

# Lab 3.03 - War (Card Game) Create a program that lets a user play a **simplified** version of the card game ['War'](<http://www.pagat.com/war/war.html>). In this version, the users will share a single deck of cards and cards will not be added back to the deck after they have been played. ## Video Explanation of the Card Game War [! [Card Game: War](<https://img.youtube.com/vi/G4DhzhDIXFM/0.jpg>)](<https://youtu.be/G4DhzhDIXFM>) ### Your game should \* Start with a given shuffled deck variable (shuffle function comes from python's random library, more details below) \* Ask for player1 and player2's names. \* Have a function `player\_turn`, with the contract shown below: ``python # name: player\_turn # purpose: takes in a player name and draws/removes a card from the deck, prints "user drew card x", and returns the value # Arguments: player\_name as string, deck as list # returns: integer `` \* Have a function `compare\_scores` that takes in the two integers representing the cards drawn and compares the card values. Make sure to write the contract for `compare\_scores`! \* For simplicity Jacks will be represented as 11, Queens will be represented as 12, Kings will be represented as 13, and Aces will be represented as 14 \* For simplicity the suit does not matter \* Include a while loop that keeps the game running until there are no cards in the deck. \* If there is a tie, there is "war". Take the next two cards an whoever wins that gets all four cards (including the previous tied cards). If there is another tie, continue taking the next two cards until there a winner. The winner takes all the "war" cards. \* Keep track of the score. \* Player who won the most number of cards wins. \* Declare the name of the winner and final score at the end of the game. ## Sample Output Player 1's name: Pat Player 2's name: Sam Pat drew card 8 Sam drew card 9 Sam has high card Pat: 0 Sam: 2 Pat drew card 9 Sam drew card 8 Pat has high card Pat: 2 Sam: 2 Pat drew card 7 Sam drew card 7 War Pat: 2 Sam: 2 Pat drew card 5 Sam drew card 6 Sam has high card Sam wins war of 4 cards Pat: 2 Sam: 6 Pat drew card 13 Sam drew card 14 Sam has high card Pat: 2 Sam: 8 Pat drew card 6 Sam drew card 12 Sam has high card Pat: 2 Sam: 10 Pat drew card 4 Sam drew card 8 Sam has high card Pat: 2 Sam: 12 Pat drew card 12 Sam drew card 2 Pat has high card Pat: 4 Sam: 12 Pat drew card 7 Sam drew card 13 Sam has high card Pat: 4 Sam: 14 Pat drew card 10 Sam drew card 6 Pat has high card Pat: 6 Sam: 14 Pat drew card 9 Sam drew card 7 Pat has high card Pat: 8 Sam: 14 Pat drew card 4 Sam drew card 13 Sam has high card Pat: 8 Sam: 16 Pat drew card 3 Sam drew card 3 War Pat: 8 Sam: 16 Pat drew card 11 Sam drew card 3 Pat has high card Pat wins war of 4 cards Pat: 12 Sam: 16 Pat drew card 4 Sam drew card 10 Sam has high card Pat: 12 Sam: 18 Pat drew card 12 Sam drew card 11 Pat has high card Pat: 14 Sam: 18 Pat drew card 4 Sam drew card 11 Sam has high card Pat: 14 Sam: 20 Pat drew card 8 Sam drew card 5 Pat has high card Pat: 16 Sam: 20 Pat drew card 12 Sam drew card 9 Pat has high card Pat: 18 Sam: 20 Pat drew card 5 Sam drew card 6 Sam has high card Pat: 18 Sam: 22 Pat drew card 10 Sam drew card 13 Sam has high card Pat: 18 Sam: 24 Pat drew card 2 Sam drew card 2 War Pat: 18 Sam: 24 Pat drew card 14 Sam drew card 14 War Pat: 18 Sam: 24 Pat drew card 2 Sam drew card 5 Sam has high card Sam wins war of 6 cards Pat: 18 Sam: 30 Pat drew card 11 Sam drew card 14 Sam has high card Pat: 18 Sam: 32 Pat drew card 10 Sam drew card 3 Pat has high card Pat: 20 Sam: 32 Final Score Pat: 20 Sam: 32 Winner: Sam ### Deck Shuffling While seemingly simple-- shuffling a deck is a somewhat complicated problem. Luckily, Python's random library has a built in shuffle algorithm. Feel free to read the documentation, but we have provided a simple wrapper function that will return to you a shuffled deck of cards. ``python import random # name: shuffled\_deck # purpose: will return a shuffled deck to the user # input: # returns: a list representing a shuffled deck def shuffled\_deck(): basic\_deck = list(range(2, 15)) \* 4 random.shuffle(basic\_deck) return basic\_deck `` ### Bonus Instead of closing the program when the deck is empty, create a way for the user to play again.