Next step was to optimize the elevation data gathering. First, I re-wrote the program so that the intersections were generated first, then elevation data retrieved for the relevant points. This took about 4 hours to do, because there were issues with finding problems in code, and figuring out the best way to reconfigure the program. Intersections\_build() was split up, because the first part gets relevant intersections, and the second part assigns pts to the paths. Since a copy of the point is made, it was not possible t assign copies and then change the data.

I also experimented with bottle.py beginning on April 8th. I wrote a short program to make a local server (which in theory could be hosted from a domain) to retrieve the elevation data from the Elevation API (since CORS restrictions do not apply in Python) and then pass it on to the Ajax request from script.js. This took about 3 hours, with experimenting and then needing to find and add the appropriate response headers (since initially the ‘no “Access-Control-Allow-Origin’ error message was present). I used the Bottle Documentation as a source to fix this problem on April 12, 2015.

To decrease the number of queries used, I looked up the format for the Encoded Polyline Algorithm. I tried to make a parser for that in Python, but I had problems with rendering negatives in binary. I later discovered a built-in function under google.maps.geometry.encoding.encodePath() that accomplished this. I substituted this format in lieu of “lat,lng|lat,lng|…”, but there were errors.