LARSON—MATH 353–CLASSROOM WORKSHEET 06
Getting Started—Monty Hall Paradox.

1. Log in to CoCalc.

   (a) Start the Chrome browser.

   (b) Go to `https://cocalc.com`

   (c) Login (**your VCU email address** is probably your username).

   (d) You should see an existing Project for our class. Click on that.

   (e) Click "New", then "Worksheets", then call it **c06**.

2. **A problem to think about: the Monty Hall Paradox**. There are 3 doors to choose from (on say a game show). There is a car behind one door and a goat behind each of the other two. You choose one door (say, Door 1). Maybe it has the car behind it (or maybe not)? At least one of the other doors has a goat behind it—and one of these doors is opened (say, Door 2). You are then given an opportunity to switch your choice of door (in this example, you could switch from Door 1 to Door 3). Should you?

   How can you investigate this question experimentally? **What can you do to extend our classroom investigation?**

   In order to do sophisticated calculations, or to allow for multiple inputs, you will need to write *programs*. The classic "hello world!" program is the first example. It includes a `print` statement. Other program features, in almost any language, include *conditional statements* (if..then..) and *loops*.

3. Type in the following program and evaluate.

```
def write_string(string_name):
    print(string_name)
```

   Now type `write_string("hello world!")` and evaluate.

4. Type in the following function definition and evaluate.

```
def absolute(x):
    if x>=0:
        return x
    else:
        return -x
```

   Now test it. Evaluate `absolute(4)`, `absolute(-4)`, etc.

5. Now *use* the program you just wrote in another program. Evaluate and test the following.

```
def abs_plus_five(x):
    return absolute(x)+5
```

6. You don't have to add five, you can add *any* number by adding a *parameter*.

```
def abs_plus(x,y):
    return absolute(x)+y
```

7. Now test it. Try `abs_plus(4,5)`, `abs_plus(-4,5)`, `abs_plus(-4,23)`, etc.

   A *string* is a sequence of *characters* (letters, numerals, symbols, etc). If you put a sequence of characters between quotes, you are telling Sage to treat what's between the quotes as a string (instead of as a *keyword*). Strings can be manipulated, and have places that can be filled in.

8. Type and evaluate `print 'This string has {}'.format('17 characters')`. Now try replacing '17 characters' with any other string.

9. Type and evaluate the following program.

```
def superstring(x):
    print 'This string has {}'.format(x)
```

10. Now test your function. Type and evaluate `superstring('black letters')`.

11. **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 353—c06 worksheet attached" (so that it will be properly recorded).

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