LARSON—MATH 255–CLASSROOM WORKSHEET 26
Python Classes

1. (a) Start the Chrome browser.
   (b) Go to `http://cocalc.com`
   (c) You should see an existing Project for our class. Click on that.
   (d) Click "New", then "Sage Worksheet", then call it **c26**.
   (e) For each problem number, label it in theSAGE cell where the work is. So for Problem 1, the first line of the cell should be `#Problem 1`.

Problems

2. (**Estimating** π) We can randomly place points in a 1-by-1 square. We also know the equation of a circle centered in this square. How can we use those ideas to estimate a value for π?!?!

3. (**The Birthday Problem**) 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)?. What could you code to investigate this problem?

Python Classes!

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 will define 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.

4. The following code defines a class called `Character`, together with a single method which *constructs* new Characters. What we have in mind is a thing (think of it as a person) that has a *name*.

```
class Character():
    def __init__(self, name):
        self.name = name
```

Now we can create characters. Try `c1=Character("John")` to create a character $c1$ with the name "John". We can create as many as we want. Try `c2=Character("Jenn")`.

5. Our characters can't do anything yet. So let's add a method so they can introduce themselves.

```
class Character():
    def __init__(self, name):
        self.name = name
    def hello(self):
        print("Hello world! I am {}.".format(self.name))
```

Evaluate. We must create new characters in order to use the newly defined abilities. Try `c3=Character("Bilbo")`. Then try `c3.hello()`

6. Now lets add attributes to our Characters: intelligence, health, strength, and fortitude. We will randomly initialize these as integers from 1 to 10

```
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)
    def hello(self):
        print("Hello world! I am {}.".format(self.name))
```

7. How can we learn about their attributes? . We would also like to be able to get a status report on these values. So we will add a `status()` method.

```
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.points = 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))
```

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

9. Perhaps we should award our characters "points" in certain situations? We can add a `points` value when we initialize the character. And also add it to our status reports. And there should be a way to change the number of points. So lets add a `change_points()` method to the `Character` class. We'll have to update the `status` method at the same time.

   ```
   def change_points(self, amount):
       self.points = self.points + amount
   ```

   Evaluate. Let `c6=Character("LittleJohn")`. Then try `c6.status()`.

10. Our characters may have to fight trolls. Define the following function.

    ```
    def fight_troll(character):
        if character.health > 5 and character.strength > 5:
            character.change_points(5)
            print("I have defeated the troll!")
        elif character.health < 4 or character.strength < 4:
            character.change_points(-5)
            print("You have defeated me this time!")
        else:
            print("Run away!")
    ```

11. Oh oh. LittleJohn has encountered a troll. Lets see what happens. Evaluate `fight_troll(c6)`. Then check his status with `c6.status()`.

12. Things may happen to our characters. Gandalf may drink a potion that effects his intelligence. Let's add a method so we can change a character's initial intelligence. We must be careful never to leave the range of 1 to 10.

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

13. Then try `sam.status()`, then `drink_potion(sam)`, then `sam.status()` again.

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

Getting your classwork recorded

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

1. 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).
2. Send me an email with an informative header like "Math 255 - c26 worksheet attached" (so that it will be properly recorded).
3. Remember to attach today's classroom worksheet!