卒業研究報告書

題目

# 人狼ゲームのコンピュータ AI の作成

指導教員

石水 隆　講師

報告者

17–1–037–0142

藤田 尚也

近畿大学理工学部情報学科

令和 3 年 2 月 1 日提出

# 概要

ゲーム AI は様々なものが開発されており，完全情報ゲームでは人間が勝つことが困難な AI が開発されている．そして不完全情報ゲームにおいても強い AI が登場し始めている．しかしコミュニケーションが必要なゲームは，AI にとって苦手な分野である．そこで不完全情報コミュニケーションゲームの中でも人気が高い人狼ゲームの AI の開発を目指す．

人狼ゲームは 3 人以上が集まり，人狼側と市民側に分かれ，話し合いによって勝利を目指すゲームである．本研究では人狼ゲームを簡略化したワンナイト人狼のルールを使用する．3 人プレイを想定し，AI を開発する．開発には Python 言語を用いる．ワンナイト人狼は夜のアクションのターンと昼の話し合いの時間の 1 回ずつで勝敗が決まる．人数 +2 の役職の中から割り振られ，夜のアクションを行う．昼の時間で話し合い，それを踏まえ投票を行い，人狼が処刑されれば市民側の勝利，人狼が処刑されなければ人狼側の勝利である．役職は人狼 2，村人 1，占い師 1，怪盗 1 の中からランダムに選ぶ．話し合いには選択肢を用い，また発言回数を 3 回に制限する．選択肢の内容は主に役職の公表，疑い，発言しないなどである．選択肢は役職によって確率を変えて選び，2 回目以降は他のプレイヤーの発言によって確率を変更する．投票の際は他のプレイヤーの発言の疑わしさを数値化し，その合計が最も高いプレイヤーに投票する．それに加え，役職の能力によって，投票先を決める．作成するプログラムはプレイ人数を 0 人から 3 人の中から選べるよう設計する．3 人であれば通常のワンナイト人狼，0 人であればコンピュータ同士の対戦ができる．

これらの手順で AI の開発に取り組み，人狼ゲームの AI を作成した．実際にプレイした結果，話し合いが選択肢による会話であっても，役職の能力が勝敗に大きく関わっていることが分かった．

# 目次

# 1 序論

## 1.1 本研究の背景

昨今の様々なゲーム AI が開発されており，とりわけオセロ，将棋，チェス，囲碁等の完全情報ゲームでは人間が勝つことが困難なゲーム AI が開発されている．

また，麻雀やポーカー等の不完全情報ゲームは，AI の作成が難しく，発展途上であるが，これらも強い AI が登場している [1][2].

不完全情報ゲームに加え，プレイヤー間の交渉が必要なゲームも AI の苦手な分野である．そこで，本研究では，不完全情報ゲームの一つであり，交渉が重要な要素となる人狼ゲームの AI の開発を目指す．

## 1.2 人狼ゲームとは

人狼ゲームとは 3 人以上のプレイヤーが集まり，話し合いを通して自身が所属する陣営の勝利を目指す不完全情報ゲームである．

話し合いの際には，各プレイヤーに割り当てられた役職に関しての情報を共有する，偽りの役職になりすますなどの戦略を用いることで勝利に近づくことができる．

## 1.3 人狼ゲームに関する既知の結果

2015 年，人狼知能プロジェクトが発足した [3]．人狼知能プロジェクトは，「人間と自然なコミュニケーションを取りながら人狼をプレイできるエージェントの構築」を目指しており，人狼知能プラットフォームなどを開発中である．また，人狼知能プロジェクトでは人狼知能国際大会を開催している．

西崎らは，ワンナイト人狼における投票行動の分析を行っている [6][7]．西崎らの結果から，ワンナイト人狼では占い師の能力の結果や，議論の内容，話し始める順序が勝敗に大きく影響していることが示される．

## 1.4 本研究の目的

本研究の目的は，実際に人とコンピュータがプレイできる人狼ゲームのプレイヤー AI を作成することである．不完全情報コミュニケーションゲームの AI を作成することで，ゲームの，また AI の幅が広がると考える．

## 1.5 本報告書の構成

本報告書は，人狼ゲームについて，人狼ゲームのコンピュータ AI，結果，考察，結論・今後の課題の順に述べる．

# 2　人狼ゲームについて

## 2.1　人狼ゲーム

人狼ゲーム [4][5] は，3 人以上のプレイヤーが人狼側，市民側に分かれて行うゲームである．役職を人数分用意し，役職の種類はゲーム開始前に決めることができ，決めた役職はゲーム開始時にランダムに割り当てられる．人数が増えるほど人狼の数を増やす必要がある．この役職の種類や数によってゲームの戦略が大きく変わる．

ゲームは昼のターン，夜のターンを交互に繰り返し，昼のターンは話し合いを行って処刑するプレイヤーを多数決の投票で決定する．2 人以上選ばれた場合はそれらのプレイヤーのみで決戦投票を行い，1 人を処刑する．夜のターンは人狼が誰か 1 人を襲撃し，役職のあるプレイヤーは，その役職の能力を実行する．これを繰り返し，人狼が全滅するか，人狼の人数が，市民側の人数以上になるまでゲームは続く．

話し合いは主に役職に関する会話で行われる．市民側は勝つために，能力を持ったプレイヤーがその結果を共有し，人狼を見つける手がかりとする．その際能力により何が分かったのか，後述する占い師ならば，なぜそのプレイヤーを占ったのか，その理由を説明するなど，しっかり話す事で信用を得る．人狼は基本的に投票でなければ処刑されないため，いかに投票を避けられるかが肝心である．そのため重要な役職を騙るか，他のプレイヤーの挙動がおかしいなどと指摘し，他のプレイヤーに注目を集めさせる．これらの話し合いの仕方は一例だが，プレイヤーによりその戦略は多岐に渡る．

## 2.2　人狼ゲームの役職

人狼ゲームには様々な役職が存在する．例をあげると，他のプレイヤーと役職を交換しその能力を使用することができる怪盗，処刑されたプレイヤーが人間かどうかを知ることができる霊媒師，夜のターンに人狼の襲撃から守ることができる狩人，人狼チームに所属するが能力がなく占い師に占われても人間とでる狂人などの役職がよく用いられる．

どのようなルールでも用いられることが多い役職は人狼，村人，占い師である．人狼は他の人狼プレイヤーを知ることができ，夜に襲撃を行ってプレイヤーを 1 人減らすことができる．この役職が無ければ人狼ゲームは成り立たない．村人は何も能力を持たない市民側の役職である．能力を持たないため，ゲームを動かすことが難しいが，この役職のプレイヤーがどのプレイヤーが人狼かを見極めることによって，ゲームの勝敗に大きく関わる．占い師は夜のターンに他のプレイヤーが人狼側か市民側かを知ることができる市民側の役職である．この役職はどのプレイヤーが人狼か判断する手がかりになるため，非常に重要な役職である．

## 2.3　ワンナイト人狼

人狼ゲームを 1 夜で決着がつくようにした人狼ゲームとしてワンナイト人狼がある．ワンナイト人狼とは，夜のターンと昼のターンそれぞれ 1 回のみで決着がつくルールであり，夜のターンから開始される．昼のターンに話し合い，その後の投票で人狼が処刑されれば市民側の勝利，市民側が処刑されれば人狼側の勝利である．投票で同票プレイヤーがいた場合は再投票がなく，同票のプレイヤー全員が処刑される．この場合も人狼が処刑されていれば市民側の勝利である．

役職は必ずプレイヤー人数 +2 の数の役職を用意する．例えばプレイヤーが 4 人なら人狼 2，村人 3，占い

師 1 のように役職を合計 6 つ用意し，その中からランダムに 4 つ，各プレイヤーに 1 つずつ役職が割り当てられる．選ばれなかった役職 2 つは伏せられる．選ばれなかった役職があることで，プレイヤーの中に占い師がいないことがあるなど，どの役職が存在するか分からない．

役職が 2 つ余るので，プレイヤーの中に人狼がいない場合もある．その場合は投票の際，全員が 1 票ずつ投票されている状態である，「平和村」にすることでプレイヤー全員の勝利となるが，誰か 1 人でも処刑されると，プレイヤー全員の負けとなる．また平和村となった際，プレイヤーに人狼がいれば人狼側の勝利である．

役職の能力は通常の人狼ゲームと異なる点がある．人狼は，1 夜のみのため夜の襲撃は行えないが仲間を確認することができる．占い師は選ばれなかった役職 2 つか，他のプレイヤー 1 人か，のどちらかを知ることができる．怪盗は役職の交換は行えるが，交換した役職の能力は使用できない．

# 3　人狼ゲームのコンピュータ AI

## 3.1　人狼ゲームに AI を用いる問題点

人狼ゲームはプレイヤー同士の話し合いによって勝敗が大きく変わるゲームである．そのため，プレイヤーは様々な嘘をつき、またプレイヤー自身の発言を信用させるなどの戦略を用いる必要がある．

しかし，AI に嘘を理解させることが難しく，また周囲の発言に合わせて嘘をつくことも難しい．

そこで人狼ゲームを簡略化したワンナイト人狼を使用し，昼のターンの話し合いでは発言回数と発言内容の制限を行う．

本研究では 3 人でのゲームを想定し，人狼 2，村人 1，占い師 1，怪盗 1 の 4 種類，5 つの役職を使用する．人狼側は人狼のみで，その他の役職は市民側である．表 1 に各役職の能力をまとめる．

表 2 はこのゲームでの選ばれる役職全ての組み合わせと，その組み合わせになる確率である．

表 1　役職の能力

| 役職 | 能力 |
|---|---|
| 人狼 | 人狼の仲間がいるかどうか確認できる<br>仲間がいればそのプレイヤーを知ることができる |
| 村人 | 何も能力を持たない |
| 占い師 | 他のプレイヤー 1 人が人狼か人間か確認できる<br>もしくは選ばれていない 2 枚の役職を知ることができる |
| 怪盗 | 他のプレイヤーと役職を交換できる<br>交換した役職の能力は使用できない<br>交換されたプレイヤーは交換されていることを認知しない |

## 3.2　話し合いの時間

話し合いには選択肢を用い，発言や役職発表する回数を 3 回に制限する．選択肢は 21 個用意し，発言はプレイヤー 3 人が同時に行う．表 3 に発言の選択肢を記す．選択肢には番号が振られ，表中の X と Y は他のプレイヤーの番号を指す．

これらの選択肢は大きく分けると自身の役職を公表，他のプレイヤーを疑う，平和村にする，発言しないの

表 2　役職の組み合わせ

| | プレイヤー 3 人の役職 | | | 残った役職 | | 確率 |
|---|---|---|---|---|---|---|
| a | 人狼 | 村人 | 占い師 | 人狼 | 怪盗 | 20% |
| b | 人狼 | 村人 | 怪盗 | 人狼 | 占い師 | 20% |
| c | 人狼 | 占い師 | 怪盗 | 人狼 | 村人 | 20% |
| d | 人狼 | 人狼 | 占い師 | 村人 | 怪盗 | 10% |
| e | 人狼 | 人狼 | 村人 | 占い師 | 怪盗 | 10% |
| f | 人狼 | 人狼 | 怪盗 | 村人 | 占い師 | 10% |
| g | 村人 | 占い師 | 怪盗 | 人狼 | 人狼 | 10% |

4 通りである.

選択肢の選び方は各役職ごとに確率によって選び，各役職の戦略によって変わる．以下の節では各役職の戦略を述べる [6][7].

表 3　話し合いの選択肢

| | 選択肢 |
|---|---|
| 1 | 私は村人です。 |
| 2 | 私は人狼です。 |
| 3 | 私は占い師です。他の 2 枚は人狼と人狼でした。 |
| 4 | 私は占い師です。他の 2 枚は人狼と怪盗でした。 |
| 5 | 私は占い師です。他の 2 枚は人狼と村人でした。 |
| 6 | 私は占い師です。他の 2 枚は怪盗と村人でした。 |
| 7 | 私は占い師です。PlayerX は人狼でした。 |
| 8 | 私は占い師です。PlayerX は人間でした。 |
| 9 | 私は占い師です。PlayerY は人狼でした。 |
| 10 | 私は占い師です。PlayerY は人間でした。 |
| 11 | 私は怪盗です。PlayerX と交換して村人でした。 |
| 12 | 私は怪盗です。PlayerY と交換して村人でした。 |
| 13 | 私は怪盗です。PlayerX と交換して占い師でした。 |
| 14 | 私は怪盗です。PlayerY と交換して占い師でした。 |
| 15 | 私は怪盗です。交換しませんでした。 |
| 16 | 私は PlayerX を信じます。 |
| 17 | 私は PlayerY を信じます。 |
| 18 | 私は PlayerX が怪しいです。 |
| 19 | 私は PlayerY が怪しいです。 |
| 20 | 平和村にしましょう。 |
| 21 | 発言しない |

### 3.2.1 村人の戦略

村人は能力がなく何も情報がないので，疑いを招かないよう余計な発言は避ける．

表 14 は最初の発言の選択肢を選ぶ確率である．表記していない選択肢の番号は選ばれる確率が 0% である．村人だと発言するか，発言しないの 2 択である．早めに村人と名乗り出ることで余計な疑いをかけられないようにする．発言しないは人狼がおかしな発言をしないかの様子見である．

2 回目の発言は，1 回目に発言していなかった場合，村人と発言する確率を 90% に上げ，発言しない確率を 10% に下げる．発言していた場合，他のプレイヤーに村人がいる，占い師が他の 2 枚に村人がいると言っている，同じ役職を 2 人が名乗り出ているなどの矛盾を選択肢 18，19 で指摘する．

3 回目の発言は，一度も発言していなかった場合，必ず村人と名乗り出るようにする．発言していた場合は 2 回目と同様に矛盾を指摘する．

表 4　村人が選ぶ選択肢の確率

| 選択肢 | 選ぶ確率 |
|---|---|
| 1 | 80% |
| 21 | 20% |

### 3.2.2 人狼の戦略

人狼は仲間がいない場合，村人だと嘘をつき他のプレイヤーに判断を委ねる，怪盗や占い師と偽り信頼を得るという 2 通りの戦略を用いる．仲間がいる場合は大胆な発言を行い，プレイヤーの中に怪盗がいないかを探る．

表 15 は人狼が最初の発言で選ぶ選択肢の確率である．仲間がいる場合，様々な選択の可能性を作ることで，仲間ではないプレイヤーが怪盗かどうかを探る．そのプレイヤーが占い師か怪盗と名乗り出ているなら怪盗である可能性がある．

2 回目の発言は，1 回目に発言をしていなかった場合，1 回目の確率の発言しない確率を 45% に上げ，他の確率を少し下げる．発言していた場合は発言せず様子をみる．3 回目の発言は一度も発言していない場合，2 回目の確率の発言しない確率を 25% にし，他の確率を少し上げる．一度でも発言していた場合は発言せず様子を見る．

仲間がいない場合，自身の役職が村人であると発言する確率を一番高くしている．これは占い師が，他の 2 枚を占っていたとき，表 2 の a の組み合わせの場合，占い師に本当の村人がどちらかを迷わせることができる．また c の組み合わせの場合，村人はいないが怪盗が人狼と交換していた場合，疑いを怪盗に向けることができる．2 種類の組み合わせで有効な発言なため，一番無難な発言である．しかし b の組み合わせでは村人と怪盗どちらにも疑われるため危険な発言である．

選択肢 3 の発言は表 2 の b の組み合わせで，怪盗が村人と交換している場合，平和村だと思わせることができるが，確率は低く，他の組み合わせの場合は疑われるため危険である．選択肢 4 の発言は a と b の組み合わせの場合，村人を信頼させられる．選択肢 8，10 は他のプレイヤーが人間であることは確実なため矛盾は起きず，村人を迷わせることができるが，占い師に疑われてしまうため確率は低くしている．選択肢 13，14 は実際に占い師を指していれば絶大な信頼を得られるが，違っていると疑われてしまうため，賭けのような発言で

ある．

発言しない選択肢は，目立たないようにするために，また怪盗が人狼と交換していた場合，その怪盗を疑わせるための様子見である．

2 回目以降の発言は，村人だと発言していた場合は村人，占い師だと発言していた場合は占い師と同じ戦略を用いる．怪盗だと発言していた場合，発言しないか，怪盗だと発言したプレイヤーがいた場合はそのプレイヤーに疑いが向くようにする．発言していなかった場合，1 回目の確率の村人だと発言する確率をさらに上げる．

表 5　人狼が選ぶ選択肢の確率

| 選択肢 | 仲間がいない | 仲間がいる |
|---|---|---|
| 1 | 48% | 23% |
| 2 | 0% | 5% |
| 3 | 2% | 7% |
| 4 | 6% | 7% |
| 5 | 3% | 7% |
| 6 | 0% | 7% |
| 7 | 0% | 7% |
| 8 | 2% | 7% |
| 9 | 0% | 7% |
| 10 | 2% | 7% |
| 11 | 0% | 4% |
| 12 | 0% | 4% |
| 13 | 6% | 4% |
| 14 | 6% | 4% |
| 21 | 25% | 0% |

#### 3.2.3　占い師の戦略

占い師は能力によって情報を持っているため，その情報を発言し，周りの信用を得るよう行動する．

占い師が占う対象は，選ばれていない役職を占う確率が 90%，他のプレイヤーを占う確率がそれぞれ 5% ずつである．選ばれていない役職を占う確率が高い理由は，役職を 2 つ知ることができ，情報量が多いからである．そのため占い結果によって戦略が決定しやすい．

プレイヤーを占った場合，人間だと分かったとしても，怪盗が人狼と交換している可能性があり，また占っていないプレイヤーの役職が分からないので人狼がいるかどうかも分からない．人狼だと分かった場合も同じく怪盗により交換されている可能性があるので，確実な投票ができない．

1 回目の発言は占い結果によって変わる．他の 2 枚がともに人狼の場合，選択肢 3 を 100% で選択し，2 回目以降は選択肢 20 を 100% で選択する．他の 2 枚が人狼，怪盗の場合，プレイヤーの中に怪盗がいないため，人狼を見つけるために選択肢 4 を 100% 選択する．2 回目以降は村人の戦略と同じように他プレイヤーの矛盾を指摘する．他の 2 枚が人狼，村人の場合，プレイヤーの中に怪盗がいるため，選択肢 5 を選択する確率を

90% にし，様子を見るために選択肢 21 を選択する確率を 10% にする．2 回目以降は 1 回目に発言していなければ，選択肢 5 を選択する確率を 100% にし，発言していれば他プレイヤーの矛盾を指摘する．他の 2 枚が村人，怪盗の場合，他のプレイヤー 2 人が人狼であるため，怪盗であると思わせる必要がある．そのため選択肢 21 を選ぶ確率を 90% にし，様子をみる確率を高めにする．そして選択肢 2 を選択する確率を 10% にし，人狼だと名乗り出ることで疑わせる．1 回目の発言から怪しい動きをすると，不自然なので確率は低くしている．2 回目は選択肢 7,9,13,14,15 を選択する確率をそれぞれ 20% にし，怪盗であると思わせる．

プレイヤーを占った場合，その結果が人狼ならば選択肢 7 もしくは 9 を 100% 選択する．人間であれば選択肢 8 もしくは 10 を選ぶ確率を 90% にし，選択肢 21 を選ぶ確率を 10% にする．発言しない確率を残すことで，人狼がおかしな発言をしないか様子をみる．2 回目の発言は発言していた場合，発言せずに様子をみる．発言していなかった場合，選択肢 8 もしくは 10 を選ぶ確率を 100% にする．3 回目の発言は，他のプレイヤーの今までの発言から，怪しい発言や矛盾を指摘する．

#### 3.2.4 怪盗の戦略

怪盗は人狼と交換するか，市民側と交換するかで戦略が大きく変わる．交換するプレイヤーは 50% ずつの確率で選ぶ．交換しないという選択肢もあるが，村人と同じ扱いになってしまうためコンピュータは選択しない．

表 17 は PlayerX と役職を交換した場合の怪盗が，最初の発言で選ぶ選択肢の確率である．

村人と交換した場合は，必ず最初に名乗り出る．1 回目の発言で適当に発言し，プレイヤーの役職を当てられる可能性は低いので大きな信頼が得られる．占い師と交換した場合も同様に，必ず最初に名乗り出る．2 回目以降は占い師が他の 2 枚が人狼だと発言していた場合選択肢 20 を 100% で選択する．それ以外の場合は発言をしない．

人狼と交換した場合は人狼側になるので，人狼と悟られないように発言する必要がある．選択肢 4 は表 2 の b の組み合わせの場合，完全に占い師になりすますことができるため疑われにくい．c の組み合わせの場合，占い師に疑われてしまうが，人狼にとってはどちらかが嘘を付いていることはわかるが，どちらが怪盗かは分からないため，投票が割れやすい．投票が割れれば人狼の勝利となるため，人狼と交換した怪盗には都合がいい．f の組み合わせの場合，人狼同士は仲間であると認識しているので，選択肢 4 の発言は嘘を付いていることになる．そのため怪盗がいることが疑われ，投票が割れる可能性がある．このようにどの組み合わせでも不利にはならないので選択肢 4 が選ばれる確率が一番高い．

選択肢 7 は交換したプレイヤーが人狼だと分かっているため，占い師になりすますことができる．しかし占い師に疑われる上，他のプレイヤーにも人狼と交換した怪盗だと疑われやすい．

選択肢 14 は表 2 の c の組み合わせの場合，占い師から高い信頼を得ることができる．また f の組み合わせの場合，怪盗がいることが疑われ，投票が割れる可能性がある．しかし b の組み合わせの場合，村人から疑われやすい．2 つの組み合わせで不利にならない発言なため確率を高くしている．

選択肢 21 は占い師が名乗り出るか様子をみる．

2 回目以降の発言は人狼と同じ戦略を用いる．

### 3.3 投票の時間

投票の時間では，他のプレイヤーの発言の記録を参照し，投票先を決定する．他のプレイヤーの発言に怪しい点があれば，その疑わしさを数値化して加算していき，その数値の合計が最も高いプレイヤーに投票する．

表 6　怪盗が選ぶ選択肢の確率

| | 交換した役職 | | |
|---|---|---|---|
| 選択肢 | 村人 | 占い師 | 人狼 |
| 4 | 0% | 0% | 40% |
| 7 | 0% | 0% | 10% |
| 11 | 100% | 0% | 0% |
| 12 | 0% | 0% | 0% |
| 13 | 0% | 100% | 0% |
| 14 | 0% | 0% | 30% |
| 21 | 0% | 0% | 20% |

また発言次第では確率によって投票する．

以下に本研究で作成した AI が用いている疑わしさの算出基準及び，投票基準を述べる．

表 7，表 8 はそれぞれ村人と占い師の他のプレイヤーの発言に対する疑わしさの数値である．他のプレイヤーが嘘を付いている事が確実な発言は，数値が高く設定されている．発言しないような目立たない行動をよく取るプレイヤーは怪しいので，少し加算している．

村人の場合，他のプレイヤーがもう 1 人のプレイヤーに占い結果が人狼だと発言しているなら，怪盗が人狼と交換している可能性もあるため，数値に関係なく発言したプレイヤーに 70%，もう 1 人のプレイヤーには 30% の確率で投票する．

占い師の場合，プレイヤーを占って人狼だった場合，怪盗が自分と交換していたら 100% の確率でそのプレイヤーに投票する．占っていないプレイヤーが村人だと名乗り出た場合，占ったプレイヤーは 90%，占っていないプレイヤーには 10% の確率で投票する．それ以外の場合，怪盗が人狼と交換している可能性が高いので，占っていないプレイヤーに 100% の確率で投票する．

プレイヤーを占って人間だった場合，占っていないプレイヤーが怪盗で自分と交換していたと発言していたなら，占ったプレイヤーに 20% の確率で，占っていないプレイヤーには 80% の確率で投票する．これは平和村の可能性があるので，平和村を狙って投票が割れやすくするためである．それ以外は占っていないプレイヤーに 100% の確率で投票する．

他の 2 枚を占った結果が村人と怪盗だった場合，どちらでも同じなのでランダムに投票する．他の 2 枚が人狼と人狼だった場合，平和村にする．

怪盗の投票基準は，交換したプレイヤーが人狼ならそのプレイヤーに投票し，交換したプレイヤーが市民側なら，交換していないプレイヤーに投票する．人狼と交換した場合，そのプレイヤーは市民側，自身は人狼側になるので，味方陣営に投票する事がない．占い師と交換し，その占い師が他の 2 枚をともに人狼と言っているなら，平和村にする．

人狼の投票基準は，基本的には投票が割れても人狼側の勝利になるので，ランダムに投票する．しかし怪盗がいる可能性があるので，それは見極める必要がある．怪盗は占い師か，怪盗だと発言する事が多いので，その発言をしたプレイヤーに投票する．2 人以上いる場合はランダムに投票する．特に自分を占って人狼だと発言しているプレイヤーには，優先的に投票する．仲間がいた場合，仲間ではないプレイヤーに投票する．ただし，仲間ではないプレイヤーが怪盗か占い師と名乗りでている場合，怪盗の可能性があるので，そのプレイ

ヤーに 90% の確率で投票し，仲間のプレイヤーは 10% の確率で投票する．

表 7　村人の算出基準

| 他のプレイヤーの発言 | 数値 |
|---|---|
| 怪盗が他のプレイヤーの村人と交換している | +200 |
| 怪盗が自分の占い師と交換したと発言している | +200 |
| 人狼か村人だと発言している | +100 |
| 占い結果が他の 2 枚に村人がいる | +100 |
| 占い結果が自分を人狼だと発言している | +100 |
| 一度も発言していない | +50 |
| 占い師がでた後に占い師というプレイヤーがいる | +20 |
| 発言しない | +10 |
| 怪盗が自分の村人と交換したと発言している | -50 |

表 8　占い師の算出基準

| 占い結果 | 他のプレイヤーの発言 | 数値 |
|---|---|---|
| 他の二枚が人狼と怪盗 | 人狼，占い師，怪盗だと名乗り出ている | +100 |
| | 一度も発言していない | +50 |
| | 占い結果のあとに村人と発言している | +20 |
| | 発言しない | +10 |
| 他の二枚が人狼と村人 | 怪盗が自分と交換して占い師と発言している | -500 |
| | 占い師，村人と名乗り出ている | +100 |
| | 占い結果が他のプレイヤーを人狼だと発言している | +300 |
| | 一度も発言しない | +50 |
| | 占い結果を発言後怪盗が名乗り出る | +50 |
| | 最初の発言で人狼だと名乗り出ている | +50 |
| | 占い結果発言後人狼だと名乗り出ている | -20 |
| | 発言しない | +10 |

## 3.4　作成したプログラム

本研究では Python 言語を用いてワンナイト人狼のアプリケーションを作成した．作成したプログラムは表 9 の通りである．付録に本研究で作成したプログラムを示す．

初めに 3 人でプレイすることができるワンナイト人狼を作成した．その後プレイヤー人数を設定するとコンピュータが追加できるよう作成した．プレイヤー人数を 0 人に設定すると，コンピュータ同士の対戦が可能である．

表 9　作成したプログラム

| ファイル名 | 説明 |
|---|---|
| werewolf.py | プレイ人数を確認しゲームを進める |
| viewer.py | GUI を用い実際にゲームをプレイする |
| choices.py | 選択肢を用意し各プレイヤーの発言を記録する |
| vote.py | 投票を集計し処刑者を決める |
| villager.py | 村人の夜のアクションや選択肢，投票先を決める |
| wolf.py | 人狼の夜のアクションや選択肢，投票先を決める |
| fortune_teller.py | 占い師の夜のアクションや選択肢，投票先を決める |
| phantom_thief.py | 怪盗の夜のアクションや選択肢，投票先を決める |
| computer.py | コンピュータがプレイする |
| cp_werewolf.py | コンピュータのみでゲームを進める |
| newgame.ipynb | このプログラムを実行しゲームを始める |

#### 3.4.1　プログラムの仕様

本節では，本研究で作成したプログラムの仕様について述べる．newgame.ipynb を実行することでゲームが開始する．このプログラムのプレイヤー数を表す people を 3 にすると通常の人狼ゲームがプレイできる．2 にするとプレイヤー 3 がコンピュータに置き替わり，2 人でプレイすることができる．1 にするとプレイヤー 2，プレイヤー 3 がコンピュータに置き替わり，1 人でプレイすることができる．0 にするとコンピュータ同士の対戦となり，対戦データを取得することができる．

図 1，図 2 に本研究で作成したプログラムを people を 1 に設定して実行し，ゲームを進めている様子を示す．

初めにスタート画面のスタートボタンを押し，ゲームが始まる．次に夜のターンになるので，OK ボタンを押す．プレイヤー 1 の役職確認が始まり，役職を見るボタンを押すと，「あなたは占い師です」のように役職が表示されるので，確認する．この画面では，村人と人狼は OK ボタンを押し，占い師は図のようにプレイヤーか，選ばれていない役職の中から占いたい対象のボタンを選択し，怪盗は交換するプレイヤーか，交換しないボタンを選択する．それぞれの役職でボタンを押した次の画面は，夜のアクションの結果が表示されるので確認し，OK ボタンを押す．次の画面では夜が明けるので発言したい内容を選択する．選択すると全員の発言内容が表示され，確認したなら話し合いを続けるボタンを押す．話し合いを 3 回繰り返すとこのボタンが話し合いを終了するボタンに変わる．そのボタンを押すと，投票するか平和村にするかを選択する．投票するボタンを選択すると，他のプレイヤーが表示されるので，投票したいプレイヤーの横の投票ボタンを押す．次の画面では最も投票が多かったプレイヤーが処刑され，OK ボタンを押すとゲーム結果が表示される．

夜のアクション，発言を選択する，投票の 3 つの画面は 2 人プレイならば 2 回，3 人プレイなら 3 回のように人数分表示される．


ワンナイト人狼
ワンナイト
人狼
スタート
ワンナイト人狼
恐ろしい夜の時間になりました。。。
各プレイヤーは役職を確認して行動してください。
他のプレイヤーに役職を見せてはいけません。
OK
ワンナイト人狼
Player1の方だけ操作してください
役職を見る
ワンナイト人狼
あなたは占い師です。
他のプレイヤーの役職を確認することができます。
役職を確認するプレイヤーを選んでください。
Player2
Player3
選ばれていない役職
ワンナイト人狼
選ばれていない役職は
村人と人狼です。
OK
ワンナイト人狼
Player1の方は発言を選択してください。
私は村人です。
私は人狼です。
私は占い師です。他の2枚は人狼と人狼でした。
私は占い師です。他の2枚は人狼と怪盗でした。
私は占い師です。他の2枚は人狼と村人でした。
私は占い師です。他の2枚は怪盗と村人でした。
私は占い師です。Player2は人狼でした。
私は占い師です。Player2は人間でした。
私は占い師です。Player3は人狼でした。
私は占い師です。Player3は人間でした。
私は怪盗です。Player2と交換して村人でした。
私は怪盗です。Player3と交換して村人でした。
私は怪盗です。Player2と交換して占い師でした。
私は怪盗です。Player3と交換して占い師でした。
私は怪盗です。交換しませんでした。
私はPlayer2を信じます。
私はPlayer3を信じます。
私はPlayer2が怪しいです。
私はPlayer3が怪しいです。
平和村にしましょう。
発言しない

図１　実行結果１

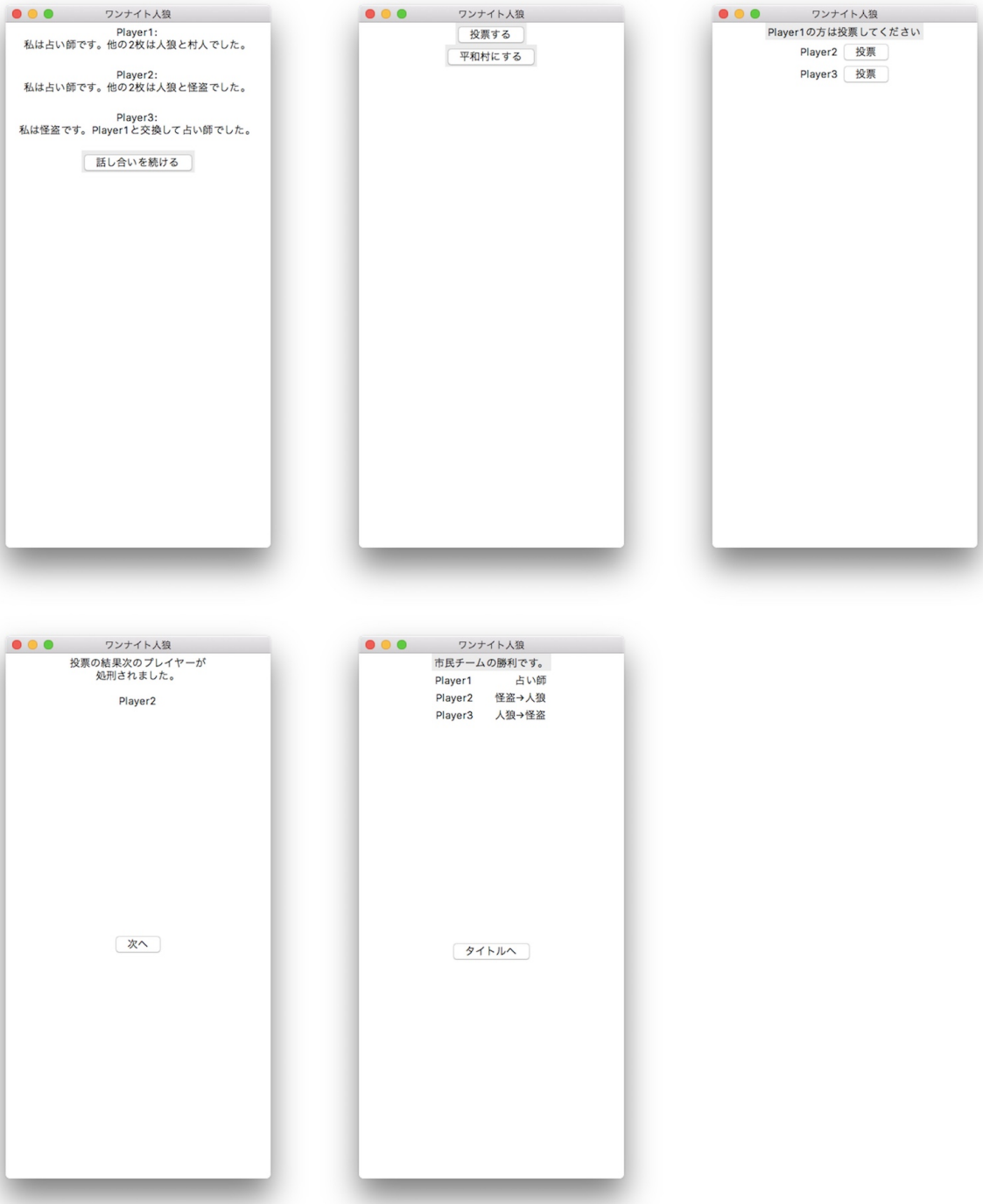
ワンナイト人狼
Player1:
私は占い師です。他の2枚は人狼と村人でした。
Player2:
私は占い師です。他の2枚は人狼と怪盗でした。
Player3:
私は怪盗です。Player1と交換して占い師でした。
話し合いを続ける
ワンナイト人狼
投票する
平和村にする
ワンナイト人狼
Player1の方は投票してください
Player2 投票
Player3 投票
ワンナイト人狼
投票の結果次のプレイヤーが
処刑されました。
Player2
次へ
ワンナイト人狼
市民チームの勝利です。
Player1 占い師
Player2 怪盗→人狼
Player3 人狼→怪盗
タイトルへ

図 2　実行結果 2

### 3.4.2 werewolf.py

ゲームの開始，役職の用意，ゲームの結果判定など全体の流れに関するプログラムである．

表 10 に werewolf.py に含まれるクラス，各メソッドを示す．表中のメソッド get_wolf_win get_job_counts までのメソッドはコンピュータの対戦で用いるためのデータである．

表 10 werewolf.py の各メソッド

| クラス | |
|---|---|
| progress | 人狼ゲームの進行に関するクラス |
| コンストラクタ | |
| __init__(self, people) | 人数を確認 |
| メソッド (関数) | |
| start(self) | ゲームを開始させる．people が 0 ならコンピュータのみで開始 |
| position_set(self) | 役職を用意する |
| random_list(self) | 役職をランダムに選ぶためのリストを準備 |
| choice_list(self, n) | プレイヤー番号によって違う選択肢を作成 |
| choice_log(self) | 発言を記録する |
| log1(self) | プレイヤー 1 の発言記録を返す |
| log2(self) | プレイヤー 2 の発言記録を返す |
| log3(self) | プレイヤー 3 の発言記録を返す |
| total_voted(self) | プレイヤーごとの投票を集計する |
| most_voted(self) | 最も投票されたプレイヤーの番号を返す |
| game_result(self) | ゲーム結果を判定し，結果を返す |
| get_wolf_win(self) | 人狼側の勝利数を返す |
| get_citizen_win(self) | 市民側の勝利数を返す |
| get_role_win(self) | 役職別の勝利数を返す |
| get_p1(self) | プレイヤー 1 の役職別勝利数を返す |
| get_p2(self) | プレイヤー 2 の役職別勝利数を返す |
| get_p3(self) | プレイヤー 3 の役職別勝利数を返す |
| players_win(self) | プレイヤー別の勝利数を返す |
| get_job_counts(self) | プレイヤー別の役職になった回数 |

### 3.4.3 viewer.py

GUI に関するプログラムである．役職，選択肢，話し合い，投票，ゲーム結果などをウィンドウに表示する．

表 11 に viewer.py に含まれるクラス，各メソッドを示す．

### 3.4.4 choices.py

選択肢に関するプログラムである．選択肢の作成，発言の記録を行う．

表 11　viewer.py の各メソッド

| クラス | |
|---|---|
| Viewer(tk.Tk) | tkinter モジュールを継承したクラス |
| コンストラクタ | |
| __init__(self, players, werewolf, p_number) | 各プレイヤーの役職情報，インスタンス，人数を受け取る |
| メソッド (関数) | |
| start_frame(self) | スタート画面を作成 |
| confirm(self) | 夜のターン開始画面作成 |
| player_turn(self) | 役職割り当て画面作成 |
| role_confirm(self) | 役職確認画面作成 |
| execution(self) | 夜のアクション画面作成 |
| discussion(self) | 話し合いの時間画面作成 |
| distinction(self, event) | 選択肢を判別し記録する |
| remark(self) | 発言を表示する画面作成 |
| confirmation(self) | 投票か平和村か選ぶ画面作成 |
| vote_time(self) | 投票画面作成 |
| check(self, n) | 投票を整理 |
| judge(self, is_peace = False) | 処刑する画面作成 |
| result(self) | ゲーム結果表示画面作成 |
| restart(self) | タイトルへ戻った場合全て初期化しスタート画面に戻る |
| change_page(self, page) | ページ遷移用の関数 |

表 12 に choices.py に含まれるクラス，各メソッドを示す．

### 3.4.5　vote.py

投票に関するプログラムである．

表 13 に vote.py に含まれるクラス，各メソッドを示す．

### 3.4.6　villager.py

村人の役職に関するプログラムである．

表 14 に villager.py に含まれるクラス，各メソッドを示す．

### 3.4.7　wolf.py

人狼の役職に関するプログラムである．

表 15 に wolf.py に含まれるクラス，各メソッドを示す．

### 3.4.8　fortune_teller.py

占い師の役職に関するプログラムである．

表 12　choices.py の各メソッド

| クラス | |
|---|---|
| Choices | 選択肢に関するクラス |
| コンストラクタ | |
| __init__(self) | 各種変数を準備 |
| メソッド (関数) | |
| make_choice(self,n) | 選択肢を作る |
| get_coming_outs(self) | 選択肢リストを返す |
| choice_log(self) | 発言を記録する |
| log1(self) | プレイヤー 1 の発言記録を返す |
| log2(self) | プレイヤー 2 の発言記録を返す |
| log3(self) | プレイヤー 3 の発言記録を返す |

表 13　vote.py の各メソッド

| クラス | |
|---|---|
| Vote | 投票に関するクラス |
| コンストラクタ | |
| __init__(self, p) | プレイヤーの数だけリストを準備 |
| メソッド (関数) | |
| total_voted(self, n) | 各プレイヤーに投票された数を合計する |
| most_voted(self) | 最も投票の多かったプレイヤーを集計 |
| punishmented_index(self) | 処刑者のリストを返す |

表 14　vilager.py の各メソッド

| クラス | |
|---|---|
| Villager | 村人クラス |
| コンストラクタ | |
| __init__(self, status) | 役職名を村人にする |
| メソッド (関数) | |
| role(self) | 役職を返す |
| action(self) | 夜のアクションがないことを表示する |
| target(self, n, players) | 空のメソッド |
| another_choices(self, choices, n) | プレイヤー番号によって違う選択肢を取得 |
| choice_rate(self, choices, log_list, c, pn) | コンピュータが確率で選んだ発言を返す |
| doubt(self, pn, choices, log_list) | コンピュータの投票先を決める |

表 15　wolf.py の各メソッド

| クラス | |
|---|---|
| Wolf | 人狼クラス |
| コンストラクタ | |
| __init__(self, status) | 役職名を人狼にする |
| メソッド (関数) | |
| role(self) | 役職を返す |
| action(self) | 夜のアクション結果を返す |
| target(self, n, players) | 仲間を確認する |
| another_choices(self, choices, n) | プレイヤー番号によって違う選択肢を取得 |
| choice_rate(self, choices, log_list, c, pn) | コンピュータが確率で選んだ発言を返す |
| doubt(self, pn, choices, log_list) | コンピュータの投票先を決める |

表 16 に fortune_teller.py に含まれるクラス，各メソッドを示す．

表 16　fortune_teller.py の各メソッド

| クラス | |
|---|---|
| FortuneTeller | 占い師クラス |
| コンストラクタ | |
| __init__(self, status) | 役職名を占い師にする |
| メソッド (関数) | |
| role(self) | 役職を返す |
| action(self) | 夜のアクション結果を返す |
| target_text(self, people, count) | 占いの対象を選ぶボタンのテキストを返す |
| target(self, n, players) | コンピュータの占いの対象を決める |
| another_choices(self, choices, n) | プレイヤー番号によって違う選択肢を取得 |
| case(self) | 占い結果によって場合分けする |
| choice_rate(self, choices, log_list, c, pn) | コンピュータが確率で選んだ発言を返す |
| doubt(self, pn, choices, log_list) | コンピュータの投票先を決める |

#### 3.4.9　phantom_thief.py

怪盗の役職に関するプログラムである．

表 17 に phantom_thief.py に含まれるクラス，各メソッドを示す．

#### 3.4.10　computer.py

コンピュータのプレイヤーに関するプログラムである．実際のプレイヤーに代わり，コンピュータがプレイする．

表 17　phantom_thief.py の各メソッド

| クラス | |
|---|---|
| FortuneTeller | 怪盗クラス |
| コンストラクタ | |
| __init__(self, status) | 役職名を怪盗にする |
| メソッド (関数) | |
| role(self) | 役職を返す |
| action(self) | 夜のアクション結果を返す |
| target_number | 交換したプレイヤーの番号を返す |
| target_text(self, people, count) | 交換の対象を選ぶボタンのテキストを返す |
| target(self, n, players) | コンピュータの交換の対象を決める |
| get_target(self) | コンピュータが交換したプレイヤー番号を返す |
| another_choices(self, choices, n) | プレイヤー番号によって違う選択肢を取得 |
| choice_rate(self, choices, log_list, c, pn) | コンピュータが確率で選んだ発言を返す |
| doubt(self, pn, choices, log_list) | コンピュータの投票先を決める |

表 18 に computer.py に含まれるクラス，各メソッドを示す．

表 18　computer.py の各メソッド

| クラス | |
|---|---|
| Computer | 実際のプレイヤーに代わりプレイするクラス |
| コンストラクタ | |
| __init__(self, position, number, ch) | 役職やプレイヤー番号を受け取る |
| メソッド (関数) | |
| divine(self, players) | 占い師か怪盗の場合ターゲットを選ぶ |
| cp_target(self) | 選んだターゲットを取得 |
| choice(self, log_list, count) | 発言を選ぶ |
| another_choices(self, choices, n) | プレイヤー別の選択肢を保存 |
| first(self, log_list) | 1 回目の発言を決める |
| second((self, log_list) | 2 回目の発言を決める |
| third(self, log_list) | 3 回目の発言を決める |
| doubt(self, log_list) | 投票先を決める |

#### 3.4.11　cp_werewolf.py

コンピュータ同士の対戦に関するプログラムである．コンピュータ同士の対戦の場合，viewer.py は使用せず，こちらのプログラムを使用する．

表 19 に cp_werewolf.py に含まれるクラス，各メソッドを示す．

表 19　cp_werewolf.py の各メソッド

| クラス | |
|---|---|
| Cp | コンピュータ同士の人狼を進めるクラス |
| コンストラクタ | |
| __init__(self, players, werewolf) | コンピュータの準備 |
| メソッド (関数) | |
| action(self) | 夜のアクション |
| discussion(self) | 話し合い |
| vote_time(self) | 投票 |
| result(self) | ゲーム結果 |
| get_wolf_win(self) | 人狼側の勝利数を返す |
| get_citizen_win(self) | 市民側の勝利数を返す |
| get_role_win(self) | 役職別の勝利数を返す |
| get_p1(self) | プレイヤー 1 の役職別勝利数を返す |
| get_p2(self) | プレイヤー 2 の役職別勝利数を返す |
| get_p3(self) | プレイヤー 3 の役職別勝利数を返す |
| players_win(self) | プレイヤー別の勝利数を返す |

#### 3.4.12　newgame.ipynb

ゲームを実行するプログラムである．クラスやコンストラクタ，関数は含まれない．

変数 people を 1，2，3 のいずれかに設定し実行すると，ワンナイト人狼をプレイできる．people を 0 に設定し実行すると，コンピュータ同士の対戦を複数回繰り返し，人狼側と市民側の勝利数や，各プレイヤーの勝利数などのデータを見ることができる．

## 4　結果

2.3 節で述べたワンナイト人狼のルールから，夜のアクション及び話し合いを行わずランダムに投票を行った場合，人狼の勝率は 67.5%，市民側の勝率は 32.5% である．そして 1 人のプレイヤーが処刑され続けた場合の勝率は，役職が人狼になる確率が 40% であるため，人狼側が 60%，市民側が 40% となる．

また本研究で作成したワンナイト人狼の AI を，話し合いを行わず夜のアクションだけ行い，コンピュータ同士で 100,000 回実行した．投票の際，人狼は仲間がいれば仲間には投票しない，占い師は占った結果が人狼だった場合のみ投票，怪盗は交換したプレイヤーが人狼ならそのプレイヤーに投票，それ以外なら交換していないプレイヤーに投票し，これら以外の場合は全てランダムに投票した．その結果，人狼側の勝率は約 68.3%，市民側の勝率は約 31.7% であった．

このことから，ワンナイト人狼は話し合いがなければ，人狼側の勝率が高いゲームである．下記の結果はこれを基準に比較する．

本研究で作成したワンナイト人狼の AI を人数を 1 人に設定しプレイした．1 セット 10 ゲームとし，10 セット合計 100 ゲーム行った．表 20 はその結果である．P2（プレイヤー 2），P3（プレイヤー 3）はコン

ピュータである．役職別の勝利数は，怪盗の交換前の役職で集計している．

表 20 から，話し合いが加わることで市民側の勝率が大幅に上がった．役職別の勝率を見ると怪盗が一番高い結果となった．またプレイ回数が 100 ゲームと少ないが，コンピュータではないプレイヤー 1 が一番勝率が低い結果となった．

表 21 にコンピュータ同士のゲーム結果を示す．ゲームを 100,000 回行った結果を集計している．また人狼が発言しない結果も加えて集計している．プレイヤーの中に人狼がいなかった場合は，プレイヤーの勝利数に差が開かないため，省いて集計している．また役職別勝率は怪盗が交換する前の役職で集計している．

通常プレイでの結果は表 20 の結果と同じく，市民側の勝率が高く，怪盗の勝率が一番高かった．人狼は発言しないという条件をつけると，人狼側の勝率がさらに下がった．

表 20　ゲーム結果 (試行回数 100 回)

| | 陣営勝利数 | | プレイヤー別勝利数 | | | P1 役職別勝利数/回数 | | | |
|---|---|---|---|---|---|---|---|---|---|
| セット | 人狼側 | 市民側 | P1 | CPU2 | CPU3 | 人狼 | 村人 | 占い師 | 怪盗 |
| 1 | 4 | 6 | 6 | 6 | 5 | 4/7 | 1/1 | 0/1 | 1/1 |
| 2 | 4 | 6 | 7 | 7 | 5 | 3/4 | 1/2 | 3/4 | 0/0 |
| 3 | 3 | 7 | 7 | 5 | 6 | 1/2 | 0/1 | 5/5 | 1/2 |
| 4 | 6 | 4 | 4 | 7 | 6 | 2/3 | 2/4 | 0/1 | 0/2 |
| 5 | 2 | 8 | 6 | 6 | 8 | 0/1 | 1/3 | 2/2 | 3/3 |
| 6 | 5 | 5 | 3 | 6 | 7 | 3/7 | 2/3 | 0/3 | 1/1 |
| 7 | 4 | 6 | 5 | 4 | 7 | 3/7 | 0/1 | 1/1 | 1/1 |
| 8 | 5 | 5 | 6 | 8 | 5 | 3/4 | 0/2 | 1/2 | 2/2 |
| 9 | 4 | 6 | 5 | 6 | 8 | 1/2 | 1/2 | 3/6 | 0/0 |
| 10 | 4 | 6 | 5 | 7 | 6 | 2/5 | 0/2 | 1/2 | 2/2 |
| 合計 | 41 | 59 | 54 | 62 | 63 | 19/38 | 8/21 | 16/27 | 11/14 |
| 勝率 | 41.0% | 59.0% | 54.0% | 62.0% | 63.0% | 50.0% | 38.1% | 59.3% | 71.4% |

表 21　コンピュータ同士のゲーム結果 (試行回数 100,000 回)

| | 陣営勝率 | | 役職別勝率 | | | |
|---|---|---|---|---|---|---|
| 条件 | 人狼側 | 市民側 | 人狼 | 村人 | 占い師 | 怪盗 |
| 通常プレイ | 43.39% | 56.61% | 54.87% | 51.87% | 56.22% | 62.87% |
| 人狼は発言しない | 33.88% | 66.12% | 49.12% | 66.52% | 73.03% | 61.66% |

以下では，表 20 および表 21 の結果を統計的に検証する．

選択肢による話し合いを行っても，市民側の勝率は変わらないと仮定し，検証する．

勝率 $p$ の勝負を $N$ 回行った場合，標準偏差 $s$ は以下の式で表される．

$$s = \sqrt{N * p * (1 - p)}$$

表 20 の市民側の勝率より，$p = 0.59$ とすると，$N = 100$ ならば標準偏差は

$$\sqrt{100 * 0.59 * 0.41} = 4.92$$

となり，表 21 の市民側の勝率より，$p = 0.57$ とすると，$N = 100,000$ ならば標準偏差は

$$\sqrt{100,000 * 0.56 * 0.44} = 156.97$$

となる．信頼区間 95% となるのは，平均値との差がそれぞれ $4.92 * 1.96 = 9.64$, $156.97 * 1.96 = 307.66$ の区間である．従って，勝利回数は 95% の確率で，勝率が 59% ならば，100 ゲームでは $59 \pm 9.64$ 回に収まり，勝率が 56% ならば，100,000 ゲームでは $56,000 \pm 307.$ 回に収まる．

どちらの範囲も，ランダム投票の場合の，市民側の勝率である 32.5% が含まれないため，選択肢での話し合いがある場合の表 20，表 21 の結果は統計上有意である．

また人狼が発言しない場合，発言する場合と比べ，勝率は変わらないと仮定し，表 21 の通常プレイ及び人狼が発言しない場合の結果を統計的に検証する．

表 21 通常プレイの人狼側の勝率より，$p = 0.43$ とすると，$N = 100,000$ ならば標準偏差は

$$\sqrt{100,000 * 0.43 * 0.57} = 156.56$$

となり，表 21 の人狼は発言しない場合の人狼側の勝率より，$p = 0.34$ とすると，$N = 100,000$ ならば標準偏差は

$$\sqrt{100,000 * 0.34 * 0.66} = 149.80$$

となる．信頼区間 95% となるのは，平均値との差がそれぞれ $156.56 * 1.96 = 306.86$, $149.80 * 1.96 = 293.60$ の区間である．従って，100,000 ゲームでの勝利回数は 95% の確率で，勝率が 43% ならば，$43,000 \pm 306.86$ 回，勝率が 33% ならば，$33,000 \pm 293.$ 回に収まる．

どちらの信頼区間も，勝利数が重なる範囲がないため，表 21 の通常プレイ及び人狼が発言しない場合の結果は有意である．よって人狼の発言によって勝率は変わる．

## 5 考察

表 20 の結果では，市民側の勝率が人狼側の勝率を上回った．西崎らの分析 [7] では占い師の能力結果が投票に大きく関係していることが示されており，本研究の選択肢の話し合いにおいても，占い師の結果により，市民側に有利な情報が出たことが関係していると考えられる．また，怪盗の勝率が高かったことから，ワンナイト人狼では怪盗の能力も大きく関係していると考える．

プレイヤー 1 が勝率が一番低くなった理由として，コンピュータの戦略が堅実であったことが考えられる．プレイヤー 2，プレイヤー 3 の勝率が近いため，あまり目立たないよう騙っていたからであると考える．しかし，プレイヤー 1 は一番勝率が高い怪盗の回数が，一番少ないため，役職が偏ったことも原因であると考える．

表 21 でも同じく市民側の勝率が人狼側の勝率を上回り，表 20 の勝率と大きな差は無かった．人狼側の勝率が下がった理由として，人狼の発言が嘘だとわかりやすいと考えたが，人狼の発言を制限すると，人狼側の勝率が大きく下がった．一方で占い師の勝率が大幅に上昇していることから，人狼の発言は特に占い師を騙せていると考える．

本研究では，話し合いに選択肢を用いたことで戦略の幅に大きな制限がかかってしまっている．さらに，発言のタイミングが同時であるため，第一声が誰かというような，役職を見分ける要素が無くなっており，また相手の発言に合わせて相槌を打ったり，話を変えるなどの戦略にも制限がかかってしまっている．よって本研究のワンナイト人狼の AI を拡張する際は，話し合いの時間に自然言語を用いれば，戦略が大きく広がり，ゲーム性が増すと考える．

## 6　結論・今後の課題

本研究ではワンナイト人狼の AI を作成した．勝率の結果から実際にコンピュータとプレイができていると言える．選択肢での発言であっても，役職の情報があれば大きく戦況を動かせることが分かった．

今後の課題は，本研究で作成したワンナイト人狼は 3 人までしかプレイできないため，人数を増やしてプレイできるようにすることである．人数が 4 人になれば戦略も大きく変わり，投票基準もより複雑になる．さらに，処刑されたら勝ちである吊り人のような役職を追加すれば，嘘を付いているからという理由では簡単に投票できなくなるため，さらに深い話し合いが重要になる．

## 謝辞

本研究を行うにあたり，ご指導頂いた指導教員の石水隆講師に心より感謝致します。また，日常の議論を通じて多くの知識や示唆を頂戴いたしました情報論理工学研究室の皆様に深く感謝致します.

## 参考文献


[1] Noam, Brown and Tuomas Sandholm (2019) “Superhuman AI for multiplayer poker” Science, 365(6456): 885-890.

[2] 麻雀 AI Microsoft Suphx が人間のトッププレイヤーに匹敵する成績を達成, Japan News Center, Mictosoft (2019/8/29) https://news.microsoft.com/ja-jp/2019/08/29/190829-mahjong-ai-microsoft-suphx/

[3] 人狼知能プロジェクト, (2015) http://aiwolf.org/

[4] 鳥海不二夫, 片上大輔, 大澤博隆, 稲葉通将, 篠田孝祐, 狩野芳伸. 人狼知能　—だます・見破る・説得する人工知能—. 森北出版, (2016).

[5] 狩野芳伸, 大槻恭士, 園田亜斗夢, 中田洋平, 箕輪峻, 鳥海不二夫. 人狼知能で学ぶ AI プログラミング. マイナビ出版, (2017).

[6] 西崎絵麻, 坂口早紀, 尾崎知伸. ワンナイト人狼における投票行動の分析. The 31st Annual Conference of the Japanese Society for Artificial Intelligence, 2H1-5, pp.1-4, 人工知能学会, (2017).

[7] 西崎絵麻, 尾崎知伸. ワンナイト人狼を対象とした投票行動の特徴分析. 人工知能学会研究会, SIG-KBS-B508-09, pp.52-59, (2017).

[8] 玉井日菜子. 3 者間人狼における他者の投票行動を考慮した戦略の検討. The 33st Annual Conference of the Japanese Society for Artificial Intelligence, 3F3-OS-14a-05, pp.1-4, 人工知能学会, (2019).

# 付録 A　付録

- werewolf.py

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:



from random import randint
import villager
import wolf
import phantom_thief
import fortune_teller
import viewer
import vote
import choices
import cp_werewolf

# 人狼ゲームを進めるクラス
class Progress:

    # コンストラクタ
    def __init__(self, people):
        #プレイヤーの数
        self.people = people

        #クラスのインスタンス生成vote
        self.vote_time = vote.Vote(3)

        #クラスのインスタンス生成choice
        self.ch = choices.Choices()

    #ゲームを開始させる関数
    def start(self):
        #ゲームスタート
```

```
        if self.people in [1,2,3]:
            root = viewer.Viewer(self.positions_set(), self, self.
                people)
            root.mainloop()
        #コンピュータのゲームスタート
        else:
            self.cp = cp_werewolf.Cp(self.positions_set(), self)

    #役職を用意する関数
    def positions_set(self):
        #  ランダムな数を用意
        number = self.random_list()

        #  全ての役職を順不同準備
        positions = list(range(5))
        positions[number[0]] = villager.Villager('村人')
        positions[number[1]] = wolf.Wolf('人狼')
        positions[number[2]] = wolf.Wolf('人狼')
        positions[number[3]] = fortune_teller.FortuneTeller('占い師')
        positions[number[4]] = phantom_thief.PhantomThief('怪盗')

        # 役職をあらかじめ決める場合
        positions2 = list()
        positions2.append(phantom_thief.PhantomThief('怪盗'))
        positions2.append(villager.Villager('村人'))
        positions2.append(fortune_teller.FortuneTeller('占い師'))
        positions2.append(wolf.Wolf('人狼'))
        positions2.append(wolf.Wolf('人狼'))

        #役職のリストを返す
        return positions

    #ランダムな数字のリストを用意する関数
    def random_list(self):
        #空のリスト準備
        numbers = list()
```

```
        #重複しないようランダムな数字を役職の数だけ準備
        while len(numbers) < 5:
            n = randint(0,4)
            if not n in numbers:
                numbers.append(n)

        #準備したリストを返す
        return numbers

    #選択肢を取得する
    def choice_list(self, n):
        self.ch.make_choice(n)
        return self.ch.get_coming_outs()

     #過去の発言を記録する
    def choice_log(self, log):
        self.ch.choice_log(log)

     #の発言記録Player1
    def log1(self):
        return self.ch.log_1

    #の発言記録Player2
    def log2(self):
        return self.ch.log_2

    #の発言記録Player3
    def log3(self):
        return self.ch.log_3

    #投票数
    def total_voted(self, player_number):
        #クラスのメソッド呼び出しvote
        self.vote_time.total_voted(player_number)

    #最も投票されたプレイヤー
    def most_voted(self):
```

```
        #クラスのメソッドの戻り値を返すvote
        return self.vote_time.most_voted()

    #ゲーム結果
    def game_result(self, players):
        #プレイヤーに人狼がいたか
        is_wolf = False

        punishment_list = list()

        #処刑されたプレイヤーのインデックス
        index = self.vote_time.punishmented_index()

        #平和村の処理
        if len(index) == 0:
            for n in range(len(players)-2):
                if players[n].status == '人狼':
                    return '人狼チームの勝利です。'
            return '市民チームの勝利です。'

        #処刑者のリスト
        for n in index:
            punishment_list.append(players[n])

        #処刑者に人狼がいればにTrue
        for n in punishment_list:
            if n.status == '人狼':
                is_wolf = True

        #人狼が処刑されれば市民の勝ち
        if is_wolf:
            return '市民チームの勝利です。'
        else:
            return '人狼チームの勝利です。'

    def get_wolf_win(self):
        return self.cp.get_wolf_win()
```

```
    def get_citizen_win(self):
        return self.cp.get_citizen_win()

    def get_role_win(self):
        return self.cp.get_role_win()

    def get_p1(self):
        return self.cp.get_p1()

    def get_p2(self):
        return self.cp.get_p2()

    def get_p3(self):
        return self.cp.get_p3()

    def players_win(self):
        return self.cp.players_win()

    def get_job_counts(self):
        return self.cp.job_counts
```

- viewew.py

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import tkinter as tk
import werewolf
import computer

# に関するクラスGUI
class Viewer(tk.Tk):
    #ウィンドウ作成
    def __init__(self, players, werewolf, p_number):
        #呪文
```

```
        tk.Tk.__init__(self)

        self.win_players = [0] * 3

        self.wc = [0] * 2

        #各プレイヤーの役職情報
        self.players = players

#           for n in players:
#               print(n.status)

        #プレイヤーの人数
        self.p_number = p_number

        #クラスのインスタンスwerewolf
        self.werewolf = werewolf

        #カウント用変数の準備
        self.count = 1
        self.count2 = 1

        #コンピュータの準備、リストの番目の役職になる2
        self.cp2 = computer.Computer(players[1], 2, self.werewolf.ch)

        #コンピュータの準備、リストの番目の役職になる3
        self.cp3 = computer.Computer(players[2], 3, self.werewolf.ch)

        #怪盗の交換が行われたか
        self.is_changed = False

        #ウィンドウに関する設定
        self.title('ワンナイト人狼')
        self.geometry('335x665')
        self.grid_rowconfigure(0, weight=1)
        self.grid_columnconfigure(0, weight=1)
```

```
        #スタート画面のフレームを貼り付け
        self.start_frame()

    #スタート画面
    def start_frame(self):
        #スタート画面のフレーム生成
        frame1 = tk.Frame(self, bg='white',  width=335, height=665)
        frame1.grid(row=0, column=0, sticky='nsew')

        #各ラベルの生成
        title = tk.Label(frame1, text='ワンナイト
            ', font=('',18), bg='white')
        title2 = tk.Label(frame1, text='人狼
            ', font=('', 50, 'bold'), fg='Red', bg='white')
        start_button = tk.Button(frame1, text='スタート
            ', highlightbackground='white',
                        highlightthickness=1, command=lambda:self.
                            change_page(self.confirm()))

        #各ラベルの貼り付け
        title.pack(expand=True, anchor=tk.S)
        title2.pack(expand=True, anchor=tk.N)
        start_button.pack(expand=True)

        #スタート画面フレームを返す
        return frame1

    #ホーム画面
    def confirm(self):
        #スタート確認画面のフレーム作成
        frame2 = tk.Frame(self, bg='white',  width=335, height=665)
        frame2.grid(row=0, column=0, sticky='nsew')

        #各ラベルの生成
        title = tk.Label(frame2, text='恐ろしい夜の時間になりました。。。
            ', bg='white')
        explain = tk.Label(frame2, text='各プレイヤーは役職を確認して行動してく
            ださい。\他のプレイヤーに役職を見せてはいけません。n')
```

```
        ok_button = tk.Button(frame2, text='OK', highlightbackground='
            white',
                              highlightthickness=1, command=lambda: self.
                                  change_page(self.player_turn()))

        #各ラベルの貼り付け
        title.pack()
        explain.pack(expand=True, anchor=tk.S)
        ok_button.pack(expand=True)

        #スタート確認画面のフレームを返す
        return frame2

    #役職割り当て画面
    def player_turn(self):
        #用フレーム作成Player
        self.frame3 = tk.Frame(self, bg='white',  width=335, height
            =665)
        self.frame3.grid(row=0, column=0, sticky='nsew')

        #案内ラベルの生成
        self.title = tk.Label(self.frame3, text=f'Player{self.countの方
            だけ操作してください}', bg='white')

        #現在のプレイヤーの役職を保存
        self.job = self.players[self.count-1]

        #次の画面に遷移するボタンの生成
        self.enter = tk.Button(self.frame3, text='役職を見る
            ', highlightbackground='white',
                               highlightthickness=1, command=lambda: self.
                                   role_confirm())

        #各ラベルの貼り付け
        self.title.pack()
        self.enter.pack(expand=True)
```

```
        #フレームを返す
        return self.frame3

    #  ラベル情報リセット　役職確認画面
    def role_confirm(self):

        #ラベルを削除
        self.title.pack_forget()
        self.enter.pack_forget()

        #役職情報ラベルの生成、貼り付け
        self.info = tk.Label(self.frame3, text=self.job.role())
        self.info.pack(expand=True, anchor=tk.S)

        #プレイヤーの役職が村人もしくは人狼の場合
        if self.job.status == '村人' or self.job.status == '人狼':
            #夜のアクションに遷移するボタンの生成、貼り付け
            self.entry = tk.Button(self.frame3, text=('OK'),
                highlightbackground='white',
                            highlightthickness=1, command=lambda: self.
                                execution())
            self.entry.pack(expand=True)
        #プレイヤーの役職がそれ以外の場合
        else:
            #対象のプレイヤーの選択肢ボタン
            self.target_buttons = list()
            #ターゲットの選択肢
            target_texts = self.job.target_text(len(self.players)-2,
                self.count)
            #ターゲットの数だけボタンを生成、貼り付け
            for n in range(len(target_texts)):
                self.target_buttons.append(tk.Button(self.frame3,text=
                    target_texts[n],
                                                highlightbackground='white
                                                    ',highlightthickness=1)
                                                    )
                if n == 0:
```

```
                self.target_buttons[n].pack(expand=True, anchor=tk
                    .S)
            elif n == len(target_texts) - 1:
                self.target_buttons[n].pack(expand=True, anchor=tk
                    .N)
            else:
                self.target_buttons[n].pack()
            self.target_buttons[n].bind("<ButtonPress>", self.
                execution)

# 夜のアクション実行画面
def execution(self, event=None):
    #ラベルを削除
    self.info.pack_forget()
    if self.job.status == '村人' or self.job.status == '人狼':
        self.entry.pack_forget()
    else:
        for n in range(len(self.target_buttons)):
            self.target_buttons[n].pack_forget()

    #夜のアクション結果ラベルの生成
    if self.job.status == '村人':
        result = tk.Label(self.frame3, text=self.job.action())
    elif self.job.status == '人狼':
        result = tk.Label(self.frame3, text=self.job.action(self.
            players, self.count))
    elif self.job.status == '占い師':
        result = tk.Label(self.frame3,
                          text=self.job.action(event.widget.cget("
                              text"), self.players))
    else:
        result = tk.Label(self.frame3,
                          text=self.job.action(event.widget.cget("
                              text"), self.players))
        if event.widget.cget("text") != '交換しない':
            self.is_changed = True
            self.num = self.count-1
```

```
            self.target_number = self.job.target_number()

    #最後のプレイヤーでない場合のボタン
    if self.count < self.p_number:
        #カウントをする+1
        self.count += 1
        #次のプレイヤーに遷移するボタンの生成
        next_button = tk.Button(self.frame3, text=('OK'),
            highlightbackground='white',
                highlightthickness=1, command=lambda: self.
                    player_turn())
    #最後のプレイヤーの場合のボタン
    else:
        #カウントリセット
        self.count = 0
        #コンピュータの夜のアクション
        if self.p_number == 1:
            if not (self.players[1].status == '村人'):
                self.cp2.divine(self.players)
                if self.players[1].status == '怪盗':
                    self.is_changed = True
                    self.num = 1
                    self.target_number = self.cp2.cp_target()
            if not (self.players[2].status == '村人'):
                self.cp3.divine(self.players)
                if self.players[2].status == '怪盗':
                    self.is_changed = True
                    self.num = 2
                    self.target_number = self.cp3.cp_target()
        elif self.p_number == 2:
            if not (self.players[2].status == '村人'):
                self.cp3.divine(self.players)
                if self.players[2].status == '怪盗':
                    self.is_changed = True
                    self.num = 2
                    self.target_number = self.cp3.cp_target()
        #話し合いの時間に遷移するボタンの生成
```

```
        next_button = tk.Button(self.frame3, text=('OK'),
            highlightbackground='white',
                highlightthickness=1, command=lambda: self.
                    change_page(self.discussion()))

    #各ラベルの貼り付け
    result.pack(expand=True)
    next_button.pack(expand=True, anchor=tk.N)

#話し合いの時間画面
def discussion(self):
    self.count += 1
    #話し合いの時間用フレーム作成
    self.frame4 = tk.Frame(self, bg='white',  width=335, height
        =665)
    self.frame4.grid(row=0, column=0, sticky='nsew')
    self.frame4.grid_columnconfigure(0, weight=1)
    self.frame4.grid_columnconfigure(2, weight=1)

    #案内表示を作成、貼り付け
    self.title = tk.Label(self.frame4, text = f'Player{self.countの
        方は発言を選択してください。}')
    self.title.pack()

    #選択肢を用意
    self.choices = self.werewolf.choice_list(self.count)

    #選択肢の数だけボタン作成、貼り付け
    for n in self.choices:
        choice_button = tk.Button(self.frame4, text=n,
            highlightbackground='white',
                highlightthickness=1)
        choice_button.bind("<ButtonPress>", self.distinction)
        choice_button.pack()

    #フレームを返す
    return self.frame4
```

```
#選択肢を判別し、記録する
def distinction(self, event):
    self.werewolf.choice_log(event.widget.cget('text'))
    #各プレイヤーの選択肢を保存
    if self.p_number == 2:
        self.cp3.another_choices(self.choices, self.count)
    elif self.p_number == 1:
        self.cp2.another_choices(self.choices, self.count)
        self.cp3.another_choices(self.choices, self.count)
        if self.count == 1:
            self.choices = self.werewolf.choice_list(3)
            self.cp2.another_choices(self.choices, 3)
            self.choices = self.werewolf.choice_list(2)
            self.cp3.another_choices(self.choices, 2)

    if self.count >= self.p_number:
        #コンピュータの発言を記録
        log_list = [self.werewolf.log1(), self.werewolf.log2(),
            self.werewolf.log3()]
        if self.p_number == 2:
            cp_text = self.cp3.choice(log_list, self.count2)
            self.werewolf.choice_log(cp_text)
        elif self.p_number == 1:
            cp2_text = self.cp2.choice(log_list, self.count2)
            cp3_text = self.cp3.choice(log_list, self.count2)
            self.werewolf.choice_log(cp2_text)
            self.werewolf.choice_log(cp3_text)
        self.remark()
    else:
        self.discussion()

#選択した発言を表示する
def remark(self):
    frame8 = tk.Frame(self, bg='white',  width=335, height=665)
    frame8.grid(row=0, column=0, sticky='nsew')
    self.count = 0
```

```
        self.count2 += 1
        p1 = tk.Label(frame8, text='Player1:\n' + self.werewolf.log1()
            [-1] + '\n', bg='white')
        p2 = tk.Label(frame8, text='Player2:\n' + self.werewolf.log2()
            [-1] + '\n', bg='white')
        p3 = tk.Label(frame8, text='Player3:\n' + self.werewolf.log3()
            [-1] + '\n', bg='white')
        p1.pack()
        p2.pack()
        p3.pack()

        if self.count2 > 3:
            end_button = tk.Button(frame8, text='話し合いを終了する',
                command=lambda: self.change_page(self.confirmation()))
            end_button.pack()
        else:
            continue_button = tk.Button(frame8, text='話し合いを続ける
                ', command=lambda: self.change_page(self.discussion()))
            continue_button.pack()
#               end_button = tk.Button(frame8, text話し合いを終了する
    ='', command=lambda: self.change_page(self.confirmation()))
#               end_button.pack()

    #投票か平和村にするか確認
    def confirmation(self):
        frame9 = tk.Frame(self, bg='white',  width=335, height=665)
        frame9.grid(row=0, column=0, sticky='nsew')

        vote_button = tk.Button(frame9, text='投票する
            ', command=lambda: self.change_page(self.vote_time()))
        peace_button = tk.Button(frame9, text='平和村にする
            ', command=lambda: self.change_page(self.judge(True)))

        vote_button.pack()
        peace_button.pack()

        return frame9
```

```
#投票時間
def vote_time(self):
    self.count += 1
    player_labels = list()
    vote_buttons = list()
    #投票時間フレーム作成
    frame5 = tk.Frame(self, bg='white',  width=335, height=665)
    frame5.grid(row=0, column=0, sticky='nsew')
    frame5.grid_columnconfigure(0, weight=1)
    frame5.grid_columnconfigure(1, weight=1)

    title = tk.Label(frame5, text=f'Player{self.countの方は投票してくだ
        さい}')

    #投票先プレイヤー情報表示
    for n in range(len(self.players)-2):
        if n+1 == self.count:
            continue
        player_labels.append(tk.Label(frame5, text=f'Player{n+1}',
            bg='white'))
        vote_buttons.append(tk.Button(frame5, text='投票
            ',highlightbackground='white',
              highlightthickness=1, command=self.check(n+1)))

    #各ラベル貼り付け
    title.grid(column=0, row=0, columnspan=2)
    for n in range(len(player_labels)):
        player_labels[n].grid(column=0, row=n+1, sticky = tk.E)
        vote_buttons[n].grid(column=1, row=n+1, sticky = tk.W)

    #フレームを返す
    return frame5

#投票の整理、投票が終わるかの判定
def check(self, n):
    def check2():
```

```
            self.werewolf.total_voted(n)

            #プレイヤー全員の投票が終われば処刑画面へ
            if self.count >= self.p_number:
                log_list = [self.werewolf.log1(), self.werewolf.log2()
                    , self.werewolf.log3()]
                #コンピュータの投票
                if self.p_number == 2:
                    self.werewolf.total_voted(self.cp3.doubt(log_list)
                        )
                elif self.p_number == 1:
                    self.werewolf.total_voted(self.cp2.doubt(log_list)
                        )
                    self.werewolf.total_voted(self.cp3.doubt(log_list)
                        )
                self.change_page(self.judge())
            else:
                self.change_page(self.vote_time())
        return check2

    #処刑画面
    def judge(self, is_peace = False):
        #処刑画面用フレーム作成
        frame6 = tk.Frame(self, bg='white',  width=335, height=665)
        frame6.grid(row=0, column=0, sticky='nsew')
        if is_peace:
            execution = tk.Label(frame6, text='平和村になりました
                ', bg='white')
        else:
            execution = tk.Label(frame6, text=self.werewolf.most_voted
                (), bg='white')
        next_page = tk.Button(frame6, text='次へ
            ', highlightbackground='white',
                        highlightthickness=1, command=lambda:self.
                            change_page(self.result()))

        execution.pack()
```

```
        next_page.pack(expand = True)

        #フレームを返す
        return frame6

    #ゲーム結果
    def result(self):
        #怪盗の交換が行われた場合役職を交換する
        if self.is_changed:
            self.players[self.target_number], self.players[self.num] =
                \
            self.players[self.num], self.players[self.target_number]
        #ゲーム結果画面作成
        frame7 = tk.Frame(self, bg='white',  width=335, height=665)
        frame7.grid(row=0, column=0, sticky='nsew')

        position = list()

        t = self.werewolf.game_result(self.players)

        if t == '人狼チームの勝利です。':
            self.wc[0] += 1
            for n in range(3):
                if self.players[n].status == '人狼':
                    self.win_players[n] += 1
        else:
            self.wc[1] += 1
            for n in range(3):
                if self.players[n].status != '人狼':
                    self.win_players[n] += 1
        print(self.wc)
        print(self.win_players)

#           results = tk.Label(frame7, text=self.werewolf.game_result(
    self.players))
        results = tk.Label(frame7, text=t)
```

```
#怪盗が交換していた場合
if self.is_changed:
    for n in range(len(self.players)-2):
        if n == self.target_number:
            position.append(tk.Label(frame7, text=f'Player{n
                +1}' + \
                                    f'{self.players[self.num
                                        ].status→
                                        }{self.players[n].
                                        status}'.rjust(6, '
                                        '), bg='white'))
        elif n == self.num:
            position.append(tk.Label(frame7, text=f'Player{n
                +1}' + \
                                    f'{self.players[self.
                                        target_number].status→
                                        }{self.players[n].
                                        status}'.rjust(6, '
                                        '), bg='white'))
        else:
            position.append(tk.Label(frame7, text=f'Player{n
                +1}' + f'{self.players[n].status}'.rjust(6, '
                '), bg='white'))
#全プレイヤーの役職公開
else:
    for n in range(len(self.players)-2):
        position.append(tk.Label(frame7, text=f'Player{n
            +1}' + f'{self.players[n].status}'.rjust(3, '
            '), bg='white'))

title_page = tk.Button(frame7, text='タイトルへ
    ',highlightbackground='white',
                highlightthickness=1, command=lambda:self.
                    restart())

#各ラベル貼り付け
results.pack()
for n in range(len(position)):
```

```
            position[n].pack()
        title_page.pack(expand=True)
        #フレームを返す
        return frame7

    #タイトルへ戻った場合全て初期化
    def restart(self):
        #変数の初期化
        self.werewolf = werewolf.Progress(self.p_number)
        self.players = self.werewolf.positions_set()
        self.count = 1
        self.count2 = 1
        #コンピュータの準備、リストの番目の役職になる2
        self.cp2 = computer.Computer(self.players[1], 2, self.werewolf
            .ch)
        #コンピュータの準備、リストの番目の役職になる3
        self.cp3 = computer.Computer(self.players[2], 3, self.werewolf
            .ch)
        self.is_changed = False
        self.change_page(self.start_frame())
#           for n in self.players:
#               print(n.status)

    #ページ遷移用関数
    def change_page(self, page):
        page.tkraise()
```

- choices.py

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


class Choices:

    def __init__(self):
```

```
        #カウンター
        self.count = 0

        #各プレイヤーの選択記録用リスト
        self.log_1 = list()
        self.log_2 = list()
        self.log_3 = list()

    def make_choice(self, n):
        #選択肢リストの作成
        self.coming_outs = list()
        if n == 1:
            a = 2
            b = 3
        elif n == 2:
            a = 1
            b = 3
        elif n == 3:
            a = 1
            b = 2
        self.coming_outs.append('私は村人です。')
        self.coming_outs.append('私は人狼です。')
        self.coming_outs.append('私は占い師です。他の枚は人狼と人狼でした。2')
        self.coming_outs.append('私は占い師です。他の枚は人狼と怪盗でした。2')
        self.coming_outs.append('私は占い師です。他の枚は人狼と村人でした。2')
        self.coming_outs.append('私は占い師です。他の枚は怪盗と村人でした。2')
        self.coming_outs.append(f'私は占い師です。Player{a}は人狼でした。')
        self.coming_outs.append(f'私は占い師です。Player{a}は人間でした。')
        self.coming_outs.append(f'私は占い師です。Player{b}は人狼でした。')
        self.coming_outs.append(f'私は占い師です。Player{b}は人間でした。')
        self.coming_outs.append(f'私は怪盗です。Player{a}と交換して村人でした。
            ')
        self.coming_outs.append(f'私は怪盗です。Player{b}と交換して村人でした。
            ')
        self.coming_outs.append(f'私は怪盗です。Player{a}と交換して占い師でした。
            ')
        self.coming_outs.append(f'私は怪盗です。Player{b}と交換して占い師でした。
            ')
```

```
        self.coming_outs.append('私は怪盗です。交換しませんでした。')
        self.coming_outs.append(f'私はPlayer{a}を信じます。')
        self.coming_outs.append(f'私はPlayer{b}を信じます。')
        self.coming_outs.append(f'私はPlayer{a}が怪しいです。')
        self.coming_outs.append(f'私はPlayer{b}が怪しいです。')
        self.coming_outs.append('平和村にしましょう。')
        self.coming_outs.append('発言しない')

    #選択肢リストを返す
    def get_coming_outs(self):
        return self.coming_outs

    #過去の発言を記録する
    def choice_log(self, log):
        if self.count % 3 == 0:
            self.log_1.append(log)
        elif self.count % 3 == 1:
            self.log_2.append(log)
        else:
            self.log_3.append(log)
        self.count += 1

    #の発言記録Player1
    def log1(self):
        return self.log_1

    #の発言記録Player2
    def log2(self):
        return self.log_2

    #の発言記録Player3
    def log3(self):
        return self.log_3
```

- vote.py

```
#!/usr/bin/env python
# coding: utf-8
```

```
# In[ ]:

# 投票に関するクラス
class Vote:
    #コンストラクタ
    def __init__(self, p):
        #プレイヤーの数だけリスト作成
        self.players = [0] * p
        #処刑されるプレイヤーのリスト
        self.punishment = list()

    #各プレイヤーに投票された数を合計する関数
    # n 投票されたプレイヤーの番号
    def total_voted(self, n):
        for m in range(3):
            if m == n-1:
                self.players[m] += 1

    #最も投票が多かったプレイヤー
    def most_voted(self):
        text = '投票の結果次のプレイヤーが\処刑されました。n\n\n'

        #投票が割れた場合
        if max(self.players) <= 1:
            return '投票の結果平和村となりました。'
        #それ以外
        else:
            for n in range(len(self.players)):
                if self.players[n] == max(self.players):
                    text += f'Player{n+1}'
                    self.punishment.append(n)
            return text

    #処刑者のリストを返す
    def punishmented_index(self):
        return self.punishment
```

- villager.py

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

import numpy as np

# 村人クラス
class Villager:

    def __init__(self, status):

        self.status = status

    # 役割
    def role(self):
        return 'あなたは村人です'

    # 能力
    def action(self):
        return '夜のアクションはありません。\話し合いで人狼を探し当ててください。n'

    # 空メソッド
    def target(self, n, players):
        pass

    #ごとの選択肢をメモPlayer
    def another_choices(self, choices, n):
        if n == 1:
            self.p1 = choices
        elif n == 2:
            self.p2 = choices
        else:
            self.p3 = choices

    #選択肢を確率で返す
    #自分の選択肢choices 各プレイヤーの発言記録log_list
```

```
#c 何回目の発言か pn 自分のプレイヤー番号
def choice_rate(self, choices, log_list, c, pn):
    if pn == 1:
        #のログPlayer2
        log1 = log_list[1]
        #のログPlayer3
        log2 = log_list[2]
        #自分のログ
        log3 = log_list[0]
    elif pn == 2:
        #のログPlayer1
        log1 = log_list[0]
        #のログPlayer3
        log2 = log_list[2]
        #自分のログ
        log3 = log_list[1]
    else:
        #のログPlayer1
        log1 = log_list[0]
        #のログPlayer2
        log2 = log_list[1]
        #自分のログ
        log3 = log_list[2]



    if c >= 2:
        if pn == 1:
            p1_choices = self.p2
            p2_choices = self.p3
        elif pn == 2:
            p1_choices = self.p1
            p2_choices = self.p3
        else:
            p1_choices = self.p1
            p2_choices = self.p2
```

```
#選択肢を選ぶ確率
#最初の発言
if c==1:
    rate_list = [0.80, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
        0.00, 0.00, 0.00,
                 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                     0.00, 0.00, 0.00, 0.20]
#回目の発言2
elif c==2:
    if log3[0] == choices[0]:
        rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
            0.00, 0.00, 0.00,
                     0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                         0.00, 0.00, 0.00, 1.00]
    else:
        rate_list = [0.90, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
            0.00, 0.00, 0.00,
                     0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                         0.00, 0.00, 0.00, 0.10]

    #全員が発言しなかった場合
    if log1[0] == log2[0] == log3[0] == choices[0]:
        rate_list[0] = 0.30
        rate_list[3] = 0.05
        rate_list[-1] = 0.55
        rate_list[14] = 0.05
        rate_list[1] = 0.05
    #村人もしくは人狼と名乗る人がいた場合
    elif log1[0] in choices[0:2] or log2[0] in choices[0:2]:
        if log3[0] == choices[0]:
            rate_list = [0.00] * 21
            if log1[0] in choices[0:2]:
                rate_list[-4] = 1.00
            else:
                rate_list[-3] = 1.00
    #占い結果が同じ場合
```

```
elif (log1[0] in p1_choices[2:6] and log2[0] in p2_choices
    [2:6]):
    rate_list[0] = 0.00
    rate_list[-1] = 0.00
    if log3[0] == choices[-1]:
        rate_list[0] = 0.10
        rate_list[15] = 0.45
        rate_list[16] = 0.45
    else:
        rate_list[15] = 0.50
        rate_list[16] = 0.50
elif pn == 1 and (log1[0] == log2[0] == p1_choices[7]):
    rate_list[0] = 0.00
    rate_list[-1] = 0.00
    if log3[0] == choices[-1]:
        rate_list[0] = 0.10
        rate_list[15] = 0.45
        rate_list[16] = 0.45
    else:
        rate_list[15] = 0.50
        rate_list[16] = 0.50
elif pn == 2 and (log1[0] == p1_choices[7] and log2[0] ==
    p2_choices[9]):
    rate_list[0] = 0.00
    rate_list[-1] = 0.00
    if log3[0] == choices[-1]:
        rate_list[0] = 0.10
        rate_list[15] = 0.45
        rate_list[16] = 0.45
    else:
        rate_list[15] = 0.50
        rate_list[16] = 0.50
elif pn == 3 and (log1[0] == log2[0] == p1_choices[9]):
    rate_list[0] = 0.00
    rate_list[-1] = 0.00
    if log3[0] == choices[-1]:
        rate_list[0] = 0.10
```

```
            rate_list[15] = 0.45
            rate_list[16] = 0.45
        else:
            rate_list[15] = 0.50
            rate_list[16] = 0.50
    #怪盗が同じ場合
    elif pn == 1 and (log1[0] == log2[0] == p1_choices[10]):
        rate_list[0] = 0.00
        rate_list[-1] = 0.00
        if log3[0] == choices[-1]:
            rate_list[0] = 0.10
            rate_list[15] = 0.45
            rate_list[16] = 0.45
        else:
            rate_list[15] = 0.50
            rate_list[16] = 0.50
    elif pn == 2 and (log1[0] == p1_choices[10] and log2[0] ==
        p2_choices[11]):
        rate_list[0] = 0.00
        rate_list[-1] = 0.00
        if log3[0] == choices[-1]:
            rate_list[0] = 0.10
            rate_list[15] = 0.45
            rate_list[16] = 0.45
        else:
            rate_list[15] = 0.50
            rate_list[16] = 0.50
    elif pn == 3 and (log1[0] == log2[0] == p1_choices[11]):
        rate_list[0] = 0.00
        rate_list[-1] = 0.00
        if log3[0] == choices[-1]:
            rate_list[0] = 0.10
            rate_list[15] = 0.45
            rate_list[16] = 0.45
        else:
            rate_list[15] = 0.50
            rate_list[16] = 0.50
```

```
#平和村の処理
#占い師が他の役職を共に人狼と言っている且つ怪盗が自分と交換して村人と言っている
elif pn == 1 and (log1[0] == choices[2] and log2[0] == choices[10]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 2 or pn == 3) and (log1[0] == p1_choices[2] and log2[0] == p2_choices[11]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 1 or pn == 2) and (log2[0] == p2_choices[2] and log1[0] == p1_choices[10]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif pn == 3 and (log2[0] == p2_choices[2] and log1[0] == p1_choices[11]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
#怪盗が占い師と交換している　占いが他のまい人狼2
elif pn == 1 and (log2[0] == p2_choices[13] and log1[0] == p1_choices[2]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 3 or pn == 2) and (log2[0] == p2_choices[12] and log1[0] in p1_choices[2]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 1 or pn == 2) and (log1[0] == p1_choices[13] and log2[0] == p2_choices[2]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif pn == 3 and (log1[0] == p1_choices[12] and log2[0] == p2_choices[2]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
#怪盗が占い師と交換している　占いがプレイヤー
```

```
elif pn == 1 and (log2[0] == p2_choices[13] and log1[0] ==
    p1_choices[7]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif pn == 1 and (log2[0] == p2_choices[13] and log1[0] ==
    p1_choices[9]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 3 or pn == 2) and (log2[0] == p2_choices[12]
   and log1[0] in p1_choices[7]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 3 or pn == 2) and (log2[0] == p2_choices[12]
   and log1[0] in p1_choices[9]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 1 or pn == 2) and (log1[0] == p1_choices[13]
   and log2[0] == p2_choices[7]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 1 or pn == 2) and (log1[0] == p1_choices[13]
   and log2[0] == p2_choices[9]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif pn == 3 and (log1[0] == p1_choices[12] and log2[0] ==
    p2_choices[7]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif pn == 3 and (log1[0] == p1_choices[12] and log2[0] ==
    p2_choices[9]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
#怪盗が自分と交換し、占い師がプレイヤーを占っている場合
elif pn == 1 and (log2[0] == p2_choices[10] and log1[0] ==
    p1_choices[7]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
```

```
elif pn == 1 and (log2[0] == p2_choices[10] and log1[0] ==
    p1_choices[9]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 3 or pn == 2) and (log2[0] == p2_choices[11]
   and log1[0] in p1_choices[7]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 3 or pn == 2) and (log2[0] == p2_choices[11]
   and log1[0] in p1_choices[9]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 1 or pn == 2) and (log1[0] == p1_choices[10]
   and log2[0] == p2_choices[7]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif (pn == 1 or pn == 2) and (log1[0] == p1_choices[10]
   and log2[0] == p2_choices[9]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif pn == 3 and (log1[0] == p1_choices[11] and log2[0] ==
    p2_choices[7]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
elif pn == 3 and (log1[0] == p1_choices[11] and log2[0] ==
    p2_choices[9]):
    rate_list = [0.00] * 21
    rate_list[-2] = 1.00
#怪盗が自分を占い師という
elif pn == 1 and (log1[0] == p1_choices[12] or log2[0] ==
   p2_choices[12]):
    rate_list = [0.00] * 21
    if log1[0] == p1_choices[12] and log2[0] == p2_choices
        [12]:
        rate_list[-3] = 0.50
        rate_list[-4] = 0.50
    elif log1[0] == p1_choices[12]:
```

```
            rate_list[-4] = 1.00
        else:
            rate_list[-3] = 1.00
    elif pn == 2 and (log1[0] == p1_choices[12] or log2[0] ==
        p2_choices[13]):
        rate_list = [0.00] * 21
        if log1[0] == p1_choices[12] and log2[0] == p2_choices
            [13]:
            rate_list[-3] = 0.50
            rate_list[-4] = 0.50
        elif log1[0] == p1_choices[12]:
            rate_list[-4] = 1.00
        else:
            rate_list[-3] = 1.00
    elif pn == 3 and (log1[0] == p1_choices[13] or log2[0] ==
        p2_choices[13]):
        rate_list = [0.00] * 21
        if log1[0] == p1_choices[13] and log2[0] == p2_choices
            [13]:
            rate_list[-3] = 0.50
            rate_list[-4] = 0.50
        elif log1[0] == p1_choices[13]:
            rate_list[-4] = 1.00
        else:
            rate_list[-3] = 1.00
    #誰かが占い師の場合
    elif log1[0] in p1_choices[3:10]:
        rate_list = [0.00] * 21
        #占い師が他の枚に村人がいると言った場合2
        if log1[0] in choices[4:6]:
            if log3[0] == choices[0]:
                rate_list[-4] = 1.00
            else:
                rate_list[0] = 0.50
                rate_list[-4] = 0.50
        elif log2[0] in choices[4:6]:
            if log3[0] == choices[0]:
```

```
                rate_list[-4] = 1.00
            else:
                rate_list[0] = 0.50
                rate_list[-3] = 0.50
        #占い結果が自分を人狼と言った場合
        elif (pn == 3 and log1[0] == p1_choices[8]) or (pn ==
            1 or pn == 2 and log1[0] == p1_choices[6]):
            if log3[0] == choices[0]:
                rate_list[-4] = 1.00
            else:
                rate_list[0] = 0.50
                rate_list[-4] = 0.50
        #占い結果が自分以外を人狼と言った場合
        elif log1[0] == p1_choices[6] or log1[0] == p1_choices
            [8]:
            if log3[0] == choices[0]:
                rate_list[-3] = 1.00
            else:
                rate_list[0] = 0.50
                rate_list[-3] = 0.50
        else:
            if log3[0] == choices[0]:
                rate_list[-1] = 1.00
            else:
                rate_list[0] = 0.50
                rate_list[-1] = 0.50
    #誰かが占い師の場合
    elif log2[0] in p2_choices[3:10]:
        rate_list = [0.00] * 21
        #占い師が他の枚に村人がいると言った場合2
        if log1[0] in choices[4:6]:
            if log3[0] == choices[0]:
                rate_list[-4] = 1.00
            else:
                rate_list[0] = 0.50
                rate_list[-4] = 0.50
        elif log2[0] in choices[4:6]:
```

```
            if log3[0] == choices[0]:
                rate_list[-4] = 1.00
            else:
                rate_list[0] = 0.50
                rate_list[-3] = 0.50
        #占い結果が自分を人狼と言った場合
        elif ((pn == 3 or pn == 2) and log2[0] == p2_choices
            [8]) or (pn == 1 and log2[0] == p2_choices[6]):
            if log3[0] == choices[0]:
                rate_list[-3] = 1.00
            else:
                rate_list[0] = 0.50
                rate_list[-3] = 0.50
        #占い結果が自分以外を人狼と言った場合
        elif log1[0] == p1_choices[6] or log1[0] == p1_choices
            [8]:
            if log3[0] == choices[0]:
                rate_list[-4] = 1.00
            else:
                rate_list[0] = 0.50
                rate_list[-4] = 0.50
        else:
            if log3[0] == choices[0]:
                rate_list[-1] = 1.00
            else:
                rate_list[0] = 0.50
                rate_list[-1] = 0.50
    #怪盗が自分の役職と交換した場合
    elif pn == 1 and (log1[0] == p1_choices[10] or log2[0] ==
        p2_choices[10]):
        rate_list = [0.00] * 21
        if log1[0] == p1_choices[10]:
            rate_list[-6] = 1.00
        else:
            rate_list[-5] = 1.00
    elif pn == 2 and log1[0] == p1_choices[10]:
        rate_list = [0.00] * 21
```

```
            rate_list[-6] = 1.00
        elif pn == 2 and log2[0] == p2_choices[11]:
            rate_list = [0.00] * 21
            rate_list[-5] = 1.00
        elif pn == 3 and (log1[0] == p1_choices[11] or log2[0] ==
            p2_choices[11]):
            rate_list = [0.00] * 21
            if log1[0] == p1_choices[11]:
                rate_list[-6] = 1.00
            else:
                rate_list[-5] = 1.00
    #回目の発言3
    else:
        #過去に村人宣言していない場合
        if not (choices[0] in log3[0:2]):
            rate_list = [1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                0.00, 0.00, 0.00,
                        0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 0.00]
        else:
            rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                0.00, 0.00, 0.00,
                        0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 1.00]
        #一度も発言していないプレイヤーがいる場合
        if choices[-1] == log1[0] and choices[-1] == log1[1]:
            rate_list = [0.00] * 21
            rate_list[-4] = 1.00
            if ((pn == 3 or np == 2) and log2[0] == p2_choices[7])
                or (pn == 1 and log2[0] == p2_choices[9]):
                rate_list[-4] = 0.00
                rate_list[-3] = 1.00
        elif choices[-1] == log2[0] and choices[-1] == log2[1]:
            rate_list = [0.00] * 21
            rate_list[-3] = 1.00
            if (pn == 3 and log1[0] == p1_choices[7]) or (pn == 1
                or pn == 2 and log1[0] == p1_choices[9]):
```

```
            rate_list[-3] = 0.00
            rate_list[-4] = 1.00
    #平和村にすると言っていた場合
    elif log3[1] == choices[-2]:
        rate_list = [0.00] * 21
        rate_list[-2] = 1.00
    #怪盗が交換した占い師が出てきた場合
    elif (pn == 3 and (log1[0] == p1_choices[12] and log2[1]
        in p2_choices[2:10])) or\
    ((pn == 3 or pn == 2) and (log2[0] == p2_choices[12] and
        log1[1] in p1_choices[2:10])) or\
    ((pn == 1 or pn == 2) and (log1[0] == p1_choices[13] and
        log2[1] in p2_choices[2:10])) or\
    (pn == 1 and (log2[0] == p2_choices[13] and log1[1] in
        p1_choices[2:10])):
        if log2[1] == choices[2] or log2[1] == p2_choices[7]
            or log2[1] == p2_choices[9]:
            rate_list = [0.00] * 21
            rate_list[-2] = 1.00
        elif log1[1] == choices[2] or log1[1] == p1_choices[7]
            or log1[1] == p1_choices[9]:
            rate_list = [0.00] * 21
            rate_list[-2] = 1.00
        else:
            rate_list = [0.00] * 21
            rate_list[-3] = 0.50
            rate_list[-4] = 0.50
    #占い師が名乗り出た場合
    elif log1[1] in p1_choices[2:10] or log2[1] in p1_choices
        [2:10]:
        #村人と言っていた場合
        if not (choices[0] in log3[0:2]):
            rate_list[0] = 0.00
            rate_list[-1] = 1.00
            #占い結果が自分を人狼と言った場合
            if (pn == 3 and log1[1] == p1_choices[8]) or (pn
                == 1 or pn == 2 and log1[1] == p1_choices[6]):
```

```
        rate_list[-1] = 0.00
        rate_list[-4] = 1.00
    elif ((pn == 3 or pn == 2) and log2[1] ==
        p2_choices[8]) or (pn == 1 and log2[1] ==
        p2_choices[6]):
        rate_list[-1] = 0.00
        rate_list[-3] = 1.00
    #占い師が他の枚に村人がいると言った場合2
    elif log1[1] in choices[4:6]:
        rate_list[-1] = 0.00
        rate_list[-4] = 1.00
    elif log2[1] in choices[4:6]:
        rate_list[-1] = 0.00
        rate_list[-3] = 1.00
    #占い師が被っている場合
    elif log1[1] in p1_choices[2:10] and log2[1] in
        p2_choices[2:10]:
        rate_list[-1] = 0.00
        rate_list[-3] = 0.25
        rate_list[-4] = 0.25
        rate_list[-5] = 0.25
        rate_list[-6] = 0.25
    elif log1[0] in p1_choices[2:10] and log2[1] in
        p2_choices[2:10]:
        if log1[0] in choices[4:6]:
            rate_list[-5] = 1.00
            rate_list[-1] = 0.00
        elif (pn == 3 and log1[1] == p1_choices[8]) or
            (pn == 1 or pn == 2 and log1[1] ==
            p1_choices[6]):
            rate_list[-5] = 1.00
            rate_list[-1] = 0.00
        else:
            rate_list[-3] = 1.00
            rate_list[-1] = 0.00
    elif log2[0] in p2_choices[2:10] and log2[1] in
        p2_choices[2:10]:
```

```
            if log2[0] in choices[4:5]:
                rate_list[-6] = 1.00
                rate_list[-1] = 0.00
            elif ((pn == 3 or pn == 2) and log2[1] ==
                p2_choices[8]) or (pn == 1 and log2[1] ==
                p2_choices[6]):
                rate_list[-6] = 1.00
                rate_list[-1] = 0.00
            else:
                rate_list[-4] = 1.00
                rate_list[-1] = 0.00
        else:
            if log1[1] in p1_choices[2:10]:
                rate_list[-1] = 0.80
                rate_list[-6] = 0.20
            else:
                rate_list[-1] = 0.80
                rate_list[-5] = 0.20
    #村人と言ってなかった場合
    else:
        rate_list[0] = 0.70
        rate_list[-1] = 0.30
        #占い結果が自分を人狼と言った場合
        if (pn == 3 and log1[1] == p1_choices[8]) or (pn
            == 1 or pn == 2 and log1[1] == p1_choices[6]):
            rate_list[0] = 0.50
            rate_list[-1] = 0.00
            rate_list[-4] = 0.50
        elif ((pn == 3 or pn == 2) and log2[1] ==
            p2_choices[8]) or (pn == 1 and log2[1] ==
            p2_choices[6]):
            rate_list[0] = 0.50
            rate_list[-1] = 0.00
            rate_list[-3] = 0.50
        #占い師が他の枚に村人がいると言った場合2
        elif log1[1] in choices[4:6]:
            rate_list[-1] = 0.00
```

```
        rate_list[0] = 0.50
        rate_list[-4] = 0.50
    elif log2[1] in choices[4:6]:
        rate_list[-1] = 0.00
        rate_list[0] = 0.50
        rate_list[-3] = 0.50
    #占い師が被っている場合
    elif log1[1] in p1_choices[2:10] and log2[1] in
        p2_choices[2:10]:
        rate_list[-1] = 0.00
        rate_list[0] = 0.60
        rate_list[-3] = 0.10
        rate_list[-4] = 0.10
        rate_list[-5] = 0.10
        rate_list[-6] = 0.10
    elif log1[0] in p1_choices[2:10] and log2[1] in
        p2_choices[2:10]:
        if log1[1] in choices[4:6]:
            rate_list[0] = 0.50
            rate_list[-5] = 0.50
            rate_list[-1] = 0.00
        elif (pn == 3 and log1[1] == p1_choices[8]) or
            (pn == 1 or pn == 2 and log1[1] ==
           p1_choices[6]):
            rate_list[0] = 0.50
            rate_list[-5] = 0.50
            rate_list[-1] = 0.00
        else:
            rate_list[0] = 0.50
            rate_list[-3] = 0.50
            rate_list[-1] = 0.00
    elif log2[0] in p2_choices[2:10] and log1[1] in
        p1_choices[2:10]:
        if log2[0] in choices[4:6]:
            rate_list[0] = 0.50
            rate_list[-6] = 0.50
            rate_list[-1] = 0.00
```

```
            elif ((pn == 3 or pn == 2) and log2[1] ==
                p2_choices[8]) or (pn == 1 and log2[1] ==
                p2_choices[6]):
                rate_list[0] = 0.50
                rate_list[-6] = 0.50
                rate_list[-1] = 0.00
            else:
                rate_list[0] = 0.50
                rate_list[-4] = 0.50
                rate_list[-1] = 0.00
        else:
            if log1[1] in p1_choices[2:10]:
                rate_list[0] = 0.70
                rate_list[-1] = 0.10
                rate_list[-6] = 0.20
            else:
                rate_list[0] = 0.70
                rate_list[-1] = 0.10
                rate_list[-5] = 0.20
#村人か人狼が名乗り出た場合
elif log1[1] in choices[0:2] or log2[1] in choices[0:2]:
    if choices[0] in log3[0:1]:
        rate_list = [0.00] * 21
        if log1[1] in choices[0:2]:
            rate_list[-4] = 1.00
        else:
            rate_list[-3] = 1.00
#怪盗が自分の役職と交換した場合
elif pn == 1 and (log1[1] == p1_choices[10] or log2[1] ==
    p2_choices[10]):
    rate_list = [0.00] * 21
    if log1[1] == p1_choices[10]:
        rate_list[-6] = 1.00
    else:
        rate_list[-5] = 1.00
elif pn == 2 and log1[1] == p1_choices[10]:
    rate_list = [0.00] * 21
```

```
                rate_list[-6] = 1.00
            elif pn == 2 and log2[1] == p2_choices[11]:
                rate_list = [0.00] * 21
                rate_list[-5] = 1.00
            elif pn == 3 and (log1[1] == p1_choices[11] or log2[1] ==
                p2_choices[11]):
                rate_list = [0.00] * 21
                if log1[1] == p1_choices[11]:
                    rate_list[-6] = 1.00
                else:
                    rate_list[-5] = 1.00
            #後出しで怪盗が占い師をとったと言った場合
            elif log1[0] in p1_choices[2:10] and log2[1] in p2_choices
                [12:14]:
                if not (choices[0] in log3[0:2]):
                    rate_list[0] = 0.50
                    rate_list[-3] = 0.50
                    rate_list[-1] = 0.00
                else:
                    rate_list[0] = 0.00
                    rate_list[-3] = 1.00
                    rate_list[-1] = 0.00
            elif log2[0] in p2_choices[2:10] and log1[1] in p1_choices
                [12:14]:
                if not (choices[0] in log3[0:2]):
                    rate_list[0] = 0.50
                    rate_list[-4] = 0.50
                    rate_list[-1] = 0.00
                else:
                    rate_list[0] = 0.00
                    rate_list[-4] = 1.00
                    rate_list[-1] = 0.00
        return np.random.choice(choices, p=rate_list)

    #投票先を決める
    def doubt(self, pn, choices, log_list):
        log1 = list()
```

```
log2 = list()
log3 = list()

#のログPlayer1
log1 = log_list[0]
#のログPlayer2
log2 = log_list[1]
#のログPlayer3
log3 = log_list[2]

#平和村を願う場合
if log1[1] == log2[1] == log3[1] == choices[-2]:
    return 0
if len(log1) == 3:
    if log1[2] == log2[2] == log3[2] == choices[-2]:
        return 0
#プレイヤーの場合1
if pn == 1:
    #疑わしさ
    p2_level = 0
    p3_level = 0

    #人狼、村人と名乗るプレイヤーは怪しい
    for n in range(3):
        if log2[n] in self.p2[0:2]:
            p2_level += 100
        if log3[n] in self.p3[0:2]:
            p3_level += 100
    #占い結果がおかしいプレイヤーは怪しい
    for n in range(3):
        #占い結果に村人がいる、自分を人狼と占っている
        if log2[n] in self.p2[4:7]:
            p2_level += 100
        if log3[n] in self.p3[4:7]:
            p3_level += 100
    #怪盗がおかしい
    for n in range(3):
```

```
        #他のプレイヤーを村人と言っている、自分を村人以外と言っている
        if log2[n] in self.p2[11:13]:
            p2_level += 200
        if log3[n] in self.p3[11:13]:
            p3_level += 200
    #一度も発言しないのは怪しい
    if log2[0] == log2[1] == log2[2] == choices[-1]:
        p2_level += 50
    if log3[0] == log3[1] == log3[2] == choices[-1]:
        p3_level += 50
    #占い師だと後出しは怪しい
    for n in range(2):
        if log2[n] == self.p2[2:10] and log3[n+1] == self.p3
            [2:10]:
            p3_level += 30
        if log3[n] == self.p3[2:10] and log2[n+1] == self.p2
            [2:10]:
            p2_level += 30
    if log2[0] == log3[2]:
        p3_level += 30
    if log3[0] == log2[2]:
        p2_level += 30
    #発言しないほど疑わしい
    for n in range(3):
        if log2[n] == choices[0]:
            p2_level += 10
        if log3[n] == choices[0]:
            p3_level += 10
    #怪盗が自分と交換して村人と言っていた
    if log2[0] == self.p2[10]:
        p2_level -= 50
    if log3[0] == self.p3[10]:
        p3_level -= 50
    #占い師が他のプレイヤーを人狼と言っていた場合例外
    if log2[0] == self.p2[8] or log3[0] == self.p3[8]:
        if log2[0] == self.p2[8] and log3[0] == self.p3[8]:
            return np.random.choice([2,3])
```

```
            elif log2[0] == self.p2[8]:
                return np.random.choice([2,3], p=[0.70, 0.30])
            elif log3[0] == self.p3[8]:
                return np.random.choice([2,3], p=[0.30, 0.70])
        #投票先を決定
        if p2_level == p3_level:
            return np.random.choice([2,3])
        elif p2_level > p3_level:
            return 2
        else:
            return 3
    #プレイヤーの場合2
    elif pn == 2:
        #疑わしさ
        p1_level = 0
        p3_level = 0

        #人狼、村人と名乗るプレイヤーは怪しい
        for n in range(3):
            if log1[n] in self.p1[0:2]:
                p1_level += 100
            if log3[n] in self.p3[0:2]:
                p3_level += 100
        #占い結果がおかしいプレイヤーは怪しい
        for n in range(3):
            #占い結果に村人がいる、自分を人狼と占っている
            if log1[n] in self.p1[4:7]:
                p1_level += 100
            if log3[n] in self.p3[4:7] or log3[n] in self.p3[8]:
                p3_level += 100
        #怪盗がおかしいのは怪しい
        for n in range(3):
            #他のプレイヤーを村人と言っている、自分を村人以外と言っている
            if log1[n] in self.p1[11:13]:
                p1_level += 200
            if log3[n] == self.p3[10] or log3[n] == self.p3[13]:
                p3_level += 200
```

```
#一度も発言しないのは怪しい
if log1[0] == log1[1] == log1[2] == choices[-1]:
    p1_level += 50
if log3[0] == log3[1] == log3[2] == choices[-1]:
    p3_level += 50
#占い師だ、と後出しは怪しい
for n in range(2):
    if log1[n] == self.p1[2:10] and log3[n+1] == self.p3
        [2:10]:
        p3_level += 20
    if log3[n] == self.p3[2:10] and log1[n+1] == self.p1
        [2:10]:
        p1_level += 20
if log1[0] == log3[2]:
    p3_level += 20
if log3[0] == log1[2]:
    p1_level += 20
#発言しないほど疑わしい
for n in range(3):
    if log1[n] == choices[0]:
        p1_level += 10
    if log3[n] == choices[0]:
        p3_level += 10
#怪盗が自分と交換して村人と言っていた
if log1[0] == self.p1[10]:
    p1_level -= 50
if log3[0] == self.p3[11]:
    p3_level -= 50
#占い師が他のプレイヤーを人狼と言っていた場合例外
if log1[0] == self.p1[8] or log3[0] == self.p3[6]:
    if log1[0] == self.p1[8] and log3[0] == self.p3[8]:
        return np.random.choice([1,3])
    elif log1[0] == self.p1[8]:
        return np.random.choice([1,3], p=[0.70, 0.30])
    elif log3[0] == self.p3[6]:
        return np.random.choice([1,3], p=[0.30, 0.70])
#投票先を決定
```

```
        if p1_level == p3_level:
            return np.random.choice([1,3])
        elif p1_level > p3_level:
            return 1
        else:
            return 3
    #プレイヤーの場合3
    else:
        #疑わしさ
        p1_level = 0
        p2_level = 0

        #人狼、村人と名乗るプレイヤーは怪しい
        for n in range(3):
            if log1[n] in self.p1[0:2]:
                p1_level += 100
            if log2[n] in self.p2[0:2]:
                p2_level += 100
        #占い結果がおかしいプレイヤーは怪しい
        for n in range(3):
            #占い結果に村人がいる、自分を人狼と占っている
            if log1[n] in self.p1[4:6] or log1[n] in self.p1[8]:
                p1_level += 100
            if log2[n] in self.p2[4:6] or log2[n] in self.p2[8]:
                p2_level += 100
        #怪盗がおかしいのは怪しい
        for n in range(3):
            #他のプレイヤーを村人と言っている、自分を村人以外と言っている
            if log1[n] == self.p1[10] or log1[n] == self.p1[13]:
                p1_level += 200
            if log2[n] == self.p2[10] or log2[n] == self.p2[13]:
                p2_level += 200
        #一度も発言しないのは怪しい
        if log1[0] == log1[1] == log1[2] == choices[-1]:
            p1_level += 50
        if log2[0] == log2[1] == log2[2] == choices[-1]:
            p2_level += 50
```

```
#占い師だ、と後出しは怪しい
for n in range(2):
    if log1[n] == self.p1[2:10] and log2[n+1] == self.p2
        [2:10]:
        p2_level += 20
    if log2[n] == self.p2[2:10] and log1[n+1] == self.p1
        [2:10]:
        p1_level += 20
if log1[0] == log2[2]:
    p2_level += 20
if log2[0] == log1[2]:
    p1_level += 20
#発言しないほど疑わしい
for n in range(3):
    if log1[n] == choices[0]:
        p1_level += 10
    if log2[n] == choices[0]:
        p2_level += 10
#怪盗が自分と交換して村人と言っていた
if log1[0] == self.p1[11]:
    p1_level -= 50
if log2[0] == self.p2[11]:
    p2_level -= 50
#占い師が他のプレイヤーを人狼と言っていた場合例外
if log1[0] == self.p1[6] or log2[0] == self.p2[6]:
    if log1[0] == self.p1[6] and log2[0] == self.p2[6]:
        return np.random.choice([1,2])
    elif log1[0] == self.p1[6]:
        return np.random.choice([1,2], p=[0.70, 0.30])
    elif log2[0] == self.p2[6]:
        return np.random.choice([1,2], p=[0.30, 0.70])
#投票先を決定
if p1_level == p2_level:
    return np.random.choice([1,2])
elif p1_level > p2_level:
    return 1
else:
```

```
                return 2
```

- wolf.py

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

import numpy as np

# 人狼に関するクラス
class Wolf:

    # コンストラクタ
    def __init__(self, status):
        self.status = status

    # 役割
    def role(self):
        return 'あなたは人狼です。\仲間を確認し市民側を欺いてください。n\人狼が人も処
            刑されなければ勝利です。n1'

    # 能力
    def action(self, players, count):
        is_wolf = False
        for n in range(len(players)-2):
            if n == count-1:
                continue
            if players[n].status == '人狼':
                player = n+1
                is_wolf = True
                break
        if is_wolf:
            return f'仲間はPlayer{player}です}'
        else:
            return '仲間はいませんでした。'

    #人狼の仲間の有無
```

```
    def target(self, n, players):
        #人狼の仲間がいない
        self.wolf_team = 0
        for m in range(len(players)-2):
            if m == n-1:
                continue
            if players[m].status == '人狼':
                #プレイヤーnが仲間+1
                self.wolf_team = m+1
                break

    #ごとの選択肢をメモPlayer
    def another_choices(self, choices, n):
        if n == 1:
            self.p1 = choices
        elif n == 2:
            self.p2 = choices
        else:
            self.p3 = choices

# 一度も発言しない戦略     #
#     def choice_rate(self, choices, log_list, c, pn):
#         rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
    0.00, 0.00,
#                      0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
    0.00, 0.00, 1.00]
#         return np.random.choice(choices, p=rate_list)

    #選択肢を確率で返す
    #自分の選択肢choices 各プレイヤーの発言記録log_list
    #c 何回目の発言か pn 自分のプレイヤー番号
    def choice_rate(self, choices, log_list, c, pn):
        if pn == 1:
            #のログPlayer2
            log1 = log_list[1]
            #のログPlayer3
            log2 = log_list[2]
```

```
        #自分のログ
        log3 = log_list[0]
    elif pn == 2:
        #のログPlayer1
        log1 = log_list[0]
        #のログPlayer3
        log2 = log_list[2]
        #自分のログ
        log3 = log_list[1]
    else:
        #のログPlayer1
        log1 = log_list[0]
        #のログPlayer2
        log2 = log_list[1]
        #自分のログ
        log3 = log_list[2]

    #各プレイヤーごとの選択肢
    if c >= 2:
        if pn == 1:
            p1_choices = self.p2
            p2_choices = self.p3
        elif pn == 2:
            p1_choices = self.p1
            p2_choices = self.p3
        else:
            p1_choices = self.p1
            p2_choices = self.p2

    #選択肢を選ぶ確率
    #最初の発言
    if c == 1:
        if self.wolf_team == 0:
            #選択肢を選ぶ確率
            rate_list = [0.48, 0.00, 0.02, 0.06, 0.03, 0.00, 0.00,
                0.02, 0.00, 0.02,
                        0.00, 0.00, 0.06, 0.06, 0.00, 0.00, 0.00,
```

```
                                    0.00, 0.00, 0.00, 0.25]
#                   rate_list = [1.00, 0.00, 0.00, 0.00, 0.00, 0.00,
    0.00, 0.00, 0.00, 0.00,
#                                0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
    0.00, 0.00, 0.00, 0.00]
            else:
                rate_list = [0.23, 0.05, 0.07, 0.07, 0.07, 0.07, 0.07,
                    0.07, 0.07, 0.07,
                                0.04, 0.04, 0.04, 0.04, 0.00, 0.00, 0.00,
                                    0.00, 0.00, 0.00, 0.00]
        #回目の発言2
        elif c == 2:
            #仲間がいた場合
            if self.wolf_team != 0:
                rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                    0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                                0.00, 0.00, 0.00, 1.00]
                #発言していなかった場合
                if log3[0] == choices[-1]:
                    rate_list = [0.10, 0.01, 0.04, 0.04, 0.04, 0.04,
                        0.04, 0.04, 0.04, 0.04,
                                    0.03, 0.03, 0.03, 0.03, 0.00, 0.00,
                                        0.00, 0.00, 0.00, 0.00, 0.45]
            #仲間がいなかった場合
            else:
                rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                    0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                                0.00, 0.00, 0.00, 1.00]
                #発言していなかった場合
                if log3[0] == choices[-1]:
                    rate_list = [0.62, 0.01, 0.00, 0.03, 0.03, 0.00,
                        0.00, 0.00, 0.00, 0.00,
                                    0.02, 0.02, 0.02, 0.02, 0.00, 0.00,
                                        0.00, 0.00, 0.00, 0.00, 0.23]
                #発言していた場合
```

```
#村人と言っていた場合
elif log3[0] == choices[0]:
    #村人もしくは人狼と名乗る人がいた場合
    if log1[0] in choices[0:1] or log2[0] in choices
        [0:2]:
        rate_list = [0.00] * 21
        if log1[0] in choices[0:2]:
            rate_list[-4] = 1.00
        else:
            rate_list[-3] = 1.00
    #怪盗が占い師と交換している　占いがプレイヤー
    elif pn == 1 and (log2[0] == p2_choices[13] and
        log1[0] == p1_choices[9]):
        rate_list = [0.00] * 21
        rate_list[-2] = 1.00
    elif (pn == 3 or pn == 2) and (log2[0] ==
        p2_choices[12] and log1[0] in p1_choices[7]):
        rate_list = [0.00] * 21
        rate_list[-2] = 1.00
    elif (pn == 1 or pn == 2) and (log1[0] ==
        p1_choices[13] and log2[0] == p2_choices[9]):
        rate_list = [0.00] * 21
        rate_list[-2] = 1.00
    elif pn == 3 and (log1[0] == p1_choices[12] and
        log2[0] == p2_choices[7]):
        rate_list = [0.00] * 21
        rate_list[-2] = 1.00
    #誰かが占い師の場合
    elif log1[0] in p1_choices[2:10]:
        #占い師が他の枚に村人がいると言った場合2
        if log1[0] in p1_choices[4:6]:
            rate_list = [0.00] * 21
            rate_list[-4] = 1.00
        #占い結果が自分を人狼と言った場合
        elif (pn == 3 and log1[0] == p1_choices[8]) or
            (pn == 1 or pn == 2 and log1[0] ==
            p1_choices[6]):
```

```
                rate_list = [0.00] * 21
                rate_list[-4] = 1.00
            #占い結果が自分以外を人狼と言った場合
            elif log1[0] == p1_choices[6] or log1[0] ==
                p1_choices[8]:
                rate_list = [0.00] * 21
                rate_list[-3] = 1.00
        #誰かが占い師の場合
        elif log2[0] in p2_choices[2:10]:
            #占い師が他の枚に村人がいると言った場合2
            if log1[0] in p1_choices[4:6]:
                rate_list = [0.00] * 21
                rate_list[-4] = 1.00
            #占い結果が自分を人狼と言った場合
            elif ((pn == 3 or pn == 2) and log2[0] ==
                p2_choices[8]) or (pn == 1 and log2[0] ==
                p2_choices[6]):
                rate_list = [0.00] * 21
                rate_list[-3] = 1.00
            #占い結果が自分以外を人狼と言った場合
            elif log1[0] == p1_choices[6] or log1[0] ==
                p1_choices[8]:
                rate_list = [0.00] * 21
                rate_list[0] = 0.50
                rate_list[-4] = 0.50
            else:
                rate_list = [0.00] * 21
                rate_list[0] = 0.50
                rate_list[-1] = 0.50
    #占い師と言っていた場合
    elif log3[0] in choices[2:10]:
        if log1[0] in p1_choices[2:10]:
            rate_list = [0.00] * 21
            rate_list[-4] = 1.00
        elif log2[0] in p2_choices[2:10]:
            rate_list = [0.00] * 21
            rate_list[-3] = 1.00
```

```
            #怪盗と言っていた場合
            elif log3[0] in choices[10:14]:
                if log1[0] in p1_choices[10:14]:
                    rate_list = [0.00] * 21
                    rate_list[-4] = 1.00
                elif log2[0] in p2_choices[10:14]:
                    rate_list = [0.00] * 21
                    rate_list[-3] = 1.00
        #4回目の発言3
        else:
            rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                0.00, 0.00, 0.00,
                        0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 1.00]
            #村人と発言していて村人か人狼が名乗り出た場合
            if log1[1] in choices[0:2] or log2[1] in choices[0:2]:
                if choices[0] in log3[0:1]:
                    rate_list = [0.00] * 21
                    if log1[1] in choices[0:1]:
                        rate_list[-4] = 1.00
                    else:
                        rate_list[-3] = 1.00
            #一度でも発言があった場合
            elif log3[0] != choices[-1] or log3[1] != choices[-1]:
                rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                    0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                                0.00, 0.00, 0.00, 1.00]
            else:
                rate_list = [0.38, 0.01, 0.03, 0.03, 0.03, 0.03, 0.03,
                    0.03, 0.03, 0.03,
                            0.03, 0.03, 0.03, 0.03, 0.00, 0.00, 0.00,
                                0.00, 0.00, 0.00, 0.25]

        return np.random.choice(choices, p=rate_list)

    #投票先を決める
```

```
def doubt(self, pn, choices, log_list):
    log1 = list()
    log2 = list()
    log3 = list()

    if pn == 1:
        #のログPlayer2
        log2 = log_list[1]
        #のログPlayer3
        log3 = log_list[2]
    elif pn == 2:
        #のログPlayer1
        log1 = log_list[0]
        #のログPlayer3
        log3 = log_list[2]
    else:
        #のログPlayer1
        log1 = log_list[0]
        #のログPlayer2
        log2 = log_list[1]

    #自分がプレイヤー1
    if pn == 1:
        if self.wolf_team == 0:
            if self.p2[12] in log2 or self.p2[13] in log2:
                return 2
            elif self.p3[12] in log3 or self.p3[13] in log3:
                return 3

            if self.p2[6] in log2 or self.p3[6] in log3:
                if self.p2[6] in log2 and self.p3[6] in log3:
                    return np.random.choice([2,3])
                elif self.p2[6] in log2:
                    return 2
                elif self.p3[6] in log3:
                    return 3
            elif choices[4] in log2 or choices[4] in log3:
```

```
                if choices[4] in log2 and choices[4] in log3:
                    return np.random.choice([2,3])
                elif choices[4] in log2:
                    return 3
                elif choices[4] in log3:
                    return 2
            elif choices[3] in log2 or choices[3] in log3:
                if choices[3] in log2 and choices[3] in log3:
                    return np.random.choice([2,3])
                elif choices[3] in log2:
                    return 2
                elif choices[3] in log3:
                    return 3
            else:
                return np.random.choice([2,3])
        elif self.wolf_team == 2:
            if log3[0] in self.p3[6:15] or log3[1] in self.p3
               [6:15] or log3[2] in self.p3[6:15]:
                return np.random.choice([2,3], p=[0.20, 0.80])
            else:
                return 3
        elif self.wolf_team == 3:
            if log2[0] in self.p2[6:15] or log2[1] in self.p2
               [6:15] or log2[2] in self.p2[6:15]:
                return np.random.choice([2,3], p=[0.80, 0.20])
            else:
                return 2
    #自分がプレイヤー2
    elif pn == 2:
        if self.wolf_team == 0:
            if self.p1[12] in log1 or self.p1[13] in log1:
                return 1
            elif self.p3[12] in log3 or self.p3[13] in log3:
                return 3

            if self.p1[6] in log1 or self.p3[6] in log3:
                if self.p1[6] in log1 and self.p3[6] in log3:
```

```
                return np.random.choice([1,3])
            elif self.p1[6] in log1:
                return 1
            elif self.p3[6] in log3:
                return 3
        elif choices[4] in log1 or choices[4] in log3:
            if choices[4] in log1 and choices[4] in log3:
                return np.random.choice([1,3])
            elif choices[4] in log1:
                return 3
            elif choices[4] in log3:
                return 1
        elif choices[3] in log1 or choices[3] in log3:
            if choices[3] in log1 and choices[3] in log3:
                return np.random.choice([1,3])
            elif choices[3] in log1:
                return 1
            elif choices[3] in log3:
                return 3
        else:
            return np.random.choice([1,3])
    elif self.wolf_team == 1:
        if log3[0] in self.p3[6:15] or log3[1] in self.p3
           [6:15] or log3[2] in self.p3[6:15]:
            return np.random.choice([1,3], p=[0.10, 0.90])
        else:
            return 3
    elif self.wolf_team == 3:
        if log1[0] in self.p1[6:15] or log1[1] in self.p1
           [6:15] or log1[2] in self.p1[6:15]:
            return np.random.choice([1,3], p=[0.90, 0.10])
        else:
            return 1
#自分がプレイヤー3
else:
    if self.wolf_team == 0:
        if self.p1[12] in log1 or self.p1[13] in log1:
```

```
            return 1
        elif self.p2[12] in log3 or self.p2[13] in log3:
            return 2

        if self.p1[6] in log1 or self.p2[6] in log2:
            if self.p1[6] in log1 and self.p2[6] in log2:
                return np.random.choice([1,2])
            elif self.p1[6] in log1:
                return 1
            elif self.p2[6] in log2:
                return 2
        elif choices[4] in log1 or choices[4] in log2:
            if choices[4] in log1 and choices[4] in log2:
                return np.random.choice([1,2])
            elif choices[4] in log1:
                return 2
            elif choices[4] in log2:
                return 1
        elif choices[3] in log1 or choices[3] in log2:
            if choices[3] in log1 and choices[3] in log2:
                return np.random.choice([1,2])
            elif choices[3] in log1:
                return 1
            elif choices[3] in log2:
                return 2
        else:
            return np.random.choice([1,2])
    elif self.wolf_team == 2:
        if log1[0] in self.p1[6:15] or log1[1] in self.p1
           [6:15] or log1[2] in self.p1[6:15]:
            return np.random.choice([1,2], p=[0.90, 0.10])
        else:
            return 1
    else:
        if log2[0] in self.p2[6:15] or log2[1] in self.p2
           [6:15] or log2[2] in self.p2[6:15]:
            return np.random.choice([1,2], p=[0.10, 0.90])
```

```
                else:
                    return 2
```

- fortun_teller.py

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

import numpy as np

# 占い師クラス
class FortuneTeller:

    # コンストラクタ
    def __init__(self, status):
        self.status = status

        self.target_status = list()

    # 役割
    def role(self):
        return 'あなたは占い師です。\他のプレイヤーの役職を確認することができます。
            n\役職を確認するプレイヤーを選んでください。n'

    #能力
    def action(self, target, players):
        for n in range(len(players)-2):
            if target == f'Player{n+1}':
                if players[n].status == '人狼':
                    return f'{targetは}{players[n].statusでした。}'
                else:
                    return f'{targetは人間でした。}'
        return f'選ばれていない役職は\n{players[-1].statusと
            }{players[-2].statusです。}'

    #占いの対象のボタンテキスト
    def target_text(self, people, count):
```

```
        target_lists = list()
        for n in range(people):
            if n+1 == count:
                continue
            target_lists.append(f'Player{n+1}')
        target_lists.append('選ばれていない役職')
        return target_lists

    #コンピュータが選んだ対象と役職
    def target(self, n, players):
        #対象　プレイヤーは各、他の枚選ぶ確率1~35%290%
        if n == 1:
            self.target_index = np.random.choice([0,1,2,3], p=[0.00,
                0.05, 0.05, 0.90])
        elif n == 2:
            self.target_index = np.random.choice([0,1,2,3], p=[0.05,
                0.00, 0.05, 0.90])
        else:
            self.target_index = np.random.choice([0,1,2,3], p=[0.05,
                0.05, 0.00, 0.90])

        #他の枚を占う2
        if self.target_index == 3:
            self.target_status.append(players[-1].status)
            self.target_status.append(players[-2].status)
        #他のプレイヤーの役職
        else:
            self.target_status.append(players[self.target_index].
                status)

    #ごとの選択肢をメモPlayer
    def another_choices(self, choices, n):
        if n == 1:
            self.p1 = choices
        elif n == 2:
            self.p2 = choices
        else:
```

```
        self.p3 = choices


#占い結果による場合分け
def case(self):
    #他の枚を占ったパターン24
    if len(self.target_status) == 2:
        if self.target_status[0] == '人狼
            ' and self.target_status[1] == '人狼':
            return 1
        elif '人狼' in self.target_status and '怪盗
            ' in self.target_status:
            return 2
        elif '人狼' in self.target_status and '村人
            ' in self.target_status:
            return 3
        elif '怪盗' in self.target_status and '村人
            ' in self.target_status:
            return 4
    else:
        if self.target_index == 0:
            if self.target_status[0] == '人狼':
                return 5
            else:
                return 6
        if self.target_index == 1:
            if self.target_status[0] == '人狼':
                return 7
            else:
                return 8
        if self.target_index == 2:
            if self.target_status[0] == '人狼':
                return 9
            else:
                return 10


#選択肢を確率で返す
```

```
#自分の選択肢choices 各プレイヤーの発言記録log_list
#c 何回目の発言か pn 自分のプレイヤー番号
def choice_rate(self, choices, log_list, c, pn):
    if pn == 1:
        #のログPlayer2
        log1 = log_list[1]
        #のログPlayer3
        log2 = log_list[2]
        #自分のログ
        log3 = log_list[0]
    elif pn == 2:
        #のログPlayer1
        log1 = log_list[0]
        #のログPlayer3
        log2 = log_list[2]
        #自分のログ
        log3 = log_list[1]
    else:
        #のログPlayer1
        log1 = log_list[0]
        #のログPlayer2
        log2 = log_list[1]
        #自分のログ
        log3 = log_list[2]

    #各プレイヤーごとの選択肢
    if c >= 2:
        if pn == 1:
            p1_choices = self.p2
            p2_choices = self.p3
        elif pn == 2:
            p1_choices = self.p1
            p2_choices = self.p3
        else:
            p1_choices = self.p1
            p2_choices = self.p2
```

```
#選択肢を選ぶ確率
#最初の発言
if c == 1:
    #占い結果によって確率を変える
    rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
        0.00, 0.00, 0.00,
                0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                    0.00, 0.00, 0.00, 0.10]
    #他の枚を占ったパターン24
    if len(self.target_status) == 2:
        #共に人狼の場合
        if self.case() == 1:
            rate_list[2] = 1.00
            rate_list[-1] = 0.00
        #人狼怪盗の場合
        elif self.case() == 2:
            rate_list[3] = 1.00
            rate_list[-1] = 0.00
        #人狼村人の場合
        elif self.case() == 3:
            rate_list[4] = 0.90
        #怪盗村人の場合
        elif self.case() == 4:
            rate_list[-1] = 0.90
            rate_list[1] = 0.10
    #他のプレイヤーを占った場合
    else:
        #自分がPLayer1
        if pn == 1:
            #が人狼Player2
            if self.case() == 7:
                rate_list[6] = 1.00
                rate_list[-1] = 0.00
            #が人間Player2
            elif self.case() == 8:
                rate_list[7] = 0.90
            #が人狼Player3
```

```
                elif self.case() == 9:
                    rate_list[8] = 1.00
                    rate_list[-1] = 0.00
                #が人間Player3
                elif self.case() == 10:
                    rate_list[9] = 0.90
            #自分がかPlayer23
            else:
                #が人狼Player1
                if self.case() == 5:
                    rate_list[6] = 1.00
                    rate_list[-1] = 0.00
                #が人間Player1
                elif self.case() == 6:
#                   rate_list[7] = 0.90
                    rate_list[-1] = 1.00
                #かが人狼Player23
                elif self.case() == 7 or self.case() == 9:
                    rate_list[8] = 1.00
                    rate_list[-1] = 0.00
                #かが人間Player23
                else:
                    rate_list[9] = 0.90
        #回目の発言2
        elif c == 2:
            rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                0.00, 0.00, 0.00,
                        0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 1.00]
            #発言しなかった場合
            if log3[0] == choices[-1]:
                #他の枚を占ったパターン24
                if len(self.target_status) == 2:
                    #人狼怪盗の場合
                    if self.case() == 2:
                        rate_list[3] = 1.00
                        rate_list[-1] = 0.00
```

```
            #人狼村人の場合
            elif self.case() == 3:
                rate_list[4] = 1.00
                rate_list[-1] = 0.00
            #怪盗村人の場合
            elif self.case() == 4:
                if not(log3[0] == choices[1]):
                    rate_list[7] = 0.20
                    rate_list[9] = 0.20
                    rate_list[13] = 0.20
                    rate_list[14] = 0.20
                    rate_list[15] = 0.20
                    rate_list[-1] = 0.00
        #他のプレイヤーを占った場合
        else:
            #自分がPLayer1
            if pn == 1:
                #が人間Player2
                if self.case() == 8:
                    rate_list[7] = 1.00
                    rate_list[-1] = 0.00
                #が人間Player3
                elif self.case() == 10:
                    rate_list[9] = 1.00
                    rate_list[-1] = 0.00
            #自分がかPlayer23
            else:
                #が人間Player1
                if self.case() == 6:
                    rate_list[7] = 1.00
                    rate_list[-1] = 0.00
                #かが人間Player23
                else:
                    rate_list[9] = 1.00
                    rate_list[-1] = 0.00
    #占い結果が人狼だった場合2
    elif self.case() == 1:
```

```
            rate_list[-1] = 0.00
            rate_list[-2] = 1.00
        #占い結果が人間で、怪盗が村人と交換していた時
        #プレイヤーが人間だった1
        elif self.case() == 6:
            if pn == 2:
                #プレイヤーがと交換して村人だった13
                if log1[0] == p1_choices[11]:
                    rate_list[-1] = 0.00
                    rate_list[-2] = 1.00
                #プレイヤーがと交換して村人だった31
                elif log2[0] == p2_choices[10]:
                    rate_list[-1] = 0.00
                    rate_list[-2] = 1.00
            elif pn == 3:
                #プレイヤーがと交換して村人だった12
                if log1[0] == p1_choices[10]:
                    rate_list[-1] = 0.00
                    rate_list[-2] = 1.00
                #プレイヤーがと交換して村人だった21
                elif log2[0] == p2_choices[10]:
                    rate_list[-1] = 0.00
                    rate_list[-2] = 1.00
        #プレイヤーは人間だった2
        elif self.case() == 8:
            if pn == 1:
                #プレイヤーがと交換して村人だった23
                if log1[0] == p1_choices[11]:
                    rate_list[-1] = 0.00
                    rate_list[-2] = 1.00
                #プレイヤーがと交換して村人だった32
                elif log2[0] == p2_choices[11]:
                    rate_list[-1] = 0.00
                    rate_list[-2] = 1.00
            elif pn == 3:
                #プレイヤーがと交換して村人だった12
                if log1[0] == p1_choices[10]:
```

```
                rate_list[-1] = 0.00
                rate_list[-2] = 1.00
            #プレイヤーがと交換して村人だった21
            elif log2[0] == p2_choices[10]:
                rate_list[-1] = 0.00
                rate_list[-2] = 1.00
    elif self.case() == 10:
        if pn == 1:
            #プレイヤーがと交換して村人だった23
            if log1[0] == p1_choices[11]:
                rate_list[-1] = 0.00
                rate_list[-2] = 1.00
            #プレイヤーがと交換して村人だった32
            elif log2[0] == p2_choices[11]:
                rate_list[-1] = 0.00
                rate_list[-2] = 1.00
        elif pn == 2:
            #プレイヤーがと交換して村人だった13
            if log1[0] == p1_choices[11]:
                rate_list[-1] = 0.00
                rate_list[-2] = 1.00
            #プレイヤーがと交換して村人だった31
            elif log2[0] == p2_choices[10]:
                rate_list[-1] = 0.00
                rate_list[-2] = 1.00

    #占い結果を言って他にも占い師がいた場合
    elif log1[0] in p1_choices[2:10]:
        rate_list[-1] = 0.00
        rate_list[-4] = 1.00
    elif log2[0] in p2_choices[2:10]:
        rate_list[-1] = 0.00
        rate_list[-3] = 1.00
#4回目の発言3
else:
    rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
        0.00, 0.00, 0.00,
```

```
                0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                    0.00, 0.00, 0.00, 1.00]
#平和村だった場合
if log3[1] == choices[-2]:
    rate_list[-1] = 0.00
    rate_list[-2] = 1.00
#他の枚を占っていた2
elif self.case() == 2 or self.case() == 3:
    #誰も発言していない
    if (choices[-1] == log1[0] and choices[-1] == log1[1])
        and \
    (choices[-1] == log2[0] and choices[-1] == log2[1]):
        rate_list[-1] = 0.00
        rate_list[-3] = 0.50
        rate_list[-4] = 0.50
    #一度も発言していないプレイヤーがいる
    elif choices[-1] == log1[0] and choices[-1] == log1
        [1]:
        rate_list[-1] = 0.00
        rate_list[-4] = 1.00
    elif choices[-1] == log2[0] and choices[-1] == log2
        [1]:
        rate_list[-1] = 0.00
        rate_list[-3] = 1.00
    #他の枚が人狼と怪盗2
    elif self.case() == 2:
        #怪盗か人狼か占い師と言ったプレイヤーがいる
        if log1[0] in p1_choices[1:15] or log1[1] in
            p1_choices[1:15]:
            rate_list[-1] = 0.00
            rate_list[-4] = 1.00
        elif log2[0] in p2_choices[1:15] or log2[1] in
            p2_choices[1:15]:
            rate_list[-1] = 0.00
            rate_list[-3] = 1.00
        #村人ですと同時に言ったプレイヤーがいる
        elif (log1[0] == choices[0] and log2[0] == choices
```

```
            [0]) or \
        (log1[1] == choices[0] and log2[1] == choices[0]):
            rate_list[-1] = 0.00
            rate_list[-3] = 0.50
            rate_list[-4] = 0.50
        #村人ですと後出しで言っていた場合
        elif log1[0] == choices[0] and log2[1] == choices
            [0]:
            rate_list[-1] = 0.00
            rate_list[-3] = 0.70
            rate_list[-4] = 0.30
        elif log1[1] == choices[0] and log2[0] == choices
            [0]:
            rate_list[-1] = 0.00
            rate_list[-3] = 0.30
            rate_list[-4] = 0.70
    #他の枚が人狼と村人2
    elif self.case() == 3:
        #直前に全く同じことを言っている
        if log1[1] == log2[1] and not(log1[1] == choices
            [-1]):
            rate_list[-1] = 0.00
            rate_list[-3] = 0.50
            rate_list[-4] = 0.50
        #怪盗が自分と交換したと先出ししている
        elif (pn == 1 or pn == 2) and log1[0] ==
            p1_choices[12]:
            rate_list[-1] = 0.00
            rate_list[-3] = 1.00
        elif pn == 3 and log1[0] == p2_choices[13]:
            rate_list[-1] = 0.00
            rate_list[-3] = 1.00
        elif pn == 1 and log2[0] == p2_choices[12]:
            rate_list[-1] = 0.00
            rate_list[-4] = 1.00
        elif (pn == 2 or pn == 3) and log2[0] ==
            p2_choices[13]:
```

```
        rate_list[-1] = 0.00
        rate_list[-4] = 1.00
    #怪盗が自分と交換したと後出ししている
    elif (pn == 1 or pn == 2) and log1[1] ==
        p1_choices[12]:
        rate_list[-1] = 0.00
        rate_list[-3] = 0.10
        rate_list[-4] = 0.90
    elif pn == 3 and log1[1] == p1_choices[13]:
        rate_list[-1] = 0.00
        rate_list[-3] = 0.10
        rate_list[-4] = 0.90
    elif pn == 1 and log2[1] == p2_choices[12]:
        rate_list[-1] = 0.00
        rate_list[-3] = 0.90
        rate_list[-4] = 0.10
    elif (pn == 2 or pn == 3) and log2[1] ==
        p2_choices[13]:
        rate_list[-1] = 0.00
        rate_list[-3] = 0.90
        rate_list[-4] = 0.10
    #怪盗が自分以外の占い師と交換したと言った場合
    elif (pn == 1 or pn == 2) and p1_choices[13] in
        log1:
        rate_list[-1] = 0.00
        rate_list[-3] = 0.90
        rate_list[-4] = 0.10
    elif pn == 3 and p1_choices[12] in log1:
        rate_list[-1] = 0.00
        rate_list[-3] = 0.90
        rate_list[-4] = 0.10
    elif pn == 1 and p2_choices[13] in log2:
        rate_list[-1] = 0.00
        rate_list[-3] = 0.10
        rate_list[-4] = 0.90
    elif (pn == 2 or pn == 3) and p2_choices[12] in
        log2:
```

```
            rate_list[-1] = 0.00
            rate_list[-3] = 0.10
            rate_list[-4] = 0.90
        #人狼と言ったプレイヤーがいる
        elif choices[1] == log1[0]:
            rate_list[-1] = 0.00
            rate_list[-3] = 0.20
            rate_list[-4] = 0.80
        elif choices[1] == log1[1]:
            rate_list[-1] = 0.00
            rate_list[-3] = 0.80
            rate_list[-4] = 0.20
        elif choices[1] == log2[0]:
            rate_list[-1] = 0.00
            rate_list[-3] = 0.80
            rate_list[-4] = 0.20
        elif choices[1] == log2[1]:
            rate_list[-1] = 0.00
            rate_list[-3] = 0.20
            rate_list[-4] = 0.80
        #村人と言ったプレイヤーがいる
        elif choices[0] in log1:
            rate_list[-1] = 0.00
            rate_list[-4] = 1.00
        elif choices[0] in log2:
            rate_list[-1] = 0.00
            rate_list[-3] = 1.00
        #占い師と言ったプレイヤーがいる
        elif log1[0] in p1_choices[2:10] or log1[1] in
            p1_choices[2:10]:
            rate_list[-1] = 0.00
            rate_list[-4] = 1.00
        elif log2[0] in p2_choices[2:10] or log2[1] in
            p2_choices[2:10]:
            rate_list[-1] = 0.00
            rate_list[-3] = 1.00
#誰かを占っている
```

```
elif self.case() in [5,6,7,8,9,10]:
    #プレイヤーが人間1
    if self.case() == 6:
        if pn == 2:
            #怪盗が自分と交換したと先出ししている、村人と名乗るプレイ
                ヤーがいる
            if log1[0] == p1_choices[12] and choices[0] in
                log2:
                rate_list[-1] = 0.00
                rate_list[-3] = 0.34
                rate_list[-2] = 0.66
        else:
            if log1[0] == p1_choices[13] and choices[0] in
                log2:
                rate_list[-1] = 0.00
                rate_list[-3] = 0.34
                rate_list[-2] = 0.66
    #プレイヤーが人間2
    elif self.case() == 8:
        if pn == 1:
            #怪盗が自分と交換したと先出ししている、村人と名乗るプレイ
                ヤーがいる
            if log1[0] == p1_choices[12] and choices[0] in
                log2:
                rate_list[-1] = 0.00
                rate_list[-3] = 0.34
                rate_list[-2] = 0.66
        else:
            if log2[0] == p2_choices[13] and choices[0] in
                log1:
                rate_list[-1] = 0.00
                rate_list[-4] = 0.34
                rate_list[-2] = 0.66
    #プレイヤーが人間3
    elif self.case() == 10:
        if pn == 1:
            #怪盗が自分と交換したと先出ししている、村人と名乗るプレイ
                ヤーがいる
```

```
            if log2[0] == p2_choices[12] and choices[0] in
                log1:
                rate_list[-1] = 0.00
                rate_list[-4] = 0.34
                rate_list[-2] = 0.66
        else:
            if log2[0] == p2_choices[13] and choices[0] in
                log1:
                rate_list[-1] = 0.00
                rate_list[-4] = 0.34
                rate_list[-2] = 0.66
    #怪盗が自分と交換したと後出ししている
    elif (pn == 1 or pn == 2) and log1[1] == p1_choices
        [12]:
        rate_list[-1] = 0.00
        rate_list[-3] = 0.10
        rate_list[-4] = 0.90
    elif pn == 3 and log1[1] == p1_choices[13]:
        rate_list[-1] = 0.00
        rate_list[-4] = 0.10
        rate_list[-3] = 0.90
    elif pn == 1 and log2[1] == p2_choices[12]:
        rate_list[-1] = 0.00
        rate_list[-3] = 0.10
        rate_list[-4] = 0.90
    elif (pn == 2 or pn == 3) and log2[1] == p2_choices
        [13]:
        rate_list[-1] = 0.00
        rate_list[-4] = 0.10
        rate_list[-3] = 0.90
    #占い師と言ったプレイヤーがいる
    elif log1[0] in p1_choices[2:10] or log1[1] in
        p1_choices[2:10]:
        rate_list[-1] = 0.00
        rate_list[-4] = 1.00
    elif log2[0] in p2_choices[2:10] or log2[1] in
        p2_choices[2:10]:
```

```
                    rate_list[-1] = 0.00
                    rate_list[-3] = 1.00
                #人狼と言ったプレイヤーがいる
                elif choices[1] in log1:
                    rate_list[-1] = 0.00
                    rate_list[-4] = 1.00
                elif choices[1] in log2:
                    rate_list[-1] = 0.00
                    rate_list[-3] = 1.00
                #怪盗が自分以外の占い師と交換したと言った場合
                elif (pn == 1 or pn == 2) and p1_choices[13] in log1:
                    rate_list[-1] = 0.00
                    rate_list[-4] = 1.00
                elif pn == 3 and p1_choices[12] in log1:
                    rate_list[-1] = 0.00
                    rate_list[-4] = 1.00
                elif pn == 1 and p2_choices[13] in log2:
                    rate_list[-1] = 0.00
                    rate_list[-3] = 1.00
                elif (pn == 2 or pn == 3) and p2_choices[12] in log2:
                    rate_list[-1] = 0.00
                    rate_list[-3] = 1.00
            return np.random.choice(choices, p=rate_list)

        #投票先を決める
        def doubt(self, pn, choices, log_list):
            #のログPlayer1
            log1 = log_list[0]
            #のログPlayer2
            log2 = log_list[1]
            #のログPlayer3
            log3 = log_list[2]

    # 他の枚が人狼の場合          #2
    #           if self.case() == 1:
    #               return 0
            #平和村を願う場合
```

```
if log1[1] == log2[1] == log3[1] == choices[-2]:
    return 0
if len(log1) == 3:
    if log1[2] == log2[2] == log3[2] == choices[-2]:
        return 0
#自分がプレイヤー1
if pn == 1:
    #プレイヤーが人狼2
    if self.case() == 7:
        #怪盗が自分と交換していた場合
        if log3[0] == self.p3[12]:
            return 2
        elif choices[0] in log3:
            return np.random.choice([2,3], p=[0.90, 0.10])
        else:
            return 3
    #プレイヤーが人狼3
    elif self.case() == 9:
        #怪盗が自分と交換していた場合
        if log2[0] == self.p2[12]:
            return 3
        elif choices[0] in log2:
            return np.random.choice([2,3], p=[0.10, 0.90])
        else:
            return 2
    #プレイヤーが人間2
    elif self.case() == 8:
        #怪盗が自分と交換していた場合
        if log2[0] == self.p2[12]:
            if choices[0] in log3:
                return np.random.choice([2,3], p=[0.20, 0.80])
            else:
                return 3
        #占ったプレイヤーが村人と言っていた場合
        elif choices[0] in log2:
            return np.random.choice([2,3], p=[0.10, 0.90])
        #怪盗とも村人とも言っていない
```

```
        else:
            return 2
    #プレイヤーが人間3
    elif self.case() == 10:
        #怪盗が自分と交換していた場合
        if log3[0] == self.p3[12]:
            if choices[0] in log2:
                return np.random.choice([2,3], p=[0.80, 0.20])
            else:
                return 2
        #占ったプレイヤーが村人と言っていた場合
        elif choices[0] in log3:
            return np.random.choice([2,3], p=[0.90, 0.10])
        #怪盗とも村人とも言っていない
        else:
            return 3
    #他の枚が怪盗と人狼2
    elif self.case() == 2:
        #疑わしさ
        p2_level = 0
        p3_level = 0

        #人狼、占い師、怪盗と名乗るプレイヤーは怪しい
        for n in range(3):
            if log2[n] in self.p2[1:15]:
                p2_level += 100
            if log3[n] in self.p3[1:15]:
                p3_level += 100
        #一度も発言しないのは怪しい
        if log2[0] == log2[1] == log2[2] == choices[-1]:
            p2_level += 50
        if log3[0] == log3[1] == log3[2] == choices[-1]:
            p3_level += 50
        #村人と後出しも怪しい
        for n in range(2):
            if log2[n] == log3[n+1] == choices[0]:
                p3_level += 20
```

```
            if log3[n] == log2[n+1] == choices[0]:
                p2_level += 20
        if log2[0] == log3[2]:
            p3_level += 20
        if log3[0] == log2[2]:
            p2_level += 20
        #発言しないほど疑わしい
        for n in range(3):
            if log2[n] == choices[0]:
                p2_level += 10
            if log3[n] == choices[0]:
                p3_level += 10
        #投票先を決定
        if p2_level == p3_level:
            return np.random.choice([2,3])
        elif p2_level > p3_level:
            return 2
        else:
            return 3
    #他の枚が村人と人狼2
    elif self.case() == 3:
        #疑わしさ
        p2_level = 0
        p3_level = 0

        #怪盗が自分と交換していたら他プレイヤーの人狼が確定
        if log2[0] == self.p2[12]:
            p3_level += 500
        if log3[0] == self.p3[12]:
            p2_level += 500
        #占い師、村人と名乗るプレイヤーは怪しい
        for n in range(3):
            if log2[n] in self.p2[2:10]:
                p2_level += 100
            if log3[n] in self.p3[2:10]:
                p3_level += 100
            if log2[n] == self.p2[8]:
```

```
                p2_level += 300
            if log3[n] == self.p3[8]:
                p3_level += 300
            if log2[n] == choices[3]:
                p2_level += 100
            if log3[n] == choices[3]:
                p3_level += 100
            if log2[n] in self.p2[0]:
                p2_level += 100
            if log3[n] in self.p3[0]:
                p3_level += 100
        #一度も発言しないのは怪しい
        if log2[0] == log2[1] == log2[2] == choices[-1]:
            p2_level += 50
        if log3[0] == log3[1] == log3[2] == choices[-1]:
            p3_level += 50
        #占い後に現れる怪盗は怪しい
        if log1[0] == choices[4] and (self.p2[12] in log2
            [1:3]):
            p2_level += 50
        if log1[0] == choices[4] and (self.p3[12] in log3
            [1:3]):
            p3_level += 50
        if log1[1] == choices[4] and (self.p2[12] == log2[2]):
            p2_level += 50
        if log1[1] == choices[4] and (self.p3[12] == log3[2]):
            p3_level += 50
        #いきなり人狼と名乗るプレイヤーは怪しい
        if log2[0] == choices[1]:
            p2_level += 50
        if log3[0] == choices[1]:
            p3_level += 50
        #発言しないほど疑わしい
        for n in range(3):
            if log2[n] == choices[0]:
                p2_level += 10
            if log3[n] == choices[0]:
```

```
                p3_level += 10
            #占い後人狼と名乗るプレイヤーは怪盗に取られた可能性
            if log1[0] == choices[4] and (self.p2[2] in log2[1:3])
                :
                p2_level -= 20
            if log1[0] == choices[4] and (self.p3[2] in log3[1:3])
                :
                p3_level -= 20
            if log1[1] == choices[4] and (self.p2[2] == log2[2]):
                p2_level -= 20
            if log1[1] == choices[4] and (self.p3[2] == log3[2]):
                p3_level -= 20
            #投票先を決定
            if p2_level == p3_level:
                return np.random.choice([2,3])
            elif p2_level > p3_level:
                return 2
            else:
                return 3
        #他が人狼
        else:
            return np.random.choice([2,3])
    #自分がプレイヤー2
    elif pn == 2:
        #プレイヤーが人狼1
        if self.case() == 5:
            #怪盗が自分と交換していた場合
            if log3[0] == self.p3[13]:
                return 1
            elif choices[0] in log3:
                return np.random.choice([1,3], p=[0.90, 0.10])
            else:
                return 3
        #プレイヤーが人狼3
        elif self.case() == 9:
            #怪盗が自分と交換していた場合
            if log1[0] == self.p1[12]:
```

```
            return 3
        elif choices[0] in log1:
            return np.random.choice([1,3], p=[0.10, 0.90])
        else:
            return 1
    #プレイヤーが人間1
    elif self.case() == 6:
        #怪盗が自分と交換していた場合
        if log1[0] == self.p1[12]:
            if choices[0] in log3:
                return np.random.choice([1,3], p=[0.20, 0.80])
            else:
                return 3
        #占ったプレイヤーが村人と言っていた場合
        elif choices[0] in log1:
            return np.random.choice([1,3], p=[0.10, 0.90])
        #怪盗とも村人とも言っていない
        else:
            return 1
    #プレイヤーが人間3
    elif self.case() == 10:
        #怪盗が自分と交換していた場合
        if log3[0] == self.p3[13]:
            if choices[0] in log1:
                return np.random.choice([1,3], p=[0.80, 0.20])
            else:
                return 1
        #占ったプレイヤーが村人と言っていた場合
        elif choices[0] in log3:
            return np.random.choice([1,3], p=[0.90, 0.10])
        #怪盗とも村人とも言っていない
        else:
            return 3
    #他の枚が怪盗と人狼2
    elif self.case() == 2:
        #疑わしさ
        p1_level = 0
```

```
        p3_level = 0

        #人狼、占い師、怪盗と名乗るプレイヤーは怪しい
        for n in range(3):
            if log1[n] in self.p1[1:15]:
                p1_level += 100
            if log3[n] in self.p3[1:15]:
                p3_level += 100
        #一度も発言しないのは怪しい
        if log1[0] == log1[1] == log1[2] == choices[-1]:
            p1_level += 50
        if log3[0] == log3[1] == log3[2] == choices[-1]:
            p3_level += 50
        #村人と後出しも怪しい
        for n in range(2):
            if log1[n] == log3[n+1] == choices[0]:
                p3_level += 20
            if log3[n] == log1[n+1] == choices[0]:
                p1_level += 20
        if log1[0] == log3[2]:
            p3_level += 20
        if log3[0] == log1[2]:
            p1_level += 20
        #発言しないほど疑わしい
        for n in range(3):
            if log1[n] == choices[0]:
                p1_level += 10
            if log3[n] == choices[0]:
                p3_level += 10
        #投票先を決定
        if p1_level == p3_level:
            return np.random.choice([1,3])
        elif p1_level > p3_level:
            return 1
        else:
            return 3
    #他の枚が村人と人狼2
```

```
elif self.case() == 3:
    #疑わしさ
    p1_level = 0
    p3_level = 0

    #怪盗が自分と交換していたら他プレイヤーの人狼が確定
    if log1[0] == self.p1[12]:
        p3_level += 500
    if log3[0] == self.p3[13]:
        p1_level += 500
    #占い師、村人と名乗るプレイヤーは怪しい
    for n in range(3):
        if log1[n] in self.p1[2:10]:
            p1_level += 100
        if log3[n] in self.p3[2:10]:
            p3_level += 100
        if log1[n] == self.p1[8]:
            p1_level += 300
        if log3[n] == self.p3[6]:
            p3_level += 300
        if log1[n] == choices[3]:
            p1_level += 100
        if log3[n] == choices[3]:
            p3_level += 100
        if log1[n] in self.p1[0]:
            p1_level += 100
        if log3[n] in self.p3[0]:
            p3_level += 100
    #一度も発言しないのは怪しい
    if log1[0] == log1[1] == log1[2] == choices[-1]:
        p1_level += 50
    if log3[0] == log3[1] == log3[2] == choices[-1]:
        p3_level += 50
    #占い後に現れる怪盗は怪しい
    if log2[0] == choices[4] and (self.p1[12] in log1
        [1:3]):
        p1_level += 50
```

```
        if log2[0] == choices[4] and (self.p3[13] in log3
            [1:3]):
            p3_level += 50
        if log2[1] == choices[4] and (self.p1[12] == log1[2]):
            p1_level += 50
        if log2[1] == choices[4] and (self.p3[13] == log3[2]):
            p3_level += 50
        #いきなり人狼と名乗るプレイヤーは怪しい
        if log1[0] == choices[1]:
            p1_level += 50
        if log3[0] == choices[1]:
            p3_level += 50
        #発言しないほど疑わしい
        for n in range(3):
            if log1[n] == choices[0]:
                p1_level += 10
            if log3[n] == choices[0]:
                p3_level += 10
        #占い後人狼と名乗るプレイヤーは怪盗に取られた可能性
        if log2[0] == choices[4] and (self.p1[2] in log1[1:3])
            :
            p1_level -= 20
        if log2[0] == choices[4] and (self.p3[2] in log3[1:3])
            :
            p3_level -= 20
        if log2[1] == choices[4] and (self.p1[2] == log1[2]):
            p1_level -= 20
        if log2[1] == choices[4] and (self.p3[2] == log3[2]):
            p3_level -= 20
        #投票先を決定
        if p1_level == p3_level:
            return np.random.choice([1,3])
        elif p1_level > p3_level:
            return 1
        else:
            return 3
    #他が人狼
```

```
        else:
            return np.random.choice([1,3])
    #自分がプレイヤー3
    else:
        #プレイヤーが人狼1
        if self.case() == 5:
            #怪盗が自分と交換していた場合
            if log2[0] == self.p2[13]:
                return 1
            elif choices[0] in log2:
                return np.random.choice([1,2], p=[0.90, 0.10])
            elif log2[0] in self.p2[2:9]:
                return 2
            else:
                return 2
        #プレイヤーが人狼2
        elif self.case() == 7:
            #怪盗が自分と交換していた場合
            if log1[0] == self.p1[13]:
                return 2
            elif choices[0] in log1:
                return np.random.choice([1,2], p=[0.10, 0.90])
            else:
                return 1
        #プレイヤーが人間1
        elif self.case() == 6:
            #怪盗が自分と交換していた場合
            if log1[0] == self.p1[13]:
                if choices[0] in log2:
                    return np.random.choice([1,2], p=[0.20, 0.80])
                else:
                    return 2
            #占ったプレイヤーが村人と言っていた場合
            elif choices[0] in log1:
                return np.random.choice([1,2], p=[0.10, 0.90])
            #怪盗とも村人とも言っていない
            else:
```

```
            return 1
        #プレイヤーが人間2
        elif self.case() == 8:
            #怪盗が自分と交換していた場合
            if log2[0] == self.p2[13]:
                if choices[0] in log1:
                    return np.random.choice([1,2], p=[0.80, 0.20])
                else:
                    return 1
            #占ったプレイヤーが村人と言っていた場合
            elif choices[0] in log2:
                return np.random.choice([1,2], p=[0.90, 0.10])
            #怪盗とも村人とも言っていない
            else:
                return 2
        #他の枚が怪盗と人狼2
        elif self.case() == 2:
            #疑わしさ
            p1_level = 0
            p2_level = 0

            #人狼、占い師、怪盗と名乗るプレイヤーは怪しい
            for n in range(3):
                if log1[n] in self.p1[1:15]:
                    p1_level += 100
                if log2[n] in self.p2[1:15]:
                    p2_level += 100
            #一度も発言しないのは怪しい
            if log1[0] == log1[1] == log1[2] == choices[-1]:
                p1_level += 50
            if log2[0] == log2[1] == log2[2] == choices[-1]:
                p2_level += 50
            #村人と後出しも怪しい
            for n in range(2):
                if log1[n] == log2[n+1] == choices[0]:
                    p2_level += 20
                if log2[n] == log1[n+1] == choices[0]:
```

```
                p1_level += 20
        if log1[0] == log2[2]:
            p2_level += 20
        if log2[0] == log1[2]:
            p1_level += 20
        #発言しないほど疑わしい
        for n in range(3):
            if log1[n] == choices[0]:
                p1_level += 10
            if log2[n] == choices[0]:
                p2_level += 10
        #投票先を決定
        if p1_level == p2_level:
            return np.random.choice([1,2])
        elif p1_level > p2_level:
            return 1
        else:
            return 2
    #他の枚が村人と人狼2
    elif self.case() == 3:
        #疑わしさ
        p1_level = 0
        p2_level = 0

        #怪盗が自分と交換していたら他プレイヤーの人狼が確定
        if log1[0] == self.p1[13]:
            p2_level += 500
        if log2[0] == self.p2[13]:
            p1_level += 500
        #占い師、村人と名乗るプレイヤーは怪しい
        for n in range(3):
            if log1[n] in self.p1[2:10]:
                p1_level += 100
            if log2[n] in self.p2[2:10]:
                p2_level += 100
            if log1[n] == self.p1[6]:
                p1_level += 300
```

```
            if log2[n] == self.p2[6]:
                p2_level += 300
            if log1[n] == choices[3]:
                p1_level += 100
            if log2[n] == choices[3]:
                p2_level += 100
            if log1[n] in self.p1[0]:
                p1_level += 50
            if log2[n] in self.p2[0]:
                p2_level += 50
        #一度も発言しないのは怪しい
        if log1[0] == log1[1] == log1[2] == choices[-1]:
            p2_level += 50
        if log2[0] == log2[1] == log2[2] == choices[-1]:
            p2_level += 50
        #占い後に現れる怪盗は怪しい
        if log3[0] == choices[4] and (self.p1[13] in log1
            [1:3]):
            p1_level += 50
        if log3[0] == choices[4] and (self.p2[13] in log2
            [1:3]):
            p2_level += 50
        if log3[1] == choices[4] and (self.p1[13] == log1[2]):
            p1_level += 50
        if log3[1] == choices[4] and (self.p2[13] == log2[2]):
            p2_level += 50
        #いきなり人狼と名乗るプレイヤーは怪しい
        if log1[0] == choices[1]:
            p1_level += 50
        elif log2[0] == choices[1]:
            p2_level += 50
        #発言しないほど疑わしい
        for n in range(3):
            if log1[n] == choices[0]:
                p1_level += 10
            if log2[n] == choices[0]:
                p2_level += 10
```

```
                #占い後人狼と名乗るプレイヤーは怪盗に取られた可能性
                if log3[0] == choices[4] and (self.p1[2] in log1[1:3])
                    :
                    p1_level -= 20
                if log3[0] == choices[4] and (self.p2[2] in log2[1:3])
                    :
                    p2_level -= 20
                if log3[1] == choices[4] and (self.p1[2] == log1[2]):
                    p1_level -= 20
                if log3[1] == choices[4] and (self.p2[2] == log2[2]):
                    p2_level -= 20
                #投票先を決定
                if p1_level == p2_level:
                    return np.random.choice([1,2])
                elif p1_level > p2_level:
                    return 1
                else:
                    return 2
            #他が人狼
            else:
                return np.random.choice([1,2])
```

- phantom_thief.py

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

import numpy as np

# 怪盗クラス
class PhantomThief:

    # コンストラクタ
    def __init__(self, status):
        self.status = status

    # 役割
```

```
def role(self):
    return 'あなたは怪盗です。\他のプレイヤーと役職を交換することができます。n\役
        職を交換するプレイヤーを選んでください。n'

# 能力
def action(self, target, players):
    for n in range(len(players)-2):
        if target == f'Player{n+1}':
            self.number = n
            return f'{targetの}{players[n].statusと役職を交換しました。
                }'
    return '役職を交換しませんでした。'

#交換したプレイヤー
def target_number(self):
    return self.number

#対象のテキスト
def target_text(self, people, count):
    target_lists = list()
    for n in range(people):
        if n+1 == count:
            continue
        target_lists.append(f'Player{n+1}')
    target_lists.append('交換しない')
    return target_lists

#コンピュータが選んだ対象と役職
def target(self, n, players):
    #対象　各50%
    if n == 1:
        self.target_index = np.random.choice([0,1,2], p=[0.00,
            0.50, 0.50])
    elif n == 2:
        self.target_index = np.random.choice([0,1,2], p=[0.50,
            0.00, 0.50])
    else:
```

```
            self.target_index = np.random.choice([0,1,2], p=[0.50,
                0.50, 0.00])
        self.target_status = players[self.target_index].status

#怪盗の交換先
def get_target(self):
    return self.target_index

#ごとの選択肢をメモPlayer
def another_choices(self, choices, n):
    if n == 1:
        self.p1 = choices
    elif n == 2:
        self.p2 = choices
    else:
        self.p3 = choices

#選択肢を確率で返す
#自分の選択肢choices 各プレイヤーの発言記録log_list
#c 何回目の発言か pn 自分のプレイヤー番号
def choice_rate(self, choices, log_list, c, pn):
    if pn == 1:
        #のログPlayer2
        log1 = log_list[1]
        #のログPlayer3
        log2 = log_list[2]
        #自分のログ
        log3 = log_list[0]
    elif pn == 2:
        #のログPlayer1
        log1 = log_list[0]
        #のログPlayer3
        log2 = log_list[2]
        #自分のログ
        log3 = log_list[1]
    else:
        #のログPlayer1
```

```
        log1 = log_list[0]
        #のログPlayer2
        log2 = log_list[1]
        #自分のログ
        log3 = log_list[2]

    #各プレイヤーごとの選択肢
    if c >= 2:
        if pn == 1:
            p1_choices = self.p2
            p2_choices = self.p3
        elif pn == 2:
            p1_choices = self.p1
            p2_choices = self.p3
        else:
            p1_choices = self.p1
            p2_choices = self.p2

    #選択肢を選ぶ確率
    #最初の発言
    if c == 1:
        #選択肢を選ぶ確率
        rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
            0.00, 0.00, 0.00,
                    0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                        0.00, 0.00, 0.00, 1.00]
        if self.target_status == '人狼':
            if (pn == 2 or pn == 3) and self.target_index == 0:
                rate_list[-1] = 0.20
                rate_list[3] = 0.40
                rate_list[6] = 0.10
                rate_list[13] = 0.30
#                   rate_list[-1] = 0.50
#                   rate_list[6] = 0.50
            elif pn == 1 and self.target_index == 1:
                rate_list[-1] = 0.20
                rate_list[3] = 0.40
```

```
                rate_list[6] = 0.10
                rate_list[13] = 0.30
#                   rate_list[-1] = 0.00
#                   rate_list[6] = 1.00
            elif pn == 3 and self.target_index == 1:
                rate_list[-1] = 0.20
                rate_list[3] = 0.40
                rate_list[8] = 0.10
                rate_list[12] = 0.30
#                   rate_list[-1] = 0.50
#                   rate_list[6] = 0.50
            elif self.target_index == 2:
                rate_list[-1] = 0.20
                rate_list[3] = 0.40
                rate_list[8] = 0.10
                rate_list[12] = 0.30
#                   rate_list[-1] = 0.00
#                   rate_list[6] = 0.50
        elif self.target_status == '村人':
            if (pn == 2 or pn == 3) and self.target_index == 0:
                rate_list[-1] = 0.00
                rate_list[10] = 1.00
            elif pn == 1 and self.target_index == 1:
                rate_list[-1] = 0.00
                rate_list[10] = 1.00
            elif pn == 3 and self.target_index == 1:
                rate_list[-1] = 0.00
                rate_list[11] = 1.00
            elif self.target_index == 2:
                rate_list[-1] = 0.00
                rate_list[11] = 1.00
        elif self.target_status == '占い師':
            if (pn == 2 or pn == 3) and self.target_index == 0:
                rate_list[-1] = 0.00
                rate_list[12] = 1.00
            elif pn == 1 and self.target_index == 1:
                rate_list[-1] = 0.00
```

```
            rate_list[12] = 1.00
        elif pn == 3 and self.target_index == 1:
            rate_list[-1] = 0.00
            rate_list[13] = 1.00
        elif self.target_index == 2:
            rate_list[-1] = 0.00
            rate_list[13] = 1.00
#回目の発言2
elif c == 2:
    #発言せず占い師がいた場合
    if log3[0] == choices[-1] and (log1[0] in p1_choices[2:10]
        or log2[0] in p2_choices[2:10]):
        rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
            0.00, 0.00, 0.00,
                        0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 0.00, 1.00]
    #発言しなかった場合
    elif log3[0] == choices[-1]:
        #人狼の選択肢
        rate_list = [0.10, 0.01, 0.04, 0.04, 0.04, 0.04, 0.04,
            0.04, 0.04, 0.04,
                    0.03, 0.03, 0.03, 0.03, 0.00, 0.00, 0.00,
                        0.00, 0.00, 0.00, 0.45]
    #平和村が確定していた場合　占い師と交換している
    elif self.target_status == '占い師':
        rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
            0.00, 0.00, 0.00,
                        0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 0.00, 1.00]
        if log1[0] == choices[2] or log2[0] == choices[2]:
            rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                0.00, 0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                                0.00, 0.00, 0.00, 1.00, 0.00]
    #平和村が確定していた場合　村人と交換している
    elif self.target_status == '村人':
        rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
```

```
                0.00, 0.00, 0.00,
                        0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                            0.00, 0.00, 0.00, 1.00]
            if log1[0] == choices[2] or log2[0] == choices[2]:
                rate_list[-1] = 0.00
                rate_list[-2] = 1.00
            elif pn == 1 and self.target_index == 1:
                if log2[0] == p2_choices[7] or log2[0] ==
                    p2_choices[9]:
                     rate_list[-1] = 0.10
                     rate_list[-2] = 0.90
            elif pn == 1 and self.target_index == 2:
                if log1[0] == p1_choices[7] or log1[0] ==
                    p1_choices[9]:
                     rate_list[-1] = 0.10
                     rate_list[-2] = 0.90
            elif pn == 2 and self.target_index == 0:
                if log2[0] == p2_choices[7] or log2[0] ==
                    p2_choices[9]:
                     rate_list[-1] = 0.10
                     rate_list[-2] = 0.90
            elif pn == 2 and self.target_index == 2:
                if log1[0] == p1_choices[7] or log1[0] ==
                    p1_choices[9]:
                     rate_list[-1] = 0.10
                     rate_list[-2] = 0.90
            elif pn == 3 and self.target_index == 0:
                if log2[0] == p2_choices[7] or log2[0] ==
                    p2_choices[9]:
                     rate_list[-1] = 0.10
                     rate_list[-2] = 0.90
            elif pn == 3 and self.target_index == 1:
                if log1[0] == p1_choices[7] or log1[0] ==
                    p1_choices[9]:
                     rate_list[-1] = 0.10
                     rate_list[-2] = 0.90
        else:
```

```
            rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                0.00, 0.00, 0.00,
                                0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                                    0.00, 0.00, 0.00, 0.00, 1.00]
        #回目の発言3
        else:
            #一度も発言していない場合
            if log3[0] == choices[-1] and log3[1] == choices[-1]:
                rate_list = [0.38, 0.01, 0.03, 0.03, 0.03, 0.03, 0.03,
                    0.03, 0.03, 0.03,
                            0.03, 0.03, 0.03, 0.03, 0.00, 0.00, 0.00,
                                0.00, 0.00, 0.00, 0.25]
            #平和村といっていた場合
            elif log3[1] == choices[-2]:
                rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                    0.00, 0.00, 0.00,
                                0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                                    0.00, 0.00, 0.00, 1.00, 0.00]
            else:
                rate_list = [0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                    0.00, 0.00, 0.00,
                                0.00, 0.00, 0.00, 0.00, 0.00, 0.00,
                                    0.00, 0.00, 0.00, 0.00, 1.00]

        return np.random.choice(choices, p=rate_list)

    #投票先を決める
    def doubt(self, pn, choices, log_list):
        #のログPlayer1
        log1 = log_list[0]
        #のログPlayer2
        log2 = log_list[1]
        #のログPlayer3
        log3 = log_list[2]

        #平和村を願う場合
        if log1[1] == log2[1] == log3[1] == choices[-2]:
```

```
        return 0
    if len(log1) == 3:
        if log1[2] == log2[2] == log3[2] == choices[-2]:
            return 0


#自分がプレイヤー1
    if pn == 1:
        if self.target_index == 1:
            #人狼と交換した場合交換したプレイヤーに投票
            if self.target_status == '人狼':
                return 2
            #交換していないプレイヤーに投票
            else:
                return 3
        else:
            #人狼と交換した場合交換したプレイヤーに投票
            if self.target_status == '人狼':
                return 3
            #交換していないプレイヤーに投票
            else:
                return 2
#自分がプレイヤー2
    elif pn == 2:
        if self.target_index == 0:
            if self.target_status == '人狼':
                return 1
            else:
                return 3
        else:
            if self.target_status == '人狼':
                return 3
            else:
                return 1
#自分がプレイヤー3
    else:
        if self.target_index == 0:
```

```
                if self.target_status == '人狼':
                    return 1
                else:
                    return 2
            else:
                if self.target_status == '人狼':
                    return 2
                else:
                    return 1
```

- computer.py

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

import choices

class Computer:

    #コンストラクタ
    #position 役職
    def __init__(self, position, number, ch):
        self.position = position
        self.player_number = number
        ch.make_choice(number)
        self.choices = ch.get_coming_outs()

    #占い師怪盗のターゲット選択,
    def divine(self, players):
        self.position.target(self.player_number, players)

    def cp_target(self):
        return self.position.get_target()

    #発言の選択
    def choice(self, log_list, count):
        if count == 1:
```

```
            return self.first(log_list)
        elif count == 2:
            return self.second(log_list)
        else:
            return self.third(log_list)

    #別の選択肢Player
    def another_choices(self, choices, n):
        self.position.another_choices(choices, n)

    #回目の発言内容1
    def first(self, log_list):
#         if self.position.status == 人狼'':
#             return self.choices[-1]
#         else:
#         return self.choices[-1]
        return self.position.choice_rate(self.choices, log_list, 1,
            self.player_number)

    #回目の選択肢確率2
    def second(self, log_list):
#         if self.position.status == 人狼'':
#             return self.choices[-1]
#         else:
#         return self.choices[-1]
        return self.position.choice_rate(self.choices, log_list, 2,
            self.player_number)

    #回目の選択肢確率3
    def third(self, log_list):
#         if self.position.status == 人狼'':
#             return self.choices[-1]
#         else:
#         return self.choices[-1]
        return self.position.choice_rate(self.choices, log_list, 3,
            self.player_number)
```

```
        #投票先を決める
        def doubt(self, log_list):
            return self.position.doubt(self.player_number, self.choices,
                log_list)
```

- cp_werewolf.py

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:

import werewolf
import computer

#コンピュータ同士で対戦させるクラス
class Cp():
    #ゲーム開始
    def __init__(self, players, werewolf):

        #各プレイヤーの役職情報
        self.players = players

        #各プレイヤーの役職回数記録
        self.job_counts = [[0 for n in range(4)] for m in range(3)]
        #平和村の場合
        if not (self.players[-1].status == '人狼
            ' and self.players[-2].status == '人狼'):
# 占い師がいるときだけカウント                #
#             if not (self.players[-1].status == 占い師
    '' or self.players[-2].status == 占い師 ''):
            for n in range(3):
                if players[n].status == '村人':
                    self.job_counts[n][0] = 1
                elif players[n].status == '占い師':
                    self.job_counts[n][1] = 1
                elif players[n].status == '怪盗':
                    self.job_counts[n][2] = 1
                else:
```

```
                self.job_counts[n][3] = 1

        #クラスのインスタンスwerewolf
        self.werewolf = werewolf

#           for n in range(3):
#               print(players[n].status)

        #つのコンピュータの準備3
        self.cp = list()
        self.cp.append(computer.Computer(players[0], 1, self.werewolf.
            ch))
        self.cp.append(computer.Computer(players[1], 2, self.werewolf.
            ch))
        self.cp.append(computer.Computer(players[2], 3, self.werewolf.
            ch))

        #怪盗の交換が行われたか
        self.is_changed = False

        #発言を記録したリスト
        self.log_list = [self.werewolf.log1(), self.werewolf.log2(),
            self.werewolf.log3()]

        #夜のアクションへ
        self.action()

    #夜のアクション
    def action(self):
        #コンピュータの夜のアクション
        for n in range(3):
            self.cp[n].divine(self.players)

        for n in range(3):
            if self.players[n].status == '怪盗':
                self.is_changed = True
                self. target_number = self.players[n].get_target()
```

```
                self.num = n
#                   print(f怪盗は '{self.target_numberと交換した} ')

        #話し合いへ
        self.discussion()

    def discussion(self):
        #選択肢を用意
        for n in range(3):
            for m in range(3):
                choice = self.werewolf.choice_list(m+1)
                self.cp[n].another_choices(choice, m+1)
        #発言を記録
        for n in range(3):
            for m in range(3):
                cp_text = self.cp[m].choice(self.log_list, n+1)
                self.werewolf.choice_log(cp_text)

        #投票へ
        self.vote_time()

    #投票時間
    def vote_time(self):
        cp_vote = list()

        #コンピュータの投票先
        for n in range(3):
            cp_vote.append(self.cp[n].doubt(self.log_list))


        #全員平和村を願う場合
        if cp_vote[0] == 0 or cp_vote[1] == 0 or cp_vote[2] == 0:
            #平和村にし、結果へ
            self.result()
        else:
            #投票先の集計
            for n in range(3):
```

```
                self.werewolf.total_voted(cp_vote[n])
            #結果へ
            self.result()

    #ゲーム結果
    def result(self):
        #怪盗の交換が行われた場合役職を交換する
        if self.is_changed:
            self.players[self.target_number], self.players[self.num] =
                \
            self.players[self.num], self.players[self.target_number]

        #最も投票されたプレイヤーの確認
        self.werewolf.most_voted()
        #どちらのチームの勝利か
        win_lose = self.werewolf.game_result(self.players)

        #勝利数記録用のカウント
        self.wolf_win = 0
        self.citizen_win = 0

        #役職別勝利数
        self.role_win = [0] * 4

        #プレイヤー別役職勝利数
        self.p1_win_role = [0] * 4
        self.p2_win_role = [0] * 4
        self.p3_win_role = [0] * 4

        #平和村はカウントしない勝利数記録
        #平和村の場合
        if self.players[-1].status == '人狼
            ' and self.players[-2].status == '人狼':
            self.wolf_win = 0
            self.citizen_win = 0
        #怪盗の交換があった
        elif self.is_changed:
```

```
#人狼チームが勝った
if win_lose == '人狼チームの勝利です。':
    self.wolf_win = 1
    #怪盗が現在人狼
    if self.players[self.num].status == '人狼':
        #怪盗の勝利数加算
        self.role_win[2] = 1
        #怪盗だったプレイヤーと初めから人狼のプレイヤーは勝利数加算
        if self.num == 0:
            self.p1_win_role[2] = 1
            if self.players[1].status == '人狼':
                self.role_win[3] = 1
                self.p2_win_role[3] = 1
            elif self.players[2].status == '人狼':
                self.role_win[3] = 1
                self.p3_win_role[3] = 1
        elif self.num == 1:
            self.p2_win_role[2] = 1
            if self.players[0].status == '人狼':
                self.role_win[3] = 1
                self.p1_win_role[3] = 1
            elif self.players[2].status == '人狼':
                self.role_win[3] = 1
                self.p3_win_role[3] = 1
        else:
            self.p3_win_role[2] = 1
            if self.players[0].status == '人狼':
                self.p1_win_role[3] = 1
            elif self.players[1].status == '人狼':
                self.p2_win_role[3] = 1
    #怪盗が現在人狼以外
    else:
        self.role_win[3] = 1
        #人狼の勝利数加算
        if self.players[0].status == '人狼':
            self.p1_win_role[3] = 1
        elif self.players[1].status == '人狼':
```

```
                self.p2_win_role[3] = 1
            elif self.players[2].status == '人狼':
                self.p3_win_role[3] = 1
    #市民チームが勝った
    else:
        self.citizen_win = 1
        #怪盗が現在人狼
        if self.players[self.num].status == '人狼':
            #怪盗に取られた人狼は勝利数加算
            if self.target_number == 0:
                self.p1_win_role[3] = 1
            elif self.target_number == 1:
                self.p2_win_role[3] = 1
            elif self.target_number == 2:
                self.p3_win_role[3] = 1

            #それ以外の市民側は加算
            for n in range(3):
                if self.players[n].status == '村人':
                    self.role_win[0] = 1
                    if n == 0:
                        self.p1_win_role[0] = 1
                    elif n == 1:
                        self.p2_win_role[0] = 1
                    else:
                        self.p3_win_role[0] = 1
                elif self.players[n].status == '占い師':
                    self.role_win[1] = 1
                    if n == 0:
                        self.p1_win_role[1] = 1
                    elif n == 1:
                        self.p2_win_role[1] = 1
                    else:
                        self.p3_win_role[1] = 1
        #怪盗が現在人狼以外
        else:
            self.citizen_win = 1
```

```
#交換していた状態を戻す
self.players[self.target_number], self.players[
    self.num] = \
self.players[self.num], self.players[self.
    target_number]
#市民側のプレイヤー勝利数加算
for n in range(3):
    if self.players[n].status == '村人':
        self.role_win[0] = 1
        if n == 0:
            self.p1_win_role[0] = 1
        elif n == 1:
            self.p2_win_role[0] = 1
        else:
            self.p3_win_role[0] = 1
    elif self.players[n].status == '占い師':
        self.role_win[1] = 1
        if n == 0:
            self.p1_win_role[1] = 1
        elif n == 1:
            self.p2_win_role[1] = 1
        else:
            self.p3_win_role[1] = 1
    elif self.players[n].status == '怪盗':
        self.role_win[2] = 1
        if n == 0:
            self.p1_win_role[2] = 1
        elif n == 1:
            self.p2_win_role[2] = 1
        else:
            self.p3_win_role[2] = 1

#元に戻す
self.players[self.target_number], self.players[
    self.num] = \
self.players[self.num], self.players[self.
    target_number]
```

```
#怪盗の交換が無かった
else:
    #人狼チームが勝った
    if win_lose == '人狼チームの勝利です。':
        self.wolf_win = 1
        if self.players[0].status == '人狼':
            self.role_win[3] += 1
            self.p1_win_role[3] = 1
        if self.players[1].status == '人狼':
            self.role_win[3] += 1
            self.p2_win_role[3] = 1
        if self.players[2].status == '人狼':
            self.role_win[3] += 1
            self.p3_win_role[3] = 1
    #市民チームが勝った
    else:
        self.citizen_win = 1
        for n in range(3):
            if self.players[n].status == '村人':
                self.role_win[0] = 1
                if n == 0:
                    self.p1_win_role[0] = 1
                elif n == 1:
                    self.p2_win_role[0] = 1
                else:
                    self.p3_win_role[0] = 1
            elif self.players[n].status == '占い師':
                self.role_win[1] = 1
                if n == 0:
                    self.p1_win_role[1] = 1
                elif n == 1:
                    self.p2_win_role[1] = 1
                else:
                    self.p3_win_role[1] = 1
            elif self.players[n].status == '怪盗':
                self.role_win[2] = 1
                if n == 0:
```

```
                    self.p1_win_role[2] = 1
                elif n == 1:
                    self.p2_win_role[2] = 1
                else:
                    self.p3_win_role[2] = 1

#人狼の勝利数を加算
def get_wolf_win(self):
    return self.wolf_win

#市民側の勝利数を加算
def get_citizen_win(self):
    return self.citizen_win

#役職別の勝利数を加算
def get_role_win(self):
    return self.role_win

#プレイヤーの役職別勝利数を加算1
def get_p1(self):
    return self.p1_win_role

#プレイヤーの役職別勝利数を加算2
def get_p2(self):
    return self.p2_win_role

#プレイヤーの役職別勝利数を加算3
def get_p3(self):
    return self.p3_win_role

#プレイヤー別の勝利数を加算
def players_win(self):
    win_players = [0] * 3
    if self.wolf_win == 1:
        for n in range(3):
            if self.players[n].status == '人狼':
                win_players[n] = 1
```

```
        elif self.citizen_win == 1:
            for n in range(3):
                if self.players[n].status != '人狼':
                    win_players[n] = 1
        return win_players
```

- newgame.ipynb

```
import werewolf

#プレイヤー数
people = 0

win_wolf = 0
win_citizen = 0
win_players = [0] * 3
role_win = [0] * 4
role_count = [[0 for n in range(4)] for m in range(3)]
player_role_win = [[0 for n in range(4)] for m in range(3)]



if people in [1,2,3]:
    #のクラスオブジェクト生成werewolf
    new_game = werewolf.Progress(people)
    #メソッド呼び出し
    new_game.start()
else:
    for n in range(100000):
        #のクラスオブジェクト生成werewolf
        new_game = werewolf.Progress(people)
        #ゲームスタート
        new_game.start()
        #役職の回数記録
        for n in range(3):
            for m in range(4):
                role_count[n][m] += new_game.get_job_counts()[n][m]
        #勝利数記録
        win_wolf += new_game.get_wolf_win()
        win_citizen += new_game.get_citizen_win()
```

```
    x = new_game.players_win()
    for n in range(3):
        win_players[n] += x[n]
    for n in range(4):
        player_role_win[0][n] += new_game.get_p1()[n]
        player_role_win[1][n] += new_game.get_p2()[n]
        player_role_win[2][n] += new_game.get_p3()[n]
    for n in range(4):
        role_win[n] = player_role_win[0][n] + player_role_win[1][n
            ] + player_role_win[2][n]

print(f'人狼チームの勝利数{win_wolf}')
print(f'市民チームの勝利数{win_citizen}')
print(f'人狼チーム勝率
    {round((win_wolf/(win_wolf + win_citizen)) * 100, 2)}%')
print(f'市民チーム勝率
    {round((win_citizen/(win_wolf + win_citizen)) * 100, 2)}%')
for n in range(3):
    print(f'Player{n +1}{win_players[n勝]}')
role = ['村人', '占い師', '怪盗', '人狼']
print('役職別勝利数')
for n in range(4):
    print(f'{role[n]}{role_win[n勝]}')
for n in range(3):
    print(f'プレイヤー{nの戦績+1}')
    print('役職ごとの回数')
    for m in range(4):
        print(f'{role[m]}{role_count[n][m回]}')
    print('役職ごとの勝利数')
    for m in range(4):
        print(f'{role[m]}{player_role_win[n][m勝]}')
    print('役職ごとの勝率')
    for m in range(4):
        if role_count[n][m] == 0:
            print(f'{role[m]}0.0%')
        else:
            print(f'{role[m]}{round((player_role_win[n][m]/
```

```
role_count[n][m]) * 100, 2)}%')
```