# Fixing Cards

April 9, 2019

## 1 Introduction

In this activity we are going to debug hands for Poker. You may want to take a few minutes to familiarize yourself with the ranking of the different hands. Note that we will not be using wild cards so there is no way to get five of a kind.

## 2 Starting Off

Make sure to download the `poker.zip` archive from the homework section of D2L and unpack it into PyCharm. There are four classes that we need to deal with. Three of them are in the `Card.py` class: `Suit`, `Rank`, and `Card`. `Suit` and `Rank` are both enumerations; they function as special named constants. Rather than using magic numbers (e.g. 1, 2, 3, 4) or strings (e.g. "club", "diamond", "heart", "spade"), Python can pick the values for us. It means we don't have to remember which number corresponded to which suit nor remember if it was "diamond" or "diamonds" or if it was capital or lowercase. The autocomplete features will also be able to finish things for us. The `Card` class represents a single card and has some static methods for constructing a whole deck of cards (and even shuffling it). The `PokerHand` class in `Poker.py` is where most of our work is going to be.

PyCharm can identify problems for us. There are some whitespace issues with `Poker.py` and the Linter test would complain if we were to submit this to Mimir. If you go to `is_flush`, there is a lot of extra whitespace at the end. This is on line 32 of `Poker.py`, so we can use Ctrl+G to bring up dialog to let us jump to a particular line (useful for very long files). Rather than delete this whitespace ourselves, we can have PyCharm do it for us with Code -> Code Cleanup. This will remove the extra whitespace here and in other places as well as fix some other things.

There are other coding standard issues that PyCharm can tell us about. If you use Code -> Inspect Code, PyCharm will bring up a list of suggestions. Two of the warnings will be that `FromString` and `IsKOfAKind` are not lowercase. The Python standard is that methods and variable names should be in `snake_case` (all lowercase letters with underscores between words). Unfortunatly, both methods are used in a bunch of places. You might want to use Edit -> Find -> Replace to change them, but they are use in multiple files and you might miss one or have a conflict with something else with the same name and accidentally change that too. Fortunatly, there is a better way. From the context menu, choose Refactor -> Rename to change them to `from_string` and `is_k_of_a_kind`. While you are at it, you can change `suitcounts` and `rankcounts` to be two words.

## 3 Tests

We have a test suite, so lets try running them and see what happens. Switch over to `PokerTests.py` and select Run -> Run.... From the dialog, select "Unittests in PokerTests.py". After this you will be able to rerun the tests by selecting the green arrow in the upper-right corner.

### 3.1 Hashing

You will immediately find that all of the tests failed for the same reason. Select `test_str` from the Test Results panel. You will see the stack trace from the failed test. There will be several links embedded in the trace; if you click on the bottom one you will jump to where the error occured. Can you figure out what is wrong with this one? Remember that Python `set`s are backed by a hash table.

### 3.2 String Parsing

Next look at `test_from_str2`. This test is telling us that we are trying to get a `Rank` that is not allowed (the enumeration is protecting us from a mistake). This time, we don't want to click on the bottom link because that is part of the `enum.py` library file; instead we want the bottommost one from `Card.py`. It looks like we are getting bad input; 0 is not one of the valid `Rank`s.

Finding how we got the problem can be a little tricky, so lets try debugging. Click on the link from the unit test to jump to the test. Next, click on the margin next to the line number to set a break point (which appears as a red circle). A little above it, in the margin next to the test name is a green arrow. Click on it and select Debug. The program will pause execution at your breakpoint, giving you time to analyze what is going on. You will see a stack trace in one panel and a list of variables in another panel.

Above these panels, there are several buttons. Going from left to right, you have Step Over, Step Into, Step Into My Code, Step Out, and Run To Cursor. Step into the `from_string` method. We'll now be looking at a kind of scary list comprehension that involves a regular expression. We're trying to parse the string '10S' and the regex is supposed to break it into the composite cards. Let's check to see what actually gets passed into the `Card`'s `from_symbol` method. Use the Step Into My Code method to get there. If you use the ordinary Step Into, you will end up int the `re` library and will need to use Step Out to back up.

It looks like only '0S' is getting passed in rather than the full '10S'. It looks like the regex is only grabbing the second digit. We can add a `*` to the regex to have it match any number of digits. Do this and rerun the test.

It looks like the test is still failing. Now we have an unknown suit. See if you can find and fix this problem.

### 3.3 Poker Hands

Let's make sure that we can detect all of the hands correctly. It looks like the `is_straight` and `is_straight_flush` tests are failing. Looking at the tests, we see that both A-K-Q-J-10 and 5-4-3-2-A are supposed to count as straights, but Ace is listed as low in our Rank enum. In Poker, the Ace is always high so except for in the 5-4-3-2-A straight, so it is probably best to change the ordering. We will then have to detect this straight specifically in `is_straight`. If you hold Ctrl and click on `is_straight` in the test you can jump there. This method has an inner helper method that you will want to edit.

## 3.4 Ordering Hands

The rest of our 5 card tests deal with trying to decide which hand beats which, with `h1 > h2` if `h1` is a better hand than h2. Both `test_order` and `test_order_equal` seem to be having the same problem. I assumed that Python `dict`s are like your `HashMap` project, but I was wrong. This is an easy fix, but there are still other problems.

It looks like hands in the same category are not being compared correctly. You'll need to use the debugger to figure out what is going wrong.

## 3.5 7 Card Poker

Some variants of Poker use more cards and players then select their 5 best cards to score. See if you can get the PokerTests7 tests to pass.