CS 1110, LAB 08: CLASSES

http://www.cs.cornell.edu/courses/cs1110/2013sp/labs/lab08/lab08.pdf

L. LEE AND S. MARSCHNER

First Name: _____ **Last Name**: _____ **NetID**: _____

Files to download. Create a *new* directory on your hard drive and download the following modules into that directory. (You can get them all bundled in a single zip file at http://www.cs.cornell.edu/courses/cs1110/2013sp/labs/lab08/lab08files.zip)

`card.py` (http://www.cs.cornell.edu/courses/cs1110/2013sp/labs/lab08/card.py)
`blackjack.py` (http://www.cs.cornell.edu/courses/cs1110/2013sp/labs/lab08/blackjack.py)
`blackjacktest.py` (http://www.cs.cornell.edu/courses/cs1110/2013sp/labs/lab08/blackjacktest.py)
`cunittest2.py` (http://www.cs.cornell.edu/courses/cs1110/2013sp/labs/lab08/cunittest2.py)

Getting credit for lab completion. When done, show your code and/or this handout to a staff member, who will ask you a few questions to see that you understood the material and then swipe your ID card to record your success. As always, if you do not finish during the lab, you have until the beginning of lab next time it meets — **in this case, March 26-27 because of Spring Break** — to finish it: show it to your lab TA at the beginning of that lab. But you should always do your best to finish during lab hours. Remember that labs are graded on effort, not correctness.

---

1. Your Mission: Writing a Version of Blackjack

You will finish a class definition for `Blackjack` that a casino could use to run multiple blackjack games simultaneously.

A player wins at blackjack by ending with a hand that has more points than the dealer's, but not more than 21 points — if someone exceeds 21 points, they are said to have "gone bust" and immediately lose.

Points come from the ranks of the cards in a hand: 10 points for each face card (Jack, Queen, or King), 11 points for an ace, and the rank of the card otherwise, e.g., a 4 of anything is 4 points.

Play begins with two cards being dealt to the player and one card to the dealer. All cards in each hand are always visible to all participants. The player can opt to "hit" — get an additional card from the deck — or "stay" — turn over play to the dealer. If the player eventually stays without going bust, then the dealer draws cards until they go bust or decide to stop.

Your bonus for completing the lab is the ability to relax and play a few rounds of the game yourself; here is a sample interaction transcript for when everything's working:

```
[llee: lab08] python blackjack.py
Face cards are 10 points, aces are 11 points, all other cards are at face value

Your hand:
8 of Spades
6 of Clubs
```

```
Dealer's hand:
9 of Spades

Type h for new card, s to stop: h
You drew the 6 of Spades
Type h for new card, s to stop: s
Dealer drew the 3 of Spades
Dealer drew the 4 of Spades
Dealer drew the 8 of Hearts
Dealer went bust, you win!
The final scores were player: 20; dealer: 24
```

2. The *New* Module `card`

We'll be working with a slightly re-defined and re-written version of the Card class from Lab 5.[1] You do not need to change anything in the `card` module. However, since it contains a class definition, albeit one whose methods all have "magic names" (unlike in `Blackjack`), you may find it useful to look at as you complete `Blackjack`.

3. Writing the blackjack class definitions

Proceed in an iterative fashion: for each step outlined below,

(1) Read what is expected of you, given in the directions in this handout and the spec of the function under consideration.
(2) Look at the appropriate test cases in module `blackjacktest`. This will help your understanding of what is required.
(3) Only then, code up what needs to be coded. Remove lines that say "pass" or comments that say "Implement me" or the like.
(4) Test your code using our unit test `blackjacktest`. You do not need to add test cases to it.

Make sure each function passes its test cases *before* moving on to implement the next function. This is important because many of the functions here build on earlier ones.

3.1. **Fix the function headers.** There is something wrong in at least one of the headers of the methods for class `Blackjack`. You can tell by running the test module `blackjacktest` in the command shell;[2] open a command shell in the directory you've put the lab files in, and type `python blackjacktest.py`.

What error do you get, and how should you fix the error? Write your answers below, and *then fix* all *method headers that require this correction.*

[1] Sustainable computing: reduce and re-use code. (Properly attributing it, of course.) The major changes are that we've removed poker-specific functions and changed the documentation to satisfy the templates we've settled on in lecture.

Why did we originally have poker functions in the module `card`? At the time of Lab 5, it was convenient to put all the required code in one place so that students wouldn't have too many files to deal with, but now we've gotten to the point where exemplifying better file organization is appropriate.

[2] You're going to need to use the Command Shell in order to play the game, so you might as well get used to it now. See the relevant course web page for reference: http://www.cs.cornell.edu/courses/cs1110/2013sp/materials/command.php

3.2. **Implement and test `__init__`.** You'll probably want to make use of standard list operations; for reference, look at section 5.1 here: http://docs.python.org/2/tutorial/datastructures.html or the relevant lecture handouts. Our solution is three lines long. Write yours here:

3.3. **Read over but don't change `_score`.** Note the leading underscore; this is meant to be a private helper function for the class (and for you).

3.4. **Implement `dscore` and `pscore`.** Use the private helper function. The syntax for getting this right might take a little getting used to, so write down your code for `dscore` here for a staff member to take a look at:

3.5. **Implement and test `pbust` and `dbust`.** Use `dscore` and `pscore`.

3.6. **Implement and test `__str__`.** . Note that it's "higher up" in the file, just after `__init__`, as is conventional. Use `dscore` and `pscore`.

3.7. **Just for fun: play some blackjack!** Open a command shell in the directory you've put the lab files in, and type `python blackjack.py`. Follow the directions on the screen: 'h' is for 'hit', and 's' is for stay. Our dealer is following a common house protocol; you can read the code for `play_a_game` to find out what it is.[3] Good luck, and may the odds be ever in your favor!

[3]You may also note that this function uses two *while-loops*, one nested in another. We'll learn about these later in the course, but you can observe what they do here.