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

2025/11/07


SmartHR

kinoppyd

- SmartHR のプログラマ
- メタプログラミングRuby大好き

@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


kinoppyd
@GhostBrain
プロモーションする
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
メタプログラミング Ruby 第2版
Paolo Perrotta
O'REILLY
午前11:21 · 2019年11月25日

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）

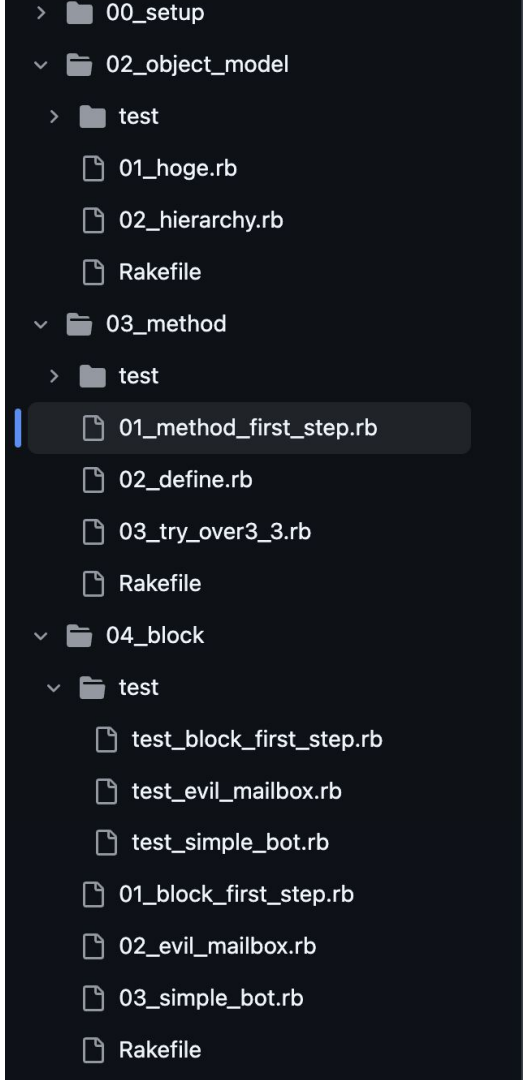
用意したtestをパスしてもらいます

- 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hogehogeメソッドを持つ
# 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?メソッドが定義されている

class Hoge
end
```



https://github.com/kinoppyd/reading-metaprogramming-ruby/blob/master/02_object_model/01_hoge.rb

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

SmartHR Tech Blog

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。

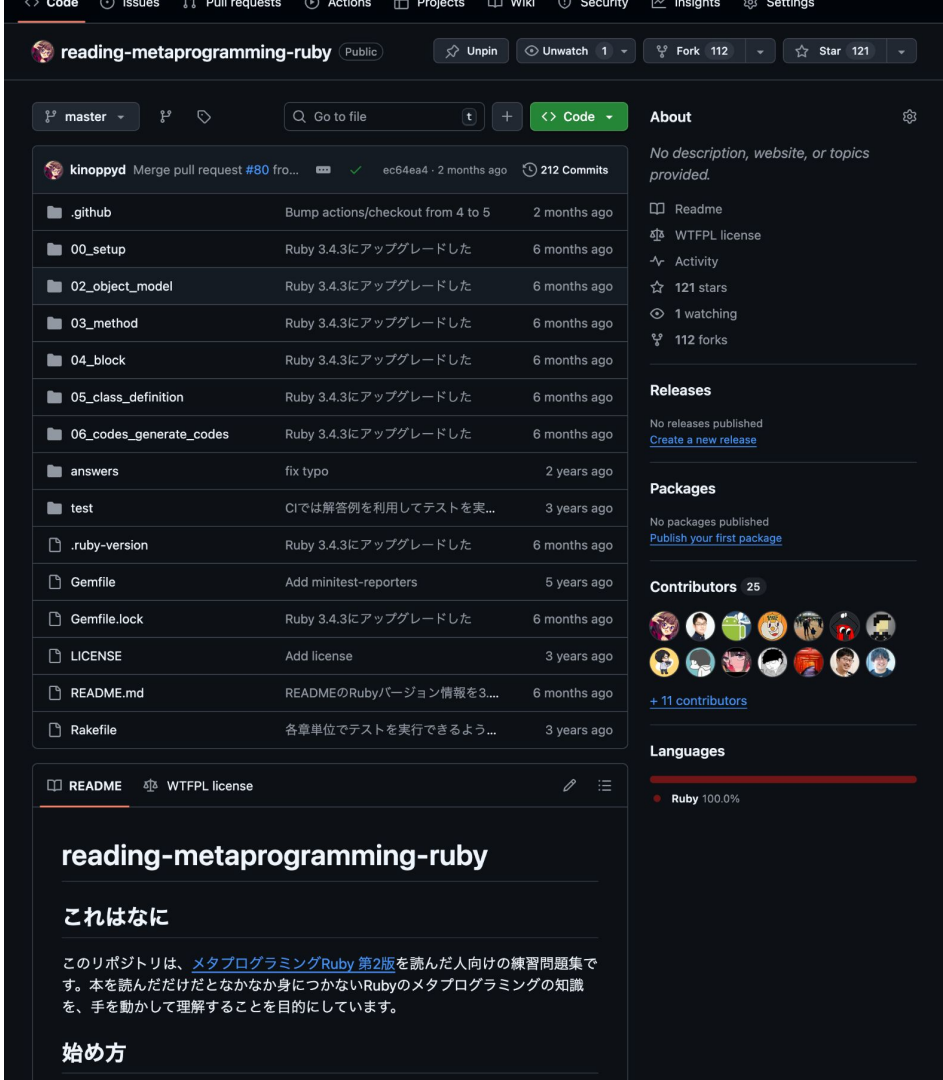
https://tech.smarthr.jp/entry/2020/03/13/123745

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 !



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.

@netwillnet

@willnet

https://blog.willnet.in

willnet

Contact

"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

Top
About
Clients
Service
Profile
Company
Contact

技術顧問業をしています

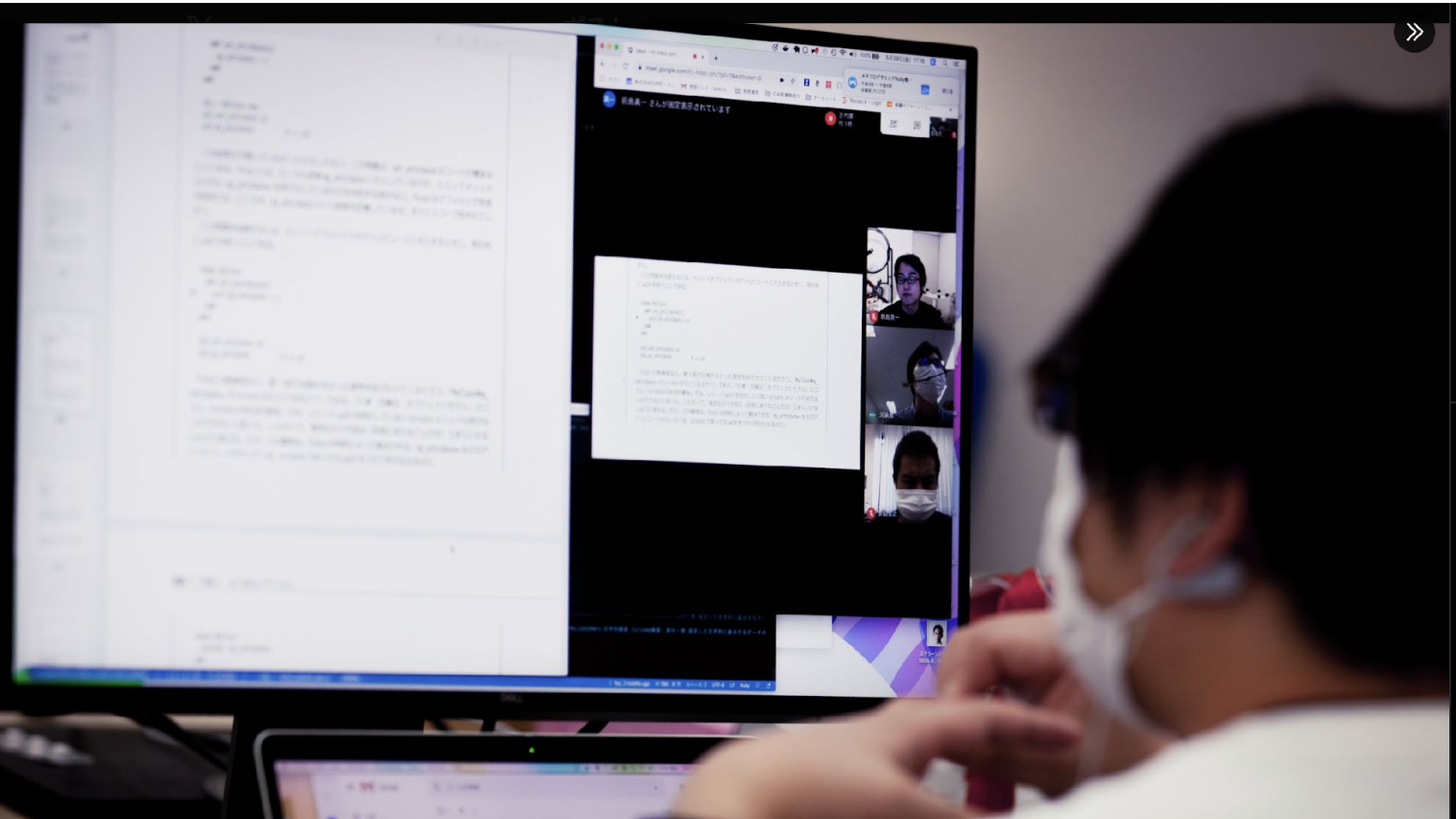
読書会が終わりメタプログラミング問題集が残った

- @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
4
10
返信をポスト
返信

SmartHR読書会から大きく変えた点

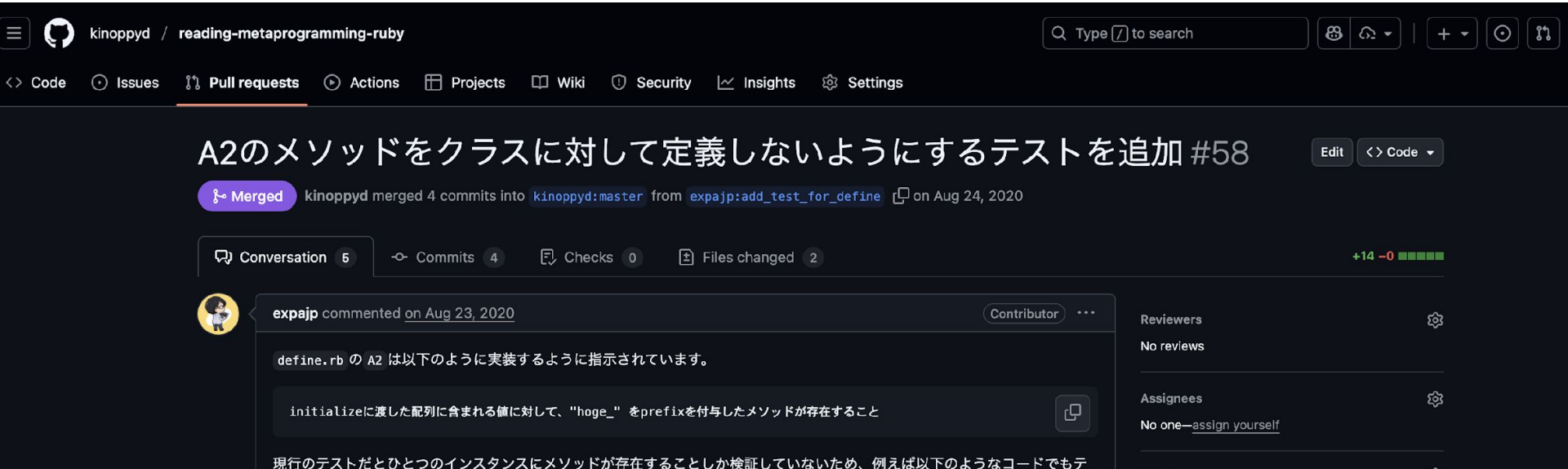
- GitHub上でコード差分を眺めつつ@willnetによる解説をつけた
  - SmartHR社内の読書会では時間の問題もありとにかく解くのがメインになっていた
  - 正答は必ずしも1つとは限らない
  - 他の人が書いた別解を見ることで勉強になる
- @willnetの解答も合わせて解説した

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時間時間を取る
  - 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



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
```

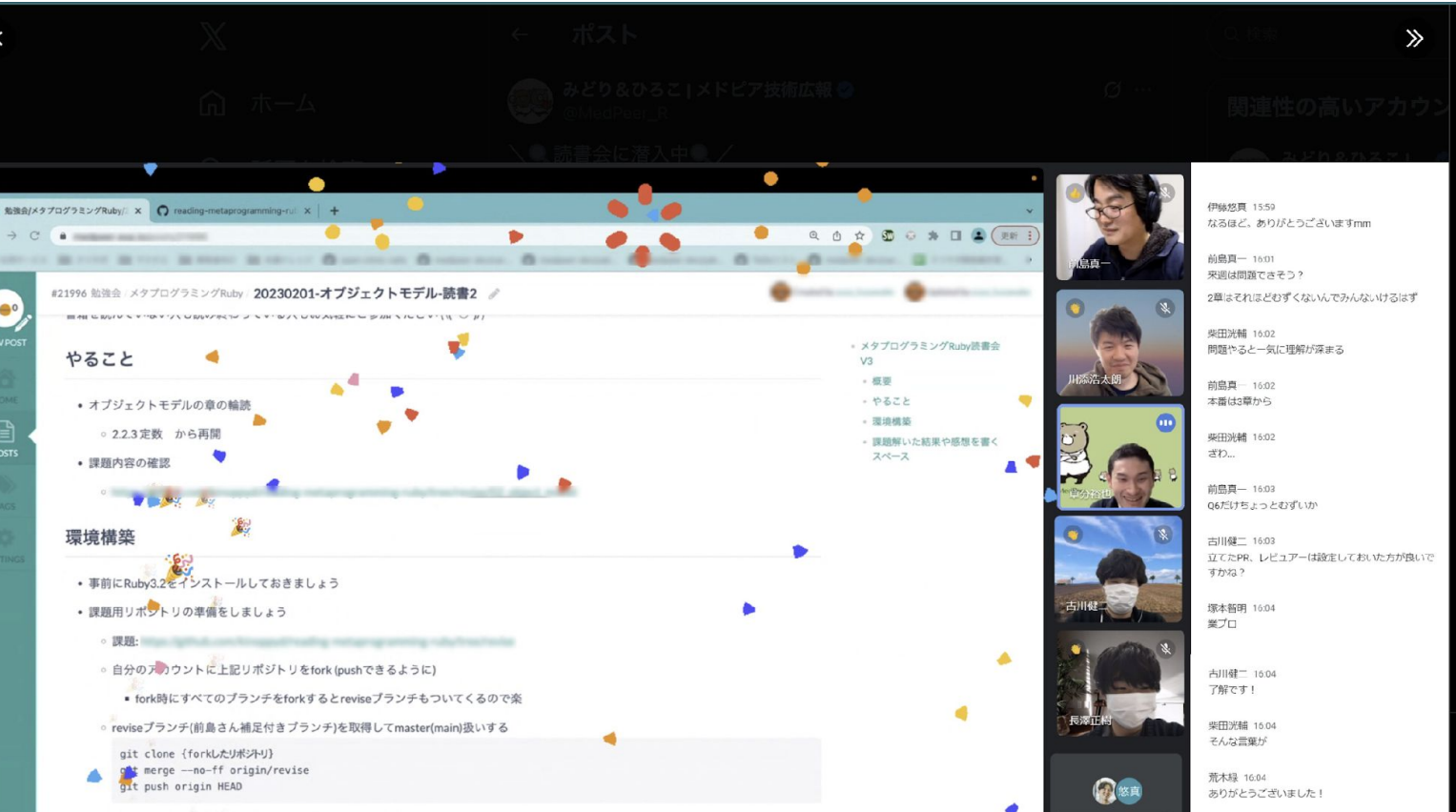
```ruby
# Q1. 問題の解説
#
# define_methodとdefine_singleton_methodとmethod_missingの素振り用の問題です。
# define_singleton_methodは3章にはまだ出てきていませんが、これを知らないと3章の問題を解くのが難しくなるので覚えておいてください
# respond_to_missing?は、respond_to?メソッド実行時にメソッドが定義されていない場合に呼ばれるメソッドです。method_missingを定義する場合は
# 必ず定義しておきましょう。
#
class F1
  define_method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


#21996 勉強会 / メタプログラミングRuby / 20230201-オブジェクトモデル-読書2
やること
オブジェクトモデルの章の輪読
2.2.3 定数　から再開
課題内容の確認
環境構築
事前にRuby3.2をインストールしておきましょう
課題用リポジトリの準備をしましょう
課題:
自分のアカウントに上記リポジトリをfork (pushできるように)
fork時にすべてのブランチをforkするとreviseブランチもついてくるので楽
reviseブランチ(前島さん補足付きブランチ)を取得してmaster(main)扱いする
git clone {forkしたリポジトリ}
git merge --no-ff origin/revise
git push origin HEAD
メタプログラミングRuby読書会 V3
概要
やること
環境構築
課題解いた結果や感想を書くスペース
伊藤悠真 15:59
なるほど、ありがとうございますmm
前島真一 16:01
来週は問題できそう？
2章はそれほどむずくないんでみんないけるはず
柴田洸輔 16:02
問題やると一気に理解が深まる
前島真一 16:02
本番は3章から
柴田洸輔 16:02
ざわ...
前島真一 16:03
Q6だけちょっとむずいか
古川健二 16:03
立てたPR、レビュアーは設定しておいた方が良いですかね？
塚本晋明 16:04
楽プロ
古川健二 16:04
了解です！
柴田洸輔 16:04
そんな言葉が
荒木緑 16:04
ありがとうございました！


みどり&ひろこ | メドピア...
@MedPeer_R
＼🔍読書会に潜入中🔍／
エンジニアが週1で自主開催している読書会に潜入！
今回は20人近くのエンジニアが参加していました。
Rubyに関する著書
『メタプログラミングRuby』を読み進めており、
次回はみんなで問題を解いていくんだそう✊
#メドピア #エンジニア #Ruby
🟢みどり🟢
午後6:13 · 2023年2月1日 · 2,314 件の表示
返信をポスト
返信

基礎問題を作っ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

誰でもメタプロ問題に着手できるようにした #69

Edit

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

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

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プロ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、僕が主催していろいろ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ったらいいかわかりづらい
- 問題そのものがかなり難しめ
- 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

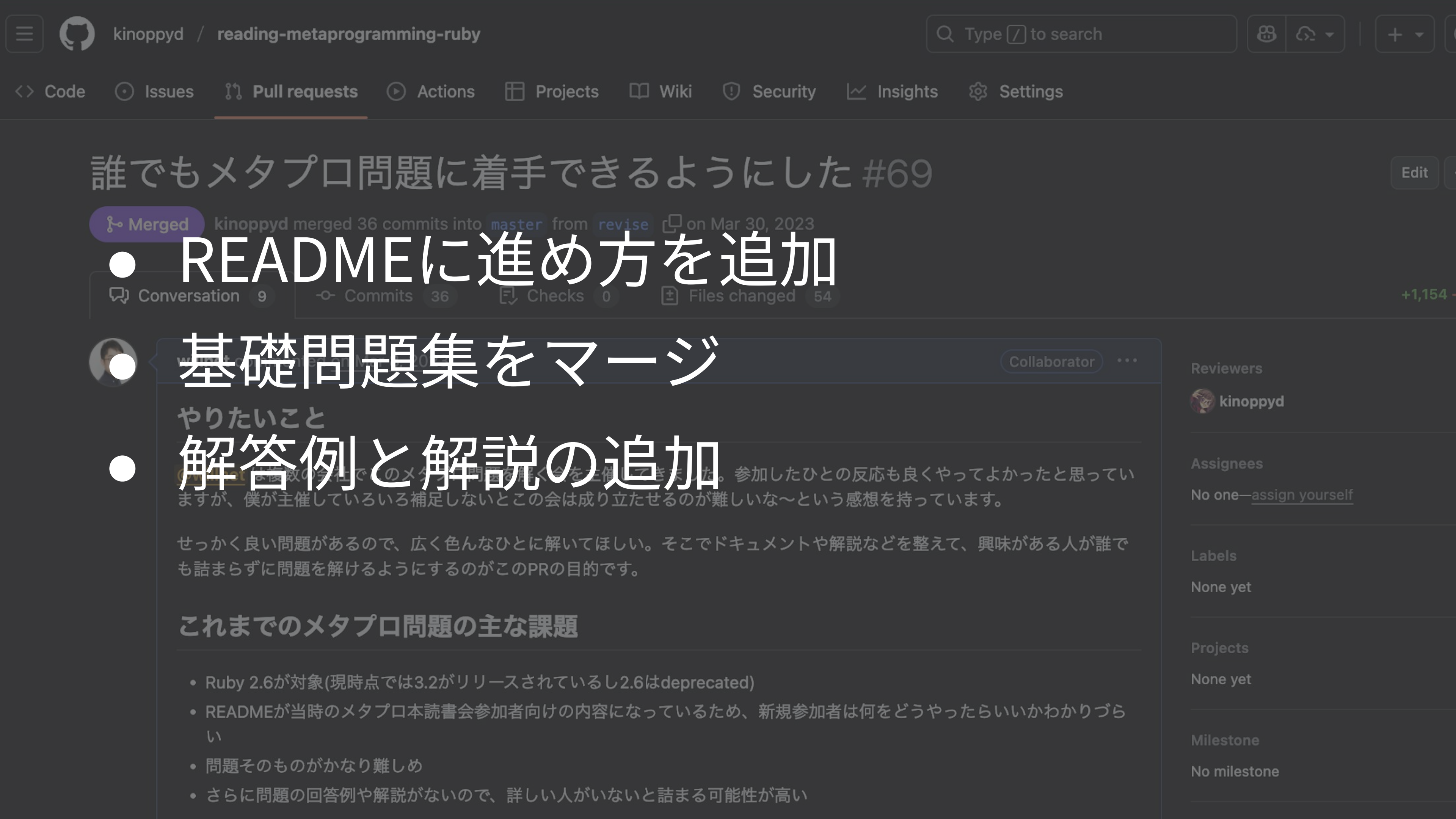
Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

- README に進め方を追加
- 基礎問題集をマージ
- 解答例と解説の追加



SmartHR Tech Blog

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

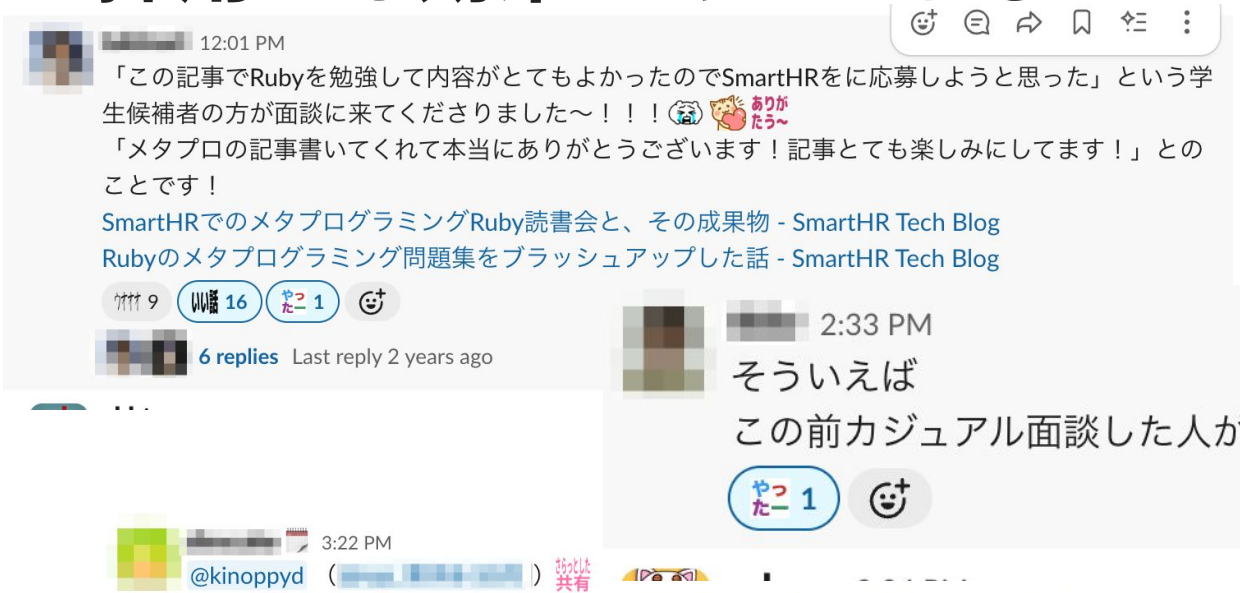
上記エントリでは メタプログラミングRuby 第2版の読書会を、単に本を読むだけでなく、毎週新しい問題を作り参加者に解いてもらう、という流れで開催したことを紹介しました。

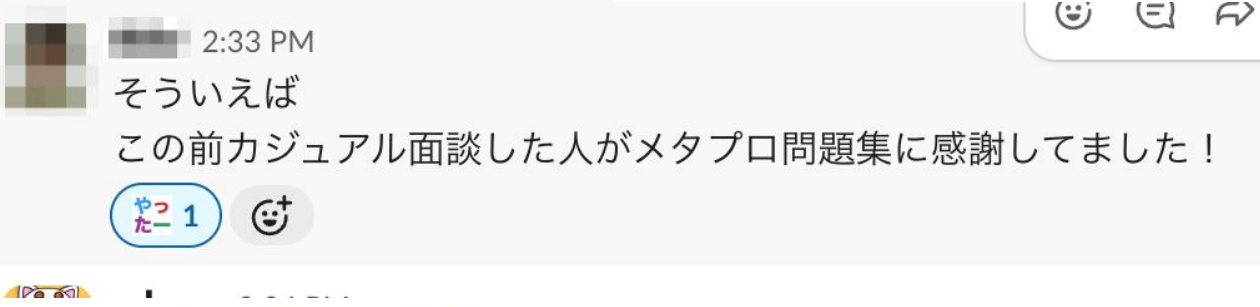
Rubyによるメタプログラミングは、普段の仕事ではなかなか使う機会がないテクニックも多く*1、一回本を通読しただけですべてを覚えるのは難しいと感じています。しかし、その読書会では実際に手を動かして解く問題を用意したため、通常の読書会よりも内容が身につく度合いが大きかったのではないか、と感じました。

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
総合
一般
世の中
政治と経済
暮らし
学び
テクノロジー
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
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)

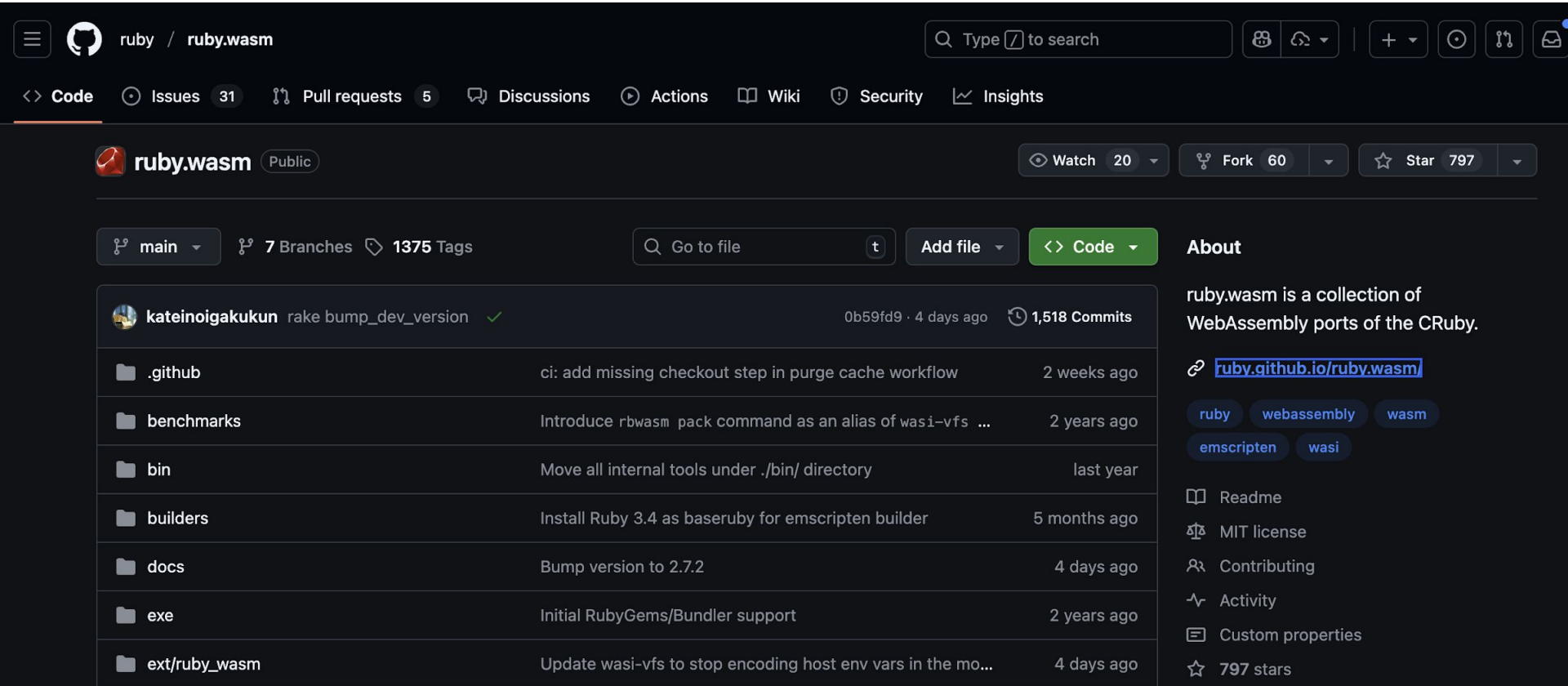
採用にも効果があったらしい





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



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？

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

Enter your code here

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

Run Test | Reset | Show Answer

Test Result

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

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

Run Test Reset Show Answer

Test Result

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

Enter your code here

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

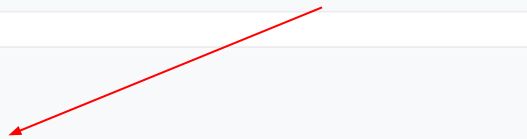
Run Test Reset Show Answer

Test Result

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

Enter your code here

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

Run Test Reset Show Answer

Test Result

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

Enter your code here

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

Run Test Reset Show Answer

Test Result

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

テスト実行

Enter your code here

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

Run Test Reset Show Answer

Test Result

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

テスト結果

Enter your code here

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

Run Test Reset Show Answer

Test Result

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

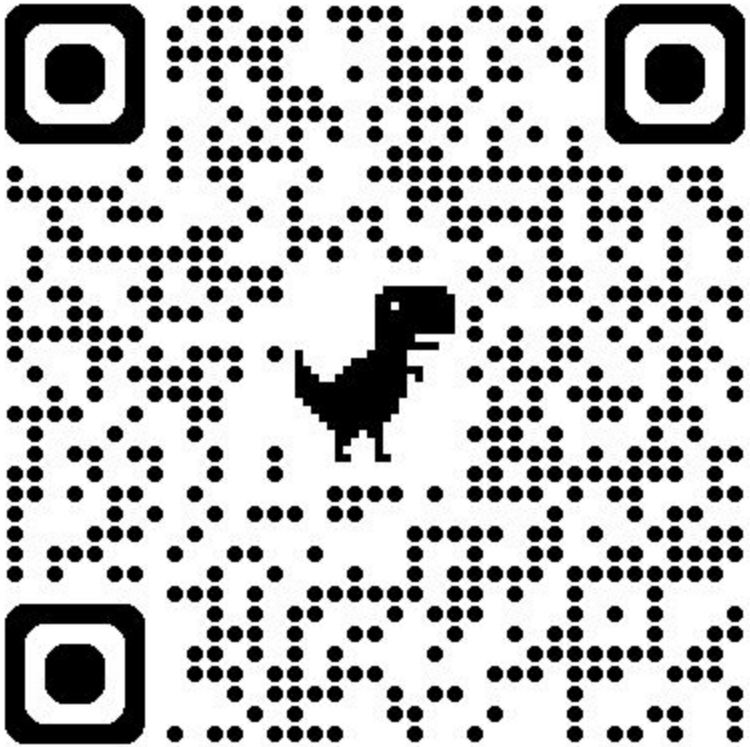
解答例と解説の表示

Enter your code here

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

Run Test Reset Show Answer

Test Result



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っ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

みなさんも問題作ってみませんか？