

1.4 Quick Start

The quickest way to get up and running with Rich is to import the alternative `print` function which takes the same arguments as the built-in `print` and may be used as a drop-in replacement. Here's how you would do that:

```
from rich import print
```

You can then print strings or objects to the terminal in the usual way. Rich will do some basic syntax *highlighting* and format data structures to make them easier to read.

Strings may contain *Console Markup* which can be used to insert color and styles in to the output.

The following demonstrates both console markup and pretty formatting of Python objects:

```
>>> print("[italic red]Hello[/italic red] World!", locals())
```

This writes the following output to the terminal (including all the colors and styles):

If you would rather not shadow Python's built-in `print`, you can import `rich.print` as `rprint` (for example):

```
from rich import print as rprint
```

Continue reading to learn about the more advanced features of Rich.

1.5 Rich in the REPL

Rich may be installed in the REPL so that Python data structures are automatically pretty printed with syntax highlighting. Here's how:

```
>>> from rich import pretty
>>> pretty.install()
>>> ["Rich and pretty", True]
```

You can also use this feature to try out Rich *renderables*. Here's an example:

```
>>> from rich.panel import Panel
>>> Panel.fit("[bold yellow]Hi, I'm a Panel", border_style="red")
```

Read on to learn more about Rich renderables.

1.5.1 IPython Extension

Rich also includes an IPython extension that will do this same pretty install + pretty tracebacks. Here's how to load it:

```
In [1]: %load_ext rich
```

You can also have it load by default by adding "`rich`" to the `c.InteractiveShellApp.extension` variable in IPython Configuration.

1.6 Rich Inspect

Rich has an `inspect()` function which can generate a report on any Python object. It is a fantastic debug aid, and a good example of the output that Rich can generate. Here is a simple example: