Python Bonsai

This problem asks you to create two classic kinds of recursive images using the `turtle` package: a tree made of line segments and the Koch Snowflake. If you are so inclined, you can make your tree more realistic than just a stick tree and get extra credit. You can use the turtle commands to play with color, make the trunk wider, etc.

In a typical Python installation, the `turtle` drawing module is installed and ready to use.

The `turtle` package provides us with a virtual turtle that can be controlled through Python commands. Start by using the turtle package at the Python command line. To do so, load in the turtle package with the command `import turtle`. Then, play around with the turtle for a few minutes. Try commands like these:

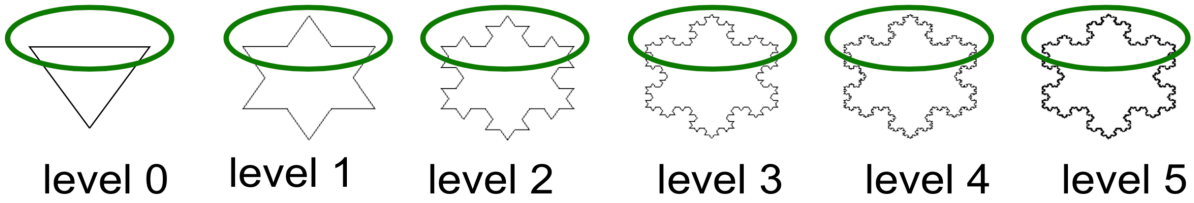
```
>>> turtle.forward(100)    <-- turtle goes forward 100 steps

>>> turtle.right(90)       <-- turtle turns right 90 degrees

>>> turtle.penup()         <-- turtle lifts its pen up off of the paper

>>> turtle.forward(100)    <-- turtle goes forward 100 steps

>>> turtle.pendown()       <-- turtle puts its pen down on the paper

>>> turtle.pencolor("red") <-- turtle uses red pen

>>> turtle.circle(100)     <-- turtle draws circle of radius 100

>>> turtle.pencolor("blue") <-- turtle changes to blue pen (most other common
colors work too!)

>>> turtle.forward(50)     <-- turtle moves forward 50 steps

>>> turtle.xcor()          <-- turtle returns its current x-coordinate

>>> turtle.ycor()          <-- turtle returns its current y-coordinate
```

To see the complete set of `turtle` commands go to the official Python turtle page.

Note: When you run a program with turtle commands, a special window will open where the drawing will take place. This window may appear behind some of your other open windows, so you may need to move windows to see it. If you run the program again, that same window will be used again. However, when you want to finally close that drawing window, you must type `turtle.bye()` at the IDLE prompt.

We have already seen in class how to draw a svTree at various recursion levels. In the first part of HW 4, you will be drawing Koch snowflake at a given recursion level.

The Koch Snowflake Fractal:



Note that the snowflake is an inverted triangle with each side growing flake recursively. Each side has the circled pattern at various recursion levels.

First write a recursive function **snowflakeSide(level,length) :**
For base case, draw a line of given length. For recursion case, recursively call the function 4 times at level one less, one for flat side, one after turning left 60 degrees, another after turning right 120 degrees and another (after turning left how many degrees ?) to be level with first line.

Now write the non-recursive function **snowflake(level, length)** that calls **snowflakeSide** 3 times one for each side of a triangle (turning right 120 degrees 3 times)

NOTE: All these functions do not need to have a return statement.

Have fun with flakes!!