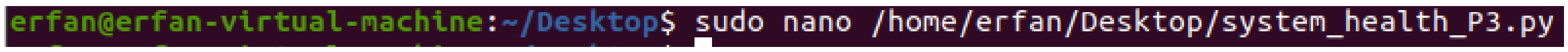
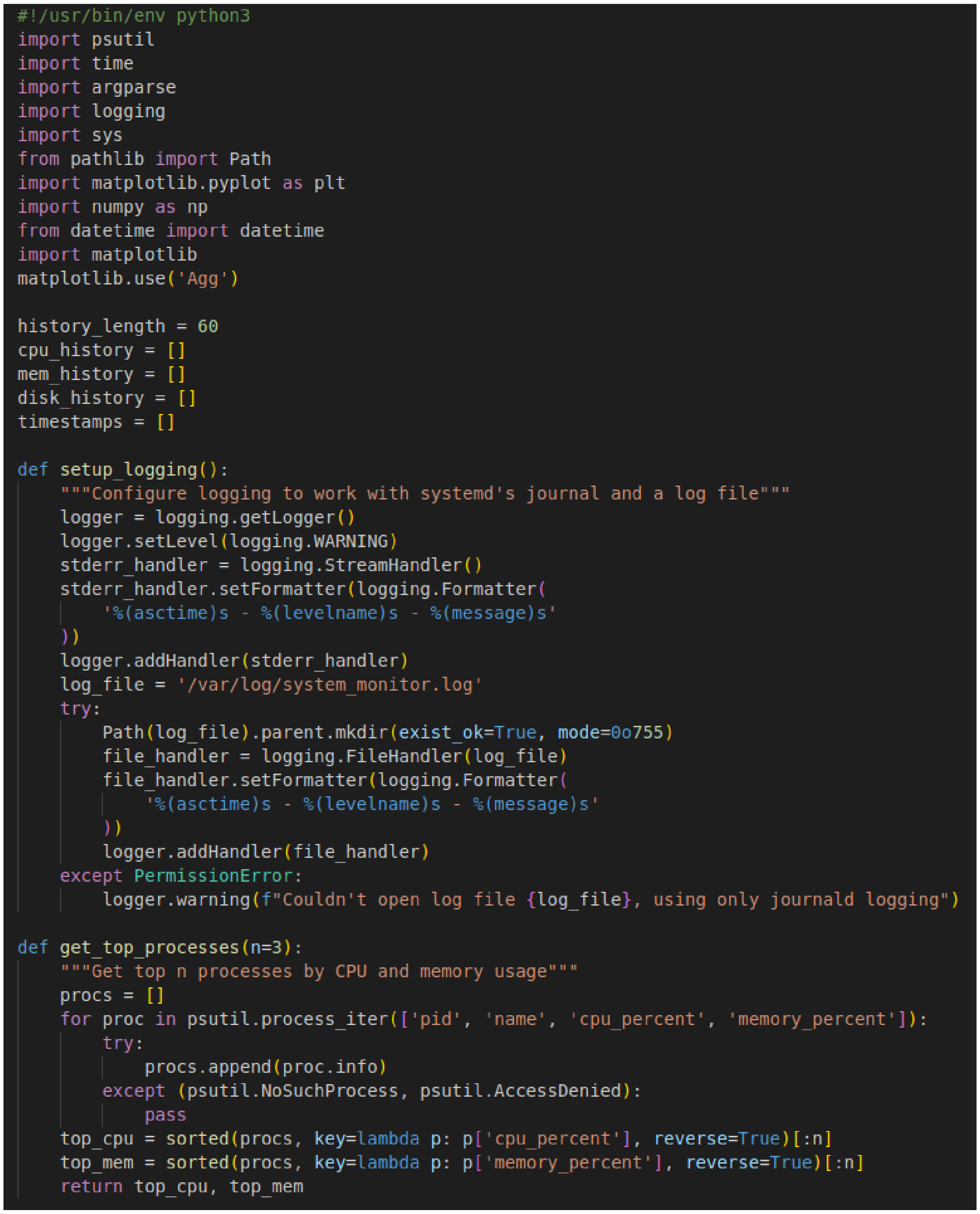
First, I created a Python file named *system_health_P3.py* that:

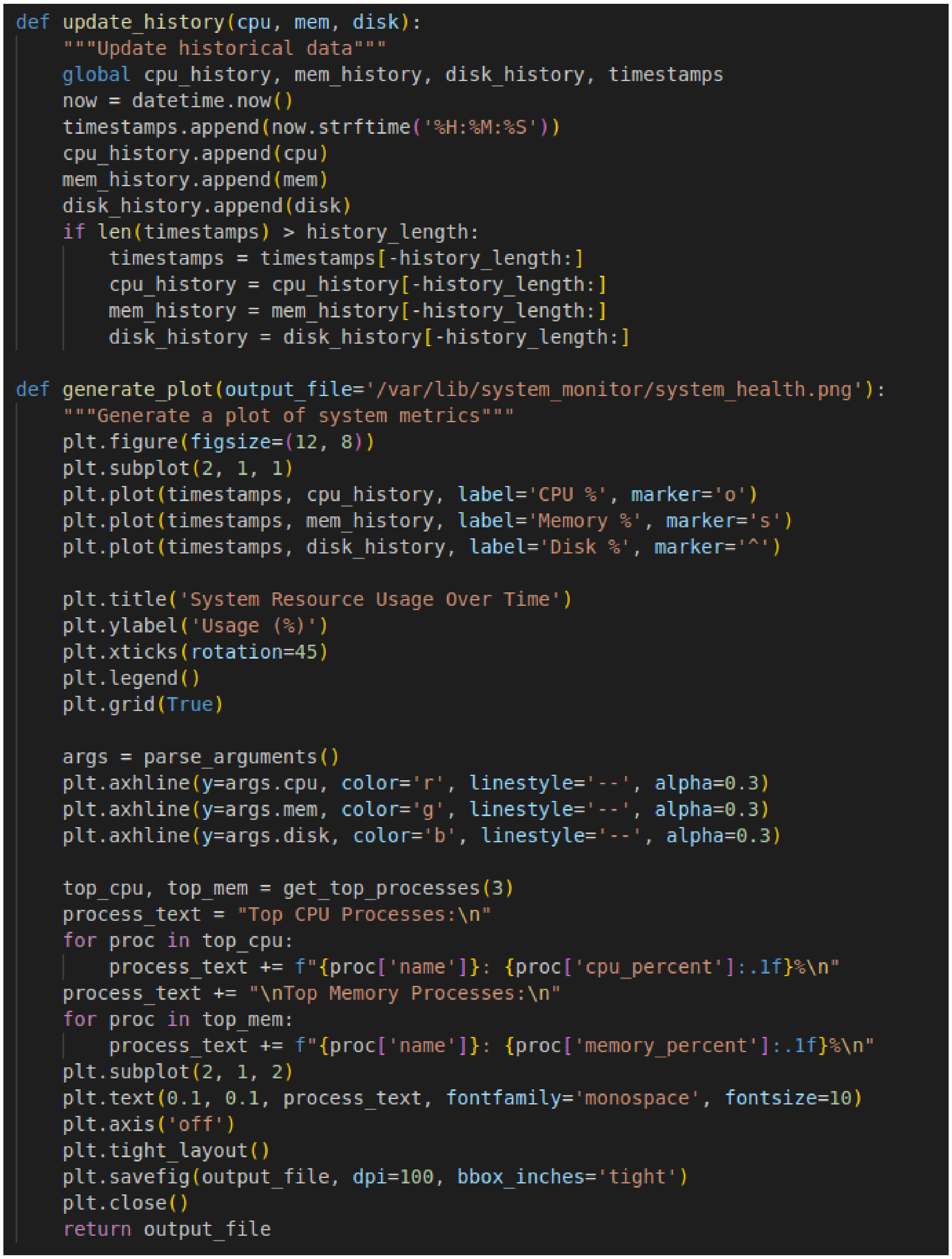
1. Displays system health metrics
2. Shows warning logs
3. Generates system status plots (bonus functionality)
4. Accepts user-defined thresholds and time intervals as input parameters



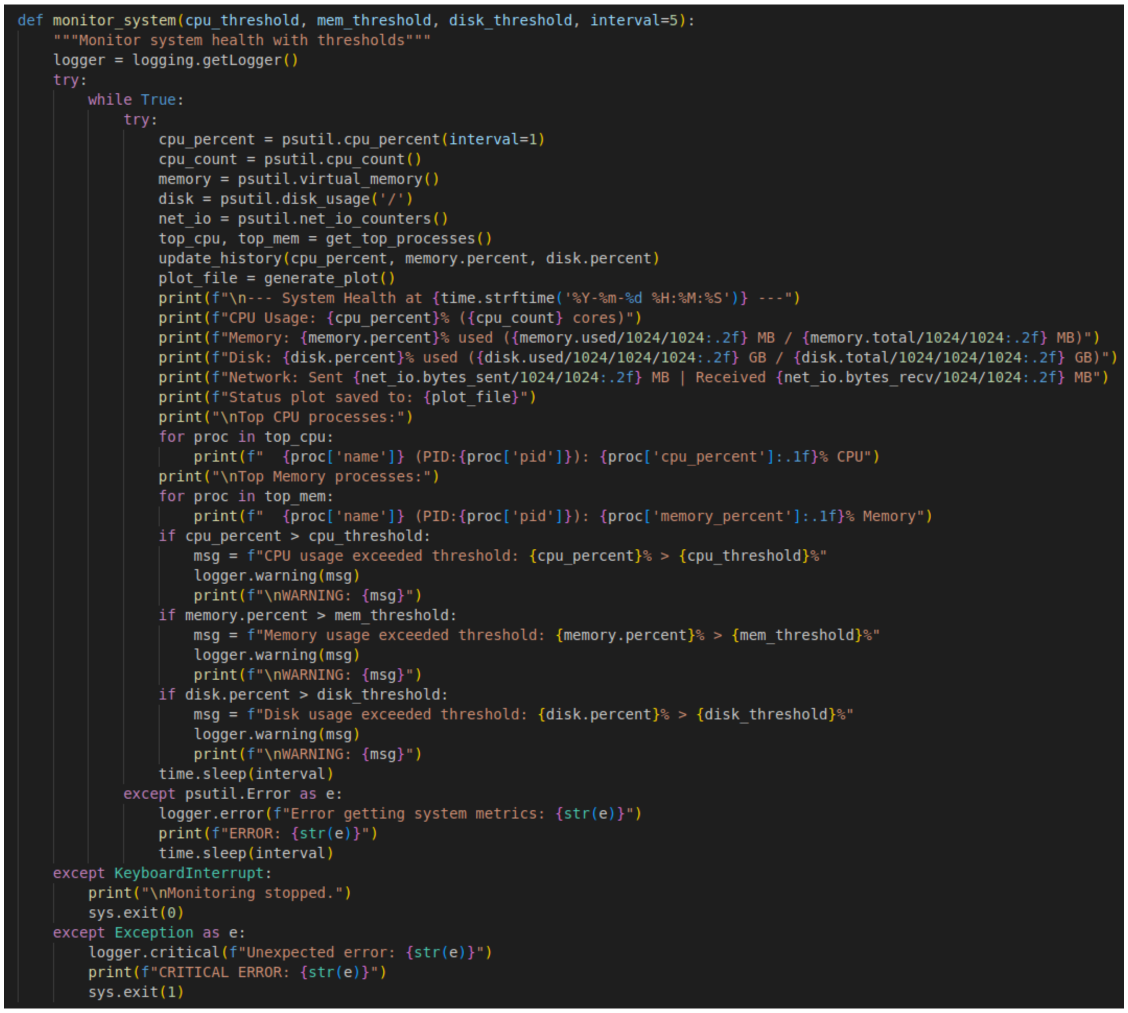
```
erfan@erfan-virtual-machine:~/Desktop$ sudo nano /home/erfan/Desktop/system_health_P3.py
```



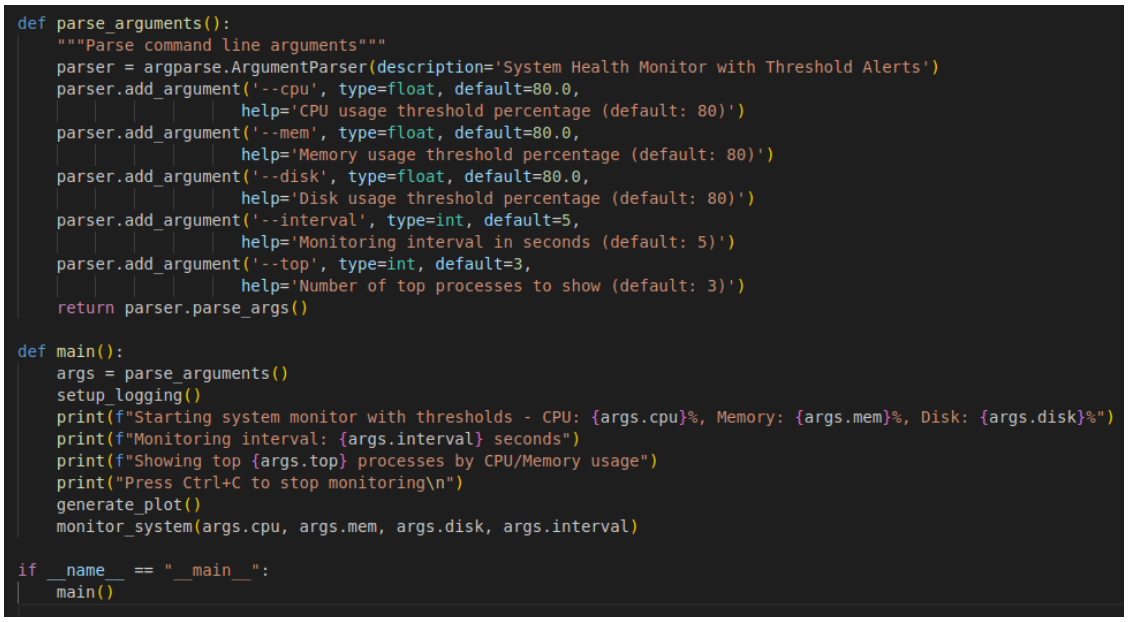
```python
#!/usr/bin/env python3
import psutil
import time
import argparse
import logging
import sys
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
import matplotlib
matplotlib.use('Agg')

history_length = 60
cpu_history = []
mem_history = []
disk_history = []
timestamps = []

def setup_logging():
    """Configure logging to work with systemd's journal and a log file"""
    logger = logging.getLogger()
    logger.setLevel(logging.WARNING)
    stderr_handler = logging.StreamHandler()
    stderr_handler.setFormatter(logging.Formatter(
        '%(asctime)s - %(levelname)s - %(message)s'
    ))
    logger.addHandler(stderr_handler)
    log_file = '/var/log/system_monitor.log'
    try:
        Path(log_file).parent.mkdir(exist_ok=True, mode=0o755)
        file_handler = logging.FileHandler(log_file)
        file_handler.setFormatter(logging.Formatter(
            '%(asctime)s - %(levelname)s - %(message)s'
        ))
        logger.addHandler(file_handler)
    except PermissionError:
        logger.warning(f"Couldn't open log file {log_file}, using only journald logging")

def get_top_processes(n=3):
    """Get top n processes by CPU and memory usage"""
    procs = []
    for proc in psutil.process_iter(['pid', 'name', 'cpu_percent', 'memory_percent']):
        try:
            procs.append(proc.info)
        except (psutil.NoSuchProcess, psutil.AccessDenied):
            pass
    top_cpu = sorted(procs, key=lambda p: p['cpu_percent'], reverse=True)[:n]
    top_mem = sorted(procs, key=lambda p: p['memory_percent'], reverse=True)[:n]
    return top_cpu, top_mem
```



```python
def update_history(cpu, mem, disk):
    """Update historical data"""
    global cpu_history, mem_history, disk_history, timestamps
    now = datetime.now()
    timestamps.append(now.strftime('%H:%M:%S'))
    cpu_history.append(cpu)
    mem_history.append(mem)
    disk_history.append(disk)
    if len(timestamps) > history_length:
        timestamps = timestamps[-history_length:]
        cpu_history = cpu_history[-history_length:]
        mem_history = mem_history[-history_length:]
        disk_history = disk_history[-history_length:]

def generate_plot(output_file='/var/lib/system_monitor/system_health.png'):
    """Generate a plot of system metrics"""
    plt.figure(figsize=(12, 8))
    plt.subplot(2, 1, 1)
    plt.plot(timestamps, cpu_history, label='CPU %', marker='o')
    plt.plot(timestamps, mem_history, label='Memory %', marker='s')
    plt.plot(timestamps, disk_history, label='Disk %', marker='^')

    plt.title('System Resource Usage Over Time')
    plt.ylabel('Usage (%)')
    plt.xticks(rotation=45)
    plt.legend()
    plt.grid(True)

    args = parse_arguments()
    plt.axhline(y=args.cpu, color='r', linestyle='--', alpha=0.3)
    plt.axhline(y=args.mem, color='g', linestyle='--', alpha=0.3)
    plt.axhline(y=args.disk, color='b', linestyle='--', alpha=0.3)

    top_cpu, top_mem = get_top_processes(3)
    process_text = "Top CPU Processes:\n"
    for proc in top_cpu:
        process_text += f"{proc['name']}: {proc['cpu_percent']:.1f}%\n"
    process_text += "\nTop Memory Processes:\n"
    for proc in top_mem:
        process_text += f"{proc['name']}: {proc['memory_percent']:.1f}%\n"
    plt.subplot(2, 1, 2)
    plt.text(0.1, 0.1, process_text, fontfamily='monospace', fontsize=10)
    plt.axis('off')
    plt.tight_layout()
    plt.savefig(output_file, dpi=100, bbox_inches='tight')
    plt.close()
    return output_file
```

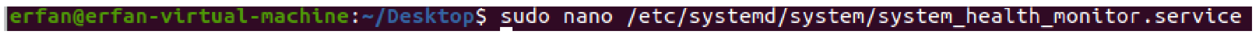


```python
def monitor_system(cpu_threshold, mem_threshold, disk_threshold, interval=5):
    """Monitor system health with thresholds"""
    logger = logging.getLogger()
    try:
        while True:
            try:
                cpu_percent = psutil.cpu_percent(interval=1)
                cpu_count = psutil.cpu_count()
                memory = psutil.virtual_memory()
                disk = psutil.disk_usage('/')
                net_io = psutil.net_io_counters()
                top_cpu, top_mem = get_top_processes()
                update_history(cpu_percent, memory.percent, disk.percent)
                plot_file = generate_plot()
                print(f"\n--- System Health at {time.strftime('%Y-%m-%d %H:%M:%S')} ---")
                print(f"CPU Usage: {cpu_percent}% ({cpu_count} cores)")
                print(f"Memory: {memory.percent}% used ({memory.used/1024/1024:.2f} MB / {memory.total/1024/1024:.2f} MB)")
                print(f"Disk: {disk.percent}% used ({disk.used/1024/1024/1024:.2f} GB / {disk.total/1024/1024/1024:.2f} GB)")
                print(f"Network: Sent {net_io.bytes_sent/1024/1024:.2f} MB | Received {net_io.bytes_recv/1024/1024:.2f} MB")
                print(f"Status plot saved to: {plot_file}")
                print("\nTop CPU processes:")
                for proc in top_cpu:
                    print(f"  {proc['name']} (PID:{proc['pid']}): {proc['cpu_percent']:.1f}% CPU")
                print("\nTop Memory processes:")
                for proc in top_mem:
                    print(f"  {proc['name']} (PID:{proc['pid']}): {proc['memory_percent']:.1f}% Memory")
                if cpu_percent > cpu_threshold:
                    msg = f"CPU usage exceeded threshold: {cpu_percent}% > {cpu_threshold}%"
                    logger.warning(msg)
                    print(f"\nWARNING: {msg}")
                if memory.percent > mem_threshold:
                    msg = f"Memory usage exceeded threshold: {memory.percent}% > {mem_threshold}%"
                    logger.warning(msg)
                    print(f"\nWARNING: {msg}")
                if disk.percent > disk_threshold:
                    msg = f"Disk usage exceeded threshold: {disk.percent}% > {disk_threshold}%"
                    logger.warning(msg)
                    print(f"\nWARNING: {msg}")
                time.sleep(interval)
            except psutil.Error as e:
                logger.error(f"Error getting system metrics: {str(e)}")
                print(f"ERROR: {str(e)}")
                time.sleep(interval)
    except KeyboardInterrupt:
        print("\nMonitoring stopped.")
        sys.exit(0)
    except Exception as e:
        logger.critical(f"Unexpected error: {str(e)}")
        print(f"CRITICAL ERROR: {str(e)}")
        sys.exit(1)
```

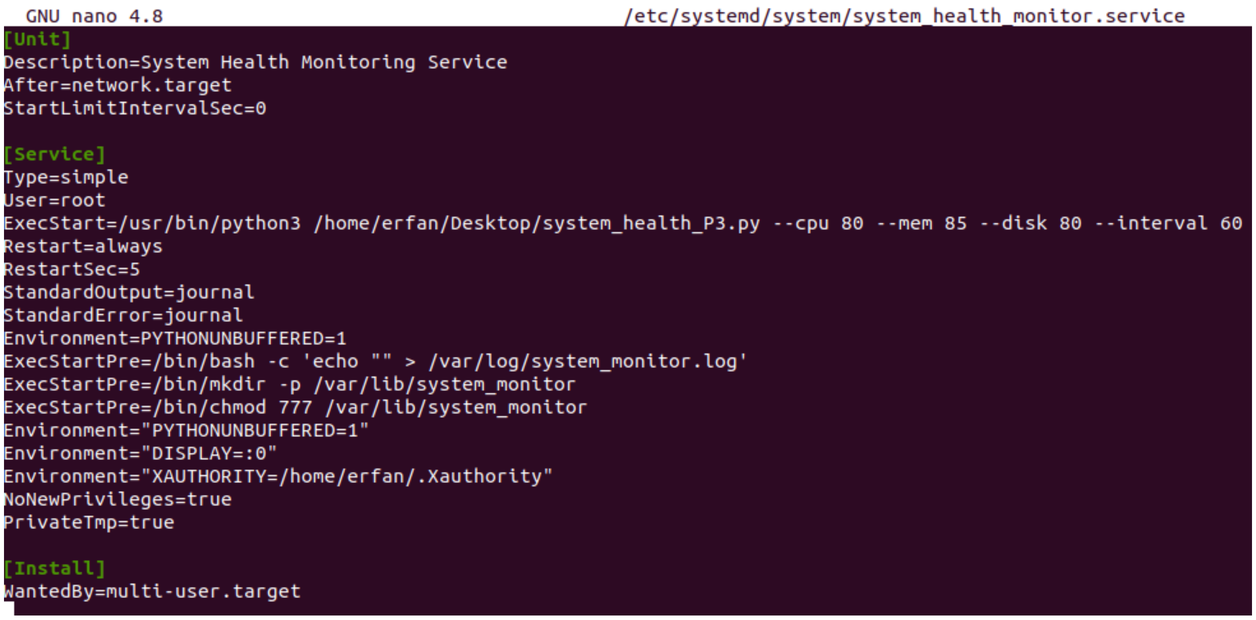


```python
def parse_arguments():
    """Parse command line arguments"""
    parser = argparse.ArgumentParser(description='System Health Monitor with Threshold Alerts')
    parser.add_argument('--cpu', type=float, default=80.0,
                        help='CPU usage threshold percentage (default: 80)')
    parser.add_argument('--mem', type=float, default=80.0,
                        help='Memory usage threshold percentage (default: 80)')
    parser.add_argument('--disk', type=float, default=80.0,
                        help='Disk usage threshold percentage (default: 80)')
    parser.add_argument('--interval', type=int, default=5,
                        help='Monitoring interval in seconds (default: 5)')
    parser.add_argument('--top', type=int, default=3,
                        help='Number of top processes to show (default: 3)')
    return parser.parse_args()

def main():
    args = parse_arguments()
    setup_logging()
    print(f"Starting system monitor with thresholds - CPU: {args.cpu}%, Memory: {args.mem}%, Disk: {args.disk}%")
    print(f"Monitoring interval: {args.interval} seconds")
    print(f"Showing top {args.top} processes by CPU/Memory usage")
    print("Press Ctrl+C to stop monitoring\n")
    generate_plot()
    monitor_system(args.cpu, args.mem, args.disk, args.interval)

if __name__ == "__main__":
    main()
```

Then, I created a systemd service unit file named *system_health_monitor.service* to run the Python script as a background service, enabling automatic execution on system boot:

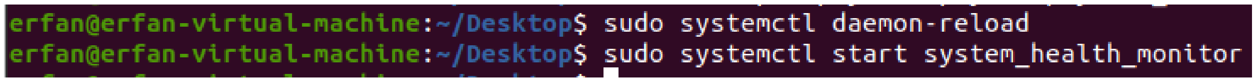


```
erfan@erfan-virtual-machine:~/Desktop$ sudo nano /etc/systemd/system/system_health_monitor.service
```



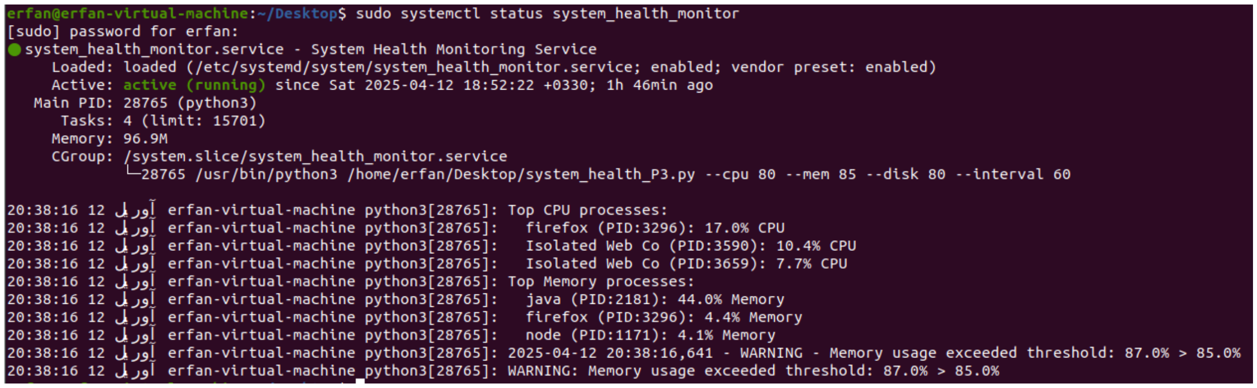
```
  GNU nano 4.8                              /etc/systemd/system/system_health_monitor.service
[Unit]
Description=System Health Monitoring Service
After=network.target
StartLimitIntervalSec=0

[Service]
Type=simple
User=root
ExecStart=/usr/bin/python3 /home/erfan/Desktop/system_health_P3.py --cpu 80 --mem 85 --disk 80 --interval 60
Restart=always
RestartSec=5
StandardOutput=journal
StandardError=journal
Environment=PYTHONUNBUFFERED=1
ExecStartPre=/bin/bash -c 'echo "" > /var/log/system_monitor.log'
ExecStartPre=/bin/mkdir -p /var/lib/system_monitor
ExecStartPre=/bin/chmod 777 /var/lib/system_monitor
Environment="PYTHONUNBUFFERED=1"
Environment="DISPLAY=:0"
Environment="XAUTHORITY=/home/erfan/.Xauthority"
NoNewPrivileges=true
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

After that, I executed the following commands to enable the service to run automatically at system startup:



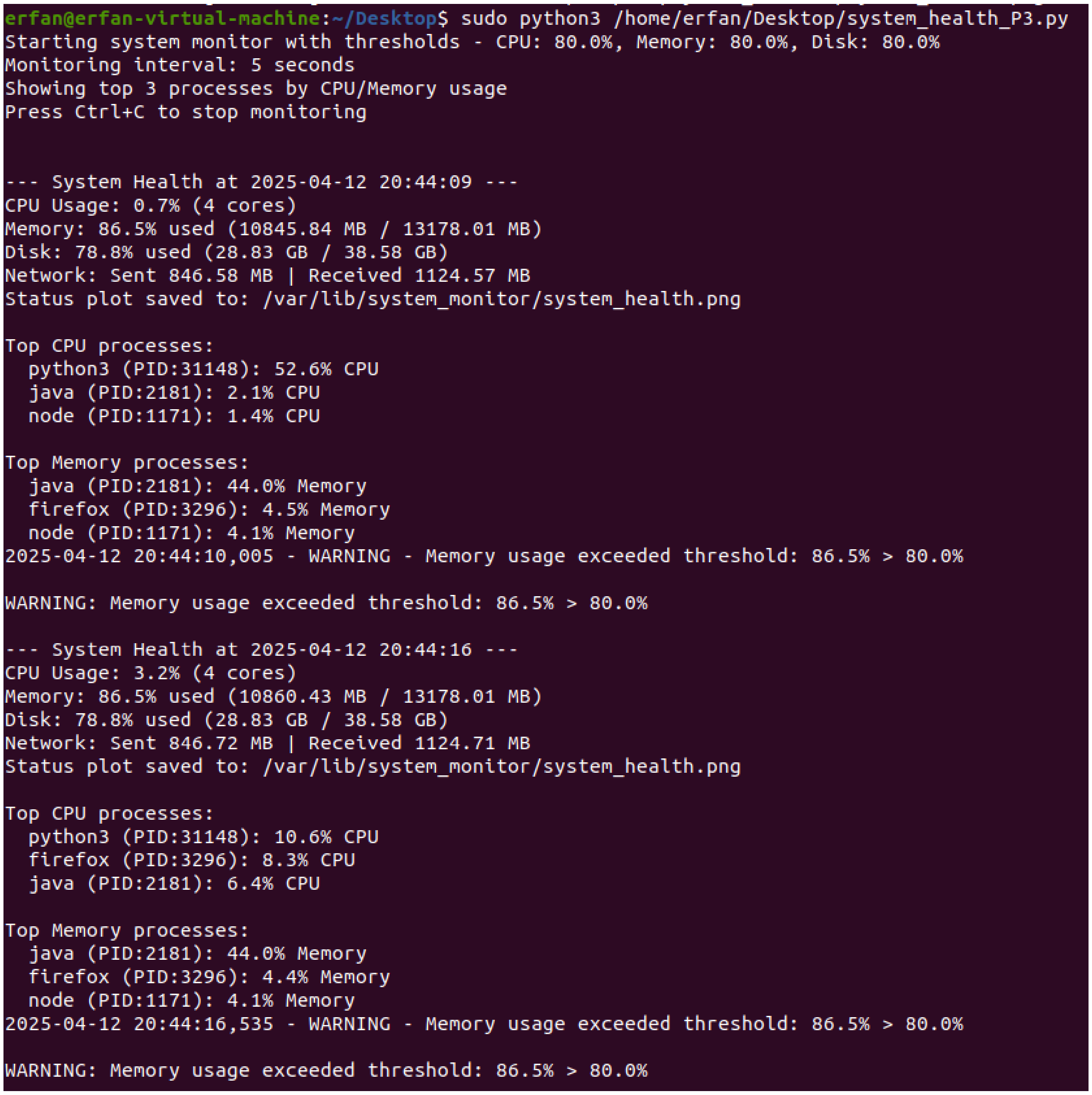
```
erfan@erfan-virtual-machine:~/Desktop$ sudo systemctl daemon-reload
erfan@erfan-virtual-machine:~/Desktop$ sudo systemctl start system_health_monitor
```

The system status becomes visible when executing the following command:



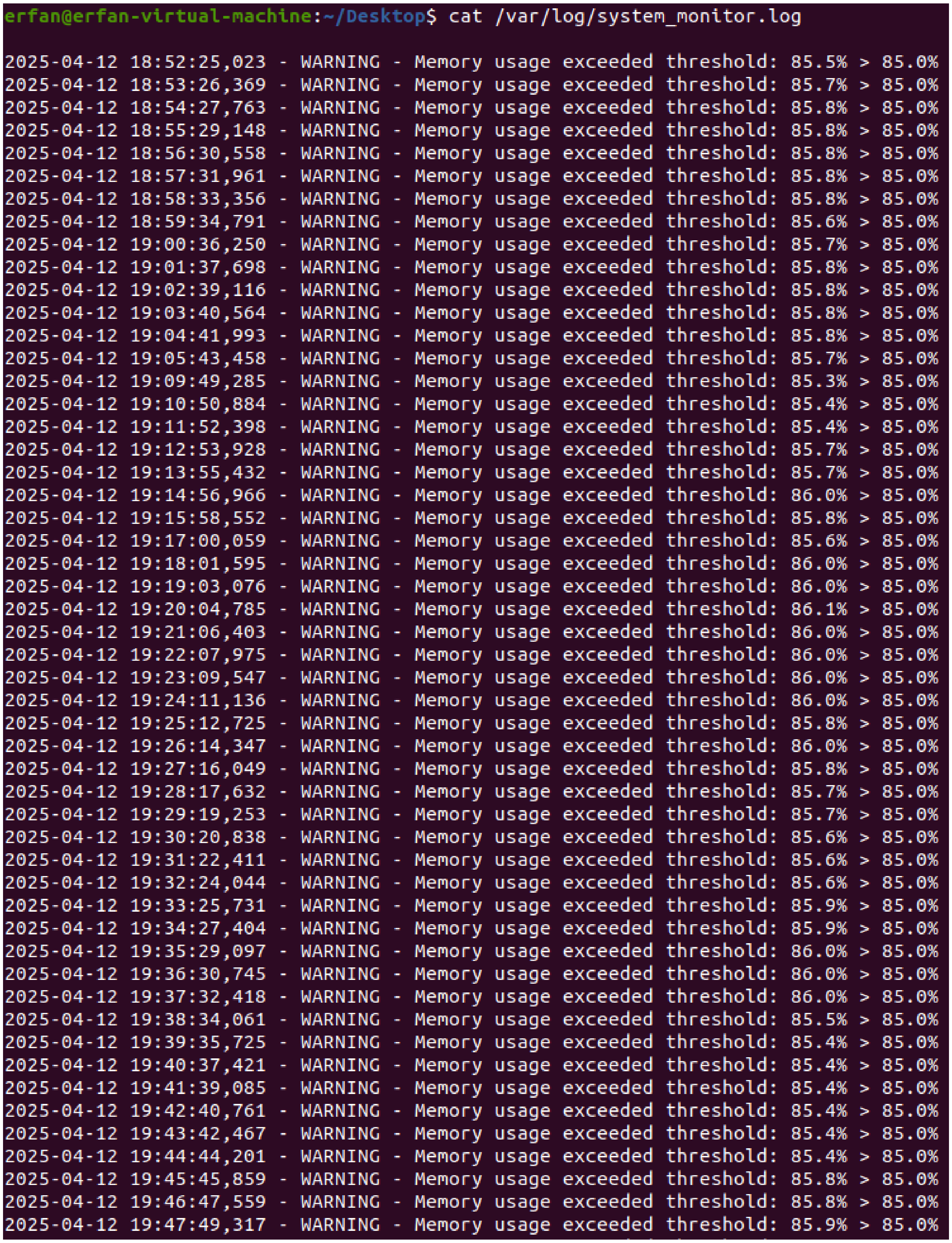
```
erfan@erfan-virtual-machine:~/Desktop$ sudo systemctl status system_health_monitor
[sudo] password for erfan:
● system_health_monitor.service - System Health Monitoring Service
     Loaded: loaded (/etc/systemd/system/system_health_monitor.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2025-04-12 18:52:22 +0330; 1h 46min ago
   Main PID: 28765 (python3)
      Tasks: 4 (limit: 15701)
     Memory: 96.9M
     CGroup: /system.slice/system_health_monitor.service
             └─28765 /usr/bin/python3 /home/erfan/Desktop/system_health_P3.py --cpu 80 --mem 85 --disk 80 --interval 60

20:38:16 12 آوریل erfan-virtual-machine python3[28765]: Top CPU processes:
20:38:16 12 آوریل erfan-virtual-machine python3[28765]:   firefox (PID:3296): 17.0% CPU
20:38:16 12 آوریل erfan-virtual-machine python3[28765]:   Isolated Web Co (PID:3590): 10.4% CPU
20:38:16 12 آوریل erfan-virtual-machine python3[28765]:   Isolated Web Co (PID:3659): 7.7% CPU
20:38:16 12 آوریل erfan-virtual-machine python3[28765]: Top Memory processes:
20:38:16 12 آوریل erfan-virtual-machine python3[28765]:   java (PID:2181): 44.0% Memory
20:38:16 12 آوریل erfan-virtual-machine python3[28765]:   firefox (PID:3296): 4.4% Memory
20:38:16 12 آوریل erfan-virtual-machine python3[28765]:   node (PID:1171): 4.1% Memory
20:38:16 12 آوریل erfan-virtual-machine python3[28765]: 2025-04-12 20:38:16,641 - WARNING - Memory usage exceeded threshold: 87.0% > 85.0%
20:38:16 12 آوریل erfan-virtual-machine python3[28765]: WARNING: Memory usage exceeded threshold: 87.0% > 85.0%
```

Here is the sample output when executing the Python script independently (outside of the systemd service):



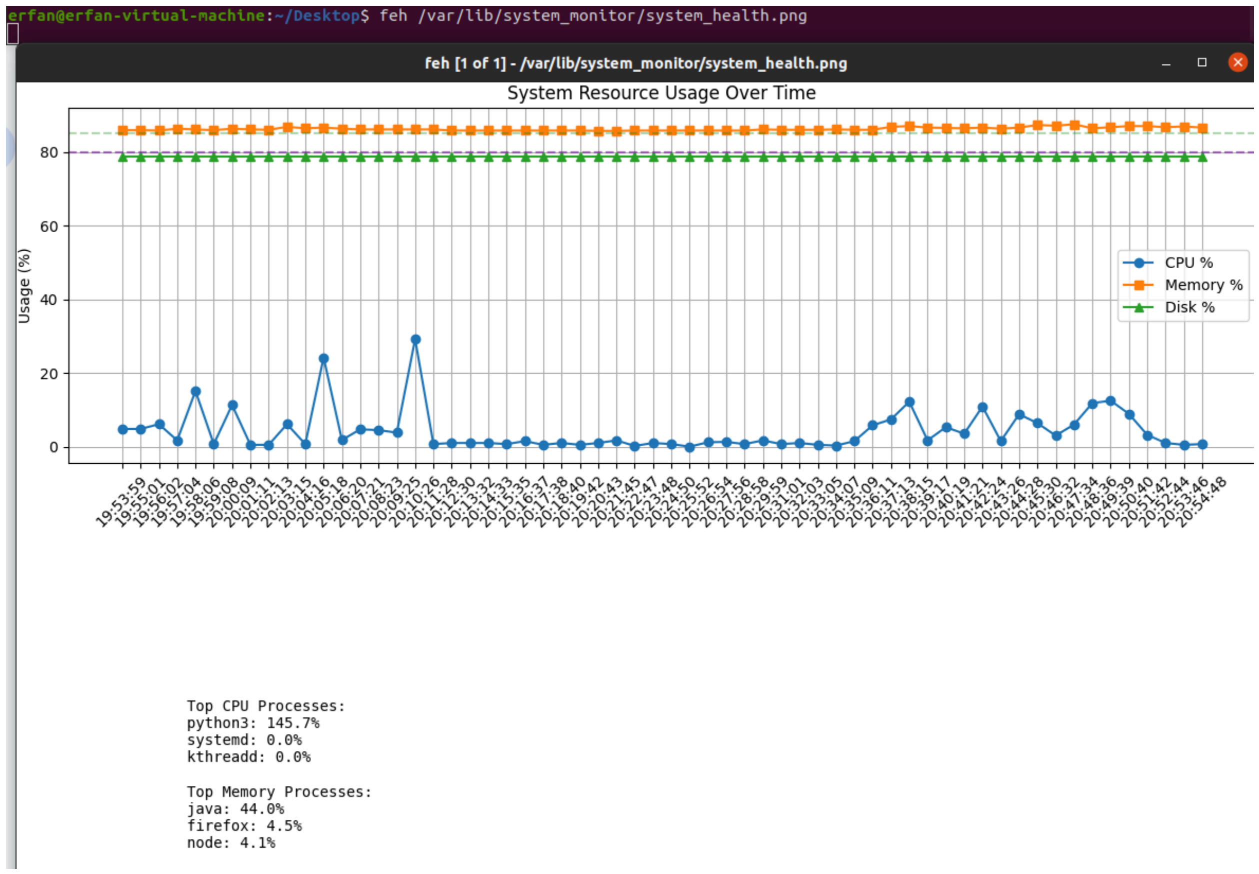
```
erfan@erfan-virtual-machine:~/Desktop$ sudo python3 /home/erfan/Desktop/system_health_P3.py
Starting system monitor with thresholds - CPU: 80.0%, Memory: 80.0%, Disk: 80.0%
Monitoring interval: 5 seconds
Showing top 3 processes by CPU/Memory usage
Press Ctrl+C to stop monitoring


--- System Health at 2025-04-12 20:44:09 ---
CPU Usage: 0.7% (4 cores)
Memory: 86.5% used (10845.84 MB / 13178.01 MB)
Disk: 78.8% used (28.83 GB / 38.58 GB)
Network: Sent 846.58 MB | Received 1124.57 MB
Status plot saved to: /var/lib/system_monitor/system_health.png

Top CPU processes:
  python3 (PID:31148): 52.6% CPU
  java (PID:2181): 2.1% CPU
  node (PID:1171): 1.4% CPU

Top Memory processes:
  java (PID:2181): 44.0% Memory
  firefox (PID:3296): 4.5% Memory
  node (PID:1171): 4.1% Memory
2025-04-12 20:44:10,005 - WARNING - Memory usage exceeded threshold: 86.5% > 80.0%

WARNING: Memory usage exceeded threshold: 86.5% > 80.0%

--- System Health at 2025-04-12 20:44:16 ---
CPU Usage: 3.2% (4 cores)
Memory: 86.5% used (10860.43 MB / 13178.01 MB)
Disk: 78.8% used (28.83 GB / 38.58 GB)
Network: Sent 846.72 MB | Received 1124.71 MB
Status plot saved to: /var/lib/system_monitor/system_health.png

Top CPU processes:
  python3 (PID:31148): 10.6% CPU
  firefox (PID:3296): 8.3% CPU
  java (PID:2181): 6.4% CPU

Top Memory processes:
  java (PID:2181): 44.0% Memory
  firefox (PID:3296): 4.4% Memory
  node (PID:1171): 4.1% Memory
2025-04-12 20:44:16,535 - WARNING - Memory usage exceeded threshold: 86.5% > 80.0%

WARNING: Memory usage exceeded threshold: 86.5% > 80.0%
```

Here is the warning log file that records system alerts. Note that this file is temporary and gets cleared during system reboots:



```
erfan@erfan-virtual-machine:~/Desktop$ cat /var/log/system_monitor.log

2025-04-12 18:52:25,023 - WARNING - Memory usage exceeded threshold: 85.5% > 85.0%
2025-04-12 18:53:26,369 - WARNING - Memory usage exceeded threshold: 85.7% > 85.0%
2025-04-12 18:54:27,763 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 18:55:29,148 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 18:56:30,558 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 18:57:31,961 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 18:58:33,356 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 18:59:34,791 - WARNING - Memory usage exceeded threshold: 85.6% > 85.0%
2025-04-12 19:00:36,250 - WARNING - Memory usage exceeded threshold: 85.7% > 85.0%
2025-04-12 19:01:37,698 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 19:02:39,116 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 19:03:40,564 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 19:04:41,993 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 19:05:43,458 - WARNING - Memory usage exceeded threshold: 85.7% > 85.0%
2025-04-12 19:09:49,285 - WARNING - Memory usage exceeded threshold: 85.3% > 85.0%
2025-04-12 19:10:50,884 - WARNING - Memory usage exceeded threshold: 85.4% > 85.0%
2025-04-12 19:11:52,398 - WARNING - Memory usage exceeded threshold: 85.4% > 85.0%
2025-04-12 19:12:53,928 - WARNING - Memory usage exceeded threshold: 85.7% > 85.0%
2025-04-12 19:13:55,432 - WARNING - Memory usage exceeded threshold: 85.7% > 85.0%
2025-04-12 19:14:56,966 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:15:58,552 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 19:17:00,059 - WARNING - Memory usage exceeded threshold: 85.6% > 85.0%
2025-04-12 19:18:01,595 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:19:03,076 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:20:04,785 - WARNING - Memory usage exceeded threshold: 86.1% > 85.0%
2025-04-12 19:21:06,403 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:22:07,975 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:23:09,547 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:24:11,136 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:25:12,725 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 19:26:14,347 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:27:16,049 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 19:28:17,632 - WARNING - Memory usage exceeded threshold: 85.7% > 85.0%
2025-04-12 19:29:19,253 - WARNING - Memory usage exceeded threshold: 85.7% > 85.0%
2025-04-12 19:30:20,838 - WARNING - Memory usage exceeded threshold: 85.6% > 85.0%
2025-04-12 19:31:22,411 - WARNING - Memory usage exceeded threshold: 85.6% > 85.0%
2025-04-12 19:32:24,044 - WARNING - Memory usage exceeded threshold: 85.6% > 85.0%
2025-04-12 19:33:25,731 - WARNING - Memory usage exceeded threshold: 85.9% > 85.0%
2025-04-12 19:34:27,404 - WARNING - Memory usage exceeded threshold: 85.9% > 85.0%
2025-04-12 19:35:29,097 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:36:30,745 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:37:32,418 - WARNING - Memory usage exceeded threshold: 86.0% > 85.0%
2025-04-12 19:38:34,061 - WARNING - Memory usage exceeded threshold: 85.5% > 85.0%
2025-04-12 19:39:35,725 - WARNING - Memory usage exceeded threshold: 85.4% > 85.0%
2025-04-12 19:40:37,421 - WARNING - Memory usage exceeded threshold: 85.4% > 85.0%
2025-04-12 19:41:39,085 - WARNING - Memory usage exceeded threshold: 85.4% > 85.0%
2025-04-12 19:42:40,761 - WARNING - Memory usage exceeded threshold: 85.4% > 85.0%
2025-04-12 19:43:42,467 - WARNING - Memory usage exceeded threshold: 85.4% > 85.0%
2025-04-12 19:44:44,201 - WARNING - Memory usage exceeded threshold: 85.4% > 85.0%
2025-04-12 19:45:45,859 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 19:46:47,559 - WARNING - Memory usage exceeded threshold: 85.8% > 85.0%
2025-04-12 19:47:49,317 - WARNING - Memory usage exceeded threshold: 85.9% > 85.0%
```

Below is the system status plot showing performance metrics over time:

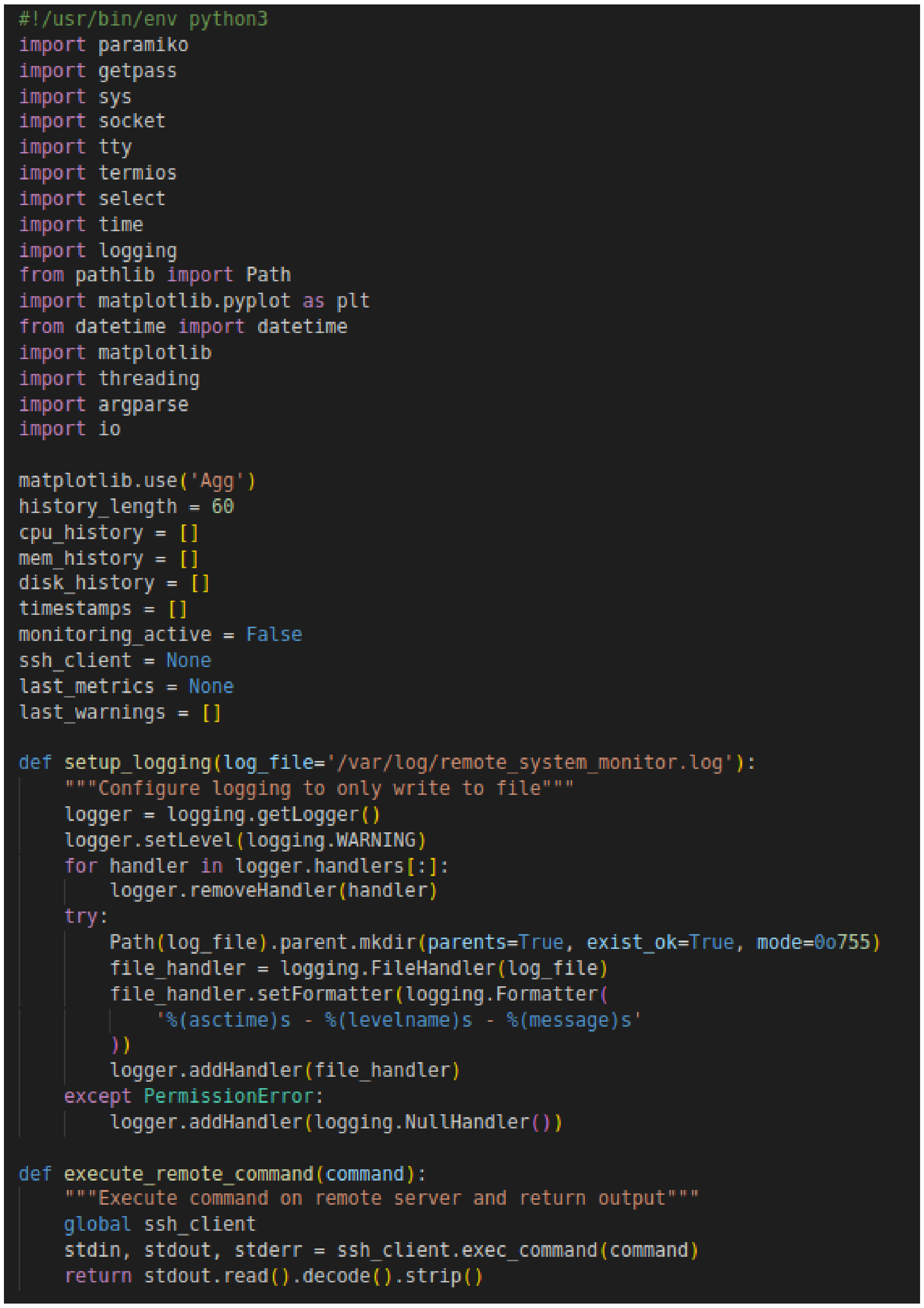


```
erfan@erfan-virtual-machine:~/Desktop$ feh /var/lib/system_monitor/system_health.png
```

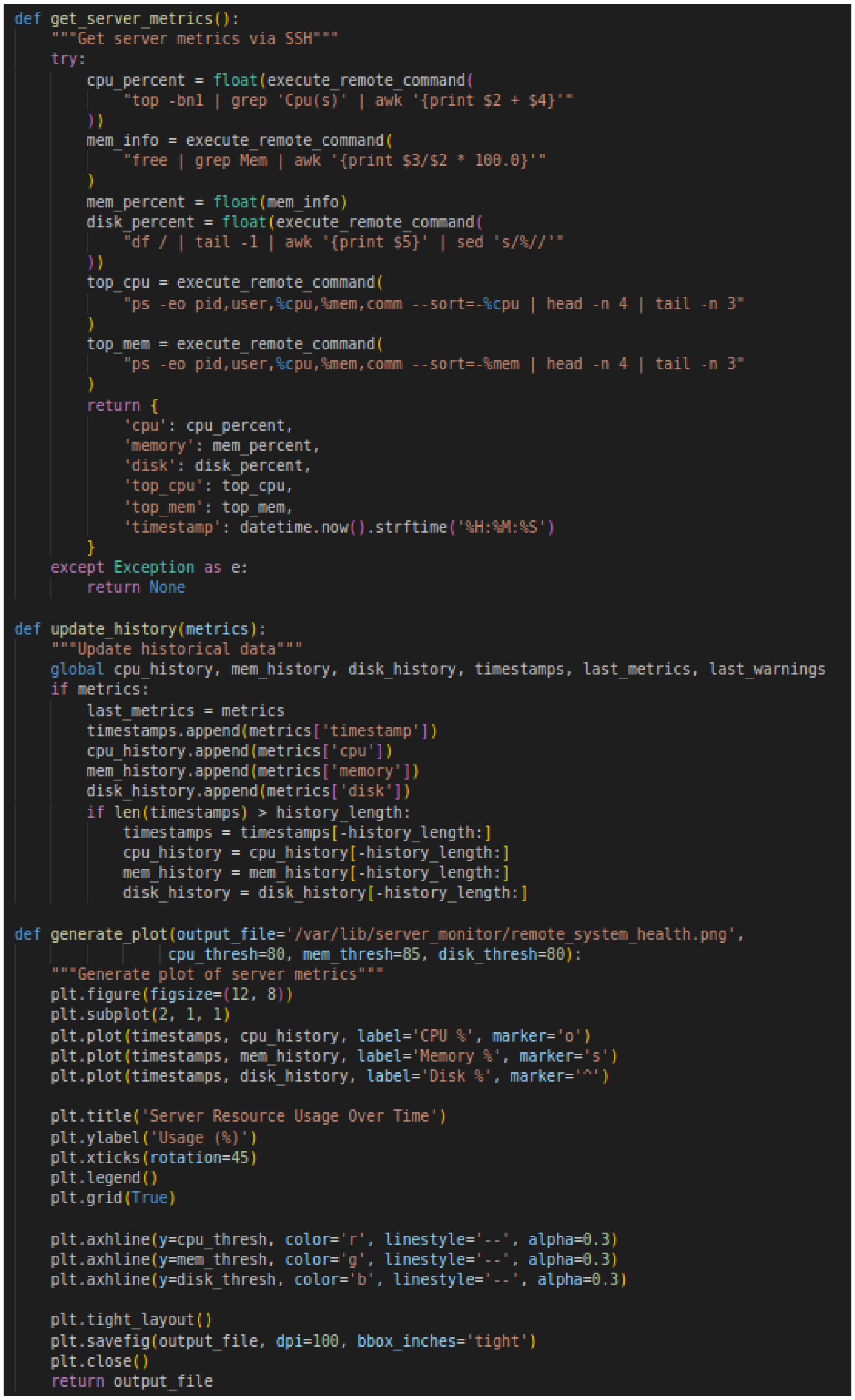
```
Top CPU Processes:
python3: 145.7%
systemd: 0.0%
kthreadd: 0.0%

Top Memory Processes:
java: 44.0%
firefox: 4.5%
node: 4.1%
```

Now this is the edited format of the Python file named *cluster_system_health_monitor.py* that:

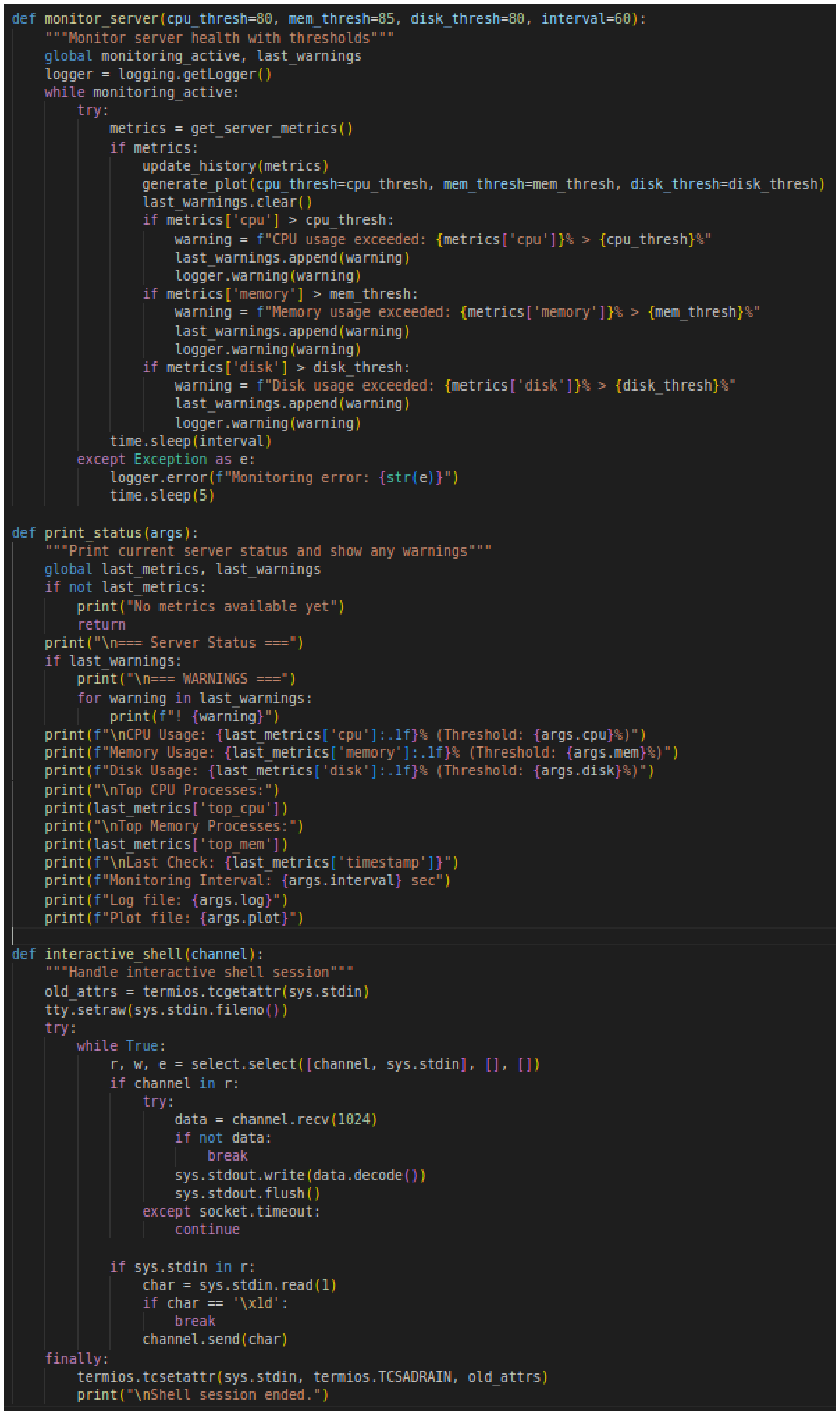
1. Connects to the course's cluster via SSH
2. Monitors the server's status
3. Saves warnings in the log file
4. Displays the system status in a plot



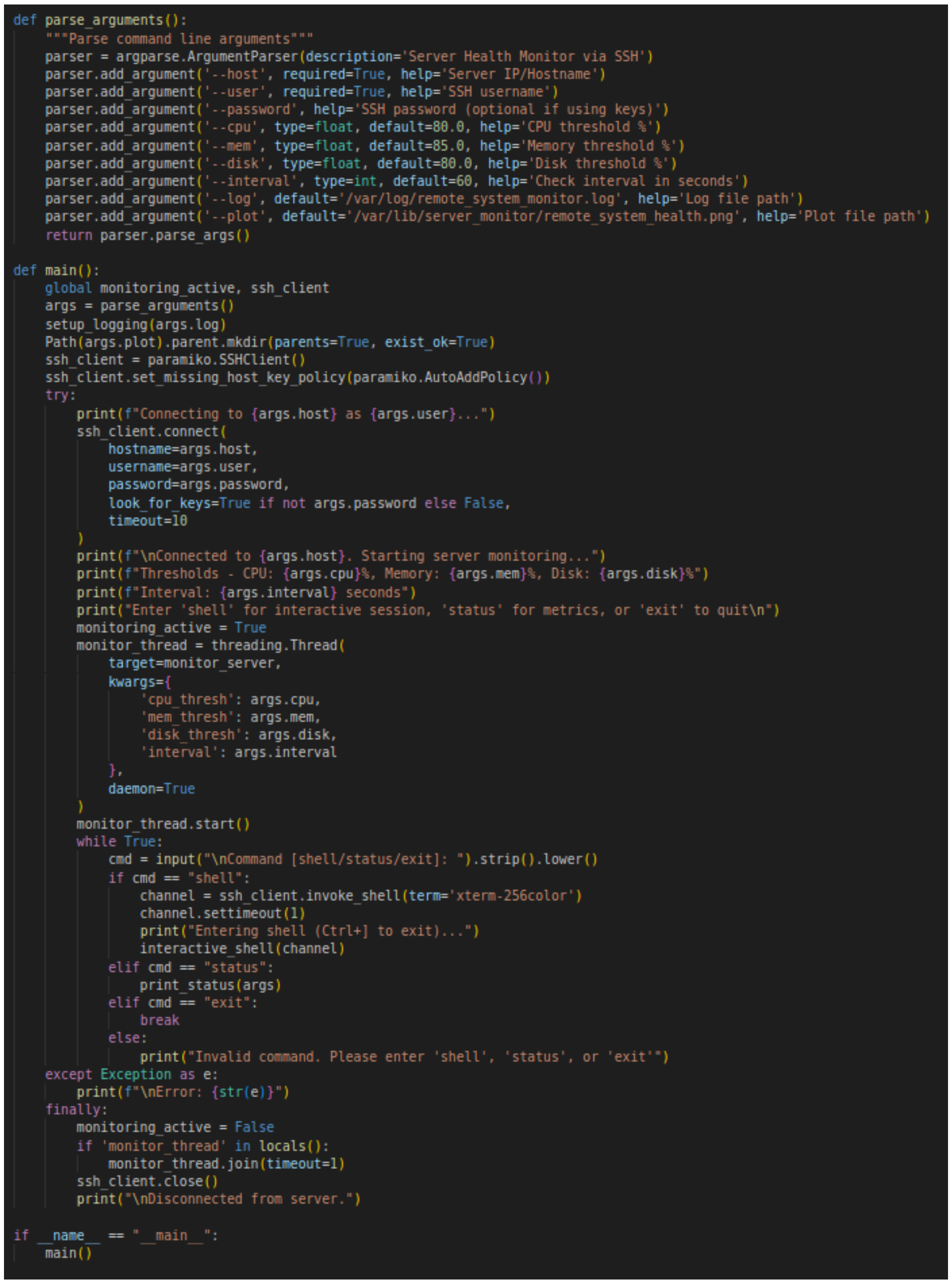
```python
#!/usr/bin/env python3
import paramiko
import getpass
import sys
import socket
import tty
import termios
import select
import time
import logging
from pathlib import Path
import matplotlib.pyplot as plt
from datetime import datetime
import matplotlib
import threading
import argparse
import io

matplotlib.use('Agg')
history_length = 60
cpu_history = []
mem_history = []
disk_history = []
timestamps = []
monitoring_active = False
ssh_client = None
last_metrics = None
last_warnings = []

def setup_logging(log_file='/var/log/remote_system_monitor.log'):
    """Configure logging to only write to file"""
    logger = logging.getLogger()
    logger.setLevel(logging.WARNING)
    for handler in logger.handlers[:]:
        logger.removeHandler(handler)
    try:
        Path(log_file).parent.mkdir(parents=True, exist_ok=True, mode=0o755)
        file_handler = logging.FileHandler(log_file)
        file_handler.setFormatter(logging.Formatter(
            '%(asctime)s - %(levelname)s - %(message)s'
        ))
        logger.addHandler(file_handler)
    except PermissionError:
        logger.addHandler(logging.NullHandler())

def execute_remote_command(command):
    """Execute command on remote server and return output"""
    global ssh_client
    stdin, stdout, stderr = ssh_client.exec_command(command)
    return stdout.read().decode().strip()
```



```python
def get_server_metrics():
    """Get server metrics via SSH"""
    try:
        cpu_percent = float(execute_remote_command(
            "top -bn1 | grep 'Cpu(s)' | awk '{print $2 + $4}'"
        ))
        mem_info = execute_remote_command(
            "free | grep Mem | awk '{print $3/$2 * 100.0}'"
        )
        mem_percent = float(mem_info)
        disk_percent = float(execute_remote_command(
            "df / | tail -1 | awk '{print $5}' | sed 's/%//'"
        ))
        top_cpu = execute_remote_command(
            "ps -eo pid,user,%cpu,%mem,comm --sort=-%cpu | head -n 4 | tail -n 3"
        )
        top_mem = execute_remote_command(
            "ps -eo pid,user,%cpu,%mem,comm --sort=-%mem | head -n 4 | tail -n 3"
        )
        return {
            'cpu': cpu_percent,
            'memory': mem_percent,
            'disk': disk_percent,
            'top_cpu': top_cpu,
            'top_mem': top_mem,
            'timestamp': datetime.now().strftime('%H:%M:%S')
        }
    except Exception as e:
        return None

def update_history(metrics):
    """Update historical data"""
    global cpu_history, mem_history, disk_history, timestamps, last_metrics, last_warnings
    if metrics:
        last_metrics = metrics
        timestamps.append(metrics['timestamp'])
        cpu_history.append(metrics['cpu'])
        mem_history.append(metrics['memory'])
        disk_history.append(metrics['disk'])
        if len(timestamps) > history_length:
            timestamps = timestamps[-history_length:]
            cpu_history = cpu_history[-history_length:]
            mem_history = mem_history[-history_length:]
            disk_history = disk_history[-history_length:]

def generate_plot(output_file='/var/lib/server_monitor/remote_system_health.png',
                  cpu_thresh=80, mem_thresh=85, disk_thresh=80):
    """Generate plot of server metrics"""
    plt.figure(figsize=(12, 8))
    plt.subplot(2, 1, 1)
    plt.plot(timestamps, cpu_history, label='CPU %', marker='o')
    plt.plot(timestamps, mem_history, label='Memory %', marker='s')
    plt.plot(timestamps, disk_history, label='Disk %', marker='^')

    plt.title('Server Resource Usage Over Time')
    plt.ylabel('Usage (%)')
    plt.xticks(rotation=45)
    plt.legend()
    plt.grid(True)

    plt.axhline(y=cpu_thresh, color='r', linestyle='--', alpha=0.3)
    plt.axhline(y=mem_thresh, color='g', linestyle='--', alpha=0.3)
    plt.axhline(y=disk_thresh, color='b', linestyle='--', alpha=0.3)

    plt.tight_layout()
    plt.savefig(output_file, dpi=100, bbox_inches='tight')
    plt.close()
    return output_file
```



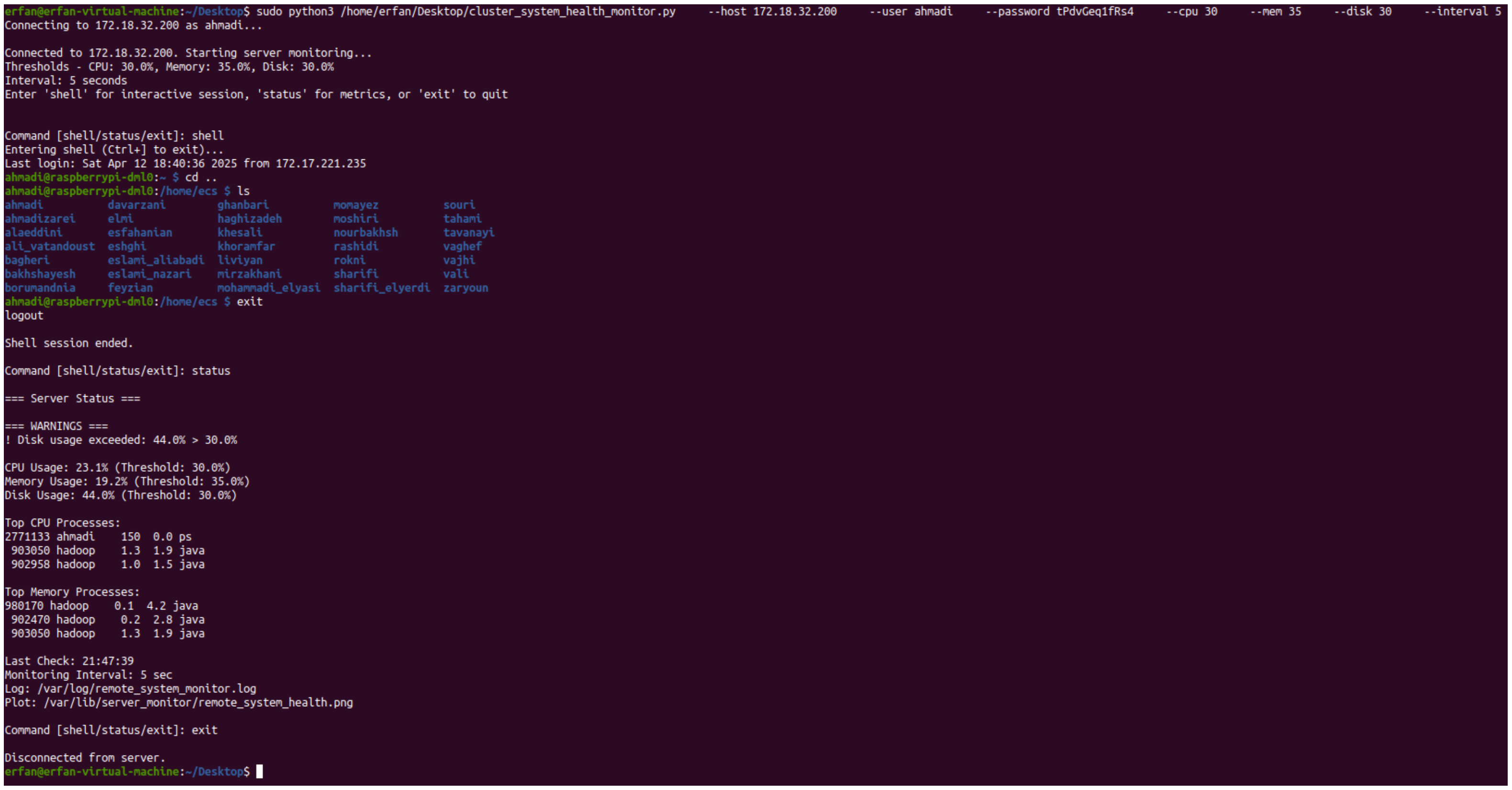
```python
def monitor_server(cpu_thresh=80, mem_thresh=85, disk_thresh=80, interval=60):
    """Monitor server health with thresholds"""
    global monitoring_active, last_warnings
    logger = logging.getLogger()
    while monitoring_active:
        try:
            metrics = get_server_metrics()
            if metrics:
                update_history(metrics)
                generate_plot(cpu_thresh=cpu_thresh, mem_thresh=mem_thresh, disk_thresh=disk_thresh)
                last_warnings.clear()
                if metrics['cpu'] > cpu_thresh:
                    warning = f"CPU usage exceeded: {metrics['cpu']}% > {cpu_thresh}%"
                    last_warnings.append(warning)
                    logger.warning(warning)
                if metrics['memory'] > mem_thresh:
                    warning = f"Memory usage exceeded: {metrics['memory']}% > {mem_thresh}%"
                    last_warnings.append(warning)
                    logger.warning(warning)
                if metrics['disk'] > disk_thresh:
                    warning = f"Disk usage exceeded: {metrics['disk']}% > {disk_thresh}%"
                    last_warnings.append(warning)
                    logger.warning(warning)
            time.sleep(interval)
        except Exception as e:
            logger.error(f"Monitoring error: {str(e)}")
            time.sleep(5)

def print_status(args):
    """Print current server status and show any warnings"""
    global last_metrics, last_warnings
    if not last_metrics:
        print("No metrics available yet")
        return
    print("\n=== Server Status ===")
    if last_warnings:
        print("\n=== WARNINGS ===")
        for warning in last_warnings:
            print(f"! {warning}")
    print(f"\nCPU Usage: {last_metrics['cpu']:.1f}% (Threshold: {args.cpu}%)")
    print(f"Memory Usage: {last_metrics['memory']:.1f}% (Threshold: {args.mem}%)")
    print(f"Disk Usage: {last_metrics['disk']:.1f}% (Threshold: {args.disk}%)")
    print("\nTop CPU Processes:")
    print(last_metrics['top_cpu'])
    print("\nTop Memory Processes:")
    print(last_metrics['top_mem'])
    print(f"\nLast Check: {last_metrics['timestamp']}")
    print(f"Monitoring Interval: {args.interval} sec")
    print(f"Log file: {args.log}")
    print(f"Plot file: {args.plot}")

def interactive_shell(channel):
    """Handle interactive shell session"""
    old_attrs = termios.tcgetattr(sys.stdin)
    tty.setraw(sys.stdin.fileno())
    try:
        while True:
            r, w, e = select.select([channel, sys.stdin], [], [])
            if channel in r:
                try:
                    data = channel.recv(1024)
                    if not data:
                        break
                    sys.stdout.write(data.decode())
                    sys.stdout.flush()
                except socket.timeout:
                    continue

            if sys.stdin in r:
                char = sys.stdin.read(1)
                if char == '\x1d':
                    break
                channel.send(char)
    finally:
        termios.tcsetattr(sys.stdin, termios.TCSADRAIN, old_attrs)
        print("\nShell session ended.")
```



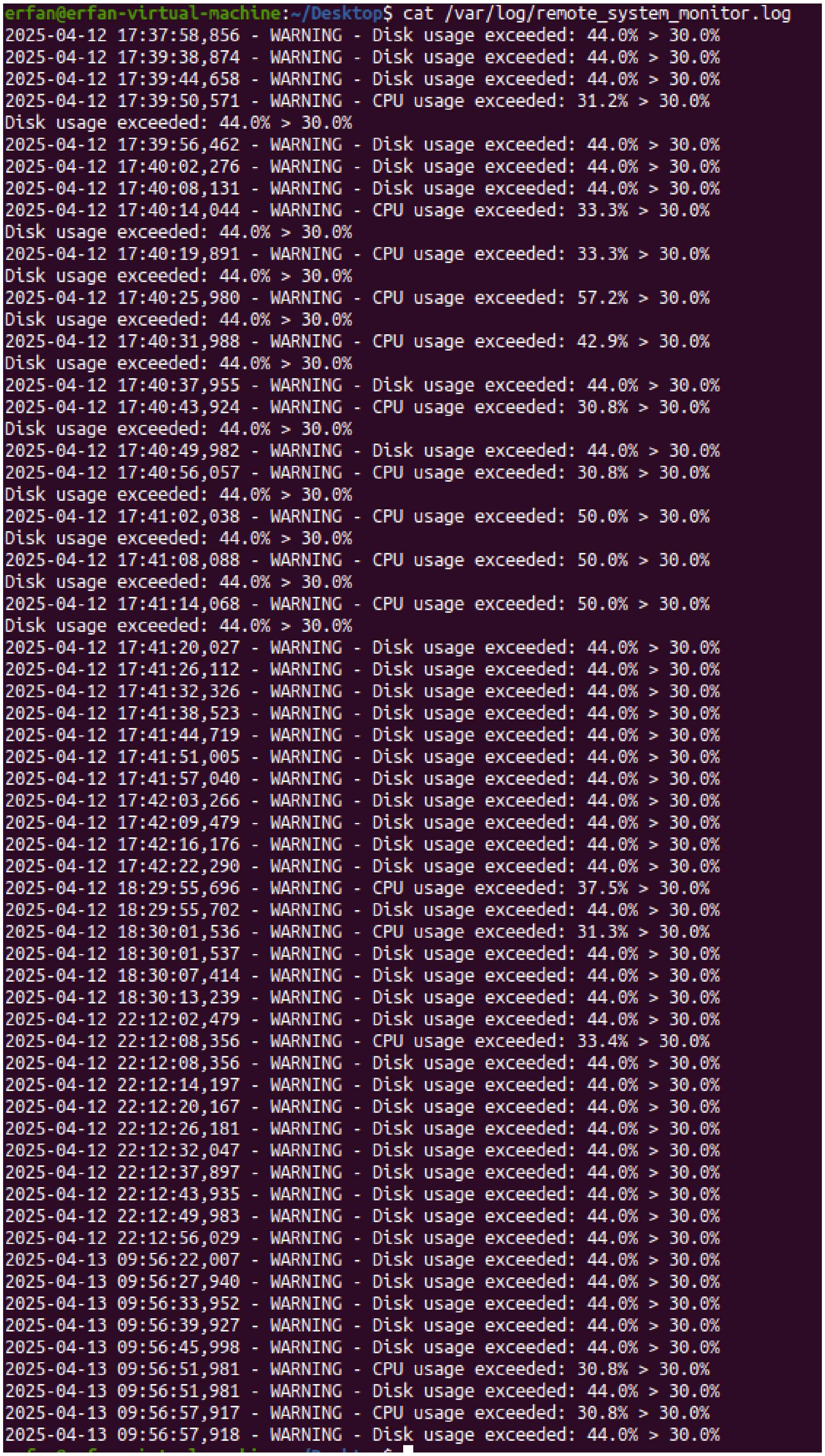
```python
def parse_arguments():
    """Parse command line arguments"""
    parser = argparse.ArgumentParser(description='Server Health Monitor via SSH')
    parser.add_argument('--host', required=True, help='Server IP/Hostname')
    parser.add_argument('--user', required=True, help='SSH username')
    parser.add_argument('--password', help='SSH password (optional if using keys)')
    parser.add_argument('--cpu', type=float, default=80.0, help='CPU threshold %')
    parser.add_argument('--mem', type=float, default=85.0, help='Memory threshold %')
    parser.add_argument('--disk', type=float, default=80.0, help='Disk threshold %')
    parser.add_argument('--interval', type=int, default=60, help='Check interval in seconds')
    parser.add_argument('--log', default='/var/log/remote_system_monitor.log', help='Log file path')
    parser.add_argument('--plot', default='/var/lib/server_monitor/remote_system_health.png', help='Plot file path')
    return parser.parse_args()

def main():
    global monitoring_active, ssh_client
    args = parse_arguments()
    setup_logging(args.log)
    Path(args.plot).parent.mkdir(parents=True, exist_ok=True)
    ssh_client = paramiko.SSHClient()
    ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        print(f"Connecting to {args.host} as {args.user}...")
        ssh_client.connect(
            hostname=args.host,
            username=args.user,
            password=args.password,
            look_for_keys=True if not args.password else False,
            timeout=10
        )
        print(f"\nConnected to {args.host}. Starting server monitoring...")
        print(f"Thresholds - CPU: {args.cpu}%, Memory: {args.mem}%, Disk: {args.disk}%")
        print(f"Interval: {args.interval} seconds")
        print("Enter 'shell' for interactive session, 'status' for metrics, or 'exit' to quit\n")
        monitoring_active = True
        monitor_thread = threading.Thread(
            target=monitor_server,
            kwargs={
                'cpu_thresh': args.cpu,
                'mem_thresh': args.mem,
                'disk_thresh': args.disk,
                'interval': args.interval
            },
            daemon=True
        )
        monitor_thread.start()
        while True:
            cmd = input("\nCommand [shell/status/exit]: ").strip().lower()
            if cmd == "shell":
                channel = ssh_client.invoke_shell(term='xterm-256color')
                channel.settimeout(1)
                print("Entering shell (Ctrl+] to exit)...")
                interactive_shell(channel)
            elif cmd == "status":
                print_status(args)
            elif cmd == "exit":
                break
            else:
                print("Invalid command. Please enter 'shell', 'status', or 'exit'")
    except Exception as e:
        print(f"\nError: {str(e)}")
    finally:
        monitoring_active = False
        if 'monitor_thread' in locals():
            monitor_thread.join(timeout=1)
        ssh_client.close()
        print("\nDisconnected from server.")

if __name__ == "__main__":
    main()
```

This is how I run the program:

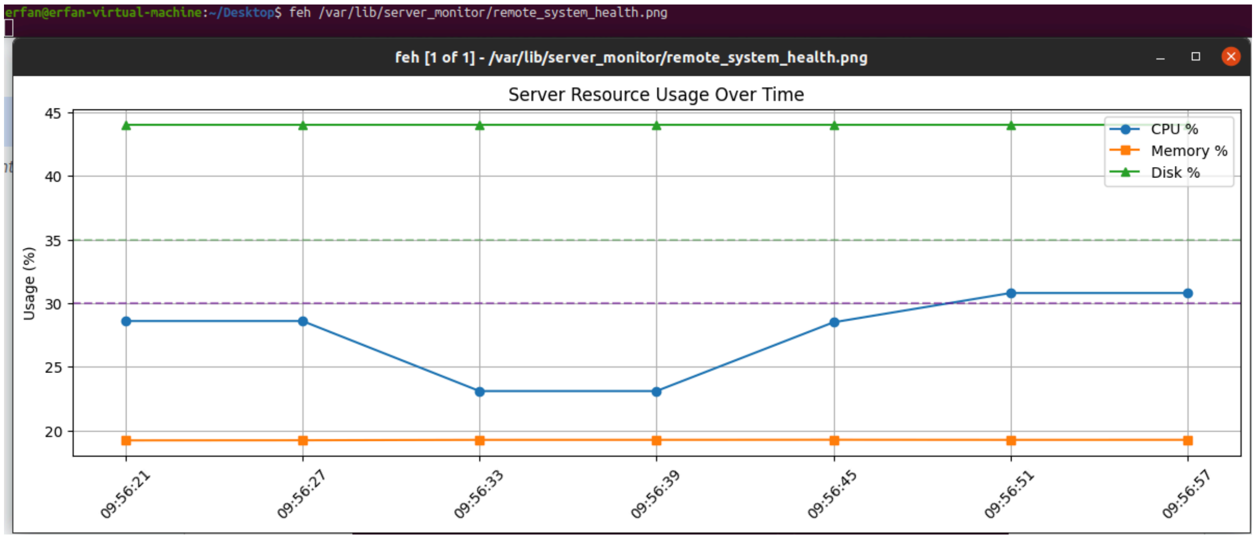
1. I provide thresholds, server information, and time interval as arguments
2. After running:
    - If I execute *shell*, it connects to the server terminal (exit with *exit* command)
    - If I run *status*, it displays the status from the last interval



```
erfan@erfan-virtual-machine:~/Desktop$ sudo python3 /home/erfan/Desktop/cluster_system_health_monitor.py     --host 172.18.32.200     --user ahmadi     --password tPdvGeq1fRs4     --cpu 30     --mem 35     --disk 30     --interval 5
Connecting to 172.18.32.200 as ahmadi...

Connected to 172.18.32.200. Starting server monitoring...
Thresholds - CPU: 30.0%, Memory: 35.0%, Disk: 30.0%
Interval: 5 seconds
Enter 'shell' for interactive session, 'status' for metrics, or 'exit' to quit


Command [shell/status/exit]: shell
Entering shell (Ctrl+] to exit)...
Last login: Sat Apr 12 18:40:36 2025 from 172.17.221.235
ahmadi@raspberrypi-dml0:~ $ cd ..
ahmadi@raspberrypi-dml0:/home/ecs $ ls
ahmadi          davarzani        ghanbari          momayez          souri
ahmadizarei     elmi             haghizadeh        moshiri          tahami
alaeddini       esfahanian       khesali           nourbakhsh       tavanayi
ali_vatandoust  eshghi           khoramfar         rashidi          vaghef
bagheri         eslami_aliabadi  liviyan           rokni            vajhi
bakhshayesh     eslami_nazari    mirzakhani        sharifi          vali
borumandnia     feyzian          mohammadi_elyasi  sharifi_elyerdi  zaryoun
ahmadi@raspberrypi-dml0:/home/ecs $ exit
logout

Shell session ended.

Command [shell/status/exit]: status

=== Server Status ===

=== WARNINGS ===
! Disk usage exceeded: 44.0% > 30.0%

CPU Usage: 23.1% (Threshold: 30.0%)
Memory Usage: 19.2% (Threshold: 35.0%)
Disk Usage: 44.0% (Threshold: 30.0%)

Top CPU Processes:
2771133 ahmadi    150  0.0 ps
 903050 hadoop    1.3  1.9 java
 902958 hadoop    1.0  1.5 java

Top Memory Processes:
980170 hadoop    0.1  4.2 java
 902470 hadoop    0.2  2.8 java
 903050 hadoop    1.3  1.9 java

Last Check: 21:47:39
Monitoring Interval: 5 sec
Log: /var/log/remote_system_monitor.log
Plot: /var/lib/server_monitor/remote_system_health.png

Command [shell/status/exit]: exit

Disconnected from server.
erfan@erfan-virtual-machine:~/Desktop$
```



```
erfan@erfan-virtual-machine:~/Desktop$ cat /var/log/remote_system_monitor.log
2025-04-12 17:37:58,856 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:39:38,874 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:39:44,658 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:39:50,571 - WARNING - CPU usage exceeded: 31.2% > 30.0%
Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:39:56,462 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:40:02,276 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:40:08,131 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:40:14,044 - WARNING - CPU usage exceeded: 33.3% > 30.0%
Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:40:19,891 - WARNING - CPU usage exceeded: 33.3% > 30.0%
Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:40:25,980 - WARNING - CPU usage exceeded: 57.2% > 30.0%
Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:40:31,988 - WARNING - CPU usage exceeded: 42.9% > 30.0%
Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:40:37,955 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:40:43,924 - WARNING - CPU usage exceeded: 30.8% > 30.0%
Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:40:49,982 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:40:56,057 - WARNING - CPU usage exceeded: 30.8% > 30.0%
Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:41:02,038 - WARNING - CPU usage exceeded: 50.0% > 30.0%
Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:41:08,088 - WARNING - CPU usage exceeded: 50.0% > 30.0%
Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:41:14,068 - WARNING - CPU usage exceeded: 50.0% > 30.0%
Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:41:20,027 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:41:26,112 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:41:32,326 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:41:38,523 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:41:44,719 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:41:51,005 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:41:57,040 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:42:03,266 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:42:09,479 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:42:16,176 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 17:42:22,290 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 18:29:55,696 - WARNING - CPU usage exceeded: 37.5% > 30.0%
2025-04-12 18:29:55,702 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 18:30:01,536 - WARNING - CPU usage exceeded: 31.3% > 30.0%
2025-04-12 18:30:01,537 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 18:30:07,414 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 18:30:13,239 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 22:12:02,479 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 22:12:08,356 - WARNING - CPU usage exceeded: 33.4% > 30.0%
2025-04-12 22:12:08,356 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 22:12:14,197 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 22:12:20,167 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 22:12:26,181 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 22:12:32,047 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 22:12:37,897 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 22:12:43,935 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 22:12:49,983 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-12 22:12:56,029 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-13 09:56:22,007 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-13 09:56:27,940 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-13 09:56:33,952 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-13 09:56:39,927 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-13 09:56:45,998 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-13 09:56:51,981 - WARNING - CPU usage exceeded: 30.8% > 30.0%
2025-04-13 09:56:51,981 - WARNING - Disk usage exceeded: 44.0% > 30.0%
2025-04-13 09:56:57,917 - WARNING - CPU usage exceeded: 30.8% > 30.0%
2025-04-13 09:56:57,918 - WARNING - Disk usage exceeded: 44.0% > 30.0%
```



```
erfan@erfan-virtual-machine:~/Desktop$ feh /var/lib/server_monitor/remote_system_health.png
```