For this part of the project, you need to write test cases for the poker hand evaluation. In case you aren't familiar with poker, here is a quick introduction to what each poker hand ranking means, how they compare, and how ties are broken. Even if you are familiar with poker, you will likely benefit from reading this guide, as we will comment on some tricky cases from a programming perspective.

For any variant, a player ultimately selects 5 cards from what they have to be the hand they evaluate. A player will always pick the 5 cards which give her the best hand. Note that in the hand evaluation code, the algorithm will need to consider 5 or more cards, and decide which 5 cards to use. What we show here are the final 5 cards to be evaluated (although at the end, we will give a few cautionary notes on examining more than 5 cards).

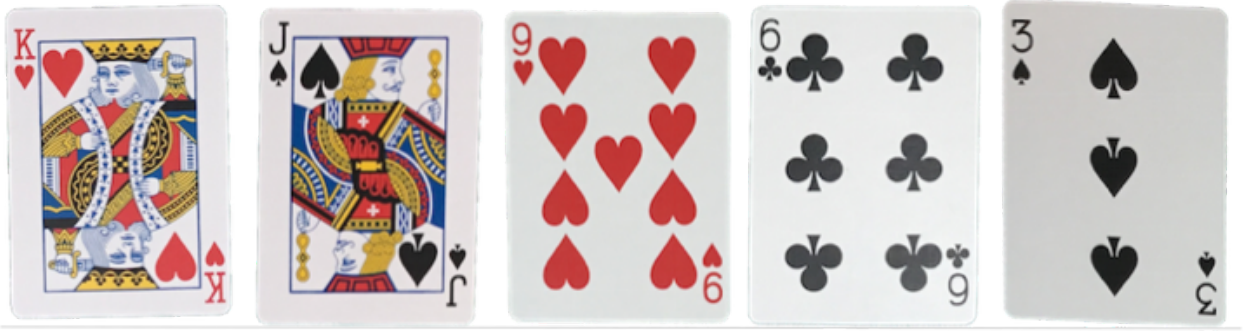
Before diving into the ranks, it is important to note that in poker, cards are generally ordered:

A > K > Q > J > 0 > 9 > 8 > 7 > 6 > 5 > 4 > 3 > 2

The only exception is that an Ace can be ranked below a 2 in an "Ace Low" Straight (which we will discuss later.

We will start with the lowest ranked hand, which has **nothing** special (that is, it does not have any of the other patterns you will see in the rest of these hands). A poker player calls a hand with nothing in it by the highest card, so Hand 0 below would be called "King High".

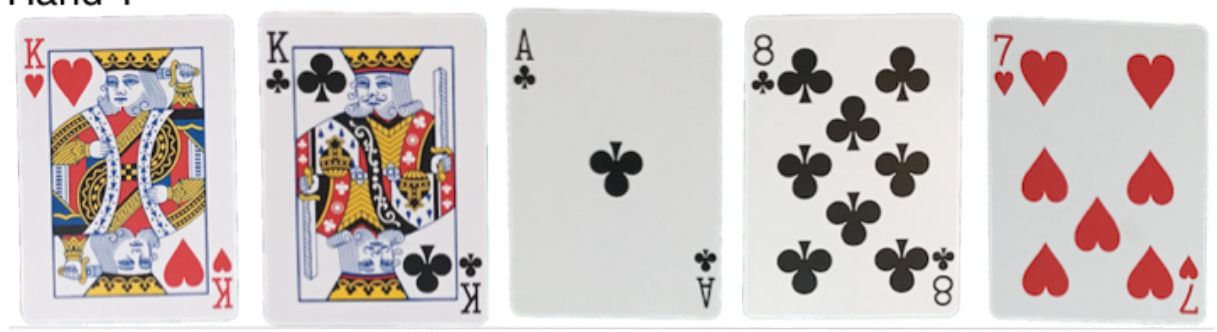
Hand 0



If multiple players tie with "nothing special" hands, then the tie is broken by comparing the ranks of each card from highest to lowest. So a "King High" hand beats a "Queen High" hand. If there are two "King High" hands, then the second highest cards in each hand are compared. If they tie, the third highest are compared, and so on. If all 5 cards are equal in rank, the hands tie.

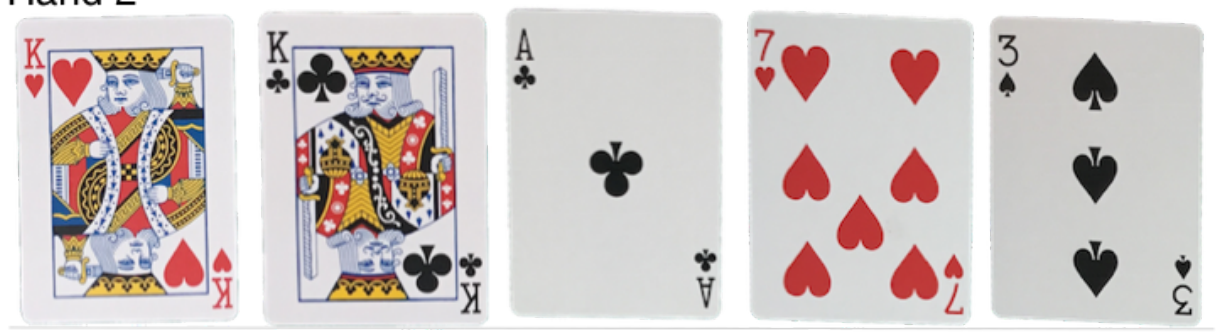
The next highest ranked hand is a **pair**: two cards of the same value. Below, you can see two examples of hands with a pair of kings. Ties are broken by first comparing the values of the pairs in each hand. A pair of Kings beats a pair of any lower values, regardless of what the tie breaker cards are (and a pair of Kings loses to a pair of aces).

Hand 1



Pair Tiebreakers

Hand 2



Pair Tiebreakers

How do you compare Hand 1 and Hand 2, which both have a pair of Kings? As they tie on their pair, you consider the tiebreaker cards from highest to lowest. Both hands have an Ace, so you go to the next highest card. Hand 1 has an 8, and Hand 2 has a 7, so Hand 1 wins.

The next higher poker hand is **two pairs**, which are illustrated below, in Hands 3 and 4:

Hand 3

Two Pairs Tiebreaker

Hand 4

Two Pairs Tiebreaker

Tiebreaking follows the same principle we have seen before. Compare the highest pairs in each hand first. If one of them is higher/lower than the other, that determines the hand. Here, both are pairs of Aces, so we move to the second pair. Again, if these differed, which one is higher determines who wins. In this example, they are the same, so we compare the 2 (Hand 3) to the Jack (Hand 4) and see that Hand 4 beats Hand 3. Note that since Hand 3 is higher rank (two pairs) than Hand 0 (nothing), 1 (pair), or 2 (pair), Hand 3 beats any of those hands without comparison of values.

The next higher hand is **three of a kind,** which means three cards of the same value**,** as illustrated below in Hands 5, 6, and 7.

Hand 5

Three of a Kind | Tiebreakers

Hand 6

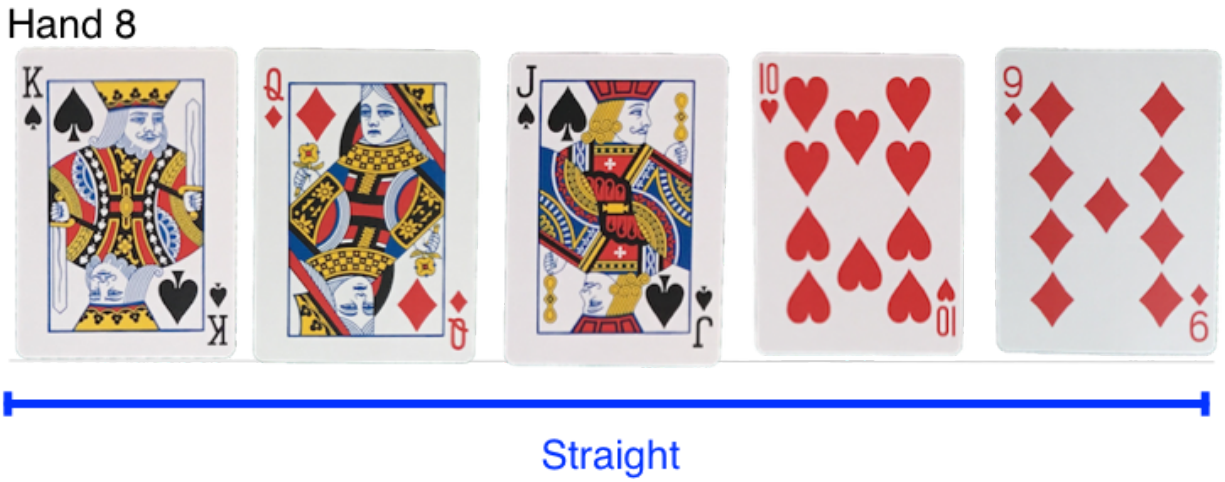
Three of a Kind | Tiebreakers

Hand 7

Three of a Kind | Tiebreakers

Tiebreaking again follows the same principles: compare the values of the three of a kind. If they differ, the higher one wins. If they are the same, compare the highest tiebreaker first, then if that ties, compare the lower. Here, all three hands have 3 kings, but hand 6 beats hand 5 because of the 7. Both Hand 6 and Hand 5 beat Hand 7 because the Ace beats the Queen, so we don't compare the four to the seven or three.

The next higher hand ranking is a straight, which is 5 cards such that the cards are sequentially ordered (each 1 rank below the previous). Tie-breaking for straights is done by comparing the values of the cards in the straights (only the first card needs to be compared---if the first card ties, all cards tie). Hand 8 shows a straight:

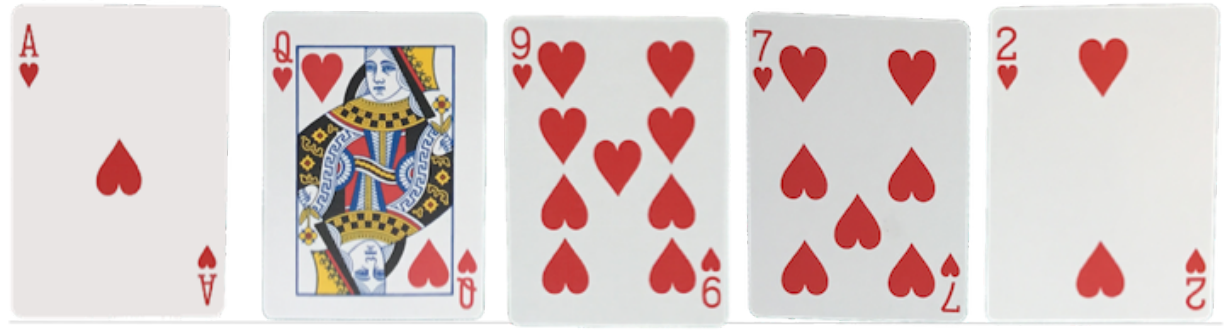


Hand 9 also shows a straight, which is called an “Ace Low” straight. An ace can be low (meaning rank below a 2) in a 5-4-3-2-A straight:

Note that ace-low straights requires special treatment in the hand evaluation, code, as the ace has to be separately considered as a low value (instead of its normal high ranking). This suggests you should both write test cases which consider this special case, as well as take care when writing code to find straights.

The next higher ranked hand is a flush, which is 5 cards of the same suit. We should two examples Hand 10 in which all cards are hearts, and Hand 11, in which all cards are spades:

Hand 10



Flush

Hand 11

Flush

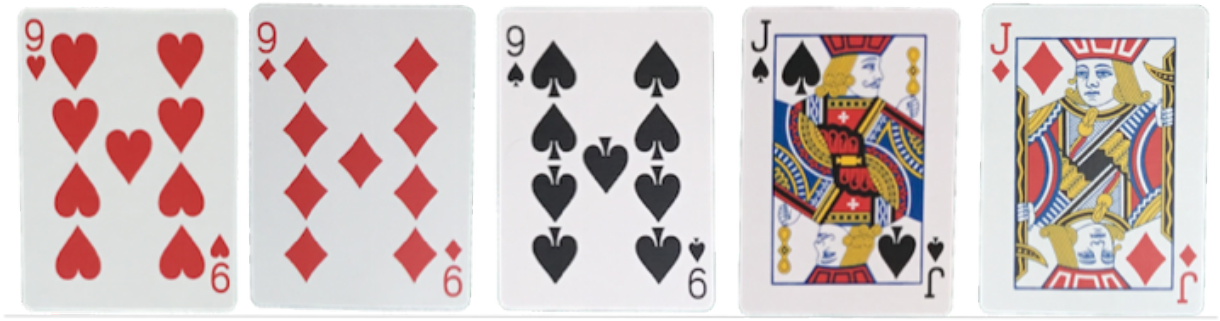
As usual, ties are broken by examining the value of the cards from highest to lowest. Hands 10 and 11 both have Aces, then Queens, so we examine the third card. Hand 11 has a ten, while Hand 10 has a nine. The ten in Hand 11 is ranked higher than the nine in Hand 10, so Hand 11 wins.

The next higher ranked hand is a full house, which means 3 cards of one value, and then a pair of another value. Below, we should two examples of a full house. Hand 12 has three Kings and two 10s (commonly called “Kings full of 10s”). You might recognize this as the hand that Drew had in the project introduction video. Hand 13 has “Nines full of Jacks”

Hand 12

Full House

Hand 13



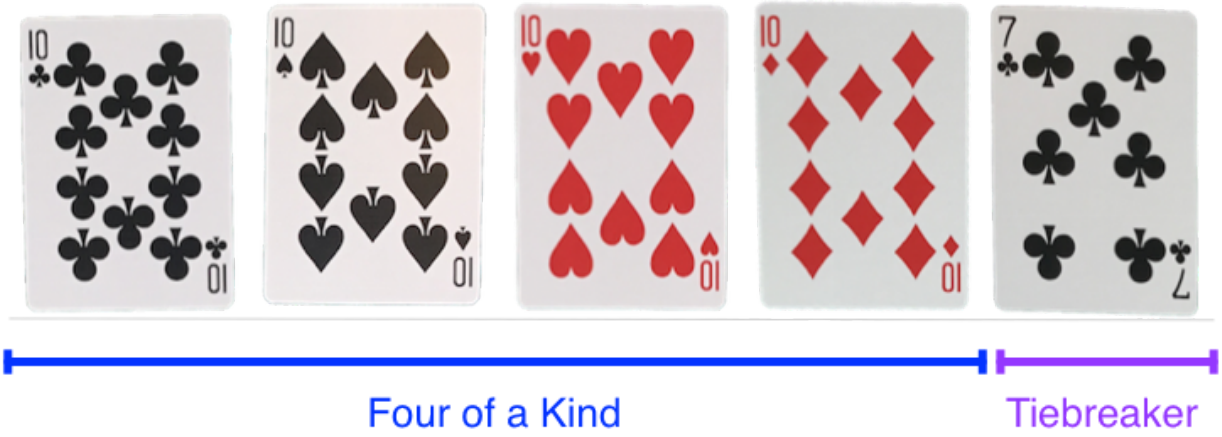
Full House

Ties are broken by first comparing the “three of a kind” part of the full house. In Hands 12 and 13, these are Kings versus Nines, so Hand 12 wins. If two hands tie on the “three of a kind” part, the pair is used to break the tie.

The next higher ranked hand is four of a kind. As the name suggests, this is four cards of the same value. Hands 14, 15, and 16 show three examples of four of a kind hands

Hand 14

Hand 15



Hand 16

As you might suspect by now, ties are broken by examining the values of the four of a kind part, and using the remaining card's value if those tie. Here, Hand 16 wins over Hands 14 and 15

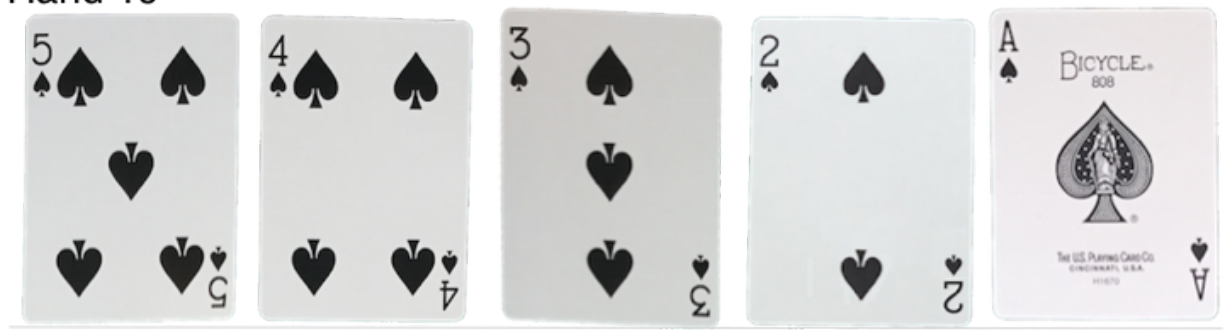
because the Kings are higher than the Tens. Hand 14 beats Hand 15 because, while both have Tens, Hand 14's tiebreaker ranks higher than Hand 15's.

The last hand ranking is a straight flush: five cards which are both a flush (all the same suit) and a straight (all sequentially ordered). As with straights, an Ace-low Straight flush is possible.

Hand 18

Straight Flush

Hand 19



Straight Flush (Ace Low)

Hand 17

Straight Flush

Ties are broken in the same way as straights and flushes: comparison of the values of the cards (as with a straight, if the highest card ties, all the others will tie since the cards are in order). Here, Hand 17 wins over the other two hands (you might recognize this as Genevieve's hand from the project introduction video). In fact, it wins over every other hand in poker, except the

three other A-K-Q-J-10 straight flushes (in the other three suits), which it ties with. Hand 18 wins over hand 19.

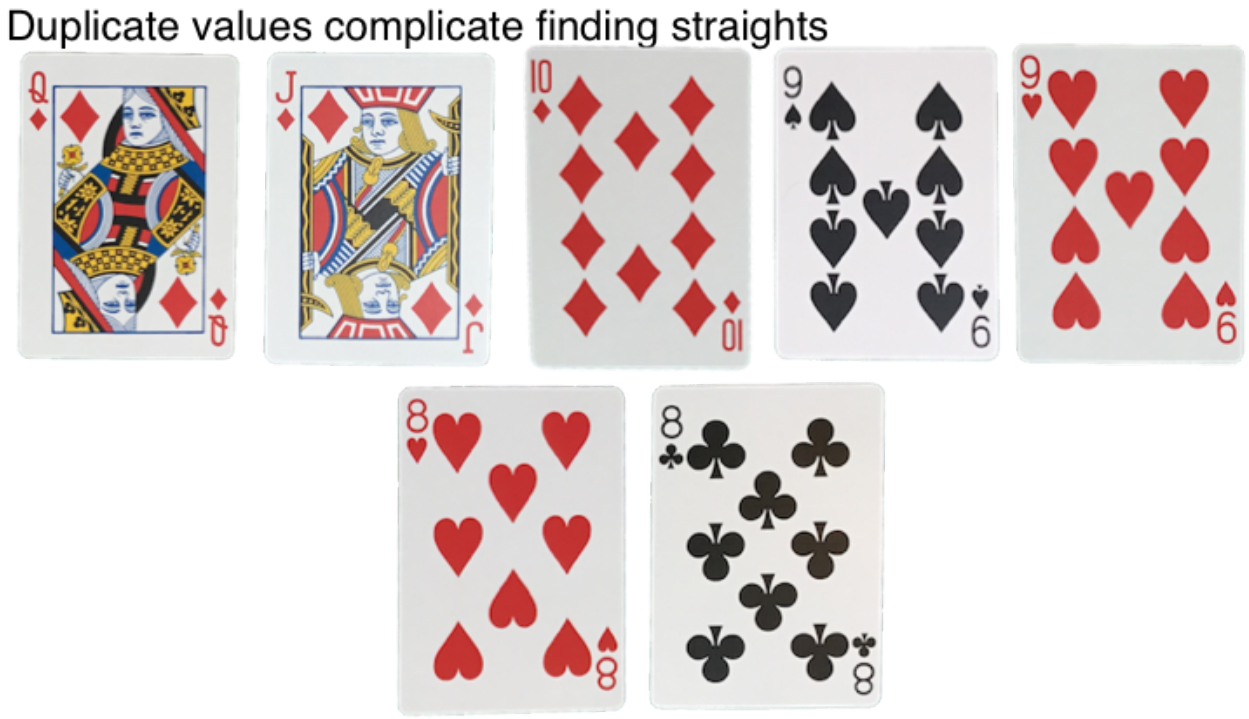
Now that you know the basic rules of poker, we want to give you a few words of caution as relates to implementing poker in code (hint: anytime you need to be cautious of something, that thing is a good idea for testcases!). A lot of these come from the fact that many variants (such as Texas Hold 'Em) have more than 5 cards to pick the hand from.

The first is that you cannot check if a hand has a straight flush by checking if it has a flush and if it has a straight. Consider the following hand:

There is a straight (A-K-Q-J-10), and a flush in diamonds (Q-J-10-5-4), but there is not a straight flush, as the straight and the flush use different cards.

The second is that when looking to see if a hand has a straight, you have to account for the possibility that there are multiple cards of equal rank in the hand. For example, consider the

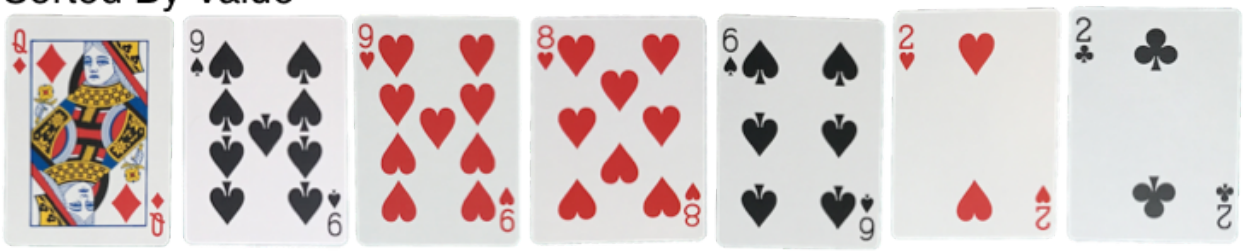
following hand:



This hand has a straight (Q-J-10-9-8), but we need to make sure our code can find it. If we sort the cards by rank, then assume we can check if each card's rank is 1 less than the current card's rank we run into problems.

The other thing to be careful of is that the program needs to not only determine what rank the hand has, but pick out the 5 cards that make it up, and put them in order of "importance" (that is, order them so that the first cards are used to break ties). The algorithm will first sort the cards by value (as in the top of the image below), then find out what hand rank the hand is (two pairs), then pick out the 5 cards that make up the final hand and order them by importance (bottom of the image). Here the pair of nines is most "important" since we compare the top pairs first in breaking ties amongst two-pair hands. The lower pair is second, then the highest remaining card.

Sorted By Value



Final Hand (Ordered By Importance)

We mention this because you will want to test not only that the hand rankings are correct (e.g., that this came up with “two pairs”) but that the final 5 card hand is correct in terms of both which cards were selected, and that they were appropriately ordered.