Colorize the Windows Console

André Burgaud

2009-05-10

The recipe described in this post demonstrates how to write foreground and background colored text in a Windows Command Line with Python and `ctypes`.

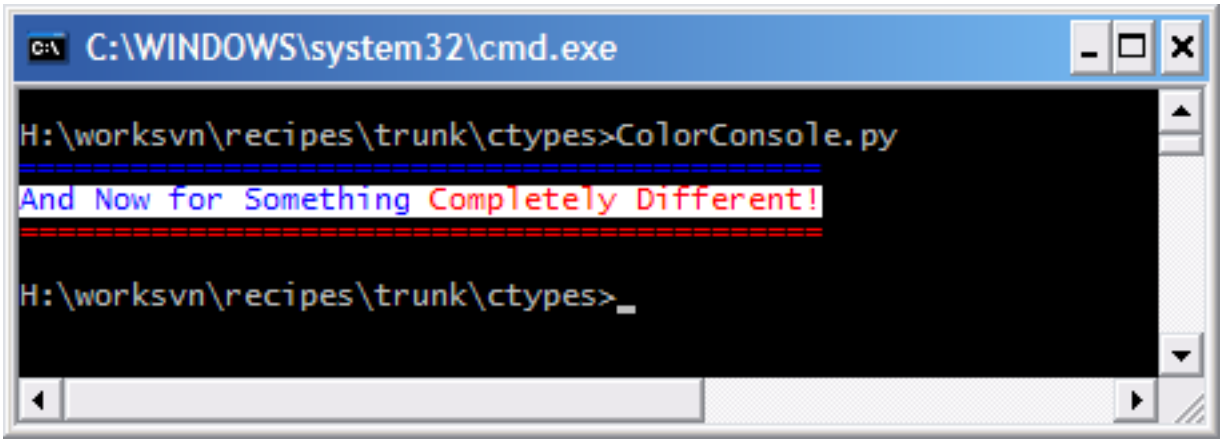

Figure 1: Colors in the Windows Terminal

I originally wrote this article in the, now defunct, `ctypes` wiki. I decided to update the corresponding code and to publish it on my personal web site.

ctypes is a foreign function library for Python written by Thomas Heller. It is part of the Python standard library since Python 2.5, but works also with Python 2.3 and 2.4 provided that you download the corresponding libraries available from SourceForge.

Often overlooked on Windows, command line programs are very powerful, but at times, a little boring or simply overwhelming when too much information is displayed at once. Having the ability to change the color and intensity of the text allows to bring contrast to the information output on the screen. for example, UPX (the Ultimate Packer for eXecutables) takes advantage of this concept and it inspired me to introduce this feature into my Python command line programs.

Coloring text in a command line window is not available by default in Python. Using ctypes allows to call the appropriate Win32 library functions, in particular SetConsoleTextAttribute and GetConsoleScreenBufferInfo.

Here is a code example showing a way to wrap `SetConsoleTextAttribute()` and `GetConsoleScreenBufferInfo()`:

color_console.py

```
"""
Colors text in console mode application (win32).
Uses ctypes and Win32 methods SetConsoleTextAttribute and
GetConsoleScreenBufferInfo.
```

```
$Id: color_console.py 534 2009-05-10 04:00:59Z andre $
"""

from ctypes import windll, Structure, c_short, c_ushort, byref

SHORT = c_short
WORD = c_ushort

class COORD(Structure):
  """struct in wincon.h."""
  _fields_ = [
    ("X", SHORT),
    ("Y", SHORT)]

class SMALL_RECT(Structure):
  """struct in wincon.h."""
  _fields_ = [
    ("Left", SHORT),
    ("Top", SHORT),
    ("Right", SHORT),
    ("Bottom", SHORT)]

class CONSOLE_SCREEN_BUFFER_INFO(Structure):
  """struct in wincon.h."""
  _fields_ = [
    ("dwSize", COORD),
    ("dwCursorPosition", COORD),
    ("wAttributes", WORD),
    ("srWindow", SMALL_RECT),
    ("dwMaximumWindowSize", COORD)]

# winbase.h
STD_INPUT_HANDLE = -10
STD_OUTPUT_HANDLE = -11
STD_ERROR_HANDLE = -12

# wincon.h
FOREGROUND_BLACK     = 0x0000
FOREGROUND_BLUE      = 0x0001
FOREGROUND_GREEN     = 0x0002
FOREGROUND_CYAN      = 0x0003
FOREGROUND_RED       = 0x0004
FOREGROUND_MAGENTA   = 0x0005
FOREGROUND_YELLOW    = 0x0006
FOREGROUND_GREY      = 0x0007
FOREGROUND_INTENSITY = 0x0008 # foreground color is intensified.

BACKGROUND_BLACK     = 0x0000
BACKGROUND_BLUE      = 0x0010
BACKGROUND_GREEN     = 0x0020
BACKGROUND_CYAN      = 0x0030
BACKGROUND_RED       = 0x0040
BACKGROUND_MAGENTA   = 0x0050
```

```
BACKGROUND_YELLOW    = 0x0060
BACKGROUND_GREY      = 0x0070
BACKGROUND_INTENSITY = 0x0080 # background color is intensified.

stdout_handle = windll.kernel32.GetStdHandle(STD_OUTPUT_HANDLE)
SetConsoleTextAttribute = windll.kernel32.SetConsoleTextAttribute
GetConsoleScreenBufferInfo = windll.kernel32.GetConsoleScreenBufferInfo

def get_text_attr():
  """Returns the character attributes (colors) of the console screen
  buffer."""
  csbi = CONSOLE_SCREEN_BUFFER_INFO()
  GetConsoleScreenBufferInfo(stdout_handle, byref(csbi))
  return csbi.wAttributes

def set_text_attr(color):
  """Sets the character attributes (colors) of the console screen
  buffer. Color is a combination of foreground and background color,
  foreground and background intensity."""
  SetConsoleTextAttribute(stdout_handle, color)
```

Using the module above to bring some colors to the Windows console can be achieved in Python 2.6 with the following code:

test_color.py (Python 2.6)

```
"""
Test module color_console (Python 2.6). Does not work with Python 3.0 (mainly
due to the usage of print. In Python 3.0 print is a builtin function and no more
a statement.
"""

import color_console as cons

def test():
  """Simple test for color_console."""
  default_colors = cons.get_text_attr()
  default_bg = default_colors & 0x0070
  cons.set_text_attr(cons.FOREGROUND_BLUE | default_bg |
                     cons.FOREGROUND_INTENSITY)
  print '========================================='
  cons.set_text_attr(cons.FOREGROUND_BLUE | cons.BACKGROUND_GREY |
                     cons.FOREGROUND_INTENSITY | cons.BACKGROUND_INTENSITY)
  print 'And Now for Something',
  cons.set_text_attr(cons.FOREGROUND_RED | cons.BACKGROUND_GREY |
                     cons.FOREGROUND_INTENSITY | cons.BACKGROUND_INTENSITY)
  print 'Completely Different!',
  cons.set_text_attr(default_colors)
  print
  cons.set_text_attr(cons.FOREGROUND_RED | default_bg |
                     cons.FOREGROUND_INTENSITY)
  print '========================================='
  cons.set_text_attr(default_colors)
```

```
if __name__ == "__main__":
  test()
```

In Python 3.0, some slight changes are necessary and we obtain:

test_color.py (Python 3)

```
"""
Test module color_console (Python 3.0). Does not work with Python 2.6.
"""

import color_console as cons
import sys

def test():
  """Simple Pyton 3.0 test for color_console."""
  default_colors = cons.get_text_attr()
  default_bg = default_colors & 0x0070
  default_fg = default_colors & 0x0007
  cons.set_text_attr(cons.FOREGROUND_BLUE | default_bg |
                     cons.FOREGROUND_INTENSITY)
  print('================================================')
  cons.set_text_attr(cons.FOREGROUND_BLUE | cons.BACKGROUND_GREY |
                cons.FOREGROUND_INTENSITY | cons.BACKGROUND_INTENSITY)
  print('And Now for Something', end=' ')
  sys.stdout.flush() # Force writing first part of the line in blue
  cons.set_text_attr(cons.FOREGROUND_RED | cons.BACKGROUND_GREY |
                     cons.FOREGROUND_INTENSITY | cons.BACKGROUND_INTENSITY)
  print('Completely Different!')
  cons.set_text_attr(default_colors)
  cons.set_text_attr(cons.FOREGROUND_RED | default_bg |
                     cons.FOREGROUND_INTENSITY)
  print('================================================')
  cons.set_text_attr(default_colors)

if __name__ == "__main__":
  test()
```

It is possible to write the test code compliant with both Python 2.6 and Python 3.0, but I like to use the natural *Python way* and as such prefer to use the new Python 3.0 features where they are and how they are.

Resources

- Source code for this article (released under the MIT License)
- Console Screen Buffers