

---

# ansicolortags Documentation

*Release public*

**Lilian Besson**

07/06/2016, 07h:38m:10s

## Contents

<b>1</b>	<b>Installation</b>	<b>1</b>
1.1	Dependencies . . . . .	1
1.2	How to install it ? . . . . .	1
	1. Directly, with pip . . . . .	1
	2. Download, extract, and use setup.py . . . . .	2
	3. Without installing it . . . . .	2
<b>2</b>	<b>Examples</b>	<b>2</b>
2.1	The function <code>printc</code> . . . . .	2
2.2	<code>python -m ansicolortags --help</code> . . . . .	2
2.3	<code>python -m ansicolortags --test</code> . . . . .	3
2.4	<code>--ANSI</code> or <code>--noANSI</code> option . . . . .	4
2.5	<code>python -m ansicolortags --generate</code> . . . . .	4
<b>3</b>	<b>Complete documentation</b>	<b>5</b>
<b>4</b>	<b>Author</b>	<b>5</b>
<b>5</b>	<b>Index and tables</b>	<b>5</b>
<b>6</b>	<b>Table of contents</b>	<b>6</b>
6.1	Installation . . . . .	6
	Dependencies . . . . .	6
	How to install it ? . . . . .	6
	1. Directly, with pip . . . . .	6
	2. Download, extract, and use setup.py . . . . .	6
	3. Without installing it . . . . .	6
6.2	ansicolortags module . . . . .	6
	List of functions . . . . .	7
	To print a string . . . . .	7
	To clean the terminal or the line . . . . .	7
	Others functions . . . . .	7
	Example of use (module) . . . . .	7
	Example of use (script) . . . . .	8
	Auto detection . . . . .	8
	About . . . . .	8
	Complete documentation . . . . .	8

Copyright . . . . .	9
6.3 TODOs for ansicolortags package . . . . .	11
Other changes . . . . .	11
<b>7 The MIT License (MIT)</b>	<b>11</b>
<b>8 Cloud of words</b>	<b>12</b>
<b>Python Module Index</b>	<b>13</b>

---

Welcome to the documentation for `ansicolortags`, a Python 2 or 3 module to use ANSI colors in a terminal.

- This project is open-source [here on BitBucket](#).
  - This project is also available from Pypi, so a quick overview and the last release can be downloaded [from Pypi](#): <https://pypi.python.org/pypi/ansicolortags> !
- 

# 1 Installation

## 1.1 Dependencies

The project is entirely written in pure Python, supporting both version 2 (2.7+) and version 3 (3.1+). For more details about the Python language, see [its official website](#).

## 1.2 How to install it ?

There is three different ways:

### 1. Directly, with pip

The preferred way is to install `ansicolortags` directly with `pip` ([what is pip?](#)):

```
pip install ansicolortags
```

On GNU/Linux, it might be necessary to give it `sudo` rights:

```
sudo pip install ansicolortags
```

### 2. Download, extract, and use setup.py

Or you can also :

1. download the file, [from PyPi](#);
2. extract it (with `tar xzfv`, or a graphical solution, like File Roller);
3. then go in the subdirectory (`cd ansicolortags-v0.3/`);
4. and finally with Python distutils setup tool:

```
$ python setup.py install
```

Or maybe with sudo rights, if the first try did not work :

```
$ sudo python setup.py install
```

### 3. Without installing it

Note that installation *is not mandatory* : a third solution is to simply include **JUST** the file `ansicolortags.py`, and embed it in your own projects. The project can be used without installing *anything elsewhere*.

## 2 Examples

### 2.1 The function `printc`

The main function of this module is `printc`, e.g. `printc("my string")` This function works exactly like `print("my string")`.

For instance, a quick description of super hero's costumes can be done like this:

```
>>> printc("<reset><white>Batman's costume is <black>black<white>, Aquaman's costume is <blue>blue<white> and Superman's costume is red and blue")
Batman's costume is black, Aquaman's costume is blue and green, and Superman's costume is red and blue
```

(Sorry, but it is hard to embed colors in the output of a Python command in a Sphinx generated web-pages.)

### 2.2 `python -m ansicolortags --help`

This command shows the help of the script, colored with the tools defined in the script:

```
Terminal - 1:9 bash - 7 Jun, Juin 06 - 10:09:33 - lillian@jarvis:~/publis/ansicolortags.py - (bash4.3)
(10:09:27)lillian@jarvis:~/publis/ansicolortags.py$ python2.7 ./ansicolortags.py
usage: ansicolortags.py [-h] [--version] [--noANSI] [--ANSI] [-t] [-g]
                        [-f FILE]

ANSI Colors utility module and script (ansicolortags.py).

optional arguments:
  -h, --help            show this help message and exit
  --version            show program's version number and exit
  --noANSI            If present, ANSI escape code from ansicolortags are 'disabled'.
  --ANSI              If present, ANSI escape code from ansicolortags are 'forced' to be printed (even if the output is detected to be a pipe).
  -t, --test          Launch a complete test of all ANSI Colors code defined here.

Generation of a GNU Bash color aliases file:
  -g, --generate      Print all ANSI Colors as 'export name = value'.
  -f FILE, --file FILE If present, and with --generate option, don't print the values, but export them in the file FILE (e.g. FILE = ~/.color.sh)

Use this file ~/color.sh with other GNU Bash scripts, simply by sourcing him with:
source ~/color.sh # in a GNU Bash script

About the convention for the names of the colors:
- for the eight colors black, red, green, yellow, blue, magenta, cyan, white:
  + the name in minuscule is for color with bold (example 'yellow'),
  + the name starting with 'B' is for color without bold (example 'Byellow'),
  + the name starting with a capital letter is for the background color (example 'Yellow');
- for the special effects (blink, italic i), bold (b), underline (u), negative, **not always supported**:
  + the name in minuscule is to turn on the effect (example 'u' to underline),
  + the name starting in capital letter is to turn down the effect (example 'U' to stop underline);
- for the other special effects (nocolors, default, Default, clear, el), the effect is **immediate** (and seems to be well supported).

About
====
This project can be found on-line:
- here on BitBucket : https://bitbucket.org/lbesson/ansicolortags.py,
- here on PyPI : https://pypi.python.org/pypi/ansicolortags,
- and its documentation can be found here on Read the Docs : http://ansicolortags.readthedocs.io/.

The reference page for ANSI code is : http://en.wikipedia.org/wiki/ANSI\_escape\_code.

Copyright
====
Version 0.1, (C) 2012-2016, Lillian Besson.
```

## 2.3 python -m ansicolortags --test

This command shows a complete test of all tags defined in the module:

```
Terminal - 1:~/ansicolortags.py - [Jun, Jun 06 - 10:13:06] dillan@jarvis:~/publis/ansicolortags.py - (bashv4.3)
10:12:34 lillian@jarvis:~/publis/ansicolortags.py$ python3 -m ansicolortags.py --test
Launching full test for ANSI colors, now the text is printed with default value of the terminal...
The color 'black' is used to make the following effect : !! This is a sample text for 'black' !!...
The color 'red' is used to make the following effect : !! This is a sample text for 'red' !!...
The color 'green' is used to make the following effect : !! This is a sample text for 'green' !!...
The color 'yellow' is used to make the following effect : !! This is a sample text for 'yellow' !!...
The color 'blue' is used to make the following effect : !! This is a sample text for 'blue' !!...
The color 'magenta' is used to make the following effect : !! This is a sample text for 'magenta' !!...
The color 'cyan' is used to make the following effect : !! This is a sample text for 'cyan' !!...
The color 'white' is used to make the following effect : !! This is a sample text for 'white' !!...
The color 'Bblack' is used to make the following effect : ...
The color 'Bred' is used to make the following effect : !! This is a sample text for 'Bred' !!...
The color 'Bgreen' is used to make the following effect : !! This is a sample text for 'Bgreen' !!...
The color 'BYellow' is used to make the following effect : !! This is a sample text for 'BYellow' !!...
The color 'Bblue' is used to make the following effect : !! This is a sample text for 'Bblue' !!...
The color 'Bmagenta' is used to make the following effect : !! This is a sample text for 'Bmagenta' !!...
The color 'Bcyan' is used to make the following effect : !! This is a sample text for 'Bcyan' !!...
The color 'Bwhite' is used to make the following effect : !! This is a sample text for 'Bwhite' !!...
The color 'Bblack' is used to make the following effect : !! This is a sample text for 'Black' !!...
The color 'Red' is used to make the following effect : !! This is a sample text for 'Red' !!...
The color 'Green' is used to make the following effect : !! This is a sample text for 'Green' !!...
The color 'Yellow' is used to make the following effect : !! This is a sample text for 'Yellow' !!...
The color 'Blue' is used to make the following effect : !! This is a sample text for 'Blue' !!...
The color 'Magenta' is used to make the following effect : !! This is a sample text for 'Magenta' !!...
The color 'Cyan' is used to make the following effect : !! This is a sample text for 'Cyan' !!...
The color 'White' is used to make the following effect : ...
The color 'Blink' is used to make the following effect : !! This is a sample text for 'Blink' !!...
The color 'Blink' is used to make the following effect : !! This is a sample text for 'Blink' !!...
The color 'nocolors' is used to make the following effect : !! This is a sample text for 'nocolors' !!...
The color 'default' is used to make the following effect : !! This is a sample text for 'default' !!...
The color 'Default' is used to make the following effect : !! This is a sample text for 'Default' !!...
The color 'Italic' is used to make the following effect : !! This is a sample text for 'Italic' !!...
The color 'Italic' is used to make the following effect : !! This is a sample text for 'Italic' !!...
The color 'b' is used to make the following effect : !! This is a sample text for 'b' !!...
The color 'B' is used to make the following effect : !! This is a sample text for 'B' !!...
The color 'u' is used to make the following effect : !! This is a sample text for 'u' !!...
The color 'U' is used to make the following effect : !! This is a sample text for 'U' !!...
The color 'neg' is used to make the following effect : !! This is a sample text for 'neg' !!...
The color 'neg' is used to make the following effect : !! This is a sample text for 'neg' !!...
The color 'clear' is used to make the following effect : ...

!! This is a sample text for 'el' !!...
!! This is a sample text for 'clear' !!...

The color 'reset' is used to make the following effect : !! This is a sample text for 'reset' !!...
The color 'bell' is used to make the following effect : !! This is a sample text for 'bell' !!...
The color 'title' is used to make the following effect : ...
The color 'warning' is used to make the following effect : \A!! This is a sample text for 'warning' !!...
The color 'question' is used to make the following effect : \Q!! This is a sample text for 'question' !!...
The color 'ERROR' is used to make the following effect : ERROR!! This is a sample text for 'ERROR' !!...
The color 'WARNING' is used to make the following effect : WARNING!! This is a sample text for 'WARNING' !!...
The color 'INFO' is used to make the following effect : INFO!! This is a sample text for 'INFO' !!...
```

## 2.4 --ANSI or --noANSI option

You can force to use ANSI codes (even if they appear to not be supported by the output, e.g. a pipe) with the `--ANSI` flag option, or force to disable them with the `--noANSI` flag option:

```
Terminal - 1:~/ansicolortags.py - [Jun, Jun 06 - 10:16:57] dillan@jarvis:~/publis/ansicolortags.py - (bashv4.3)
10:16:51 lillian@jarvis:~/publis/ansicolortags.py$ python3 -m ansicolortags.py --help | head
usage: ansicolortags.py [-h] [--version] [--noANSI] [--ANSI [-t] [-g]]
                        [-f FILE]

ANSI Colors utility module and script (ansicolortags.py).

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  --noANSI              If present, ANSI escape code from ansicolortags are *disabled*.
  --ANSI                If present, ANSI escape code from ansicolortags are *forced* to be printed (even if the output is detected to be a pipe).
10:16:54 lillian@jarvis:~/publis/ansicolortags.py$ python3 -m ansicolortags.py --help --ANSI | head
usage: ansicolortags.py [-h] [--version] [--noANSI] [--ANSI [-t] [-g]]
                        [-f FILE]

ANSI Colors utility module and script (ansicolortags.py).

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  --noANSI              If present, ANSI escape code from ansicolortags are *disabled*.
  --ANSI                If present, ANSI escape code from ansicolortags are *forced* to be printed (even if the output is detected to be a pipe).
```

```
Terminal-1:9:ansicolortags.py- "[Jun. juin 06 - 10:16:25]<lilian@jarvis:~/publis/ansicolortags.py> (bashv4.3)"
[10:16:22]lilian@jarvis#1[~/publis/ansicolortags.py] python3.5 ./ansicolortags.py --help --noANSI
usage: ansicolortags.py [-h] [--version] [--noANSI] [-t] [-g]
                        [-f FILE]
ANSI Colors utility module and script (ansicolortags.py).

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  --noANSI             If present, ANSI escape code from ansicolortags are *disabled*.
  -ANSI               If present, ANSI escape code from ansicolortags are *forced* to be printed (even if the output is detected to be a pipe).
  -t, --test          Launch a complete test of all ANSI Colors code defined here.

Generation of a GNU Bash color aliases file:
  -g, --generate       Print all ANSI Colors as 'export name = value'.
  -f FILE, --file FILE If present, and with --generate option, don't print the values, but export them in the file FILE (e.g. FILE = ~/.color.sh)

Use this file ~/.color.sh with other GNU Bash scripts, simply by sourcing him with:
source ~/.color.sh # in a GNU Bash script

About the convention for the names of the colors:
* for the eight colors black, red, green, yellow, blue, magenta, cyan, white:
+ the name in minuscule is for color **with bold** (example 'yellow'),
+ the name starting with 'B' is for color **without bold** (example 'Byellow'),
+ the name starting with a capital letter is for the background color (example 'Yellow');
* for the special effects (blink, italic (i), bold (b), underline (u), negative), **not always supported**:
+ the name in minuscule is to **turn on** the effect (example 'u' to underline),
+ the name starting in capital letter is to **turn down** the effect (example 'U' to stop underline);
* for the other special effects (nocolors, default, Default, clear, el), the effect is **immediate** (and seems to be well supported).

About
====
This project can be found on-line:
- here on Bitbucket : https://bitbucket.org/lbesson/ansicolortags.py,
- here on PyPI : https://pypi.python.org/pypi/ansicolortags,
- and its documentation can be found here on Read the Docs : http://ansicolortags.readthedocs.io/.

The reference page for ANSI code is : https://en.wikipedia.org/wiki/ANSI\_escape\_code.

Copyrigh
=====
Version 0.1, (C) 2012-2016, Lilian Besson.
```

## 2.5 python -m ansicolortags --generate

This command can be used to generate a .color.sh file, to be used in any GNU Bash script:

```
Terminal-1:9:ansicolortags.py- "[Jun. juin 06 - 10:19:31]<lilian@jarvis:~/publis/ansicolortags.py> (bashv4.3)"
[10:17:39]lilian@jarvis#9[~/publis/ansicolortags.py] python2 ./ansicolortags.py --generate
#!/bin/sh
#
# From ansicolortags.py module, auto generated with the --generate command
# More information on https://bitbucket.org/lbesson/ansicolortags.py
#
# About the convention for the names of the colors :
# * for the eight colors black, red, green, yellow, blue, magenta, cyan, white:
# + the name in minuscule is for color **with bold** (example 'yellow'),
# + the name starting with 'B' is for color **without bold** (example 'Byellow'),
# + the name starting with a capital letter is for the background color (example 'Yellow').
# * for the special effects (blink, italic, bold, underline, negative), **not always supported** :
# + the name in minuscule is to **turn on** the effect,
# + the name starting in capital letter is to **turn off** the effect.
# * for the other special effects (nocolors, default, Default, clear, el), the effect is **immediate** (and seems to be well supported).
#
# About
# =====
# Use this file .color.sh in other GNU Bash scripts, simply by sourcing him with
# $ source ~/.color.sh
#
# Copyrigh
# =====
# (C) Lilian Besson, 2012-2016.
#
# List of colors
# =====
export black="\033[01;30m"
export red="\033[01;31m"
export green="\033[01;32m"
export yellow="\033[01;33m"
export blue="\033[01;34m"
export magenta="\033[01;35m"
export cyan="\033[01;36m"
export white="\033[01;37m"
export Bblack="\033[02;30m"
export Bred="\033[02;31m"
export Bgreen="\033[02;32m"
export Byellow="\033[02;33m"
export Bblue="\033[02;34m"
export Bmagenta="\033[02;35m"
export Bcyan="\033[02;36m"
export Bwhite="\033[02;37m"
export Black="\033[40m"
export Red="\033[41m"
export Green="\033[42m"
export Yellow="\033[43m"
```

## 3 Complete documentation

And, a detailed description of every function is available on the documentation of the module ansicolortags.

## 4 Author

Hi, I am Lilian Besson, a French student at ÉNS de Cachan, in Mathematics and computer science (CS).

If needed, feel free to contact me :

1. either with [this web page](#);
2. or via my bitbucket account `lbesson`;
3. or via email [here](#) (remove the `[]` and change DOT to `.` and AT to `@`).

You can use [this form](#) to inform me of a bug on `ansicolortags.py`:  
<https://bitbucket.org/lbesson/ansicolortags.py/issues?status=new&status=open> !

## 5 Index and tables

- [genindex](#)
- [modindex](#)
- [search](#)

## 6 Table of contents

### 6.1 Installation

#### Dependencies

The project is entirely written in pure Python, supporting both version 2 (2.7+) and version 3 (3.1+). For more details about the Python language, see [its official website](#).

#### How to install it ?

There is three different ways:

##### 1. Directly, with pip

The preferred way is to install `ansicolortags` directly with `pip` ([what is pip?](#)):

```
pip install ansicolortags
```

On GNU/Linux, it might be necessary to give it `sudo` rights:

```
sudo pip install ansicolortags
```

##### 2. Download, extract, and use `setup.py`

Or you can also :

1. download the file, [from PyPi](#);

2. extract it (with `tar xzfv`, or a graphical solution, like File Roller);
3. then go in the subdirectory (`cd ansicolortags-v0.3/`);
4. and finally with Python distutils setup tool:

```
$ python setup.py install
```

Or maybe with sudo rights, if the first try did not work :

```
$ sudo python setup.py install
```

### 3. Without installing it

Note that installation *is not mandatory* : a third solution is to simply include **JUST** the file `ansicolortags.py`, and embed it in your own projects. The project can be used without installing *anything elsewhere*.

## 6.2 ansicolortags module

An efficient and simple ANSI colors module (and also a powerful script), with functions to print text using colors. <https://bitbucket.org/lbesson/ansicolortags.py>

The names of the colors follow these conventions:

- for the eight ANSI colors (black, red, green, yellow, blue, magenta, cyan, white):
  - the name in minuscule is for color **with bold** (example ‘yellow’),
  - the name starting with ‘B’ is for color **without bold** (example ‘Byellow’),
  - the name starting with a **capital** letter is for the background color (example ‘Yellow’).
- for the special effects (blink, *italic*, **bold**, underline, negative), they might not always be supported, but they usually are:
  - the name in minuscule is used to turn *on* the effect (example ‘i’ to turn on italic),
  - the name starting in capital letter is used to turn *down* the effect (example ‘I’ to turn off italic).
- for the other special effects (`nocolors`, `default`, `Default`, `clear`, `el`), the effect is **immediate** (and seems to be well supported).

### List of functions

#### To print a string

- `sprint()` (page 9): give a string,
- `printc()` (page 9): like `print()`, but with interpreting tags to put colors. **This is the most useful function in this module !**
- `writec()` (page 10): like `printc`, but using any file object (and no new line added at the end of the string).

## To clean the terminal or the line

- `erase()` (page 9): erase all ANSI colors tags in the string (like `sprint`, but erasing and not interpreting color tags),
- `clearLine()` (page 9), `clearScreen()` (page 9): to clear the current line or screen,
- `Reset()` (page 9): to return to default foreground and background, and stopping all *fancy* effects (like blinking or reverse video).

## Others functions

- `notify()` (page 9): try to display a *system* notification. **Only on GNU/Linux with notify-send installed.**
- `xtitle()` (page 11): try to set the *title* of the terminal. Warning: **not always supported.**

## Example of use (module)

To store a string, use `sprint()` (page 9) (i.e. print to a string, *sprint*), like this:

```
>>> example = sprint("France flag is <blue>blue<white>white<red>red<white>, Italy flag have <green>green")
```

The string `example` can then be printed, with colors, with:

```
>>> print(example) # Sorry, but in the documentation it is hard to show colors :)
France flag is bluewhitered, Italy flag have green on it.
```

To directly print a string colored by tags, use `printc()` (page 9):

```
>>> printc("Batman's costum is <black>black<white>, Aquaman's costum is <blue>blue<white> and <green>green")
```

### See also:

This is the most useful function. To do the same, but on any file, use `writelc()` (page 10).

Moreover, the function `erase()` (page 9) can also be useful to simply delete all *valid* color tags

```
>>> print(erase("Batman's costum is <black>black<white>, Aquaman's costum is <blue>blue<white> and <green>green")
Batman's costum is black, Aquaman's costum is blue and green, and this is a non-valid <tag>, so it is not printed")
```

In this last example, an `<el>` tag (`el`) is used to erase the current content of the line, useful to make a *dynamical* print

```
>>> writelc("<red>Computing <u>len(str(2**562016))<reset>..."); tmp = len(str(2**562016)); writelc("<el>
Done !")
```

The first part of the line 'Computing len(str(2\*\*562016))...' have disappeared after the computation! (which takes about one second).

## Example of use (script)

- To show the help `$ ansicolortags.py --help;`
- To run a test `$ ansicolortags.py --test;`
- To produce a GNU Bash color aliases file `$ ansicolortags.py --generate --file ~/.color_aliases.sh.`



## Auto detection

This script can normally detect if ANSI codes are supported :

1. `$ ansicolortags.py --help` : will print with colors if colors seems to be supported;
2. `$ ansicolortags.py --help --noANSI` : will print without any colors, even if it is possible;
3. `$ ansicolortags.py --help --ANSI` : will force the use of colors, even if they seems to be not supported.

And, the module part behaves exactly like the script part.

---

## About

This project can be found on-line:

- [here on BitBucket](#);
- [here on PyPi](#).
- The reference page for ANSI code is : [here on Wikipedia](#).
- The reference page for XTitle escape code is : [here](#).

## Complete documentation

The doc is available on-line, on [Read the Docs](http://ansicolortags.readthedocs.io/) : <http://ansicolortags.readthedocs.io/> !

## Copyrigh

© Lilian Besson, 2012-2016.

`ansicolortags.Reset()` → unit

Try to reset the current ANSI codes buffer, using `reset`.

`ansicolortags.clearLine()` → unit

Try to clear the current line using ANSI code `el`.

`ansicolortags.clearScreen()` → unit

Try to clear the screen using ANSI code `clear`.

`ansicolortags.erase(chainWithTags, left='<', right='>', verbose=False)` → string

Parse a string containing color tags, when color is one of the previous define name, and then return it, with color tags **erased**.

Example:

```
>>> print(erase("<blue>This is blue.<white> And <this> is white.<red> Now this is red because I  
This is blue. And <this> is white. Now this is red because I am <angry> !")
```

This example seems exactly the same that the previous one in the documentation, but it's not (it is impossible to put color in the output of a Python example in Sphinx documentation, so there is **no color in output** in the examples... but be sure there is the real output !).

`ansicolortags.notify` (*msg*='', *obj*='Notification sent by `ansicolortags.notify`', *icon*=None, *verb*=False) → bool  
Notification using subprocess and `notify-send` (GNU/Linux command-line program). Also print the informations directly to the screen (only if *verb*=True).

**Warning:** This does not use any *ANSI escape* codes, but the common *notify-send* GNU/Linux command line program. It will probably fail (but cleanly) on Windows or Mac OS X.

- Return True if and only if the title have been correctly changed.
- Fails simply if `notify-send` is not found.

`ansicolortags.printc` (*chainWithTags*, *left*='<', *right*'>', *\*\*kw*) → unit  
A shortcut to `print(sprint(chainWithTags))` : analyze all tags (convert the tags like <red> to their ANSI code value, like red), and print the result.

This is the more useful function in this package.

**Hint:** I suggest to use `ansicolortags.py` in your own project with the following piece of code:

```
try:
    from ansicolortags import printc
except ImportError:
    print("WARNING: ansicolortags was not found, disabling colors instead.\nPlease install it with pip")
    def printc(*a, **kw):
        print(*a, **kw)
```

**Hint:** During the last 4 years, a lot of the small Python scripts I wrote try to use this module to add some colors: for example, `FreeSMS.py`, `plotnotes.py`, `strapdown2html.py`, `calc_interets.py`...

`ansicolortags.sprint` (*chainWithTags*, *left*='<', *right*'>', *verbose*=False) → string  
Parse a string containing color tags, when color is one of the previous define name, and then return it, with color tags changed to concrete ANSI color codes.

**Tags are delimited** by `left` and `right`. By default, it's *HTML / Pango style* whit '<' and '>', but you can change them.

For example, a custom style even closer to HTML could be: `left='<span color='` and `right = '</span>'` is also possible.

**Warning:** It is more prudent to put nothing else than ANSI Colors (i.e. values in `colorList`) between '<' and '>' in `chainWithTags`. The behavior of the function in case of false tags is **not perfect**. Moreover, a good idea could be to try not to use '<' or '>' for anything else than tags. I know, it's not perfect. But, the syntax of color tags is so simple and so beautiful with this limitation that you will surely forgive me this, *won't you* ;) ?

Example:

```
>>> print(sprint("<blue>This is blue.<white> And <this> is white.<red> Now this is red because I am <angry> !"))
This is blue. And <this> is white. Now this is red because I am <angry> !
```

This function is used in all the following, so all other function can also use `left` and `right`

`ansicolortags.tocolor(mystring) → string`

Convert a string to a color. `mystring` **have to be** in `colorDict` to be recognized (and interpreted). Default value if `mystring` is not one of the color name is "" the empty string.

`ansicolortags.writec(chainWithTags="" , out=sys.stdout, left='<', right='>', flush=True) → unit`

Useful to print colored text **to a file**, represented by the object `out`. Also useful to print colored text, but without any trailing 'n' character.

Example:

In this example, before the long computation begin, it prints 'Computing  $2^{(2^{(2^{2^4})})}$ ....', and when the computation is done, erases the current line (with `<el>` tag, `el`), and prints ' Done !' in green, and the result of the computation:

```
>>> writec("<red>Computing<reset> 2**(2**(2**4))...."); tmp = 2**(2**(2**4)); writec("<el><green> Done !")
```

This example show how to use this module to write colored data in a file. Be aware that this file now contains ANSI escape sequences. For example, `$ cat /tmp/colored-text.txt` will well print the colors, but editing the file will show *hard values* of escape code:

```
>>> my_file = open('/tmp/colored-text.txt', mode = 'w') # Opens an random file.
>>> write("<blue>this is blue.<white>And <this> is white.<red>Now this is red because I am <angr>")
```

Remark: Can also be used to simply reinitialize the ANSI colors buffer, but the function `Reset()` (page 9) is here for this:

```
>>> writec("<reset>")
```

**Warning:** The file `out` **will be flushed** by this function if `flush` is set to `True` (this is default behavior). If you prefer no to, use `flush=False` option:

```
>>> writec(chainWithTags_1, out=my_file, flush=False)
>>> # many things...
>>> writec(chainWithTags_n, out=my_file, flush=False)
>>> my_file.flush() # only flush here!
```

`ansicolortags.xtitle(new_title="" , verb=False) → 0 or 1`

**Modify the current terminal title.** Returns 0 if one of the two solutions worked, 1 otherwise.

An experimental try is with **ANSI escape code**, if the simple way by calling the `xtitle` program does not work (or if it is not installed).

**Note:** The second solution simply uses the two *ANSI Tags* `<title>` (title) and `<bell>` (bell). So, you can also do it with:

```
>>> ansicolortags.writec("<title>This is the new title of the terminal<bell>")
```

But this function `xtitle` is better: it tries two ways, and returns a signal to inform about his success.

## 6.3 TODOs for ansicolortags package

### Other changes

1. Improve support of Windows.
2. Add automated tests.

3. Dirty to use all these `exec()`, should be avoided... Find something cleaner!

---

This project is currently in version 0.1, release public. Last update of this doc was made 07/06/2016, 07h:38m:10s.

---

**Note:** This project is based on my old [ANSIColors-balises](#) project, which was only for Python 2.7.

---

## 7 The MIT License (MIT)

The MIT License (MIT) Copyright © 2016 Lilian Besson (Naareen), <https://bitbucket.org/lbesson/> <naareen at crans dot org>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

---



## Python Module Index

### a

ansicolortags, 6

## Index

### A

ansicolortags (module), 6

### C

clearLine() (in module ansicolortags), 9

clearScreen() (in module ansicolortags), 9

### E

erase() (in module ansicolortags), 9

### N

notify() (in module ansicolortags), 9

### P

putc() (in module ansicolortags), 9

### R

Reset() (in module ansicolortags), 9

### S

sprint() (in module ansicolortags), 9

### T

toColor() (in module ansicolortags), 10

### W

wputc() (in module ansicolortags), 10

### X

xtitle() (in module ansicolortags), 11