

Ruby 講義

第7回 Wikipediaアクセス解析

五十嵐邦明

twitter : igaiga555

<http://www.facebook.com/kuniaki.igarashi>



2013.5.23 at 一橋大学
ニフティ株式会社寄附講義
社会科学における情報技術と
コンテンツ作成III

五十嵐邦明 講師 株式会社万葉



twitter: igaiga555

<https://github.com/igaiga/>

<http://www.facebook.com/kuniaki.igarashi>

濱崎 健吾

Teaching Assistant
fluxflex, inc(米国法人)



twitter: hmsk

<https://github.com/hmsk/>

<http://www.facebook.com/hamachang>

以下の2ファイルを wikipedia 解析
演習で使うので DL してください。

<http://bit.ly/wpdatamini>

<http://bit.ly/wpdata2013>

**TODO
休講連絡？**

先週の復習

ファイルから1行ずつ読み込み



教科書
p.51-55

サンプルコード

```
filename = "20120301-000000-ja.txt"
file = File.open(filename, "r:UTF-8")
while text = file.gets
  puts text
end
file.close
```

ファイルから各行読み込み

```
filename = "20120301-000000-ja.txt"  
file = File.open(filename, "r", encoding: "UTF-8")  
while text = file.gets  
  puts text  
end
```

while: 条件成立中は繰り返し
この場合、条件は
text = file.gets
になります。

text に1行目のデータが入って**end**まで処理

while の行へ戻り、

は1行読み込むメソッド

text に2行目のデータが入って**end**まで処理

... 全行繰り返し

ファイル終端で読み込めない

終端で**file.gets**すると条件不成立になり繰り返し終了

next文



教科書
p.102

```
while text = file.gets
```

```
... (A) ...
```

```
next
```

```
... (B) ...
```

```
end
```

繰り返し中にnext文を書くと、それ以降の処理を行わず、
繰り返し先頭に戻します。

... (A) ... の部分の処理が行われてnextへくると、

... (B) ... の部分の処理は行われず、

繰り返し先頭へ戻ります。

教科書p.102の図6.2が分かりやすいです。

unless文



教科書
p.76

unless 条件

処理

end

条件が偽の時に処理を実行

if文と対になる文

if文とは逆に条件が偽の時に処理実行

以下の2つの文は同じ意味

puts "hello" if x != 3

puts "hello" unless x == 3

正規表現



教科書 p.44-46
p.267-290

文字列がパターンに一致（マッチ）するか調べる道具

ものすごく便利で、強力で、奥が深いです。
ここではごく基本的な説明だけを行います。

文字列 =~ /正規表現パターン/

`=~` → 正規表現マッチを行う演算子

左辺の文字列中に正規表現パターンが含まれるかを判定します。

パターンが英数字や漢字だけからなる場合、

単純に文字列にパターンが含まれるかどうかを判定します。

`"Ruby" =~ /Ruby/ #=> 0`

マッチした場合はマッチ位置を返します

`"Ruby" =~ /Diamond/ #=> nil`

マッチしない場合はnilを返します

正規表現



教科書 p.44-46
p.267-290

ほかにも便利な検索の機能があります。

```
text =~ /^ja/
```

正規表現パターン中の [^] は行先頭の意味

/[^]ja/ は 対象文字列が ja から始まればマッチする
パターンです。

```
"ja title count" =~ /^ja/ #=> 0
```

行先頭が ja から始まるのでマッチ

```
"xxxja" =~ /^ja/ #=> nil
```

jaはあるが、行先頭ではないのでマッチしない

もっと知りたい人はこの辺のページを読んでみてください。

<http://doc.ruby-lang.org/ja/2.0.0/doc/spec=2fregexp.html>

<http://www.namaraii.com/rubytips/>

next, unless, 正規表現

next unless text =~ /**ja**/

unless は条件 **text =~ /ja/** が不成立だったらその前の部分(**next**)を実行。

条件 **/ja/** は正規表現でjaから始まる の意味。

nextは以降の処理を行わずに次の繰り返しへ進む。

→**text** が ja から始まらなかった場合、
以降の処理は行わず、
次の繰り返し (次の行) へ進む。

String#split メソッド

splitは文字列を空白文字で区切ってArrayにします。

```
data = list.split
```

```
"ja.b %E3%81%84%E3%82%8D 1 6661".split  
→["ja.b", "%E3%81%84%E3%82%8D","1","6661"]
```

CGI.unescape メソッド

%XX%XXっていう謎の文字列を、読める形式に変換します。

```
require "cgi"  
CGI.unescape("%E3%81%84%E3%82%  
8D%E3%81%AF") #=> "いろは"
```

この変換をURLデコードと呼びます。

例外処理



教科書

p.153-168

begin

例外を発生させる可能性のある処理

rescue Exception => 变数

例外が起こった場合の処理

end

コード実行中、うまく処理できない場合などに例外を発生させるメソッドがあります。例外が発生した場合、**rescue** 節で例外を捕まえ、その際に実行する特殊処理を書くことができます。

もしも、例外を**rescue** で捕まえない場合は、プログラムはそこでエラー終了します。

来週の講義にて
以下の2ファイルをWikipedia解析演習
で使うのでDLしておいてください。

<http://bit.ly/wpdatasmall>

<http://bit.ly/wpdata2013>

今週

ここから

目次

Wikipediaアクセス解析(後編)

Wikipediaのアクセス数の解析の実習2週目です。
必要なRubyの知識を説明しながら、いよいよ解析します。

ブロック
リファレンスマニュアル

Wikipediaのアクセス数解析

wikipediaは1時間ごとのアクセス数データを公開しています。

<http://dumps.wikimedia.org/other/pagecounts-raw/>

Index of page view statistics for 2012-05

Pagecount files for 2012-05

Check the [hashes](#) after your download, to make sure your files arrived intact.

- [pagecounts-20120501-000000.gz](#), size 69M
- [pagecounts-20120501-010000.gz](#), size 67M
- [pagecounts-20120501-020000.gz](#), size 67M
- [pagecounts-20120501-030000.gz](#), size 66M
- [pagecounts-20120501-040000.gz](#), size 67M
- [pagecounts-20120501-050000.gz](#), size 77M
- [pagecounts-20120501-060000.gz](#), size 75M
- [pagecounts-20120501-070000.gz](#), size 80M

Wikipediaアクセス数データ

ja.b %C3%84 1 6499

ja.b %C3%88%C2%B1%C3%AF%C2%BF%C2%BD%C3%A7%C2%AC%C2%AC%C3%AF
%C2%BD%C2%B3%C3%A6%C3%AF%C2%BF%C2%BD%C3%AF%C2%BF%C2%BD
%C3%AF%C2%BD%C2%AC%C3%AF%C2%BD%C2%AC973%C3%A8%C2%AD%C3%AF
%C2%BF%C2%BD%C3%AF%C2%BD%C2%A1 1 6656

ja.b %E3%81%95%E3%81%BE%E3%81%96%E3%81%BE%E3%81%AA%E9%9D
%A2%E3%81%8B%E3%82%89%E8%A6%8B%E3%81%9F%E6%97%A5%E6%9C%AC_
%E5%9C%BO%E7%90%86_%E6%B0%97%E5%80%99 1 18210

ja.b %E3%82%A6%E3%82%A3%E3%82%AD%E3%83%9A
%E3%83%87%E3%82%A3%E3%82%A2%E3%81%AE%E6%9B%88%E3%81%8D
%E6%96%B9_%E3%83%9D%E3%83%BC%E3%82%BF%E3%83%AB%E3%83%BB
%E3%83%97%E3%83%AD%E3%82%B8%E3%82%A7%E3%82%AF
%E3%83%88%E6%A1%88%E5%86%85 1 12093

ja.b %E3%82%A6%E3%82%A3%E3%82%AD%E3%83%9A
%E3%83%87%E3%82%A3%E3%82%A2%E3%81%AE%E6%9B%88%E3%81%8D
%E6%96%B9_%E5%85%A5%E9%96%80%E7%B7%A8-
%E3%82%A6%E3%82%A3%E3%82%AD%E3%83%9A
%E3%83%87%E3%82%A3%E3%82%A2%E3%81%A8%E3%81%AF%EF%BC%9F 1
15837

...

っていうデータが数十万行

Wikipediaアクセス数データ

データ構造を見るためにちょっと読みやすく変えたもの

ja.b アーティキュレーションを表す記号 1 7642

ja.b カテゴリ:stab 1 68732

ja.b カテゴリ:大学入試 3 45061

ja.b カテゴリ:民法 1 47619

ja.b カテゴリ:社会学 1 6661

ja.b カテゴリ:User_bg 1 7931

ja.b カテゴリ:User_uk-3 1 6599

ja.b ガス事業法第2条 1 8541

ja.b ガリア戦記 1 12936

ja.b ガリア戦記/参照画像一覧 1 54089

ja.b コントラクトブリッジ/ルール 2 7957

ja.b コントラクトブリッジ/ルール/スコアリング 1 14903

...

っていうデータが数十万行

Wikipediaアクセス数解析

言語種別 ページタイトル アクセス数 容量

ja.b %E3%81%84%E3%82%8D 1 6661

スペース区切りで以下の4項目が書かれている

言語種別： ja から始まるのが日本のデータ

ページタイトル： アクセスされたページ名

ただし、プログラムで扱い易い形式(%XX)になっている

アクセス数： アクセスされた回数

容量： そのページのデータサイズ

Wikipediaアクセス数データ

言語種別 ページタイトル アクセス数 容量

ja.b %C3%84 1 6499
ja.b %C3%88%C2%B1%C3%AF%C2%BF%C2%BD%C3%A7%C2%AC%C2%AC %C3%AF%C2%BD%C2%B3%C3%A6%C3%AF%C2%BF%C2%BD%C3%AF%C2%BF %C2%BD%C3%AF%C2%BD%C2%AC%C3%AF%C2%BD %C2%AC973%C3%A8%C2%AD%C3%AF%C2%BF%C2%BD%C3%AF%C2%BD %C2%A1 1 6656
ja.b %E3%81%95%E3%81%BE%E3%81%96%E3%81%BE%E3%81%AA%E9%9D %A2%E3%81%8B%E3%82%89%E8%A6%8B%E3%81%9F%E6%97%A5%E6%9C %AC_%E5%9C%BO%E7%90%86_%E6%BO%97%E5%80%99 1 18210
...

このデータを解析して、ある1時間のアクセス数トップ20をコードを書いて調べてみます。
簡単に言うと、「アクセス数」欄の数が大きいものから20個、その「ページタイトル」を表示させる

Wikipediaのアクセス数解析

コードで書く際の処理の流れ

- ・データファイルを開く
- ・データファイルから1行読み込む
- ・日本語データ以外はパス
- ・データ1行からタイトルとカウントを取得
- ・取得データをいれものに詰めてとておく
- ・データファイル全行について繰り返し
- ・データファイルを閉じる
- ・貯まったデータをカウント順にソート（並べ替え）
- ・トップ20件表示

Wikipediaのアクセス数解析

```
# encoding: utf-8
require "cgi"
filename = "20120301-000000-ja.txt"
file = File.open(filename, "r:UTF-8")
list = []
while text = file.gets
begin
  next unless text =~ /^ja/
  data = text.split
  h = { :title => CGI.unescape(data[1]), :count => data[-2] }
  list.push h
rescue Exception => e
  #p e
end
end
file.close

# count順にソート
result = list.sort_by do |i|
  i[:count].to_i
end

# トップ20表示
result.reverse.first(20).each do |i|
  puts i
end
```

30行くらいで書けます

Wikipediaのアクセス数解析

コードで書く際の処理の流れ

- ・データファイルを開く
- ・データファイルから1行読み込む
- ・日本語データ以外はパス
- ・データ1行からタイトルとカウントを取得
- ・取得データをいれものに詰めてとっておく
- ・データファイル全行について繰り返し
- ・データファイルを閉じる
- ・貯まったデータをカウント順にソート（並べ替え）
- ・トップ20件表示

先週はここまでできました。

Wikipediaのアクセス数解析

コードで書く際の処理の流れ

- ・データファイルを開く
- ・データファイルから1行読み込む
- ・日本語データ以外はパス
- ・データ1行からタイトルとカウントを取得
- ・取得データをいれものに詰めてとておく
- ・データファイル全行について繰り返し
- ・データファイルを閉じる
- ・貯まったデータをカウント順にソート（並べ替え）
- ・トップ20件表示

今週はここを説明

Wikipediaのアクセス数解析

```
# encoding: utf-8
require "cgi"
filename = "20120301-000000-ja.txt"
file = File.open(filename, "r:UTF-8")
list = []
while text = file.gets
begin
  next unless text =~ /^ja/
  data = text.split
  h = { :title => CGI.unescape(data[1]), :count => data[-2] }
  list.push h
rescue Exception => e
  #p e
end
end
file.close

# count順にソート
result = list.sort_by do |i|
  i[:count].to_i
end

# トップ20表示
result.reverse.first(20).each do |i|
  puts i
end
```

今週はここを説明

Wikipediaのアクセス数解析

```
# encoding: utf-8
require "cgi"
filename = "20120301-000000-ja.txt"
file = File.open(filename, "r:UTF-8")
list = []
while text = file.gets
begin
  next unless text =~ /^ja/
  data = text.split
  h = { :title => CGI.unescape(data[1]), :count => data[-2] }
  list.push h
rescue Exception => e
  #p e
end
end
file.close
```

ここまで、できしたこと

count順にソート
データファイルの各行を解析して、
結果を変数list(Arrayオブジェクト)へ詰める

```
# トップ20表示
result.reverse.first(20).each do |i|
  puts i
end
```

データファイル各行の解析結果

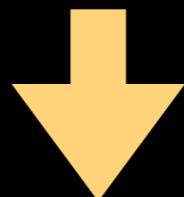
言語種別 ページタイトル アクセス数 容量

ja.b %E3%81%84%E3%82%8D 1 6661

ja.b %E3%81%95%E3%81%BE 3 2342

ja.b %E4%BC%81%E6%A5%AD 2 1656

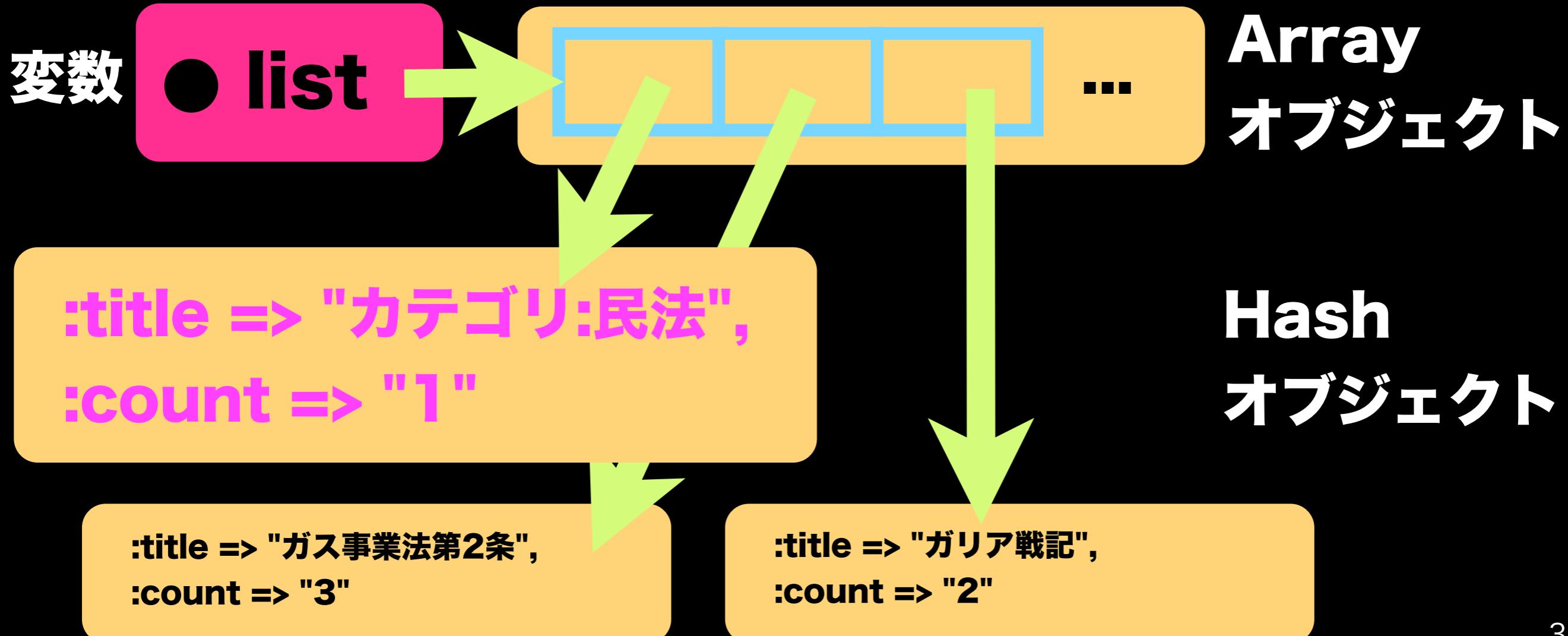
変数listの中身



```
[  
  {:title=>"カテゴリ:民法", :count=>"1"},  
  {:title=>"ガス事業法第2条", :count=>"3"},  
  {:title=>"ガリア戦記", :count=>"2"},  
 ...  
 ]
```

変数listの中身の概念図

```
[{:title=>"カテゴリ:民法", :count=>"1"},  
{:title=>"ガス事業法第2条", :count=>"3"},  
{:title=>"ガリア戦記", :count=>"2"},  
...]
```

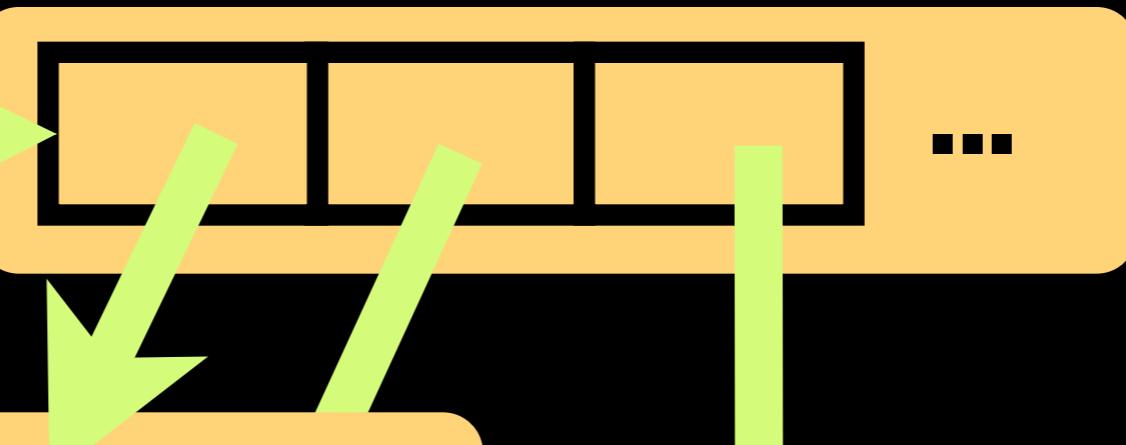
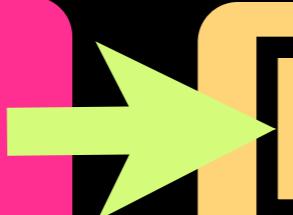


count順に並べ変えたい =アクセスが多いランキングを見たい

```
[{:title=>"カテゴリ:民法", :count=>"1"},  
{:title=>"ガス事業法第2条", :count=>"3"},  
{:title=>"ガリア戦記", :count=>"2"},  
...]
```

変数

● list



Array
オブジェクト

:title => "カテゴリ:民法",

:count => "1"

Hash
オブジェクト

:title => "ガス事業法第2条",
:count => "3"

:title => "ガリア戦記",
:count => "2"

count順に並べ変えたい

```
[{:title=>"カテゴリ:民法", :count=>"1"},  
{:title=>"ガス事業法第2条", :count=>"3"},  
{:title=>"ガリア戦記", :count=>"2"},  
...]
```

変数

list



Array
オブジェクト

```
[{:title=>"?", :count=>"XXXXXX"},  
{:title=>"?", :count=>"XXXXXX"},  
{:title=>"?", :count=>"XXX"},  
...]
```

:title => "ガス事業法第2条",
:count => "3"

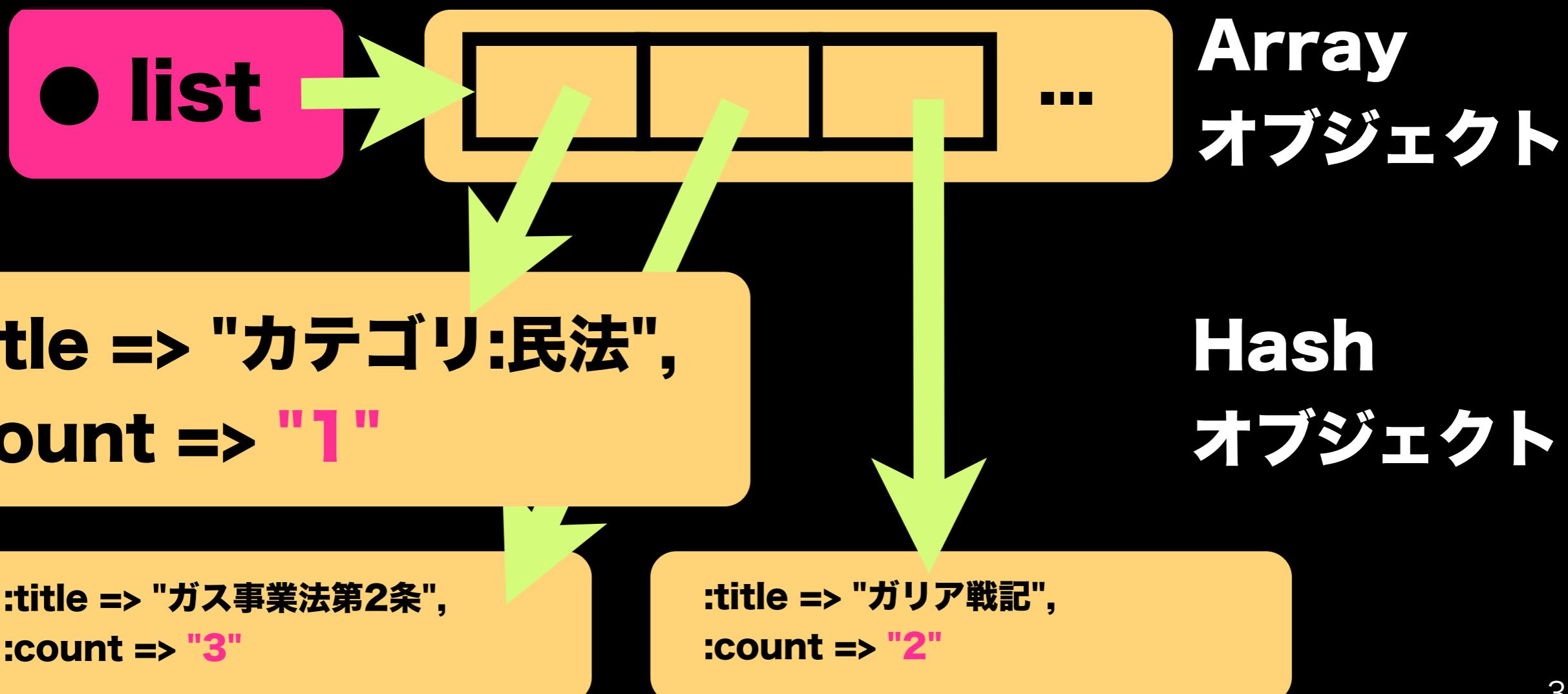
:title => "ガリア戦記",
:count => "2"

Hash
オブジェクト

count順に並べ変えるには

```
[{:title=>"カテゴリ:民法", :count=>"1"},  
 {:title=>"ガス事業法第2条", :count=>"3"},  
 {:title=>"ガリア戦記", :count=>"2"}]  
list(Array)の中身について  
{:count}を調べて並びかえる
```

変数



Array
オブジェクト

:title => "カテゴリ:民法",

:count => "1"

Hash
オブジェクト

:title => "ガス事業法第2条",
:count => "3"

:title => "ガリア戦記",
:count => "2"

count順並び替えソースコード

```
# count順にソート
result = list.sort_by do |i|
  i[:count].to_i
end
```

詳細を説明していきます。

Array#sort_by メソッド

Arrayの中身をブロック(後述)の評価結果で並び替え

サンプルコード

```
array = [5,1,3]
result = array.sort_by do |i|
  i
end
p result
```

[1,3,5]

実行結果

ブロック

do ~ end で書かれる処理のかたまり

桃色の部分がブロック

```
array.sort_by do |i|
```

```
i
```

```
end
```

以前出てきた each メソッドについてたのも実はブロックです。

```
array.each do |i|
```

```
  puts i
```

```
end
```

メソッド+ブロック

`Array#sort_by` や `Array#each` は

ブロックを添えて呼び出します。

ブロックを添えて呼び出す `Array` のメソッドは

大抵 `Array` の全要素について繰り返し実行します。

```
array.sort_by do |i|
```

```
i
```

```
end
```

`sort_by` : 全要素をブロックの評価結果で並び替え

```
array.each do |i|
```

```
  puts i
```

```
end
```

`each` : 全要素についてブロックを実行

ブロックの評価結果

`sort_by`は「Arrayをブロックの評価結果で並び替え」するメソッドです。ブロックの評価結果はブロックで最後に実行された文になります。

```
array.sort_by do |i|
  i
end
```

←ブロックで最後に実行された文が
ブロックの評価結果

Array#sort_by メソッド

Arrayの中身をブロックの評価結果で並び替え

サンプルコード

```
array = [5,1,3]
result = array.sort_by do |i|
  i
end
p result
```

1回目 i = 5 評価結果 5
2回目 i = 1 評価結果 1
3回目 i = 3 評価結果 3

[1,3,5]

実行結果

count順並び替えソースコード

もしも変数listが以下の3つしか入っていないとすると

```
[{:title=>"カテゴリ:民法", :count=>"1"},  
 {:title=>"ガス事業法第2条", :count=>"3"},  
 {:title=>"ガリア戦記", :count=>"2"}]
```

```
result = list.sort_by do |i|  
   i[:count].to_i  
end
```

1回目の i → {:title=>"カテゴリ:民法", :count=>"1"}

2回目の i → {:title=>"ガス事業法第2条", :count=>"3"}

3回目の i → {:title=>"ガリア戦記", :count=>"2"}

count順並び替えソースコード

```
result = list.sort_by do |i|
  i[:count].to_i
end
```

i[:count]はHashの :count キーの値を読み出し
i[:count]には文字列オブジェクト "1" が入っているので、
to_i で整数オブジェクト 1 にする

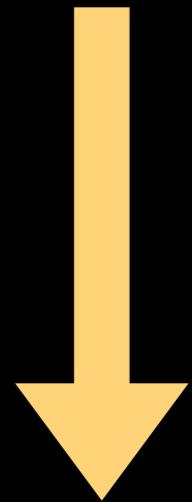
	ブロック	評価結果
変数 i		
1回目	{:title=>"カテゴリ:民法", :count=>"1"}	1
2回目	{:title=>"ガス事業法第2条", :count=>"3"}	3
3回目	{:title=>"ガリア戦記", :count=>"2"}	2

count順並び替えソースコード

```
[{:title=>"カテゴリ:民法", :count=>"1"},  
 {:title=>"ガス事業法第2条", :count=>"3"},  
 {:title=>"ガリア戦記", :count=>"2"}]
```

変数list

```
result = list.sort_by do |i|  
 i[:count].to_i  
end
```



```
[{:title=>"カテゴリ:民法", :count=>"1"},  
 {:title=>"ガリア戦記", :count=>"2"},  
 {:title=>"ガス事業法第2条", :count=>"3"}]
```

変数result

ただ、これだとカウントが少ない順なので、後で逆順にします。

Wikipediaのアクセス数解析

コードで書く際の処理の流れ

- ・データファイルを開く
- ・データファイルから1行読み込む
- ・日本語データ以外はパス
- ・データ1行からタイトルとカウントを取得
- ・取得データをいれものに詰めてとておく
- ・データファイル全行について繰り返し
- ・データファイルを閉じる
- ・貯まったデータをカウント順にソート（並べ替え）
- ・トップ20件表示

ここができた

Wikipediaのアクセス数解析

コードで書く際の処理の流れ

- ・データファイルを開く
- ・データファイルから1行読み込む
- ・日本語データ以外はパス
- ・データ1行からタイトルとカウントを取得
- ・取得データをいれものに詰めてとておく
- ・データファイル全行について繰り返し
- ・データファイルを閉じる
- ・貯まったデータをカウント順にソート（並べ替え）
- ・トップ20件表示

最後の部分です

Wikipediaのアクセス数解析

```
# encoding: utf-8
require "cgi"
filename = "20120301-000000-ja.txt"
file = File.open(filename, "r:UTF-8")
list = []
while text = file.gets
begin
  next unless text =~ /^ja/
  data = text.split
  h = { :title => CGI.unescape(data[1]), :count => data[-2] }
  list.push h
rescue Exception => e
  #p e
end
end
file.close
```

```
# count順にソート
result = list.sort_by do |i|
  i[:count].to_i
end
```

```
# トップ20表示
result.reverse.first(20).each do |i|
  puts i
end
```

ここができた

Wikipediaのアクセス数解析

```
# encoding: utf-8
require "cgi"
filename = "20120301-000000-ja.txt"
file = File.open(filename, "r:UTF-8")
list = []
while text = file.gets
begin
  next unless text =~ /^ja/
  data = text.split
  h = { :title => CGI.unescape(data[1]), :count => data[-2] }
  list.push h
rescue Exception => e
  #p e
end
end
file.close

# count順にソート
result = list.sort_by do |i|
  i[:count].to_i
end

# トップ20表示
result.reverse.first(20).each do |i|
  puts i
end
```

いよいよ最後の部分

トップ20表示コード

```
# トップ20表示  
result.reverse.first(20).each do |i|  
  puts i  
end
```

詳細を説明していきます。

ランキングを出す戦略1

```
[{:title=>"?", :count=>"1"},  
{:title=>"?", :count=>"1"},  
...  
{:title=>"?", :count=>"XXX"},  
{:title=>"?", :count=>"XXXX"},  
{:title=>"?", :count=>"XXXXX"}]
```

現在の変数**result**の中身
countの少ない順にソート
された**Array**オブジェクト

↓ **count**の多い順にしたい
= 全要素を逆順にする

```
[{:title=>"?", :count=>"XXXXXX"},  
{:title=>"?", :count=>"XXXX"},  
{:title=>"?", :count=>"XXX"},  
...  
{:title=>"?", :count=>"1"},  
{:title=>"?", :count=>"1"}]
```

調べてみよう

Arrayオブジェクトの全要素を逆順にするには？

Rubyのリファレンスマニュアルでメソッドを探せます。
例えば以下のページでリファレンスマニュアルを検索できます。

<http://miyamae.github.com/rubydoc-ja/2.0.0/>
or 「サクサク Ruby リファレンス」でgoogle検索

The screenshot shows the Ruby 1.9.3 Reference Manual website. The search bar at the top contains the word "Array". The main content area displays the documentation for the Array class. The title "Ruby 1.9.3 リファレンスマニュアル" is visible above the main content. The Array class definition is shown in a blue header: "オブジェクト指向スクリプト言語 Ruby リファレンスマニュアル". Below this, a list of methods is provided:

- Ruby オフィシャルサイト <http://www.ruby-lang.org/ja/>
- version 1.9 対応リファレンス
- 原著：まつもとゆきひろ
- 最新版URL: <http://www.ruby-lang.org/ja/documentation/>

The left sidebar lists several other methods:

- Array
- <Enumerable
- 配列クラスです。配列は任意の Ruby オブジェクトを要素として
- Array#&
- self & other -> Array
- 集合の積演算です。両方の配列に含まれる要素からなる新しい配列
- Array#*
- self * times -> Array
- 配列の内容を times 回 繰り返した新しい配列を作成し返します。
- Array#+
- self + other -> Array
- 自身と other の内容を繋げた配列を生成して返します。
- Array#-
- self - other -> Array
- 自身から other の要素を取り除いた配列を生成して返します

リファレンスマニュアル検索



1. 画面左上の検索窓に「Array」と入力します。
2. 表示候補の最初の行「Array」をクリックします。

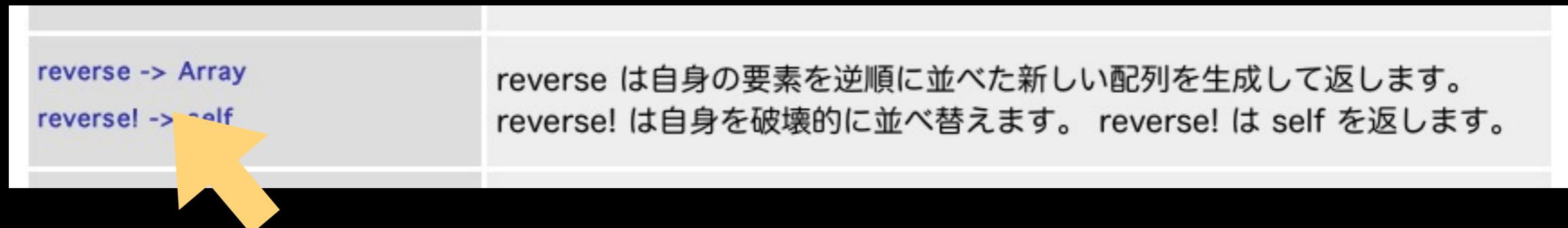
3. 右側にArrayのメソッド一覧などの説明が表示されます。

A screenshot of the Ruby 1.9.3 Reference Manual page for the 'Array' class. The title 'class Array' is prominently displayed in a large blue header. Below the title, the text 'クラスの継承リスト: Array < Enumerable < Object < Kernel < BasicObject' (Inheritance list: Array < Enumerable < Object < Kernel < BasicObject) is shown. A section titled '要約' (Summary) contains the text: '配列クラスです。配列は任意の Ruby オブジェクトを要素として持つことができます。一般的には配列は配列式を使って' (An array class. An array can hold any Ruby object as an element. Generally, arrays are used as array literals). The left side of the screen shows the search results for 'Array' from the previous step.

下の方へ探していくと・・・

リファレンスマニュアル検索

メソッド名と、そのメソッドの説明が並んでます。



4. 使えそうなメソッドのあたりをつけたらメソッド名をクリック

A screenshot of the Ruby 1.9.3 Reference Manual. The URL in the address bar is 'Ruby 1.9.3 リファレンスマニュアル > ライブラリ一覧 > 組み込みライブラリ > Arrayクラス > reverse'. The title of the page is 'instance method Array#reverse'. Below the title, it says 'reverse -> Array' and 'reverse! -> self'. The explanatory text reads: 'reverse は自身の要素を逆順に並べた新しい配列を生成して返します。 reverse! は自身を破壊的に並べ替えます。 reverse! は self を返します。' At the bottom, there are two code snippets:

```
a = ["a", 2, true]
p a.reverse      #=> [true, 2, "a"]
p a              #=> ["a", 2, true] (変化なし)

a = ["a", 2, true]
p a.reverse!
p a              #=> [true, 2, "a"]
```

メソッド名

説明

サンプルコード

reverseを使えばできそうです。

リファレンスマニュアル検索

5. シンプルなコード例を作ってirbで試してみる

\$ irb

[3,2,1].reverse

[1,2,3]

実行結果

reverseを使えばできそうです。

「逆引きRuby」ページもオススメ

<http://www.namaraii.com/rubytips/>

- 配列
 - プログラムで配列を定義する
 - 配列要素をカンマ区切りで出力する
 - 配列の要素数を取得する
 - 配列に要素を追加する
 - 配列の先頭または末尾から要素を取りだす
 - 部分配列を取りだす
 - 配列を任意の値で埋める
 - 配列を空にする
 - 配列同士を結合する
 - 配列同士の和・積を取る
 - 複数の要素を変更する
 - 配列の配列をフラットな配列にする
 - 配列をソートする
 - 条件式を指定したソート
 - 配列を逆順にする
 - 指定した位置の要素を取り除く
 - 一致する要素を全て取り除く
 - 配列から重複した要素を取り除く
 - 配列から指定条件を満たす要素を取り除く
 - 配列から指定条件を満たす要素を抽出する
 - 配列中の要素を探す
 - 配列の配列を検索する
 - 配列の各要素にブロックを実行し配列を作成する
 - 配列の各要素に対して繰り返しブロックを実行する
 - 配列の要素をランダムに抽出する
 - 実行するたびに取り出される要素が異なります。

配列

配列を使うためにはArrayクラスを使います。

プログラムで配列を定義する

プログラム内で配列を定義するには以下のように記述します。

```
fruits = ["apple", "orange", "lemon"]
scores = [55, 49, 100, 150, 0]
```

配列を参照する場合はArray#[]メソッドを使い、引数として配列の要素番号を指定します。要素番号は0から始まります。上の例ではfruits[0]は"apple"、scores[3]は150を返します。

以下のように配列をネスト（入れ子）にすることもできます。

```
fruits = [3, ["apple", 250], ["orange", 400], ["lemon", 300]]
p fruits[0] #=> 3
p fruits[1][1] #=> 250
p fruits[3][0] #=> "lemon"
```

配列要素をカンマ区切りで出力する

Array#joinメソッドを使うと配列の要素を任意の文字列で区切った文字列を取得することができます。

```
p ["apple", "orange", "lemon"].join(',') #=> "apple,orange,lemon"
p ["apple", "orange", "lemon"].join('#') #=> "apple#orange#lemon"
p [55, 49, 100, 150, 0].join(',') #=> "55,49,100,150,0"
p [3, ["apple", 250], ["orange", 400], ["lemon", 300]].join(',') #=> "3,apple,250,orange,400,lemon,300"
```

プロも使う検索方法

Rubyのオブジェクトに直接聞く方法もあります。

```
$ irb
```

```
[] .methods
```

```
#=> ["zip", "pop", "find_index", "rassoc", "enum_slice", "map!",  
"minmax", "methods", "send", "shuffle!", "replace", ... ]
```

`methods` メソッドで使えるメソッド一覧が表示されます。英単語でなんとなく意味が分かるのでこれで見つけるときも多いです。

```
require 'pp'  
pp [].methods.sort
```

`pp` と `sort` を使うと見易いです。

演習

`sort_by` と
リファレンスマニュアルを検索 の演習です。
両方の初級問題から先に解いてみてください。

sort_byの演習

a1. Arrayオブジェクト

```
[{:name => "apple", :price => 50},  
 {:name => "grape", :price => 100},  
 {:name => "orange", :price => 30} ]
```

を:priceが安い順に並び替えて表示するコードを書いてください。

a2. Arrayオブジェクト [3,1,-5] を絶対値が小さい順に並び替えて表示するコードを書いてください。

ヒント：絶対値の取得は `1.abs` メソッド

a3. 【上級】 あるArrayオブジェクト(例えば[3,1,5]とか[1,2,3]とか)が小さい順に並び替え済みかどうか判定するコードを書いてください。

a4. 【上級】 要素に重複の無いArray(例：[3,4,2,1,5])をsort_byを使い、reverseと同様の結果を得られるコードを書いて下さい。

リファレンスマニュアル検索演習

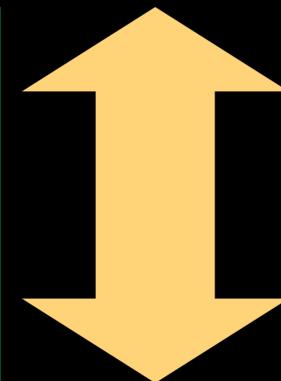
- b1. `["080","1234","5678"]` から
"080-1234-5678" という文字列を作るメソッドをリファレンスマニュアルから探して、コードを作って実行してください。
- b2. 【上級】 `[1,2,3]` を、各要素を3倍した値で置き換えるメソッドをリファレンスマニュアルから探し、コードを書いて実行してください。

ランキングを出す戦略2

reverseを使って逆順にできました。

```
[{:title=>"?", :count=>"XXXXX"},  
 {:title=>"?", :count=>"XXXX"},  
 {:title=>"?", :count=>"XXX"},  
 ...  
 {:title=>"?", :count=>"1"},  
 {:title=>"?", :count=>"1"}]
```

全部で
数万行



先頭から20個

全部表示するのは大変なので、先頭20件だけ取得したい

リファレンスマニュアルで検索してみましょう。

`first(n) -> Array`

先頭の n 要素を配列で返します。n は 0 以上でなければなりません。

Ruby 1.9.3 リファレンスマニュアル > ライブラリー一覧 > 組み込みライブラリ > `Array` クラス > `first`

instance method `Array#first`

`first -> object | nil`

配列の先頭の要素を返します。要素がなければ `nil` を返します。

```
p [0, 1, 2].first #=> 0  
p [].first      #=> nil
```

[SEE ALSO] `Array#last`

`first(n) -> Array`

先頭の n 要素を配列で返します。n は 0 以上でなければなりません。

[PARAM] n:

取得したい要素の個数を整数で指定します。

[EXCEPTION] `ArgumentError`:

n が負値の場合発生します。

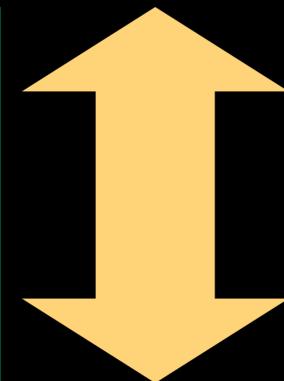
```
ary = [0, 1, 2]  
p ary.first(0)  
p ary.first(1)  
p ary.first(2)  
p ary.first(3)  
p ary.first(4)  
# => []  
[0]  
[0, 1]  
[0, 1, 2]  
[0, 1, 2]
```

これが
使えそう

ランキングを出す戦略2

firstを使って最初の20個を取得できました。

```
[{:title=>"?", :count=>"XXXXX"},  
 {:title=>"?", :count=>"XXXX"},  
 {:title=>"?", :count=>"XXX"},  
 ...]
```



先頭から20個

20個の要素を持つArrayオブジェクトができたので、
前にやったeach文で繰り返し処理を行い、
各要素を表示すればランキングを表示できそうです。

では、ここまで処理をつなげてみましょう。

トップ20表示コード

```
result.reverse.first(20).each do |i|
  puts i
end
```

このようにつなげて書くことができます。

reverseメソッドも、

firstメソッドも、

Arrayオブジェクトを返すからです。

このような書き方をメソッドチェインと言います。

Wikipediaのアクセス数解析

コードで書く際の処理の流れ

- ・データファイルを開く
- ・データファイルから1行読み込む
- ・日本語データ以外はパス
- ・データ1行からタイトルとカウントを取得
- ・取得データをいれものに詰めてとっておく
- ・データファイル全行について繰り返し
- ・データファイルを閉じる
- ・貯まったデータをカウント順にソート（並べ替え）
- ・トップ20件表示

全部できました！

Wikipediaのアクセス数解析

```
# encoding: utf-8
require "cgi"
filename = "20120301-000000-ja.txt"
file = File.open(filename, "r:UTF-8")
list = []
while text = file.gets
begin
  next unless text =~ /^ja/
  data = text.split
  h = { :title => CGI.unescape(data[1]), :count => data[-2] }
  list.push h
rescue Exception => e
  #p e
end
end
file.close
```

```
# count順にソート
result = list.sort_by do |i|
  i[:count].to_i
end
```

```
# トップ20表示
result.reverse.first(20).each do |i|
  puts i
end
```

では、実行してみましょう！

Wikipediaのアクセス数解析演習

c1. 前述のコードを実行してください。.rbファイルと同じフォルダにデータファイルを置いて実行してください。

3行目↓をデータファイルの名前に変更する必要があります。

```
filename = "20120301-000000-ja.txt"
```

データファイルは2つ用意しています。

まずは小さい方でコードがうまく動くことを確認してみましょう。

zip形式で圧縮してあるので、解凍して利用してください。

(Winはファイル選択して右クリックメニューから「すべて展開」、Macはファイルをダブルクリック)

20120301-000000-ja.txt : 練習用の小さいデータ

<http://bit.ly/wpdatamini> からダウンロードできます。

pagecounts-20130503-040000 : DLできる実際のデータ

<http://bit.ly/wpdata2013> からダウンロードできます。

※Winで出力が "\u4FDD\u5143\u306E\u4E71" となる人は10行目付近を以下のコードで差し替えてください。cp932は文字コードの一種です。

```
旧 : h = {:title => CGI.unescape(data[1]), :count => data[-2]}
```

```
新 : h = {:title => CGI.unescape(data[1]).encode("cp932"), :count => data[-2]}
```

Wikipediaのアクセス数解析演習

c2. 【上級】データを配布しているサイトから任意のデータをダウンロードして解析してください。

<http://dumps.wikimedia.org/other/pagecounts-raw/>

ヒント：.gz形式で圧縮してあるので、解凍が必要です。

VM, Mac の場合はターミナルから \$ gunzip ファイル名 で解凍できます。

Windows の場合は例えばLhaplusを使って解凍できます。

<http://www.forest.impress.co.jp/lib/arc/archive/archiver/lhaplus.html>

解答したファイルを .rb ファイルと同じフォルダに配置し、

コード3行目のfilenameを解凍したファイル名に変更してください。

まとめ

ブロック

do ~ end で書かれる処理のかたまり

桃色の部分がブロック

```
array.sort_by do |i|
```

```
i
```

```
end
```

以前出てきた each メソッドについてたのも実はブロックです。

```
array.each do |i|
```

```
  puts i
```

```
end
```

メソッド + ブロック

`Array#sort_by` や `Array#each` は

ブロックを添えて呼び出します。

ブロックを添えて呼び出す `Array` のメソッドは

大抵 `Array` の全要素について繰り返し実行します。

```
array.sort_by do |i|
```

```
i
```

```
end
```

`sort_by` : 全要素をブロックの評価結果で並び替え

```
array.each do |i|
```

```
  puts i
```

```
end
```

`each` : 全要素についてブロックを実行

ブロックの評価結果

`sort_by`は「Arrayをブロックの評価結果で並び替え」するメソッドです。ブロックの評価結果はブロックで最後に実行された文になります。

```
array.sort_by do |i|
  i
end
```

←ブロックで最後に実行された文が
ブロックの評価結果

Array#sort_by メソッド

Arrayの中身をブロックの評価結果で並び替え

サンプルコード

```
array = [5,1,3]
result = array.sort_by do |i|
  i
end
p result
```

1回目 i = 5 評価結果 5
2回目 i = 1 評価結果 1
3回目 i = 3 評価結果 3

[1,3,5]

実行結果

リファレンスマニュアル検索



1. 画面左上の検索窓に「Array」と入力します。
2. 表示候補の最初の行「Array」をクリックします。

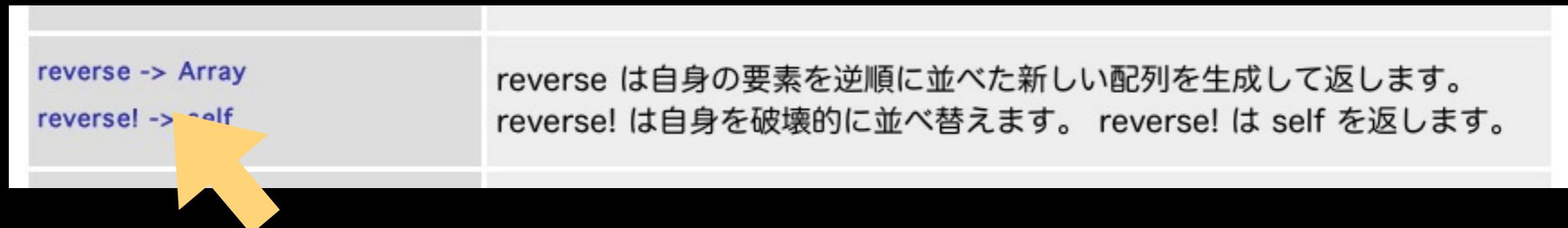
3. 右側にArrayのメソッド一覧などの説明が表示されます。

A screenshot of the Ruby 1.9.3 Reference Manual page for the 'Array' class. The title 'class Array' is prominently displayed in a large blue header. Below the title, the text 'クラスの継承リスト: Array < Enumerable < Object < Kernel < BasicObject' (Inheritance list: Array < Enumerable < Object < Kernel < BasicObject) is shown. A section titled '要約' (Summary) contains the text: '配列クラスです。配列は任意の Ruby オブジェクトを要素として持つことができます。一般的には配列は配列式を使って' (An array class. An array can hold any Ruby object as an element. Generally, arrays are used as array literals). The left side of the screen shows the search results for 'Array' from the previous step.

下の方へ探していくと・・・

リファレンスマニュアル検索

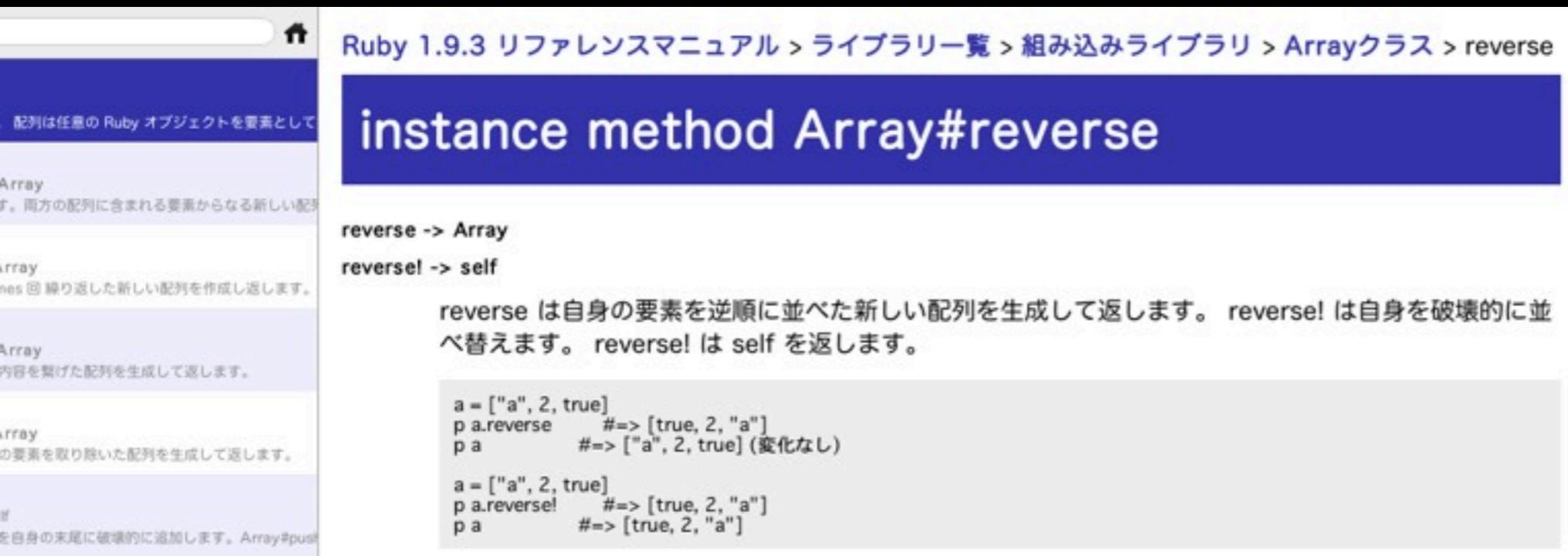
メソッド名と、そのメソッドの説明が並んでます。



reverse -> Array
reverse! -> self

reverse は自身の要素を逆順に並べた新しい配列を生成して返します。
reverse! は自身を破壊的に並べ替えます。 reverse! は self を返します。

4. 使えそうなメソッドのあたりをつけたらメソッド名をクリック



Ruby 1.9.3 リファレンスマニュアル > ライブラリー一覧 > 組み込みライブラリ > Arrayクラス > reverse

instance method Array#reverse

reverse -> Array
reverse! -> self

reverse は自身の要素を逆順に並べた新しい配列を生成して返します。 reverse! は自身を破壊的に並べ替えます。 reverse! は self を返します。

```
a = ["a", 2, true]
p a.reverse      #=> [true, 2, "a"]
p a              #=> ["a", 2, true] (変化なし)

a = ["a", 2, true]
p a.reverse!
p a              #=> [true, 2, "a"]
```

メソッド名

説明

サンプルコード

reverseを使えばできそうです。

リファレンスマニュアル検索

5. 分かり易いパターンでirbで試してみる

\$ irb

[3,2,1].reverse

[1,2,3]

実行結果

reverseを使えばできそうです。

演習回答

sort_byの演習

a1. Arrayオブジェクト

```
[{:name => "apple", :price => 50},  
 {:name => "grape", :price => 100},  
 {:name => "orange", :price => 30} ]
```

を:price安い順に並び替えて表示するコードを書いてください。

```
array = [{:name => "apple", :price => 50}, {:name =>  
 "grape", :price => 100}, {:name => "orange", :price =>  
 30}]  
  
result = array.sort_by do |i|  
   i[:price]  
end  
  
p result
```

sort_byの演習

a2. Arrayオブジェクト [3,1,-5] を絶対値が小さい順に並び替えて表示するコードを書いてください。

ヒント：絶対値の取得は `i.abs` メソッド

```
array = [3,1,-5]
result = array.sort_by do |i|
  i.abs
end
p result
```

sort_byの演習

a3. 【上級】 あるArrayオブジェクト(例えば[3,1,5]とか[1,2,3]とか)が小さい順に並び替え済みかどうか判定するコードを書いてください。

```
array = [3,1,5]
result = array.sort_by do |i|
  i
end
p result == array
```

Arrayオブジェクトは == 演算子で同じかどうか判定できます。
sort_by は対象のオブジェクトそのものを並び替えるのではなく、
対象のオブジェクトを並び替えた新しいオブジェクトを返します。
なので、array の中身はsort_by 後も変わらず [3,1,5] です。
対象のオブジェクトを並び替える sort_by! メソッドもあります。
対象のオブジェクトを書き換えるメソッドは破壊的メソッドと呼び、
メソッド名末尾に ! がつくものが多いです。

sort_byの演習

a4. 【上級】 要素に重複の無いArray(例：[3,4,2,1,5])をsort_byを使い、reverseと同様の結果を得られるコードを書いて下さい。

```
array = [3,4,2,1,5]
result = array.sort_by do |i|
  -array.index(i)
end
p result
```

array.index(value) はarrayの中のvalueが入っている位置を返します。

[2,3,5,7,11].index(2) →0

[2,3,5,7,11].index(3) →1

[2,3,5,7,11].index(5) →2

という具合です。

リファレンスマニュアル検索演習

b1. ["080","1234","5678"] から

"080-1234-5678" という文字列を作るメソッドをリファレンスマニュアルから探して、コードを書いて実行してください。

```
["080","1234","5678"].join("-")
```

b2. 【上級】 [1,2,3] を、各要素を3倍した値で置き換えるメソッドをリファレンスマニュアルから探し、コードを作って実行してください。

```
[1,2,3].map! do |i|
  i * 3
end
```

参考文献

ウィキペディア記事閲覧回数の特徴分析
人工知能学会研究会資料 SIG-SWO-A901-03
曾根 広哲、山名 早人
早稲田大学大学院基幹理工学研究科 早稲田大学理工学術院 国立情報学研究所
<http://sigsw.org/papers/SIG-SWO-A901/SIG-SWO-A901-03.pdf>

參考資料

書式

Rubyコード

`puts "abc"`

実行結果

"abc"

実行結果

shellコマンド

`$ ls`

課題チェック用の付箋の書き方

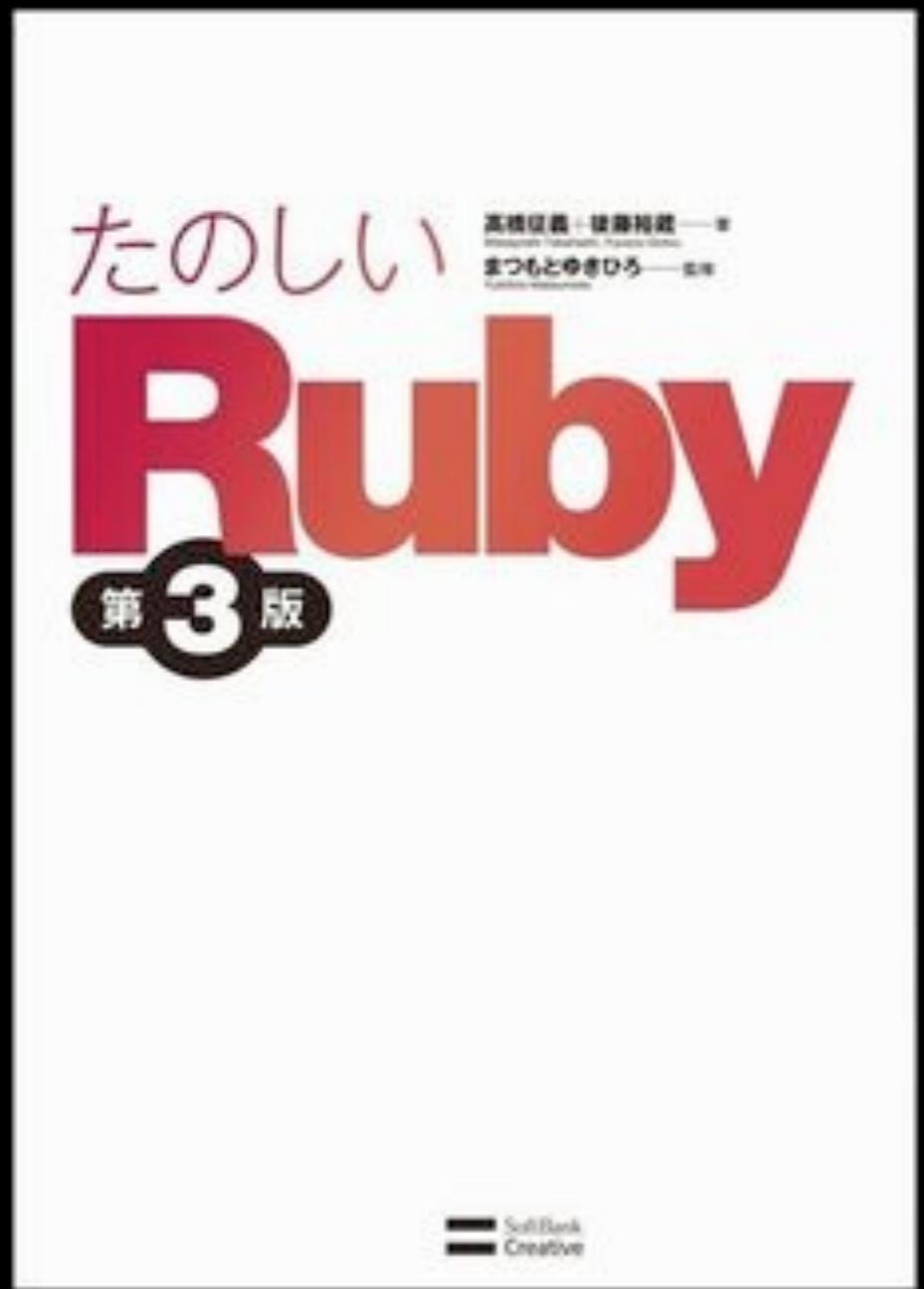
上の方に学籍番号と名前を書いてください

学籍番号

名前

※この辺に講師陣がクリアした課題番号を書いていきます。

教科書：たのしいRuby



<http://www.amazon.co.jp/dp/4797357401/>



お買い求めは
大学生協または
ジュンク堂池袋店で

講義資料置き場

過去の資料がDLできます。

<https://github.com/igaiga/hitotsubashi-ruby-2013>

雑談・質問用facebookグループ

<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
- 参加者はスタッフ(講師・TA)と昨年、今年の受講者です
- 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