Add descriptive header code here

A good reference:

https://autogis-site.readthedocs.io/en/latest/lessons/Raster/overview.html

In [3]:
```
import rasterio
```

1. Point to the rasters and read them in:

Three layers are needed for this notebook:
GoodSlope.tif
GoodDistance.tif
FedLands_Teton.shp

In [4]:
```
# Path to the good slope tif
slopef = r"C:\Users\xfour\Downloads\Lab5c_NewRasters\goodslope.tif"
```

In [7]:
```
# Path to the good distance tif
trailf = r"C:\Users\xfour\Downloads\Lab5c_NewRasters\goodtrail.tif"
```

In [9]:
```
sloperas = rasterio.open(slopef)

# Your turn: Now open distf and call it distras

trailras = rasterio.open(trailf)
```

In [10]:
```
#Check the raster type to see that they got read in properly:
print(type(sloperas))
print(type(trailras))
```

```
<class 'rasterio.io.DatasetReader'>
<class 'rasterio.io.DatasetReader'>
```

2. Check the coordinate systems of the rasters

In [19]:
```
#Print and compare the coordinate systems of each raster
print(sloperas.crs)

# Your turn: Print the distras crs
print(trailras.crs)

#Your turn: compare the 2 crs's with an if statement

if sloperas.crs == trailras.crs:
        print("Both rasters have the same CRS.")
else:
        print("The CRSs are different.")
```

```
EPSG:26912
EPSG:26912
Both rasters have the same CRS.
```

Since the coordinate systems are not the same, they will need to be projected. (Technically, the coordinate systems are the same, it's the datums that are different.)

In [20]:
```
# Print out the file format
print ("The GoodSlope tif is a: " + str(sloperas.driver))
```

```
The GoodSlope tif is a: GTiff
```

3. Read some raster information:

In [21]:
```
# Read the raster stats
import numpy as np
array = sloperas.read()

# Calculate statistics for each band
stats = []
for band in array:
    stats.append({
        'min': band.min(),
        'mean': band.mean(),
        'median': np.median(band),
        'max': band.max()})

# Show stats for each channel
stats
```

Out[21]:
```
[{'min': 0, 'mean': 2.172041095453197, 'median': 1.0, 'max': 255}]
```

In [22]:
```
#  Your turn: Read the stats for the other raster...
# Read the raster stats
import numpy as np
array = trailras.read()

# Calculate statistics for each band
stats = []
for band in array:
    stats.append({
        'min': band.min(),
        'mean': band.mean(),
        'median': np.median(band),
        'max': band.max()})

# Show stats for each channel
stats
```

Out[22]:
```
[{'min': 0, 'mean': 1.6559984611186835, 'median': 0.0, 'max': 255}]
```

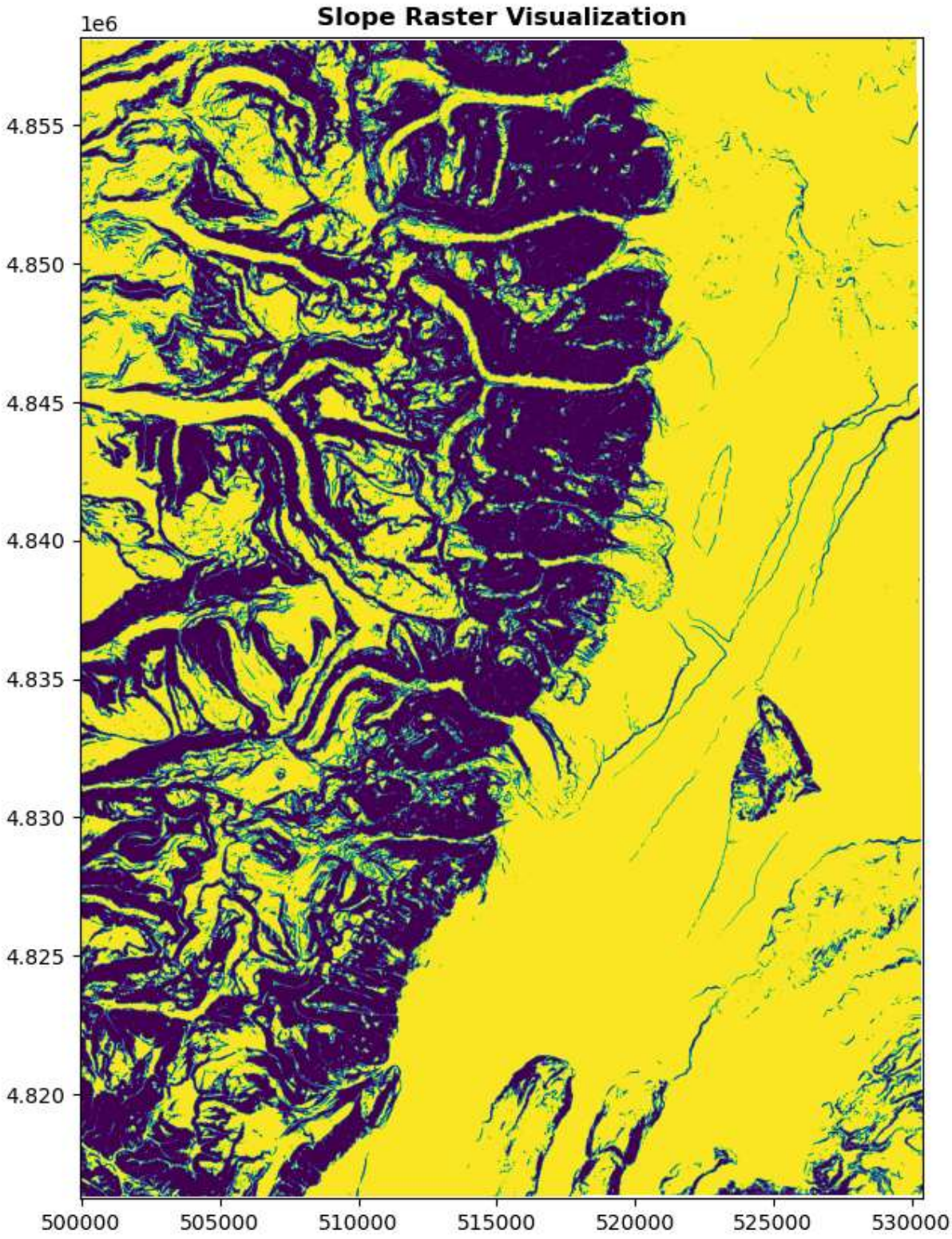
4. Plot the data

In [26]:

```
#Let's try this visualization thing...
from rasterio.plot import show
import os
import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots(figsize=(10, 10))  # Set the size of the figure
show(sloperas, ax=ax, title="Slope Raster Visualization")  # Display the raster

plt.show()
```

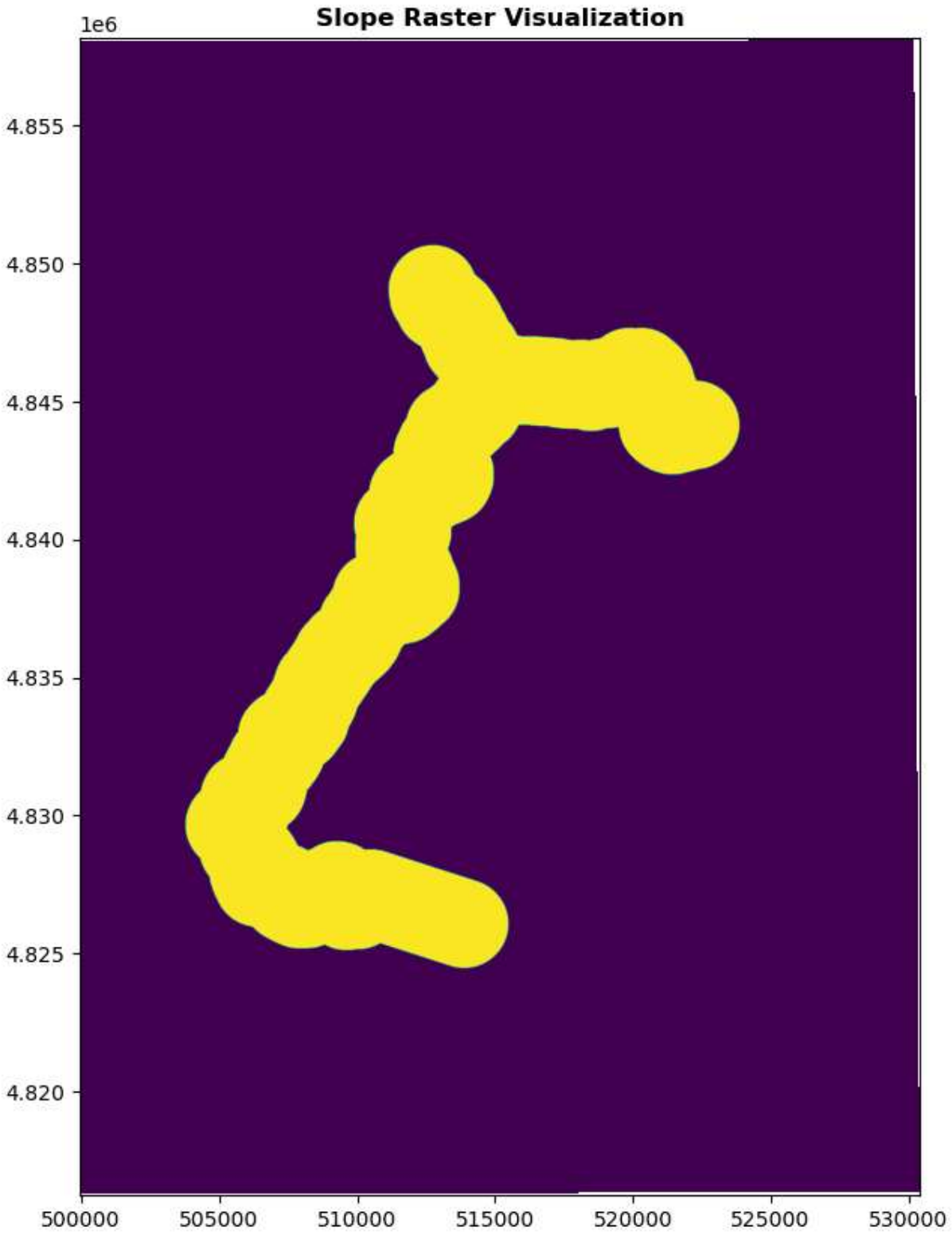


In [27]:
```
# Your turn: Now plot the other raster:
fig, ax = plt.subplots(figsize=(10, 10))  # Set the size of the figure
show(trailras, ax=ax, title="Slope Raster Visualization")  # Display the raster

plt.show()


# Can you figure out how to show two plots side by side?
```

```
# Hint - examples in the
# https://autogis-site.readthedocs.io/en/latest/index.html
```



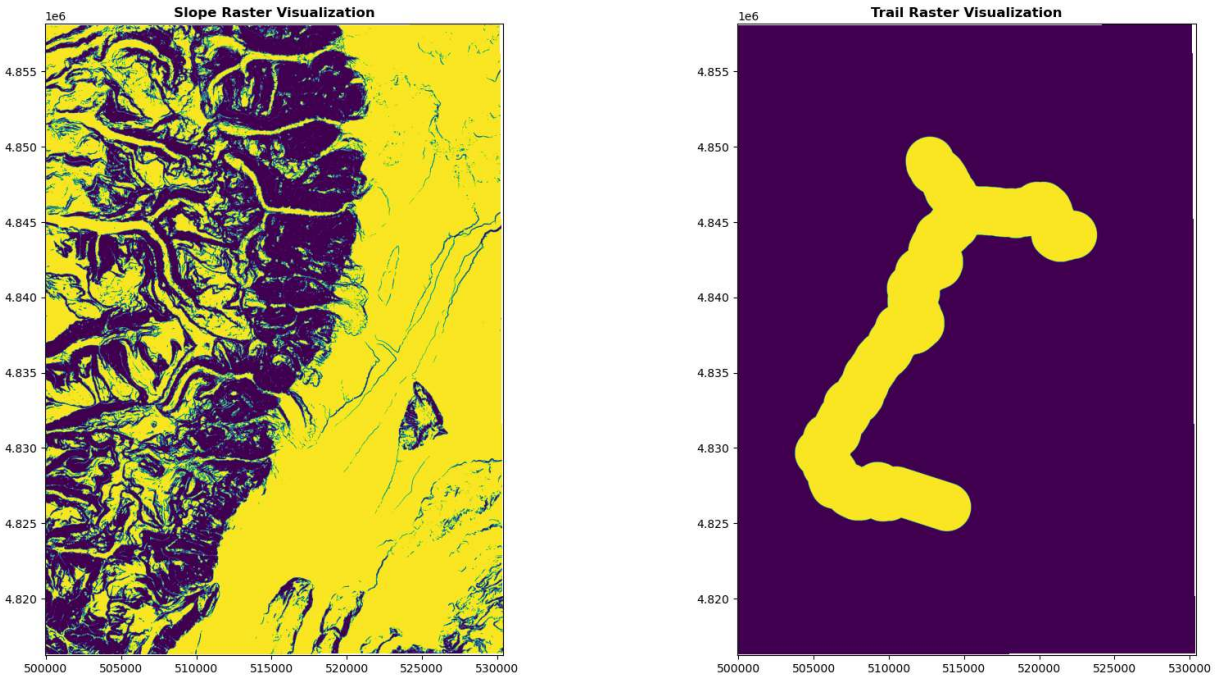
In [29]:
```
# Can you figure out how to show two plots side by side?

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))  # 1 row, 2 columns

# Show the slope raster in the first subplot
show(sloperas, ax=ax1, title="Slope Raster Visualization")
```

```
# Show the trail raster in the second subplot
show(trailras, ax=ax2, title="Trail Raster Visualization")

plt.show()
```

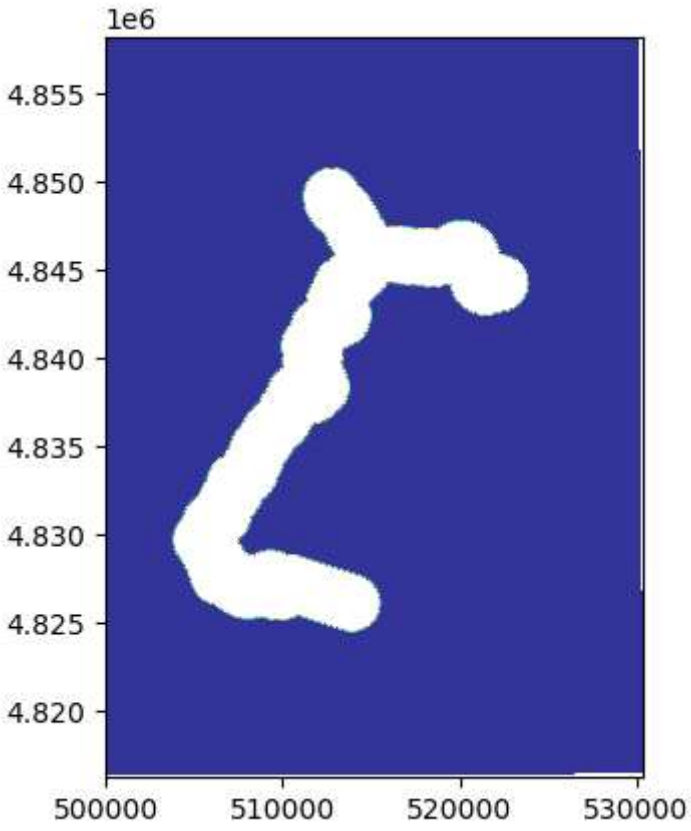


5. Clip the raster to the correct extent:

With rasterio, input layers must match exactly: extent, row and columns, cell size, CRS... in order to overlay

In [30]:
```
#Let's try to clip this raster!
#Maybe even both rasters!
#Following the steps from:  https://automating-gis-processes.github.io/CSC/notebook
from rasterio.mask import mask
from shapely.geometry import box
import geopandas as gpd
from fiona.crs import from_epsg
```

In [32]:
```
show(trailras, cmap='terrain')
```



Out[32]: <Axes: >

In [33]:
```
# File path for the shapefile to use for clipping
bnd_shp = r"C:\Users\xfour\Downloads\Lab5c_NewRasters\FedLands_Tetons.shp"

# Turn the shp into a geopandas dataframe
bnd = gpd.read_file(bnd_shp)
```

In [34]:
```
# Check that the coordinate systems are the same:
if bnd.crs != sloperas.crs:
    print ("Need to project the vector")
    # Make the vector CRS the same as the raster
    bnd = bnd.to_crs(crs=sloperas.crs.data)
else:
    print ("Coordinate systems are the same "+ str(bnd.crs))
```

```
Coordinate systems are the same EPSG:26912
```

So here's what we have:

bnd: This is the geopandas representation of the shapefile we want to clip by

sloperas and distras: The two rasters that we want to clip, then overlay

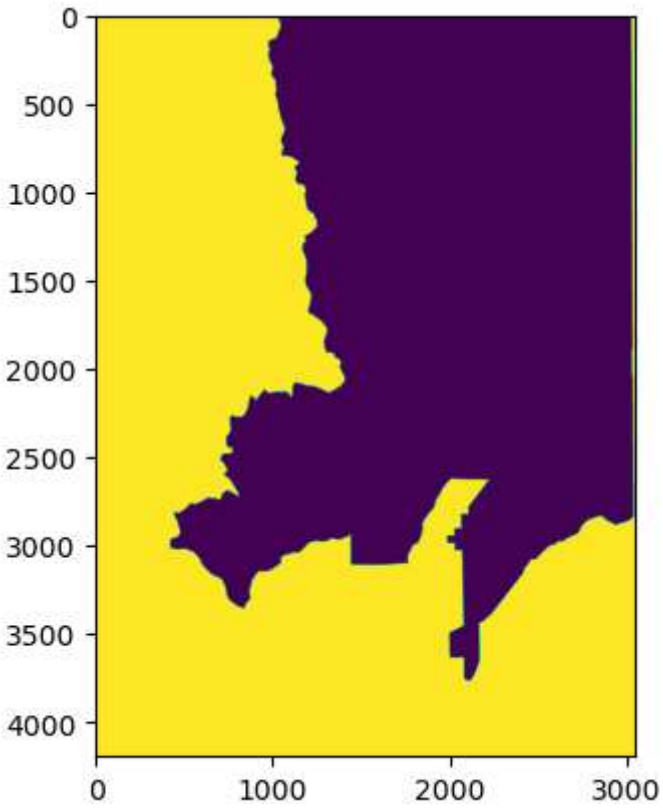
In [35]:
```
def getFeatures(gdf):
    """Function to parse features from GeoDataFrame in such a manner that rasterio
    import json
    return [json.loads(gdf.to_json())['features'][1]['geometry']]
    # Indexing in line above retrieves just one feature
```

```
coords = getFeatures(bnd)
# If you print this, be prepared for a few pages of coordinates:
#print (coords)
```

In [36]:
```
# Now let's actually do the clip:
# Mask is shown as just "mask" because of the way we imported it. It's really raste
# Refer to https://rasterio.readthedocs.io/en/stable/topics/masking-by-shapefile.ht
out_img, out_transform = mask(sloperas, coords, True)
```

In [37]:
```
# Update the metadata for the clipped version:
out_meta=  sloperas.meta.copy()
```

In [38]:
```
show(out_img)
```

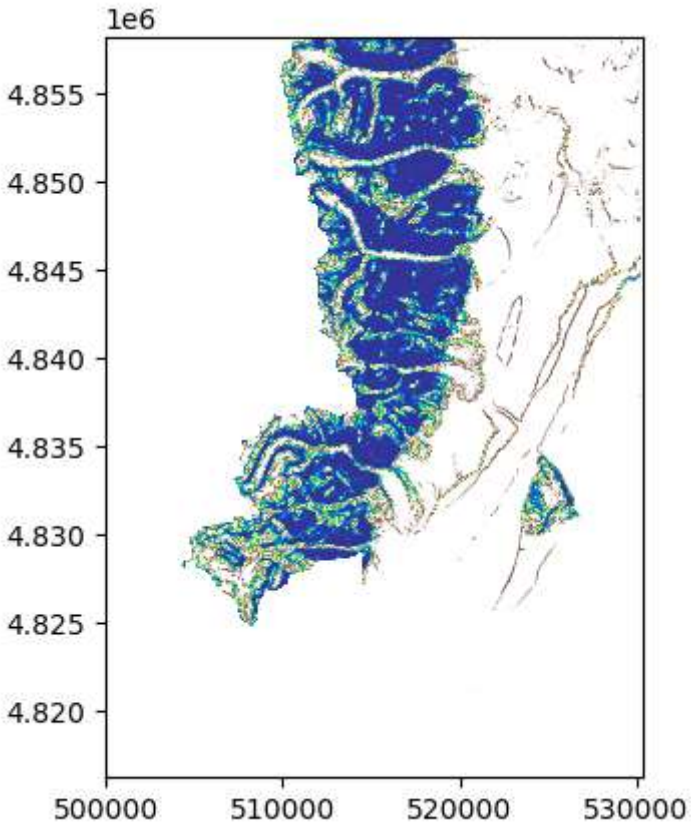


Out[38]: <Axes: >

In [39]:
```
# Save the output as a new TIF:
# Output raster
out_tif = r"C:\Users\xfour\Downloads\Lab5c_NewRasters\slope_clp.tif"
with rasterio.open(out_tif, "w", **out_meta) as dest:
        dest.write(out_img)
```

In [40]:
```
# Open the clipped raster file
slope_clp = rasterio.open(out_tif)

# Visualize
show(slope_clp, cmap='terrain')
```



Out[40]: <Axes: >

In [43]:
```
slopeval = sloperas.read()
distval = trailras.read()
```

The rest of Lab 5c:

Perform these raster based operations to get additional practice with it:

1. Find and clip another raster to another shapefile
2. Overlay the distance and slope rasters using map algebra: experiment with addition and multiplication to execute both a binary and a weighted overlay situation.
3. Use the RMNP data from Lab 4 /5b to see what you can do with the elevation layer and Zonal Statistics: Average elevation for each lake (or subset of lakes, or just one lake) Add elevation attribute to each Elk point

OR:
--Explore the EarthLab courses and tutorials in Python https://www.earthdatascience.org/ and pick out a 2-3 topics that interest you (Aim for ~1 hour of time) --Follow the Mosaic lesson from https://autogis-site.readthedocs.io/en/latest/notebooks/Raster/raster-mosaic.html using data that you find/download yourself and create a mosaic

In [ ]:
```
### Add new cells here as needed to complete the rest of Lab 5c
```

In [46]:
```
# Path to the good distance tif
elevf = r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\RMNPelev.tif"
```

```
elevras = rasterio.open(elevf)
```

In [48]:
```
lake_shp = r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\RMNP_Lakes.sh

lakes = gpd.read_file(lake_shp)
```

In [56]:
```
# Check that the coordinate systems are the same:
if lakes.crs != elevras.crs:
    print ("Need to project the vector")
    # Make the vector CRS the same as the raster
    lakes = lakes.to_crs(crs=elevras.crs.data)
else:
    print ("The Coordinate systems are the same: "+ str(lakes.crs))
```

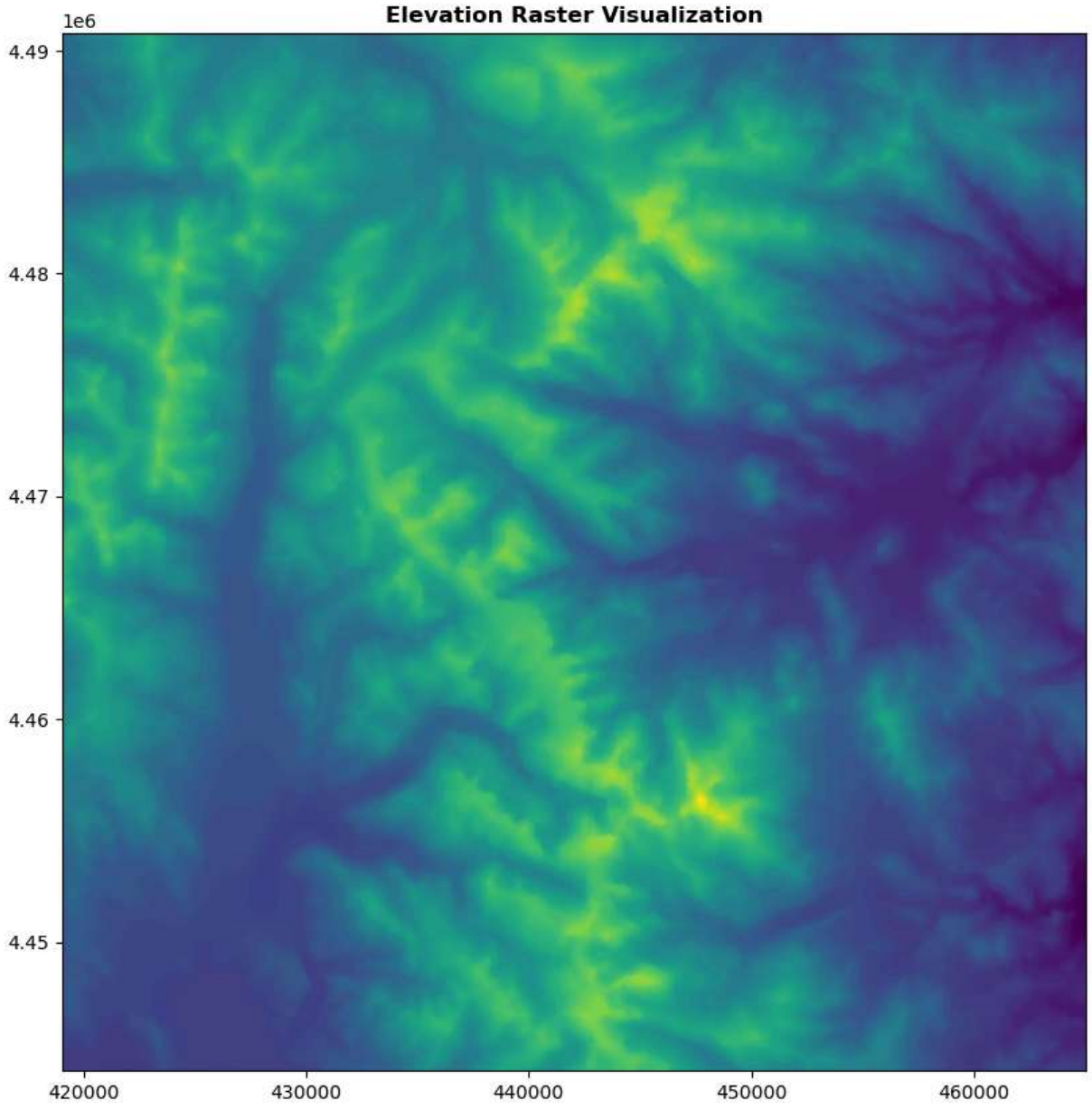
```
The Coordinate systems are the same: +init=epsg:26913 +type=crs
```

In [53]:
```
def getFeatures(gdf):
    import json
    return [json.loads(gdf.to_json())['features'][1]['geometry']]
coords = getFeatures(lakes)
```

In [58]:
```
#Let's try this visualization thing...
from rasterio.plot import show
import os
import matplotlib.pyplot as plt
%matplotlib inline

fig, ax = plt.subplots(figsize=(10, 10))  # Set the size of the figure
show(elevras, ax=ax, title="Elevation Raster Visualization")  # Display the raster

plt.show()
```



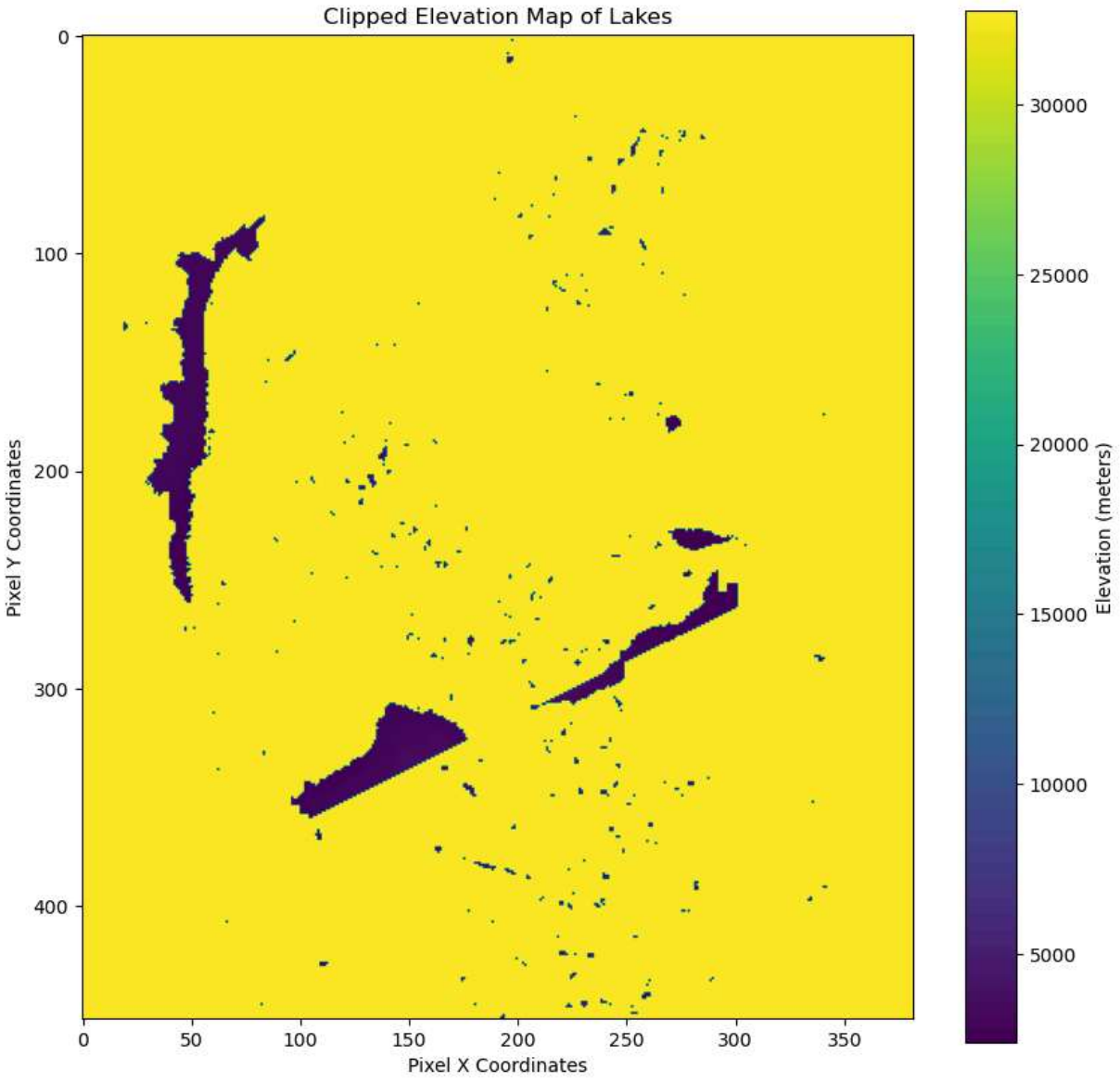
In [62]:
```
from rasterio.mask import mask
```

In [89]:
```
lakes_geom = lakes.geometry.values  # Extract the geometry in a format suitable for
out_image, out_transform = mask(elevras, lakes_geom, crop=True)
clipped_ras = rasterio.open('clipped_raster.tif', 'w', driver='GTiff',
                            height=out_image.shape[1],
                            width=out_image.shape[2],
                            count=1,
                            dtype=out_image.dtype,
                            crs=elevras.crs,
                            transform=out_transform)
clipped_ras.write(out_image[0], 1)
clipped_ras.close()
```

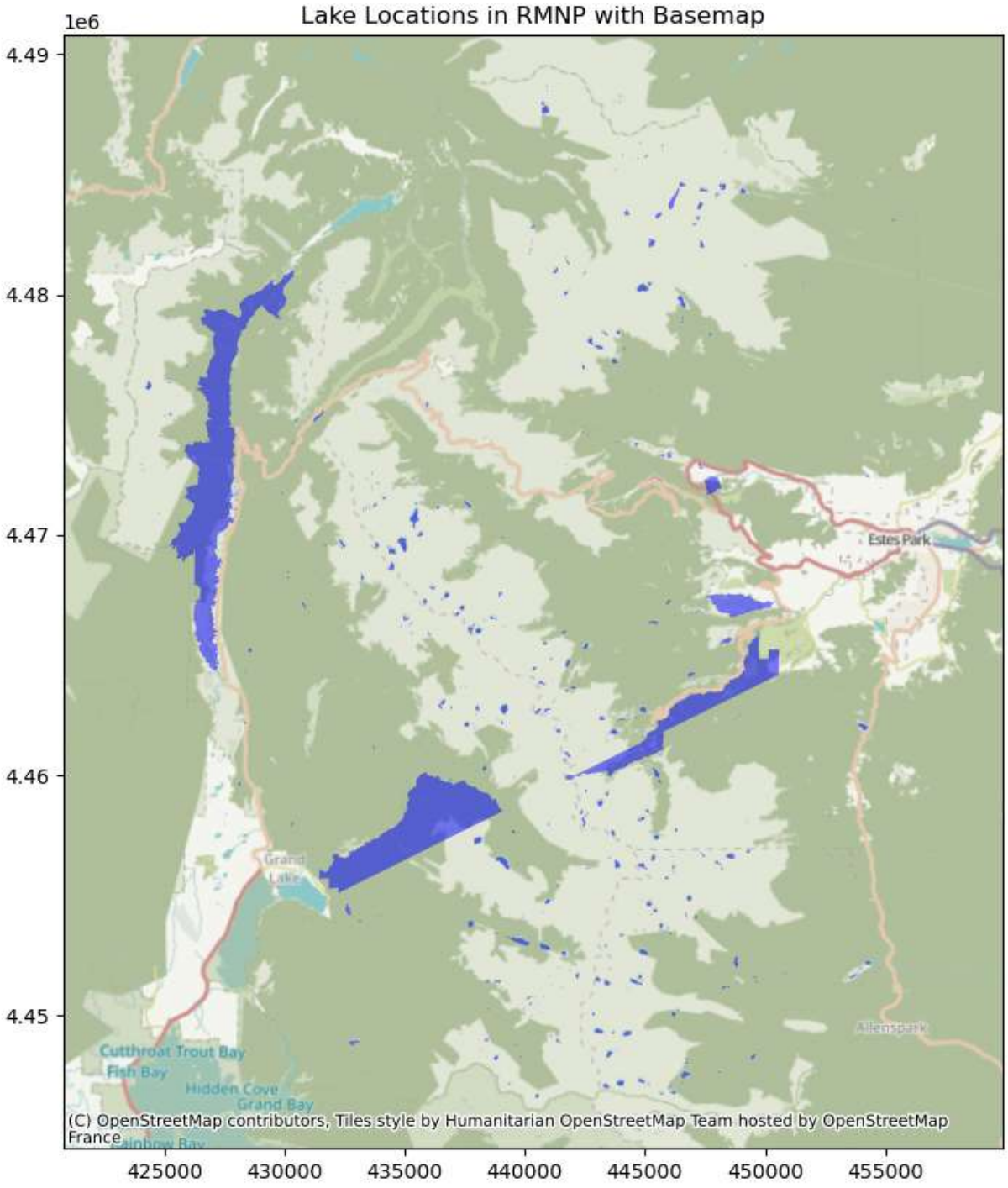
In [90]:
```
with rasterio.open('clipped_raster.tif') as src:
    # Read the first and only band
    image = src.read(1)
```

```
    # Setup the figure and axes
    plt.figure(figsize=(10, 10))

    # Show the image, using a colormap and a color bar
    plt.imshow(image, cmap='viridis')  # You can change 'viridis' to other colormap
    plt.colorbar(label='Elevation (meters)')  # Adjust label as appropriate
    plt.title('Clipped Elevation Map of Lakes')
    plt.xlabel('Pixel X Coordinates')
    plt.ylabel('Pixel Y Coordinates')
    plt.show()
```



In [68]:
```
import contextily as ctx
fig, ax = plt.subplots(figsize=(10, 10))
lakes.plot(ax=ax, alpha=0.5, color='blue')
ctx.add_basemap(ax, crs=lakes.crs.to_string())
ax.set_title('Lake Locations in RMNP with Basemap')
plt.show()
```



In [69]:
```
pip install rasterstats
```

```
Collecting rasterstats
  Downloading rasterstats-0.19.0-py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: affine in c:\users\xfour\anaconda3\lib\site-packages
(from rasterstats) (2.4.0)
Requirement already satisfied: click>7.1 in c:\users\xfour\anaconda3\lib\site-packag
es (from rasterstats) (8.1.7)
Requirement already satisfied: cligj>=0.4 in c:\users\xfour\anaconda3\lib\site-packa
ges (from rasterstats) (0.7.2)
Requirement already satisfied: fiona in c:\users\xfour\anaconda3\lib\site-packages
(from rasterstats) (1.9.5)
Requirement already satisfied: numpy>=1.9 in c:\users\xfour\anaconda3\lib\site-packa
ges (from rasterstats) (1.26.4)
Requirement already satisfied: rasterio>=1.0 in c:\users\xfour\anaconda3\lib\site-pa
ckages (from rasterstats) (1.3.10)
Collecting simplejson (from rasterstats)
  Downloading simplejson-3.19.2-cp311-cp311-win_amd64.whl.metadata (3.2 kB)
Requirement already satisfied: shapely in c:\users\xfour\anaconda3\lib\site-packages
(from rasterstats) (2.0.1)
Requirement already satisfied: colorama in c:\users\xfour\anaconda3\lib\site-package
s (from click>7.1->rasterstats) (0.4.6)
Requirement already satisfied: attrs in c:\users\xfour\anaconda3\lib\site-packages
(from rasterio>=1.0->rasterstats) (23.1.0)
Requirement already satisfied: certifi in c:\users\xfour\anaconda3\lib\site-packages
(from rasterio>=1.0->rasterstats) (2024.2.2)
Requirement already satisfied: snuggs>=1.4.1 in c:\users\xfour\anaconda3\lib\site-pa
ckages (from rasterio>=1.0->rasterstats) (1.4.7)
Requirement already satisfied: click-plugins in c:\users\xfour\anaconda3\lib\site-pa
ckages (from rasterio>=1.0->rasterstats) (1.1.1)
Requirement already satisfied: setuptools in c:\users\xfour\anaconda3\lib\site-packa
ges (from rasterio>=1.0->rasterstats) (68.2.2)
Requirement already satisfied: six in c:\users\xfour\anaconda3\lib\site-packages (fr
om fiona->rasterstats) (1.16.0)
Requirement already satisfied: pyparsing>=2.1.6 in c:\users\xfour\anaconda3\lib\site
-packages (from snuggs>=1.4.1->rasterio>=1.0->rasterstats) (3.0.9)
Downloading rasterstats-0.19.0-py3-none-any.whl (16 kB)
Downloading simplejson-3.19.2-cp311-cp311-win_amd64.whl (75 kB)
   ---------------------------------------- 0.0/75.3 kB ? eta -:--:--
   ---------------------------------------- 75.3/75.3 kB 2.0 MB/s eta 0:00:00
Installing collected packages: simplejson, rasterstats
Successfully installed rasterstats-0.19.0 simplejson-3.19.2
Note: you may need to restart the kernel to use updated packages.
```

In [71]:
```python
from rasterstats import zonal_stats
```

In [72]:
```python
lake_shp = r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\RMNP_Lakes.sh
lakes = gpd.read_file(lake_shp)
elevf = r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\RMNPelev.tif"

# Calculate zonal statistics
stats = zonal_stats(lakes, elevf, stats="mean")

# Add the average elevation to the lakes GeoDataFrame
lakes['avg_elevation'] = [stat['mean'] for stat in stats]
```

```
# Now you can view the updated GeoDataFrame
print(lakes[['avg_elevation']])
```

```
     avg_elevation
0              NaN
1      3515.000000
2      3474.000000
3      3472.250000
4              NaN
..             ...
730    3364.000000
731    3377.000000
732            NaN
733            NaN
734    3404.333333

[735 rows x 1 columns]
```

In [77]:
```
elk_shp = r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\Elk843cow.shp"
elk = gpd.read_file(elk_shp)

# Open the elevation raster
with rasterio.open(elevf) as elev_ras:
    # Extracting coordinates and transforming them into the raster's coordinate sys
    # if necessary (though both should be in the same CRS)
    elk['elevation'] = [list(elev_ras.sample([(geom.x, geom.y)]))[0][0] for geom in

# View the result to ensure it's correct
print(elk[['elevation']])
```

```
     elevation
0        32767
1        32767
2        32767
3        32767
4        32767
..         ...
422      32767
423      32767
424      32767
425      32767
426      32767

[427 rows x 1 columns]
```

In [87]:
```
lakes.to_file(r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\LakesEl.sh
elk.to_file(r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\Elk843El.shp
```

```
C:\Users\xfour\AppData\Local\Temp\ipykernel_12668\2084059619.py:1: UserWarning: Colu
mn names longer than 10 characters will be truncated when saved to ESRI Shapefile.
  lakes.to_file(r"C:\Users\xfour\Downloads\RMNPDataLesson4\RMNPDataLesson4\LakesEl.s
hp")
```