

Decks of Cards

In this problem, you will be modeling a deck of playing cards.

I've provided a class called `Card`, so **you don't need to write that class**. The documentation for `Card` is as follows:

```
class Card
| A simple class that models a playing card
|
| Methods defined here:
|
| __init__(self, rank, suit)
|     rank - the rank of the Card - for example 2, 3, J, Q, A (string)
|     suit - the suit of the Card - for example Clubs, Spades (string)
|
| __str__(self)
|     Return a string representing this Card in the format:
|     [rank] of [suit]
```

I've also provided two Python `list` constants in **deck.py**: `CARD_RANKS` and `CARD_SUITS`.

Write a new class called `Deck` in the provided file **deck.py**, which will store a collection of `Cards`. `Deck` should have an initializer (constructor) that takes no parameters (besides `self`), and then adds each of the 52 possible `Cards` to its collection.

Write a `__str__` method for `Deck` that will return a string displaying each `Card` in the current top-to-bottom order of the `Deck`, for example:

```
Q of Hearts
3 of Spades
7 of Diamonds
...etc, where Q of Hearts is at the top of the Deck.
```

Write a method `deal` that takes a single `int` parameter `num_to_deal` that represents the number of `Cards` to deal off of the `Deck`. The method should remove `num_to_deal` cards from the top of the `Deck`, and print the string representation of the `Cards` in the order that they were removed. So, if a `Deck` `d` started with 52 `Cards`, and I call `d.deal(10)`, then there should only be 42 `Cards` remaining in the `Deck` `d`.

Write a method `shuffle` that will randomize the order of the `Cards` in the `Deck`. For this method, you'll want to use Python's `random` module (see: <http://docs.python.org/library/random.html>)

Write a method `cut`, which should take the first half of the `Deck` and swap it with the second half of the `Deck`. For a `Deck` with an odd number of `Cards`, have the remainder `Card` be part of the second half of the `Deck`. The `cut` method does not need to return anything.

For example, for a Deck `d` with these Cards:

```
7 of Hearts  
9 of Spades  
10 of Diamonds  
J of Spades  
A of Clubs
```

If these two statements were executed:

```
d.cut()  
print d
```

Then the following would be printed:

```
10 of Diamonds  
J of Spades  
A of Clubs  
7 of Hearts  
9 of Spades
```

Raise exceptions where appropriate. Feel free to define your own exception classes.

Your code will be evaluated based on correctness, style, design and documentation.