

# Ruby 講義

## 第5回 Ruby入門

Kuniaki IGARASHI/igaiga

2012.5.10 at 一橋大学

社会科学における情報技術とコンテンツ作成III  
(ニフティ株式会社寄附講義)

○ 剰余金の配当に関するお知らせ

○ ニフティ、「@nifty EMOBILE LTE 定額にねんプラン」の提供を開...

○ 「@nifty温泉」で「母の日 全国一斉！100のありがとう風呂」特設サイト公...

○ 「スマブレ！」のサービス停止について

○ ニフティとサンリオウェーブ、iOS向けアプリ「Hello Kitty Worl...

○ 平成24年3月期 決算短信

○ 特別損失の計上に関するお知らせ

○ 「シェフモ」登録会員数150万人を突破、「2012年 主婦の全国節電調査（冬季...

ニフティとなら、きっとかなう。  
With Us, You Can.

# ニフティ株式会社

HOME

NIFTY

ニュースリリース

アット・ニフティ

楽しいサービスがいっぱい

@nifty

アクセスマップ

大森から西新宿へ移転いたしました

@nifty Web募金

東日本大震災復興支援  
募金受付中



2012年4月25日 IR 特別損失の計上に関するお知らせ

2012年4月25日 IR 剰余金の配当に関するお知らせ

2012年4月25日 IR 平成24年3月期決算短信

2012年4月11日 IR ニフティ、「@nifty EMOBILE LTE 定額にねんプラン」の提供を開始

2012年4月11日 IR ニフティとサンリオウェーブ、iOS向けアプリ「Hello Kitty World」を台湾で提供開始

2012年4月10日 お知らせ 「@nifty温泉」で「母の日 全国一斉！100のありがとう風呂」特設サイト公開



講師

五十嵐邦明

株式会社万葉

エンジニア



いりや  
@igatt



Teaching Assistant 濱崎 健吾  
クックパッド株式会社 エンジニア



# 講義資料置き場

講義資料置き場をつくりました。  
過去の資料がDLできます。

<https://github.com/hitotsubashi-ruby/lecture2012>  
or

<http://bit.ly/ruby-lecture>

# 雑談・質問用facebookグループ

facebookグループを作りました

<https://www.facebook.com/groups/hitotsubashi.rb>

- 加入/非加入は自由です
- 加入/非加入は成績に関係しません
- 参加者一覧は公開されます
- 書き込みは参加者のみ見えます
- 希望者はアクセスして参加申請してください
- 雑談、質問、議論など何でも気にせずどうぞ～
- 質問に答えられる人は答えてあげてください
- 講師陣もお答えします
- 入ったら軽く自己紹介おねがいします

先週の

おさらい

# shell のコマンド

**ls : ファイル一覧を見る**

**cd : フォルダ移動**

**mkdir : フォルダ作成**

**pwd : 今いるフォルダ名を表示**

**cat : ファイルの中身を表示**



# 変数

教科書 p.21

オブジェクト へのラベル・荷札

変数 = オブジェクト

変数にオブジェクトを代入する

```
name = "igarashi"
```

(変数nameに"igarashi"オブジェクトを代入)

```
puts name  
=> "igarashi"
```

# 条件判断

**if - end**

if 条件

条件が成立した時に実行したい処理

end

**if - else - end**

if 条件

条件が成立した時に実行したい処理

else

条件が不成立の時に実行したい処理

end

**比較演算子 ==, !=**

# メソッドの定義、呼び出し

```
def メソッド名  
  メソッドで実行したい処理  
end
```

メソッド定義  
には **def** を  
使う

```
def hello  
  print "Hello, Ruby.\n"  
end
```

定義

```
hello()
```

呼び出し

メソッド呼び出しはメソッド名に()をつけます。  
そして、この()は省略可能です。(曖昧にならない限り)

# 別のファイルを 取り込む

```
require "../hello"
```

**実行フォルダにある hello.rb を取り込む**



# 対応する教科書のページ

**変数** p.21～23

**コメント** p.23～24

**条件判断・繰り返し** p.24～28

**メソッド** p.29

**別のファイルの読み込み(require)**

p.30～31

今週

ここから

# 目次

**Array**

**Hash**

# 配列(Array)

教科書

p.33~

## ほかのオブジェクトの入れもの

作り方の例：

```
names = ["五十嵐", "濱崎"]
```

```
numbers = [1,3,5]
```

[ と ] で囲い, で区切る。

文字列や数字ほか、どんなオブジェクトも入ります。

空っぽの配列は [] です。

概念図は教科書p.35 図2.2 を参照。

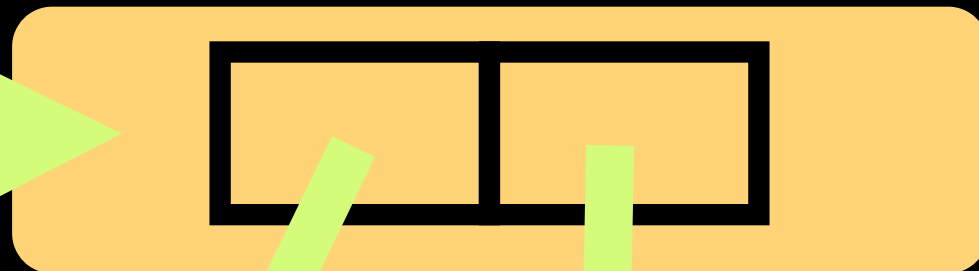
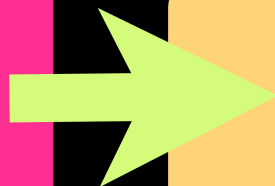


# 配列 (Array) 概念図

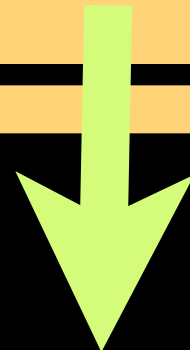
```
names = ["五十嵐", "濱崎"]
```

変数

● names



Array  
オブジェクト



五十嵐

濱崎

String  
オブジェクト

# 配列から読み込む

## 番号(index)を指定して読み込み

```
names = ["五十嵐", "濱崎"]
```

```
names[0] → "五十嵐"
```

```
names[1] → "濱崎"
```

最初の要素は0番です。1始まりではないので注意です。

また、末尾からの番号でも読み込めます。(-1始まり)

```
names[-1] → "濱崎"
```

```
names[-2] → "五十嵐"
```

# 配列へ追加する

配列の末尾にオブジェクトを追加するには  
**push**メソッドを使います。

```
names = ["五十嵐", "濱崎"]
```

```
names.push("山田")
```

```
p names → ["五十嵐", "濱崎", "山田"]
```

配列に入っている要素数を調べるには **size**メソッド

```
names = ["五十嵐", "濱崎", "山田"]
```

```
p names.size → 3
```

# 配列の繰り返し処理

教科書

p.38~

**eachで中身を順番に処理する  
ものすごくーく大事！！！！**

```
配列.each do |変数|
```

```
  繰り返したい処理
```

```
end
```



# 配列の繰り返し処理

教科書

p.38～

```
names = ["五十嵐", "濱崎", "山田"]  
names.each do |name|  
  puts name  
end
```

→ "五十嵐"

"濱崎"

"山田"

nameの両脇  
にある記号 |  
はパイプと読み  
ます。

キーボードの右  
上の方にある  
(たぶん)。

nameの中身が1回目は"五十嵐"、2回目は "濱崎",  
3回目は"山田"となり、繰り返しputs文を実行

# 配列 (Array) の演習

**1. 配列 [1,3,5] の全要素を表示するコードを書いてください。**

**2. 配列 [1,3,5] の全要素を加えた結果を表示するコードを書いてください。**

**ヒント: ある変数に数  $N$  を足すのは  $x = x + N$  です。これは  $x += N$  とも書けます。**

# 配列(Array)の演習【上級】

**3.【上級】 配列 [1,1,2,2,3]を[1,2,3]にするコードを1行で書いてください。**

ヒント：リファレンスでArrayのメソッドを探してみましょう。

<http://miyamae.github.com/rubydoc-ja/1.9.3/>

**4.【上級】 上記2.をeachを使わずに書いてください**

ヒント：injectを使います。使い方はリファレンスから探しましょう。

<http://miyamae.github.com/rubydoc-ja/1.9.3/>

# ハッシュ(Hash)

教科書

p.40~

これもほかのオブジェクトの入れもので使います。

使い道の例としては、、、





**COOKPAD**  
http://cookpad.com

レシピ検索No.1

[詳細検索](#) | [最近見たレシピ](#)



[筍](#) [塩麴](#) [ボンゴレ](#) [あさり](#) [新玉ねぎ](#) [もっと見る...](#)

検索

レシピをさがす

レシピをのせる

クックパッド

«hmskpad のレシピ (13品)



レシピID:1188379

## オーソドックスなとんかつ



揚げ物楽しい, 豚肉安い, ソースも簡単



[hmskpad](#)

### 材料 (1人分)

豚ロース	1枚くらい
こしょう	少々
サラダ油	豚ロースが全て浸かるくらい

### ■衣

小麦粉 (薄力粉)	100gくらい
卵	1個弱
パン粉	100gくらい

こういうページを表現したいときに

# オーソドックスなとんかつ ←title

揚げ物楽しい、豚肉安い、ソースも簡単

 [hmskpad](#)

description ↑

author ↑

## 材料 (1人分)

豚ロース	1枚くらい
こしょう	少々
サラダ油	豚ロースが全て浸かるくらい

## ■ 衣

小麦粉 (薄力粉)	100gくらい
卵	1個弱
パン粉	100gくらい

## ■ ソース

ケチャップ	大さじ2
ウスターソース	100ccくらい

ingredients →

データにラベルを付けると扱い易いです





# オーソドックスなとんかつ ← title

レシピID : 1188379



揚げ物楽しい, 豚肉安い, ソースも簡単

description ↑

author ↑

 hmskpad

豚ロース	1枚くらい
こしょう	少々
サラダ油	豚ロースが全て浸かるくらい

## ■ 衣

小麦粉 (薄力粉)	100gくらい
卵	1個弱
パン粉	100gくらい

ingredients →

Hashを使う  
とこんな感じ  
でまとめられ  
ます

```
recipe = {  
  :title => "オーソドックスなとんかつ",  
  :author => "hmskpad",  
  :description => "揚げ物楽しい、豚肉安い、  
ソースも簡単",  
  :ingredients => [省略] }
```

# ハッシュ(Hash)

教科書

p.40~

キーと値の組を持てるオブジェクトの入れもの。なんでも入ります。

作り方の例：

```
recipe = {  
  :title => "♥日向夏のジャム♥",  
  :author => "濱崎" }
```

↑ キー

↑ 値

{ }で囲う キー => 値 区切りは , 空配列は {}  
ここでキーに使われている：始まりのものは何？

# シンボル

教科書

p.41~

ラベルとして使う文字列的なもの

`:title`

シンボルにするには先頭に `:` を付ける

ハッシュのキーによく使います

文字列との変換もできます

シンボルへ `"foo".to_sym → :foo`

文字列へ `:foo.to_s → "foo"`

# ハッシュから読み込む

```
recipe = { :title => "♥日向夏のジャム♥",  
           :author => "濱崎" }  
p recipe[:title] → "♥日向夏のジャム♥"  
p recipe[:author] → "濱崎"
```

ハッシュ名[キー] で読み込みます。



# ハッシュへ追加する

ハッシュ名[キー] = 格納したいオブジェクト

同じキーの要素は追加不可(上書き)

ハッシュオブジェクト内でキーは唯一のもの(ユニーク)

```
recipe = { :title => "♥日向夏のジャム♥",  
  :author => "濱崎" }
```

```
recipe[:url] = "http://cookpad.com/recipe/xxx"
```

追加後

```
p recipe → { :title => "♥日向夏のジャム♥",  
  :author => "濱崎",  
  :url => "http://cookpad.com/recipe/xxx" }
```

# ハッシュの繰り返し処理

**Arrayと同じですが、変数を2個とります。**

```
ハッシュ.each do |キーの変数, 値の変数|  
  繰り返したい処理  
end
```

```
recipe = { :title => "♥日向夏のジャム♥", :author => "濱崎" }  
recipe.each do |k, v|  
  print k, " - ", v, "\n"  
end  
→ title - ♥日向夏のジャム♥  
   author - 濱崎
```

# ArrayとHashの使い分け

**Array : 順番が決まってるいれもの**

- 並び順が重要なもの
- データを重複させたい場合にも使える

**Hash : キー(名札)を付けられるいれもの**

- 順番が保持されなくても困らないもの

※Ruby 1.9 からはHashも順番を保持します。

- キーが重複しない場合に利用

# nil

教科書

p.47

**「ない」ことを表すオブジェクト**

**例えば、ハッシュで存在しないキーを読もう  
とするとこの nil が返ってきます。**

**if 文などで条件判断をする場合、**

**nil は 偽（不成立）になります。**

**偽（不成立）になるのは false と nil の2つだけです。**

**それ以外の全ての値は真（成立）になります。**

pp

p より見易いデバッグ用メソッド

```
require "pp"
```

```
pp 見たい変数
```

ppには require文 が必要

# Hashの演習

**1. ハッシュの全要素を表示する、右のコードを実行してください。**

```
recipe = {  
  :title => "いちごのコンフィ",  
  :author => "濱崎" }  
recipe.each do |k, v|  
  print k, " - ", v, "\n"  
end
```

**2. 1.のコード中のrecipeの :author キーに対応する値を "五十嵐" とする代入文を書いてください。**



# Hashの演習【上級】

**3.【上級】 右のnumsを値が偶数の要素だけにしてください。**

ヒント：リファレンスでHashのメソッドを探してみましょう。  
<http://miyamae.github.com/rubydoc-ja/1.9.3/>  
剰余(割り算した余り)を求める演算子は % です。

```
nums = {  
  :a => 1,  
  :b => 2,  
  :c => 3,  
  :d => 4,  
  :e => 5}
```

まとめ

# 配列(Array)

教科書

p.33~

## ほかのオブジェクトの入れもの

作り方の例：

```
names = ["五十嵐", "濱崎"]
```

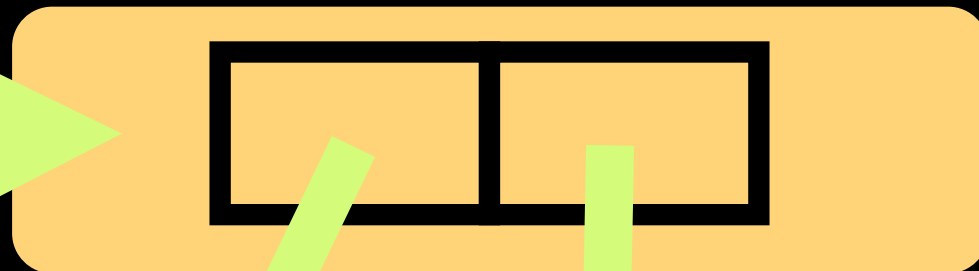
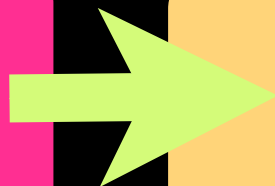
```
numbers = [1,3,5]
```

# 配列(Array)概念図

```
names = ["五十嵐", "濱崎"]
```

変数

● names



Array  
オブジェクト



五十嵐

濱崎

String  
オブジェクト

# 配列から読み込む

## 番号(index)を指定して読み込み

```
names = ["五十嵐", "濱崎"]
```

```
names[0] → "五十嵐"
```

```
names[1] → "濱崎"
```

最初の要素は0番です。1始まりではないので注意です。

# 配列へ追加する

配列の末尾にオブジェクトを追加するには  
pushメソッドを使います。

```
names = ["五十嵐", "濱崎"]
```

```
names.push("山田")
```

```
p names → ["五十嵐", "濱崎", "山田"]
```

# 配列の繰り返し処理

教科書

p.38~

**eachで中身を順番に処理する  
ものすごくーく大事！！！！**

```
配列.each do |変数|
```

```
  繰り返したい処理
```

```
end
```

# ハッシュ(Hash)

教科書

p.40~

## ほかのオブジェクトの入れもの

キーと値の組を持てます。なんでも入ります。

作り方の例：

```
recipe = {  
  :title => "♥日向夏のジャム♥",  
  :author => "濱崎" }
```

↑ キー

↑ 値



# オーソドックスなとんかつ ←title

レシピID : 1188379



揚げ物楽しい, 豚肉安い, ソースも簡単

description ↑

author ↑

 hmskpad

豚ロース	1枚くらい
こしょう	少々
サラダ油	豚ロースが全て浸かるくらい

## ■ 衣

小麦粉 (薄力粉)	100gくらい
卵	1個弱
パン粉	100gくらい

ingredients →

Hashを使う  
とこんな感じ  
でまとめられ  
ます

```
recipe = {  
  :title => "オーソドックスなとんかつ",  
  :author => "hmskpad",  
  :description => "揚げ物楽しい、豚肉安い、  
ソースも簡単",  
  :ingredients => [省略] }
```

# シンボル

教科書

p.41～

ラベルとして使う文字列的なもの

`:title`

シンボルにするには先頭に `:` を付ける

# ハッシュから読み込む

```
recipe = { :title => "♥日向夏のジャム♥",  
           :author => "濱崎" }  
p recipe[:title] → "♥日向夏のジャム♥"  
p recipe[:author] → "濱崎"
```

ハッシュ名[キー] で読み込みます。

# ハッシュへ追加する

ハッシュ名[キー] = 格納したいオブジェクト

同じキーの要素は追加不可(上書き)

ハッシュオブジェクト内でキーは唯一のもの(ユニーク)

```
recipe = { :title => "♥日向夏のジャム♥",  
  :author => "濱崎" }
```

```
recipe[:url] = "http://cookpad.com/recipe/xxx"
```

追加後

```
p recipe → { :title => "♥日向夏のジャム♥",  
  :author => "濱崎",  
  :url => "http://cookpad.com/recipe/xxx" }
```

# ハッシュの繰り返し処理

**Arrayと同じですが、変数を2個とります。**

```
ハッシュ.each do |キーの変数, 値の変数|  
  繰り返したい処理  
end
```

```
recipe = { :title => "♥日向夏のジャム♥", :author => "濱崎" }  
recipe.each do |k, v|  
  print k, " - ", v, "\n"  
end  
→ title - ♥日向夏のジャム♥  
   author - 濱崎
```

# ArrayとHashの使い分け

**Array : 順番が決まってるいれもの**

- 並び順が重要なもの
- データを重複させたい場合に利用

**Hash : キー(名札)を付けられるいれもの**

- 順番が保持されなくても困らないもの

※Ruby 1.9 からはHashも順番を保持します。

- キーが重複しない場合に利用

# nil

教科書

p.47

**「ない」ことを表すオブジェクト**

**例えば、ハッシュで存在しないキーを読もうとするとこの nil が返ってきます。**

**if 文などで条件判断をする場合、**

**nil は偽（不成立）になります。**

**偽（不成立）になるのは false と nil の2つだけです。**

**それ以外の全ての値は真（成立）になります。**

解答



# 配列 (Array) の演習

1. 配列 [1,3,5] の全要素を表示するコードを書いてください。

2. 配列 [1,3,5] の全要素を加えた結果を表示するコードを書いてください。

ヒント: ある変数に数 N を足すのは  $x = x + N$  です。これは  $x += N$  とも書けます。

3. 【上級】 配列 [1,1,2,2,3] を [1,2,3] にするコードを1行で書いてください。

ヒント: リファレンスでArrayのメソッドを探してみましょう。

<http://miyamae.github.com/rubydoc-ja/1.9.3/>

4. 【上級】 上記2.をeachを使わずに書いてください

ヒント: injectを使います。使い方はリファレンスから探しましょう。

<http://miyamae.github.com/rubydoc-ja/1.9.3/>

# 配列(Array)の演習解答

1. 配列 [1,3,5] の全要素を表示するコードを書いてください。

```
array = [1,3,5]  
array.each do |x|  
  puts x  
end
```

# 配列(Array)の演習解答

2. 配列 [1,3,5] の全要素を加えた結果を表示するコードを書いてください。

ヒント: ある変数に数 N を足すのは  $x = x + N$  です。これは  $x += N$  とも書けます。

```
sum = 0
array = [1,3,5]
array.each do |x|
  sum = sum + x
end
puts sum
```

# 配列(Array)の演習解答

**3. 【上級】 配列 [1,1,2,2,3]を[1,2,3]にするコードを1行で書いてください。**

ヒント：リファレンスでArrayのメソッドを探してみましょう。

<http://miyamae.github.com/rubydoc-ja/1.9.3/>

```
p [1,1,2,2,3].uniq!
```

**配列中の重複を除き、各値1つずつにするには  
Array#uniq! を使います。**

<http://miyamae.github.com/rubydoc-ja/1.9.3/#!/method/-array/i/uniq=21.html>

# 配列(Array)の演習解答

## 4. 【上級】 上記2.をeachを使わずに書いてください

ヒント : injectを使います。使い方はリファレンスから探しましょう。

<http://miyamae.github.com/rubydoc-ja/1.9.3/>

```
p [1,3,5].inject(0){|sum, i| sum + i }
```

inject は便利でかっこいいメソッドです。

<http://miyamae.github.com/rubydoc-ja/1.9.3/#!/method/-enumerable/i/reduce.html>

ちなみに、なぜArrayクラスのメソッドではないinjectを使えるかというと、ArrayはEnumerableというモジュールのメソッドも使えるからです。Arrayだけでなく、eachを持つ全てのクラスはEnumerableを使うことができます。

# Hashの演習

1. ハッシュの全要素を表示する、右のコードを実行してください。

```
recipe = {  
  :title => "いちごのコンフィ",  
  :author => "濱崎" }  
recipe.each do |k, v|  
  print k, " - ", v, "\n"  
end
```

2. 1.のコード中のrecipeの :author キーに対応する値を "五十嵐" とする代入文を書いてください。

3. 【上級】 右のnumsを値が偶数の要素だけにしてください。

```
nums = {  
  :a => 1,  
  :b => 2,  
  :c => 3,  
  :d => 4,  
  :e => 5}
```

ヒント：リファレンスでHashのメソッドを探してみましょう。  
<http://miyamae.github.com/rubydoc-ja/1.9.3/>  
剰余(割り算した余り)を求める演算子は % です。

# Hashの演習解答

2. 1.のコード中のrecipeの :author キーに対応する値を "五十嵐" とする代入文を書いてください。

```
recipe = {  
  :title => "いちごのコンフィ",  
  :author => "濱崎" }  
recipe[:author] = "五十嵐"
```

# Hashの演習解答

3. 【上級】 右のnumsを値が偶数の要素だけにしてください。

```
nums = { :a => 1, :b => 2,  
         :c => 3, :d => 4, :e => 5}  
nums.select! do |k,v|  
  v % 2 == 0  
end  
p nums
```

**select!** は、あとに続く

**do - end** の結果が**true** の要素だけを残すメソッドです。





**過去資料置き場**

**<https://github.com/hitotsubashi-ruby/lecture2012>**

**facebook group**

**<https://www.facebook.com/groups/hitotsubashi.rb>**