Golf course elevation gatherer

Useful links

- https://cloud.google.com/
- https://developers.google.com/maps/documentation/elevation/start
- https://geographiclib.sourceforge.io/html/python/
- https://mymaps.google.com/

Getting started & your API key

- To use this tool, one must first set up an account on https://cloud.google.com/, this is so that one may attain an API key for Googles' elevation API. Ensure that you've selected the Elevation API when registration (it may be apart of the maps API)
    - After setting up an account, go to the console on the above link. To attain the actual key, go to API's, click elevation API, then click credentials. There, you'll find your key (at the time of writing, to query the elevation server, the URL is:
    - https://maps.googleapis.com/maps/api/elevation/json?locations=latitude,longitude&key=YOUR_API_KEY (This link is the one used in the program)

How to use the program

- Before attempting to run, download geographiclib for python (linked above).
- Insert your API key on line 9(!!!!!!!)
- To use this program, one must have FOUR coordinate points. For example, say you want to make a shape that looks like this:

  X - - - - - - - - X
  - -
  - -
  X - - - - - - - - X

- AT THE BOTTOM OF THE PROGRAM, you will find four arrays, point1,point2,point3,point4. The coordinates for point1 would be the top left point on our parallelogram, point2 would be coord at top right, point3 would be coord at bottom left, point4 would be coord at bottom right.
- Then, look at the toCSV function. Change the path to your preferred path. This program outputs a csv in the form: lat,lon,elevation

How it works

X - - - - - - - - X
- -
- -
X - - - - - - - - X

- Take a look at our shape again, with p1 at top left, p2 at top right, p3 at bottom left, p4 at bottom right. Lets say that the length of the top line is x meters, and we want coordinates every 5m
    - First, we find out how far apart p1 and p2 are. The calculation that is done is found in the create rectangle function
        - info1 = geo.Inverse(p1[0],p1[1],p2[0],p2[1])
        - dist1 = info1['s12'] (OUR DISTANCE)
        - azimuth1 = info1['azi1']
            - P1 and P2's lat and lon are input into the geo.Inverse function, and it returns a dictionary with various values, like the distance between two coordinates.
    - Next, we need to find out the distance between point1 and point3, (top left, bottom left), using the same code, with p1 and p3's values.
- We now have enough information for our for loop, where the outermost will iterate from the distance from point1 to point3 every 5m, and the inner will iterate from the outer loops current point, to the distance from calculated by "dist1"
    - NOTE: This means that that the side lengths for the rectangle will ALWAYS be:
      Two sides = dist1
      Two sides = dist2

    So a shape like the below may not turn out as expected!

    - The bottom side would be as long as the top side! With some minor changes, if deemed necessary, you could change this behavior.

- Then, we use the azimuth (direction), and step size (every 5m) to calculate a new coordinate. We calculate a new azimuth at the beginning of each iteration, so that we are plotting the points along the correct path.
- Simply: calculate two distances, calculate bearing for new line of coordinates, query the server for information, calculate new bearing at next iteration, query server for info

To check if your shape is correct:

- Go to mymaps.google.com
    - Create new map
        - Import data under layer on @ top left
        - Change your csv to remove elevations
        - Import csv