PyColorTerm Documentation

Release 0.2.1

Diego Navarro Mellén

November 05, 2013

Contents

Contents:

CHAPTER 1

PyColorTerm

PyColorTerm allows you to write colored and styled lines out in the terminal from Python and in a pythonic way

- Free software: BSD license
- Documentation: http://pycolorterm.rtfd.org.

1.1 Features

- Get your line prints pretty with color and style formatting
- Python 3 ready!
- Pythonic

1.2 Getting started

1.2.1 Installation

```
$ pip install pycolorterm
```

1.2.2 Usage

```
from pycolorterm.pycolorterm import print_pretty

print_pretty('This is <BOLD>awesome<END> <FG_RED>because<END> you can <UNDERSCORE>mix<END> <BG_BLUE>m
```

1.2.3 Result

1.3 Did you like it?

CHAPTER 2

Installation

At the command line:

```
$ easy_install pycolorterm
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv pycolorterm
$ pip install pycolorterm
```

CHAPTER 3

Usage

To use PyColorTerm in a project

3.1 Using shortcut function

```
from pycolorterm.pycolorterm import print_pretty

print_pretty('{FG_RED}This is a test in RED')

print_pretty('{FG_RED}This{END} is a {BG_BLUE}{UNDERSCORE}more{END} complex {FG_GREEN}{BOLD}test')
```

New in version 0.2.1: Changed the markup from *{}* to <> to allow string pre-formatting

```
print_pretty('<FG_RED>You can add {}<END>'.format('Variables here'))
```

3.2 Using context manager

```
from pycolorterm.pycolorterm import pretty_output, styles

with pretty_output(styles['FG_RED']) as out:
    out.write('This is a test in RED')

with pretty_output(styles['FG_BLUE']) as out:
    out.write('This is a test in BLUE')

with pretty_output(styles['BOLD'], styles['FG_GREEN']) as out:
    out.write('This is a bold text in green')

with pretty_output(styles['BOLD'], styles['BG_GREEN']) as out:
    out.write('This is a text with green background')

with pretty_output(styles['FG_GREEN']) as out:
    out.write('This is a green text with ' + styles['BOLD'] + 'bold' + styles['END'] + ' text include

with pretty_output() as out:
    out.write(styles['BOLD'] + 'Use this' + styles['END'] + ' even with ' + styles['BOLD'] + styles['
```

```
with pretty_output() as out:
    out.write('This is {BOLD}awesome{END} {FG_RED}because{END} you can {UNDERSCORE}mix{END} {BG_BLUE}
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/dnmellen/pycolorterm/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

PyColorTerm could always use more documentation, whether as part of the official PyColorTerm docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/dnmellen/pycolorterm/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *pycolorterm* for local development.

1. Fork the *pycolorterm* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/pycolorterm.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv pycolorterm
   $ cd pycolorterm/
   $ python setup.py develop
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
  $ flake8 pycolorterm tests
        $ python setup.py test
  $ tox

To get flake8 and tox, just pip install them into your virtualenv.
```

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/dnmellen/pycolorterm/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_pycolorterm
```

CHAPTER 5

Credits

5.1 Development Lead

- Diego Navarro Mellén <dnmellen@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2013-09-20)

- First release on PyPI.

6.2 0.1.1 (2013-09-20)

- Revision with 100% code coverage

6.3 0.2.0 (2013-09-24)

- Added new shortcut function to print with format (print_pretty)

6.4 0.2.1 (2013-09-28)

- Changed markup syntax from { } to <>

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*