Loops and Breaks

by Miriam Tocino

Coding a Spanish Omelette

What They Do

Automate **repetitive tasks**
quickly and easily.

Different Ways

Different Ways

- ruby loops: **until, while, loop**

Different Ways

- ruby loops: **until, while, loop**
- ruby iteration: **each, times, for**

Ruby Loops

until, while, loop

Used to **repeat statements**
until a condition is met.

Ruby Iteration

each, times, for

Used to **loop through values** in Arrays, Hashes, and other Ruby types.

Coding a Spanish Omelette

Ingredients:

Coding a Spanish Omelette

Ingredients:

5 potatoes

Coding a Spanish Omelette

Ingredients:

5 potatoes

6 eggs

Coding a Spanish Omelette

Ingredients:

5 potatoes

6 eggs

1 onion

Coding a Spanish Omelette

Ingredients:

5 potatoes

6 eggs

1 onion

Olive oil

Coding a Spanish Omelette

Ingredients:

5 potatoes

6 eggs

1 onion

Olive oil

Salt

While Loop

Execute, add, repeat

Repeat action **while** certain **condition is true**.
As soon as the **condition stops being true**, the loop stops.

Syntax

```
while conditional
    # Do something
end
```

Syntax

true

```
while conditional
    # Do something
end
```

While Loop

```
$ ruby recipe.rb
```

```
# Scrape the Potatoes
```

While Loop

```
$ ruby recipe.rb
```

```ruby
# Scrape the Potatoes
counter = 1
```

While Loop

```
$ ruby recipe.rb
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6


end
```

While Loop

```
$ ruby recipe.rb
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."

end
```

While Loop

```
$ ruby recipe.rb
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."
  counter = counter + 1
end
```

While Loop

```
$ ruby recipe.rb
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."
  counter = counter + 1
end
```

While Loop

```
$ ruby recipe.rb

Scrape potato 1.
Scrape potato 2.
Scrape potato 3.
Scrape potato 4.
Scrape potato 5.
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."
  counter = counter + 1
end
```

Infinite Loop

```
$ ruby recipe.rb
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."
  counter = counter + 1
end
```

Infinite Loop

```
$ ruby recipe.rb
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."
  counter = counter + 1
end
```

Infinite Loop

```
$ ruby recipe.rb
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."

end
```

Infinite Loop

```
$ ruby recipe.rb
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."

end
```

Infinite Loop

```
$ ruby recipe.rb

Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
…
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."

end
```

Infinite Loop

```
$ ruby recipe.rb

Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
```

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."

end
```

Infinite Loop

```
$ ruby recipe.rb

Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
Scrape potato 1.
```

An infinite loop is
a **loop that never exits**.

```ruby
# Scrape the Potatoes
counter = 1
while counter < 6
  puts "Scrape potato #{counter}."

end
```

Until Loop

Execute, add, repeat

Repeat action **while** certain **condition is false**.
As soon as the **condition is true**, the loop stops.

Syntax

```
until conditional
    # Do something
end
```

Syntax

false

```
until conditional
    # Do something
end
```

Until Loop

```
$ ruby recipe.rb
```

```
# Cut potatoes into thick slices
```

Until Loop

```
$ ruby recipe.rb
```

```ruby
# Cut potatoes into thick slices
counter = 1
```

Until Loop

```
$ ruby recipe.rb
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6


end
```

Until Loop

```
$ ruby recipe.rb
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."

end
```

Until Loop

```
$ ruby recipe.rb
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Until Loop

```
$ ruby recipe.rb
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Until Loop

```
$ ruby recipe.rb

Cut potato 1.
Cut potato 2.
Cut potato 3.
Cut potato 4.
Cut potato 5.
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

There’s always
another way with Ruby.

Assignment Operators

```
# More Assignment Operators
```

```
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```ruby
# More Assignment Operators
counter = counter + 1
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```ruby
# More Assignment Operators
counter = counter + 1
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```ruby
# More Assignment Operators
counter = counter + 1
counter += 1
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```ruby
# More Assignment Operators
counter = counter + 1
counter += 1
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```ruby
# More Assignment Operators
counter = counter + 1
counter += 1

counter = counter - 1
counter -= 1
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```
# More Assignment Operators
counter = counter + 1
counter += 1

counter = counter - 1
counter -= 1
```

```
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```
# More Assignment Operators
counter = counter + 1
counter += 1

counter = counter - 1
counter -= 1

counter = counter * 1
counter *= 1
```

```
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```
# More Assignment Operators
counter = counter + 1
counter += 1

counter = counter - 1
counter -= 1

counter = counter * 1
counter *= 1
```

```
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```
# More Assignment Operators
counter = counter + 1
counter += 1

counter = counter - 1
counter -= 1

counter = counter * 1
counter *= 1

counter = counter / 1
counter /= 1
```

```
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```
# More Assignment Operators
counter = counter + 1
counter += 1

counter = counter - 1
counter -= 1

counter = counter * 1
counter *= 1

counter = counter / 1
counter /= 1
```

```
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```
# More Assignment Operators
counter = counter + 1
counter += 1

counter = counter - 1
counter -= 1

counter = counter * 1
counter *= 1

counter = counter / 1
counter /= 1
```

```
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter = counter + 1
end
```

Assignment Operators

```
# More Assignment Operators
counter = counter + 1
counter += 1

counter = counter - 1
counter -= 1

counter = counter * 1
counter *= 1

counter = counter / 1
counter /= 1
```

```
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."

end
```

Assignment Operators

```ruby
# More Assignment Operators
counter = counter + 1
counter += 1

counter = counter - 1
counter -= 1

counter = counter * 1
counter *= 1

counter = counter / 1
counter /= 1
```

```ruby
# Cut potatoes into thick slices
counter = 1
until counter == 6
  puts "Cut potato #{counter}."
  counter += 1
end
```

Keep Cooking

Keep Cooking

```
puts "Chop the onion."
puts "Heat the oil in a large frying pan."
```

For Loop

Specified number of times

Whenever **you know**
how many times you will be looping.

Syntax

```
for variable in range
    # Do something
end
```

For Loop

```
$ ruby recipe.rb
```

```
# Add potatoes and onion to the pan
```

For Loop

```
$ ruby recipe.rb
```

```ruby
# Add potatoes and onion to the pan
for counter in 1...6
```

For Loop

```
$ ruby recipe.rb
```

```ruby
# Add potatoes and onion to the pan
for counter in 1...6


end
```

For Loop

```
$ ruby recipe.rb
```

```ruby
# Add potatoes and onion to the pan
for counter in 1...6
  puts "Add potato #{counter}
       to the pan."
end
```

For Loop

```
$ ruby recipe.rb
```

```ruby
# Add potatoes and onion to the pan
for counter in 1...6
  puts "Add potato #{counter}
       to the pan."
end

puts "Add onion to the pan."
```

For Loop

```
$ ruby recipe.rb
```

```ruby
# Add potatoes and onion to the pan
for counter in 1...6
  puts "Add potato #{counter}
       to the pan."
end

puts "Add onion to the pan."
```

For Loop

```
$ ruby recipe.rb

Add potato 1 to the pan.
Add potato 2 to the pan.
Add potato 3 to the pan.
Add potato 4 to the pan.
Add potato 5 to the pan.

Add onion to the pan.
```

```ruby
# Add potatoes and onion to the pan
for counter in 1...6
  puts "Add potato #{counter}
       to the pan."
end

puts "Add onion to the pan."
```

For Loop

```
$ ruby recipe.rb

Add potato 1 to the pan.
Add potato 2 to the pan.
Add potato 3 to the pan.
Add potato 4 to the pan.
Add potato 5 to the pan.

Add onion to the pan.
```

```ruby
# Add potatoes and onion to the pan
for counter in 1...6
  puts "Add potato #{counter}
       to the pan."
end

puts "Add onion to the pan."
```

For Loop

```
$ ruby recipe.rb

Add potato 1 to the pan.
Add potato 2 to the pan.
Add potato 3 to the pan.
Add potato 4 to the pan.
Add potato 5 to the pan.

Add onion to the pan.
```

```ruby
# Add potatoes and onion to the pan

  puts "Add potato #{counter}
       to the pan."
end

puts "Add onion to the pan."
```

For Loop

```
$ ruby recipe.rb

Add potato 1 to the pan.
Add potato 2 to the pan.
Add potato 3 to the pan.
Add potato 4 to the pan.
Add potato 5 to the pan.

Add onion to the pan.
```

```ruby
# Add potatoes and onion to the pan
for counter in 1..5
  puts "Add potato #{counter}
       to the pan."
end

puts "Add onion to the pan."
```

Loop Method

Repeat and break

It will run the statements
inside the loop **until some condition is met**.

Syntax

```
loop do
    # Do something
    break if conditional
end
```

Syntax

```
loop do
  # Do something
  break if conditional
end
```

true

The Loop Method

```
$ ruby recipe.rb
```

```
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do



end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1


end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  puts "#{minutes} minutes left."

end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  puts "#{minutes} minutes left."

end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  puts "#{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  puts "#{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb

29 min left.
28 min left.
27 min left.
26 min left.
25 min left.
24 min left.
23 min left.
22 min left.
21 min left.
20 min left.
19 min left.
18 min left.
…
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  puts "#{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  puts "#{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1




  puts "#{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1




  puts "#{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1

  puts "#{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  if minutes % 5 == 0
    puts "It's time to stir!"
  end
  puts "#{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  if minutes % 5 == 0
    puts "It's time to stir!"
  end
  puts "#{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb

29 minutes left.
28 minutes left.
27 minutes left.
26 minutes left.
25 minutes left.
It's time to stir!
24 minutes left.
23 minutes left.
22 minutes left.
21 minutes left.
20 minutes left.
It's time to stir!
…
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  if minutes % 5 == 0
    puts "It's time to stir!"
  end
  puts "#{minutes} minutes left."
  break if minutes == 0
end
```

Next

Sometimes you might want to
skip over certain steps in the loop.

Syntax

```
loop do
  next if conditional
  # Do something
  break if conditional
end
```

Syntax

```
loop do
  next if conditional
  # Do something
  break if conditional
end
```

true

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  puts "It’s time to stir.
        #{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1

  puts "It’s time to stir.
        #{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1

  puts "It’s time to stir.
        #{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  next if minutes % 5 != 0
  puts "It’s time to stir.
        #{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  next if minutes % 5 != 0
  puts "It’s time to stir.
        #{minutes} minutes left."
  break if minutes == 0
end
```

The Loop Method

```
$ ruby recipe.rb

25 min left.
It’s time to stir.
20 min left.
It’s time to stir.
15 min left.
It’s time to stir.
10 min left.
It’s time to stir.
5 min left.
It’s time to stir.
0 min left.
```

```ruby
# Stew gently for 30 minutes,
# stirring every 5 min
# until the potatoes are softened.
minutes = 30
loop do
  minutes -= 1
  next if minutes % 5 != 0
  puts "It’s time to stir.
        #{minutes} minutes left."
  break if minutes == 0
end
```

Keep Cooking

Keep Cooking

```
puts "Strain potatoes and onions through a colander into a large bowl."
```

Each Method

Powerful iterator

Used to **iterate over a collection of items**,
like an array, one at a time.

Syntax

```
mycollection.each do |item|
  # Do something
end
```

Syntax

```
mycollection.each do |item|
  # Do something
end

mycollection.each { |item| # Do something }
```

Each Operator

```
$ ruby recipe.rb
```

```
# Break the eggs
and beat them separately.
```

Each Operator

```
$ ruby recipe.rb
```

```
# Break the eggs
and beat them separately.
eggs = [1, 2, 3, 4, 5, 6]
```

Each Operator

```
$ ruby recipe.rb
```

```ruby
# Break the eggs
and beat them separately.
eggs = [1, 2, 3, 4, 5, 6]
eggs.each do |egg|

end
```

Each Operator

```
$ ruby recipe.rb
```

```ruby
# Break the eggs
and beat them separately.
eggs = [1, 2, 3, 4, 5, 6]
eggs.each do |egg|
  puts "Break egg #{egg}."
end
```

Each Operator

```
$ ruby recipe.rb
```

```ruby
# Break the eggs
and beat them separately.
eggs = [1, 2, 3, 4, 5, 6]
eggs.each do |egg|
  puts "Break egg #{egg}."
end

puts "Beat the eggs separately."
```

Each Operator

```
$ ruby recipe.rb
```

```ruby
# Break the eggs
and beat them separately.
eggs = [1, 2, 3, 4, 5, 6]
eggs.each do |egg|
  puts "Break egg #{egg}."
end

puts "Beat the eggs separately."
```

Each Operator

```
$ ruby recipe.rb

Break egg 1.
Break egg 2.
Break egg 3.
Break egg 4.
Break egg 5.
Break egg 6.

Beat the eggs separately.
```

```ruby
# Break the eggs
and beat them separately.
eggs = [1, 2, 3, 4, 5, 6]
eggs.each do |egg|
  puts "Break egg #{egg}."
end

puts "Beat the eggs separately."
```

Keep Cooking

LISTO!

Keep Cooking

```
puts "Then stir into the bowl the potatoes with plenty of salt."
puts "Heat a little of the strained oil in a smaller pan."
puts "Tip everything into the pan and cook on a moderate heat."
puts "When almost set, invert on a plate and slide back into the pan."
puts "Cook a few more minutes."
puts "Slide on to a plate and cool for 10 minutes before serving."
```

Times Iterator

Compact *for* loop

It **takes a number** and execute the statements in the block **that number of times**.

Syntax

```
number.times do
  # Do something
end
```

For Loop

```
$ ruby recipe.rb
```

```
# How do you like the recipe?
```

For Loop

```
$ ruby recipe.rb
```

```ruby
# How do you like the recipe?
3.times do

end
```

For Loop

```
$ ruby recipe.rb
```

```ruby
# How do you like the recipe?
3.times do
  puts "It is very tasty!"
end
```

For Loop

```
$ ruby recipe.rb
```

```ruby
# How do you like the recipe?
3.times do
  puts "It is very tasty!"
end
```

For Loop

```
$ ruby recipe.rb

It is very tasty!
It is very tasty!
It is very tasty!
```

```ruby
# How do you like the recipe?
3.times do
  puts "It is very tasty!"
end
```

For Loop

```
$ ruby recipe.rb

It is very tasty!
It is very tasty!
It is very tasty!
```

```ruby
# How do you like the recipe?
3.times do
  puts "It is very tasty!"
end
```

```ruby
3.times { puts "It is very tasty!" }
```

Final Review

Final Review

```
while conditional
    # Do something
end
```

Final Review

```
while conditional
    # Do something
end
```

```
until conditional
    # Do something
end
```

Final Review

```
while conditional
    # Do something
end
```

```
until conditional
    # Do something
end
```

```
for variable in range
    # Do something
end
```

Final Review

```
while conditional
    # Do something
end
```

```
until conditional
    # Do something
end
```

```
for variable in range
    # Do something
end
```

```
loop do
    next if conditional
    # Do something
    break if conditional
end
```

Final Review

```
while conditional
    # Do something
end
```

```
until conditional
    # Do something
end
```

```
for variable in range
    # Do something
end
```

```
loop do
    next if conditional
    # Do something
    break if conditional
end
```

```
mycollection.each do |item|
    # Do something
end
```

Final Review

```
while conditional
    # Do something
end
```

```
until conditional
    # Do something
end
```

```
for variable in range
    # Do something
end
```

```
loop do
    next if conditional
    # Do something
    break if conditional
end
```

```
mycollection.each do |item|
    # Do something
end
```

```
number.times do
    # Do something
end
```

Thank you!

It’s time for questions

Check here the **recipe.rb** file:

https://github.com/Codaisseur/workshops