loops

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 opposed to apply functions, where the iterations happen kind of under the hood, and the apply functions can be 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

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

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

for (length in lengths) {
  mass <- 0.73 * length^2
  print(mass)
  #we can't use return() in for loops
}
```

```
## [1] 129.1297
## [1] 164.25
## [1] 7300
```

Exercise 1

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

```
## [1] 3
## [1] 6
## [1] 9
## [1] 12
## [1] 15
```

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

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

```
## [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 n the following vector.

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

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

```r
#To access the second element in the vector, I need to use the index number 2 inside the square bracket
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] "jasmines"
```

Exercise 3

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

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

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 use 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.

```
# To create an empty vector, we use the function vector()
my_results <- vector(mode = "character", length = length(flowers))
my_results
```

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

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

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

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

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

#Exercise 4

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

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

Looping over multiple object with indices

We have three vectors:

```
dino_names <- c("T-rex", "Ankylosaur", "Triceratops")
# We have different a values 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))
```

We can iterate through these values within a loop

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

```
## [1] "T-rex"
## [1] 0 0 0
## [1] "Ankylosaur"
## [1] 164.25   0.00   0.00
## [1] "Triceratops"
## [1] 164.250000   9.353074   0.000000
```

```
dino_masses
```

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

#Exercise 5

```
lengths = c(1.1, 2.2, 1.6)
widths = c(3.5, 2.4, 2.8)
areas <- vector(mode = "numeric", length = 3)
for (i in 1:length(lengths)) {
  areas[i] <- lengths[i] * widths[i]
}
areas
```

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