CS320 Lab:
Dirty Dealing

Prof. Craig Partridge

Lab #6

A standard feature of classic American Western movies is the "card sharp", a gambler who cheats, often with a marked deck of cards or by hiding valuable cards up their sleeve.

In this lab, we ask you to implement a card shuffling routine that results in a deck that allows a card sharp to win more than they should.

1 Basics of a Crooked Deck

Removing certain cards from a deck of playing cards will, depending on the cards and the game being played, give an advantage to certain players.

For instance, in blackjack (also known as "21"), players try to beat the dealer's hand. Hands are valued by adding up the value of the cards, with face cards (Jacks, Queens, and Kings) being valued at 10, and aces being either 1 or 11. The player seeks a hand that has a higher value than the dealer's hand but does not exceed 21 points in value.

The usual odds in blackjack slightly favor the dealer. But you can change the odds by removing certain cards from the deck. For instance, removing all the 4 cards reduces the dealer's chance of winning by 0.52%. Removing an ace improves the dealer's chance of winning by 0.59%.

In poker, the situation is more complicated, but knowing that specific cards are removed from play, does give a player a tremendous advantage. For instance, if they know that all 5s are out, they will know not to try to complete straights (runs of sequential cards) that need to include a 5.

2 The Assignment

We are asking you to implement a crooked deck of cards, by shuffling specific cards to the bottom of the deck. Games rarely play all the cards in the deck as this gives an advantage to players who count cards, so putting cards at the bottom of the deck effectively removes them from play.

We are giving you the initial declaration of a class called `DirtyDeck` that implements a dirty deck of cards. We have implemented `__init__()`, `__contains__()`, `__len__()` `__iter__()` and

`__str__()`. `__init__()` takes an optional keyword, `hide=rank`, which instructs your code to put cards with the given rank at the bottom of the deck. Note that a rank of 10 only removes the 10 cards, not the Jack, Queen or King.

You are free to enhance or improve the course code for DirtyDeck, as long as all the provided methods continue to work (you cannot remove them).

`DirtyDeck` makes use of another class `PlayingCard`, which we are also providing, to represent playing cards. Do **not** change the PlayingCard class.

Your task is to implement:

- `shuffle()` updates the deck to contain a full deck of cards and does a Fischer-Yates shuffle. If `self.__hide__` is not `None`, you should push cards that have the rank in `__hide__` to the bottom of the otherwise properly shuffled deck; and
- `deal()` deals a single card from the top of the card deck. If the only 25% of the full deck remains, `deal()` raises a `ResourceWarning("low␣deck")` exception.

2.1 Requirements

Reminder that every assignment has a code portion, which is worth 60 points (40 for correctness, 20 for literate programming), and a reflection (worth 40 points).

2.1.1 Code Requirements

The requirements for the two methods are described above.

Note that we will test to ensure that your shuffle implementation returns properly mixed cards, and correctly places the specified cards at the end of the deck.

Python supports a shuffle routine in `random.shuffle()`. You are free to use it for debugging, but it may **not** be used in your solution.

2.1.2 The Reflective Essay

This lab, with its card cheating theme, seemed like an excellent time to let you produce another reflective essay as a short comic or few pages in graphic novel style. You can mix images and text in whatever way helps you tell us how you thought about solving this lab. The text boxes and text conversations in the comic should not exceed 250 words.

Reminder, don't let the imagery distract you from the message. As with all reflections we want to know how you worked your way to a solution in this lab. Specifically, we'd like to hear about/see:

- The challenges inherent in debugging code that contains randomness;
- How you dealt with those challenges in your coding process.

Unlike other assignments, you are free to use ChatGPT or other LLMs to generate your images. *The text in the images should still be yours.* You can also use your own art, or art done by a friend. In the event that mixing art and text doesn't work for you, you can submit a 250 word essay without artwork.

We will be grading your text and only grading artwork in terms of how it moves the narrative forward. That beautifully rendered western saloon? We won't grade it. That the comic uses shooting yourself in the foot as a metaphor for finishing the lab, we will grade according to how it improves the narrative.

Note that ChatGPT may limit the number of images you can generate each day, so start early.

2.2 Tips

Reminder of the following useful list operations:

- `append(x)` appends the value `x` to a list; and
- `pop(i)` removes and returns the value in position `i` from the list (if `i` is not specified, then `pop()` removes and returns the last item in list).

For random functions, you need to know:

- `random.randint(x, y)` returns a random integer between x and y inclusive; and
- `random.seed(x)` seeds the random number generator - you only need to do this for debugging, and the usual approach is to seed with 0 - this gives you the same random sequence over and over again. Don't forget to take out the call to `random.seed(0)` after debugging!