```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu May 30 13:43:58 2019

@author: torjus

THIS PYTHON FILE IS USED TO CONDTUCT THE TRADE FLOW ANALYSIS

"""

#TRADE FLOW ANALYSIS - SINGLE PORT
import areas
import AIS_Plotting as PAIS
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
from datetime import datetime
import statistics as st
from geopy.distance import great_circle
from shapely.geometry import MultiPoint
import pandas as pd
import AIS_Plotting as PAIS
import dataHandling
from datetime import timedelta


import json
import geog
import shapely.geometry
import time


import sqlite3
import time
import datetime

from scipy import stats

import sklearn.cluster as cluster
import scipy.cluster.hierarchy as hcluster
import loc_check as LC
import networkx as nx
from itertools import cycle , groupby
from pylab import boxplot
import matplotlib.colors as mcolors
import unixTimeConvert as UTC
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
from datetime import timedelta
from shapely.geometry import MultiPoint
from math import radians, cos, sin, asin, sqrt


def genAllTerminalAreas(df):
    polygons = list()
    for i in range(0,len(df['longitude'])):
        p = createTerminalArea(df['longitude'][i],df['latitude'][i],df['diameter'][i])
        p.append(p[0]) #Adding the first point
        polygons.append(p)

    return polygons

def createTerminalArea(x,y,d_km):
    p = shapely.geometry.Point([x, y])

    n_points = 10 #number of points in polygon
    d =  d_km * 1000  #Diameter in meters
    angles = np.linspace(0, 360, n_points)
```

```
    polygon = geog.propagate(p, angles, d)
    polygon=polygon.tolist()
    return polygon

def insidePolygon(row,polygons):
    x = row.longitude
    y = row.latitude
    for i, polygon in enumerate(polygons):
        lon,lat = zip(*polygon)
        xMax = max(lon)
        xMin = min(lon)
        yMax = max(lat)
        yMin = min(lat)
        if x<xMax:
            if x>xMin:
                if y<yMax:
                    if y>yMin:
                        if rayCasting(x,y,polygon):
                            return i
    return -1 #If not inside any defined polygon

def assignPolygon(df,polygons):
    start = time.perf_counter()
    df_out=df.copy()
    df_out['node']=-2 #-2 means not yet classified
    df_out['node'] = df.apply(insidePolygon,args=[polygons],axis=1)
    end = time.perf_counter()
    tot = end-start
    return df_out,tot

def obtainUniqueVisits(df_poly,nodes,df_V):
    portlogList = [0] * len(nodes) #This will be the output list of portlogs for all nodes
    for i in range(0,len(nodes)):
        mmsiInNode = df_poly['mmsi'][df_poly['node']==i].unique()
        columns = ['DateTime','mmsi','node','LNG']
        df_portLog = pd.DataFrame(columns = columns )
        for mmsi in mmsiInNode:
            df_temp = df_poly[df_poly['mmsi']==mmsi]
            df_temp['diff']=df_temp['node'].diff()
            df_temp = df_temp[df_temp['diff']!=0]
            df_temp2 = df_temp[df_temp['node']==i]
            df_temp2['timediff']=df_temp2['DateTime'].diff()
            df_temp2['timediff'][0]=timedelta(days=999) #Setting big value to secure not removing first vi
            df_temp2=df_temp2[df_temp2['timediff']>=timedelta(days=1)]

            for j in range(0,len(df_temp2)):
                date = df_temp2['DateTime'][j]
                userid = df_temp2['mmsi'][j]
                node = df_temp2['node'][j]
                LNG = df_V['gasCap'][df_V['mmsi']==userid]
                if len(LNG)==0:
                    LNG = 0
                else:
                    LNG = LNG.iloc[0]

                portvisit = pd.Series([date,userid,node,LNG],index=df_portLog.columns)
                df_portLog = df_portLog.append(portvisit,ignore_index = True)

        portlogList[i]=df_portLog
    return portlogList

def rayCasting(x,y,poly):
    n = len(poly)
    inside =False

    p1x,p1y = poly[0]
    for i in range(n+1):
        p2x,p2y = poly[i % n]
        if y > min(p1y,p2y):
            if y <= max(p1y,p2y):
                if x <= max(p1x,p2x):
                    if p1y != p2y:
                        xinters = (y-p1y)*(p2x-p1x)/(p2y-p1y)+p1x
                    if p1x == p2x or x <= xinters:
```

```
                        inside = not inside
        p1x,p1y = p2x,p2y

    return inside

def importVesselData():
    df_V = pd.read_excel('LNGC_GPV_Database.xlsx',sheet_name='LNGC_All')
    return df_V
```