Ruby 初級者向けレッスン 第 13 回

okkez @ Ruby 関西, サカイ@小波ゼミ

2007 年 06 月 16 日

演習問題

色々なメソッドを定義してみましょう

Part 1

- 引数を三つ受け取り、引数の内容に応じて計算結果を返すメソッド
  - 三つの引数のうち一つは演算子を表す文字列です
  - 残りの二つは数値です
  - eval 系メソッドの使用は禁止です
  - 最低限、四則演算を実装してください。それ以上実装するのは自由です。

Part 2

- 先ほど作成したメソッドに以下の機能を追加してください
  - ゼロ除算した場合は例外メッセージを日本語で表示させる
  - 対応していない演算子が与えられた場合は例外を発生させる

Part 3

- 先ほどまでに作成したメソッドの引数で受け取ることができる数値の個数を無制限にしてください
  - ただし、演算子は一つでいいです。
  - 例えば +,1,2,3 のような引数が渡された場合の結果は 6 です。

解答例

Part1

まずはベタベタに。特に解説はいらないと思います。

```
01: # answer1.rb
02:
03: def calc(operator, op1, op2)
04:   case operator
05:   when '+'
06:     op1 + op2
07:   when '-'
08:     op1 - op2
09:   when '*'
10:     op1 * op2
11:   when '/'
12:     op1 / op2
13:   end
14: end
15:
16: p calc('+', 1, 2) #=> 3
17: p calc('-', 1, 2) #=> -1
18: p calc('*', 1, 2) #=> 2
19: p calc('/', 1, 2) #=> 0
```

Part2

これもベタベタに。

```
01: # answer2.rb
02:
03: def calc(operator, op1, op2)
04:   case operator
05:   when '+'
06:     op1 + op2
07:   when '-'
08:     op1 - op2
09:   when '*'
10:     op1 * op2
11:   when '/'
12:     begin
13:       op1 / op2
14:     rescue ZeroDivisionError
```

```
15:       STDERR.puts ’ゼロで除算してはいけません’
16:     end
17:   else
18:     raise ArgumentError, "不正な演算子です: #{operator}"
19:   end
20: end
21:
22: p calc(’/’, 1, 0)
23: p calc(’%’, 1, 2)
```

Part3

Part1,Part2 の解答を素直に発展させるとこうなります。この解答のポイントは初期値なしの Enumerable#inject を使用している点です。詳細はリファレンスマニュアルを参照してください。

```
01: # answer3.rb
02:
03: def calc(operator, *op)
04:   case operator
05:   when ’+’
06:     op.inject{|a, b| a + b}
07:   when ’-’
08:     op.inject{|a, b| a - b}
09:   when ’*’
10:     op.inject{|a, b| a * b}
11:   when ’/’
12:     begin
13:       op.inject{|a, b| a / b.to_f }
14:     rescue ZeroDivisionError
15:       STDERR.puts ’ゼロで除算してはいけません’
16:     end
17:   else
18:     raise ArgumentError, "不正な演算子です: #{operator}"
19:   end
20: end
21:
22: p calc(’+’, 1, 2, 3, 4, 5)
23: p calc(’*’, 1, 2, 3, 4, 5)
24: p calc(’/’, 10, 2, 5)
25: p calc(’%’, 1, 2)
```

nov さんによるエレガントな解答。ポイントを箇条書きであげておきます。

- 最初に Fixnum クラスの演算子っぽいメソッドでアリティが 1 のものを選んでいます。

```
["%", "<<", "&", ">>", "*", "+", "-", "/", "|", "^",
 "**", "<", "<=>", "==", ">", "===", ">=", "<=", "=~", "[]"]
```

- KNOWN_OPERATORS に第一引数で指定された演算子が存在しない場合は例外発生
- line.13 以降で e に特異メソッドを定義して ZeroDivisionError に元から存在するエラーメッセージを上書きしている
- line.19 の raise はさっき rescue した例外をそのまま raise する

```
01: # answer3a.rb (thanks nov)
02:
03: KNOWN_OPERATORS = Fixnum.public_instance_methods.reject do |op|
04:   /\w/ === op.to_s or Fixnum.instance_method(op).arity != 1
05: end
06:
07: def calc1(op, *x)
08:   raise ArgumentError unless KNOWN_OPERATORS.member?(op)
09:   begin
10:     x.inject do |a, b|
11:       a.__send__(op, b)
12:     end
13:   rescue ZeroDivisionError => e
14:     class << e
15:       def message
16:         "ゼロで割っちゃダメ"
17:       end
18:     end
19:     raise
20:   end
21: end
```

うじひさくんによる純粋関数型言語「Ruby」で書かれた解答。私はこの言語にあまり詳しくないので解説はできません。すみません。

```
01: class Proc
02:   def to_ary
03:     [call(nil)]
04:   end
05:
06:   def inspect
07:     call nil
08:   end
09: end
10:
```

```
11: calc = lambda {|op|
12:   raise ’オペレータは四則演算のみ’ unless %w(+ - * /).include? op
13:   lambda {|n|
14:     lambda {|m|
15:       raise ’0で割り算できません’ if op == ’/’ && m == 0
16:       return n unless m
17:       lambda {|j| calc[op][n.__send__(op, m)][j] }
18:     }
19:   }
20: }
21:
22: puts calc[’+’][9][6][3][1][2]
23: p    calc[’+’][9][6][3][1][2]
24: puts calc[’*’][2][5]
25: p    calc[’/’][2][5.0]
26: p    calc[’*’][2][0]
27: p    calc[’/’][1][0]
```

今回は禁止していた eval を使用してみた解答例です。

- line.3 独自の例外を定義しています
- これくらい制限をつけておけば eval を使っても大丈夫でしょう

```
01: # answer3c.rb
02:
03: class InvalidOperatorError < ArgumentError
04: end
05:
06: def calc(op, *x)
07:   unless %w[+ - * /].include?(op)
08:     raise InvalidOperatorError, ’そんな演算子では計算できません！’
09:   end
10:   unless x.partition{|v| v.kind_of?(Numeric) }.last.empty?
11:     raise ArgumentError,
12:     "引数が変ですよ : #{x.map{|v| v.inspect }.join(’,’)}"
13:   end
14:   eval(x.join(op))
15: end
16:
17: puts calc(’+’, 1, 2, 3)
18: puts calc(’-’, 1, 2, 3)
19: puts calc(’*’, 1, 2, 3)
20: puts calc(’/’, 20, 2, 5)
21: begin
```

```
22:   puts calc('%', 2, 1)
23: rescue
24:   p $!
25: end
26: begin
27:   puts calc('+', '1', '1')
28: rescue
29:   p $!
30: end
```

疑問、質問は learn-ruby-kansai のメーリングリストまで。