CHAPTER

TWO

CONSOLE API

For complete control over terminal formatting, Rich offers a *Console* class. Most applications will require a single Console instance, so you may want to create one at the module level or as an attribute of your top-level object. For example, you could add a file called "console.py" to your project:

```python
from rich.console import Console
console = Console()
```

Then you can import the console from anywhere in your project like this:

```python
from my_project.console import console
```

The console object handles the mechanics of generating ANSI escape sequences for color and style. It will auto-detect the capabilities of the terminal and convert colors if necessary.

2.1 Attributes

The console will auto-detect a number of properties required when rendering.

- *size* is the current dimensions of the terminal (which may change if you resize the window).
- *encoding* is the default encoding (typically "utf-8").
- *is_terminal* is a boolean that indicates if the Console instance is writing to a terminal or not.
- *color_system* is a string containing the Console color system (see below).

2.2 Color systems

There are several "standards" for writing color to the terminal which are not all universally supported. Rich will auto-detect the appropriate color system, or you can set it manually by supplying a value for `color_system` to the *Console* constructor.

You can set `color_system` to one of the following values:

- `None` Disables color entirely.
- `"auto"` Will auto-detect the color system.
- `"standard"` Can display 8 colors, with normal and bright variations, for 16 colors in total.
- `"256"` Can display the 16 colors from "standard" plus a fixed palette of 240 colors.
- `"truecolor"` Can display 16.7 million colors, which is likely all the colors your monitor can display.
- `"windows"` Can display 8 colors in legacy Windows terminal. New Windows terminal can display "truecolor".