for 표현식을 이용한 반복문

names의 리스트를 가진 테라폼 코드

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

파이썬과 같은 범용 프로그래밍 언어에서 모든 이름을 대문자로 변환하는 방법

```
upper_case_names = []
for name in names:
   upper_case_names.append(name.upper())

print upper_case_names

# ['NEO', 'TRINITY', 'MORPHEUS']
```

리스트 내포를 이용한 방법

```
names = ["neo", "trinity", "morpheus"]
upper_case_names = [name.upper() for name in names]

print upper_case_names

# ['NEO', 'TRINITY', 'MORPHEUS']
```

파이썬에서는 조건을 지정하여 결과 목록을 필터링할 수도 있다.

```
names = ["neo", "trinity", "morpheus"]

short_upper_case_names = [names.upper() for name in names if len(name) < 5]

print upper_case_names
```

```
# ['NEO']
```

테라폼의 for 표현식

```
[for <ITEM> in <LIST> : <OUTPUT>]
```

LIST : 반복할 리스트

ITEM : LIST의 각 항목에 할당할 로컬 변수의 이름

OUTPUT : ITEM을 어떤 식으로든 변환하는 표현식

var.names의 이름 목록을 대문자로 변환하는 테라폼 코드

```
provider "local" {
}

variable "names" {
  description = "A list of names"
  type        = list(string)
  default     = ["neo", "trinity", "morpheus"]
}

output "upper_names" {
  value = [ for name in var.names : upper(name) ]
}
```

실행

```
$ terraform apply
...
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

upper_names = [
  "NEO",
  "TRINITY",
  "MORPHEUS",
]
```

파이썬의 리스트 내포 구문과 마찬가지로 조건을 지정하여 결과 리스트를 필터링할 수 있다.

```
provider "local" {
}

variable "names" {
  description = "A list of names"
  type        = list(string)
  default     = ["neo", "trinity", "morpheus"]
}

output "short_upper_names" {
  value = [ for name in var.names : upper(name) if length(name) < 5]
}
```

```
$ terraform apply
...
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

upper_names = [
  "NEO",
]
```

다음과 같은 구문을 사용하여 맵을 반복할 수 있다.

```
[ for <KEY>, <VALUE> in <MAP> : <OUTPUT> ]
```

MAP : 반복되는 맵

key, value : MAP의 키-값 쌍에 할당할 로컬 변수

OUTPUT : KEY와 VALUE를 어떤 식으로든 변환하는 표현식

```
provider "local" {
}

variable "hero_thousand_faces" {
  description = "map"
```

```
  type        = map(string)
  default     = {
    neo       = "hero"
    trinity   = "love interest"
    morpheus  = "mentor"
  }
}

output "bios" {
  value = [for name, role in var.hero_thousand_faces : "${name} is the ${role}"]
}
```

```
$ terraform apply
...
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

bios = [
  "morpheus is the mentor",
  "neo is the hero",
  "trinity is the love interest",
]
```

for 표현식을 리스트가 아닌 맵을 출력하는데 사용할 수도 있다.

```
# 리스트를 반복하고 맵을 출력
[for <ITEM> in <LIST> : <OUTPUT_KEY> => <OUTPUT_VALUE>]

# 맵을 반복하고 리스트를 출력
{for <KEY>, <VALUE> in <MAP> : <OUTPUT_KEY> => <OUTPUT_VALUE>}
```

식을 대괄호 대신 중괄호로 묶고 각 반복마다 단일 값을 출력하는 대신 키와 값을 화살표로 구분하여 출력하는 것이 유일한 차이점이다. 예를 들어 다음은 맵을 변환하여 모든 키와 값을 대문자로 만드는 방법이다.

```
provider "local" {
}

variable "hero_thousand_faces" {
  description = "map"
  type        = map(string)
```

```
  default     = {
    neo       = "hero"
    trinity   = "love interest"
    morpheus  = "mentor"
  }
}

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

```
$ terraform apply
...
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

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