# 1 Accomplishments

Over the past week, I have accomplished the following:

## 1.1 Octahedron (oc_.py files)

1. Extruded all eight interior triangles of the octahedron, as presented in oc4.py.
2. Converted the initial list of midpoint tuples to one that was produced wholly through the midpt function (oc3.py), thereby shortening my code.

## 1.2 Tetrahedron (tetra_.py files)

1. Created tetra1.py, which creates a regular tetrahedron centered at the origin.
2. Created tetra2.py, which has a tetrahedron function that takes a list of points and creates a faced tetrahedron. Attempted to create a function that would scale the tetrahedron (and created the failed programs tetra3.py and tetra4.py to deal with the issue, before noticing that tuples do not have a good method for scaling AND that I would not necessarily need to scale anything to have functional code).
3. Decided on a method for stellating the initial structures, thereby adhering to my Proposal.
4. Wrote pseudo-code for the tetrahedron and cube stellations, presented in section 2.
5. Researched the calculation of Hausdorff dimension for fractals which start with dimensions higher than one, but did not find a satisfactory method for approximating the dimension of my fractals.
6. Updated my webpage.

# 2 Pseudo-Code

## 2.1 Tetrahedron

1. Using the midpt function, as defined in oc3.py, find the midpoints of each side of each original triangle. (The base case has 4 original triangles.)

2. Create a triangle from the midpoints of each original triangle. Store this new triangle in an array (A).

3. Define the other three equilateral triangles (each of which contains one original corner and two original midpoints. Place these in another array (B).

4. For each object in A:

   - Extrude the triangle according to the extrude function (oc3.py).
   - Move the triangular sides of the extruded tetrahedron to B.
   - Delete the triangle from A.

5. For each object in B:

   - Subdivide the triangle into four triangles, as in steps 1-3.

## 2.2 Cube (Hexahedron)

1. Create a variable for side length (L).

2. Using the midpt function, as defined in oc3.py, find the points that divide the sides of the original squares into three (the base case has six squares). Instead of multiplying by 1/2, multiply by 1/3 and 2/3.

3. Compare the tuples for the dividing points. If the x-coordinates are the same, then the points of the new (center) square will be, in relation to one of the corners:

   - (0, 1/3L, 1/3L)
   - (0, 1/3L, 2/3L)
   - (0, 2/3L, 1/3L)
   - (0, 2/3L, 2/3L)

   The same coordinates will be true for other variable equalities. However, the zeros will be in different positions.

4. Place the new square in array A.

5. Break up the remaining area of the original square into eight other squares. Place these squares in B.

6. For each object in A:

   - Extrude the square according to an extrude function similar to that in oc3.py.

- Move the square of the extruded cube to B.
- Delete the square from A.

7. For each object in B:
   - Subdivide the square into nine squares, as in steps 1-5.

## 2.3 Needed Skills/ Further Research

1. Understand how arrays and lists work in Python.
2. Improve the efficiency of my code.
3. Figure out how to add user options for the code.