

(continued from previous page)

```
with progress:
    for n in progress.track(range(100)):
        progress.print(n)
        sleep(0.1)
```

16.2.10 Print / log

The `Progress` class will create an internal `Console` object which you can access via `progress.console`. If you print or log to this console, the output will be displayed *above* the progress display. Here's an example:

```
with Progress() as progress:
    task = progress.add_task("twiddling thumbs", total=10)
    for job in range(10):
        progress.console.print(f"Working on job #{job}")
        run_job(job)
    progress.advance(task)
```

If you have another `Console` object you want to use, pass it in to the `Progress` constructor. Here's an example:

```
from my_project import my_console

with Progress(console=my_console) as progress:
    my_console.print("[bold blue]Starting work!")
    do_work(progress)
```

16.2.11 Redirecting stdout / stderr

To avoid breaking the progress display visuals, Rich will redirect `stdout` and `stderr` so that you can use the built-in `print` statement. This feature is enabled by default, but you can disable by setting `redirect_stdout` or `redirect_stderr` to `False`

16.2.12 Customizing

If the `Progress` class doesn't offer exactly what you need in terms of a progress display, you can override the `get_renderables` method. For example, the following class will render a `Panel` around the progress display:

```
from rich.panel import Panel
from rich.progress import Progress

class MyProgress(Progress):
    def get_renderables(self):
        yield Panel(self.make_tasks_table(self.tasks))
```

16.2.13 Reading from a file

Rich provides an easy way to generate a progress bar while reading a file. If you call `open()` it will return a context manager which displays a progress bar while you read. This is particularly useful when you can't easily modify the code that does the reading.

The following example demonstrates how we might show progress when reading a JSON file: