Loops

Luna L Sanchez Reyes

2023-04-06

For loops are another way that we can tell a computer to repeat tasks for us. They are versatile and very explicit, so that means that we are controlling everything that is run on each iteration of the loop (mostly). As ooposed to apply functions, where the iterations happen kind of under the hood, and the apply functions can only be used to loop over (iterate) on one function.

Loops can let us iterate over multiple functions and whole blocks of code.

The general structure of a for loop

The general syntax to run a for loop is as follows:

```
for (variable_used_inside_the_loop in object_with_values) {
  do something with(variable_used_inside_the_loop)
}
```

An example:

```
lengths <- c(13.3, 15, 100)

for (value in 1:3) {
  mass <- 0.73 * value^2
  print(mass)
  # we can't use return() in for loops
}
```

```
## [1] 0.73
## [1] 2.92
## [1] 6.57
```

Exercise 1.

The code below prints the numbers 1 through 5 one line at a time. Modify it to print each of these numbers multiplied by 3.

```
numbers <- c(1, 2, 3, 4, 5)
for (number in numbers){
  print(number * 3 /100)
}
```

```
## [1] 0.03
## [1] 0.06
## [1] 0.09
## [1] 0.12
## [1] 0.15
```

2. Write a for loop that loops over the following vector and prints out the mass in kilograms (mass_kg = 2.2 * mass_lb)

```r
mass_lbs <- c(2.2, 3.5, 9.6, 1.2)

for (mass_lbs in mass_lbs) {
  mass_kgs <- 2.2 *mass_lbs
  print(mass_kgs)
}
```

```
## [1] 4.84
## [1] 7.7
## [1] 21.12
## [1] 2.64
```

Looping over using an index

What is an index in R? It is the numeric position of values inside any data structure in R. For example in the following vector

```r
flowers <- c("lilacs", "daisies", "jasmins")
str(flowers)
```

```
##  chr [1:3] "lilacs" "daisies" "jasmins"
```

```r
# To access the second element in the vector, we need to use the number 2 as index inside the square br
flowers[2]
```

```
## [1] "daisies"
```

We can use numbers as indices to loop over values inside a vector.

```r
for (i in 1:3) {
  print(i)
  print(flowers[i])
}
```

```
## [1] 1
## [1] "lilacs"
## [1] 2
## [1] "daisies"
## [1] 3
## [1] "jasmins"
```

Exercise 3.

Complete the code below so that it prints out the name of each bird one line at a time.

```r
birds = c('robin', 'woodpecker', 'blue jay', 'sparrow', "chicken")
for (j in 1:length(birds)){
  print(birds[j])
}
```

```
## [1] "robin"
## [1] "woodpecker"
## [1] "blue jay"
## [1] "sparrow"
## [1] "chicken"
```

Storing results from a for loop using indices

So far we have just printed some values and results from some equations.

Usually what we need is to save the results of running a for loop, so that we can use them later.

When we are using a function what do we do to store the results of the function? We assign the result to a variable name:

```
my_results <- 0.73 * lengths^2
```

But in for loops we do not have that option. We can't do:

```
my_result <- for (variable in vector) {

}
```

The only way to save results from each iteration of the loop is by saving them into an empty object.

Run the following loop:

```
for (i in 1:length(flowers)) {
  upper <- toupper(flowers[i])
  print(upper)
}
```

```
## [1] "LILACS"
## [1] "DAISIES"
## [1] "JASMINS"
```

To store the output of running the function `toupper()` we need to create an empty vector. To create an empty vector, we use the function `vector()`.

```
my_results <- vector(mode = "character", length = length(flowers))
my_results
```

```
## [1] "" "" ""
```

Now we can use this empty vector and indices inside a loop to store results:

```
for (i in 1:length(flowers)) {
  upper <- toupper(flowers[i])
  my_results[i] <- upper
}
my_results
```

```
## [1] "LILACS"  "DAISIES" "JASMINS"
```

Exercise 4.

Complete the code below so that it stores one area for each radius.

```
radius <- c(1.3, 2.1, 3.5)
areas <- vector(mode = "numeric", length = length(radius))
for (whatever in 1:length(radius)){
  areas[whatever] <- pi * radius[whatever] ^ 2
}
areas
```

```
## [1]  5.309292 13.854424 38.484510
```

Looping over multiple object with indices

We have 3 vectors:

```r
dino_names <- c("T-rex", "Ankylosaur", "Triceratops")
# We have different a an dbvalues for each of these dino species
a_values <- c(0.73, 5.4, 100)
b_values <- c(2, 0.5, 1.2)
dino_lengths <- c(15, 3, 20)
dino_masses <- vector(mode = "numeric", length = length(dino_names))
dino_masses
```

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

We can iterate through these values within a loop

```r
#dino_masses <- vector()
for (i in 1:length(dino_names)) {
  print(dino_names[i])
  mass <- a_values[i] * dino_lengths[i]^b_values[i]
  print(mass)
  dino_masses[i] <- mass
}
```

```
## [1] "T-rex"
## [1] 164.25
## [1] "Ankylosaur"
## [1] 9.353074
## [1] "Triceratops"
## [1] 3641.128
```

```r
dino_masses
```

```
## [1]  164.250000    9.353074 3641.128406
```

```r
dino_masses[4] <- NA
dino_masses[100] <- NA
dino_masses
```

```
##   [1]  164.250000    9.353074 3641.128406          NA          NA          NA
##   [7]          NA          NA          NA          NA          NA          NA
##  [13]          NA          NA          NA          NA          NA          NA
##  [19]          NA          NA          NA          NA          NA          NA
##  [25]          NA          NA          NA          NA          NA          NA
##  [31]          NA          NA          NA          NA          NA          NA
##  [37]          NA          NA          NA          NA          NA          NA
##  [43]          NA          NA          NA          NA          NA          NA
##  [49]          NA          NA          NA          NA          NA          NA
##  [55]          NA          NA          NA          NA          NA          NA
##  [61]          NA          NA          NA          NA          NA          NA
##  [67]          NA          NA          NA          NA          NA          NA
##  [73]          NA          NA          NA          NA          NA          NA
##  [79]          NA          NA          NA          NA          NA          NA
##  [85]          NA          NA          NA          NA          NA          NA
##  [91]          NA          NA          NA          NA          NA          NA
##  [97]          NA          NA          NA          NA
```

Exercise 5. Complete the code below to calculate an area for each pair of lengths and widths, store the areas in a vector, and after they are all calculated print them out:

```r
lengths = c(1.1, 2.2, 1.6)
widths = c(3.5, 2.4, 2.8)
areas <- vector(mode = "numeric", length = 3)
areas <- vector(mode = "numeric", length = length(widths))
areas
```

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

```r
for (i in 1:length(lengths)) {
  areas[i] <- lengths[i] * widths[i]
}
areas
```

```
## [1] 3.85 5.28 4.48
```