Homework #3: Color Graph (100 pts)

For this assignment, you must implement another portion of your CS325 web app. Specifically, you must implement Python source code that inputs a directed graph and outputs two items: 1) the number of unique color combinations between adjacent nodes and 2) the graph structure.

**INPUT** - list of edges, one per line, in the form:

```
COLOR_# -> COLOR_#
COLOR_# -> COLOR_#
COLOR_# -> COLOR_#
...
```

COLOR is either red, orange, yellow, green, blue, indigo, or violet. # is a non-negative integer.

**OUTPUT (1)** - count (N) of each unique color combination, in the form:

```
COLOR -> COLOR = N
```

**OUTPUT (2)** - graph structure, in the form:

```
COLOR_#
  ===> COLOR_#
  ===> COLOR_#
COLOR_#
  ===> COLOR_#
...
```

Your solution must:
- Use the /hw3 route for GET and POST requests
- Use a textarea html element to input the graph in the format specified above
- Have a Process button that runs your solution on the given input
- Output #1 in the format specified above
- Output #2 in the format specified above

HINTS:
- Python dictionary
- Node class with name, color, and list of adjacent Nodes
- Work incrementally, start with simple examples
EXAMPLE

NOTE: You are not required to have a Generate button (and associated inputs) that generates a random graph. Also, you are not required to render the graph visually.

Initial site:

Input a directed graph into the textarea (or randomly Generate).
Click “Process”:

### Color Combinations:
- blue -> blue = 5
- blue -> green = 1
- green -> blue = 1
- green -> orange = 1
- orange -> blue = 1
- orange -> green = 1

### Graph Structure

- **blue_1**
  - ==> blue_2
  - ==> blue_3
- **blue_2**
  - ==> blue_1
  - ==> green_0
- **blue_3**
  - ==> blue_1
  - ==> green_0
- **green_0**
  - ==> blue_3
  - ==> orange_4
- **orange_4**
  - ==> blue_3
  - ==> green_0