```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
These tools are for building domain polygons
Author: Alexandra Christensen
Created: Thu Jan 14 17:36:39 2021
Updated: 12/01/2022

"""
#import rasterio
#from osgeo import gdal
#from rasterio._fill import _fillnodata
import numpy as np
#import os
from shapely.geometry import Polygon, mapping, LinearRing, MultiPolygon,
Point, LineString, MultiPoint
import geopandas as gpd
import warnings; warnings.filterwarnings('ignore', 'GeoSeries.notna',
UserWarning)
#import math
#from string import Template
import pandas as pd
#from rasterstats import zonal_stats
#from deltas_download import download_NASADEM2
import sys
sys.path.insert(1, '/scratch_lg/loac_hydro/alchrist/anuga/code')
from deltas_mesh import segmentize, lesspoints, removedegenerate
#from raster_bridges import make_distance
#import fnmatch
#from  make_tides import maketides
#from pathlib import Path
from sklearn.neighbors import BallTree
pd.options.mode.chained_assignment = None
#from shapely import geometry

def findconnectedwater(input_polys):
    largest_one =
input_polys.loc[input_polys.area==max(input_polys.area)].index[0]
    output_polys =
input_polys.loc[input_polys.area==max(input_polys.area)].copy(deep=True)
    input_polys = input_polys.drop(largest_one)
    output_polys = output_polys.reset_index(drop=True)
    # added= 1
    # while added <10  and input_polys.shape[0]>0:
    #     input_polys = input_polys.reset_index(drop=True)
    for index,row in input_polys.iterrows():
        if
len(input_polys[input_polys.geometry.touches(row['geometry'])])==0:
            input_polys = input_polys.drop([index],axis=0)
    input_polys = input_polys.reset_index(drop=True)
    for index in range(input_polys.shape[0]):
        if
output_polys.loc[0].geometry.intersects(input_polys.loc[index].geometry)=
=True:
```

```
            output_polys.loc[1] = input_polys.loc[index]
            input_polys = input_polys.drop([index],axis=0)
            output_polys = output_polys.dissolve(by='dissolve')
            output_polys = output_polys.reset_index(drop=True)

output_polys.insert(output_polys.shape[1],"dissolve",np.zeros((output_pol
ys.shape[0],1)))
        # added = added+1
    return output_polys
def delete_holes(input_polys):
    crs = input_polys.crs
    exploded = input_polys.explode(index_parts = True)
    interiors = exploded.interiors
    print('There are %s holes' %(len(interiors)))
    holes = gpd.GeoDataFrame()
    holes['geometry'] = None
    holes.crs = crs
    i=0
    for polygon in interiors:
        for interior in range(len(polygon)):
            interior_x = np.array(polygon[interior].coords.xy[0])
            interior_y = np.array(polygon[interior].coords.xy[1])
            newxy = np.column_stack((interior_x,interior_y))
            temppolygon = Polygon(newxy)
            holes.loc[i,'geometry'] = temppolygon
            i=i+1
    holes.geometry = holes.buffer(0.1)
    polys_no_holes =
gpd.overlay(input_polys,holes,how='union',keep_geom_type=True)
    polys_no_holes['dissolve'] = 1
    polys_no_holes = polys_no_holes.dissolve(by='dissolve')
    polys_no_holes = polys_no_holes.explode(index_parts =
True).reset_index(drop=True)
    polys_no_holes.geometry = polys_no_holes.buffer(0.001)
    print('holes removed')

    return polys_no_holes,holes


def get_nearest(src_points, candidates, k_neighbors=1):
    tree = BallTree(candidates, leaf_size=15, metric='minkowski')
    distances, indices = tree.query(src_points, k=k_neighbors)
    distances = distances.transpose()
    indices = indices.transpose()
    closest = indices[0]
    closest_dist = distances[0]
    return (closest, closest_dist)
def nearest_neighbor(left_gdf, right_gdf, return_dist=False):
    left_geom_col = left_gdf.geometry.name
    right_geom_col = right_gdf.geometry.name
    right = right_gdf.copy().reset_index(drop=True)
    left_radians = np.array(left_gdf[left_geom_col].apply(lambda geom:
(geom.x, geom.y)).to_list())
```

```
    right_radians = np.array(right[right_geom_col].apply(lambda geom:
(geom.x , geom.y)).to_list())
    closest, dist = get_nearest(src_points=left_radians,
candidates=right_radians)
    closest_points = right.loc[closest]
    closest_points = closest_points.reset_index(drop=True)
    if return_dist:
        # Convert to meters from radians
        closest_points['distance'] = dist
    return closest_points
def getpolygonpoints(filled_area,ulx,uly,lrx,lry,triangle_length):
    filled_area = filled_area.explode(index_parts = True)
    filled_area = filled_area.reset_index(drop=True)
    filled_area = segmentize(filled_area,1)
    filled_area['dissolve'] = 0
    filled_area = filled_area.dissolve(by = 'dissolve')
    crs = filled_area.crs
    xcoords = []
    ycoords = []
    if filled_area.geometry[0].type == "MultiPolygon":
        filled_area.geometry[0] = MultiPolygon(Polygon(p.exterior) for p
in filled_area.geometry[0])
        g = [a for a in filled_area.geometry[0]]
        for b in g:
            xcoords.extend(np.array(b.exterior.coords.xy[0]))
            ycoords.extend(np.array(b.exterior.coords.xy[1]))
    else:
        filled_area.geometry[0] =
Polygon(filled_area.geometry[0].exterior)
        g = filled_area.geometry[0]
        xcoords = np.array(g.exterior.coords.xy[0])
        ycoords = np.array(g.exterior.coords.xy[1])
    newxy = np.column_stack((xcoords,ycoords))
    temppoints = gpd.GeoDataFrame(newxy, geometry=[Point(x,y) for x,y in
zip(newxy[:,0],newxy[:,1])],crs=crs)
    newxy = lesspoints(newxy,(triangle_length))
    # newxy[(newxy[:,0]>lrx),0] = lrx
    # newxy[(newxy[:,0]<ulx),0] = ulx
    # newxy[(newxy[:,1]>uly),1] = uly
    # newxy[(newxy[:,1]<lry),1] = lry
    #newxy = (removedegenerate(newxy[:,0],newxy[:,1]))
    points = gpd.GeoDataFrame(newxy, geometry=[Point(x,y) for x,y in
zip(newxy[:,0],newxy[:,1])],crs=crs)
    points = points.drop([0,1],axis=1)
    return filled_area,points
def removenearbypoints(points,points_all, folder, delta,
desc,meshscale,base_length=220):
    nearest = nearest_neighbor(points,points_all,return_dist=True)
    nearest = nearest.rename(columns={'geometry':'closest_river'})
    points = points.join(nearest)
    for index,row in points.iterrows():
        #print(row['distance'])
        if row['distance'] < base_length:
            points.loc[index,'geometry'] =
points.loc[index,'closest_river']
    points = points.drop(['closest_river','distance'],axis=1)
    newxy = np.array(points['geometry'].apply(lambda geom: (geom.x,
geom.y)).to_list())
    #print('##################### %s domain defined with
%s%s_%s_points.csv' %(desc,folder,delta,desc))
    #print('%s%s_%s_points_%s.csv' %(folder,delta,desc,meshscale))

    return points, newxy
```