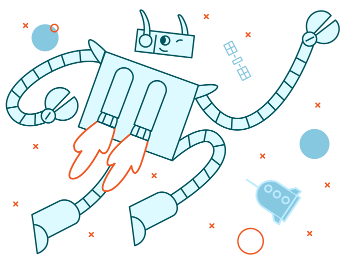
Ping Performance Analyzer with subprocess



Goal

Develop a Python script that accepts a list of IP addresses as command line arguments, pings each, and reports their response times.

Background

In the midst of preparing for a major product launch, your tech company faces intermittent network issues that threaten to delay development. Suspecting that the problem might be linked to network latency or loss of connectivity with key servers, the IT department has tasked you with developing a tool to quickly assess network health. Your mission is to create a script that can ping a list of critical servers and report back on their response times.

Task Description

1. **Script Creation**: Create a Python script named `pinger.py`. The script should take a list of IP addresses from command line arguments using `sys.argv`.

2. **Using subprocess**: Implement the ping functionality using the `subprocess.check_output` function to execute the `ping` command for each provided IP address.

    - This function helps us run external commands in Python and get their output.

- Example: `ping_output = subprocess.check_output(['ping', '-c', '1', ip_address])`

3. **Output Format**: For each IP, the script should print:

```
IP Address: [IP] | Responded: [Yes/No] | Response Time: [Time in ms]
```

   If an IP does not respond, do not show a response time.

4. **Test**: Test your script on the following IPs:

- 8.8.8.8
- 1.1.1.1
- 192.168.1.1
- 216.58.205.46
- 151.101.1.69
- 13.107.21.200

Example usage and output

```
python pinger.py 8.8.8.8 1.1.1.1
```

```
IP Address: 8.8.8.8 | Responded: Yes | Response Time: 14 ms
IP Address: 1.1.1.1 | Responded: Yes | Response Time: 11 ms
```

To submit

Submit the `pinger.py` script.

Sentinel