README for `Contract Bridge Simulator`, Computer Programming HW3

B07202020
Hao-Chien Wang

December 1, 2019

1 Program Description

This program simulate a contract bridge game. It doesn't have an AI to play with the user so it is not really a game. The rules of contract bridge can be found below, with some definitions of the terms that will be used below (such as bids and trumpsuit):

`https://en.wikipedia.org/wiki/Contract_bridge`

The computer will deal the cards and show the gaming process. The user enters the bid he wants to bid and the card he want to play. The computer will show error messages when an invalid operation is done.

2 Usage

Run `python3 main.py`. The players are represented by N, E, S and W to indicate North, East, South and West, respectively. The game autimatically starts and deal the cards. N always starts the bidding. During the bidding stage, enter number between 1 and 7 to indicate levels (when prompted), and use `s`, `h`, `d`, `c`, `nt` to represent spades, hearts, diamonds and no trump. Double and redouble are not yet supported. Passing out will result in an custom exception.

When the bidding ended and the final contract is determined, the game

starts. Each player enter a string started by the first letter of a suit and followed by the rank to represent the card he want to play (e.g. `h4` means 4 of hearts, `sa` means ace of spades, `d10` means 10 of diamonds, `ck` means king of clubs, etc) . Cards that are against the rules (e.g. cards that one doesn't own or doesn't follow the leading suit) are not accepted. The game ends when the declarer succeeds or fail to make the contract.

3 Functions and Classes

All functions and classes are defined in `ob.py`. `main.py` imports them and run the game. All the functions, classes and their definitions are listed below.

3.1 Functions

cardKey: Defines the value of cards, in order to compare. When both `trump Sute` and `leading_suit` are `None`, the function is used to sort the cards. `leadingSuit` can be either `Spades`, `Hearts`, `Diamonds` or `Clubs`. `trumpSuit` can be all the suits above or `NoTrump`. The value of a normal card is an integer between 0 and 12, while A has 12 and 2 has 0. A trump suit card has an extra value of 13. Any card that doesn't follow the leading suit has a value of 0.

3.2 Classes and NamedTuples

3.2.1 Card

A named tuple representing cards, with slots `['rank', 'suit']`.
Example: `(rank='k', suit='Spades')`

3.2.2 Bid

A named tuple representing bids, with slots `['level', 'suit']`
Example: `(level=3, suit='NoTrump')`, `(level=4, suit='Diamonds')`

3.2.3 Hand

This class represents a hand of cards.

Attributes:

- `cards`: A dictionary that maps strings of the names of suits to lists containing the cards.

Methods:

- `__init__`: The constructor requires a list of card that is dealt. It is then sorted and saved in a dictionary of lists (self.cards).
- `__str__`: print all the cards in a more readable way.
- `has`: to tell whether this player has cards in a certain suit.

3.2.4 Deck

This class represents a deck of cards. It can shuffle and deal cards.

Attributes:

- `_cards`: A list to save all cards.

Methods:

- `__init__`: Constructor to generate all cards.
- `shuffle`: Shuffle the cards.
- `deal`: deal cards. return a list of four players (Hand objects) with dealt cards. Sorted by N, E, S, W.

3.2.5 Auction

This class represent the whole bidding process.

Attributes

- `_bidlist`: A list to save all the bid objects or `'pass'`.
- `_declarer`: The declarer. Determined after the bidding is done. initialized with `None`.
- `_playerList`: A list to save the hand of the players. Sorted by N, E, S, W.

Methods

- `_contractAccepted`: To tell whether the auction has ended and the contract has been determined. That is, a bid is given and all the other three players has bid `pass`. Returns `True` if so, `False` otherwise.
- `_isLarger`: A function to determine whether a bid is larger than the last non-pass bid. Returns `True` if so, `False` otherwise.
- `bid`: The method to start bidding. Takes no arguments and return nothing.
- `declarer`: The method to return the declarer. Returns `None` if not yet determined.
- `_lastBid`: To find the last non-pass bid. Takes no arguments and return the bid.
- `_newBid`: Starting a new bid. prompts the player to enter the needed information. Returns `True` if the bid is successful and `False` if the bid is invalid.

3.2.6 Tricks

This class represent the whole playing process, that is, 13 tricks.

Attributes

- `_scoreTable`: A dictionary that maps `'EW'` and `'NS'` to two lists, storing the tricks they have won.
- `_declarer`: A number saving the declarer. 0, 1, 2, 3 means N, E, S, W, respectively.
- `_playerList`: A list of Hand objects, saving the hands of the players.
- `_cardTable`: A list saving the cards played in each tricks.
- `_trumpSuit`: A string storing the trump suit of the game.
- `_contract`: A bid object saving the final contract.
- `__init__`:

Methods

- `__init__`: Constructor. Takes an auction abject as an argumen and save the required information in the attribute variables.
- `declarer`: Returns the declarer of the game in terms of a string. Takes no arguments.
- `_newTrick`: A method to start a new trick. Takes the opener and the number of that trick as arguments. It asks each player to play a card and returns the player taking the trick.
- `_illegalCard`: A method to tell whether a card input is valid. Takes the card, the hand of the player playing that card, and the leading suit of that trick. If the card follows the leading suit and is in the hand, `True` is returned. `False` otherwise.
- `_playCard`: Asks the player to play a card. Takes the hand of the player, the number of the player (0 as N and 3 as W), and the leading suit (if the player is not the leader of the suit) as arguments. It calls `_illegalCard` to verify the card, then returns the card.
- `play`: The function to call every tricks, and tell whether the game has ended. Takes no argument and returns nothing.