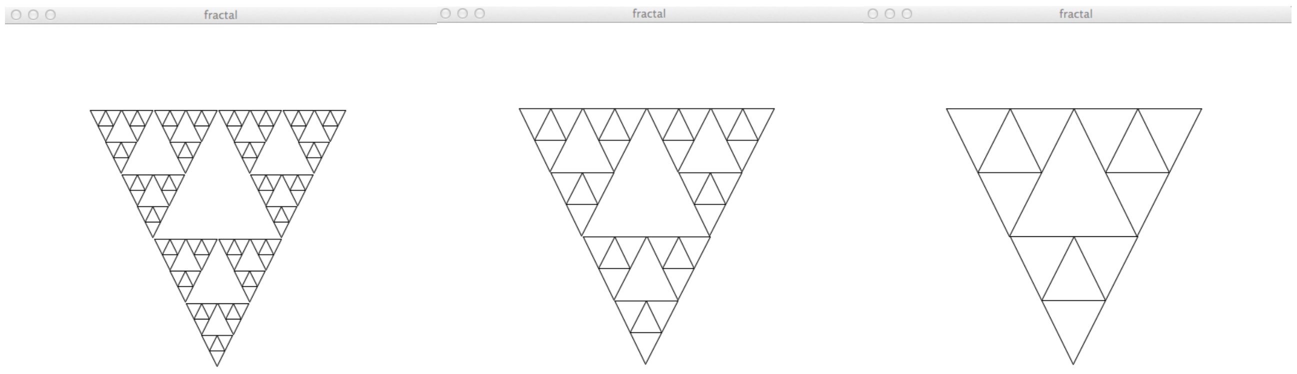
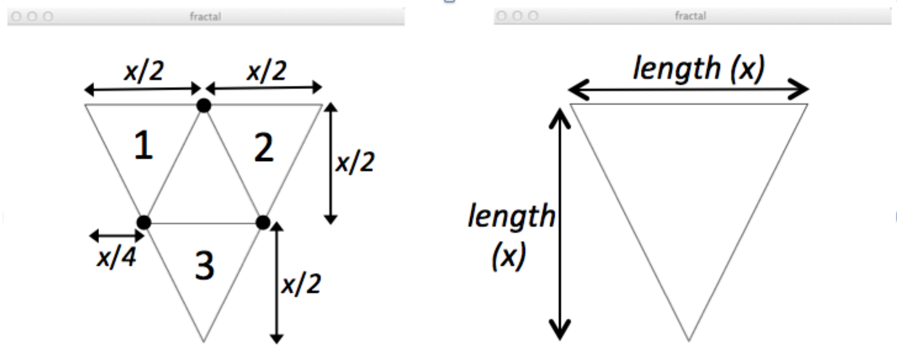
Fractals

One application of recursion is to create fractal images. You will create the specific image described below using recursion.

Deliverables: Your Python program in a file *user1_user2.py*.

The following images display a certain fractal at five different levels of detail: 4, 3, 2, 1, and 0. The fractal is made up of triangles. To increase from one detail level to the next, we subdivide each triangle into three equal triangles. The lower 2 images show the details of the dimensions of the 3 subtriangles created to move from detail level 0 to detail level 1.





In a file `fractal.py`, write a function to draw this fractal using the graphics.py module. Your function definition should begin as follows.

```
def drawFractal(win, length, detail, x, y):
```

where *win* is the window to draw to, *length* is the height (and width) of the current triangle, *detail* is the number of further levels of subdivision to go through, and *x* & *y* represent the upper left point of the current triangle.

You should read *length* and *detail* as command line arguments from the user. Your initial (x,y) point for the upper left corner should be (100,100).

For example, the four images above were produced using four different initial calls to `drawFractal()`, with details of 0, 1, 2, and 3, all with length 300.0. Just to give you an idea, my drawFractal function is 7 lines long. The majority of your grade will come from correctly solving the problem, but more efficient/elegant solutions will receive more points.

The easiest way to draw a triangle in Zelle's graphics is to create a Polygon with a list of 3 points as the argument, e.g.

```
triPoints = [ Point(10,10), Point(50,10), Point(30,50) ]
tri = Polygon( triPoints )
```

will create a triangle with its upper left corner at the point (10,10) in the window, and a width and height of 40.