

★ Python 3 para impacientes ★

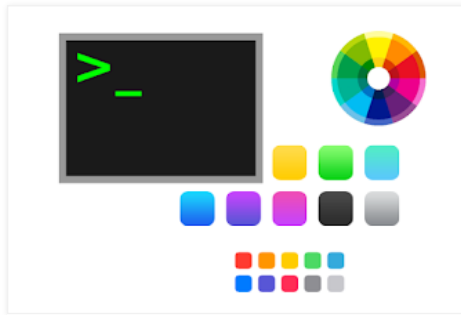


"Simple es mejor que complejo" (Tim Peters)

Python	IPython	EasyGUI	Tkinter	JupyterLab	Numpy
--------	---------	---------	---------	------------	-------

domingo, 18 de septiembre de 2016

Dar color a las salidas en la consola



En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades.

Hay un método basado en los [códigos ANSI](#) que no requiere instalar absolutamente nada pero con limitaciones con algunas versiones del sistema operativo **Windows**.

También, existen varios módulos de terceros que, aunque conlleva realizar una instalación, permiten aplicar con facilidad estilos y colores a las salidas en los sistemas operativos más extendidos: **Windows**, **Mac** y **GNU/Linux**.

En particular, en esta entrada para la tarea que tenemos por delante vamos a mostrar cómo se utilizan los [códigos ANSI](#) y el módulo [Colorama](#), éste último, por la gran aceptación que ha tenido entre los desarrolladores.

Aplicar formatos con códigos ANSI

En los sistemas con **Linux** y **Mac** se pueden utilizar los códigos de secuencias o códigos de escape del estándar ANSI para cambiar los formatos de las salidas. En el caso de Windows, en la actualidad, los códigos ANSI son soportados a partir de **Windows 10 update TH2** (con versiones anteriores de DOS y Windows, dependiendo de la versión, era necesario cargar el controlador ANSI.SYS, ANSI.COM u otro software de terceros en el inicio del sistema).

Un **código de formato** lo forma el carácter Escape seguido por tres números enteros separados por un punto y coma (;): el primero de estos números (un valor de 0 a 7) establece el estilo del texto (negrita, subrayado, etc); el segundo número (de 30 a 37) fija el color del texto y el último número (de 40 a 47) el color del fondo:

Estilos*	Cod. ANSI
Sin efecto	0
Negrita	1
Débil	2
Cursiva	3
Subrayado	4
Inverso	5
Oculto	6
Tachado	7

Buscar

Python para impacientes

[Python](#)
[IPython](#)
[EasyGUI](#)
[Tkinter](#)
[JupyterLab](#)
[Numpy](#)

Anexos

[Guía urgente de MySQL](#)
[Guía rápida de SQLite3](#)

Entradas + populares

[Dar color a las salidas en la consola](#)

En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

[Instalación de Python, paso a paso](#)

Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

[Variables de control en Tkinter](#)

Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

[Añadir, consultar, modificar y suprimir elementos en Numpy](#)

Acceder a los elementos de un array. [], [].
 ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

[Cálculo con arrays Numpy](#)

Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebido...

[Operaciones con fechas y horas. Calendarios](#)

Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

[Threading: programación con hilos \(I\)](#)

En programación, la técnica que permite que una aplicación ejecute simultáneamente varias operaciones en el mismo espacio de proceso se...

[Convertir, copiar, ordenar, unir y dividir arrays Numpy](#)

Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

[Tkinter: interfaces gráficas en Python](#)

Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

[Gráficos en IPython](#)

Unos de los motivos que inspiraron el desarrollo de IPython fue contar con una

(*) Hay que tener presente que algunos estilos no están soportados por todas las consolas.

Colores	Texto	Fondo
Negro	30	40
Rojo	31	41
Verde	32	42
Amarillo	33	43
Azul	34	44
Morado	35	45
Cian	36	46
Blanco	37	47

El carácter Escape se puede expresar en octal "\033", en hexadecimal "\x1b", o bien, con `chr(27)`. Después, le seguirá el carácter "[", el código de formato y el carácter "m", teniendo en cuenta que si se omite alguno de los tres valores se asumirá el valor que tenga en el formato anterior o el valor predeterminado de la consola.

`\033[cod_formato;cod_color_texto;cod_color_fondom`

En el siguiente ejemplo la secuencia de Escape se expresa de las formas comentadas. Se utilizan los estilos negrita y subrayado; y los colores amarillo y morado para el texto. Sin embargo, no se establece ningún color de fondo. En este caso se asume el color de fondo predeterminado de la consola.

```
print(chr(27)+"[1;33m"+"Texto en negrita de color amarillo")
print("\x1b[1;33m"+"Texto en negrita de color amarillo")
print("\033[4;35m"+"Texto en negrita y subrayado de color morado")
print("\033[4;35m"+"Texto en negrita y subrayado de color morado")
```

```
Texto en negrita de color amarillo
Texto en negrita de color amarillo
Texto en negrita y subrayado de color morado
Texto en negrita y subrayado de color morado
```

El ejemplo que sigue muestra que es posible cambiar de formato añadiendo uno nuevo al final de cada línea:

```
print("\033[2J\033[1;1f") # Borrar pantalla y situar cursor
print("\033[1;33m"+"Texto en negrita color amarillo"+'\033[0;m')
print("\033[;36m"+"Texto normal de color cian")
print("\033[4;35;47m"+"Texto subr morado sobre blanco"+'\033[0;m')
print("\033[4;35m"+"Texto normal subr color morado"+'\033[0;m')
```

```
Texto en negrita de color amarillo
Texto normal de color cian
Texto normal subrayado de color morado sobre fondo blanco
Texto normal subrayado de color morado
```

A continuación, un ejemplo que construye una tabla con todos los formatos posibles, recorriendo y cambiando estilos y colores:

```
def construye_tabla_formatos():
    for estilo in range(8):
        for colortexto in range(30,38):
            cad_cod = ''
            for colorfondo in range(40,48):
                fmto = ';'.join([str(estilo),
                                str(colortexto),
                                str(colorfondo)])
                cad_cod+="\033["+fmto+"m "+fmto+" \033[0m"
            print(cad_cod)
            print('\n')

construye_tabla_formatos()
```

herramienta que uniera la posibilidad de realizar cálculos...

Archivo

septiembre 2016 (2) ▼

python.org



pypi.org



Sitios

- [ActivePython](#)
- [Anaconda](#)
- [Bpython](#)
- [Django](#)
- [Flask](#)
- [Ipython](#)
- [IronPython](#)
- [Matplotlib](#)
- [MicroPython](#)
- [Numpy](#)
- [Pandas](#)
- [Pillow](#)
- [PortablePython](#)
- [PyBrain](#)
- [PyCharm](#)
- [PyDev](#)
- [PyGame](#)
- [Pypi](#)
- [PyPy](#)
- [Pyramid](#)
- [Python.org](#)
- [PyTorch](#)
- [SciPy.org](#)
- [Spyder](#)
- [Tensorflow](#)
- [TurboGears](#)

El módulo Colorama

Aplicar formatos con Colorama

Colorama es un módulo desarrollado por Arnon Yaari que facilita la aplicación de formatos a las salidas por la consola. Por un lado evita tener que expresar con valores numéricos los estilos y colores. En su lugar, se utilizan sus nombres en inglés que favorecen la lectura del código. Además, lo más importante, es que el módulo se puede usar con **GNU/Linux**, **Mac** y **Windows**; y como mejora ofrece la posibilidad de desplazar el cursor a una posición de pantalla antes de aplicar un formato.

Para instalar el módulo **Colorama**:

\$ pip3 install colorama

Estilos*	(Style)
Débil	DIM
Normal	NORMAL
Brillante	BRIGHT
Reset	RESET_ALL

(*) En **Colorama** los estilos se han reducido a los que son soportados por mayor número de sistemas. En **Windows** con **Style.NORMAL** se obtiene el mismo resultado que con **Style.DIM**

Colores Texto/Fondo	(Fore/Back)
Negro	BLACK
Rojo	RED
Verde	GREEN
Amarillo	YELLOW
Azul	BLUE
Morado	MAGENTA
Cian	CYAN
Blanco	WHITE
Reset	RESET

El ejemplo siguiente muestra la sencillez de uso de este módulo. Después de la línea de **"import"** se inicializa **Colorama** con el método **init()** y, a continuación, se aplican formatos a distintas salidas de texto. También, se utilizan los métodos **RESET** y **RESET_ALL** para restablecer estilos y colores. En las líneas finales se muestran los distintos niveles de intensidad posibles:

```
from colorama import init, Fore, Back, Style

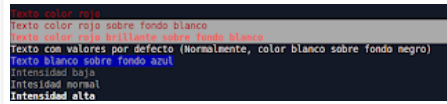
# Para restablecer colores después de cada impresión inicializar
# el módulo con init(autoreset=True) en lugar de init().

init()
print(Fore.RED+"Texto color rojo")
print(Back.WHITE+"Texto color rojo sobre fondo blanco")
print(Back.WHITE+Style.BRIGHT+"Txt rojo brill.s/blanco"+Back.RESET)
print(Style.RESET_ALL + "Texto con valores por defecto")
print(Fore.WHITE+Back.BLUE+"Texto blanco sobre azul"+Back.RESET)
print("Texto blanco sobre fondo negro")

# Niveles de intensidad

print(Style.DIM + Fore.WHITE + "Intensidad baja")
```

```
print(Style.NORMAL + "Intensidad normal")
print(Style.BRIGHT + "Intensidad alta")
```



Desplazar el cursor antes de dar formato

Colorama incluye la clase **Cursor** que permite, antes de dar formato, desplazar el cursor a una posición absoluta de la pantalla o una posición relativa en relación a la posición actual:

- **Cursor.POS(fila, columna)**: desplaza el cursor a la fila y columna indicadas.
- **Cursor.UP(número)**: desplaza el cursor a la línea anterior.
- **Cursor.DOWN(número)**: desplaza el cursor a la línea siguiente.
- **Cursor.FORWARD(número)**: avanza el cursor un número de caracteres en la línea actual.
- **Cursor.BACK(número)**: retrocede el cursor un número de caracteres en la línea actual.

En el siguiente ejemplo se simula la copia de una serie de archivos. Para mostrar las salidas en una misma posición de la pantalla se utilizan algunos de los métodos anteriores.

```
from time import sleep
from colorama import Cursor, init, Fore
init()
print("Copiando archivos... ")
for arch in ["111", "222", "333", "444", "555"]:
    sleep(1)
    print(Cursor.UP(1)+Cursor.FORWARD(20)+Fore.YELLOW+str(arch))

print(Cursor.POS(25,2) + Fore.GREEN + ">>> Proceso finalizado")

# Correspondencias con secuencias de Escape ANSI:

# "\033[númA" - Línea arriba
# "\033[númB" - Línea abajo
# "\033[númC" - avanzar caracter
# "\033[númD" - retroceder caracter
# "\033[x;yf" - desplazar cursor a coordenada de pantalla
```



[Ir al índice del tutorial de Python](#)

Publicado por Pherkad en [10:12](#)



Etiquetas: [Python3](#)

[Entrada más reciente](#)

[Inicio](#)

[Entrada antigua](#)