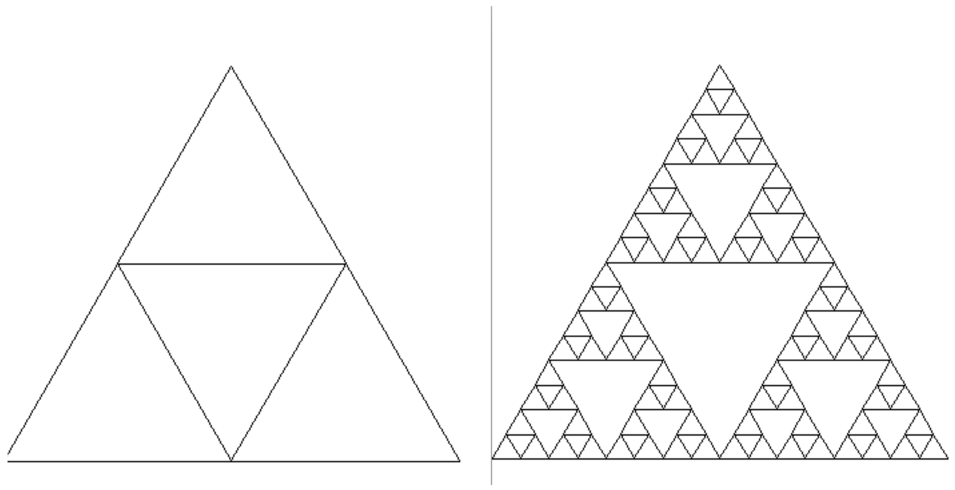
Computer Modelling for Scientists CSCI 2202
Lab F, 23 March 2021

Recursive Graphics: Drawing the Sierpinski Triangle.

$n = 1$ $n = 4$



The Sierpinski triangle is a fractal that can be drawn in iteratively and recursively:

(1) First draw it by an iterative process described below:
  - (a) Set up arrays to define the three corners of an equilateral triangle.
  - (b) Plot a point at one of the corners. (Refer to its coordinates as (x, y)).
  - (c) Randomly pick one of the three corners, and plot a point at the mid-point of (x, y) and the corner picked.
  - (d) Update (x, y) to the coordinates of the mid-point in the previous step.
  - (e) Repeat steps *(ii)* to *(iv)* 100000 times. The surprising thing is that the image you obtain is the same in each simulation. This is the Sierpinski triangle.

(2) Next draw this recursively:
  - (a) Notice that in the figure below, The $n = 1$ Sierpinski triangle is made up of 3 smaller triangles.
  - (b) Define a *recursive* function `sierpinski`. It takes the level of recursion `n`, the length of a side of the equilateral triangle `side`, and the coordinates `x, y` of a point, that allows you to define the corners of the triangle being drawn. An alternative is to pass the coordinates of all 3 corners instead of the side length.

(c) At this point you have a choice: Either *(i)*
In the base case ($n = 0$), define the coordinates of the corners of the triangle and use to draw the basic triangle using the corners you defined in the previous step (this will draw the fractal "bottom-up") Or *(ii)*
For the base case, do nothing, just return. This will draw the fractal "top-down".

(d) For *(i)* when not at the base case, make recursive calls with one less level of recursion and half the length of the side. You will need to determine the corners of the triangle for each recursive call.
For *(ii)* when not at the base case, draw a triangle with the given corner (x, y) coordinates. Calculate the mid-point of each side of your triangle. Perform the appropriate recursion by providing the correct corner points for each recursive call.

(e) Finally, set the x and y scales to fit your drawing to the screen and call `sierpinski` with the appropriate value for the parameters.

(f) Once your program is working, animate the drawing: For this, use the `pause(TD)` function in `matplotlib.pyplot` at the position in your code where it shows the drawing as it builds-up. `TD` is the time delay: set to ~ 1.