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 a certain 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 a certain 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 it counter was less than 11

and because 1 is always less than 11, the loop would never have ended!

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 that includes the numbers from 1 to **10**

1…10 -> a range that 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

anther 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 can be anything you like!)

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 method 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 operator that 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 applying .each to

Iterators

3. Times

does something to an item a **specified number of times**

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

Thank you! :)

le wagon