Programming for Everybody

3. Loops & Iterators

le wagon

Why looping?

To repeat an action in Ruby!

Ex: displaying posts on your blog page

While loop

The number of times we'll be looping is **unknown**

Repeats an action in Ruby `while` a certain condition is **true**

Checks to see if said condition is **true**, and `while` it is, the loop keeps running; as soon as the condition stops being **true**, the loop stops

Until loop

The number of times the action will be repeated is also **unknown**

Repeats an action in Ruby while a certain condition is **false**

Checks to see if said condition is **false**, and while it is, the loop keeps running; as soon as the condition becomes **true**, the loop stops

Beware of infinite loops!

```
counter = 1

while counter < 11
      puts counter
      counter = counter + 1  (same as counter +=1)
end
```

If we'd forgotten to **increment the counter,** the loop would have kept checking if 1 is less than 11, therefore always evaluating to **true**.

We would have been stuck in an infinite loop! 😱

For loop

The number of times the action will be repeated is **known**

To repeat an action in Ruby within a certain **range** of elements

1..10 -> a range which includes the numbers from 1 to **10**

1…10 -> a range which includes the numbers from 1 to **9**

Next

Used to skip over certain steps in the loop

```
for number in 1..5
    next if number % 2 == 0
    print number
end
```

(skips printing all the even numbers)

Iterators

Another way to loop in Ruby!

An **iterator** is a Ruby method that repeatedly invokes a `block` of code

That `block` of code is the bit that contains the instructions to be repeated

(and those instructions may be anything you want!) 🙌🏻

Iterators

1. Loop

It’s the simplest iterator of all:

```
loop { print "Hello, world!" }
```

is the same as:

```
loop do
  print "Hello, world!"
end
```

Iterators

1. Loop (cont.)

when using the loop iterator, we need to use “**break"** to break the loop as soon as a certain condition is met

```
number = 0

loop do
      number += 1
      print number
      break if number > 5
end
```

(the loop stops after printing the numbers from 1 to 6)

Iterators

2. Each

a more powerful iterator which can apply an expression to **each element** of a **collection**, one at a time

```
collection_name.each do | item |
    #do something to each item
end
```

the name between | | can be anything -> it’s just a placeholder for each element of the collection you’re calling **.each** on

Iterators

3. Times

Does something a **specified number of times**

```
10.times do
    #do something
end
```

Thank you.