

Ruby 講義

第12回 Ruby入門

Kuniaki IGARASHI/igaiga

2012.6.28 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)

○ 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」の提供を開始

○ 「@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

○ 「スマブレ！」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

○ 「シユフモ」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

アット・ニフティ
楽しいサービスがいっぱい

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

- 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19日 IR [「@nifty EMOBILE LTE 定額にねんプラン」の提供を開始](#)
- 2012年4月19日 IR [ニフティとサンリオウェーブ、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](#)

提供

講師

五十嵐邦明

株式会社万葉

エンジニア



GARASHI

.9.25 at 高専カンファ

いがいが
⑤

Teaching Assistant 演習 健二
クックパッド株式会社 エンジニア



先週の
おさらい

継承

既に定義されているクラスを拡張して新しいクラスを作ることを継承といいます。

(既にあるたい焼きの型を利用して、少し違う新しいたい焼きの型をつくるようなものです。)

継承

たとえばBookクラスとMagazineクラス
(雑誌)を作るとします。

```
class Book
  attr_accessor :title, :price
end
```

```
class Magazie
  attr_accessor :title, :price, :number
end
```

別々に定義を書いてもいいのですが、**共通項**
もたくさんあります。

継承

たとえばBookクラスとMagazineクラス
(雑誌)を作るとします。

```
class Book
  attr_accessor :title, :price
end
```

```
class Magazie
  attr_accessor :title, :price, :number
end
```

別々に定義を書いてもいいのですが、共通項
もたくさんあります。

継承

そんなときは、継承を使うと便利です。

継承を使った書き方

```
class Book  
  attr_accessor :title, :price  
end
```

```
class Magazine < Book  
  attr_accessor :number  
end
```

継承を使わない書き方

```
class Book  
  attr_accessor :title, :price  
end
```

```
class Magazine  
  attr_accessor :title, :price, :number  
end
```

←→
同じ動作

class Magazine < Book と書くことで、Bookクラスを継承した Magazineクラスを定義できます。 MagazineクラスはBookクラスの性質を受け継ぎます。何を受け継ぐかを次のページで解説します。

継承

継承する場合の書式

```
class クラス名 < スーパークラス名  
  クラスの定義  
end
```

スーパークラスとは、継承元の
クラス(親クラス)です。

継承したクラスは、親クラスの全てのインスタンス変数、メソッドなどを受け継ぎます。

```
class Book  
  attr_accessor :title, :price  
end
```

```
class Magazine < Book  
  attr_accessor :number  
end
```

例えばBookクラスを継承した
Magazineクラスは、titleとpriceを
受け継いでいます。例えば、↓のような
コードを書くことができます。

```
magazine = Magazine.new  
magazine.title = "CanCam"  
p magazine.title #=> "CanCam"
```

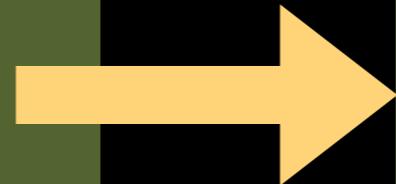
Module

メソッドを共同利用する仕組み

Module

クラスでモジュールをincludeすると、moduleに定義してあるメソッドをクラスへ追加することができます。

```
module Greeting
  def hello
    puts "Hello!"
  end
end
```



```
class Alice
  include Greeting
end

alice = Alice.new
alice.hello #=> "Hello!"
```

モジュールを定義します。
モジュールにはメソッドを
定義します。

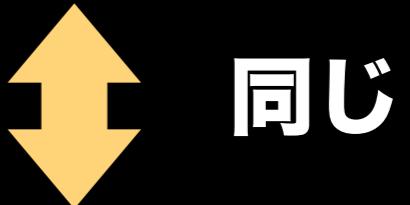
include したAliceクラスに
はhelloメソッドが追加され
ます。

Module

include した Alice クラスは右のように書いたことと同じになります。

```
module Greeting
  def hello
    puts "Hello!"
  end
end
```

```
class Alice
  include Greeting
end
```



```
class Alice
  def hello
    puts "Hello!"
  end
end
```

Module

複数のクラスで同じメソッドを利用したいときにmoduleを使うと重複をなくせるので便利です。

```
module Greeting
  def hello
    puts "Hello!"
  end
end
```

もしもhelloメソッドで表示する"Hello!"を"こんにちは"に変えたい場合はこのモジュールだけ変更すれば良い

```
class Alice
  include Greeting
end

alice = Alice.new
alice.hello #=> "Hello!"
```

```
class Bob
  include Greeting
end
```

```
bob = Bob.new
bob.hello #=> "Hello!"
```

Moduleの文法

```
module モジュール名  
#メソッド定義  
end
```

```
class Sample  
include モジュール名  
end
```

module 定義

include(読み込み)

includeとrequireの違い

```
class Sample  
  include Greeting  
end
```

```
require "filename"
```

include は Sample クラスに Greeting モジュールで定義されているメソッドを追加する命令

require は他のファイル(.rb)で定義されているメソッドやクラス、モジュールを読めるようにする命令

目次

Gem

—
Fragments from
my work box
With love from
Audrey

Gem

Rubyのライブラリ管理システム

Gem

ライブラリ：
他のプログラムから読み込んで利用するためのプログラム

たくさんの便利なライブラリがネット上に公開されています。それらをインストールするなど、管理してくれる仕組みがGemです。

Gemの例

- Excelファイルを読み書きする
 - twitterなど有名なWebサービスへのアクセスを簡単にしてくれる
 - 画像をリサイズしたりエフェクトをかける
- などなど数万種類のライブラリが公開されています

Gemの使い方

1. shell からgemコマンドを使ってインストール

```
$ gem install gem名
```

※ubuntuの場合は \$ sudo gem install gem名

2. コードで require "gem名"

```
require "gem名"  
# ここでgem を使うコードを書きます。
```

※ubuntuの場合は \$ sudo gem install gem名

Spreadsheet

Excelファイルを操作するgem

spreadsheet

Excelファイルを読み書きするgem
Excelがインストールされていなくても、
Windows以外でも動作します。

shell から以下のコマンドを実行するとインストー
ルできます。

```
$ gem install spreadsheet
```

※ubuntuの場合は \$ sudo gem install spreadsheet

spreadsheet gem 演習

spreadsheet gem をインストール

\$ gem install spreadsheet

Excelファイルを新規作成

```
require "spreadsheet"
```

```
book = Spreadsheet::Workbook.new
```

```
sheet = book.create_worksheet
```

```
sheet[0,0] = "testing..."
```

```
book.write("example.xls")
```

Excelファイルを開いて別名で保存

```
require "spreadsheet"
```

```
book = Spreadsheet.open("example.xls")
```

```
sheet = book.worksheet(0)
```

```
p str = sheet[0,0]
```

```
sheet[1,0] = "hello"
```

```
book.write("example2.xls")
```

Twitter

twitterを検索したり操作したりするgem

The Twitter Ruby Gem

Follow @gem Documentation Code

A Ruby wrapper for the Twitter API.

Installation

```
gem install twitter
```

Mailing List

Please direct any questions about the library to the mailing list.

Usage Examples

Return @sferik's location

```
Twitter.user("sferik").location
```

Return @sferik's most recent Tweet

```
Twitter.user_timeline("sferik").first.text
```

Return the text of the Tweet at <https://twitter.com/sferik/statuses/27558893223>

```
Twitter.status(27558893223).text
```

Find the 3 most recent marriage proposals to @justinbieber

```
Twitter.search("to:justinbieber marry me", :rpp => 3, :result_type => "recent") map do |status|
  "#{status.from_user}: #{status.text}"
end
```

<http://twitter.rubyforge.org/>

twitter gem 演習

twitter gem をインストール

\$ gem install twitter

twitterでユーザーがつぶやいている「最近聞いている曲」を調べる
nowplaying という言葉入りのtweetを最新100件検索する

```
require 'twitter'  
  
    この言葉を含む      件数  
Twitter.search("nowplaying", :rpp => 100,  
               :result_type => "recent").each do |status|  
  puts "#{status.from_user}: #{status.text}"  
end      つぶやいたユーザー      つぶやき内容
```

演習解説 文字列表示

```
puts "#{status.from_user}: #{status.text}"
```

"#{変数名}" と書くと、文字列に変数の内容を埋め込むことができます。

```
name = "igarashi"
```

```
puts "Author #{name}" #=> Author igarashi
```

変数nameの中身が表示される

以下の2つのコードは同じ動作です。

```
name = "igarashi"
```

```
text = "Author #{name}"
```



```
text = "Author igarashi"
```

同じ動作

演習解説 文字列表示

Rubyには2種類の文字列リテラル(表現)があります。

ダブルクオートとシングルクオートです。

前のページで説明した変数の中身を表示する場合はダブルクオートを使う必要があります。

ダブルクオート："#{変数名}" と書くと変数の中身を表示

```
name = "igarashi"
```

```
puts "Author #{name}" #=> Author igarashi
```

変数nameの中身が表示される

シングルクオート：'#{変数名}' と書くとそのまま表示

```
name = "igarashi"
```

```
puts 'Author #{name}' #=> Author #{name}
```

変数nameの中身が表示されず、そのまま出力される

演習解説 引数のHashの省略形

この部分、見慣れない書き方をしています。

```
Twitter.search("nowplaying", :rpp => 100, :result_type => "recent")
```

省略しないで書くと以下のようになります。

```
Twitter.search("nowplaying", {:rpp => 100, :result_type => "recent"})
```

1番目の引数は文字列 2番目の引数はHashで[]を省略

Rubyでよく使われるテクニックで、たとえば複数のオプションを引数として渡したいときにHashを渡すことがあります。キーが説明文の役目をしてくれて読みやすいからです。

その際に曖昧にならなければHashの[]を省略して書けます。

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