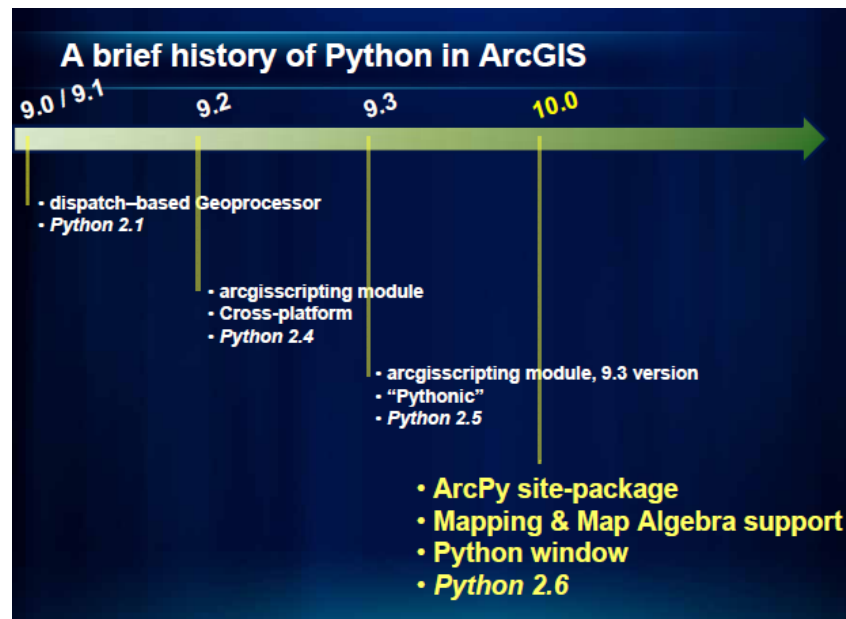


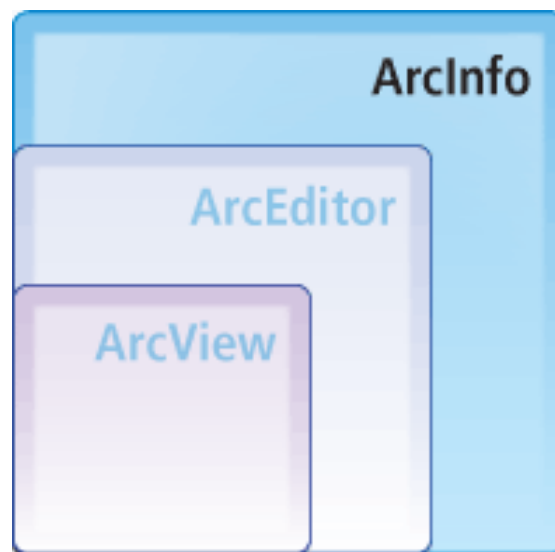
## Chapter 6: Geoprocessing with Tools and Toolboxes

### 6.0 Introduction

- All geoprocessing tools are available through Arcpy.
- All tools are exposed as native methods of the Arcpy site package.
- Previous versions of ArcGIS require that an object of the geoprocessor module (encapsulation of all gp functions) be first created. In version 10 of ArcGIS you need only import the Arcpy site package.



- Access to these tools is dependent on the type of ArcGIS Desktop license that you have.



- Presently, ArcGIS has 800+ geoprocessing tools.

## 6.1 Gaining access

- You must first import the Arcpy site package

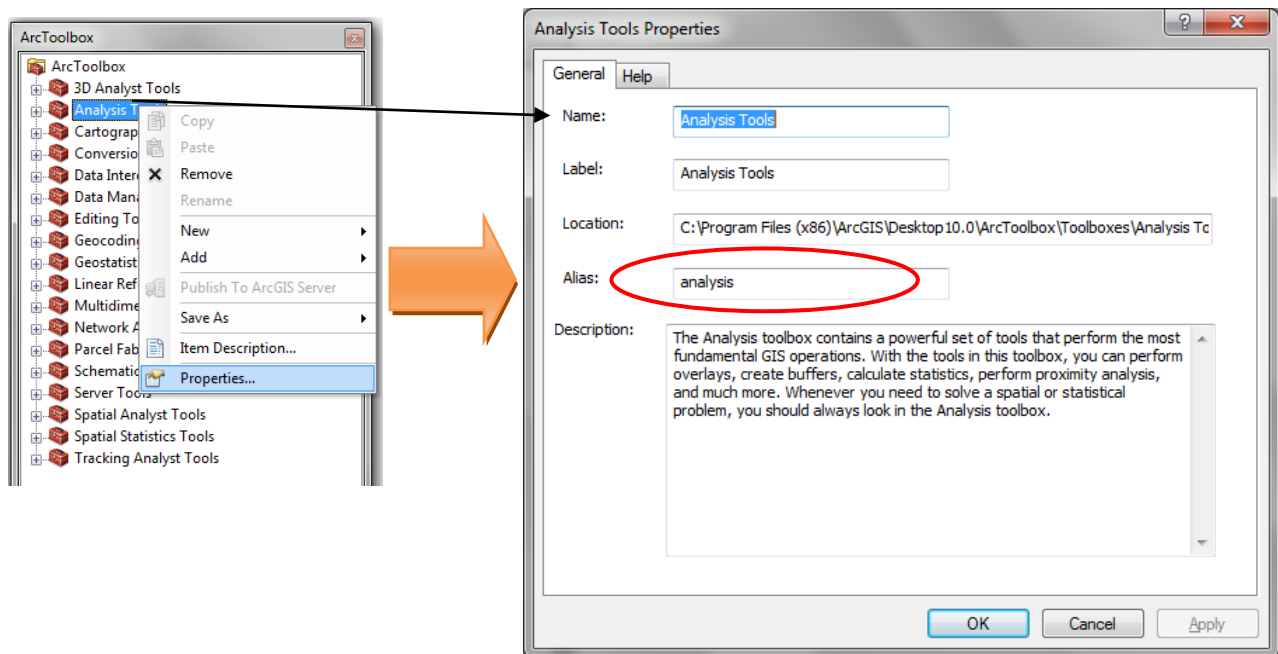
## 6.2 Toolbox alias

- Each toolbox in ArcGIS has an alias property
- The alias is used when running tools in the Python window or in scripts and uniquely identifies the tool.
- Tools within a toolbox cannot have the same name, but it is possible for different toolboxes to have tools with the same name. For example, there are two tools named Clip, one in the Analysis toolbox (alias name analysis) and one in the Coverage toolbox (alias name arc). When you use Clip in the Python window, you are required to choose either Clip\_analysis or Clip\_arc.
- The alias cannot contain spaces or special characters, such as an underscore (\_).
- The table below contains the aliases of all system toolboxes. You should never use these aliases for your custom toolboxes.

System toolbox	Alias
3D Analyst Tools	3d
Analysis	analysis
Cartography	cartography
Conversion	conversion
Coverage	arc
Data Interoperability	interop
Data Management	management
Editing	edit
Geocoding	geocoding
Geostatistical Analyst	ga
Linear Referencing	lr
Multidimension	md

Network Analyst	na
Parcel Fabric	fabric
Samples	samples
Schematics	schematics
Server	server
Spatial Analyst	sa
Spatial Statistics	stats
Tracking Analyst	ta

- The alias of any toolbox can be found by right clicking on the toolbox and selecting the properties option.



- In Python you call a tool using the general syntax <Tool name>\_<Toolbox alias>. To call the Buffer tool we would use buffer\_analysis

## 6.3 Tool organization

- Geoprocessing tools are organized in two different ways. All tools are available **as functions on ArcPy** but are also available **in modules matching the toolbox alias name**.
- Although most of the examples in the help show tools organized as functions available from ArcPy, both approaches are equally valid. Which approach you use will come down to a matter of personal preference and coding habits. In the following example, the [GetCount](#) tool is shown using both approaches.

```
import arcpy

inFeatures = "c:/temp/rivers.shp"

# Tools can be accessed as functions on the arcpy module, and
# from modules matching the toolbox name.
#
arcpy.GetCount_management(inFeatures)
arcpy.management.GetCount(inFeatures)
```

<tool name>\_<Toolbox alias>

- How would the syntax for the buffer tool look like using both approaches?
  - `Arcpy.Buffer_analysis (parameter1,...)`
  - `Arcpy.analysis.buffer(parameter1,...)`

## 6.4 Calling tools in Python

- Each tool has a set of either mandatory and/or optional parameters that it needs in order for the tools run properly.
- In Python the general syntax is as discussed above.

Example:

`Buffer_analysis (in_features, out_feature_class, buffer_distance_or_field, {line_side}, {line_end_type}, {dissolve_option}, {dissolve_field})`

Search Buffer (Analysis) for a description of each parameter

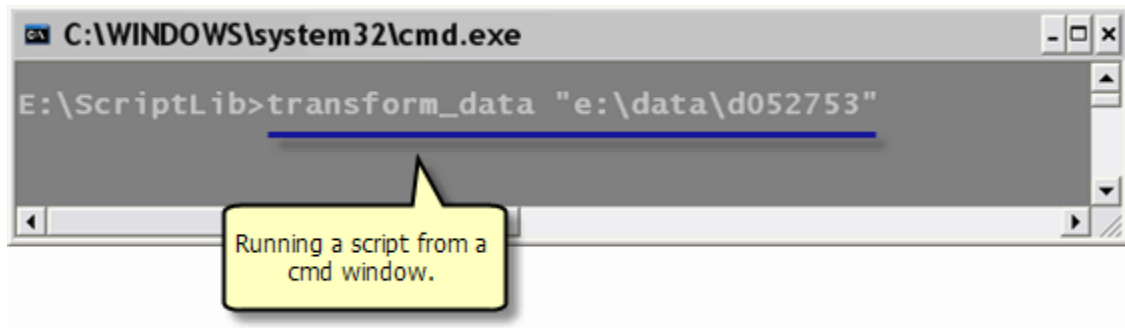
Step through the sample code and try to understand each line.

- NOTICE: in the above the example the suffix is `_analysis`. This is used to distinguish the version of analysis being used.
  - There are two buffer tools: one in analysis tools toolbox and the other in the coverage tools toolbox
  - The suffix is a way of distinguishing between the version of tool being used.

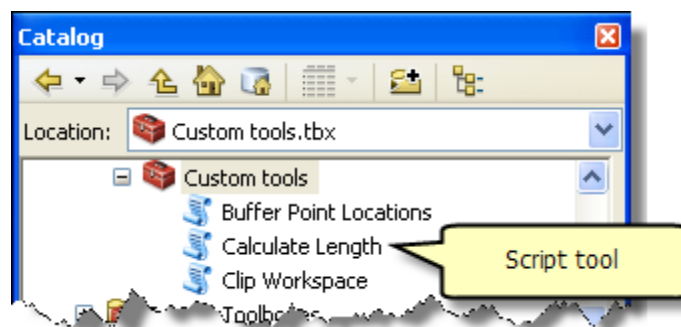
## 6.5 Script tools

Scripts that you create can be executed with one of two basic methods: outside ArcGIS and within ArcGIS.

- Outside ArcGIS means that the script is executed from the operating system command prompt, as shown below, or within a development application, like PythonWin. Scripts executed in this manner are referred to as **stand-alone scripts**.



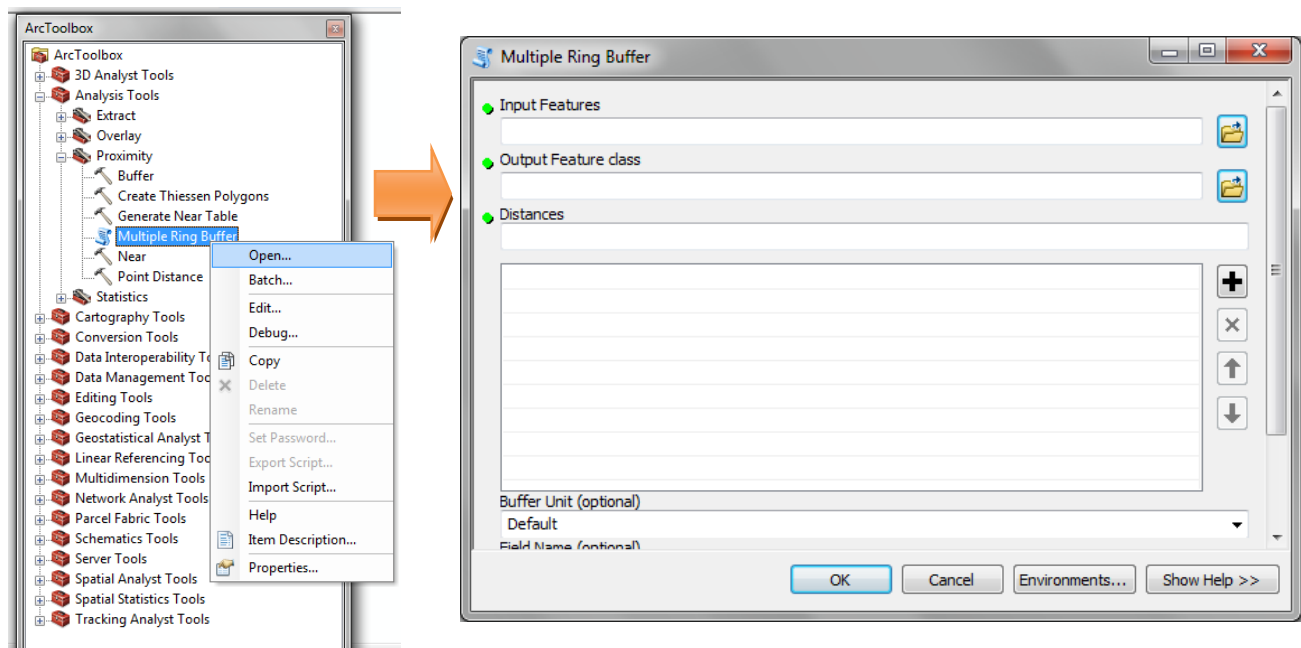
- No visual interface
  - Can be scheduled to batch process data (for example, during off hours for more efficient processing)
  - Easier to debug because of the in-built Python error handling capabilities
  - Uses sys.argv which contains a list of available Python command line elements
- Within ArcGIS means you create a script tool inside a toolbox. A script tool is like any other tool—it can be opened and executed from the tool dialog box, used in models and the Python window, and called from other scripts and script tools.
  - Familiar user interface for executing geoprocessing functionality
  - Much more difficult to debug during development



Creating a script tool for your script is easy, and there are many advantages to doing so.

- A script tool that you create is an integral part of geoprocessing

- Just like a system tool—you can open it from the Search or Catalog window, use it in ModelBuilder and the Python window, and call it from another script.
- You can write messages to the progress dialog box and the Results window.
  - For example, error prevention alerting the user to different errors that may exist: no data, invalid file format,...
- Using the built-in documentation tools, you can provide documentation.
- When the script is run as a script tool, the arcpy object you import is fully aware of the application it was called from, such as ArcMap. All environment settings made in the application, such as `arcpy.env.overwriteOutput` and `arcpy.env.scratchWorkspace`, are available from the geoprocessing object you create.
- Build your own library of tools for specific applications
- Buttons available for browsing to input and output making it more visually appealing




- Almost all tools require parameters which you set either using the tool's dialog or command line
  - Parameters need not always be hard coded allowed code to become more flexible.
  - Users using the script can input their own value at run time. For example, the name of the output files being created.

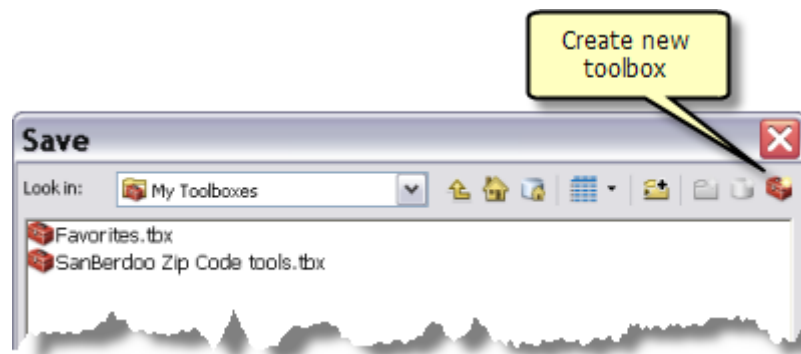
## 6.6 Creating a script tool

- Requires 3 things
  - A script
  - A custom toolbox
  - Parameters specific for the script

- You can create your own toolbox and add tools and toolsets to it. Any toolbox you create is called a custom toolbox, as opposed to a system toolbox (a toolbox installed with ArcGIS). Script and model tools that you create need to be stored in a custom toolbox.

Steps:

- In the Catalog window, navigate to the folder or geodatabase where you want the toolbox to be created.
- Right-click the folder or geodatabase and click New > Toolbox.
  - The default name of the toolbox (Toolbox.tbx or Toolbox) will be highlighted, and you can enter the name for your toolbox. If the toolbox name is not highlighted, right-click the toolbox and click Rename.
  - When saving a new model in ModelBuilder, you can create a new toolbox with the New Toolbox button , as illustrated below.



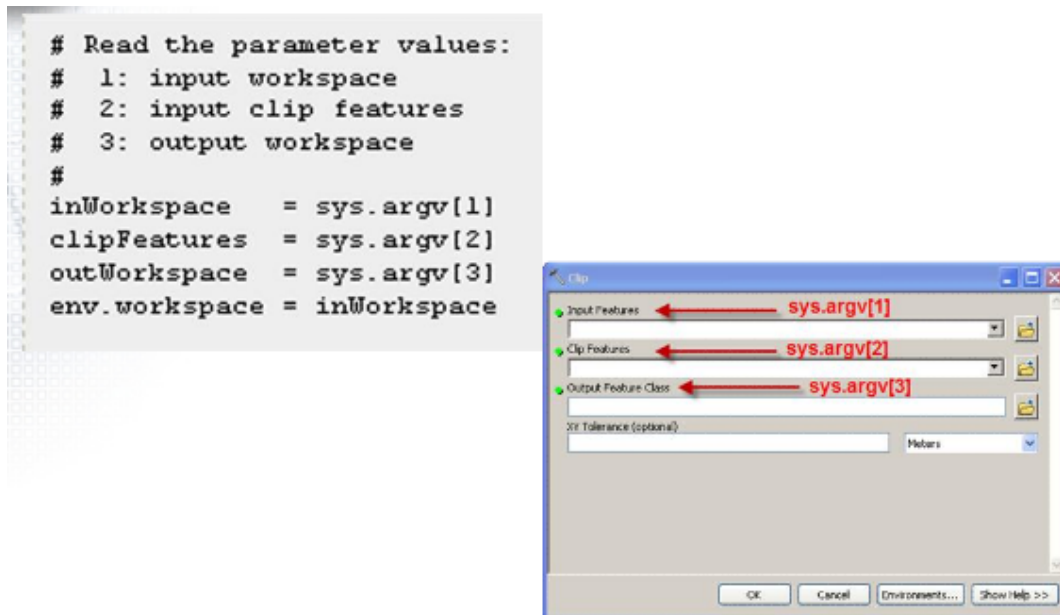
- A good practice is to give your new toolbox an alias when you create it. The toolbox alias is used to identify the toolbox in scripting. Right-click the toolbox and click Properties. On the General tab, enter an alias name.
- To create a script tool, right-click your custom toolbox and click Add > Script. This opens the Add Script wizard that takes you step by step through the process of creating a script tool. After completing the steps, your toolbox will contain a new script tool. You can always modify properties (such as parameter names and data types) of this script tool by right-clicking the script tool and choosing Properties.
- Caution:

Do not create a toolbox in your system temp folder since ArcGIS will delete toolboxes in your system temp folder. The system temp folder is typically C:\Users\<username>\AppData\Local\Temp (Windows Vista and Windows 7) or C:\Documents and Settings\<username>\Local Settings\Temp (Windows XP). You can determine the location of your temp folder as follows:

1. From the Windows Start menu, click Run.
  2. In the Run window, enter cmd. This will open a cmd window.
  3. In the cmd window, enter echo %TEMP% and press ENTER. The location of your system temp folder will be displayed.
- All system toolboxes are read only and therefore nothing can be added to them
  - The toolbox should be given a name and an ALIAS
  - Parameter values entered by the user are sent your script
    - To retrieve user values either GetParameterAsText() or sys.argv[] can be used
    - sys.argv[] has a finite size of text input which can accept; 1024 characters per parameter

## 6.7 sys.argv[]

- allows you to capture parameter input
  - a series or words inputted when you call a script
  - each word must be separated by an empty space
  - words are stored in a zero-based list object
    - sys.argv[0] stores the script name
    - the first parameter is sys.argv[1]



## 6.8 GetParameterAsText()

- Also used to capture input parameters
- Unlike sys.argv[], this method can only be used with ArcToolbox
  - Not for use with PythonWin or command prompt
- Zero based with the first parameter stored at position 0 and incremented by 1 thereafter
- Much harder to debug since it can't be tested in PythonWin



```
# Import arcpy site-package
#
import arcpy
from arcpy import env

# Read the parameter values:
# 1: input workspace
# 2: input clip features
# 3: output workspace
#
inWorkspace = arcpy.GetParameterAsText(0)
clipFeatures = arcpy.GetParameterAsText(1)
outWorkspace = arcpy.GetParameterAsText(2)
env.workspace = inWorkspace
```

## 6.9 Script properties

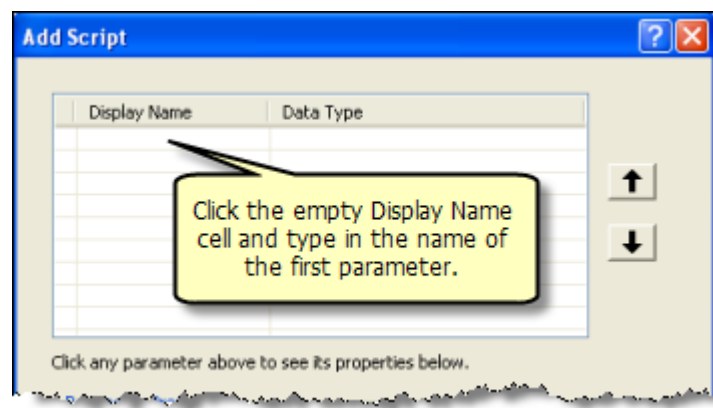
- **Name**
  - Must be a unique ArcToolbox name.
  - **No spaces** or restricted characters.
- **Label**
  - The text to display in the toolbox that references the Name.
  - Doesn't have to be unique.
- **Description**
  - General description of what functionality the script provides.
- **Stylesheet**
  - Used to customize the properties of dialog items and background of the tool.

## 6.10 Running script in process

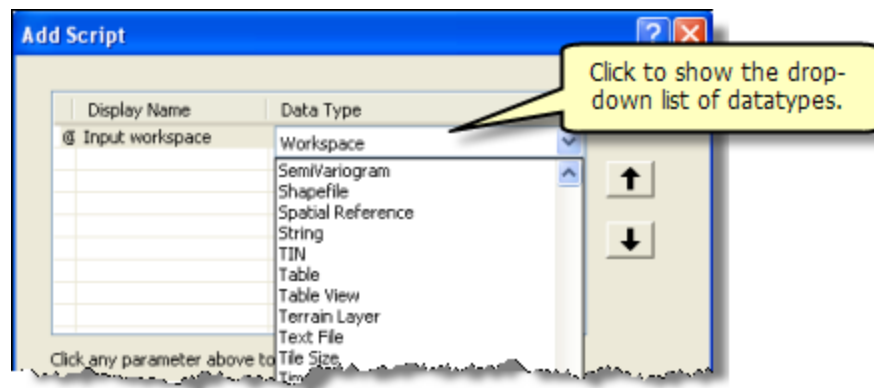
- The Run Python script in process option became available in ArcGIS 9.3. Prior to 9.3, Python scripts were run out-of-process.
- Running out-of-process requires that ArcGIS start another process (think of this as starting another program), which takes time.
- When running out-of-process, there are also performance issues with message communication between the two processes. Starting at version 9.3, Python is incorporated into ArcGIS so that scripts can be run in-process, removing all startup time and messaging overhead.
- Running in-process requires that all modules loaded with the Python import directive have the necessary logic to enable them to run in-process. All standard Python libraries, such as os, string, and time, have the necessary logic. However, non standard modules obtained from third parties may not have the necessary logic to run in-process. If you are experiencing unexplainable problems when your script runs, try unchecking the in-process option and running your script again. If the problem goes away when running out-of-process, then there is most likely an issue with one of the modules you imported. In this case, leave the option unchecked.
- The Run Python script in process option ONLY applies to Python scripts.

### 6.11 Setting script tool parameters

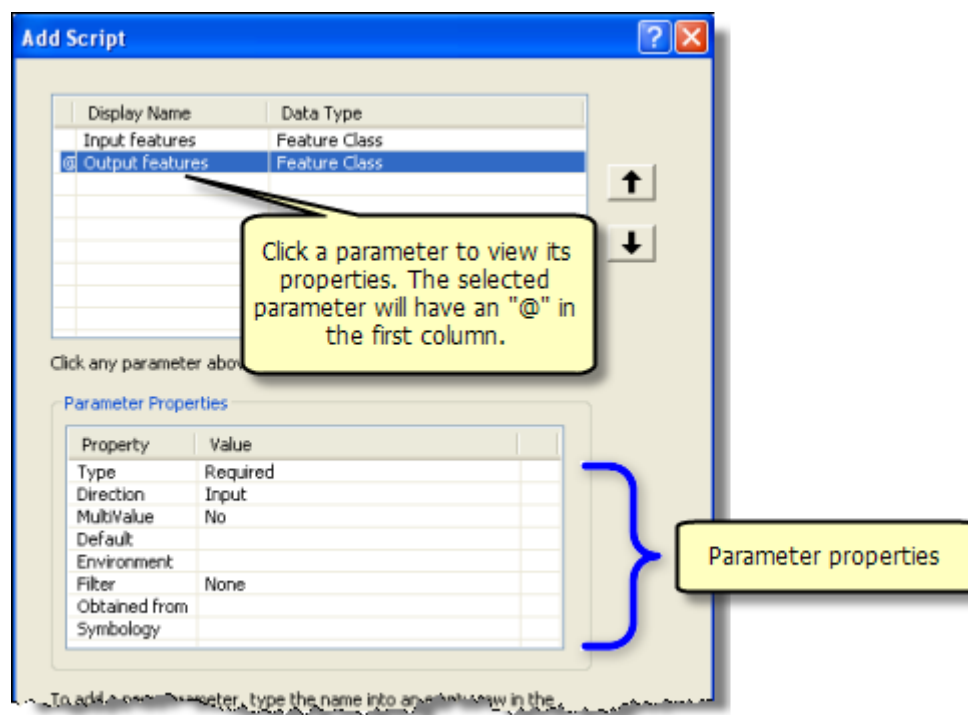
- Script tool parameters can be set when using the Add Script wizard. You can also add, delete, and modify script tool parameters from a tool's Properties dialog box. To access script tool properties, right-click the tool, click Properties, then click the Parameters tab.
- Whether you are setting parameters in the Add Script wizard or in the Properties dialog box, the procedures (as described here) are the same.
- To add a new parameter, click the first empty cell in the Display Name column and type the name of the parameter. This is the name that will be displayed on the tool dialog box and can contain spaces. For Python syntax, the parameter name will be the display name with spaces replaced by underscores (\_).



- After entering the display name of the parameter, choose a data type for the parameter by clicking in the Data Type cell, as shown below.



Each parameter has additional properties you can set, as shown below.



Property	Description
<a href="#">Type</a>	Can be Required, Optional, or Derived. Derived means that the user of your tool does not enter a value for the parameter. Derived types are always output parameters.
<a href="#">Direction</a>	Can be Input or Output. If the parameter Type is Derived, direction is always equal to Output.
<a href="#">Multivalue</a>	Multivalue is Yes if you want a list of values, No if you want a single value.
<a href="#">Default</a> or <a href="#">Schema</a>	The default value for the parameter. When the parameter data type is either feature set or record set, <b>Default</b> is replaced with <b>Schema</b> .
<a href="#">Environment</a>	If the default value for the parameter is to come from an environment setting, this property contains the name of the environment setting.
<a href="#">Filter</a>	If you want only certain datasets or values to be entered for a parameter, you can specify a filter. There are six types of filters, and the type of filter you can choose depends on the data type of the parameter.
<a href="#">Obtained from</a>	This property applies to derived output parameters and input parameter data types. For derived output parameters, <b>Obtained from</b> can be set to the parameter containing the definition of the output. For input parameters, <b>Obtained from</b> is set to the parameter containing the information needed for input.
<a href="#">Symbology</a>	This property applies only to output parameters. The value is the location of a layer file (.lyr) that contains the symbology for displaying the output.

## Type

There are three choices for Type:

- A **Required** parameter requires an input value from the user. The tool cannot be executed until the user supplies a value.
- An **Optional** parameter does not require a value from the user.
- A **Derived** parameter is only for output parameters (see Direction below). A derived output parameter does not show on the tool dialog box.

There are five uses for a derived output parameter, as follows:

- The output is the same as the input, such as Calculate Field. Calculate Field changes the values of a particular field on the input table—it doesn't create a new table or modify the schema of the input. Other tools whose output is the same as the input can be found in the Editing toolbox.
- The tool modifies the schema of the input, such as Add Field. Add Field adds a field to the input table—it doesn't create a new output table.
- The tool creates output using information in other parameters, such as the Create Feature Class tool. With the Create Feature Class tool, you specify the workspace and the name of the new feature class, and the feature class is created for you.
- The tool outputs a scalar value as opposed to a dataset. Get Count, for example, outputs a long (the number of records). Whenever a tool outputs a scalar value, the output is Derived.

- The tool will create data in a known location. For example, you may have a script that updates an existing table in a known workspace. The user doesn't need to provide this table in the dialog box or in scripting.
- If your script tool has derived output, you need to set the value of the derived output parameter in your script using the `SetParameterAsText()` function.

## 6.12 Exercise 5