input_generator.py

This is a python script used to get the latitude and longitude of the corner points of the field when user knows the distance of each point with respect to one reference point. (Ref Point lat/long must be known)

importing headers

In [1]:
```
from __future__ import print_function
import math
```

Custom Class for taking lat/lon points

POINT class takes two parameters x and y and save then as lon and lat variables of the class object.

In [2]:
```
class POINT:
        def __init__(self,x,y):
                self.lat = x
                self.lon = y
```

Functions used:

1. def get_location_metres(original_location, dNorth, dEast)

Returns a POINT object containing the latitude/longitude dNorth and dEast metres from the specified original_location. The function is useful when you want to move the vehicle around specifying locations relative to the current vehicle position. This function is relatively accurate over small distances (10m within 1km) except close to the poles.

Reference: http://gis.stackexchange.com/questions/2951/algorithm-for-offsetting-a-latitude-longitude-by-some-amount-of-meters (http://gis.stackexchange.com/questions/2951/algorithm-for-offsetting-a-latitude-longitude-by-some-amount-of-meters)

In [3]:
```
def get_location_metres(original_location, dNorth, dEast):

        #Radius of "spherical" earth
        earth_radius=6378137.0

        #Coordinate offsets in radians
        dLat = dNorth/earth_radius
        dLon = dEast/(earth_radius*math.cos(math.pi*original_location.la
t/180))

        #New position in decimal degrees
        newlat = original_location.lat + (dLat * 180/math.pi)
        newlon = original_location.lon + (dLon * 180/math.pi)
        new_location = POINT(newlat,newlon)
        return new_location
```

2. def get_distance_metres(aLocation1, aLocation2)

Returns the ground distance in metres between two POINT class objects. This method is an approximation, and will not be accurate over large distances and close to the earth's poles.

Reference: https://github.com/diydrones/ardupilot/blob/master/Tools/autotest/common.py (https://github.com/diydrones/ardupilot/blob/master/Tools/autotest/common.py)

In [4]:
```
def get_distance_metres(aLocation1, aLocation2):

        dlat = aLocation2.lat - aLocation1.lat
        dlong = aLocation2.lon - aLocation1.lon
        return math.sqrt((dlat*dlat) + (dlong*dlong)) * 1.113195e5
```

Main Body:

Taking user input for land info:

User enters the latitude and longitude of the reference point of the land under consideration. We further check if the values entered is valid or not.

In [5]:
```
print("   This code generates the input lat long values \n")
print("Please enter the geo location of the reference point of your fiel
d: \n")
in_lat = 0.0
in_lon = 0.0
while True:
        try:
                in_lat = float(input("Please enter the latitute of poin
t:\n"))
                if(in_lat<0 or in_lat>90):
                        print("Latitude value must be between 0 and 90")
                        continue
                in_lon = float(input("Please enter the longitude of poin
t:\n"))
                if(in_lon<0 or in_lon>180):
                        print("Langitude value must be between 0 and 18
0")
                        continue
                break
        except:
                print("Oops!  That was no valid lat/lon.  Try again...")
```

```
   This code generates the input lat long values

Please enter the geo location of the reference point of your field:

Please enter the latitute of point:
10.0
Please enter the longitude of point:
10.0
```

Create out custom POINT object for input:

In [6]:
```
first_point = POINT(in_lat,in_lon)
```

Create a input.txt file to store all the calculated points.

Store the first point taken from user in file.

In [7]:
```
input_file = open("input.txt","w+")
input_file.write(str(first_point.lat) + "," + str(first_point.lon) + '\n
')
```

Taking user input for number of points to be generated:

In [8]:
```
number_of_points = int(input("Please enter the more number of point to g
enerate:\n"))
```

```
Please enter the more number of point to generate:
5
```

User inputs the distance of each point from reference point

User inputs the each corner point distance in form of (NORTH,EAST) i.e. if next point is 10 m north and 20 m east of initial reference , user enters 10 20 when asked.

In [9]:
```
for i in range(number_of_points):
        print("Enter the next point distance in meter (north,east) from
refrence point:\n")
        move_north = int(input("North distance to point: "))
        move_east = int(input("East distance to point: "))
        new_point = get_location_metres(first_point,move_north,move_eas
t)
        input_file.write(str(new_point.lat) + "," + str(new_point.lon) +
'\n')

print("All generated points are stored in input.txt.\n")
```

```
Enter the next point distance in meter (north,east) from refrence point:

North distance to point: -50
East distance to point: 50
Enter the next point distance in meter (north,east) from refrence point:

North distance to point: 0
East distance to point: 100
Enter the next point distance in meter (north,east) from refrence point:

North distance to point: 50
East distance to point: 100
Enter the next point distance in meter (north,east) from refrence point:

North distance to point: 100
East distance to point: 50
Enter the next point distance in meter (north,east) from refrence point:

North distance to point: 50
East distance to point: 0
All generated points are stored in input.txt.
```