Exercise 10

Working with rasters

Exercise data

Exercise data for this book can be downloaded from 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 10 is posted as a zip file called PythonScripting_Ex10_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.

Preview the data

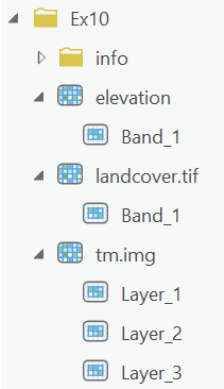
Before starting to work with the exercise data, you will preview the data in ArcGIS® Pro.

1. **Start ArcGIS Pro and click Start Without a Template to create a new blank project.**

2. **Make sure the Catalog pane is visible by clicking Catalog Pane on the View tab. Dock the Catalog pane to the right side of the ArcGIS Pro interface.**

3. **Create a new folder connection to the location of the exercise data by right-clicking Folders > Add Folder Connection, and navigating to the folder—e.g., C:\PythonPro\Ex10.**

4. **Examine the contents of this folder.**



The elevation raster is in Esri GRID format (no file extension), the landcover raster is in TIFF format (.tif), and the tm raster is in IMAGINE format (.img). The tm raster consists of three bands, whereas the other two rasters consist of a single band.

5. **Save the project as Exercise 10 to the C:\PythonPro\Ex10 folder. Then, close ArcGIS Pro.**

List the rasters

The `ListRasters()` function can be used to list all the rasters in a workspace.

1. **Start IDLE.**

2. **Create a new script file, and save your script as list_rasters.py to the C:\PythonPro\Ex10 folder.**

3. **Enter the following lines of code:**

```
import arcpy

arcpy.env.workspace = "C:/PythonPro/Ex10"

rasterlist = arcpy.ListRasters()

for raster in rasterlist:

    print(raster)
```

4. **Save and run the script.**

   Running the script prints a list of rasters in the current workspace, as follows:

```
elevation

landcover.tif

tm.img
```

   In this case, elevation is a raster in Esri GRID format and therefore has no file extension.

Describe the rasters

The `da.Describe()` function can describe raster properties.

1. **In IDLE, create a new script file, and save your script as describe_rasters.py to the C:\PythonPro\Ex10 folder.**

2. **Enter the following code:**

```
import arcpy

arcpy.env.workspace = "C:/PythonPro/Ex10"

raster = "tm.img"

desc = arcpy.da.Describe(raster)

print("Raster base name: " + desc["baseName"])

print("Raster data type: " + desc["dataType"])

print("Raster file extension: " + desc["extension"])
```

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

Running the script prints several raster properties, as follows:

```
Raster base name: tm

Raster data type: RasterDataset

Raster file extension: img
```

Printing here employs string concatenation, but the same output can also be achieved using the `format()` method or f-strings. For example, line 5 of the script can be written as follows:

```
print("Raster base name: {0}".format(desc["baseName"]))
```

More specific properties depend on whether the raster data element is a raster dataset, raster band, or raster catalog. The tm.img raster is a raster dataset, which makes it possible to access additional properties.

4. **Add the following lines to the script:**

```
print("Raster spatial reference: " +
desc["spatialReference"].name)

print("Raster format: " + desc["format"])

print("Raster compression: " + desc["compressionType"])

print("Raster number of bands: " + str(desc["bandCount"]))
```

Notice that a property that consists of a number and is combined using the plus sign (+) must be converted into a string for proper printing.

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

   Running the script prints additional properties that are unique to raster datasets:

```
Raster spatial reference: GCS_North_American_1983

Raster format: IMAGINE Image

Raster compression: RLE

Raster number of bands: 3
```

   In contrast to the tm.img raster, landcover.tif is a single-band raster.

6. **Modify line 3 of the script as follows:**

```
raster = "landcover.tif"
```

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

   One of the differences in the results is that the number of bands is one. Additional raster properties can be accessed for individual raster bands. For single-band rasters, this is implicit, and the raster band does not need to be specified.

8. **Modify the script as follows:**

```
import arcpy

arcpy.env.workspace = "C:/PythonPro/Ex10"
```

```
raster = "landcover.tif"

desc = arcpy.da.Describe(raster)

x = desc["meanCellHeight"]

y = desc["meanCellWidth"]

spatialref = desc["spatialReference"]

units = spatialref.linearUnitName

print("The raster resolution is {0} by {1} {2}.".format(x,
y, units))
```

9. **Save and run the script.**

Running the script prints the cell size in the units of the coordinate system of the raster, as follows:

```
The raster resolution is 30.0 by 30.0 Meter.
```

For multiband rasters, however, individual bands must be specified.

10. **Modify line 3 of the script as follows:**

```
raster = "tm.img"
```

11. **Save and run the script.**

The result is an error:

```
KeyError: 'meanCellHeight'
```

The `da.Describe()` function returns a dictionary, and `meanCellHeight` is not a key in this dictionary because the raster is a multiband raster. Determining the resolution of the tm.img raster requires examining the individual bands. This can be accomplished using the `children` property.

12. **Modify the script as follows:**

```
import arcpy

arcpy.env.workspace = "C:/PythonPro/Ex10"

raster = "tm.img"

desc = arcpy.da.Describe(raster)

for rband in desc["children"]:

    bandname = rband["baseName"]

    x = rband["meanCellHeight"]

    y = rband["meanCellWidth"]

    spatialref = desc["spatialReference"]

    units = spatialref.angularUnitName

    print("The resolution of {0} is {1:.7f} by {2:.7f}
{3}."

          .format(bandname, x, y, units))
```

13. **Save and run the script.**

Running the script prints the raster resolution in the units of the coordinate system. Because the tm.img raster is in a geographic coordinate system, the units are angular units, not linear units, as follows:

```
The resolution of Layer_1 is 0.0002778 by 0.0002778 Degree.

The resolution of Layer_2 is 0.0002778 by 0.0002778 Degree.

The resolution of Layer_3 is 0.0002778 by 0.0002778 Degree.
```

Determining whether the raster is in a geographic or projected coordinate system is not in the script but can be accomplished using the type property of the spatial reference—for example:

```
if spatialref.type == "Geographic"
```

Use raster objects in geoprocessing

When working with raster datasets in Python scripting, it is common to work with raster objects. Most raster geoprocessing tools return raster objects, which can be saved as rasters when necessary.

1. **In IDLE, create a new script file, and save your script as raster_objects.py to the C:\PythonPro\Ex10 folder.**

2. **Enter the following code:**

```
import arcpy

arcpy.env.workspace = "C:/PythonPro/Ex10"

outraster = arcpy.sa.Slope("elevation")
```

3. **Save and run the script**

   Next, you will determine the permanent state of the raster.

4. **Add the following lines to the script:**

```
desc = arcpy.da.Describe(outraster)

print(desc["datasetType"])

print(desc["permanent"])
```

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

   The result is

```
RasterDataset

True
```

The output from the Slope tool is a raster dataset but this is only temporary. When you exit the script, the temporary raster will be deleted. The `save()` method can be used to make the raster permanent.

6. **Replace lines 4–6 with the following line of code:**

```
outraster.save("slope")
```

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

8. **Start ArcGIS Pro and open Exercise 10.**

9. **In the Catalog pane, navigate to the C:\PythonPro\Ex10 folder, and confirm that the slope raster has been created.**

In working with rasters, it is common to keep most of the intermediate raster datasets in memory as raster objects, and save only the final raster dataset of interest.

10. **Close ArcGIS Pro.**

Use map algebra operators

The `arcpy.sa` and `arcpy.ia` modules contain several map algebra operators, which you'll use next. These operators make it easier to write map algebra expressions in Python.

1. **In IDLE, create a new script file, and save your script as raster_algebra.py to the C:\PythonPro\Ex10 folder.**

2. **Enter the following code:**

```
import arcpy

from arcpy.sa import *

arcpy.env.workspace = "C:/PythonPro/Ex10"

elevraster = arcpy.Raster("elevation")
```

This code creates a raster object by referencing a raster on disk. Using raster objects makes it easier to use raster datasets in code.

3. **Add the following lines of code and run it:**

```
outraster = elevraster * 3.281

outraster.save("elev_ft")
```

Running this code converts the elevation values from meters to feet and saves the raster object as a permanent raster. The map algebra operator (*) is used instead of the Times tool to make the code shorter. Map algebra operators include arithmetic, bitwise, Boolean, and relational operators.

4. **Replace lines 5–6 with the following lines of code:**

```
slope = AspectSlope(elevraster)

goodslope = slope < 20

goodelev = elevraster < 2500

goodfinal = goodslope & goodelev

goodfinal.save("final")
```

Running this code creates a new raster indicating slope less than 20 degrees and elevation less than 2,500 meters. The code example illustrates how map algebra operators can be used to carry out a series of geoprocessing operations. All the outputs are temporary raster objects, and only the result is saved.

5. **Start ArcGIS Pro. Navigate to the C:\PythonPro\Ex10 folder, and confirm that the final raster has been created.**

6. **Add the final raster to a new map to confirm the Boolean nature of the cell values.**

7. **Close ArcGIS Pro.**

Work with classes to define raster tool parameters

Many raster tools have parameters that have a varying number of arguments. The `arcpy.sa` and `arcpy.ia` modules have several classes to make it easier to work with these parameters. You will use the Reclassify tool with the `RemapRange` class as one of the parameters.

1. **In IDLE, create a new script file, and save your script as reclass.py to the C:\PythonPro\Ex10 folder.**

2. **Enter the following code:**

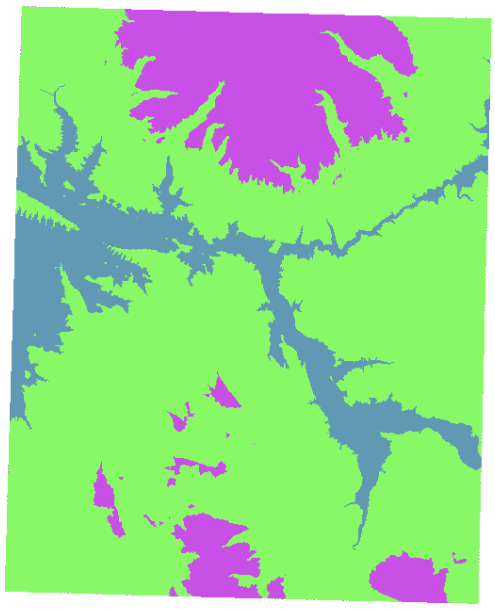
```
import arcpy
from arcpy.sa import *
arcpy.env.workspace = "C:/PythonPro/Ex10"
myremap = RemapRange([[1000,2000,1], [2000,3000,2],
[3000,4000,3]])
outreclass = Reclassify("elevation", "VALUE", myremap)
outreclass.save("elev_recl")
```

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

4. **In ArcGIS Pro, navigate to the C:\PythonPro\Ex10 folder, and confirm that the**

elev_recl raster has been created.

5. **Add the elev_recl raster to the current map to confirm the reclassified elevation.**



For discrete data, such as land cover, the `RemapValue` class is commonly used.

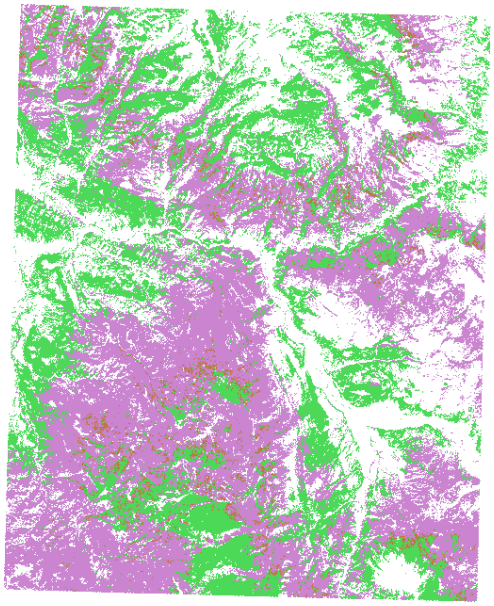
6. **Close ArcGIS Pro.**
7. **In the reclass.py script, replace lines 4–6 of the script that pertain to the reclassification of the elevation raster with the following:**

```
myremap = RemapValue([[41,1], [42,2], [43,3]])

outreclass = Reclassify("landcover.tif", "VALUE", myremap,
"NODATA")

outreclass.save("lc_recl")
```

8. **Save and run the script.**

9. **In ArcGIS Pro, navigate to the C:\PythonPro\Ex10 folder, and confirm that the lc_recl raster has been created.**

10. **Add the lc_recl raster to the current map to confirm the reclassified landcover.**



11. **Close ArcGIS Pro.**

    There is no need to save your work.

End of exercise 10.