Just one problem on this *entirely optional and extra credit problem set*! Do not forget to adjust the README to indicate you have completed the assignment before your final commit!

Get Assignment link: https://classroom.github.com/a/qQvEgsmR

1. (6 points (bonus)) When learning to work with classes, I found it useful initially to work with objects which are very tactile and which I can easily envision defining an "object" for. To that end, in this problem you will be working with playing cards. I have already provided code to you which defines a `Card` class, which stores both a rank and a suit. Internally, both of these values are stored as numbers, but you will notice when you print a card to the screen that those numbers are converted to characters that better describe a typical playing card. Internally, a rank of 1 corresponds to an Ace, 2–10 corresponds to the number cards, 11 corresponds to a Jack, 12 to a Queen, and 13 to a King. Suits range from 0 to 3 and correspond to club, diamond, heart, spade. Getter functions are provided for both rank and suit that will return to you their numeric representation. Make sure you understand what is happening in this function before continuing.

   Your task in this problem is to define a new class to represent an entire *deck* of playing cards. Your standard deck of playing cards has 52 cards, with Ace through King in each of the 4 suits. You will be importing and using the `Card` class I already defined for you to represent the individual elements comprising your `Deck`. The primary (and only) attribute of your `Deck` will be a list of all the cards in the deck, and then you will have several methods to support the types of actions you would commonly want to make with a deck of cards. In particular, your `Deck` class should include:

   - A constructor which takes no extra arguments but which initializes a deck of the standard 52 playing cards whenever a new deck object is created. This can be done by appending the necessary 52 card objects to your deck list attribute. Loops are your friend here.
   - A `shuffle` method with will shuffle or randomize the order of the cards currently within your deck. (The random class has a shuffle method that should make this pretty straightforward.)
   - A `draw` method which takes one argument n and draws the "top" n cards from your deck, returning a list of those cards. Those cards should also be removed from the deck so that subsequent calls to `draw` will draw new cards.
   - A getter method called `get_deck` which returns a *copy* of the current list of cards in the deck.
   - An `__str__` or `__repr__` method which prints out a nice string representation of the cards still in the deck. Note that you may need to loop through and call `str` directly on the individual cards to have them display correctly as strings. Using the `join` string method may also be useful.

Creating and using some of these methods of your `Deck` class may result in output similar to below:

```
>>> D = Deck()
>>> print(D)
[A♣, 2♣, 3♣, 4♣, 5♣, 6♣, 7♣, 8♣, 9♣, T♣, J♣, Q♣, K♣, A♦, 2♦, 3♦, 4♦, 5♦, 6♦,
 7♦, 8♦, 9♦, T♦, J♦, Q♦, K♦, A♠, 2♠, 3♠, 4♠, 5♠, 6♠, 7♠, 8♠, 9♠, T♠, J♠, Q♠,
 K♠, A♥, 2♥, 3♥, 4♥, 5♥, 6♥, 7♥, 8♥, 9♥, T♥, J♥, Q♥, K♥]
>>> D.shuffle()
>>> print(D)
[K♦, 9♥, A♠, A♦, 7♥, Q♦, 2♦, 3♠, 2♠, J♥, K♠, Q♥, 7♣, J♣, 2♥, 8♥, 4♠, K♣, 7♦,
 8♠, Q♣, 3♥, 4♣, T♣, 8♦, T♠, 6♦, A♥, 5♥, A♣, 4♦, J♠, 5♠, 9♦, 9♣, 9♠, 8♣, 2♣,
 6♣, 7♠, 3♦, 4♥, K♥, 3♣, J♦, 5♣, 6♠, 5♦, 6♥, Q♠, T♥, T♦]
>>> print(D.draw(3))
[K♦, 9♥, A♠]
>>> print(D)
[A♦, 7♥, Q♦, 2♦, 3♠, 2♠, J♥, K♠, Q♥, 7♣, J♣, 2♥, 8♥, 4♠, K♣, 7♦, 8♠, Q♣, 3♥,
 4♣, T♣, 8♦, T♠, 6♦, A♥, 5♥, A♣, 4♦, J♠, 5♠, 9♦, 9♣, 9♠, 8♣, 2♣, 6♣, 7♠, 3♦,
 4♥, K♥, 3♣, J♦, 5♣, 6♠, 5♦, 6♥, Q♠, T♥, T♦]
>>> print(len(D.get_deck()))
49
```