Card game (WAR)

To play a hand of Blackjack the following steps must be followed:

\*\*Create a deck of 52 cards

-Shuffle the deck

-Ask the Player for their bet

-Make sure that the Player's bet does not exceed their available chips

-Deal two cards to the Dealer and two cards to the Player

-Show only one of the Dealer's cards, the other remains hidden

-Show both of the Player's cards

-Ask the Player if they wish to Hit, and take another card

-If the Player's hand doesn't Bust (go over 21), ask if they'd like to Hit again.

-If a Player Stands, play the Dealer's hand. The dealer will always Hit until the Dealer's value meets or exceeds 17

-Determine the winner and adjust the Player's chips accordingly

-Ask the Player if they'd like to play again

-----: Import the random module. This will be used to shuffle the deck prior to dealing. Then, declare variables to store suits, ranks and values. You can develop your own system, or copy ours below. Finally, declare a Boolean value to be used to control while loops. This is a common practice used to control the flow of the game.

import random

suits = ('Hearts', 'Diamonds', 'Spades', 'Clubs')

ranks = ('Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine', 'Ten', 'Jack', 'Queen', 'King', 'Ace')

values = {'Two':2, 'Three':3, 'Four':4, 'Five':5, 'Six':6, 'Seven':7, 'Eight':8, 'Nine':9, 'Ten':10, 'Jack':10,

'Queen':10, 'King':10, 'Ace':11}

playing = True

class Card:

def \_\_init\_\_(self,suit,rank):

self.suit = suit

self.rank = rank

def \_\_str\_\_(self):

return self.rank + ' of ' + self.suit

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)) # build Card objects and add them to the list

def \_\_str\_\_(self):

deck\_comp = '' # start with an empty string

for card in self.deck:

deck\_comp += '\n '+card.\_\_str\_\_() # add each Card object's print string

return 'The deck has:' + deck\_comp

def shuffle(self):

random.shuffle(self.deck)

def deal(self):

single\_card = self.deck.pop()

return single\_card

test\_deck = Deck()

print(test\_deck)

class Hand:

def \_\_init\_\_(self):

self.cards = [] # start with an empty list as we did in the Deck class

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]

def adjust\_for\_ace(self):

pass

test\_deck = Deck()

test\_deck.shuffle()

test\_player = Hand()

test\_player.add\_card(test\_deck.deal())

test\_player.add\_card(test\_deck.deal())

test\_player.value

for card in test\_player.cards:

print(card)

class Hand:

def \_\_init\_\_(self):

self.cards = [] # start with an empty list as we did in the Deck class

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 # add to self.aces

def adjust\_for\_ace(self):

while self.value > 21 and self.aces:

self.value -= 10

self.aces -= 1

class Chips:

def \_\_init\_\_(self):

self.total = 100 # This can be set to a default value or supplied by a user input

self.bet = 0

def win\_bet(self):

self.total += self.bet

def lose\_bet(self):

self.total -= self.bet

def take\_bet(chips):

while True:

try:

chips.bet = int(input('How many chips would you like to bet? '))

except ValueError:

print('Sorry, a bet must be an integer!')

else:

if chips.bet > chips.total:

print("Sorry, your bet can't exceed",chips.total)

else:

break

def hit(deck,hand):

hand.add\_card(deck.deal())

hand.adjust\_for\_ace()

def hit\_or\_stand(deck,hand):

global playing # to control an upcoming while loop

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

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)

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.")

while True:

# Print an opening statement

print('Welcome to BlackJack! Get as close to 21 as you can without going over!\n\

Dealer hits until she reaches 17. Aces count as 1 or 11.')

# 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() # remember the default value is 100

# 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

# 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 loop

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("\nPlayer's winnings stand at",player\_chips.total)

# Ask to play again

new\_game = input("Would you like to play another hand? Enter 'y' or 'n' ")

if new\_game[0].lower()=='y':

playing=True

continue

else:

print("Thanks for playing!")

break