LARSON—MATH 255–CLASSROOM WORKSHEET 33
Python Classes & the Birthday Problem

1. (a) Start the Chrome browser.
   (b) Go to `http://cocalc.com`
   (c) Login using **your VCU email address** .
   (d) Click on our class Project.
   (e) Click "New", then "Worksheets", then call it **c33**.
   (f) For each problem number, label it in the Sage cell where the work is. So for Problem 2, the first line of the cell should be `#Problem 2`.

Every integer in Python is an *instance* of the Python `int` class. Included in that class are built-in functions (called *methods*) that work only on `ints`. Every Integer in Sage is an *instance* of the Sage `Integer` class. Included in that class are built-in functions (called *methods*) that work only on `Integers`; one example is the `.is_prime()` method. Another example is the Sage *Graph* class: every graph is an instance of that class. The methods here include `.size()` and `.order()`.

Our Own Class. In order to have a slightly deeper understanding of Python and Sage classes (and object-oriented programming) we defined our own Sage class. We'll design a general class of Dungeons and Dragons character, sample concrete character objects, methods that can be accessed by any character objects, and functions that can be used on the characters.

```
class Character():
    def __init__(self, name):
        self.name = name
        self.intelligence=randint(1,10)
        self.health=randint(1,10)
        self.strength=randint(1,10)
        self.fortitude=randint(1,10)
        self.point = randint(1,10)
    def hello(self):
        print "Hello world! I am {}.".format(self.name)
    def status(self):
        print("My intelligence is {}".format(self.intelligence))
        print("My health is {}" .format(self.health))
        print("My strength is {}" .format(self.strength))
        print("My fortitude is {}" .format(self.fortitude))
        print("My points are {}".format(self.points))
    def change_intelligence(self,amount):
        new = self.intelligence + amount
        if new < 1:
            self.intelligence = 1
        elif new > 10:
            self.intelligence = 10
        else:
            self.intelligence = new
    def change_points(self, amount):
        self.points = self.points + amount
```

2. Evaluate. We must create new characters in order to use the newly defined abilities. Try `sam=Character("Samwise")`. Then try `sam.hello()`

3. This creates an *object* of the `Character` type. The name form the program environment's point of view is "sam". The `.name` built-in to the class is "Samwise"—but that's not useable for our programs—this is data that's stored as part of the created object.

4. Now define the following function.

```
def drink_potion(character):
    if random() < 0.5:
        character.change_intelligence(3)
        print("I feel smarter!")
    else:
        character.change_intelligence(-3)
        print("Uh oh!")
```

   Try `sam.status()`, then `drink_potion(sam)`, then `sam.status()` again.

5. Then try `sam.status()`.

6. Now make your own character with your own name—and check all of these things that we did for "sam".

   You see these *classes*, *objects* and *methods* can get very interesting!

   The Birthday Problem.

7. (Guess) **How many students do we need in a classroom so that there is a better than 50% chance that at least one pair of them have the same birthday (Month & Day)?**

8. What could you code to investigate this problem?

   Getting your classwork recorded

   When you are done, before you leave class...

   (a) Click the "Make pdf" (Adobe symbol) icon and make a pdf of this worksheet. (If Cocalc hangs, click the printer icon, then "Open", then print or make a pdf using your browser).

   (b) Send me an email with an informative header like "Math 255—c33 worksheet attached" (so that it will be properly recorded).

   (c) Remember to attach today's classroom worksheet!