WhatColorIsX Documentation

Release 1.0.1

Tom Milligan

November 24, 2015

Contents

WhatColorIsX is a simple python module that aims to answer one question well - what colour is this string? Useful for automating colour generation for multiple items, WhatColorIsX can also be used to examine local files.

Note: All variable, function and object names in WhatColorIsX use the American spelling, **color**, for consistency with other code.

Contents:

1.1 About

WhatColorIsX was initially developed to replace the boring task of manually assigning colours to objects on another project. The values it returns are generally more relevant than a randomly generated colour - although it throws up some surprises sometimes!

Useage can be as simple as `whatcoloris string` from the command-line; see the *Examples* section for more options.

1.1.1 Thanks

Thanks to Valentine Lab for the colour module, which is super easy to use and partly inspired this module. If you need to post-process the output of WhatColorIsX, I highly recommend giving it a look for it's lightweight simplicity.

And as always, thanks to all the contributors of Pillow, for their hard work.

1.2 Installation

```
$ pip install WhatColorIsX
```

You may find you need to `pip install Pillow` as a dependency first, although it will be attempted automatically.

1.3 Examples

1.3.1 Import to your project

For almost all cases, call the `new()` factory function, then get the colour value from the *`color()`* method:

```
import WhatColorIsX

brick = WhatColorIsX.new('brick')
brick_color = brick.color()
fish = WhatColorIsX.new('fish')
fish_color_bright = fish.color(bright_hue=True)
```

If you already have PIL images that you want to process, you can use the same syntax:

```
from WhatColorIsX import whatcoloris_image
from PIL import Image

img = Image.open('images/cat.jpg')
cat = WhatColorIsX.new(img)
cat_color = cat.color()
```

1.3.2 Run from the command line

Use the *whatcoloris command*:

```
$ whatcoloris sky
#769ab8
$ whatcoloris images/dog.png
#6c5a47
$ whatcoloris grass -b
#65ff00
```

1.3.3 Visual Demo

Using this python script, a folder of image files can be composited along with their calculated colours. The main function of WhatColorIsX is to do this *without* a source image, using only a string.

See an example output here.

1.4 Reference

1.4.1 `WhatColorIsX` Module

The *`WhatColorIsX`* module provides an object of the same name (lowercase), which can determining the colour of:

- A string
- A local file
- A `PIL.Image.Image`

The `whatcolorisx` Class

class `WhatColorIsX.`**`whatcolorisx`**(*input*, *images_to_try=10*)

The whatcolorisx object. Can also be created by the `new()` factory function.

Parameters

- **`input`** (*string*) – The search term to pass to Google image search. If given with a .jpg or .png extension, it is treated as a local file path. Will also accept a *PIL.Image.Image* object.
- **`images_to_try`** (*int*) – The number of images to try processing before raising *`InvalidSearchResults`*

Returns An *`whatcolorisx`* object.

Raises *`InvalidSearchResults`* if no valid image is returned by the search

Methods

`whatcolorisx.`**`color`**`(`*bright_hue=False*, *method='average_color'*`)`

Returns the colour of *`whatcolorisx.img`*.

If bright_hue is set to True, a bright hue will be returned.

Parameters

- **`bright_hue`** (*bool*) – force a bright colour value *(saturation = 1.0, luminance = 0.5)*
- **`method`** (*string*) – The helper method that will pick the colour from the image. Options are *`average_color()`* or *`common_color()`*

Returns the guessed colour of the input string in 6-digit hexadecimal format *(e.g. #ffffff)*

Return type string

Helper methods

`whatcolorisx.`**`average_color`**`()`

Returns the average colour of *`whatcolorisx.img`*.

Recommended for most uses.

Returns RGB value in a three-member tuple

Return type tuple

`whatcolorisx.`**`common_color`**`()`

Returns the most common colour of *`whatcolorisx.img`*.

Not recommended for complex images which may be over or under-exposed; there is a high chance a black or white color will be returned.

Returns RGB value in a three-member tuple

Return type tuple

Attributes

`whatcolorisx.`**`input`**

The initial input to the *`whatcolorisx`* object.

`whatcolorisx.`**`img`**

The `PIL.Image.Image` image generated from *`input`*.

Exceptions

exception `WhatColorIsX.`**`InvalidSearchResults`**

Raised if no valid image is returned by Google Search

1.4.2 `whatcoloris` command

The *whatcoloris command* can be run from the command-line, and provides quick use of the *`WhatColorIsX.whatcolorisx.color()`* method.

Usage

```
$ whatcoloris -h
usage: whatcoloris [-h] [-b] [-m {average_color,common_color}]
                   [--images_to_try IMAGES_TO_TRY]
                   x

Returns colour of string based on Google image search.

positional arguments:
  x                     string/file to find colour of

optional arguments:
  -h, --help            show this help message and exit
  -b, --bright_hue      return a bright colour; hsl=(x,1.0,0.5)
  -m {average_color,common_color}, --method {average_color,common_color}
                        Helper method to use for colour picking. Defaults to
                        average
  --images_to_try IMAGES_TO_TRY
                        number of images to try processing before erroring
```

1.5 Development

1.5.1 Installation

WhatColorIsX can be installed for development as normal:

- clone the GitHub repo
- run `python setup.py develop`
- install dev dependencies using `pip install -r requirements_dev.txt`.

1.5.2 Roadmap

Some ideas:

- improve relevance of colour value
  - discard/differentiate background
  - look at center of image
- return list of *n* colour suggestions
  - use multiple images *(heavy internet, light computation)*
  - use clustering/peak detection *(light internet, heavy computation/installation size)*

1.5.3 Guidelines

Please ensure any new code you write:

- is documented
  - has docstrings in the source code

  - is added to the `docs` (prefrably using autodoc)
  - `sphinx-build -b html .  ./_build` to check html output
- is covered by tests
  - write tests and add them to `tests`
  - run tests using `nosetests` or `coverage run source=WhatColorIsX.py setup.py test`
  - check coverage using `coverage report`

Pull Requests on GitHub are always welcome!

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index