2019年2月16日

Ruby初級者向けレッスン 69回
― ブロック ―

ひがき @ Ruby関西

お品書き

- ブロックとは?
  - 繰り返し
  - Enumerable
- メソッドにブロックを渡す
- ブロックで値を受け取る
- メソッドでブロックを受け取る
- ブロックに値を渡す

ブロックとは?

- 一連の処理をひとまとめにしたもの

```
f = open('hello.txt')
f.read
    :
f.close
```

- メソッドと何が違うの?

繰り返し

```
a = ['a', 'b', 'c']
a.each do |i|
  puts i
end

# >> a
# >> b
# >> c
```

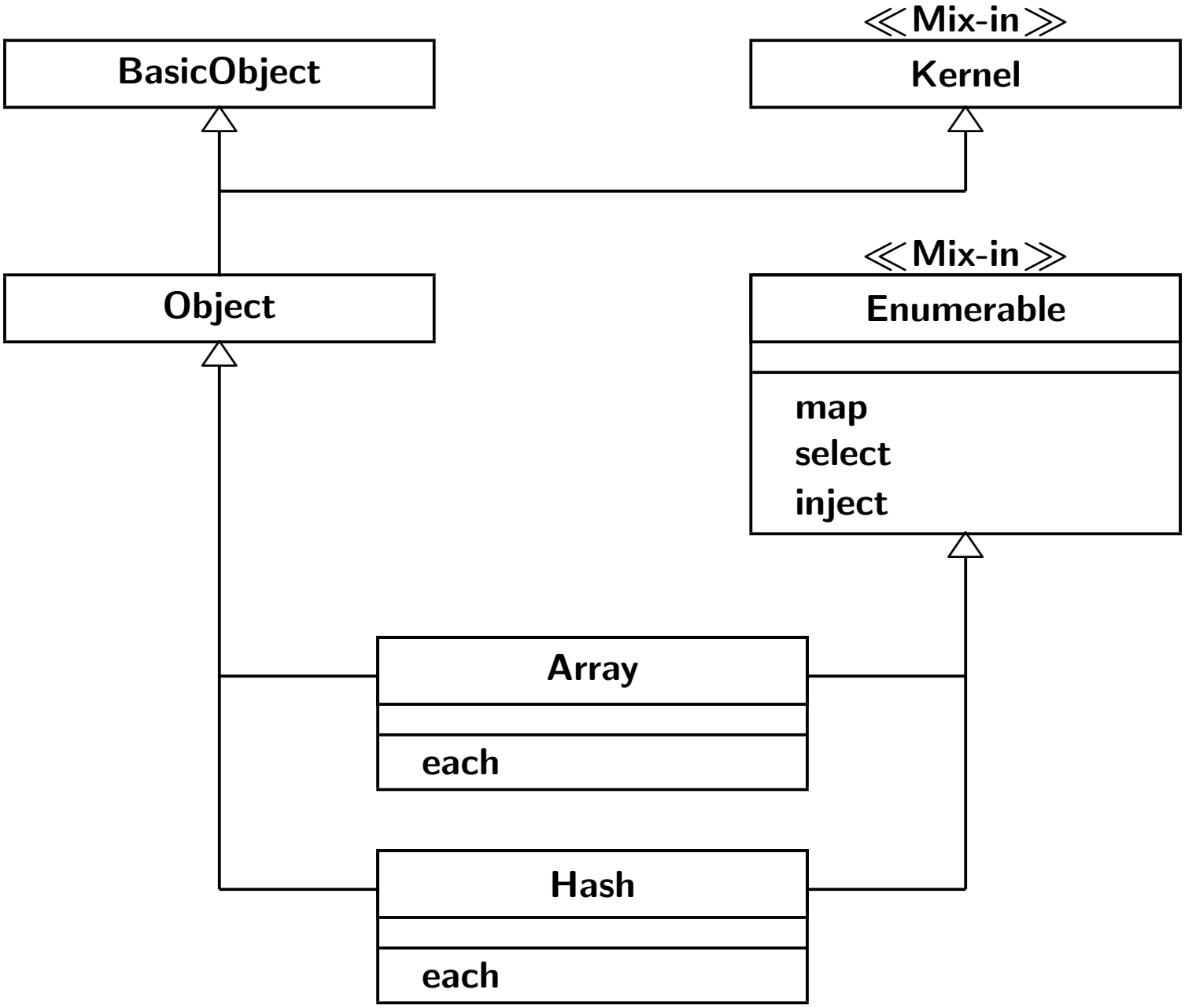
繰り返し

```
a = ['a', 'b', 'c']
a.each{|i| puts i}



# >> a
# >> b
# >> c
```

Enumerable

- 繰り返しを行なうクラスのためのモジュール
- クラスには each メソッドが必要

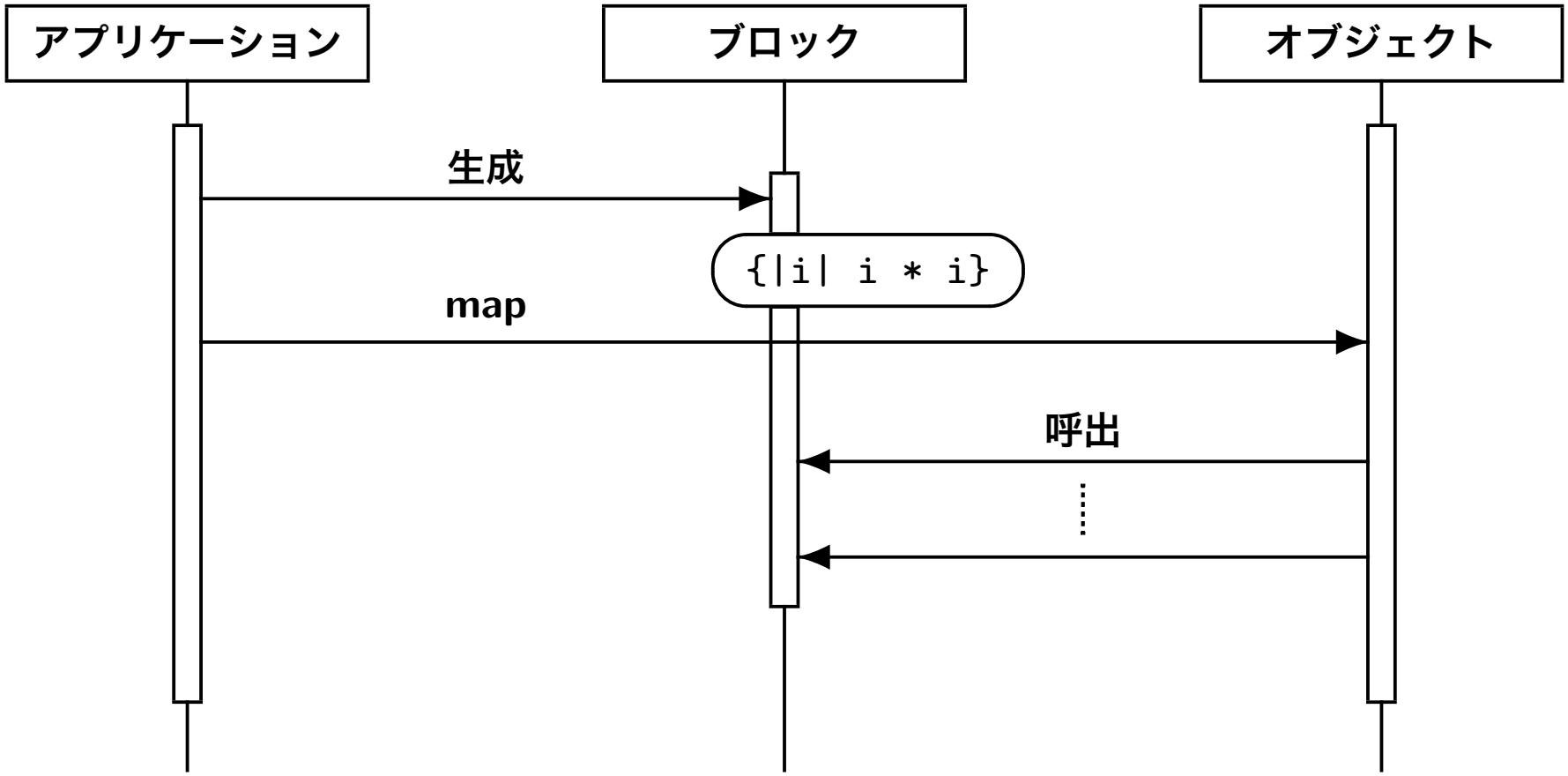
```
Array.ancestors
    # => [Array, Enumerable, Object, Kernel,
Hash.ancestors
    # => [Hash, Enumerable, Object, Kernel,
```


≪Mix-in≫
BasicObject
Kernel
≪Mix-in≫
Object
Enumerable
map
select
inject
Array
each
Hash
each

便利な例

```
a = [0, 1, 2, 3]        # => [0, 1, 2, 3]

a.map{|i| i * i}        # => [0, 1, 4, 9]
a.select{|i| i.even?}       # => [0, 2]
a.inject{|s, i| s + i}      # => 6
a.find{|i| i.odd?}          # => 1
a.all?{|i| i.even?}         # => false
a.any?{|i| i.even?}         # => true
```

array.map{|i| i * i}



ブロックを渡す

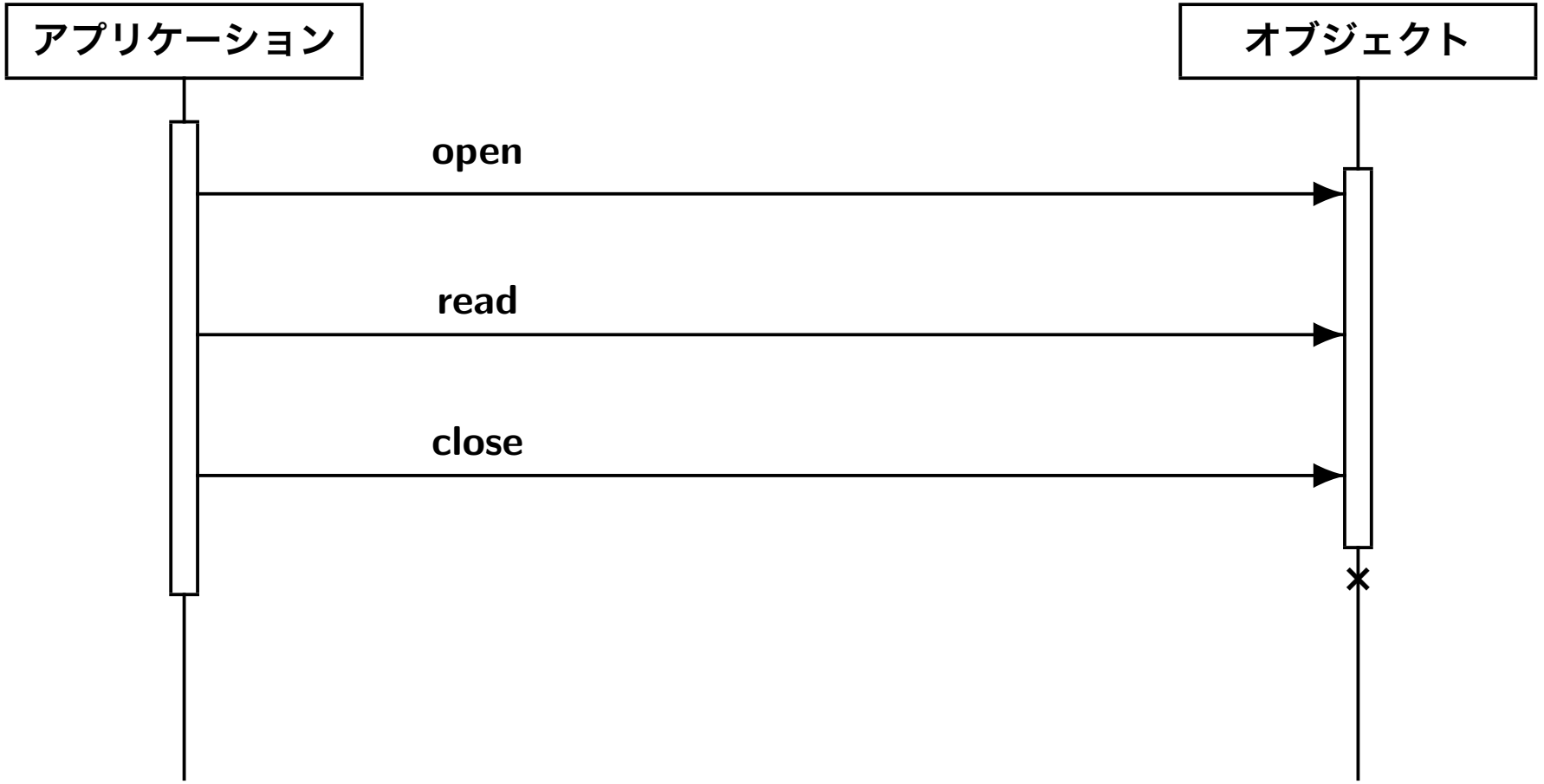
- メソッドには、ブロックをひとつ渡せる。
- ブロックをどう使うかは、メソッド次第。

```
open('hello.txt') # => #<File:hello.txt>
open('hello.txt'){|f| f.read}
                          # => "こんにちは\n"
```

ファイル入出力の例

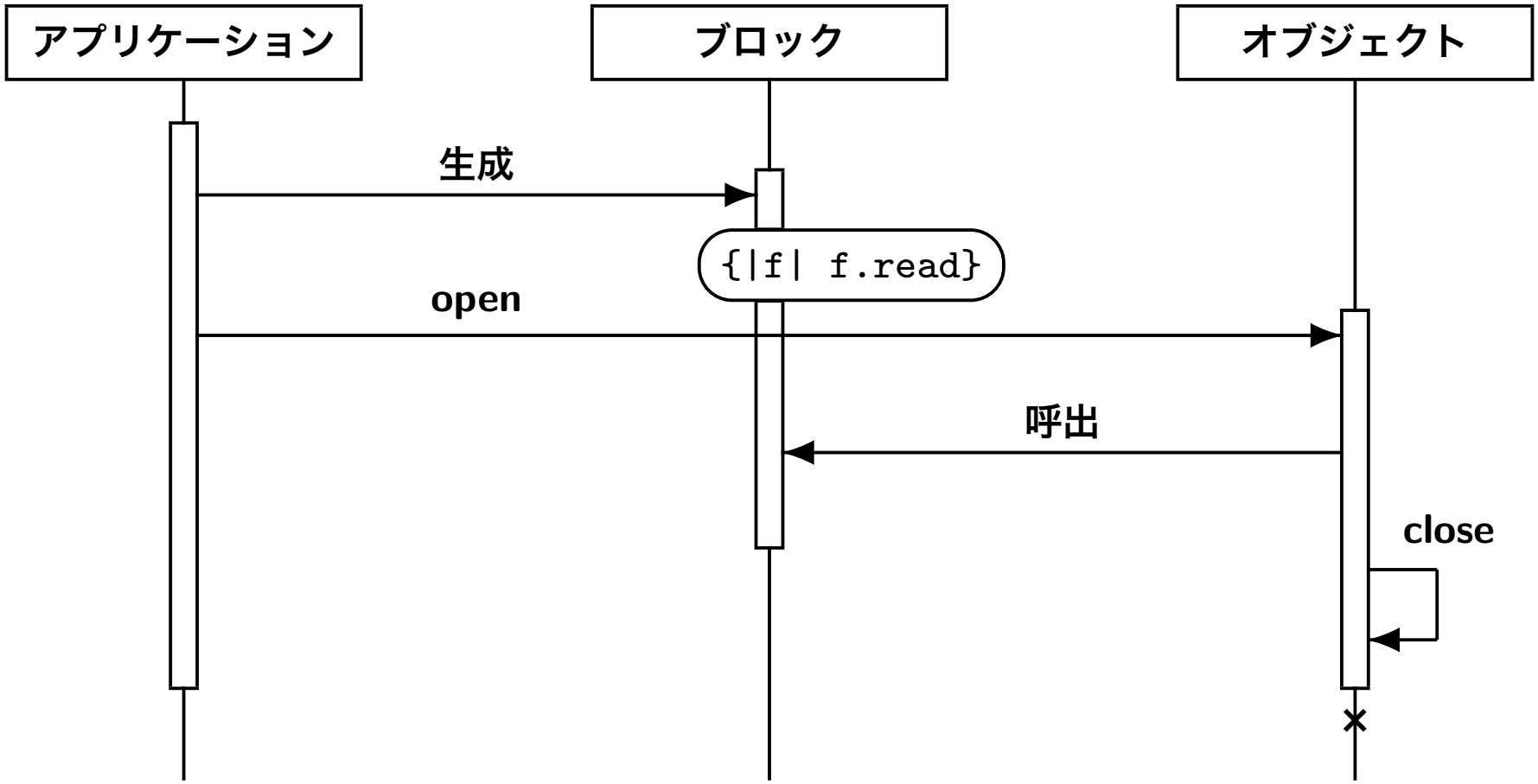
```
begin
  f = open('hello.txt')
  f.read
ensure
  f.close unless f.nil?
end
```

ブロックのない open



ブロック付き open

```
open('hello.txt'){|f| f.read}
```



値を受け取る

- ブロックは、値を受け取れる。
- 何を幾つ受け取れるかは、メソッド次第。

値は受け取らなくてもいい

```
2.times{|i| puts i}
# >> 0
# >> 1

2.times{puts 'こんにちは'}
# >> こんにちは
# >> こんにちは
```

Hash の例

```
people = {matz: 53, dhh: 39}
                # => {:matz=>53, :dhh=>39}

people.each do |person|
  p person
end

# >> [:matz, 53]
# >> [:dhh, 39]
```

Hash の例 (2)

```
people = {matz: 53, dhh: 39}
                # => {:matz=>53, :dhh=>39}

people.each do |name, age|
  puts "#{name}(#{age})"
end

# >> matz(53)
# >> dhh(39)
```

each_cons の例

```
kobe = ["大阪", "尼崎", "芦屋", "三宮"]

kobe.each_cons(2){|path| p path}

# >> ["大阪", "尼崎"]
# >> ["尼崎", "芦屋"]
# >> ["芦屋", "三宮"]
```

each_cons の例 (2)

```
kobe.each_cons(2) do |from, to|
  p "#{from} - #{to}"
end

# >> "大阪 - 尼崎"
# >> "尼崎 - 芦屋"
# >> "芦屋 - 三宮"
```

each_cons の例 (3)

```
kobe.each_cons(3){|i| p i}
# >> ["大阪", "尼崎", "芦屋"]
# >> ["尼崎", "芦屋", "三宮"]

kobe.each_cons(3){|i, j| p [i, j]}
# >> ["大阪", "尼崎"]
# >> ["尼崎", "芦屋"]
```

ブロックを受け取るメソッド

- こんな感じで呼びたい

```
x3{puts 'Ruby3'}


# >> Ruby3
# >> Ruby3
# >> Ruby3
```

ブロックを受け取る

```
def x3
  yield
  yield
  yield
end

x3{puts 'Ruby3'}
```

```
def x3(&block)
  block.call
  block.call
  block.call
end

# >> Ruby3
# >> Ruby3
# >> Ruby3
```

値を渡す

```
def x3
  yield 0
  yield 1
  yield 2
end
```

```
def x3(&block)
  block.call 0
  block.call 1
  block.call 2
end
```

```
x3{|i| puts "Ruby#{i}"}  # >> Ruby0
                         # >> Ruby1
                         # >> Ruby2
```

ふたつの値を渡す

```
def x3
  yield  'Ruby', 0  # 文字列 と 数値
  yield  'Ruby', 1
  yield ['Ruby', 2] # ひとつの配列
end
```

ふたつの値を渡す (2)

```
x3{|i, j| p "#{i} #{j}"}


# >> "Ruby 0"
# >> "Ruby 1"
# >> "Ruby 2"
```

ふたつの値を渡す (3)

```
x3{|i| p i}    # ひとつの変数で受けとる


# >> "Ruby"
# >> "Ruby"
# >> ["Ruby", 2]
```

ブロックは Proc

```
block = Proc.new do |i, j|
  puts "#{i}#{j}"
end

x3(&block)

# >> Ruby0
# >> Ruby1
# >> Ruby2
```

演習問題 0

今日のレッスンで分からなかったこと、疑問に思ったことをグループで話し合ってみよう。

演習問題 1

0 から 9 までの数値をもつ配列 a がある。

- 各要素を順番に表示しよう。
- 各要素を文字列にした配列を作ろう。
- 各要素を 2倍した値を持つ配列を作ろう。
- 全要素の合計値を計算しよう。

```
a = (0..9).to_a
  # => [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

演習問題 2

0 から 9 までの数値をもつ配列 a がある。
奇数の要素だけを持つ配列を作ろう。
ただし odd? メソッドは使用禁止。
(これは Enumerable を使ったパズルです)

```
a = [*0..9]
  # => [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
a. Enumerable メソッド { odd? 禁止 }
```

演習問題 3

Enumerable#map を自作してみよう。

```
module Enumerable
  def my_map
    ......
  end
end
```

ただし Enumerable#map と Enumerable#map! は使用禁止。

参考

- 公式サイト

  https://www.ruby-lang.org/
- るりま

  http://docs.ruby-lang.org/ja/
- 解答例

  https://github.com/higaki/

  learn_ruby_kansai_85