Ruby 講義
第8回 Ruby入門

Kuniaki IGARASHI/igaiga

2012.5.31 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)

ニフティ株式会社

English　お問い合わせ　サイト内検索　検索

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…
「スマプレ！」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
「シュフモ」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.

ニフティ株式会社提供

HOME　会社概要　紹介　社会・環境活　ニュースリリー　IR情　採用情

採用情報
募集要項や働く環境はこちら　NIFTY

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

ニュースリリース一覧へ　RSS

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2　24
2012年4月1　プレス　LE LTE 定額にねんプラン」の提供を開始
2012年4月1　ス　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

Teaching Assistant 濱崎 健吾

クックパッド株式会社　エンジニア

先週の

おさらい

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
```

実行結果

```
[1,3,5]
```

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。

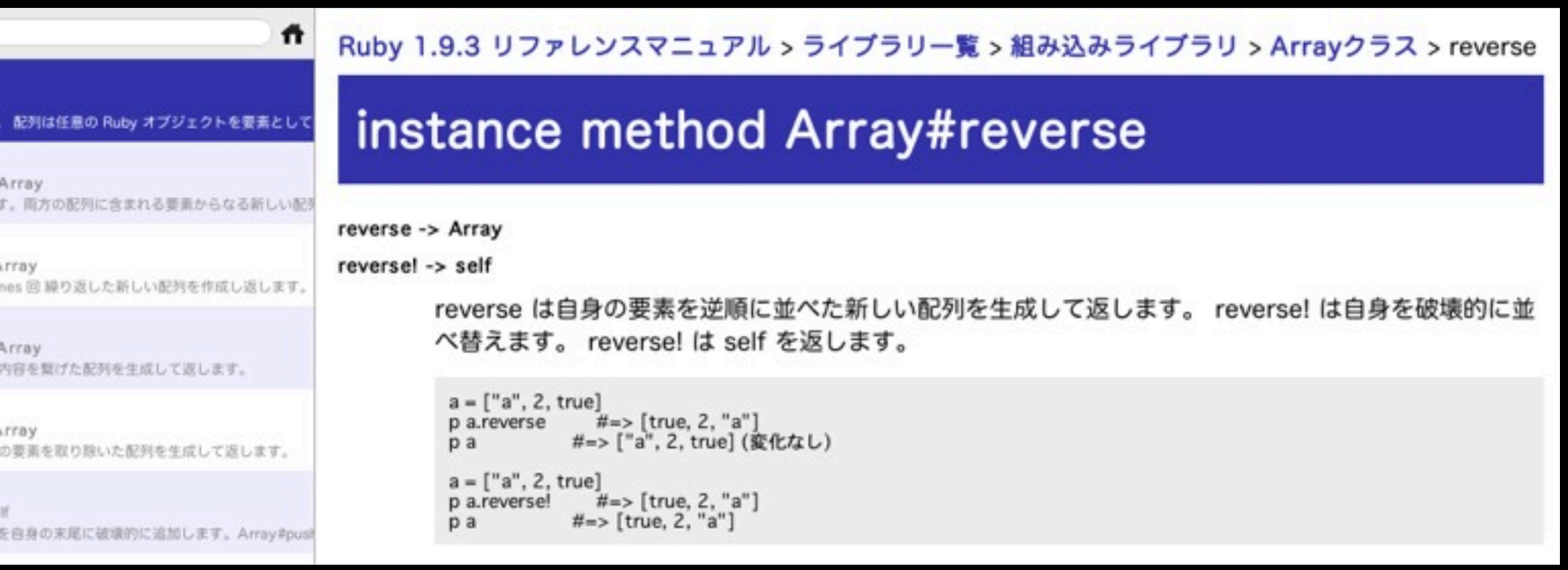
下の方へ探していくと・・・

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



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]
```

reverseを使えばできそうです。

今週

ここから

目次

演習特集

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%B0%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%B8%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%B8%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:スタブ 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%B0%E7%90%86_%E6%B0%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

Wikipediaのアクセス数解析

```
# encoding: utf-8
require "cgi"
filename = "pagecounts-20120526-000000-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
```

前回、エラーメッセージがたくさん出た理由

#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
#<ArgumentError: invalid byte sequence in UTF-8>
...
こんなの出てました。

調べたところ、以下のようになってました

```
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
```

文字コード変換にギブアップするデータは、以下の例外が発生
#<ArgumentError: invalid byte sequence in UTF-8>

例外が発生した場合、この処理が行われる。
捕まえた例外が変数eに入り、それを表示している。

調査の結果、どうやらWikipediaサイトで公開されているアクセス数データには、ときどき文字コード変換にギブアップするデータの行があるようです。
ここでは、ｐｅの行を＃ｐｅとコメントアウトすることにします。
（変換できないデータの行は無視することになります。）
※正確に解析したい場合はもう少しがんばる方法もありますが、今回は数十万行のうちの数行なので無視できると考えます。
※それはそれとして、配ったデータファイルも一部問題があったので再配布します。

Wikipediaのアクセス数解析

```
# encoding: utf-8
require "cgi"
filename = "pagecounts-20120526-000000-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

1. 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 pagecounts-20120526-000000-ja.txt を置いて実行してください。http://bit.ly/wpdata0531 からダウンロードできます。zip圧縮してあります。ブラウザでダウンロードして解凍するか、Linuxの人は下記の※1の方法でもかまいません。
（実行時間が長いので、実際のデータファイルから日本語部分だけ抜き出して小さくしてあります。）

※1:Linuxでのダウンロード方法の例
端末を開いてソースコードがあるフォルダで以下を実行。wgetはshellで使えるダウンローダーです。unzip はzip解凍コマンドです。

```
$ wget http://bit.ly/wpdata0531
$ unzip wpdata0531
```

※2: 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は文字コードで、shift_jisのwin版です。

```
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

2.【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 の場合は端末から $ 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
また、 コード内3行目くらいのfilenameを解凍したファイル名にする必要があります。

演習の前に

1つだけ

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？

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

"abc".upcase!

実行前

abc

オブジェクト

upcase!

実行後

ABC

オブジェクト

オブジェクト
が変身する

"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

変換した
新しい
オブジェクト
が複製される

upcase! を使うコード

```
"abc".upcase!
```

実行前 → upcase! → 実行後

abc → 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
`puts a`	

upcase を使うコード

```
"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

変換した
新しい
オブジェクト
が複製される

```
a = "abc"
```

a → abc

```
b = a.upcase
puts b
```

a → abc　b → ABC

複製されたオブジェクトを
別の変数に入れる

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!
```

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）

演習問題集

どれから解いても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
1.odd? → true, 2.odd? → false
ちなみに偶数か調べるのは Fixnum#even? メソッドです。

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 のキーの中に a 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=>"year!", :bob=>"yo!", :linda => "wow!" }

↑というHash を↓にできればOKです。

{:alice=>"YEAR!", :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 = "900"
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、900 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

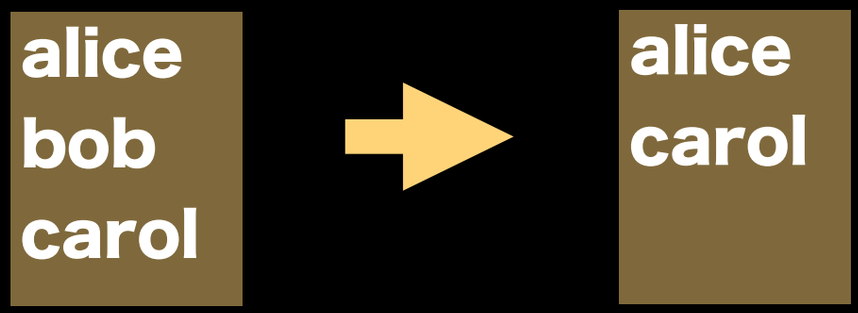
引数にハッシュを受け取るメソッドを書いてください。そのメソッドの中で、引数で受け取ったハッシュのキー :text の値を表示してください。
（例えば、メソッド名を print_text として、メソッド呼び出し側は以下のようになります。）

```
print_text({:text => ”TEXT!!”})
```

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
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"]

まとめ

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
1.odd? → true, 2.odd? → false
ちなみに偶数か調べるのは Fixnum#even? メソッドです。

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

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!"}]
```

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 = [6,2,1]
p array.count { |i| i < 3 }
```

演習問題 4

Hash のキーの中に a 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=>"year!", :bob=>"yo!", :linda => "wow!" }

↑というハッシュオブジェクトを↓にできればOKです。

{:alice=>"YEAR!", :bob=>"yo!", :linda => "WOW!" }

```
h = {:alice=>"year!", :bob=>"yo!", :linda => "wow!" }
h.each do |key, value|
  h[key] = value.upcase if key =~ /a/
end
p h
#=> {:alice=>"YEAR!", :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 = "900"
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、900 を表示）

考え方の一例：
arrayを作ってこの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 = "900"
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のキー :text の値を表示してください。（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。）

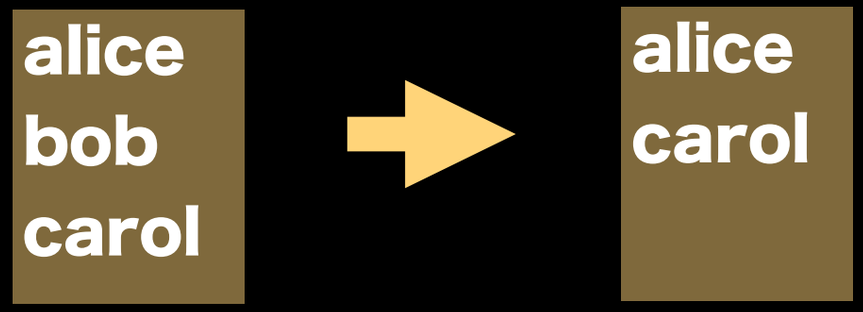
```
print_text({:text => ”TEXT!!”})
```

```
def print_text(h)
  puts h[:text]
end
print_text({:text=>”TEXT!!”}) #=> TEXT!!
```

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：



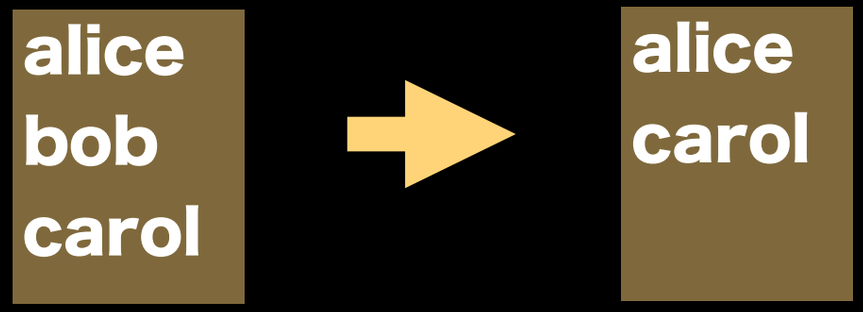

```
# ファイルから読み込み
included = [] # 出力用データを格納するArray
in_filename = 'in.txt'
File.open(in_filename, 'r:UTF-8') do |in_file|
  while text = in_file.gets
    included << text if text =~ /a/
  end
end
# ファイルへ書き込み
out_filename = 'out.txt'
File.open(out_filename, 'w:UTF-8') do |out_file|
  included.each do |i|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

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