blackjack Documentation

Release 0.1

Adam Li

Oct 01, 2018

CONTENTS:

CHAPTER

ONE

INSTALL

This is where you write how to get a new laptop to run this project.

If you are running python3

```
python3 -m venv .venv
pip install -r requirements.txt
python3 app.py
```

Useful Commands:

1. h: hit for a new card
2. s: stand on a hand
3. d: double a hand (optional)
4. p: split a hand (optional and only possible if you start with two of the same cards, such as 4,4, or A,A, or J,J)

CHAPTER

TWO

RUNNING THE BLACKJACK MODULE

Running it is as simple as running any python script.

```
python3 app.py
```

CHAPTER

THREE

TESTS

Here, I describe how I deployed my tests and performed unit testing on the entire blackjack game backend functions.

Within the directory of the blackjack_kpcb app, you can run:

```
pytest ./
```

CHAPTER

FOUR

DOCUMENTATION FOR THE CODE

4.1 Objects - Cards (API description of objects that have a card functionality)

Card module.

class blackjack.objects.cards.deck.**Card**(*suit=0*, *rank=2*)
Class to represent a standard playing card.

Attributes: suit: integer 0-3 (0=clubs, 1=diam, 2=heart, 3=spade) rank: integer 1-13 (10=j, 11=q, 12=k, 13=A

get_rank()
Gets the rank of this specific card.

Returns

get_suit()
Gets the suit of this specific card.

Returns

class blackjack.objects.cards.deck.**Deck**(*numdecks=1*)
Class for representing a deck of cards.

Essentially creates a stack of cards as a data structure to hold the deck. Note that the number of cards in our end deck does not have to be 52, but can be multiples depending on the number of decks we use.

Attributes: cards: list of Card objects.

deal_card()
Function to deal a card from the deck by popping a card from our stack of cards.

Returns (Card) a card object.

shuffle()
Function to shuffle in place the stack of cards in our deck object.

Returns

blackjack.objects.cards.deck.**cmp**(*a*, *b*)
A helper function to compare two objects. Returns a positive number if this > other; negative if other > this; and 0 if they are equivalent.

Parameters

- **a** – first object
- **b** – second object

Returns True/False

class blackjack.objects.cards.hand.**Hand**

Class to represent a hand of playing cards. The state of a hand has two states: hittable, or stood. It is represented by the attribute: stood.

It has some added functionality by also allowing doubling, and splitting, which are some more unique features of the blackjack game.

add_card(*card*)

Function used to add a card into the user's hand.

Will not add a card, if the state of the hand is in stood.

Parameters **card** –

Returns

can_double()

Function to determine if a hand can double or not.

Returns (bool) True or False, depending on the state of the current hand.

can_hit(*ishouse=False*)

Helper function to determine if this hand is hittable. :return: (bool)

get_cards(*numerical=False*)

Get a list of the cards, or get the numerical value of each of the cards 1-10.

Parameters **numerical** – (bool) a flag to return the list of cards (strs), or a list of their values.

Returns (list) a list of the cards in this hand.

get_soft_value()

Function to return the soft value of the hand.

Returns total_val (int)

get_total_value()

Function to get the total value of the hand.

Returns (int) the best value of your hand, or the hard value of the hand.

get_value()

Function to return the total value of the hand.

Returns total_val (int)

is_splittable()

Function to determine if a hand can split or not.

Returns (bool) True or False, depending on the state of the current hand (same cards)

restart()

Function to reset a hand. Empties the list of hands.

Returns

split_cards()

Function to perform the splitting of the cards. It will pop a card from the current cards list and create a new hand from that card.

Returns (Hand) returns a new hand object that is formed from the popped card.

stand()

Makes hand stand and freezes the hand from hitting.

Returns

4.2 Objects - Users (API description of objects that have a user functionality)

User module.

class blackjack.objects.users.base.**BaseUser**
A base class for any cards user that will be playing a card cards that involves hands of cards.

class blackjack.objects.users.house.**House**
The House class for the House user in a blackjack game.

You would call this class in conjunction with `Player` and `BlackjackGame`.

Note:

An example of initialization is this: house = House()

---

deal_hand(*hand*)
Deals a hand to the house user.

Parameters **hand** – a Hand object that will act as the house's hand.

Returns None

face_up_card
A property of the house. They will always have a defined face up card, which is the first card they are dealt.

Returns the first card they are dealt.

restart()
A helper function to reset the house's hand to None.

Returns None

class blackjack.objects.users.player.**Player**(*name*)
A class/object that is a player. This is user of the cards.

This object handles all cards state information for a player, such as their hands, and winning, or losing each hand.

Member name (str) the unique name of a player.

cash_out()
Function for player to cash out.

To be implemented in a future version :return:

deal_hand(*hand*)
Deals an extra hand to the player. This is called when a player splits.

It can also be called in future versions, when a player decides to add more then 1 hand.

Parameters **hand** – (Hand) a hand object that will be appended to the player's hands.

Returns

get_hands()
Access function to return the hands a player has

Returns (list) a list of the hand objects belonging to a player

get_total_bet()
Function to get the total bet of a player during a play state.

To be implemented in a future version :return:

lose(*ihand*)
Function to handle when a player loses a hand

Parameters **ihand** – (int) the index of the hand the player is on (0-3)

Returns

place_bet(*bet*)
Function for player to place a bet.

To be implemented in a future version :param bet: :return:

restart()
A function to restart a player's cards mode.

It empties out all hands for a player. :return:

win(*ihand*)
Function to handle when a player wins a hand.

Parameters **ihand** – (int) the index of the hand the player is on (0-3)

Returns

4.3 Objects - Game (API description of objects that have a game functionality)

Game module.

class blackjack.objects.game.blackjack.**PlayBlackjack**(*numdecks*)
The blackjack cards class/object. It is a wrapper for synchronizing all the interactions between cards objects and user objects.

All players are stored in a dictionary, accessed by their name.

The house is stored as a direct object, since there is only one house user.

The game state is stored as an attribute: in_play.

Attributes: numdecks: (int) the number of decks to use in this game

add_player(*player*)
Function to add player to the game. It adds the player by their name to the dictionary of players.

Parameters **player** – (Player) is a player to be added with a unique name identifier.

Returns

check_dealer_blackjack()
Function to check if a dealer hand has a blackjack.

Returns True/False (bool)

check_player_blackjack(*hand*)
Function to check if a player hand has a blackjack.

Returns True/False (bool)

deal()
Function to deal the initial hand to players. This is the function to begin game play state.

You can not deal if the game has already started.

Returns

determine_outcomes()
Function to determine the outcomes of the game.

Returns

double(*player*, *hand*)
Function to double your bet for only one more card.

Returns

get_players()
Function to return all the player objects for this game.

Returns (list) of Player objects

hit(*playerhand*)
Function to hit the player's hand.

If bust, then in_play gets set to false and the cards ends. The user is allowed then to rebet.

If player loses, then the player loses bet.

Returns

remove_player(*player*)
Removes a player from the game.

Parameters **player** – (Player) the player to be removed from the game

Returns 1 if successful

restart()
Function to restart the game by initializing it to a beginning game state of not in play, shuffle the decks of cards and reset player's hands.

Returns

split(*player*, *hand*)
Function to split a hand and add a hand to that player.

Parameters

- **player** –
- **hand** –

Returns

stand(*hand=None*)
Function to stand on player's hand.

If the player loses, then player loses bet.

The dealer's hand is hit until it reaches 17.

Returns

4.4 Teacher (API description of the teacher for blackjack; basic strategy)

Configuration module.

class blackjack.teacher.base.**BaseTeacher**

suggest_action()

Function that suggests an action based on the current state of the cards.

Most learners will have the ability to process and remember previous cards state information by nature of their models.

Parameters

- **house** –
- **players** –

Returns

class blackjack.teacher.base.**BasicStrategyTeacher**

suggest_action(*hhand*, *phand*)

Function that suggests an action based on the current state of the cards.

Most learners will have the ability to process and remember previous cards state information by nature of their models.

Parameters

- **house** –
- **players** –

Returns

4.5 Utility (API description of the utility functions)

Utility module