CISC106 Fall 2011 Lab06

- This lab an all subsequent labs will be due Sunday at 11:55 PM EDT on Sakai.
- The preparation problems below are to develop your understanding without creating extra work for you or the TA; these problems will not be graded. Be sure to read and understand them - they will help with the problems you must submit for grading
- Review the code examples from your notes in class.
- You may work with one or two other people on your lab (max size is three!). These people must be in your same lab section. If you do, **one** of you should be designated to submit the assignment on Sakai. **All of your names** should appear on code that you develop together[1].
- Whom do you think deducts more points: a happy TA, or a frustrated TA? Make your work easy to read! It isn't just good software engineering, it is good for your grade!
- EVERY python program/function must include header, doc string that contains a human-readable desciption of what the function does, and must be followed by a good series of tests, as discussed in class. Always test boundaries. Do not test erroneous input (e.g. a factorial function does not need to correctly handle strings).
- EVERY .py file must have a comment line at the very top containing your name(s), lab section, and a brief description of what the file is.
- Write the tests first! Real software engineers do this for very good reasons - so should you!

Preparation (do not submit for grading)

1. Start up the python interpreter and `import random`. Call `random.seed`(490177). Then call `random.randint`$(0, 1000000)$ several times and see what numbers you get.
2. Call `random.seed`(490177) again. Then go back to calling `random.randint`$(0, 1000000)$. What do you notice? What happens if you call `random.seed` with another seed besides 490177?

Programs (to be graded)

1. The *Monty Hall Problem* is a probability puzzle named after the host of the 1960s/70s game show *Let's Make a Deal*. The problem statement is as such:[2]

   > You're a contestant on a gameshow. Before you are three doors - aptly labeled 1, 2 and 3. The show's host explains to you that you that behind *one* of these doors is a car, and that a goat resides behind each of the other two. You must choose one of these three doors, at which point the host will open one of the other two doors revealing its contents. You will then be given the option to switch to other

[1] If you would like to work with someone but don't know whom, your TA may be able to help connect you to other students looking for lab partners.

[2] For more info on the Monty Hall Problem, visit the wikipedia article: `http://en.wikipedia.org/wiki/Monty_Hall_problem`

un-opened door. Whether or not you choose to switch, if the door you choose contains the car, you win that car. If it contains a goat, you win nothing. You choose your door, and then the hosts opens one of the other two doors to reveal a goat. It's now time to decide what to do next - should you stay or should you go? Which choice, if either, gives you a better chance of winning the car?

Your job is to write a simple simulation of one play of the game. In the simulation, you should randomly generate the contestant's choice of doors *and* the door containing the car. You can do this by importing the `random` package and then calling `random.randint`$(1, 3)$ - which will generate a random number between 1 and 3 (inclusive.) You should then consider what the win conditions are for a contestant who chooses to switch and for one who chooses *not* to switch. When thinking about these conditions, keep in mind the fact that *the host is aware of which door contains the car* when he performs the reveal.

2. Next, you should write a function `run_test`, which takes a number *iterations* and a boolean specifying whether or not to switch doors. It will then run your simulation *iterations* times, counting the number of times the contestant wins the car. After running the simulations, it should display the raw number of wins as well as the win ratio. Try calling `run_test` several times, varying whether or not to switch, to see which strategy wins you the car more often.[3]

You should implemnt your simulation in a file called lab06.py. Submit that file along with any other docs required by your TA on Sakai.

[3]One final note - be sure to intialize the random number generator before calling `run_test` by calling `random.seed()`