GIS Programming Lab 6 Instructions

TA: Christy Attaway

Pre-lab reading:

Map creation:

https://github.tamu.edu/TAMU-GEOG-676-GIS-Programming/Content/blob/master/modules/23.md

Arcpy messaging & progress:

https://github.tamu.edu/TAMU-GEOG-676-GIS-Programming/Content/blob/master/modules/24.md

Advanced tool parameters: (optional)

https://github.tamu.edu/TAMU-GEOG-676-GIS-Programming/Content/blob/master/modules/26.md

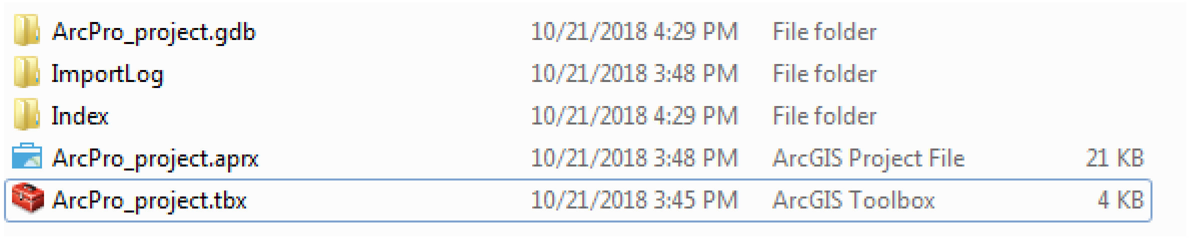
Tasks:

- Create a script that can generate either a unique value or graduated color map
- Turn said script into a toolbox that can be accessed from the Geoprocessing pane in ArcGIS Pro
- Utilize a progressor inside the tool to inform the user how far along the script is in generating the map

For this homework assignment, you will need to create a toolbox whose job is to create either a GraduatedColorsRenderer or UniqueValueRenderer based map. Once you have a working toolbox, you will need to add in a progressor who's label tracks what portion of the code is executing. This means the progressor should increment every so often and the label text should change with each progression.

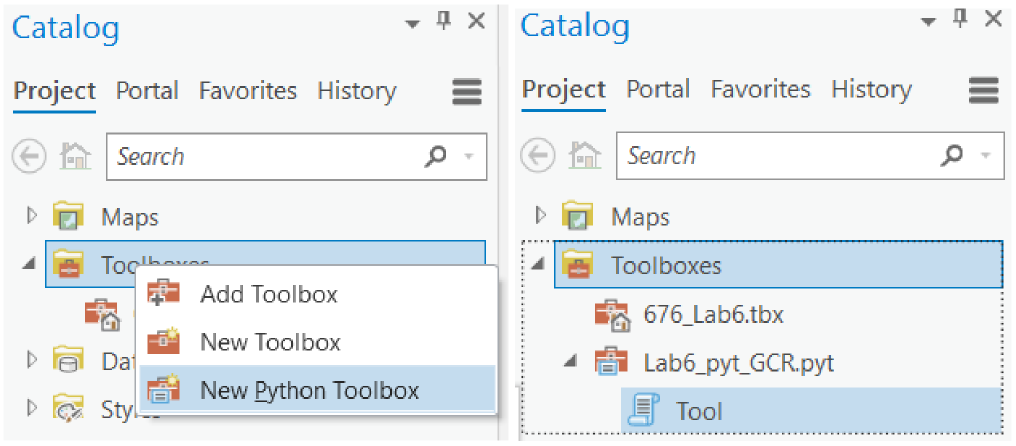
STEP 1: Generate a map

1. Create a new ArcGIS Pro project.
2. Add Campus.gdb and load one feature (GarageParking layer)
3. Save the project. You'll notice a toolbox (tbx) file within the project folder and in the Catalog tab.



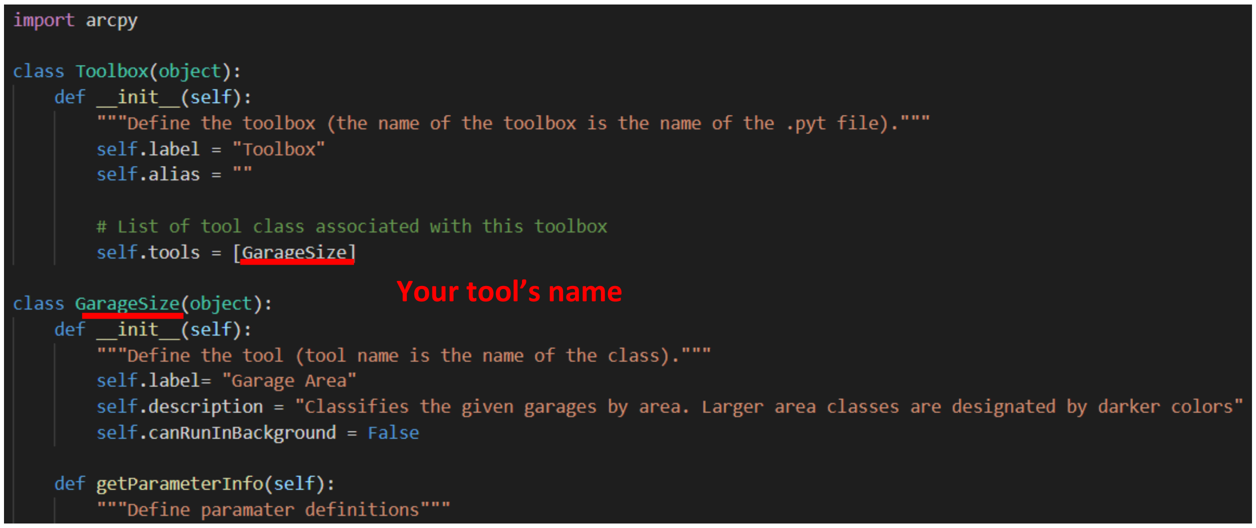
ArcPro_project.gdb	10/21/2018 4:29 PM	File folder	
ImportLog	10/21/2018 3:48 PM	File folder	
Index	10/21/2018 4:29 PM	File folder	
ArcPro_project.aprx	10/21/2018 3:48 PM	ArcGIS Project File	21 KB
ArcPro_project.tbx	10/21/2018 3:45 PM	ArcGIS Toolbox	4 KB

4. To make an editable toolbox, right click Toolboxes and select New Python Toolbox. Name it after the tool you choose (GraduatedColorsRenderer or UniqueValueRenderer) and you should see it added to your project with a generic tool inside the dropdown. We're going to change the toolbox to make this tool serve a unique purpose!



STEP 2: Edit your toolbox

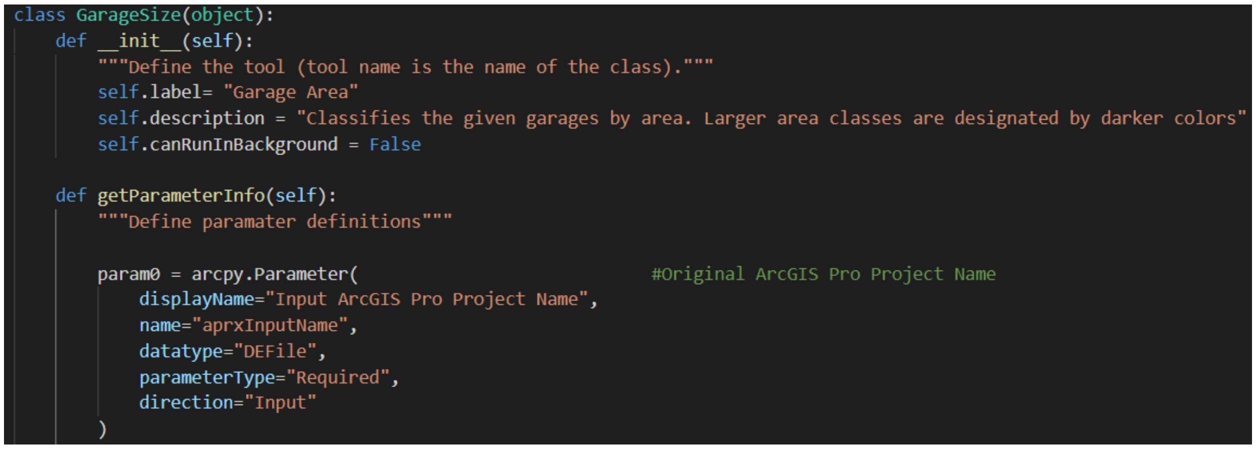
1. At this point I recommend making a copy of your toolbox and saving it as a .py so that you can edit it more easily (with colors). Don't forget to change it back to a .pyt when you're done.
2. Do not edit anything besides the tool name in the first part, otherwise the toolbox will not work.



```
import arcpy

class Toolbox(object):
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the .pyt file)."""
        self.label = "Toolbox"
        self.alias = ""

        # List of tool class associated with this toolbox
        self.tools = [GarageSize]

class GarageSize(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label= "Garage Area"
        self.description = "Classifies the given garages by area. Larger area classes are designated by darker colors"
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define paramater definitions"""
```

3. Now add your parameters. At minimum you'll need:
   a. Input project
   b. Layer to classify (which layer you want to use to generate a color map)
   c. Output location
   d. Output project name

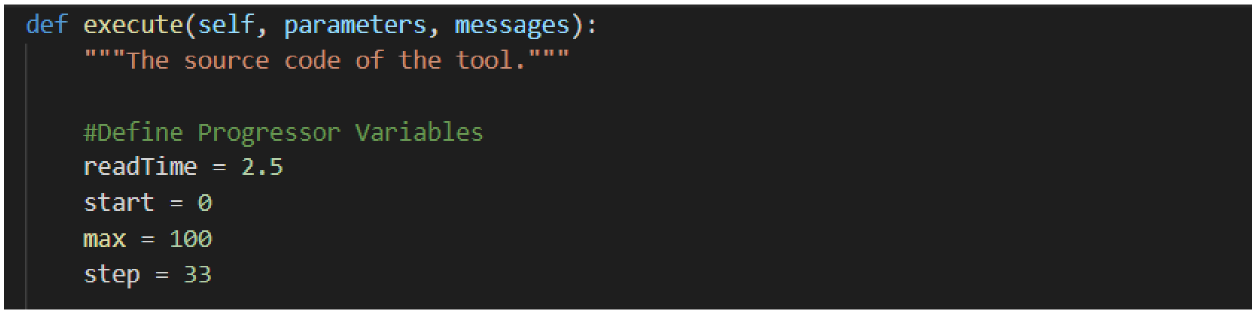
The first one should look something like this:



```
class GarageSize(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label= "Garage Area"
        self.description = "Classifies the given garages by area. Larger area classes are designated by darker colors"
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define paramater definitions"""

        param0 = arcpy.Parameter(                          #Original ArcGIS Pro Project Name
            displayName="Input ArcGIS Pro Project Name",
            name="aprxInputName",
            datatype="DEFile",
            parameterType="Required",
            direction="Input"
        )
```

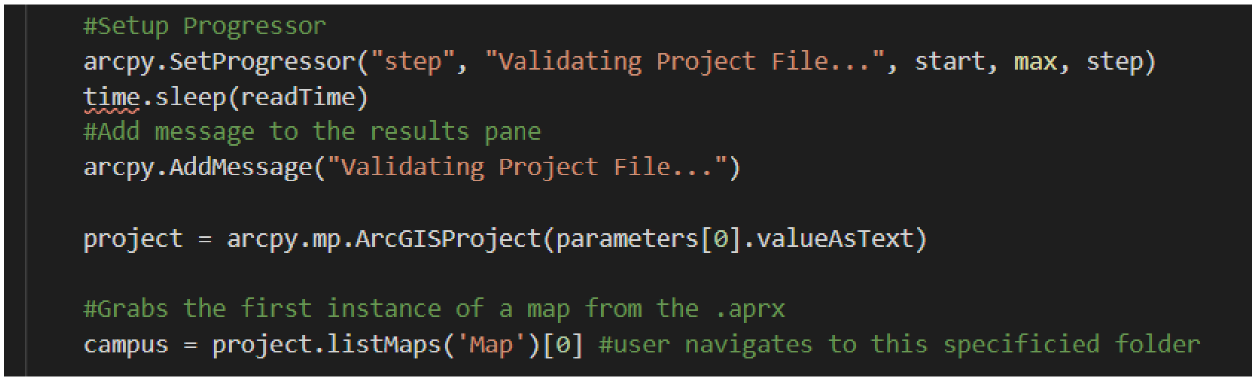
Check the link at the bottom of the instructions for your data types.

4. Next is the progressor within the execute section.
    a. First, you'll define progressor variables



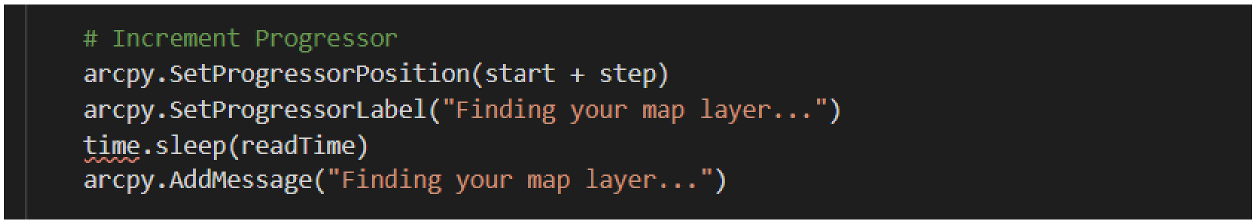
```python
def execute(self, parameters, messages):
    """The source code of the tool."""

    #Define Progressor Variables
    readTime = 2.5
    start = 0
    max = 100
    step = 33
```

    b. Set up your progressor with a message of your choice and read in your project file for your input project parameter.



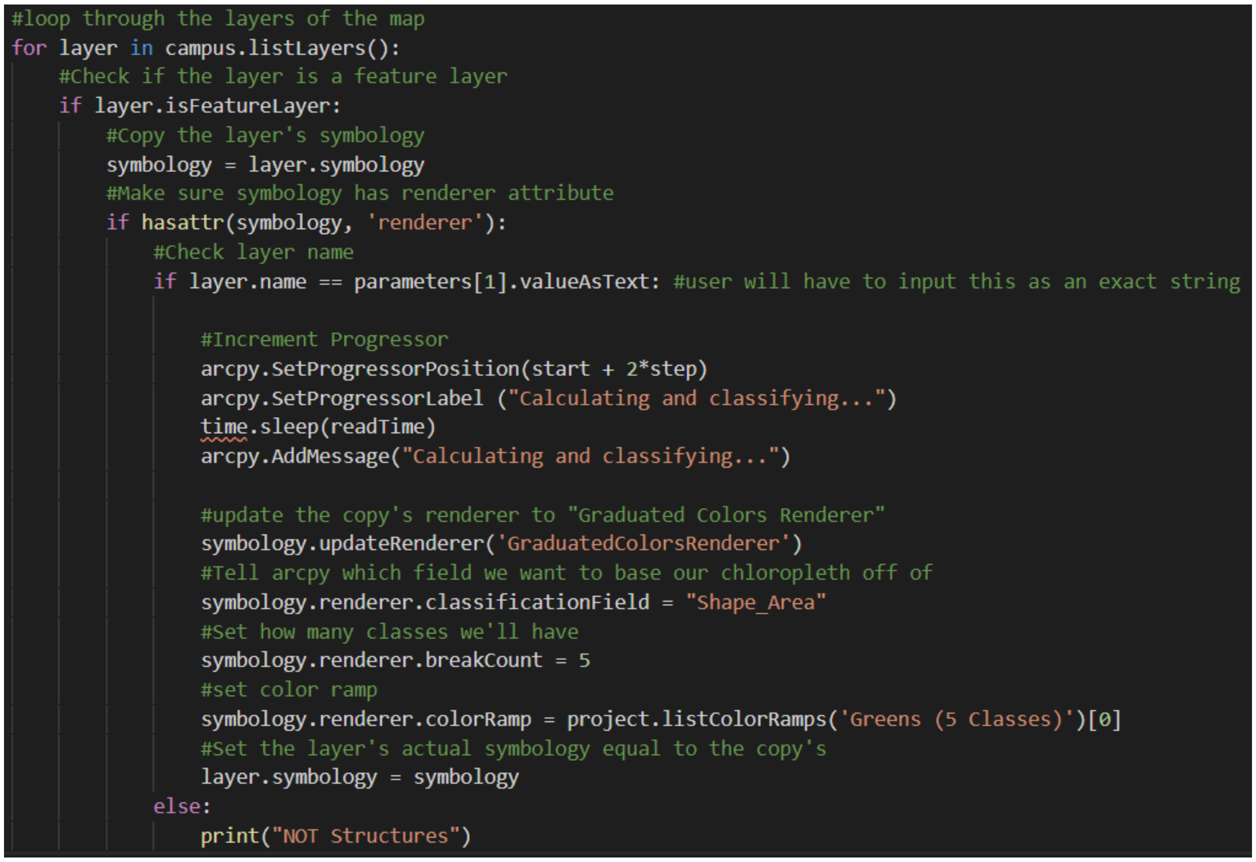
```python
    #Setup Progressor
    arcpy.SetProgressor("step", "Validating Project File...", start, max, step)
    time.sleep(readTime)
    #Add message to the results pane
    arcpy.AddMessage("Validating Project File...")

    project = arcpy.mp.ArcGISProject(parameters[0].valueAsText)

    #Grabs the first instance of a map from the .aprx
    campus = project.listMaps('Map')[0] #user navigates to this specificied folder
```

    c. Now that the progressor is initialized, set up new labels as it increments through the tool. You'll need to do this again after your for loop classification.



```python
    # Increment Progressor
    arcpy.SetProgressorPosition(start + step)
    arcpy.SetProgressorLabel("Finding your map layer...")
    time.sleep(readTime)
    arcpy.AddMessage("Finding your map layer...")
```

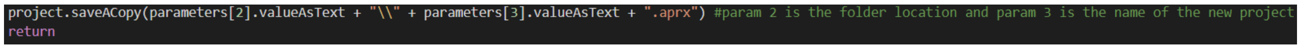
5. For loop
    a. To classify your layer, you'll need to set up the following filters:



```
#loop through the layers of the map
for layer in campus.listLayers():
    #Check if the layer is a feature layer
    if layer.isFeatureLayer:
        #Copy the layer's symbology
        symbology = layer.symbology
        #Make sure symbology has renderer attribute
        if hasattr(symbology, 'renderer'):
            #Check layer name
            if layer.name == parameters[1].valueAsText: #user will have to input this as an exact string

                #Increment Progressor
                arcpy.SetProgressorPosition(start + 2*step)
                arcpy.SetProgressorLabel ("Calculating and classifying...")
                time.sleep(readTime)
                arcpy.AddMessage("Calculating and classifying...")

                #update the copy's renderer to "Graduated Colors Renderer"
                symbology.updateRenderer('GraduatedColorsRenderer')
                #Tell arcpy which field we want to base our chloropleth off of
                symbology.renderer.classificationField = "Shape_Area"
                #Set how many classes we'll have
                symbology.renderer.breakCount = 5
                #set color ramp
                symbology.renderer.colorRamp = project.listColorRamps('Greens (5 Classes)')[0]
                #Set the layer's actual symbology equal to the copy's
                layer.symbology = symbology
            else:
                print("NOT Structures")
```

    b. Make sure to add your own labels and decide what field you'll use for classification.
    c. You can also adjust the break count and color to your personal preference!

6. Increment your progressor again to end the process. Save a copy of your classification using your last parameters.



```
project.saveACopy(parameters[2].valueAsText + "\\" + parameters[3].valueAsText + ".aprx") #param 2 is the folder location and param 3 is the name of the new project
return
```

STEP 4: Submission

- Screenshot of your .py code with Terminal window illustrating that it can be run without any errors.
- Screenshot of your Toolbox in ArcGIS Pro with no error messages popping up after running.
- Link to your Github page that contains the Python codes (Python script and toolbox)

Notes:

- If you're having difficulty with one (GraduatedColorsRenderer or UniqueValueRenderer) try the other one before giving up.
- Make sure you're checking if your toolbox is saved as a .py or a .pyt
- Keep your workspace clean. If you make copies of the toolbox to try different code, make sure you stay organized and remember where you save everything.
- Triple check that your data type is correctly listed in your parameters! ****
- https://pro.arcgis.com/en/pro-app/arcpy/geoprocessing_and_python/defining-parameter-data-types-in-a-python-toolbox.htm