Colorama

Hace que las secuencias de caracteres de escape ANSI (para producir texto de terminal en color y posicionamiento del cursor) funcionen en MS Windows.

PyPI para versiones | Github para código fuente | Colorama para empresas en Tidelift

Si Colorama le resulta útil, por favor, haga una donación con Paypal a los autores. ¡Gracias!

Instalación

Probado en CPython 2.7, 3.7, 3.8, 3.9 y 3.10 y Pypy 2.7 y 3.8.

No hay requisitos aparte de la biblioteca estándar.

pip install colorama

o

conda install -c anaconda colorama

Descripción

Las secuencias de caracteres de escape ANSI se han utilizado durante mucho tiempo para producir texto de terminal en color y posicionamiento del cursor en Unix y Mac. Colorama también hace que esto funcione en Windows, envolviendo stdout, eliminando las secuencias ANSI que encuentra (que aparecerían como galimatías en la salida) y convirtiéndolas en las llamadas win32 adecuadas para modificar el estado de la terminal. En otras plataformas, Colorama no hace nada.

Esto tiene como resultado proporcionar una API multiplataforma simple para imprimir texto de terminal en color desde Python, y tiene el efecto secundario positivo de que las aplicaciones o bibliotecas existentes que usan secuencias ANSI para producir salida en color en Linux o Mac ahora también pueden funcionar en Windows, simplemente llamando a colorama.just_fix_windows_console() (desde v0.4.6) o colorama.init() (todas las versiones, pero puede tener otros efectos secundarios; consulte a continuación).

Un enfoque alternativo es instalar ansi.sys en máquinas Windows, que proporciona el mismo comportamiento para todas las aplicaciones que se ejecutan en terminales. Colorama está pensado para situaciones en las que eso no es fácil (por ejemplo, tal vez su aplicación no tiene un instalador).

Los scripts de demostración en el repositorio de código fuente imprimen un texto en color utilizando secuencias ANSI. Compare su salida bajo el manejo ANSI integrado de Gnome-terminal, frente al símbolo del sistema de Windows utilizando Colorama:

Secuencias ANSI en Ubuntu bajo gnome-terminal. Las mismas secuencias ANSI en Windows, utilizando Colorama.

Estas capturas de pantalla muestran que, en Windows, Colorama no admite el "texto tenue" ANSI; se ve igual que el "texto normal".

Uso

Inicialización

Si lo único que desea de Colorama es que los escapes ANSI funcionen en Windows, ejecute:

```
from colorama import just_fix_windows_console
just_fix_windows_console()
```

Si está en una versión reciente de Windows 10 o superior, y su stdout/stderr apuntan a una consola de Windows, esto activará el interruptor de configuración mágico para habilitar la compatibilidad con ANSI integrada de Windows.

Si está en una versión anterior de Windows, y su stdout/stderr apuntan a una consola de Windows, esto envolverá sys.stdout o sys.stderr en un objeto de archivo mágico que intercepta las secuencias de escape ANSI y emite las llamadas Win32 adecuadas para emularlas.

En todas las demás circunstancias, no hace nada en absoluto. Básicamente, la idea es que esto hace que Windows actúe como Unix con respecto al manejo de escapes ANSI.

Es seguro llamar a esta función varias veces. Es seguro llamar a esta función en plataformas que no sean Windows, pero no hará nada. Es seguro llamar a esta función cuando uno o ambos de sus stdout/stderr se redirigen a un archivo; no hará nada con esos flujos.

De manera alternativa, puede usar la interfaz anterior con más funciones (pero también más posibles errores de ejecución):

```
from colorama import init
init()
```

Esto hace lo mismo que just_fix_windows_console, excepto por las siguientes diferencias:

No es seguro llamar a init varias veces; puede terminar con múltiples capas de envoltura y compatibilidad con ANSI rota.

Colorama aplicará una heurística para adivinar si stdout/stderr soportan ANSI, y si cree que no, entonces envolverá sys.stdout y sys.stderr en un objeto de archivo mágico que elimina las secuencias de escape ANSI antes de imprimirlas. Esto sucede en todas las plataformas y puede ser conveniente si desea escribir su código para emitir secuencias de escape ANSI incondicionalmente y dejar que Colorama decida si realmente deben imprimirse. Pero tenga en cuenta que la heurística de Colorama no es particularmente inteligente.

init también acepta argumentos de palabras clave explícitos para habilitar/deshabilitar varias funciones; consulte a continuación.

Para dejar de usar Colorama antes de que finalice su programa, simplemente llame a deinit(). Esto restaurará stdout y stderr a sus valores originales, de modo que Colorama se deshabilite. Para volver a usar Colorama, llame a reinit(); es más económico que llamar a init() nuevamente (pero hace lo mismo).

La mayoría de los usuarios deberían depender de colorama >= 0.4.6 y usar just_fix_windows_console. La antigua interfaz init será compatible indefinidamente por compatibilidad con versiones anteriores, pero no tenemos previsto solucionar ningún problema con ella, también por compatibilidad con versiones anteriores.

Salida en color

La impresión multiplataforma de texto en color se puede realizar utilizando la abreviatura constante de Colorama para secuencias de escape ANSI. Estas son deliberadamente rudimentarias, consulte a continuación.

```
from colorama import Fore, Back, Style

print(Fore.RED + 'some red text')

print(Back.GREEN + 'and with a green background')

print(Style.DIM + 'and in dim text')

print(Style.RESET_ALL)

print('back to normal now')
```

...o simplemente imprimiendo manualmente ANS

Secuencias a partir de su propio código:

```
print('\033[31m' + 'some red text')

print('\033[39m') # y restablezca el color predeterminado
```

...o bien, Colorama se puede utilizar junto con bibliotecas ANSI existentes, como la venerable Termcolor, la fabulosa Blessings o la increíble _Rich.

Si desea que las constantes Fore, Back y Style de Colorama sean más capaces, considere utilizar una de las bibliotecas de alta capacidad mencionadas anteriormente para generar colores, etc., y utilice Colorama solo para su propósito principal: convertir esas secuencias ANSI para que también funcionen en Windows:

DE FORMA IGUAL, no envíe PR que agreguen la generación de nuevos tipos ANSI a Colorama. Solo nos interesa convertir códigos ANSI en llamadas API de win32, no en atajos como el mencionado anteriormente para generar caracteres ANSI.

```
from colorama import just_fix_windows_console
from termcolor import coloured

# use Colorama para que Termcolor funcione también en Windows
just_fix_windows_console()

# luego use Termcolor para toda la salida de texto en color
print(colored('¡Hola, mundo!', 'green', 'on_red'))
```

Las constantes de formato disponibles son:

Primer plano: NEGRO, ROJO, VERDE, AMARILLO, AZUL, MAGENTA, CIAN, BLANCO, REINICIAR.

Posterior: NEGRO, ROJO, VERDE, AMARILLO, AZUL, MAGENTA, CIAN, BLANCO, REINICIAR.

Estilo: DIM, NORMAL, BRIGHT, REINICIAR_ALL

Style.RESET_ALL restablece el primer plano, el fondo y el brillo. Colorama realizará este restablecimiento automáticamente al salir del programa.

Estos son bastante compatibles, pero no forman parte del estándar:

Adelante: LIGHTBLACK_EX, LIGHTRED_EX, LIGHTGREEN_EX, LIGHTYELLOW_EX, LIGHTBLUE_EX, LIGHTMAGENTA_EX, LIGHTCYAN_EX, LIGHTWHITE_EX

Atrás: LIGHTBLACK_EX, LIGHTRED_EX, LIGHTGREEN_EX, LIGHTYELLOW_EX, LIGHTBLUE_EX, LIGHTMAGENTA_EX, LIGHTCYAN_EX, LIGHTWHITE_EX

Posicionamiento del cursor

Se admiten códigos ANSI para reposicionar el cursor. Consulte demos/demo06.py para ver un ejemplo de cómo generarlos.

Argumentos de palabras clave de inicialización

init() acepta algunos **kwargs para anular el comportamiento predeterminado.

init(autoreset=False):

Si se encuentra enviando repetidamente secuencias de reinicio para desactivar los cambios de color al final de cada impresión, entonces init(autoreset=True) automatizará eso:

```
from colorama import init
init(autoreset=True)
print(Fore.RED + 'some red text')
print('automatically back to default color again')
```

init(strip=None):

Pase True o False para anular si los códigos ANSI deben eliminarse de la salida. El comportamiento predeterminado es eliminarlos si está en Windows o si la salida está redirigida (no es una tty).

init(convert=None):

Pase True o False para anular si se deben convertir los códigos ANSI en la salida en llamadas win32. El comportamiento predeterminado es convertirlos si está en Windows y la salida es a una tty (terminal).

init(wrap=True):

En Windows, Colorama funciona reemplazando sys.stdout y sys.stderr con objetos proxy, que anulan el método .write() para hacer su trabajo. Si este ajuste le causa problemas, puede desactivarlo pasando init(wrap=False). El comportamiento predeterminado es ajustar si autoreset, strip o convert son True.

Cuando el ajuste está desactivado, la impresión a color en plataformas que no sean Windows seguirá funcionando normalmente. Para generar una salida en color multiplataforma, puede usar el proxy AnsiToWin32 de Colorama directamente:

```
import sys
from colorama import init, AnsiToWin32
init(wrap=False)
stream = AnsiToWin32(sys.stderr).stream

# Python 2
print >>stream, Fore.BLUE + 'blue text on stderr'

# Python 3
print(Fore.BLUE + 'blue text on stderr', file=stream)
```

Secuencias ANSI reconocidas

Las secuencias ANSI generalmente tienen la forma:

```
ESC [ <param> ; <param> ... <command>
```

Donde <param> es un entero y <command> es una sola letra. Se pasan cero o más parámetros a un <command>. Si no se pasan parámetros, generalmente es sinónimo de pasar un solo cero. No existen espacios en la secuencia; Se han insertado aquí simplemente para facilitar su lectura.

Las únicas secuencias ANSI que Colorama convierte en llamadas win32 son:

```
ESC [ 0 m # restablecer todo (colores y brillo)
ESC [ 1 m # brillante
ESC [ 2 m # tenue (se ve igual que el brillo normal)
```

ESC [22 m # brillo normal

PRIMER PLANO:

ESC [30 m # negro

ESC [31 m # rojo

ESC [32 m # verde

ESC [33 m # amarillo

ESC [34 m # azul

ESC [35 m # magenta

ESC [36 m # cian

ESC [37 m # blanco

ESC [39 m # restablecer

FONDO

ESC [40 m # negro

ESC [41 m # rojo

ESC [42 m # verde

ESC [43 m # amarillo

ESC [44 m # azul

ESC [45 m # magenta

ESC [46 m # cian

ESC [47 m # blanco

ESC [49 m # restablecer

posicionamiento del cursor

ESC [y;x H # posición del cursor en x horizontal, y vertical

ESC [y;x f # posición del cursor en x horizontal, y vertical

ESC [n A # mover el cursor n líneas hacia arriba

ESC [n B # mover el cursor n líneas hacia abajo

ESC [n C # mover el cursor n caracteres hacia adelante

ESC [n D # mover el cursor n caracteres hacia atrás

limpiar la pantalla

ESC [modo J # limpiar la pantalla

limpiar la línea

ESC [modo K # limpiar la línea

Se pueden combinar varios parámetros numéricos del comando 'm' en una sola secuencia:

ESC [36 ; 45 ;

1 m # texto cian brillante sobre fondo magenta

Todas las demás secuencias ANSI del formato ESC [<param> ; <param> ... <command> se eliminan silenciosamente de la salida en Windows.

No se reconocen ni se eliminan otras formas de secuencia ANSI, como códigos de un solo carácter o caracteres iniciales alternativos. Sin embargo, sería genial agregarlos. Avísame si te resultaría útil a través de los problemas en GitHub.

Estado y problemas conocidos

Personalmente, solo lo he probado en Windows XP (CMD, Console2), Ubuntu (gnome-terminal, xterm) y OS X.

Algunas secuencias ANSI válidas no se reconocen.

Si estás trabajando en el código, consulta README-hacking.md. ESPECIALMENTE, consulta la explicación allí de por qué no queremos PR que permitan a Colorama generar nuevos tipos de códigos ANSI.

Consulte los problemas pendientes y la lista de deseos:
https://github.com/tartley/colorama/issues

Si algo no funciona para usted, o no hace lo que esperaba o deseaba, me encantaría saberlo en esa lista de problemas, estaría encantado de recibir parches y estaría feliz de otorgar acceso de confirmación a cualquiera que envíe uno o dos parches que funcionen.

Licencia

Soporte profesional

Tidelift

El soporte profesional para colorama está disponible como parte de la suscripción a Tidelift. Tidelift ofrece a los equipos de desarrollo de software una única fuente para comprar y mantener su software, con garantías de nivel profesional de los expertos que mejor lo conocen, al mismo tiempo que se integra perfectamente con las herramientas existentes.

Gracias

¡Consulte el REGISTRO DE CAMBIOS para obtener más agradecimientos!

Marc Schlaich (schlamar) por una corrección de setup.py para Python2.5.

Marc Abramowitz informó y arregló un fallo al salir con la salida estándar cerrada, lo que proporcionó una solución al debate sobre setuptools/distutils del problema n.° 7 y otras correcciones.

Usuario ‘eryksun’, por su orientación sobre la instanciación correcta de ctypes.windll.

Matthew McCormick por señalar cortésmente un fallo de larga data en sistemas que no son Windows.

Ben Hoyt, por una magnífica corrección en sistemas Windows de 64 bits.

Jesse de Empty Square por enviar una corrección para los ejemplos en el README.

Usuario ‘jamessp’, una corrección de documentación observante para el posicionamiento del cursor.

Usuario ‘vaal1239’, Dave Mckee y Lackner Kristof por una pequeña pero muy necesaria corrección para Win7.

Julien Stuyck, por sugerir sabiamente actualizaciones compatibles con Python3 para el README.

Daniel Griffith por múltiples parches fabulosos.

Oscar Lesta por una valiosa corrección para evitar que los caracteres ANSI se envíen a salidas que no son tty.

Roger Binns, por sus numerosas sugerencias, valiosos comentarios e informes de errores.

Tim Golden por su reflexión y sus comentarios muy apreciados sobre la idea inicial.

Usuario ‘Zearin’ por las actualizaciones del archivo README.

John Szakmeister por agregar compatibilidad con colores claros

Charles Merriam por agregar documentación a las demostraciones

Jurko por una solución para CPython2.5 de Windows de 64 bits sin ctypes

Florian Bruhin por una solución cuando stdout o stderr son None

Thomas Weininger por solucionar ValueError en Windows

Remi Rampin por una mejor integración de Github y soluciones al archivo README

Simeon Visser por cerrar un identificador de archivo usando ‘with’ y actualizar los clasificadores para incluir Python 3.3 y 3.4

Andy Neff por solucionar el problema de RESET de los colores LIGHT_EX.

Jonathan Hartley por la idea inicial y la implementación.