

# デザインプロジェクトII

## 「ボードゲームAIの開発」

---

担当 藤原

(小林)

# もくじ

---

内容

講義の進め方

成績評価方法

# 教員の紹介

藤原 洋志 (ふじわら ひろし)

## 教育分野

応用プログラミング言語 (3年生・前期)  
プログラミング言語論 (3年生・前期)

## 研究分野

アルゴリズム理論, 最適化理論  
スキーレンタル問題, ビンパッキング問題,  
円周上への質点配置問題, 木生成問題,  
周波数割当問題, 通貨交換問題, 関数最適化

先が分かれば...

スキー回数が...

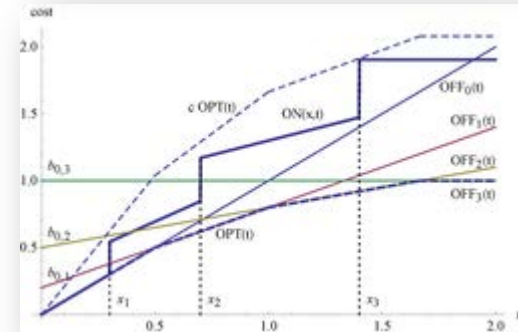
- 10回以上 → 買う
- 10回未満 → レンタル

先が分かなければ...

?



レンタル	買う
1万円	10万円



### ビンパッキング問題の例

ON (オンラインアルゴリズム):



OPT (最適なオフラインアルゴリズム):



# 内容

## 対戦用ボードゲームの人工知能を開発する



# コンピュータチェス

## ディープ・ブルーとガルリ・カスパロフ



# どうぶつしょうぎ

## どうぶつしょうぎの対戦AIを開発する



# どうぶつしょうぎ

## Amazonで991円



おもちゃ

Amazonポイント: 0  
マイストア キフト券 タイムセール Amazonで売る ヘルプ In English

prime day プライム特典利用で ポイント最大3,500円分

小林一樹さん  
アカウントサービス

今すぐ登録  
プライム

ほしい物  
リスト

カート

おもちゃ Amazonランキング 女の子 男の子 赤ちゃん・知育玩具 パズル ゲーム ブロック ホビー バーゲン

< 検索結果に戻る



画像をクリックして拡大イメージを表示

### どうぶつしょうぎ

幻冬舎エデュケーション  
★★★★☆ 225件のカスタマーレビュー

参考価格: ¥1,543  
価格: **¥ 991** 対象商品¥ 2,000以上の注文で通常配送無料 詳細  
OFF: ¥ 552 (36%)

**在庫あり。** 在庫状況について  
住所からお届け予定日を確認 既定の住所を使用 詳細

7/6 水曜日にお届けするには、今から**5 時間 15 分**以内に「お急ぎ便」または「当日お急ぎ便」を選択して注文を確定してください（有料オプション。Amazonプライム会員は無料）

この商品は、Amazon.co.jp が販売、発送します。ギフトラッピングを利用できます。

新品の出品: 32 ¥ 800より

- 対象年齢: 4歳から
- こどもやママも楽しめる! 将棋の入門版。

[もっと見る](#)

シェアする

☐ お急ぎ便無料でカートに入れる Amazonプライム無料体験について

数量: 1

カートに入れる

1-Click 注文を有効にしてください

ほしい物リストに追加する

#### こちらからもご購入いただけます

**¥ 1,291** [カートに入れる](#)

+ 関東への配送料無料  
発売元: クローズバイジャパン

**¥ 1,291** [カートに入れる](#)

+ 関東への配送料無料  
発売元: テラフォーマー

**¥ 941** [カートに入れる](#)

+ ¥ 350 関東への配送料  
発売元: Future Family

新品の出品: 32 ¥ 800より

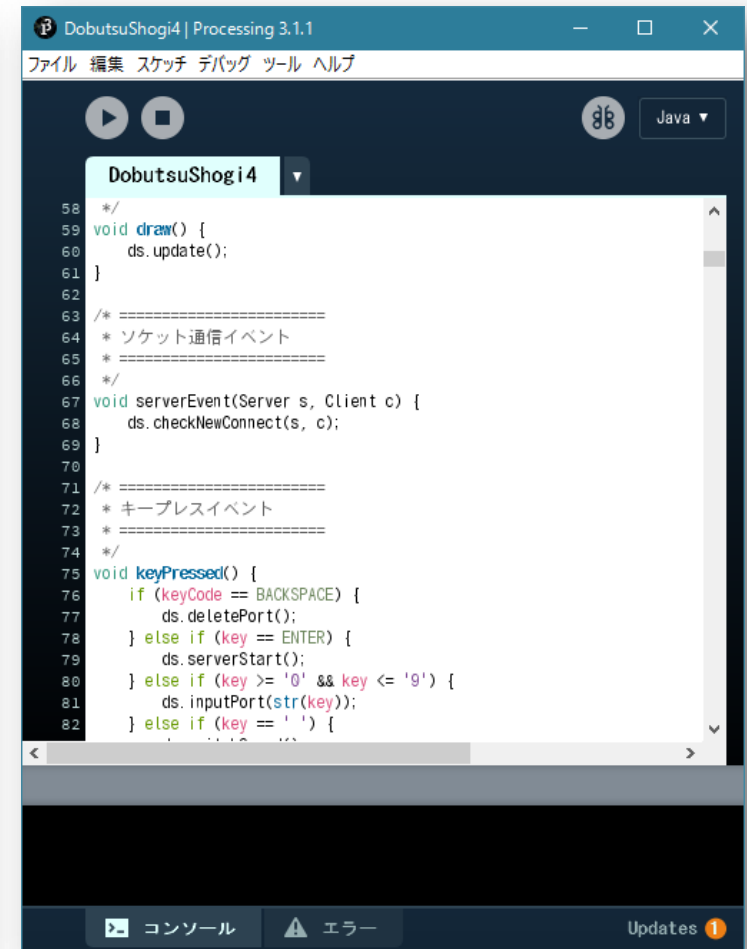
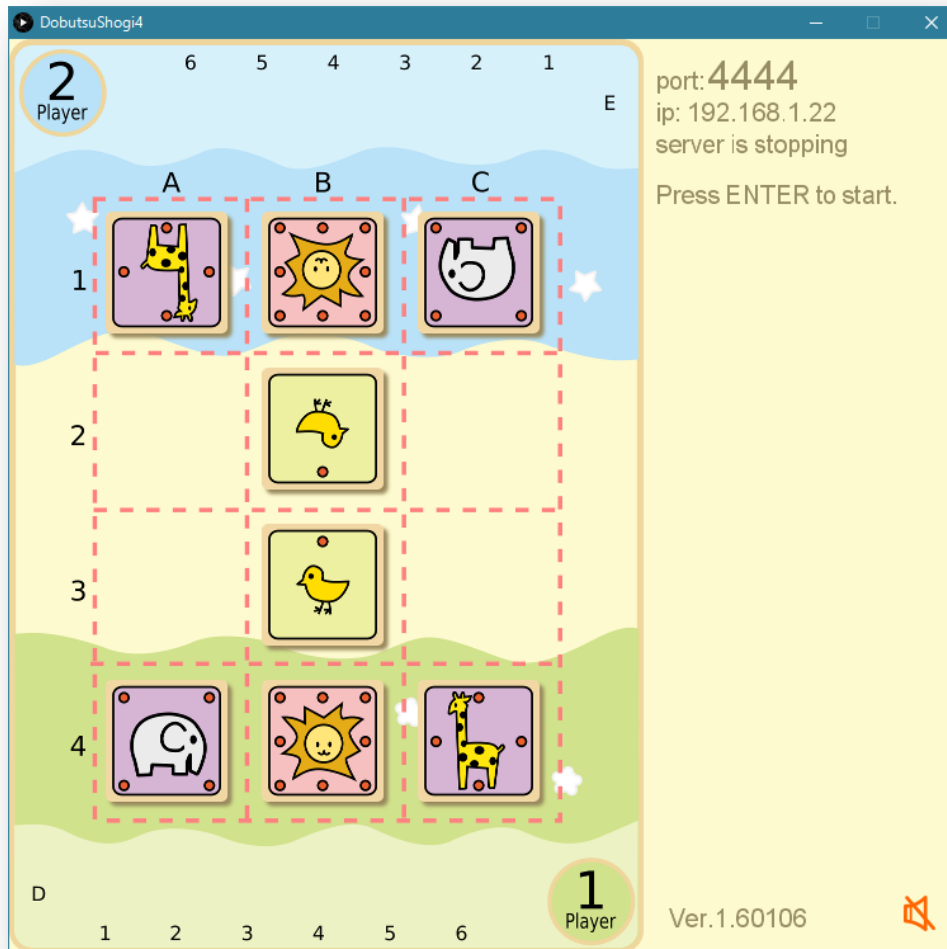
#### キャンペーンおよび追加情報

- Amazonランキング大賞 2016上半期: [おもちゃ総合ランキング](#) | [ホビー総合ランキング](#) | [総合ランキング](#)
- Amazon.co.jpが販売する一部の商品は、お一人様のご注文数量を限定させていただいております。限定数量を超えるご注文の際には、キャンセルさせていただく場合がございますので、あらかじめご了承ください。



# どうぶつしょうぎ対戦可視化サーバ

## どうぶつしょうぎの対戦AIを開発する





# 京都将棋

## 京都銀閣金鷄秘譜将棋

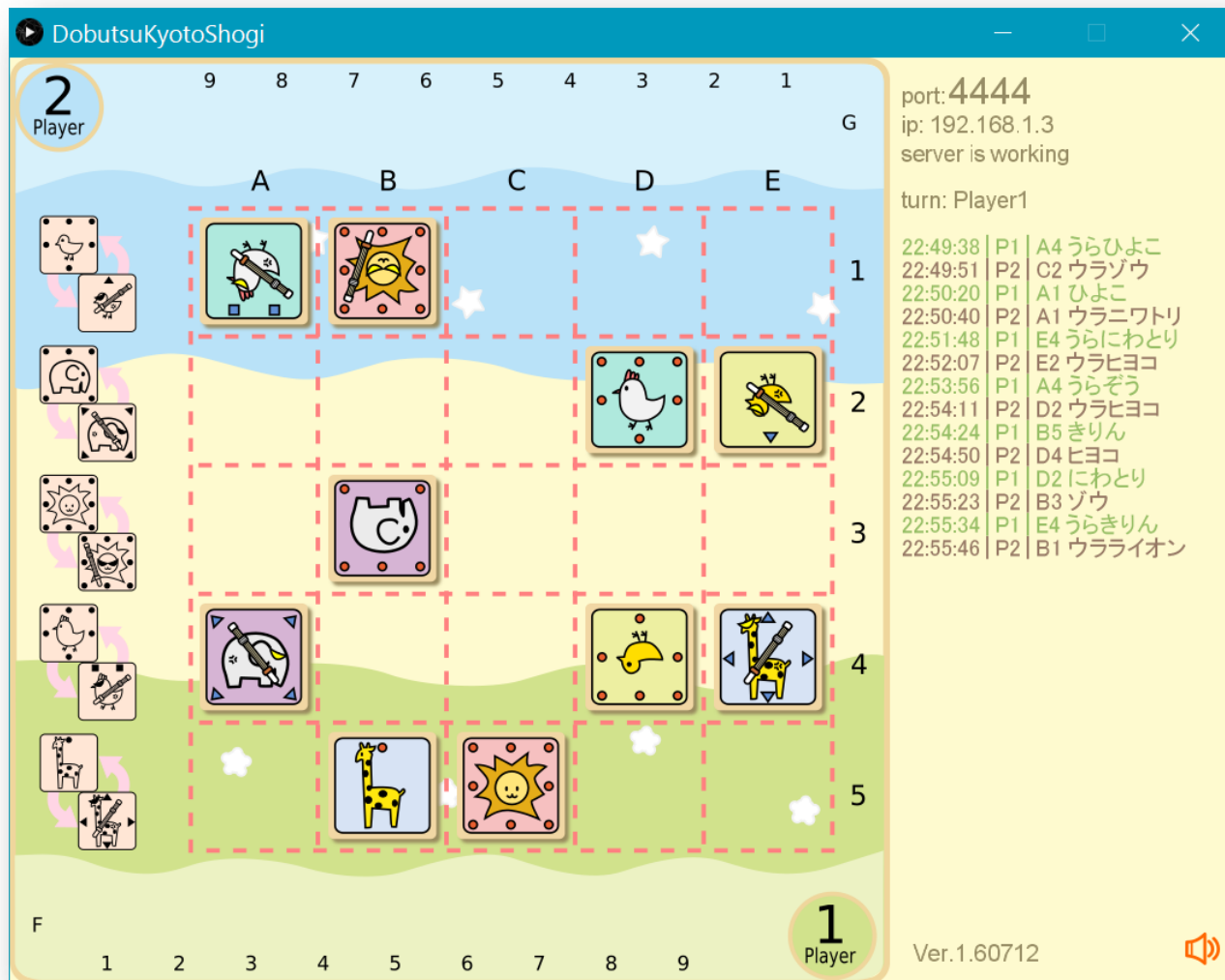
一手ごとにコマが裏返る5x5マスの将棋



玉 ⇄ 玉  
香 ⇄ と  
銀 ⇄ 角  
金 ⇄ 桂  
飛 ⇄ 歩

# どうぶつきょうとしょうぎサーバ

## 京都将棋の対戦AIを開発する



# どうぶつしょうぎAI

## 相手のAIに勝てる独自のAIを作るのが課題

```
tenum2.py (~/.Processing/DobutsuShogi2/Python) - GVIM
ファイル(F) 編集(E) ツール(T) シンタックス(S) パッケージ(P) ウインドウ(W) TeX-Suite TeX-Environments TeX-Elements TeX-Math ヘルプ(H)
📄 📁 📂 📅 📆 📇 📈 📉 📊 📋 📌 📍 📎 📏 📐 📑 📒 📓 📔 📕 📖 📗 📘 📙 📚 📛 📜 📝 📞 📟 📠 📡 📢 📣 📤 📥 📦 📧 📨 📩 📪 📫 📬 📭 📮 📯 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿 📠 📡 📢 📣 📤 📥 📦 📧 📨 📩 📪 📫 📬 📭 📮 📯 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿
/k/P/D/DobutsuShogi2.pde 3/k/P/D/P/tenum2.py

1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3
4 import socket
5 import sys
6 import re
7 import time
8 # import threading
9 import itertools
10 import copy
11
12 BUFSIZE = 1024
13
14 # 1手=1つのインスタンス
15 class Te:
16
17     def __str__(self):
18         return "mv " + self.src + " " + self.dst
19
20     def __init__(self, teStr):
21         teStr = teStr.strip()
22
23         if teStr.split(" ")[0] != "mv":
24             raise ValueError("not mv")
25
26         self.src = teStr.split(" ")[1]
27         self.dst = teStr.split(" ")[2]
28
29 # 1つの局面=1つのインスタンス
30 # インスタンスが常に保持している情報は、辞書 masuToKoma
31 # のみ
32 class Kyokumen:
33
34     komadaiAlpha = ["1": "D", "2": "E"]
35     flip = ["1": "2", "2": "1"]
36
37     # 駒の動き
38     # プレーヤー1を基準
39     kiki = [{"l": [(0, -1), (1, -1), (-1, -1), (1, 0),
40 (-1, 0),
41 (1, 1), (-1, 1), (0, 1)],
42 "e": [(1, -1), (-1, -1), (1, 1), (-1, 1)],
43 "g": [(0, -1), (1, 0), (-1, 0), (0, 1)],
44 "c": [(0, -1)],
45 "h": [(0, -1), (1, -1), (-1, -1), (1, 0), (-1, 0),
46 (1, 1), (-1, 1), (0, 1)],
47 "g": [(0, -1), (1, 0), (-1, 0), (0, 1)],
48 "c": [(0, -1)],
49 "h": [(0, -1), (1, -1), (-1, -1), (1, 0), (-1, 0),
50 (1, 1), (-1, 1), (0, 1)]}]
51
52 # 指定したマスにある駒の種類と現在の持ち主を返す
53 # 指定したマスに駒がなければ None
54 # マスの指定はXStrオブジェクトと整数を用いて行う
55
56 # 盤面簡易プリント用文字列
57 def debugStrings(self):
58     returnStr = "#" * 20 + "\n\n"
59
60     # 持ち駒を文字列化
61     returnStr = returnStr + "Player2: "
62
63     for n in range(1,6):
64         komaStr = self.getMochigoma("2", n)
65
66         if komaStr != None:
67             returnStr = returnStr + komaStr + " "
68
69     returnStr = returnStr + "\n\n"
70
71     # 盤面を文字列化
72     for y in range(1,7):
73         for x in range("ABC"):
74             komaStr = self.getKoma(x + y)
75
76             if komaStr == None:
77                 komaStr = "--"
78
79             returnStr = returnStr + komaStr
80
81     returnStr = returnStr + "\n"
82
83     # 持ち駒を文字列化
84     returnStr = returnStr + "Player1: "
85
86     for n in range(1,6):
87         komaStr = self.getMochigoma("1", n)
88
89         if komaStr != None:
90             returnStr = returnStr + komaStr + " "
91
92     returnStr = returnStr + "\n\n"
93
94     return returnStr
95
96 # 指定したマスにある駒の種類と現在の持ち主を返す
97 # 指定したマスに駒がなければ None
98 # マスの指定はXStrオブジェクトと整数を用いて行う
99
100 # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
101 # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
102 # playerStr は "1" あるいは "2"
103 # 返り値は例えば "e1" という文字列
104 def getMochigoma(self, playerStr, n):
105
106     return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
107
108 # 指定した手番の持ち駒台の空いているところに駒を置く
109 # koma は例えば "e1" という文字列
110 def putMochigoma(self, playerStr, koma):
111
112     # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
113     for n in range(1,7):
114         # 持ち駒台の n 番目に駒がなければそこに置く
115         if self.getMochigoma(playerStr, n) == None:
116             self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
117             break
118
119 # srcX, srcY のマス目の駒を動かす手を、
120 # プレーヤーがどちらかに関係なく生成するジェネレータ
121 # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
122 # 手とは、例えば ("B4", "C3") のようなタプル
123 def getTesFrom(self, src):
124     srcKoma = self.getKoma(src)
125
126     # プレーヤー2なら向き逆転
127     if srcKoma[1] == "1":
128         muki = 1
129     elif srcKoma[1] == "2":
130         muki = -1
131
132     # 駒の先行先について
133     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
134         deltaX = deltaX * muki
135         deltaY = deltaY * muki
136
137         # 先行が盤外の場合は飛ばす
138         if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
139             continue
140
141         # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
142         # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
143         # playerStr は "1" あるいは "2"
144         # 返り値は例えば "e1" という文字列
145         def getMochigoma(self, playerStr, n):
146
147             return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
148
149         # 指定した手番の持ち駒台の空いているところに駒を置く
150         # koma は例えば "e1" という文字列
151         def putMochigoma(self, playerStr, koma):
152
153             # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
154             for n in range(1,7):
155                 # 持ち駒台の n 番目に駒がなければそこに置く
156                 if self.getMochigoma(playerStr, n) == None:
157                     self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
158                     break
159
160         # srcX, srcY のマス目の駒を動かす手を、
161         # プレーヤーがどちらかに関係なく生成するジェネレータ
162         # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
163         # 手とは、例えば ("B4", "C3") のようなタプル
164         def getTesFrom(self, src):
165             srcKoma = self.getKoma(src)
166
167             # プレーヤー2なら向き逆転
168             if srcKoma[1] == "1":
169                 muki = 1
170             elif srcKoma[1] == "2":
171                 muki = -1
172
173             # 駒の先行先について
174             for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
175                 deltaX = deltaX * muki
176                 deltaY = deltaY * muki
177
178                 # 先行が盤外の場合は飛ばす
179                 if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
180                     continue
181
182                 # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
183                 # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
184                 # playerStr は "1" あるいは "2"
185                 # 返り値は例えば "e1" という文字列
186                 def getMochigoma(self, playerStr, n):
187
188                     return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
189
190                 # 指定した手番の持ち駒台の空いているところに駒を置く
191                 # koma は例えば "e1" という文字列
192                 def putMochigoma(self, playerStr, koma):
193
194                     # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
195                     for n in range(1,7):
196                         # 持ち駒台の n 番目に駒がなければそこに置く
197                         if self.getMochigoma(playerStr, n) == None:
198                             self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
199                             break
200
201                 # srcX, srcY のマス目の駒を動かす手を、
202                 # プレーヤーがどちらかに関係なく生成するジェネレータ
203                 # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
204                 # 手とは、例えば ("B4", "C3") のようなタプル
205                 def getTesFrom(self, src):
206                     srcKoma = self.getKoma(src)
207
208                     # プレーヤー2なら向き逆転
209                     if srcKoma[1] == "1":
210                         muki = 1
211                     elif srcKoma[1] == "2":
212                         muki = -1
213
214                     # 駒の先行先について
215                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
216                         deltaX = deltaX * muki
217                         deltaY = deltaY * muki
218
219                     # 先行が盤外の場合は飛ばす
220                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
221                         continue
222
223                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
224                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
225                     # playerStr は "1" あるいは "2"
226                     # 返り値は例えば "e1" という文字列
227                     def getMochigoma(self, playerStr, n):
228
229                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
230
231                     # 指定した手番の持ち駒台の空いているところに駒を置く
232                     # koma は例えば "e1" という文字列
233                     def putMochigoma(self, playerStr, koma):
234
235                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
236                         for n in range(1,7):
237                             # 持ち駒台の n 番目に駒がなければそこに置く
238                             if self.getMochigoma(playerStr, n) == None:
239                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
240                                 break
241
242                     # srcX, srcY のマス目の駒を動かす手を、
243                     # プレーヤーがどちらかに関係なく生成するジェネレータ
244                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
245                     # 手とは、例えば ("B4", "C3") のようなタプル
246                     def getTesFrom(self, src):
247                         srcKoma = self.getKoma(src)
248
249                     # プレーヤー2なら向き逆転
250                     if srcKoma[1] == "1":
251                         muki = 1
252                     elif srcKoma[1] == "2":
253                         muki = -1
254
255                     # 駒の先行先について
256                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
257                         deltaX = deltaX * muki
258                         deltaY = deltaY * muki
259
260                     # 先行が盤外の場合は飛ばす
261                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
262                         continue
263
264                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
265                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
266                     # playerStr は "1" あるいは "2"
267                     # 返り値は例えば "e1" という文字列
268                     def getMochigoma(self, playerStr, n):
269
270                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
271
272                     # 指定した手番の持ち駒台の空いているところに駒を置く
273                     # koma は例えば "e1" という文字列
274                     def putMochigoma(self, playerStr, koma):
275
276                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
277                         for n in range(1,7):
278                             # 持ち駒台の n 番目に駒がなければそこに置く
279                             if self.getMochigoma(playerStr, n) == None:
280                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
281                                 break
282
283                     # srcX, srcY のマス目の駒を動かす手を、
284                     # プレーヤーがどちらかに関係なく生成するジェネレータ
285                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
286                     # 手とは、例えば ("B4", "C3") のようなタプル
287                     def getTesFrom(self, src):
288                         srcKoma = self.getKoma(src)
289
290                     # プレーヤー2なら向き逆転
291                     if srcKoma[1] == "1":
292                         muki = 1
293                     elif srcKoma[1] == "2":
294                         muki = -1
295
296                     # 駒の先行先について
297                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
298                         deltaX = deltaX * muki
299                         deltaY = deltaY * muki
300
301                     # 先行が盤外の場合は飛ばす
302                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
303                         continue
304
305                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
306                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
307                     # playerStr は "1" あるいは "2"
308                     # 返り値は例えば "e1" という文字列
309                     def getMochigoma(self, playerStr, n):
310
311                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
312
313                     # 指定した手番の持ち駒台の空いているところに駒を置く
314                     # koma は例えば "e1" という文字列
315                     def putMochigoma(self, playerStr, koma):
316
317                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
318                         for n in range(1,7):
319                             # 持ち駒台の n 番目に駒がなければそこに置く
320                             if self.getMochigoma(playerStr, n) == None:
321                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
322                                 break
323
324                     # srcX, srcY のマス目の駒を動かす手を、
325                     # プレーヤーがどちらかに関係なく生成するジェネレータ
326                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
327                     # 手とは、例えば ("B4", "C3") のようなタプル
328                     def getTesFrom(self, src):
329                         srcKoma = self.getKoma(src)
330
331                     # プレーヤー2なら向き逆転
332                     if srcKoma[1] == "1":
333                         muki = 1
334                     elif srcKoma[1] == "2":
335                         muki = -1
336
337                     # 駒の先行先について
338                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
339                         deltaX = deltaX * muki
340                         deltaY = deltaY * muki
341
342                     # 先行が盤外の場合は飛ばす
343                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
344                         continue
345
346                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
347                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
348                     # playerStr は "1" あるいは "2"
349                     # 返り値は例えば "e1" という文字列
350                     def getMochigoma(self, playerStr, n):
351
352                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
353
354                     # 指定した手番の持ち駒台の空いているところに駒を置く
355                     # koma は例えば "e1" という文字列
356                     def putMochigoma(self, playerStr, koma):
357
358                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
359                         for n in range(1,7):
360                             # 持ち駒台の n 番目に駒がなければそこに置く
361                             if self.getMochigoma(playerStr, n) == None:
362                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
363                                 break
364
365                     # srcX, srcY のマス目の駒を動かす手を、
366                     # プレーヤーがどちらかに関係なく生成するジェネレータ
367                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
368                     # 手とは、例えば ("B4", "C3") のようなタプル
369                     def getTesFrom(self, src):
370                         srcKoma = self.getKoma(src)
371
372                     # プレーヤー2なら向き逆転
373                     if srcKoma[1] == "1":
374                         muki = 1
375                     elif srcKoma[1] == "2":
376                         muki = -1
377
378                     # 駒の先行先について
379                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
380                         deltaX = deltaX * muki
381                         deltaY = deltaY * muki
382
383                     # 先行が盤外の場合は飛ばす
384                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
385                         continue
386
387                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
388                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
389                     # playerStr は "1" あるいは "2"
390                     # 返り値は例えば "e1" という文字列
391                     def getMochigoma(self, playerStr, n):
392
393                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
394
395                     # 指定した手番の持ち駒台の空いているところに駒を置く
396                     # koma は例えば "e1" という文字列
397                     def putMochigoma(self, playerStr, koma):
398
399                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
400                         for n in range(1,7):
401                             # 持ち駒台の n 番目に駒がなければそこに置く
402                             if self.getMochigoma(playerStr, n) == None:
403                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
404                                 break
405
406                     # srcX, srcY のマス目の駒を動かす手を、
407                     # プレーヤーがどちらかに関係なく生成するジェネレータ
408                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
409                     # 手とは、例えば ("B4", "C3") のようなタプル
410                     def getTesFrom(self, src):
411                         srcKoma = self.getKoma(src)
412
413                     # プレーヤー2なら向き逆転
414                     if srcKoma[1] == "1":
415                         muki = 1
416                     elif srcKoma[1] == "2":
417                         muki = -1
418
419                     # 駒の先行先について
420                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
421                         deltaX = deltaX * muki
422                         deltaY = deltaY * muki
423
424                     # 先行が盤外の場合は飛ばす
425                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
426                         continue
427
428                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
429                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
430                     # playerStr は "1" あるいは "2"
431                     # 返り値は例えば "e1" という文字列
432                     def getMochigoma(self, playerStr, n):
433
434                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
435
436                     # 指定した手番の持ち駒台の空いているところに駒を置く
437                     # koma は例えば "e1" という文字列
438                     def putMochigoma(self, playerStr, koma):
439
440                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
441                         for n in range(1,7):
442                             # 持ち駒台の n 番目に駒がなければそこに置く
443                             if self.getMochigoma(playerStr, n) == None:
444                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
445                                 break
446
447                     # srcX, srcY のマス目の駒を動かす手を、
448                     # プレーヤーがどちらかに関係なく生成するジェネレータ
449                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
450                     # 手とは、例えば ("B4", "C3") のようなタプル
451                     def getTesFrom(self, src):
452                         srcKoma = self.getKoma(src)
453
454                     # プレーヤー2なら向き逆転
455                     if srcKoma[1] == "1":
456                         muki = 1
457                     elif srcKoma[1] == "2":
458                         muki = -1
459
460                     # 駒の先行先について
461                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
462                         deltaX = deltaX * muki
463                         deltaY = deltaY * muki
464
465                     # 先行が盤外の場合は飛ばす
466                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
467                         continue
468
469                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
470                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
471                     # playerStr は "1" あるいは "2"
472                     # 返り値は例えば "e1" という文字列
473                     def getMochigoma(self, playerStr, n):
474
475                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
476
477                     # 指定した手番の持ち駒台の空いているところに駒を置く
478                     # koma は例えば "e1" という文字列
479                     def putMochigoma(self, playerStr, koma):
480
481                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
482                         for n in range(1,7):
483                             # 持ち駒台の n 番目に駒がなければそこに置く
484                             if self.getMochigoma(playerStr, n) == None:
485                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
486                                 break
487
488                     # srcX, srcY のマス目の駒を動かす手を、
489                     # プレーヤーがどちらかに関係なく生成するジェネレータ
490                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
491                     # 手とは、例えば ("B4", "C3") のようなタプル
492                     def getTesFrom(self, src):
493                         srcKoma = self.getKoma(src)
494
495                     # プレーヤー2なら向き逆転
496                     if srcKoma[1] == "1":
497                         muki = 1
498                     elif srcKoma[1] == "2":
499                         muki = -1
500
501                     # 駒の先行先について
502                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
503                         deltaX = deltaX * muki
504                         deltaY = deltaY * muki
505
506                     # 先行が盤外の場合は飛ばす
507                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
508                         continue
509
510                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
511                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
512                     # playerStr は "1" あるいは "2"
513                     # 返り値は例えば "e1" という文字列
514                     def getMochigoma(self, playerStr, n):
515
516                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
517
518                     # 指定した手番の持ち駒台の空いているところに駒を置く
519                     # koma は例えば "e1" という文字列
520                     def putMochigoma(self, playerStr, koma):
521
522                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
523                         for n in range(1,7):
524                             # 持ち駒台の n 番目に駒がなければそこに置く
525                             if self.getMochigoma(playerStr, n) == None:
526                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
527                                 break
528
529                     # srcX, srcY のマス目の駒を動かす手を、
530                     # プレーヤーがどちらかに関係なく生成するジェネレータ
531                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
532                     # 手とは、例えば ("B4", "C3") のようなタプル
533                     def getTesFrom(self, src):
534                         srcKoma = self.getKoma(src)
535
536                     # プレーヤー2なら向き逆転
537                     if srcKoma[1] == "1":
538                         muki = 1
539                     elif srcKoma[1] == "2":
540                         muki = -1
541
542                     # 駒の先行先について
543                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
544                         deltaX = deltaX * muki
545                         deltaY = deltaY * muki
546
547                     # 先行が盤外の場合は飛ばす
548                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
549                         continue
550
551                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
552                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
553                     # playerStr は "1" あるいは "2"
554                     # 返り値は例えば "e1" という文字列
555                     def getMochigoma(self, playerStr, n):
556
557                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
558
559                     # 指定した手番の持ち駒台の空いているところに駒を置く
560                     # koma は例えば "e1" という文字列
561                     def putMochigoma(self, playerStr, koma):
562
563                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
564                         for n in range(1,7):
565                             # 持ち駒台の n 番目に駒がなければそこに置く
566                             if self.getMochigoma(playerStr, n) == None:
567                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
568                                 break
569
570                     # srcX, srcY のマス目の駒を動かす手を、
571                     # プレーヤーがどちらかに関係なく生成するジェネレータ
572                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
573                     # 手とは、例えば ("B4", "C3") のようなタプル
574                     def getTesFrom(self, src):
575                         srcKoma = self.getKoma(src)
576
577                     # プレーヤー2なら向き逆転
578                     if srcKoma[1] == "1":
579                         muki = 1
580                     elif srcKoma[1] == "2":
581                         muki = -1
582
583                     # 駒の先行先について
584                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
585                         deltaX = deltaX * muki
586                         deltaY = deltaY * muki
587
588                     # 先行が盤外の場合は飛ばす
589                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
590                         continue
591
592                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
593                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
594                     # playerStr は "1" あるいは "2"
595                     # 返り値は例えば "e1" という文字列
596                     def getMochigoma(self, playerStr, n):
597
598                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
599
600                     # 指定した手番の持ち駒台の空いているところに駒を置く
601                     # koma は例えば "e1" という文字列
602                     def putMochigoma(self, playerStr, koma):
603
604                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
605                         for n in range(1,7):
606                             # 持ち駒台の n 番目に駒がなければそこに置く
607                             if self.getMochigoma(playerStr, n) == None:
608                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
609                                 break
610
611                     # srcX, srcY のマス目の駒を動かす手を、
612                     # プレーヤーがどちらかに関係なく生成するジェネレータ
613                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
614                     # 手とは、例えば ("B4", "C3") のようなタプル
615                     def getTesFrom(self, src):
616                         srcKoma = self.getKoma(src)
617
618                     # プレーヤー2なら向き逆転
619                     if srcKoma[1] == "1":
620                         muki = 1
621                     elif srcKoma[1] == "2":
622                         muki = -1
623
624                     # 駒の先行先について
625                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
626                         deltaX = deltaX * muki
627                         deltaY = deltaY * muki
628
629                     # 先行が盤外の場合は飛ばす
630                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
631                         continue
632
633                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
634                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
635                     # playerStr は "1" あるいは "2"
636                     # 返り値は例えば "e1" という文字列
637                     def getMochigoma(self, playerStr, n):
638
639                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
640
641                     # 指定した手番の持ち駒台の空いているところに駒を置く
642                     # koma は例えば "e1" という文字列
643                     def putMochigoma(self, playerStr, koma):
644
645                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
646                         for n in range(1,7):
647                             # 持ち駒台の n 番目に駒がなければそこに置く
648                             if self.getMochigoma(playerStr, n) == None:
649                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
650                                 break
651
652                     # srcX, srcY のマス目の駒を動かす手を、
653                     # プレーヤーがどちらかに関係なく生成するジェネレータ
654                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
655                     # 手とは、例えば ("B4", "C3") のようなタプル
656                     def getTesFrom(self, src):
657                         srcKoma = self.getKoma(src)
658
659                     # プレーヤー2なら向き逆転
660                     if srcKoma[1] == "1":
661                         muki = 1
662                     elif srcKoma[1] == "2":
663                         muki = -1
664
665                     # 駒の先行先について
666                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
667                         deltaX = deltaX * muki
668                         deltaY = deltaY * muki
669
670                     # 先行が盤外の場合は飛ばす
671                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
672                         continue
673
674                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
675                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
676                     # playerStr は "1" あるいは "2"
677                     # 返り値は例えば "e1" という文字列
678                     def getMochigoma(self, playerStr, n):
679
680                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
681
682                     # 指定した手番の持ち駒台の空いているところに駒を置く
683                     # koma は例えば "e1" という文字列
684                     def putMochigoma(self, playerStr, koma):
685
686                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
687                         for n in range(1,7):
688                             # 持ち駒台の n 番目に駒がなければそこに置く
689                             if self.getMochigoma(playerStr, n) == None:
690                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
691                                 break
692
693                     # srcX, srcY のマス目の駒を動かす手を、
694                     # プレーヤーがどちらかに関係なく生成するジェネレータ
695                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
696                     # 手とは、例えば ("B4", "C3") のようなタプル
697                     def getTesFrom(self, src):
698                         srcKoma = self.getKoma(src)
699
700                     # プレーヤー2なら向き逆転
701                     if srcKoma[1] == "1":
702                         muki = 1
703                     elif srcKoma[1] == "2":
704                         muki = -1
705
706                     # 駒の先行先について
707                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
708                         deltaX = deltaX * muki
709                         deltaY = deltaY * muki
710
711                     # 先行が盤外の場合は飛ばす
712                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
713                         continue
714
715                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
716                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
717                     # playerStr は "1" あるいは "2"
718                     # 返り値は例えば "e1" という文字列
719                     def getMochigoma(self, playerStr, n):
720
721                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
722
723                     # 指定した手番の持ち駒台の空いているところに駒を置く
724                     # koma は例えば "e1" という文字列
725                     def putMochigoma(self, playerStr, koma):
726
727                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
728                         for n in range(1,7):
729                             # 持ち駒台の n 番目に駒がなければそこに置く
730                             if self.getMochigoma(playerStr, n) == None:
731                                 self.masuToKoma[Kyokumen.komadaiAlpha[playerStr] + str(n)] = koma
732                                 break
733
734                     # srcX, srcY のマス目の駒を動かす手を、
735                     # プレーヤーがどちらかに関係なく生成するジェネレータ
736                     # srcX, srcY に駒がなければエラー(呼び出し側でチェックすべし)
737                     # 手とは、例えば ("B4", "C3") のようなタプル
738                     def getTesFrom(self, src):
739                         srcKoma = self.getKoma(src)
740
741                     # プレーヤー2なら向き逆転
742                     if srcKoma[1] == "1":
743                         muki = 1
744                     elif srcKoma[1] == "2":
745                         muki = -1
746
747                     # 駒の先行先について
748                     for (deltaX, deltaY) in Kyokumen.kiki[srcKoma[0]]:
749                         deltaX = deltaX * muki
750                         deltaY = deltaY * muki
751
752                     # 先行が盤外の場合は飛ばす
753                     if src[0] == "A" and deltaX < 0 or src[0] == "C" and deltaX > 0:
754                         continue
755
756                     # 指定した手番の持ち駒台の n 番目(1<=n<=6)にある駒の種類と現在の持ち主を返す
757                     # 持ち駒台の n 番目(1<=n<=6)に駒がなければ None
758                     # playerStr は "1" あるいは "2"
759                     # 返り値は例えば "e1" という文字列
760                     def getMochigoma(self, playerStr, n):
761
762                         return self.masuToKoma.get(Kyokumen.komadaiAlpha[playerStr] + str(n))
763
764                     # 指定した手番の持ち駒台の空いているところに駒を置く
765                     # koma は例えば "e1" という文字列
766                     def putMochigoma(self, playerStr, koma):
767
768                         # 持ち駒台のうち、最初に空いているところ(つまり末尾)に置く
769                         for n in range(1,7):
770                             # 持ち駒台の n 番目に駒がなければそこに置く
771                             if self.getMochigoma(playerStr, n) == None:
772                                 self.mas
```

# 講義の進め方

ひとりずつ**個人ごと**にプログラムを開発する

他の受講生のプログラムと対戦する

7回の講義中、**対戦会を3回**を開催(第3回講義、第5回講義、第7回講義)  
大会の運営は受講生で協力して運営

対戦とコーディングを繰り返す

対戦結果をもとにプログラムを改良

基本的には**自分で調査**してコーディング

適宜講義時間中に**ヒント**を提供

共通事項に関して**積極的に情報交換**

# 成績評価方法

## 成績

対戦成績（勝率）とプログラム内容をもとに評価

※探索アルゴリズムが実装されていること

対戦プログラムの発表

授業時間中に作成したプログラムを発表

テスト | ありません

レポート | あるかも