tint Documentation

Release 0.3

Christian Schramm

March 12, 2015

Contents

Note: Install *tint* with

```
pip install tint
```

or clone from github.

Match human readable color names to sRBG values (and vice versa).

tint was created to solve the problem of normalizing color strings of various languages and systems to a well defined set of color names. In order to do that, *tint* establishes a registry for color names and corresponding sRGB values. *tint* ships with default color definitions for english ("en"), which is mostly constructed from wikipedia plus the webcolors definitions.

You may query the aforementioned registry for a sRGB hex value by passing it a color name – if there's no exact match, a fuzzy match is applied. Together with the hex value, a matching score is returned: 100 is best (exact match), lower means worse.

Also, you may want to find the best name for a given sRGB value. Again, *tint* tries to match exactly, and failing that, it will find a color name with the minimal perceptual difference according to CIEDE2000. The color name and the color distance are returned: A distance of 0 is best, higher means worse.

Examples

```
>>> import tint
>>> tint_registry = tint.TintRegistry()
>>> tint_registry.match_name("a darker greenish color")
MatchResult(hex_code=u'013220', score=66)
>>> tint_registry.find_nearest("013220", "en")
FindResult(color_name=u'dark green', distance=0)
>>> tint_registry.add_colors("limited", [("cyan", "00ffff"), ("yellow", "ffff00")])
>>> tint_registry.find_nearest("013220", system="limited")
FindResult(color_name=u'cyan', distance=72.54986912349503)
```

class `tint.`**`TintRegistry`**(*load_defaults=True*)

A registry for color names, categorized by color systems.

Parameters **load_defaults** (*bool, optional*) – Load default color systems provided by *tint*. Currently, only "en" is provided by default. Defaults to True.

`add_colors`(*system*, *colors*)

Add color definition to a given color system.

You may add to already existing color system. Previously existing color definitions of the same (normalized) name will be overwritten, regardless of the color system.

Parameters

- **system** (*string*) – The color system the colors should be added to (e.g. `"en"`).
- **color_definitions** (*iterable of tuples*) – Color name / sRGB value pairs (e.g. `[("white", "ffffff"), ("red", "ff0000")]`)

Examples

```
>>> color_definitions = {"greenish": "336633", "blueish": "334466"}
>>> tint_registry = TintRegistry()
>>> tint_registry.add_colors("vague", color_definitions.iteritems())
```

add_colors_from_file(*system*, *f_or_filename*)
Add color definition to a given color system.

You may pass either a file-like object or a filename string pointing to a color definition csv file. Each line in that input file should look like this:

```
café au lait,a67b5b
```

i.e. a color name and a sRGB hex code, separated by by comma (`,`). Note that this is standard excel-style csv format without headers.

You may add to already existing color system. Previously existing color definitions of the same (normalized) name will be overwritten, regardless of the color system.

Parameters

- **system** (*string*) – The color system the colors should be added to (e.g. `"en"`).
- **color_definitions** (*filename, or file-like object*) – Either a filename, or a file-like object pointing to a color definition csv file (excel style).

find_nearest(*hex_code*, *system*, *filter_set=None*)
Find a color name that's most similar to a given sRGB hex code.

In normalization terms, this method implements "normalize an arbitrary sRGB value to a well-defined color name".

Parameters

- **system** (*string*) – The color system. Currently, *"en"*' is the only default system.
- **filter_set** (*iterable of string, optional*) – Limits the output choices to fewer color names. The names (e.g. `["black", "white"]`) must be present in the given system. If omitted, all color names of the system are considered. Defaults to None.

Returns A named tuple with the members *color_name* and *distance*.

Raises `ValueError` – If argument *system* is not a registered color system.

Examples

```
>>> tint_registry = TintRegistry()
>>> tint_registry.find_nearest("54e6e4", system="en")
FindResult(color_name=u'bright turquoise', distance=3.730288645055483)
>>> tint_registry.find_nearest("54e6e4", "en", filter_set=("white", "black"))
FindResult(color_name=u'white', distance=25.709952192116894)
```

match_name(*in_string*, *fuzzy=False*)
Match a color to a sRGB value.

The matching will be based purely on the input string and the color names in the registry. If there's no direct hit, a fuzzy matching algorithm is applied. This method will never fail to return a sRGB value, but depending on the score, it might or might not be a sensible result – as a rule of thumb, any score less then 90 indicates that there's a lot of guessing going on. It's the callers responsibility to judge if the return value should be trusted.

In normalization terms, this method implements "normalize an arbitrary color name to a sRGB value".

Parameters

- **in_string** (*string*) – The input string containing something resembling a color name.

- **fuzzy** (*bool, optional*) – Try fuzzy matching if no exact match was found. Defaults to `False`.

Returns A named tuple with the members *hex_code* and *score*.

Raises `ValueError` – If `fuzzy` is `False` and no match is found

Examples

```
>>> tint_registry = TintRegistry()
>>> tint_registry.match_name("rather white", fuzzy=True)
MatchResult(hex_code=u'ffffff', score=95)
```

t

A

F

M

T