

```

import math
from dbfpy import dbf
import matplotlib.pyplot as plt

#create a plot
fig = plt.figure(1, figsize = [10,10], dpi=90)
axScatter = plt.subplot(111)

def candMedian(dataPoints):
    #Calculate the first candidate median as the geometric mean
    tempX = 0.0
    tempY = 0.0

    for i in range(0,len(dataPoints)):
        tempX += dataPoints[i][0]
        tempY += dataPoints[i][1]

    return [tempX/len(dataPoints),tempY/len(dataPoints)]

def numersum(testMedian,dataPoint):
    # Provides the denominator of the weiszfeld algorithm depending on whether you are adjusting the candidate x or y
    return 1/math.sqrt((testMedian[0]-dataPoint[0])**2 + (testMedian[1]-dataPoint[1])**2)

def denomsum(testMedian, dataPoints):
    # Provides the denominator of the weiszfeld algorithm
    temp = 0.0
    for i in range(0,len(dataPoints)):
        temp += 1/math.sqrt((testMedian[0] - dataPoints[i][0])**2 + (testMedian[1] - dataPoints[i][1])**2)
    return temp

def objfunc(testMedian, dataPoints):
    # This function calculates the sum of linear distances from the current candidate median to all points
    # in the data set, as such it is the objective function we are minimising.
    temp = 0.0
    for i in range(0,len(dataPoints)):
        temp += math.sqrt((testMedian[0]-dataPoints[i][0])**2 + (testMedian[1]-dataPoints[i][1])**2)
    return temp

# Use the above functions to calculate the median
# Test Data - later to be read from a file
#dataPoints = [[3,4],[3,3],[6,8],[9,3],[3,5],[1,2],[6,3]]

```

```

# Data read from dbf file exported, and randomly offset, from ArcGIS 9.3
dataPoints = []
filestring = r"C:\Users\Daniel\Documents\AddressLayerSwk\AddressDBFs\TestMediansBradford.dbf"
db = dbf.Dbf(filestring)
for i in range(0,len(db)):
    rec = db[i]
    x = float(rec['easting'])
    y = float(rec['northing'])
    dataPoints.append([x,y])
    # add data points to scatter plot
    axScatter.scatter(x,y)

axScatter.set_aspect(1.)
# Create a starting 'median'
testMedian = candMedian(dataPoints)
print testMedian
#add mean to scatter plot
axScatter.scatter(testMedian[0],testMedian[1],s=150,color='green', marker='x')

# numIter depends on how long it take to get a suitable convergence of objFunc
numIter = 50

#minimise the objective function.
for x in range(0,numIter):
    print objfunc(testMedian,dataPoints)
    denom = denomsum(testMedian,dataPoints)
    nextx = 0.0
    nexty = 0.0

    for y in range(0,len(dataPoints)):
        nextx += (dataPoints[y][0] * numersum(testMedian,dataPoints[y]))/denom
        nexty += (dataPoints[y][1] * numersum(testMedian,dataPoints[y]))/denom

    testMedian = [nextx,nexty]

# add final median to scatter plot (to see progression add this line into the loop)
axScatter.scatter(testMedian[0],testMedian[1],s=150,color='red', marker='x')

#create a legend for plot
leg = axScatter.legend(('Mean Centre','Median Centre','Data Points'),scatterpoints = 1)

print testMedian
plt.show()

```