```
import os
import re
from collections import Counter
import csv


log_file_path = 'sample.log'  # Ensure your log file is in the same directory or provide the full path
with open('/content/log_analysis.py', 'r') as file:
  for line in file:
    line = line.strip()


import re
ip_match = re.search(r'^\d+\.\d+\.\d+\.\d+', line)
if ip_match:
  ip_address = ip_match.group()
  print(f"Extracted IP: {ip_address}")


try:
  ip_address = re.search(r'^\d+\.\d+\.\d+\.\d+', line).group()
except AttributeError:
  print("No IP address found in line.")
```

```
No IP address found in line.
```

```
import re
from collections import Counter
def extract_ip_addresses(line):
  ip_match = re.search(r'^\d+\.\d+\.\d+\.\d+', line)
  if ip_match:
    return ip_match.group()
  return None


def count_ip_requests(log_file_path):
  ip_counter = Counter()
  with open(log_file_path, 'r') as file:
    for line in file:
      ip_address = extract_ip_addresses(line)
      if ip_address:
        ip_counter[ip_address] += 1
  return ip_counter


def display_sorted_ip_counts(ip_counter):
  sorted_ips = ip_counter.most_common()
  print("IP Address Request Counts:")
  for ip, count in sorted_ips:
    print(f"{ip}: {count}")


if __name__ == "__main__":
  log_file_path = 'sample.log'
  ip_counts = count_ip_requests(log_file_path)
  display_sorted_ip_counts(ip_counts)
```

```
IP Address Request Counts:
203.0.113.5: 8
198.51.100.23: 8
192.168.1.1: 7
10.0.0.2: 6
192.168.1.100: 5
```

```
import re
from collections import Counter
def extract_endpoint(line):
  endpoint_match = re.search(r'\"[A-Z]+ (.+?) HTTP', line)
  if endpoint_match:
    return endpoint_match.group(1)
  return None
def count_endpoint_accesses(log_file_path):
  endpoint_counter = Counter()
  with open(log_file_path, 'r') as file:
    for line in file:
      endpoint = extract_endpoint(line)
      if endpoint:
        endpoint_counter[endpoint] += 1
  return endpoint_counter
def find_most_frequent_endpoint(endpoint_counter):
  most_frequent = endpoint_counter.most_common(1)
  if most_frequent:
    endpoint, count = most_frequent[0]
    print(f"Most Frequently Accessed Endpoint: {endpoint} (Accessed {count} times)")
if __name__ == "__main__":
  log_file_path = 'sample.log'
  endpoint_counts = count_endpoint_accesses(log_file_path)
  find_most_frequent_endpoint(endpoint_counts)
```

```
Most Frequently Accessed Endpoint: /login (Accessed 13 times)
```

```
import re
from collections import Counter
def detect_failed_logins(log_file_path, threshold=10):
  failed_login_counter = Counter()
  with open(log_file_path, 'r') as file:
    for line in file:
      if '401' in line and 'Invalid credentials' in line:
        ip_match = re.search(r'^\d+\.\d+\.\d+\.\d+', line)
        if ip_match:
          ip_address = ip_match.group()
          failed_login_counter[ip_address] += 1
  return failed_login_counter, threshold
def flag_suspicious_ips(failed_login_counter, threshold):
  suspicious_ips = {ip: count for ip, count in failed_login_counter.items() if count > threshold}
  return suspicious_ips
def display_suspicious_activity(suspicious_ips):
  print("\nSuspicious Activities:")
  for ip, count in suspicious_ips.items():
    print(f"{ip}: {count} failed attempts")
if __name__ == "__main__":
  log_file_path = 'sample.log'
  failed_login_counter, threshold = detect_failed_logins(log_file_path)
  suspicious_ips = flag_suspicious_ips(failed_login_counter, threshold)
  display_suspicious_activity(suspicious_ips)
```

```

Suspicious Activities:
```

```
def display_results(ip_counts, endpoint_counts, failed_logins):
  print("IP Request Counts:")
  for ip, count in ip_counts.most_common():
    print(f"{ip}: {count}")
  print("\nMost Accessed Endpoint:")
  most_accessed = endpoint_counts.most_common(1)
  if most_accessed:
    print(f"{most_accessed[0][0]} (Accessed {most_accessed[0][1]} times)")
  print("\nSuspicious Activities:")
  for ip, count in failed_logins.items():
    if count > 10:
      print(f"{ip}: {count} failed attempts")


import re

from collections import Counter

import csv



def parse_log(file_path):

    # Initialize counters for IPs, endpoints, and failed logins

    ip_counter = Counter()

    endpoint_counter = Counter()

    failed_login_counter = Counter()



    try:

        with open(file_path, 'r') as file:

            for line in file:

                # Extract IP address

                ip_match = re.search(r'^\d+\.\d+\.\d+\.\d+', line)

                if ip_match:

                    ip = ip_match.group()

                    ip_counter[ip] += 1



                # Extract endpoint

                endpoint_match = re.search(r'\"[A-Z]+ (.+?) HTTP', line)

                if endpoint_match:

                    endpoint = endpoint_match.group(1)

                    endpoint_counter[endpoint] += 1



                # Detect failed logins

                if '401' in line and 'Invalid credentials' in line:

                    failed_login_counter[ip] += 1

    except FileNotFoundError:

        print(f"Error: The file {file_path} was not found.")

    except Exception as e:
        print(f"An error occurred: {e}")


    return ip_counter, endpoint_counter, failed_login_counter


def write_to_csv(ip_counts, endpoint_counts, failed_logins, output_file):

    with open(output_file, 'w', newline='') as csvfile:

        fieldnames = ['IP Address', 'Request Count', 'Endpoint', 'Endpoint Access Count', 'Failed Login Cou

        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        writer.writeheader()


        # Write IP request counts

        for ip, count in ip_counts.items():

            writer.writerow({'IP Address': ip, 'Request Count': count})


        # Write endpoint access counts

        for endpoint, count in endpoint_counts.items():

            writer.writerow({'Endpoint': endpoint, 'Endpoint Access Count': count})


        # Write suspicious activity counts

        for ip, count in failed_logins.items():

            if count > 10:  # Assuming threshold is 10

                writer.writerow({'IP Address': ip, 'Failed Login Count': count})


def display_results(ip_counts, endpoint_counts, failed_logins):

    print("IP Request Counts:")

    for ip, count in ip_counts.most_common():

        print(f"{ip}: {count}")


    print("\nMost Accessed Endpoint:")

    most_accessed = endpoint_counts.most_common(1)

    if most_accessed:

        print(f"{most_accessed[0][0]} (Accessed {most_accessed[0][1]} times)")
    print("\nSuspicious Activities:")

    for ip, count in failed_logins.items():

        if count > 10:  # Assuming threshold is 10

            print(f"{ip}: {count} failed attempts")



if __name__ == "__main__":

    log_file_path = 'sample.log'

    output_csv = 'log_analysis_results.csv'



    # Ensure 'parse_log' function returns the expected values

    ip_counts, endpoint_counts, failed_logins = parse_log(log_file_path)

    display_results(ip_counts, endpoint_counts, failed_logins)

    write_to_csv(ip_counts, endpoint_counts, failed_logins, output_csv)
```

```
IP Request Counts:
203.0.113.5: 8
198.51.100.23: 8
192.168.1.1: 7
10.0.0.2: 6
192.168.1.100: 5

Most Accessed Endpoint:
/login (Accessed 13 times)

Suspicious Activities:
```

```
import re
from collections import Counter

def extract_ip_addresses(line):
  ip_match = re.search(r'^\d+\.\d+\.\d+\.\d+', line)
  if ip_match:
    return ip_match.group()
  return None

def extract_endpoint(line):
  endpoint_match = re.search(r'\"[A-Z]+ (.+?) HTTP', line)
  if endpoint_match:
    return endpoint_match.group(1)
  return None

def parse_log(log_file_path):
  ip_counter = Counter()
  endpoint_counter = Counter()
  failed_login_counter = Counter()

  with open(log_file_path, 'r') as file:
    for line in file:
      # Extract IP address
      ip_address = extract_ip_addresses(line)
      if ip_address:
        ip_counter[ip_address] += 1
      # Extract endpoint
      endpoint = extract_endpoint(line)
      if endpoint:
        endpoint_counter[endpoint] += 1

      # Detect failed login attempts
      if '401' in line and 'Invalid credentials' in line:
        ip_match = re.search(r'^\d+\.\d+\.\d+\.\d+', line)
        if ip_match:
          failed_login_counter[ip_match.group()] += 1

  return ip_counter, endpoint_counter, failed_login_counter
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
/content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
def parse_log(file_path):
  pass
def count_requests(log_data):
  pass
def find_most_accessed(log_data):
  pass
def write_to_csv(data, output_file):
  pass
def parse_log(file_path):
  with open(file_path, 'r') as file:
    pass
ip_address_counts = Counter()
endpoint_access_counts = Counter()
suspicious_login_attempts = Counter()
from collections import Counter
def count_ip_requests(log_lines):
  ip_counter = Counter()
  for line in log_lines:
    ip_address = extract_ip_from_line(line)
    if ip_address:
      ip_counter[ip_address] += 1
      return ip_counter
```

```
import re

def extract_ip_from_line(line):
    ip_match = re.search(r'^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}', line)  # Improved regex
    if ip_match:
        return ip_match.group(0)
    return None
```