

squares = [red, blue, orange, purple, red]

squares.size => 5 (There are 5 of them!)

each Does something (or nothing!) to each item, one at a time.

In our case, squares.

This "loop" will happen 5 times. Because there are 5 items in our squares collection.

This variable name can be called anything. frankenstein. x. We called it square, because we have a collection of squares. It's the temporary name for the ITEM that's currently being passed to each.

These are the same
squares.each do |square|
 you can now do something to the square.
end
[red, blue, orange, purple, red].each do |square|
 you can now do something to the square.
end

1. [red, blue, orange, purple, red].each do |square|
 red ==> this is the value of square on the first time
end

2. [red, blue, orange, purple, red].each do |square|
 blue ==> this is the value of square on the second time
end

3. [red, blue, orange, purple, red].each do |square|
 orange ==> this is the value of square on the third time
end

4. [red, blue, orange, purple, red].each do |square|
 purple ==> this is the value of square on the fourth time
end

5. [red, blue, orange, purple, red].each do |square|
 red ==> this is the value of square on the fifth time (Yup, we had two green squares in this collection).
end

select. Looks through a collection and SELECTS a sub-set of that collection, based on WHATEVER YOU WANT!

Let's find all the red squares.

Why? Because red is the color of money, so if we find all the red squares, we can pretend we are RICH!

Let's call our red squares: green\_squares.

This starts as an empty collection, but when we're done it will be full of green squares

select happens for EACH item in the array. Hey! It's like EACH. But it gives you something back.

green\_squares = squares.select do |square|
 square.color == "green"
end

This thing? It's our CONDITION. Let's pretend you can ask a square for its color. We have decided that if you ask a square for its color, and it says "green", we want to save it in green\_squares. If it doesn't say "green", well, forget it.

Remember! This variable can be named anything we want.

1. green\_squares = [red, blue, orange, purple, red].select do |square|
 red.color == "green" (=> TRUE)
end
This square goes into green\_squares!

2. green\_squares = [red, blue, orange, purple, red].select do |square|
 blue.color == "green" (=> FALSE)
end
This square does not go into green\_squares!

3. green\_squares = [red, blue, orange, purple, red].select do |square|
 orange.color == "green" (=> FALSE)
end
This square does not go into green\_squares!

4. green\_squares = [red, blue, orange, purple, red].select do |square|
 purple.color == "green" (=> FALSE)
end
This square does not go into green\_squares!

5. green\_squares = [red, blue, orange, purple, red].select do |square|
 red.color == "green" (=> TRUE)
end
ANOTHER GREEN SQUARE OMG!

So. At the end of select we have 2 collections!

squares = [red, blue, orange, purple, red] What we started with.

AND

green\_squares = [red, red] Our collection of pretend money.

