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.

Creating a script tool is straightforward, and several interesting options allow you to make sure things run smoothly. The basic process is to add the script to a toolbox and define the input and output parameters. As with many items in Python, the inputs are indexed in the script, and these index numbers determine the order in which they will appear in the user dialog box. Although there is a Python system command for getting user input, a special function in ArcPy allows you to specify the input type when you create your script tool. This function is named arcpy.GetParamterAsText(), and you increase the index number for each input parameter. Options include defining the type of entry that can be made and pulling a list of values from a domain or attribute table.

Once your script tool is created from a script, it will perform just like the regular ArcGIS tools. You can place it on toolbars, access it from the Catalog window, call it in other scripts, and even index it in the Search window to make it easier to find.

Scenario

You completed a script to perform a single- and multifamily building count for all the box zones in Oleander, and the Fire Department will do periodic counts of other building types in a single box on an as-needed basis. With some simple modifications, you can have the script get input from the user to decide what box zone to work on and which building type to search for. The resulting building count can be stored in a new field with a predetermined name and then displayed in the geoprocessing Results window. Because the building count will differ as the city changes, it will be helpful to run this script to get fresh counts each time.

Data

The data is the box map layer and the individual box zone files from the City of Oleander geodatabase. Also in the map document are the building footprints. The field UseCode has a numeric code for each use type. The following list contains the use type followed by the field name you should use to store the building count for each type:

1 = Single Family (SFCount)
2 = Multi-Family (MFCount)
3 = Commercial (ComCount)
4 = Industrial (IndCount)
5 = City Property (CityCount)
6 = Storage Sheds (ShedCount)
7 = Schools (SchCount)
8 = Church (ChurCount)

SCRIPTING TECHNIQUES

The new technique shown here is how to create a script tool from a script. Two important steps will help lead to predictable results. The first step is to carefully track any input or output variables. Make notations in the code to identify the order in which the input values are accepted from the user. The script tool must consume these values in the same order. The index numbers are used to track this order, starting with zero and moving up from there.

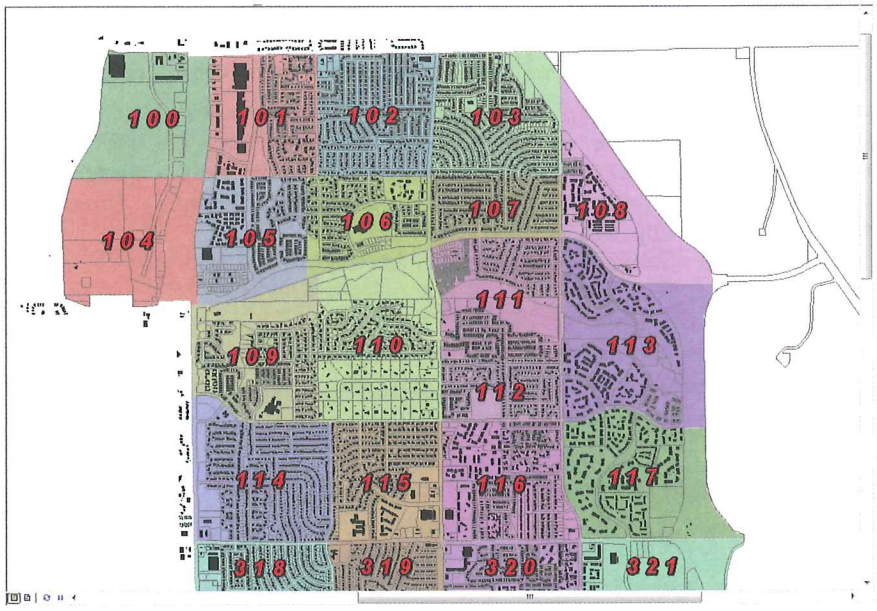
Although this script does not use it, there is a special function for returning a value from a script tool. This function is primarily used to feed a return value to a model, but it may also be used to send a value to another script or model. This function is arcpy.SetParameterAsText(), which is tracked with an index number and is set as output in the script tool.

The second step is to ensure that the input value type is set correctly. Setting these value types will cause the script to use the standard ArcMap input dialog boxes, giving your custom tool the look and feel of a system tool. But remember that whatever type you set, the dialog box will expect you to select that type of item. For instance, if you set the input to Feature Class, the dialog box will have you select an existing feature class. But if the tool is set to create a new feature class, you will not be able to give it a valid name because you will be prompted to select an existing feature class. In this case, you would set the input to a string, and then use the string to create the feature class. However, if you are setting a default workspace, and you set the input to workspace, only valid, existing workspaces can be selected, which would be correct. Use the rule that whatever data type you set, an existing item of that data type will be selected.

Another interesting technique is the use of an input filter. You can set a filter for certain data input types. This filter can be as simple as a predefined list of values, or it may include complex code, which you will learn in tutorial 2-7. The list will limit the choices of the user—these types of data integrity rules are always a good idea. In addition to the list, you will add documentation to the script so that the user can better understand how to use the tool—another good practice. Make it a goal with all your scripts to document the code thoroughly for future programmers, set as many data integrity rules as possible to make it easier on the user, and add as much detailed documentation as possible to avoid user confusion.

Build script tools

1. Open the map document Tutorial 2-6. The box zones and the building footprints are as shown:



2. Examine the data, and determine your course of action. Write your pseudo code outlining the necessary steps. Completed pseudo code is shown at the end of this tutorial.

3. Start a new script in your IDE named Tutorial2-6.py. Although this script is being written to your IDE, it will later become an ArcGIS script tool. Set up the template lines in your script to store the description, and load the ArcPy module. Set the workspace environment to C:\EsriPress\GISTPython\Data\City of Oleander.gdb.

```
#-------------------------------------------------------------------------------
# Name:         Tutorial 2-6
#               Dwelling count with user input
# Purpose:      Prompt the user for the building type and the box number.
#               Count the buildings in the selected box zone and store the results
#               in the attribute table.
#
# 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
    env.workspace = r"C:\EsriPress\GISTPython\Data\City of Oleander.gdb"

    # ENTER CODE

    # Determine results

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

You must prompt the user for the box number and the building type. Use the arcpy.GetParameterAsText() function with index numbers 0 and 1, but later when this becomes a script tool, you will see how to make input statements a list type input.

4. Write the code to get the box zone number as a variable named boxNumber and the building type code as a variable named buildingType with index numbers 0 and 1, respectively, as shown:

```
# Get input from the user
#    The first will be the box number to act upon - index 0
#    The second will be the building type to count - index 1
boxNumber = arcpy.GetParameterAsText(0)
buildingType = arcpy.GetParameterAsText(1)
```

Remember these index numbers because they must be listed in the same order in the script tool. The first input will be the box number in which the user wants to perform a count, and the second input will be the building type to use for the count.

5. Use the MakeFeatureLayer tool to get the correct box zone file and the proper set of building footprints, as shown:

```
# Make feature layers from the user input
boxLayer = arcpy.MakeFeatureLayer_management(\
r"C:\EsriPress\GISTPython\Data\City of Oleander.gdb\FireBoxMaps\FireBoxMap_" \
 + str(boxNumber))
buildLayer = arcpy.MakeFeatureLayer_management(\
r"C:\EsriPress\GISTPython\Data\City of Oleander.gdb\Planimetrics\BldgFootprints", \
"\"UseCode\" = '" + buildingType + "'")
```

Note that because you are making a variable equal to the MakeFeatureLayer tool, you do not need to add an output layer name to the tool's syntax.

Do the selection and count just as you did in tutorial 2-5.

6. Write the code to select the buildings based on the specified box zone, and count the number of selected features, as shown:

```
# Use the specified file of box zone to select specified type of building
arcpy.SelectLayerByLocation_management(buildLayer, "HAVE_THEIR_CENTER_IN", boxLayer)

# Count the selected features
bldgCount = int(arcpy.GetCount_management(buildLayer).getOutput(0))
```

This code will get the building count. Write this code to the geoprocessing Results window, and note that there is a special ArcPy function to do this—in fact, there are three. The differences are basically in the color of the text when the message is printed. The commands are as follows:

- arcpy.AddMessage, which adds text to the Results window in black letters
- arcpy.AddWarning, which adds text to the Results window in green letters
- arcpy.AddError, which adds text to the Results window in red letters

Note that none of these commands will interrupt the script, but each will print a message in its associated color. Next, use one of these commands to add a statement about the feature count to the Results window.

7. **Add the message command to put the feature count in the Results window, as shown:**

```
# Display the results in the geoprocessing Results window
arcpy.AddMessage("The count of buildings is" + str(bldgCount) + ".")
```

Next, create a field to store the results, and calculate it equal to the count. The field name will depend on the building type the user specified. Remember that the ArcPy AddField function will work, even if the field already exists.

8. **Write code that uses the buildingType variable to determine the proper field name, and add it to the selected feature class. Then calculate the count value to that field, as shown:**

```
# Create a field to store the results

# 1 = Single Family (SFCount)
# 2 = Multi-Family (MFCount)
# 3 = Commercial (ComCount)
# 4 = Industrial (IndCount)
# 5 = City Property (CityCount)
# 6 = Storage Sheds (ShedCount)
# 7 = Schools (SchCount)
# 8 = Church (ChurCount)

if buildingType == 1:
    newField = "SFCount"
elif buildingType == 2:
    newField = "MFCount"
elif buildingType == 3:
    newField = "ComCount"
elif buildingType == 4:
    newField = "IndCount"
elif buildingType == 5:
    newField = "CityCount"
elif buildingType == 6:
    newField = "ShedCount"
elif buildingType == 7:
    newField = "SchCount"
else:
    newField = "ChurCount"

arcpy.AddField_management(boxLayer,newField,"LONG")

# Store the results in the field
arcpy.CalculateField_management(boxLayer,newField,bldgCount)
```

This graphic shows a lot of code, but it is basically three steps: determine the field name to use, add that field, and calculate the value.

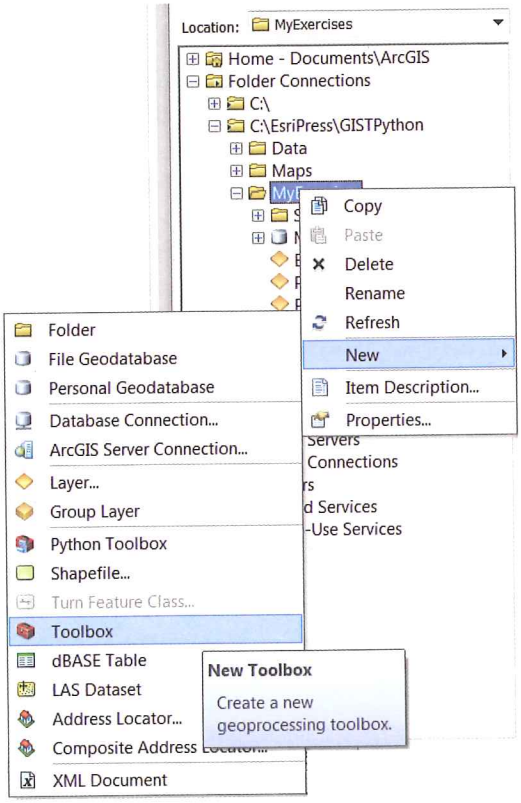
9. **Now that your code entry is complete, save your Python script to your MyExercises folder where you installed the student data, and close your IDE.**

Here's the pseudo code for the script:

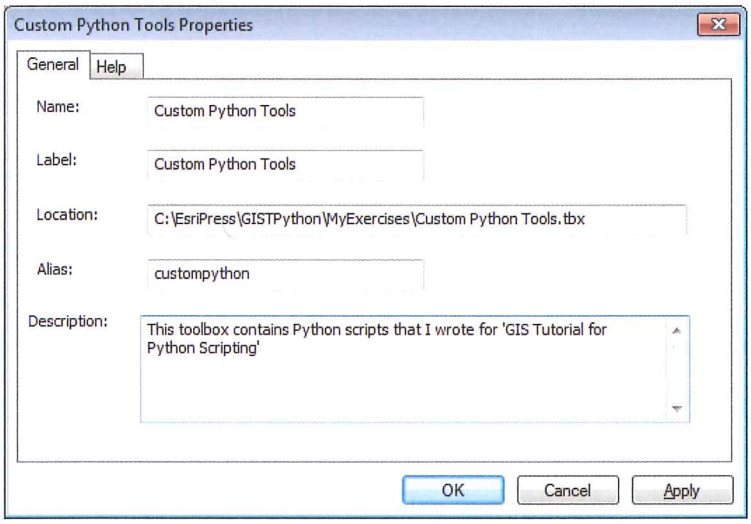
```
# Import the modules, and set the environments
# Get input from the user
#    The first will be the box number to act upon
#    The second will be the building type to count
# Make feature layers from the user input
# Use the specified file of box zone to select specified type of building
# Count the selected features
# Display the results in the geoprocessing Results window
# Create a field to store the results

# 1 = Single Family (SFCount)
# 2 = Multi-Family (MFCount)
# 3 = Commercial (ComCount)
# 4 = Industrial (IndCount)
# 5 = City Property (CityCount)
# 6 = Storage Sheds (ShedCount)
# 7 = Schools (SchCount)
# 8 = Church (ChurCount)

# Store the results in the field
```

Script tools must be stored in an existing toolbox, so create a new one in your MyExercises folder.

10. **In ArcMap, open the Catalog window, and navigate to your MyExercises folder where you installed the student data. Right-click the folder, and click New > Toolbox, as shown in the graphic. Name the toolbox Custom Python Tools.**



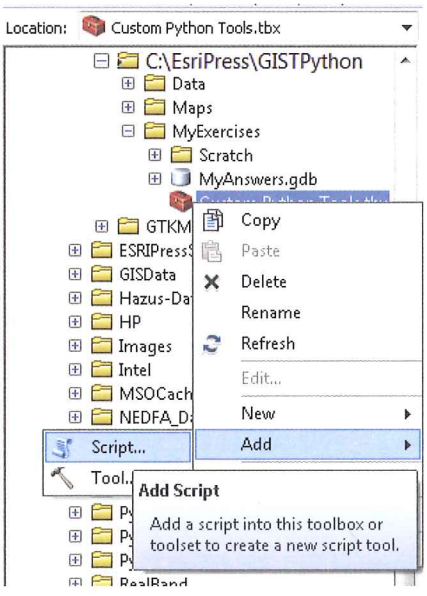
11. **Right-click the toolbox, and click Properties. For an alias, type custompython, and add a description, as shown in the graphic. Click OK.**



Adding aliases to your toolboxes is important. When you access tools from a toolbox in a Python script, use the tool name and an underscore and then the toolbox alias. Have you noticed that the ArcPy tools you have been using all have a toolbox alias after the tool name? Using the alias helps distinguish between two tools that might have the same name. You will not use the alias here, but it is good practice to assign one for later use if you index your custom toolboxes.

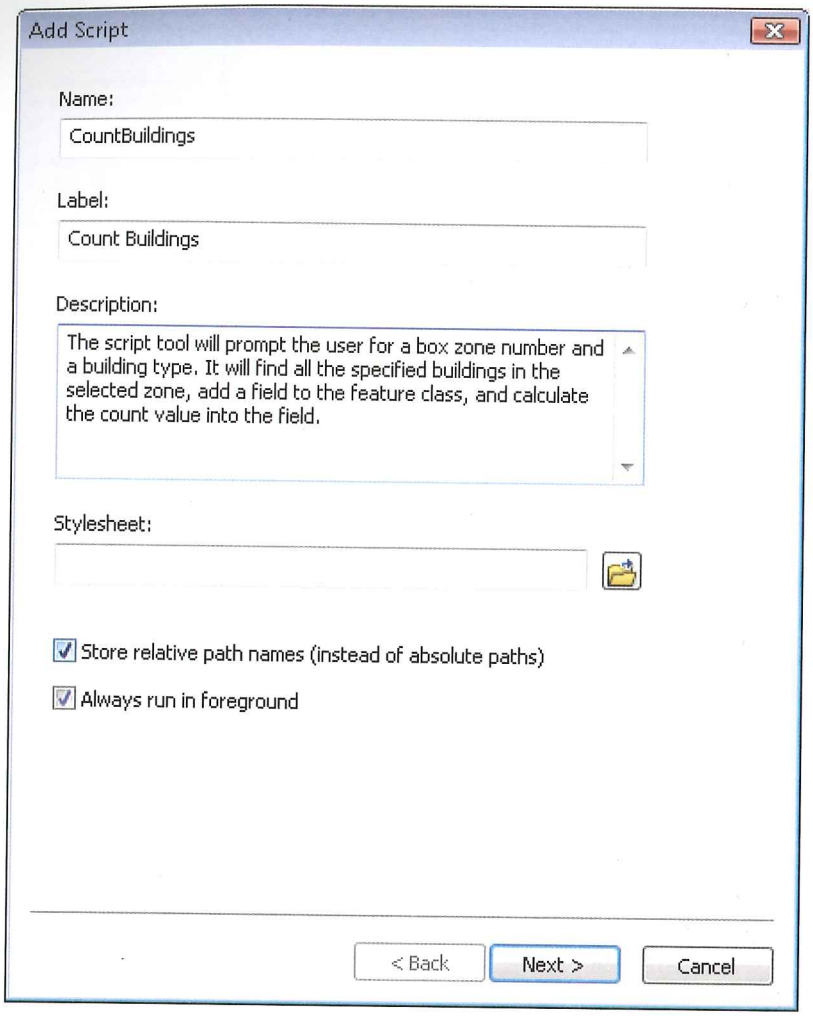
Start the process by adding your script to the toolbox. There are several steps, so be careful not to miss one.

12. **Right-click the Custom Python Tools toolbox and click Add > Script, as shown:**

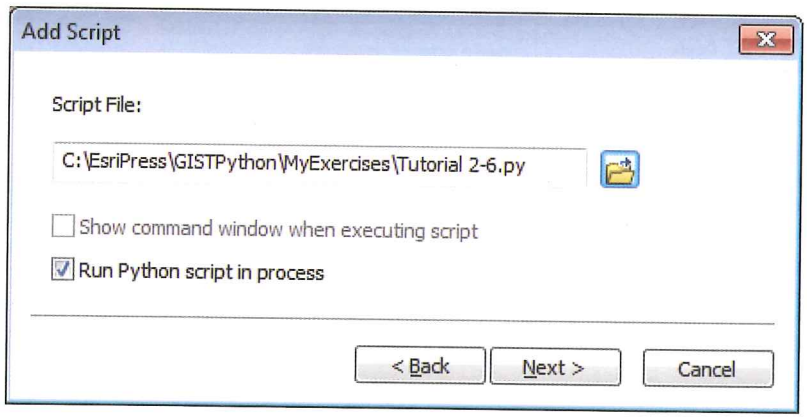


This opens the script tool dialog box, and you can set the parameters.

13. Name the tool CountBuildings, label it Count Buildings, and give the tool an appropriate description. Also, select the "Store relative path names" check box, as shown in the graphic. Click Next.

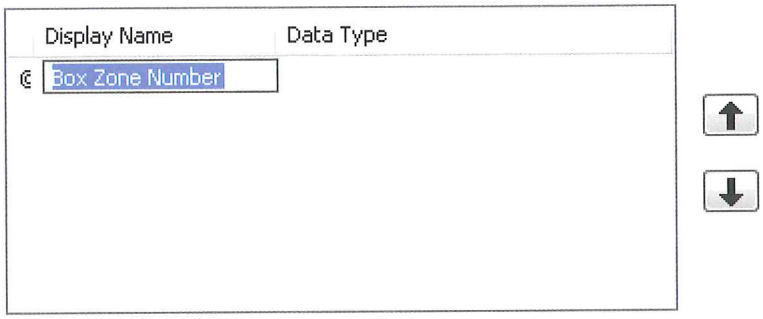


14. Click the Navigate button, and select your script from your MyExercises folder, as shown in the graphic. Click Next.



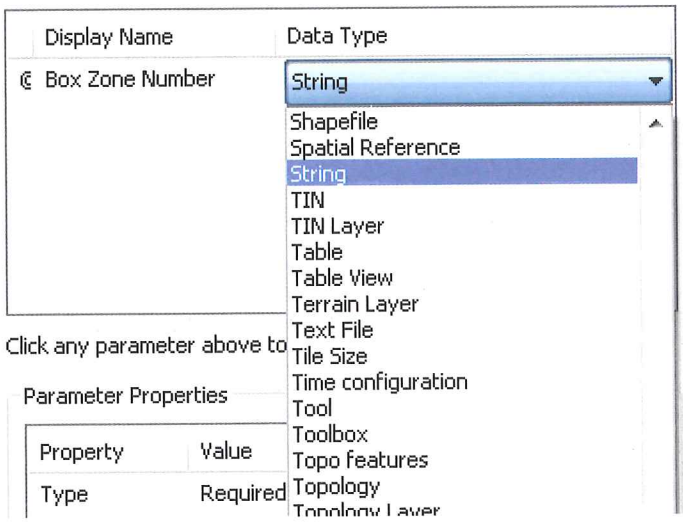
Next, define the input parameters for the two values you need from the user. Remember that the index numbers you assigned will get the box zone number first and the building type second.

15. **As shown, click the first cell under Display Name and type Box Zone Number, which will act as a text prompt to the user.**

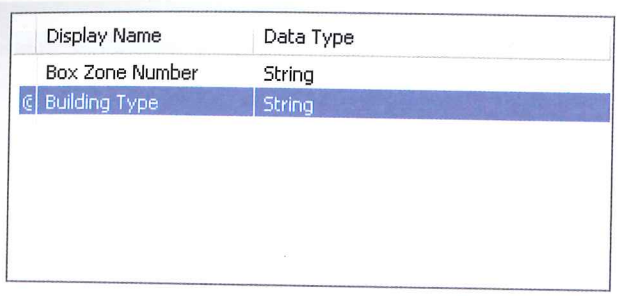


Setting the data type is next. When you click the Data Type arrow, a long list of data types appears. It is important to realize that if you choose a particular type of file, such as a feature class, shapefile, or dBASE file, the input must be a file of this type and must already exist. The advantage of setting the data type is that the selection dialog box will only take files of the specified type. For instance, if you set the data type to Workspace and navigate to C:\EsriPress\GISTPython\Data, you will see only geodatabases listed and none of the text files or shapefiles that might be there. If you are just getting text that will be used to create one of these types of files, your input should be a string. For this input, you can select String to allow alphanumeric entries.

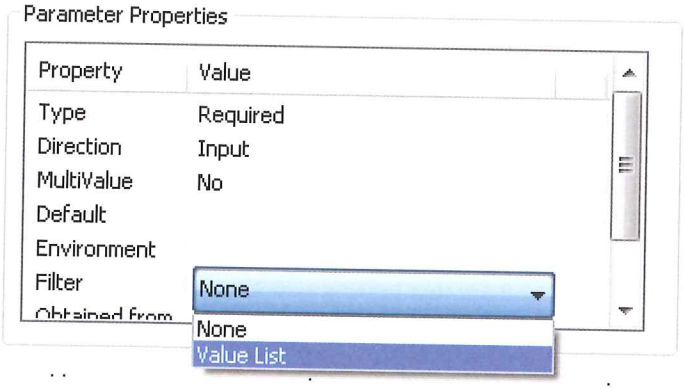
16. **Click the Data Type box and select String, as shown:**



17. On the next line, type the display name as Building Type, and set it to String, as shown in the graphic. Note that in the Parameter Properties box both of these items are set as Required and Input.



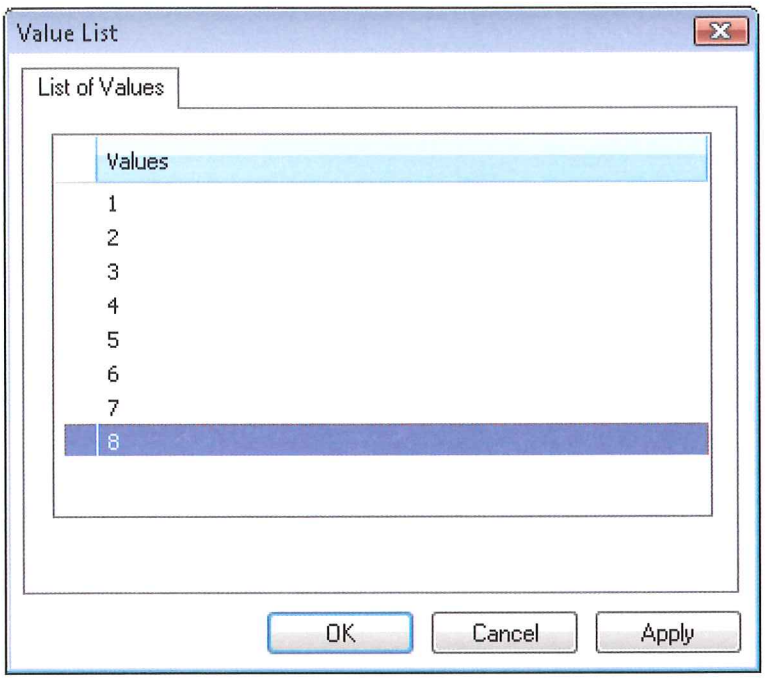
18. With Building Type still highlighted, move to the Parameter Properties panel, and click the cell next to Filter. In the drop-down list, select Value List, as shown:



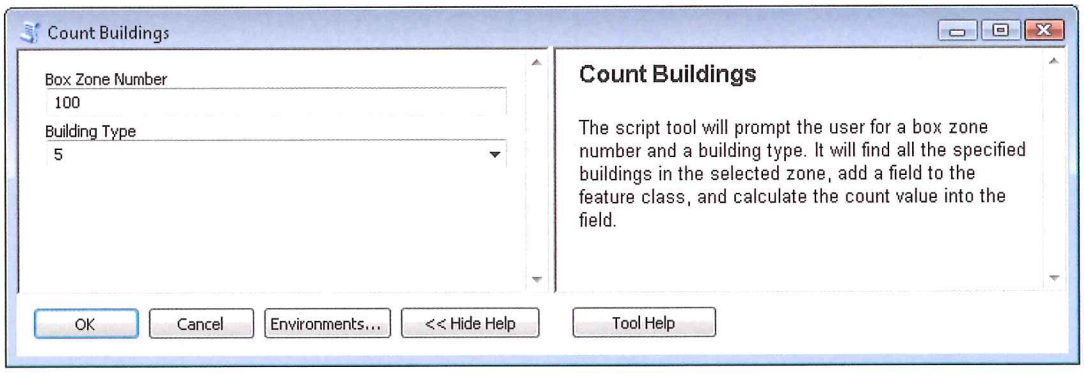
This dialog box will let you build a value list to make data entry easier for the user. This value list can also prevent users from entering an invalid value. The table lists the filter types you can apply with this setting, as shown:

Filter type	Values
Value List	A list of string or numeric values. Used with String, Long, Double, and Boolean parameter data types.
Range	A minimum and maximum value. Used with Long and Double data types.
Feature Class	A list of allowable feature class types: "Point", "Multipoint", "Polyline", "Polygon", "MultiPatch", "Sphere", "Annotation", and "Dimension". More than one value can be supplied to the filter.
File	A list of file suffixes, for example, "txt; e00; ditamap".
Field	A list of allowable field types: "Short", "Long", "Single", "Double", "Text", "Date", "OID", "Geometry", "Blob", "Raster", "GUID", "GlobalID", and "XML". More than one value can be supplied to the filter.
Workspace	A list of allowable workspace types: "File System", "Local Database", or "Remote Database". More than one value can be supplied.

19. **In the Value List, enter the numbers 1 through 8, as shown in the graphic. Click OK, and then click Finish to complete the script tool creation process.**

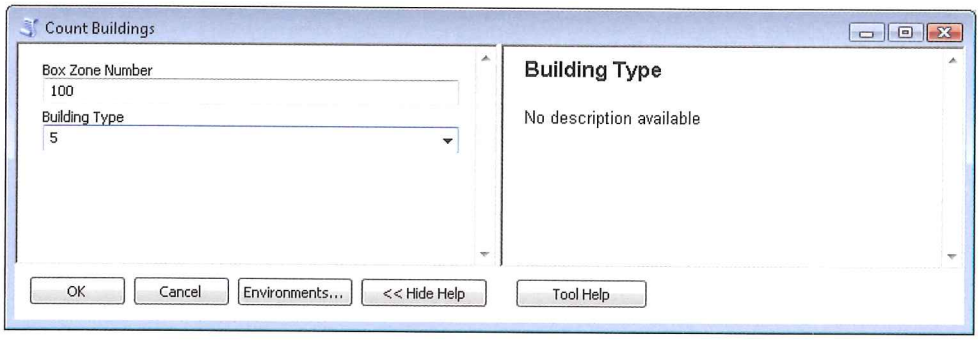


20. **Run the script tool by double-clicking it in the toolbox. Enter a box zone number of 100, and select a building type of 5, as shown in the graphic. Click OK to run the tool, and make sure to clear the check box "Close this dialog when completed successfully." If the dialog box closes too quickly, or if the script is running in the background, you can open the geoprocessing Results window to see the building count.**



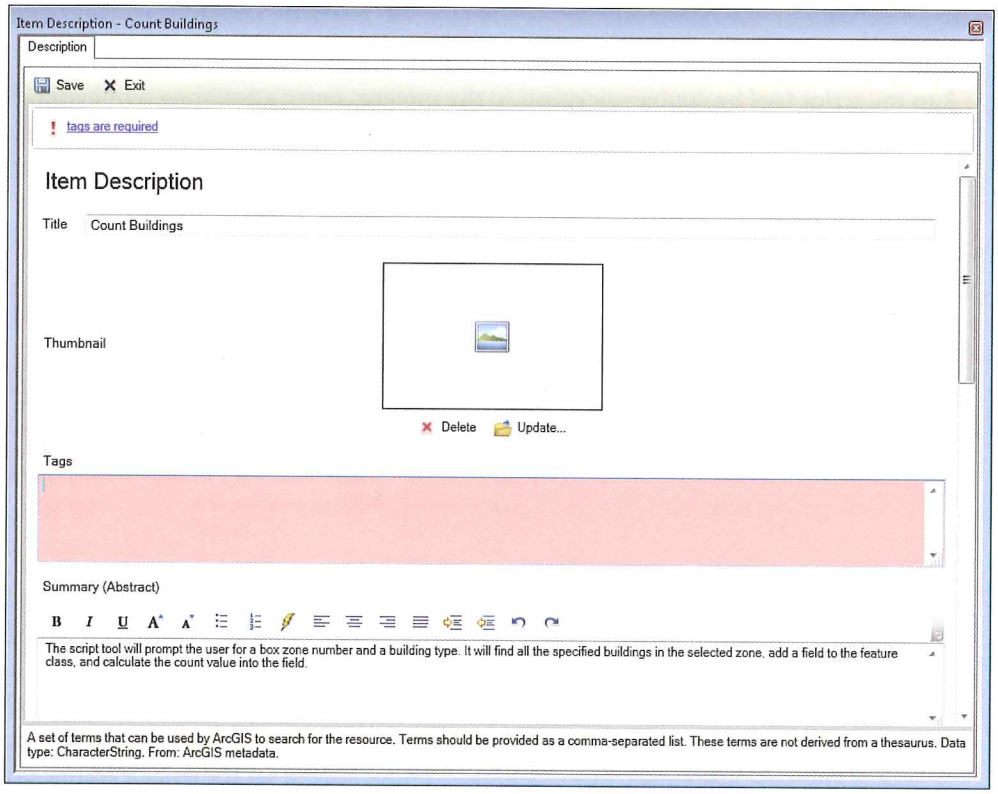
If you completed the steps correctly, the tool will run successfully. If the tool fails, right-click the tool in the toolbox and click Edit. This opens an editing tool that lets you make changes to the script. The default editing tool is Notepad. Refer to appendix A for how to set the default editing program for Python scripts used in ArcGIS.

Did you notice that the description you typed earlier during the tool creation process became the tool Help? You may have also noticed that when you clicked the entry boxes to type values, you did not get a Help message, as shown:



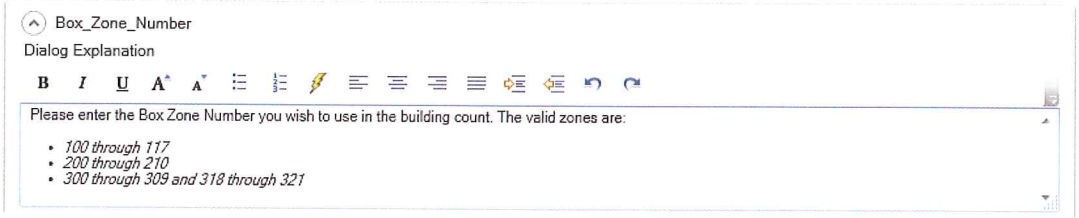
There is a way to add more relevant help to the tool, enhance its appearance, and give it the same look and feel as a system tool.

21. **Right-click your new tool and click Item Description. In the resulting window, click Edit. Scroll through the window, and note the variety of items that can be entered (and that tags are required), as shown:**

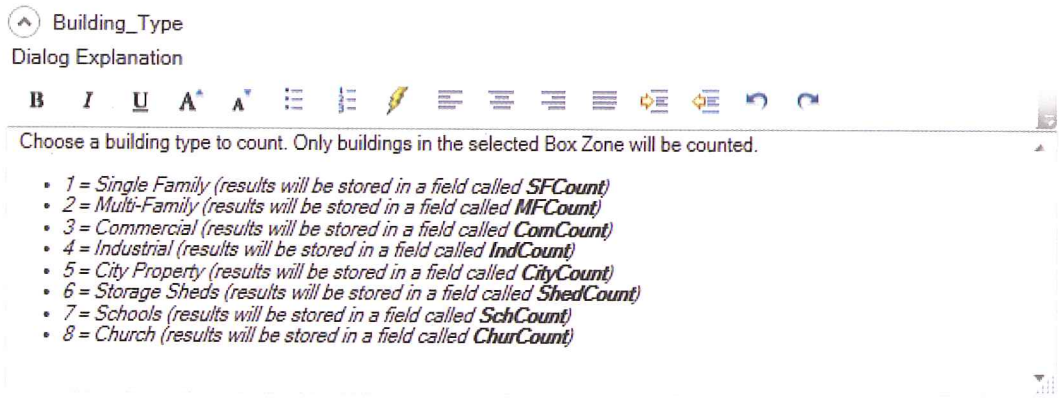


As you click each of the entry lines, a small Help text is displayed at the bottom of the window to guide you toward inputting the correct information.

22. Scroll down and click the chevron next to Box_Zone_Number. Here, you can type more descriptive instructions for the user. Notice that there are many formatting options, including bold, italic, and bullet lists. Fill in detailed instructions, as shown:



23. Add a better description for the parameter Building_Type, as shown:



24. Explore the other description items, and enter tags that will identify the purpose of this script. When you are done, click Save and close the Item Description dialog box.

25. Run the script again, and click in each of the entry boxes to see the resulting Help messages, as shown: