Clarity – Data Operations Engineer Code Challenge

This challenge is meant to test your software development skills using Python as programming language. You are free to use any tools you feel comfortable with.

Your answer will be evaluated based on:
- Correctness, does it work?
- Is it tested?
- Is it well designed?
- Would this code be easy to extend or maintain?
- Is the code easy to comprehend?
- Performs well?

Your implementation must be your own, making use of your standard library. Also, please feel free to consult any relevant documentation. Please provide a complete solution, including any instructions an engineer would need to build and run your software, e.g. a README, Makefile, setup.py, build.xml, etc.

Challenge description

A log file contains newline-terminated, space-separated text formatted like:

<unix_timestamp> <hostname> <hostname>

For example:

```
1366815793 quark garak
1366815795 brunt quark
1366815811 lilac garak
```

Each line represents connection from a host (left) to another host (right) at a given time. The lines are roughly sorted by timestamp. They might be out of order by maximum 5 minutes.
Implement a tool that parse log files like these, we provide you a input Data Example.

Goals to Achieve

1. Parse the data with a time_init, time_end (Required ~3h)

Build a tool, that given the name of a file (with the format described above), an **init_datetime** , an **end_datetime**, and a **Hostname**, returns:

- A list of hostnames connected to the given host during the given period.

2. Unlimited Input Parser (Optional ~3h)

The tool should both parse previously written log files and terminate or collect input from a new log file while it's being written and run indefinitely.

The script will output, once every hour:

- A list of hostnames connected to a given (configurable) host during the last hour.
- A list of hostnames received connections from a given (configurable) host during the last hour.
- The hostname that generated most connections in the last hour.

Both the number of log lines and hostnames can be very high. Consider implementing a CPU and memory-efficient solution. Please feel free to make assumptions as necessary with proper documentation.

Good luck!