

## Exercise 11

### Map scripting

#### Exercise data

Exercise data for this book can be downloaded from

[links.esri.com/PythonPro3rdEditionData](https://links.esri.com/PythonPro3rdEditionData). . This is a link to the ArcGIS Online group called

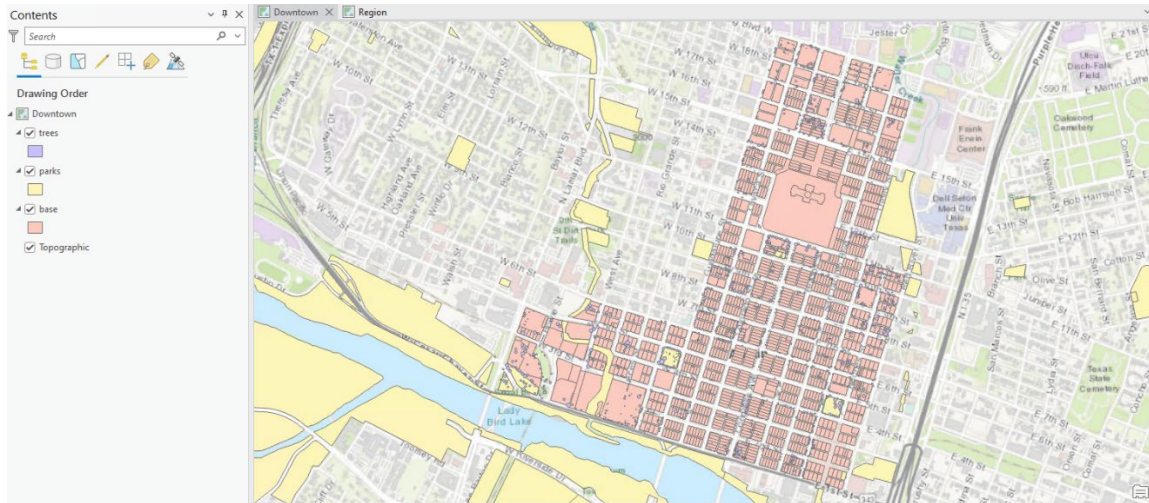
*Python Scripting for ArcGIS Pro – 2024 (Esri Press)*. The data for exercise 11 is posted as a zip file called `PythonScripting_Ex11_Data.zip`. Download this file and extract it to a folder of your choice. The instructions use a folder called `C:\PythonPro`, but you can use a different folder provided you update any paths.

#### Open and save a project

Map scripting can open a project, examine some of the properties of the project, and save a copy of the project.

- 1. Start ArcGIS Pro.**
- 2. Open the project `C:\PythonPro\Ex11\Austin.aprx`.**

Notice how the project includes two maps: Downtown and Region, each with a handful of feature layers.



### 3. Open the Python window and run the following code:

```
aprx = arcpy.mp.ArcGISProject("CURRENT")
```

Running this code creates an `ArcGISProject` object by referencing the project that is currently open. You can now access the properties and methods of this object.

### 4. Run the following code:

```
print(aprx.defaultGeodatabase)
```

The result is:

C:\PythonPro\Ex11\Austin\_Data.gdb.

Although using the `CURRENT` keyword is convenient when working with the Python windows, stand-alone scripts must reference the full path and name of the project.

### 5. Save changes and close ArcGIS Pro.

### 6. Start IDLE.

### 7. Create a new script file, and save your script as `projects.py` to the

C:\PythonPro\Ex11 folder.

## 8. Enter the following lines of code:

```
import arcpy

aprx = arcpy.mp.ArcGISProject("CURRENT")

print(aprx.defaultGeodatabase)
```

## 9. Save and run the script.

The code results in an error because CURRENT is not a meaningful reference outside ArcGIS Pro, even if the software application is up and running. Instead, the project must be referenced by its file name.

## 10. Correct line 2 of the code as follows:

```
aprx = arcpy.mp.ArcGISProject("Austin.aprx")
```

## 11. Save and run the script.

This time no error occurs, and the path of the default geodatabase prints. Because the projects.py script and the Austin\_Downtown.aprx file reside in the same directory, it is not necessary to specify the full path. However, when referencing a project in a different directory, the full path is necessary:

```
aprx_path = "C:/PythonPro/Ex11/Austin.aprx"

aprx = arcpy.mp.ArcGISProject(aprx_path)
```

Also note that setting a workspace using `arcpy.env.workspace` has no effect, so unless the script and .aprx file reside in the same directory, you must specify the full path.

**Note:** The code in the rest of the exercise assumes that that script and .aprx file reside in the same directory, and therefore only the file name is used, not the full path.

Map scripting often modifies the existing .aprx file, and for any changes to take effect, you must save the .aprx file. However, to review the changes, it is common to save the results to a new file using the `saveACopy()` method. In addition, it is common to add the `del` statement to remove the reference to a project to avoid unwanted locks on the .aprx file. You will practice this in the next step.

## 12. Modify the script as follows:

```
import arcpy

aprx = arcpy.mp.ArcGISProject("Austin.aprx")


aprx.saveACopy("Austin_Copy.aprx")

del aprx
```

## 13. Save and run the script.

Note that the line `del aprx` does not delete the project but only the reference to the project file in the script to avoid a lock on the file.

## 14. Use File Explorer to confirm that the new Austin\_Copy.aprx file has been created in the C:\PythonPro\Ex11 folder.

 Austin.aprx

 Austin\_Copy.aprx

In the remaining sections, you will use the `saveACopy()` method at the end of some of the scripts to save your changes.

## Work with maps

Map scripting can work with one or more maps in a project. First, you will determine a list of all the maps in a project.

1. In IDLE, create a new script file, and save your script as [list\\_maps.py](#) to the `C:\PythonPro\Ex11` folder.
2. Enter the following lines of code:

```
import arcpy

aprx = arcpy.mp.ArcGISProject("Austin.aprx")

maps = aprx.listMaps()

for m in maps:

    print(m.name)

    print(m.mapUnits)

del aprx
```

3. Save and run the script.

Running the script prints the names and map units of the two maps in the project to the interactive interpreter:

Downtown

Foot\_US

Region

Foot\_US

Some properties also can be modified.

**4. Modify the code as follows:**

```
import arcpy

aprx = arcpy.mp.ArcGISProject("Austin.aprx")

m = aprx.listMaps("Region")[0]

m.name = "County"

aprx.saveACopy("Austin_County.aprx")

del aprx
```

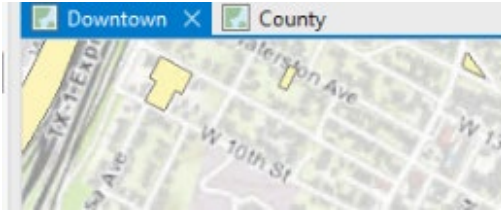
This script references one specific map by name, changes the name of the map, and then saves the project as a copy.

**5. Save and run the script.**

**6. Start ArcGIS Pro.**

**7. Open the project C:\PythonPro\Ex11\Austin\_County.aprx.**

**8. Confirm that the map previously called Region has been renamed to County.**



## 9. Close ArcGIS Pro.

### Work with map layers

Map scripting also can work with map layers. In the next example, you will create a list of all the layers in a map and modify the properties of a specific layer.

1. In IDLE, create a new script file, and save your script as [work\\_layers.py](#) to the **C:\PythonPro\Ex11** folder.
2. Enter the following lines of code:

```
import arcpy

aprx = arcpy.mp.ArcGISProject("Austin.aprx")

maps = aprx.listMaps()

for m in maps:

    print("Map: " + m.name)

    lyrs = m.listLayers()

    for lyr in lyrs:

        print(lyr.name)

del aprx
```

### 3. Save and run the script.

Running the script prints the names of the two maps in the project and the layers present in each map:

```
Map: Downtown
```

```
parks
```

```
trees
```

```
base
```

```
Topographic
```

```
Map: Region
```

```
facilities
```

```
hospitals
```

```
parks
```

```
Topographic
```

You can check the type of layer by using the `is*` properties of the `Layer` object.

### 4. Modify the code as follows:



```
import arcpy

aprx = arcpy.mp.ArcGISProject("Austin.aprx")

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
```

## 5. Save and run the script.

Running the script prints the layer type of all the layers in the specified map:

```
parks is a feature layer
```

```
trees is a feature layer
```

```
base is a feature layer
```

```
Topographic is a basemap layer
```

You can manipulate the layers in a map using methods of the `Map` object. The following code illustrates how to modify the basemap:

**6. Modify the code as follows:**

```
import arcpy

aprx = arcpy.mp.ArcGISProject("Austin.aprx")

m = aprx.listMaps("Downtown")[0]

m.addBasemap("Light Gray Canvas")

aprx.saveACopy("Austin_Canvas.aprx")

del aprx
```

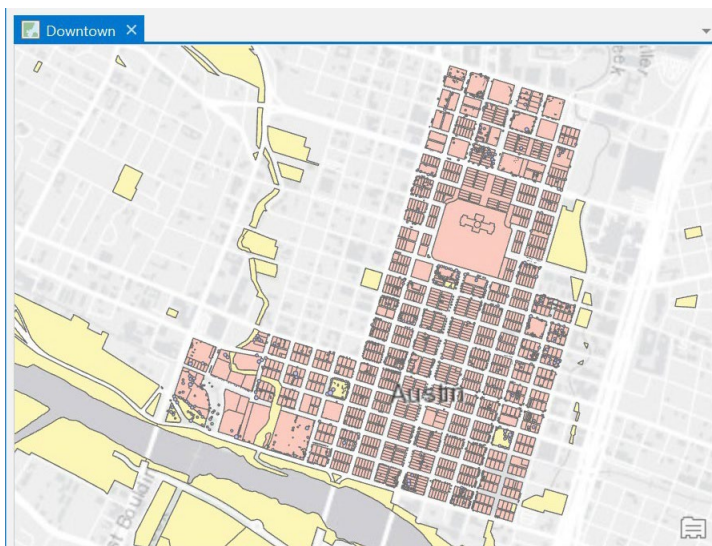
**7. Save and run the script.**

Running the script replaces the existing basemap.

**8. Start ArcGIS Pro.**

**9. Open the project C:\PythonPro\Ex11\Austin\_Canvas .aprx.**

**10. Confirm that the basemap in the Downtown map has been replaced.**



## 11. Close ArcGIS Pro.

### Modifying layer symbology

Symbology can be modified using properties of the `Layer` object. The following script creates a Symbology object, which is then modified, and this new symbology is applied to the layer of interest.

1. In IDLE, create a new script file, and save your script as `layer_symbology.py` to the `C:\PythonPro\Ex11` folder.
2. Enter the following lines of code:

```
import arcpy

aprx = arcpy.mp.ArcGISProject("Austin.aprx")

m = aprx.listMaps("Downtown")[0]

lyr = m.listLayers("parks")[0]

sym = lyr.symbology

red = {"RGB": [100, 175, 0, 100]}

if lyr.isFeatureLayer and hasattr(sym, "renderer"):

    sym.renderer.symbol.color = red

    lyr.symbology = sym

aprx.saveACopy("Austin_Symbology.aprx")

del aprx
```

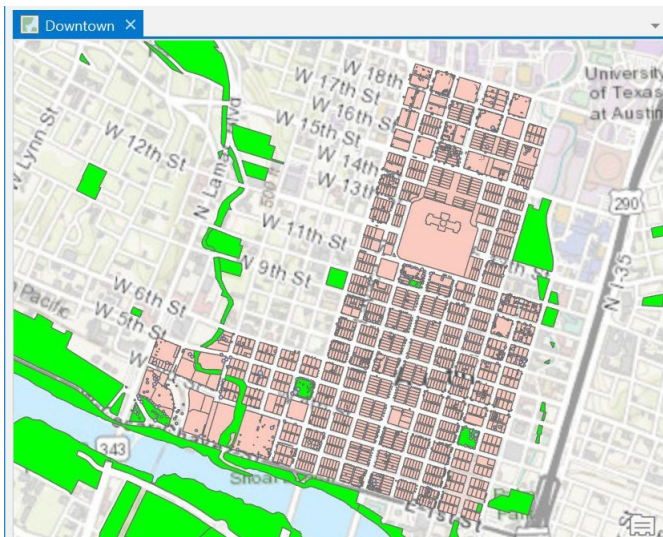
### **3. Save and run the script.**

The script performs two checks to ensure the layer of interest is of the right type and supports the renderer. Although not required, these checks make the script more robust because it prevents errors when modifying the symbology of layers that don't support this type of symbology. Running the script modifies the fill color of the polygon of the layer of interest.

### **4. Start ArcGIS Pro.**

### **5. Open the project C:\PythonPro\Ex11\Austin\_Symbology.aprx.**

### **6. Confirm that the color of the parks feature layer in the Downtown map has been modified to green.**



### **7. Close ArcGIS Pro.**

End of Exercise 11.