Loops in Ruby

If there is something computer programmers like to do not do, it is to repeat the same thing several times.

That is why from the very beginning this was kind of the whole point of creating computers. Why calculating this over and over again?? what if we teach a dog how to do it. Well instead of dog training, humans created computers.

It was easy to fall in an infinite loop back in the day and end up unplugging the dog to stop it. Today Ruby makes looping easy and intuitive.

LOOPS: execute the same block of code a specific number of times.

Syntax of some loops

While: executes code while conditional is true.

while

conditional [do]

code

end

---------------------------------

code while condition

----------------------

begin

code

end while conditional

Until: executes code while conditional is false.

until conditional [do]

code

end

-----------------------

code until conditional

-----------------------

begin

code

end until conditional

For: executes code once for each element in the expression.

for variable [, variable ...] in expression [do]

code

end

The challenge here is to define the specific number of times that we want to execute and obviously what code the want to be executed.

Which one to use will depend on how do you like to set the conditional and just execute some code (while, until), if you are going to loop over a Hash or an Array without creating new scope for local variables (for) or if you want to execute related to a sequence of numbers (for).

The statement above is just an interpretation you can experiment with them, but as important it is to understand how this different loops work is to understand what iterator variables and the conditionals are:

Defining and understanding iterators;

An iterator is a variable that will increase or decrees in order to count how many times the code has been repeated.

This variable usually define equal to 0 or 1 outside the loop and increase =+ 1 every time the code is executed. Is not weird to use it counting backwards or counting in a different scale. But the final idea is to use this in the conditional that stops the loop and some times in the code that you want to execute i (iterators) times.

Defining and understanding conditional;

Loops go on until the conditional is true or false. This is why you use the iterator and a conditional when setting up a loop.

On loops like each they stop when they have no more elements to pass into the code so we can say they have an automatic break.

Break, redo, retry, next and retry\_if are statements that you can use inside your loops in order to control its performance giving you some independence from the conditional.