UNIVERSITY OF WATERLOO

A new course announcement has been posted. Click **here** to view the new announcement.

# Bridge - Playing

WARNING: For the purposes of Academic Integrity, the final project is treated like a final exam. This means penalties are increased for any academic offences (including, but not limited to receiving 0 in the entire course!) You have been warned.
**Reminder:** Do not discuss this assignment with anyone except the instructors on the EdX discussion forums!

## Playing

Once the bidding phase has been completed, assuming a non-passed bid was made, the next phase of Bridge commences which involves playing the cards in hand one at a time to earn tricks. Recall that a **trick** is a collection of 4 cards, once from each player.

Remember that the players are traditionally denoted using the cardinal directions. There is a North player, an East player, a South player and a West player. The North and South players are partners as are the East and West players. Order of play begins with a starting player and cyclically goes through the players in the order North, East, South, West, North, East,... and so on.

Once bidding is done, assuming the bidding has a non-passed bid that was made, a contract is established (refer to the bidding part for more on contracts). This establishes a **declarer**, the person who made the last numeric bid, and the **dummy**, the declarer's partner. The dummy does not play cards in the hand. Instead, after the player sitting next to the declarer has played the first card, the dummy reveals their hand and the declarer plays both their own hand and the dummy's hand. However, cards from the dummy can only be played when it is their turn and everyone, including the opposing team can see the dummy's hand.

Cards are played one at a time. Cards played must follow suit from the original suit lead, that is, the suit of the card played must match the suit of the first card if possible. If not possible, any card may be played. The first player playing a card in a trick can also play whatever card they wish. Once all four cards in a trick have been played (one from each player) the largest card wins the trick and that player starts the next trick.

The winner of a trick is the player who played the highest trump card if the contract is not a "NT" or "No Trump" contract or the player who plays the highest card in the suit lead if no trump cards are present. Remember that the order of cards is `'2', '3', ..., '10', 'J', 'Q', 'K', 'A'`.

Saving your game can only be done after a contract has been established and only if a trick is not already in progress.

The `Game` object below is self-explanatory except for two boolean values stored in `ns_vulnerable` and `ew_vulnerable` which will be explained in the scoring section next. For now you should test with these values being either `True` or `False` but the meaning of these values won't be explained or used until the next section.

You job is to complete the following class methods and functions based on the above description of how a game is played.

```
from a10q2 import *
import check



PLAYERS = ["North", "East", "South", "West"]
PLAYERS_DICT = {"North":0, "East":1, "South":2, "West":3}
NUM_PLAYERS = len(PLAYERS)

HIGH_CARDS = ['J', 'Q', 'K', 'A']
SUITS = ['C', 'D', 'H', 'S']
VALUES = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10',
          'J', 'Q', 'K']


class Game:
  '''
  Fields:
    contract (list Bid (anyof Bid None))
    cur_player (Str)
    declarer (Str)
    declarer_tricks (Nat)
    players (list Player Player Player Player)
    ns_vulnerable (Bool)
    ew_vulnerable (Bool)

  Requires:
      contract[0] is a numeric bid
      contract[1] is one of
         None
         Bid("double", None)
         Bid("redouble", None) or
      cur_player is one of "North", "East", "South", "West"
      players.name are in some cyclic permutation of
         "North", "East", "South", "West"
      declarer is one of the player names above
      0 <=  declarer_tricks <= 13
      players[k].hand have no repetition amongst the cards for all k.

  '''


  ##PROVIDED METHODS

  def __init__(self, ctract, start_player, declarer, d_tricks, the_players,
               ns_vul, ew_vul):
    '''
    Initialized a valid Bridge Game.

    Effects: Mutates self

    __init__: Game Str (anyof Str None) -> None
```

A brief summary of the functions and methods you need to submit and complete is given below:

a. Code the `Game` class method

`save(self, file_name)`

which consumes a string `file_name` and saves the current Bridge game. The format of the saved file should contain the following information in order:

- The contract number (note you cannot saved an all-pass game!)
- The contract suit

- One of the words indicating doubling and/or redoubling from the following list: None, double, redouble
- The current player's name (that is, who is supposed to play a card next)
- The declarer's name
- The number of tricks the declaring team has won
- Whether or not North or South are vulnerable with a word from: True, False
- Whether or not East or West are vulnerable with a word from: True, False
- The number of cards the players have left to play (note that all players must have the same hand size when games are saved!)
- Four instances of the following:
    - A player's name
    - Each of the cards from the player's hand (in the order it was stored in the game object) one per line in the format `vs` where `v` is a value and `s` is a suit.

b. Write the code for the `Game` class method

```
load(file_name)
```

which consumes a string `file_name` which corresponds to a file that was saved using `save` above and returns a Bridge `Game` object properly initialized given the information from the same file.

c. Write the code for the `Game` class method

```
followed_suit(hand, card, suit_lead)
```

which consumes a Player's `hand`, a `card` from that hand and what suit was lead in `suit_lead` and returns `True` if the `card` played was a legal play and `False` otherwise.

Loading last submission entry...

main.py - Add File

```
from a10q2 import *
import check


PLAYERS = ["North", "East", "South", "West"]
PLAYERS = ["North", "East", "South", "West"]
PLAYERS_DICT = {"North":0, "East":1, "South":2, "West":3}
NUM_PLAYERS = len(PLAYERS)

HIGH_CARDS = ['J', 'Q', 'K', 'A']
SUITS = ['C', 'D', 'H', 'S']
VALUES = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10',
          'J', 'Q', 'K']


class Game:
  '''
  Fields:
    contract (list Bid (anyof Bid None))
    cur_player (Str)
    declarer (Str)
    declarer_tricks (Nat)
    players (list Player Player Player Player)
    ns_vulnerable (Bool)
    ew_vulnerable (Bool)

  Requires:
      contract[0] is a numeric bid
      contract[1] is one of
         None
         Bid("double", None)
         Bid("redouble", None) or
      cur_player is one of "North", "East", "South", "West"
      players.name are in some cyclic permutation of
         "North", "East", "South", "West"
      declarer is one of the player names above
      0 &lt;=  declarer_tricks &lt;= 13
      players[k].hand have no repetition amongst the cards for all k.

  '''

  ##PROVIDED METHODS

  def __init__(self, ctract, start_player, declarer, d_tricks, the_players,
               ns_vul, ew_vul):
    '''
    Initialized a valid Bridge Game.

    Effects: Mutates self

    __init__: Game Str (anyof Str None) -> None
```

Code Output

Run Code Save Code Reset Code Show History

Visualize

Submit Code View Submissions

## Discussion

Hide Discussion

**Topic:** Module_10 / Final Project: Bridge - Playing

**Add a Post**

Show all posts by recent activity

There are no posts in this topic yet.

✖