Blessed Documentation

Release 1.17.11

Erik Rose, Jeff Quast, Avram Lubkin

Nov 10, 2020
# CONTENTS

1 Introduction 3  
1.1 Examples 3  
1.2 Requirements 5  
1.3 Brief Overview 5  
1.4 Before And After 6  

2 Terminal 7  
2.1 Capabilities 8  
2.2 Compound Formatting 8  
2.3 Clearing The Screen 8  
2.4 Hyperlinks 8  
2.5 Styles 9  
2.6 Full-Screen Mode 9  
2.7 Pipe Savvy 9  

3 Colors 11  
3.1 24-bit Colors 11  
3.2 256 Colors 25  
3.3 16 Colors 25  
3.4 Monochrome 27  

4 Keyboard 29  
4.1 inkey() 29  
4.2 Keycodes 30  
4.3 delete 33  
4.4 Alt/meta 33  

5 Location 35  
5.1 Example 36  
5.2 Context Manager 36  
5.3 Finding The Cursor 37  

6 Measuring 39  
6.1 Resizing 39  

7 Examples 41  
7.1 bounce.py 41  
7.2 cnn.py 41  
7.3 detect-multibyte.py 41  
7.4 editor.py 42  
7.5 keymatrix.py 42
Blessed is an easy, practical library for making terminal apps, by providing an elegant, well-documented interface to Colors, Keyboard input, and screen position and Location capabilities.

```python
from blessed import Terminal

term = Terminal()

print(term.home + term.clear + term.move_y(term.height // 2))
print(term.black_on_darkkhaki(term.center('press any key to continue.')))

with term.cbreak(), term.hidden_cursor():
    inp = term.inkey()

print(term.move_down(2) + 'You pressed ' + term.bold(repr(inp)))
```

It’s meant to be fun and easy, to do basic terminal graphics and styling with Python using blessed. Terminal is the only class you need to import and the only object you should need for Terminal capabilities.

Whether you want to improve CLI apps with colors, or make fullscreen applications or games, blessed should help get you started quickly. Your users will love it because it works on Windows, Mac, and Linux, and you will love it because it has plenty of documentation and examples!

Full documentation at https://blessed.readthedocs.io/en/latest/

### 1.1 Examples

Fig. 1: x11-colorpicker.py, bounce.py, worms.py, and plasma.py, from our repository.

Exemplary 3rd-party examples which use blessed,
Fig. 2: **Voltron** is an extensible debugger UI toolkit written in Python.

Fig. 3: **cursewords** is “graphical” command line program for solving crossword puzzles in the terminal.

Fig. 4: **GitHeat** builds an interactive heatmap of git history.

Fig. 5: **Dashing** is a library to quickly create terminal-based dashboards.

Fig. 6: **Enlighten** is a console progress bar library that allows simultaneous output without redirection.

Fig. 7: **macht** is a clone of the (briefly popular) puzzle game, 2048.
1.2 Requirements

*Blessed* works with Windows, Mac, Linux, and BSD’s, on Python 2.7, 3.4, 3.5, 3.6, 3.7, and 3.8.

1.3 Brief Overview

*Blessed* is more than just a Python wrapper around *curses*:

- **Styles, Colors**, and maybe a little positioning without necessarily clearing the whole screen first.
- Works great with Python’s new `f-strings` or any other kind of string formatting.
- Provides up-to-the-moment `Location` and terminal height and width, so you can respond to terminal size changes.
- Avoids making a mess if the output gets piped to a non-terminal, you can output sequences to any file-like object such as `StringIO`, files, pipes or sockets.
- Uses `terminfo(5)` so it works with any terminal type and capability: No more C-like calls to `tigetstr` and `tparm`.
- Non-obtrusive calls to only the capabilities database ensures that you are free to mix and match with calls to any other curses application code or library you like.
- Provides context managers `Terminal.fullscreen()` and `Terminal.hidden_cursor()` to safely express terminal modes, curses development will no longer fudge up your shell.
- Act intelligently when somebody redirects your output to a file, omitting all of the special sequences colors, but still containing all of the text.

*Blessed* is a fork of *blessings*, which does all of the same above with the same API, as well as following enhancements:

- Windows support, new since Dec. 2019!
- Dead-simple keyboard handling: safely decoding unicode input in your system’s preferred locale and supports application/arrow keys.
- 24-bit color support, using `Terminal.color_rgb()` and `Terminal.on_color_rgb()` and all X11 Colors by name, and not by number.
- Determine cursor location using `Terminal.get_location()`, enter key-at-a-time input mode using `Terminal.cbreak()` or `Terminal.raw()` context managers, and read timed key presses using `Terminal.inkey()`.
- Allows the printable length of strings that contain sequences to be determined by `Terminal.length()`, supporting additional methods `Terminal.wrap()` and `Terminal.center()`, terminal-aware variants of the built-in function `textwrap.wrap()` and method `str.center()`, respectively.
- Allows sequences to be removed from strings that contain them, using `Terminal.strip_seqs()` or sequences and whitespace using `Terminal.strip()`.

---

1.2. Requirements
### 1.4 Before And After

With the built-in `curses` module, this is how you would typically print some underlined text at the bottom of the screen:

```python
from curses import tigetstr, setupterm, tparm
from fcntl import ioctl
from os import isatty
import struct
import sys
from termios import TIOCGWINSZ

# If we want to tolerate having our output piped to other commands or
# files without crashing, we need to do all this branching:
if hasattr(sys.stdout, 'fileno') and isatty(sys.stdout.fileno()):
    setupterm()
    sc = tigetstr('sc')
    cup = tigetstr('cup')
    rc = tigetstr('rc')
    underline = tigetstr('smul')
    normal = tigetstr('sgr0')
else:
    sc = cup = rc = underline = normal = ''

# Save cursor position.
print(sc)

if cup:
    # tigetnum('lines') doesn't always update promptly, hence this:
    height = struct.unpack('hhhh', ioctl(0, TIOCGWINSZ, '\000' * 8))[0]

    # Move cursor to bottom.
    print(tparm(cup, height - 1, 0))

print('This is {under}underlined{normal}!

# Restore cursor position.
print(rc)
```

The same program with `Blessed` is simply:

```python
from blessed import Terminal

term = Terminal()

with term.location(0, term.height - 1):
    print('This is ' + term.underline('underlined') + '!

print(term.normal)
```
Blessed provides just one top-level object: `Terminal`. Instantiating a `Terminal` figures out whether you’re on a terminal at all and, if so, does any necessary setup:

```python
>>> import blessed
>>> term = blessed.Terminal()
```

This is the only object, named `term` here, that you should need from blessed, for all of the remaining examples in our documentation.

You can proceed to ask it all sorts of things about the terminal, such as its size:

```python
>>> term.height, term.width
(34, 102)
```

Support for `Colors`:

```python
>>> term.number_of_colors
56
```

And create printable strings containing sequences for `Colors`:

```python
>>> term.green_reverse('ALL SYSTEMS GO')
'\x1b[32m\x1b[7mALL SYSTEMS GO\x1b[m'
```

When printed, these codes make your terminal go to work:

```python
>>> print(term.white_on_firebrick3('SYSTEM OFFLINE'))
```

And thanks to f-strings since python 3.6, it’s very easy to mix attributes and strings together:

```python
>>> print(f"{term.yellow}Yellow is brown, {term.bright_yellow}" + f"Bright yellow is actually yellow!{term.normal}")
```
2.1 Capabilities

Any capability in the terminfo(5) manual, under column **Cap-name** can be an attribute of the **Terminal** class, such as ‘smul’ for ‘begin underline mode’.

There are a lot of interesting capabilities in the terminfo(5) manual page, but many of these will return an empty string, as they are not supported by your terminal. They can still be used, but have no effect. For example, **blink** only works on a few terminals, does yours?

```python
>>> print(term.blink("Insert System disk into drive A:"))
```

2.2 Compound Formatting

If you want to do lots of crazy formatting all at once, you can just mash it all together:

```python
>>> print(term.underline_bold_green_on_yellow('They live! In sewers!'))
```

This compound notation comes in handy for users & configuration to customize your app, too!

2.3 Clearing The Screen

Blessed provides syntactic sugar over some screen-clearing capabilities:

- **clear** Clear the whole screen.
- **clear_eol** Clear to the end of the line.
- **clear_bol** Clear backward to the beginning of the line.
- **clear_eos** Clear to the end of screen.

Suggest to always combine **home** and **clear**, and, in almost all emulators, clearing the screen after setting the background color will repaint the background of the screen:

```python
>>> print(term.home + term.on_blue + term.clear)
```

2.4 Hyperlinks

Maybe you haven’t noticed, because it’s a recent addition to terminal emulators, is that they can now support hyperlinks, like to HTML, or even file:// URLs, which allows creating clickable links of text.

```python
>>> print(f"blessed {term.link('https://blessed.readthedocs.org', 'documentation')}"
```

Hover your cursor over **documentation**, and it should highlight as a clickable URL.
2.5 Styles

In addition to Colors, blessed also supports the limited amount of styles that terminals can do. These are:

- **bold** Turn on ‘extra bright’ mode.
- **reverse** Switch fore and background attributes.
- **normal** Reset attributes to default.
- **underline** Enable underline mode.
- **no_underline** Disable underline mode.

**Note:** While the inverse of underline is no_underline, the only way to turn off bold or reverse is normal, which also cancels any custom colors.

2.6 Full-Screen Mode

If you’ve ever noticed how a program like vim(1) restores you to your unix shell history after exiting, it’s actually a pretty basic trick that all terminal emulators support, that blessed provides using the fullscreen() context manager over these two basic capabilities:

- **enter_fullscreen** Switch to alternate screen, previous screen is stored by terminal driver.
- **exit_fullscreen** Switch back to standard screen, restoring the same terminal screen.

```python
with term.fullscreen(), term.cbreak():
    print(term.move_y(term.height // 2) +
          term.center('press any key').rstrip())
    term.inkey()
```

2.7 Pipe Savvy

If your program isn’t attached to a terminal, such as piped to a program like less(1) or redirected to a file, all the capability attributes on Terminal will return empty strings for any Colors, Location, or other sequences. You’ll get a nice-looking file without any formatting codes gumming up the works.

If you want to override this, such as when piping output to less -R, pass argument value True to the force_styling parameter.

In any case, there is a does_styling attribute that lets you see whether the terminal attached to the output stream is capable of formatting. If it is False, you may refrain from drawing progress bars and other frippery and just stick to content:

```python
if term.does_styling:
    with term.location(x=0, y=term.height - 1):
        print('Progress: [=======> ]')
        print(term.bold("60\%"))
```
Doing colors with blessed is easy, pick a color name from the Color chart below, any of these named are also attributes of the Terminal!

These attributes can be printed directly, causing the terminal to switch into the given color. Or, as a callable, which terminates the string with the normal attribute. The following three statements are equivalent:

```python
>>> print(term.orangered + 'All systems are offline' + term.normal)
>>> print(f'{term.orangered}All systems are offline{term.normal}')
>>> print(term.orangered('All systems are offline'))
```

To use a background color, prefix any color with `on_`:

```python
>>> print(term.on_darkolivegreen('welcome to the army'))
```

And combine two colors using “on_” as in “foreground_on_background”:

```python
>>> print(term.peru_on_seagreen('All systems functioning within defined parameters.'))
```

### 3.1 24-bit Colors

Most Terminal emulators, even Windows, has supported 24-bit colors since roughly 2016. To test or force-set whether the terminal emulator supports 24-bit colors, check or set the terminal attribute `number_of_colors()`:

```python
>>> print(term.number_of_colors == 1 << 24)
True
```

Even if the terminal only supports 256, or worse, 16 colors, the nearest color supported by the terminal is automatically mapped:

```python
>>> term.number_of_colors = 1 << 24
>>> term.darkolivegreen
'\x1b[38;2;85;107;47m'
```

```python
>>> term.number_of_colors = 256
>>> term.darkolivegreen
'\x1b[38;5;58m'
```

```python
>>> term.number_of_colors = 16
>>> term.darkolivegreen
'\x1b[90m'
```
And finally, the direct \((r, g, b)\) values of 0-255 can be used for \texttt{color\_rgb()} and \texttt{on\_color\_rgb()} for foreground and background colors, to access each and every color!

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td></td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>red2</td>
<td></td>
<td>93.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>red3</td>
<td></td>
<td>80.4%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>snow</td>
<td></td>
<td>100.0%</td>
<td>98.0%</td>
<td>98.0%</td>
<td>0.0%</td>
<td>2.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>snow2</td>
<td></td>
<td>93.3%</td>
<td>91.4%</td>
<td>91.4%</td>
<td>0.0%</td>
<td>2.1%</td>
<td>93.3%</td>
</tr>
<tr>
<td>snow3</td>
<td></td>
<td>80.4%</td>
<td>78.8%</td>
<td>78.8%</td>
<td>0.0%</td>
<td>2.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>snow4</td>
<td></td>
<td>54.5%</td>
<td>53.7%</td>
<td>53.7%</td>
<td>0.0%</td>
<td>1.4%</td>
<td>54.5%</td>
</tr>
<tr>
<td>brown</td>
<td></td>
<td>64.7%</td>
<td>16.5%</td>
<td>16.5%</td>
<td>0.0%</td>
<td>74.5%</td>
<td>64.7%</td>
</tr>
<tr>
<td>brown1</td>
<td></td>
<td>100.0%</td>
<td>25.1%</td>
<td>25.1%</td>
<td>0.0%</td>
<td>74.9%</td>
<td>100.0%</td>
</tr>
<tr>
<td>brown2</td>
<td></td>
<td>93.3%</td>
<td>23.1%</td>
<td>23.1%</td>
<td>0.0%</td>
<td>75.2%</td>
<td>93.3%</td>
</tr>
<tr>
<td>brown3</td>
<td></td>
<td>80.4%</td>
<td>20.0%</td>
<td>20.0%</td>
<td>0.0%</td>
<td>75.1%</td>
<td>80.4%</td>
</tr>
<tr>
<td>brown4</td>
<td></td>
<td>54.5%</td>
<td>13.7%</td>
<td>13.7%</td>
<td>0.0%</td>
<td>74.8%</td>
<td>54.5%</td>
</tr>
<tr>
<td>darkred</td>
<td></td>
<td>54.5%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>indianred</td>
<td></td>
<td>80.4%</td>
<td>36.1%</td>
<td>36.1%</td>
<td>0.0%</td>
<td>55.1%</td>
<td>80.4%</td>
</tr>
<tr>
<td>indianred1</td>
<td></td>
<td>100.0%</td>
<td>41.6%</td>
<td>41.6%</td>
<td>0.0%</td>
<td>58.4%</td>
<td>100.0%</td>
</tr>
<tr>
<td>indianred2</td>
<td></td>
<td>93.3%</td>
<td>38.8%</td>
<td>38.8%</td>
<td>0.0%</td>
<td>58.4%</td>
<td>93.3%</td>
</tr>
<tr>
<td>indianred3</td>
<td></td>
<td>80.4%</td>
<td>33.3%</td>
<td>33.3%</td>
<td>0.0%</td>
<td>58.5%</td>
<td>80.4%</td>
</tr>
<tr>
<td>indianred4</td>
<td></td>
<td>54.5%</td>
<td>22.7%</td>
<td>22.7%</td>
<td>0.0%</td>
<td>58.3%</td>
<td>54.5%</td>
</tr>
<tr>
<td>firebrick</td>
<td></td>
<td>69.8%</td>
<td>13.3%</td>
<td>13.3%</td>
<td>0.0%</td>
<td>80.9%</td>
<td>69.8%</td>
</tr>
<tr>
<td>firebrick1</td>
<td></td>
<td>100.0%</td>
<td>18.8%</td>
<td>18.8%</td>
<td>0.0%</td>
<td>81.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>firebrick2</td>
<td></td>
<td>93.3%</td>
<td>17.3%</td>
<td>17.3%</td>
<td>0.0%</td>
<td>81.5%</td>
<td>93.3%</td>
</tr>
<tr>
<td>firebrick3</td>
<td></td>
<td>80.4%</td>
<td>14.9%</td>
<td>14.9%</td>
<td>0.0%</td>
<td>81.5%</td>
<td>80.4%</td>
</tr>
<tr>
<td>firebrick4</td>
<td></td>
<td>54.5%</td>
<td>10.2%</td>
<td>10.2%</td>
<td>0.0%</td>
<td>81.3%</td>
<td>54.5%</td>
</tr>
<tr>
<td>webmaroon</td>
<td></td>
<td>50.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>50.2%</td>
</tr>
<tr>
<td>rosybrown</td>
<td></td>
<td>73.7%</td>
<td>56.1%</td>
<td>56.1%</td>
<td>0.0%</td>
<td>23.9%</td>
<td>73.7%</td>
</tr>
<tr>
<td>rosybrown1</td>
<td></td>
<td>100.0%</td>
<td>75.7%</td>
<td>75.7%</td>
<td>0.0%</td>
<td>24.3%</td>
<td>100.0%</td>
</tr>
<tr>
<td>rosybrown2</td>
<td></td>
<td>93.3%</td>
<td>70.6%</td>
<td>70.6%</td>
<td>0.0%</td>
<td>24.4%</td>
<td>93.3%</td>
</tr>
<tr>
<td>rosybrown3</td>
<td></td>
<td>80.4%</td>
<td>60.8%</td>
<td>60.8%</td>
<td>0.0%</td>
<td>24.4%</td>
<td>80.4%</td>
</tr>
<tr>
<td>rosybrown4</td>
<td></td>
<td>54.5%</td>
<td>41.2%</td>
<td>41.2%</td>
<td>0.0%</td>
<td>24.5%</td>
<td>54.5%</td>
</tr>
<tr>
<td>lightcoral</td>
<td></td>
<td>94.1%</td>
<td>50.2%</td>
<td>50.2%</td>
<td>0.0%</td>
<td>46.7%</td>
<td>94.1%</td>
</tr>
<tr>
<td>salmon</td>
<td></td>
<td>98.0%</td>
<td>50.2%</td>
<td>44.7%</td>
<td>1.7%</td>
<td>54.4%</td>
<td>98.0%</td>
</tr>
<tr>
<td>mistyrose</td>
<td></td>
<td>100.0%</td>
<td>89.4%</td>
<td>88.2%</td>
<td>1.7%</td>
<td>11.8%</td>
<td>100.0%</td>
</tr>
<tr>
<td>mistyrose2</td>
<td></td>
<td>93.3%</td>
<td>83.5%</td>
<td>82.4%</td>
<td>1.8%</td>
<td>11.8%</td>
<td>93.3%</td>
</tr>
<tr>
<td>mistyrose3</td>
<td></td>
<td>80.4%</td>
<td>71.8%</td>
<td>71.0%</td>
<td>1.4%</td>
<td>11.7%</td>
<td>80.4%</td>
</tr>
<tr>
<td>coral1</td>
<td></td>
<td>100.0%</td>
<td>44.7%</td>
<td>33.7%</td>
<td>2.8%</td>
<td>66.3%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>coral2</td>
<td>93.3%</td>
<td>41.6%</td>
<td>31.4%</td>
<td>2.7%</td>
<td>66.4%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>coral3</td>
<td>80.4%</td>
<td>35.7%</td>
<td>27.1%</td>
<td>2.7%</td>
<td>66.3%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>coral4</td>
<td>54.5%</td>
<td>24.3%</td>
<td>18.4%</td>
<td>2.7%</td>
<td>66.2%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>tomato</td>
<td>100.0%</td>
<td>38.8%</td>
<td>27.8%</td>
<td>2.5%</td>
<td>72.2%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>tomato2</td>
<td>93.3%</td>
<td>36.1%</td>
<td>25.9%</td>
<td>2.5%</td>
<td>72.3%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>tomato3</td>
<td>80.4%</td>
<td>31.0%</td>
<td>22.4%</td>
<td>2.5%</td>
<td>72.2%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>tomato4</td>
<td>54.5%</td>
<td>21.2%</td>
<td>14.9%</td>
<td>2.6%</td>
<td>72.7%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>mistyrose4</td>
<td>54.5%</td>
<td>49.0%</td>
<td>48.2%</td>
<td>2.1%</td>
<td>11.5%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>salmon1</td>
<td>100.0%</td>
<td>54.9%</td>
<td>41.2%</td>
<td>3.9%</td>
<td>58.8%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>salmon2</td>
<td>93.3%</td>
<td>51.0%</td>
<td>38.4%</td>
<td>3.8%</td>
<td>58.8%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>salmon3</td>
<td>80.4%</td>
<td>43.9%</td>
<td>32.9%</td>
<td>3.9%</td>
<td>59.0%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>salmon4</td>
<td>54.5%</td>
<td>29.8%</td>
<td>22.4%</td>
<td>3.9%</td>
<td>59.0%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>coral</td>
<td>100.0%</td>
<td>49.8%</td>
<td>31.4%</td>
<td>4.5%</td>
<td>68.6%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>orangered</td>
<td>100.0%</td>
<td>27.1%</td>
<td>0.0%</td>
<td>4.5%</td>
<td>100.0%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>orangered2</td>
<td>93.3%</td>
<td>25.1%</td>
<td>0.0%</td>
<td>4.5%</td>
<td>100.0%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>orangered3</td>
<td>80.4%</td>
<td>21.6%</td>
<td>0.0%</td>
<td>4.5%</td>
<td>100.0%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>orangered4</td>
<td>54.5%</td>
<td>14.5%</td>
<td>0.0%</td>
<td>4.4%</td>
<td>100.0%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>darksalmon</td>
<td>91.4%</td>
<td>58.8%</td>
<td>47.8%</td>
<td>4.2%</td>
<td>47.6%</td>
<td>91.4%</td>
<td></td>
</tr>
<tr>
<td>lightsalmon</td>
<td>100.0%</td>
<td>62.7%</td>
<td>47.8%</td>
<td>4.8%</td>
<td>52.2%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>lightsalmon2</td>
<td>93.3%</td>
<td>58.4%</td>
<td>44.7%</td>
<td>4.7%</td>
<td>52.1%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>lightsalmon3</td>
<td>80.4%</td>
<td>50.6%</td>
<td>38.4%</td>
<td>4.8%</td>
<td>52.2%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>lightsalmon4</td>
<td>54.5%</td>
<td>34.1%</td>
<td>25.9%</td>
<td>4.8%</td>
<td>52.5%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>sienna</td>
<td>62.7%</td>
<td>32.2%</td>
<td>17.6%</td>
<td>5.4%</td>
<td>71.9%</td>
<td>62.7%</td>
<td></td>
</tr>
<tr>
<td>sienna1</td>
<td>100.0%</td>
<td>51.0%</td>
<td>27.8%</td>
<td>5.3%</td>
<td>72.2%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>sienna2</td>
<td>93.3%</td>
<td>47.5%</td>
<td>25.9%</td>
<td>5.3%</td>
<td>72.3%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>sienna3</td>
<td>80.4%</td>
<td>40.8%</td>
<td>22.4%</td>
<td>5.3%</td>
<td>72.2%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>sienna4</td>
<td>54.5%</td>
<td>27.8%</td>
<td>14.9%</td>
<td>5.4%</td>
<td>72.7%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>seashell</td>
<td>100.0%</td>
<td>96.1%</td>
<td>93.3%</td>
<td>6.9%</td>
<td>6.7%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>chocolate</td>
<td>82.4%</td>
<td>41.2%</td>
<td>11.8%</td>
<td>6.9%</td>
<td>85.7%</td>
<td>82.4%</td>
<td></td>
</tr>
<tr>
<td>chocolate1</td>
<td>100.0%</td>
<td>49.8%</td>
<td>14.1%</td>
<td>6.9%</td>
<td>85.9%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>chocolate2</td>
<td>93.3%</td>
<td>46.3%</td>
<td>12.9%</td>
<td>6.9%</td>
<td>86.1%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>chocolate3</td>
<td>80.4%</td>
<td>40.0%</td>
<td>11.4%</td>
<td>6.9%</td>
<td>85.9%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>chocolate4</td>
<td>54.5%</td>
<td>27.1%</td>
<td>7.5%</td>
<td>6.9%</td>
<td>86.3%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>seashell2</td>
<td>93.3%</td>
<td>89.8%</td>
<td>87.1%</td>
<td>7.3%</td>
<td>6.7%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>seashell3</td>
<td>80.4%</td>
<td>77.3%</td>
<td>74.9%</td>
<td>7.1%</td>
<td>6.8%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>seashell4</td>
<td>54.5%</td>
<td>52.5%</td>
<td>51.0%</td>
<td>7.4%</td>
<td>6.5%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>peachpuff</td>
<td>100.0%</td>
<td>85.5%</td>
<td>72.5%</td>
<td>7.9%</td>
<td>27.5%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>peachpuff2</td>
<td>93.3%</td>
<td>79.6%</td>
<td>67.8%</td>
<td>7.7%</td>
<td>27.3%</td>
<td>93.3%</td>
<td></td>
</tr>
</tbody>
</table>

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>peachpuff3</td>
<td></td>
<td>80.4%</td>
<td>68.6%</td>
<td>58.4%</td>
<td>7.7%</td>
<td>27.3%</td>
<td>80.4%</td>
</tr>
<tr>
<td>peachpuff4</td>
<td></td>
<td>54.5%</td>
<td>46.7%</td>
<td>39.6%</td>
<td>7.9%</td>
<td>27.3%</td>
<td>54.5%</td>
</tr>
<tr>
<td>sandybrown</td>
<td></td>
<td>95.7%</td>
<td>64.3%</td>
<td>37.6%</td>
<td>7.7%</td>
<td>60.7%</td>
<td>95.7%</td>
</tr>
<tr>
<td>tan1</td>
<td></td>
<td>100.0%</td>
<td>64.7%</td>
<td>31.0%</td>
<td>8.1%</td>
<td>69.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>tan2</td>
<td></td>
<td>93.3%</td>
<td>60.4%</td>
<td>28.6%</td>
<td>8.2%</td>
<td>69.3%</td>
<td>93.3%</td>
</tr>
<tr>
<td>tan4</td>
<td></td>
<td>54.5%</td>
<td>35.3%</td>
<td>16.9%</td>
<td>8.2%</td>
<td>69.1%</td>
<td>54.5%</td>
</tr>
<tr>
<td>peru</td>
<td></td>
<td>80.4%</td>
<td>52.2%</td>
<td>24.7%</td>
<td>8.2%</td>
<td>69.3%</td>
<td>80.4%</td>
</tr>
<tr>
<td>linen</td>
<td></td>
<td>98.0%</td>
<td>94.1%</td>
<td>90.2%</td>
<td>8.3%</td>
<td>8.0%</td>
<td>98.0%</td>
</tr>
<tr>
<td>bisque3</td>
<td></td>
<td>80.4%</td>
<td>71.8%</td>
<td>62.0%</td>
<td>8.9%</td>
<td>22.9%</td>
<td>80.4%</td>
</tr>
<tr>
<td>darkorange1</td>
<td></td>
<td>100.0%</td>
<td>49.8%</td>
<td>0.0%</td>
<td>8.3%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>darkorange2</td>
<td></td>
<td>93.3%</td>
<td>46.3%</td>
<td>0.0%</td>
<td>8.3%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>darkorange3</td>
<td></td>
<td>80.4%</td>
<td>40.0%</td>
<td>0.0%</td>
<td>8.3%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>darkorange4</td>
<td></td>
<td>54.5%</td>
<td>27.1%</td>
<td>0.0%</td>
<td>8.3%</td>
<td>100.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>tan</td>
<td></td>
<td>82.4%</td>
<td>70.6%</td>
<td>54.9%</td>
<td>9.5%</td>
<td>33.3%</td>
<td>82.4%</td>
</tr>
<tr>
<td>bisque</td>
<td></td>
<td>100.0%</td>
<td>89.4%</td>
<td>76.9%</td>
<td>9.0%</td>
<td>23.1%</td>
<td>100.0%</td>
</tr>
<tr>
<td>bisque2</td>
<td></td>
<td>93.3%</td>
<td>83.5%</td>
<td>71.8%</td>
<td>9.1%</td>
<td>23.1%</td>
<td>93.3%</td>
</tr>
<tr>
<td>bisque4</td>
<td></td>
<td>54.5%</td>
<td>49.0%</td>
<td>42.0%</td>
<td>9.4%</td>
<td>23.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>burlywood</td>
<td></td>
<td>87.1%</td>
<td>72.2%</td>
<td>52.9%</td>
<td>9.4%</td>
<td>39.2%</td>
<td>87.1%</td>
</tr>
<tr>
<td>burlywood1</td>
<td></td>
<td>100.0%</td>
<td>82.7%</td>
<td>60.8%</td>
<td>9.3%</td>
<td>39.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>burlywood2</td>
<td></td>
<td>93.3%</td>
<td>77.3%</td>
<td>56.9%</td>
<td>9.3%</td>
<td>39.1%</td>
<td>93.3%</td>
</tr>
<tr>
<td>burlywood3</td>
<td></td>
<td>80.4%</td>
<td>66.7%</td>
<td>49.0%</td>
<td>9.4%</td>
<td>39.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>burlywood4</td>
<td></td>
<td>54.5%</td>
<td>45.1%</td>
<td>33.3%</td>
<td>9.3%</td>
<td>38.8%</td>
<td>54.5%</td>
</tr>
<tr>
<td>darkorange</td>
<td></td>
<td>100.0%</td>
<td>54.9%</td>
<td>0.0%</td>
<td>9.2%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>navajowhite</td>
<td></td>
<td>100.0%</td>
<td>87.1%</td>
<td>67.8%</td>
<td>10.0%</td>
<td>32.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>navajowhite2</td>
<td></td>
<td>93.3%</td>
<td>81.2%</td>
<td>63.1%</td>
<td>10.0%</td>
<td>32.4%</td>
<td>93.3%</td>
</tr>
<tr>
<td>antiquewhite</td>
<td></td>
<td>98.0%</td>
<td>92.2%</td>
<td>84.3%</td>
<td>9.5%</td>
<td>14.0%</td>
<td>98.0%</td>
</tr>
<tr>
<td>antiquewhite1</td>
<td></td>
<td>100.0%</td>
<td>93.7%</td>
<td>85.9%</td>
<td>9.3%</td>
<td>14.1%</td>
<td>100.0%</td>
</tr>
<tr>
<td>antiquewhite2</td>
<td></td>
<td>93.3%</td>
<td>87.5%</td>
<td>80.0%</td>
<td>9.3%</td>
<td>14.3%</td>
<td>93.3%</td>
</tr>
<tr>
<td>antiquewhite3</td>
<td></td>
<td>80.4%</td>
<td>75.3%</td>
<td>69.0%</td>
<td>9.2%</td>
<td>14.1%</td>
<td>80.4%</td>
</tr>
<tr>
<td>antiquewhite4</td>
<td></td>
<td>54.5%</td>
<td>51.4%</td>
<td>47.1%</td>
<td>9.6%</td>
<td>13.7%</td>
<td>54.5%</td>
</tr>
<tr>
<td>wheat</td>
<td></td>
<td>96.1%</td>
<td>87.1%</td>
<td>70.2%</td>
<td>10.9%</td>
<td>26.9%</td>
<td>96.1%</td>
</tr>
<tr>
<td>wheat1</td>
<td></td>
<td>100.0%</td>
<td>90.6%</td>
<td>72.9%</td>
<td>10.9%</td>
<td>27.1%</td>
<td>100.0%</td>
</tr>
<tr>
<td>wheat2</td>
<td></td>
<td>93.3%</td>
<td>84.7%</td>
<td>68.2%</td>
<td>10.9%</td>
<td>26.9%</td>
<td>93.3%</td>
</tr>
<tr>
<td>wheat3</td>
<td></td>
<td>80.4%</td>
<td>72.9%</td>
<td>58.8%</td>
<td>10.9%</td>
<td>26.8%</td>
<td>80.4%</td>
</tr>
<tr>
<td>wheat4</td>
<td></td>
<td>54.5%</td>
<td>49.4%</td>
<td>40.0%</td>
<td>10.8%</td>
<td>26.6%</td>
<td>54.5%</td>
</tr>
<tr>
<td>orange</td>
<td></td>
<td>100.0%</td>
<td>64.7%</td>
<td>0.0%</td>
<td>10.8%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>orange2</td>
<td></td>
<td>93.3%</td>
<td>60.4%</td>
<td>0.0%</td>
<td>10.8%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>orange3</td>
<td></td>
<td>80.4%</td>
<td>52.2%</td>
<td>0.0%</td>
<td>10.8%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
</tbody>
</table>

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>orange4</td>
<td></td>
<td>54.5%</td>
<td>35.3%</td>
<td>0.0%</td>
<td>10.8%</td>
<td>100.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>oldlace</td>
<td></td>
<td>99.2%</td>
<td>96.1%</td>
<td>90.2%</td>
<td>10.9%</td>
<td>9.1%</td>
<td>99.2%</td>
</tr>
<tr>
<td>moccasin</td>
<td></td>
<td>100.0%</td>
<td>89.4%</td>
<td>71.0%</td>
<td>10.6%</td>
<td>29.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>papayawhip</td>
<td></td>
<td>100.0%</td>
<td>93.7%</td>
<td>83.5%</td>
<td>10.3%</td>
<td>16.5%</td>
<td>100.0%</td>
</tr>
<tr>
<td>navajowhite3</td>
<td></td>
<td>80.4%</td>
<td>70.2%</td>
<td>54.5%</td>
<td>10.1%</td>
<td>32.2%</td>
<td>80.4%</td>
</tr>
<tr>
<td>navajowhite4</td>
<td></td>
<td>54.5%</td>
<td>47.5%</td>
<td>36.9%</td>
<td>10.0%</td>
<td>32.4%</td>
<td>54.5%</td>
</tr>
<tr>
<td>blanchedalmond</td>
<td></td>
<td>100.0%</td>
<td>92.2%</td>
<td>80.4%</td>
<td>10.0%</td>
<td>19.6%</td>
<td>100.0%</td>
</tr>
<tr>
<td>goldenrod</td>
<td></td>
<td>85.5%</td>
<td>64.7%</td>
<td>12.5%</td>
<td>11.9%</td>
<td>85.3%</td>
<td>85.5%</td>
</tr>
<tr>
<td>goldenrod1</td>
<td></td>
<td>100.0%</td>
<td>75.7%</td>
<td>14.5%</td>
<td>11.9%</td>
<td>85.5%</td>
<td>100.0%</td>
</tr>
<tr>
<td>goldenrod2</td>
<td></td>
<td>93.3%</td>
<td>70.6%</td>
<td>13.3%</td>
<td>11.9%</td>
<td>85.7%</td>
<td>93.3%</td>
</tr>
<tr>
<td>goldenrod3</td>
<td></td>
<td>80.4%</td>
<td>60.8%</td>
<td>11.4%</td>
<td>11.9%</td>
<td>85.9%</td>
<td>80.4%</td>
</tr>
<tr>
<td>goldenrod4</td>
<td></td>
<td>54.5%</td>
<td>41.2%</td>
<td>7.8%</td>
<td>11.9%</td>
<td>85.6%</td>
<td>54.5%</td>
</tr>
<tr>
<td>floralwhite</td>
<td></td>
<td>100.0%</td>
<td>98.0%</td>
<td>94.1%</td>
<td>11.1%</td>
<td>5.9%</td>
<td>100.0%</td>
</tr>
<tr>
<td>darkgoldenrod</td>
<td></td>
<td>72.2%</td>
<td>52.5%</td>
<td>4.3%</td>
<td>11.8%</td>
<td>94.0%</td>
<td>72.2%</td>
</tr>
<tr>
<td>darkgoldenrod1</td>
<td></td>
<td>100.0%</td>
<td>72.5%</td>
<td>5.9%</td>
<td>11.8%</td>
<td>94.1%</td>
<td>100.0%</td>
</tr>
<tr>
<td>darkgoldenrod2</td>
<td></td>
<td>93.3%</td>
<td>67.8%</td>
<td>5.5%</td>
<td>11.8%</td>
<td>94.1%</td>
<td>93.3%</td>
</tr>
<tr>
<td>darkgoldenrod3</td>
<td></td>
<td>80.4%</td>
<td>58.4%</td>
<td>4.7%</td>
<td>11.8%</td>
<td>94.1%</td>
<td>80.4%</td>
</tr>
<tr>
<td>darkgoldenrod4</td>
<td></td>
<td>54.5%</td>
<td>39.6%</td>
<td>3.1%</td>
<td>11.8%</td>
<td>94.2%</td>
<td>54.5%</td>
</tr>
<tr>
<td>cornsilk</td>
<td></td>
<td>100.0%</td>
<td>97.3%</td>
<td>86.3%</td>
<td>13.3%</td>
<td>13.7%</td>
<td>100.0%</td>
</tr>
<tr>
<td>cornsilk2</td>
<td></td>
<td>93.3%</td>
<td>91.0%</td>
<td>80.4%</td>
<td>13.6%</td>
<td>13.9%</td>
<td>93.3%</td>
</tr>
<tr>
<td>cornsilk3</td>
<td></td>
<td>80.4%</td>
<td>78.4%</td>
<td>69.4%</td>
<td>13.7%</td>
<td>13.7%</td>
<td>80.4%</td>
</tr>
<tr>
<td>lightgoldenrod1</td>
<td></td>
<td>100.0%</td>
<td>92.5%</td>
<td>54.5%</td>
<td>13.9%</td>
<td>45.5%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lightgoldenrod2</td>
<td></td>
<td>93.3%</td>
<td>86.3%</td>
<td>51.0%</td>
<td>13.9%</td>
<td>45.4%</td>
<td>93.3%</td>
</tr>
<tr>
<td>lightgoldenrod3</td>
<td></td>
<td>80.4%</td>
<td>74.5%</td>
<td>43.9%</td>
<td>14.0%</td>
<td>45.4%</td>
<td>80.4%</td>
</tr>
<tr>
<td>gold</td>
<td></td>
<td>100.0%</td>
<td>84.3%</td>
<td>0.0%</td>
<td>14.1%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>gold2</td>
<td></td>
<td>93.3%</td>
<td>78.8%</td>
<td>0.0%</td>
<td>14.1%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>gold3</td>
<td></td>
<td>80.4%</td>
<td>67.8%</td>
<td>0.0%</td>
<td>14.1%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>gold4</td>
<td></td>
<td>54.5%</td>
<td>45.9%</td>
<td>0.0%</td>
<td>14.0%</td>
<td>100.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>cornsilk4</td>
<td></td>
<td>54.5%</td>
<td>53.3%</td>
<td>47.1%</td>
<td>14.0%</td>
<td>13.7%</td>
<td>54.5%</td>
</tr>
<tr>
<td>lemonchiffon2</td>
<td></td>
<td>93.3%</td>
<td>91.4%</td>
<td>74.9%</td>
<td>14.9%</td>
<td>19.7%</td>
<td>93.3%</td>
</tr>
<tr>
<td>lightgoldenrod</td>
<td></td>
<td>93.3%</td>
<td>86.7%</td>
<td>51.0%</td>
<td>14.0%</td>
<td>45.4%</td>
<td>93.3%</td>
</tr>
<tr>
<td>lightgoldenrod4</td>
<td></td>
<td>54.5%</td>
<td>50.6%</td>
<td>29.8%</td>
<td>14.0%</td>
<td>45.3%</td>
<td>54.5%</td>
</tr>
<tr>
<td>khaki</td>
<td></td>
<td>94.1%</td>
<td>90.2%</td>
<td>54.9%</td>
<td>15.0%</td>
<td>41.7%</td>
<td>94.1%</td>
</tr>
<tr>
<td>khaki1</td>
<td></td>
<td>100.0%</td>
<td>96.5%</td>
<td>56.1%</td>
<td>15.3%</td>
<td>43.9%</td>
<td>100.0%</td>
</tr>
<tr>
<td>khaki2</td>
<td></td>
<td>93.3%</td>
<td>90.2%</td>
<td>52.2%</td>
<td>15.4%</td>
<td>44.1%</td>
<td>93.3%</td>
</tr>
<tr>
<td>khaki3</td>
<td></td>
<td>80.4%</td>
<td>77.6%</td>
<td>45.1%</td>
<td>15.4%</td>
<td>43.9%</td>
<td>80.4%</td>
</tr>
<tr>
<td>khaki4</td>
<td></td>
<td>54.5%</td>
<td>52.5%</td>
<td>30.6%</td>
<td>15.3%</td>
<td>43.9%</td>
<td>54.5%</td>
</tr>
<tr>
<td>darkkhaki</td>
<td></td>
<td>74.1%</td>
<td>71.8%</td>
<td>42.0%</td>
<td>15.4%</td>
<td>43.4%</td>
<td>74.1%</td>
</tr>
</tbody>
</table>

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemonchiffon</td>
<td></td>
<td>100.0%</td>
<td>98.0%</td>
<td>80.4%</td>
<td>15.0%</td>
<td>19.6%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lemonchiffon3</td>
<td></td>
<td>80.4%</td>
<td>78.8%</td>
<td>64.7%</td>
<td>15.0%</td>
<td>19.5%</td>
<td>80.4%</td>
</tr>
<tr>
<td>lemonchiffon4</td>
<td></td>
<td>54.5%</td>
<td>53.7%</td>
<td>43.9%</td>
<td>15.4%</td>
<td>19.4%</td>
<td>54.5%</td>
</tr>
<tr>
<td>palegoldenrod</td>
<td></td>
<td>93.3%</td>
<td>91.0%</td>
<td>66.7%</td>
<td>15.2%</td>
<td>28.6%</td>
<td>93.3%</td>
</tr>
<tr>
<td>beige</td>
<td></td>
<td>96.1%</td>
<td>96.1%</td>
<td>86.3%</td>
<td>16.7%</td>
<td>10.2%</td>
<td>96.1%</td>
</tr>
<tr>
<td>olive</td>
<td></td>
<td>50.2%</td>
<td>50.2%</td>
<td>0.0%</td>
<td>16.7%</td>
<td>100.0%</td>
<td>50.2%</td>
</tr>
<tr>
<td>ivory</td>
<td></td>
<td>100.0%</td>
<td>100.0%</td>
<td>94.1%</td>
<td>16.7%</td>
<td>5.9%</td>
<td>100.0%</td>
</tr>
<tr>
<td>ivory2</td>
<td></td>
<td>93.3%</td>
<td>93.3%</td>
<td>87.8%</td>
<td>16.7%</td>
<td>5.9%</td>
<td>93.3%</td>
</tr>
<tr>
<td>ivory3</td>
<td></td>
<td>80.4%</td>
<td>80.4%</td>
<td>75.7%</td>
<td>16.7%</td>
<td>5.9%</td>
<td>80.4%</td>
</tr>
<tr>
<td>ivory4</td>
<td></td>
<td>54.5%</td>
<td>54.5%</td>
<td>51.4%</td>
<td>16.7%</td>
<td>5.8%</td>
<td>54.5%</td>
</tr>
<tr>
<td>yellow</td>
<td></td>
<td>100.0%</td>
<td>100.0%</td>
<td>0.0%</td>
<td>16.7%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>yellow2</td>
<td></td>
<td>93.3%</td>
<td>93.3%</td>
<td>0.0%</td>
<td>16.7%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>yellow3</td>
<td></td>
<td>80.4%</td>
<td>80.4%</td>
<td>0.0%</td>
<td>16.7%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>yellow4</td>
<td></td>
<td>54.5%</td>
<td>54.5%</td>
<td>0.0%</td>
<td>16.7%</td>
<td>100.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>lightyellow</td>
<td></td>
<td>100.0%</td>
<td>100.0%</td>
<td>87.8%</td>
<td>16.7%</td>
<td>12.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lightyellow2</td>
<td></td>
<td>93.3%</td>
<td>93.3%</td>
<td>82.0%</td>
<td>16.7%</td>
<td>12.2%</td>
<td>93.3%</td>
</tr>
<tr>
<td>lightyellow3</td>
<td></td>
<td>80.4%</td>
<td>80.4%</td>
<td>70.6%</td>
<td>16.7%</td>
<td>12.2%</td>
<td>80.4%</td>
</tr>
<tr>
<td>lightyellow4</td>
<td></td>
<td>54.5%</td>
<td>54.5%</td>
<td>47.8%</td>
<td>16.7%</td>
<td>12.2%</td>
<td>54.5%</td>
</tr>
<tr>
<td>lightgoldenrodyellow</td>
<td></td>
<td>98.0%</td>
<td>98.0%</td>
<td>82.4%</td>
<td>16.7%</td>
<td>16.0%</td>
<td>98.0%</td>
</tr>
<tr>
<td>olivedrab</td>
<td></td>
<td>42.0%</td>
<td>55.7%</td>
<td>13.7%</td>
<td>22.1%</td>
<td>75.4%</td>
<td>55.7%</td>
</tr>
<tr>
<td>olivedrab1</td>
<td></td>
<td>75.3%</td>
<td>100.0%</td>
<td>24.3%</td>
<td>22.1%</td>
<td>75.7%</td>
<td>100.0%</td>
</tr>
<tr>
<td>olivedrab2</td>
<td></td>
<td>70.2%</td>
<td>93.3%</td>
<td>22.7%</td>
<td>22.1%</td>
<td>75.6%</td>
<td>93.3%</td>
</tr>
<tr>
<td>olivedrab3</td>
<td></td>
<td>60.4%</td>
<td>80.4%</td>
<td>19.6%</td>
<td>22.2%</td>
<td>75.6%</td>
<td>80.4%</td>
</tr>
<tr>
<td>olivedrab4</td>
<td></td>
<td>41.2%</td>
<td>54.5%</td>
<td>13.3%</td>
<td>22.1%</td>
<td>75.5%</td>
<td>54.5%</td>
</tr>
<tr>
<td>darkolivegreen</td>
<td></td>
<td>33.3%</td>
<td>42.0%</td>
<td>18.4%</td>
<td>22.8%</td>
<td>56.1%</td>
<td>42.0%</td>
</tr>
<tr>
<td>darkolivegreen1</td>
<td></td>
<td>79.2%</td>
<td>100.0%</td>
<td>43.9%</td>
<td>22.8%</td>
<td>56.1%</td>
<td>100.0%</td>
</tr>
<tr>
<td>darkolivegreen2</td>
<td></td>
<td>73.7%</td>
<td>93.3%</td>
<td>40.8%</td>
<td>22.9%</td>
<td>56.3%</td>
<td>93.3%</td>
</tr>
<tr>
<td>darkolivegreen3</td>
<td></td>
<td>63.5%</td>
<td>80.4%</td>
<td>35.3%</td>
<td>22.9%</td>
<td>56.1%</td>
<td>80.4%</td>
</tr>
<tr>
<td>darkolivegreen4</td>
<td></td>
<td>43.1%</td>
<td>54.5%</td>
<td>23.9%</td>
<td>22.9%</td>
<td>56.1%</td>
<td>54.5%</td>
</tr>
<tr>
<td>greenyellow</td>
<td></td>
<td>67.8%</td>
<td>100.0%</td>
<td>18.4%</td>
<td>23.2%</td>
<td>81.6%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lawngreen</td>
<td></td>
<td>48.6%</td>
<td>98.8%</td>
<td>0.0%</td>
<td>25.1%</td>
<td>100.0%</td>
<td>98.8%</td>
</tr>
<tr>
<td>chartreuse</td>
<td></td>
<td>49.8%</td>
<td>100.0%</td>
<td>0.0%</td>
<td>25.0%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>chartreuse2</td>
<td></td>
<td>46.3%</td>
<td>93.3%</td>
<td>0.0%</td>
<td>25.1%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>chartreuse3</td>
<td></td>
<td>40.0%</td>
<td>80.4%</td>
<td>0.0%</td>
<td>25.0%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>chartreuse4</td>
<td></td>
<td>27.1%</td>
<td>54.5%</td>
<td>0.0%</td>
<td>25.1%</td>
<td>100.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>green</td>
<td></td>
<td>0.0%</td>
<td>100.0%</td>
<td>0.0%</td>
<td>33.3%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>green2</td>
<td></td>
<td>0.0%</td>
<td>93.3%</td>
<td>0.0%</td>
<td>33.3%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>green3</td>
<td></td>
<td>0.0%</td>
<td>80.4%</td>
<td>0.0%</td>
<td>33.3%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>green4</td>
<td>0.0%</td>
<td>54.5%</td>
<td>0.0%</td>
<td>33.3%</td>
<td>100.0%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>webgreen</td>
<td>0.0%</td>
<td>50.2%</td>
<td>0.0%</td>
<td>33.3%</td>
<td>100.0%</td>
<td>50.2%</td>
<td></td>
</tr>
<tr>
<td>honeydew</td>
<td>94.1%</td>
<td>100.0%</td>
<td>94.1%</td>
<td>33.3%</td>
<td>5.9%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>honeydew2</td>
<td>87.8%</td>
<td>93.3%</td>
<td>87.8%</td>
<td>33.3%</td>
<td>5.9%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>honeydew3</td>
<td>75.7%</td>
<td>80.4%</td>
<td>75.7%</td>
<td>33.3%</td>
<td>5.9%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>honeydew4</td>
<td>51.4%</td>
<td>54.5%</td>
<td>51.4%</td>
<td>33.3%</td>
<td>5.8%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>darkgreen</td>
<td>0.0%</td>
<td>39.2%</td>
<td>0.0%</td>
<td>33.3%</td>
<td>100.0%</td>
<td>39.2%</td>
<td></td>
</tr>
<tr>
<td>palegreen</td>
<td>59.6%</td>
<td>98.4%</td>
<td>59.6%</td>
<td>33.3%</td>
<td>39.4%</td>
<td>98.4%</td>
<td></td>
</tr>
<tr>
<td>palegreen1</td>
<td>60.4%</td>
<td>100.0%</td>
<td>60.4%</td>
<td>33.3%</td>
<td>39.6%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>palegreen3</td>
<td>48.6%</td>
<td>80.4%</td>
<td>48.6%</td>
<td>33.3%</td>
<td>39.5%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>palegreen4</td>
<td>32.9%</td>
<td>54.5%</td>
<td>32.9%</td>
<td>33.3%</td>
<td>39.6%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>limegreen</td>
<td>19.6%</td>
<td>80.4%</td>
<td>19.6%</td>
<td>33.3%</td>
<td>75.6%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>lightgreen</td>
<td>56.5%</td>
<td>93.3%</td>
<td>56.5%</td>
<td>33.3%</td>
<td>39.5%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>forestgreen</td>
<td>13.3%</td>
<td>54.5%</td>
<td>13.3%</td>
<td>33.3%</td>
<td>75.5%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>darkseagreen</td>
<td>56.1%</td>
<td>73.7%</td>
<td>56.1%</td>
<td>33.3%</td>
<td>23.9%</td>
<td>73.7%</td>
<td></td>
</tr>
<tr>
<td>darkseagreen1</td>
<td>75.7%</td>
<td>100.0%</td>
<td>75.7%</td>
<td>33.3%</td>
<td>24.3%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>darkseagreen2</td>
<td>70.6%</td>
<td>93.3%</td>
<td>70.6%</td>
<td>33.3%</td>
<td>24.4%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>darkseagreen3</td>
<td>60.8%</td>
<td>80.4%</td>
<td>60.8%</td>
<td>33.3%</td>
<td>24.4%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>darkseagreen4</td>
<td>41.2%</td>
<td>54.5%</td>
<td>41.2%</td>
<td>33.3%</td>
<td>24.5%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>seagreen</td>
<td>18.0%</td>
<td>54.5%</td>
<td>34.1%</td>
<td>40.7%</td>
<td>66.9%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>seagreen1</td>
<td>32.9%</td>
<td>100.0%</td>
<td>62.4%</td>
<td>40.6%</td>
<td>67.1%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>seagreen2</td>
<td>30.6%</td>
<td>93.3%</td>
<td>58.0%</td>
<td>40.6%</td>
<td>67.2%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>seagreen3</td>
<td>26.3%</td>
<td>80.4%</td>
<td>50.2%</td>
<td>40.7%</td>
<td>67.3%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>mediumseagreen</td>
<td>23.5%</td>
<td>70.2%</td>
<td>44.3%</td>
<td>40.8%</td>
<td>66.5%</td>
<td>70.2%</td>
<td></td>
</tr>
<tr>
<td>mintcream</td>
<td>96.1%</td>
<td>100.0%</td>
<td>98.0%</td>
<td>41.7%</td>
<td>3.9%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>springgreen</td>
<td>0.0%</td>
<td>100.0%</td>
<td>49.8%</td>
<td>41.6%</td>
<td>100.0%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>springgreen2</td>
<td>0.0%</td>
<td>93.3%</td>
<td>46.3%</td>
<td>41.6%</td>
<td>100.0%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>springgreen3</td>
<td>0.0%</td>
<td>80.4%</td>
<td>40.0%</td>
<td>41.6%</td>
<td>100.0%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>springgreen4</td>
<td>0.0%</td>
<td>54.5%</td>
<td>27.1%</td>
<td>41.6%</td>
<td>100.0%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>mediumspringgreen</td>
<td>0.0%</td>
<td>98.0%</td>
<td>60.4%</td>
<td>43.6%</td>
<td>100.0%</td>
<td>98.0%</td>
<td></td>
</tr>
<tr>
<td>aquamarine</td>
<td>49.8%</td>
<td>100.0%</td>
<td>83.1%</td>
<td>44.4%</td>
<td>50.2%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>aquamarine2</td>
<td>46.3%</td>
<td>93.3%</td>
<td>77.6%</td>
<td>44.4%</td>
<td>50.4%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>aquamarine3</td>
<td>40.0%</td>
<td>80.4%</td>
<td>66.7%</td>
<td>44.3%</td>
<td>50.2%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>aquamarine4</td>
<td>27.1%</td>
<td>54.5%</td>
<td>45.5%</td>
<td>44.5%</td>
<td>50.4%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>turquoise</td>
<td>25.1%</td>
<td>87.8%</td>
<td>81.6%</td>
<td>48.3%</td>
<td>71.4%</td>
<td>87.8%</td>
<td></td>
</tr>
<tr>
<td>lightseagreen</td>
<td>12.5%</td>
<td>69.8%</td>
<td>66.7%</td>
<td>49.1%</td>
<td>82.0%</td>
<td>69.8%</td>
<td></td>
</tr>
<tr>
<td>mediumturquoise</td>
<td>28.2%</td>
<td>82.0%</td>
<td>80.0%</td>
<td>49.4%</td>
<td>65.6%</td>
<td>82.0%</td>
<td></td>
</tr>
<tr>
<td>teal</td>
<td>0.0%</td>
<td>50.2%</td>
<td>50.2%</td>
<td>50.0%</td>
<td>100.0%</td>
<td>50.2%</td>
<td></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>aqua</td>
<td></td>
<td>0.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>50.0%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>cyan2</td>
<td></td>
<td>0.0%</td>
<td>93.3%</td>
<td>93.3%</td>
<td>50.0%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>cyan3</td>
<td></td>
<td>0.0%</td>
<td>80.4%</td>
<td>80.4%</td>
<td>50.0%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>cyan4</td>
<td></td>
<td>0.0%</td>
<td>54.5%</td>
<td>54.5%</td>
<td>50.0%</td>
<td>100.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>azure</td>
<td></td>
<td>94.1%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>50.0%</td>
<td>9.9%</td>
<td>100.0%</td>
</tr>
<tr>
<td>azure2</td>
<td></td>
<td>87.8%</td>
<td>93.3%</td>
<td>93.3%</td>
<td>50.0%</td>
<td>5.9%</td>
<td>93.3%</td>
</tr>
<tr>
<td>azure3</td>
<td></td>
<td>75.7%</td>
<td>80.4%</td>
<td>80.4%</td>
<td>50.0%</td>
<td>5.9%</td>
<td>80.4%</td>
</tr>
<tr>
<td>azure4</td>
<td></td>
<td>51.4%</td>
<td>54.5%</td>
<td>54.5%</td>
<td>50.0%</td>
<td>5.8%</td>
<td>80.4%</td>
</tr>
<tr>
<td>cadetblue</td>
<td></td>
<td>37.3%</td>
<td>62.0%</td>
<td>62.7%</td>
<td>50.5%</td>
<td>40.6%</td>
<td>62.7%</td>
</tr>
<tr>
<td>lightcyan</td>
<td></td>
<td>87.8%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>50.0%</td>
<td>12.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lightcyan2</td>
<td></td>
<td>82.0%</td>
<td>93.3%</td>
<td>93.3%</td>
<td>50.0%</td>
<td>12.2%</td>
<td>93.3%</td>
</tr>
<tr>
<td>lightcyan3</td>
<td></td>
<td>70.6%</td>
<td>80.4%</td>
<td>80.4%</td>
<td>50.0%</td>
<td>12.2%</td>
<td>80.4%</td>
</tr>
<tr>
<td>lightcyan4</td>
<td></td>
<td>47.8%</td>
<td>54.5%</td>
<td>54.5%</td>
<td>50.0%</td>
<td>12.2%</td>
<td>54.5%</td>
</tr>
<tr>
<td>turquoise1</td>
<td></td>
<td>0.0%</td>
<td>96.1%</td>
<td>100.0%</td>
<td>50.7%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>turquoise2</td>
<td></td>
<td>0.0%</td>
<td>89.8%</td>
<td>93.3%</td>
<td>50.6%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>turquoise3</td>
<td></td>
<td>0.0%</td>
<td>77.3%</td>
<td>80.4%</td>
<td>50.7%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>turquoise4</td>
<td></td>
<td>0.0%</td>
<td>52.5%</td>
<td>54.5%</td>
<td>50.6%</td>
<td>100.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>darkslategray</td>
<td></td>
<td>18.4%</td>
<td>31.0%</td>
<td>31.0%</td>
<td>50.0%</td>
<td>40.5%</td>
<td>31.0%</td>
</tr>
<tr>
<td>darkslategray1</td>
<td></td>
<td>59.2%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>50.0%</td>
<td>40.8%</td>
<td>100.0%</td>
</tr>
<tr>
<td>darkslategray2</td>
<td></td>
<td>55.3%</td>
<td>93.3%</td>
<td>93.3%</td>
<td>50.0%</td>
<td>40.8%</td>
<td>93.3%</td>
</tr>
<tr>
<td>darkslategray3</td>
<td></td>
<td>47.5%</td>
<td>80.4%</td>
<td>80.4%</td>
<td>50.0%</td>
<td>41.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>darkslategray4</td>
<td></td>
<td>32.2%</td>
<td>54.5%</td>
<td>54.5%</td>
<td>50.0%</td>
<td>41.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>darkturquoise</td>
<td></td>
<td>0.0%</td>
<td>80.8%</td>
<td>82.0%</td>
<td>50.2%</td>
<td>100.0%</td>
<td>82.0%</td>
</tr>
<tr>
<td>paleturquoise</td>
<td></td>
<td>68.6%</td>
<td>93.3%</td>
<td>93.3%</td>
<td>50.0%</td>
<td>26.5%</td>
<td>93.3%</td>
</tr>
<tr>
<td>paleturquoise1</td>
<td></td>
<td>73.3%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>50.0%</td>
<td>26.7%</td>
<td>100.0%</td>
</tr>
<tr>
<td>paleturquoise2</td>
<td></td>
<td>68.2%</td>
<td>93.3%</td>
<td>93.3%</td>
<td>50.0%</td>
<td>26.9%</td>
<td>93.3%</td>
</tr>
<tr>
<td>paleturquoise3</td>
<td></td>
<td>58.8%</td>
<td>80.4%</td>
<td>80.4%</td>
<td>50.0%</td>
<td>26.8%</td>
<td>80.4%</td>
</tr>
<tr>
<td>paleturquoise4</td>
<td></td>
<td>40.0%</td>
<td>54.5%</td>
<td>54.5%</td>
<td>50.0%</td>
<td>26.6%</td>
<td>54.5%</td>
</tr>
<tr>
<td>cadetblue1</td>
<td></td>
<td>59.6%</td>
<td>96.1%</td>
<td>100.0%</td>
<td>51.6%</td>
<td>40.4%</td>
<td>100.0%</td>
</tr>
<tr>
<td>cadetblue2</td>
<td></td>
<td>55.7%</td>
<td>89.8%</td>
<td>93.3%</td>
<td>51.6%</td>
<td>40.3%</td>
<td>93.3%</td>
</tr>
<tr>
<td>cadetblue3</td>
<td></td>
<td>47.8%</td>
<td>77.3%</td>
<td>80.4%</td>
<td>51.6%</td>
<td>40.5%</td>
<td>93.3%</td>
</tr>
<tr>
<td>cadetblue4</td>
<td></td>
<td>32.5%</td>
<td>52.5%</td>
<td>54.5%</td>
<td>51.5%</td>
<td>40.3%</td>
<td>93.3%</td>
</tr>
<tr>
<td>powderblue</td>
<td></td>
<td>69.0%</td>
<td>87.8%</td>
<td>90.2%</td>
<td>51.9%</td>
<td>23.5%</td>
<td>90.2%</td>
</tr>
<tr>
<td>lightblue4</td>
<td></td>
<td>40.8%</td>
<td>51.4%</td>
<td>54.5%</td>
<td>53.8%</td>
<td>25.2%</td>
<td>54.5%</td>
</tr>
<tr>
<td>skyblue</td>
<td></td>
<td>52.9%</td>
<td>80.8%</td>
<td>92.2%</td>
<td>54.8%</td>
<td>42.6%</td>
<td>92.2%</td>
</tr>
<tr>
<td>lightblue</td>
<td></td>
<td>67.8%</td>
<td>84.7%</td>
<td>90.2%</td>
<td>54.1%</td>
<td>24.8%</td>
<td>90.2%</td>
</tr>
<tr>
<td>lightblue1</td>
<td></td>
<td>74.9%</td>
<td>93.7%</td>
<td>100.0%</td>
<td>54.2%</td>
<td>25.1%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lightblue2</td>
<td></td>
<td>69.8%</td>
<td>87.5%</td>
<td>93.3%</td>
<td>54.2%</td>
<td>25.2%</td>
<td>93.3%</td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>lightblue3</td>
<td></td>
<td>60.4%</td>
<td>75.3%</td>
<td>80.4%</td>
<td>54.2%</td>
<td>24.9%</td>
<td>80.4%</td>
</tr>
<tr>
<td>deepskyblue</td>
<td></td>
<td>0.0%</td>
<td>74.9%</td>
<td>100.0%</td>
<td>54.2%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>deepskyblue2</td>
<td></td>
<td>0.0%</td>
<td>69.8%</td>
<td>93.3%</td>
<td>54.2%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>deepskyblue3</td>
<td></td>
<td>0.0%</td>
<td>60.4%</td>
<td>80.4%</td>
<td>54.1%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>deepskyblue4</td>
<td></td>
<td>0.0%</td>
<td>40.8%</td>
<td>54.5%</td>
<td>54.2%</td>
<td>100.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>lightskyblue3</td>
<td></td>
<td>55.3%</td>
<td>71.4%</td>
<td>80.4%</td>
<td>56.0%</td>
<td>31.2%</td>
<td>80.4%</td>
</tr>
<tr>
<td>skyblue1</td>
<td></td>
<td>52.9%</td>
<td>80.8%</td>
<td>100.0%</td>
<td>56.8%</td>
<td>47.1%</td>
<td>100.0%</td>
</tr>
<tr>
<td>skyblue2</td>
<td></td>
<td>49.4%</td>
<td>75.3%</td>
<td>93.3%</td>
<td>56.8%</td>
<td>47.1%</td>
<td>93.3%</td>
</tr>
<tr>
<td>skyblue3</td>
<td></td>
<td>42.4%</td>
<td>65.1%</td>
<td>80.4%</td>
<td>56.7%</td>
<td>47.3%</td>
<td>80.4%</td>
</tr>
<tr>
<td>skyblue4</td>
<td></td>
<td>29.0%</td>
<td>43.9%</td>
<td>54.5%</td>
<td>56.9%</td>
<td>46.8%</td>
<td>54.5%</td>
</tr>
<tr>
<td>lightskyblue</td>
<td></td>
<td>52.9%</td>
<td>80.8%</td>
<td>98.0%</td>
<td>56.4%</td>
<td>46.0%</td>
<td>98.0%</td>
</tr>
<tr>
<td>lightskyblue1</td>
<td></td>
<td>69.0%</td>
<td>88.6%</td>
<td>100.0%</td>
<td>56.1%</td>
<td>31.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lightskyblue2</td>
<td></td>
<td>64.3%</td>
<td>82.7%</td>
<td>93.3%</td>
<td>56.1%</td>
<td>31.1%</td>
<td>93.3%</td>
</tr>
<tr>
<td>lightskyblue3</td>
<td></td>
<td>37.6%</td>
<td>48.2%</td>
<td>54.5%</td>
<td>56.2%</td>
<td>30.9%</td>
<td>54.5%</td>
</tr>
<tr>
<td>aliceblue</td>
<td></td>
<td>94.1%</td>
<td>97.3%</td>
<td>100.0%</td>
<td>57.8%</td>
<td>5.9%</td>
<td>100.0%</td>
</tr>
<tr>
<td>steelblue</td>
<td></td>
<td>27.5%</td>
<td>51.0%</td>
<td>70.6%</td>
<td>57.6%</td>
<td>61.1%</td>
<td>70.6%</td>
</tr>
<tr>
<td>steelblue1</td>
<td></td>
<td>38.8%</td>
<td>72.2%</td>
<td>100.0%</td>
<td>57.6%</td>
<td>61.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>steelblue2</td>
<td></td>
<td>36.1%</td>
<td>67.5%</td>
<td>93.3%</td>
<td>57.5%</td>
<td>61.3%</td>
<td>93.3%</td>
</tr>
<tr>
<td>steelblue3</td>
<td></td>
<td>31.0%</td>
<td>58.0%</td>
<td>80.4%</td>
<td>57.5%</td>
<td>61.5%</td>
<td>80.4%</td>
</tr>
<tr>
<td>steelblue4</td>
<td></td>
<td>21.2%</td>
<td>39.2%</td>
<td>54.5%</td>
<td>57.6%</td>
<td>61.2%</td>
<td>54.5%</td>
</tr>
<tr>
<td>slategray</td>
<td></td>
<td>43.9%</td>
<td>50.2%</td>
<td>56.5%</td>
<td>58.3%</td>
<td>22.2%</td>
<td>56.5%</td>
</tr>
<tr>
<td>slategray1</td>
<td></td>
<td>77.6%</td>
<td>88.6%</td>
<td>100.0%</td>
<td>58.5%</td>
<td>22.4%</td>
<td>100.0%</td>
</tr>
<tr>
<td>slategray2</td>
<td></td>
<td>72.5%</td>
<td>82.7%</td>
<td>93.3%</td>
<td>58.5%</td>
<td>22.3%</td>
<td>93.3%</td>
</tr>
<tr>
<td>slategray3</td>
<td></td>
<td>62.4%</td>
<td>71.4%</td>
<td>80.4%</td>
<td>58.3%</td>
<td>22.4%</td>
<td>80.4%</td>
</tr>
<tr>
<td>slategray4</td>
<td></td>
<td>42.4%</td>
<td>48.2%</td>
<td>54.5%</td>
<td>58.6%</td>
<td>22.3%</td>
<td>54.5%</td>
</tr>
<tr>
<td>dodgerblue</td>
<td></td>
<td>11.8%</td>
<td>56.5%</td>
<td>100.0%</td>
<td>58.2%</td>
<td>88.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>dodgerblue2</td>
<td></td>
<td>11.0%</td>
<td>52.5%</td>
<td>93.3%</td>
<td>58.3%</td>
<td>88.2%</td>
<td>93.3%</td>
</tr>
<tr>
<td>dodgerblue3</td>
<td></td>
<td>9.4%</td>
<td>45.5%</td>
<td>80.4%</td>
<td>58.2%</td>
<td>88.3%</td>
<td>80.4%</td>
</tr>
<tr>
<td>dodgerblue4</td>
<td></td>
<td>6.3%</td>
<td>30.6%</td>
<td>54.5%</td>
<td>58.3%</td>
<td>88.5%</td>
<td>54.5%</td>
</tr>
<tr>
<td>lightslategray</td>
<td></td>
<td>46.7%</td>
<td>53.3%</td>
<td>60.0%</td>
<td>58.3%</td>
<td>22.2%</td>
<td>60.0%</td>
</tr>
<tr>
<td>lightsteelblue</td>
<td></td>
<td>69.0%</td>
<td>76.9%</td>
<td>87.1%</td>
<td>59.4%</td>
<td>20.7%</td>
<td>87.1%</td>
</tr>
<tr>
<td>lightsteelblue1</td>
<td></td>
<td>79.2%</td>
<td>88.2%</td>
<td>100.0%</td>
<td>59.4%</td>
<td>20.8%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lightsteelblue2</td>
<td></td>
<td>73.7%</td>
<td>82.4%</td>
<td>93.3%</td>
<td>59.3%</td>
<td>21.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>lightsteelblue3</td>
<td></td>
<td>63.5%</td>
<td>71.0%</td>
<td>80.4%</td>
<td>59.3%</td>
<td>21.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>lightsteelblue4</td>
<td></td>
<td>43.1%</td>
<td>48.2%</td>
<td>54.5%</td>
<td>59.2%</td>
<td>20.9%</td>
<td>54.5%</td>
</tr>
<tr>
<td>cornflowerblue</td>
<td></td>
<td>39.2%</td>
<td>58.4%</td>
<td>92.9%</td>
<td>60.7%</td>
<td>57.8%</td>
<td>92.9%</td>
</tr>
<tr>
<td>royalblue</td>
<td></td>
<td>25.5%</td>
<td>41.2%</td>
<td>88.2%</td>
<td>62.5%</td>
<td>71.1%</td>
<td>88.2%</td>
</tr>
<tr>
<td>royalblue1</td>
<td></td>
<td>28.2%</td>
<td>46.3%</td>
<td>100.0%</td>
<td>62.5%</td>
<td>71.8%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

continues on next page

### 3.1. 24-bit Colors
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>royalblue2</td>
<td><img src="image1" alt="Image" /></td>
<td>26.3%</td>
<td>43.1%</td>
<td>93.3%</td>
<td>62.5%</td>
<td>71.8%</td>
<td>93.3%</td>
</tr>
<tr>
<td>royalblue3</td>
<td><img src="image2" alt="Image" /></td>
<td>22.7%</td>
<td>37.3%</td>
<td>80.4%</td>
<td>62.5%</td>
<td>71.7%</td>
<td>80.4%</td>
</tr>
<tr>
<td>royalblue4</td>
<td><img src="image3" alt="Image" /></td>
<td>15.3%</td>
<td>25.1%</td>
<td>54.5%</td>
<td>62.5%</td>
<td>71.9%</td>
<td>54.5%</td>
</tr>
<tr>
<td>blue</td>
<td><img src="image4" alt="Image" /></td>
<td>0.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>66.7%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>blue2</td>
<td><img src="image5" alt="Image" /></td>
<td>0.0%</td>
<td>0.0%</td>
<td>93.3%</td>
<td>66.7%</td>
<td>100.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>blue3</td>
<td><img src="image6" alt="Image" /></td>
<td>0.0%</td>
<td>0.0%</td>
<td>80.4%</td>
<td>66.7%</td>
<td>100.0%</td>
<td>80.4%</td>
</tr>
<tr>
<td>blue4</td>
<td><img src="image7" alt="Image" /></td>
<td>0.0%</td>
<td>0.0%</td>
<td>54.5%</td>
<td>66.7%</td>
<td>100.0%</td>
<td>54.5%</td>
</tr>
<tr>
<td>navy</td>
<td><img src="image8" alt="Image" /></td>
<td>0.0%</td>
<td>0.0%</td>
<td>50.2%</td>
<td>66.7%</td>
<td>100.0%</td>
<td>50.2%</td>
</tr>
<tr>
<td>lavender</td>
<td><img src="image9" alt="Image" /></td>
<td>90.2%</td>
<td>90.2%</td>
<td>98.0%</td>
<td>66.7%</td>
<td>8.0%</td>
<td>98.0%</td>
</tr>
<tr>
<td>ghostwhite</td>
<td><img src="image10" alt="Image" /></td>
<td>97.3%</td>
<td>97.3%</td>
<td>100.0%</td>
<td>66.7%</td>
<td>2.7%</td>
<td>100.0%</td>
</tr>
<tr>
<td>midnightblue</td>
<td><img src="image11" alt="Image" /></td>
<td>9.8%</td>
<td>9.8%</td>
<td>43.9%</td>
<td>66.7%</td>
<td>77.7%</td>
<td>43.9%</td>
</tr>
<tr>
<td>slateblue</td>
<td><img src="image12" alt="Image" /></td>
<td>41.6%</td>
<td>35.3%</td>
<td>80.4%</td>
<td>69.0%</td>
<td>56.1%</td>
<td>80.4%</td>
</tr>
<tr>
<td>slateblue1</td>
<td><img src="image13" alt="Image" /></td>
<td>51.4%</td>
<td>43.5%</td>
<td>100.0%</td>
<td>69.0%</td>
<td>56.5%</td>
<td>100.0%</td>
</tr>
<tr>
<td>slateblue3</td>
<td><img src="image14" alt="Image" /></td>
<td>41.2%</td>
<td>34.9%</td>
<td>80.4%</td>
<td>69.0%</td>
<td>56.6%</td>
<td>80.4%</td>
</tr>
<tr>
<td>slateblue4</td>
<td><img src="image15" alt="Image" /></td>
<td>27.8%</td>
<td>23.5%</td>
<td>54.5%</td>
<td>69.0%</td>
<td>56.8%</td>
<td>54.5%</td>
</tr>
<tr>
<td>lightslateblue</td>
<td><img src="image16" alt="Image" /></td>
<td>51.8%</td>
<td>43.9%</td>
<td>100.0%</td>
<td>69.0%</td>
<td>56.1%</td>
<td>100.0%</td>
</tr>
<tr>
<td>slateblue2</td>
<td><img src="image17" alt="Image" /></td>
<td>47.8%</td>
<td>40.4%</td>
<td>93.3%</td>
<td>69.0%</td>
<td>56.7%</td>
<td>93.3%</td>
</tr>
<tr>
<td>darkslateblue</td>
<td><img src="image18" alt="Image" /></td>
<td>28.2%</td>
<td>23.9%</td>
<td>54.5%</td>
<td>69.0%</td>
<td>56.1%</td>
<td>93.3%</td>
</tr>
<tr>
<td>mediumslateblue</td>
<td><img src="image19" alt="Image" /></td>
<td>48.2%</td>
<td>40.8%</td>
<td>93.3%</td>
<td>69.0%</td>
<td>56.3%</td>
<td>93.3%</td>
</tr>
<tr>
<td>mediumpurple</td>
<td><img src="image20" alt="Image" /></td>
<td>57.6%</td>
<td>43.9%</td>
<td>85.9%</td>
<td>72.1%</td>
<td>48.9%</td>
<td>85.9%</td>
</tr>
<tr>
<td>mediumpurple1</td>
<td><img src="image21" alt="Image" /></td>
<td>67.1%</td>
<td>51.0%</td>
<td>100.0%</td>
<td>72.1%</td>
<td>49.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>mediumpurple2</td>
<td><img src="image22" alt="Image" /></td>
<td>62.4%</td>
<td>47.5%</td>
<td>93.3%</td>
<td>72.1%</td>
<td>49.2%</td>
<td>93.3%</td>
</tr>
<tr>
<td>mediumpurple3</td>
<td><img src="image23" alt="Image" /></td>
<td>53.7%</td>
<td>40.8%</td>
<td>80.4%</td>
<td>72.1%</td>
<td>49.3%</td>
<td>80.4%</td>
</tr>
<tr>
<td>mediumpurple4</td>
<td><img src="image24" alt="Image" /></td>
<td>36.5%</td>
<td>27.8%</td>
<td>54.5%</td>
<td>72.1%</td>
<td>48.9%</td>
<td>54.5%</td>
</tr>
<tr>
<td>purple1</td>
<td><img src="image25" alt="Image" /></td>
<td>60.8%</td>
<td>18.8%</td>
<td>100.0%</td>
<td>75.3%</td>
<td>81.2%</td>
<td>100.0%</td>
</tr>
<tr>
<td>purple2</td>
<td><img src="image26" alt="Image" /></td>
<td>56.9%</td>
<td>17.3%</td>
<td>93.3%</td>
<td>75.3%</td>
<td>81.5%</td>
<td>93.3%</td>
</tr>
<tr>
<td>purple3</td>
<td><img src="image27" alt="Image" /></td>
<td>49.0%</td>
<td>14.9%</td>
<td>80.4%</td>
<td>75.3%</td>
<td>81.5%</td>
<td>80.4%</td>
</tr>
<tr>
<td>purple4</td>
<td><img src="image28" alt="Image" /></td>
<td>33.3%</td>
<td>10.2%</td>
<td>54.5%</td>
<td>75.4%</td>
<td>81.3%</td>
<td>54.5%</td>
</tr>
<tr>
<td>blueviolet</td>
<td><img src="image29" alt="Image" /></td>
<td>54.1%</td>
<td>16.9%</td>
<td>88.6%</td>
<td>75.3%</td>
<td>81.0%</td>
<td>88.6%</td>
</tr>
<tr>
<td>rebeccapurple</td>
<td><img src="image30" alt="Image" /></td>
<td>40.0%</td>
<td>20.0%</td>
<td>60.0%</td>
<td>75.0%</td>
<td>66.7%</td>
<td>60.0%</td>
</tr>
<tr>
<td>indigo</td>
<td><img src="image31" alt="Image" /></td>
<td>29.4%</td>
<td>0.0%</td>
<td>51.0%</td>
<td>76.3%</td>
<td>100.0%</td>
<td>51.0%</td>
</tr>
<tr>
<td>purple</td>
<td><img src="image32" alt="Image" /></td>
<td>62.7%</td>
<td>12.5%</td>
<td>94.1%</td>
<td>76.9%</td>
<td>86.7%</td>
<td>94.1%</td>
</tr>
<tr>
<td>darkorchid</td>
<td><img src="image33" alt="Image" /></td>
<td>60.0%</td>
<td>19.6%</td>
<td>80.0%</td>
<td>77.8%</td>
<td>75.5%</td>
<td>80.0%</td>
</tr>
<tr>
<td>darkorchid1</td>
<td><img src="image34" alt="Image" /></td>
<td>74.9%</td>
<td>24.3%</td>
<td>100.0%</td>
<td>77.8%</td>
<td>75.7%</td>
<td>100.0%</td>
</tr>
<tr>
<td>darkorchid2</td>
<td><img src="image35" alt="Image" /></td>
<td>69.8%</td>
<td>22.7%</td>
<td>93.3%</td>
<td>77.8%</td>
<td>75.6%</td>
<td>93.3%</td>
</tr>
<tr>
<td>darkorchid3</td>
<td><img src="image36" alt="Image" /></td>
<td>60.4%</td>
<td>19.6%</td>
<td>80.4%</td>
<td>77.8%</td>
<td>75.6%</td>
<td>80.4%</td>
</tr>
<tr>
<td>darkorchid4</td>
<td><img src="image37" alt="Image" /></td>
<td>40.8%</td>
<td>13.3%</td>
<td>54.5%</td>
<td>77.8%</td>
<td>75.5%</td>
<td>54.5%</td>
</tr>
<tr>
<td>darkviolet</td>
<td><img src="image38" alt="Image" /></td>
<td>58.0%</td>
<td>0.0%</td>
<td>82.7%</td>
<td>78.4%</td>
<td>100.0%</td>
<td>82.7%</td>
</tr>
</tbody>
</table>

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>mediumorchid1</td>
<td>87.8%</td>
<td>40.0%</td>
<td>100.0%</td>
<td>80.0%</td>
<td>60.0%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>mediumorchid2</td>
<td>82.0%</td>
<td>37.3%</td>
<td>93.3%</td>
<td>80.0%</td>
<td>60.1%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>mediumorchid3</td>
<td>70.6%</td>
<td>32.2%</td>
<td>80.4%</td>
<td>79.9%</td>
<td>60.0%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>mediumorchid4</td>
<td>47.8%</td>
<td>21.6%</td>
<td>54.5%</td>
<td>80.0%</td>
<td>60.4%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>mediumorchid</td>
<td>72.9%</td>
<td>33.3%</td>
<td>82.7%</td>
<td>80.0%</td>
<td>59.7%</td>
<td>82.7%</td>
<td></td>
</tr>
<tr>
<td>plum</td>
<td>86.7%</td>
<td>62.7%</td>
<td>86.7%</td>
<td>83.3%</td>
<td>27.6%</td>
<td>86.7%</td>
<td></td>
</tr>
<tr>
<td>plum1</td>
<td>100.0%</td>
<td>73.3%</td>
<td>100.0%</td>
<td>83.3%</td>
<td>26.7%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>plum2</td>
<td>93.3%</td>
<td>68.2%</td>
<td>93.3%</td>
<td>83.3%</td>
<td>26.9%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>plum3</td>
<td>80.4%</td>
<td>58.8%</td>
<td>80.4%</td>
<td>83.3%</td>
<td>26.8%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>plum4</td>
<td>54.5%</td>
<td>40.0%</td>
<td>54.5%</td>
<td>83.3%</td>
<td>26.6%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>orchid</td>
<td>85.5%</td>
<td>43.9%</td>
<td>83.9%</td>
<td>84.0%</td>
<td>48.6%</td>
<td>85.5%</td>
<td></td>
</tr>
<tr>
<td>orchid3</td>
<td>54.5%</td>
<td>27.8%</td>
<td>53.7%</td>
<td>83.8%</td>
<td>48.9%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>violet</td>
<td>93.3%</td>
<td>51.0%</td>
<td>93.3%</td>
<td>83.3%</td>
<td>45.4%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>magenta2</td>
<td>93.3%</td>
<td>0.0%</td>
<td>93.3%</td>
<td>83.3%</td>
<td>100.0%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>magenta3</td>
<td>80.4%</td>
<td>0.0%</td>
<td>80.4%</td>
<td>83.3%</td>
<td>100.0%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>fuchsia</td>
<td>100.0%</td>
<td>0.0%</td>
<td>100.0%</td>
<td>83.3%</td>
<td>100.0%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>thistle</td>
<td>84.7%</td>
<td>74.9%</td>
<td>84.7%</td>
<td>83.3%</td>
<td>11.6%</td>
<td>84.7%</td>
<td></td>
</tr>
<tr>
<td>thistle1</td>
<td>100.0%</td>
<td>88.2%</td>
<td>100.0%</td>
<td>83.3%</td>
<td>11.8%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>thistle2</td>
<td>93.3%</td>
<td>82.4%</td>
<td>93.3%</td>
<td>83.3%</td>
<td>11.8%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>thistle3</td>
<td>80.4%</td>
<td>71.0%</td>
<td>80.4%</td>
<td>83.3%</td>
<td>11.7%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>thistle4</td>
<td>54.5%</td>
<td>48.2%</td>
<td>54.5%</td>
<td>83.3%</td>
<td>11.5%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>webpurple</td>
<td>50.2%</td>
<td>0.0%</td>
<td>50.2%</td>
<td>83.3%</td>
<td>100.0%</td>
<td>50.2%</td>
<td></td>
</tr>
<tr>
<td>darkmagenta</td>
<td>54.5%</td>
<td>0.0%</td>
<td>54.5%</td>
<td>83.3%</td>
<td>100.0%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>orchid1</td>
<td>100.0%</td>
<td>51.4%</td>
<td>98.0%</td>
<td>84.0%</td>
<td>48.6%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>orchid2</td>
<td>93.3%</td>
<td>47.8%</td>
<td>91.4%</td>
<td>84.1%</td>
<td>48.7%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>orchid3</td>
<td>80.4%</td>
<td>41.2%</td>
<td>78.8%</td>
<td>84.0%</td>
<td>48.8%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>maroon1</td>
<td>100.0%</td>
<td>20.4%</td>
<td>70.2%</td>
<td>89.6%</td>
<td>79.6%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>maroon2</td>
<td>93.3%</td>
<td>18.8%</td>
<td>65.5%</td>
<td>89.6%</td>
<td>79.8%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>maroon3</td>
<td>80.4%</td>
<td>16.1%</td>
<td>56.5%</td>
<td>89.5%</td>
<td>80.0%</td>
<td>80.4%</td>
<td></td>
</tr>
<tr>
<td>maroon4</td>
<td>54.5%</td>
<td>11.0%</td>
<td>38.4%</td>
<td>89.5%</td>
<td>79.9%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>violetred</td>
<td>81.6%</td>
<td>12.5%</td>
<td>56.5%</td>
<td>89.4%</td>
<td>84.6%</td>
<td>81.6%</td>
<td></td>
</tr>
<tr>
<td>mediumvioletred</td>
<td>78.0%</td>
<td>8.2%</td>
<td>52.2%</td>
<td>89.5%</td>
<td>89.4%</td>
<td>78.0%</td>
<td></td>
</tr>
<tr>
<td>deeppink</td>
<td>100.0%</td>
<td>7.8%</td>
<td>57.6%</td>
<td>91.0%</td>
<td>92.2%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>deeppink2</td>
<td>93.3%</td>
<td>7.1%</td>
<td>53.7%</td>
<td>91.0%</td>
<td>92.4%</td>
<td>93.3%</td>
<td></td>
</tr>
<tr>
<td>deeppink4</td>
<td>54.5%</td>
<td>3.9%</td>
<td>31.4%</td>
<td>91.0%</td>
<td>92.8%</td>
<td>54.5%</td>
<td></td>
</tr>
<tr>
<td>hotpink</td>
<td>100.0%</td>
<td>41.2%</td>
<td>70.6%</td>
<td>91.7%</td>
<td>58.8%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>hotpink1</td>
<td>100.0%</td>
<td>43.1%</td>
<td>70.6%</td>
<td>92.0%</td>
<td>56.9%</td>
<td>100.0%</td>
<td></td>
</tr>
<tr>
<td>hotpink4</td>
<td>54.5%</td>
<td>22.7%</td>
<td>38.4%</td>
<td>91.8%</td>
<td>58.3%</td>
<td>54.5%</td>
<td></td>
</tr>
</tbody>
</table>

continues on next page

3.1. 24-bit Colors
<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>deeppink3</td>
<td></td>
<td>80.4%</td>
<td>6.3%</td>
<td>46.3%</td>
<td>91.0%</td>
<td>92.2%</td>
<td>80.4%</td>
</tr>
<tr>
<td>hotpink2</td>
<td></td>
<td>93.3%</td>
<td>41.6%</td>
<td>65.5%</td>
<td>92.3%</td>
<td>55.5%</td>
<td>93.3%</td>
</tr>
<tr>
<td>hotpink3</td>
<td></td>
<td>80.4%</td>
<td>37.6%</td>
<td>56.5%</td>
<td>92.7%</td>
<td>53.2%</td>
<td>80.4%</td>
</tr>
<tr>
<td>violetred1</td>
<td></td>
<td>100.0%</td>
<td>24.3%</td>
<td>58.8%</td>
<td>92.4%</td>
<td>75.7%</td>
<td>100.0%</td>
</tr>
<tr>
<td>violetred2</td>
<td></td>
<td>93.3%</td>
<td>22.7%</td>
<td>54.9%</td>
<td>92.4%</td>
<td>75.6%</td>
<td>93.3%</td>
</tr>
<tr>
<td>violetred3</td>
<td></td>
<td>80.4%</td>
<td>19.6%</td>
<td>47.1%</td>
<td>92.5%</td>
<td>75.6%</td>
<td>80.4%</td>
</tr>
<tr>
<td>violetred4</td>
<td></td>
<td>54.5%</td>
<td>13.3%</td>
<td>32.2%</td>
<td>92.4%</td>
<td>75.5%</td>
<td>54.5%</td>
</tr>
<tr>
<td>maroon</td>
<td></td>
<td>69.0%</td>
<td>18.8%</td>
<td>37.6%</td>
<td>93.8%</td>
<td>72.7%</td>
<td>69.0%</td>
</tr>
<tr>
<td>lavenderblush4</td>
<td></td>
<td>54.5%</td>
<td>51.4%</td>
<td>52.5%</td>
<td>93.8%</td>
<td>5.8%</td>
<td>54.5%</td>
</tr>
<tr>
<td>lavenderblush</td>
<td></td>
<td>100.0%</td>
<td>94.1%</td>
<td>96.1%</td>
<td>94.4%</td>
<td>5.9%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lavenderblush2</td>
<td></td>
<td>93.3%</td>
<td>87.8%</td>
<td>89.8%</td>
<td>94.0%</td>
<td>5.9%</td>
<td>93.3%</td>
</tr>
<tr>
<td>lavenderblush3</td>
<td></td>
<td>80.4%</td>
<td>75.7%</td>
<td>77.3%</td>
<td>94.4%</td>
<td>5.9%</td>
<td>80.4%</td>
</tr>
<tr>
<td>palevioletred</td>
<td></td>
<td>85.9%</td>
<td>43.9%</td>
<td>57.6%</td>
<td>94.5%</td>
<td>48.9%</td>
<td>85.9%</td>
</tr>
<tr>
<td>palevioletred1</td>
<td></td>
<td>100.0%</td>
<td>51.0%</td>
<td>67.1%</td>
<td>94.5%</td>
<td>49.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>palevioletred2</td>
<td></td>
<td>93.3%</td>
<td>47.5%</td>
<td>62.4%</td>
<td>94.6%</td>
<td>49.2%</td>
<td>93.3%</td>
</tr>
<tr>
<td>palevioletred3</td>
<td></td>
<td>80.4%</td>
<td>40.8%</td>
<td>53.7%</td>
<td>94.6%</td>
<td>49.3%</td>
<td>80.4%</td>
</tr>
<tr>
<td>palevioletred4</td>
<td></td>
<td>54.5%</td>
<td>27.8%</td>
<td>36.5%</td>
<td>94.6%</td>
<td>48.9%</td>
<td>54.5%</td>
</tr>
<tr>
<td>pink1</td>
<td></td>
<td>100.0%</td>
<td>71.0%</td>
<td>77.3%</td>
<td>96.4%</td>
<td>29.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>pink2</td>
<td></td>
<td>93.3%</td>
<td>66.3%</td>
<td>72.2%</td>
<td>96.4%</td>
<td>29.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>pink3</td>
<td></td>
<td>80.4%</td>
<td>56.9%</td>
<td>62.0%</td>
<td>96.4%</td>
<td>29.3%</td>
<td>80.4%</td>
</tr>
<tr>
<td>pink4</td>
<td></td>
<td>54.5%</td>
<td>38.8%</td>
<td>42.4%</td>
<td>96.2%</td>
<td>28.8%</td>
<td>54.5%</td>
</tr>
<tr>
<td>crimson</td>
<td></td>
<td>86.3%</td>
<td>7.8%</td>
<td>23.5%</td>
<td>96.7%</td>
<td>90.9%</td>
<td>86.3%</td>
</tr>
<tr>
<td>pink</td>
<td></td>
<td>100.0%</td>
<td>75.3%</td>
<td>79.6%</td>
<td>97.1%</td>
<td>24.7%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lightpink</td>
<td></td>
<td>100.0%</td>
<td>71.4%</td>
<td>75.7%</td>
<td>97.5%</td>
<td>28.6%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lightpink1</td>
<td></td>
<td>100.0%</td>
<td>68.2%</td>
<td>72.5%</td>
<td>97.7%</td>
<td>31.8%</td>
<td>100.0%</td>
</tr>
<tr>
<td>lightpink2</td>
<td></td>
<td>93.3%</td>
<td>63.5%</td>
<td>67.8%</td>
<td>97.6%</td>
<td>31.9%</td>
<td>93.3%</td>
</tr>
<tr>
<td>lightpink3</td>
<td></td>
<td>80.4%</td>
<td>54.9%</td>
<td>58.4%</td>
<td>97.7%</td>
<td>31.7%</td>
<td>80.4%</td>
</tr>
<tr>
<td>lightpink4</td>
<td></td>
<td>54.5%</td>
<td>37.3%</td>
<td>39.6%</td>
<td>97.7%</td>
<td>31.7%</td>
<td>54.5%</td>
</tr>
<tr>
<td>black</td>
<td></td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>gray1</td>
<td></td>
<td>1.2%</td>
<td>1.2%</td>
<td>1.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>1.2%</td>
</tr>
<tr>
<td>gray2</td>
<td></td>
<td>2.0%</td>
<td>2.0%</td>
<td>2.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>2.0%</td>
</tr>
<tr>
<td>gray3</td>
<td></td>
<td>3.1%</td>
<td>3.1%</td>
<td>3.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>3.1%</td>
</tr>
<tr>
<td>gray4</td>
<td></td>
<td>3.9%</td>
<td>3.9%</td>
<td>3.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>3.9%</td>
</tr>
<tr>
<td>gray5</td>
<td></td>
<td>5.1%</td>
<td>5.1%</td>
<td>5.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>5.1%</td>
</tr>
<tr>
<td>gray6</td>
<td></td>
<td>5.9%</td>
<td>5.9%</td>
<td>5.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>5.9%</td>
</tr>
<tr>
<td>gray7</td>
<td></td>
<td>7.1%</td>
<td>7.1%</td>
<td>7.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>7.1%</td>
</tr>
<tr>
<td>gray8</td>
<td></td>
<td>7.8%</td>
<td>7.8%</td>
<td>7.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>7.8%</td>
</tr>
<tr>
<td>gray9</td>
<td></td>
<td>9.0%</td>
<td>9.0%</td>
<td>9.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>9.0%</td>
</tr>
</tbody>
</table>
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>gray10</td>
<td></td>
<td>10.2%</td>
<td>10.2%</td>
<td>10.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>10.2%</td>
</tr>
<tr>
<td>gray11</td>
<td></td>
<td>11.0%</td>
<td>11.0%</td>
<td>11.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>11.0%</td>
</tr>
<tr>
<td>gray12</td>
<td></td>
<td>12.2%</td>
<td>12.2%</td>
<td>12.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>12.2%</td>
</tr>
<tr>
<td>gray13</td>
<td></td>
<td>12.9%</td>
<td>12.9%</td>
<td>12.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>12.9%</td>
</tr>
<tr>
<td>gray14</td>
<td></td>
<td>14.1%</td>
<td>14.1%</td>
<td>14.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>14.1%</td>
</tr>
<tr>
<td>gray15</td>
<td></td>
<td>14.9%</td>
<td>14.9%</td>
<td>14.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>14.9%</td>
</tr>
<tr>
<td>gray16</td>
<td></td>
<td>16.1%</td>
<td>16.1%</td>
<td>16.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>16.1%</td>
</tr>
<tr>
<td>gray17</td>
<td></td>
<td>16.9%</td>
<td>16.9%</td>
<td>16.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>16.9%</td>
</tr>
<tr>
<td>gray18</td>
<td></td>
<td>18.0%</td>
<td>18.0%</td>
<td>18.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>18.0%</td>
</tr>
<tr>
<td>gray19</td>
<td></td>
<td>18.8%</td>
<td>18.8%</td>
<td>18.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>18.8%</td>
</tr>
<tr>
<td>gray20</td>
<td></td>
<td>20.0%</td>
<td>20.0%</td>
<td>20.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>20.0%</td>
</tr>
<tr>
<td>gray21</td>
<td></td>
<td>21.2%</td>
<td>21.2%</td>
<td>21.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>21.2%</td>
</tr>
<tr>
<td>gray22</td>
<td></td>
<td>22.0%</td>
<td>22.0%</td>
<td>22.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>22.0%</td>
</tr>
<tr>
<td>gray23</td>
<td></td>
<td>23.1%</td>
<td>23.1%</td>
<td>23.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>23.1%</td>
</tr>
<tr>
<td>gray24</td>
<td></td>
<td>23.9%</td>
<td>23.9%</td>
<td>23.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>23.9%</td>
</tr>
<tr>
<td>gray25</td>
<td></td>
<td>25.1%</td>
<td>25.1%</td>
<td>25.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>25.1%</td>
</tr>
<tr>
<td>gray26</td>
<td></td>
<td>25.9%</td>
<td>25.9%</td>
<td>25.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>25.9%</td>
</tr>
<tr>
<td>gray27</td>
<td></td>
<td>27.1%</td>
<td>27.1%</td>
<td>27.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>27.1%</td>
</tr>
<tr>
<td>gray28</td>
<td></td>
<td>27.8%</td>
<td>27.8%</td>
<td>27.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>27.8%</td>
</tr>
<tr>
<td>gray29</td>
<td></td>
<td>29.0%</td>
<td>29.0%</td>
<td>29.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>29.0%</td>
</tr>
<tr>
<td>gray30</td>
<td></td>
<td>30.2%</td>
<td>30.2%</td>
<td>30.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>30.2%</td>
</tr>
<tr>
<td>gray31</td>
<td></td>
<td>31.0%</td>
<td>31.0%</td>
<td>31.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>31.0%</td>
</tr>
<tr>
<td>gray32</td>
<td></td>
<td>32.2%</td>
<td>32.2%</td>
<td>32.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>32.2%</td>
</tr>
<tr>
<td>gray33</td>
<td></td>
<td>32.9%</td>
<td>32.9%</td>
<td>32.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>32.9%</td>
</tr>
<tr>
<td>gray34</td>
<td></td>
<td>34.1%</td>
<td>34.1%</td>
<td>34.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>34.1%</td>
</tr>
<tr>
<td>gray35</td>
<td></td>
<td>34.9%</td>
<td>34.9%</td>
<td>34.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>34.9%</td>
</tr>
<tr>
<td>gray36</td>
<td></td>
<td>36.1%</td>
<td>36.1%</td>
<td>36.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>36.1%</td>
</tr>
<tr>
<td>gray37</td>
<td></td>
<td>36.9%</td>
<td>36.9%</td>
<td>36.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>36.9%</td>
</tr>
<tr>
<td>gray38</td>
<td></td>
<td>38.0%</td>
<td>38.0%</td>
<td>38.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>38.0%</td>
</tr>
<tr>
<td>gray39</td>
<td></td>
<td>38.8%</td>
<td>38.8%</td>
<td>38.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>38.8%</td>
</tr>
<tr>
<td>gray40</td>
<td></td>
<td>40.0%</td>
<td>40.0%</td>
<td>40.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>40.0%</td>
</tr>
<tr>
<td>dimgray</td>
<td></td>
<td>41.2%</td>
<td>41.2%</td>
<td>41.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>41.2%</td>
</tr>
<tr>
<td>gray42</td>
<td></td>
<td>42.0%</td>
<td>42.0%</td>
<td>42.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>42.0%</td>
</tr>
<tr>
<td>gray43</td>
<td></td>
<td>43.1%</td>
<td>43.1%</td>
<td>43.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>43.1%</td>
</tr>
<tr>
<td>gray44</td>
<td></td>
<td>43.9%</td>
<td>43.9%</td>
<td>43.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>43.9%</td>
</tr>
<tr>
<td>gray45</td>
<td></td>
<td>45.1%</td>
<td>45.1%</td>
<td>45.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>45.1%</td>
</tr>
<tr>
<td>gray46</td>
<td></td>
<td>45.9%</td>
<td>45.9%</td>
<td>45.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>45.9%</td>
</tr>
<tr>
<td>gray47</td>
<td></td>
<td>47.1%</td>
<td>47.1%</td>
<td>47.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>47.1%</td>
</tr>
</tbody>
</table>

continues on next page

3.1. 24-bit Colors

23
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>gray48</td>
<td></td>
<td>47.8%</td>
<td>47.8%</td>
<td>47.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>47.8%</td>
</tr>
<tr>
<td>gray49</td>
<td></td>
<td>49.0%</td>
<td>49.0%</td>
<td>49.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>49.0%</td>
</tr>
<tr>
<td>gray50</td>
<td></td>
<td>49.8%</td>
<td>49.8%</td>
<td>49.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>49.8%</td>
</tr>
<tr>
<td>webgray</td>
<td></td>
<td>50.2%</td>
<td>50.2%</td>
<td>50.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>50.2%</td>
</tr>
<tr>
<td>gray51</td>
<td></td>
<td>51.0%</td>
<td>51.0%</td>
<td>51.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>51.0%</td>
</tr>
<tr>
<td>gray52</td>
<td></td>
<td>52.2%</td>
<td>52.2%</td>
<td>52.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>52.2%</td>
</tr>
<tr>
<td>gray53</td>
<td></td>
<td>52.9%</td>
<td>52.9%</td>
<td>52.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>52.9%</td>
</tr>
<tr>
<td>gray54</td>
<td></td>
<td>54.1%</td>
<td>54.1%</td>
<td>54.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>54.1%</td>
</tr>
<tr>
<td>gray55</td>
<td></td>
<td>54.9%</td>
<td>54.9%</td>
<td>54.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>54.9%</td>
</tr>
<tr>
<td>gray56</td>
<td></td>
<td>56.1%</td>
<td>56.1%</td>
<td>56.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>56.1%</td>
</tr>
<tr>
<td>gray57</td>
<td></td>
<td>56.9%</td>
<td>56.9%</td>
<td>56.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>56.9%</td>
</tr>
<tr>
<td>gray58</td>
<td></td>
<td>58.0%</td>
<td>58.0%</td>
<td>58.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>58.0%</td>
</tr>
<tr>
<td>gray59</td>
<td></td>
<td>58.8%</td>
<td>58.8%</td>
<td>58.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>58.8%</td>
</tr>
<tr>
<td>gray60</td>
<td></td>
<td>60.0%</td>
<td>60.0%</td>
<td>60.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>60.0%</td>
</tr>
<tr>
<td>gray61</td>
<td></td>
<td>61.2%</td>
<td>61.2%</td>
<td>61.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>61.2%</td>
</tr>
<tr>
<td>gray62</td>
<td></td>
<td>62.0%</td>
<td>62.0%</td>
<td>62.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>62.0%</td>
</tr>
<tr>
<td>gray63</td>
<td></td>
<td>63.1%</td>
<td>63.1%</td>
<td>63.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>63.1%</td>
</tr>
<tr>
<td>gray64</td>
<td></td>
<td>63.9%</td>
<td>63.9%</td>
<td>63.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>63.9%</td>
</tr>
<tr>
<td>gray65</td>
<td></td>
<td>65.1%</td>
<td>65.1%</td>
<td>65.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>65.1%</td>
</tr>
<tr>
<td>gray66</td>
<td></td>
<td>65.9%</td>
<td>65.9%</td>
<td>65.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>65.9%</td>
</tr>
<tr>
<td>darkgray</td>
<td></td>
<td>66.3%</td>
<td>66.3%</td>
<td>66.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>66.3%</td>
</tr>
<tr>
<td>gray67</td>
<td></td>
<td>67.1%</td>
<td>67.1%</td>
<td>67.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>67.1%</td>
</tr>
<tr>
<td>gray68</td>
<td></td>
<td>67.8%</td>
<td>67.8%</td>
<td>67.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>67.8%</td>
</tr>
<tr>
<td>gray69</td>
<td></td>
<td>69.0%</td>
<td>69.0%</td>
<td>69.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>69.0%</td>
</tr>
<tr>
<td>gray70</td>
<td></td>
<td>70.2%</td>
<td>70.2%</td>
<td>70.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>70.2%</td>
</tr>
<tr>
<td>gray71</td>
<td></td>
<td>71.0%</td>
<td>71.0%</td>
<td>71.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>71.0%</td>
</tr>
<tr>
<td>gray72</td>
<td></td>
<td>72.2%</td>
<td>72.2%</td>
<td>72.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>72.2%</td>
</tr>
<tr>
<td>gray73</td>
<td></td>
<td>72.9%</td>
<td>72.9%</td>
<td>72.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>72.9%</td>
</tr>
<tr>
<td>gray74</td>
<td></td>
<td>74.1%</td>
<td>74.1%</td>
<td>74.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>74.1%</td>
</tr>
<tr>
<td>gray</td>
<td></td>
<td>74.5%</td>
<td>74.5%</td>
<td>74.5%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>74.5%</td>
</tr>
<tr>
<td>gray75</td>
<td></td>
<td>74.9%</td>
<td>74.9%</td>
<td>74.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>74.9%</td>
</tr>
<tr>
<td>silver</td>
<td></td>
<td>75.3%</td>
<td>75.3%</td>
<td>75.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>75.3%</td>
</tr>
<tr>
<td>gray76</td>
<td></td>
<td>76.1%</td>
<td>76.1%</td>
<td>76.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>76.1%</td>
</tr>
<tr>
<td>gray77</td>
<td></td>
<td>76.9%</td>
<td>76.9%</td>
<td>76.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>76.9%</td>
</tr>
<tr>
<td>gray78</td>
<td></td>
<td>78.0%</td>
<td>78.0%</td>
<td>78.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>78.0%</td>
</tr>
<tr>
<td>gray79</td>
<td></td>
<td>78.8%</td>
<td>78.8%</td>
<td>78.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>78.8%</td>
</tr>
<tr>
<td>gray80</td>
<td></td>
<td>80.0%</td>
<td>80.0%</td>
<td>80.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>80.0%</td>
</tr>
<tr>
<td>gray81</td>
<td></td>
<td>81.2%</td>
<td>81.2%</td>
<td>81.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>81.2%</td>
</tr>
</tbody>
</table>

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Image</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>gray82</td>
<td></td>
<td>82.0%</td>
<td>82.0%</td>
<td>82.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>82.0%</td>
</tr>
<tr>
<td>lightgray</td>
<td></td>
<td>82.7%</td>
<td>82.7%</td>
<td>82.7%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>82.7%</td>
</tr>
<tr>
<td>gray83</td>
<td></td>
<td>83.1%</td>
<td>83.1%</td>
<td>83.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>83.1%</td>
</tr>
<tr>
<td>gray84</td>
<td></td>
<td>83.9%</td>
<td>83.9%</td>
<td>83.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>83.9%</td>
</tr>
<tr>
<td>gray85</td>
<td></td>
<td>85.1%</td>
<td>85.1%</td>
<td>85.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>85.1%</td>
</tr>
<tr>
<td>gray86</td>
<td></td>
<td>85.9%</td>
<td>85.9%</td>
<td>85.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>85.9%</td>
</tr>
<tr>
<td>gainsboro</td>
<td></td>
<td>86.3%</td>
<td>86.3%</td>
<td>86.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>86.3%</td>
</tr>
<tr>
<td>gray87</td>
<td></td>
<td>87.1%</td>
<td>87.1%</td>
<td>87.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>87.1%</td>
</tr>
<tr>
<td>gray88</td>
<td></td>
<td>87.8%</td>
<td>87.8%</td>
<td>87.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>87.8%</td>
</tr>
<tr>
<td>gray89</td>
<td></td>
<td>89.0%</td>
<td>89.0%</td>
<td>89.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>89.0%</td>
</tr>
<tr>
<td>gray90</td>
<td></td>
<td>89.8%</td>
<td>89.8%</td>
<td>89.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>89.8%</td>
</tr>
<tr>
<td>gray91</td>
<td></td>
<td>91.0%</td>
<td>91.0%</td>
<td>91.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>91.0%</td>
</tr>
<tr>
<td>gray92</td>
<td></td>
<td>92.2%</td>
<td>92.2%</td>
<td>92.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>92.2%</td>
</tr>
<tr>
<td>gray93</td>
<td></td>
<td>92.9%</td>
<td>92.9%</td>
<td>92.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>92.9%</td>
</tr>
<tr>
<td>gray94</td>
<td></td>
<td>94.1%</td>
<td>94.1%</td>
<td>94.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>94.1%</td>
</tr>
<tr>
<td>gray95</td>
<td></td>
<td>94.9%</td>
<td>94.9%</td>
<td>94.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>94.9%</td>
</tr>
<tr>
<td>gray96</td>
<td></td>
<td>96.1%</td>
<td>96.1%</td>
<td>96.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>96.1%</td>
</tr>
<tr>
<td>gray97</td>
<td></td>
<td>96.9%</td>
<td>96.9%</td>
<td>96.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>96.9%</td>
</tr>
<tr>
<td>gray98</td>
<td></td>
<td>98.0%</td>
<td>98.0%</td>
<td>98.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>98.0%</td>
</tr>
<tr>
<td>gray99</td>
<td></td>
<td>98.8%</td>
<td>98.8%</td>
<td>98.8%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>98.8%</td>
</tr>
<tr>
<td>gray100</td>
<td></td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

### 3.2 256 Colors

The built-in capability `color()` accepts a numeric index of any value between 0 and 254, I guess you could call this “Color by number...”, it not recommended, there are many common cases where the colors do not match across terminals!

### 3.3 16 Colors

Recommended for common CLI applications.

Traditional terminals are only capable of 8 colors:

- black
- red
- green
- yellow
- blue
- magenta
• cyan
• white

Prefixed with on_, the given color is used as the background color:

• on_black
• on_red
• on_green
• on_yellow
• on_blue
• on_magenta
• on_cyan
• on_white

The same colors, prefixed with bright_ or bold_, such as bright_blue, provides the other 8 colors of a 16-color terminal:

• bright_black
• bright_red
• bright_green
• bright_yellow
• bright_blue
• bright_magenta
• bright_cyan
• bright_white

Combined, there are actually three shades of grey for 16-color terminals, in ascending order of intensity:

• bright_black: is dark grey.
• white: a mild white.
• bright_white: pure white (ffffff).

Note:

• bright_black is actually a very dark shade of grey!
• yellow is brown, only high-intensity yellow (bright_yellow) is yellow!
• purple is magenta.

Warning: Terminal emulators use different values for any of these 16 colors, the most common of these are displayed at https://en.wikipedia.org/wiki/ANSI_escape_code#3/4_bit. Users can customize these 16 colors as a common “theme”, so that one CLI application appears of the same color theme as the next.

When exact color values are needed, 24-bit Colors should be preferred, by their name or RGB value.
3.4 Monochrome

One small consideration for targeting legacy terminals, such as a vt220, which do not support colors but do support reverse video: select a foreground color, followed by reverse video, rather than selecting a background color directly:

the same desired background color effect as on_background:

```python
>>> print(term.on_green('This will not standout on a vt220'))
>>> print(term.green_reverse('Though some terminals standout more than others'))
```

The second phrase appears as black on green on both color terminals and a green monochrome vt220.
The built-in function `input()` (or `raw_input()`) is pretty good for a basic game:

```python
name = input("What is your name? ")
if sum(map(ord, name)) % 2:
    print(f"{name}?! What a beautiful name!")
else:
    print(f"How interesting, {name} you say?"

But it has drawbacks – it’s no good for interactive apps! This function **will not return until the return key is pressed**, so we can’t do any exciting animations, and we can’t understand or detect arrow keys and others required to make awesome, interactive apps and games!

*Blessed* fixes this issue with a context manager, `cbreak()`, and a single function for all keyboard input, `inkey()`.

### 4.1 inkey()

Let’s just dive right into a rich “event loop”, that awaits a keypress for 3 seconds and tells us what key we pressed.

```python
print(f"{term.home}{term.black_on_skyblue}{term.clear}"
print("press 'q' to quit.
with term.cbreak():
    val =''
    while val.lower() != 'q':
        val = term.inkey(timeout=3)
        if not val:
            print("It sure is quiet in here ...")
        elif val.is_sequence:
            print("got sequence: {0}:{1}.format((str(val), name, val.code))
        elif val:
            print("got {0}.format(val)
        print(f'bye!{term.normal}')
```

cbreak() enters a special mode that ensures `os.read()` on an input stream will return as soon as input is available, as explained in cbreak(3). This mode is combined with `inkey()` to decode multibyte sequences, such as `\0x1bO`, into a unicode-derived Keystroke instance.

The Keystroke returned by `inkey()` is unicode – it may be printed, joined with, or compared to any other unicode strings. It also has these special attributes:

- `is_sequence` (bool): Whether it is an “application” key.
- `code` (int): The keycode, for equality testing.
• `name (str)`: a human-readable name of any “application” key.

## 4.2 Keycodes

When the `is_sequence` property tests `True`, the value of `code` represents a unique application key of the keyboard. `code` may then be compared with attributes of `Terminal`, which are duplicated from those found in `curses(3)`, or those constants in `curses` beginning with phrase `KEY_`, as follows:

Table 1: All Terminal class attribute Keyboard codes, by name

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Example Sequence(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY_BACKSPACE</td>
<td>263</td>
<td>‘\x08’, ‘\x7f’</td>
</tr>
<tr>
<td>KEY_BEGIN</td>
<td>354</td>
<td></td>
</tr>
<tr>
<td>KEY_BTAB</td>
<td>353</td>
<td></td>
</tr>
<tr>
<td>KEY_C1</td>
<td>351</td>
<td></td>
</tr>
<tr>
<td>KEY_C3</td>
<td>352</td>
<td></td>
</tr>
<tr>
<td>KEY_CANCEL</td>
<td>355</td>
<td></td>
</tr>
<tr>
<td>KEY_CATAB</td>
<td>342</td>
<td></td>
</tr>
<tr>
<td>KEY_CENTER</td>
<td>350</td>
<td></td>
</tr>
<tr>
<td>KEY_CLEAR</td>
<td>333</td>
<td></td>
</tr>
<tr>
<td>KEY_CLOSE</td>
<td>356</td>
<td></td>
</tr>
<tr>
<td>KEY_COMMAND</td>
<td>357</td>
<td></td>
</tr>
<tr>
<td>KEY_COPY</td>
<td>358</td>
<td></td>
</tr>
<tr>
<td>KEY_CREATE</td>
<td>359</td>
<td></td>
</tr>
<tr>
<td>KEY_CTAB</td>
<td>341</td>
<td></td>
</tr>
<tr>
<td>KEY_DELETE</td>
<td>330</td>
<td>‘\x1b[3~’</td>
</tr>
<tr>
<td>KEY_DL</td>
<td>328</td>
<td></td>
</tr>
<tr>
<td>KEY_DOWN</td>
<td>258</td>
<td>‘\x1b[1B’, ‘\x1b[OB’</td>
</tr>
<tr>
<td>KEY_EIC</td>
<td>332</td>
<td></td>
</tr>
<tr>
<td>KEY_END</td>
<td>360</td>
<td>‘\x1b[F’, ‘\x1b[K’, ‘\x1b[8~’, ‘\x1b[OF’</td>
</tr>
<tr>
<td>KEY_ENTER</td>
<td>343</td>
<td>‘\n’, ‘\r’, ‘\x1bOM’</td>
</tr>
<tr>
<td>KEY_EOL</td>
<td>335</td>
<td></td>
</tr>
<tr>
<td>KEY_EOS</td>
<td>334</td>
<td></td>
</tr>
<tr>
<td>KEY_ESCAPE</td>
<td>361</td>
<td>‘\x1b’</td>
</tr>
<tr>
<td>KEY_F0</td>
<td>264</td>
<td></td>
</tr>
<tr>
<td>KEY_F1</td>
<td>265</td>
<td>‘\x1bOP’</td>
</tr>
<tr>
<td>KEY_F10</td>
<td>274</td>
<td></td>
</tr>
<tr>
<td>KEY_F11</td>
<td>275</td>
<td></td>
</tr>
<tr>
<td>KEY_F12</td>
<td>276</td>
<td></td>
</tr>
<tr>
<td>KEY_F13</td>
<td>277</td>
<td></td>
</tr>
<tr>
<td>KEY_F14</td>
<td>278</td>
<td></td>
</tr>
<tr>
<td>KEY_F15</td>
<td>279</td>
<td></td>
</tr>
<tr>
<td>KEY_F16</td>
<td>280</td>
<td></td>
</tr>
<tr>
<td>KEY_F17</td>
<td>281</td>
<td></td>
</tr>
<tr>
<td>KEY_F18</td>
<td>282</td>
<td></td>
</tr>
<tr>
<td>KEY_F19</td>
<td>283</td>
<td></td>
</tr>
<tr>
<td>KEY_F2</td>
<td>266</td>
<td>‘\x1bOQ’</td>
</tr>
<tr>
<td>KEY_F20</td>
<td>284</td>
<td></td>
</tr>
<tr>
<td>KEY_F21</td>
<td>285</td>
<td></td>
</tr>
<tr>
<td>KEY_F22</td>
<td>286</td>
<td></td>
</tr>
</tbody>
</table>

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Example Sequence(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY_F23</td>
<td>287</td>
<td></td>
</tr>
<tr>
<td>KEY_F3</td>
<td>267</td>
<td>‘\x1bOR’</td>
</tr>
<tr>
<td>KEY_F4</td>
<td>268</td>
<td>‘\x1bOS’</td>
</tr>
<tr>
<td>KEY_F5</td>
<td>269</td>
<td></td>
</tr>
<tr>
<td>KEY_F6</td>
<td>270</td>
<td></td>
</tr>
<tr>
<td>KEY_F7</td>
<td>271</td>
<td></td>
</tr>
<tr>
<td>KEY_F8</td>
<td>272</td>
<td></td>
</tr>
<tr>
<td>KEY_F9</td>
<td>273</td>
<td></td>
</tr>
<tr>
<td>KEY_FIND</td>
<td>362</td>
<td>‘\x1b[1~’</td>
</tr>
<tr>
<td>KEY_HELP</td>
<td>363</td>
<td></td>
</tr>
<tr>
<td>KEY_HOME</td>
<td>262</td>
<td>‘\x1b[H’, ‘\x1b[7~’, ‘\x1b[OH’</td>
</tr>
<tr>
<td>KEY_IL</td>
<td>329</td>
<td></td>
</tr>
<tr>
<td>KEY_INSERT</td>
<td>331</td>
<td>‘\x1b[2~’</td>
</tr>
<tr>
<td>KEY_KP_0</td>
<td>520</td>
<td>‘\x1bOp’</td>
</tr>
<tr>
<td>KEY_KP_1</td>
<td>521</td>
<td>‘\x1bOq’</td>
</tr>
<tr>
<td>KEY_KP_2</td>
<td>522</td>
<td>‘\x1bOr’</td>
</tr>
<tr>
<td>KEY_KP_3</td>
<td>523</td>
<td>‘\x1bOs’</td>
</tr>
<tr>
<td>KEY_KP_4</td>
<td>524</td>
<td>‘\x1bOt’</td>
</tr>
<tr>
<td>KEY_KP_5</td>
<td>525</td>
<td>‘\x1bOu’</td>
</tr>
<tr>
<td>KEY_KP_6</td>
<td>526</td>
<td>‘\x1bOv’</td>
</tr>
<tr>
<td>KEY_KP_7</td>
<td>527</td>
<td>‘\x1bOw’</td>
</tr>
<tr>
<td>KEY_KP_8</td>
<td>528</td>
<td>‘\x1bOx’</td>
</tr>
<tr>
<td>KEY_KP_9</td>
<td>529</td>
<td>‘\x1bOy’</td>
</tr>
<tr>
<td>KEY_KP_ADD</td>
<td>514</td>
<td>‘\x1bOk’</td>
</tr>
<tr>
<td>KEY_KP_DECIMAL</td>
<td>517</td>
<td>‘\x1bOn’</td>
</tr>
<tr>
<td>KEY_KP_DIVIDE</td>
<td>518</td>
<td>‘\x1bOo’</td>
</tr>
<tr>
<td>KEY_KP_EQUAL</td>
<td>519</td>
<td>‘\x1bOX’</td>
</tr>
<tr>
<td>KEY_KP_MULTIPLY</td>
<td>513</td>
<td>‘\x1bOj’</td>
</tr>
<tr>
<td>KEY_KP_SEPARATOR</td>
<td>515</td>
<td>‘\x1bOL’</td>
</tr>
<tr>
<td>KEY_KP_SUBTRACT</td>
<td>516</td>
<td>‘\x1bOm’</td>
</tr>
<tr>
<td>KEY_LEFT</td>
<td>260</td>
<td>‘\x1b[D’, ‘\x1b[OD’</td>
</tr>
<tr>
<td>KEY_LL</td>
<td>347</td>
<td></td>
</tr>
<tr>
<td>KEY_MARK</td>
<td>364</td>
<td></td>
</tr>
<tr>
<td>KEY_MAX</td>
<td>511</td>
<td></td>
</tr>
<tr>
<td>KEY_MESSAGE</td>
<td>365</td>
<td></td>
</tr>
<tr>
<td>KEY_MIN</td>
<td>257</td>
<td></td>
</tr>
<tr>
<td>KEY_MOUSE</td>
<td>409</td>
<td></td>
</tr>
<tr>
<td>KEY_MOVE</td>
<td>366</td>
<td></td>
</tr>
<tr>
<td>KEY_NEXT</td>
<td>367</td>
<td></td>
</tr>
<tr>
<td>KEY_OPEN</td>
<td>368</td>
<td></td>
</tr>
<tr>
<td>KEY_OPTIONS</td>
<td>369</td>
<td></td>
</tr>
<tr>
<td>KEY_PGDOWN</td>
<td>338</td>
<td>‘\x1b[U’, ‘\x1b[6~’</td>
</tr>
<tr>
<td>KEY_PGUP</td>
<td>339</td>
<td>‘\x1b[V’, ‘\x1b[5~’</td>
</tr>
<tr>
<td>KEY_PREVIOUS</td>
<td>370</td>
<td></td>
</tr>
<tr>
<td>KEY_PRINT</td>
<td>346</td>
<td></td>
</tr>
<tr>
<td>KEY_REDO</td>
<td>371</td>
<td></td>
</tr>
<tr>
<td>KEY_REFERENCE</td>
<td>372</td>
<td></td>
</tr>
<tr>
<td>KEY_REFRESH</td>
<td>373</td>
<td></td>
</tr>
<tr>
<td>KEY_REPLACE</td>
<td>374</td>
<td></td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Example Sequence(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY_RESET</td>
<td>345</td>
<td></td>
</tr>
<tr>
<td>KEY_RESIZE</td>
<td>410</td>
<td></td>
</tr>
<tr>
<td>KEY_RESTART</td>
<td>375</td>
<td></td>
</tr>
<tr>
<td>KEY_RESUME</td>
<td>376</td>
<td></td>
</tr>
<tr>
<td>KEY_RIGHT</td>
<td>261</td>
<td>‘\x1b[C’, ‘\x1b[OC’</td>
</tr>
<tr>
<td>KEY_SAVE</td>
<td>377</td>
<td></td>
</tr>
<tr>
<td>KEY_SBEG</td>
<td>378</td>
<td></td>
</tr>
<tr>
<td>KEY_SCANCEL</td>
<td>379</td>
<td></td>
</tr>
<tr>
<td>KEY_SCOMMAND</td>
<td>380</td>
<td></td>
</tr>
<tr>
<td>KEY_SCOPY</td>
<td>381</td>
<td></td>
</tr>
<tr>
<td>KEY_SCREATE</td>
<td>382</td>
<td></td>
</tr>
<tr>
<td>KEY_SDC</td>
<td>383</td>
<td></td>
</tr>
<tr>
<td>KEY_SDL</td>
<td>384</td>
<td></td>
</tr>
<tr>
<td>KEY_SDOWN</td>
<td>336</td>
<td>‘\x1b[1;2B’</td>
</tr>
<tr>
<td>KEY_SELECT</td>
<td>385</td>
<td>‘\x1b[4~’</td>
</tr>
<tr>
<td>KEY_SEND</td>
<td>386</td>
<td></td>
</tr>
<tr>
<td>KEY_SEOL</td>
<td>387</td>
<td></td>
</tr>
<tr>
<td>KEY_SEXIT</td>
<td>388</td>
<td></td>
</tr>
<tr>
<td>KEY_SFIND</td>
<td>389</td>
<td></td>
</tr>
<tr>
<td>KEY_SHELP</td>
<td>390</td>
<td></td>
</tr>
<tr>
<td>KEY_SHOME</td>
<td>391</td>
<td></td>
</tr>
<tr>
<td>KEY_SIC</td>
<td>392</td>
<td></td>
</tr>
<tr>
<td>KEY_SLEFT</td>
<td>393</td>
<td>‘\x1b[1;2D’</td>
</tr>
<tr>
<td>KEY_SMMESSAGE</td>
<td>394</td>
<td></td>
</tr>
<tr>
<td>KEY_SMOVE</td>
<td>395</td>
<td></td>
</tr>
<tr>
<td>KEY_SNEXT</td>
<td>396</td>
<td></td>
</tr>
<tr>
<td>KEY_SOPTIONS</td>
<td>397</td>
<td></td>
</tr>
<tr>
<td>KEY_SPREVIOUS</td>
<td>398</td>
<td></td>
</tr>
<tr>
<td>KEY_SPRINT</td>
<td>399</td>
<td></td>
</tr>
<tr>
<td>KEY_SREDO</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td>KEY_SREPLACE</td>
<td>401</td>
<td></td>
</tr>
<tr>
<td>KEY_SRESET</td>
<td>344</td>
<td></td>
</tr>
<tr>
<td>KEY_SRIGHT</td>
<td>402</td>
<td>‘\x1b[1;2C’</td>
</tr>
<tr>
<td>KEY_SRSUME</td>
<td>403</td>
<td></td>
</tr>
<tr>
<td>KEY_SSAVE</td>
<td>404</td>
<td></td>
</tr>
<tr>
<td>KEY_SSUSPEND</td>
<td>405</td>
<td></td>
</tr>
<tr>
<td>KEY_STAB</td>
<td>340</td>
<td></td>
</tr>
<tr>
<td>KEY_SUNDO</td>
<td>406</td>
<td></td>
</tr>
<tr>
<td>KEY_SUP</td>
<td>337</td>
<td>‘\x1b[1;2A’</td>
</tr>
<tr>
<td>KEY_SUSPEND</td>
<td>407</td>
<td></td>
</tr>
<tr>
<td>KEY_TAB</td>
<td>512</td>
<td>‘\n’</td>
</tr>
<tr>
<td>KEY_UNDO</td>
<td>408</td>
<td></td>
</tr>
<tr>
<td>KEY_UP</td>
<td>259</td>
<td>‘\x1b[A’, ‘\x1b[OA’</td>
</tr>
<tr>
<td>KEY_UP_LEFT</td>
<td>348</td>
<td></td>
</tr>
<tr>
<td>KEY_UP_RIGHT</td>
<td>349</td>
<td></td>
</tr>
</tbody>
</table>

All such keystrokes can be decoded by blessed, there is a demonstration program, keymatrix.py that tests how many of them you can find!
4.3 delete

Typically, backspace is \^H (8, or 0x08) and delete is \^? (127, or 0x7f).

On some systems however, the key for backspace is actually labeled and transmitted as “delete”, though its function in the operating system behaves just as backspace. Blessed usually returns “backspace” in most situations.

It is highly recommend to accept both KEY_DELETE and KEY_BACKSPACE as having the same meaning except when implementing full screen editors, and provide a choice to enable the delete mode by configuration.

4.4 Alt/meta

Programs with GNU readline, like bash, have Alt combinators, such as ALT+u to uppercase the word after cursor. This is achieved by the configuration option altSendsEscape or metaSendsEscape in xterm.

The default for most terminals, however, is for this key to be bound by the operating system, or, used for inserting international keys, (where the combination ALT+u, a is used to insert the character ä).

It is therefore a recommendation to avoid alt or meta keys entirely in applications.

And instead prefer the ctrl-key combinations, maybe along with raw(), to avoid instructing users to custom-configure their terminal emulators to communicate Alt sequences.

If you still wish to optionall decode them, ALT+z becomes Escape + z (or, in raw form \x1bz). This is detected by blessings as two keystrokes, KEY_ESCAPE and 'z'. Blessings currently provides no further assistance in detecting these key combinations.
If you just want to move the location of the cursor before writing text, and aren’t worried about returning, do something like this:

```python
>>> print(term.home + term.clear)
>>> print(term.move_down(2) + term.move_right(20) + term.bright_red('fire!'))
>>> print(term.move_xy(20, 7) + term.bold('Direct hit!'))
>>> print(term.move_y(term.height - 3))
```

There are four direct movement capabilities:

- `move_xy(x, y)` Position cursor at given `x`, `y`.
- `move_x(x)` Position cursor at column `x`.
- `move_y(y)` Position cursor at row `y`.
- `home` Position cursor at (0, 0).

And four relative capabilities:

- `move_up` or `move_up(y)` Position cursor 1 or `y` row cells above the current position.
- `move_down(y)` Position cursor 1 or `y` row cells below the current position.

---

**Note:**

`move_down` or is often valued as `\n`, which additionally returns the carriage to `column 0, and, depending on your terminal emulator, may also destroy any characters to end of line.

`move_down(1)` is always a safe non-destructive one-notch movement in the downward direction.

- `move_left` or `move_left(x)` Position cursor 1 or `x` column cells left of the current position.
- `move_right` or `move_right(x)` Position cursor 1 or `x` column cells right of the current position.
5.1 Example

The source code of bounce.py combines a small bit of Keyboard input with many of the Terminal location capabilities, home, width, height, and move_xy are used to create a classic game of tennis:

```python
# std imports
from math import floor

# local
from blessed import Terminal

def roundxy(x, y):
    return int(floor(x)), int(floor(y))

term = Terminal()

x, y, xs, ys = 2, 2, 0.4, 0.3
with term.cbreak(), term.hidden_cursor():
    # clear the screen
    print(term.home + term.black_on_olivedrab4 + term.clear)

    # loop every 20ms
    while term.inkey(timeout=0.02) != 'q':
        # erase,
        txt_erase = term.move_xy(*roundxy(x, y)) + ''

        # bounce,
        if x >= (term.width - 1) or x <= 0:
            xs *= -1
        if y >= term.height or y <= 0:
            ys *= -1

        # move,
        x, y = x + xs, y + ys

        # draw!
        txt_ball = term.move_xy(*roundxy(x, y)) + ''
        print(txt_erase + txt_ball, end='', flush=True)
```

5.2 Context Manager

A `contextlib.contextmanager(location())` is provided to move the cursor to an \((x, y)\) screen position and restore the previous position on exit:

```python
with term.location(0, term.height - 1):
    print('Here is the bottom. ')

print('This is back where I came from. ')
```

All parameters to `location()` are optional, we can use it without any arguments at all to restore the cursor location:
with term.location():
    print(term.move_xy(1, 1) + 'Hi Mom!' + term.clear_eol)

**Note:** calls to `location()` may not be nested.

## 5.3 Finding The Cursor

We can determine the cursor’s current position at anytime using `get_location()`.

This uses a kind of “answer back” sequence that your terminal emulator responds to. Because the terminal may not respond, or may take some time to respond, the `timeout` keyword argument can be specified to return coordinates (-1, -1) after a blocking timeout:

```python
>>> term.get_location(timeout=5)
(32, 0)
```

The return value of `get_location()` mirrors the arguments of `location()`:

```python
with term.location(12, 12):
    val = term.get_location()
    print(val)
```

Produces output, `(12, 12)`

Although this wouldn’t be suggested in most applications because of its latency, it certainly simplifies many applications, and, can also be timed, to make a determination of the round-trip time, perhaps even the bandwidth constraints, of a remote terminal!
Any string containing sequences can be measured by blessed using the `length()` method. This means that blessed can measure, right-align, center, or word-wrap its own output!

The `height` and `width` properties always provide a current readout of the size of the window:

```python
>>> term.height, term.width
(34, 102)
```

By combining the measure of the printable width of strings containing sequences with the terminal width, the `center()`, `ljust()`, `rjust()`, and `wrap()` methods “just work” for strings that contain sequences.

```python
with term.location(y=term.height // 2):
    print(term.center(term.bold('press return to begin!')))
    term.inkey()  
```

In the following example, `wrap()` word-wraps a short poem containing sequences:

```python
from blessed import Terminal

term = Terminal()

poem = (term.bold_cyan('Plan difficult tasks'),
        term.cyan('through the simplest tasks'),
        term.bold_cyan('Achieve large tasks'),
        term.cyan('through the smallest tasks'))

for line in poem:
    print('
'.join(term.wrap(line, width=25, subsequent_indent=' ' * 4)))
```

### 6.1 Resizing

To detect when the size of the window changes, you can author a callback for `SIGWINCH` signals:

```python
import signal
from blessed import Terminal

term = Terminal()

def on_resize(sig, action):
    print(f'height={term.height}, width={term.width}')
```

(continues on next page)
signal.signal(signal.SIGWINCH, on_resize)

# wait for keypress
term.inkey()

Note: This is not compatible with Windows! We hope to make a cross-platform API for this in the future https://github.com/jquast/blessed/issues/131.

Sometimes it is necessary to make sense of sequences, and to distinguish them from plain text. The `split_seqs()` method can allow us to iterate over a terminal string by its characters or sequences:

```python
>>> term.split_seqs(term.bold('bbq'))
['\x1b[1m', 'b', 'b', 'q', '\x1b(B', '\x1b[m']
```

Will display something like, `['\x1b[1m', 'b', 'b', 'q', '\x1b(B', '\x1b[m']`

Method `strip_seqs()` can remove all sequences from a string:

```python
>>> phrase = term.bold_black('coffee')
>>> phrase
'\x1b[1m\x1b[30mcoffee\x1b(B\x1b[m'
>>> term.strip_seqs(phrase)
'coffee'
```
A few programs are provided with blessed to help interactively test the various API features, but also serve as examples of using blessed to develop applications.

These examples are not distributed with the package – they are only available in the github repository. You can retrieve them by cloning the repository, or simply downloading the “raw” file link.

Fig. 1: x11_colorpicker.py, bounce.py, worms.py, and plasma.py.

7.1 bounce.py

https://github.com/jquast/blessed/blob/master/bin/bounce.py

This is a very brief, basic primitive non-interactive version of a “classic tennis” video game. It demonstrates basic timed refresh of a bouncing terminal cell.

7.2 cnn.py

https://github.com/jquast/blessed/blob/master/bin/cnn.py

This program uses 3rd-party BeautifulSoup and requests library to fetch the cnn website and display news article titles using the link() method, so that they may be clicked.

7.3 detect-multibyte.py

https://github.com/jquast/blessed/blob/master/bin/detect-multibyte.py

This program also demonstrates how the get_location() method can be used to reliably test whether the terminal emulator of the connecting client is capable of rendering multibyte characters as a single cell.
7.4 editor.py

https://github.com/jquast/blessed/blob/master/bin/editor.py

This program demonstrates using the directional keys and noecho input mode. It acts as a (very dumb) fullscreen editor, with support for saving a file, as well as including a rudimentary line-editor.

7.5 keymatrix.py

https://github.com/jquast/blessed/blob/master/bin/keymatrix.py

This program displays a “gameboard” of all known special KEY_NAME constants. When the key is depressed, it is highlighted, as well as displaying the unicode sequence, integer code, and friendly-name of any key pressed.

7.6 on_resize.py

https://github.com/jquast/blessed/blob/master/bin/on_resize.py

This program installs a SIGWINCH signal handler, which detects screen resizes while also polling for input, displaying keypresses.

This demonstrates how a program can react to screen resize events.

7.7 plasma.py

https://github.com/jquast/blessed/blob/master/bin/plasma.py

This demonstrates using only on_color_rgb() and the built-in colorsys module to quickly display all of the colors of a rainbow in a classic demoscene plasma effect.

7.8 progress_bar.py

https://github.com/jquast/blessed/blob/master/bin/progress_bar.py

This program demonstrates a simple progress bar. All text is written to stderr, to avoid the need to “flush” or emit newlines, and makes use of the move_x (hpa) capability to “overstrike” the display a scrolling progress bar.

7.9 resize.py

https://github.com/jquast/blessed/blob/master/bin/resize.py

This program demonstrates the get_location() method, behaving similar to resize(1) : set environment and terminal settings to current window size. The window size is determined by eliciting an answerback sequence from the connecting terminal emulator.
7.10  tprint.py

https://github.com/jquast/blessed/blob/master/bin/tprint.py

This program demonstrates how users may customize FormattingString styles. Accepting a string style, such as “bold” or “bright_red” as the first argument, all subsequent arguments are displayed by the given style. This shows how a program could provide user-customizable compound formatting names to configure a program’s styling.

7.11  worms.py

https://github.com/jquast/blessed/blob/master/bin/worms.py

This program demonstrates how an interactive game could be made with blessed. It is similar to NIBBLES.BAS or “snake” of early mobile platforms.

7.12  x11_colorpicker.py

https://github.com/jquast/blessed/blob/master/bin/x11_colorpicker.py

This program shows all of the X11 colors, demonstrates a basic keyboard-interactive program and color selection, but is also a useful utility to pick colors!
8.1 color.py

Sub-module providing color functions.

References,
- Measuring Colour by R.W.G. Hunt and M.R. Pointer

**rgb_to_xyz** *(red, green, blue)*
Convert standard RGB color to XYZ color.

Parameters
- **red** *(int)* – RGB value of Red.
- **green** *(int)* – RGB value of Green.
- **blue** *(int)* – RGB value of Blue.

Returns Tuple (X, Y, Z) representing XYZ color

Return type tuple

D65/2° standard illuminant

**xyz_to_lab** *(x_val, y_val, z_val)*
Convert XYZ color to CIE-Lab color.

Parameters
- **x_val** *(float)* – XYZ value of X.
- **y_val** *(float)* – XYZ value of Y.
- **z_val** *(float)* – XYZ value of Z.

Returns Tuple (L, a, b) representing CIE-Lab color

Return type tuple

D65/2° standard illuminant

**rgb_to_lab** *(red, green, blue)*
Convert RGB color to CIE-Lab color.

Parameters
- **red** *(int)* – RGB value of Red.
**Blessed Documentation, Release 1.17.11**

- **green** (*int*) – RGB value of Green.
- **blue** (*int*) – RGB value of Blue.

**Returns**  Tuple (L, a, b) representing CIE-Lab color

**Return type**  *tuple*

D65/2° standard illuminant

**dist_rgb** (*rgb1*, *rgb2*)

Determine distance between two rgb colors.

**Parameters**
- **rgb1** (*tuple*) – RGB color definition
- **rgb2** (*tuple*) – RGB color definition

**Returns**  Square of the distance between provided colors

**Return type**  *float*

This works by treating RGB colors as coordinates in three dimensional space and finding the closest point within the configured color range using the formula:

\[d^2 = (r2 - r1)^2 + (g2 - g1)^2 + (b2 - b1)^2\]

For efficiency, the square of the distance is returned which is sufficient for comparisons

**dist_rgb_weighted** (*rgb1*, *rgb2*)

Determine the weighted distance between two rgb colors.

**Parameters**
- **rgb1** (*tuple*) – RGB color definition
- **rgb2** (*tuple*) – RGB color definition

**Returns**  Square of the distance between provided colors

**Return type**  *float*

Similar to a standard distance formula, the values are weighted to approximate human perception of color differences

For efficiency, the square of the distance is returned which is sufficient for comparisons

**dist_cie76** (*rgb1*, *rgb2*)

Determine distance between two rgb colors using the CIE94 algorithm.

**Parameters**
- **rgb1** (*tuple*) – RGB color definition
- **rgb2** (*tuple*) – RGB color definition

**Returns**  Square of the distance between provided colors

**Return type**  *float*

For efficiency, the square of the distance is returned which is sufficient for comparisons

**dist_cie94** (*rgb1*, *rgb2*)

Determine distance between two rgb colors using the CIE94 algorithm.

**Parameters**
- **rgb1** (*tuple*) – RGB color definition
• rgb2 (tuple) – RGB color definition

Returns Square of the distance between provided colors

Return type float

For efficiency, the square of the distance is returned which is sufficient for comparisons

\texttt{dist\_cie2000}(rgb1, rgb2)

Determine distance between two rgb colors using the CIE2000 algorithm.

Parameters

• rgb1 (tuple) – RGB color definition

• rgb2 (tuple) – RGB color definition

Returns Square of the distance between provided colors

Return type float

For efficiency, the square of the distance is returned which is sufficient for comparisons

\section*{8.2 colorspace.py}

Color reference data.

References,

• https://github.com/freedesktop/xorg-rgb/blob/master/rgb.txt

• https://github.com/ThomasDickey/xterm-snapshots/blob/master/256colres.h

• https://github.com/ThomasDickey/xterm-snapshots/blob/master/XTerm-col.ad

• https://en.wikipedia.org/wiki/ANSI_escape_code#Colors

• https://gist.github.com/XVilka/8346728

• https://devblogs.microsoft.com/commandline/24-bit-color-in-the-windows-console/

• http://jdebp.eu/Softwares/nosh/guide/TerminalCapabilities.html

\texttt{class RGBColor(red, green, blue)}

Named tuple for an RGB color definition.

Create new instance of RGBColor(red, green, blue)

\texttt{RGB\_256TABLE = (RGBColor(red=0, green=0, blue=0), RGBColor(red=205, green=0, blue=0), RGBColor(red=0, green=205, blue=0), ... RGBColor(red=218, green=218, blue=218), RGBColor(red=228, green=228, blue=228), RGBColor(red=238, green=238, blue=238))}

Curses color indices of 8, 16, and 256-color terminals

\texttt{X11\_COLORNAMES\_TO\_RGB = \{'aliceblue': RGBColor(red=240, green=248, blue=255), 'antiquewhite': RGBColor(red=250, green=237, blue=222), ... \}}

X11 Color names to (XTerm-defined) RGB values from xorg-rgb/rgb.txt
8.3 formatters.py

Sub-module providing sequence-formatting functions.

```python
_make_colors()
    Return set of valid colors and their derivatives.

    Return type  set

    Returns  Color names with prefixes

COLORS = {'aliceblue', 'antiquewhite', 'antiquewhite1', 'antiquewhite2', 'antiquewhite3', '

Valid colors and their background (on), bright, and bright-background derivatives.

COMPOUNDABLES = {'blink', 'bold', 'italic', 'reverse', 'standout', 'underline'}

Attributes that may be compounded with colors, by underscore, such as 'reverse_indigo'.

class ParameterizingString(cap, normal='', name='<not specified>')

A Unicode string which can be called as a parameterizing termcap.

For example:

```python
>>> from blessed import Terminal
>>> term = Terminal()
>>> color = ParameterizingString(term.color, term.normal, 'color')
>>> color(9)('color #9')
'\x1b[91mcolor #9\x1b(B\x1b[m'
```

Class constructor accepting 3 positional arguments.

Parameters

- cap (str) – parameterized string suitable for curses.tparm()
- normal (str) – terminating sequence for this capability (optional).
- name (str) – name of this terminal capability (optional).

`__call__`(*args)

Returning `FormattingString` instance for given parameters.

Return evaluated terminal capability (self), receiving arguments `*args`, followed by the terminating sequence (self.normal) into a `FormattingString` capable of being called.

Raises

- `TypeError` – Mismatch between capability and arguments
- `curses.error` – `curses.tparm()` raised an exception

Return type  `FormattingString` or `NullCallableString`

Returns  Callable string for given parameters

class ParameterizingProxyString(fmt_pair, normal='', name='<not specified>')

A Unicode string which can be called to proxy missing termcap entries.

This class supports the function `get_proxy_string()`, and mirrors the behavior of `ParameterizingString`, except that instead of a capability name, receives a format string, and callable to filter the given positional `*args` of `ParameterizingProxyString.__call__()` into a terminal sequence.

For example:
```python
>>> from blessed import Terminal
>>> term = Terminal('screen')
>>> hpa = ParameterizingString(term.hpa, term.normal, 'hpa')
>>> hpa(9)
'u''
>>> fmt = u'\x1b[{0}G'
>>> fmt_arg = lambda *arg: (arg[0] + 1,)
>>> hpa = ParameterizingProxyString((fmt, fmt_arg), term.normal, 'hpa')
>>> hpa(9)
'u'\x1b[10G'
```

Class constructor accepting 4 positional arguments.

**Parameters**

- `fmt_pair (tuple)` – Two element tuple containing: - format string suitable for displaying terminal sequences - callable suitable for receiving `__call__` arguments for formatting string
- `normal (str)` – terminating sequence for this capability (optional).
- `name (str)` – name of this terminal capability (optional).

```
__call__(*args)

Returning `FormattingString` instance for given parameters.
Arguments are determined by the capability. For example, `hpa (move_x)` receives only a single integer, whereas `cup (move)` receives two integers. See documentation in `terminfo(5)` for the given capability.
```

**Return type** `FormattingString`

**Returns** Callable string for given parameters

class `FormattingString (sequence, normal='')`

A Unicode string which doubles as a callable.

This is used for terminal attributes, so that it may be used both directly, or as a callable. When used directly, it simply emits the given terminal sequence. When used as a callable, it wraps the given (string) argument with the 2nd argument used by the class constructor:

```python
>>> from blessed import Terminal
>>> term = Terminal()
>>> style = FormattingString(term.bright_blue, term.normal)
>>> print(repr(style))
u'\x1b[94m'
>>> style('Big Blue')
u'\x1b[94mBig Blue\x1b(B\x1b[m'
```

Class constructor accepting 2 positional arguments.

**Parameters**

- `sequence (str)` – terminal attribute sequence.
- `normal (str)` – terminating sequence for this attribute (optional).

```
__call__(*args)

Return text joined by `sequence` and `normal`.
```

**Raises** `TypeError` – Not a string type

**Return type** `str`

**Returns** Arguments wrapped in `sequence` and `normal`
class `FormattingOtherString` *(direct, target)*
A Unicode string which doubles as a callable for another sequence when called.

This is used for the `move_up()`, `down`, `left`, and `right()` family of functions:

```python
>>> from blessed import Terminal
>>> term = Terminal()
>>> move_right = FormattingOtherString(term.cuf1, term.cuf)
>>> print(repr(move_right))
'u\x1b[C'
>>> print(repr(move_right(666)))
'u\x1b[666C'
>>> print(repr(move_right()))
'u\x1b[C'
```

Class constructor accepting 2 positional arguments.

Parameters

- **direct** *(str)* – capability name for direct formatting, eg `'x' + term.right`.
- **target** *(str)* – capability name for callable, eg `'x' + term.right(99)`.

__call__(*args)

Return text by target.

class `NullCallableString`
A dummy callable Unicode alternative to `FormattingString`.

This is used for colors on terminals that do not support colors, it is just a basic form of unicode that may also act as a callable.

Class constructor.

__call__(*args)

Allow empty string to be callable, returning given string, if any.

When called with an int as the first arg, return an empty Unicode. An int is a good hint that I am a `ParameterizingString`, as there are only about half a dozen string-returning capabilities listed in terminfo(5) which accept non-int arguments, they are seldom used.

When called with a non-int as the first arg (no no args at all), return the first arg, acting in place of `FormattingString` without any attributes.

get_proxy_string *(term, attr)*

Proxy and return callable string for proxied attributes.

Parameters

- **term** *(Terminal)* – `Terminal` instance.
- **attr** *(str)* – terminal capability name that may be proxied.

Return type None or `ParameterizingProxyString`.

Returns `ParameterizingProxyString` for some attributes of some terminal types that support it, where the terminfo(5) database would otherwise come up empty, such as `move_x` attribute for `term.kind` of `screen`. Otherwise, None.

split_compound *(compound)*

Split compound formatting string into segments.

```python
>>> split_compound('bold_underline_bright_blue_on_red')
['bold', 'underline', 'bright_blue', 'on_red']
```
Parameters `compound` *(str)* – a string that may contain compounds, separated by underline (_)._;

Return type list

Returns List of formating string segments

**resolve_capability** *(term, attr)*

Resolve a raw terminal capability using `tigetstr()`.

Parameters

- *term* *(Terminal)* – Terminal instance.
- *attr* *(str)* – terminal capability name.

Returns string of the given terminal capability named by `attr`, which may be empty (u") if not found or not supported by the given *kind*.

Return type *str*

**resolve_color** *(term, color)*

Resolve a simple color name to a callable capability.

This function supports `resolve_attribute()`.

Parameters

- *term* *(Terminal)* – Terminal instance.
- *color* *(str)* – any string found in set COLORS.

Returns a string class instance which emits the terminal sequence for the given color, and may be used as a callable to wrap the given string with such sequence.

Returns `NullCallableString` when `number_of_colors` is 0, otherwise `FormattingString`.

Return type `NullCallableString` or `FormattingString`

**resolve_attribute** *(term, attr)*

Resolve a terminal attribute name into a capability class.

Parameters

- *term* *(Terminal)* – Terminal instance.
- *attr* *(str)* – Sugary, ordinary, or compound formatted terminal capability, such as “red_on_white”, “normal”, “red”, or “bold_on_black”.

Returns a string class instance which emits the terminal sequence for the given terminal capability, or may be used as a callable to wrap the given string with such sequence.

Returns `NullCallableString` when `number_of_colors` is 0, otherwise `FormattingString`.

Return type `NullCallableString` or `FormattingString`
8.4 keyboard.py

Sub-module providing ‘keyboard awareness’.

```python
class Keystroke (ucs='', code=None, name=None):
    A unicode-derived class for describing a single keystroke.
    
    A class instance describes a single keystroke received on input, which may contain multiple characters as a
    multibyte sequence, which is indicated by properties is_sequence returning True.
    
    When the string is a known sequence, code matches terminal class attributes for comparison, such as term.
    KEY_LEFT.
    
    The string-name of the sequence, such as u'KEY_LEFT' is accessed by property name, and is used by the
    __repr__() method to display a human-readable form of the Keystroke this class instance represents. It may
    otherwise by joined, split, or evaluated just as any other unicode string.

    Class constructor.

    static __new__ (cls, ucs='', code=None, name=None)
        Class constructor.

    property is_sequence
        Whether the value represents a multibyte sequence (bool).

    property name
        String-name of key sequence, such as u'KEY_LEFT' (str).

    property code
        Integer keycode value of multibyte sequence (int).

get_keyboard_codes ()
    Return mapping of keycode integer values paired by their curses key-name.

    Return type dict
    Returns Dictionary of (code, name) pairs for curses keyboard constant values and their mnemonic
    name. Such as key 260, with the value of its identity, u'KEY_LEFT'.

    These keys are derived from the attributes by the same of the curses module, with the following exceptions:
    • KEY_DELETE in place of KEY_DC
    • KEY_INSERT in place of KEY_IC
    • KEY_PGUP in place of KEY_PPAGE
    • KEY_PGDOW in place of KEY_NPAGE
    • KEY_ESCAPE in place of KEY_EXIT
    • KEY_SUP in place of KEY_SR
    • KEY_SDOWN in place of KEY_SF

    This function is the inverse of get_curses_keycodes(). With the given override “mixins” listed above,
    the keycode for the delete key will map to our imaginary KEY_DELETE mnemonic, effectively erasing the
    phrase KEY_DC from our code vocabulary for anyone that wishes to use the return value to determine the
    key-name by keycode.

get_keyboard_sequences (term)
    Return mapping of keyboard sequences paired by keycodes.

    Parameters term (blessed.Terminal) — Terminal instance.
```
Returns mapping of keyboard unicode sequences paired by keycodes as integer. This is used as the argument `mapper` to the supporting function `resolve_sequence()`.

Return type OrderedDict

Initialize and return a keyboard map and sequence lookup table, (sequence, keycode) from `Terminal` instance `term`, where `sequence` is a multibyte input sequence of unicode characters, such as `u'\x1b[D'`, and `keycode` is an integer value, matching curses constant such as `term.KEY_LEFT`.

The return value is an OrderedDict instance, with their keys sorted longest-first.

`_alternative_left_right(term)`
Determine and return mapping of left and right arrow keys sequences.

Parameters `term(blessed.Terminal)` – `Terminal` instance.

Return type dict

Returns Dictionary of sequences `term._cuf1`, and `term._cub1`, valued as `KEY_RIGHT`, `KEY_LEFT` (when appropriate).

This function supports `get_terminal_sequences()` to discover the preferred input sequence for the left and right application keys.

It is necessary to check the value of these sequences to ensure we do not use `u' '` and `u''` for `KEY_RIGHT` and `KEY_LEFT`, preferring their true application key sequence, instead.

`DEFAULT_SEQUENCE_MIXIN = (('
', 343), ('\r', 343), ('\x08', 263), ('\t', 512), ('\x1b', 361), ('\x7f', 263), ('\x1b[\bA', ... 260), ('\x1b[\bF', 360), ('\x1b[\bH', 262), ('\x1b[\bP', 265), ('\x1b[\bQ', 266), ('\x1b[\bR', 267), ('\x1b[\bS', 268))`

In a perfect world, terminal emulators would always send exactly what the terminfo(5) capability database plans for them, accordingly by the value of the `TERM` name they declare.

But this isn’t a perfect world. Many vt220-derived terminals, such as those declaring ‘xterm’, will continue to send vt220 codes instead of their native-declared codes, for backwards-compatibility.

This goes for many: rxvt, putty, iTerm.

These “mixins” are used for all terminals, regardless of their type.

Furthermore, curses does not provide sequences sent by the keypad, at least, it does not provide a way to distinguish between keypad 0 and numeric 0.

`CURSES_KEYCODE_OVERRIDE_MIXIN = (('KEY_DELETE', 330), ('KEY_INSERT', 331), ('KEY_PGUP', 339), ('KEY_PGDOWN', 338), ... 337), ('KEY_SDOWN', 336), ('KEY_UP_LEFT', 348), ('KEY_UP_RIGHT', 349), ('KEY_CENTER', 350), ('KEY_BEGIN', 354))`

Override mixins for a few curses constants with easier mnemonics: there may only be a 1:1 mapping when only a keycode (int) is given, where these phrases are preferred.

`_CURSES_KEYCODE_ADDINS = ('TAB', 'KP_MULTIPLY', 'KP_ADD', 'KP_SEPARATOR', 'KP_SUBTRACT', 'KP_EQUAL', 'KP_0', 'KP_1', 'KP_2', 'KP_3', 'KP_4', 'KP_5', 'KP_6', 'KP_7', 'KP_8', 'KP_9')`

Though we may determine keynames and codes for keyboard input that generate multibyte sequences, it is also especially useful to aliases a few basic ASCII characters such as `KEY_TAB` instead of `u'\t'` for uniformity.

Furthermore, many key-names for application keys enabled only by context manager `keypad()` are surprisingly absent. We inject them here directly into the curses module.
8.5 sequences.py

Module providing ‘sequence awareness’.

class Sequence(sequence_text, term)
A “sequence-aware” version of the base str class.

This unicode-derived class understands the effect of escape sequences of printable length, allowing a properly implemented rjust(), ljust(), center(), and length().

Class constructor.

Parameters

- sequence_text (str) – A string that may contain sequences.
- term (blessed.Terminal) – Terminal instance.

ljust (width, fillchar=’ ’)
Return string containing sequences, left-adjusted.

Parameters

- width (int) – Total width given to left-adjust text. If unspecified, the width of the attached terminal is used (default).
- fillchar (str) – String for padding right-of text.

Returns String of text, left-aligned by width.

Return type str

rjust (width, fillchar=’ ’)
Return string containing sequences, right-adjusted.

Parameters

- width (int) – Total width given to right-adjust text. If unspecified, the width of the attached terminal is used (default).
- fillchar (str) – String for padding left-of text.

Returns String of text, right-aligned by width.

Return type str

center (width, fillchar=’ ’)
Return string containing sequences, centered.

Parameters

- width (int) – Total width given to center text. If unspecified, the width of the attached terminal is used (default).
- fillchar (str) – String for padding left and right-of text.

Returns String of text, centered by width.

Return type str

length()
Return the printable length of string containing sequences.

Strings containing term.left or \b will cause “overstrike”, but a length less than 0 is not ever returned. So _\b+ is a length of 1 (displays as +), but \b alone is simply a length of 0.
Some characters may consume more than one cell, mainly those CJK Unified Ideographs (Chinese, Japanese, Korean) defined by Unicode as half or full-width characters.

For example:

```python
>>> from blessed import Terminal
>>> from blessed_sequences import Sequence
>>> term = Terminal()
>>> msg = term.clear + term.red(u'')
>>> Sequence(msg, term).length()
10
```

**Note:** Although accounted for, strings containing sequences such as `term.clear` will not give accurate returns, it is not considered lengthy (a length of 0).

---

**strip** *(chars=None)*
Return string of sequences, leading and trailing whitespace removed.

- **Parameters** `chars` *(str)* – Remove characters in chars instead of whitespace.
- **Return type** `str`
- **Returns** string of sequences with leading and trailing whitespace removed.

**lstrip** *(chars=None)*
Return string of all sequences and leading whitespace removed.

- **Parameters** `chars` *(str)* – Remove characters in chars instead of whitespace.
- **Return type** `str`
- **Returns** string of sequences with leading removed.

**rstrip** *(chars=None)*
Return string of all sequences and trailing whitespace removed.

- **Parameters** `chars` *(str)* – Remove characters in chars instead of whitespace.
- **Return type** `str`
- **Returns** string of sequences with trailing removed.

**strip_seqs**
Return text stripped of only its terminal sequences.

- **Return type** `str`
- **Returns** Text with terminal sequences removed

**padd** *(strip=False)*
Return non-destructive horizontal movement as destructive spacing.

- **Parameters** `strip` *(bool)* – Strip terminal sequences
- **Return type** `str`
- **Returns** Text adjusted for horizontal movement

**class SequenceTextWrapper** *(width, term, **kwargs)*
Object for wrapping/filling text. The public interface consists of the wrap() and fill() methods; the other methods are just there for subclasses to override in order to tweak the default behaviour. If you want to completely replace the main wrapping algorithm, you’ll probably have to override _wrap_chunks().

Several instance attributes control various aspects of wrapping:
width (default: 70)  the maximum width of wrapped lines (unless break_long_words is false)

initial_indent (default: “”) string that will be prepended to the first line of wrapped output. Counts towards the line’s width.

subsequent_indent (default: “”) string that will be prepended to all lines save the first of wrapped output; also counts towards each line’s width.

expand_tabs (default: true)  Expand tabs in input text to spaces before further processing. Each tab will become 0 .. ‘tabsize’ spaces, depending on its position in its line. If false, each tab is treated as a single character.

tabsize (default: 8)  Expand tabs in input text to 0 .. ‘tabsize’ spaces, unless ‘expand_tabs’ is false.

replace_whitespace (default: true)  Replace all whitespace characters in the input text by spaces after tab expansion. Note that if expand_tabs is false and replace_whitespace is true, every tab will be converted to a single space!

fix_sentence_endings (default: false)  Ensure that sentence-ending punctuation is always followed by two spaces. Off by default because the algorithm is (unavoidably) imperfect.

break_long_words (default: true)  Break words longer than ‘width’. If false, those words will not be broken, and some lines might be longer than ‘width’.

break_on_Hyphens (default: true)  Allow breaking hyphenated words. If true, wrapping will occur preferably on whitespaces and right after hyphens part of compound words.

drop_whitespace (default: true)  Drop leading and trailing whitespace from lines.

max_lines (default: None)  Truncate wrapped lines.

placeholder (default: ‘ […]’)  Append to the last line of truncated text.

Class initializer.

This class supports the wrap() method.

_wrap_chunks (chunks)
Sequence-aware variant of textwrap.TextWrapper._wrap_chunks().

Raises ValueError – self.width is not a positive integer

Return type  list

Returns  text chunks adjusted for width

This simply ensures that word boundaries are not broken mid-sequence, as standard python textwrap would incorrectly determine the length of a string containing sequences, and may also break consider sequences part of a “word” that may be broken by hyphen (–), where this implementation corrects both.

_handle_long_word (reversed_chunks, cur_line, cur_len, width)
Sequence-aware textwrap.TextWrapper._handle_long_word().

This simply ensures that word boundaries are not broken mid-sequence, as standard python textwrap would incorrectly determine the length of a string containing sequences, and may also break consider sequences part of a “word” that may be broken by hyphen (–), where this implementation corrects both.

iter_parse (term, text)
Generator yields (text, capability) for characters of text.

value for capability may be None, where text is str of length 1. Otherwise, text is a full matching sequence of given capability.

measure_length (text, term)
Deprecated since version 1.12.0.
Return type  int

Returns  Length of the first sequence in the string

8.6  terminal.py

Module containing Terminal, the primary API entry point.

class Terminal (kind=None, stream=None, force_styling=False)
    An abstraction for color, style, positioning, and input in the terminal.

    This keeps the endless calls to tigetstr() and tparm() out of your code, acts intelligently when somebody
    pipes your output to a non-terminal, and abstracts over the complexity of unbuffered keyboard input. It uses the
    terminfo database to remain portable across terminal types.

    Initialize the terminal.

    Parameters

        • kind (str) – A terminal string as taken by curses.setupterm(). Defaults to the
          value of the TERM environment variable.

          Note: Terminals withing a single process must share a common kind. See __CUR_TERM.

        • stream (file) – A file-like object representing the Terminal output. Defaults to the
          original value of sys.__stdout__, like curses.initscr() does.

          If stream is not a tty, empty Unicode strings are returned for all capability values, so things
          like piping your program output to a pipe or file does not emit terminal sequences.

        • force_styling (bool) – Whether to force the emission of capabilities even if sys.
          __stdout__ does not seem to be connected to a terminal. If you want to force styling to
          not happen, use force_styling=None.

          This comes in handy if users are trying to pipe your output through something like less
          -r or build systems which support decoding of terminal sequences.

__getattr__(attr)
    Return a terminal capability as Unicode string.

    For example, term.bold is a unicode string that may be prepended to text to set the video attribute
    for bold, which should also be terminated with the pairing normal. This capability returns a callable,
    so you can use term.bold("hi") which results in the joining of (term.bold, "hi", term.
    normal).

    Compound formatters may also be used. For example:

>>> term.bold_blink_red_on_green("merry x-mas!")

For a parameterized capability such as move (or cup), pass the parameters as positional arguments:

>>> term.move(line, column)

See the manual page terminfo(5) for a complete list of capabilities and their arguments.

property kind
    Read-only property: Terminal kind determined on class initialization.

    Return type  str
**property does_styling**

Read-only property: Whether this class instance may emit sequences.

**Return type** bool

**property is_a_tty**

Read-only property: Whether stream is a terminal.

**Return type** bool

**property height**

Read-only property: Height of the terminal (in number of lines).

**Return type** int

**property width**

Read-only property: Width of the terminal (in number of columns).

**Return type** int

**property pixel_height**

Read-only property: Height of the terminal (in pixels).

**Return type** int

**property pixel_width**

Read-only property: Width of the terminal (in pixels).

**Return type** int

**location** *(x=None, y=None)*

Context manager for temporarily moving the cursor.

**Parameters**

- **x** *(int)* – horizontal position, from left, 0, to right edge of screen, self.width - 1.
- **y** *(int)* – vertical position, from top, 0, to bottom of screen, self.height - 1.

**Returns** a context manager.

**Return type** Iterator

Move the cursor to a certain position on entry, do any kind of I/O, and upon exit let you print stuff there, then return the cursor to its original position:

```python
term = Terminal()
with term.location(y=0, x=0):
    for row_num in range(term.height-1):
        print('Row #{row_num}')
print(term.clear_eol + 'Back to original location.')
```

Specify x to move to a certain column, y to move to a certain row, both, or neither. If you specify neither, only the saving and restoration of cursor position will happen. This can be useful if you simply want to restore your place after doing some manual cursor movement.

Calls cannot be nested: only one should be entered at a time.

**Note:** The argument order (x, y) differs from the return value order (y, x) of `get_location()`, or argument order (y, x) of `move()`. This is for API Compaibility with the blessings library, sorry for the trouble!
get_location (timeout=None)
Return tuple (row, column) of cursor position.

Parameters timeout (float) – Return after time elapsed in seconds with value (-1, -1) indicating that the remote end did not respond.

Return type tuple

Returns cursor position as tuple in form of (y, x). When a timeout is specified, always ensure the return value is checked for (-1, -1).

The location of the cursor is determined by emitting the u7 terminal capability, or VT100 Query Cursor Position when such capability is undefined, which elicits a response from a reply string described by capability u6, or again VT100’s definition of \x1b[%i%d;%dR when undefined.

The (y, x) return value matches the parameter order of the move_xy() capability. The following sequence should cause the cursor to not move at all:

```python
>>> term = Terminal()
>>> term.move_yx(*term.get_location())
```

And the following should assert True with a terminal:

```python
>>> term = Terminal()
>>> given_y, given_x = 10, 20
>>> with term.location(y=given_y, x=given_x):
...     result_y, result_x = term.get_location()
...     assert given_x == result_x, (given_x, result_x)
...     assert given_y == result_y, (given_y, result_y)
```

fullscreen()
Context manager that switches to secondary screen, restoring on exit.

Under the hood, this switches between the primary screen buffer and the secondary one. The primary one is saved on entry and restored on exit. Likewise, the secondary contents are also stable and are faithfully restored on the next entry:

```
with term.fullscreen():
    main()
```

Note: There is only one primary and one secondary screen buffer. fullscreen() calls cannot be nested, only one should be entered at a time.

hidden_cursor()
Context manager that hides the cursor, setting visibility on exit.

```
with term.hidden_cursor():
    main()
```

Note: hidden_cursor() calls cannot be nested: only one should be entered at a time.

move_xy (x, y)
A callable string that moves the cursor to the given (x, y) screen coordinates.

Parameters
- x (int) – horizontal position, from left, 0, to right edge of screen, self.width - 1.
• \textit{y (int)} – vertical position, from top, 0, to bottom of screen, \textit{self.height - 1}.

\textbf{Return type} \textit{ParameterizingString}

\textbf{Returns} Callable string that moves the cursor to the given coordinates

\textbf{move_yx (y, x)}

A callable string that moves the cursor to the given \((y, x)\) screen coordinates.

\textbf{Parameters}

\begin{itemize}
  \item \textit{y (int)} – vertical position, from top, 0, to bottom of screen, \textit{self.height - 1}.
  \item \textit{x (int)} – horizontal position, from left, 0, to right edge of screen, \textit{self.width - 1}.
\end{itemize}

\textbf{Return type} \textit{ParameterizingString}

\textbf{Returns} Callable string that moves the cursor to the given coordinates

\textbf{property move_left}

Move cursor 1 cells to the left, or callable string for \(n>1\) cells.

\textbf{property move_right}

Move cursor 1 or more cells to the right, or callable string for \(n>1\) cells.

\textbf{property move_up}

Move cursor 1 or more cells upwards, or callable string for \(n>1\) cells.

\textbf{property move_down}

Move cursor 1 or more cells downwards, or callable string for \(n>1\) cells.

\textbf{property color}

A callable string that sets the foreground color.

\textbf{Return type} \textit{ParameterizingString}

The capability is unparameterized until called and passed a number, at which point it returns another string which represents a specific color change. This second string can further be called to color a piece of text and set everything back to normal afterward.

This should not be used directly, but rather a specific color by name or \textit{color_rgb()} value.

\textbf{color_rgb (red, green, blue)}

Provides callable formatting string to set foreground color to the specified RGB color.

\textbf{Parameters}

\begin{itemize}
  \item \textit{red (int)} – RGB value of Red.
  \item \textit{green (int)} – RGB value of Green.
  \item \textit{blue (int)} – RGB value of Blue.
\end{itemize}

\textbf{Return type} \textit{FormattingString}

\textbf{Returns} Callable string that sets the foreground color

If the terminal does not support RGB color, the nearest supported color will be determined using \textit{color_distance_algorithm}.

\textbf{property on_color}

A callable capability that sets the background color.

\textbf{Return type} \textit{ParameterizingString}

\textbf{on_color_rgb (red, green, blue)}

Provides callable formatting string to set background color to the specified RGB color.
Parameters

• **red** (*int*) – RGB value of Red.

• **green** (*int*) – RGB value of Green.

• **blue** (*int*) – RGB value of Blue.

**Return type** *FormattingString*

**Returns** Callable string that sets the foreground color

If the terminal does not support RGB color, the nearest supported color will be determined using *color_distance_algorithm*.

**formatter** (*value*)

Provides callable formatting string to set color and other text formatting options.

**Parameters** *value* (*str*) – Sugary, ordinary, or compound formatted terminal capability, such as “red_on_white”, “normal”, “red”, or “bold_on_black”.

**Return type** *FormattingString* or NullCallableString

**Returns** Callable string that sets color and other text formatting options

Calling `term.formatter('bold_on_red')` is equivalent to `term.bold_on_red`, but a string that is not a valid text formatter will return a NullCallableString. This is intended to allow validation of text formatters without the possibility of inadvertently returning another terminal capability.

**rgb_downconvert** (*red*, *green*, *blue*)

Translate an RGB color to a color code of the terminal’s color depth.

**Parameters**

• **red** (*int*) – RGB value of Red (0-255).

• **green** (*int*) – RGB value of Green (0-255).

• **blue** (*int*) – RGB value of Blue (0-255).

**Return type** *int*

**Returns** Color code of downconverted RGB color

**property normal**

A capability that resets all video attributes.

**Return type** *str*

normal is an alias for sgr0 or exit_attribute_mode. Any styling attributes previously applied, such as foreground or background colors, reverse video, or bold are reset to defaults.

**link** (*url*, *text*, *url_id=''*)

Display text that when touched or clicked, navigates to url.

Optional *url_id* may be specified, so that non-adjacent cells can reference a single target, all cells painted with the same “id” will highlight on hover, rather than any individual one, as described in “Hovering and underlining the id parameter” of gist [https://gist.github.com/egmontkob/eb114294efbcd5adb1944c9f3cb5fed8](https://gist.github.com/egmontkob/eb114294efbcd5adb1944c9f3cb5fed8).

**Parameters**

• **url** (*str*) – Hyperlink URL.

• **text** (*str*) – Clickable text.

• **url_id** (*str*) – Optional ‘id’.
Return type  str

Returns  String of text as a hyperlink to url.

**property stream**
Read-only property: stream the terminal outputs to.

This is a convenience attribute. It is used internally for implied writes performed by context managers `hidden_cursor()`, `fullscreen()`, `location()`, and `keypad()`.

**property number_of_colors**
Number of colors supported by terminal.

Common return values are 0, 8, 16, 256, or 1 << 24.

This may be used to test whether the terminal supports colors, and at what depth, if that’s a concern.

**property color_distance_algorithm**
Color distance algorithm used by `rgb_downconvert()`.

The slowest, but most accurate, ‘cie2000’, is default. Other available options are ‘rgb’, ‘rgb-weighted’, ‘cie76’, and ‘cie94’.

**ljust** *(text, width=None, fillchar=’ ‘)*
Left-align text, which may contain terminal sequences.

Parameters
• text (str) – String to be aligned
• width (int) – Total width to fill with aligned text. If unspecified, the whole width of the terminal is filled.
• fillchar (str) – String for padding the right of text

Return type  str

Returns  String of text, left-aligned by width.

**rjust** *(text, width=None, fillchar=’ ‘)*
Right-align text, which may contain terminal sequences.

Parameters
• text (str) – String to be aligned
• width (int) – Total width to fill with aligned text. If unspecified, the whole width of the terminal is used.
• fillchar (str) – String for padding the left of text

Return type  str

Returns  String of text, right-aligned by width.

**center** *(text, width=None, fillchar=’ ‘)*
Center text, which may contain terminal sequences.

Parameters
• text (str) – String to be centered
• width (int) – Total width in which to center text. If unspecified, the whole width of the terminal is used.
• fillchar (str) – String for padding the left and right of text

Return type  str
**length** *(text)*

Return printable length of a string containing sequences.

**Parameters**
- **text** *(str)* – String to measure. May contain terminal sequences.

**Return type** int

**Returns**
The number of terminal character cells the string will occupy when printed

Wide characters that consume 2 character cells are supported:

```python
>>> term = Terminal()
>>> term.length(term.clear + term.red(u''))
10
```

**Note:** Sequences such as ‘clear’, which is considered as a “movement sequence” because it would move the cursor to \((y, x)\)(0, 0), are evaluated as a printable length of 0.

**strip** *(text, chars=None)*

Return text without sequences and leading or trailing whitespace.

**Return type** str

**Returns**
Text with leading and trailing whitespace removed

```python
>>> term.strip(u' \x1b[0;3m xyz ')
u'xyz'
```

**rstrip** *(text, chars=None)*

Return text without terminal sequences or trailing whitespace.

**Return type** str

**Returns**
Text with terminal sequences and trailing whitespace removed

```python
>>> term.rstrip(u' \x1b[0;3m xyz ')
u' xyz'
```

**lstrip** *(text, chars=None)*

Return text without terminal sequences or leading whitespace.

**Return type** str

**Returns**
Text with terminal sequences and leading whitespace removed

```python
>>> term.lstrip(u' \x1b[0;3m xyz ')
u'xyz '
```

**strip_seqs** *(text)*

Return text stripped of only its terminal sequences.

**Return type** str

**Returns**
Text with terminal sequences removed

```python
>>> term.strip_seqs(u'\x1b[0;3mxyz')
u'xyz'
>>> term.strip_seqs(term.cuf(5) + term.red(u'test'))
u' test'
```
Note: Non-destructive sequences that adjust horizontal distance (such as \b or term.cuf(5)) are replaced by destructive space or erasing.

split_seqs(text, **kwds)
Return text split by individual character elements and sequences.

Parameters
- text (str) – String containing sequences
- kwds – remaining keyword arguments for re.split().

Return type  list[str]
Returns List of sequences and individual characters

wrap(text, width=None, **kwargs)
Text-wrap a string, returning a list of wrapped lines.

Parameters
- text (str) – Unlike textwrap.wrap(), text may contain terminal sequences, such as colors, bold, or underline. By default, tabs in text are expanded by string.expandtabs().
- width (int) – Unlike textwrap.wrap(), width will default to the width of the attached terminal.
- kwargs – See textwrap.TextWrapper

Return type  list
Returns List of wrapped lines

See textwrap.TextWrapper for keyword arguments that can customize wrapping behaviour.

getch()
Read, decode, and return the next byte from the keyboard stream.

Return type  unicode
Returns a single unicode character, or u'' if a multi-byte sequence has not yet been fully received.

This method name and behavior mimics curses getch(void), and it supports inkey(), reading only one byte from the keyboard string at a time. This method should always return without blocking if called after kbhit() has returned True.

Implementors of alternate input stream methods should override this method.

ungetch(text)
Buffer input data to be discovered by next call to inkey().

Parameters text (str) – String to be buffered as keyboard input.

kbhit(timeout=None)
Return whether a keypress has been detected on the keyboard.

This method is used by inkey() to determine if a byte may be read using getch() without blocking. The standard implementation simply uses the select.select() call on stdin.
Parameters `timeout (float)` – When `timeout` is 0, this call is non-blocking, otherwise blocking indefinitely until keypress is detected when None (default). When `timeout` is a positive number, returns after `timeout` seconds have elapsed (float).

Return type **bool**

Returns True if a keypress is awaiting to be read on the keyboard attached to this terminal. When input is not a terminal, False is always returned.

cbreak()

Allow each keystroke to be read immediately after it is pressed.

This is a context manager for `tty.setcbreak()`.

This context manager activates ‘rare’ mode, the opposite of ‘cooked’ mode: On entry, `tty.setcbreak()` mode is activated disabling line-buffering of keyboard input and turning off automatic echo of input as output.

**Note:** You must explicitly print any user input you would like displayed. If you provide any kind of editing, you must handle backspace and other line-editing control functions in this mode as well!

Normally, characters received from the keyboard cannot be read by Python until the Return key is pressed. Also known as *cooked* or *canonical input* mode, it allows the tty driver to provide line-editing before shuffling the input to your program and is the (implicit) default terminal mode set by most unix shells before executing programs.

Technically, this context manager sets the `termios` attributes of the terminal attached to `sys.__stdin__`.

**Note:** `tty.setcbreak()` sets VMIN = 1 and VTIME = 0, see [http://www.unixwiz.net/techtips/termios-vmin-vtime.html](http://www.unixwiz.net/techtips/termios-vmin-vtime.html)

raw()

A context manager for `tty.setraw()`.

Although both `break()` and `raw()` modes allow each keystroke to be read immediately after it is pressed, Raw mode disables processing of input and output.

In cbreak mode, special input characters such as ^C or ^S are interpreted by the terminal driver and excluded from the stdin stream. In raw mode these values are receive by the `inkey()` method.

Because output processing is not done, the newline '\n' is not enough, you must also print carriage return to ensure that the cursor is returned to the first column:

```python
with term.raw():
    print("printing in raw mode", end="\r\n")
```

keypad()

Context manager that enables directional keypad input.

On entering, this puts the terminal into “keyboard_transmit” mode by emitting the keypad_xmit (smkx) capability. On exit, it emits keypad_local (rmkx).

On an IBM-PC keyboard with numeric keypad of terminal-type `xterm`, with numlock off, the lower-left diagonal key transmits sequence `\x1b[F`, translated to `Terminal` attribute `KEY_END`.

However, upon entering `keypad()`, `\x1b[OP` is transmitted, translating to `KEY_LL` (lower-left key), allowing you to determine diagonal direction keys.
inkey (timeout=None, esc_delay=0.35)
Read and return the next keyboard event within given timeout.

Generally, this should be used inside the raw() context manager.

Parameters

• timeout (float) – Number of seconds to wait for a keystroke before returning. When
  None (default), this method may block indefinitely.

• esc_delay (float) – To distinguish between the keystroke of KEY_ESCAPE, and
  sequences beginning with escape, the parameter esc_delay specifies the amount of
  time after receiving escape (chr(27)) to seek for the completion of an application key
  before returning a Keystroke instance for KEY_ESCAPE.

Return type Keystroke.

Returns Keystroke, which may be empty (u'') if timeout is specified and keystroke is
not received.

Note: When used without the context manager cbreak(), or raw(), sys.__stdin__ remains
line-buffered, and this function will block until the return key is pressed!

class WINSZ (ws_row, ws_col, ws_xpixel, ws_ypixel)
Structure represents return value of termios.TIOCGWINSZ.

    ws_row
    rows, in characters

    ws_col
    columns, in characters

    ws_xpixel
    horizontal size, pixels

    ws_ypixel
    vertical size, pixels

Create new instance of WINSZ(ws_row, ws_col, ws_xpixel, ws_ypixel)

    _CUR_TERM = None
Bugs or suggestions? Visit the issue tracker and file an issue. We welcome your bug reports and feature suggestions!
Are you stuck and need support? Give stackoverflow a try. If you’re still having trouble, we’d like to hear about it!
Open an issue in the issue tracker with a well-formed question.
Would you like to contribute? That’s awesome! Pull Requests are always welcome!

### 9.1 Fork

_Blessed_ is a fork of _blessings_. Apologies for the fork, I just couldn’t get the _Keyboard_, and later _Location_ or _Measuring_ code accepted upstream after two major initiatives, the effort was better spent in a fork, where the code is accepted.

Furthermore, a project in the node.js language of the same name, _blessed_, is not related, or a fork of each other in any way.

### 9.2 License

_Blessed_ is under the MIT License. See the LICENSE file. Please enjoy!

### 9.3 Running Tests

Install and run tox:

```
pip install --upgrade tox
```

Py.test is used as the test runner, and with the tox target supporting positional arguments, you may for example use `loopenfailing` with python 3.7, stopping at the first failing test case, and looping (retrying) after a filesystem save is detected:

```
tox -epy37 -- -fx
```

The test runner (tox) ensures all code and documentation complies with standard python style guides, pep8 and pep257, as well as various static analysis tools.
Warning: When you contribute a new feature, make sure it is covered by tests. Likewise, some bug fixes should include a test demonstrating the bug.

9.4 Further Reading

As a developer's API, blessed is often bundled with frameworks and toolsets that dive deeper into Terminal I/O programming than Terminal offers. Here are some recommended readings to help you along:

- The terminfo(5) manpage of your preferred posix-like operating system. The capabilities available as attributes of Terminal are directly mapped to those listed in the column Cap-name.
- The termios(3) of your preferred posix-like operating system.
- The TTY demystified by Linus Åkesson.
- A Brief Introduction to Termios by Nelson Elhage.
- Richard Steven’s Advance Unix Programming (“AUP”) provides two very good chapters, “Terminal I/O” and “Pseudo Terminals”.
- GNU’s The Termcap Manual by Richard M. Stallman.
- Chapter 4 of CUNY’s course material for Introduction to System Programming, by Stewart Weiss
- Chapter 11 of the IEEE Open Group Base Specifications Issue 7, “General Terminal Interface”
- The GNU C Library documentation, section Low-Level Terminal Interface
- The source code of many popular terminal emulators. If there is ever any question of “the meaning of a terminal capability”, or whether or not your preferred terminal emulator actually handles them, read the source! Many modern terminal emulators are now based on libvte.
- The source code of the tty(4), pty(7), and the given “console driver” for any posix-like operating system. If you search thoroughly enough, you will eventually discover a terminal sequence decoder, usually a case switch that translates \x1b[0m into a “reset color” action towards the video driver. Though tty.c linked here is probably not the most interesting, it can get you started:
  - FreeBSD
  - OpenBSD
  - Illumos (Solaris)
  - Minix
  - Linux
- Thomas E. Dickey has been maintaining xterm, as well as a primary maintainer of many related packages such as ncurses for quite a long while. His consistent, well-documented, long-term dedication to xterm, curses, and the many related projects is world-renown.
- termcap & terminfo (O’Reilly Nutshell) by Linda Mui, Tim O’Reilly, and John Strang.
- Note that System-V systems, also known as Unix98 (SunOS, HP-UX, AIX and others) use a Streams interface. On these systems, the ioctl(2) interface provides the PUSH and POP parameters to communicate with a Streams device driver, which differs significantly from Linux and BSD.

Many of these systems provide compatible interfaces for Linux, but they may not always be as complete as the counterpart they emulate, most especially in regards to managing pseudo-terminals.
9.5 The misnomer of ANSI

When people say ‘ANSI’, they are discussing:

- **Standard ECMA-48**: Control Functions for Coded Character Sets
- **ANSI X3.64** from 1981, when the American National Standards Institute adopted the ECMA-48 as standard, which was later withdrawn in 1997 (so in this sense it is *not* an ANSI standard).
- The **ANSI.SYS** driver provided in MS-DOS and clones. The popularity of the IBM Personal Computer and MS-DOS of the era, and its ability to display colored text further populated the idea that such text “is ANSI”.
- The various code pages used in MS-DOS Personal Computers, providing “block art” characters in the 8th bit (int 127-255), paired with ECMA-48 sequences supported by the MS-DOS **ANSI.SYS** driver to create artwork, known as **ANSI art**.
- The ANSI terminal database entry and its many descendants in the **terminfo database**. This is mostly due to terminals compatible with SCO UNIX, which was the successor of Microsoft’s Xenix, which brought some semblance of the Microsoft DOS **ANSI.SYS** driver capabilities.
- **Select Graphics Rendition (SGR)** on vt100 clones, which include many of the common sequences in **ECMA-48**.
- Any sequence started by the **Control-Sequence-Inducer** is often mistakenly termed as an “ANSI Escape Sequence” though not appearing in **ECMA-48** or interpreted by the **ANSI.SYS** driver. The adjoining phrase “Escape Sequence” is so termed because it follows the ASCII character for the escape key (ESC, \x1b).
CHAPTER
TEN

VERSION HISTORY

1.17

• introduced: Hyperlinks, method link(), #116.

• introduced: 24-bit color support, detected by term.number_of_colors == 1 << 24, and 24-bit color foreground method color_rgb() and background method on_color_rgb(), as well as 676 common X11 color attribute names are now possible, such as term.aquamarine_on_wheat, #60.

• introduced: term.move_xy, recommended over built-in move capability, as the argument order, (x, y) matches the return value of get_location(), and all other common graphics library calls, #65.

• introduced: move_up(), move_down(), Terminal.move_left(), move_right() which are strings that move the cursor one cell in the respective direction, are now also callables for moving n cells to the given direction, such as term.move_right(9).

• introduced: pixel_width and pixel_height for libsixel support or general curiosity.

• introduced: formatter() which returns callable formatters for valid text formatters such as ‘red’ or ‘bold_on_red’, returning a NullCallableString if passed an invalid text formatter.

• bugfix: prevent ValueError: I/O operation on closed file on sys.stdin in multi-processing environments, where the keyboard wouldn’t work, anyway.

• bugfix: prevent error condition, ValueError: underlying buffer has been detached in rare conditions where sys.__stdout__ has been detached in test frameworks, #126.

• bugfix: off-by-one error in get_location(), now accounts for %i in cursor_report, #94.

• bugfix: split_seqs() and related functions failed to match when the color index was greater than 15, #101.

• bugfix: Context Managers, fullscreen(), hidden_cursor(), and keypad() now flush the stream after writing their sequences.

• bugfix: chr(127), \x7f has changed from keycode term.DELETE to the more common match, term.BACKSPACE, #115 by jwezel.

• bugfix: ensure FormattingOtherString may be pickled.

• bugfix: Use UTF-8 for keyboard if input encoding cannot be determined.

• deprecated: the curses move() capability is no longer recommended, suggest to use move_xy(), which matches the return value of get_location().

• deprecated: superscript, subscript, shadow, and dim are no longer “compoundable” with colors, such as in phrase Terminal.blue_subscript('a'). These attributes are not typically supported, anyway. Use Unicode text or 256 or 24-bit color codes instead.
1.17.11

- deprecated: additional key names, such as KEY_TAB, are no longer “injected” into the curses module namespace.
- bugfix: briefly tried calling `curses.setupterm()` with os.devnull as the file descriptor, reverted. #59.
- deprecated: `inkey()` no longer raises RuntimeError when `stream` is not a terminal, programs using `inkey()` to block indefinitely if a keyboard is not attached. #69.
- deprecated: using argument `_intr_continue` to method `kbhit()`, behavior is as though such value is always True since 1.9.
- bugfix: Now imports on 3.10+
- bugfix: Fix detection of shift+arrow keys when using tmux. #178.

1.16

- introduced: Windows support?! PR #110 by avylove.

1.15

- enhancement: disable timing integration tests for keyboard routines.
- enhancement: Support python 3.7. PR #102.
- enhancement: Various fixes to test automation PR #108

1.14

- bugfix: `wrap()` misbehaved for text containing newlines, #74.
- bugfix: TypeError when using `PYTHONOPTIMIZE=2` environment variable, #84.
- bugfix: define `blessed.__version__` value, #92.
- bugfix: detect sequences `\x1b[0K` and `\x1b2K`, #95.

1.13

- enhancement: `split_seqs()` introduced, and 4x cost reduction in related sequence-aware functions, #29.
- deprecated: `blessed.sequences.measure_length` function superseded by `iter_parse()` if necessary.
- deprecated: warnings about “binary-packed capabilities” are no longer emitted on strange terminal types, making best effort.

1.12

- enhancement: `get_location()` returns the `(row, col)` position of the cursor at the time of call for attached terminal.
- enhancement: a keyboard now detected as `stdin` when `stream` is `sys.stderr`.

1.11

- enhancement: `inkey()` can return more quickly for combinations such as Alt + Z when MetaSendsEscape is enabled, #30.
- enhancement: `FormattingString` may now be nested, such as `t.red('red', t.underline('rum'))`, #61

1.10

- workaround: provide `sc` and `rc` for Terminals of kind='ansi', repairing `location()` #44.
• bugfix: length of simple SGR reset sequence \x1b[m was not correctly determined on all terminal types, #45.

• deprecated: __intr_continue arguments introduced in 1.8 are now marked deprecated in 1.10: beginning with python 3.5, the default behavior is as though this argument is always True, PEP-475, blessed does the same.

1.9

• enhancement: break_long_words now supported by Terminal.wrap()

• Ignore curses.error message 'tparm() returned NULL': this occurs on win32 or other platforms using a limited curses implementation, such as PDCurses, where curses.tparm() is not implemented, or no terminal capability database is available.

• Context manager keypad() emits sequences that enable “application keys” such as the diagonal keys on the numpad. This is equivalent to curses.window.keypad().

• bugfix: translate keypad application keys correctly.

• enhancement: no longer depend on the ‘2to3’ tool for python 3 support.

• enhancement: allow civis and cnorm (hide_cursor, normal_hide) to work with terminal-type ansi by emulating support by proxy.

• enhancement: new public attribute: kind: the very same as given Terminal.__init__.kind keyword argument. Or, when not given, determined by and equivalent to the TERM Environment variable.

1.8

• enhancement: export keyboard-read function as public method getch(), so that it may be overridden by custom terminal implementers.

• enhancement: allow inkey() and kbhit() to return early when interrupted by signal by passing argument _intr_continue=False.

• enhancement: allow hpa and vpa (move_x, move_y) to work on tmux(1) or screen(1) by emulating support by proxy.

• enhancement: add rstrip() and lstrip(), strips both sequences and trailing or leading whitespace, respectively.

• enhancement: include wcwidth library support for length(): the printable width of many kinds of CJK (Chinese, Japanese, Korean) ideographs and various combining characters may now be determined.

• enhancement: better support for detecting the length or sequences of externally-generated ecma-48 codes when using xterm or alixterm.

• bugfix: when locale.getpreferredencoding() returns empty string or an encoding that is not valid for codecs.getincrementaldecoder, fallback to ASCII and emit a warning.

• bugfix: ensure FormattingString and ParameterizingString may be pickled.

• bugfix: allow ~.inkey and related to be called without a keyboard.

• change: term.keyboard_fd is set None if stream or sys.stdout is not a tty, making term.inkey(), term.cbreak(), term.raw(), no-op.

• bugfix: \x1bOH (KEY_HOME) was incorrectly mapped as KEY_LEFT.

1.7

• Forked github project erikrose/blessings to jquast/blessed, this project was previously known as blessings version 1.6 and prior.
• introduced: context manager `cbreak()`, which is equivalent to entering terminal state by `tty.setcbreak()` and returning on exit, as well as the lesser recommended `raw()`, pairing from `tty.setraw()`.

• introduced: `inkey()`, which will return one or more characters received by the keyboard as a unicode sequence, with additional attributes `code` and `name`. This allows application keys (such as the up arrow, or home key) to be detected. Optional value `timeout` allows for timed poll.

• introduced: `center()`, `rjust()`, `ljust()`, allowing text containing sequences to be aligned to detected horizontal screen width, or by `width` specified.

• introduced: `wrap()` method. Allows text containing sequences to be word-wrapped without breaking mid-sequence, honoring their printable width.

• introduced: `strip()`, strips all sequences and whitespace.

• introduced: `strip_seqs()` strip only sequences.

• introduced: `rstrip()` and `lstrip()` strips both sequences and trailing or leading whitespace, respectively.

• bugfix: cannot call `curses.setupterm()` more than once per process (from `Terminal.__init__()`: Previously, blessed pretended to support several instances of different Terminal kind, but was actually using the kind specified by the first instantiation of `Terminal`. A warning is now issued. Although this is misbehavior is still allowed, a `warnings.WarningMessage` is now emitted to notify about subsequent terminal misbehavior.

• bugfix: resolved issue where `number_of_colors` fails when `does_styling` is `False`. Resolves issue where piping tests output would fail.

• bugfix: warn and set `does_styling` to `False` when the given `kind` is not found in the terminal capability database.

• bugfix: allow unsupported terminal capabilities to be callable just as supported capabilities, so that the return value of `color(n)` may be called on terminals without color capabilities.

• bugfix: for terminals without underline, such as vt220, `term.underline('text')` would emit 'text' + `term.normal`. Now it emits only 'text'.

• enhancement: some attributes are now properties, raise exceptions when assigned.

• enhancement: `pypy` is now a supported python platform implementation.

• enhancement: removed `pokemon curses.error exceptions`.

• enhancement: do not ignore `curses.error` exceptions, unhandled curses errors are legitimate errors and should be reported as a bug.

• enhancement: converted nose tests to pytest, merged travis and tox.

• enhancement: pytest fixtures, paired with a new `@as_subprocess` decorator are used to test a multitude of terminal types.

• enhancement: test accessories `@as_subprocess` resolves various issues with different terminal types that previously went untested.

• deprecation: python2.5 is no longer supported (as tox does not support).

1.6

• Add `does_styling`. This takes `force_styling` into account and should replace most uses of `is_a_tty`.

• Make `is_a_tty` a read-only property like `does_styling`. Writing to it never would have done anything constructive.
Blessed Documentation, Release 1.17.11

- Add `fullscreen()` and `hidden_cursor()` to the auto-generated docs.

1.5.1

- Clean up fabfile, removing the redundant `test` command.
- Add Travis support.
- Make `python setup.py test` work without spurious errors on 2.6.
- Work around a tox parsing bug in its config file.
- Make context managers clean up after themselves even if there’s an exception (Vitja Makarov #29 <https://github.com/erikrose/blessings/pull/29>).
- Parameterizing a capability no longer crashes when there is no tty (<Vitja Makarov #31 <https://github.com/erikrose/blessings/pull/31>)

1.5

- Add syntactic sugar and documentation for `enter_fullscreen` and `exit_fullscreen`.
- Add context managers `fullscreen()` and `hidden_cursor()`.
- Now you can force a `Terminal` to never to emit styles by passing keyword argument `force_styling=None`.

1.4

- Add syntactic sugar for cursor visibility control and single-space-movement capabilities.
- Endorse the `location()` context manager for restoring cursor position after a series of manual movements.
- Fix a bug in which `location()` that wouldn’t do anything when passed zeros.
- Allow tests to be run with `python setup.py test`.

1.3

- Added `number_of_colors`, which tells you how many colors the terminal supports.
- Made `color(n)` and `on_color(n)` callable to wrap a string, like the named colors can. Also, make them both fall back to the `setf` and `setb` capabilities (like the named colors do) if the termcap entries for `setaf` and `setab` are not available.
- Allowed `color` to act as an unparametrized string, not just a callable.
- Made `height` and `width` examine any passed-in stream before falling back to stdout (This rarely if ever affects actual behavior; it’s mostly philosophical).
- Made caching simpler and slightly more efficient.
- Got rid of a reference cycle between `Terminal` and `FormattingString`.
- Updated docs to reflect that terminal addressing (as in `location()`) is 0-based.

1.2

- Added support for Python 3! We need 3.2.3 or greater, because the curses library couldn’t decide whether to accept strs or bytes before that (https://bugs.python.org/issue10570).
- Everything that comes out of the library is now unicode. This lets us support Python 3 without making a mess of the code, and Python 2 should continue to work unless you were testing types (and badly). Please file a bug if this causes trouble for you.
- Changed to the MIT License for better world domination.
• Added Sphinx docs.

1.1

• Added nicely named attributes for colors.
• Introduced compound formatting.
• Added wrapper behavior for styling and colors.
• Let you force capabilities to be non-empty, even if the output stream is not a terminal.
• Added `is_a_tty` to determine whether the output stream is a terminal.
• Sugared the remaining interesting string capabilities.
• Allow `location()` to operate on just an x or y coordinate.

1.0

• Extracted Blessed from `nose-progressive`.
INDEXES

- genindex
- modindex
b

blessed.color, 45
blessed.colors, 47
blessed.formatters, 48
blessed.keyboard, 52
blessed.sequences, 54
blessed.terminal, 57
Symbols

_CURSES_KEYCODE_ADDINS (in module blessed.keyboard), 53
_CUR_TERM (in module blessed.terminal), 66
__call__() (FormattingOtherString method), 50
__call__() (FormattingString method), 49
__call__() (NullCallableString method), 50
__call__() (ParameterizingProxyString method), 49
__call__() (ParameterizingString method), 48
__getattr__() (Terminal method), 57
__new__() (Keystroke static method), 52
_alternative_left_right() (in module blessed.keyboard), 53
_handle_long_word() (SequenceTextWrapper method), 56
_make_colors() (in module blessed.formatters), 48
_wrap_chunks() (SequenceTextWrapper method), 56

B
blessed.color module, 45
blessed.colorspace module, 47
blessed.formatters module, 48
blessed.keyboard module, 52
blessed.sequences module, 54
blessed.terminal module, 57

C
cbreak() (Terminal method), 65
center() (Sequence method), 54
center() (Terminal method), 62
code() (Keystroke property), 52
color() (Terminal property), 60
color_distance_algorithm() (Terminal property), 62
color_rgb() (Terminal method), 60
COLORS (in module blessed.formatters), 48

D
DEFAULT_SEQUENCE_MIXIN (in module blessed.keyboard), 53
dist_cie2000() (in module blessed.color), 47
dist_cie76() (in module blessed.color), 46
dist_cie94() (in module blessed.color), 46
dist_rgb() (in module blessed.color), 46
dist_rgb_weighted() (in module blessed.color), 46
does_styling() (Terminal property), 57

F
formatter() (Terminal method), 61
FormattingOtherString (class in blessed.formatters), 49
FormattingString (class in blessed.formatters), 49
fullscreen() (Terminal method), 59

G
get_keyboard_codes() (in module blessed.keyboard), 52
get_keyboard_sequences() (in module blessed.keyboard), 52
get_location() (Terminal method), 58
get_proxy_string() (in module blessed.formatters), 50
getch() (Terminal method), 64

H
height() (Terminal property), 58
hidden_cursor() (Terminal method), 59

I
inkey() (Terminal method), 65
is_a_tty() (Terminal property), 58
is_sequence() (Keystroke property), 52
iter_parse() (in module blessed.sequences), 56
K
  kbhit() (Terminal method), 64
  keypad() (Terminal method), 65
  Keystroke (class in blessed.keyboard), 52
  kind() (Terminal property), 57

L
  length() (Sequence method), 54
  length() (Terminal method), 63
  link() (Terminal method), 61
  ljust() (Sequence method), 54
  ljust() (Terminal method), 62
  location() (Terminal method), 58
  lstrip() (Sequence method), 55
  lstrip() (Terminal method), 63

M
  measure_length() (in module blessed.sequences), 56
  module
    blessed.color, 45
    blessed.colorspace, 47
    blessed.formatters, 48
    blessed.keyboard, 52
    blessed.sequences, 54
    blessed.terminal, 57
  move_down() (Terminal property), 60
  move_left() (Terminal property), 60
  move_right() (Terminal property), 60
  move_up() (Terminal property), 60
  move_xy() (Terminal method), 59
  move_yx() (Terminal method), 60

N
  name() (Keystroke property), 52
  normal() (Terminal property), 61
  NullCallableString (class in blessed.formatters), 50
  number_of_colors() (Terminal property), 62

O
  on_color() (Terminal property), 60
  on_color_rgb() (Terminal method), 60

P
  padd() (Sequence method), 55
  ParameterizingProxyString (class blessed.formatters), 48
  ParameterizingString (class blessed.formatters), 48
  pixel_height() (Terminal property), 58
  pixel_width() (Terminal property), 58

R
  raw() (Terminal method), 65
  resolve_attribute() (in module blessed.formatters), 51
  resolve_capability() (in module blessed.formatters), 51
  resolve_color() (in module blessed.formatters), 51
  RGB_256TABLE (in module blessed.colorspace), 47
  rgb_downconvert() (Terminal method), 61
  rgb_to_lab() (in module blessed.color), 45
  rgb_to_xyz() (in module blessed.color), 45
  RGBColor (class in blessed.colorspace), 47
  rjust() (Sequence method), 54
  rjust() (Terminal method), 62
  rstrip() (Sequence method), 55
  rstrip() (Terminal method), 63

S
  Sequence (class in blessed.sequences), 54
  SequenceTextWrapper (class in blessed.sequences), 55
  split_compound() (in module blessed.formatters), 55
  split_seqs() (Terminal method), 60
  stream() (Terminal property), 62
  strip() (Sequence method), 55
  strip() (Terminal method), 63
  strip_seqs() (Sequence method), 55
  strip_seqs() (Terminal method), 63

T
  Terminal (class in blessed.terminal), 57

U
  ungetch() (Terminal method), 64

W
  width() (Terminal property), 58
  WINSZ (class in blessed.terminal), 66
  wrap() (Terminal method), 64
  ws_col (WINSZ attribute), 66
  ws_row (WINSZ attribute), 66
  ws_xpixel (WINSZ attribute), 66
  ws_ypixel (WINSZ attribute), 66

X
  in X11_COLORNAMES_TO_RGB (in module blessed.colorspace), 47
  in xyz_to_lab() (in module blessed.color), 45