Introduction to For-Loops

Jose Toledo Luna

2024-07-21

Table of contents

A for-loop serves the purpose of cycling through a collection of objects, such as a vector, list, matrix, or dataframe, and consistently applying a specific set of operations to each element within the data structure

The syntax of a for-loop in R consists of a `variable` which takes items from the `iterable` one by one, where the `iterable` is the collection of objects provided *(vector, list, matrix, etc..)*

Lastly, inside the for-loop within the curly braces `{ }` is the `loop body` which are statements that are executed once for each item in the `iterable` provided

```
for(variable in iterable) {
  loop body
}
```

Utilizing for-loops helps maintain code cleanliness and prevents unnecessary duplication of code blocks

To start with a basic example, consider printing the numbers from `1 to 5` inclusive, this is our `iterable` and is constructed using any sequence operator. This can be `a:b` or the built-in function `seq()`

```r
for (index in 1:5){
  print(index)
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

Here `index` is our `variable`, the variable name can be anything but it is usually in the context of the problem

For example, this can be anything from `car`,`letters`, `months`, etc.., in most cases usually the variable name `i` suffices

Our `loop body` is simply to `print()` the current index

Using a For-Loop on a vector

In the next example, we will print the numbers from `1 to 5` and then double each number before printing it

```r
for(i in 1:5){
  print(i*2)
}
```

```
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
```

However, there are scenarios where we not only want to print these values but also store them after each iteration. To do that, we need to start by creating an empty object. This object can be new or an existing one, but it's important to keep in mind that after each iteration, the object may be modified

The function `numeric()` creates a numeric vector of all `0` of a specified `length`. Here the `length`, should be the the same size as the `iterable`

```r
vec <- numeric(length = 5)
```

```r
vec
```

```
[1] 0 0 0 0 0
```

```r
for(i in 1:5){
  print(paste0('Current Iteration: ',i) )    # (1)

  vec[i] <- (i*2)                            # (2)

  print(vec)                                 # (3)
}
```

(1) Print the current iteration *(optional)*
(2) Update the ith element of the vector `vec` by doubling the current index
(3) Print the updated vector `vec` *(optional)*

```
[1] "Current Iteration: 1"
[1] 2 0 0 0 0
[1] "Current Iteration: 2"
[1] 2 4 0 0 0
[1] "Current Iteration: 3"
[1] 2 4 6 0 0
[1] "Current Iteration: 4"
[1] 2 4 6 8 0
[1] "Current Iteration: 5"
[1]  2  4  6  8 10
```

Here is the resulting vector

```r
vec
```

```
[1]  2  4  6  8 10
```

> ⚠ Warning
>
> In R, for-loops tend to be slow. To mitigate the performance issues associated with for-loops, it is often recommended to use vectorized operations or apply functions

For example,

```r
vec <- 2*(1:5)
vec
```

```
[1]  2  4  6  8 10
```

It is clear there was no need to use a for-loop in the previous example it was simply for teaching purposes

We don't need to iterate sequentially from a regular sequence `1:N`. We can iterate through the elements of an existing object. For example,

```r
for(i in 3:6){
  print(i)
}
```

```
[1] 3
[1] 4
[1] 5
[1] 6
```

```r
for(i in seq(from = 1, to = 11, by = 2)){
  print(i)
}
```

```
[1] 1
[1] 3
[1] 5
[1] 7
[1] 9
[1] 11
```

```r
for(pet in c('dog','cat','hamster','turtle')){
  print(pet)
}
```

```
[1] "dog"
[1] "cat"
[1] "hamster"
[1] "turtle"
```

However, if you are trying to update the ith element of an object this approach might not be the best

```r
vec <- numeric(length=3)

for(index in c(2,4,6) ){
  print( vec[index] )
}
```

```
[1] 0
[1] NA
[1] NA
```

An `NA` is produced because our vector `vec` has three elements and we are trying to access the 4th and 6th element which do not exist

Instead use the function `seq_along(x)` which will create a regular sequence from `1:length(x)`

```r
seq_along(c(2,4,6))
```

```
[1] 1 2 3
```

```r
seq_along(c('dog','cat','hamster','turtle'))
```

```
[1] 1 2 3 4
```

```r
for(index in seq_along(c(2,4,6)) ){
  print(index)
}
```

```
[1] 1
[1] 2
[1] 3
```