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 “Worksheets”, then call it c04.

2. Evaluate “pi”. Use n(_) to find a decimal approximation for π.

3. Draw the graphs of the following equations by hand. Find the solutions by hand.
   \[\begin{align*}
   x^2 + y^2 &= 4 \\
   y &= x + 1
   \end{align*}\]
   Now use solve() to find the intersection points of the graphs of this system of equations. First use the Help by typing solve?.

4. Consider the following system. Sketch the graphs of these lines on the same coordinate system, then solve to get the exact point of intersection.
   \[\begin{align*}
   2x + y &= 20 \\
   -x + y &= 0
   \end{align*}\]

5. Consider the following system. Sketch the graphs of these equations on the same coordinate system, then solve to get the exact points of intersection.
   \[\begin{align*}
   y &= x^2 \\
   y &= x
   \end{align*}\]
6. Type in the following program and evaluate.

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

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

7. You can define functions that produce interactive windows called *Sage Interacts*. Type in the following program and evaluate. “hello world!” is a default input and can be changed to any other plotable function.

```python
@interact
def i_write_string(f=input_box("hello world!")):
    print f
```

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

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

10. Make a *comment* using “#” in the cell where you defined `absolute(x)` like “This function returns the absolute value”. 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.