Seventeen Cards

code_v2.1_20180527

Overview

This release is intended to
1. get you familiar with the communication between the host and the player, and
2. provide player templates based on which you can develop your own player.

Note

For the player codes, i.e., Ref01.py, Ref01*2.py, Ref02.py, and Ref02*2.py, the only difference between v2.1 and v2 is just one additional line -- second line in the following code segment.

```
########## ########## ########## ########## SHOW HANDS
response = iQ.get() # opponent cards J,HA,DA,CQ,CA <=== added in
v2.1
response = iQ.get() # win, lose, or tie
if response == 'win': # win the pot and ante (myBet == oppBet)
    balance += (myBet1 + myBet2 + ANTE) * 2
elif response == 'tie': # split the pot and get ante back
    balance += myBet1 + myBet2 + ANTE
```

If you have started working on your player based on any v2 player, you can make it work by including the line yourself.

For all the other files, use the v2.1 distribution.

Bug Fixes

v2.0

1. There is a bug in the card changing process -- the host did not return replacement cards!
   This is fixed in v2.
2. The host *randomly* determines the game result. This is fixed.

3. The player who gets to change cards first can exchange up to 5 cards.
   This is not true for the second player (but allowed in v0 and v1).
   If the first player changes *N* cards, the second player can change up to 7-*N* cards.

v2.1

- Before showing the game result (win, lose, or tie), the host will inform each player the opponent's cards.

What's new in v2?

1. Now there are two players: Ref01 and Ref02.
   Ref01 is a more deterministic player.
   Ref02 is similar to G01 in v0 and v1 but with the above mentioned bugs fixed.
2. Easier to include your own player in the PK game.
3. v2 introduces the **timeout** mechanism.
   If a player fails to respond within the specified amount of time (currently, 10 seconds), host will terminate the game, with the timeout-ed player to blame.
4. v2 pays attention to player **misbehavior**, for example, over-bet, under-bet, changing too many cards.
   If any misbehavior is observed, host will terminate the game, with the misbehaved player to blame.
5. The "fate17_instructions" pdf file is modified to fix the card changing bug add the instruction that shows the opponent's cards.

Contents

This release includes four Python source files, a README file, the game flow, and the game instruction file.

PK.py

This is the management program. It coordinates the game play.

Ref01.py

This player set the betting target according to the cards in hand.
The card changing strategy is straightforward -- give up the singletonn cards

except for straight.
The player's name is ***sandy***.

Ref01_2.py

A copy of "Ref01.py". The only difference is that the player name is ***sandy_2***.
If you want Sandy to play against herself, you should use Ref01 and Ref01_2 as the players.

Ref02.py

This is similar to G01.py who plays randomly.
The difference is that the bugs in G01.py are fixed.
The player name is ***rambo***.

Ref02_2.py

This is to "Ref02" as Ref01_2 is to Ref01.

utility.py

Tools and classes defined to facilitate the game play.

README.pdf

This file.

fate17_gameFlow

The game flow.

fate17_instructions

List all instructions and show where they appear in the game flow.
Updated to include the missing instruction that gives back the replacement cards.

Host Instructions & Player Responses

To try the release, run the "PK.py" file.
You will see the instructions the host send to the players and the player responses.
The following is an example.

```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Game 981 of 1000
> rambo: first
> sandy: second
```

```
> rambo: ante
> sandy: ante
> rambo: cards J,CQ,CK,HQ,SJ
> sandy: cards HA,DA,DK,SQ,CA
```

This is the initialization stage.
The format of instructions sent to players is
> *name*: *instruction*
where *name* is the player name that you define in your player source program, e.g., `name = 'sandy'` in "Ref01.py", and *instruction* is the message that the player, sandy or rambo here, receives.

Thus, `> rambo: first` informs rambo that he is the first player of this game, `> sandy: ante` informs sandy that she has to deduct 5 coins from her balance, and `> rambo: cards J,CQ,CK,HQ,SJ` indicates the five cards dealt to rambo.

```
<<<<<<<<<<<<<<<<<<<< BETTING INTERVAL I
> rambo: action:bet,check
rambo > bet 15
> sandy: opponent bet 15
> sandy: action:call,fold
sandy > call 15
> rambo: opponent call
> rambo's balance: -3417
> sandy's balance: -4623
```

This is the first betting interval.
An instruction starting with `action:` means that a response is expected.
For instance, after rambo receives `action:bet,check`, he has to decide and tell the host whether to `bet` or `check`.
To bet, send to the host the message `bet n` where *n* is the amount of bet. To check, simply send the `check` message to the host.
Note that the host will forward rarmbo's response to sandny.
Here, rambo decides to bet 15; so, sandy receives the instruction `opponent bet 15`.

```
<<<<<<<<<<<<<<<<<<<<<<<< CHANGING CARDS
> rambo: action:change
rambo > change CQ,J,CK
> rambo: cards SK,DJ,HK                 # added in v2.
> sandy: opponent change 3 cards
> sandy: action:change
sandy > change SQ,DK
> sandy: cards SA,HJ                    # added i v2.
> rambo: opponent change 2 cards
```

This is the card changing stage.
The bet setter gets to change cards first. Here rambo decides to change three cards; therefore, he replies with `change` followed by the cards separated by ','. Host will send rambo three replacement cards. On the other hand, sandy changes two cards.
Note that the other player will be informed of the number of changed cards, not the cards.

```
<<<<<<<<<<<<<<<<<<< BETTING INTERVAL II
> rambo: action:bet,check
rambo > bet 30
> sandy: opponent bet 30
> sandy: action:call,fold
sandy > call 30
> rambo: opponent call
> sandy's balance: -4653
> rambo's balance: -3447
```

This is the second betting interval; it is the same as interval 1 except for the lower and upper bounds of bet.

```
<<<<<<<<<<<<<<<<<<<<<<<<<<<< SHOW HANDS
> rambo: opponent cards HA,DA,CA,SA,HJ    # added in v2.1
> sandy: opponent cards HQ,SJ,SK,DJ,HK    # added in v2.1
> sandy: win
> rambo: lose
> rambo's balance: -3447
> sandy's balance: -4553
```

If the game stands, the winner will be determined and the balance shown.

Adjustment

Currently, the number of games is 1000 (`gameCount = 1000` at line 302 of PK.py).
Feel free to adjust the game count.

Designing Your Own Player

There are now two players, i.e., Ref01 and Ref02, that you can reference. **Ref01** is a somewhat deterministic player; **Ref02**, on the other hand, is a random player.

A player file must define the following.
1. The global variable `name` which is the player name.
2. The function `play(gameCount,iQ,rQ)` where *gameCount* is the number of

games to play, *iQ* the instruction queue, and *rQ* the response queue.

A player communicates with the host via the instruction queue *iQ* and the response queue *rQ*.
The `iQ.get()` expression gets one host instruction which is a string.
On the other hand, `rQ.put()` with the response as the argument passes your decision to the host.

If you receive an instruction that starts with "action", you must respond within 10 seconds; otherwise, you will be timeout-ed.

It is very important that you **strictly** follow the communication protocol. Otherwise, your program will hang and nothing happens.

Note

1. If you want to used codes in *utility.py*, copy and paste them to your own program.
2. PK.py and utility.py may be changed or renamed without notice.
3. To specify the players in the PK game, just modify P1 and/or P2 near the end of PK.py.
   For example, if you want your player (in file: T03_XYZ.py) to PK against sandy, P1 and P2 should be set as follows.
   ```
   P1 = 'Ref01'
   P2 = 'T03_XYZ'
   ```
   To avoid file/player name conflict in the future, you should start your file and player names with **T_XX** where **XX** is your two-digit team number, e.g., 03 for team 3 and 12 for team 12.