# Write a while loop

Let's get you started with building a while loop from the ground up. Have another look at its recipe:

while (condition) {

expr

}

Remember that the condition part of this recipe should become FALSE at some point during the execution. Otherwise, the while loop will go on indefinitely.

If your session expires when you run your code, check the body of your *while* loop carefully.

Have a look at the code on the right; it initializes the speed variables and already provides a while loop template to get you started.

##### Instructions

Code a while loop with the following characteristics:

* The condition of the while loop should check if speed is higher than 30.
* Inside the body of the while loop, print out "Slow down!".
* Inside the body of the while loop, decrease the speed by 7 units and assign this new value to speed again. This step is crucial; otherwise your while loop will never stop and your session will expire.

**If your session expires when you run your code, check the body of your *while* loop carefully: it's likely that you made a mistake.**

# Throw in more conditionals

In the previous exercise, you simulated the interaction between a driver and a driver's assistant: When the speed was too high, "Slow down!" got printed out to the console, resulting in a decrease of your speed by 7 units.

There are several ways in which you could make your driver's assistant more advanced. For example, the assistant could give you different messages based on your speed or provide you with a current speed at a given moment.

A while loop similar to the one you've coded in the previous exercise is already available in the editor. It prints out your current speed, but there's no code that decreases the speed variable yet, which is pretty dangerous. Can you make the appropriate changes?

##### Instructions

* If the speed is greater than 48, have R print out "Slow down big time!", and decrease the speed by 11.
* Otherwise, have R simply print out "Slow down!", and decrease the speed by 6.

If the session keeps timing out and throwing an error, you are probably stuck in an infinite loop! Check the body of your while loop and make sure you are assigning new values to speed.

# Loop over a vector

In the previous video, Filip told you about two different strategies for using the for loop. To refresh your memory, consider the following loops that are equivalent in R:

primes <- c(2, 3, 5, 7, 11, 13)

# loop version 1

for (p in primes) {

print(p)

}

# loop version 2

for (i in 1:length(primes)) {

print(primes[i])

}

Remember our linkedin vector? It's a vector that contains the number of views your LinkedIn profile had in the last seven days. The linkedin vector has already been defined in the editor on the right so that you can fully focus on the instructions!

##### Instructions

Write a for loop that iterates over all the elements of linkedin and prints out every element separately. Do this in two ways: using the loop version 1 and the loop version 2 in the example code above.

# Loop over a list

Looping over a list is just as easy and convenient as looping over a vector. There are again two different approaches here:

primes\_list <- list(2, 3, 5, 7, 11, 13)

# loop version 1

for (p in primes\_list) {

print(p)

}

# loop version 2

for (i in 1:length(primes\_list)) {

print(primes\_list[[i]])

}

Notice that you need double square brackets - [[ ]] - to select the list elements in loop version 2.

Suppose you have a list of all sorts of information on New York City: its population size, the names of the boroughs, and whether it is the capital of the United States. We've already prepared a list nyc with all this information in the editor (source: Wikipedia).

##### Instructions

As in the previous exercise, loop over the nyc list in two different ways to print its elements:

* Loop directly over the nyc list (loop version 1).
* Define a looping index and do subsetting using double brackets (loop version 2).