gsamarakoon / Fun-projects-for-Python Public

Code Issues Pull requests Actions Projects Security Insights

master

Fun-projects-for-Python / A game of BlackJack.ipynb

gsamarakoon upload History

1 contributor

266 lines (266 sloc) | 8.75 KB

In [ ]:

```
import random

suits = ('Hearts', 'Diamonds', 'Spades', 'Clubs')
ranks = ('Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine', 'T
values = {'Two':2, 'Three':3, 'Four':4, 'Five':5, 'Six':6, 'Seven':7, 'Eight
         'Queen':10, 'King':10, 'Ace':11}

playing = True

#creating car class#

class Card:

    def __init__(self, suit, rank):
        self.suit = suit
        self.rank = rank

    def __str__(self):
        return self.rank + ' of ' + self.suit

#creating Deck, shuffle function and single dealing#

class Deck:

    def __init__(self):
        self.deck = []  # start with an empty list#
        for suit in suits:
            for rank in ranks:
                self.deck.append(Card(suit, rank))

    def __str__(self):
        deck_comp = '' #strating competition deck empty#
        for card in self.deck:
            deck_comp += '\n' + card.__str__() #add each card object;s strin
        return 'The deck has' + deck_comp

    def shuffle(self):
        random.shuffle(self.deck)

    def deal(self):
        single_card = self.deck.pop()
        return single_card

#creating a hand#

class Hand:
    def __init__(self):
        self.cards = []  # start with an empty list as we did in the Deck cl
        self.value = 0   # start with zero value
        self.aces = 0    # add an attribute to keep track of aces

    def add_card(self,card):
        self.cards.append(card)
        self.value += values[card.rank]
        if card.rank == 'Ace':
            self.aces += 1

    def adjust_for_ace(self):
```

```
        while self.value > 21 and self.aces:
            self.value -= 10
            self.aces -= 1

#creating Chips balance for comeptitor#

class Chips:

    def __init__(self):
        self.total = 100  # This can be set to a default value or supplied b
        self.bet = 0

    def win_bet(self):
        self.total += self.bet

    def lose_bet(self):
        self.total -+ self.bet

#Taking bets#

def take_bet(chips):
    while True:
        try:
            chips.bet = int(input('How many chips would you liek to bet?  ')
        except ValueError:
            print("Sorry, a bet must be an integer!")
        else:
            if chips.bet > chips.total:
                print('Sorry, your bet cannot exceed {} '.format(chips.total
            else:
                break

# taking hits#

def hit(deck,hand):
    hand.add_card(deck.deal())
    hand.adjust_for_ace()

#player to take hits or stand#

def hit_or_stand(deck,hand):
    global playing

    while True:
        x = input("Would you like to Hit or Stand? Enter 'h' or 's'")

        if x[0].lower() == 'h':
            hit(deck,hand)  # hit() function defined above

        elif x[0].lower() == 's':
            print("Player stands. Dealer is playing.")
            playing = False

        else:
            print("Sorry, please try again.")
            continue
        break

#functions to display cards#
```

quedarse o dejar de jugar

M

muestra la mano

C

```
def show_some(player,dealer):
    print("\nDealer's Hand")
    print("<card hidden>")
    print(' ', dealer.cards[1])
    print("\nPlayer's Hand: ", *player.cards, sep= '\n')

def show_all(player,dealer):
    print("\nDealer's Hand:", *dealer.cards, sep="\n")
    print("Dealer's Hand =",dealer.value)
    print("\nPlayer's Hand: ", *player.cards, sep= '\n')
    print("Player's Hand = ", player.value)
```

base de los escenarios

M

```
#functions to handle game scenarios#

def player_busts(player,dealer,chips):
    print("Player busts!")
    chips.lose_bet()

def player_wins(player,dealer,chips):
    print("Player wins!")
    chips.win_bet()

def dealer_busts(player,dealer,chips):
    print("Dealer busts!")
    chips.win_bet()

def dealer_wins(player,dealer,chips):
    print("Dealer wins!")
    chips.lose_bet()

def push(player,dealer):
    print("Dealer and Player tie! It's a push.")
```

M

MENU

reparte cartas

```
#NOW FOR THE GAME
while True:
    # Print an opening statement
    print("Welcome to my kickass Blackjack game.")

    # Create & shuffle the deck, deal two cards to each player
    deck = Deck()
    deck.shuffle()

    player_hand = Hand()
    player_hand.add_card(deck.deal())
    player_hand.add_card(deck.deal())

    dealer_hand = Hand()
    dealer_hand.add_card(deck.deal())
    dealer_hand.add_card(deck.deal())

    # Set up the Player's chips
    player_chips = Chips()

    # Prompt the Player for their bet
    take_bet(player_chips)

    # Show cards (but keep one dealer card hidden)
    show_some(player_hand, dealer_hand)

    while playing:  # recall this variable from our hit_or_stand function
```

piden o se quedan

escenarios basados segun el dealer

salir o volver a jugar

```
        # Prompt for Player to Hit or Stand
        hit_or_stand(deck, player_hand)

        # Show cards (but keep one dealer card hidden)
        show_some(player_hand,dealer_hand)

        # If player's hand exceeds 21, run player_busts() and break out of l
        if player_hand.value >21:
            player_busts(player_hand, dealer_hand, player_chips)

            break

    # If Player hasn't busted, play Dealer's hand until Dealer reaches 17
    if player_hand.value <= 21:

        while dealer_hand.value <17:
            hit(deck, dealer_hand)

        # Show all cards
        show_all(player_hand,dealer_hand)

        # Run different winning scenarios
        if dealer_hand.value > 21:
            dealer_busts(player_hand,dealer_hand,player_chips)

        elif dealer_hand.value > player_hand.value:
            dealer_wins(player_hand,dealer_hand,player_chips)

        elif dealer_hand.value < player_hand.value:
            player_wins(player_hand,dealer_hand,player_chips)

        else:
            push(player_hand,dealer_hand)


    # Inform Player of their chips total
    print("\nPlayers winnings stand at", player_chips.total)

    # Ask to play again
    new_game = input("would you like to play again? Enter 'y' or 'n'")
    if new_game[0].lower() == 'y':
        playing = True
        continue
    else:
        print('Thanks for playing! ')

        break
```

In [ ]: