Python Reader

Back to Code in Place

Nested Stuctures

Ultimate CS106A: Reverse a Dict

A normal dictionary pairs keys to their values. In this problem, we are going to create a new dictionary where the keys are the values of another dictionary, and the values are the other dictionary's keys.

Normal Dict:
Key -> Value
Reversed Dict:
Value -> Keys

Here's the code for how we can do that:

```python
def reverse(original):
    # should create and return a new dictionary where
    # the values of the original are now keys!
    reversed = {}
    for key in original:
        value = original[key]
        if not value in reversed:
            reversed[value] = []
        reversed[value].append(key)
    return reversed
```

The creation of this dictionary gives us an important insight: dictionaries themselves can contain complex data structures as values, like lists!

The XKCD Color Survey

In 2010, the creator of the webcomic XKCD released a survey. In this survey, volunteers around the world were shown a randomly-chosen color, and asked to name the color. After filtering the responses, the result was about 2.8 million RGB triplets and their names. We are going to put together what we learned about dictionaries and files to analyze this data!

Here is how the file was formatted: color-name, red value, green value, blue value.

The data shows that people perceive colors differently. For example, volunteers labeled all of the following RGB triplets as blue: (124,164,176), (33,115,229), (73,97,236), (120,158,209), (41,201,234), (107,148,220).

For reference, here is what those different shades of blue look like:

Back to Code in Place

How can we organize all the data that was gathered? One way to do this is to associate each color name with a list of colors represented by RGB values.

Each set of RGB values is represented by a list. This means that each color name is associated with a list of lists, where each element of the list is another list! The outer list represents a different shade of that color, while the inner list represents the RGB values that make up that color.

Below shows us what that dictionary might look like:

```
{
"clover green": [[100, 216, 135], [72, 218, 111],
[57, 109, 40], [9, 190, 78], [4, 217, 90], [36,
164, 33], [85, 195, 120], [137, 207, 101], [155,
213, 167], [41, 141, 12], [35, 195, 118], [63,
169, 115], [2, 184, 86], [49, 189, 100], [147,
200, 8], [63, 160, 43], [87, 121, 8], [49, 183,
44], [61, 190, 119]],
"sal": [[184, 207, 244], [48, 199, 109], [247, 4,
25], [6, 101, 127], [196, 124, 36], [148, 30, 23],
[106, 51, 249], [186, 63, 96], [209, 234, 226],
[115, 18, 254], [59, 251, 10], [209, 84, 209],
[254, 164, 39], [154, 165, 137], [82, 196, 178],
[120, 250, 248], [175, 59, 33], [67, 52, 126],
[224, 211, 50], [9, 255, 249], [138, 43, 154],
[218, 158, 7], [213, 79, 90]],
"marzipan": [[202, 197, 102], [34, 80, 112], [127,
162, 51], [90, 171, 24], [134, 198, 156], [163,
138, 126], [212, 248, 154], [133, 25, 118], [75,
143, 86], [46, 108, 0], [9, 242, 107], [29, 120,
25], [237, 209, 155], [215, 82, 187], [200, 79,
52], [12, 78, 60], [18, 52, 183], [186, 61, 232],
[169, 201, 232], [173, 216, 142]]
}
```

Below is code we can use to convert information from the file into a dictionary. Notice how this code uses so many of the things you've learned in the past 5 weeks: decomposition, for-each loops, if-statements, lists, dictionaries, and files. You've come such a long way, and we are so proud of everything you've learned!

```python
def load_data():
    data = {}
    file = open(DATA_FILE)
    n_colors = 0
    for line in file:
        line = line.strip()
        add_color(data, line)
        n_colors += 1
    # print(len(data), n_colors)
    file.close()
    return data

def add_color(data, line):
    parts = line.split(',')
    color_name = parts[0]
    color_rgb = color_from_line(line)
    if color_name not in data:
        data[color_name] = []
    data[color_name].append(color_rgb)

def color_from_line(line):
    parts = line.split(',')
    r = int(parts[1])
    g = int(parts[2])
    b = int(parts[3])
    return [r, g, b]
```