#####################################

### WELCOME TO YOUR OOP PROJECT #####

#####################################

# For this project you will be using OOP to create a card game. This card game will

# be the card game "War" for two players, you an the computer. If you don't know

# how to play "War" here are the basic rules:

#

# The deck is divided evenly, with each player receiving 26 cards, dealt one at a time,

# face down. Anyone may deal first. Each player places his stack of cards face down,

# in front of him.

#

# The Play:

#

# Each player turns up a card at the same time and the player with the higher card

# takes both cards and puts them, face down, on the bottom of his stack.

#

# If the cards are the same rank, it is War. Each player turns up three cards face

# down and one card face up. The player with the higher cards takes both piles

# (six cards). If the turned-up cards are again the same rank, each player places

# another card face down and turns another card face up. The player with the

# higher card takes all 10 cards, and so on.

#

# There are some more variations on this but we will keep it simple for now.

# Ignore "double" wars

#

# https://en.wikipedia.org/wiki/War\_(card\_game)

from random import shuffle

# Two useful variables for creating Cards.

SUITE = 'H D S C'.split()

RANKS = '2 3 4 5 6 7 8 9 10 J Q K A'.split()

class Deck:

"""

This is the Deck Class. This object will create a deck of cards to initiate

play. You can then use this Deck list of cards to split in half and give to

the players. It will use SUITE and RANKS to create the deck. It should also

have a method for splitting/cutting the deck in half and Shuffling the deck.

"""

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

'creating Deck'

self.items = [(s,r) for s in SUITE for r in RANKS ]

def shuffle\_deck(self):

print 'shuffle cards'

shuffle(self.items)

def split\_deck(self):

print 'split Deck'

return (self.items[:26],self.items[26:])

class Hand:

'''

This is the Hand class. Each player has a Hand, and can add or remove

cards from that hand. There should be an add and remove card method here.

'''

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

self.cards=cards

def add\_card(self,card):

print 'add a card to deck'

self.cards.extend(card)

def remove\_card(self):

return self.cards.pop()

class Player:

"""

This is the Player class, which takes in a name and an instance of a Hand

class object. The Payer can then play cards and check if they still have cards.

"""

def \_\_init\_\_(self,name,hand):

self.name=name

self.hand=hand

def play\_card(self):

card=self.hand.remove\_card()

print ('{} card to be played by {}'.format(card,self.name))

return card

def cards\_for\_war(self):

war\_cards=[]

if len(self.hand.cards)<3:

return war\_cards

else:

for i in range(3):

war\_cards.append(self.hand.remove\_card())

return war\_cards

def still\_has\_cards(self):

return len(self.hand.cards)!=0

######################

#### GAME PLAY #######

######################

print("Welcome to War, let's begin...")

# Use the 3 classes along with some logic to play a game of war!

deck=Deck()

deck.shuffle\_deck()

name=input("What is your name? ")

half1,half2=deck.split\_deck()

player\_comp=Player('computer',Hand(half1))

player\_user=Player(name,Hand(half2))

war\_count=0

rounds=0

while(player\_comp.still\_has\_cards() and player\_user.still\_has\_cards()):

print("It is time for a new round!")

print("Here are the current standings: ")

print(player\_user.name+" count: "+str(len(player\_user.hand.cards)))

print(player\_comp.name+" count: "+str(len(player\_comp.hand.cards)))

print("Both players play a card!")

print('\n')

rounds+=1

card\_user=player\_user.play\_card()

card\_comp=player\_comp.play\_card()

#comparing both

table\_cards=[]

table\_cards.append(card\_comp)

table\_cards.append(card\_user)

if card\_comp[1]==card\_user[1]:

war\_count+=1

print 'war'

table\_cards.extend(player\_user.cards\_for\_war())

table\_cards.extend(player\_comp.cards\_for\_war())

if RANKS.index(card\_comp[1])<RANKS.index(card\_user[1]):

player\_user.hand.add\_card(table\_cards)

else:

player\_comp.hand.add\_card(table\_cards)

else:

if RANKS.index(card\_comp[1])<RANKS.index(card\_user[1]):

player\_user.hand.add\_card(table\_cards)

else:

player\_comp.hand.add\_card(table\_cards)

print ('rounds were '+ str(rounds))

print ('wars were '+ str(war\_count))

Footer

© 2022 GitHub, Inc.

Footer navigation

Terms

Privacy