Due Sunday Mar 27 11:59 PM EST

# CSE 4256

Dr. Alan Weide

Spring Term 2022

# Homework 8

## Code Quality

Any submitted homework that raises any errors when I invoke Python with the submitted `main.py` file will receive no credit. Additionally, any submitted homework that contains any failing test cases in the submitted `tests.py` will receive no credit. As discussed in class, I will enforce the *spirit* of this rule and not the *letter* of this rule. In particular, simply deleting all code from `main.py` or not having any test cases in `tests.py` will result in no credit being given.

Beyond that, while you do not need to correctly answer *every* question in order to earn a grade of "satisfactory" on assignments in this course, please at least *attempt* every question. You may be surprised how quickly things click once you start working on them!

## Assignment

This assignment should be completed as a collection of several Python source (`.py`) files, one of which should be named `tests.py` and should contain test cases written with either the `unittest` or `pytest` framework.

(1) In the slides, it was noted that every function in the `random` module can be implemented by transforming the result of `random()` in some way. Use this fact to implement the methods `uniform(a, b)`, `randRange(start, stop)`, and `choice(seq)`. The only function from the `random` module you may use is `random()`, although you may use the other methods you have implemented.

(2) Use the `namedtuple` function to create a type called `Card` that represents a standard playing card identifiable by its `suit` (one of `'CLUBS'`, `'SPADES'`, `'HEARTS'`, or `'DIAMONDS'`) and `rank` (one of the numbers 1–13, with 1 to be interpreted as "Ace" and 11–13 as "Jack", "Queen", and "King", in order).

(3) Implement the function `std_card_deck()` that returns a `deque` of the standard 52 playing cards. The returned deque should be ordered as follows: [(CLUBS, 1), ..., (CLUBS, 13), (SPADES, 1), ..., (SPADES, 13), (HEARTS, 1), ..., (HEARTS, 13), (DIAMONDS, 1), ..., (DIAMONDS, 13)].

(4) Write a function called `riffle_shuffle(deck)` that simulates a "riffle shuffle" of a deck of cards, treating the right end of `deck` as the "top". Use the following algorithm:

- Split the deck into `d1` and `d2`, of nearly-equal lengths, leaving `deck` empty
- As long as there are cards in both decks, choose the bottom card from either `d1` or `d2`, *at random*, and place it at the top of `deck`
- When one of the decks runs out, place the rest of the cards from the other deck on the top of `deck`

(5) Note that a single riffle shuffle is not a particularly effective shuffling technique. Implement the function `mix_deck(deck)` to "fully" shuffle the deck. The only functions you may use are built-in functions and those that you have implemented in this assignment. Your function should be "efficient" in that it should run in $O(\texttt{len}(\texttt{deck}))$ time, keeping in mind the relative runtimes of the various `deque` methods.

(6) Implement the function `deal(deck, n_players)` which deals the cards in `deck` into `n_players` hands of approximately-equal size.

(7) In Homework 4, you were asked to count the number of times each letter appeared in a string, and to report the letter that appeared most often. Rewrite your solution to those problems using the `Counter` type and the built-in **`filter`** function (and, if you want, the **`map`** function).

(8) In a file called `tests.py`, write a reasonably complete test suite for the functions you've implemented using either the `unittest` or `pytest` framework. All of your tests should be run when the command `python3 -m unittest` or `pytest` is executed at the command line from your project directory.

## Challenge Activities

Some homeworks (such as this one) will have additional challenge activities. These activities **do not contribute to your grade**, but they are problems that I find interesting or challenging.

(9) Implement some of the non-uniform real-valued distribution functions from the `random` module, such as `expovariate(lambd)` or `normalvariate(mu, sigma)`. The equations for these distributions (and some pseudocode for generating them!) are readily available online. As in problem 1, the only function from the `random` module you may use is `random()`.

## Submission

To submit this assignment, upload a **`.zip`** file named "lastnamefirstname.zip" (obviously replacing lastname with your own last name and firstname with your own first name) containing all of your Python files to the "Homework 8" assignment on Carmen. As always, be sure to note all group members who contributed to the assignment and what those contributions were.