Mod 4 Homework - `DeckOfCards`

Implement a class called `DeckOfCards` that supports the following operations with (at worst) the noted running times.

`DeckOfCards` ADT

- `__init__(cards)`
  - *O*(n)
  - Initializes a new DeckOfCards with the ordered collection of strings `cards`
  - `cards[0]` should end up as the **bottom** card
  - `cards[-1]` should end up as the **top**
- `deal_top()`
  - *O*(1)
  - removes and returns the top card in the deck
- `deal_bottom()`
  - *O*(1)
  - removes and returns the bottom card in the deck
- `is_cheating(card)`
  - *O*(1)
  - returns `True` if `card` **has been dealt**, `False` otherwise

Cards are just strings. Any strings should work, but strings like *four of diamonds* or *queen of hearts* may be useful for keeping the card deck abstraction in mind while testing.

You will probably need to use a linked data structure to achieve *O*(1) for dealing to the top and bottom, but these do not support *O*(1) for `is_cheating` (i.e. `contains`). Dictionaries and sets support membership testing with *O*(1): think about how you can use both a linked data structure and a dictionary or set in the same class.

Special Behavior

- Raise `RuntimeError` if someone tries to deal from an empty deck.

External Modules

Do not use any imported modules (`math`, `collections`, ...) when implementing functionality. It is okay to use imported modules for testing; e.g. the `random` and `string` modules for generating random items.

It is okay to import modules you write yourself for this assignment; e.g. any data structures you write yourself.

Tests

You will be graded on your tests for this assignment.
You should have a file `TestDeckOfCards.py` that includes unit tests for all functionality in your class `DeckOfCards`. These tests can be a series of `assert` statements or you can use the `unittest` module, the choice is yours.

As a reminder for testing:

- test the *public interface* of your class. You should test `DeckOfCards` methods, but do not need to test any underlying data structures for this assignment. You *should* write tests for your data structures, but they will not be graded.
- Exceptions are part of the public interface: test that your code raises the correct Errors in the correct circumstances.
- Boolean functions like `is_cheating` should test `True` **and** `False` scenarios.
- Use comments (if using asserts) or clear function names (if using `unittest`) for your tests, and format tests so they are readable. **Particularly messy/difficult to decipher test suites will incur a 10 point penalty**.

The code below generates and shuffles 52 cards in the list `my_lst`. Feel free to use it.

```
import random
random.seed(658) # Change the seed to generate a new order of cards

my_lst = []
suits = ['hearts', 'diamonds', 'clubs', 'spades']
values = ['two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten',
'jack', 'queen', 'king', 'ace']
for suit in suits:
    for value in values:
        my_lst.append(value + ' of ' + suit)
random.shuffle(my_lst)
```

Examples

Any examples below are intended to be illustrative, not exhaustive. Your code may have bugs even if it behaves as below. Write your own tests, and think carefully about edge cases.

```
>>> from DeckOfCards import DeckOfCards
>>> cards = ["four of hearts", "queen of diamonds", "ace of spades"]
>>> deck = DeckOfCards(cards)
>>> deck.is_cheating("ace of spades")
False
>>> deck.deal_top()
'ace of spades'
>>> deck.is_cheating("ace of spades")
True
>>> deck.deal_bottom()
'four of hearts'
>>> deck.deal_bottom()
'queen of diamonds'
>>> deck.deal_top()
Traceback (most recent call last):
  ...
RuntimeError: Attempt to deal_top from empty DeckOfCards
>>>
```

Submitting

At a minimum, submit the following files:

- `DeckOfCards.py`
- `TestDeckOfCards.py`

Students must submit to Mimir **individually** by the due date (typically, the second Wednesday after this module opens at 11:59 pm EST) to receive credit.

Grading

- 30 - Functionality (includes Exceptions)
  - 10 - `deal_top`
  - 10 - `deal_bottom`
  - 10 - `is_cheating`
- 40 - Running Times
  - 10 - `init`
  - 10 - `deal_top`
  - 10 - `deal_bottom`
  - 10 - `is_cheating`
- 30 - Tests (includes Exceptions)
  - 10 - `deal_top`
  - 10 - `deal_bottom`
  - 10 - `is_cheating`

Feedback

If you have any feedback on this assignment, please leave it here.

We check this feedback regularly. It has resulted in:

- A simplified, clear **Submitting** section on all assignments
- A simplified, clear **Grading** section on all assignments
- Clearer instructions on several assignments (particularly in the recursion module)