

# Ruby 講義

## 第8回 各種演習

五十嵐邦明

twitter : igaiga555

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



2013.6.6 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>

# 目次

## 演習特集

Wikipediaアクセス解析

演習時間1

演習時間2

演習の解答と解説

**演習の前に**

**つづけ**

**Rubyの説明**

# 破壊的メソッドの説明

## upcase と upcase!

`String#upcase` と `String#upcase!` はどちらも文字列を大文字にするメソッドです。

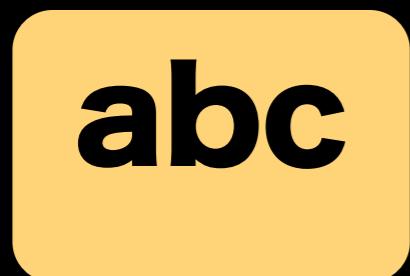
irb で実行するとどちらも同じように見えますが、`!` の有無で何が違うのでしょうか？

```
"abc".upcase #=> "ABC"
```

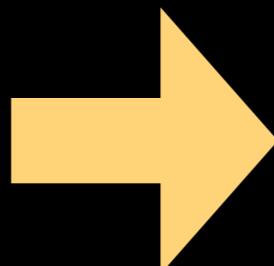
```
"abc".upcase! #=> "ABC"
```

# "abc".upcase!

実行前



upcase!



実行後



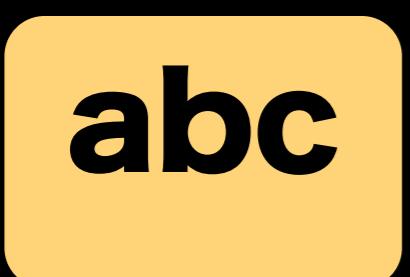
オブジェクト

オブジェクト

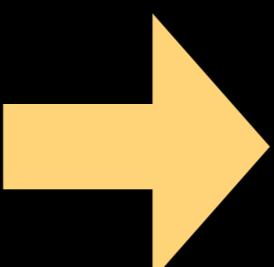
オブジェクト  
が変身する

# "abc".upcase

実行前



upcase



実行後



旧オブジェクト



変換した  
新しい  
オブジェクト  
が複製される

オブジェクト

新オブジェクト

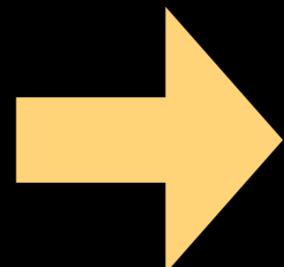
# upcase! の使い方

"abc".upcase!

実行前

abc

upcase!



実行後

ABC

オブジェクト  
が変身する

オブジェクト

オブジェクト

a = "abc"

a

abc



a.upcase!

a

ABC

変数aが指すオブジェクトが"abc"から"ABC"に変身  
変数aをそのまま使う

# upcase の使い方

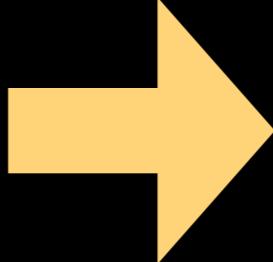
"abc".upcase

実行前

abc

オブジェクト

upcase



実行後

abc

旧オブジェクト

ABC

新オブジェクト

a = "abc"

a

abc

b = a.upcase

a

abc

b

ABC

複製されたオブジェクトを別の変数(ここではb)に入れると

変換した  
新しい  
オブジェクト  
が複製される

# 破壊的メソッド

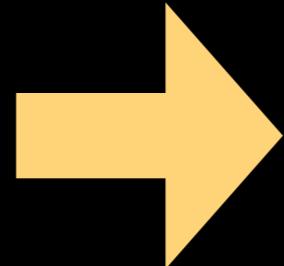
`upcase!` のようにオブジェクトの内容を  
変更するものを「破壊的メソッド」と言います

`"abc".upcase!`

実行前

abc

upcase!



実行後

ABC

オブジェクト  
が変身する

オブジェクト

オブジェクト

メソッド名末尾に `!` がついたら破壊的メソッド  
(ただし、破壊的メソッドでも `!` が付かないものもある)

**Wikipedia**

**アクセス数解析**

**復習**

# 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

- ①データファイルを開く
- ②データファイルから1行読み込む
- ③日本語データ以外はパス
- ④データ1行からタイトルとカウントを取得
- ⑤取得データをいれものに詰めてとっておく
- ⑥データファイル全行について繰り返し
- ⑦データファイルを閉じる
- ⑧貯まったデータをカウント順にソート（並べ替え）
- ⑨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
```

# 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ファイルとデータファイルは同じフォルダに置いてください。
- ・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>)

**pagecounts-20130503-040000**

DLできる実際のデータ(<http://bit.ly/wpdata2013>)

※Winで出力が \u4FDD\u5143\u306E... のようになる場合は10行目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。

# 演習問題集

どちらから解いてもOKです。

# 演習問題 1

**Arrayオブジェクトの要素が奇数の項目を全て足すコードを書いてください。**

**例えば array = [2,3,5,7,11] の場合、  
(3+5+7+11=) 26 が表示されればOKです。**

**ヒント：奇数かどうか調べるのは Fixnum#odd? メソッド  
ちなみに偶数か調べるのは Fixnum#even? メソッドです。**

**1.odd? #=> true**

**2.odd? #=> false**

# 演習問題 2

```
[{:title => "a", :price => 70},  
 {:title => "b", :price => 200},  
 {:title => "c", :price => 50}]
```

というオブジェクトから、以下のようなオブジェクトを作るコードを書いてください。

```
[{:title=>"a", :price=>70, :special=>"Low price!"},  
 {:title=>"b", :price=>200},  
 {:title=>"c", :price=>50, :special=>"Low price!"}]
```

# 演習問題 3

あるArrayオブジェクトが与えられたとき、(例えば [6,2,3]) その中で3以下の数字がいくつあるか表示するコードを書いてください。

# 演習問題 4

Hash のバリューの中に！という文字が含まれるときに、そのバリューを大文字に変換した Hash を作るコードを書いてください。

例えば、

```
{:alice=>"yeah", :bob=>"yo!", :linda => "wow!" }
```

↑ という Hash を ↓ にできればOKです。

```
{:alice=>"yeah", :bob=>"YO!", :linda => "WOW!" }
```

# 演習問題 5

文字列 "write" 中の e を ten に置換  
し、"written"にするコードを書いてください。

ヒント：文字列の置換は String#gsub! を使います。

# 演習問題 6

**text1 = "123"**

**text2 = "55"**

**text3 = "1024"**

という3つの文字列オブジェクトがあるとき、数値として最も大きいものを表示するコードを書いてください。

(この場合、1024 を表示)

考え方の一例：

arrayを作り、この3つを数値オブジェクトに変換して格納して、maxメソッドを呼ぶと最も大きい数値を返します。

ヒント：文字列オブジェクトを数値オブジェクトにするのはto\_iメソッド  
"123".to\_i #=> 123

# 演習問題 7

引数にハッシュを受け取るメソッドを書いてください。そのメソッドの中で、引数で受け取ったハッシュにキー :text がなかったとき、“This object does not have :text key.” と出力するコードを書いてください。

(例えば、メソッド名を `print_text` として、メソッド呼び出し側は以下のようになります。)

```
print_text({:title => "the Ruby book"})
```

# 演習問題 8

alice,bob,carolの試験の点数が次のようにHashに格納されています。

alice = {**:english => 90, :math => 60, :history => 75**}

bob = {**:english => 50, :math => 90, :history => 80**}

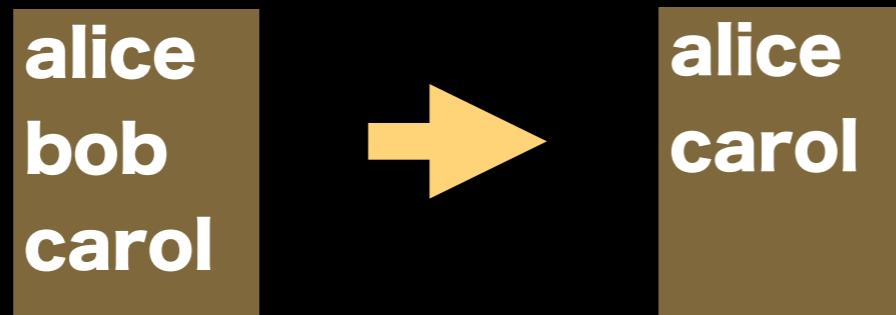
carol = {**:english => 18, :math => 50, :history => 100**}

各科目の平均点を出力してください

# 演習問題【上級】S1

あるテキストファイルから、`a`という文字列を含む行だけを抽出した別のテキストファイルを作ってください。

例：



ヒント："sentence" という文字列が書かれたテキストファイルを作るのは以下のコードになります。

```
out_filename = 'out.txt'  
out_file = File.open(out_filename, 'w:UTF-8')  
out_file.puts "sentence"  
out_file.close
```

# 演習問題【上級】S2

日本の都道府県で、男性と女性の人口差が最も大きいのはどこか、データから解析してください。

ヒント：データは以下にあります。

<http://www.e-stat.go.jp/SG1/estat>List.do?id=000001109855>  
男女別人口及び世帯の種類(2区分)別世帯数 の CSV

ヒント：文字例をカンマで区切ってArrayにするには  
**String#split** を使います。

"a,b,c".split(",") #=> ["a","b","c"]

# 演習解答

解答は一例です。Rubyのコードはいろいろな書き方が可能です。ここに挙げたコードだけが正解ではないです。大切なのは、読み手への心配りと思いやりです。

# 記法の説明

ときどきでてくる

#=>

というマークは実行結果を表します。

p 1+2 #=> 3

p array #=> [1,2,3]

コードではないので打たなくて大丈夫です。もし打ったとしても、#から始まる箇所はコメントとして扱われる  
ので、何も起きません。

# 演習問題 1

Arrayオブジェクトの要素が奇数の項目を全て足すコードを書いてください。  
例えば `array = [2,3,5,7,11]` の場合、  $3+5+7+11 = 26$  が表示されれば  
OKです。

ヒント：奇数かどうか調べるのは `Fixnum#odd?` メソッド  
ちなみに偶数か調べるのは `Fixnum#even?` メソッドです。

1. `i.odd? #=> true`  
2. `i.odd? #=> false`

```
array = [2,3,5,7,11]
sum = 0
array.each do |i|
  sum += i if i.odd?
end
p sum #=> 26
```

1行でかっこよく書くこともできます。（上級者向け）

```
[2,3,5,7,11].select{|i|i.odd?}.inject(:+)
```

# 演習問題 2

```
[{:title => "a", :price => 70},  
 {:title => "b", :price => 200},  
 {:title => "c", :price => 50}]
```

というオブジェクトから、以下のようなオブジェクトを作るコードを書いてください。

```
[{:title=>"a", :price=>70, :special=>"Low price!"},  
 {:title=>"b", :price=>200},  
 {:title=>"c", :price=>50, :special=>"Low price!"}]
```

```
items = [{:title => "a", :price => 70},  
          {:title => "b", :price => 200},  
          {:title => "c", :price => 50}]  
  
items.each do |item|  
  item[:special] = 'Low price!' if item[:price] < 100  
end  
  
p items
```

```
[{:title=>"a", :price=>70, :special=>"Low price!"},  
 {:title=>"b", :price=>200},  
 {:title=>"c", :price=>50, :special=>"Low price!"}]
```

実行結果

# 演習問題 3

あるArrayオブジェクトが与えられたとき、(例えば [6,2,3]) その中で3以下の数字がいくつあるか表示するコードを書いてください。

```
array = [6,2,3]
count = 0
array.each do |i|
  count += 1 if i <= 3
end
p count #=> 2
```

別解 countメソッドにブロックを渡すことでカウントできます。  
ブロックは do end の変わりに {} で書くこともできます。

```
array = [6,2,3]
p array.count { |i| i <= 3 }
```

# 演習問題 4

Hash の値の中に ! という文字が含まれるときに、その値を大文字に変換したHashを作るコードを書いてください。

例えば、

```
{:alice=>"yeah", :bob=>"yo!", :linda => "wow!" }
```

↑ というHash を ↓ にできればOKです。

```
{:alice=>"yeah", :bob=>"YO!", :linda => "WOW!" }
```

```
h = {:alice=>"yeah", :bob=>"yo!", :linda => "wow!" }
h.each do |key, value|
  h[key] = value.upcase if value =~ /!/
end
p h
#=> {:alice=>"yeah", :bob=>"YO!", :linda=>"WOW!"}
```

# 演習問題 5

文字列 "write" 中の e を ten に置換  
し、"written"にするコードを書いてください。  
ヒント：文字列の置換は String#gsub!を使います。

```
"write".gsub!(/e/, "ten") #=> "written"
```

gsub!は破壊的メソッドです。対象のオブジェクトを書き換えます。 !のない gsub というメソッドもあります。 gsubは置換した新しいStringオブジェクトを返します。違いは以下のコードを実行するのが分かり易いです。

```
string = "write"  
string.gsub!(/e/, "ten")  
p string #=> "written"
```

```
string = "write"  
string.gsub(/e/, "ten")  
p string #=> "write"
```

# 演習問題 6

```
text1 = "123"
```

```
text2 = "55"
```

```
text3 = "1024"
```

という3つの文字列オブジェクトがあるとき、数値として最も大きいものを表示するコードを書いてください。 (この場合、1024 を表示)

考え方の一例：

arrayを作つてこの3つを数値として格納して、maxメソッドを呼ぶと最も大きい数値を返します。

ヒント：文字列オブジェクトを数値オブジェクトにするのはto\_iメソッド

```
"123".to_i #=> 123
```

```
text1 = "123"
```

```
text2 = "55"
```

```
text3 = "1024"
```

```
array = []
```

```
array.push text1.to_i
```

```
array.push text2.to_i
```

```
array.push text3.to_i
```

```
p array.max
```

# 演習問題 7

引数にハッシュを受け取るメソッドを書いてください。そのメソッドの中で、引数で受け取ったハッシュにキー :text がなかったとき、"This object does not have :text key." と出力するコードを書いてください。

(例えば、メソッド名を print\_text として、メソッド呼び出し側は以下のようになります。)

```
print_text({:title => "the Ruby book"})
```

```
def print_text(h)
  puts "This object does not have :text key." unless h[:text]
end

print_text({:text => "the Ruby book"}) #=> 出力なし
print_text({:foo => "bar"}) #=> "This object does not have :text key."
```

# 演習問題 8

alice,bob,carolの試験の点数が次のようにHashに格納されています。

alice = {**:english => 90, :math => 60, :history => 75**}

bob = {**:english => 50, :math => 90, :history => 80**}

carol = {**:english => 18, :math => 50, :history => 100**}

各科目の平均点を出力してください

```
alice = {:english => 90, :math => 60, :history => 75}
```

```
bob = {:english => 50, :math => 90, :history => 80}
```

```
carol = {:english => 18, :math => 50, :history => 100}
```

```
puts "english average", (alice[:english]+bob[:english]+carol[:english])/3
```

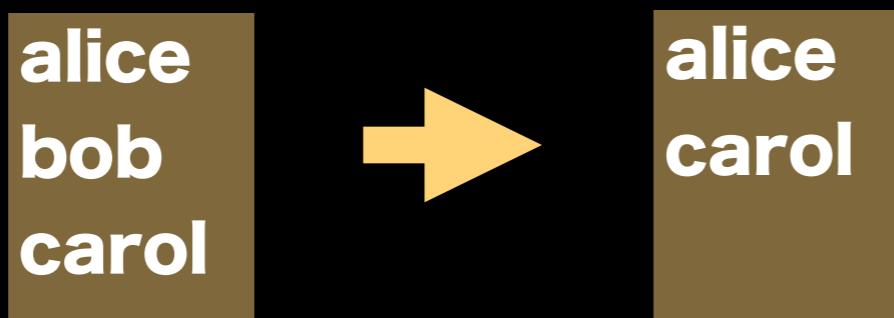
```
puts "math average", (alice[:math]+bob[:math]+carol[:math])/3
```

```
puts "history average", (alice[:history]+bob[:history]+carol[:history])/3
```

# 演習問題【上級】S1 解1

あるテキストファイルから、aという文字列を含む行だけを抽出した別のテキストファイルを作ってください。

例：



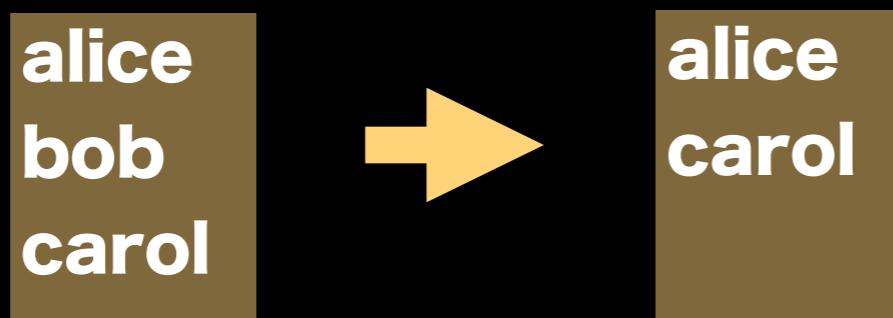
```
# ファイルから読み込み
included = [] # 出力用データを格納するArray
in_filename = 'in.txt'
File.open(in_filename, 'r:UTF-8') do |in_file|
  while text = in_file.gets
    included << text if text =~ /a/
  end
end
# ファイルへ書き込み
out_filename = 'out.txt'
File.open(out_filename, 'w:UTF-8') do |out_file|
  included.each do |i|
    out_file.puts i
  end
end
```

2つのファイルをopenします。  
出力側は"w:UTF-8"(書き込み)  
です。  
また、File.openにブロックを  
渡すとブロック完了時に自動で  
closeします

# 演習問題【上級】S1 解2

あるテキストファイルから、aという文字列を含む行だけを抽出した別のテキストファイルを作ってください。

例：



```
in_filename = 'in.txt'  
out_filename = 'out.txt'  
File.open(in_filename, 'r:UTF-8') do |in_file|  
  File.open(out_filename, 'w:UTF-8') do |out_file|  
    while text = in_file.gets  
      out_file.puts text if text =~ /a/  
    end  
  end  
end
```

in と out を同時に開く書き方です。結果格納用のArrayが不要になるのでスッキリです。

# 演習問題【上級】S2

日本の都道府県で、男性と女性の人口差が最も大きいのはどこか、データから解析してください。

ヒント：データは以下にあります。

<http://www.e-stat.go.jp/SG1/estat>List.do?id=000001109855>

男女別人口及び世帯の種類(2区分)別世帯数 の CSV

ヒント：文字例をカンマで区切ってArrayにするには

`String#split` を使います。

`"a,b,c".split(",") #=> ["a","b","c"]`

私もまだ書いていないので、誰か書けたらコードを私まで送ってください。 :)



# 參考資料

# 書式

Rubyコード

`puts "abc"`

実行結果

"abc"

実行結果

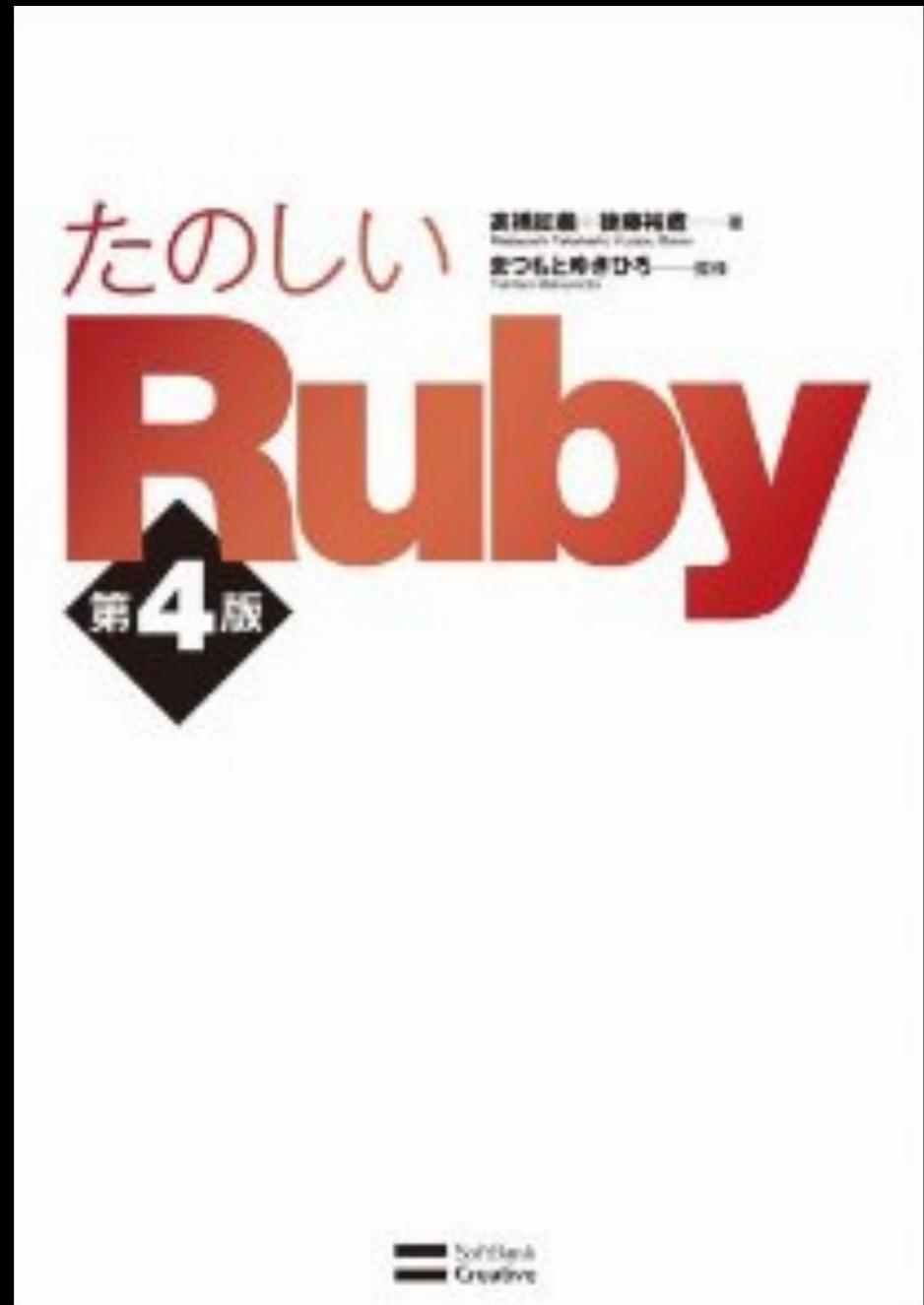
文中では `#=>` で書きます

`p 1+2 #=> 3`

shellコマンド

`$ ls`

# 教科書：たのしいRuby



<http://www.amazon.co.jp/dp/4797372273/>



お買い求めは  
大学生協または  
ジュンク堂池袋店で

# 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。

# 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