CIS 211
Spring 2014

Project 3: Decks

Due Monday April 21
Upload via Blackboard by 11:00 P.M.

Goals

The next project is an extension of the Card project from last week. Here you will get a chance to define a new class that extends one of Python's built-in types. Along the way we will also explore random number generators, a topic described in the first part of the textbook.

Reading

Introduction to Computing Using Python, sections 6.4–6.5 and 8.1–8.5, plus resources available on the class web site.

Programming Projects

1. Create a new file named `Deck.py` and add the definition of a class named Deck. Your new class should be derived from Python's `list` class. An instance of this class will be a list of 52 Card objects (which are defined in `Card.py`).

   When the constructor is called it should return a list of all 52 cards in order from 2♣ up through A♠.

Important*: don't define a class that has a list as an attribute. Define a class that is a new type of list. See the lecture notes on the difference between "has-a" and "is-a"*

   Define three methods for your class:
   • `shuffle()` should rearrange the cards into a new random permutation
   • `deal(n)` should remove the first `n` cards from the deck and return them in a list
   • `restore(a)` should add the cards in list `a` to the end of the deck

```
>>> d = Deck()

>>> len(d)
52

>>> type(d[0])
<class 'Card.Card'>

>>> d
[2♣, 3♣, 4♣, ... Q♠, K♠, A♠]

>>> d.shuffle()
>>> d
[Q♣, A♦, 7♦, 9♦, 8♦, 3♠, 8♠, ... 5♣, 9♣, K♦]

>>> h = d.deal(5)
>>> h
[Q♣, A♦, 7♦, 9♦, 8♦]

>>> d
[3♠, 8♠, ... 5♣, 9♣, K♦]

>>> len(d)
47
```

```
>>> d.restore(h)
>>> d
[3♠, 8♠, ... 9♣, K♦, Q♣, A♦, 7♦, 9♦, 8♦]

>>> len(d)
52

>>> d.sort()
>>> d
[2♣, 3♣, 4♣, ... Q♠, K♠, A♠]
```

Make sure you understand why the last expression above worked. Why are you able to sort a deck of cards even though you did not define a `sort` method in your class?

2. Define a new class named `PinochleDeck` that has `Deck` as its base class. An instance of this class should be a list of 48 cards. The game of Pinochle uses only 9s and above, and there are two copies of each card. A new deck should be sorted.

To make a random permutation of a list you can use a function named `shuffle` from Python's `random` library (see p. 197 in the textbook).

```
>>> d = PinochleDeck()
>>> d
[9♣, 9♣, 10♣, 10♣, ... Q♠, Q♠, K♠, K♠, A♠, A♠]
>>> d.shuffle()
>>> h = d.deal(12)
>>> h.sort()
>>> h
[A♣, 9♦, 10♦, J♦, J♦, 9♥, A♥, A♥, 9♠, 10♠, K♠, A♠]
```

Card.pyc

Near the front of your `Deck.py` file you need to have a statement that imports the Card class:

```
from Card import *
```

You can either use the Card class you wrote for the last project, or you can download a compiled binary of the class written by the instructors. This file, called `Card.pyc` (note the 'c' on the end of the name), is available on Piazza and the CIS web page. If you want to use this file download it and save it in the same directory as your `Deck.py` file.

Extra Credit Ideas

- Since the `Deck` class is a subclass of Python's `list` class users can do anything to a `Deck` that they can do to a list, including some things they shouldn't. For example, it's easy to attach a string to the end of a deck of cards:

```
>>> d
[2♣, 3♣, 4♣, ... Q♠, K♠, A♠]
>>> d.append('howdy')
>>> len(d)
53
>>> d.shuffle()
>>> d
[5♣, 'howdy', 2♣, J♠, ... 8♥, J♣, Q♦]
```

A function like `points` that expects a list of `Card` objects isn't going to like that.

Figure out what sorts of things you don't want to happen with decks of cards and add code to your class definition that raises an error message when the operation is invoked.

- Have the `restore` function verify its argument is a list of Card objects.
- Write your own version of a method that makes a random permutation instead of using `random.shuffle`.
- Add a second (optional) argument to the `deal` method that specifies the number of hands to create. For example, `deal(5,2)` will make 2 hands with 5 cards each, where the cards are dealt in the traditional fashion, *i.e.* alternate cards to each hand.

Testing

We will test your classes and functions by running the tests in a file named `test_Decks.py`. You can download the file and run the tests yourself before you submit your project. This is what you will see in your shell window when your code passes all the tests:

```
$ python3 -m unittest test_Decks.py
....
----------------------------------------------------------
Ran 4 tests in 0.001s

OK
```

What to Turn In

Documentation Write a short description (two or three paragraphs total) of what you did for this project. This is your chance to tell the graders about anything you think you did well. The documentation should be in a file named `writeup` with an extension that identifies the file format (.doc for Microsoft Word, .pdf for Adobe PDF, .txt for plain text, .rtf for rich text format).

Code Save both your class definitions in a single file named `Deck.py`.

Create a package (tar or zip format) that includes your writeup and code and upload the package via Blackboard.