You can also use `for` expressions to output a map rather than a list using the following syntax:

```
# Loop over a list and output a map
{for <ITEM> in <LIST> : <OUTPUT_KEY> => <OUTPUT_VALUE>}

# Loop over a map and output a map
{for <KEY>, <VALUE> in <MAP> : <OUTPUT_KEY> => <OUTPUT_VALUE>}
```

The only differences are that (a) you wrap the expression in curly braces rather than square brackets, and (b) rather than outputting a single value each iteration, you output a key and value, separated by an arrow. For example, here is how you can transform a map to make all the keys and values uppercase:

```
output "upper_roles" {
  value = {for name, role in var.hero_thousand_faces :
upper(name) => upper(role)}
}
```

Here's the output from running this code:

```
upper_roles = {
  "MORPHEUS" = "MENTOR"
  "NEO" = "HERO"
  "TRINITY" = "LOVE INTEREST"
}
```

Loops with the for String Directive

Earlier in the book, you learned about string interpolations, which allow you to reference Terraform code within strings:

```
"Hello, ${var.name}"
```

String directives allow you to use control statements (e.g., for-loops and if-statements) within strings using a syntax similar to string interpolations, but

instead of a dollar sign and curly braces (`${...}`), you use a percent sign and curly braces (`%{...}`).

Terraform supports two types of string directives: for-loops and conditionals. In this section, we'll go over for-loops; we'll come back to conditionals later in the chapter. The `for` string directive uses the following syntax:

```
%{ for <ITEM> in <COLLECTION> }<BODY>%{ endfor }
```

where `COLLECTION` is a list or map to loop over, `ITEM` is the local variable name to assign to each item in `COLLECTION`, and `BODY` is what to render each iteration (which can reference `ITEM`). Here's an example:

```
variable "names" {
  description = "Names to render"
  type        = list(string)
  default     = ["neo", "trinity", "morpheus"]
}

output "for_directive" {
  value = "%{ for name in var.names }${name}, %{ endfor }"
}
```

When you run `terraform apply`, you get the following output:

```
$ terraform apply

(...)

Outputs:

for_directive = "neo, trinity, morpheus, "
```

There's also a version of the `for` string directive syntax that gives you the index in the for-loop:

```
%{ for <INDEX>, <ITEM> in <COLLECTION> }<BODY>%{ endfor }
```