Python les-materialen

# Warmup Project Exercise

## Simple War Game

Before we launch in to the OOP Milestone 2 Project, let’s walk through together on using OOP for a more robust and complex application, such as a game. We will use Python OOP to simulate a simplified version of the game war. Two players will each start off with half the deck, then they each remove a card, compare which card has the highest value, and the player with the higher card wins both cards. In the event of a time

## Single Card Class

### Creating a Card Class with outside variables

Here we will use some outside variables that we know don’t change regardless of the situation, such as a deck of cards. Regardless of what round,match, or game we’re playing, we’ll still need the same deck of cards.

# We'll use this later  
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':11, 'Queen':12, 'King':13, 'Ace':14}

class Card:  
   
 def \_\_init\_\_(self,suit,rank):  
 self.suit = suit  
 self.rank = rank  
 self.value = values[rank]  
   
 def \_\_str\_\_(self):  
 return self.rank + ' of ' + self.suit

Create an example card

suits[0]

'Hearts'

ranks[0]

'Two'

two\_hearts = Card(suits[0],ranks[0])

two\_hearts

<\_\_main\_\_.Card at 0x1dfaff6b898>

print(two\_hearts)

Two of Hearts

two\_hearts.rank

'Two'

two\_hearts.value

2

values[two\_hearts.rank]

2

## Deck Class

### Using a class within another class

We just created a single card, but how can we create an entire Deck of cards? Let’s explore doing this with a class that utilizes the Card class.

A Deck will be made up of multiple Cards. Which mean’s we will actually use the Card class within the \_\_init\_\_ of the Deck class.

class Deck:  
   
 def \_\_init\_\_(self):  
 # Note this only happens once upon creation of a new Deck  
 self.all\_cards = []   
 for suit in suits:  
 for rank in ranks:  
 # This assumes the Card class has already been defined!  
 self.all\_cards.append(Card(suit,rank))  
   
 def shuffle(self):  
 # Note this doesn't return anything  
 random.shuffle(self.all\_cards)  
   
 def deal\_one(self):  
 # Note we remove one card from the list of all\_cards  
 return self.all\_cards.pop()

### Create a Deck

mydeck = Deck()

len(mydeck.all\_cards)

52

mydeck.all\_cards[0]

<\_\_main\_\_.Card at 0x1dfaff269e8>

print(mydeck.all\_cards[0])

Two of Hearts

mydeck.shuffle()

print(mydeck.all\_cards[0])

Five of Spades

my\_card = mydeck.deal\_one()

print(my\_card)

King of Clubs

# Player Class

Let’s create a Player Class, a player should be able to hold instances of Cards, they should also be able to remove and add them from their hand. We want the Player class to be flexible enough to add one card, or many cards so we’ll use a simple if check to keep it all in the same method.

We’ll keep this all in mind as we create the methods for the Player class.

### Player Class

class Player:  
   
 def \_\_init\_\_(self,name):  
 self.name = name  
 # A new player has no cards  
 self.all\_cards = []   
   
 def remove\_one(self):  
 # Note we remove one card from the list of all\_cards  
 # We state 0 to remove from the "top" of the deck  
 # We'll imagine index -1 as the bottom of the deck  
 return self.all\_cards.pop(0)  
   
 def add\_cards(self,new\_cards):  
 if type(new\_cards) == type([]):  
 self.all\_cards.extend(new\_cards)  
 else:  
 self.all\_cards.append(new\_cards)  
   
   
 def \_\_str\_\_(self):  
 return f'Player {self.name} has {len(self.all\_cards)} cards.'

jose = Player("Jose")

jose

<\_\_main\_\_.Player at 0x1dfaff8b940>

print(jose)

Player Jose has 0 cards.

two\_hearts

<\_\_main\_\_.Card at 0x1dfaff6b898>

jose.add\_cards(two\_hearts)

print(jose)

Player Jose has 1 cards.

jose.add\_cards([two\_hearts,two\_hearts,two\_hearts])

print(jose)

Player Jose has 4 cards.

## War Game Logic

player\_one = Player("One")

player\_two = Player("Two")

## Setup New Game

new\_deck = Deck()

new\_deck.shuffle()

### Split the Deck between players

len(new\_deck.all\_cards)/2

26.0

for x in range(26):  
 player\_one.add\_cards(new\_deck.deal\_one())  
 player\_two.add\_cards(new\_deck.deal\_one())

len(new\_deck.all\_cards)

0

len(player\_one.all\_cards)

26

len(player\_two.all\_cards)

26

## Play the Game

import pdb

game\_on = True

round\_num = 0  
while game\_on:  
   
 round\_num += 1  
 print(f"Round {round\_num}")  
   
 # Check to see if a player is out of cards:  
 if len(player\_one.all\_cards) == 0:  
 print("Player One out of cards! Game Over")  
 print("Player Two Wins!")  
 game\_on = False  
 break  
   
 if len(player\_two.all\_cards) == 0:  
 print("Player Two out of cards! Game Over")  
 print("Player One Wins!")  
 game\_on = False  
 break  
   
 # Otherwise, the game is still on!  
   
 # Start a new round and reset current cards "on the table"  
 player\_one\_cards = []  
 player\_one\_cards.append(player\_one.remove\_one())  
   
 player\_two\_cards = []  
 player\_two\_cards.append(player\_two.remove\_one())  
   
 at\_war = True  
  
 while at\_war:  
  
  
 if player\_one\_cards[-1].value > player\_two\_cards[-1].value:  
  
 # Player One gets the cards  
 player\_one.add\_cards(player\_one\_cards)  
 player\_one.add\_cards(player\_two\_cards)  
   
   
 # No Longer at "war" , time for next round  
 at\_war = False  
   
 # Player Two Has higher Card  
 elif player\_one\_cards[-1].value < player\_two\_cards[-1].value:  
  
 # Player Two gets the cards  
 player\_two.add\_cards(player\_one\_cards)  
 player\_two.add\_cards(player\_two\_cards)  
   
 # No Longer at "war" , time for next round  
 at\_war = False  
  
 else:  
 print('WAR!')  
 # This occurs when the cards are equal.  
 # We'll grab another card each and continue the current war.  
   
 # First check to see if player has enough cards  
   
 # Check to see if a player is out of cards:  
 if len(player\_one.all\_cards) < 5:  
 print("Player One unable to play war! Game Over at War")  
 print("Player Two Wins! Player One Loses!")  
 game\_on = False  
 break  
  
 elif len(player\_two.all\_cards) < 5:  
 print("Player Two unable to play war! Game Over at War")  
 print("Player One Wins! Player One Loses!")  
 game\_on = False  
 break  
 # Otherwise, we're still at war, so we'll add the next cards  
 else:  
 for num in range(5):  
 player\_one\_cards.append(player\_one.remove\_one())  
 player\_two\_cards.append(player\_two.remove\_one())

Round 1  
Round 2  
Round 3  
Round 4  
Round 5  
Round 6  
Round 7  
Round 8  
Round 9  
Round 10  
Round 11  
Round 12  
Round 13  
Round 14  
Round 15  
Round 16  
Round 17  
Round 18  
Round 19  
Round 20  
Round 21  
Round 22  
Round 23  
Round 24  
Round 25  
Round 26  
Round 27  
Player One out of cards! Game Over

## Game Setup in One Cell

player\_one = Player("One")  
player\_two = Player("Two")  
  
new\_deck = Deck()  
new\_deck.shuffle()  
  
for x in range(26):  
 player\_one.add\_cards(new\_deck.deal\_one())  
 player\_two.add\_cards(new\_deck.deal\_one())  
   
game\_on = True

round\_num = 0  
while game\_on:  
   
 round\_num += 1  
 print(f"Round {round\_num}")  
   
 # Check to see if a player is out of cards:  
 if len(player\_one.all\_cards) == 0:  
 print("Player One out of cards! Game Over")  
 print("Player Two Wins!")  
 game\_on = False  
 break  
   
 if len(player\_two.all\_cards) == 0:  
 print("Player Two out of cards! Game Over")  
 print("Player One Wins!")  
 game\_on = False  
 break  
   
 # Otherwise, the game is still on!  
   
 # Start a new round and reset current cards "on the table"  
 player\_one\_cards = []  
 player\_one\_cards.append(player\_one.remove\_one())  
   
 player\_two\_cards = []  
 player\_two\_cards.append(player\_two.remove\_one())  
   
 at\_war = True  
  
 while at\_war:  
  
  
 if player\_one\_cards[-1].value > player\_two\_cards[-1].value:  
  
 # Player One gets the cards  
 player\_one.add\_cards(player\_one\_cards)  
 player\_one.add\_cards(player\_two\_cards)  
   
   
 # No Longer at "war" , time for next round  
 at\_war = False  
   
 # Player Two Has higher Card  
 elif player\_one\_cards[-1].value < player\_two\_cards[-1].value:  
  
 # Player Two gets the cards  
 player\_two.add\_cards(player\_one\_cards)  
 player\_two.add\_cards(player\_two\_cards)  
   
 # No Longer at "war" , time for next round  
 at\_war = False  
  
 else:  
 print('WAR!')  
 # This occurs when the cards are equal.  
 # We'll grab another card each and continue the current war.  
   
 # First check to see if player has enough cards  
   
 # Check to see if a player is out of cards:  
 if len(player\_one.all\_cards) < 5:  
 print("Player One unable to play war! Game Over at War")  
 print("Player Two Wins! Player One Loses!")  
 game\_on = False  
 break  
  
 elif len(player\_two.all\_cards) < 5:  
 print("Player Two unable to play war! Game Over at War")  
 print("Player One Wins! Player One Loses!")  
 game\_on = False  
 break  
 # Otherwise, we're still at war, so we'll add the next cards  
 else:  
 for num in range(5):  
 player\_one\_cards.append(player\_one.remove\_one())  
 player\_two\_cards.append(player\_two.remove\_one())

Round 1  
Round 2  
WAR!  
Round 3  
WAR!  
WAR!  
Round 4  
WAR!  
Round 5  
Round 6  
Round 7  
Round 8  
Round 9  
Round 10  
Round 11  
Round 12  
Round 13  
WAR!  
Round 14  
Round 15  
WAR!  
Round 16  
Round 17  
Round 18  
Round 19  
Round 20  
Round 21  
Round 22  
Round 23  
Round 24  
Round 25  
Round 26  
Round 27  
Round 28  
Round 29  
WAR!  
Round 30  
Round 31  
Round 32  
WAR!  
Round 33  
Round 34  
Round 35  
Round 36  
Round 37  
Round 38  
Round 39  
Round 40  
Round 41  
Round 42  
WAR!  
WAR!  
Round 43  
Round 44  
Round 45  
Round 46  
Round 47  
Round 48  
Round 49  
Round 50  
Round 51  
Round 52  
Round 53  
Round 54  
Round 55  
Round 56  
Round 57  
Round 58  
Round 59  
Round 60  
Round 61  
Round 62  
Round 63  
Round 64  
Round 65  
Round 66  
Round 67  
Round 68  
Round 69  
Round 70  
Round 71  
Round 72  
Round 73  
Round 74  
Round 75  
Round 76  
Round 77  
Round 78  
Round 79  
Round 80  
Round 81  
Round 82  
Round 83  
Round 84  
Round 85  
Round 86  
Round 87  
Round 88  
Round 89  
Round 90  
Round 91  
Round 92  
Round 93  
Round 94  
Round 95  
Round 96  
Round 97  
Round 98  
Round 99  
Round 100  
Round 101  
Round 102  
Round 103  
Round 104  
Round 105  
Round 106  
Round 107  
Round 108  
WAR!  
Round 109  
Round 110  
Round 111  
Round 112  
Round 113  
Round 114  
Round 115  
Round 116  
Round 117  
Round 118  
Round 119  
Round 120  
Round 121  
Round 122  
Round 123  
Round 124  
Round 125  
Round 126  
Round 127  
Round 128  
Round 129  
Round 130  
Round 131  
Round 132  
Round 133  
Round 134  
WAR!  
Round 135  
Round 136  
WAR!  
Round 137  
WAR!  
Round 138  
Round 139  
Round 140  
Round 141  
Round 142  
Round 143  
Round 144  
Round 145  
Round 146  
Round 147  
Round 148  
Round 149  
Round 150  
Round 151  
Round 152  
Round 153  
Round 154  
Round 155  
Round 156  
Round 157  
Round 158  
Round 159  
Round 160  
Round 161  
Round 162  
Round 163  
Round 164  
Round 165  
Round 166  
Round 167  
Round 168  
Round 169  
Round 170  
Round 171  
Round 172  
Round 173  
Round 174  
Round 175  
Round 176  
Round 177  
Round 178  
Round 179  
Round 180  
Round 181  
Round 182  
Round 183  
Round 184  
Round 185  
Round 186  
Round 187  
Round 188  
Round 189  
Round 190  
Round 191  
Round 192  
Round 193  
Round 194  
Round 195  
Round 196  
Round 197  
Round 198  
Round 199  
Round 200  
Round 201  
Round 202  
Round 203  
Round 204  
Round 205  
Round 206  
Round 207  
Round 208  
Round 209  
Round 210  
Round 211  
Round 212  
Round 213  
Round 214  
Round 215  
Round 216  
Round 217  
Round 218  
Round 219  
Round 220  
Round 221  
Round 222  
Round 223  
WAR!  
Round 224  
Round 225  
Round 226  
Round 227  
Round 228  
Round 229  
Round 230  
Round 231  
Round 232  
Round 233  
Round 234  
Round 235  
Round 236  
Round 237  
Round 238  
Round 239  
Round 240  
Round 241  
Round 242  
Round 243  
Round 244  
Round 245  
Round 246  
Round 247  
Round 248  
Round 249  
Round 250  
Round 251  
Round 252  
Round 253  
Round 254  
Round 255  
Round 256  
Round 257  
WAR!  
Round 258  
Round 259  
Round 260  
Round 261  
Round 262  
Round 263  
Round 264  
Round 265  
Round 266  
Round 267  
Round 268  
Round 269  
Round 270  
Round 271  
Round 272  
Round 273  
Round 274  
Round 275  
Round 276  
Round 277  
Round 278  
Round 279  
Round 280  
Round 281  
Round 282  
Round 283  
Round 284  
Round 285  
Round 286  
Round 287  
Round 288  
Round 289  
Round 290  
Round 291  
Round 292  
Round 293  
Round 294  
Round 295  
Round 296  
Round 297  
Round 298  
Round 299  
Round 300  
Round 301  
Round 302  
Round 303  
Round 304  
Round 305  
Round 306  
Round 307  
WAR!  
Round 308  
Round 309  
Round 310  
Round 311  
Round 312  
Round 313  
Round 314  
Round 315  
WAR!  
Round 316  
Round 317  
Round 318  
Round 319  
Round 320  
Round 321  
Round 322  
Round 323  
Round 324  
Round 325  
Round 326  
Round 327  
Round 328  
Round 329  
Round 330  
Round 331  
Round 332  
Round 333  
Round 334  
Round 335  
Round 336  
Round 337  
Round 338  
Round 339  
Round 340  
Round 341  
Round 342  
Round 343  
Round 344  
Round 345  
Round 346  
Round 347  
Round 348  
Round 349  
WAR!  
Round 350  
WAR!  
Player Two unable to play war! Game Over at War  
Player One Wins! Player One Loses!

len(player\_one.all\_cards)

27

len(player\_two.all\_cards)

25

print(player\_one\_cards[-1])

Ace of Diamonds

print(player\_two\_cards[-1])

Four of Hearts

## Great Work!

Other links that may interest you: \* https://www.reddit.com/r/learnpython/comments/7ay83p/war\_card\_game/ \* https://codereview.stackexchange.com/questions/131174/war-card-game-using-classes \* https://gist.github.com/damianesteban/6896120 \* https://lethain.com/war-card-game-in-python/ \* https://hectorpefo.github.io/2017-09-13-Card-Wars/ \* https://www.wimpyprogrammer.com/the-statistics-of-war-the-card-game