Ruby はじめに

[Ruby][Perl]

http://www1.tf.chiba-u.jp/~shin/tutorial/index.rb

演算

http://www.ruby-lang.org/ja/man/html/_B1E9BBBBBBD2BCB0.html

四則演算 (整数、浮動小数点数)

```
puts 1+2
puts 1.0/3.0
```

べき乗

```
puts 2**10
```

剰余

```
puts 10 % 3
```

絶対値

```
puts ((1-3).abs)
```

乱数

```
puts rand
puts rand(11) # 0-10
```

乱数 Seed を使う

```
srand 999
puts rand
```

Math

```
puts Math::PI
puts Math::E
puts Math.sqrt(5)
puts Math.log(100)
puts Math::E**Math.log(100)
```

各種演算

```
puts 'hello' + ' world'  # 文字列の足し算
puts '0' * 4             # 文字列の掛け算 0000
```

```
puts 0.to_s * 4          # 数値 -> 文字列変換
puts '9'.to_i * 2        # 文字列 -> 整数変換
puts 1.to_f / 3.to_f     # 整数 -> 浮動小数点数変換
```

標準入力 gets(tring)

```
puts gets
```

末尾の改行を除く chomp

```
puts
val = gets.chomp
puts '<-- ' + val + ' -->
```

演算子もメソッド

```
puts 1.+ 1
```

文字列メソッド

```
puts 'hello'.reverse
puts 'hello is ' + 'hello'.length.to_s + ' chars.'
puts 'abcd'.upcase
puts 'ABCD'.downcase
puts 'abcd efg'.capitalize
```

センタリング、左寄せ、右寄せ

```
puts '0123456789' * 2
puts 'hello'.center(20)
puts 'hello'.ljust(20)
puts 'hello'.rjust(20)
```

配列

```
dates = ['sun', 'mon', 'tue', 'wed', 'thu', 'fri', 'sat']
puts dates[0]
puts dates[1]
puts dates[2]
```

sort メソッド

```
dates = ['sun', 'mon', 'tue', 'wed', 'thu', 'fri', 'sat']
dates2 = dates.sort
dates2.each do |date|
  puts date
end
```

ハッシュ

```
has = {}
has['one'] = 1
has['two'] = 2
has['three'] = 3

puts has['one']
```

制御

http://www.ruby-lang.org/ja/man/html/_C0A9B8E6B9BDC2A4.html#unless

■比較

・比較演算子

```
puts 1 > 0.5
puts 0.5 == 0.5
puts 1 != 1.0
```

・ショートサーキット

```
c1 = true
c2 = false

def chk1
  c2 = true
  return true
end
def chk2
  c1 = false
  return false
end

if chk1 || chk2
  puts true
end
puts c1.to_s + "," + c2.to_s
```

■分岐

if

```
t = gets.chomp.to_i
if t < 12
  puts 'good moring.'
else
  if t == 12
    puts 'good afternoon.'
  else
    puts 'good evening.'
  end
end
```

・elsif

```
a = true
b = false
c = true
if a and b and c
  puts "all true"
elsif a or b or c
  puts "any one or more true"
else
  puts "all false"
end
```

unless

if と逆の条件判定

```
c1 = false

unless c1
  puts "c1 false"
else
  puts "c1 true"
end
```

case

```
def greet(hour)
  case hour
  when 5 .. 11
    msg = "morning"
  when 12
    msg = "afternoon"
  when 13 .. 20
    msg = "evening"
  else
    msg = "night"
  end
  return "good " + msg
end

puts greet(22)
```

■繰り返し

for

```
ary = ['a','b','c','d']
for c in ary
  puts c
end
```

while

```
s = gets.chomp;
while s != 'q'
  puts s
  s = gets.chomp;
end
```

each メソッド

```
dates = ['sun', 'mon', 'tue', 'wed', 'thu', 'fri', 'sat']
dates.each do |date|
  puts date
end
```

times メソッド

```
3.times do
  puts 'hello'
end
```

例外

例外を発生

・ランタイム

```
def foo
  raise "Runtime Error!"
end
foo
```

・文法

```
def bar
  raise SyntaxError.new("invalid syntax")
end
bar
```

例外処理

・begin、rescue、else、ensure

```
def foo(x , y)
  begin
    ret =  x / y
  rescue ZeroDivisionError
    ret = nil
  else
    ret = nil
  ensure

  end
  return ret
end
puts foo(100, 0)
```

メソッドの作成

```
def greeting time
  if time < 12
    puts 'good moring.'
  else
    if time == 12
      puts 'good afternoon.'
    else
      puts 'good evening.'
    end
  end
end

greeting 12
```

戻値

```
def greeting time
  msg = ''
  if time < 12
```

```
    msg = 'good moring.'
  else
    if time == 12
      msg = 'good afternoon.'
    else
      msg = 'good evening.'
    end
  end
  msg # メソッドからの返り値は単純にメソッドの最後の行の値
end

puts greeting(21)
```

不定個の引数を渡す

「*」を使う

```
def prtPrms(*prms)
  for p in prms
    print p, " "
  end
end

prtPrms(1, 2, 3)
```

特異メソッド

・特異メソッドは特定のインスタンスに固有のメソッド

```
class Foo
end

f = Foo.new
def f.hello
  puts 'hello.'
end
f.hello
```

クラスメソッド

```
class Foo
 def Foo.name
   puts 'foo'
 end
end

Foo.name
```

ブロック付きメソッド

```
def foo
  puts "pre process."
  yield
  puts "post process."
end

foo { puts "main process." }
```

Web ページ取得例

```
require "open-uri"
```

```
open("http://typea.info", :proxy => 'http://proxy:8080') { |uri|
  puts uri.read
}
```

テキストファイル表示例

```
File.open("C:\\test.txt") {|file|
  while w = file.gets
    print w
  end
}
```

変数

http://www.ruby-lang.org/ja/man/html/_CAD1BFF4A4C8C4EABFF4.html

種類	書式	スコープ
ローカル変数	小文字または、'_' ではじめる	宣言した位置からその変数が宣言されたブロック、メソッド定義、またはクラス / モジュール定義の終りまでで
インスタンス変数	@ ではじめる	インスタンス変数はそのクラスまたはサブクラスのメソッド。特異メソッドから参照不可
クラス変数	@@ で始める	クラスの特異メソッド、インスタンスメソッドなどから参照／代入
グローバル変数	$ ではじめる	プログラムのどこからでも
定数	大文字ではじめる	定数が定義されたクラス / モジュール定義の中。再代入で警告

・メンバー変数 class Hoge

```
  def sayName
    puts @name
  end
  def giveName name
    @name = name #@ をつける
  end
end

h = Hoge.new
h.giveName('foo')
h.sayName
```

・定数

```
CONST_VALUE = 'constant'     # 大文字ではじめる
CONST_VALUE = 'change value' #warning
puts CONST_VALUE
```

クラス

クラスの拡張

組み込み整数クラスを拡張してみる

・同じクラスを再定義すると、前のクラス定義に追加されていきます。
・メソッドを再定義した場合には後のものが上書きしますので、前のものは失われます。

```
class Integer
  def to_chinese
    s = self.to_s

    cs = ['零', '壱','弐','参','四','伍',
          '六', '七','八','九']

    ret = ''
    i = 0
    while i < s.length
      ret += cs[s[i ,1].to_i]
      i += 1
    end
    return ret
  end
end

puts 123456.to_chinese
```

クラスを作る

```
class Hoge
  def sayName
    puts "hoge"
  end
end

h = Hoge.new
h.sayName
```

アクセッサ

メンバー変数は public にはできない。
アクセッサを定義するか、attr_reader、attr_writer、 attr_accessor を使う

```
class Foo
  def initialize(name, age)
    @name = name
    @age = age
  end

  attr_reader :name
  # 以下と同じ
  #def name
  # return @name
  #end

  attr_accessor :age
end
f = Foo.new('Foo', 12)
f.age = 20
puts f.name + ' ' + f.age.to_s
```

initialize メソッド

定義されていれば、生成時に呼び出される

```
class Hoge
  def initialize
    @name = 'bar'
  end
  def sayName
    puts @name
  end
  def giveName name
    @name = name
  end
end

Hoge.new.sayName
```

initialize に引数を渡す

```
class Hoge
  def initialize name
    @name = name
  end
  def sayName
    puts @name
  end
end

Hoge.new('hogehoge').sayName
```

クラス変数

・1.6 より実装。`@@' で始まる変数

```
class Foo
  @@name = 'foo'
end

puts Foo.new
```

手続きオブジェクト

[初めての JavaScript(関数)]
[クロージャ]

```
alert = Proc.new do
  puts "Error!"
end

alert.call
```

引数を取れる

```
alert = Proc.new do | reason |
  puts "Error! " + reason
end

alert.call 'Zero Divide'
alert.call 'Memory Leak'
```

モジュール

・名前空間

・mix-in

名前空間 (メソッド)

```
module Foo
  def self.hoge
    puts "Foo.hoge"
  end
end

module Bar
  def self.hoge
    puts "Bar.hoge"
  end
end

Foo::hoge
Bar::hoge
```

名前空間 (クラス)

```
module Foo
  class FooCls
    def hoge
      puts "Foo.hoge"
    end
  end
end

module Bar
  class BarCls
    def hoge
      puts "Bar.hoge"
    end
  end
end

Foo::FooCls.new.hoge
Bar::BarCls.new.hoge
```

mix-in

```
module Foo
  def foo
    puts "Foo"
  end
end

module Bar
  def bar
    puts "Bar"
  end
end

class MixinCls
  include Foo
  include Bar
end

m = MixinCls.new
m.foo
m.bar
```

オブジェクト指向

■継承

```
class Foo
  def initialize(name)
```

```
    @name = name
  end
  def hoge
    puts @name
  end
end

class Bar < Foo
  def initialize(name)
    @name = name
  end
end

Bar.new('hoge').hoge
```

オーバーライド

```
class Foo
  def name(name)
    @name = name
  end
  def hoge
    puts "Foo:" + @name
  end
end

class Bar < Foo
  def hoge
    puts "Bar:" + @name
  end
end

b = Bar.new
b.name('Bar')
b.hoge
```