Didn’t get this implemented but we got some functions that we planned to use to calculate heading from subsequent (Lat,Long) positions for each vessel in the data set and then calculate the angle difference between the headings of each ship in our possible COLREG interaction

**Using 4 input variables (lon1, lat1, lon2, lat2)**

from math import radians, cos, sin, asin, sqrt

def haversine(lon1, lat1, lon2, lat2):

    """

    Calculate the great circle distance between two points

    on the earth (specified in decimal degrees)

    """

    # convert decimal degrees to radians

    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula

    dlon = lon2 - lon1

    dlat = lat2 - lat1

    a = sin(dlat/2)\*\*2 + cos(lat1) \* cos(lat2) \* sin(dlon/2)\*\*2

    c = 2 \* asin(sqrt(a))

    r = 6371 # Radius of earth in kilometers. Use 3956 for miles

    return c \* r

**Bearing with two input variables (pointA, pointB).**

|  |
| --- |
| defcalculate\_initial\_compass\_bearing(pointA, pointB): |
|  | """ |
|  | Calculates the bearing between two points. |
|  |  |
|  | The formulae used is the following: |
|  | θ = atan2(sin(Δlong).cos(lat2), |
|  | cos(lat1).sin(lat2) − sin(lat1).cos(lat2).cos(Δlong)) |
|  |  |
|  | :Parameters: |
|  | - `pointA: The tuple representing the latitude/longitude for the |
|  | first point. Latitude and longitude must be in decimal degrees |
|  | - `pointB: The tuple representing the latitude/longitude for the |
|  | second point. Latitude and longitude must be in decimal degrees |
|  |  |
|  | :Returns: |
|  | The bearing in degrees |
|  |  |
|  | :Returns Type: |
|  | float |
|  | """ |
|  | if (type(pointA) != tuple) or (type(pointB) != tuple): |
|  | raise TypeError("Only tuples are supported as arguments") |
|  |  |
|  | lat1 = math.radians(pointA[0]) |
|  | lat2 = math.radians(pointB[0]) |
|  |  |
|  | diffLong = math.radians(pointB[1] - pointA[1]) |
|  |  |
|  | x = math.sin(diffLong) \* math.cos(lat2) |
|  | y = math.cos(lat1) \* math.sin(lat2) - (math.sin(lat1) |
|  | \* math.cos(lat2) \* math.cos(diffLong)) |
|  |  |
|  | initial\_bearing = math.atan2(x, y) |
|  |  |
|  | # Now we have the initial bearing but math.atan2 return values |
|  | # from -180° to + 180° which is not what we want for a compass bearing |
|  | # The solution is to normalize the initial bearing as shown below |
|  | initial\_bearing = math.degrees(initial\_bearing) |
|  | compass\_bearing = (initial\_bearing + 360) % 360 |
|  |  |
|  | return compass\_bearing |