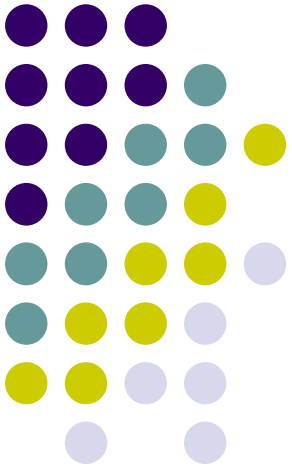
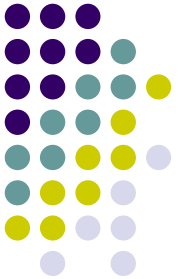
Ruby クイズ第二回

2006/12/16

cuzic

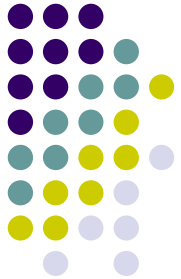


自己紹介



- cuzic は「きゅーじっく」と読む
- 趣味はプログラミングと政治・経済と世界史
- 何気に親指シフトキーボード使い
  - 昔、富士通のワープロで使われていたキーボード

Ruby クイズとは

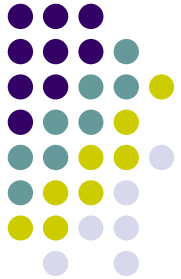


- 同名の Ruby でパズル的な問題を解くやつとはまったく別
  - 名前だけパクった
- Ruby 中級者が上級者になるときにハマりやすいポイントを分かりやすい選択問題で学ぶ企画

- 今回はいろいろパワーアップ！！

Ruby クイズ第一問

Ruby の文字列



- 以下は Ruby の文字列への記述である。正しいものを選べ

(1) Ruby での文字列はマルチスレッド処理を前提とした設計になっており、他のスレッドにより突然変更されることがないよう値が不変の文字列クラスと文字列構築用の文字列クラスがある。

(2) Ruby の文字列では、ヌル文字 (\000 に相当する文字) 以外の任意のバイト列を扱える

(3) Ruby の認識するマルチバイトエンコーディングには、EUC, SJIS, JIS, NONE の4種類がある。

(4) Ruby での文字列は任意の長さのバイト列を扱うことができ、日本語文字も特殊なグローバル変数に漢字コードを設定することによって正しく処理可能である。

選択肢

(ア) 1　　(イ) 2　　(ウ) 3　　(エ) 4

Ruby クイズ第一問 解答

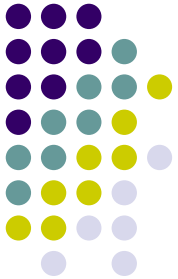
Ruby の文字列

解答　(エ)

(4) Ruby での文字列は任意の長さのバイト列を扱うことができ、日本語文字も特殊なグローバル変数に漢字コードを設定することによって正しく処理可能である。

これが正解です。Ruby では String クラスを用いて、任意の長さのバイト列を正しく扱えます。

Ruby クイズ第一問　解説 (1)
Ruby の文字列



(1)　Ruby での文字列はマルチスレッド処理を前提とした設計になっており、他のスレッドにより突然変更されることがないよう値が不変の文字列クラスと文字列構築用の文字列クラスがある。

これは Java や C# についての記述です。Ruby は文字列の構築と文字列の保持でもすべて String を利用します。そのため Ruby の文字列はシンプルですが、マルチスレッド処理では注意が必要です。

(2) Ruby の文字列では、ヌル文字 (\000 に相当する文字) 以外の任意のバイト列を扱える

Ruby では ヌル文字が文字列中にたとえあったとしても正しく扱うことができます。一部ヌル文字非対応のメソッドがありますが、例外的です。

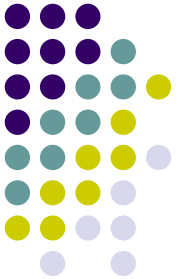
Ruby クイズ第一問　解説 (2)

Ruby の文字列

(3) Ruby の認識するマルチバイトエンコーディングには、EUC, SJIS, JIS, NONE の４種類がある。

Ruby の認識するマルチバイト文字列エンコーディングは “EUC” “SJIS” “UTF8” “NONE” のいずれかです。

Ruby クイズ第二問
文字列リテラル



- 次の出力は？

```
p <<EOD % “Ruby”
I love #{%{#{%q{%s}}}}
EOD
```

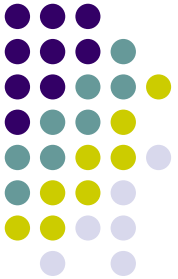
(ア) “I love #{%q{Ruby}}\n”

(イ) “I love Ruby\n”

(ウ) 文法エラーで出力されない

(エ) “I love Ruby”

Ruby クイズ第二問　解答
文字列リテラル



- 答え（イ）

```
p <<EOD % “Ruby”
I love #{%{#{%q{%s}}}}
EOD
#=> “I love Ruby\n”
```

Ruby クイズ第二問　解説(1)
文字列リテラル

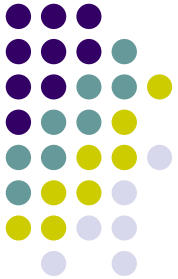
- 5つの文字列リテラル

  “” , ‘’ , %{}, %q{}, ヒアドキュメント
- 式展開可能な文字列リテラル

  “”, %{}, <<EOD
- 文字列そのままを利用する文字列リテラル

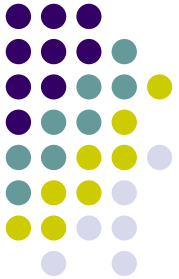
  ‘’ , %q{}, <<‘EOD’
- %{}, %q{} の使いどころ
  - % 記法は、文字列中に \ や “ を含めたいときに有用
  - 特に %r{} による正規表現リテラルは覚えたいテク
  - 始まりの区切り文字が (,{,[,< の場合は終わりの区切り文字は),},],> になる。

Ruby クイズ第二問　解説(2)
文字列リテラル



- 文字列は、% メソッドを持つ
  - str % a,b,c,…は sprintf str, a,b,c,… と等価
  - sprintf は文字列をフォーマット表記する C 以来の伝統的方法
  - % メソッドは Python 由来(？)
- ヒアドキュメントは開始ラベル `<<EOD' が文法要素としての式に相当
  - メソッド呼び出しも可能
  - 引数をヒアドキュメントにすると、メソッド呼び出しがすっきりできる場合がある

Ruby クイズ第三問　問題
Ruby の文字列の文法



- 次のうちパースエラーまたは実行時エラーになるものは？

(1)　“Perl” = “Ruby”

(2)　“Perl”.replace “Ruby”

(3) a = “Perl”
　　a[“Perl”] = “Ruby”

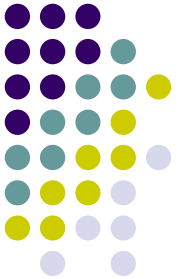
(4) a = “Perl”
　　a[/Perl/,0] = “Ruby”

(5) class A
　　def to_str
　　"class A“
　　end
　　end
　　“” + A.new

選択肢

(ア) 1　　（イ）1,2　　（ウ）1, 5　　（エ）1, 3, 4　　（オ）3, 4

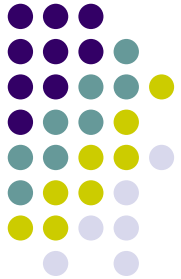
Ruby クイズ第三問　解答
Ruby の文字列の文法



答　ア

- “Perl” = “Ruby” が parse error
  他は、すべて正常に実行可能

Ruby クイズ第三問　解説(1)
Ruby の文字列の文法



(1) “Perl” = “Ruby”

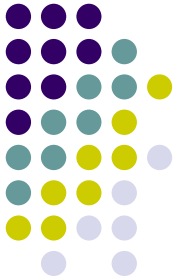
#=> parse error, unexpected '=', expecting $

文字列リテラルを代入文の左辺にすることは Ruby の文法上できません

(2) “Perl”.replace “Ruby”

文字列リテラルはレシーバとして String の任意のメソッド呼び出しが可能です。破壊的に変更するようなメソッドも呼び出せます。

Ruby クイズ第三問　解説(2)
Ruby の文字列の文法



(3) a = “Perl”

　　a[“Perl”] = “Ruby”

(4) a = “Perl”

　　a[/Perl/,0] = “Ruby”

String#[]= メソッド

self[*substr*]=*val* の形式では、文字列中の *substr* に一致する最初の部分文字列を *val* に置き換えます

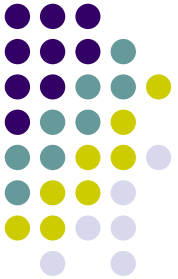
self[*regexp*, *nth*]=*val* の形式では、正規表現 *regexp* の *nth* 番目の括弧にマッチする最初の部分文字列を文字列*val*で置き換えます

Ruby クイズ第三問　解説(3)
Ruby の文字列の文法

(1)
```
class A
  def to_str
    "class A“
  end
end
“” + A.new
#=> “class A”
```

String クラスは引数として文字列をとるメソッドが多くあります。
そのようなメソッドに、文字列以外のオブジェクトを送ると、文字列への暗黙の変換が行われます。
文字列への暗黙の変換は to_str メソッドによって行われます。
暗黙の変換が出来ない場合は、エラーとなります。

Ruby クイズ第四問　問題
部分文字列の置換



- 実行に対する正しい出力の組み合わせを選べ

(1)
```
a = “dog”
b = a[0,2]
b = “pi”
p a
```

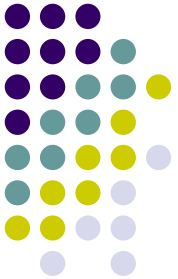
(2)
```
a = “dog”
b = a[0,2]
b.replace “pi”
p a
```

(3)
```
a = “dog”
a[0,2] = “pi”
p a
```

選択肢

（ア）1: “dog”　2:”dog”　3: “dog”
（イ）1: “dog”　2:”dog”　3: “pig”
（ウ）1: “dog”　2:”pig”　3: “pig”
（エ）1: “pig”　2:”pig”　3: “pig”

Ruby クイズ第四問　解答
部分文字列の置換



- 答:イ

(1)
```
a = “dog”
b = a[0,2]
b = “pi”
p a
#=> “dog”
```

(2)
```
a = “dog”
b = a[0,2]
b.replace “pi”
p a
#=> “dog”
```

(3)
```
a = “dog”
a[0,2] = “pi”
p a
#=> “pig”
```

Ruby クイズ第四問　解説

部分文字列の置換

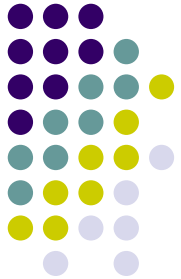
(1)
```
a = “dog”
b = a[0,2]
b = “pi”
p a
#=> “dog”
```

(2)
```
a = “dog”
b = a[0,2]
b.replace “pi”
p a
#=> “dog”
```

(3)
```
a = “dog”
a[0,2] = “pi”
p a
#=> “pig”
```

- *self*#[*index*,*length*] の形式で得られる部分文字列を変更しても、元の文字列は変更されません。
- *self*#[]= というメソッド呼び出しは、*self*#[] とは異なるメソッド呼び出しです。

Ruby クイズ第五問
フォーマット文字列



- 以下の対応関係を適切に結べ

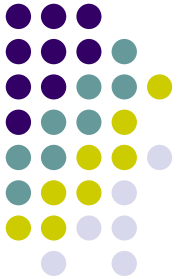
(1) p sprintf("% 6d",2007)
(2) p sprintf("%+6d",2007)
(3) p sprintf("%06d",2007)
(4) p sprintf("%-6d",2007)

(a) "002007"
(b) " 2007"
(c) "2007 "
(d) " +2007“

- 選択肢

(ア) 1-b , 2-d , 3-c , 4-a
（イ）1-d , 2-b , 3-a , 3-c
(ウ) 1-b , 2-d , 3-a , 4-c
(エ) 1-b, 2-a , 3-d , 4-c

Ruby クイズ第五問　解答
フォーマット文字列



答えは ウ

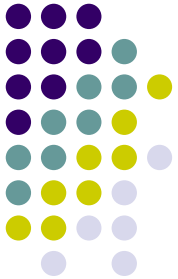
(1) p sprintf("% 6d",2007)
　　“002007”

(2) p sprintf("%+6d",2007)
　　“ +2007”

(3) p sprintf("%06d",2007)
　　“002007”

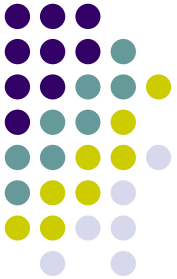
(4) p sprintf("%-6d",2007)
　　“2007 “

Ruby クイズ第五問　解説 (1)
フォーマット文字列



- sprintf はフラグと幅そして指示子だけ覚えましょう
  sprintf “%[フラグ][幅]指示子”
- フラグ
  - ‘ ‘ スペース
    - 右詰めで正の値なら、符号なし。スペースで左端を埋める
  - ‘+’
    - 右詰めで常に符号付。スペースで左端を埋める
  - ‘0’
    - 右詰めで正の値なら符号なし。負なら先頭に - 。0で左端を埋める
  - ‘-’
    - 左詰めで、正なら符号なし。スペースで右端を埋める

Ruby クイズ第五問　解説 (2)
フォーマット文字列

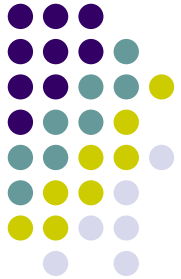


sprintf “%[フラグ]%[幅]指示子”

- 幅

  幅とは生成文字列の長さ
  結果の文字列が指定した幅を超える場合は幅の指定は無効
  幅として * を指定すると、引数で指定可能

- 主な指示子
  - %d ：10進数表現
  - %f ：小数点表現
  - %x : 16進数表現



みなさんどれくらいできましたか？

ご清聴ありがとうございました