colorama

Why?

Colors important for perception and they can be used to improve user's interaction with you program

A couple of examples

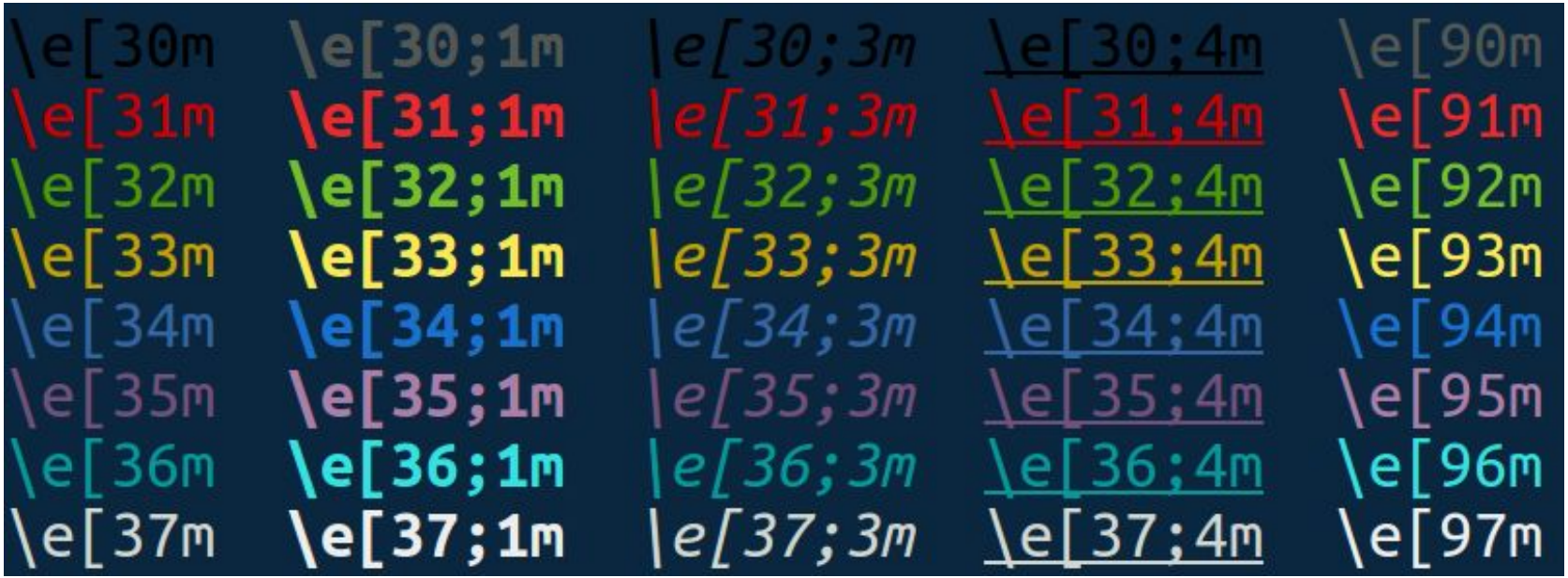
- mark errors
- highlight different parts of output with different colour

Primitive way

Most terminals are capable of color printing, and we print text via them.
So we can take advantage of their color coding directly

Minimal example

```
for code in {30..37}; do
> echo -en "\e[${code}m"'\\e['"$code"'m'"\e[0m";
> echo -en "  \e[$code;1m"'\\e['"$code"';1m'"\e[0m";
> echo -en "  \e[$code;3m"'\\e['"$code"';3m'"\e[0m";
> echo -en "  \e[$code;4m"'\\e['"$code"';4m'"\e[0m";
> echo -e "  \e[$((code+60))m"'\\e['"$((code+60))"'m'"\e[0m";
> done
```



Also, take a look at $LS_COLORS in your terminal

Wtf is going on?

We have such pattern

escape character-color code-finishing symbol

Escape character

There are variations of escape characters and color encoding on different terminals

escape characters

- `\e`
- `\033`
- `\0x1b`

Moreover they don't work on windows (not a big loss, but nevertheless)

Python

Ok, so what about python? Well, almost no difference

```
>>> print('\x1b[31;1mHi!')
Hi!
>>> print('Oh, Johny, what\'s going on?!')
Oh, Johny, what's going on?!
>>> print('Johny, I\'m bleeding!')
Johny, I'm bleeding!
>>> print('\x1b[0mOh that\'s better')
Oh that's better
```

Why this approach not ideal

Imho, there are 2 main drawbacks

- messiness - you can encapsulate all coloring stuff in your functions, but they still will be messy
- lack of standardization - these color encodings are different between different terminals

Next level!

But we have colorama module, so everything is ok) It fixes aforementioned flaws

- mnemonics for colors
- works on windows

Not so many colors, though(
And no "hotkeys" for things like underlining. For all this stuff look at curses or blessings

Beginning

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


# Default init don't work in Pycharm later about it
colorama.init()

print(Fore.BLUE, 'Paint me blue', sep='')
print('Oops')
```

```
Hi, there!)
Oops
```

Explanation

Default `colorama.init()` does nothing on UNIX, it is necessary to convert color codes to proper Windows stuff

`Fore` - stands for the font color. In order to return to normal font It needs to be closed with `Style.RESET_ALL`

Yet `colorama.init()` has various args

About init

`init(autoreset=False, convert=None, strip=None, wrap=True)`

- `autoreset` - whether to reset font setting automatically after print call, though it somehow disrupts printing from terminal(
- `convert`, `strip`, `wrap` - parameters for tuning color printing

Set `convert` and `strip` to `False` if you need to test colors in Pycharm

```
colorama.init(autoreset=True, convert=False,
              strip=False)

print(Fore.BLUE, 'Hi, there!)', sep='')
print('It\'s ok now')
```

```
Hi, there!)
It's ok now
```

Colors

Fore: BLACK, RED, GREEN, YELLOW, BLUE, MAGENTA, CYAN, WHITE, RESET

```
colors = ['BLACK', 'RED', 'GREEN', 'YELLOW', 'BLUE',
'MAGENTA', 'CYAN', 'WHITE']


for color in colors:
    print(getattr(Fore, color), 'Yeahh', sep='')
```

Backgrounds

Back: BLACK, RED, GREEN, YELLOW, BLUE, MAGENTA, CYAN, WHITE, RESET

```
backs = ['BLACK', 'RED', 'GREEN', 'YELLOW', 'BLUE',
'MAGENTA', 'CYAN', 'WHITE']

for back in backs:
    print(getattr(Back, back), 'Yeahh', sep='')
```

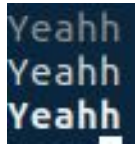
Styles

Style: DIM, NORMAL, BRIGHT, RESET_ALL

```
styles = ['DIM', 'NORMAL', 'BRIGHT']

for style in styles:
    print(getattr(Style, style), 'Yeahh', sep='')
```

Yeahh
Yeahh
Yeahh



Combined

```
print(Fore.GREEN, Style.BRIGHT, 'Bright green!)',
      sep='')
print(Fore.LIGHTYELLOW_EX, Style.BRIGHT, Back.BLUE,
      'Bright yellow!)', sep='')
```

```
Bright green!)
Bright yellow!)
```