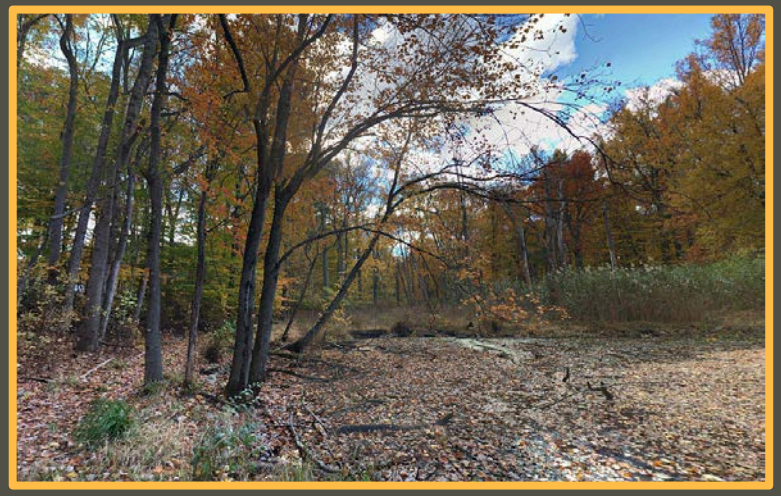
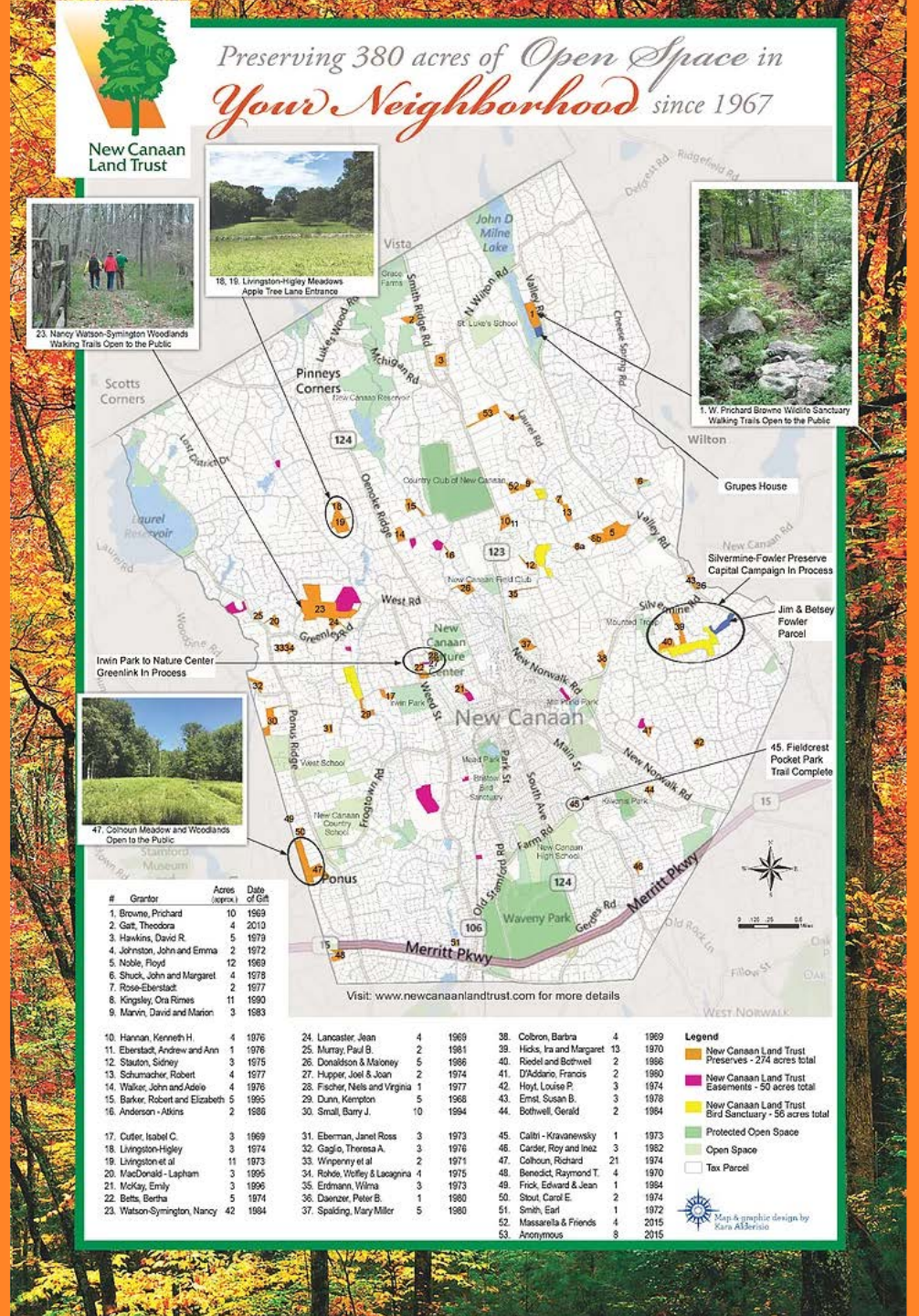
UPDATING THE NEW CANAAN LAND TRUST ANNUAL PROPERTY MAP USING PYTHON



Luke Menard
Yale School of Forestry and Environmental Studies
Geospatial Software Design, Fall 2016

BACKGROUND AND OBJECTIVES



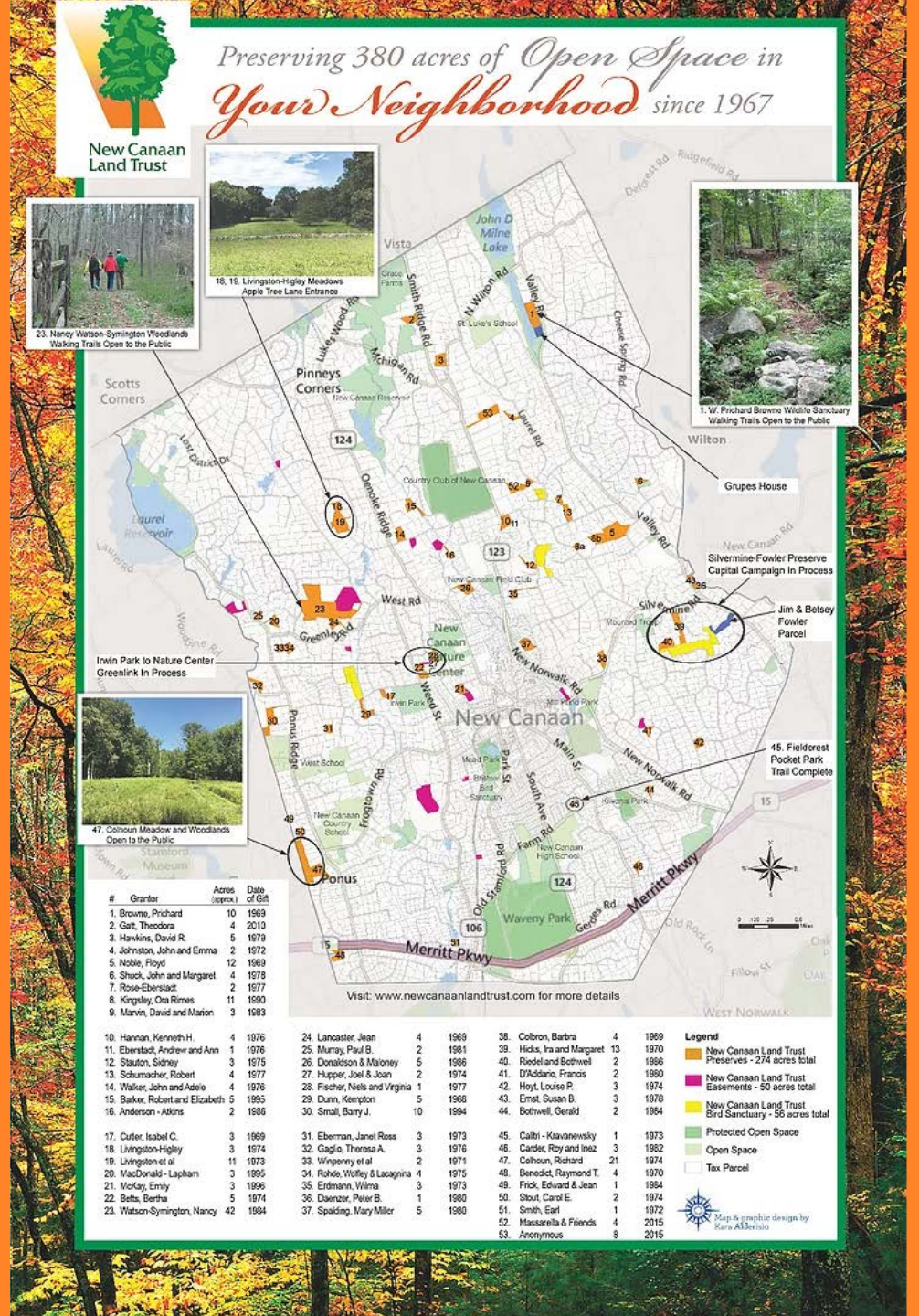
#	Grantor	Acres (approx.)	Date of Gift
1.	Browne, Prichard	10	1969
2.	Gatt, Theodora	4	2010
3.	Hawkins, David R.	5	1979
4.	Johnston, John and Emma	2	1972
5.	Noble, Floyd	12	1969
6.	Shuck, John and Margaret	4	1978
7.	Rose-Eberstadt	2	1977
8.	Kingsley, Ora Rimes	11	1990
9.	Marvin, David and Marion	3	1983
10.	Hannan, Kenneth H.	4	1976
11.	Eberstadt, Andrew and Ann	1	1976
12.	Stauton, Sidney	3	1975
13.	Schumacher, Robert	4	1977
14.	Walker, John and Adele	4	1976
15.	Barker, Robert and Elizabeth	5	1995
16.	Anderson - Atkins	2	1986
17.	Cutler, Isabel C.	3	1969
18.	Livingston-Higley	3	1974
19.	Livingston et al	11	1973
20.	MacDonald - Lapham	3	1995
21.	McKay, Emily	3	1996
22.	Betts, Bertha	5	1974
23.	Watson-Symington, Nancy	42	1984
24.	Lancaster, Jean	4	1969
25.	Murray, Paul B.	2	1981
26.	Donaldson & Maloney	5	1986
27.	Hupper, Joel & Joan	2	1974
28.	Fischer, Nells and Virginia	1	1977
29.	Dunn, Kempton	5	1968
30.	Small, Barry J.	10	1994
31.	Eberman, Janet Ross	3	1973
32.	Gaglio, Theresa A.	3	1976
33.	Winpenny et al	2	1971
34.	Rohde, Wolfley & Lacagnina	4	1975
35.	Erdmann, Wilma	3	1973
36.	Daenzer, Peter B.	1	1980
37.	Spalding, Mary Miller	5	1980
38.	Colbron, Barbra	4	1969
39.	Hicks, Ira and Margaret	13	1970
40.	Riedel and Bothwell	2	1986
41.	D'Addario, Francis	2	1980
42.	Heyl, Louise P.	3	1974
43.	Ernst, Susan B.	3	1978
44.	Bothwell, Gerald	2	1984
45.	Calitri - Kravanewsky	1	1973
46.	Carder, Roy and Inez	3	1982
47.	Colhoun, Richard	21	1974
48.	Benedict, Raymond T.	4	1970
49.	Frick, Edward & Jean	1	1984
50.	Stout, Carol E.	2	1974
51.	Smith, Earl	1	1972
52.	Massarella & Friends	4	2015
53.	Anonymous	8	2015

BACKGROUND

The New Canaan Land Trust releases an annual property map, which is distributed in hard copy to land trust members and town residents. Presently, the map provides a visual representation of the preserves, easements, and bird sanctuaries owned by the land trust. It also includes an extended legend with the grantor(s) and approximate acreage of each property.

To construct the property map in the past, the land trust enlisted a third-party mapping and graphic design consultant. New Canaan taxlot and parcel data is overlain on a gray canvas basemap. New Canaan Land Trust properties and municipal open space is color coordinated according to function. Key properites are highlighted and an image of each is included along the map border.

The reverse side of the paper map provides aerial photographs of two flagship land trust properties, the W. Pritchard Browne Wildlife Sanctuary and the Nancy Watson-Symington Woodlands. The images are superimposed with a simple outline of the walking trails on the property.



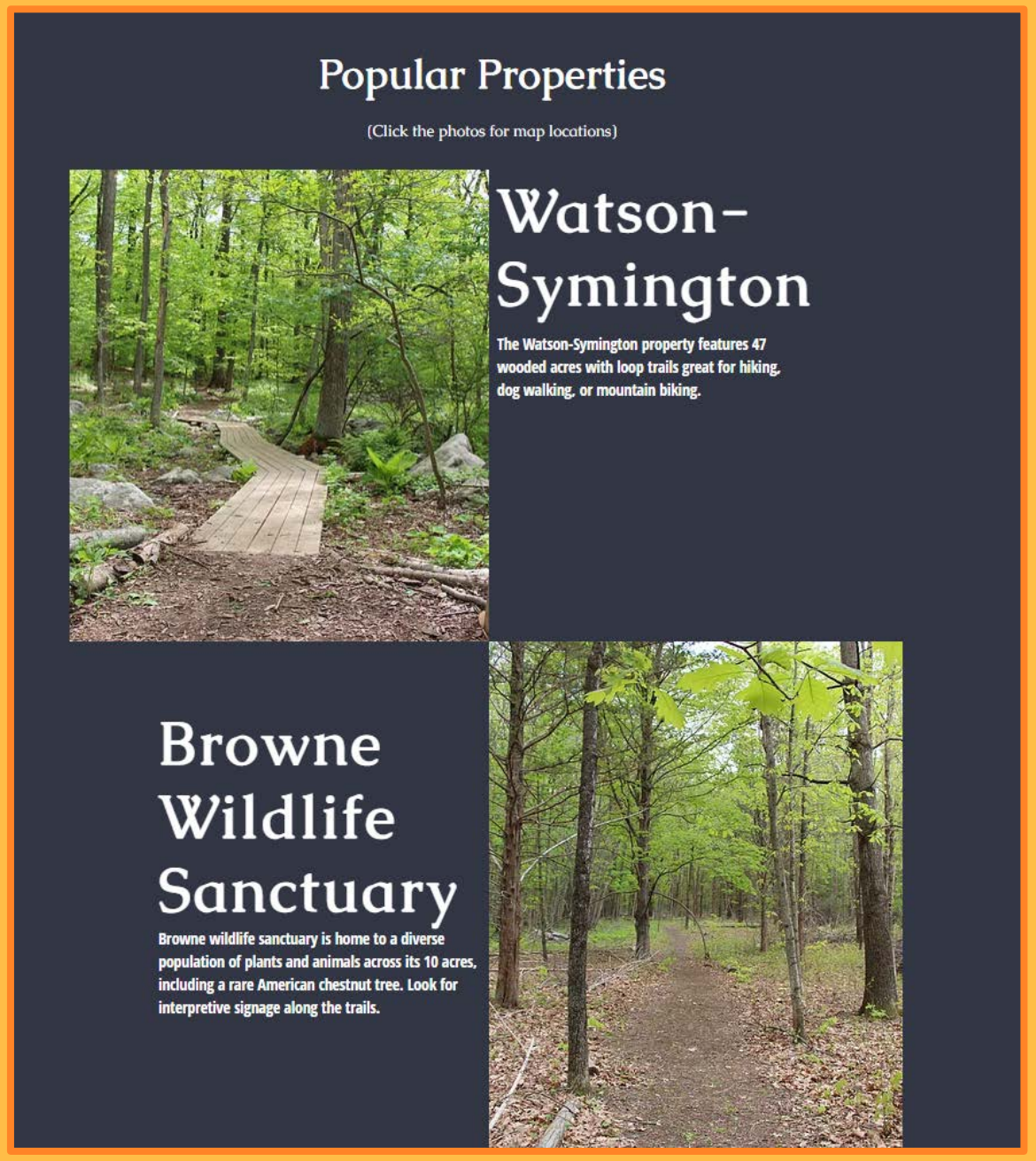
#	Grantor	Acres (approx.)	Date of Gift
1.	Browne, Prichard	10	1969
2.	Gatt, Theodora	4	2010
3.	Hawkins, David R.	5	1979
4.	Johnston, John and Emma	2	1972
5.	Noble, Floyd	12	1969
6.	Shuck, John and Margaret	4	1978
7.	Rose-Eberstadt	2	1977
8.	Kingsley, Ora Rimes	11	1990
9.	Marvin, David and Marion	3	1983
10.	Hannan, Kenneth H.	4	1976
11.	Eberstadt, Andrew and Ann	1	1976
12.	Stauton, Sidney	3	1975
13.	Schumacher, Robert	4	1977
14.	Walker, John and Adele	4	1976
15.	Barker, Robert and Elizabeth	5	1995
16.	Anderson - Atkins	2	1986
17.	Cutler, Isabel C.	3	1969
18.	Livingston-Higley	3	1974
19.	Livingston et al	11	1973
20.	MacDonald - Lapham	3	1995
21.	McKay, Emily	3	1996
22.	Betts, Bertha	5	1974
23.	Watson-Symington, Nancy	42	1984
24.	Lancaster, Jean	4	1969
25.	Murray, Paul B.	2	1981
26.	Donaldson & Maloney	5	1986
27.	Hupper, Joel & Joan	2	1974
28.	Fischer, Nells and Virginia	1	1977
29.	Dunn, Kempton	5	1968
30.	Small, Barry J.	10	1994
31.	Eberman, Janet Ross	3	1973
32.	Gaglio, Theresa A.	3	1976
33.	Winpenny et al	2	1971
34.	Rohde, Wolfley & Lacagnina	4	1975
35.	Erdmann, Wilma	3	1973
36.	Daenzer, Peter B.	1	1980
37.	Spalding, Mary Miller	5	1980
38.	Colbron, Barbra	4	1969
39.	Hicks, Ira and Margaret	13	1970
40.	Riedel and Bothwell	2	1986
41.	D'Addario, Francis	2	1980
42.	Heyl, Louise P.	3	1974
43.	Ernst, Susan B.	3	1978
44.	Bothwell, Gerald	2	1984
45.	Calitri - Kravanewsky	1	1973
46.	Carder, Roy and Inez	3	1982
47.	Calhoun, Richard	21	1974
48.	Benedict, Raymond T.	4	1970
49.	Frick, Edward & Jean	1	1984
50.	Stout, Carol E.	2	1974
51.	Smith, Earl	1	1972
52.	Massarella & Friends	4	2015
53.	Anonymous	8	2015

BACKGROUND

All of the property boundaries depicted on the map are up to date and accurate. However, the property shapefiles that the land trust has in its posession are outdated, with some incorrect property boundaries and missing land holdings. In the past, this has resulted in the need to conduct post-hoc image editing following the creation of the map. The land trust has stressed some need to have updated land parcel shapefiles to increase their capacity to map their properites themselves and ensure that this additional editing work is unecessary in the future.

Anecdotal evidence suggests that values of properties increase following the introduction of a park or wildlife sanctuary in their proximity. Many New Canaan residents associated with the land trust believe that this positive relationship exists between New Canaan Land Trust land holdings and monetary values of the properties nearby these land parcels. However, no real consideration of this trend has taken place.

Additionally, the New Canaan Land Trust suggested an interest in a more structured method of identifying ecologically and recreationally valuable properties in the town to assist with the prioritization of land for future purchases.



OBJECTIVES

Through conversations with Mike Johnson, the Executive Direction of the New Canaan Land Trust, and exploration of available data, I identified the following as primary objectives for my work in Python:

1. Update the land trust's parcel shapefiles to more accurately represent currently holdings and remove the need for post-hoc image editing.

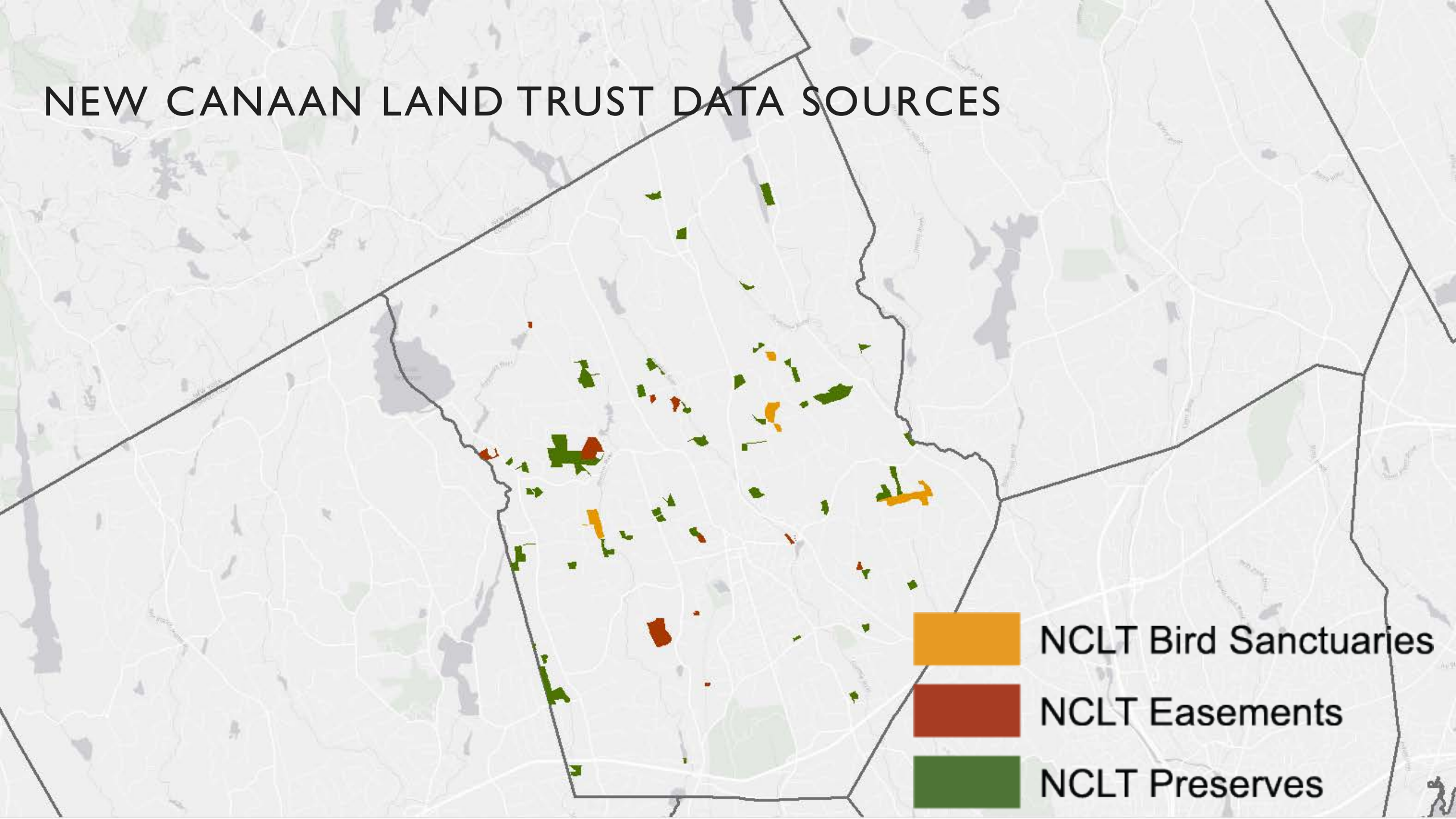
2. Conduct cursory analyses to provide a rough sense of property value fluctuations for proprties neighboring New Canaan Land Trust land holdings over time.

3. Generate interesting statistics for inclusion in flagship property profiles on the annual property map and New Canaan Land Trust website.

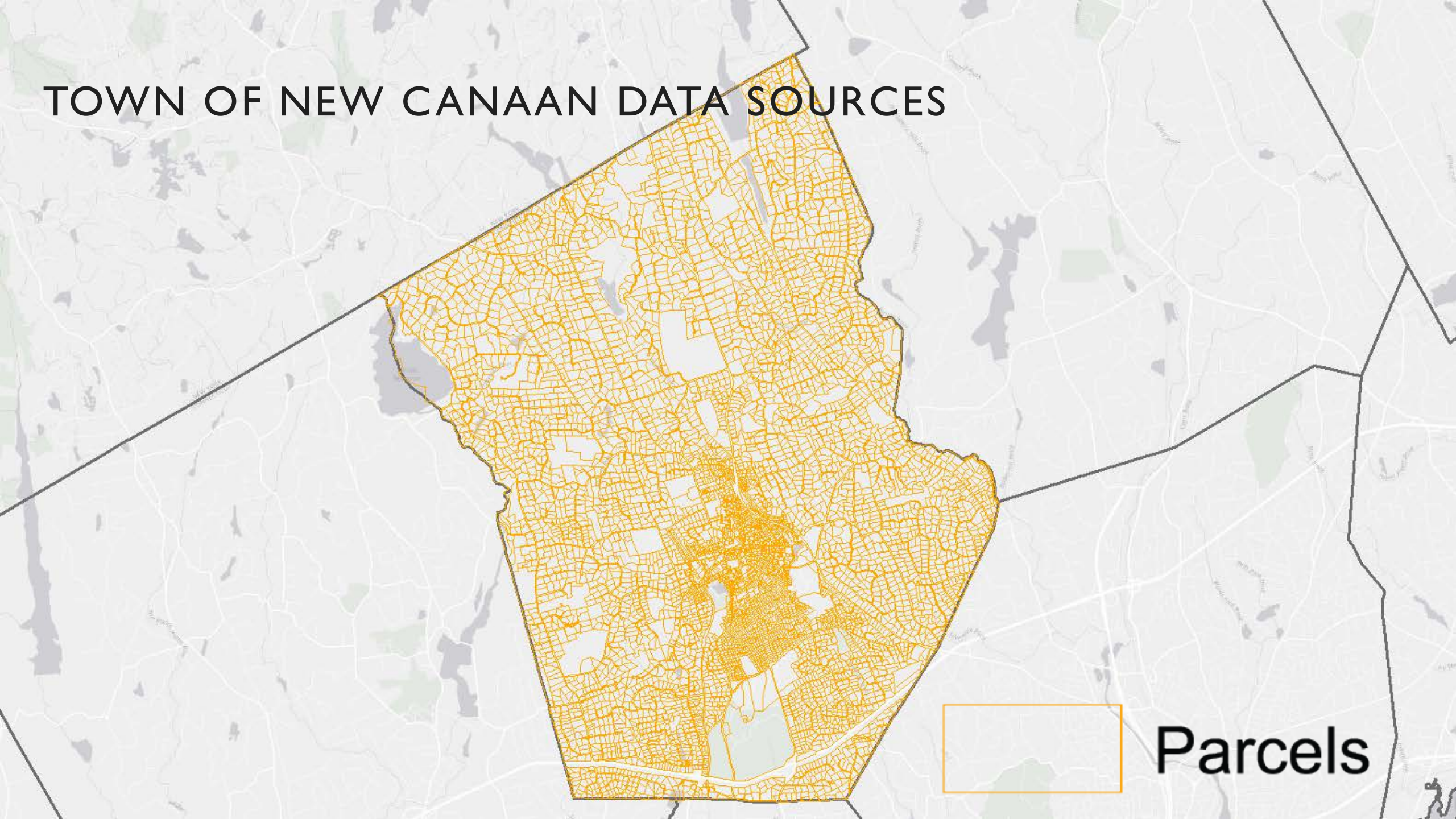
4. Conduct a site suitability assessment of New Canaan land parcels to identify high priority land for future purchases.

DATA COLLECTION

NEW CANAAN LAND TRUST DATA SOURCES



TOWN OF NEW CANAAN DATA SOURCES

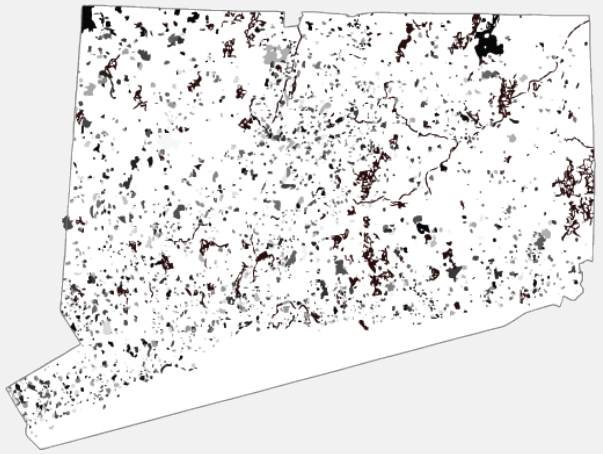


CONNECTICUT DEEP DATA SOURCES

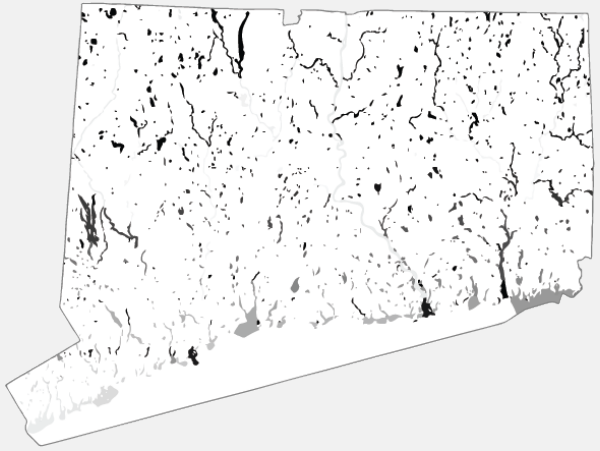
Connecticut Roads and Highways

Connecticut Town Boundaries

Municipal Protected Open Space



Water Bodies

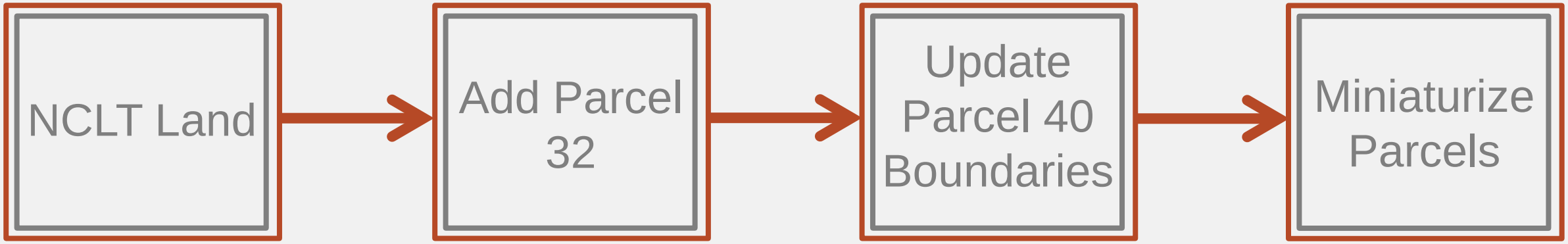


Statewide Hiking Trails

Connecticut Railroads

OBJECTIVE 1: UPDATE NCLT SHAPEFILES

OBJECTIVE I WORFLOW

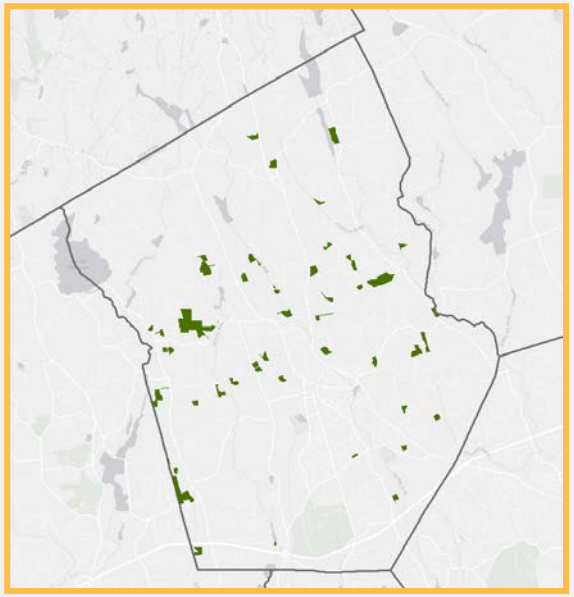


Objective: Update the land trust's parcel shapefiles to more accurately represent currently holdings and remove the need for post-hoc image editing.

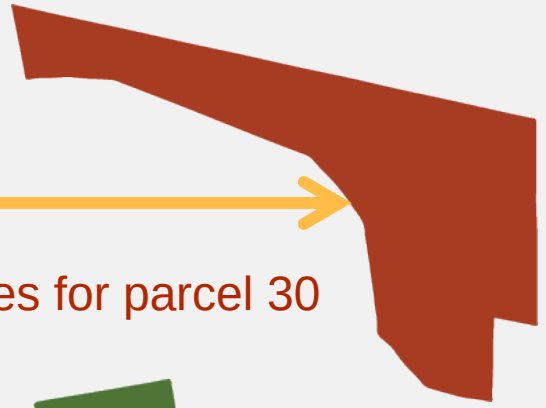
SAMPLE CODE

```
FEATURE (FID = 5)
     PART:
          (784592.808899,615926.5177)
          (784556.208923,615933.00769)
          (784538.827698,615940.116516)
          (784513.078918,615950.647522)
          (784493.33313,615977.678528)
          (784481.068909,615994.467529)
          (784473.954895,616010.112915)
          (784453.259277,616036.02771)
          (784453.248901,616055.647522)
          (784448.874878,616098.046875)
          (784436.658875,616216.457703)
          (784417.852905,616254.310486)
          (784397.770874,616287.285522)
          (784370.178894,616324.987671)
          (784346.193115,616337.665283)
          (784282.433472,616371.365479)
          (784223.307312,616402.616699)
          (784186.920288,616421.849121)
          (784133.138916,616449.28772)
          (784076.
          (784025.
          (784008.
          (784646.
          (784647.
          (784594.
          (784592.
```

```
# Loop through vertices
for feature in VertexCoordinates:

    # Reset vertex coordinates and add to PointsArray
    for vertex in feature:
        Point.X = vertex[0]
        Point.Y = vertex[1]
        PointsArray.add(Point)

    # Add the first vertex again to close the polygon
    PointsArray.add(PointsArray.getObject(0))

    # Create a polygon object from the array of vertex points
    Polygon = arcpy.Polygon(PointsArray)

    # Clear the array of vertex points for future use
    PointsArray.removeAll()

    # Append this polygon to Features list
    Features.append(Polygon)

# Create an output shapefile from Features list
arcpy.CopyFeatures_management(Features, OutputShapefile)
```



Start with NCLT parcels shapefile



Set vertices for parcel 30

Edit vertices for parcel 40

SAMPLE CODE

```
Point                       = arcpy.Point()

# Create list to hold all new features
NewFeatures                 = []

# Get attribute table and name of shape field for polygons created above
attributeTable              = arcpy.SearchCursor(OutputShapefile)
ShapeField                  = arcpy.Describe(OutputShapefile).shapeFieldName

# Loop through features
for nextRecord in attributeTable:
    # Get the next feature and centroid
    arcpy.AddMessage("FEATURE (FID = " + str(nextRecord.getValue("FID")) + ")")
    nextFeature             = nextRecord.getValue(ShapeField)
    Centroid                = nextFeature.centroid

    # Create array to hold the parts of new feature
    NewParts                = arcpy.Array()

    # Cycle through parts of the current feature
    for nextPart in nextFeature:
        arcpy.AddMessage("\tPART:")

        # Create array to hold the points for a new part to be created
        NewPoints           = arcpy.Array()

        # Cycle through original vertices and create new point from each
        for nextVertex in nextPart:
            if nextVertex:
                arcpy.AddMessage("\t\t\t(" + str(nextVertex.X) + ',' + str(nextVertex.Y) + ")")
                # If the next vertex is non-Null, create a new point and add it to the array of new points
                Point.X  = ((nextVertex.X * 2.0) + Centroid.X) / 1.5
                Point.Y  = ((nextVertex.Y * 2.0) + Centroid.Y) / 2.0
                NewPoints.add(Point)
            else:
                arcpy.AddMessage("\t\tHOLE: (beginning with a Null point)")
                # If the next vertex is Null, insert a new point that is also Null
                NewPoints.append(None)
        # After creating an array of new points for a given part, add it to this feature's array of new parts
        NewParts.append(NewPoints)

    # After creating an array of new parts for a given feature, create a new feature from that array
    newFeature = arcpy.Polygon(NewParts)

    # After creating a new feature, append it to a list of all new features
    NewFeatures.append(newFeature)

# Create the ne
arcpy.CopyFeatu

# Delete row an
del nextRecord
```

Miniaturize Polygons

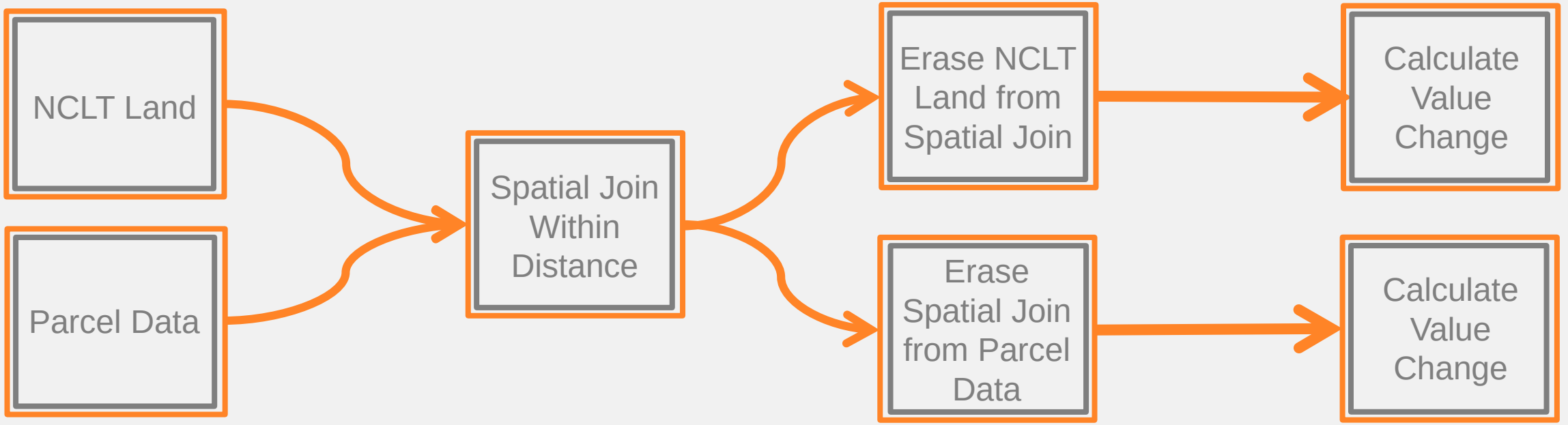
```
# CHARACTERIZE POLYGONS

#Find feature type
arcpy.AddField_management(OutputShapefile, "ShapeType", "TEXT", 10)
arcpy.CalculateField_management(OutputShapefile,"ShapeType","!shape.type!","PYTHON_9.3")

#Find length on shape in feet
arcpy.AddField_management(OutputShapefile, "Feet", "DOUBLE", 20, 5)
arcpy.CalculateField_management(OutputShapefile,"Feet","!shape.length@feet!","PYTHON_9.3")

#Find area of shape in acres
arcpy.AddField_management(OutputShapefile, "Acres", "DOUBLE", 20, 5)
arcpy.CalculateField_management(OutputShapefile,"Acres","!shape.area@acres!","PYTHON_9.3")
```

	FID	Shape *	Id	ShapeType	Feet	Acres
▶	0	Polygon	0	polygon	3368.80739	149.84814
	1	Polygon	0	polygon	2295.94282	79.57222
	2	Polygon	0	polygon	5873.54938	295.07193
	3	Polygon	0	polygon	3125.4176	107.51038
	4	Polygon	0	polygon	3647.61055	94.52174
	5	Polygon	0	polygon	2571.60124	45.53853
	6	Polygon	0	polygon	592.34946	2.36088

Characterize Polygons

OBJECTIVE III: MAP PROPERTY VALUE CHANGE

OBJECTIVE II WORFLOW

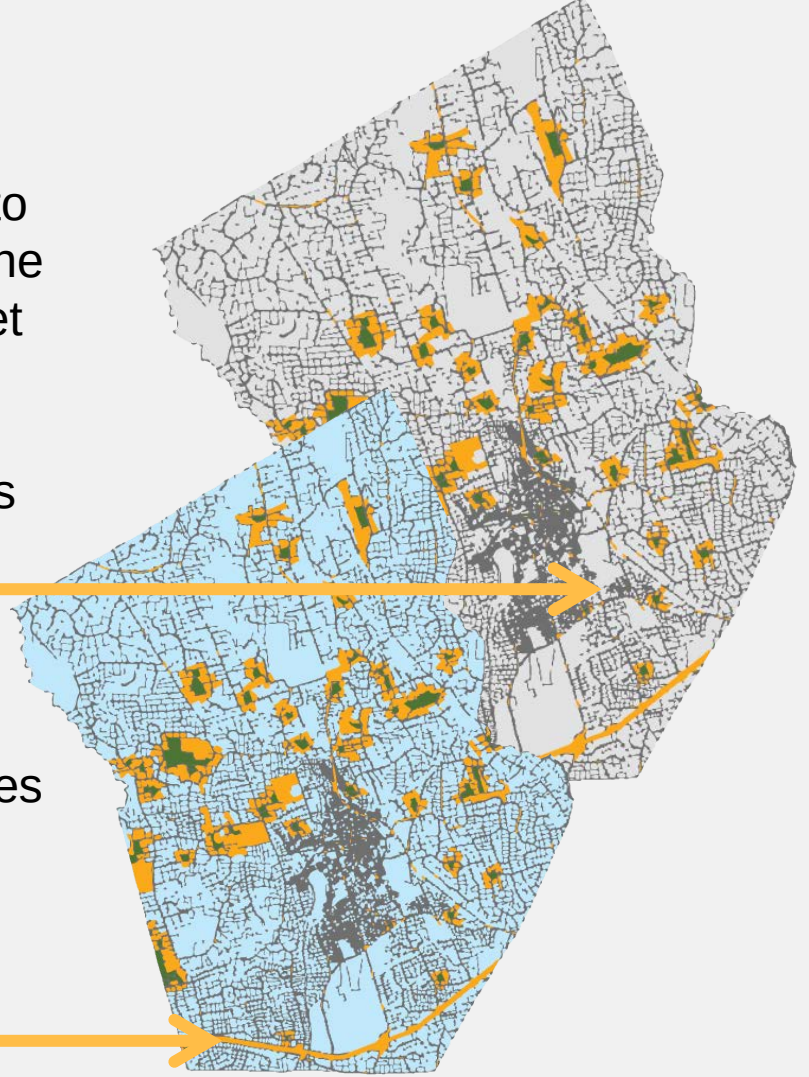


Objective: Conduct cursory analyses to provide a rough sense of property value fluctuations for proprties neighboring New Canaan Land Trust land holdings over time.

SAMPLE CODE

```
    targetFeatures = Parcels
    joinFeatures = NCLTLand
    outfc  = os.path.join(outWorkspace, "Parcels_NCLTLand")
    outfc1 = os.path.join(outWorkspace, "Parcels_NCLTLand1")

    fieldmappings = arcpy.FieldMappings()
    fieldmappings.addTable(targetFeatures)
    fieldmappings.addTable(joinFeatures)


#Spatially join parcel layer with NCLTLand layer
    #Parcels_JOIN_NCLTLand = Parcels[:-4] + "_spj.shp"
    Parcels_NCLTLand  = arcpy.SpatialJoin_analysis(targetFeatures, joinFeatures, outfc, '#', "KEEP_COMMON", fieldmappings, "WITHIN_A_DISTANCE", "100 FEET")
    Parcels_NCLTLand1 = arcpy.SpatialJoin_analysis(targetFeatures, joinFeatures, outfc1, '#', "KEEP_COMMON", fieldmappings, "WITHIN_A_DISTANCE", "100 FEET")

#Erase NCLT preserves from spatial join results
    #Join_ERASE_NCLTLand = Parcels[:-4] + "_erase.shp"
    eraseOutput1 = r"C:\Users\llm47\Desktop\Scratch\Join_ERASE_NCLTLand"
    eraseOutput1 = arcpy.Erase_analysis(Parcels_NCLTLand1, joinFeatures, eraseOutput1)

#Add a Value_Change field to the Erase shapefile, that will store the change in land values from the first to last land appraisal
    fieldName = "Val_Change"
    arcpy.AddField_management(eraseOutput1, fieldName, "Double")

#Calculate the change in land appraisal value between the final and first appraisals
    arcpy.CalculateField_management(eraseOutput1, fieldName, "[Total_Ap_7] - [Total_Ap_1]", "VB")


#_____________________________________________________________________________

#Erase spatial join results from Parcel Data
    #Parcels_ERASE_Join = Parcels[:-4] + "_erase2.shp"
    eraseOutput = r"C:\Users\llm47\Desktop\Scratch\Parcels_ERASE_Join"
    eraseOutput = arcpy.Erase_analysis(targetFeatures, Parcels_NCLTLand, eraseOutput)


#Add a Value_Change field to the Erase shapefile, that will store the change in land values from the first to last land appraisal
    arcpy.AddField_management(eraseOutput, fieldName, "Double")

#Calculate the change in land appraisal value between the final and first appraisals
    arcpy.CalculateField_management(eraseOutput, fieldName, "[Total_Ap_7] - [Total_Ap_1]", "VB")

#arcpy.CopyFeatures_management(eraseOutput, OutputMap)
```

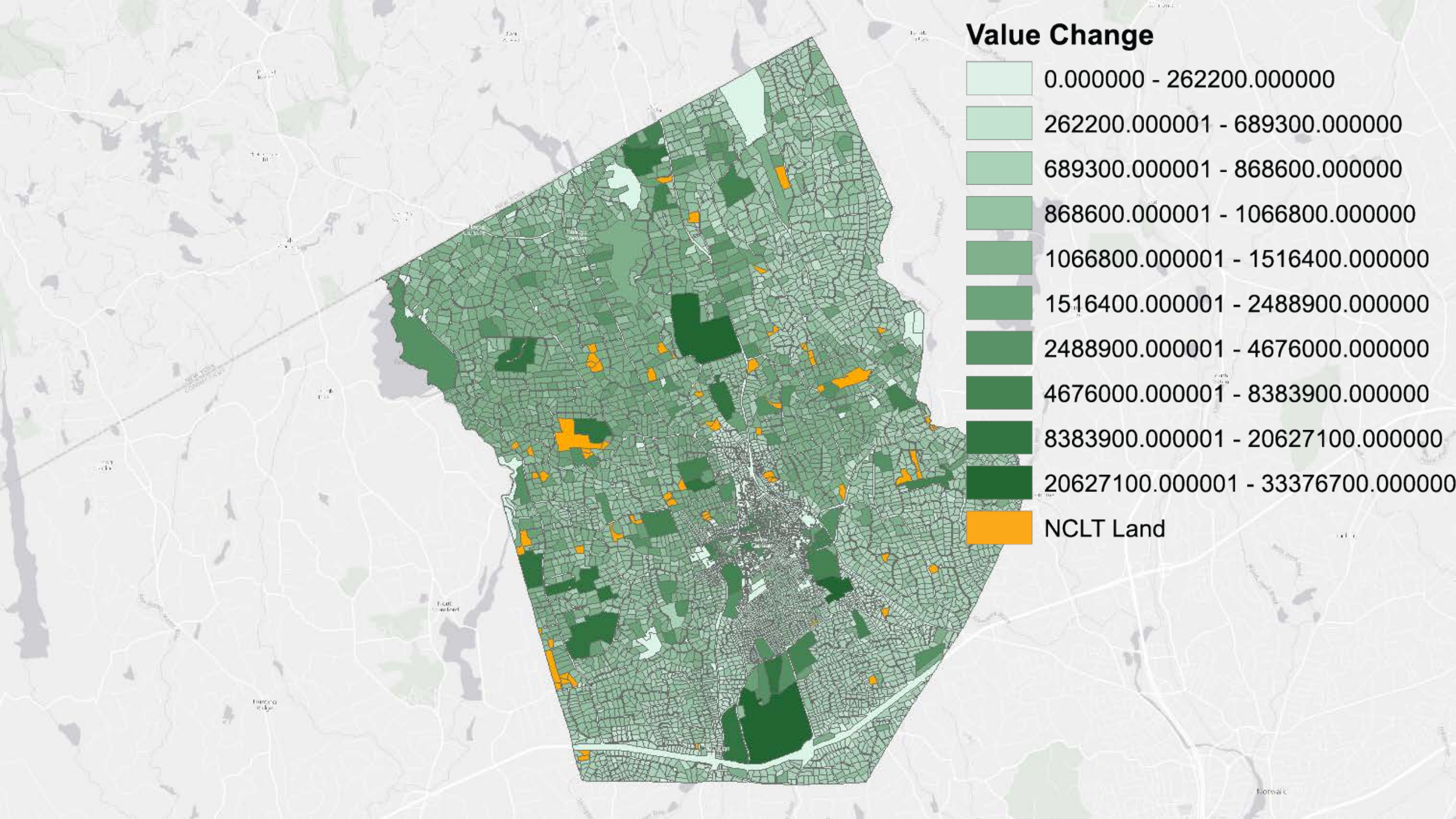
Use Spatial Join to link attributes of the NCLT land dataset to those parcels within 100 feet of land trust holdings



Use the Erase function to create separate shapefiles of neighboring parcels and all others

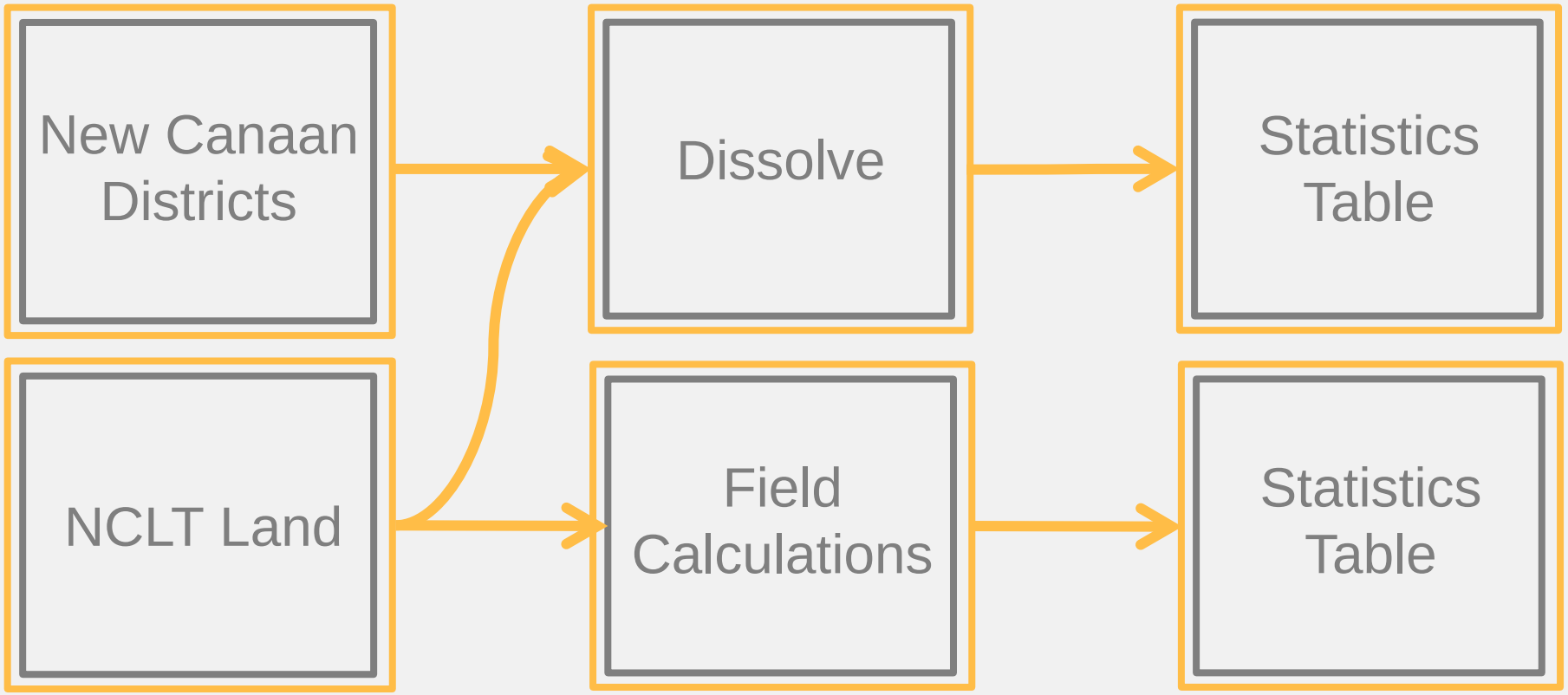
Val_Change
812700
719600
745100
794900
715200
795600
793400
795600

Add a value change field and subtract the initial land appraisal from the terminal appraisal


Value Change
0.000000 - 262200.000000
262200.000001 - 689300.000000
689300.000001 - 868600.000000
868600.000001 - 1066800.000000
1066800.000001 - 1516400.000000
1516400.000001 - 2488900.000000
2488900.000001 - 4676000.000000
4676000.000001 - 8383900.000000
8383900.000001 - 20627100.000000
20627100.000001 - 33376700.000000
NCLT Land

OBJECTIVE III: GENERATE STATISTICS

OBJECTIVE III WORFLOW



Objective: Generate interesting statistics for inclusion in flagship property profiles on the annual property map and New Canaan Land Trust website.

SAMPLE CODE

```
#Create Enumerators -- Thanks to Yanan Xin's 2013 work here
    OriginalEnumeration = arcpy.SearchCursor(Input,"","", "FID;NAME10;Total_Ap_7", "NAME10 D")
    TargetEnumeration = arcpy.UpdateCursor(OutputMap,"","", "NAME10;Total_Ap_7", "NAME10 A")


#Loop
    for record in TargetEnumeration:
        Name2 = record.getValue(Name)
        Appr2 = record.getValue(Appr)
        ID2 = record.getValue(ID)

        n=1
        while n:
            Next = OriginalEnumeration.next()
            NextName = Next.getValue(Name)
            NextAppr = Next.getValue(Appr)
            NextID = Next.getValue(ID)

            while ID2 != NextID:
                if Name2 == NextName:
                    NextAppr = Appr2 + NextAppr
                    #TargetEnumeration.deleteRow(record)
                    record = TargetEnumeration.next()
                    Name2 = record.getValue(Name)
                    Appr2 = record.getValue(Appr)
                    ID2 = record.getValue(ID)
                else:
                    break

        if ID2 == NextID:
            record.setValue(Appr, NextAppr)
            TargetEnumeration.updateRow(record)
            arcpy.AddMessage("The value of NCLT holdings in "+str(Name)+" is "+str(NextAppr)+"\n")
            nextRecord = OriginalEnumeration.reset()
            n = 0
```

	FID	Shape *	NAME10	SUM_Total_
▶	0	Polygon	New Canaan 1	55774200
	1	Polygon	New Canaan 2	17882700
	2	Polygon	New Canaan 3	6326500

Use enumerations and search cursors to dissolve the parcel data by New Canaan district and sum the amount of value of NCLT holdings in each

```
# CHARACTERIZE POLYGONS

#Find feature type
arcpy.AddField_management(OutputShapefile, "ShapeType", "TEXT", 10)
arcpy.CalculateField_management(OutputShapefile,"ShapeType","!shape.type!","PYTHON_9.3")

#Find length on shape in feet
arcpy.AddField_management(OutputShapefile, "Feet", "DOUBLE", 20, 5)
arcpy.CalculateField_management(OutputShapefile,"Feet","!shape.length@feet!","PYTHON_9.3")

#Find area of shape in acres
arcpy.AddField_management(OutputShapefile, "Acres", "DOUBLE", 20, 5)
arcpy.CalculateField_management(OutputShapefile,"Acres","!shape.area@acres!","PYTHON_9.3")
```

Use field calculator to characterize NCLT preserves and easements

	FID	Shape *	Id	ShapeType	Feet	Acres
▶	0	Polygon	0	polygon	3368.80739	149.84814
	1	Polygon	0	polygon	2295.94282	79.57222
	2	Polygon	0	polygon	5873.54938	295.07193
	3	Polygon	0	polygon	3125.4176	107.51038
	4	Polygon	0	polygon	3647.61055	94.52174
	5	Polygon	0	polygon	2571.60124	45.53853
	6	Polygon	0	polygon	592.34946	2.36088

OBJECTIVE IV: SITE SUITABILITY ASSESSMENT

CRITERIA

- **Criterion 1: Proximity to an existing New Canaan Land Trust Preserve**

  To maximize the land preserved in New Canaan, we want to prioritize new land that is not in close proximity to existing preserves.

- **Criterion 2: Proximity to highways and major roads**

  We want to preserve land that is far from major roads and highways to avoid noise and car traffic.

- **Criterion 3: Proximity to water bodies**

  We want to preserve land that is close to water, both to protect water bodies and to provide recreational opportunities to preserve visitors.

- **Criterion 4: Proximity to existing state trails**

  We want land close to state trails to maximize use for recreational opportunities for patrons.

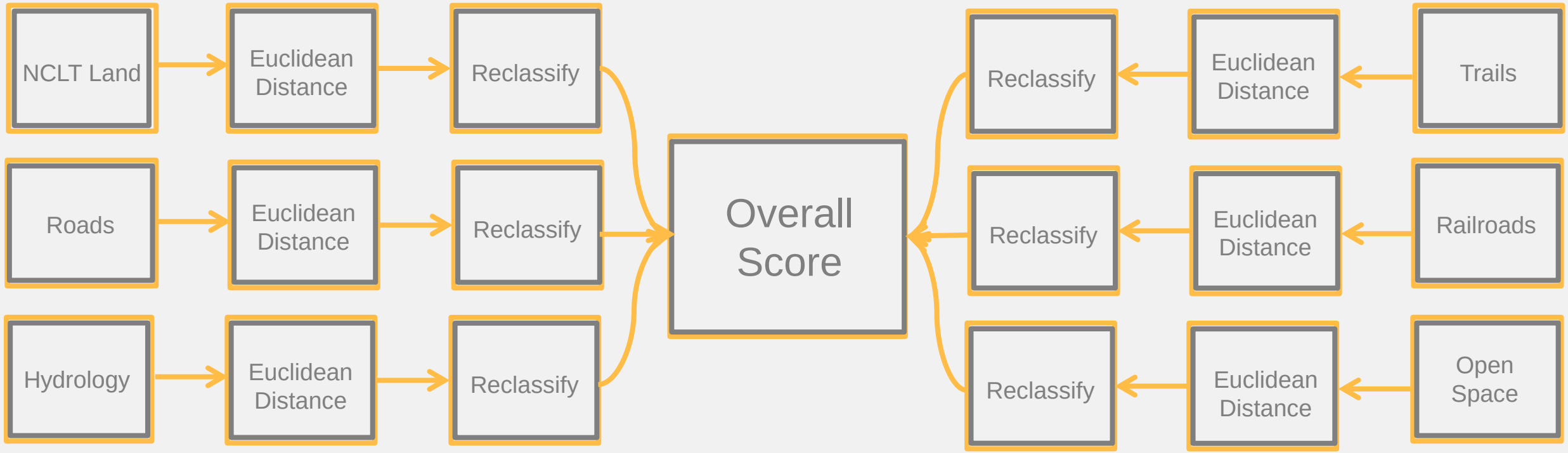
- **Criterion 5: Proximity to railroads**

  Because unused railroads can often be easily converted to walking and biking trails, we want to find land in close proximity to rail lines.

- **Criterion 6: Presence of municipal open space**

  Because municipal open space is already well regulated, we should prioritize other land.

OBJECTIVE IV WORFLOW

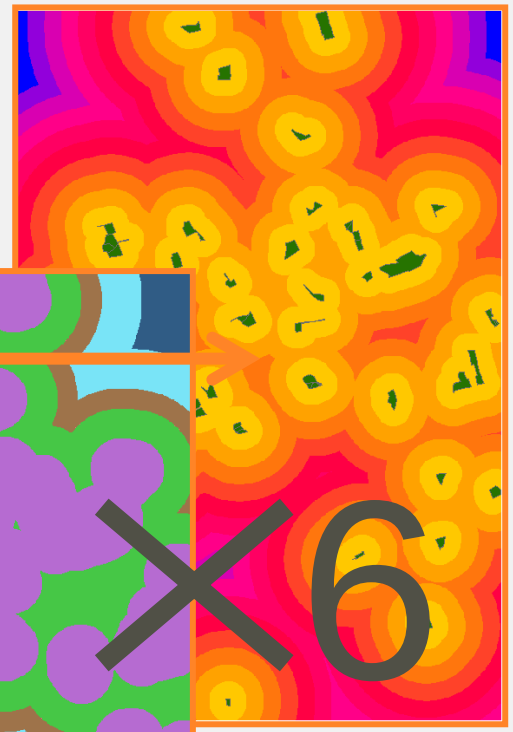


Objective: Conduct a site suitability assessment of New Canaan land parcels to identify high priority land for future purchases

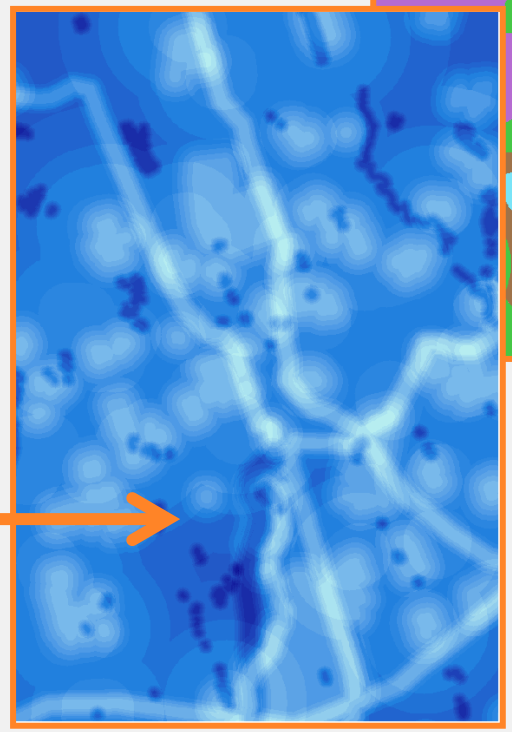
SAMPLE CODE

```
#Criterion 1: Use Euclidean Distance to estimate the distance away from existing New Caanan
#Land Trust preserves. Reclassify output to create a score grid, with higher scores allocated to
#those areas further from existing preserves.

        Pres_BUFFER = arcpy.sa.EucDistance(Preserves)
        Pres_SCORE  = arcpy.sa.Reclassify(Pres_BUFFER, "Value", RemapRange([[0, 1200, 0], [1200, 2400, 20], [2400, 3200, 40], [3200, 4800, 60], [4800, 6854, 80]]), "DATA")


#Criterion 2: Reclassify to select only state highways and interstates. Use Euclidean Distance to
#estimate the distance away from roads. Reclassify output to create a score grid, with higher
#scores allocated to those areas further from roads.

        #Roads_RECLASS = arcpy.sa.Reclassify(Roads,"Value","3 1; 4 1","NODATA")
        #Roads_BUFFER = arcpy.sa.EucDistance(Roads)
        Roads_SCORE = arcpy.sa.Reclassify(Roads_BUFFER, "Value", RemapRange([[0, 199, 0], [200, 299, 20], [300, 399, 40], [400, 450, 60], [450, 599, 80], [600, 130000, 100]]), "DATA")


#Criterion 3: Use Euclidean Distance to estimate the distance away from a water feature.
#Reclassify output to create a score grid, with higher scores allocated to regions closer to a water
#body.

        Hydro_BUFFER = arcpy.sa.EucDistance(Hydrology)
        Hydro_Score = arcpy.sa.Reclassify(Hydro_BUFFER, "Value", RemapRange([[0, 60, 100], [60, 120, 80], [120, 180, 60], [180, 240, 40], [240, 300, 20], [300,132623, 0]]), "DATA")

#Criterion 4: Use Euclidean Distance to estimate the distance away from an existing state trail.
#Reclassify output to create a score grid, with higher scores allocated to regions closer to a state
#trail.

        Trails_BUFFER = arcpy.sa.EucDistance(Trails)
        Trails_SCORE = arcpy.sa.Reclassify(Trails_BUFFER, "Value", RemapRange([[0, 200, 100], [200, 400, 80], [400, 600, 60], [600,800, 40], [800, 1000, 20], [1000,99000, 0]]), "DATA")


#Criterion 5: Use Euclidean Distance to estimate the distance away from a railroad. Reclassify
#output to create a score grid, with higher scores allocated to regions closer to a railroad.

        Railroads_BUFFER = arcpy.sa.EucDistance(Railroads)
        Railroads_SCORE = arcpy.sa.Reclassify(Railroads_BUFFER, "Value", RemapRange([[0, 200, 100], [200, 300, 80], [300, 400, 60], [400, 500, 40], [500, 600, 20], [600, 122000, 0]]), "DATA")


#Criterion 6: Use Euclidean Distance to estimate the distance away from existing municipal open space.
#Reclassify output to create a score grid, with higher scores allocated to those areas further from existing preserves.

        OS_BUFFER = arcpy.sa.EucDistance(OpenSpace)
        OS_SCORE  = arcpy.sa.Reclassify(OS_BUFFER, "Value", RemapRange([[0, 200, 0], [200, 400, 20], [400, 600, 40], [600,800, 60], [800, 1000, 80], [1000,130000, 100]]), "DATA")

# Create a layer of combined scores to estimate the best overall locations.
        Score = Pres_SCORE + Roads_SCORE + Hydro_Score + Trails_SCORE + Railroads_SCORE + OS_SCORE

#Select the areas with the highest scores.
        Score_RECLASS = arcpy.sa.Reclassify(Score, "Value", "420 1; 400 1; 380 1; 360 1; 340 1; 320 1; 300 1", "NODATA")
```

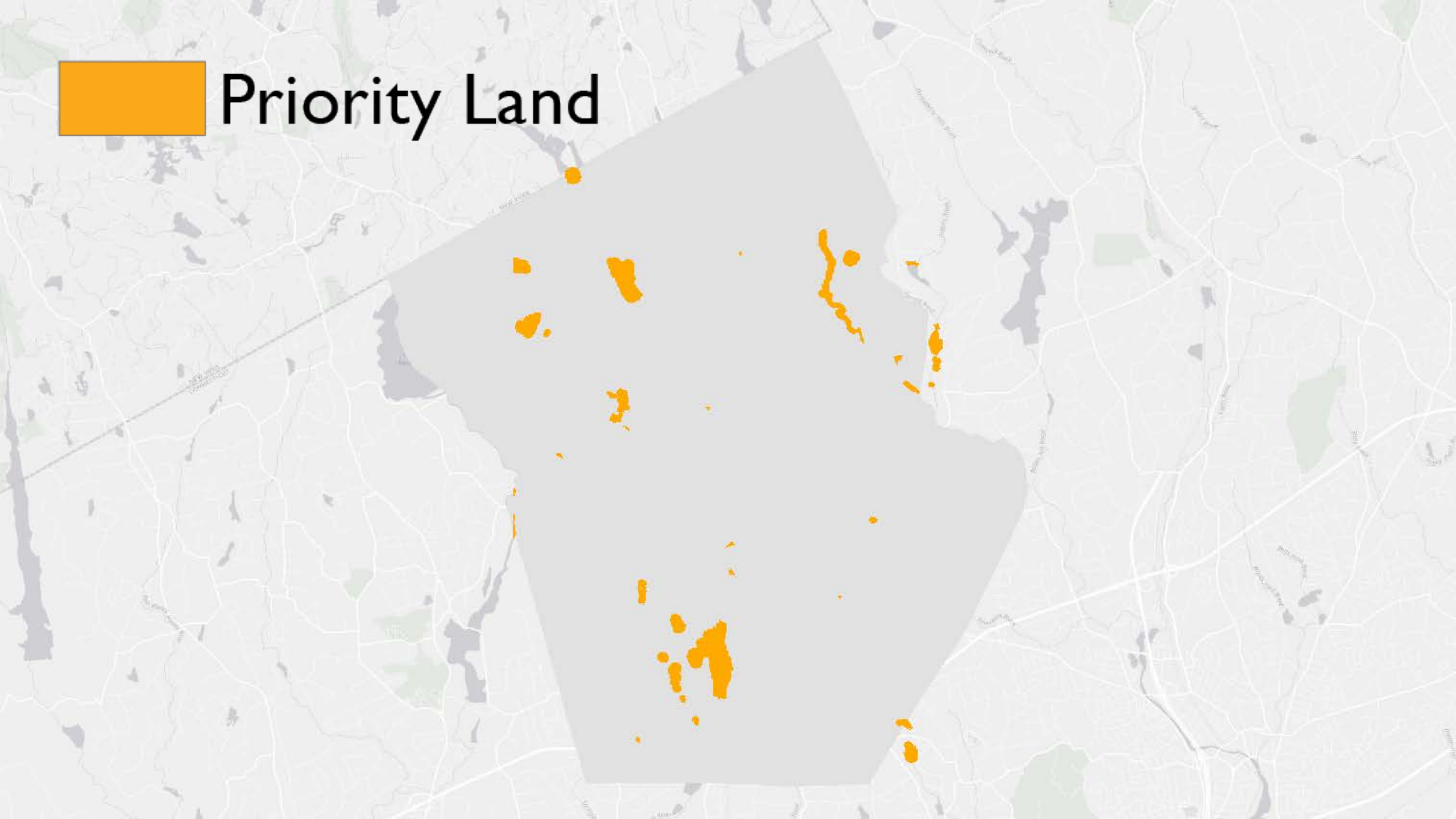
Use Euclidean Distance to calculate distance from features



Reclassify to calculate sub scores



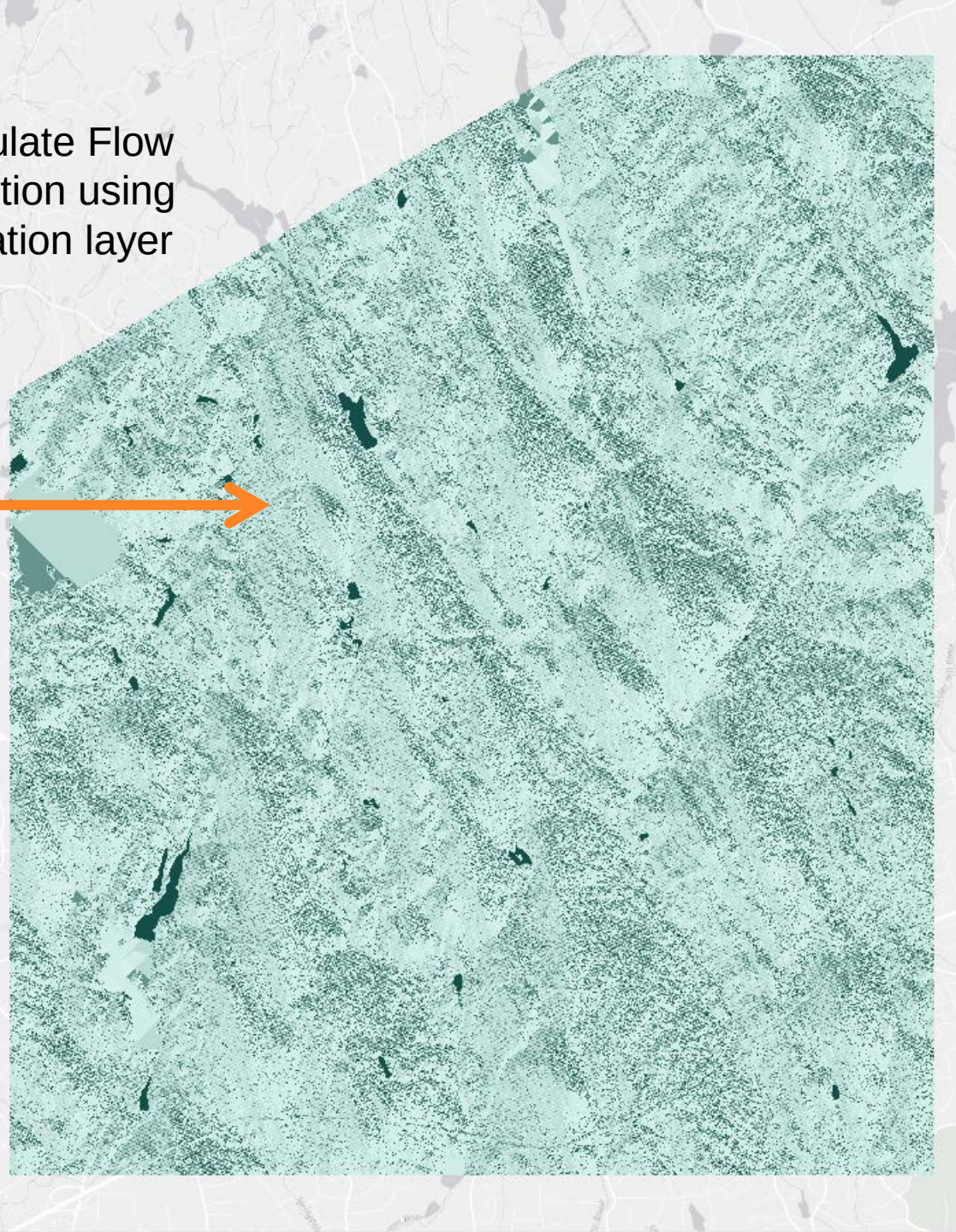
Calculate an overall score


Priority Land

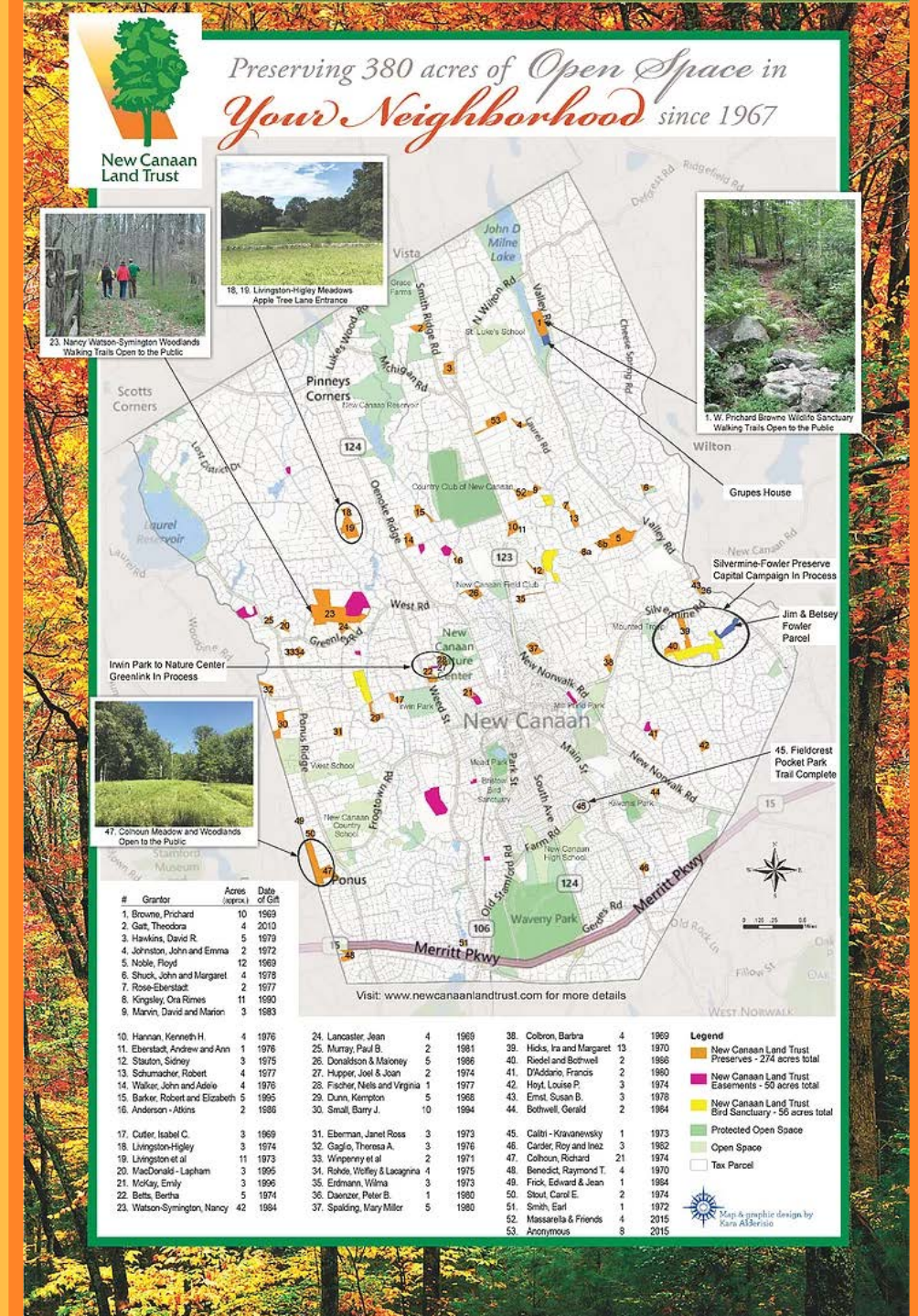
SAMPLE CODE

```
#Select the areas with the highest scores.
        Score_RECLASS = arcpy.sa.Reclassify(Score, "Value", "420 1; 400 1; 380 1; 360 1; 340 1; 320 1; 300 1", "NODATA")
        Directions = arcpy.sa.FlowDirection(Elevation)
        Accumulations = arcpy.sa.FlowAccumulation(Directions,Score_RECLASS)
        Accumulations.save(OutputName)

# Deactivate ArcGIS Spatial Analyst license
        arcpy.CheckInExtension("spatial")

    except Exception as e:
        arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
        exceptionreport = sys.exc_info()[2]
        fullermessage = traceback.format_tb(exceptionreport)[0]
        arcpy.AddError("at this location: \n\n" + fullermessage + "\n")
```

Calculate Flow Direction using Elevation layer



FUTURE PROJECTS AND ANALYSES



#	Grantor	Acres (approx.)	Date of Gift
1.	Browne, Prichard	10	1969
2.	Gatt, Theodora	4	2010
3.	Hawkins, David R.	5	1979
4.	Johnston, John and Emma	2	1972
5.	Noble, Floyd	12	1969
6.	Shuck, John and Margaret	4	1978
7.	Rose-Eberstadt	2	1977
8.	Kingsley, Ora Rimes	11	1990
9.	Marvin, David and Marion	3	1983
10.	Hannan, Kenneth H.	4	1976
11.	Eberstadt, Andrew and Ann	1	1976
12.	Stauton, Sidney	3	1975
13.	Schumacher, Robert	4	1977
14.	Walker, John and Adele	4	1976
15.	Barker, Robert and Elizabeth	5	1995
16.	Anderson - Atkins	2	1986
17.	Cutler, Isabel C.	3	1969
18.	Livingston-Higley	3	1974
19.	Livingston et al	11	1973
20.	MacDonald - Lapham	3	1995
21.	McKay, Emily	3	1996
22.	Betts, Bertha	5	1974
23.	Watson-Symington, Nancy	42	1984
24.	Lancaster, Jean	4	1969
25.	Murray, Paul B.	2	1981
26.	Donaldson & Maloney	5	1986
27.	Hupper, Joel & Joan	2	1974
28.	Fischer, Niels and Virginia	1	1977
29.	Dunn, Kempton	5	1968
30.	Small, Barry J.	10	1994
31.	Eberman, Janet Ross	3	1973
32.	Gaglio, Theresa A.	3	1976
33.	Winpenny et al	2	1971
34.	Rohde, Wolfley & Lacagnina	4	1975
35.	Erdmann, Wilma	3	1973
36.	Daenzer, Peter B.	1	1980
37.	Spalding, Mary Miller	5	1980
38.	Colbron, Barbra	4	1969
39.	Hicks, Ira and Margaret	13	1970
40.	Riedel and Bothwell	2	1986
41.	D'Addario, Francis	2	1980
42.	Hoyt, Louise P.	3	1974
43.	Ernst, Susan B.	3	1978
44.	Bothwell, Gerald	2	1984
45.	Calitri - Kravanewsky	1	1973
46.	Carder, Roy and Inez	3	1982
47.	Colhoun, Richard	21	1974
48.	Benedict, Raymond T.	4	1970
49.	Frick, Edward & Jean	1	1984
50.	Stout, Carol E.	2	1974
51.	Smith, Earl	1	1972
52.	Massarella & Friends	4	2015
53.	Anonymous	8	2015

DATA VISUALIZATION AND GRAPHIC DESIGN

- Use ArcGIS map outputs and Adobe Creative Suite to develop a compelling spatial data new annual map for the NCLT
  - Feature large map with updated parcels
  - Snapshots of flagship properties
  - Highlighted data found in phase 2 analyses

STORY MAPPING

- Use updated parcel info to create an ArcGIS Story Map that highlights properties and lives on the NCLT website

ADDITIONAL STATISTICAL ANALYSES

- Conduct a more rigorous analysis of the economic and ecological factors associated with New Canaan land preservation over time

FULL SCRIPTS

```
"""
The New Canaan Land Trust's shapefile of land parcels is outdated and the shapes of several of their
holdings have shifted over time.
Instead of changing the shapes of polygons in their source file, the team has historically done post hoc
alterations of the shapes holdings
in photoshop before distributing property maps to their members. This script will recreate NCLT
polygons and edit vertices to create
a more accurate polygon layer for use by the NCLT in the future.

"""

# Import necessary modules
import arcpy, sys, os, string, math, traceback

# Allow the output file to overwrite
arcpy.env.overwriteOutput = True

try:

   # Request user output of data type = Shapefile and direction = Output
   OutputShapefile  = arcpy.GetParameterAsText(0)
   arcpy.AddMessage("Output shapefile: \t" + OutputShapefile + "\n")
#_____________________________________________________________________________________
_____________


# Parcel 32 is missing from the current NCLT preserves shapefile
# Parcel 32 crosses the boundaries of multiple taxlots, so can't be recreated by simply selecting features
from the town's taxlot shapefile
# Create Parcel 32 by finding its xy coordinates and generating a new polygon
# The shape of parcel 40 is incorrect
# Create a new parcel 40 with updated vertices

#_____________________________________________________________________________________
_____________

   # Find coordinates of NCLT parcel polygon vertices
   #for row in arcpy.SearchCursor(InputClass, ["OID@", "SHAPE@"]):
   #   print("Feature {}:".format(row[0]))
   #    number = 0
   #    for part in row[1]:
   #       print("Part {}:".format(number))
   #       for point in part:
   #          if point:
   #             print("{}, {}".format(point.X, point.Y))
   #          else:
   #             print("Interior Ring:")
   #       number += 1
```

```
   # Create list of five polygonal features defined by the XY coordinates of its vertices from above
   VertexCoordinates = [ [ [797016.470969,631610.831902], [796645.287,631509.958088],
[796270.129942,632549.317886], [796707.377099,632650.656922], [797016.470969,631610.831902]
],
                [ [792838.717841,630514.372153], [792855.226995,629916.327089],
[792372.861162,629924.641049], [792401.479871,630378.780889], [792838.717841,630514.372153]
],
                [ [800106.204947,623087.599933], [800405.518981,622876.037036],
[800694.450161,622891.513055], [800488.049984,622463.520144], [799992.288836,622241.372951],
[799162.130063,622080.14099],  [798824.618927,622386.138081], [799570.625966,622602.74001],
[799449.622927,622715.010127], [799956.080903,622978.537159], [800106.204947,623087.599933] ],
                [ [802508.641087,617630.567868], [801802.16809,617371.153033],
[801795.20485,617711.179912],  [802080.909987,617828.462157], [801992.077159,618168.008066],
[802255.325977,618235.28385],  [802508.641087,617630.567868]
],
                [ [803012.999986,617758.552848], [802749.60386,617697.556907],
[802763.709146,618071.595857], [802379.988129,619082.151907], [802663.584018,619083.005908],
[802912.428009,618049.345902], [803012.999986,617758.552848]
],
                [ [784592.808899,615926.517639],
[784556.208801,615933.00769],[784538.827698,615940.116516],[784513.078918,615950.647461],[78
4493.333008,615977.678467],[784481.068909,615994.467529],[784473.954773,616010.112854],[7844
53.259155,616036.02771],[784453.248779,616055.647522],[784448.874756,616098.046814],[784436.
658813,616216.457642],[784417.852783,616254.310425],[784397.770935,616287.285522],[784370.17
8772,616324.98761],[784346.193054,616337.665344],[784282.433533,616371.365356],[784223.30719
,616402.616577],[784186.920288,616421.848999],[784133.138916,616449.287598],[784076.188782,6
16453.9375],[784025.458801,616450.667664],[784008.053955,616571.034302],[784646.648804,61638
5.867554],[784647.472046,616049.68512],[784594.670044,616063.583801],[784592.808899,615926.5
17639]],
                [ [801762.8888, 617594.1575], [801762.8888, 617597.9875], [801762.9588,
        617601.8075],[801763.0988,    617605.6175],[801763.2988,    617609.4075],[801763.5588,
        617613.1875],[801763.8888,    617616.9475],[801764.2788,    617620.6975],[801764.7388,
        617624.4275],[801765.2488,    617628.1475],[801765.8288,    617631.8375],[801766.4688,
        617635.5275],[801767.1688,    617639.1975],[801767.9288,    617642.8475],[801768.7588,
        617646.4775],[801769.6188,    617650.0975],[801770.5588,    617653.6975],[801771.5588,
        617657.2675],[801772.6088,    617660.8175],[801773.7288,    617664.3575],[801774.8988,
        617667.8675],[801776.1288,    617671.3575],[801777.4088,    617674.8175],[801778.7588,
        617678.2675],[801780.1488,    617681.6775],[801781.6088,    617685.0675],[801783.1188,
        617688.4375],[801784.6788,    617691.7775],[801786.2988,    617695.0775],[801787.9788,
        617698.3575],[801789.6988,    617701.6175],[801791.4788,    617704.8375],[801793.3188,
        617708.0275],[801795.2088,    617711.1775],[801797.1388,    617714.3075],[801799.1388,
        617717.3975],
                [801801.1788,   617720.4475],[801803.2688,    617723.4775],[801805.4188,
        617726.4575],[801807.6088,    617729.3975],[801809.8588,    617732.3075],[801812.1488,
        617735.1875],[801814.4988,    617738.0275],[801816.8888,    617740.8175],[801819.3288,
        617743.5775],[801821.8088,    617746.2875],[801824.3488,    617748.9575],[801826.9288,
```

```
        617751.5775],[801829.5588,     617754.1675],[801832.2288,     617756.6975],[801834.9488,
        617759.1975],[801837.7088,     617761.6375],[801840.5188,     617764.0375],[801843.3688,
        617766.3975],[801846.2588,     617768.6975],[801849.1988,     617770.9475],[801852.1788,
        617773.1475],[801855.2088,     617775.2975],[801858.2688,     617777.3975],[801861.3788,
        617779.4475],[801864.5288,     617781.4375],[801867.7188,     617783.3775],[801870.9488,
        617785.2675],[801874.2188,     617787.0875],[801877.5288,     617788.8575],[801880.8788,
        617790.5775],[801884.2688,     617792.2275],[801887.6988,     617793.8175],[801891.1688,
        617795.3475],[801894.6688,     617796.8275],[801898.2188,     617798.2275],[801901.8088,
        617799.5675],
                [801905.4288,    617800.8375],[801909.0988,     617802.047, [801912.7988,
        617803.1875],[801916.5388,     617804.2575],[801920.3388,     617805.2575],[801924.1588,
        617806.1775],[801928.0288,     617807.0275],[801931.9488,     617807.8075],[801933.9188,
        617808.1575],[801935.8988,     617808.4975],[801937.8988,     617808.8075],[801939.8988,
        617809.1075],[801941.9188,     617809.3775],[801943.9488,     617809.6275],[801945.9988,
        617809.8675],[801948.0488,     617810.0775],[801950.1188,     617810.2575],[801952.1988,
        617810.4275],[801954.3088,     617810.5675],[802080.9088,     617828.4575],[802084.7388,
        617845.2575],[801991.2088,     617950],[801975.2788, 617950],[802008.4888,
        617950],[802006.1988, 617950],[802017.2688, 617950],[802217.4188, 617950],[802231.5288,
        617950],[802248.7888, 617950],[802341.7088, 617932.0675],[802391.6788,
        617941.5875],[802408.1288,     617892.1175],[802501.7888,     617626.2875],[802396.9588,
        617608.6575],[802289.9588,     617587.6275],[802181.2688,     617565.7475],[802018.2988,
        617527.7175],[801817.9588,     617442.4975],
                [801818.4588,    617379.5575],[801815.9888,     617378.7375], [801812.6288,
        617377.3775],[801809.3688,     617375.7875],[801806.2288,     617373.9675],[801802.1688,
        617371.1575],[801777.3288,     617450.3775],[801763.0888,     617586.4475],[801762.9488,
        617590.3075],[801762.8888,     617594.1575]]]


# Create empty point
Point = arcpy.Point()

# Create empty array of points
PointsArray = arcpy.Array()

# Create empty list of polygons
Features = []

# Loop through vertices
for feature in VertexCoordinates:

   # Reset vertex coordinates and add to PointsArray
   for vertex in feature:
      Point.X = vertex[0]
      Point.Y = vertex[1]
      PointsArray.add(Point)

   # Add the first vertex again to close the polygon
   PointsArray.add(PointsArray.getObject(0))
```

```
        # Create a polygon object from the array of vertex points
        Polygon = arcpy.Polygon(PointsArray)

        # Clear the array of vertex points for future use
        PointsArray.removeAll()

        # Append this polygon to Features list
        Features.append(Polygon)

    # Create an output shapefile from Features list
    arcpy.CopyFeatures_management(Features, OutputShapefile)

#_____
_____

    # MINIATURIZE POLYGONS

    # Create object to hold new points
    Point                   = arcpy.Point()

    # Create list to hold all new features
    NewFeatures             = []

    # Get attribute table and name of shape field for polygons created above
    attributeTable          = arcpy.SearchCursor(OutputShapefile)
    ShapeField                      = arcpy.Describe(OutputShapefile).shapeFieldName

    # Loop through features
    for nextRecord in attributeTable:
        # Get the next feature and centroid
        arcpy.AddMessage("FEATURE (FID = " + str(nextRecord.getValue("FID")) + ")")
        nextFeature         = nextRecord.getValue(ShapeField)
        Centroid            = nextFeature.centroid

        # Create array to hold the parts of new feature
        NewParts            = arcpy.Array()

        # Cycle through parts of the current feature
        for nextPart in nextFeature:
            arcpy.AddMessage("\tPART:")

            # Create array to hold the points for a new part to be created
            NewPoints       = arcpy.Array()

            # Cycle through original vertices and create new point from each
            for nextVertex in nextPart:
                if nextVertex:
```

```
                arcpy.AddMessage("\t\t(" + str(nextVertex.X) + ',' + str(nextVertex.Y) + ")")
                # If the next vertex is non-Null, create a new point and add it to the array of new points
                Point.X  = ((nextVertex.X * 2.0) + Centroid.X) / 1.5
                Point.Y  = ((nextVertex.Y * 2.0) + Centroid.Y) / 2.0
                NewPoints.add(Point)
            else:
                arcpy.AddMessage("\t\tHOLE: (beginning with a Null point)")
                # If the next vertex is Null, insert a new point that is also Null
                NewPoints.append(None)
        # After creating an array of new points for a given part, add it to this feature's array of new parts
        NewParts.append(NewPoints)

    # After creating an array of new parts for a given feature, create a new feature from that array
    newFeature = arcpy.Polygon(NewParts)

    # After creating a new feature, append it to a list of all new features
    NewFeatures.append(newFeature)

# Create the new shapefile from that list of all new features
arcpy.CopyFeatures_management(NewFeatures, OutputShapefile)

# Delete row and update cursor objects to avoid locking attribute table
del nextRecord
del attributeTable

#_____________________________________________________________________________________
_____________

# CHARACTERIZE POLYGONS

#Find feature type
arcpy.AddField_management(OutputShapefile, "ShapeType", "TEXT", 10)
arcpy.CalculateField_management(OutputShapefile,"ShapeType","!shape.type!","PYTHON_9.3")

#Find length on shape in feet
arcpy.AddField_management(OutputShapefile, "Feet", "DOUBLE", 20, 5)
arcpy.CalculateField_management(OutputShapefile,"Feet","!shape.length@feet!","PYTHON_9.3")

#Find area of shape in acres
arcpy.AddField_management(OutputShapefile, "Acres", "DOUBLE", 20, 5)
arcpy.CalculateField_management(OutputShapefile,"Acres","!shape.area@acres!","PYTHON_9.3")


#_____________________________________________________________________________________
_____________________________________

except Exception as e:
    # If unsuccessful, end gracefully by indicating why
```

```
arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
# ... and where
exceptionreport = sys.exc_info()[2]
fullermessage   = traceback.format_tb(exceptionreport)[0]
arcpy.AddError("at this location: \n\n" + fullermessage + "\n")
```

```
"""
This script conducts a site suitability assessment of New Canaan land parcels to identify high priority
land for future purchases.

"""


# Import external modules
# Note that math modules isn't needed
import sys, os, string, arcpy, math, traceback

from arcpy import env
from arcpy.sa import *

# Check to see if Spatial Analyst license is available. If available, continue with script.
if arcpy.CheckExtension("spatial") == "Available":
   try:

# Activate ArcGIS Spatial Analyst license
      arcpy.CheckOutExtension("spatial")

# Read user inputs from dialog box

      Towns = arcpy.GetParameterAsText(0)
      Preserves = arcpy.GetParameterAsText(1)
      Roads = arcpy.GetParameterAsText(2)
      Hydrology = arcpy.GetParameterAsText(3)
      Trails = arcpy.GetParameterAsText(4)
      Railroads = arcpy.GetParameterAsText(5)
      OpenSpace = arcpy.GetParameterAsText(6)
      Elevation = arcpy.GetParameterAsText(7)
      OutputName = arcpy.GetParameterAsText(8)

# Reclassify Towns to identify only those of interest. Here, New Caanan is selected.
      Towns = arcpy.sa.Reclassify(Towns,"Value","90 1","NODATA")


#Criterion 1: Use Euclidean Distance to estimate the distance away from existing New Caanan
#Land Trust preserves. Reclassify output to create a score grid, with higher scores allocated to
#those areas further from existing preserves.

      Pres_BUFFER = arcpy.sa.EucDistance(Preserves)
```

```
    Pres_SCORE  = arcpy.sa.Reclassify(Pres_BUFFER, "Value", RemapRange([[0, 1200, 0], [1200, 2400, 20], [2400, 3200, 40], [3200, 4800, 60], [4800, 6834, 80]]), "DATA")


#Criterion 2: Reclassify to select only state highways and interstates. Use Euclidean Distance to
#estimate the distance away from roads. Reclassify output to create a score grid, with higher
#scores allocated to those areas further from roads.

    #Roads_RECLASS = arcpy.sa.Reclassify(Roads,"Value","3 1; 4 1","NODATA")
    Roads_BUFFER = arcpy.sa.EucDistance(Roads)
    Roads_SCORE = arcpy.sa.Reclassify(Roads_BUFFER, "Value", RemapRange([[0, 199, 0], [200, 299, 20], [300, 399, 40], [400, 450, 60], [450, 599, 80], [600, 130000, 100]]), "DATA")


#Criterion 3: Use Euclidean Distance to estimate the distance away from a water feature.
#Reclassify output to create a score grid, with higher scores allocated to regions closer to a water
#body.

    Hydro_BUFFER = arcpy.sa.EucDistance(Hydrology)
    Hydro_Score = arcpy.sa.Reclassify(Hydro_BUFFER, "Value", RemapRange([[0, 60, 100], [60, 120, 80], [120, 180, 60], [180, 240, 40], [240, 300, 20], [300,132623, 0]]), "DATA")

#Criterion 4: Use Euclidean Distance to estimate the distance away from an existing state trail.
#Reclassify output to create a score grid, with higher scores allocated to regions closer to a state
#trail.

    Trails_BUFFER = arcpy.sa.EucDistance(Trails)
    Trails_SCORE = arcpy.sa.Reclassify(Trails_BUFFER, "Value", RemapRange([[0, 200, 100], [200, 400, 80], [400, 600, 60], [600,800, 40], [800, 1000, 20], [1000,99000, 0]]), "DATA")


#Criterion 5: Use Euclidean Distance to estimate the distance away from a railroad. Reclassify
#output to create a score grid, with higher scores allocated to regions closer to a railroad.

    Railroads_BUFFER = arcpy.sa.EucDistance(Railroads)
    Railroads_SCORE = arcpy.sa.Reclassify(Railroads_BUFFER, "Value", RemapRange([[0, 200, 100], [200, 300, 80], [300, 400, 60], [400, 500, 40], [500, 600, 20], [600, 122000, 0]]), "DATA")


#Criterion 6: Use Euclidean Distance to estimate the distance away from existing municipal open space.
#Reclassify output to create a score grid, with higher scores allocated to those areas further from
existing preserves.
```

```
        OS_BUFFER = arcpy.sa.EucDistance(OpenSpace)
        OS_SCORE  = arcpy.sa.Reclassify(OS_BUFFER, "Value", RemapRange([[0, 200, 0], [200, 400, 20],
[400, 600, 40], [600,800, 60], [800, 1000, 80], [1000,130000, 100]]), "DATA")


# Create a layer of combined scores to estimate the best overall locations.
        Score = Pres_SCORE + Roads_SCORE + Hydro_Score + Trails_SCORE + Railroads_SCORE + OS_SCORE

#Select the areas with the highest scores.
        Score_RECLASS = arcpy.sa.Reclassify(Score, "Value", "420 1; 400 1; 380 1; 360 1; 340 1; 320 1; 300
1", "NODATA")
        Directions = arcpy.sa.FlowDirection(Elevation)
        Accumulations = arcpy.sa.FlowAccumulation(Directions,Score) #Note that I've gotten this to run
before, but for some reason, Arc crashes every time I use this set of files
        Accumulations.save(OutputName)

# Deactivate ArcGIS Spatial Analyst license
        arcpy.CheckInExtension("spatial")

    except Exception as e:
        arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
        exceptionreport = sys.exc_info()[2]
        fullermessage = traceback.format_tb(exceptionreport)[0]
        arcpy.AddError("at this location: \n\n" + fullermessage + "\n")

else:

# Report error message if Spatial Analyst license is unavailable
    arcpy.AddMessage ("Spatial Analyst license is unavailable")
```

```
# Import external modules
# Note that math modules isn't needed
import sys, os, string, arcpy, math, traceback

from arcpy import env
from arcpy.sa import *

# Check to see if Spatial Analyst license is available. If available, continue with script.
if arcpy.CheckExtension("spatial") == "Available":
   try:

# Activate ArcGIS Spatial Analyst license
      arcpy.CheckOutExtension("spatial")

# Read user inputs from dialog box

      Towns = arcpy.GetParameterAsText(0)
      Preserves = arcpy.GetParameterAsText(1)
      Roads = arcpy.GetParameterAsText(2)
      Hydrology = arcpy.GetParameterAsText(3)
      Trails = arcpy.GetParameterAsText(4)
      Railroads = arcpy.GetParameterAsText(5)
      OpenSpace = arcpy.GetParameterAsText(6)
      Elevation = arcpy.GetParameterAsText(7)
      OutputName = arcpy.GetParameterAsText(8)

# Reclassify Towns to identify only those of interest. Here, New Caanan is selected.
      Towns = arcpy.sa.Reclassify(Towns,"Value","90 1","NODATA")


#Criterion 1: Use Euclidean Distance to estimate the distance away from existing New Caanan
#Land Trust preserves. Reclassify output to create a score grid, with higher scores allocated to
#those areas further from existing preserves.

      Pres_BUFFER = arcpy.sa.EucDistance(Preserves)
      Pres_SCORE  = arcpy.sa.Reclassify(Pres_BUFFER, "Value", RemapRange([[0, 1200, 0], [1200, 2400,
20], [2400, 3200, 40], [3200, 4800, 60], [4800, 6834, 80]]), "DATA")


#Criterion 2: Reclassify to select only state highways and interstates. Use Euclidean Distance to
#estimate the distance away from roads. Reclassify output to create a score grid, with higher
#scores allocated to those areas further from roads.

      #Roads_RECLASS = arcpy.sa.Reclassify(Roads,"Value","3 1; 4 1","NODATA")
      Roads_BUFFER = arcpy.sa.EucDistance(Roads)
      Roads_SCORE = arcpy.sa.Reclassify(Roads_BUFFER, "Value", RemapRange([[0, 199, 0], [200, 299,
20], [300, 399, 40], [400, 450, 60], [450, 599, 80], [600, 130000, 100]]), "DATA")
```

```
#Criterion 3: Use Euclidean Distance to estimate the distance away from a water feature.
#Reclassify output to create a score grid, with higher scores allocated to regions closer to a water
#body.

    Hydro_BUFFER = arcpy.sa.EucDistance(Hydrology)
    Hydro_Score = arcpy.sa.Reclassify(Hydro_BUFFER, "Value", RemapRange([[0, 60, 100], [60, 120, 80],
[120, 180, 60], [180, 240, 40], [240, 300, 20], [300,132623, 0]]), "DATA")

#Criterion 4: Use Euclidean Distance to estimate the distance away from an existing state trail.
#Reclassify output to create a score grid, with higher scores allocated to regions closer to a state
#trail.

    Trails_BUFFER = arcpy.sa.EucDistance(Trails)
    Trails_SCORE = arcpy.sa.Reclassify(Trails_BUFFER, "Value", RemapRange([[0, 200, 100], [200, 400,
80], [400, 600, 60], [600,800, 40], [800, 1000, 20], [1000,99000, 0]]), "DATA")


#Criterion 5: Use Euclidean Distance to estimate the distance away from a railroad. Reclassify
#output to create a score grid, with higher scores allocated to regions closer to a railroad.

    Railroads_BUFFER = arcpy.sa.EucDistance(Railroads)
    Railroads_SCORE = arcpy.sa.Reclassify(Railroads_BUFFER, "Value", RemapRange([[0, 200, 100],
[200, 300, 80], [300, 400, 60], [400, 500, 40], [500, 600, 20], [600, 122000, 0]]), "DATA")


#Criterion 6: Use Euclidean Distance to estimate the distance away from existing municipal open space.
#Reclassify output to create a score grid, with higher scores allocated to those areas further from
existing preserves.

    OS_BUFFER = arcpy.sa.EucDistance(OpenSpace)
    OS_SCORE  = arcpy.sa.Reclassify(OS_BUFFER, "Value", RemapRange([[0, 200, 0], [200, 400, 20],
[400, 600, 40], [600,800, 60], [800, 1000, 80], [1000,130000, 100]]), "DATA")


# Create a layer of combined scores to estimate the best overall locations.
    Score = Pres_SCORE + Roads_SCORE + Hydro_Score + Trails_SCORE + Railroads_SCORE + OS_SCORE

#Select the areas with the highest scores.
    Score_RECLASS = arcpy.sa.Reclassify(Score, "Value", "420 1; 400 1; 380 1; 360 1; 340 1; 320 1; 300
1", "NODATA")
    Directions = arcpy.sa.FlowDirection(Elevation)
    Accumulations = arcpy.sa.FlowAccumulation(Directions,Score_RECLASS)
    Accumulations.save(OutputName)

# Deactivate ArcGIS Spatial Analyst license
    arcpy.CheckInExtension("spatial")
```

```
    except Exception as e:
        arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
        exceptionreport = sys.exc_info()[2]
        fullermessage = traceback.format_tb(exceptionreport)[0]
        arcpy.AddError("at this location: \n\n" + fullermessage + "\n")

else:

# Report error message if Spatial Analyst license is unavailable
    arcpy.AddMessage ("Spatial Analyst license is unavailable")
```