Ruby 初級者向けレッスン 40 回
— Ruby のメソッド —

こなみひでお@Ruby 関西

2010 年 12 月 4 日

1　メソッド定義

1.1　基本的な形

この形は Ruby でも他の大抵の言語でもあまり変わりません。

```
def hoge(arg1,arg2,...)
  # ここで arg1,arg2,... を使ってお仕事
  return val  # 戻り値（返り値）を返す。
end
```

C の関数だったらこう。

```
int hoge(int arg1, int arg2,...){
  int val;
  // ここで arg1,arg2,... を使ってお仕事
  return val;  // 戻り値（返り値）を返す。
}
```

1.2　Ruby のメソッドでできること

1.2.1　return を省略できる

return を省略すると，最後に評価された値が返されます。
こっちが「ふつう」の形。

```
def hoge(arg1,arg2)
  return arg1 + arg2
end
```

ところが，これでもよろしい。

```
def hoge(arg1,arg2)
  arg1 + arg2
end
```

■じゃあ，return なんていらない？ →メソッドからの出口を途中にも欲しいとか，複数の出口がほしいときには必要。たとえば次のような再帰的メソッドでは出口が複数になります。

```
def fibonacci(n)
  if n < 2 then
     return 1
  else
     return fibonacci(n-1) + fibonacci(n-2)
  end
end
```

1.2.2 Ruby のメソッドは複数の戻り値を持てる

Java とか C で書いていてちょっとはがゆいのは，return で返せるのは 1 つの値だけということ。ところが Ruby では，次のように複数の値を返せます。

```
def hogehoge(x,y)
  return x+y, x-y
end
a,b = hogehoge(10,5)
puts "a = #{a}, b = #{b}"
```

そうそう，このときにも return は省略できませんね。

1.3 メソッドへの引数の渡し方

1.3.1 オプション引数を渡す

メソッド定義では，仮引数の記述に代入文を置くと，呼び出し時に引数を省略したときのデフォルト値を設定することができます。

```
def hogera(x,y,opt1="yes",opt2=12)
  puts "#{x},#{y}: #{opt1} ,#{opt2} "
end
```

```
hogera(10,20)            # 引数 2 つを省略
hogera(10,20,"no")       # 引数 1 つを省略
hogera(10,20,"no",25)    # 省略なし
hogera(10,20,"24")       # 引数 1 つを省略のつもり
```

実行結果は次のとおり。

```
10,20: yes ,12
10,20: no ,12
10,20: no ,25
10,20: 24 ,12      ※期待に反した動作
```

1.3.2 不定数の引数をメソッドに渡す

C の printf は任意の数の引数をとることができます。そのくせこいつは C にしては特殊な関数で，こういう関数を**自作しようとすると**苦労します。

```
printf("Hello World!");
printf("x = %d\n",x);
printf("x = %d, v = %f\n",x,v);
```

Ruby では次のようにして任意の数の引数を渡すようにできます。honyara の仮引数，*args に注目して下さい。

```
def honyara(x,y,*args)
  puts "#{x},#{y}"
  p args
end
honyara(10,20,1,2,3)
honyara(10,20,1,"Hi!",true,:abc)
```

実行結果は次のとおりです。

```
10,20
[1, 2, 3]
10,20
[1, "Hi!", true, :abc]
```

つまり，受け取った側では，仮引数に *args にまとめて引き受けて，配列 args で処理しています。

1.4 メソッド呼び出しの括弧は省略できる

ここまでメソッド呼び出しでは引数を括弧に入れていました。たとえば,

```
honyara(10,20,1,"Hi!",true,:abc)
```

のように。しかし，次のように書いても有効です。

```
honyara 10,20,1,"Hi!",true,:abc
```

ただし，メソッドの戻り値を他の変数に代入したり，`if` の後に書くときには，括弧を付けるべきです。

練習 1–1　次のように，任意の数の整数データを与えるとその総和を計算するメソッドを書いて下さい。

```
sum_all(1,2,3,4,5)  #=> 15.0
```

練習 1–2　次のように，整数データの配列を与えるとその総和を計算するメソッドを書いて下さい。このメソッドはデフォルトでは float(浮動小数点数) で計算するものとし，オプション引数で `:int` というシンボルを与えると整数で計算するものとします。

```
sum_all([1,2,3,4,5])  #=> 15.0
sum_all(:int, [1,2,3,4,5])  #=> 15
```

2 正規表現

文字列を賢く取り扱うために作られた道具です．使いこなすと，文字列を検索したり，必要な情報を拾い出したりする作業がとても柔軟かつ機能的に行えるようになります．

2.1 正規表現オブジェクトのかたち

次の形の式は**正規表現リテラル**です．これらはすべて Regexp クラスのインスタンスになっています．

1. `/^[Rr]uby/`
   文字列の前後をスラッシュ / で挟んだものが基本的な正規表現リテラルの形．この例は「文字列先頭から始まる Ruby または ruby」とマッチする．ハット ^[*1]やカギ括弧 [ ] は**メタ文字**で，特別の働きを持つ．^ は文字列の先頭を意味し (→表 3)，[Rr] は R か r のいずれかを意味する (→表 4)．
2. `/Ruby/i`
   / の後に置かれている i は**オプション**の例．正規表現の機能を修飾する働きをする．i オプションがあると，大文字と小文字の違いは無視される[*2](→表 1)．
3. `%r|Ruby|`
   スラッシュではなく %r に引き続く縦棒 |(または他の記号) に文字列を挟んでも正規表現リテラルとなる．式がバックスラッシュで煩雑になるのを避けることができる．
4. `r = Regexp.new("[Rr]uby")`
   正規表現を Regexp クラスのインスタントして文字列から生成することもできます．

次はすべて文字列'<a href="http://kyojo.jp">'
にマッチします．

```
/<a href\=\"http\:\/\/kyojo\.jp\"\>/
%r|<a href="http://kyojo.jp">|
%r!<a href="http://kyojo.jp">!
Regexp.new('<a href="http://kyojo.jp">')
```

表 1　正規表現のオプション

オプション	意味	備考
i	大小文字を無視	
o	式展開を行う	
x	空白文字を無視	
m	複数行モード	
n,e,s,u	文字コード指定	n:無指定

[*1] ハット ^ はキャレットとも呼ばれます．

[*2] 大文字は upper case, 小文字は lower case といい，両者の区別を無視することを ignore cases というので i が使われるわけです．

2.2 さまざまな正規表現

表 2〜表 4 に正規表現を構成するためのメタ文字と，
それらを使った規則を示しておきます．

表 2　文字種を表すメタ文字

\w	英数字 (日本語コードの英数字も含む)
\W	非英数字 (\w) 以外
\s	空白文字 (スペース，タブ，改行)
\S	非空白文字 (\w 以外)
\d	数字 (1 バイト文字のみ)
\D	非数字
.	任意の文字 (日本語コードも含む)

表 3　位置を指定するメタ文字

^	文字列の先頭にマッチ
$	文字列の最後にマッチ

表 4　文字クラスの指定

正規表現	マッチする文字
[abc]	a,b,c のどれか
[a-z]	a,...,z のどれか
[0-9a-z]	数字と英小文字
[^abc]	a,b,c 以外のどれか

表 5　文字の繰り返しの表現: m, n は正の整数

?	直前の文字の 0 回または 1 回の「繰り返し」	
+	直前の文字の 1 回以上の繰り返し	+? で最短一致
*	直前の文字の 0 回以上の繰り返し	*? で最短一致
{m}	直前の文字の m 回の繰り返し	{m}? で最短一致
{m,}	直前の文字の m 回以上の繰り返し	{m,}? で最短一致
{m,n}	直前の文字の m 回以上 n 回までの繰り返し	{m,n}? で最短一致

2.3 正規表現を組み立ててみよう

正規表現の実例をいろいろと考えてみましょう．下に列記されている文字列にマッチする正規表現を組み立ててください．解答は下にあります．なお，正解は一通りとは限りません．

1. 5 桁の正の整数
2. 任意の桁数の正の整数
3. 任意の桁数の正の整数，ただし先頭が 0 でないもの
4. 負の値も含む任意の整数，先頭には + が付くかもしれないし，そうでないかもしれない．
5. 任意の数値，つまり正負の符号が付いている可能性があり，途中に小数点をもつかも知れない数字の列

6. 'sum', 'max1', 'array02' のように，小文字の英字の文字列の後にゼロ個以上の数字が続く文字列
7. 12:30:05 というフォーマットの時刻表記
8. 上と 2:0:8 の形の文字列（時刻の数字が 1,2 桁のいずれか）
9. < で始まり，途中に任意の文字列があって，> で終わる文字列
10. HTML のタグ（上の応用だけど一工夫必要）

2.3.1 上の解答

1. /[0-9][0-9][0-9][0-9][0-9]/
   /\d\d\d\d\d/
   /\d{5}/
2. /\d+/
   /[0-9]+/
3. /[1-9]\d*/
4. /[\+\-]?\d+/
5. /[\+\-]?\d+\.?\d*/
6. /[a-z]+\d*/
7. /\d\d\:\d\d\:\d\d/
8. /\d{1,2}\:\d{1,2}\:\d{1,2}
9. /\<.*\>/
10. /\<.+?\>/

練習 2–1　HTML のドキュメント中のハイパーリンクを URL と名前とに分解するための正規表現を書いて下さい。ハイパーリンクは次のような形をしていて，他のタグは内部にないものとします。なお，" の代わりに’ が使われることもあります。

Ruby **関西ホーム**

練習 2–2　適当な HTML ファイルをネットからダウンロードしてください，そのファイル全体を 1 行として読み込み，そこからハイパーリンクを抽出するメソッドを書いて下さい。

- ハイパーリンクは一般に複数存在しているので，配列の配列，またはハッシュで返す。
- 返される日本語エンコーディングは， Shift-JIS, EUC-JP, UTF-8 のいずれかとし，デフォルトでは UTF-8 で返すものとする。それ以外はメソッド呼び出しのときに指定する（オプション引数を使います)。

なお，ファイル全体を 1 行に読み込むためには，次のように入力レコードセパレータの$/ に nil を代入しておいて，1 行読み込みを実行すれば簡単です。

```
$/ = nil
line =  File.read("hoge.txt")
p line
```