CS 1110, LAB 6: LISTS: CARDS AND POKER HANDS

http://www.cs.cornell.edu/courses/cs1110/2017sp/labs/lab06.pdf

First Name: ____________ **Last Name**: ____________ **NetID**: ____________

Getting Credit: Deadline: the first 10 minutes of (your) lab **<u>two weeks</u> from now (Tue Mar 21 or Wed 22)**, due to the prelim. But *since this lab covers material that will be on the prelim*, don't wait until after the prelim to start it!

The checking-off procedure is the same as before.[1]

1. List Operations

Complete the following tables as usual. Box-and-folder diagrams can help. Remember that list slicing produces a new list.

Commands	**Expected Output**	**Correct?**
`lablist=['Cats','rule','!','?','!']`	(n/a)	(n/a)

`lablist.remove('!')`
`print lablist`		

`lablist.remove('C')`		
`print lablist.insert(2, '!')`		
`print lablist`		
`print lablist.index('!')`		

`copy1 = lablist`
`copy2 = lablist[:]`
`lablist[0] = 'Dogs'`
`print copy1`		

`print copy2`		

`copy2.append(' As do bats.')`
`print copy2`		

`print len(copy2)`		

Course authors: E. Andersen, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White

[1] In case you've forgotten, here's a reminder: Show this handout and/or your code to a staff member either (a) during your lab 06 session, (b) in consulting hours listed at http://www.cs.cornell.edu/courses/cs1110/2017sp/about/staff.php up to the day **before** your next scheduled lab section, or (c) in the first 10 minutes of (your) next scheduled lab (Tue Mar 21 or Wed 22). Beyond that time, the staff have been instructed not to give you credit. Labs are graded on effort, not correctness. We just want to see that you tried all the exercises, and to clarify any misunderstandings or questions you have.

2. The Module cardstuff, and Some Common Error Messages

Create a new directory on your hard drive for this lab's files. Then, download into that new directory the files you need for lab 06; get them packaged in a single zip file from the Labs section of the course web page, http://www.cs.cornell.edu/courses/cs1110/2017sp/labs .

You'll be working with the Card type. The definition and details are given in cardstuff.py, but here's all you need to know:

- Cards have two attributes, a `suit` and a `rank`, both ints. Using ints makes it easy to compare card values; for instance, ♠K may be "worth" more than ♢2.
- To translate our ints into human-readable card names, we have two lists, defined in module cardstuff named `SUIT_NAMES` and `RANK_NAMES`; these lists serve as "translation tables".

To see how this works, go to the directory you downloaded the lab files into, and begin a Python interactive session. Then, for each line below that isn't a comment, enter it.

```
import cardstuff
print cardstuff.SUIT_NAMES
# the next commands look at element 0 and 1 of the list cardstuff.SUIT_NAMES
print cardstuff.SUIT_NAMES[0]
print cardstuff.SUIT_NAMES[1]
```

So, 0 represents Clubs, 1 represents Diamonds, and in general, int `s` represents the suit given by `cardstuff.SUIT_NAMES[s]` when `s` is one of 0, 1, 2, 3.

Similarly, try these:

```
print cardstuff.RANK_NAMES
# the next commands look at some elements of the list cardstuff.RANK_NAMES
print cardstuff.RANK_NAMES[1]
print cardstuff.RANK_NAMES[2]
print cardstuff.RANK_NAMES[11]
```

Thus, 11 represents a Jack, 2 represents a 2, 1 represents an Ace, and in general, int `r` represents the rank given by `cardstuff.RANK_NAMES[r]`.[2]

What error message do you get if you enter `print cardstuff.RANK_NAMES[14]` in Python interactive mode, and why? Write your answer below.

Now, practice creating and printing some cards. Our initializer for Cards has two[3] parameters, `s` and `r`, the ints for the suit and rank of the new card, respectively.

Try the following:

[2]Small detail that you can skip if you don't care: The value `None` in element 0 of `cardstuff.RANK_NAMES` is an encoding trick that lets us talk about ranks 1 through 13, not 0 through 12. Wouldn't it have been weird if we had said that a 1 represents a 2?

[3]We'll explain later in the course why, if you look at the actual class definition in the code, the definition for the initializer method actually has *three* parameters. For now, don't worry about it.

```
c1 = cardstuff.Card(0,13)
print c1.rank, c1.suit
print cardstuff.RANK_NAMES[c1.rank] + ' of ' + cardstuff.SUIT_NAMES[c1.suit]
```

The last line you entered above is what is used, employing a little Python "method"-ology you haven't learned yet, inside our definition of the Card class to cause the `print` function to give nice output for cards. To see this, try the following:

```
print c1
```

You should see the exact same output.

What error message do you get if you enter `print cardstuff.RANK_NAMES[rank]` (that is, don't have the `c1` in the square brackets), and why?

Finally, let's practice creating a new list of two new cards. Here's one way to do it: try the following:

```
cardlist = [cardstuff.Card(1,4), cardstuff.Card(2,11)]
#check that there are exactly two cards
len(cardlist)
print cardlist[0]
print cardlist[1]
```

3. Writing Functions for Card Decks and Hands

3.1. print_cards.

3.1. print_cards. We're going to be using lists of Cards to represent card decks and card hands. In doing so, we'd like to be able to look at the contents of a list of cards. Try this:

```
print cardlist
```

Not too informative (to humans), right?

So, instead, we've almost completed for you a function print_cards, which looks like this:

```
def print_cards(clist):

    """Print cards in list clist, which is a (possibly empty) list of Cards."""
    for c in clist:

        print c
```

The line `for c in clist:` is the beginning of a *for-loop*, a construct we briefly mention in lecture 11. What it means is that the variable `c` takes on each value in `clist` in turn; for each such value, the body of code indented below the line is executed. So, try it out (still in Python Interactive Mode) to see what it does:

```
cardstuff.print_cards(cardlist) # prettier than before!
```

What would happen if instead of `print c`, the line in the for-loop said read `print "card"`, and why?

3.2. **draw_poker_hand.** OK, now for the exciting (?) part: you can try your hand (ha) at drawing poker hands from decks, by implementing cardstuff.draw_poker_hand. Take a look at its specification. There are going to be several steps involved, which we're going to break up into helper functions.

You'll want to make use of standard list operations to implement your helper functions. Look at section 5.1 here: `http://docs.python.org/2/tutorial/datastructures.html` or the relevant lecture handouts.

3.2.1. *draw.* First, we'll want to be able to randomly draw a single card from a given deck supplied as an argument, and return that card, removing it from the deck.

Look at the code skeleton and comments for `draw` that we've provided you. We've already written a line that choses a random index `i` for the card list given as argument. So all you have to do is add a line or two that (a) removes the element of the argument list at index `i`, and (b) returns that element. *Use the list method* `pop`*, documented at the webpage mentioned above.*

Test your code in the command shell:

(1) Exit python, to bring you to the command shell.
(2) Enter `python cardstufftest.py`

... and debug as appropriate until your code passes the test of cardstufftest.test_draw.

3.2.2. *poker_compare.* Now let's deal with the fact that cardstuff.draw_poker_hand wants an ordering on the list it returns. This ordering will be determined by the function poker_compare.

Using if-statements and the like, implement `cardstuff.poker_compare` *according to its specification, which explains what you must do.* You'll probably need to write some non-trivial boolean expressions inside your if statements.

Run the unit test cardstufftest.py to get a quick diagnostic on whether your implementation is correct, by seeing if you get past cardstufftest.test_compare().

3.2.3. *finish draw_poker_hand using your helper functions.* OK, now for the pièce de résistance: complete function cardstuff.draw_poker_hand. Your code should include five calls to draw_card (unless you feel like figuring out that business about for-loops) — each time appending the item returned by draw to the temporary list `output`. You'll also want to *sort* the output list, and to use some list method to *reverse* the sort you used. There's a hint or two regarding syntax given in the comments in the body of the code.

You can test by running cardstufftest.py repeatedly: you should see a random poker hand, appropriately sorted, printed out near the end each time, plus a small amount of diagnostic information.