



# メタプログラミングRuby問題集 の活用



株式会社ウィルネット  
前島 真一



株式会社SmartHR  
kinoppyd



# kinoppyd

- SmartHR のプログラマ
- メタプログラミングRuby大好き

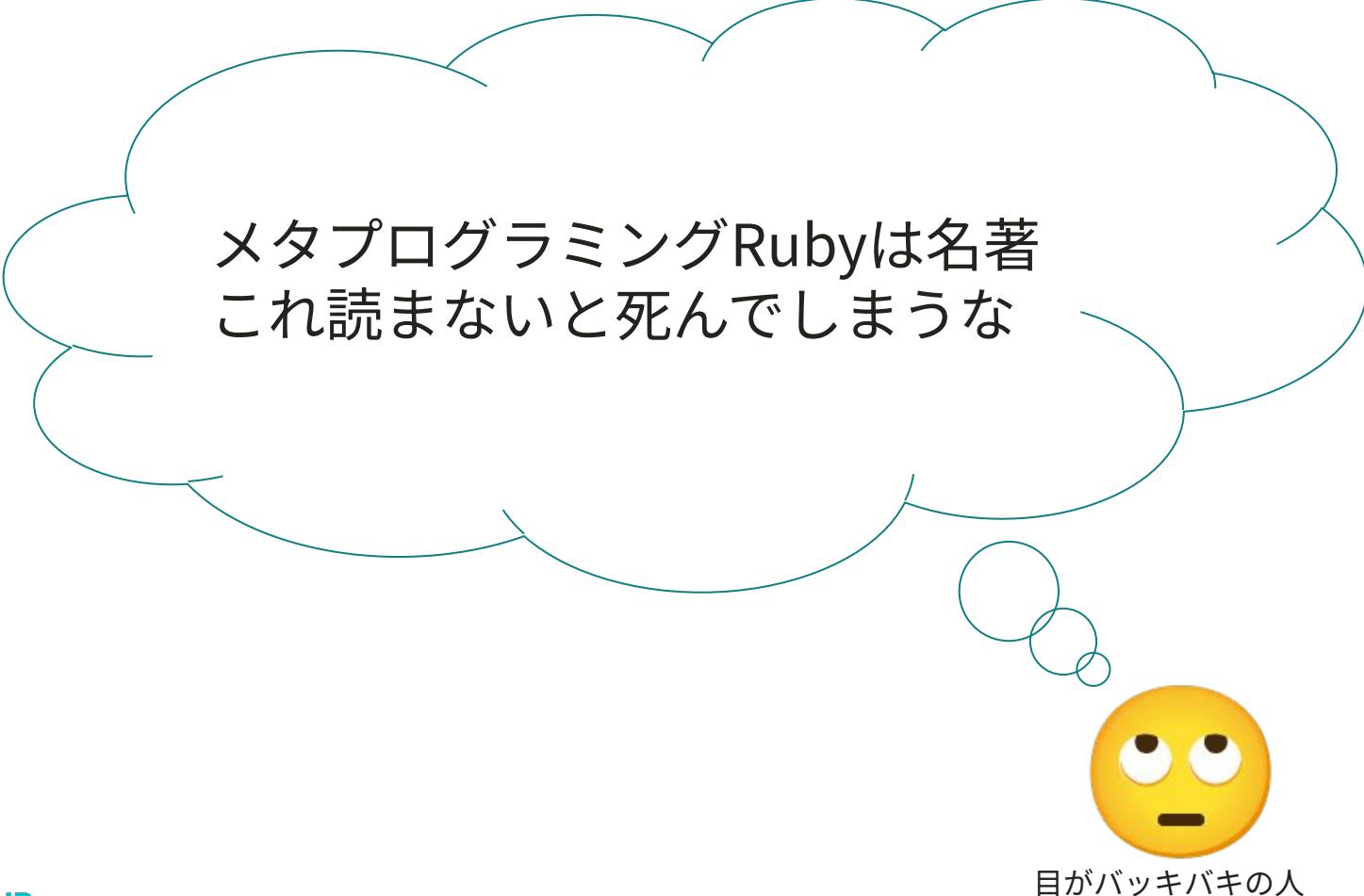


𝕏 @GhostBrain

 kinoppyd

 <https://kinoppyd.dev/blog>

メタプログラミング  
*Ruby is 名著 of 名著*



メタプログラミングRubyは名著  
これ読まないと死んでしまうな

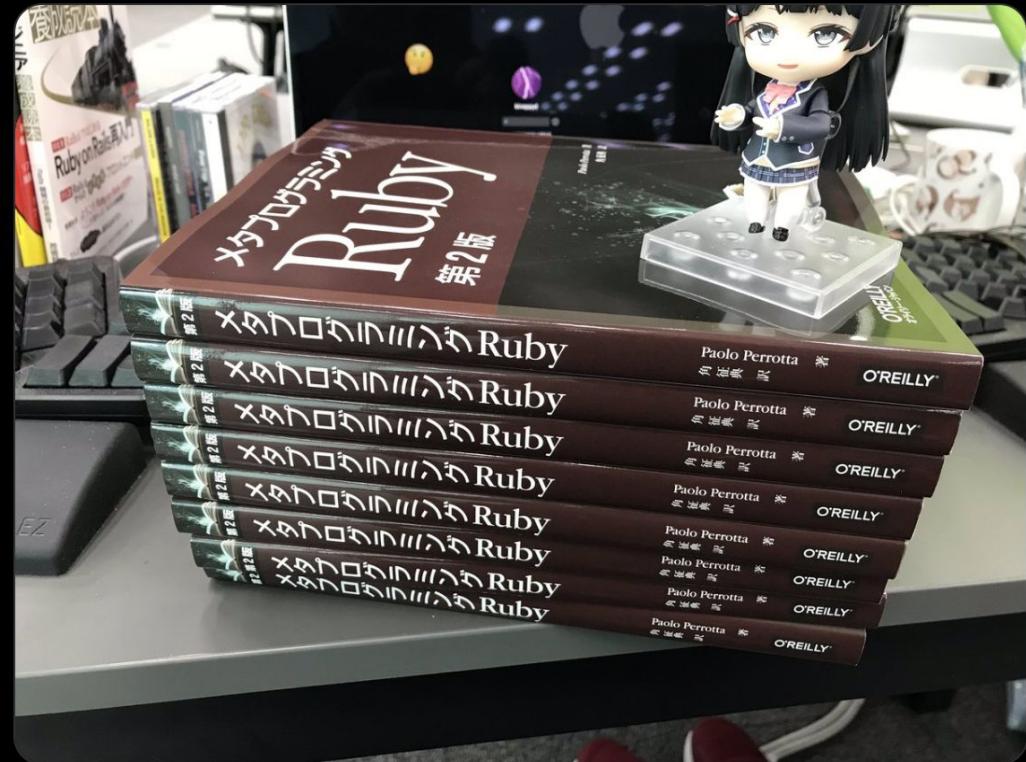
目がバッキバキの人

# 読書会やりましょう

- みんなで読むことによって底力向上
- でもこれ普通に輪読するのキツそう
- 実習形式にすればいいか

# たくさん買った

社で勉強会を開催するために買ったんですが、こんだけ同じ本が大量にあるとなんか、不思議な気持ちになる



実習！？

正気か！？

めちゃつらいぞ！？

# つらいっす、でも

- 将来JOINした人に残せるテキスト
- 作問と読書会のまとめを別の人気が担当
- PublicにすることでRubyに貢献できる

# 将来JOINした人に残せるテキスト

- 入社してくる人はRuby初めてかも
- 都度勉強会開くのはしんどい
- 鍛錬したテキストを残しておこう

# 作問と読書会のまとめを別の人気が担当

- 参加者にもグラデーションがある
- すでに詳しい人には僕と一緒に作問を
- 未読の参加者はその回のテキストまとめを

# PublicにすることでRubyに貢献できる

- 技術顧問のレビューも通ったらしいける
- 社外の目に触れてより鍛錬される
- Rubyを使う会社全体で底力あげてこ

どういう実習をしましたか？

# まずレベル分け

- 修行僧（初学者）
- ヌンチャク使い（中級者）
- マスター老師（メタプロ大好き）

# 章を要約して全員の前で講義してもらいます

- 初学者が要約
- 中級者がレビュー
- テキストは非公開（著作権に触れる）

# 用意したtestをパスしてもらいます

- メタプロ大好きなできる人が作問
- 講義の後に解いてもらう
- 問題は公開

## 実際の出題

# Q1.

# Hogeクラスは次の仕様を持つ

# "hoge" という文字列の定数Hogeを持つ

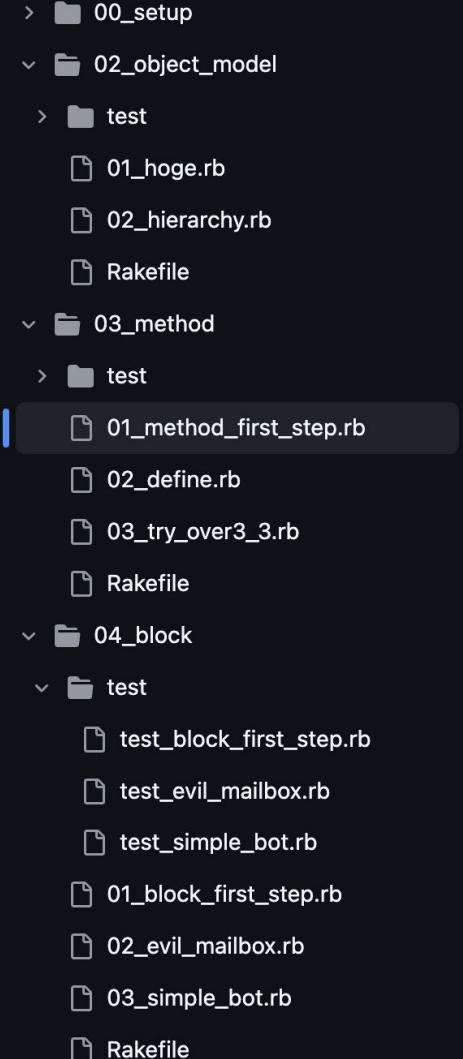
# "hoge" という文字列を返すhoge方法を持つ

# HogeクラスのスーパークラスはStringである

# 自身が"hoge"という文字列である時 (HogeクラスはStringがスーパークラスなので、当然自身は文字列である) 、trueを返すhoge?方法が定義されている

```
class Hoge
```

```
end
```



# テストをActionsでグリーンに

もう残っていませんが、当時はCIのBadgeをREADMEに貼って、全員グリーンを目指しました

## Badgesのリスト（任意）

forkした方々のなかから、ここに常に自分のバッジを表示したい人は、「まずやること」セクションで設定したバッジのURIをここに名前とともに書き、ブルリクをください。

Name	Badge
<a href="#">meganemura</a>	 Ruby no status
<a href="#">mserizawa</a>	 Ruby no status
<a href="#">wakasa51</a>	 Ruby no status
<a href="#">morizumi</a>	 Ruby no status
<a href="#">moonstruckdrops</a>	 Ruby no status
<a href="#">ykarakita</a>	 Ruby no status
<a href="#">ringo</a>	 Ruby no status
<a href="#">kabetch</a>	 Ruby no status
<a href="#">kouryou</a>	 Ruby no status
<a href="#">shunhikita</a>	 Ruby no status
<a href="#">yoshinarl</a>	 Ruby no status
<a href="#">aAnzai2017</a>	 Ruby no status
<a href="#">nagata03</a>	 Ruby no status

2020-03-13 ~ 2024-02-05

## SmartHRでのメタプログラミングRuby読書会と、その成果物

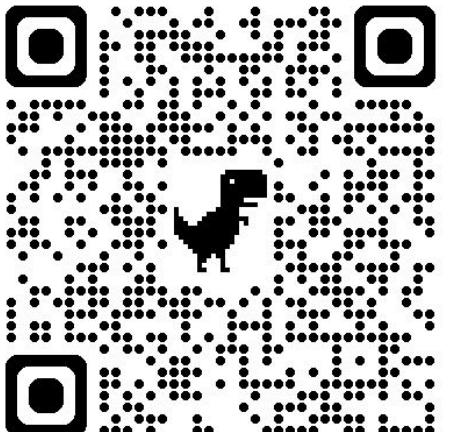
エンジニアの [kinoppyd](#) です、お久しぶりです。今日は社内 [メタプログラミングRuby 第2版](#) 読書会を開催した話と、その成果物に関して共有しようと思います。



そして何が残りましたか？

# 鍛錬された問題集！

<https://github.com/kinoppyd/reading-metaprogramming-ruby>



Star me !

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

reading-metaprogramming-ruby Public

Unpin Unwatch 1 Fork 112 Star 121

master Go to file Code

kinoppyd Merge pull request #80 fro... ec64ea4 · 2 months ago 212 Commits

.github Bump actions/checkout from 4 to 5 2 months ago

00\_setup Ruby 3.4.3にアップグレードした 6 months ago

02\_object\_model Ruby 3.4.3にアップグレードした 6 months ago

03\_method Ruby 3.4.3にアップグレードした 6 months ago

04\_block Ruby 3.4.3にアップグレードした 6 months ago

05\_class\_definition Ruby 3.4.3にアップグレードした 6 months ago

06\_codes\_generate\_codes Ruby 3.4.3にアップグレードした 6 months ago

answers fix typo 2 years ago

test CIでは解答例を利用してテストを実... 3 years ago

.ruby-version Ruby 3.4.3にアップグレードした 6 months ago

Gemfile Add minitest-reporters 5 years ago

Gemfile.lock Ruby 3.4.3にアップグレードした 6 months ago

LICENSE Add license 3 years ago

README.md READMEのRubyバージョン情報を3... 6 months ago

Rakefile 各章単位でテストを実行できるよう... 3 years ago

README WTFPL license

## reading-metaprogramming-ruby

### これはなに

このリポジトリは、[メタプログラミングRuby 第2版](#)を読んだ人向けの練習問題集です。本を読んだだけだとなかなか身につかないRubyのメタプログラミングの知識を、手を動かして理解することを目的にしています。

### 始め方

About

No description, website, or topics provided.

Readme

WTFPL license

Activity

121 stars

1 watching

112 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Contributors 25

+ 11 contributors

Languages

Ruby 100.0%

あなたも明日から  
メタプログラミング  
*Ruby*読書会が始め  
られます！

# バトンタッチ！



# Shinichi Maeshima

## Willnet Inc.

𝕏 @netwillnet

GitHub @willnet

🌐 <https://blog.willnet.in>

# "つくれる"のその先へ

つくるときの障害を取りのぞき  
楽しく継続性のあるWebサービス開発を実現します

技術顧問業をしています

[Top](#)[About](#)[Clients](#)[Service](#)[Profile](#)[Company](#)[Contact](#)

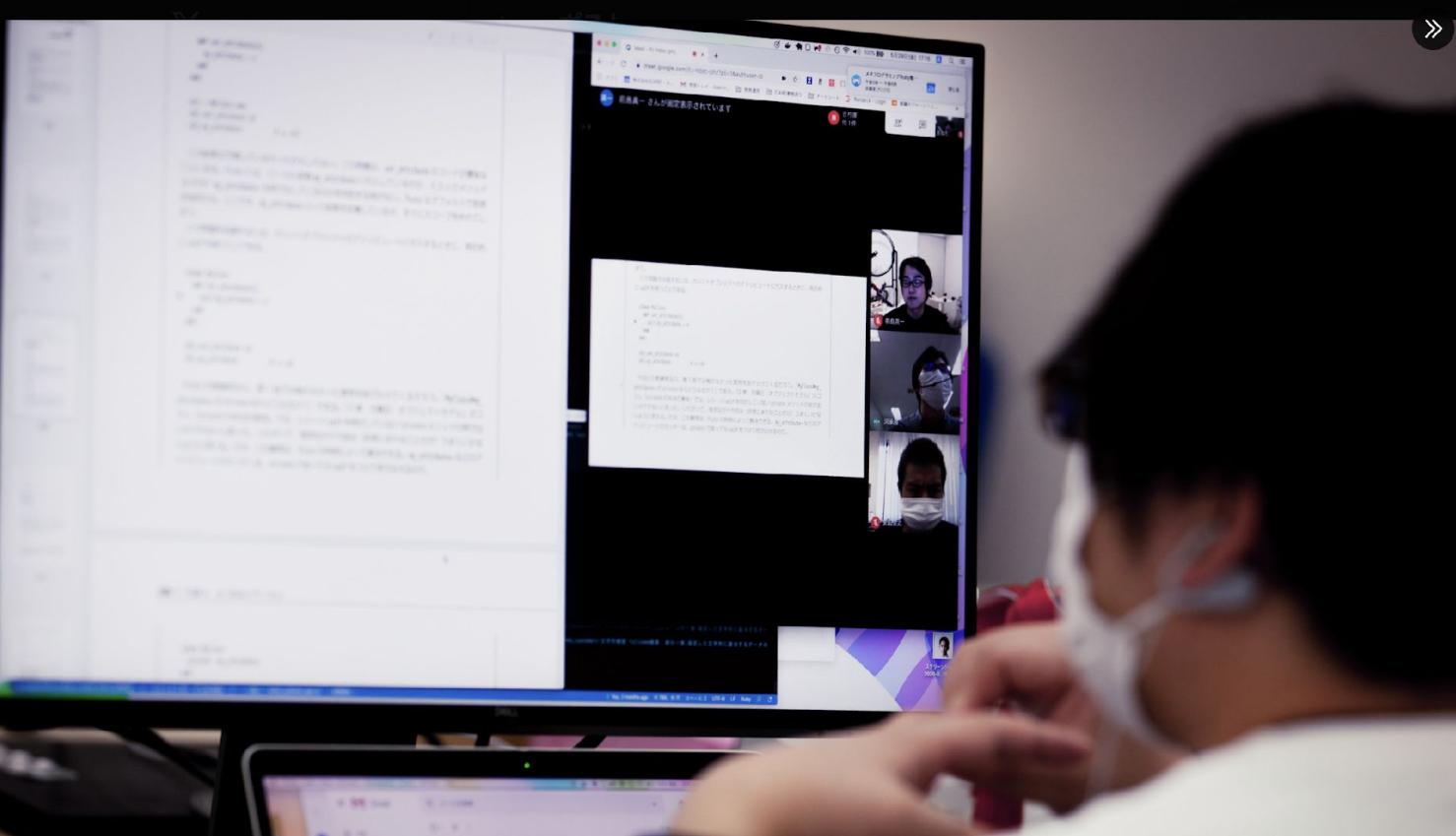
# 読書会が終わりメタプログラミング問題集が残った

- @willnetは複数社の技術顧問をしている
- 他社でもこの問題集を使ってメタプログラミングを教えようと思った

# 3社で試した

- 進め方は次のようにした
  - メタプログラミングRuby第2版を読む
  - 要約パートはなし
  - 問題を解く
- コロナ禍以降での開催だったので基本フルリモートで実施

# 実施風景1



アオキタカユキ / iCARE U...  
@dorisukeone

本日のiCARE開発部では @netwillnet さんによる「メタプログラミングRuby勉強会」。デザイナーな私としてはこの豪華な勉強会は羨ましいですな。

午後5:33 · 2020年5月29日

返信をポスト

返信

# SmartHR読書会から大きく変えた点

- GitHub上でコード差分を眺めつつ@willnetによる解説をつけていた
  - SmartHR社内の読書会では時間の問題もありとにかく解くのがメインになっていた
  - 正答は必ずしも1つとは限らない
  - 他の人が書いた別解を見ることで勉強になる
- @willnetの解答も合わせて解説した

# 会社の状況に合わせてやり方を変えてすすめた

- 基本的に週1実施
- モチベーションの高い参加者が集まっているか、みんな業務で忙しいかなどで細かい内容を調整した
- 予習ありで30分で終える
  - 先に本を読んで感想を話す
  - 先に問題を解いて参加者の解答例を見ながら解説をする
- 予習無しで1時間時間取る
  - みんなで本を音読する
  - もくもく問題を解く

# 他社での読書会を重ねるごとに問題が洗練されていく

- 実際に問題をすすめるとtypoや、問題の裏をかくような解法が見つかる
- GitHub上で公開されているため、気づいた人が修正用のPRを出せる



The screenshot shows a GitHub pull request page for the repository `reading-metaprogramming-ruby`. The pull request is titled "A2のメソッドをクラスに対して定義しないようにするテストを追加 #58". The status bar indicates the pull request is "Merged" and was merged into `kinoppyd:master` from `expajp:add_test_for_define` on Aug 24, 2020. The page displays the following details:

- Conversation:** 5 comments
- Commits:** 4
- Checks:** 0
- Files changed:** 2
- Reviewers:** No reviews
- Assignees:** No one—assign yourself

The pull request includes a comment from `expajp` dated Aug 23, 2020, which reads:

```
define.rb の A2 は以下のように実装するように指示されています。  
initializeに渡した配列に含まれる値に対して、"hoge_" をprefixを付与したメソッドが存在すること
```

# 参加者の感想

- 難しい！
  - 基礎問題を解かずにいきなり応用問題が始まっている
  - 対応する章に出てこないメソッドを使わないと解けない問題が時折ある
- コードの差分を眺めることや@willnetの解説は好評だったはず

# 対策

- 応用問題前に肩慣らしができる基礎問題を追加した
- willnet/reading-metaprogramming-ruby-first-steps
  - 応用問題を解くために必要なメソッドの素振りをする形の問題
  - @willnetが解説するときに楽ができるようにその場で読める解説と回答例もつけた

# 基礎問題例

```
# Q1.
# 次の動作をする F1 class を実装する
# - 1. "def"キーワードを使わずにF1クラスにhelloインスタンスメソッドを定義すること
#       戻り値は "hello" であること
# - 2. "def"キーワードを使わずにF1クラスにworldクラスメソッドを定義すること
#       戻り値は "world" であること
# - 3. 定義していないメソッドを実行したときにエラーを発生させず、"NoMethodError"という文字列を返すこと
# - 4. `F1.new.respond_to?(定義していないメソッド名)` を実行したときにtrueを返すこと

class F1
end
```

```
# Q1. 問題の解説
#
# define_methodとdefine_singleton_methodとmethod_missingの素振り用の問題です。
# define_singleton_methodは3章にはまだ出てきていませんが、これを知らないと3章の問題を解くのが難しくなるので覚えておいてください
# respond_to_missing?は、respond_to?メソッド実行時にメソッドが定義されていない場合に呼ばれるメソッドです。method_missingを定義する
# 場合は
# 必ず定義しておきましょう。
#
class F1
  define_method :hello do
    'hello'
  end

  define_singleton_method :world do
    'world'
  end

  def method_missing(*args)
    'NoMethodError'
  end

  def respond_to_missing?(*args)
    true
  end
end
```

# 実施風景2

# 基礎問題を作つてからのメタプロ読書会

- 応用問題はやっぱり難しいけど好評
- @willnetのお手伝い先に提供するだけであればこれで良さそう

# でもこれでいいのかな？

- でもこんな良いコンテンツを自分の目の届く範囲だけしか提供できないというのはもったいない
- 身近に技術顧問がいないひとでも問題に取り組めるようにしたい



# 誰でもメタaproblem問題に着手できるようにした #69

Merged kinoppyd merged 36 commits into master from revise on Mar 30, 2023

Conversation 9

Commits 36

Checks 0

Files changed 54

Edit



willnet commented on Mar 8, 2023

Collaborator

## やりたいこと

@willnet は複数の会社でこのメタaproblem問題を解く会を主催してきました。参加したひとの反応も良くやってよかったと思っていますが、僕が主催している補足しないとこの会は成り立たせるのが難しいな～という感想を持っています。

せっかく良い問題があるので、広く色んなひとに解いてほしい。そこでドキュメントや解説などを整えて、興味がある人が誰でも詰まらずに問題を解けるようにするのがこのPRの目的です。

## これまでのメタaproblem問題の主な課題

- Ruby 2.6が対象(現時点では3.2がリリースされているし2.6はdeprecated)
- READMEが当時のメタaproblem本読書会参加者向けの内容になっているため、新規参加者は何をどうやつたらいいかわかりづらい
- 問題そのものがかなり難しみ
- さらに問題の回答例や解説がないので、詳しい人がいないと詰まる可能性が高い

### Reviewers

kinoppyd

### Assignees

No one—assign yourself

### Labels

None yet

### Projects

None yet

### Milestone

No milestone



# 誰でもメタプロト問題に着手できるようにした #69

Merged kinoppyd merged 36 commits into master from revise  on Mar 30, 2023

## ● READMEに進め方を追加

Conversation 9 Commits 36 Checks 0 Files changed 54 +1,154

## ● 基礎問題集をマージ

Collaborator

### やりたいこと

## ● 解答例と解説の追加

お世話になりました。このPRは問題の発表でのメタプロト問題解説会を開く会を準備で進みました。参加したひとの反応も良くやってよかったと思っていますが、僕が主催しているいる補足しないとこの会は成り立たせるのが難しいな~という感想を持っています。

せっかく良い問題があるので、広く色んなひとに解いてほしい。そこでドキュメントや解説などを整えて、興味がある人が誰でも詰まらずに問題を解けるようにするのがこのPRの目的です。

## これまでのメタプロト問題の主な課題

- Ruby 2.6が対象(現時点では3.2がリリースされているし2.6はdeprecated)
- READMEが当時のメタプロト本読書会参加者向けの内容になっているため、新規参加者は何をどうやつたらいいかわかりづらい
- 問題そのものがかなり難しみ
- さらに問題の回答例や解説がないので、詳しい人がいないと詰まる可能性が高い

### Reviewers

kinoppyd

### Assignees

No one—assign yourself

### Labels

None yet

### Projects

None yet

### Milestone

No milestone

2023-05-30

## Rubyのメタプログラミング問題集をブラッシュアップした話

こんにちは。SmartHRでRails顧問業をしている[willnet](#)です。以前、[SmartHRでのメタプログラミングRuby読書会と、その成果物](#)というエントリを([kinoppyd](#)さんが)書いていました。今回のエントリはその続きの話です。

### 実際に手を動かすと身につく

上記エントリでは[メタプログラミングRuby 第2版](#)の読書会を、単に本を読むだけでなく、毎週新しい問題を作り参加者に解いてもらう、という流れで開催したことを紹介しました。

Rubyによるメタプログラミングは、普段の仕事ではなかなか使う機会がないテクニックも多く<sup>\*1</sup>、一回本を通読しただけですべて覚えるのは難しいと感じています。しかし、その読書会では実際に手を動かして解く問題を用意したため、通常の読書会よりも内容が身につく度合いが大きかったのではないか、と感じました。

# 一定の認知は得たはず？

B!

キーワード・URLを検索



マイページ

ブックマーク

あとで読む

追加

ツー

総合

一般

世の中

政治と経済

暮らし

学び

テクノロジー

おもし

> テクノロジー > Rubyのメタプログラミング問題集をブラッシュアップした話 - SmartHR Tech Blog

## Rubyのメタプログラミング問題集をブラッシュアップした話 - SmartHR Tech Blog

テクノロジー 記事元: [tech.smarthr.jp](https://tech.smarthr.jp)

41users がブックマーク 1



コメントを入力してください (省略可)

# 採用にも効果があったらしい

≡ ⊞ ⇠ ↵ :



12:01 PM

「この記事でRubyを勉強して内容がとてもよかったですSmartHRを応募しようと思った」という学生候補者の方が面談に来てくださいました～！！！ ありがたう～

「メタプロの記事書いてくれて本当にありがとうございます！記事とても楽しみにしています！」とのことです！

[SmartHRでのメタプログラミングRuby読書会と、その成果物 - SmartHR Tech Blog](#)

[Rubyのメタプログラミング問題集をブラッシュアップした話 - SmartHR Tech Blog](#)

いいね 9 リツイート 16 タグ 1

6 replies Last reply 2 years ago



2:33 PM

そういえば

この前カジュアル面談した人がメタプロ問題集に感謝してました！

いいね 1



3:22 PM

@kinoppyd ( ) 共有

今日面接した新卒の方が [SmartHRでのメタプログラミングRuby読書会と、その成果物](#) を読んで、SmartHRという会社をはじめて知ったとのことでした～

問題もとても参考になったので作ってくれた方に感謝！！という感じだったので共有する (edited)

SmartHR Tech Blog

[SmartHRでのメタプログラミングRuby読書会と、その成果物 - SmartHR Tech Blog](#)

エンジニアのkinoppydです、お久しぶりです。今日は社内メタプログラミングRuby第2版読書会を開催した話と、その成果物に関して共有しようと思います。

SmartHR社内での勉強会 社内では、いくつかの勉強会や読書会が開催されています。業務で必要な知識をみんなで増やして学習する目的でねったり、単に有志で集ま

でももっと色んな人に解いてもらいたい

...

# そんなときに現れたruby.wasm

ruby / ruby.wasm

Type  to search

Code Issues 31 Pull requests 5 Discussions Actions Wiki Security Insights

 ruby.wasm Public

Watch 20 Fork 60 Star 797

main 7 Branches 1375 Tags

Go to file Add file Code

 kateinoigakukun rake bump\_dev\_version ✓ 0b59fd9 · 4 days ago 1,518 Commits

.github ci: add missing checkout step in purge cache workflow 2 weeks ago

benchmarks Introduce rbwasm pack command as an alias of wasi-vfs ... 2 years ago

bin Move all internal tools under ./bin/ directory last year

builders Install Ruby 3.4 as baseruby for emscripten builder 5 months ago

docs Bump version to 2.7.2 4 days ago

exe Initial RubyGems/Bundler support 2 years ago

ext/ruby\_wasm Update wasi-vfs to stop encoding host env vars in the mo... 4 days ago

About

ruby.wasm is a collection of WebAssembly ports of the CRuby.

 [ruby.github.io/ruby.wasm](https://ruby.github.io/ruby.wasm)

ruby webassembly wasm  
emscripten wasi

Readme  
MIT license  
Contributing  
Activity  
Custom properties  
797 stars

ブラウザで問題を解けるようになればもっと  
メタプロ問題集にチャレンジする人が増えるの  
では？

英語版を作れば世界中の人がメタプロ問題集にチャレンジするのでは？

Section: 00\_setup

Problem: TryOut

## Detailed Specification

### TryOut Class Specifications

The constructor accepts 2 or 3 arguments. The arguments are first name, middle name, and last name in that order, with middle name being optional.

It has a `full_name` method. This returns a string that combines first name, middle name, and last name with a single space. However, when middle name is omitted, only one space is placed between first name and last name.

It has a `first_name=` method. This replaces the first name with the content of the argument.

It has an `upcase_full_name` method. This returns the result of `full_name` method in all uppercase. This method has no side effects.

It has an `upcase_full_name!` method. This is a version of `upcase_full_name` with side effects, changing first name, middle name, and last name to all uppercase, and the object remembers this state

Enter your code here

Test Result

```
1 class TryOut
2 end
3
```

Run Test

Reset

Show Answer

<https://willnet.github.io/metaprogramming-challenges-in-ruby/>

# Metaprogramming Challenges in Ruby

[About](#)[English](#)[日本語](#)

Section: 00\_setup

Problem: TryOut

英語と日本語を切り替え

## Detailed Specification

### TryOut Class Specifications

The constructor accepts 2 or 3 arguments. The arguments are first name, middle name, and last name in that order, with middle name being optional.

It has a `full_name` method. This returns a string that combines first name, middle name, and last name with a single space. However, when middle name is omitted, only one space is placed between first name and last name.

It has a `first_name=` method. This replaces the first name with the content of the argument.

It has an `upcase_full_name` method. This returns the result of `full_name` method in all uppercase. This method has no side effects.

It has an `upcase_full_name!` method. This is a version of `upcase_full_name` with side effects, changing first name, middle name, and last name to all uppercase, and the object remembers this state

Enter your code here

Test Result

```
1 class TryOut
2 end
3
```

Run Test

Reset

Show Answer

# Metaprogramming Challenges in Ruby

[About](#)[English](#)[日本語](#)**Section:**

00\_setup

**Problem:**

TryOut

## Detailed Specification

### TryOut Class Specifications

The constructor accepts 2 or 3 arguments. The arguments are first name, middle name, and last name in that order, with middle name being optional.

It has a `full_name` method. This returns a string that combines first name, middle name, and last name with a single space. However, when middle name is omitted, only one space is placed between first name and last name.

It has a `first_name=` method. This replaces the first name with the content of the argument.

It has an `upcase_full_name` method. This returns the result of `full_name` method in all uppercase. This method has no side effects.

It has an `upcase_full_name!` method. This is a version of `upcase_full_name` with side effects, changing first name, middle name, and last name to all uppercase, and the object remembers this state

問題の切り替え

## Enter your code here

```
1 class TryOut
2 end
3
```

## Test Result

[Run Test](#)[Reset](#)[Show Answer](#)

Section:

00\_setup

Problem:

TryOut

問題文

## Detailed Specification

TryOut Class Specifications



The constructor accepts 2 or 3 arguments. The arguments are first name, middle name, and last name in that order, with middle name being optional.

It has a `full_name` method. This returns a string that combines first name, middle name, and last name with a single space. However, when middle name is omitted, only one space is placed between first name and last name.

It has a `first_name=` method. This replaces the first name with the content of the argument.

It has an `upcase_full_name` method. This returns the result of `full_name` method in all uppercase. This method has no side effects.

It has an `upcase_full_name!` method. This is a version of `upcase_full_name` with side effects, changing first name, middle name, and last name to all uppercase, and the object remembers this state

Enter your code here

Test Result

```
1 class TryOut
2 end
3
```

Run Test

Reset

Show Answer

Section: 00\_setup

Problem: TryOut

## Detailed Specification

### TryOut Class Specifications

The constructor accepts 2 or 3 arguments. The arguments are first name, middle name, and last name in that order, with middle name being optional.

It has a `full_name` method. This returns a string that combines first name, middle name, and last name with a single space. However, when middle name is omitted, only one space is placed between first name and last name.

It has a `first_name=` method. This replaces the first name with the content of the argument.

It has an `upcase_full_name` method. This returns the result of `full_name` method in all uppercase. This method has no side effects.

It has an `upcase_full_name!` method. This is a version of `upcase_full_name` with side effects, changing first name, middle name, and last name to all uppercase, and the object remembers this state

Enter your code here

コードを書く  
Test Result

```
1 class TryOut
2 end
3
```

Run Test

Reset

Show Answer

Section: 00\_setup

Problem: TryOut

## Detailed Specification

### TryOut Class Specifications

The constructor accepts 2 or 3 arguments. The arguments are first name, middle name, and last name in that order, with middle name being optional.

It has a `full_name` method. This returns a string that combines first name, middle name, and last name with a single space. However, when middle name is omitted, only one space is placed between first name and last name.

It has a `first_name=` method. This replaces the first name with the content of the argument.

It has an `upcase_full_name` method. This returns the result of `full_name` method in all uppercase. This method has no side effects.

It has an `upcase_full_name!` method. This is a version of `upcase_full_name` with side effects, changing first name, middle name, and last name to all uppercase, and the object remembers this state

Enter your code here

テスト実行

Test Result

```
1 class TryOut
2 end
3
```

Run Test

Reset

Show Answer

Section: 00\_setup

Problem: TryOut

## Detailed Specification

### TryOut Class Specifications

The constructor accepts 2 or 3 arguments. The arguments are first name, middle name, and last name in that order, with middle name being optional.

It has a `full_name` method. This returns a string that combines first name, middle name, and last name with a single space. However, when middle name is omitted, only one space is placed between first name and last name.

It has a `first_name=` method. This replaces the first name with the content of the argument.

It has an `upcase_full_name` method. This returns the result of `full_name` method in all uppercase. This method has no side effects.

It has an `upcase_full_name!` method. This is a version of `upcase_full_name` with side effects, changing first name, middle name, and last name to all uppercase, and the object remembers this state

Enter your code here

テスト結果

```
1 class TryOut
2 end
3
```

Test Result

Run Test

Reset

Show Answer

Section: 00\_setup

Problem: TryOut

## Detailed Specification

### TryOut Class Specifications

The constructor accepts 2 or 3 arguments. The arguments are first name, middle name, and last name in that order, with middle name being optional.

It has a `full_name` method. This returns a string that combines first name, middle name, and last name with a single space. However, when middle name is omitted, only one space is placed between first name and last name.

It has a `first_name=` method. This replaces the first name with the content of the argument.

It has an `upcase_full_name` method. This returns the result of `full_name` method in all uppercase. This method has no side effects.

It has an `upcase_full_name!` method. This is a version of `upcase_full_name` with side effects, changing first name, middle name, and last name to all uppercase, and the object remembers this state

## 解答例と解説の表示

Enter your code here

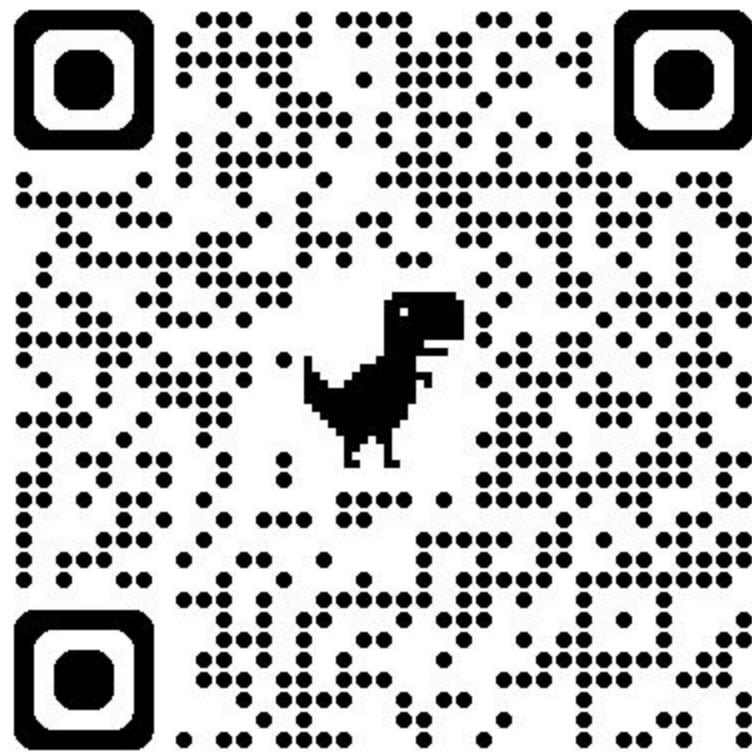
Test Result

```
1 class TryOut
2 end
3
```

Run Test

Reset

Show Answer



# 仕組み

- あらかじめ用意しているテストコード(minitest)とフォームに入力したコードをあわせて都度実行する
- 実行した後の標準出力を画面に表示する
  - 標準入出力はデフォルトだとconsole.log()としてコンソール表示されるので、jsで扱えるようにRubyVMを修正する必要があった
- 上記以外は素直にruby.wasmを利用するだけでいけた

# 実装

- 可能な限りAIエージェント(主にClaude Code)を利用してみた

# AIエージェントでwasm版を作った感想

- GitHub上の問題をそのままwebにするだけでは扱いづらかったのをAIエージェントを利用して解決した
  - 例えば1つの問題として6つのコードを要求する問題がある
  - web上で解きやすいように1つずつにバラした
  - エージェントに任せたが問題の切れ目がわからていなかったり等指示出しが難しい
  - バラしたことによりテストの再構成も必要になりエージェントは詰まりがち
- 英語化等はすんなりいった

# 今後の展望

- 英語圏での宣伝

# まとめ

- どのような形でメタプログラミングRuby問題集が生まれ、改善されたかを話しました
- 問題を作るというのは大変だけど本を読むだけよりも知識として定着しやすい
- 自分たち以外の人にも残せる
- 採用にも一定影響ある模様

みなさんも問題作つ  
てみませんか？