Problem Description

In this problem you will write a Python program that solves a "puzzler" that was presented on NPR's CarTalk radio program. In a direct quote of their radio transcript, found here `http://www.cartalk.com/content/hall-lights?question`, the problem is described as follows:

> **RAY**: This puzzler is from my "ceiling light" series. Imagine, if you will, that you have a long, long corridor that stretches out as far as the eye can see. In that corridor, attached to the ceiling are lights that are operated with a pull cord.
>
> There are gazillions of them, as far as the eye can see. Let's say there are 20,000 lights in a row.
>
> They're all off. Somebody comes along and pulls on each of the chains, turning on each one of the lights. Another person comes right behind, and pulls the chain on every second light.
>
> **TOM**: Thereby turning off lights 2, 4, 6, 8 and so on.
>
> **RAY**: Right. Now, a third person comes along and pulls the cord on every third light. That is, lights number 3, 6, 9, 12, 15, etcetera. Another person comes along and pulls the cord on lights number 4, 8, 12, 16 and so on. Of course, each person is turning on some lights and turning other lights off.
>
> If there are 20,000 lights, at some point someone is going to come skipping along and pull every 20,000th chain.
>
> When that happens, some lights will be on, and some will be off. Can you predict which lights will be on?

Goals

1. Write a Python script that builds an array of 20,000 light states and iterates through as described in the problem description. The output of the script should print a nicely formatted message of the state of each of the first 100 lightbulbs.
2. Modify the program to be run from the command line and accept a command line argument that says how many lights you should consider.
3. Modify your code such that it can both be executed at the command line (accepting arguments), or it can be `imported` in a different Python script, and the "helper" functions that you have written can then be used in within that script

Problem Extensions

1. Use the *%timeit* IPython "magic" function or directly use the timeit Python module to identify pieces of your code that may be the slowest parts. Then spend some time and try if you can increase the speed your code.

Useful Notes

1. For the first goal, you will likely be using a Python list of `bool` datatypes

2. Python scripts can be executed in the following ways:

   - At the command line you can type `python scriptname.py`
   - Within IPython you can use the *%run* magic function e.g. `%run scriptname.py`
   - You can make the script executable using the command `chmod a+x scriptname.py` and then you can tell your terminal to use Python to execute the script. This is done using a *hashbang* (also called a *shebang*). Basically you just put one of the following as the first line of your file:
     - `#!/usr/bin/python`
     - `#!/usr/bin/env python`

   Once all of this is done, the script can then be executed using just `./scriptname.py`. For details on what this does, check out this Unix.StackExchange question and read the accepted answer.

3. A simple way of collecting command line arguments is to import the `sys` module and then use `sys.argv`. A more advanced (and arguably easier) way is to use the `argparse` module.

4. The third goal will be accomplished using a piece of "boilerplate" code that many Python scripts end up using. Read this question on Stack Overflow, and see if you can figure it out yourself. The accepted answer is great!