Assignment #8

GEOG 4057

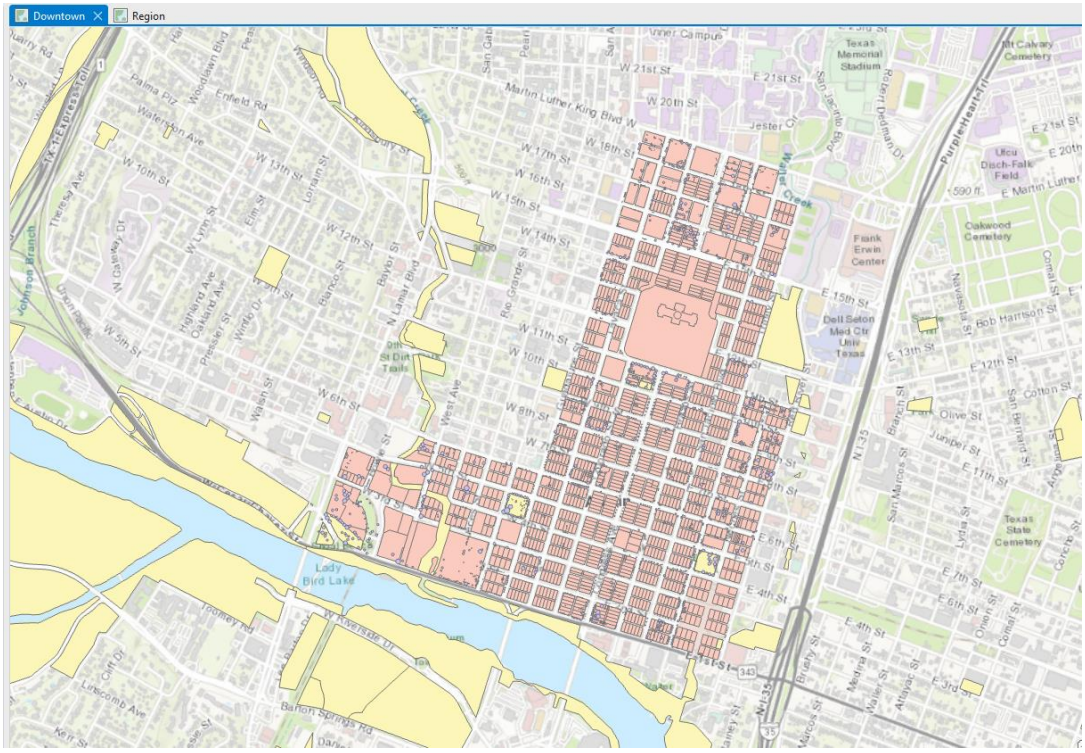
GIS programming

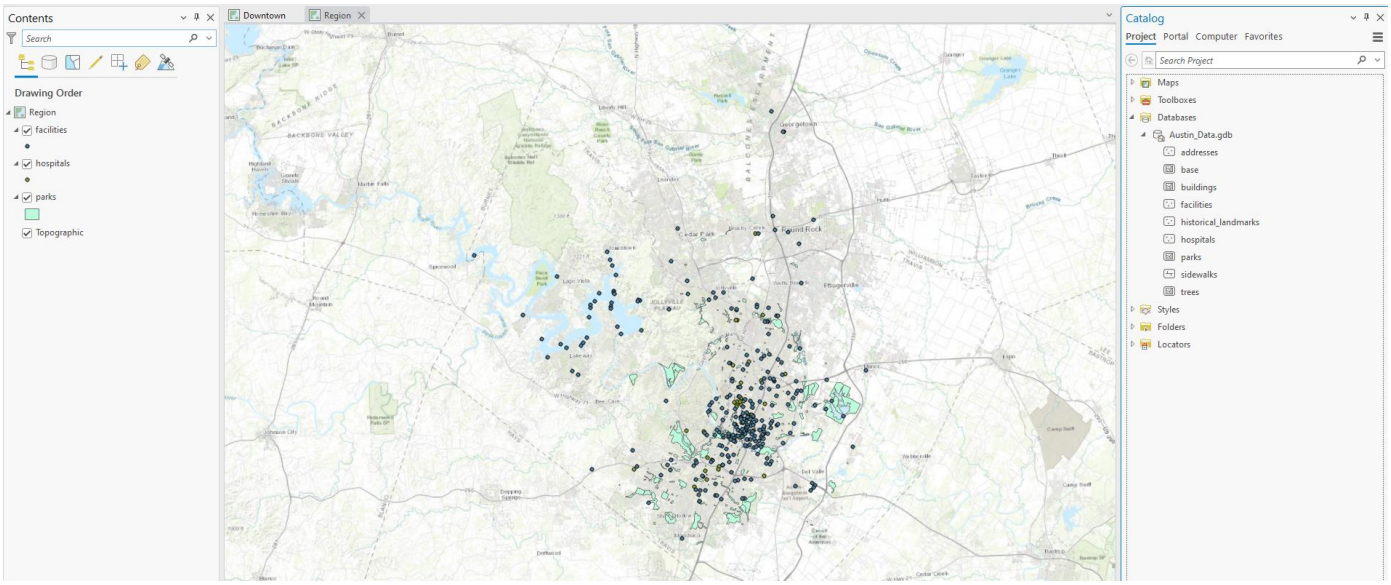
Prof. Lei Wang

Prepared by/

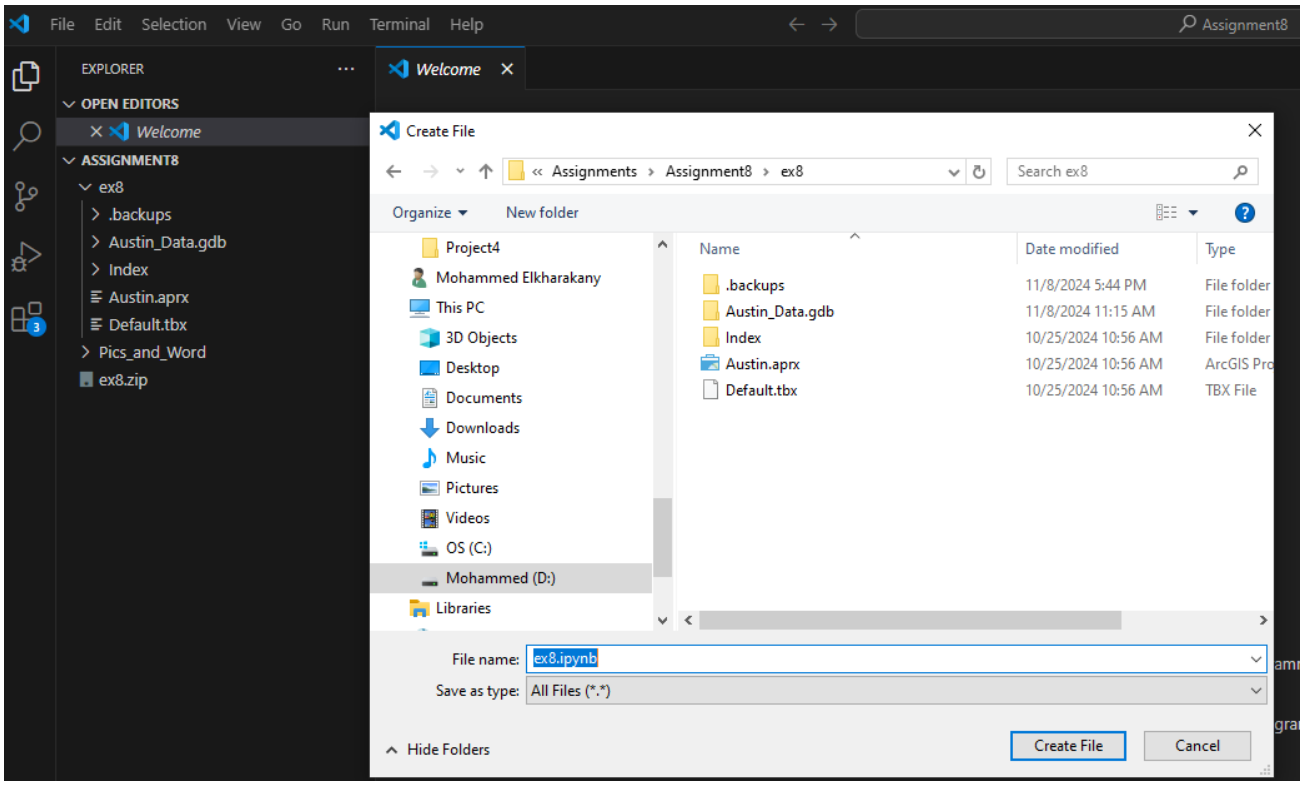
Mohammed Elkharakany

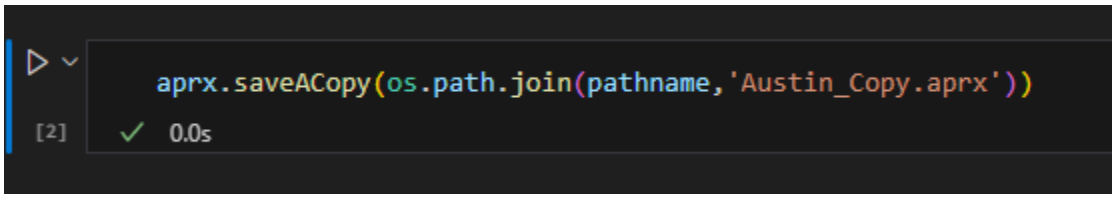
Data preparation

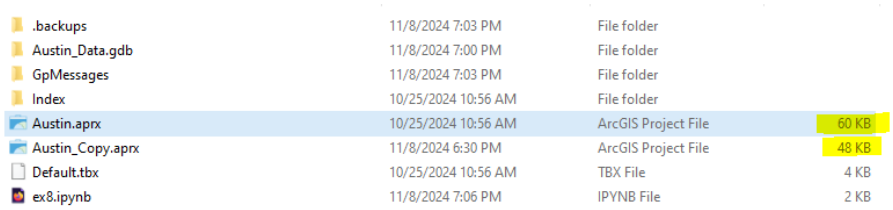




Open the project and print some information



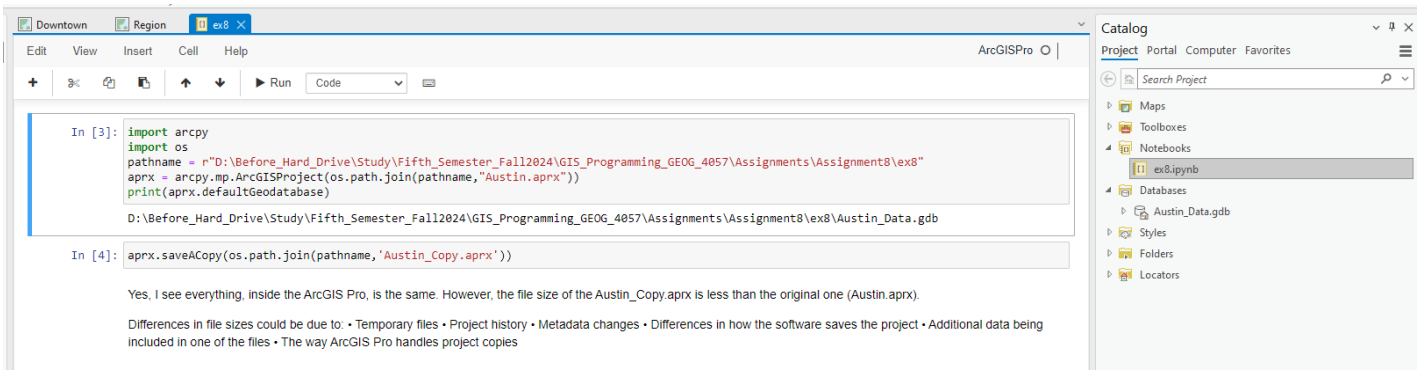




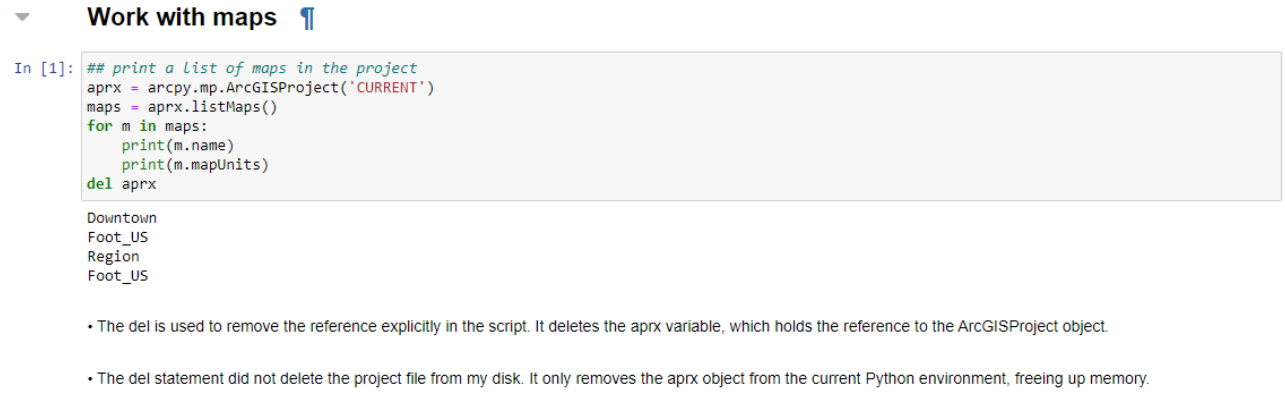
Yes, I see everything, inside the ArcGIS Pro, is the same. However, the file size of the Austin_Copy.aprx is less than the original one (Austin.aprx).

Differences in file sizes could be due to:

- Temporary files
- Project history
- Metadata changes
- Differences in how the software saves the project
- Additional data being included in one of the files
- The way ArcGIS Pro handles project copies



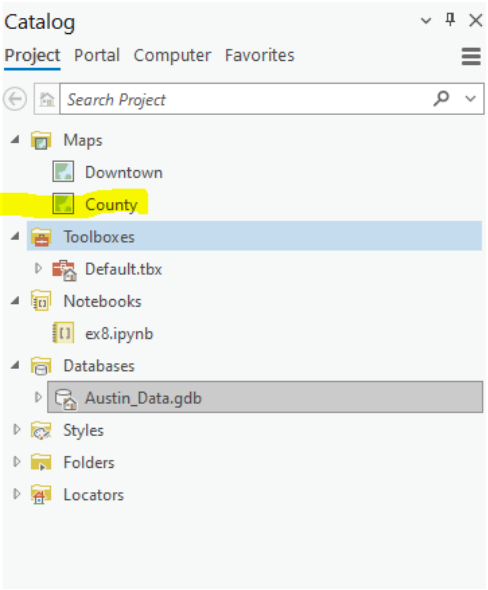
Work with maps



- The del is used to remove the reference explicitly in the script. It deletes the aprx variable, which holds the reference to the ArcGISProject object (i.e., looks like breaking the connection).
- The del statement did not delete the project file from my disk. It only removes the aprx object from the current Python environment, freeing up memory.

change the name of a map

```
In [2]: ## change the name of a map
        aprx = arcpy.mp.ArcGISProject('CURRENT')
        m = aprx.listMaps("Region")[0]
        m.name = "County"
        del aprx
```



list the layers in a map

```
In [3]: ## list the layers in a map
        aprx = arcpy.mp.ArcGISProject('CURRENT')
        maps = aprx.listMaps()
        for m in maps:
            print("Map: " + m.name)
            lyrs = m.listLayers()
            for lyr in lyrs:
                print(lyr.name)
        del aprx
```

```
Map: Downtown
trees
parks
base
Topographic
Map: County
facilities
hospitals
parks
Topographic
```

print if a layer is a basemap or a feature layer

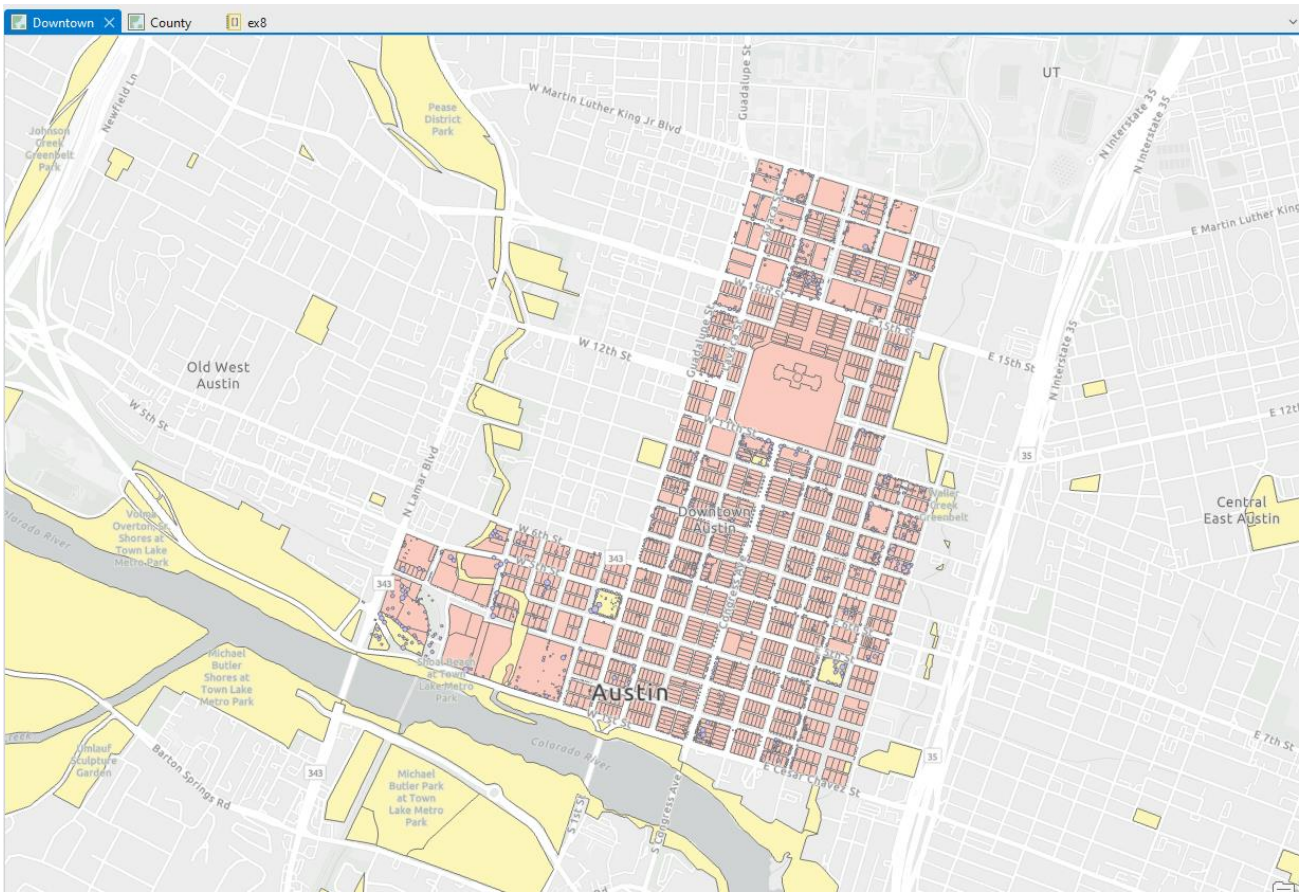
```
In [4]: ## print if a layer is a basemap or a feature layer
aprx = arcpy.mp.ArcGISProject('CURRENT')
m = aprx.listMaps("Downtown")[0]
lyrs = m.listLayers()
for lyr in lyrs:
    if lyr.isBasemapLayer:
        print(lyr.name + " is a basemap layer")
    if lyr.isFeatureLayer:
        print(lyr.name + " is a feature layer")
del aprx
```

```
trees is a feature layer
parks is a feature layer
base is a feature layer
Topographic is a basemap layer
```

Change the basemap

```
In [5]: ## change the basemap of Downtown
aprx = arcpy.mp.ArcGISProject('CURRENT')
m = aprx.listMaps("Downtown")[0]
m.addBasemap("Light Gray Canvas")
```

Yes, the basemap is changed as shown below.



Work with layers

Modify layer symbology

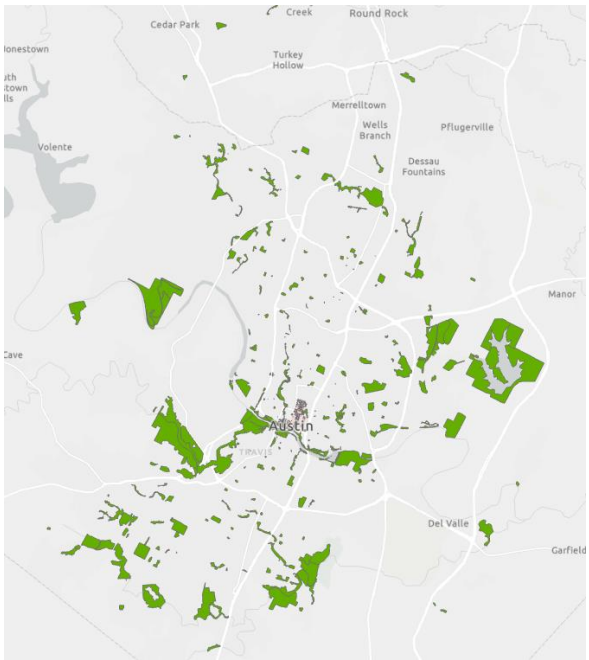
Work with layers

In [28]:
```
##Modify layer symbology
aprx = arcpy.mp.ArcGISProject("CURRENT")
m = aprx.listMaps("Downtown")[0]
lyr = m.listLayers("parks")[0]
sym = lyr.symbology
green = {"RGB": [100, 175, 0, 100]}
if lyr.isFeatureLayer and hasattr(sym, "renderer"):
    sym.renderer.symbol.color = green
    lyr.symbology = sym
```

The 'green' variable is a dictionary with a key "RGB" and a value that is a list of integers.

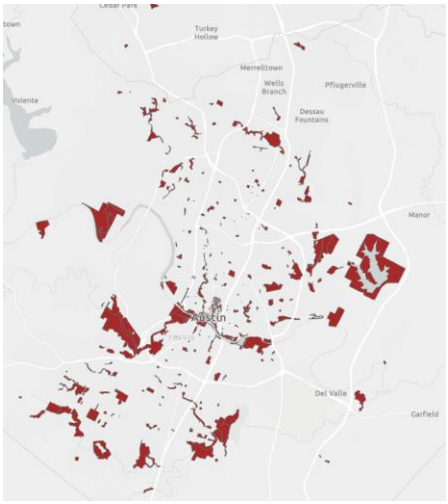
The list [100, 175, 0, 100] in the "RGB" dictionary represents the RGBA (Red, Green, Blue, Alpha) color model values:

- The first value (100) represents the red component.
- The second value (175) represents the green component.
- The third value (0) represents the blue component.
- The fourth value (100) represents the alpha (transparency or opacity) component.

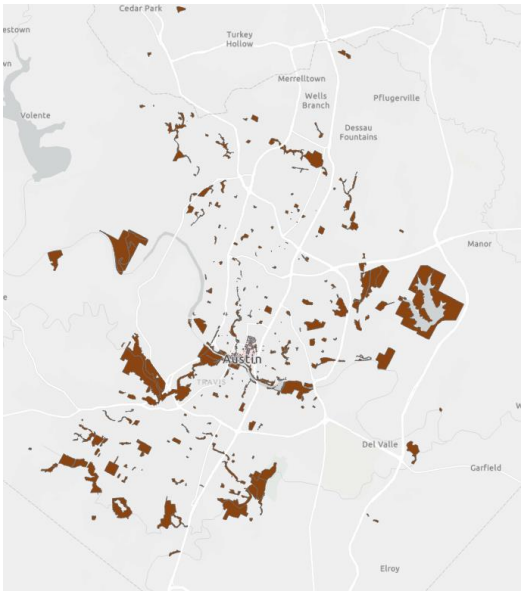


To change the color to brown, you would use appropriate RGBA values for brown, which can be represented as [165, 42, 42, 100] or [139, 69, 19, 100] in RGBA.

In [31]:
```
##Modify layer symbology to brown
aprx = arcpy.mp.ArcGISProject("CURRENT")
m = aprx.listMaps("Downtown")[0]
lyr = m.listLayers("parks")[0]
sym = lyr.symbology
green = {"RGB": [165, 42, 42, 100]}
if lyr.isFeatureLayer and hasattr(sym, "renderer"):
    sym.renderer.symbol.color = green
    lyr.symbology = sym
```



In [32]:
```
##Modify layer symbology to brown
aprx = arcpy.mp.ArcGISProject("CURRENT")
m = aprx.listMaps("Downtown")[0]
lyr = m.listLayers("parks")[0]
sym = lyr.symbology
green = {"RGB": [139, 69, 19, 100]}
if lyr.isFeatureLayer and hasattr(sym, "renderer"):
    sym.renderer.symbol.color = green
    lyr.symbology = sym
```



Work with a layout

Add a layout to the project

Work with a layout

In [9]:
```
##Add a layout to the project
aprx = arcpy.mp.ArcGISProject("CURRENT")
m = aprx.listMaps("Downtown")[0]

lyt = aprx.createLayout( 11,8.5, 'INCH', 'New Layout with Rectangles')

def MakeRec_LL(llx, lly, w, h):
    xyRecList = [[llx, lly], [llx, lly+h], [llx+w,lly+h], [llx+w,lly], [llx,lly]]
    array = arcpy.Array([arcpy.Point(*coords) for coords in xyRecList])
    rec = arcpy.Polygon(array)
    return rec

mf = lyt.createMapFrame(MakeRec_LL(0.5,0.5,10,7.5), m, "New Map Frame")
```

- The layout size is specified as 11 inches (width) by 8.5 inches (height).
- The map frame size is specified as 10 inches (width) by 7.5 inches (height).

➢ Creating layouts and map frames using Python (with ArcPy) has several advantages over manual creation in ArcGIS Pro:

- **Automation**: You can automate the creation of multiple layouts and map frames, which is useful if you need to generate maps for multiple areas or datasets.
- **Consistency**: Ensures that all layouts and map frames adhere to the same specifications
- **Time-saving**: Significantly reduces the time required for repetitive tasks.
- **Dynamic Adjustments**
- **Reproducibility**

Add layout elements

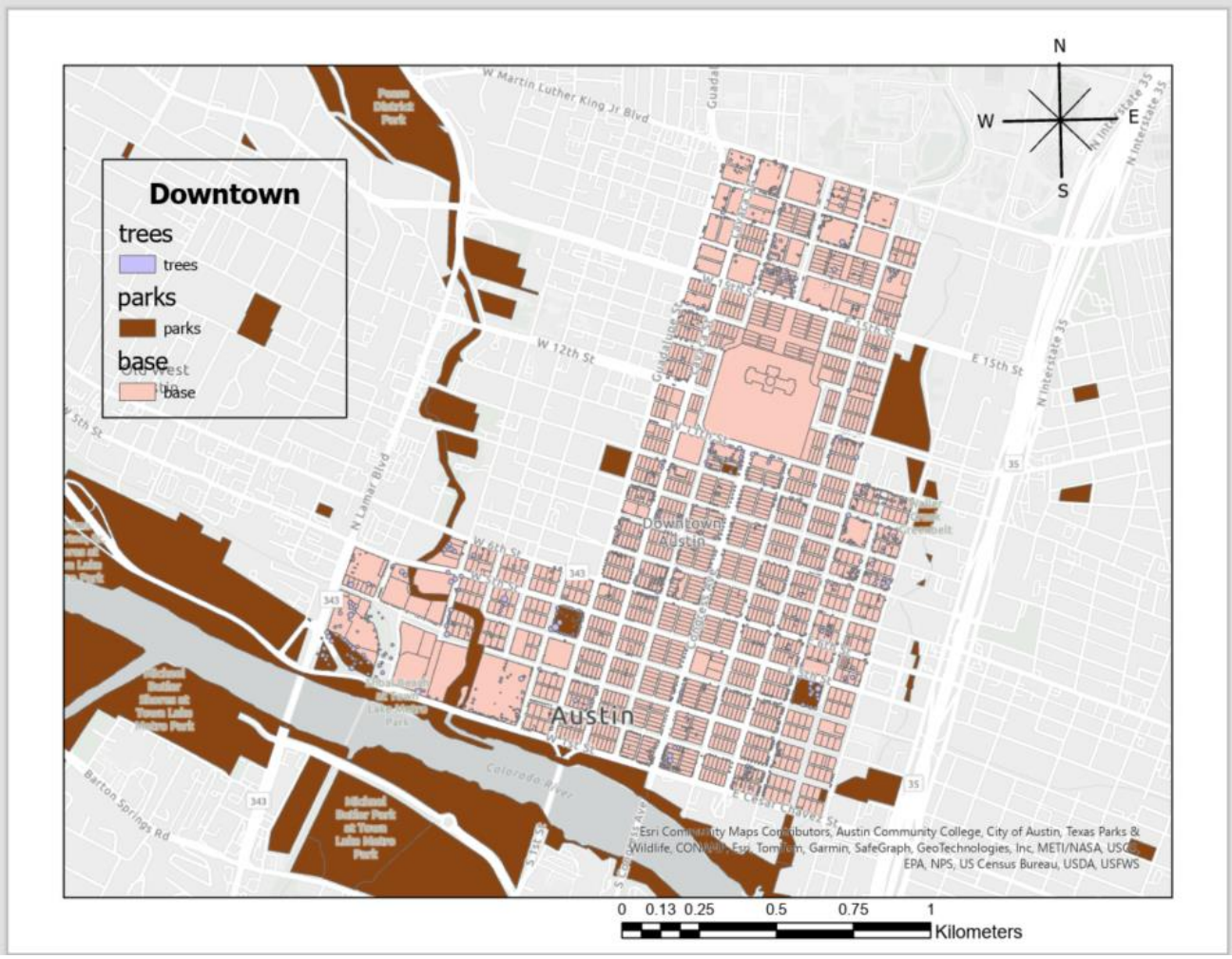
Add layout elements

In [12]:
```
##create a north arrow and a scale bar
#Create a north arrow
naStyle = aprx.listStyleItems('ArcGIS 2D', 'North_Arrow', 'Compass North 1')[0]
na = lyt.createMapSurroundElement(arcpy.Point(9.5,7.5), 'North_Arrow', mf,
                                  naStyle, "Compass North Arrow")
na.elementWidth = 0.5

#Create a scale bar
sbName = 'Double Alternating Scale Bar 1 Metric'
sbStyle = aprx.listStyleItems('ArcGIS 2D', 'Scale_bar', sbName)[0]
sbEnv = MakeRec_LL(5.5, 0.1, 4, 0.5)
sb = lyt.createMapSurroundElement(sbEnv, 'Scale_bar', mf, sbStyle, 'New Scale Bar')
```

create a legend

```
In [13]: ##create a legend
         legSi = aprx.listStyleItems('ArcGIS 2D', 'LEGEND', 'Legend 3' )[0]
         leg = lyt.createMapSurroundElement(arcpy.Point(1,7), 'LEGEND', mf, legSi, 'New Legend Element')
         leg.elementWidth = 3
         leg.elementHeight = 3
         leg.fittingStrategy = 'AdjustFontSize'
         leg.columnCount = 1
         leg.title = 'Downtown'
```



Export the layout to a pdf

```
In [21]: lyt.exportToPDF(os.path.join(pathname, 'downtown.pdf'))
```

Link to Github: https://github.com/Mohammed-Elkharakany/Assignment8.git