Log Health Monitoring Script (Python)

Overview

This project is a lightweight Python script that analyzes system log files and generates a concise summary of system health. It focuses on identifying errors, warnings, and failed login attempts to support quick inspection of log data.

Real-World Relevance

In real IT environments, servers and applications automatically generate log files that record operational events. Engineers rely on summarized log information to quickly understand system behavior without manually reading large log files. This project demonstrates the core logic behind such log monitoring in a simplified form.

What the Script Does

- Reads a log file line by line
- Detects predefined keywords related to errors, warnings, and login failures
- Counts occurrences of each event type
- Generates a text-based log summary report
- Flags potential issues when predefined limits are exceeded

Technologies Used

- Python
- File handling
- String processing
- Conditional logic and loops
- Exception handling

Input

- Text-based log file (input_logs.txt)
- Contains simulated log entries such as errors, warnings, and failed login attempts

Output

- Text-based summary report (summary_report.txt)
- Displays total error count
- Displays total warning count
- Displays total failed login attempts
- Highlights alerts when thresholds are exceeded

Source Code

```
ERROR_LIMIT = 4
LOGIN_LIMIT = 2

errors = 0
warnings = 0
logins = 0

try:
    with open("input_logs.txt") as file:
        for line in file:
            line = line.lower()

            if "error" in line:
                errors += 1
            if "warning" in line:
                warnings += 1
            if "login failed" in line:
                logins += 1

except FileNotFoundError:
    print("Log file not found.")


with open("summary_report.txt", "w") as report:
    report.write("LOG SUMMARY\n")
    report.write("-----------\n")
    report.write(f"Errors   : {errors}\n")
    report.write(f"Warnings : {warnings}\n")
    report.write(f"Logins   : {logins}\n\n")

    if errors > ERROR_LIMIT:
        report.write("Warning: High error count\n")
    if logins > LOGIN_LIMIT:
        report.write("Security alert: Login failures\n")

print("Log analysis completed.")
```

Conclusion

This is a small Python backend utility that demonstrates log analysis fundamentals. It focuses on clarity, correctness and real-world relevance rather than scale