1. Log in to your Cocalc account.
   
   (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 Worksheets”; call it c03.
   (e) For each problem number, label it in the Sage cell where the work is. So for Problem 1, the first line of the cell should be #Problem 1.

2. Find $\sin \frac{2\pi}{3}$ to 30 digits of precision. (You may have to look back at your old worksheets or code. That will be common until your skills become locked-in).

3. Define variables $a$, $b$ and $c$. Solve $ax^2 + bx + c = 0$. What does the output mean?

4. How do you use solve to solve systems of more than one equation? Use the Help by evaluating solve? and look at examples with more than one equation.

5. Consider the following system. Sketch the graphs of these lines on the same coordinate system (by hand and then with plot), then solve to get the exact point of intersection.
   
   \[
   \begin{cases} 
   2x + y = 20 \\
   -x + y = 0 
   \end{cases}
   \]

6. Consider the following system. Sketch the graphs of these equations on the same coordinate system (by hand and then with plot), then solve to get the exact points of intersection.
   
   \[
   \begin{cases} 
   y = x^2 \\
   y = x 
   \end{cases}
   \]

7. Type in the following program and evaluate. (Note that there are exactly four spaces before the word “print”).

   ```python
   def write_string(string_name):
       print string_name
   
   Now type write_string("hello world!") and evaluate.
   ```
In order to do sophisticated calculations, or to allow for multiple inputs, you will need to write *programs*. Our “hello world!” program was the first example. It included a *print* statement. Other program features, in almost any language, include *conditional statements* (if..then..) and *loops*.

8. Type in the following function definition and evaluate.

```python
#This function will implement the absolute-value function

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

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

10. The hashtag and what follows it is a *comment*. These are useful explanations or reminders and are ignored by the compiler. Add your own comment using “#” in the cell where you defined `absolute(x)` like “Math is fun!”. Evaluate to check that Sage ignores it.

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

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

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

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

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

14. Write your own function *triple_product* that takes *three* inputs (call them anything, or `x, y, z` is fine) and *returns* their product.


We’ll eventually do all the (free) sections. (We won’t do any of the “Pro” sections.)