Exercise 9
Working with rasters

List the rasters

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

1 **Start PythonWin. Create a new Python script and save as** listrasters.py **to the C:\EsriPress\Python\Data\Exercise09\Results folder.**

2 **Enter the following code:**

```
import arcpy
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise09"
rasterlist = arcpy.ListRasters()
for raster in rasterlist:
    print raster
```

3 **Save and run the script.**

Running the script prints a list of rasters to 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 `Describe` function can be used to describe raster properties.

1 **In PythonWin, create a new Python script and save as** describerasters.py **to the Results folder for exercise 9.**

2 **Enter the following code:**

```
import arcpy
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise09"
raster = "tm.img"
desc = arcpy.Describe(raster)
print "Raster base name is: " + desc.basename
print "Raster data type is: " + desc.dataType
print "Raster file extension is: " + desc.extension
```

3 **Save and run the script.**

Running the script prints a number of general raster properties, as follows:

```
Raster base name is: tm
Raster data type is: RasterDataset
Raster file extension is: img
```

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 print statements to the script:**

```
print "Raster spatial reference is: " + desc.spatialReference.name
print "Raster format is: " + desc.format
print "Raster compression type is: " + desc.compressionType
print "Raster number of bands is: " + str(desc.bandCount)
```

Notice that the band Count property which consists of a number and is combined using the plus sign (+), needs to 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 base name is: tm
Raster data type is: RasterDataset
Raster file extension is: img
Raster spatial reference is: GCS_North_American_1983
Raster format is: IMAGINE Image
Raster compression type is: RLE
Raster number of bands is: 3
```

6 **Modify the script as follows:**

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

7 **Save and run the script.** In contrast to the tm.img raster, landcover.tif is a single-band raster. 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
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise09"
raster = "landcover.tif"
desc = arcpy.Describe(raster)
x = desc.meanCellHeight
y = desc.meanCellWidth
spatialref = desc.spatialReference
units = spatialref.linearUnitName
print "The raster resolution is " + str(x) + " by " + str(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 need to be specified.

10 Modify the script as follows:

```
import arcpy
from arcpy import env
env.workspace = "C:/EsriPress/Python/Data/Exercise09"
rasterband = "tm.img/Layer_1"
desc = arcpy.Describe(rasterband)
x = desc.meanCellHeight
y = desc.meanCellWidth
spatialref = desc.spatialReference
units = spatialref.angularUnitName
print "The raster resolution of Band 1 is " + str(x) + " by " + str(y) ➔
➔ + " " + units + "."
```

11 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:

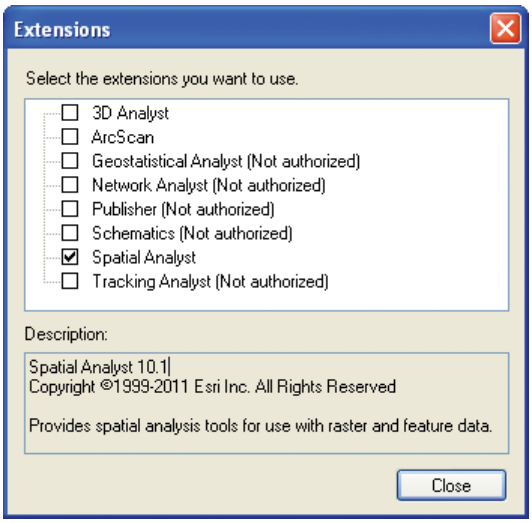
```
The raster resolution of Band 1 is 0.000277778 by 0.000277778 Degree.
```

Use raster objects in geoprocessing

To use the Spatial Analyst geoprocessing tools, you need an ArcGIS Spatial Analyst license. When working in the Python window, you can set this from within ArcMap.

1 Start ArcMap. On the menu bar, click Customize > Extensions.

2 On the Extensions dialog box, activate the ArcGIS Spatial Analyst extension by selecting that check box. ➔



3 Click Close. In 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.

4 Open the Python window. This will allow you to see immediate results from running each line of code.

5 Run the following code:

```
>>> from arcpy import env
>>> env.workspace = "C:/EsriPress/Python/Data/Exercise09"
>>> outraster = arcpy.sa.Slope("elevation")
```

The new raster outraster is added to the ArcMap table of contents. Next, you will determine the permanent state of the raster.

6 Run the following code:

```
>>> desc = arcpy.Describe(outraster)
>>> print desc.permanent
```

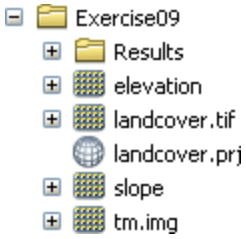
The result is `False`.

The new raster is only temporary. You can also determine the raster's status by examining the layer properties in ArcMap. When you exit ArcMap without saving the map document, the temporary raster will be deleted. The `save` method can be used to make the raster permanent.

7 Run the following code:

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

8 Open the Catalog window in ArcMap. Navigate to the C:\EsriPress\Python\Data\Exercise09 folder and confirm that the slope raster has been saved.



In working with rasters, it is common to import all the functions of the `arcpy.sa` module, which you will do next. This module shortens the code to call geoprocessing tools.

9 Run the following code:

```
>>> from arcpy.sa import *
>>> outraster2 = Aspect("elevation")
>>> outraster2.save("aspect")
```

>>> TIP

To see the result in the Catalog window, right-click a folder (in this case, the Exercise09 folder) and click Refresh. This makes any newly added data files visible.

Use map algebra operators

The `arcpy.sa` module contains a number of map algebra operators, which you'll use next. These operators make it easier to write map algebra expressions in Python.

1 **Run the following code:**

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

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

2 **Run the following code:**

```
>>> outraster3 = elevraster * 3.281
>>> outraster3.save("elev_ft")
```

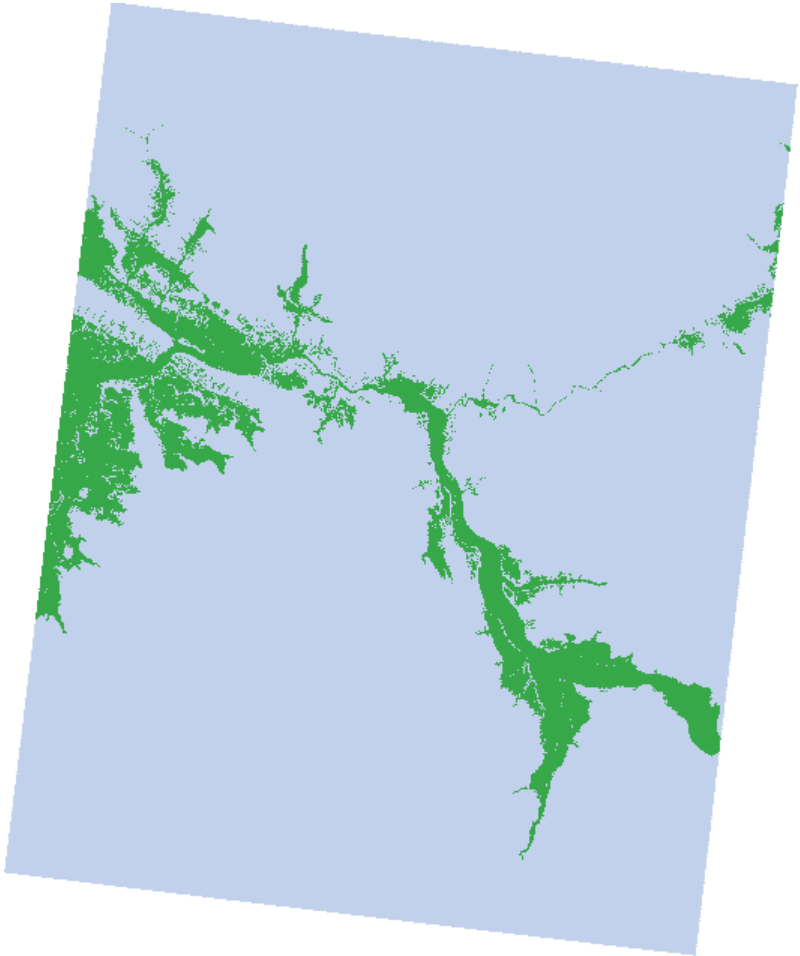
Running this code converts the elevation values from meters to feet and saves the raster object as a permanent raster. The map algebra operator (`*`) is used instead of the Times tool to make the code shorter. Map algebra operators include arithmetic, bitwise, Boolean, and relational operators.

3 **Run the following code:**

```
>>> slope = Slope(elevraster)
>>> goodslope = slope < 20
>>> goodelev = elevraster < 2000
>>> goodfinal = goodslope & goodelev
>>> goodfinal.save("final")
```

Running this code creates a new raster indicating slope less than 20 degrees and elevation less than 2,000 meters.

The result of the code, as shown in the figure, 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 final result is saved.



Work with classes to define raster tool parameters

Many raster tools have parameters that have a varying number of arguments. The `arcpy.sa` module has a number of classes to make it easier to work with these parameters.

You will be working with a stand-alone script, and the first task is to make sure a license for the Spatial Analyst extension is available.

1 Start PythonWin. Create a new Python script and save as slope.py **to the Results folder for exercise 9.**

2 Enter the following code:

```
import arcpy
from arcpy import env
from arcpy.sa import *
env.workspace = "C:/EsriPress/Python/Data/Exercise09"
if arcpy.CheckExtension("spatial") == "Available":
    arcpy.CheckOutExtension("spatial")
    outraster = arcpy.sa.Slope("elevation", "PERCENT_RISE")
    outraster.save("slope_per")
    arcpy.CheckInExtension("spatial")
else:
    print "Spatial Analyst license is not available."
```

3 Save and run the script.

In the Catalog window in ArcMap, confirm that the **slope_per** raster has been created. You now have a basic stand-alone script for working with tools from the `arcpy.sa` module. Next, you will use the Reclassify tool with the `RemapRange` class as one of the parameters.

4 Save your slope.py script as reclass.py**.**

5 In the reclass.py script, replace the two lines of code that pertained to creating and saving the slope raster with the following, making sure to maintain the indentation of these lines:

```
myremap = RemapRange([[1000,2000,1], [2000,3000,2], [3000,4000,3]])
outreclass = Reclassify("elevation", "VALUE", myremap)
outreclass.save("elev_recl")
```

6 Save and run the script.

7 In the Catalog window in ArcMap, confirm that the elev_recl **raster has been created.**

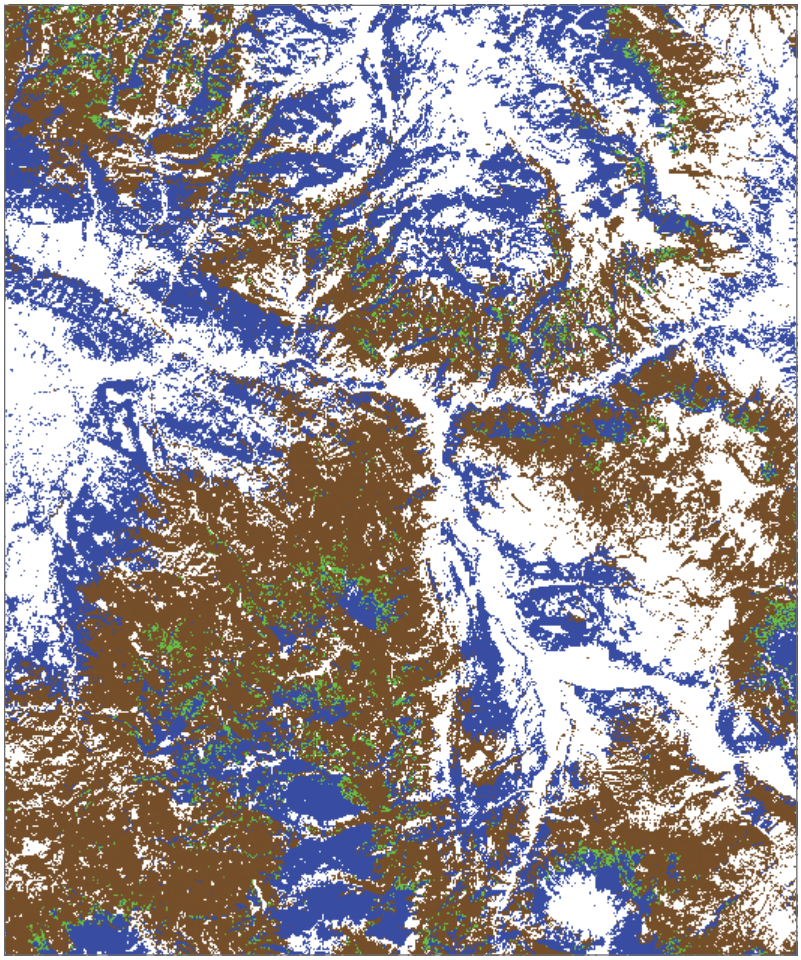
For discrete data, such as land cover, the `RemapValue` class is commonly used.

8 In the reclass.py script, replace the three lines of code that pertained 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")
```

9 Save and run the script.

10 In the Catalog window in ArcMap, confirm that the lc_recl **raster has been created.**



Other commonly used classes in raster-based analysis are the neighborhood classes, which you'll work with next.

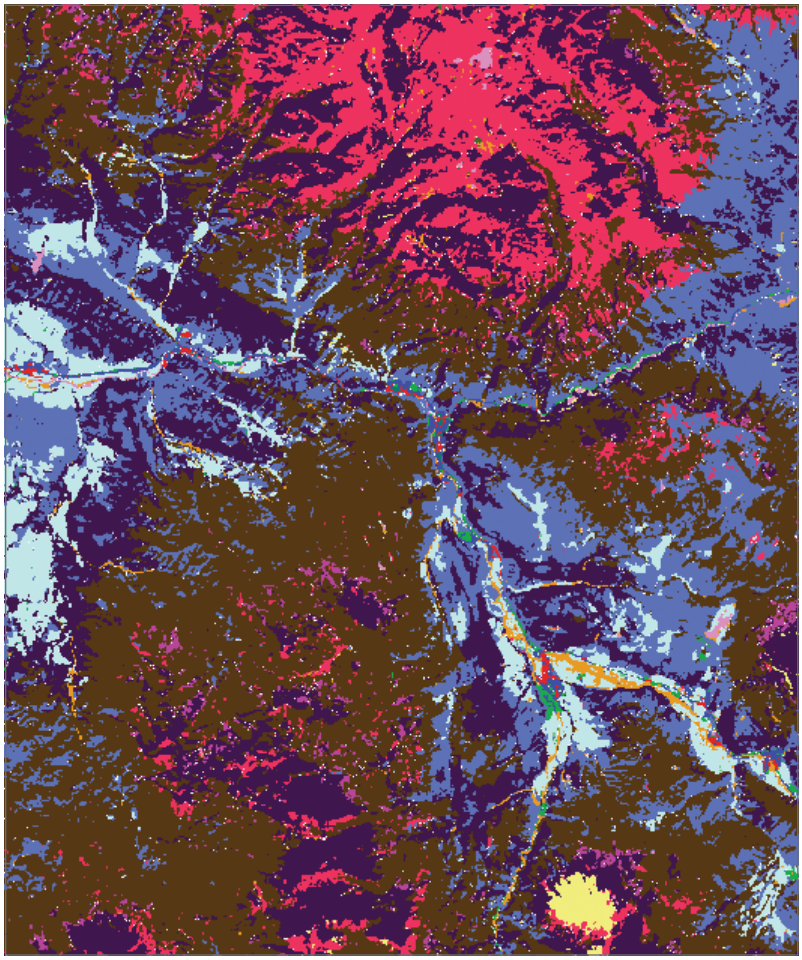
11 Save your reclass.py script as neighborhood.py**.**

12 In the neighborhood.py script, replace the three lines of code that pertained to the reclassification of the land cover raster with the following:

```
mynbr = NbrCircle(3, "CELL")
outraster = FocalStatistics("landcover.tif", mynbr, "MAJORITY")
outraster.save("lc_nbr")
```

13 Save and run the script.

14 In the Catalog window in ArcMap, confirm that the lc_nbr raster has been created.



Challenge exercises

Challenge 1

Create a script that determines what areas meet the following conditions:

- Moderate slope—between 5 and 20 degrees
- Southerly aspect—between 150 and 270 degrees
- Forested—land cover types of 41, 42, or 43

Be sure to use the map algebra expressions of the `arcpy.sa` module.

Challenge 2

Write a script that copies all the rasters in a workspace to a new file geodatabase. You can use the rasters in the Exercise09 folder as an example.