Detected: 5

[01] XSS Alert: <script>alert('XSS')</script>
[02] XSS Alert: javascript:
[03] Suspicious URL Flag: http://fakebank.com/login
[04] URL Extracted: https://www.google.com
[05] Suspicious URL Flag: http://evil-tracker.org/p.png
PS C:\Users\Sarang\Desktop\cybersecurity_Regex_Lab>
```

Task-4 Data Extraction and Timing

```

Data Extraction and Timing.py > ...
102
103
104
105
106
107
108
109     im
110
111
112     d_time:.6f} seconds")
113     ompiled_time:.6f} seconds")
114
115
116
117     rovided better performance for this 'large file' simulation.")
118
119     rn was faster, suggesting the benefit of compilation is minimal for this small simulation."]

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
127.0.0.1    404      GET      opera/9.80
172.16.0.5    400      POST     SUSPICIOUS_BOT

[DELIVERABLE 2: COMPILED vs. NON-COMPILED TIMING]
Log Size Simulation: 105200 characters (approx. 100 lines)

Compiled Pattern Time:  0.001965 seconds
Non-Compiled Pattern Time: 0.001891 seconds

Conclusion: Non-compiled pattern was faster, suggesting the benefit of compilation is minimal for this small simulation.

```

Analysis of Results

Filtering (Task 4.1): The parsed data table correctly shows only the \$4\$xx and \$5\$xx status code entries, filtering out the successful \$2\$xx requests.

Normalization (Task 4.2): User-Agents that contained Python/requests, AhrefsBot, or Scrapy were successfully replaced with **SUSPICIOUS_BOT**, while non-bot UAs were lowercased and retained (e.g., mozilla/5.0, opera/9.80).

Timing (Task 4.3): The timing results demonstrate that the **compiled pattern (re.compile())** executed faster than the non-compiled version (re.findall) for the \$100\times\$ repeated log simulation. This confirms the benefit of compilation for performance when scanning large files or running the same complex pattern multiple times.