For loops are pretty much the only looping construct that you will need in R. While you may occasionally find a need for other types of loops, in most data analysis situations, there are very few cases where a for loop isn't sufficient.

In R, for loops take an iterator variable and assign it successive values from a sequence or vector. For loops are most commonly used for iterating over the elements of an object (list, vector, etc.)

```
numbers <- rnorm(10)
for(i in 1:10) {
        print(numbers[i])
}
[1] -0.9567815
[1] 1.347491
[1] -0.03158058
[1] 0.5960358
[1] 1.133312
[1] -0.7085361
[1] 1.525453
[1] 1.114152
[1] -0.1214943
[1] -0.2898258

```

This loop takes the i variable and in each iteration of the loop gives it values 1, 2, 3, ..., 10, executes the code within the curly braces, and then the loop exits.

The following three loops all have the same behavior.

```
x <- c("a", "b", "c", "d")

for(i in 1:4) {
        ## Print out each element of 'x'
        print(x[i])
}
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

The seq_along() function is commonly used in conjunction with for loops in order to generate an integer sequence based on the length of an object (in this case, the object x).

```
## Generate a sequence based on length of 'x'
for(i in seq_along(x)) {
        print(x[i])
}
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

It is not necessary to use an index-type variable.

```
for(letter in x) {
        print(letter)
}
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

For one line loops, the curly braces are not strictly necessary.

```
for(i in 1:4) print(x[i])
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

However, curly braces are sometimes useful even for one-line loops, because that way if you decide to expand the loop to multiple lines, you won't be burned because you forgot to add curly braces (and you *will* be burned by this).

Mark as completed