Another new technique is handling the output of printable maps. These maps were exported to PDF documents, but other forms of presentation are available. It is important to make sure that the output file names are unique, which allows the user to create several sets of maps in one session and to know that the files are not being overwritten.

Study questions

1. What other data integrity techniques might be used to ensure that the tax ID number entered by the user is valid?
2. Why would zooming to selected features be better in some circumstances than panning to selected features?
3. Check ArcGIS for Desktop Help to determine the other types of map output formats.

Tutorial 3-3 Creating a map series

The techniques used for manipulating the map elements can be combined with geoprocessing tasks to create a series of custom maps. Feature lists and selections can be used to control the extent of the map.

Learning objectives

- Combining feature selection techniques and map element manipulations
- Cloning map elements
- Creating a map book

Preparation

Research the following topics in ArcGIS for Desktop Help:

- "GraphicElement (arcpy.mapping)": Clone
- "DataFrame (arcpy.mapping)": zoomToSelectedFeatures()
- "ExportToPDF (arcpy.mapping)"
- "PDFDocument (arcpy.mapping)"

Introduction

In tutorials 3-1 and 3-2, you learned how to manipulate the elements in a map document and how to control the properties of the layers in the table of contents. In this tutorial, you will combine selections and geoprocessing tasks with these other techniques to create a series of maps.

Making selections in a layer and iterating through the selected features is a common task. As each feature is accessed, the extent of the feature, or for that matter a set of features, can be used to pan or zoom the map. The panToExtent() method is used in tutorial 3-2 because the map was set to a fixed scale, but by using the zoomToSelectedFeatures() method, the scale of the map is changed to accommodate the selected features. One caveat is that the area covered by the selected set of features may vary greatly, so the map document must be able to accommodate many scales.

Although, in general, you are only able to work with elements already in the map document, and you can neither add nor delete elements, this limitation is not completely true because arcpy.mapping has the ability to clone graphic elements in your map to create copies that you can manipulate. Another common practice for using graphic elements is to include extra elements in the map document that are moved off the virtual page. If you need more elements than were originally used in the map document, make a list of all the elements in the map document, on or off the virtual page, and find one of the extras. Then move the element's location coordinates onto the virtual page. Elements not on the virtual page do not appear in prints or exported files.

Scenario

The Oleander Public Works Department must submit a water quality report to the State of Texas each year to maintain its ranking of "Superior Water Supply." Part of the process for developing the report is to take and test water samples at locations throughout the city. When the report is submitted, the department wants to include a series of small exhibit maps showing the location of each sampling station and a list of the property descriptions within 30 feet. It is supplying a point feature class with the sampling station locations, and it wants one map for each location.

Use the template map document provided, and iterate through each sampling station. Then select the properties within the specified distance, and generate the list of property descriptions. Make a table at the bottom of the map using graphic lines and text.

Each map should be exported to a PDF document, and when they are all completed, the maps can be combined to form a single map book.

Data

A map document is provided as a template, which includes the water utility system, the feature class containing the sampling stations, and a feature class containing the property information. The elements to use include the following:

- Water Utility—the data frame for the main map display.
- Title—the map title that you will change to reflect the station number.
- PropDesc—text that contains the property description of the selected properties. Clone this for each additional property.
- LeftVert and RightVert—the lines bordering the property description. Clone these for each additional property.
- TopHoriz and BotHoriz—the lines at the top and bottom of the property description. Clone the bottom line for each additional property.

In the sampling station feature class, the field named Status shows which stations are operational. Only these stations should be mapped. There is also a description that will appear on the map.

The layer Parcels has two fields to supply the legal description of the property: Prop_Des_1 and Prop_Des_2. Combine these two fields into a single text element for each parcel, and display this text on the exhibit map.

SCRIPTING TECHNIQUES

The new technique in this tutorial includes the functions to clone text and graphics. At least one of the graphic elements you wish to clone must be present in the map document. Then the item can be cloned using the clone method, and its location and size can be adjusted using the element's properties. It is a suggested practice to add a suffix value to the cloned items. This value can be any string you like, but *_clone* is customary. Each new feature gets this suffix along with a count value. For example, _clone_1, _clone_2, and _clone_3.

After you are finished with the map exhibit, use the delete method to delete the cloned items. These items will be easy to identify because of the suffix, so a ListLayoutElements results object with a wildcard can easily find them all. Deleting all the cloned elements resets the map for the next iteration.

Another technique is to establish a set of variables for each layout element. In the previous tutorials, a list was created and searched each time an element needed to be found and used, which can be resource intensive for a script that creates a large number of maps. Instead, go through the list of elements once, and establish variables for each element. Finding and modifying elements will be easier and faster after the variables are established.

The last new technique shown here is working with PDF documents. Besides exporting the map to a PDF document, you can use other methods with PDF documents. Rather than send a large number of PDF documents to another user, combine them into a single PDF document for added convenience. It is also a good way to create a single map book that can be easily shared and printed.

To accomplish this task, export the individual map pages as normal. Then create a new, empty PDF document to hold the final map book. Using a looping statement, append all the individual pages to the map book file, creating the desired single output file. As a matter of cleanup, delete the individual files so that only the map book remains.

Create a map series

1. **Start ArcMap, and open the map document Tutorial 3-3.**

Take a moment to examine the names of the graphic elements as described in the data section at the start of this tutorial. You can use this map as a reference, but this script is a stand-alone script and has no user parameters. When it runs, the script creates a PDF document of each map and, at the end, combines all the maps into a single PDF document.

2. **Write the pseudo code for this script. Yours will be more detailed, but follow this basic outline:**

- **Iterate through the features in the SamplingStations layer, making sure to use only the active station.**
- **Use the current sampling station to select all the property within 30 feet.**
- **Zoom the map document to show the currently selected features.**
- **Change the map title to reflect the sampling station.**
- **Build a chart of the property descriptions, and separate the values with lines.**
- **Create an exhibit map for each sampling station, and when completed, merge all the maps into a single PDF document.**

When you have your pseudo code ready, continue with the tutorial.

3. **Start your IDE, and create a new script named** SampleStationsMapBook.py **in your MyExercises folder.**

4. **Add the title information and author, and import the ArcPy module, as shown:**

```
#-------------------------------------------------------------------------------
# Name:        Create Sampling Stations Exhibit Map
#
# Purpose:     Individual exhibit maps will be created for each sampling station
#               in Oleander. These will show the location and a list of the property
#               within 200 feet. When completed, all the individual maps will be
#               combined into a single map book.
#
# Author:      David W. Allen, GISP
#
# Created:     10/25/2013
# Copyright:   (c) David 2013
#
#-------------------------------------------------------------------------------


# Import the modules
import arcpy
```

5. **Create the reference to the map document with an object named** thisMap, **as shown in the graphic. Note that the full path is given so that this script can be run as a stand-alone script.**

```
# Create a map document object for the map document
thisMap = arcpy.mapping.MapDocument(r"C:\EsriPress\GISTPython\Maps\Tutorial 3-3.mxd")
```

6. **Create a data frame object to reference the Water Utility data frame. This method shows a way to find the correct data frame with a single index, even if it is not the first data frame (index[0]), as shown:**

```
# Create a data frame list object of the Water Utility data frame
# Adding the index number at the end of the data frame object creation
# means that you are working with only the desired data frame
myDF = arcpy.mapping.ListDataFrames(thisMap)[0]
if myDF.name <> "Water Utility":
    myDF = arcpy.mapping.ListDataFrames(thisMap)[1]
```

7. **Use the ListLayoutElements method to create a list object of all the graphic elements in the map. Then use a for statement to loop through the elements to assign each element to a variable, as shown:**

```
# Create a list object of the map elements
myMapElements = arcpy.mapping.ListLayoutElements(thisMap)

# Use a loop to assign the elements you want to control to variables
for element in myMapElements:
    if element.name == "Title":
        mapTitle = element
    elif element.name == "PropDesc":
        mapDesc = element
    elif element.name == "LeftVert":
        mapLeftVert = element
    elif element.name == "RightVert":
        mapRightVert = element
    elif element.name == "TopHoriz":
        mapTopHoriz = element
    elif element.name == "BotHoriz":
        mapBotHoriz = element
print mapTitle.text
print mapDesc.text
```

This technique performs the loop through the element list only once, saving time and code as the elements are referenced throughout the script. Note the inclusion of print statements to track the progress of the script.

8. **Create a list object of the layers in the table of contents. Find the Parcels and SamplingStations layers, and assign them to variables for future use. Create a feature layer of the sampling stations with a wildcard to include only the operational stations, as shown:**

```
# Get a list of all the layers in the table of contents
myLayers = arcpy.mapping.ListLayers(myDF)

# Use a for loop to find the layers 'Parcels'
# and 'SamplingStations' in the list
# Assign them to new variables for easier reference

for lyr in myLayers:
    if lyr.name == "Parcels":
        print "Found the parcels!"
        layerParcels = lyr
    elif lyr.name == "SamplingStations":
        print "Found the sampling stations!"
        layerSStations = lyr

# Create a feature layer to hold only the active sampling stations
arcpy.MakeFeatureLayer_management(layerSStations,"ActiveStations_lyr",'"Status"=\'Operational\'')
```

The feature layer of the sampling stations is used to access each station for the maps. Note that the Parcels layer is not being put into a feature layer. Although you can do selections in a feature layer, you cannot use them to manipulate the map extent or scale. Because feature layers exist in virtual memory, their connections to the map document and data frame are removed. Changing the extent or scale of a feature layer does not affect the map document. You may, however, get the extent of a feature layer and use it to set the data frame extent, which would have the effect of zooming to the feature layer.

9. Add the code to create a new folder in your MyExercises folder named WaterExhibitMaps **to hold the output PDF documents, as shown:**

```
# Create a new folder to hold the output PDF documents
arcpy.CreateFolder_management(r"C:\EsriPress\GISTPython\MyExercises", "WaterExhibitMaps")
```

Each completed map document is exported to this folder as a PDF document. At the end of the script, access this folder to combine all the files into one PDF document.

The script so far includes the code to set up all the data and workspaces for the geoprocessing and map creation tasks to be done. Next, set up a cursor to step through the features in the sampling stations feature layer. Feature selections and map document manipulations will be made for each feature, producing an individual map for each station.

10. Set up a search cursor for the sampling stations feature layer. Include the field Desc **with the cursor to be used as the map title. Write a for statement to iterate through each feature, as shown:**

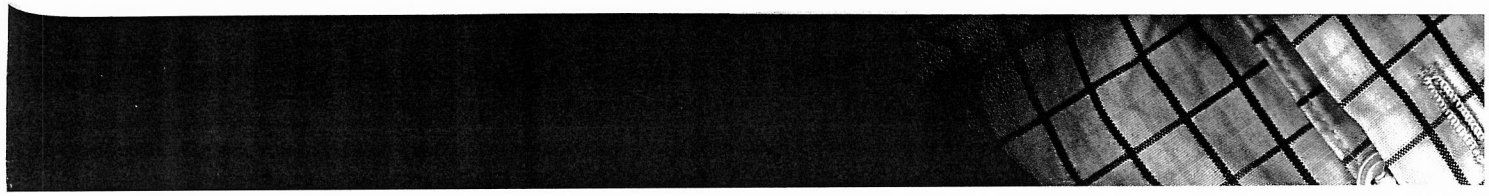
```
# Create a cursor object for the sampling stations
stationCursor = arcpy.da.SearchCursor("ActiveStations_lyr","Desc")

# Iterate through the features of the sampling stations
for feature in stationCursor:
```

The field Desc will be used to name each output map. The cursor object will contain a single attribute value for each feature, which means that index number 0 will hold the value from that field. If more fields were included in the cursor initialization, the fields would be accessed in the order in which they were called in the cursor command, with index numbers starting at one.

11. Set a variable named featureDesc **to contain the attribute value for the current feature, as shown:**

```
    # Retrieve the station name from the current feature
    featureDesc = feature[0]
```

The current feature in the cursor is referenced but not selected. To perform the selection with the 30-foot buffer, select the currently referenced feature, which can be done using the query "Desc" = 'featureDesc.' Selecting this single item makes it easy to select by location.



12. Add a select by attribute function to select the currently referenced feature. Use a select by location function to select all the parcels within 30 feet, as shown:

```
# Select the current feature in the Parcels layer
# SelectLayerByAttribute_management (in_layer_or_view,
#           {selection_type}, {where_clause})
arcpy.SelectLayerByAttribute_management("ActiveStations_lyr","NEW_SELECTION",\
'"Desc"=\'' + featureDesc + "'")

# Select the parcels within 30 feet of this feature
# SelectLayerByLocation_management (in_layer, {overlap_type},
#           {select_features}, {search_distance}, {selection_type})
arcpy.SelectLayerByLocation_management(layerParcels,"WITHIN_A_DISTANCE",\
"ActiveStations_lyr","30")
```

With the parcel feature selected, you can alter the extent of the map to show the current sampling station and the parcels around it. Accessing the extent property of the selected features and passing this along to the data frame would be fine, but the zoomToSelectedFeatures() method accomplishes this task with one line of code. Because the sampling station selection was done on a feature layer, it will have no effect on the map extent. The parcel selection, which was done on the layer in the table of contents, will control the zoom.

Once the zoom is completed, check the scale to make sure that it is not zooming in too closely. A scale of 1:1800 should be the closest zoom, and if the scale factor is larger than that, it is a good idea to increase the scale by about 10 percent. This change keeps the items at the edge of the selected set from touching the map border. Note that the arcpy.RefreshActiveView or arcpy.RefreshTOC functions are not necessary here. These functions refresh the screen and have no effect on a stand-alone script.

13. Add the code to zoom to the selected features. Then test to see if the scale factor is below 1800. If so, set the scale to 1800, and if the scale is larger, add a 10 percent margin to the scale, as shown:

```
# Zoom the map to the selected features
myDF.zoomToSelectedFeatures()
# Adjust scale to at least 1:1800
if myDF.scale < 1800:
    myDF.scale = 1800
else:
    myDF.scale = myDF.scale * 1.1
```

The map area is now set to show the desired features at an appropriate scale. The next phase is to set the map title and to build a chart showing the descriptions of the selected parcels. Refer back to the variable names assigned to the different graphic items, noted when you first opened the map document, to get the names of the objects you will change. The graphic storing the map title was set to the variable mapTitle, so to change its text, access the mapTitle.text property.

14. **Set the map title to the current station description, stored in the featureDesc variable, as shown:**

```
    # Set the text in the map
    # Change the title of the map to this name
    mapTitle.text = featureDesc
```

The chart consists of the property description and the three lines abutting the text. If you examine the map document, you will see these graphic items below the virtual page. In this location, these items will not appear on any printed or exported maps. Clone these items, and then move the clones into place to represent each property that is selected. Earlier in the script, the elements were given the variable names mapLeftVert, mapRightVert, and mapDesc. Each element has a property for the X and Y page coordinates named elementPositionX and elementPositionY.

To create the chart, set up variables to hold the page coordinates for the first set of lines and the first chart text derived from the first parcel. Clone the elements, and move them to these positions. For each additional parcel, move the page coordinates down 0.25 inches, and place a new set of cloned items in that position.

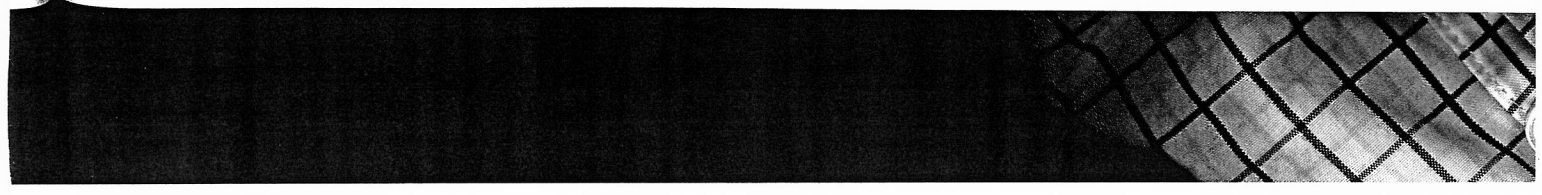
15. **Set up variables to hold the coordinates for the first set of lines and text using the coordinate values shown:**

```
    # Set up the locations for the lines and text used in the chart
    # For each text value, the existing items will be cloned and used
    # in the map.

    # Set up the coordinates for the lines and text of the chart
    botLineX = 0.32
    botLineY = 2.00
    leftLineX = 0.32
    leftLineY = 2.25
    rightLineX = 3.41
    rightLineY = 2.25
    chartTitleX = 0.38
    chartTitleY = 2.21
```

Many parcels may be selected for each map. To get all the proper descriptions into the charts, set up a cursor to scroll through the selected properties. This parcel cursor is nested inside the sampling station cursor. For each sampling station, the parcel cursor will return multiple parcels. When the end of the parcel cursor is reached, the sampling station cursor will move to the next feature, and the process will start again.

16. **Set up a search cursor for the parcels. Have the cursor return the fields Prop_Des_1 and Prop_Des_2, which will be used in the chart. Then set up the for statement to step through the selected parcels, as shown:**

```
    # Get the values of the property description and build chart
    fieldList= ["Prop_Des_1","Prop_Des_2"]
    parcelCursor = arcpy.da.SearchCursor(layerParcels,fieldList)
    for label in parcelCursor:
```



The text element containing the property description can be cloned into an object named textClone. The suffix _clone makes it easier to select and delete the cloned items before the next map is created. Once the item is cloned, it is given the concatenated value of the two property descriptions. Then it is moved into position using the coordinates shown in the preceding graphic.

17. Add the code to clone the element PropDesc, set it to the concatenation of the two property descriptions from the selected parcels, and move it into position, as shown:

```
        # Clone the chart title and set the text
        textClone = mapDesc.clone("_clone")
        textClone.text = str(label[0]) + ", " + str(label[1])
        # Move the new text into place
        textClone.elementPositionX = chartTitleX
        textClone.elementPositionY = chartTitleY
```

18. Write the code to perform a clone operation on the three lines used to make the chart, and move them into place, as shown:

```
        # Clone the lines and move them into place
        leftLine = mapLeftVert.clone("_clone")
        rightLine = mapRightVert.clone("_clone")
        botLine = mapBotHoriz.clone("_clone")
        leftLine.elementPositionX = leftLineX
        leftLine.elementPositionY = leftLineY
        rightLine.elementPositionX = rightLineX
        rightLine.elementPositionY = rightLineY
        botLine.elementPositionX = botLineX
        botLine.elementPositionY = botLineY
```

The first line of the chart is finished. If there are more parcel descriptions to add, the parcelCursor repeats, and another feature is selected. In order for this line in the chart to appear in the right location, move everything down 0.25 inches, which involves subtracting that value from the y-values of the lines and text. The x-values remain unchanged. This adjustment should be made before the cursor repeats.

Once all the items in the cursor have been processed, delete the cursor object, which is done by going back to the same indent level as the for statement and issuing the del command.

19. **Add the code to reduce all the y-coordinates by 0.25 inches, and delete the cursor, as shown:**

```
        # Move the Y coordinate values down 0.25 inches
        botLineY = botLineY - 0.25
        leftLineY = leftLineY - 0.25
        rightLineY = rightLineY - 0.25
        chartTitleY = chartTitleY - 0.25
    del parcelCursor
```

When the parcel cursor is finished running, the map for the first sampling station will be complete. Export the map to a PDF document before repeating the process for the next station.

20. **Export the map to a PDF document into the folder you created in step 9, as shown:**

```
# Export the map to a PDF document
# ExportToPDF (map_document, out_pdf, {data_frame}, {df_export_width},
#   {df_export_height}, {resolution}, {image_quality},
#   {colorspace}, {compress_vectors}, {image_compression}, {picture_symbol},
#   {convert_markers}, {embed_fonts},
#   {layers_attributes}, {georef_info}, {jpeg_compression_quality})
arcpy.mapping.ExportToPDF(thisMap, r"C:\EsriPress\GISTPython\MyExercises\WaterExhibitMaps\Project_" \
 + mapTitle.text + ".pdf", resolution=100,image_quality="NORMAL")
```

In this example, the file resolution and image quality are reduced to save file space. You can research the other settings in ArcGIS for Desktop Help.

When the map has been exported, remove all the cloned items from the map in preparation for the next map. Because the clones all have the suffix _clone, they can be easily identified using a wildcard with the ListLayoutElements() function. Use a for statement to iterate through the list and delete the elements.

21. **Create a list object with the cloned elements, and use a for statement to step through and delete the elements, as shown:**

```
    # Delete the cloned items and reset for the next map
    cloneGraphics = arcpy.mapping.ListLayoutElements(thisMap,wildcard="*clone*")
    for graphic in cloneGraphics:
        graphic.delete()
```

This is the end of the sampling stations cursor. A complete map has been created, and the map document has been reset for the next map. The cleanup at the end of the cursor's process is to delete the cursor and the map document you have been accessing. Note that these return to the same indent level as the for statement you used to step through the sampling stations cursor.

22. **Add the code to close the cursor and the map document, as shown:**

```
# Delete the cursor
del stationCursor

# Delete the map document object
del thisMap
```

Take all the PDF documents created by this script and combine them into a single PDF document. The "Getting Started with arcpy.mapping" tutorial in ArcGIS for Desktop Help has a section demonstrating this process, and the code shown in the graphic for step 23 is derived from that example.

23. **Combine all the PDF documents in the WaterExhibitMaps folder into a single PDF document. This process involves creating the file, setting the workspace, creating a list of all the PDF documents in the destination folder, iterating through that list, and appending each file to the final document, as shown:**

```
# Combine all the PDF documents into a single map book
finalPDF_filename = r"C:\EsriPress\GISTPython\MyExercises\WaterExhibitMaps\StationsMapBook.pdf"
finalPDF = arcpy.mapping.PDFDocumentCreate(finalPDF_filename)

# Create a list of all PDF documents in the new folder
arcpy.env.workspace = r"C:\EsriPress\GISTPython\MyExercises\WaterExhibitMaps"

# Copy each file with a .pdf extension to a dBASE file
# List automatically looks in the defined workspace
pdfPath = r"C:\EsriPress\GISTPython\MyExercises\WaterExhibitMaps\\"
for file in arcpy.ListFiles("*.pdf"):
    print file
    finalPDF.appendPages(pdfPath + file)

# Commit changes and delete variable reference
finalPDF.saveAndClose()
del finalPDF
```

24. **Save the SampleStationsMapBook.py script. Close the map document, and run the script. When the script is finished, look in the destination folder for the output PDF document.**

Here's the pseudo code for this script:

```
# Import the modules
# Create a map document object for the map document
# Create a data frame list object of the Water Utility data frame
# Adding the index number at the end of the data frame object creation
# means that you are working with only the desired data frame
# Get a list of all the layers in the table of contents
# Create a list object of the map elements
# Use a loop to assign the elements you want to control to variables
# Use a for loop to find the layers 'Parcels'
# and 'SamplingStations' in the list
# Assign them to new variables for easier reference
# Create a feature layer to hold only the active sampling stations
# Create a new folder to hold the output PDF documents
# Create a cursor object for the sampling stations
# Iterate through the features of the sampling stations
# Retrieve the station name from the current feature
# Select the current feature in the Parcels layer
# SelectLayerByAttribute_management (in_layer_or_view, {selection_type}, {where_clause})
# Select the parcels within 30 feet of this feature
# SelectLayerByLocation_management (in_layer, {overlap_type}, {select_features}, \
#           {search_distance}, {selection_type})
#
# Zoom the map to the selected features
# Adjust scale to at least 1:1800
# Set the text in the map
# Change the title of the map to this name
# Set up the locations for the lines and text used in the chart
# For each text value, the existing items will be cloned and used
# in the map.
# Set up the coordinates for the lines and text of the chart
# Get the values of the property description and build chart
# Clone the chart title and set the text
# Move the new text into place
# Clone the lines and move them into place
# Move the Y coordinate values down 0.25 inches
# Export the map to a PDF document
# ExportToPDF (map_document, out_pdf, {data_frame}, {df_export_width}, {df_export_height}, \
#             {resolution}, {image_quality},
#             {colorspace}, {compress_vectors}, {image_compression}, {picture_symbol}, \
#             {convert_markers}, {embed_fonts},
#             {layers_attributes}, {georef_info}, {jpeg_compression_quality})
# Delete the cloned items and reset for the next map
# Delete the cursor
# Delete the map document object
# Combine all the PDF documents into a single map book
# Create a list of all PDF documents in the new folder
# Copy each file with a .pdf extension to a dBASE file
# List automatically looks in the defined workspace
# Commit changes and delete variable reference
```

This script is rather complex, but when it works correctly, you can automatically make dozens of maps.

Exercise 3-3

Search ArcGIS for Desktop Help for the topic "Building map books with ArcGIS." Work through these examples, and research the ways the Data Driven Pages commands can be used in a Python script.

Tutorial 3-3 review

The mapping module in ArcPy is useful in creating map books, even if the page layouts are different from one map to another. The use of a cloning technique allows you to place extra graphic elements in a map layout and use these elements to build such things as charts, tables, and graphs. A graphic item can be cloned and moved into position as many times as needed.

Another interesting technique demonstrated in this tutorial is the process of creating a list of graphic elements and assigning the elements you wish to work with to a variable. This means that your cursor has to iterate through the list of elements only once, which can save a lot of time if you are doing a lot of work with elements.

You also learned the variety of tasks that can be performed involving PDF documents. Several map pages, title sheets, or even indexed values, such as street names, can be placed into separate PDF documents, and then these pages can be grouped into a single PDF document for easy printing and sharing.

Study questions

1. Would it be possible to nest more than one cursor (iteration) within another? What are the consequences of trying to do multiple iterations?
2. What other functions can be used to handle PDF documents in arcpy.mapping? Give examples.
3. What other uses might you see for the cloning technique?