1. Log in to your Sage/Cocalc account.
   (a) Start the Chrome browser.
   (b) Go to http://cocalc.com and sign in.
   (c) You should see an existing Project for our class. Click on that.
   (d) Click “New”, call it c36, then click “Sage Worksheet”.

**Sierpinski Gasket**
You can read about the Sierpinski Gasket (or Sierpinski Triangle) at http://en.wikipedia.org/wiki/Sierpinski_triangle.

2. Try 5//2 and 1//2. This Sage command behaves the same way that 5/2 and 1/2 behaves in Python.

3. The binomial coefficient $\binom{a}{b}$ is defined to equal $\frac{a!}{b!(a-b)!}$. It equals the number of subsets of cardinality $b$ there are in a set of cardinality $a$. This number has lots of interpretations and interesting combinatorial properties. See if you can figure out $\binom{3}{2}$ by hand, and then see what you get with binomial(3, 2) in Sage.

Note that the familiar Pascal’s Triangle (see: http://en.wikipedia.org/wiki/Pascal’s_triangle) can be defined using binomial coefficients.

4. We’ve seen the range() command before but, like most things in Sage, it has powerful options that we haven’t seen (or used) before. Try range(4,20), range(4,20,2), and then range(4,20,3).

5. The following program uses matrix_plot() which is a way to represent the pattern of a matrix by associating different colors to different entries. Try matrix_plot(matrix(2,[1,2,3,4])), matrix_plot(matrix(2,[1,2,3,1])), and matrix_plot(matrix(2,[1,2,3,4]),cmap="hsv").

6. Define a $3 \times 3$ matrix with all different entries and use matrix_plot() to represent it.
7. Now let's make our Gasket!

```python
def sierpinski(N):
    """
    Generates the Sierpinski triangle by taking the modulo-2
    of each element in Pascal's triangle
    """
    S=[[0]*(N//2-a//2))+
    [binomial(a,b)%2 for b in range(a+1)]+
    ([0]*(N//2-a//2)) for a in range(0,N,2)]
    return S

@interact
def _(N=slider([2 ** a for a in range(12)],
               label="Number of iterations", default=64),
               size=slider(1, 20, label="Size", step_size=1, default=9)):
    M = sierpinski(2 * N)
    matrix_plot(M, cmap="binary").show(figsize=[size, size])
```

Random Walks

8. The following Sage INTERACT starts with the origin \((0,0,0)\) and "walks" \(u (u \in (-0.5, 0.5))\) units in each of the \(x, y\) and \(z\) directions, repeats this up to 1,000 times, keeps track of all of the points that are visited and then draws all of the points, with lines from one point to the next.

```python
@interact
def rwalk3d(n=slider(50,1000,step_size=1), frame=True):
    pnt = [0,0,0]
    v = [copy(pnt)]
    for i in range(n):
        pnt[0] += random()-0.5
        pnt[1] += random()-0.5
        pnt[2] += random()-0.5
        v.append(copy(pnt))
    show(line3d(v,color="black"),aspect_ratio=[1,1,1],frame=frame)
```

9. Make a Sage INTERACT which simulates a random walk in the \(x-y\) plane, starting at the origin.