1.Multi-Server Health Aggregator

Problem Statement:

You joined as a **DevOps Intern** at a mid-sized company. Your task is to help the Operations team by automating a health check report of all their servers.
They want to input the number of servers, and for each server:

- Server name
- CPU usage (%)
- Memory usage (%)
- Disk usage (%)

Write a Python program that:

- Checks if any metric exceeds thresholds (CPU > 85%, Memory > 80%, Disk > 90%).
- Prints warnings for unhealthy servers.
- Displays the total count of unhealthy servers.

Sample input and output:

```
Enter the number of servers: 3

Enter details for server 1:

Server name: web

CPU usage (%): 65

Memory usage (%): 86

Disk usage (%): 75

Enter details for server 2:

Server name: cache

CPU usage (%): 86

Memory usage (%): 75

Disk usage (%): 50

Enter details for server 3:

Server name: jail

CPU usage (%): 92

Memory usage (%): 81

Disk usage (%): 76

Server web is unhealthy: Memory high
```

Solution:

```
num_servers = int(input("Enter the number of servers: "))
servers = []

for i in range(num_servers):
    print(f"\nEnter details for server {i+1}:")
    name = input("Server name: ")
    cpu = int(input("CPU usage (%): "))
    memory = int(input("Memory usage (%): "))
    disk = int(input("Disk usage (%): "))

    servers.append({"server": name, "cpu": cpu, "memory": memory, "disk": disk})

unhealthy_count = 0

for s in servers:
    warnings = []
    if s["cpu"] > 85:
        warnings.append("CPU high")
    if s["memory"] > 80:
        warnings.append("Memory high")
    if s["disk"] > 90:
        warnings.append("Disk high")

    if warnings:
        print(f"\nServer {s['server']} is unhealthy: {', '.join(warnings)}")
        unhealthy_count += 1

print(f"\nTotal unhealthy servers: {unhealthy_count}")
```

2.Log File Analyzer

Problem Statement:

During your internship, you’re asked to analyze the system logs.
The system admin will input a number of log entries manually in the format:
2025-09-13 10:15:00 INFO System started

Write a Python program that:

- Accepts a number of log lines from the user.
- Counts the number of INFO, WARNING, and ERROR logs.
- Displays the most frequent log level.

Solution:

```
from collections import Counter


num_logs = int(input("Enter number of log entries: "))

log_entries = []


for i in range(num_logs):

    log = input(f"Enter log entry {i+1} (e.g., '2025-09-13 10:15:00 INFO System started'): ")

    log_entries.append(log)


levels = []


for entry in log_entries:

    parts = entry.split()

    levels.append(parts[2])


counter = Counter(levels)


for level, count in counter.items():

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


most_common = counter.most_common(1)[0][0]
```

```
print(f"Most frequent log level: {most_common}")
```

Sample input and output:

```
Enter number of log entries: 3
Enter log entry 1 (e.g., '2025-09-13 10:15:00 INFO System started'): 2025-09-14 11:50:25 WARNING
high memory usage
Enter log entry 2 (e.g., '2025-09-13 10:15:00 INFO System started'): 2025-09-14 10:35:30 INFO
system stopped
Enter log entry 3 (e.g., '2025-09-13 10:15:00 INFO System started'): 2025-09-14 09:36:25 INFO User
login
WARNING: 1
INFO: 2
Most frequent log level: INFO
```

3.Resource Monitoring with Alerts

Mode Problem Statement:

Your DevOps lead asks you to build a monitoring simulation tool.
The user will input how many resource checks they want to simulate.
For each check, the CPU and Memory usage will be entered.

Write a Python program that:

- Alerts if CPU usage exceeds 85% or Memory exceeds 80%.

Solution:

```python
num_checks = int(input("Enter number of resource checks: "))

cpu_usages = []

memory_usages = []


for i in range(num_checks):

    cpu = int(input(f"CPU usage at check {i+1} (%): "))

    memory = int(input(f"Memory usage at check {i+1} (%): "))

    cpu_usages.append(cpu)

    memory_usages.append(memory)


for i in range(num_checks):

    if cpu_usages[i] > 85 or memory_usages[i] > 80:

        print(f"Alert at check {i+1}: CPU {cpu_usages[i]}%, Memory {memory_usages[i]}%")
```

Sample input and output:

```
Enter number of resource checks: 3
CPU usage at check 1 (%): 65
Memory usage at check 1 (%): 85
CPU usage at check 2 (%): 76
Memory usage at check 2 (%): 87
CPU usage at check 3 (%): 59
Memory usage at check 3 (%): 65
Alert at check 1: CPU 65%, Memory 85%
Alert at check 2: CPU 76%, Memory 87%
```

```
Enter number of resource checks: 2
CPU usage at check 1 (%): 56
Memory usage at check 1 (%): 75
CPU usage at check 2 (%): 75
Memory usage at check 2 (%): 78
```

4.User Login System

Problem Statement:

Your team is automating a simple login system where users can attempt to login.
They are allowed **3 attempts maximum** to enter valid credentials.
Predefined users and passwords are given.

Write a Python program that:

- Prompts for username and password.
- Displays successful login or locks the account after 3 failed attempts.

```
"admin": "admin123",
   "devops_user": "devops2023",
   "tester": "testme"
```

Solution:

```
users = {
   "admin": "admin123",
   "devops_user": "devops2023",
   "tester": "testme"
}

attempts = 0
while attempts < 3:
   username = input("Enter username: ")
   password = input("Enter password: ")

   if username in users and users[username] == password:
      print("Login successful!")
      break
   else:
      print("Invalid credentials.")
      attempts += 1

if attempts == 3:
    print("Account locked!")
```

Sample input and output:

```
Enter username: admin
Enter password: admin12
Invalid credentials.
Enter username: devops_user
Enter password: devop
Invalid credentials.
Enter username: tester
Enter password: test
Invalid credentials.
Account locked!

Enter username: tester
Enter password: testme
Login successful!
```

5.Automated Log Cleanup

Problem Statement:

You’re developing an automated cleanup utility for logs.
The user will enter the number of log files manually, along with file names that have timestamps embedded.
Files older than **6 months** should be marked for deletion.

Write a Python program that:

- Processes user-entered file names.
- Prints which files would be deleted and which will be kept.

Solution:

```
from datetime import datetime, timedelta


num_files = int(input("Enter number of log files: "))

files = []


for i in range(num_files):

   file = input(f"Enter file name {i+1} (format: 'app_log_YYYYMMDD.log'): ")

   files.append(file)


today = datetime.today()

threshold_date = today - timedelta(days=180)


deleted_files = []

remaining_files = []


for file in files:

   date_str = file.split('_')[2].split('.')[0]  # Extract YYYYMMDD

   file_date = datetime.strptime(date_str, "%Y%m%d")


   if file_date < threshold_date:

      deleted_files.append(file)
```

```
    else:
        remaining_files.append(file)


print("Deleted files:", deleted_files)
print("Remaining files:", remaining_files)
```

Sample input and output:

```
Enter number of log files: 3
Enter file name 1 (format: 'app_log_YYYYMMDD.log'): app_log_20250912.log
Enter file name 2 (format: 'app_log_YYYYMMDD.log'): app_log_20231203.log
Enter file name 3 (format: 'app_log_YYYYMMDD.log'): app_log_20241205.log
Deleted files: ['app_log_20231203.log', 'app_log_20241205.log']
Remaining files: ['app_log_20250912.log']
```