The use of a feature layer is important in these situations because it provides a way to make selections and process the data without putting the source data in jeopardy. These types of layers can be created for tables, called *table view*, and you will see later how a query can extract a subset of the data. For this tutorial, the feature layer was required because the selection processes only work either in a map document that is referenced from the table of contents or in a feature layer or table view. These processes will not work on layers referenced by their source path.

Study questions

1. Name a few other geoprocessing tools, and look up these tools' syntax and sample scripts in ArcGIS for Desktop Help.
2. When is it useful to work with a copy of the data in a feature layer or table view rather than access the source data directly?
3. Look up the syntax for cursors in ArcGIS for Desktop Help, and show the different types.

Tutorial 2-4 Using while statements

The iteration of a script can be controlled with a while statement, which will cause the script to continue running until a condition is met. The number of iterations will change depending on the condition.

Learning objectives

- Using cursors with while statements
- Iterating through features

Preparation

Research the following topics in ArcGIS for Desktop Help:

- "Cursor (arcpy)"
- "Copy Features (Data Management)"

Introduction

In tutorial 2-3, you learned the simple use of a cursor. A cursor can be used to go through an entire dataset feature by feature or row by row and to perform an individual task on each item it finds. By default, geoprocessing tasks act only on the selected feature, and a cursor has the effect of making the current cursor item the only selected feature.

The cursor you used in tutorial 2-3 retrieved the value of a single field to be used in a condition statement. Cursors can access any field in the input dataset, which is specified in the cursor setup. The for statement is used with the cursor to handle the looping through the dataset. The for statement and all the code included in its indent level is run for each item in the cursor. Mastering the use of cursors is important in writing a script that can handle a variety of tasks.

In addition to a cursor, this tutorial uses another control statement, the while statement. The programmer sets up a condition, and the while statement will continue running the script or portion of a script until the condition is met. Once the condition is met, the rest of the code runs. The while statement is often used to process subsets of the data or to monitor the number of times a process is performed. One thing to be careful about, however, is that if no scenario exists in which the condition of the while statement equates to False, the while statement will never end.

Scenario

The library in Oleander bought a bookmobile at the Fort Worth Police Auction and is investigating the best way to put it in service. The library has decided that the bookmobile will make periodic stops at apartment complexes because of the density of people it can reach with one stop. The question is how to implement bookmobile stops for single-family residential customers.

The head librarian has an idea that he wants to investigate. He has selected six places in the city where they would be able to set up stops. The librarian wants to serve about 200 households at each stop, but he is not sure how far those 200 households would have to travel to get to the bookmobile. The process is complicated, so you will want to build a script to test locations. Also, the librarian will be selecting more locations for further research based on your initial results.

The process is to go through each of the six sites, one by one, and do some analysis on the household count. For each site, select all the parcels within 150 feet, and add up the number of dwelling units. If the number exceeds 200, stop the selection process. If the number does not exceed 200, select all the parcels within another distance of the currently selected set, and add up the households again. If the number does not exceed 200, repeat the selection until it does. Output the selected set to a new feature class, and move on to the next site.

Data

The map document for this tutorial includes a file named BookmobileLocations that contains the six sites the librarian chose. The field Marker contains the site name. Also included are the parcels with a field named DU, which has the count of dwelling units.

SCRIPTING TECHNIQUES

This script must do two things: iterate through a set of features, and perform a select operation at each iteration. A good practice is to write and test the code for the first process, and then add the second process. This practice simplifies the troubleshooting process because you will be limiting the number of errors that can occur. Set up and test a cursor to access each of the features in the input feature class. Once this cursor is running, add the second process.

This second process includes another type of iterator called a *while statement*. As you know, a cursor will go through the records one by one and stop when all the records have been accessed. With a while statement, you will define a condition and write code to process the data. This process repeats as long as the condition you set evaluates to True. As with the other types of condition statements and iterators, the processes associated with the while statement hold at an indent level to identify where the process stops.

This tutorial requires you to imbed one type of iterator inside another, which can be tricky. Make sure that each iterator has a condition that will cause it to complete. The cursor is easy because it completes when the last record is accessed, but the while statement is not as straightforward. If you were to unknowingly set a condition that could never be false, the while statement would never end. For this reason, it is sometimes advisable to add a counter inside the while statement to limit the number of possible iterations to a number known to be past the script's reasonable operation. For example, if you feel that the script can complete its tasks in 15 iterations or less for all cases, then stop the script once it reaches 20 iterations. Stopping the script involves imbedding a count variable inside the while statements and using an if statement to end the script if the count exceeds the predetermined maximum. An example of this action is given later, and you can add this as an option.

Use while statements

1. **Open the map document Tutorial 2-4.**

2. **Write the pseudo code for this project. (Hint: you will need a cursor to go through the six sites and a while statement to keep adding more parcels until the dwelling unit count exceeds 200.) Completed pseudo code is shown at the end of this tutorial.**

3. **Create a new Python script in your IDE, and name it** BookmobileAnalysis.py**. You can use the familiar template lines to start your script.**

4. **Import the appropriate modules, and set the workspace environment to** C:\EsriPress\GISTPython\Data\City of Oleander.gdb**, as shown:**

```
try:
    # Import the modules
    import arcpy
    from arcpy import env

    # Set up the environment
    env.workspace = r"C:\EsriPress\GISTPython\Data\City of Oleander.gdb\\"
    env.overwriteOutput = True
```

Start by writing a cursor to access each of the individual sites in the BookmobileLocations feature class. Get the code for the cursor working first, and add the while statement later. Writing the code in this order will not only be easier to understand, but will also be easier to troubleshoot.

As a general housekeeping rule, make a feature layer for each of the feature classes you will be working with. In addition, the feature layer for parcels should include only the single-family households, which you can find by adding an optional selection clause (DU = 1). A feature layer is only a temporary copy of the data, so any changes made to it are also temporary unless specifically saved to the script.

5. **Make two feature layers named** Parcels_lyr **and** Locations_lyr **from the Parcels and BookmobileLocations layers, as shown. Add a query to the Parcels_lyr statement to select only the single-family units. Check the tool reference if you are unsure of the query syntax.**

```
    # Set up cursor for the bookmobile sites
    arcpy.MakeFeatureLayer_management("Parcels","Parcels_lyr",'"DU"= 1')
    arcpy.MakeFeatureLayer_management("BookmobileLocations","Locations_lyr")
```

Next, create a search cursor to go through the Locations_lyr file, which contains the setup sites for the bookmobile.

6. **Create a new search cursor named** siteCursor**. Have the cursor bring in the field Marker, which contains the unique site name for each location, as shown:**

```
    siteCursor = arcpy.da.SearchCursor("Locations_lyr","Marker")
```

This particular layer has six features, so you know that the siteCursor object will contain the six values of the field Marker. The for statement will reference each row in the attribute table, one at a time, and allow you to perform geoprocessing tasks for each value. The syntax is:

for {variable name to track the current row} in {cursor name}:

Note that the statement ends with a colon. The colon signals the start of an indent level, and all the code that maintains that level will be run once for each row. The end of the for statement is signified by going back one indent level.

7. **Construct a for statement using the siteCursor object, as shown:**

```
siteCursor = arcpy.da.SearchCursor("Locations_lyr","Marker")
for row in siteCursor:
```

The variable named row will now hold the value of the field Marker. It is an indexed list variable, so you must add an index number at the end of the variable to reference which field value to use. In this case, only one field was brought into the cursor, so row[0] will return that value. If other fields were brought into the cursor, they would be referenced in the order listed in the cursor statement, and the variable row[x] would hold their index number.

8. **Create a new variable named** siteName **that holds the value of Marker at each iteration, as shown:**

```
siteCursor = arcpy.da.SearchCursor("Locations_lyr","Marker")
for row in siteCursor:
    siteName = row[0]
```

The code from steps 6 and 7 is shown so that you can note the indent level. Maintain this level until the cursor routine is finished.

In each iteration through the cursor, select the location feature from Locations_lyr, and use that location to select all the parcels within 150 feet.

9. **Research the various select tools available in ArcMap, and note the syntax of the code. In this example, the Select_analysis tool is used. Store the selected feature in your scratch folder as** SiteTemp **(MyExercises\Scratch\Temporary Storage.gdb\SiteTemp). Be careful to correctly format the where clause parameter in this tool's syntax. Write your code, and check it against the code shown:**

```
    # Select parcels within 150 feet of the new selection
    arcpy.Select_analysis("Locations_lyr",\
    r"C:\EsriPress\GISTPython\MyExercises\Scratch\Temporary Storage.gdb\SiteTemp",\
    '"Marker" = \'' + siteName + "\'")
```

The formatting of the where clause is tricky. As an example, the resulting statement for the first site should be as follows:

"Marker" = 'Site 1'

Decode the sequence of characters used in the selection statement. Use your Python references for help.

After selecting the site feature, next select all the parcels within 150 feet. You have used this tool before and should be familiar with its syntax.

10. **Add a Select Layer By Location tool, and configure it to use the selected feature from step 9 and a 150-foot search distance, as shown:**

```
    arcpy.SelectLayerByLocation_management("Parcels_lyr","WITHIN_A_DIST
    r"C:\EsriPress\GISTPython\MyExercises\Scratch\Temporary Storage.gdb
    "150","NEW_SELECTION")
```

A condition statement called a *while loop* will be placed in this part of the code, but for now, finish the processing to be performed in the cursor. Export the selected features to a new feature class, and name it using the value from the field Marker.

11. Research a tool to export features to a new feature class in your MyAnswers geodatabase. Find the proper syntax, and write your code before referencing the code shown:

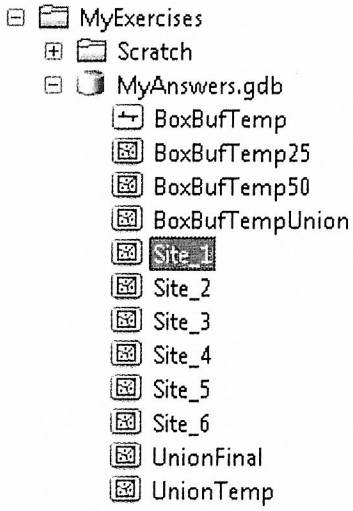
```
        # Export the selected features to a new feature class
        # Use the current site name as the file name
        arcpy.CopyFeatures_management("Parcels_lyr", r"C:\EsriPress\GISTPython\MyExercises\MyAnswers.gdb\\" \
        + siteName.replace(" ","_"))

        print siteName + " Output OK!"
        # Move to the next site and repeat
```

There are two things to note here. First, the site names have a space in them, which is not allowed in a feature class name. You can use the Python string method .replace() to change the space to an underscore. Second, a print statement was added to print the site name and to confirm that the steps were completed successfully. If no print statement was added, you would get no indication of the script's status.

Next, test the code, and see if the cursor is working as expected.

12. Save your script and run it.

The script should create new feature classes in the MyAnswers geodatabase, as shown:



Put some of the files into your map document and see if they meet your expectations.

Now move on to the while statement.

A while statement sets a condition and continues looping through its processes until the statement is false. In this project, the count of selected features must be less than 200. Check your Python resource books or ArcGIS for Desktop Help for the syntax and usage of while statements. Basically,

set up a variable to hold the count of parcels, and then start the while statement with the condition that the variable not exceed 200.

13. In the section of the code for the while statement, add a variable to get the count of features selected in the parcels layer. Then set up the while statement using this variable, as shown:

```
# Start a while statement until number of dwelling units exceeds 200
parcelCount = int(arcpy.GetCount_management("Parcels_lyr").getOutput(0))
print parcelCount
while parcelCount < 200:
    # All statements at this indent level are part of the while loop
```

Note that the while statement ends with a colon and starts a new indent level. Everything that is indented at this level runs within the while statement. To end the statement, drop back one indent level.

The variable that is controlling the while loop (parcelCount) is called a *sentry variable* because it watches over the loop and provides the value that will eventually cause it to end. This variable was initialized with an ArcPy function named GetCount(), which is useful in a lot of Python scripts. Note that because parcelCount is a Python object and not a simple variable, you must add .getOutput(0) at the end of the object to reference the value's index number.

When you use the sentry variable in the while statement, you must make sure that the value can be true on the first run. Otherwise, the loop will never run. The second thing to watch for is that the sentry variable is updated somewhere in the loop and that a condition eventually exists where the condition of the while statement is false. Otherwise, the loop will never end. Some programmers will nest a count variable in the while loop and cause the loop to end after a certain number of iterations are completed.

If the count of parcels is less than 200, make another selection to add all the parcels within a distance of the currently selected parcels to the selection set. This process will use Select Layer By Location with a distance of 150 feet—check the tool Help to determine how to set the overlap type to select features within a certain distance and how to set the selection type to add features to the current set.

14. Add the selection statement, making sure to select the parcels, and add the parcels to the currently selected features. Copy the parcelCount statement to get a new count, as shown:

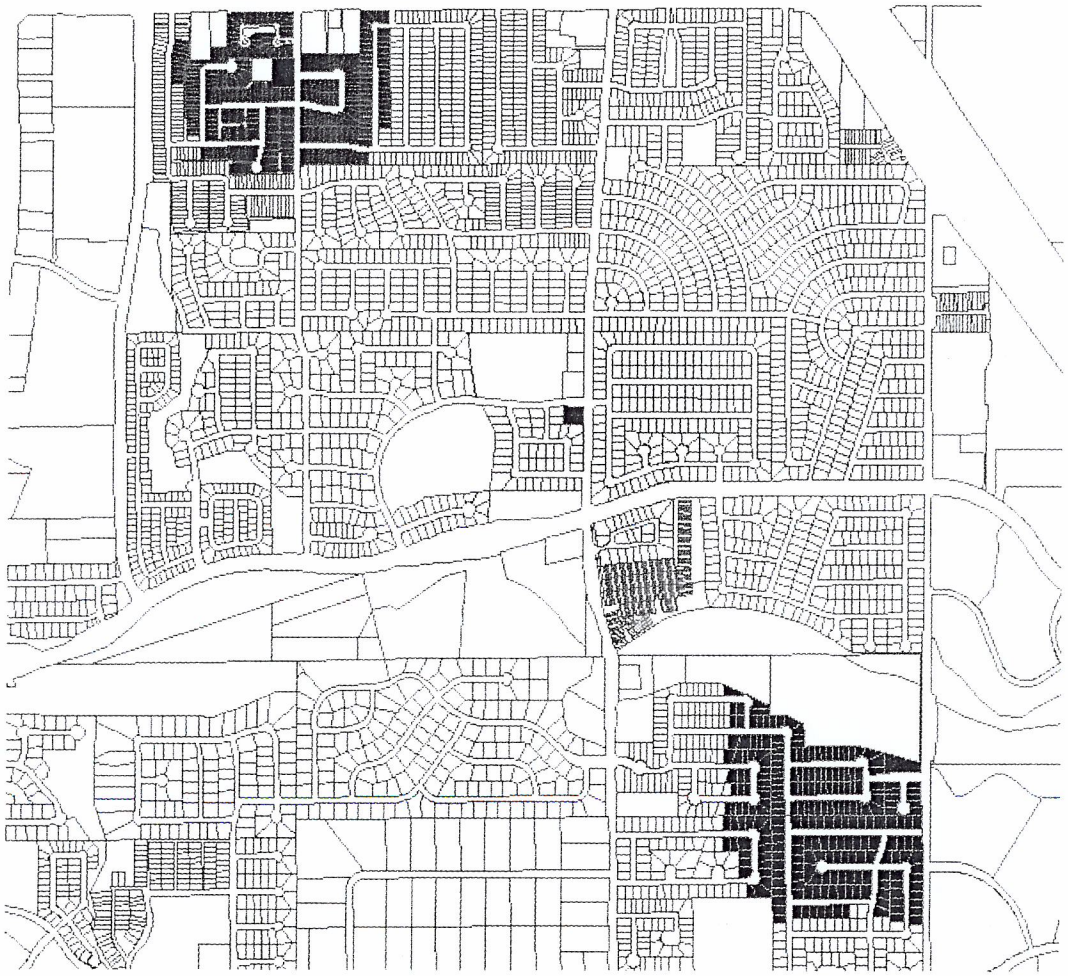
```
# Add to the selected set all property within 150 feet and redo count
# SelectLayerByLocation_management (in_layer, overlap_type, select_features,
#                                   search_distance, selection_type)
arcpy.SelectLayerByLocation_management("Parcels_lyr","WITHIN_A_DISTANCE",\
"Parcels_lyr","150","ADD_TO_SELECTION")

parcelCount = int(arcpy.GetCount_management("Parcels_lyr").getOutput(0))
print parcelCount
# Exit the while statement when count exceeds 200
```

The selection will increase the number of currently selected features, so the value of the parcelCount variable will also increase. This condition will eventually cause the while loop to end.

As a bonus, add a count variable, and limit the while loop to eight iterations. The code would look something like this:

```
# This count routine will repeat eight times
# Initialize the count variable outside the while loop
myCount = 1
# Set the starting condition as True
theCondition = True

while theCondition == True:
    # Inside the while loop check the value
    if myCount == 8:
        # Set the condition to False
        theCondition = False
    else:
        print "The current count is " + str(myCount)
        myCount = myCount + 1

print "All done! The final count equals " + str(myCount) + "."
# Script is saved in \GISTPython\Data for reference
```

The code is now finished; save and run it. Drag a few of the output layers to your map document to see the results, as shown:



Here's the pseudo code for this project:

```
# Set up cursor for the bookmobile sites
# Select parcels within 150 feet of the new selection
# Start a while statement until number of dwelling units exceeds 200
# Add to the selected set all property within 150 feet and redo count
# Exit the while statement when count exceeds 200
# Export the selected features to a new feature class
# Use the value of the field Marker as the file name
# Move to the next site and repeat
```

Exercise 2-4

A similar project has cropped up in the Public Works Department. The City of Oleander runs its own public water supply system and must maintain a state certification with annual inspections. The water system has water sampling stations where samples can be drawn and sent to the state lab for testing. Oleander wants to move up to a level 1 certification, which will require analysis of the pipes immediately adjacent to the sampling stations.

Open the map document Exercise 2-4.

Your task is to write a Python script that will select the 10 adjacent pipes around each water sampling station and output these to a new file using the description as the name. Be careful because some of the sampling stations are closed and should not be included in the analysis. (**Hint:** use the selection type dealing with common boundaries between the features rather than a set distance to select adjacent pipes.)

Tutorial 2-4 review

Nesting looping statements is a way to perform multiple processes on your data. In this case, you nested a while statement inside the cursor. Each feature was selected one at a time, and a complex process was performed.

The while statement included a condition for the looping process. As long as the condition of the while statement remains true, the loop will continue. Make sure that at some point the condition equates to false for the loop to end, or your script might never complete its processing. As a fail-safe maneuver to prevent an endless loop, a while statement with a count variable adds additional control to the number of times the processes loop.

Study questions

1. Give an example of when you might use a while statement in your data processing.
2. Is there a limit to the number of loops that can be nested in each other?
3. Besides a counter, what other control(s) could have been included to make sure it was not an endless loop? (**Hint:** use the count of the features.)

Tutorial 2-5 Using lists and for statements

A list object can be used to store a list of such items as features, files, and tables. This list object can be used in a for statement to iterate through the list. This technique provides a loop that iterates a finite number of times—once for each item in the list.

Learning objectives

- Creating and using lists
- Creating and calculating fields

Preparation

Research the following topics in ArcGIS for Desktop Help:

- "Create lists of data"
- "ListFeatureClasses (arcpy)"
- "Add Field (Data Management)"
- "Calculate Field (Data Management)"

Introduction

In tutorials 2-3 and 2-4, you learned how to iterate through features in a feature class using a cursor and a for statement. You can also use the for statement to iterate through other items by placing them in a list object. A list object is a special type of Python variable that can hold a list of values, and these values are indexed just as you have seen with the Python text-handling functions.

List objects can be defined simply by providing the values separated by commas, but ArcPy has several special functions to create lists from elements commonly used in geoprocessing tasks. These functions include listing fields, layers, and raster files. There are also functions to list map layout components, such as data frames and layout elements. The output of these tools is, in fact, tuples, which are a special type of Python list in which the values contained in the list cannot be changed. Because the use of tuples in ArcGIS is almost identical to how you would use lists, the references here will continue to treat them as standard Python list objects.

Once these ArcPy list objects are created, you can use them with a for statement to iterate through the list contained in the object. Each iteration of the for statement will expose an object that can be used for feature classes that need processing, layout elements that need updating, fields that need changing, and any number of other tasks.

Scenario

The Fire Department needs a count of all the single-family homes and multifamily structures in each of the 44 fire response zones. The secretary in that department is trying to complete this task with a felt-tipped pen and an aerial photo. You see that she has already missed a few structures, misidentified a few, and is spending an inordinate amount of time on this project. You are going to write a script that will be faster and more accurate. The Fire Department's geodatabase contains a polygon feature class for each response zone, or box. You will need to add two fields to each box: one for the single-family count and one for the multifamily count. The secretary has already added the fields to a portion of the feature classes, but the rest of the classes do not have these fields.

Data

The FireDepartment geodatabase contains 44 feature classes representing the box zones. Note that the name structure for these feature classes is the phrase "FireBoxMap" followed by the box ID number. In addition, there is a polygon feature class with the building footprints. Each building has a use code in a field named UseCode (1 = single family, 2 = multi-family).

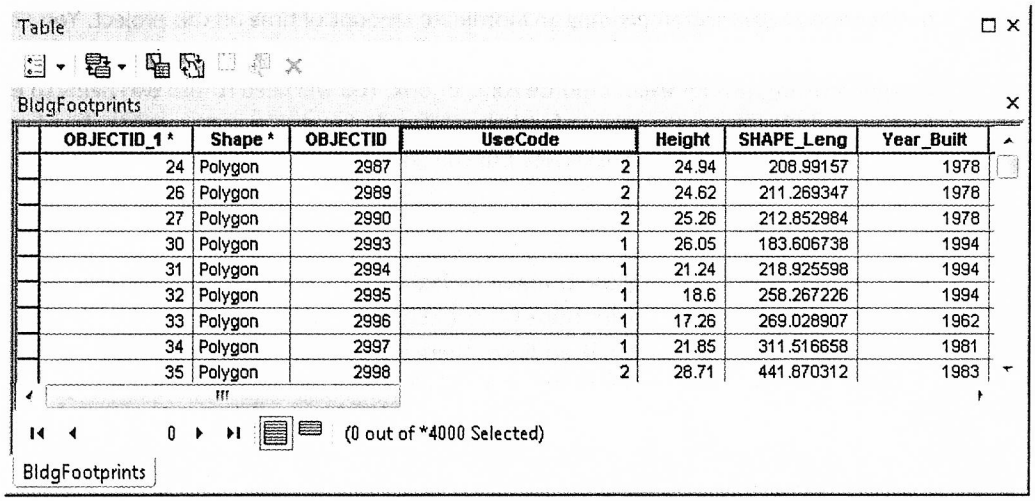
SCRIPTING TECHNIQUES

Because there are 44 different feature classes to be used in this project, the best course of action is to make a list of the feature classes and iterate through the list. A list object can be created using the ListFeatureClasses tool, which can be set to include only files that meet a specified criterion. Once the list object is created, a for statement can be used to iterate through the list. Use a combination list and for routine for feature classes, although this type of iteration can be done with any of the list tools shown at the start of this chapter.

The second technique introduced here is the use of the Calculate Field tool. This tool is used to write values into fields of the attribute table. Any number or text string can be created and stored in a variable, and then the variable can be written to the specified field. The unique aspect of this technique is that the fields are written one by one rather than as a batch, so individual steps that may not be appropriate for the entire dataset can be calculated here.

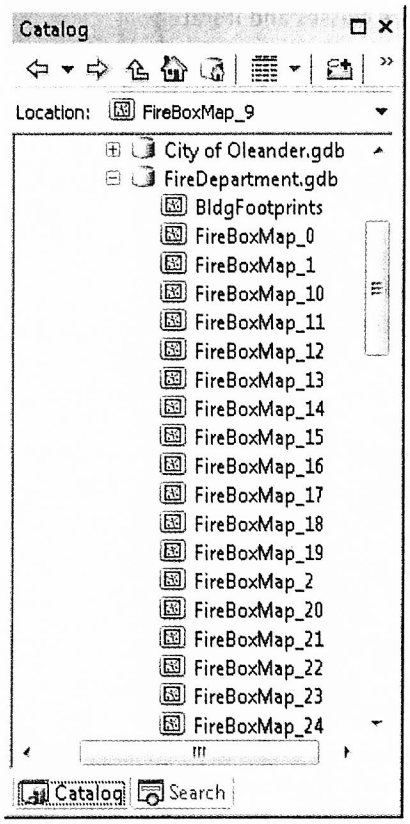
Use lists and for statements

1. Open the map document Tutorial 2-5, and note the BldgFootprints feature class. Open the attribute table, and make note of the field containing the use code (UseCode), as shown:



OBJECTID_1 *	Shape *	OBJECTID	UseCode	Height	SHAPE_Leng	Year_Built
24	Polygon	2987	2	24.94	208.99157	1978
26	Polygon	2989	2	24.62	211.269347	1978
27	Polygon	2990	2	25.26	212.852984	1978
30	Polygon	2993	1	26.05	183.606738	1994
31	Polygon	2994	1	21.24	218.925598	1994
32	Polygon	2995	1	18.6	258.267226	1994
33	Polygon	2996	1	17.26	269.028907	1962
34	Polygon	2997	1	21.85	311.516658	1981
35	Polygon	2998	2	28.71	441.870312	1983

2. In the Catalog window, note the feature classes representing the 44 boxes, as shown:



Develop a plan of action for tackling this project.

3. **Write the pseudo code for this task, which will involve getting a list of all the layers and adding fields among other tasks. Completed pseudo code for this task is found at the end of this tutorial.**

4. **Start a new Python script named DwellingCount.py. Add the comments and template components as in the previous scripts, as shown:**

```
#---------------------------------------------------------------------------
# Name:         Tutorial 2-5
#               Dwelling count
# Purpose:      Count the buildings in each box zone.
#
# Author:       David W. Allen, GISP
#
# Created:      10/25/2013
# Copyright:    (c) David 2013
#
#---------------------------------------------------------------------------

try:
    # Import the modules
    import arcpy
    from arcpy import env

    # Set up the environment

    # ENTER CODE

    # Determine results

except arcpy.ExecuteError:
    print  arcpy.GetMessages(2)
except:
    print "Process did not complete."
```

Notice that a different type of error handling has been added in the except line. The special arcpy.ExecuteError object will trap any errors raised by ArcPy statements. The second except statement will trap any other errors. Check the ArcPy documentation for more information on this object.

Start by making a list object of the feature classes in the FireDepartment geodatabase.

5. Search ArcGIS for Desktop Help to locate an ArcPy function that can be used to get a list of feature classes. Several are shown in the table. Select the one you think is most appropriate, and open and read the tool reference.

ListFields(dataset, wild_card, field_type)	Returns a list of fields found in the input value
ListIndexes(dataset, wild_card)	Returns a list of attribute indexes found in the input value
ListDatasets(wild_card, feature_type)	Returns the datasets in the current workspace
ListFeatureClasses(wild_card, feature_type)	Returns the feature classes in the current workspace
ListFiles(wild_card)	Returns the files in the current workspace
ListRasters(wild_card, raster_type)	Returns a list of rasters found in the current workspace
ListTables(wild_card, table_type)	Returns a list of tables found in the current workspace
ListWorkspaces(wild_card, workspace_type)	Returns a list of workspaces found in the current workspace
ListVersions(sde_workspace)	Returns a list of versions the connected user has permission to use

List functions

The ListFeatureClasses function will work, but you must set the workspace environment and use a wildcard to get only the box zone files.

6. Add the workspace code, and the list code using the examples from the tool reference, as shown:

```
# Set up the environment
env.workspace = r"C:\EsriPress\GISTPython\Data\FireDepartment.gdb"
env.overwriteOutput = True

# Create a list of all the box zone feature classes
fcBoxZones = arcpy.ListFeatureClasses("FireBoxMap*")
#fcBoxZones = arcpy.ListFeatureClasses("FireBoxMap_14")
```

The feature class names can now be retrieved from the list object by using an index number, starting at zero (0). The first would be fcBoxZones[0], and the 44th would be fcBoxZones[43] (remember to start the index numbers at 0). Notice that in the graphic an extra line was added to explicitly select number 14. You can use this for testing the script so that it does not have to iterate through all 44 files.

Now use a for statement to retrieve the files one by one. All the code written in the for statement will run for each file in the list object. When the last file is processed, the for statement will release the script to continue to the end.

7. **Write the for statement to go through all the files, and then check your code against the code shown:**

```
    # Start a for statement to iterate through the files
    for fc in fcBoxZones:
        # Get the first file - it's stored in fc
        print fc
```

8. **Research tools that can create the fields to hold the results. Reference the tool Help, and write the code to add two integer fields named SFCount and MFCount, as shown:**

```
        # Add two fields to hold the results
        arcpy.AddField_management (fc, "SFCount", "LONG")
        arcpy.AddField_management (fc, "MFCount", "LONG")
```

The scenario says that some of the files already have these fields added. The AddField command will automatically detect this and not duplicate a field if it already exists.

Next, get the counts using a select statement. In previous scripts, you copied the feature class to a feature layer and used a query to get only the features you needed. It is interesting to note that multiple feature layers can be created for the same feature class and used in different ways. Consider this option when preparing the select statements.

9. **Add the code to make a feature layer, and perform the selections for single-family homes, as shown:**

```
        # Select the single-family housing units (centroid within polygon)
        arcpy.MakeFeatureLayer_management("BldgFootprints","Buildings_lyr","\"UseCode\" = 1")
        print "Made feature layer"

        arcpy.SelectLayerByLocation_management ("Buildings_lyr", "HAVE_THEIR_CENTER_IN",fc)
        print "Made selection"

        # Count the single-family dwellings
        bldgCount = int(arcpy.GetCount_management("Buildings_lyr").getOutput(0))
        print str(bldgCount)
```

You have used all three of these statements before with slight differences. Be careful about the syntax for the query in the MakeFeatureLayer command. Also, the selection command uses the wording "HAVE_THEIR_CENTER_IN" for the overlap type parameter so that no building will appear in more than one box. The overlap type keywords must be typed exactly as shown in the tool reference. If you were using the overlap type "INTERSECT," a building that crossed the boundary would be selected twice. Note that some print statements were added so that you can follow the progress of the script as it runs.

Next, update the fields in the feature class to contain the count of single-family homes.

10. **Research the tool necessary to put the value bldgCount into the field SFCount, as shown:**

```
# Update the field
# CalculateField_management (in_table, field, expression, {expression_type}, {code_block})
arcpy.CalculateField_management(fc,"SFCount",bldgCount)
print "Updated the SFCount field"
```

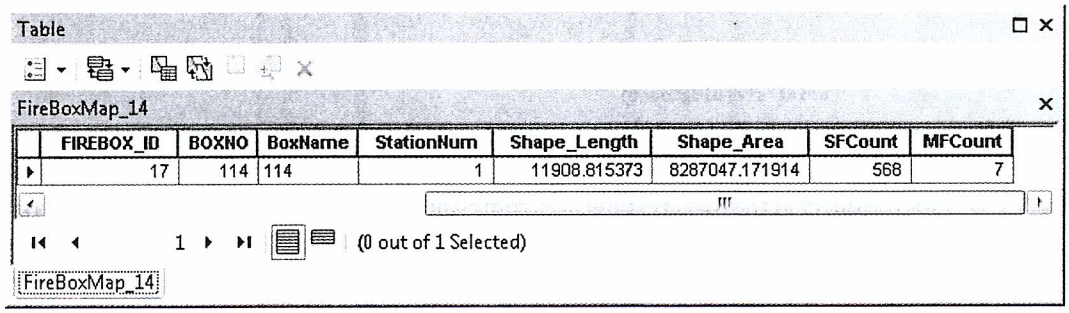
Your turn

The code for selecting the single-family housing is working. Add the code to do the selection, and update the MFCount field for the multifamily buildings, as shown in the graphic. Remember that the use code for the building footprints is 2 for multi-family.

```
# Select the multi-family housing units (centroid within polygon)
arcpy.MakeFeatureLayer_management("BldgFootprints","Buildings_lyr","\"UseCode\" = 2")
print "Made feature layer"

arcpy.SelectLayerByLocation_management ("Buildings_lyr", "HAVE_THEIR_CENTER_IN",fc)
print "Made selection"

# Count the multi-family buildings
bldgCount = int(arcpy.GetCount_management("Buildings_lyr").getOutput(0))
print str(bldgCount)

# Update the field
arcpy.CalculateField_management(fc,"MFCount",bldgCount)
print "Updated the MFCount field"

# Go to the next feature class, and do the selections again
```

The script is almost complete. The counts will be made for both building types, and the fields will be updated.

11. **Save the script. If you were using a single feature class for testing, change the wildcard expression in the list statement to select all the FireBox feature classes. Run the script.**



Here's the pseudo code for this script:

```
# Create a list of all the box zone feature classes
# Start a for statement to iterate through the files
# Get the first file
# Add two fields to hold the results
# Select the single-family housing units (centroid within polygon)
# Count the single-family dwellings
# Update the field
# Select the multi-family housing units (centroid within polygon)
# Count the multi-family buildings
# Update the field
# Go to the next feature class, and do the selections again
```

Exercise 2-5

When you take the results back to the Fire Department, you find the secretary hard at work with a ruler and another printed map. She is trying to measure the linear lane-miles for roads in each box—that is, the length of the street in miles multiplied by the number of lanes. Once again, you tell her that a script can be written to solve this problem and add the results to each of the individual box zone files.

Open the map document Exercise 2-5. The street data and all the box files are shown. Intersect each box file with the streets, and use the fields Shape_Length and Number of lanes to calculate the results into a new field named **LaneMiles**.

Tutorial 2-5 review

As seen in the table of list functions shown in step 5 of this tutorial, there are many list functions in ArcPy. The results of these functions are list objects, which basically contain all the values that are returned. Much like the cursors, a for statement is used to access the values in the list objects. This statement goes through each item in the list one by one and performs the processing tasks you specify. Lists cannot get into an endless loop because they have a finite number of values.

The items in a list object are indexed—for example, text strings in a list. You could pull out an individual item from the list by its index number, if you knew it. Otherwise, you must iterate through the list and find the individual feature class of interest. The list commands also include an optional wildcard value that will let you limit what goes into the list. This may be used to limit the list values to such things as only polygon feature classes or only tables that start with certain letters.

Study questions

1. Select one of the other list types shown in the chart of list functions in the chapter's special introduction, and provide an example of using the list type with a wildcard value.
2. The new fields and final counts were added to the source feature class, but all the selections were done to a feature layer. Why is this desirable?
3. Write an example of code that will return only a single layer from a ListFiles command.

Tutorial 2-6 Building script tools

Scripts can be placed in tools located in a custom toolbox for easy access and easy sharing.

Learning objectives

- Building script tools
- Getting and controlling user input
- Using ArcPy messages
- Creating script tool documentation

Preparation

Research the following topics in ArcGIS for Desktop Help:

- "What is a script tool?"
- "Accessing parameters in a script tool"
- "Understanding messages in script tools"
- "Value list filter"

Introduction

The scripts in the previous tutorials of this chapter all deal with predefined inputs and are designed to run independently of any user input. Although this design functions well in many situations, there are times that you will want to run scripts inside ArcMap and accept user input. To perform these tasks, your scripts must become script tools.