Formatting Style

1. Use Python style conventions for your function and variable names (pothole case: lowercase letters with words separated by underscores (_) to improve readability).
2. Choose good names for your functions and variables. For example, `num_bright_spots` is more helpful and readable than `nbs`.
3. Use a tab width of 4 (Wingware's default), if you use tabs at all. The best way to make sure your program will be formatted correctly is never to mix spaces and tabs -- use only tabs, or only spaces.
4. Put a blank space before and after every operator. For example, the first line below is good but the second line is not:

   ```
   b = 3 > x and 4 - 5 < 32
   ```

   ```
   b= 3>x and 4-5<32
   ```

5. Write a `docstring` comment for each function. (See below for guidelines on the content of your docstrings.)

   Put a blank line after every `docstring` comment.

6. Each line must be less than **80 characters** long *including tabs and spaces*. You should break up long lines using `\`.

Docstrings

1. Describe precisely *what* the function does.
2. Do not reveal *how* the function does it.
3. Make the purpose of every parameter clear.
4. Refer to every parameter by name.
5. Be clear about whether the function returns a value, and if so, what.
6. Explain any conditions that the function assumes are true. Examples: "n is an int", "n != 0", "the height and width of p are both even."
7. Be concise and grammatically correct.
8. Write the docstring as a command (e.g., "Return the first ...") rather than a statement (e.g., "Returns the first ...")

Other things to consider

- **Correctness:** Your code should perform as specified.
- **Docstrings:** For each function that you design from scratch, write a good `docstring`. (Do not change the docstrings that we have already written for you.)
- **Internal comments:** Within functions, the more complicated parts of your code should also be described using "internal" comments.
- **Programming style:** Your variable names should be meaningful and your code as simple and clear as possible.
- **Good use of helper functions:** If you find yourself repeating a task, you should add a helper function and call that function instead of duplicating the code. And if a function is more than about 20 lines long, consider introducing helper functions to do some of the work – even if they will only be called once.